A Semidefinite Relaxation based Branch-and-Bound Method for Tight Neural Network Verification

Jianglin Lan[1], Benedikt Brückner[2], Alessio Lomuscio[2]

[1] James Watt School of Engineering, University of Glasgow, UK
[2] Department of Computing, Imperial College London, UK
Jianglin.Lan@glasgow.ac.uk, b.brueckner21@imperial.ac.uk, a.lomuscio@imperial.ac.uk

Abstract

We introduce a novel method based on semidefinite program (SDP) for the tight and efficient verification of neural networks. The proposed SDP relaxation advances the present SoA in SDP-based neural network verification by adding a set of linear constraints based on eigenvectors. We extend this novel SDP relaxation by combining it with a branch-and-bound method that can provably close the relaxation gap up to zero. We show formally that the proposed approach leads to a provably tighter solution than the present SoA. We report experimental results showing that the proposed method outperforms baselines in terms of verified accuracy while retaining an acceptable computational overhead.

Introduction

Neural networks (NNs) are susceptible to adversarial attacks (Goodfellow, Shlens, and Szegedy 2014). The verification of NNs determines whether a NN satisfies a given specification such as robustness to an ℓ_p-norm perturbation on a given input (Li et al. 2020; Liu et al. 2020). If a NN is verified to be robust, then we know that no adversarial attack exists for the model, input and perturbation under analysis. This makes NN verification valuable for safety-critical systems (Tran et al. 2020; Julian and Kochenderfer 2021; Kouvaros et al. 2021; Manzanas Lopez et al. 2021).

Existing NN verification methods are either *complete* or *incomplete*. Complete methods guarantee that no false negatives or positives are generated. However, this often comes at a high computational cost, hindering their scalability to large NNs. Incomplete methods are based on over-approximations of nonlinear activation functions, possibly leading to spurious counterexamples (Henriksen and Lomuscio 2020, 2021; Wang et al. 2021; Hashemi, Kouvaros, and Lomuscio 2021). At the same time, they are computationally lighter and thus have the comparative advantage of scaling up to larger NNs. The tightness of the approximations plays a key role in the success of incomplete methods, as a looser approximation results in more verification queries remaining unsolved. Hence, it is appealing to design incomplete methods with tight convex approximations whilst retaining an acceptable computational cost.

The existing complete approaches to NN verification are based on either mixed-integer linear programming (MILP) formulations solved via branch-and-bound (BaB) algorithms (Bastani et al. 2016; Lomuscio and Maganti 2017; Tjeng, Xiao, and Tedrake 2019; Anderson et al. 2020; Botoeva et al. 2020; Bunel et al. 2020) or satisfiability modulo theories (Ehlers 2017; Katz et al. 2021). These complete verifiers often have limited scalability. Some incomplete approaches can also offer completeness guarantees by combining symbolic interval propagation with input refinement such as `ReluVal` (Wang et al. 2018c) or neuron refinement such as `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 usually require a very large number of refinements. Other incomplete approaches built on convex relaxations (Xu et al. 2020) or Lagrangian relaxations (Dvijotham et al. 2018; Wong and Kolter 2018; Chen et al. 2021) do not guarantee completeness.

This work will focus on NNs with Rectified Linear Unit (ReLU) activation functions. The tightest possible convex approximation for ReLU, the *triangle relaxation*, was introduced by (Ehlers 2017). Incomplete methods (bound propagations) based on convex relaxations like this one solve polynomial-time solvable linear program (LP) problems and can achieve SoA performance (Weng et al. 2018; Singh et al. 2019a; Tjandraatmadja et al. 2020; Müller et al. 2021). However, their efficacy is fundamentally limited by the tightness of convex relaxations, an issue known as the "convex relaxation barrier" (Salman et al. 2019). Hence, even when the optimal convex relaxation is used for each single neuron, the approximation computed for the overall model may still be too loose to certify certain properties. One way to overcome this barrier is applying convex relaxations to multiple neurons, see for example the `DeepPoly` (Singh et al. 2019a), `kPoly` (Singh et al. 2019b), `OptC2V` (Tjandraatmadja et al. 2020), and `PRIMA` (Müller et al. 2021) methods. Another solution is using stronger relaxations beyond LPs, such as semidefinite program (SDP) relaxations (Raghunathan, Steinhardt, and Liang 2018; Fazlyab, Morari, and Pappas 2022; Batten et al. 2021).

Since the ReLU activation function can be equivalently represented by a set of linear/quadratic constraints, the robustness verification problem of ReLU NNs is well-suited to be solved via SDP methods. Nevertheless, the SDP methods are applicable for any feed-forward NN whose activation functions can be represented as quadratic constraints, *e.g.*, convolutional NNs with ReLU activations (Dathathri et al. 2020). SDP becomes useful when the over-approximations due to linear relaxations make the queries not solvable by LP methods. SDP also has the advantage of being able to verify nonlinear specifications such as quadratic input/output specifications (Qin et al. 2019; ul Abdeen et al. 2022).

SDP-based NN verification was first introduced in (Raghunathan, Steinhardt, and Liang 2018) and empirically shown to be considerably tighter than the standard LP relaxations. (Zhang 2020) provides a theoretical proof that SDP relaxations can be exact for a single hidden layer, but become loose for multiple hidden layers. It has further been shown that adding linear cuts (Batten et al. 2021), nonconvex cuts (Ma and Sojoudi 2020) or complete positivity constraints (Brown et al. 2022) can tighten the standard SDP relaxation. The standard SDP approaches (Raghunathan, Steinhardt, and Liang 2018; Zhang 2020; Ma and Sojoudi 2020; Brown et al. 2022) improve the relaxation tightness but also considerably increase the computational burden, leading to poor scalability for larger NNs. To reduce the computational demand, a memory-efficient SDP relaxation was introduced in (Dathathri et al. 2020) and layer SDP relaxations were used in (Batten et al. 2021; Newton and Papachristodoulou 2021) by exploiting the cascaded structures of NNs. The `LayerSDP` method (Batten et al. 2021) provides the presently tightest solution by combining an SDP relaxation with linear cuts. However, `LayerSDP` is still considerably loose for large networks, as observed from the results in (Batten et al. 2021). This gap leads to an increased number of unsolved verification queries as the NN size grows and thus limits the applicability of the `LayerSDP` method.

We advance SDP-based NN verification through a new layer SDP relaxation and a BaB algorithm. The proposed relaxation combines the `LayerSDP` method (Batten et al. 2021) with a set of eigenvector-based linear constraints, offering a provably tighter solution than the SoA SDP methods. We further develop a BaB algorithm to reduce the SDP relaxation gaps. Theoretically, we show that the solution returned by the BaB algorithm at any iteration is tighter than the SoA SDP methods, and can, in principle, converge to the exact solution of the verification problem. Our experimental results on various standard benchmarks confirm that the proposed layer SDP-based BaB method outperforms the present SoA, whilst retaining competitive efficiency.

