A Unified View of SDP-based Neural Network Verification through Completely Positive Programming

Robin Brown
Stanford University
rabrown1@stanford.edu

Edward Schmerling
Stanford University
schmrlng@stanford.edu

Navid Azizan
Massachusetts Institute of Technology
azizan@mit.edu

Marco Pavone
Stanford University
pavone@stanford.edu

Abstract

Verifying that input-output relationships of a neural network conform to prescribed operational specifications is a key enabler towards deploying these networks in safety-critical applications. Semidefinite programming (SDP)-based approaches to Rectified Linear Unit (ReLU) network verification transcribe this problem into an optimization problem, where the accuracy of any such formulation reflects the level of fidelity in how the neural network computation is represented, as well as the relaxations of intractable constraints. While the literature contains much progress on improving the tightness of SDP formulations while maintaining tractability, comparatively little work has been devoted to the other extreme, i.e., how to most accurately capture the original verification problem before SDP relaxation. In this work, we develop an exact, convex formulation of verification as a completely positive program (CPP), and provide analysis showing that our formulation is minimal—the removal of any constraint fundamentally misrepresents the neural network computation. We leverage our formulation to provide a unifying view of existing approaches, and give insight into the source of large relaxation gaps observed in some cases.

1 INTRODUCTION

While neural networks today empower many consumer products in image and natural language understanding, they have been shown to fail in surprising and unexpected ways, potentially deterring broad deployment in safety critical settings. One avenue for inspiring confidence in neural networks is through the formal verification of safety rules specified as input-output relationships describing limits on the expected behavior of a network. In this paper, we consider algorithms that pose verification as an optimization problem, where the objective encodes a metric of satisfaction of the safety rule, the constraints represent the neural network computation, and the optimal objective value directly corresponds to confirmation or denial of the safety rule.

Sound and complete, or exact, verifiers must always return the correct decision, which inherently requires exact representation of the neural network computations. Due to the NP-completeness of verification Katz et al. (2017), exact verifiers face a complexity barrier prohibiting faster than exponential run-time in the worst case. This suggests that faithfully representing the forward pass of a neural network is at odds with tractable optimization formulations.

Sound but incomplete, or relaxed, verifiers must never return a false assertion of safety, but may conservatively suggest a network is unsafe when it is truly safe. The conservatism is a result of approximating the neural network computation, and is the trade-off for improved tractability. In the context of optimization-based verification, this is achieved by loosening exact constraints into approximate constraints; the mismatch between corresponding optimal objective values is termed the relaxation gap. In this work we aim to develop a deeper understanding of how to tune the balance between tightness and efficiency, motivated by the central challenge of devising a systematic family of relaxations with exact representation of neural network computations as a limiting case.

Contributions. Our contributions are twofold:

1. We develop an *exact, convex formulation* of the verification problem as a completely positive program (CPP); these are linear optimization problems over the cone of completely positive ma-

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

trices. While the complexity of verification is not resolved by the proposed formulation, it is packaged entirely in the complete positivity constraint, with the neural network computation being exactly represented by tractable linear constraints. This gives a clean separation between the two competing desiderata of accuracy and tractability in relaxed verification, opening the door for new classes of verifiers that predictably trade-off tightness and efficiency.

2. We also provide analysis explaining how properties of the CPP formulation evolve when the complete positivity constraint is relaxed. We find that many of the favorable properties of the CPP formulation are retained in SDP relaxations, showing that it is a convenient starting point for constructing tight SDP-based verifiers. Finally, we contextualize existing work in this shared framework, clearly laying out their similarities and differences with the proposed framework.

1.1 Related Work

Sound and Complete Verifiers. A number of exact verifiers rely on reformulating verification as a mixed integer linear program (MILP) Cheng et al. (2017); Lomuscio and Maganti (2017); Fischetti and Jo (2017); Tjeng et al. (2019) or Satifiability Modulo Theories (SMT) problem Scheibler et al. (2015) and calling an off-the-shelf solver. Others have developed custom solvers specifically meant to leverage the structure of the verification problem Katz et al. (2017); Ehlers (2017); De Palma et al. (2021); Jaeckle et al. (2021). Bunel et al. (2018) provides a unified perspective of exact verifiers through the lens of branch and bound. The primary differences across the various methods stem from (1) the way bounds are computed, (2) the type of branching that is considered, and (3) strategies to guide branching. Whether deferring to the decisions of an off-the-shelf solver, or employing verification-specific decisions, all exact verifiers ultimately rely on exhaustive search.