Verification Problem and Convex Relaxations

Notation. We use the symbol $\mathbb{R}^n$ to denote the n-dimensional Euclidean space. We use $\mathrm{diag}(X)$ to stack the main diagonals of the matrix X as a column and use $X \bullet Y$ to represent $\mathrm{trace}(X^\top Y)$. We use $\odot$ to denote the element-wise product, and $\mathbb{I}_{[a,b]}$ to denote a sequence of non-negative integers from a to b. We use $\mathbf{0}_{n\times m}$ to denote an $n \times m$ zero matrix and $\mathbf{1}_n$ to denote a $n \times 1$ vector of ones. We use $\|\cdot\|_\infty$ to refer to the standard ℓ_∞ norm of a vector in $\mathbb{R}^n$. The symbol $P_i[z]$ represents the elements of matrix P_i corresponding to the vector or matrix z. s.t. is short for subject to.

Neural Networks (NNs) and Verification Problem. We consider the feed-forward fully-connected ReLU network $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$ with L hidden layers, n_0 inputs and n_{L+1} outputs. The network output is $f(x_0) = W_L x_L + b_L$, with the input x_0, the post-activation vectors $x_{i+1} = \mathrm{ReLU}(\hat{x}_{i+1})$ and pre-activation vectors $\hat{x}_{i+1} = W_i x_i + b_i$, $i \in \mathbb{I}_{[0,L-1]}$, where $W_i \in \mathbb{R}^{n_{i+1}\times n_i}$ and $b_i \in \mathbb{R}^{n_{i+1}}$, $i \in \mathbb{I}_{[0,L]}$, are the weights and biases, respectively. We focus on NNs for image classification whereby a given input x_0 is said to be class $j^\star$ if the corresponding output has the highest value: $j^\star = \arg\max_{j\in\mathbb{I}_{[1,n_{L+1}]}} f(x_0)_j$.

We study the local robustness verification problem. Given a network 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 $f(x_0)_{j^\star} - f(x_0)_j > 0$, for all $\|x_0 - \bar{x}\|_\infty \le \epsilon$ and $j \in \mathbb{I}_{[1,n_{L+1}]}$ with $j \neq j^\star$. The verification problem can be formulated and solved as an optimisation problem (Batten et al. 2021):

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

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 post-activation bounds that can be computed (before solving the optimisation problem) using bound propagation methods (Wang et al. 2018b; Henriksen and Lomuscio 2020). The problem (1) is solved for every potential adversarial target $j \in \mathbb{I}_{[1,n_{L+1}]}$ with $j \neq j^\star$. If $\gamma^* > 0$ in all the cases, the NN is verified to be robust on $\bar{x}$ under the perturbation ϵ.

SoA Convex Relaxations. Due to the presence of the nonlinear ReLU constraint (1a), the optimisation problem (1) is non-convex and generally hard to solve. A common strategy for overcoming this issue is to take a convex relaxation of the original problem which can then be efficiently solved. Solving the convex relaxation yields an optimal objective value γ^*_{cvx} as a valid lower bound to γ^*, *i.e.*, $\gamma^* \ge \gamma^*_{\text{cvx}}$. When $\gamma^*_{\text{cvx}} > 0$, we know that $\gamma^* > 0$ always holds and the network is verified to be robust for the given specification. However, when $\gamma^*_{\text{cvx}} \le 0$, no conclusion about the robustness of the network for the given case can be drawn. It could either be the case that a valid adversarial example exists, or that the relaxation we employed was too loose. Clearly, a tighter convex relaxation with a smaller gap $\gamma^* - \gamma^*_{\text{cvx}}$ can increase the number of verification queries solved. Therefore, tightness is the core design objective of a convex relaxation method given that its computational cost is acceptable.

The LP relaxation (Ehlers 2017), based on a triangle over-approximation of each ReLU constraint, is a widely used convex relaxation method for NN verification. The LP relaxation is easy to solve but usually results in a large relaxation gap due to the inherent convex relaxation barrier (Salman et al. 2019). This motivates us to develop a tighter convex relaxation method to improve the precision of our verification approach. Convex relaxations based on SDP (Raghunathan, Steinhardt, and Liang 2018; Batten et al. 2021) have recently been shown to be tighter than LP relaxations for verification. The SDP relaxation is based on a reformulation of the ReLU constraint (1a) as a set of linear and quadratic constraints:

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

The input and post-activation constraints in (1b) and (1c) are rewritten as the quadratic constraints:

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

where $l_0 = \bar{x} - \epsilon \mathbf{1}_{n_0}$ and $u_0 = \bar{x} + \epsilon \mathbf{1}_{n_0}$. Replacing (1a)-(1c) with (2) and (3) yields an equivalent quadratically constrained quadratic programming (QCQP) problem:

$$\gamma^* := \left\{ \min\ \ c^\top x_L + c_0 \mid (2), (3) \right\}. \tag{4}$$

This QCQP problem is still non-convex due to the quadratic constraints. The techniques of polynomial lifting (Parrilo 2000; Lasserre 2009) can be used to reformulate them as linear constraints, resulting in a convex SDP relaxation to the QCQP problem. SDP-based relaxations were first utilised for NN verification in (Raghunathan, Steinhardt, and Liang 2018) where a single positive semidefinite (PSD) matrix P is used to couple all the ReLU constraints of the network together. This approach leads to a global SDP relaxation to the non-convex optimisation problem (1) with potentially smaller relaxation gaps than the LP relaxation. The global SDP relaxation is computationally demanding for large NNs as P is high-dimensional, thus limiting its applicability.

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 $i \geq 0$. This cascading structure is exploited in (Batten et al. 2021) to derive a layer-based SDP relaxation with a set of PSD matrices defined as:

$$P_i = \mathbf{x}_i \mathbf{x}_i^\top,\ i \in \mathbb{I}_{[0,L-1]}, \tag{5}$$

where $\mathbf{x}_i = [1,\ x_i^\top, x_{i+1}^\top]^\top \in \mathbb{R}^{\bar{n}_i}$ and $\bar{n}_i = 1 + n_i + n_{i+1}$.

By using (5), the constraints (2) and (3) are reformulated as a set of linear constraints on the elements of P_i:

$$P_i[x_{i+1}] \geq 0, P_i[x_{i+1}] \geq W_i P_i[x_i] + b_i, i \in \mathbb{I}_{[0,L-1]}, \tag{6a}$$

$$\operatorname{diag}(P_i[x_{i+1}x_{i+1}^\top] - W_i P_i[x_i x_{i+1}^\top]) - b_i \odot P_i[x_{i+1}] = 0,\ i \in \mathbb{I}_{[0,L-1]}, \tag{6b}$$

$$\operatorname{diag}(P_i[x_i x_i^\top]) - (l_i + u_i) \odot P_i[x_i] + l_i \odot u_i \leq 0,\ i \in \mathbb{I}_{[0,L-1]}, \tag{6c}$$

$$\operatorname{diag}(P_{L-1}[x_L x_L^\top]) - (l_L + u_L) \odot P_{L-1}[x_L] + l_L \odot u_L \leq 0, \tag{6d}$$

$$P_i[\bar{x}_{i+1}\bar{x}_{i+1}^\top] = P_{i+1}[\bar{x}_{i+1}\bar{x}_{i+1}^\top],\ i \in \mathbb{I}_{[0,L-2]}, \tag{6e}$$

$$P_i[1] = 1,\ P_i \succeq 0,\ \operatorname{rank}(P_i) = 1,\ i \in \mathbb{I}_{[0,L-1]}, \tag{6f}$$

where $\bar{x}_{i+1} = [1,\ x_{i+1}^\top]^\top$. The constraint (6e) ensures the input-output consistency (Batten et al. 2021) and (6f) is equivalent to (5) as shown in (Horn and Johnson 2012).

By replacing (2) and (3) with (6) and dropping the non-convex rank constraint $\operatorname{rank}(P_i) = 1$, the QCQP problem, and equivalently the original verification problem (1), is relaxed to a convex layer SDP (Batten et al. 2021):

$$\begin{aligned} \gamma^*_{\text{LayerSDP}} := \min_{\{P_i\}_{i=0}^{L-1}}\ & c^\top P_{L-1}[x_L] + c_0 \\ \text{s.t.}\quad & (6a), (6b), (6c), (6d), (6e), && (7a) \\ & P_i[1] = 1,\ P_i \succeq 0,\ i \in \mathbb{I}_{[0,L-1]}, && (7b) \\ & P_i[x_{i+1}] \leq A_i P_i[x_i] + B_i,\ i \in \mathbb{I}_{[0,L-1]}, && (7c) \end{aligned}$$

where (7c) is reformulated from the triangle relaxation (Ehlers 2017) with $A_i = k_i \odot W_i$, $B_i = k_i \odot (b_i - \hat{l}_{i+1}) + \text{ReLU}(\hat{l}_{i+1})$, and $k_i = (\text{ReLU}(\hat{u}_{i+1}) - \text{ReLU}(\hat{l}_{i+1}))/(\hat{u}_{i+1} - \hat{l}_{i+1})$. The vectors $\hat{u}_{i+1}$ and $\hat{l}_{i+1}$ are the upper and lower bounds of $\hat{x}_{i+1}$, respectively. These bounds can be computed together with the post-activation bounds l_{i+1} and u_{i+1} using bound propagation methods.

It is shown in (Batten et al. 2021) that the layer SDP relaxation (7), using a set of PSD matrices P_i, $i \in \mathbb{I}_{[0,L-1]}$, with much smaller sizes, is equivalent to the global SDP relaxation (Raghunathan, Steinhardt, and Liang 2018) but more computationally efficient. Also, by introducing the linear cuts in (7c), the layer SDP relaxation is tighter, *i.e.*, $\gamma^*_{\text{GlobalSDP}} \leq \gamma^*_{\text{LayerSDP}} \leq \gamma^*$. However, due to the fact that the rank-one constraint has been removed, the relaxation gap of the layer SDP is still considerable in large NNs, as empirically shown in (Batten et al. 2021), thereby limiting the scalability and applicability of the approach. We address this issue by proposing a novel SDP relaxation in the next section which produces a tighter solution than the layer SDP.

A New Layer SDP Relaxation

In this section we develop a new, enhanced SDP relaxation to the non-convex problem (1) by adding eigenvector-based constraints to a reformulated problem of the layer SDP (7).

Reformulated Convex Layer SDP. We reformulate the QCQP problem (4) into a more compact convex layer SDP relaxation to easily analyse when the relaxation becomes tight. This modified form also makes it easy to derive additional constraints for tightening the relaxation. We start by rewriting the constraints (6b), (6c) and (6d) respectively as:

$$\mathbf{x}_i^\top Q^1_{i,j} \mathbf{x}_i + c^1_{i,j} \mathbf{x}_i = 0,\ j \in \mathbb{I}_{[1,n_{i+1}]},\ i \in \mathbb{I}_{[0,L-1]}, \tag{8a}$$

$$\mathbf{x}_0^\top Q^2_j \mathbf{x}_0 + c^2_j \mathbf{x}_0 + d^2_j \leq 0,\ j \in \mathbb{I}_{[1,n_0]}, \tag{8b}$$

$$\mathbf{x}_i^\top Q^3_{i,j} \mathbf{x}_i + c^3_{i,j} \mathbf{x}_i + d^3_{i,j} \leq 0, j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}, \tag{8c}$$

where $Q^1_{i,j} = \mathbf{0}_{\bar{n}_i \times \bar{n}_i}$, $Q^1_{i,j}(1+n_i+j, 1+n_i+j) = 1$, $Q^1_{i,j}(1+n_i+j, 2:1+n_i) = -W_i(j,:)/2$, $Q^1_{i,j}(2:1+n_i, 1+n_i+j) = -W_i(j,:)^\top/2$, $c^1_{i,j} = \mathbf{0}_{1\times\bar{n}_i}$, $c^1_{i,j}(1+n_i+j) = -b_i(j)$; $Q^2_j = \mathbf{0}_{\bar{n}_0 \times \bar{n}_0}$, $Q^2_j(1+j, 1+j) = 1$, $c^2_j = \mathbf{0}_{1\times\bar{n}_0}$, $c^2_j(1+j) = -(l_0(j) + u_0(j))$, $d^2_j = l_0(j)u_0(j)$; $Q^3_{i,j} = \mathbf{0}_{\bar{n}_i\times\bar{n}_i}$, $Q^3_{i,j}(1+n_i+j, 1+n_i+j) = 1$, $c^3_{i,j} = \mathbf{0}_{1\times\bar{n}_i}$, $c^3_{i,j}(1+n_i+j) = -(l_{i+1}(j) + u_{i+1}(j))$ and $d^3_{i,j} = l_{i+1}(j)u_{i+1}(j)$.

The matrices Q^2_j and $Q^3_{i,j}$ are always PSD as they have only the non-zero element 1 on the main diagonals. However, Proposition 1 shows that this is not the case for $Q^1_{i,j}$.

Proposition 1. *The matrix $Q_{i,j}^1$ is not necessarily positive definite but it has at most one negative eigenvalue.*

Proof. We represent $Q_{i,j}^1$ as a symmetric block matrix $Q_{i,j}^1 = \begin{bmatrix} H_{i,j}^1 & \mathbf{0}_{(n_{i+1}-j)\times 1} \\ \mathbf{0}_{1\times(n_{i+1}-j)} & \mathbf{0}_{(n_{i+1}-j)\times(n_{i+1}-j)} \end{bmatrix}$, where $H_{i,j}^1 = \begin{bmatrix} D & z \\ z^\top & \alpha \end{bmatrix}$, $D = \mathbf{0}_{(n_i+j)\times(n_i+j)}$, $\alpha = 1$ and $z = \begin{bmatrix} 0, & -W_i(j,:)^\top/2, & \mathbf{0}_{(j-1)\times 1} \end{bmatrix}$. The eigenvalues of $Q_{i,j}^1$ include those of the symmetric arrowhead matrix $H_{i,j}^1$ and the $(n_{i+1}-j)$ zero eigenvalues. Let the eigenvalues of $H_{i,j}^1$ be $\lambda_1 \le \cdots \le \lambda_{n_{i+1}-j}$. By Cauchy's interlacing theorem, the eigenvalues of $H_{i,j}^1$ interlace the eigenvalues of the matrix formed by deleting the last row and column of $H_{i,j}^1$ (O'leary and Stewart 1990). Hence, we know that $\lambda_1 \le d_1 \le \lambda_2 \le \cdots \le d_{n_i+j} \le \lambda_{n_{i+1}-j}$, where $d_1, \cdots, d_{n_i+j}$ are the diagonal elements of D. Since D is a zero matrix, the above relation implies that $\lambda_1 \le 0 = \lambda_2 = \cdots = \lambda_{n_{i+1}-j}$. Therefore, the matrix $Q_{i,j}^1$ may not be positive definite and it has at most one negative eigenvalue λ_1. □