Sound but Incomplete Verifiers. Sound but incomplete verifiers bypass exhaustive search by solving convex relaxations of the verification problem, where the neuron values either appear as or can be derived from variables in the optimization problem. In our discussion, we distinguish between first-order and second-order relaxations. Broadly speaking, we categorize a relaxation as first order if the optimization variables represent degree one monomials of the neuron values—in this case, the neuron values can be directly read off the from the optimal solution. We consider a relaxation to be second order if the optimization variables represent degree two monomials of the neuron values—in this case, recovering neuron values typically requires factoring the optimal solution. Because relaxed verifiers rely on approximating the neural network computations, the optimal neuron values may not necessarily correspond with a valid forward pass of the network.

Under the category of first-order relaxations are methods that are based on linear outer bounds of activations functions Ehlers (2017); Singh et al. (2018), interval bound propagation Gowal et al. (2018); Wang et al. (2018a,b), and the dual of the relaxed or original non-convex problem Wang et al. (2018b); Wong et al. (2018); Dvijotham et al. (2018b,a). Salman et al. (2019) gives a unifying perspective of first-order relaxations based on a layer-wise convex relaxation framework, and shows that the optimal layer-wise convex relaxation exhibits a non-trivial gap. This gap highlights the inability of first-order relaxations to capture the ReLU activation, the complementarity of which is naturally represented with second-order (i.e., quadratic) constraints.

An approach for addressing these limitations is to lift to a second-order space where the quadratic ReLU constraints can be linearized via the reformulation linearization technique (RLT) Sherali and Adams (1998). Raghunathan et al. (2018) first proposed representing ReLUs with a quadratic constraint and lifting the problem to a SDP. Work in this area is primarily based on SDPs both from the primal perspective Raghunathan et al. (2018); Dathathri et al. (2020); Anderson et al. (2021); Ma and Sojoudi (2020), and from the dual perspective Fazlyab et al. (2022); Newton and Papachristodoulou (2021). One exception is Dvijotham et al. (2020), which uses diagonally dominant matrices to relax the SDP further and represent it as a linear program (LP).

One challenge specific to second-order relaxations is that the quantities of interest, i.e., neuron values, are obfuscated in the relaxation formulation. This makes it difficult to tell how the constraints interact, and to what extent they accurately capture the neural network computation. The analysis provided in this work aims also to clarify these points.

2 PRELIMINARIES

2.1 Notation and Terminology

For a matrix M, we use $M_{i,j}$ to denote the entry in the ith row and jth column, $M_{i,*}$ denotes the entire ith row, and $M_{*,j}$ denotes the entire jth column. For sets of matrices, $S_M + S_N$ denotes the Minkowski sum: $S_M + S_N := \{M + N \mid M \in S_M, N \in S_N\}$. The matrix inner product $\langle A, B\rangle$ is defined as $\langle A, B\rangle = \mathrm{Tr}(A^\top B)$. S^+ is the set of positive semidefinite (PSD) matrices; $\mathcal{N}$ is the set of entrywise non-negative square matrices (we use S^+_n and $\mathcal{N}_n$ to specify the dimension if it is unclear from context). The cone of doubly non-negative matrices is defined as $S^+ \cap \mathcal{N}$, where $\cap$ denotes the set intersection.

2.2 Problem Setting

Feedforward ReLU Networks. We consider an n-layer feedforward ReLU network representing a function f with input $z_{0,*} \in \mathbb{R}^{h_0}$ and output $f(z_{0,*}) = z_{n,*} \in \mathbb{R}^{h_n}$, with f being the composition of layer-wise functions, i.e., $f = f_n \circ f_{n-1} \circ \cdots \circ f_1$. The ith layer of f corresponds to a function $f_i : \mathbb{R}^{h_{i-1}} \to \mathbb{R}^{h_i}$ (h_i is the dimension of the hidden variable $z_{i,*}$) of the form

$$z_{i,*} = f_i(z_{i-1,*}) = \sigma(\overline{W}[i-1]z_{i-1,*} + b[i-1]). \quad (1)$$