Under the constraints in (6f), we know $P_i = \mathbf{x}_i\mathbf{x}_i^\top,\ i \in \mathbb{I}_{[0,L-1]}$. Hence, the following equations hold:

$$\begin{aligned} \mathbf{x}_i^\top Q_{i,j}^1 \mathbf{x}_i &= Q_{i,j}^1 \bullet (\mathbf{x}_i\mathbf{x}_i^\top) = Q_{i,j}^1 \bullet P_i, \\ \mathbf{x}_0^\top Q_j^2 \mathbf{x}_0 &= Q_j^2 \bullet (\mathbf{x}_0\mathbf{x}_0^\top) = Q_j^2 \bullet P_0, \\ \mathbf{x}_i^\top Q_{i,j}^3 \mathbf{x}_i &= Q_{i,j}^3 \bullet (\mathbf{x}_i\mathbf{x}_i^\top) = Q_{i,j}^3 \bullet P_i. \end{aligned} \tag{9}$$

By substituting (8) with (9) into (6) and dropping the rank-one constraint, it yields the new layer SDP relaxation:

$$\gamma^*_{\text{LayerSDP}} := \min_{\{P_i\}_{i=0}^{L-1}} c^\top P_{L-1}[x_L] + c_0$$

$$\text{s.t. } (6a), (6e), (7b), (7c), \tag{10a}$$

$$Q_{i,j}^1 \bullet P_i + c_{i,j}^1 P_i[\mathbf{x}_i] = 0, j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}, \tag{10b}$$

$$Q_j^2 \bullet P_0 + c_j^2 P_0[\mathbf{x}_0] + d_j^2 \le 0,\ j \in \mathbb{I}_{[1,n_0]}, \tag{10c}$$

$$Q_{i,j}^3 \bullet P_i + c_{i,j}^3 P_i[\mathbf{x}_i] + d_{i,j}^3 \le 0,\ j \in \mathbb{I}_{[1,n_{i+1}]},\ i \in \mathbb{I}_{[0,L-1]}. \tag{10d}$$

The only difference between the layer SDP relaxations (7) and (10) is that the constraints (6b), (6c) and (6d) are replaced by (10b), (10c) and (10d), respectively. Due to dropping the rank-one constraint, there generally exists a non-zero gap between the QCQP (4) and (10). We show below that the gap can be closed via adding extra constraints.

Enhanced Non-convex Layer SDP. Lemma 1 gives the conditions when the SDP relaxation (10) becomes exact.

Lemma 1. *Let P_i, $i \in \mathbb{I}_{[0,L-1]}$, be the optimal solution to the layer SDP relaxation* (10) *that satisfies the conditions:*

$$Q_{i,j}^1 \bullet P_i \ge \mathbf{x}_i^\top Q_{i,j}^1 \mathbf{x}_i, j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}, \tag{11a}$$

$$Q_j^2 \bullet P_0 \ge \mathbf{x}_0^\top Q_j^2 \mathbf{x}_0,\ j \in \mathbb{I}_{[1,n_0]}, \tag{11b}$$

$$Q_{i,j}^3 \bullet P_i \ge \mathbf{x}_i^\top Q_{i,j}^3 \mathbf{x}_i, j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}. \tag{11c}$$

Then P_i, $i \in \mathbb{I}_{[0,L-1]}$, is the optimal solution for the QCQP problem (4).

Proof. Under the given conditions, we know the relations:

$$\begin{aligned} &\mathbf{x}_i^\top Q_{i,j}^1 \mathbf{x}_i + c_{i,j}^1 \mathbf{x}_i \le Q_{i,j}^1 \bullet P_i + c_{i,j}^1 P_i[\mathbf{x}_i] = 0, \\ &j \in \mathbb{I}_{[1,n_{i+1}]},\ i \in \mathbb{I}_{[0,L-1]}; \\ &\mathbf{x}_0^\top Q_j^2 \mathbf{x}_0 + c_j^2 \mathbf{x}_0 + d_j^2 \le Q_j^2 \bullet P_0 + c_j^2 P_0[\mathbf{x}_0] + d_j^2 \le 0, \\ &j \in \mathbb{I}_{[1,n_0]}; \\ &\mathbf{x}_i^\top Q_{i,j}^3 \mathbf{x}_i + c_{i,j}^3 \mathbf{x}_i + d_{i,j}^3 \le Q_{i,j}^3 \bullet P_i + c_{i,j}^3 P_i[\mathbf{x}_i] + d_{i,j}^3 \\ &\quad \le 0,\ j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}. \end{aligned}$$

Since $\mathbf{x}_i^\top Q_{i,j}^1 \mathbf{x}_i + c_{i,j}^1 \mathbf{x}_i \ge 0$, the above relations imply that the optimal solution P_i, $i \in \mathbb{I}_{[0,L-1]}$, to the layer SDP relaxation (10) satisfies the original nonlinear constraints (2) and (3). Therefore, the solution P_i, $i \in \mathbb{I}_{[0,L-1]}$, is also the optimal solution for the QCQP problem. □

If the matrices $Q_{i,j}^1$, Q_j^2 and $Q_{i,j}^3$ are PSD, then (11) holds automatically under $P_i \succeq 0$. According to (8), Q_j^2 and $Q_{i,j}^3$ are PSD and thus (11b) and (11c) hold automatically. However, since $Q_{i,j}^1$ may be indefinite, it would be necessary to impose extra constraints enforcing condition (11a) to ensure exactness of the layer SDP relaxation (10). To construct the extra constraints, we perform a spectral decomposition (Lu et al. 2019) of the matrix $Q_{i,j}^1$ and get

$$Q_{i,j}^1 = \sum_{r_1 \in \mathcal{P}_{i,j}} \lambda_{i,j}^{r_1} V^{r_1} - \sum_{r_2 \in \mathcal{N}_{i,j}} \lambda_{i,j}^{r_2} V^{r_2}, \tag{12}$$

where $V^{r_1} = v_{i,j}^{r_1}(v_{i,j}^{r_1})^\top$ and $V^{r_2} = v_{i,j}^{r_2}(v_{i,j}^{r_2})^\top$. $\mathcal{P}_{i,j}$ and $\mathcal{N}_{i,j}$ are the index sets of all positive eigenvalues $\lambda_{i,j}^{r_1}$ and negative eigenvalues $-\lambda_{i,j}^{r_2}$ of $Q_{i,j}^1$, respectively.

By applying (12), (11a) is equivalently represented as

$$\Big(\sum_{r_1 \in \mathcal{P}_{i,j}} \lambda_{i,j}^{r_1} V^{r_1} - \sum_{r_2 \in \mathcal{N}_{i,j}} \lambda_{i,j}^{r_2} V^{r_2}\Big) \bullet \big(P_i - \mathbf{x}_i\mathbf{x}_i^\top\big) \ge 0. \tag{13}$$

Let $\tilde{x}_i = [x_i^\top, x_{i+1}^\top]^\top$ and $\tilde{X}_i = \tilde{x}_i\tilde{x}_i^\top$. The PSD constraint $P_i \succeq 0$ is equivalent to $\tilde{X}_i \succeq \tilde{x}_i\tilde{x}_i^\top$ by using the Schur complement, and further equivalent to $P_i \succeq \mathbf{x}_i\mathbf{x}_i^\top$ as $P_i[1] = 1$ and $P_i[\tilde{x}_i] = \tilde{x}_i$. This implies that $\big(\sum_{r_1 \in \mathcal{P}_{i,j}} \lambda_{i,j}^{r_1} V^{r_1}\big) \bullet \big(P_i - \mathbf{x}_i\mathbf{x}_i^\top\big) \ge 0$. Therefore, we can ensure (13), *i.e.*, the condition (11a), by enforcing

$$\Big(\sum_{r_2 \in \mathcal{N}_{i,j}} \lambda_{i,j}^{r_2} V^{r_2}\Big) \bullet \big(P_i - \mathbf{x}_i\mathbf{x}_i^\top\big) = 0. \tag{14}$$

The above analysis motivates us to add the eigenvector-based constraint (14) into (10) to obtain the new layer SDP:

$$\gamma^*_{\text{QCQP2}} := \min_{\{P_i\}_{i=0}^{L-1}} c^\top P_{L-1}[x_L] + c_0$$

$$\text{s.t. } (10a), (10b), (10c), (10d), \tag{15a}$$

$$\big(v_{i,j}^r (v_{i,j}^r)^\top\big) \bullet P_i = \big((v_{i,j}^r)^\top P_i[\mathbf{x}_i]\big)^2, \quad r \in \mathcal{N}_{i,j},\ j \in \mathbb{I}_{[1,n_{i+1}]},\ i \in \mathbb{I}_{[0,L-1]}. \tag{15b}$$

In view of Lemma 1, we have the results in Theorem 1 showing that with the extra constraints in (15b), the new non-convex layer SDP (15) is equivalent to the QCQP (4) .

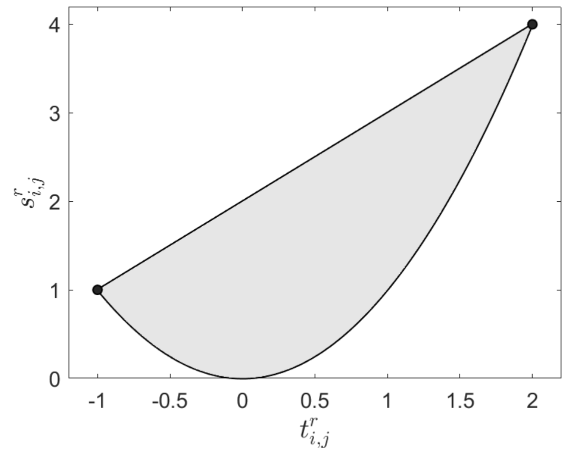

Figure 1: $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]}$ (the shaded area) when $\mathbf{l}^r_{i,j}=-1$ and $\mathbf{u}^r_{i,j}=2$.

Theorem 1. *The optimal solution P_i, $i \in \mathbb{I}_{[0,L-1]}$, of* (15) *is also optimal to* (4)*, i.e., $\gamma^*_{LayerSDP} \le \gamma^*_{QCQP2} = \gamma^*$.*

The extra constraints in (15b) result in a tighter layer SDP, but they are nonlinear, making the new layer SDP problem (15) non-convex and harder to solve than the original layer SDP relaxation (7). To address this issue, we derive a convex relaxation of (15) below.

Enhanced Convex Layer SDP Relaxation. We can replace the constraint (15b) by $s^r_{i,j} = (t^r_{i,j})^2$, with $s^r_{i,j} = \big(v^r_{i,j}(v^r_{i,j})^\top\big) \bullet P_i$ and $t^r_{i,j} = (v^r_{i,j})^\top P_i[\mathbf{x}_i]$. Since we know the bounds of the vector $\mathbf{x}_i$, $i \in \mathbb{I}_{[0,L-1]}$, we can compute the lower bound $\mathbf{l}^r_{i,j}$ and upper bound $\mathbf{u}^r_{i,j}$ of $t^r_{i,j}$, $r \in \mathcal{N}_{i,j}, j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}$ as $\mathbf{l}^r_{i,j} = \min\ (v^r_{i,j})^\top \mathbf{x}_i$ and $\mathbf{u}^r_{i,j} = \max\ (v^r_{i,j})^\top \mathbf{x}_i$. The non-convex constraint $s^r_{i,j} = (t^r_{i,j})^2$ over $\mathbf{l}^r_{i,j} \le t^r_{i,j} \le \mathbf{u}^r_{i,j}$ can be relaxed by constructing the convex hull of the original feasible set as $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]} := \mathrm{Conv}\{(t^r_{i,j}, s^r_{i,j}) \mid s^r_{i,j} = (t^r_{i,j})^2, \mathbf{l}^r_{i,j} \le t^r_{i,j} \le \mathbf{u}^r_{i,j}\}$. The convex set $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]}$ contains the two points $(\mathbf{l}^r_{i,j}, (\mathbf{l}^r_{i,j})^2)$ and $(\mathbf{u}^r_{i,j}, (\mathbf{u}^r_{i,j})^2)$. The line connecting the two end points of the parabola is $s^r_{i,j} = (\mathbf{l}^r_{i,j}+\mathbf{u}^r_{i,j})t^r_{i,j} - \mathbf{l}^r_{i,j}\mathbf{u}^r_{i,j}$. Hence, the set $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]}$ can be represented by $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]} = \{(t^r_{i,j}, s^r_{i,j}) \mid s^r_{i,j} \ge (t^r_{i,j})^2, s^r_{i,j} \le (\mathbf{l}^r_{i,j}+\mathbf{u}^r_{i,j})t^r_{i,j} - \mathbf{l}^r_{i,j}\mathbf{u}^r_{i,j}\}$. An illustration of this set is provided in Figure 1.

Under the PSD constraint $P_i \succeq 0, i \in \mathbb{I}_{[0,L-1]}$, the relation $P_i \succeq \mathbf{x}_i\mathbf{x}_i^\top$ is satisfied automatically and so is $s^r_{i,j} \ge (t^r_{i,j})^2$. By removing this redundant part, the non-convex layer SDP (15) is relaxed to the following problem:

$$
\begin{aligned}
\gamma^*_{\text{LayerSDP2}} &:= \min_{\{P_i\}_{i=0}^{L-1}} c^\top P_{L-1}[x_L] + c_0 \\
\text{s.t. } & (10a), (10b), (10c), (10d), && \text{(16a)} \\
& s^r_{i,j} = \big(v^r_{i,j}(v^r_{i,j})^\top\big) \bullet P_i,\ t^r_{i,j} = (v^r_{i,j})^\top P_i[\mathbf{x}_i], \\
& s^r_{i,j} \le (\mathbf{l}^r_{i,j} + \mathbf{u}^r_{i,j})t^r_{i,j} - \mathbf{l}^r_{i,j}\mathbf{u}^r_{i,j}, \\
& r \in \mathcal{N}_{i,j},\ j \in \mathbb{I}_{[1,n_{i+1}]},\ i \in \mathbb{I}_{[0,L-1]}. && \text{(16b)}
\end{aligned}
$$

We show the properties of the problem (16) in Theorem 2.