where $\overline{W}[i-1] \in \mathbb{R}^{h_i \times h_{i-1}}$ is the weight matrix, $b[i-1] \in \mathbb{R}^{h_i}$ is the bias. The vector of all neurons in the i-th layer is denoted by $z_{i,*}$, while $z_{i,j}$ refers specifically to the j-th neuron in the i-th layer. The function $\sigma(\hat{z}_{i,j}) = \max(0, \hat{z}_{i,j})$ is the Rectified Linear Unit (ReLU) function; on vectors, $\sigma(\cdot)$ operates elementwise. We assume that this activation is not applied in the last layer f_n. We also model the preactivation values $\hat{z}_{i,*} = \overline{W}[i-1]z_{i-1,*} + b[i-1]$, so that $z_{i,*} = \sigma(\hat{z}_{i,*})$ for $i \geq 1$, and make the identification $\hat{z}_{0,*} = z_{0,*}$.

In this paper, we use a positive/negative splitting for each neuron, denoted by $\lambda^+ \geq 0$, $\lambda^- \geq 0$, rather than the typical pre/post-activation splitting. The positive/negative and pre/post-activation splittings are related by the following equations:

$$\lambda^+_{i,j} = z_{i,j} \quad (2)$$

$$\lambda^-_{i,j} = z_{i,j} - \hat{z}_{i,j} \quad (3)$$

We will often need to make a number of routine conversions, chiefly between matrices that act on post-activation neurons in single layer, $z_{i,*}$, to those that act on the collection of all positive/negative splittings of variables. To emphasize that these conversions maintain equivalence, we refer to these converted matrices using the same letter and use an overline (e.g., $\overline{M}$) to denote matrices that act on single layers, and an unmarked matrix (e.g., M) to denote matrices that act on the collection of all variables. For inequalities, this conversion also includes the addition of a slack variable to convert to an equality constraint. We use an underline (e.g., $\underline{M}$) to denote that a slack variable has not been included and the inequality should be treated as such. Concrete examples illustrating these conversions are included in the Appendix.

Safety Rules. We consider safety rules specified as input-output relationships on f. Specifically, for all inputs from the set $\mathcal{X} \subseteq \mathbb{R}^{h_0}$, we aim to ensure that the output belongs to the set $\mathcal{Y} \subseteq \mathbb{R}^{h_n}$. We consider bounded, polytopic input sets

$$\mathcal{X} = \{x \in \mathbb{R}^{h_0} \mid \overline{A}x \leq a\}, \quad (4)$$

and output sets specified by a half-space constraint,

$$\mathcal{Y} = \{y \in \mathbb{R}^{h_n} \mid \bar{c}^\top y \geq d\}. \quad (5)$$

This is not restrictive, as this construction can be extended to polytopic output sets as well; because polytopes are the intersection of half-spaces, each inequality defining the polytope can be verified independently. In this paper, we consider the formulation of finding output violations given input constraints. However, the methods developed in this paper can be readily adapted to finding minimal adversarial disturbances by imposing constraints on the output and optimizing over the input.

Optimization Formulation. Verification can be posed as a non-convex optimization problem:

$$\begin{aligned} \texttt{OPT} = \min_{\lambda^+,\, \lambda^-} \quad & \bar{c}^\top(\lambda^+_{n,*} - \lambda^-_{n,*}) \\ \text{s.t.} \quad & \overline{A}(\lambda^+_{0,*} - \lambda^-_{0,*}) \leq a, \\ & \lambda^+_{i+1,*} - \lambda^-_{i+1,*} = \overline{W}[i]\lambda^+_{i,*} + b[i], \\ & \lambda^+_{i,j}\lambda^-_{i,j} = 0 \quad \forall i, j, \\ & \lambda^+ \geq 0,\ \lambda^- \geq 0 \end{aligned} \quad (6)$$

where the network is safe if and only if $\texttt{OPT} \geq d$.