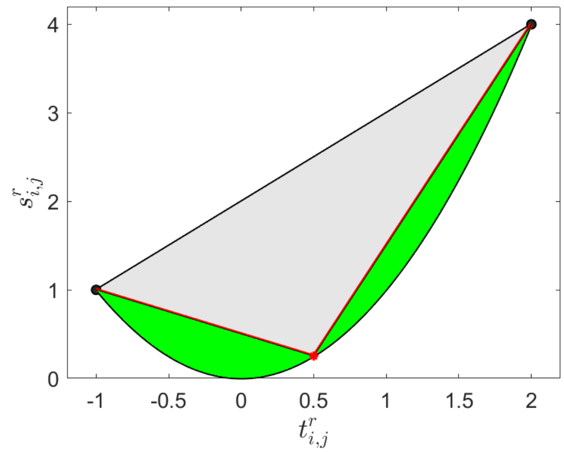

Figure 2: An illustration of BaB step reducing the over-approximation introduced by the set $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]}$ by splitting the interval $t^r_{i,j} \in [\mathbf{l}^r_{i,j}, \mathbf{u}^r_{i,j}]$ into two equal halves when $\mathbf{l}^r_{i,j} = -1$ and $\mathbf{u}^r_{i,j} = 2$. The branching step significantly reduces the total area of $\mathcal{P}_{[\mathbf{l}^r_{i,j},\mathbf{u}^r_{i,j}]}$ by removing the shaded area. After the split, only the two green areas are part of the feasible set.

Theorem 2. *The relation $\gamma^*_{LayerSDP} \le \gamma^*_{LayerSDP2} \le \gamma^*_{QCQP2} = \gamma^*$ holds. If an optimal solution of* (16) *satisfies $s^r_{i,j} = (t^r_{i,j})^2$, $\forall r \in \mathcal{N}_{i,j}, j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}$, then it is also optimal to QCQP* (4) *and $\gamma^*_{LayerSDP2} = \gamma^*_{QCQP2} = \gamma^*$.*

Proof. We prove the first part from three aspects:

1) $\gamma^*_{LayerSDP} \le \gamma^*_{LayerSDP2}$: Compared to the existing layer SDP (7), our new layer SDP (16) has extra constraints (16b), making its solution at least as tight as that of (7).

2) $\gamma^*_{LayerSDP2} \le \gamma^*_{QCQP2}$: This holds because we linearise the non-convex constraints (15b) to be (16b).

3) $\gamma^*_{QCQP2} = \gamma^*$: Theorem 1 shows that the non-convex layer SDP (15) is shown to be equivalent to the QCQP (4).

If an optimal solution of (16) satisfies $s^r_{i,j} = (t^r_{i,j})^2$, $\forall j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}$, then (16b) represents an exact relaxation of the non-convex constraints (15b). Hence, this optimal solution is also optimal to (15) and (4). We thus have $\gamma^*_{\text{LayerSDP2}} = \gamma^*_{\text{QCQP2}}$ and subsequently $\gamma^*_{\text{LayerSDP2}} = \gamma^*$. □

Theorem 2 shows that the new layer SDP relaxation (16) provides a tighter lower bound to the original verification problem when compared with the existing layer SDP relaxation (7). The lower bound can be further tightened by using the BaB method to be developed below.

A BaB Method for Solving the New Layer SDP

In this section we present a BaB method to compute the optimal solution of the non-convex layer SDP (15). As shown in Theorem 2, the relaxation (16) is exact when $s^r_{i,j} = (t^r_{i,j})^2$, $\forall r \in \mathcal{N}_{i,j},\ j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}$. If $s^r_{i,j} > (t^r_{i,j})^2$ for some $r \in \mathcal{N}_{i,j},\ j \in \mathbb{I}_{[1,n_{i+1}]}, i \in \mathbb{I}_{[0,L-1]}$, then we can split the interval $t^r_{i,j} \in [\mathbf{l}^r_{i,j}, \mathbf{u}^r_{i,j}]$ into two smaller intervals $t^r_{i,j} \in [\mathbf{l}^r_{i,j}, (\mathbf{l}^r_{i,j} + \mathbf{u}^r_{i,j})/2]$ and $t^r_{i,j} \in [(\mathbf{l}^r_{i,j} + \mathbf{u}^r_{i,j})/2, \mathbf{u}^r_{i,j}]$

to construct two new problems, so that the convex relaxation of $s^r_{i,j} = (t^r_{i,j})^2$ can be tighter over these smaller intervals. We illustrate this through the example in Figure 2. We choose to split the interval $t^r_{i,j} \in [\mathbf{l}^r_{i,j}, \mathbf{u}^r_{i,j}]$ at its middle point $(\mathbf{l}^r_{i,j} + \mathbf{u}^r_{i,j})/2$, because this produces the tightest relaxation, as shown in (Deng et al. 2018, Lemma 3.1).

The overall BaB method is presented in Algorithm 1. As shown in Line 18 of the algorithm, we choose to branch the interval with the largest distance $d^r_{i,j} = |\lambda^r_{i,j}(s^r_{i,j} - (t^r_{i,j})^2)|$. This choice of interval is inspired by the intuition that the violation of the equality constraint is largest in this direction and thus the splitting is (locally) optimal. Note that a (globally) optimal strategy of choosing the neurons to split is of independent research interest in the area of NN verification, and it is not the focus of this work. The tightness of the solution obtained by Algorithm 1 is characterised in Theorem 3.

Theorem 3. *Algorithm 1 returns the objective value* γ_{bab} *satisfying:* $\gamma^*_{LayerSDP} \leq \gamma^*_{LayerSDP2} \leq \gamma_{bab} \leq \gamma^*_{QCQP2} = \gamma^*$.

Proof. We perform each branching step by splitting the interval $t^r_{i,j} \in [\mathbf{l}^r_{i,j}, \mathbf{u}^r_{i,j}]$ into two equal halves. Solving the two problems associated with the smaller intervals gives a lower bound (objective value) that is not worse than the one before branching. Since the branch and bound is initialised using the optimal solution to the layer SDP relaxation (16), we have $\gamma^*_{\text{LayerSDP2}} \leq \gamma_{\text{bab}}$. According to Theorem 2, we also have $\gamma^*_{\text{LayerSDP}} \leq \gamma^*_{\text{LayerSDP2}} \leq \gamma_{\text{bab}} \leq \gamma^*_{\text{QCQP2}} = \gamma^*$. □

We terminate Algorithm 1 whenever $\gamma_{\text{bab}} > 0$, because any positive lower bound is sufficient to certify that the NN is robust for an input under the perturbation ϵ. In principle, we can keep branching and bounding until γ_{bab} reaches the ground true optimal objective value γ^*. In this sense, the proposed BaB algorithm is considered to be complete. In practice, we may set a maximum number of iterations in analogy with the timeout used in other BaB-based methods (Anderson et al. 2020; Kouvaros and Lomuscio 2021; Wang et al. 2021; Bunel et al. 2020).