2.3 Completely Positive Programming

Completely positive programs (CPP) are linear optimization problems over matrix variables Duŕ (2010):

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \langle M_0,\, X\rangle \\ \text{subject to} \quad & \langle M_i,\, X\rangle = m_i, \quad i \in \{1, \ldots, L\}, \\ & X \in C^* \end{aligned} \quad (7)$$

where $C^* \subset S^+ \cap \mathcal{N}$ is the cone of completely positive (CP) matrices, that is, matrices that have a factorization with entrywise non-negative entries:

$$\mathcal{C}^*_n := \{X \in \mathbb{R}^{n\times n} \mid X = \sum_k x^{(k)}(x^{(k)})^\top, \quad x^{(k)} \in \mathbb{R}^n_{\geq 0}\} \quad (8)$$

Based on the Sum of Squares (SOS) hierarchy, Parrilo (2000) constructed a hierarchy of cones $\{(\mathcal{K}^r)^*\}$ approximating the completely positive cone from the exterior. Meaning, $\mathcal{C}^* = \bigcap_{r\geq 0}(\mathcal{K}^r)^*$, and

$$S^+ \cap \mathcal{N} = (\mathcal{K}^0)^* \supset (\mathcal{K}^1)^* \supset \ldots \quad (9)$$

Optimizing over each cone, $(\mathcal{K}^r)^*$, can be posed as an SDP. In addition, the hierarchy gives a formulaic methodology for constructing relaxations of CPPs that trade off accuracy and tractability.

3 GEOMETRY OF COMPLETE POSITIVITY

Before presenting the full CPP formulation, this section will provide an introductory exploration of the

interaction between complete positivity and the verification problem. We focus on a single neuron and provide intuition for the geometry of CP matrices when representing a ReLU activation.[1] A single neuron is represented by two variables, $\lambda^+ \geq 0$ and $\lambda^- \geq 0$, denoting the positive and negative split of the neuron, respectively. The ReLU function is then defined by the quadratic constraint $\lambda^+\lambda^- = 0$. The completely positive formulation entails lifting to a matrix,

$$\begin{pmatrix} \Lambda[\lambda^+,\lambda^+] & \Lambda[\lambda^+,\lambda^-] & \lambda^+ \\ \Lambda[\lambda^+,\lambda^-] & \Lambda[\lambda^-,\lambda^-] & \lambda^- \\ \lambda^+ & \lambda^- & 1 \end{pmatrix} \in C^* \tag{10}$$

where the elements of $\Lambda \in \mathbb{R}^{2\times 2}$ correspond to cross terms of λ^+ and λ^-, and are denoted with symbolic indexing. In the lifted formulation, the ReLU is further defined by the linear constraint $\Lambda[\lambda^+,\lambda^-] = 0$. Complete positivity means there exists a factorization

$$\begin{pmatrix} \Lambda[\lambda^+,\lambda^+] & \Lambda[\lambda^+,\lambda^-] & \lambda^+ \\ \Lambda[\lambda^+,\lambda^-] & \Lambda[\lambda^-,\lambda^-] & \lambda^- \\ \lambda^+ & \lambda^- & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{+,(k)} \\ \lambda^{-,(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{+,(k)} \\ \lambda^{-,(k)} \\ \xi^{(k)} \end{pmatrix}^\top \tag{11}$$

where $\lambda^{+,(k)}, \lambda^{-,(k)}, \xi^{(k)} \geq 0$, and K is the rank of the factorization. The variables in CPPs and SDPs are weighted (convex in the case of CPPs) combinations of the factors, e.g., neurons $\lambda^\pm = \sum_k \xi^{(k)}\lambda^{\pm,(k)}$. As we will further explore in Section 4, it is important that each factor satisfies the constraints, however we are only able to impose them on their aggregate. A relaxation gap arises when constraints that hold for the sum of factors do not hold for individual factors, e.g., $\Lambda[\lambda^+,\lambda^-] = \sum_k \lambda^{+,(k)}\lambda^{-,(k)} = 0$, but $\lambda^{+,(k)}\lambda^{-,(k)} \neq 0$.

In the context of ReLUs, non-negativity of $\lambda^{+,(k)}, \lambda^{-,(k)}, \xi^{(k)}$ is critical for ensuring that each factor corresponds to a valid ReLU, even if $K \neq 1$. In terms of the pre/post-activation splitting,

$$z^{(k)} \geq 0, \quad z^{(k)} \geq \hat{z}^{(k)}, \quad \sum_{k=1}^{K} z^{(k)}(z^{(k)} - \hat{z}) = 0. \tag{12}$$

Figure 1 depicts three different cases when $K = 2$. In the diagram, each factor satisfies the ReLU constraints, i.e., $z^{(k)} = \sigma(\hat{z}^{(k)})$. Intuitively, the exactness of the CPP formulation follows from analogous arguments that each term in the factorization of an optimal solution is feasible for (6); this notion is formalized in the proof of Theorem 4.1.

4 RESULTS

In this section, we first state our main result, the exact reformulation of (6) as a CPP, in Theorem 4.1. In order to understand SDP relaxations of this reformulation, the remainder of this section is dedicated to breaking down the constraints in the CPP and illustrating how the guarantees change when the CP constraint is relaxed. The proofs of all results stated in this section are provided in the Supplemental Material.

[1] Although, for dimensions $n \leq 4$, $C_n^* = S_n^+ \cap \mathcal{N}_n$.

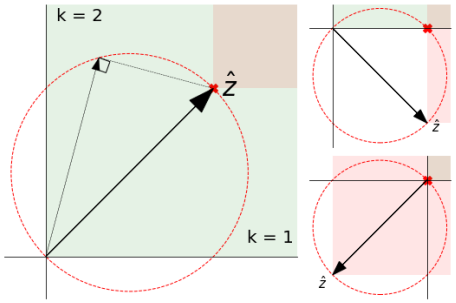


Figure 1: This figure depicts the geometry of CP matrices and ReLU constraints. $\sum_k z^{(k)}(\hat{z}^{(k)} - z^{(k)}) = 0$ constrains z to the dotted red circle, $z^{(k)} \geq \hat{z}^{(k)}$ is depicted by the shaded red region, and $z^{(k)} \geq 0$ is depicted by the shaded green region. z (indicated by the red cross) is fully determined by these constraints.

In this section, the objects of interest will be matrices of the form $\begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix}$, where Λ is a square matrix representing cross terms of elements in λ, a vector. In the proposed formulation, $\lambda^\top = \begin{pmatrix} (\lambda^+)^\top & (\lambda^-)^\top & s^\top \end{pmatrix}$ is the concatenation of λ^+, λ^-, the positive and negative splitting for each neuron, and s, slack variables used to represent inequality constraints. To emphasize the relationship with λ, we use symbolic indexing to refer to specific terms in Λ; for example, $\Lambda[\lambda^+_{i,j}, s_k]$ refers to the element in Λ corresponding to the dot product of factors corresponding to $\lambda^+_{i,j}$ and s_k.

Theorem 4.1. *Problem (6) is equivalent to the following convex optimization problem:*

$$\begin{aligned} \texttt{OPT}_{\texttt{CPP}} = \min_{\lambda,\ \Lambda} \quad & c^\top\lambda && \text{(13a)} \\ \text{s.t.} \quad & A_{j,*}\lambda = a_j, && \text{(13b)} \\ & \langle A_{j,*}^\top A_{j,*},\ \Lambda\rangle = a_j^2, && \text{(13c)} \\ & W[i]_{j,*}\lambda = b[i]_j, && \text{(13d)} \\ & \langle W[i]_{j,*}^\top W[i]_{j,*},\ \Lambda\rangle = b[i]_j^2, && \text{(13e)} \\ & \Lambda[\lambda^+_{i,j}, \lambda^-_{i,j}] = 0, && \text{(13f)} \\ & \begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in \mathcal{C}^* && \text{(13g)} \end{aligned}$$

Exactness is defined by the following conditions:

1. $\texttt{OPT} = \texttt{OPT}_{\texttt{CPP}}$*: Problem (6) and Problem (13) have the same objective value.*
2. *If* $(\lambda^*, \Lambda^*) = \arg\min$ *(13) then* λ^* *is in the convex hull of optimal solutions for Problem (6).*

The salient features of (13) are the *pairs* of linear and "self-quadratic" constraints and the CP constraint.

To motivate the following discussion, we introduce the zeroth order sum of squares (0-SOS) relaxation as a natural SDP relaxation for Problem (13):

$$\begin{aligned} \underset{\lambda,\,\Lambda}{\text{minimize}} \quad & c^\top \lambda \\ \text{subject to} \quad & (13b) - (13f), \\ & \begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in S^+ \cap \mathcal{N} \end{aligned} \tag{14}$$

This relaxation (derived from (9)) replaces the intractable CP constraint, $\begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in C^*$, with a tractable doubly non-negative constraint, $\begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in S^+ \cap \mathcal{N}$.