The proposed BaB algorithm achieves bounding via solving the layer SDP problem (16). Hence, computational complexity of our approach depends on the number of layer SDP problems to solve and the complexity of solving each problem. We solve the layer SDP using MOSEK (Andersen and Andersen 2000) which implements an interior-point algorithm. Before calling MOSEK, we formulate the SDP as the widely-used conic optimisation form where all inequality constraints are converted into equalities (Sturm 1999). According to (Nesterov 2003), solving a SDP at each iteration of the interior-point algorithm requires $\mathcal{O}(n^3 m + n^2 m^2 + m^3)$ time and $\mathcal{O}(n^2 + m^2)$ memory, where n is the dimension of the positive semidefinite matrix and m is the number of equality constraints. For the layer SDP problem (16), $n = 1 + \max_{i=0,\ldots,L-1}(\bar{n}_i + \bar{n}_{i+1})$ is the largest dimension of all P_i with $\bar{n}_i = n_i - s_i$ being the number of neurons at layer i after removing s_i inactive neurons, and $m = 2L + 4\sum_{i=1}^{L} \bar{n}_i + \sum_{i=0}^{L-1}(\bar{n}_i + g_i n_{i+1}) + \sum_{i=1}^{L-1} \bar{n}_i$ with g_i being the number of eigenvector-based constraints associated with P_i. The layer SDP (Batten et al. 2021) was also solved using MOSEK with $n = 1 + \max_{i=0,\ldots,L-1}(\bar{n}_i + \bar{n}_{i+1})$ and $m = 2L + 4\sum_{i=1}^{L} \bar{n}_i + \sum_{i=0}^{L-1} \bar{n}_i + \sum_{i=1}^{L-1} \bar{n}_i$. Hence, our layer SDP problem (16) has slightly higher computational cost than the one in (Batten et al. 2021), with $\sum_{i=0}^{L-1}(g_i n_{i+1})$ more equality constraints.

Algorithm 1: Layer SDP-based BaB algorithm

Input: NN parameters, perturbation radius ϵ, test input

1: Compute the bounds $\{\hat{l}_i\}_{i=1}^L$, $\{\hat{u}_i\}_{i=1}^L$, $\{l_i\}_{i=1}^L$ and $\{u_i\}_{i=1}^L$ using a bound propagation method.

2: Perform spectral decomposition of $Q^1_{i,j}, j \in \mathbb{I}_{[1,n_{i+1}]}$, $i \in \mathbb{I}_{[0,L-1]}$, and construct the sets of negative eigenvalues $\mathcal{N}^0 = \{\mathrm{E}_0, \cdots, \mathrm{E}_{L-1}\}$ with $\mathrm{E}_i = \{\lambda^r_{i,j}\}$, and the corresponding eigenvectors $\mathcal{V}^0 = \{\mathrm{V}_1, \cdots, \mathrm{V}_{L-1}\}$ with $\mathrm{V}_i = \{v^r_{i,j}\}$, P_i indices $\mathcal{I}^0 = \{\mathrm{I}_0, \cdots, \mathrm{I}_{L-1}\}$ with $\mathrm{I}_i = i \times \mathbf{1}_{1\times r}$, and bounds $\mathcal{L}^0 = \{\mathrm{L}_0, \cdots, \mathrm{L}_{L-1}\}$ and $\mathcal{U}^0 = \{\mathrm{U}_0, \cdots, \mathrm{U}_{L-1}\}$ with $\mathrm{L}_i = \{\mathbf{l}^r_{i,j}\}$ and $\mathrm{U}_i = \{\mathbf{u}^r_{i,j}\}$, $r \in \mathcal{N}_{i,j}$.

3: Solve `Problem 0`, (16) with the parameter set $(\mathcal{N}^0, \mathcal{V}^0, \mathcal{I}^0, \mathcal{L}^0, \mathcal{U}^0)$, for the objective value γ^0 and the distance set $\mathcal{D}^0 = \{\mathrm{D}_0, \cdots, \mathrm{D}_{L-1}\}$ with $\mathrm{D}_i = \{d^r_{i,j}\}$, $d^r_{i,j} = |\lambda^r_{i,j}(s^r_{i,j} - (t^r_{i,j})^2)|$, $r \in \mathcal{N}_{i,j}$.

4: Set $\gamma_{\text{bab}} = \gamma^0$, which is NaN when (16) is infeasible.

5: **if** (`Problem 0` is infeasible) $\vee (\gamma^0 > 0) \vee (\mathcal{N}^0 == \emptyset)$ **then** Return γ_{bab} and stop algorithm.

6: **end if**

7: Set $k = 0$. Initialise $\mathcal{P}$ as the set of active problems by adding the instance $\{\mathcal{N}^0, \mathcal{V}^0, \mathcal{I}^0, \mathcal{L}^0, \mathcal{U}^0, \mathcal{D}^0, \gamma^0\}$ to $\mathcal{P}$.

8: **loop**

9: Set $k = k + 1$.

10: **if** $\mathcal{P} == \emptyset$ **then** Return γ_{bab} and stop algorithm.

11: **end if**

12: Choose and remove from $\mathcal{P}$ the instance with the smallest objective value and denote it as $\mathcal{P}^k = \{\mathcal{N}^k, \mathcal{V}^k, \mathcal{I}^k, \mathcal{L}^k, \mathcal{U}^k, \mathcal{D}^k, \gamma^k\}$. Set $\gamma_{\text{bab}} = \gamma^k$.

13: **if** $\gamma_{\text{bab}} > 0$ **then** Return γ_{bab} and stop algorithm.

14: **end if**

15: Set $z_{i^*} = (\mathcal{L}^k_{i^*} + \mathcal{U}^k_{i^*})/2$ with i^* indexing the largest element in $\mathcal{D}^k$. Construct the parameter sets $\mathcal{S}_1 := (\mathcal{N}^k, \mathcal{V}^k, \mathcal{I}^k, \mathcal{L}^k, \bar{\mathcal{U}}^k)$ and $\mathcal{S}_2 := (\mathcal{N}^k, \mathcal{V}^k, \mathcal{I}^k, \bar{\mathcal{L}}^k, \mathcal{U}^k)$, where $\bar{\mathcal{U}}^k_{i^*} = z_{i^*}$ and $\bar{\mathcal{L}}^k_{i^*} = z_{i^*}$ when $i = i^*$, and $\bar{\mathcal{U}}^k_i = \mathcal{U}^k_i$ and $\bar{\mathcal{L}}^k_i = \mathcal{L}^k_i$ for all $i \neq i^*$.

16: Solve `Problem 1`, (16) with $\mathcal{S}_1$, for the objective value γ_1 and distance set $\mathcal{D}_1$.

17: **if** `Problem 1` is feasible **then**

18: Add instance $\{\mathcal{N}^k, \mathcal{V}^k, \mathcal{I}^k, \mathcal{L}^k, \bar{\mathcal{U}}^k, \mathcal{D}_1, \gamma_1\}$ to $\mathcal{P}$.

19: **end if**

20: Solve `Problem 2`, (16) with $\mathcal{S}_2$, for the objective value γ_2 and distance set $\mathcal{D}_2$.

21: **if** `Problem 2` is feasible **then**

22: Add instance $\{\mathcal{N}^k, \mathcal{V}^k, \mathcal{I}^k, \bar{\mathcal{L}}^k, \mathcal{U}^k, \mathcal{D}_2, \gamma_2\}$ to $\mathcal{P}$.