Next, we explore the cooperative role that the linear and self-quadratic constraints play in enforcing linear constraints on individual factors. In Theorem 4.2, we highlight the identical role they play in Problems (13) and (14) when representing equality constraints. They do, however, result in different guarantees when representing inequality constraints. These differences are explored in Corollary 4.2.1. We present results in a generic form, because none of these results rely on the specific form of the verification problem, and it allows us to focus on the core assumptions needed to derive each result. To tie things back to (13) and (14), we intersperse these results with discussion of implications for verification.

The following theorem shows that pairs of linear and self-quadratic constraints play an identical role in both the CPP and SDP-based formulations.

Theorem 4.2. *Suppose there is a factorization*

$$\begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top. \tag{15}$$

Then, $V_{i,*}\lambda^{(k)} = \xi^{(k)} v_i$ for all k if and only if

$$V_{i,*}\lambda = v_i \tag{16}$$

$$\left\langle V_{i,*}^\top V_{i,*},\, \Lambda \right\rangle = v_i^2. \tag{17}$$

This theorem underscores the importance of *pairing* linear and self-quadratic constraints to enforce linear constraints on the factors, λ^k and ξ^k. Notice that this result does not rely on complete positivity, and holds even with a weaker positive semidefinite constraint.

Network-Defining Constraints. Theorem 4.2 characterizes how constraints (13d) and (13e) act in Problems (13) and (14). They collectively ensure that the neurons in consecutive layers are related through the weights and biases of the networks.

Because the equality constraints act the same way in (13) and (14), the only potential source of a relaxation gap is due to inequality constraints. In our framework, we convert inequality constraints to equality constraints through the introduction of slack variables. For a CP matrix, each factor's slack variables are guaranteed to be non-negative, whereas for doubly non-negative matrices, only their weighted sum is guaranteed to be non-negative. That is, individual slack variables may be negative, and thus individual factors are not guaranteed to satisfy the inequalities. The following corollary highlights these differences.

Corollary 4.2.1. *Suppose there is a factorization,*

$$\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top \tag{18}$$

and for all $i = \{1, \dots L\}$, (λ, Λ) satisfy

$$V_{i,*}\lambda + s_i = v_i, \tag{19}$$

$$\left\langle V_{i,*}^\top V_{i,*},\, \Lambda[\lambda,\lambda^\top] \right\rangle + 2V_{i,*}\Lambda[\lambda,s_i] + \Lambda[s_i,s_i] = v_i^2. \tag{20}$$

Then we must have:

1. *For $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in C^*$, each factor individually satisfies the inequalities:*

$$V_{i,*}\lambda^{(k)} \le \xi^{(k)} v_i. \tag{21}$$

2. *For $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in S^+$, and $s \ge 0$ the weighted sum of factors satisfies the inequalities:*

$$V_{i,*}\lambda = \sum_k \xi^{(k)} V_{i,*}\lambda^{(k)} \le v_i. \tag{22}$$

Input and Non-negativity Constraints. This corollary shows that constraints (13b) and (13c) are sufficient for ensuring that each factor satisfies the input constraints in (13). However, we cannot derive the same result for (14) because membership in the cone $S^+ \cap \mathcal{N}$ does not necessarily imply the existence of a non-negative factorization. An analogous conclusion applies to the non-negativity of λ^+ and λ^-.

ReLU Constraints. In (13), each factor satisfies $\lambda_{i,j}^{\pm,(k)} \ge 0$. Thus the constraint $\sum_k \lambda_{i,j}^{+,(k)} \lambda_{i,j}^{-,(k)} = 0$ can only hold if $\lambda_{i,j}^{+,(k)} \lambda_{i,j}^{-,(k)} = 0$ for all k, i.e., each factor satisfies the ReLU constraints. In (14), it is only guaranteed that $\sum_k \xi^{(k)} \lambda_{i,j}^{\pm,(k)} \ge 0$, so the same conclusion cannot be drawn.