23: **end if**

24: **end loop**

Output: γ_{bab}

		BaBSDP		LayerSDP		SDP-IP		SDP-FO		LP		PRIMA	β-CROWN		OVAL	
Models	PGD	ver.	t	ver.	t	ver.	t	ver.†	t^*	ver.	t	ver.†	ver.	t	ver.	t
MLP-Adv	94	**92**	2161	91	2036	82	12079	84	17230	7	2	–	88	8	11	1.8
MLP-LP	80	**80**	284	**80**	260	**80**	50733	78	21669	78	0.5	–	**80**	0.1	20	0.1
MLP-SDP	84	**84**	6800	**84**	6643	80	43156	64	22871	6	7	–	76	65	18	21
MLP-6×100	91	**80**	1379	75	1034	52	6760	◇	–	0	2.8	51.0	78	238	–	–
MLP-9×100	86	47	625	35	446	22	2148	◇	–	1	4.4	42.8	**69**	461	–	–
MLP-6×200	96	**93**	3875	92	3180	76	25850	◇	–	3	6.3	69	84	190	–	–
CF-3 × 20	20	**18**	5315	17	3663	16	7804	◇	–	0	0.3	–	17	0.3	–	–

Table 1: Verified robustness (ver., in percentage) and runtime per image (t, in seconds) for a set of benchmarks with various sizes. Dagger (†): these numbers are directly taken from the literature: SDP-FO from (Batten et al. 2021) for the same 100 images, and PRIMA from (Müller et al. 2021) for 1000 images. Star ($*$): the runtime is estimated by evaluating 3 images. Dash (–): previously reported or re-implementation results are unavailable. Diamond (◇): the methods fail to verify any instance.

Experimental Evaluation

We conducted experiments on a Linux machine running an AMD Ryzen Threadripper 3970X 32-Core CPU with 256 GB RAM and a RTX 3090 GPU. The optimisation problems were solved using the SDP solver MOSEK.

We evaluated the proposed method on several fully-connected ReLU NNs (where "$m \times n$" means a NN with $m-1$ hidden layers each having n neurons):

1) Three small size MNIST models MLP-Adv, MLP-LP, and MLP-SDP are from (Raghunathan, Steinhardt, and Liang 2018) and tested under $\epsilon = 0.1$ as in (Raghunathan, Steinhardt, and Liang 2018; Batten et al. 2021).
2) Three medium size MNIST models MLP-6 × 100, MLP-9 × 100 and MLP-6 × 200 are from (Singh et al. 2019a) and evaluated under the same $\epsilon = 0.026, 0.026, 0.015$, respectively, as in (Singh et al. 2019a; Müller et al. 2021; Batten et al. 2021; Wang et al. 2021).
3) One CIFAR10 model CF-3 × 20 is from (Li et al. 2020) and evaluated under $\epsilon = 2/255$.

We compared the proposed layer SDP-based BaB method (referred to as BaBSDP) against several SoA incomplete methods for verification:

1) BaB-based methods: β-CROWN (Wang et al. 2021) and OVAL (Bunel et al. 2020), both of which were shown to be stronger BaB-based verifiers (β-CROWN is winner) in the VNN-COMP 2021 (Bak, Liu, and Johnson 2021).
2) Linear relaxations: the standard linear program relaxation LP (Ehlers 2017) and its variant PRIMA (Müller et al. 2021).
3) 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 received permission to run VeriNet (Henriksen and Lomuscio 2020) for obtaining input bounds. We manually passed these to Matlab and used them as an input to run Algorithm 1. All experiments were run on the first 100 images of the MNIST/CIFAR10 datasets. The PGD upper bounds of the MNIST models are taken from (Batten et al. 2021), while that of the CIFAR10 model is from (Li et al. 2020). We report the results in Table 1, where the verified robustness represents the percentage of images that are verified to be robust and the runtime represents the average MOSEK solver time for verifying an image.

Our results show that BaBSDP is more precise than the SoA SDP-based methods for the models MLP-6 × 100, MLP-9 × 100 and CF-3 × 20. BaBSDP achieves the same precision as LayerSDP for the other three benchmarks. For the MLP-LP and MLP-SDP networks, both BaBSDP and LayerSDP reach the PGD upper bounds. Compared to the baselines that are based on linear relaxations (LP and PRIMA) and BaB (OVAL), BaBSDP is more precise for all the models. Compared to the SoA BaB method β-CROWN, BaBSDP is only less precise for the model MLP-9 × 100. As expected we found BaBSDP needs a larger amount of runtime than LayerSDP, LP, OVAL and β-CROWN (which used GPU support) across all the networks. However, BaBSDP resulted to be faster than SDP-IP for all the networks. According to the results reported in (Batten et al. 2021), SDP-FO is less efficient than LayerSDP for MLP-Adv and MLP-LP, and fails to verify MLP-6×100 and MLP-9 × 100. These results confirm that our proposed BaBSDP improves the verification precision, whilst retaining a competitive computational efficiency.

Conclusions

A number of SDP relaxation-based incomplete methods have been proposed in the literature to provide tight approximations of nonlinear activation functions for verifying large NNs. However, the relaxation gaps for the present SoA are still considerable, leading to many verification queries remaining unsolved. In this paper, we proposed a new SDP relaxation with improved tightness by integrating a set of eigenvector-based constraints to the layer SDP relaxtion method. Building on this novel SDP relaxation, we developed a BaB algorithm to further reduce the relaxation gaps. Our theoretical analysis showed that the method yields tighter relaxations than the SoA SDP methods for NN verification. The experimental results demonstrated that our method achieved SoA performance on various commonly used benchmarks. This work opens up further work into combining BaB with SDP for NN verification.

Acknowledgements

Jianglin Lan is supported by a Leverhulme Trust Early Career Fellowship under Award ECF-2021-517. Benedikt Brückner is supported by the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence [grant number EP/S023356/1]. 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.

Brown, R. A.; Schmerling, E.; Azizan, N.; and Pavone, M. 2022. A Unified View of SDP-based Neural Network Verification through Completely Positive Programming. In *International Conference on Artificial Intelligence and Statistics (AISTATS22)*, 9334–9355. PMLR.

Bunel, R.; Mudigonda, P.; Turkaslan, I.; Torr, P.; Lu, J.; and Kohli, P. 2020. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(2020).

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.

Deng, Z.; Fang, S.-C.; Lu, C.; and Guo, X. 2018. A branch-and-cut algorithm using polar cuts for solving nonconvex quadratic programming problems. *Optimization*, 67(2): 359–375.

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*.

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.

Kouvaros, P.; and Lomuscio, A. 2021. Towards Scalable Complete Verification of ReLU Neural Networks via Dependency-based Branching. In *International Joint Conference on Artificial Intelligence (IJCAI21)*, 2643–2650.

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

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.

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

Lu, C.; Deng, Z.; Zhou, J.; and Guo, X. 2019. A sensitive-eigenvector based global algorithm for quadratically constrained quadratic programming. *Journal of Global Optimization*, 73(2): 371–388.

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.

O'leary, D.; and Stewart, G. 1990. Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices. *Journal of Computational Physics*, 90(2): 497–505.

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

Qin, C.; O'Donoghue, B.; Bunel, R.; Stanforth, R.; Gowal, S.; Uesato, J.; Swirszcz, G.; Kohli, P.; et al. 2019. Verification of non-linear specifications for neural networks. *arXiv preprint arXiv:1902.09592*.

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.

ul Abdeen, Z.; Yin, H.; Kekatos, V.; and Jin, M. 2022. Learning neural networks under input-output specifications. In *American Control Conference (ACC2022)*, 1515–1520. IEEE.

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