Verification-Defining Constraints. In the proposed CPP formulation, constraints (13b)-(13f) are critical for encoding the verification problem. By the results presented in this section, the removal of any of these constraints would fundamentally misrepresent the problem. For this reason, we call them the verification-defining constraints.

Strengthening Constraints. In the (0-SOS) relaxation (14), however, the verification-defining constraints are not sufficient to ensure exactness. One common strategy is to impose additional "strengthening constraints." These are constraints encoding *a priori* knowledge of the optimal solution. They will have no effect on (13), but can potentially reduce the feasible domain in relaxations, thereby generating tighter relaxations. In the context of verification, these are typically derived from bound constraints:

$$\hat{u}_{i,j} - (\lambda^+_{i,j} - \lambda^-_{i,j}) \geq 0 \quad (23) \qquad u_{i,j} - \lambda^+_{i,j} \geq 0 \quad (24)$$

$$(\lambda^+_{i,j} - \lambda^-_{i,j}) - \hat{l}_{i,j} \geq 0 \quad (25) \qquad \lambda^+_{i,j} - l_{i,j} \geq 0 \quad (26)$$

These are lower and upper bounds for each neuron's pre/post-activation values (denoted by $\hat{l}_{i,j}, \hat{u}_{i,j}$, and $l_{i,j}, u_{i,j}$ respectively)—they can be found efficiently by forward propagating bounds on the input set. These bounds can be further refined by leveraging the following characterization of the convex hull of the graph of ReLUs:

$$\lambda^+_{i,j} \leq \frac{u_{i,j} - l_{i,j}}{\hat{u}_{i,j} - \hat{l}_{i,j}}((\lambda^+_{i,j} - \lambda^-_{i,j}) - \hat{l}_{i,j}) + l_{i,j} \quad (27)$$

This inequality is commonly referred to as the triangle relaxation due to the geometry of the convex hull (Liu et al., 2021, Fig. 6.4).

In our framework, each of these inequality constraints would be imposed by introducing a slack variable and including a pair of linear and self-quadratic constraints. In principle, (23)-(27) could be further combined with additional valid inequalities to provide even stronger relaxations. However, the following Corollary shows that simply combining existing linear inequalities, e.g., by multiplication, will not actually strengthen the relaxation.

Corollary 4.2.2. *If* $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in S^+ \cap \mathcal{N}$, *and* (18), (19), (20) *hold, then the following holds:*

$$v_i v_j - v_i V_{j,*}\lambda - v_j V_{i,*}\lambda + \langle V^\top_{i,*} V_{j,*}, \Lambda[\lambda,\lambda^\top]\rangle \geq 0 \quad (28)$$

Inequality (28) is the linearized inequality representing $(v_i - V_{i,*}\lambda)(v_j - V_{j,*}\lambda) \geq 0$. In contrast to the self-quadratic constraints, this is a product of two different linear constraints. For this reason, we call constraints of this form "cross-quadratic" constraints. Corollary 4.2.2 states that if we enforce constraints of the form $V_{i,*}\lambda \leq v_i$ through the introduction of a slack variable and paired linear/self-quadratic constraints, then the derived cross-quadratic constraint will also hold. This is significant because in a number of existing works, cross-quadratic constraints derived from (23)-(26) are introduced, however the linear and self-quadratic constraints are excluded.

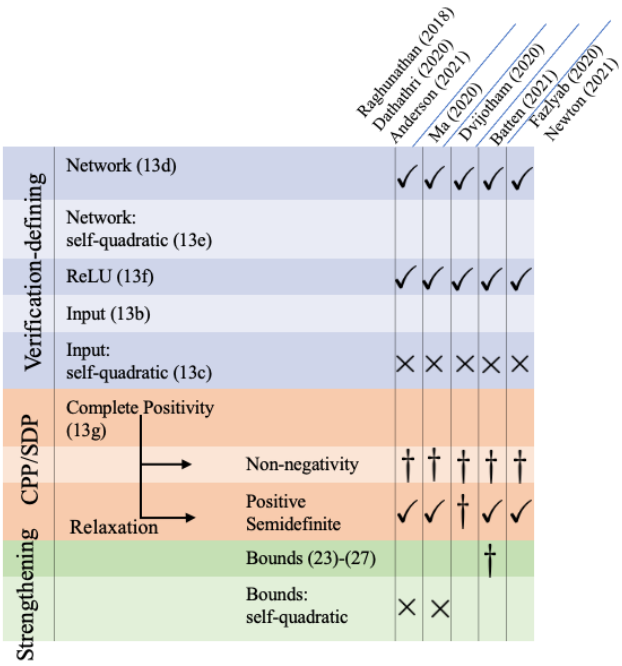


Figure 2: This table summarizes the constraints included in existing work. The † indicates partial encoding of constraints/categorizations that require additional qualification. Cross-quadratic constraints are marked with a × in the self-quadratic row.

5 UNIFYING SDP-BASED VERIFICATION

Now that we have established the notions of verification-defining and strengthening constraints, we are in a position to show how existing work fits into our framework. We discuss two seemingly disparate frameworks that have served as the basis for subsequent work, Raghunathan et al. (2018) and Fazlyab et al. (2022). In this section we show how these frameworks, as well as subsequent works, are related by couching their approaches in the language of our proposed verification-defining and strengthening constraints. A number of subsequent works focus on improving computational aspects of existing relaxations (e.g., time to solution, memory usage) without aiming to improve the relaxation gap. These works are discussed in the Appendix rather than the main body of this paper because they largely have the same constraint structure at work.

A number of works in this section construct strengthening cuts derived from pre/post-activation bounds on neurons. Without loss of generality, we will assume that $\hat{l}_{i,j} < 0 < \hat{u}_{i,j}$ (otherwise the ReLU could be treated as the identity, if $\hat{l}_{i,j} \geq 0$, or zero, if $\hat{u}_{i,j} \leq 0$). The post-activation bounds are then $l_{i,j} = 0$ and $u_{i,j} = \hat{u}_{i,j}$.

5.1 Direct SDP

To the best of our knowledge, Raghunathan et al. (2018) was the first to propose an SDP relaxation of the verification problem:

$$\begin{aligned}
\min_{z,\,Z} \quad & \bar{c}^\top z_{n,*} \\
\text{s.t.} \quad & z_{i,*} \geq 0,\ z_{i,*} \geq \overline{W}[i-1] z_{i-1,*}, \\
& Z[z_{i,j}, z_{i,j}] = \overline{W}[i-1]_{j,*} Z[z_{i-1,*}, z_{i,j}], \\
& -Z[z_{i,j}, z_{i,j}] + (u_{i,j} + l_{i,j}) z_{i,j} - l_{i,j} u_{i,j} \geq 0,\ \forall i, j, \\
& \begin{pmatrix} Z & z \\ z^\top & 1 \end{pmatrix} \in \mathcal{S}^+
\end{aligned} \tag{29}$$

Although their approach does not include biases, it is straightforward to include them. Problem (29) is equivalent to the following optimization problem:

$$\begin{aligned}
\min_{\lambda,\,\Lambda} \quad & c^\top \lambda \\
\text{s.t.} \quad & (l_{0,i} + u_{0,i}) U_{i,*} \lambda - \left\langle U_{i,*}^\top U_{i,*},\, \Lambda \right\rangle - u_i l_i \geq 0, \\
& \hat{u}_{i,j} \lambda^+_{i,j} - \Lambda[\lambda^+_{i,j}, \lambda^+_{i,j}] \geq 0,\ i \geq 1, \\
& W[i]_{j,*} \lambda = b[i]_j, \\
& \Lambda[\lambda^+_{i,j}, \lambda^-_{i,j}] = 0, \\
& \begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in S^+, \\
& \lambda \geq 0
\end{aligned} \tag{30}$$

where U is defined as $U_{i,*}\lambda = \lambda^+_{0,i} - \lambda^-_{0,i}$. In the context of (14), we see that (30) does not account for non-negativity of Λ, i.e., $\Lambda \in \mathcal{N}$, and the network self-quadratic constraints (13e). (30) also includes strengthening constraints derived from $\lambda^+_{i,j} \geq 0$ and $(\hat{u}_{i,j} - \lambda^+_{i,j}) \geq 0$.

Tightening Extensions. Based on the observation that (30) sometimes produces bounds looser than the triangle LP relaxation , Batten et al. (2021) proposes including linear cuts derived from (27). In the following, we provide intuition for the gap that the triangle inequality closes.

The PSD constraint ensures that each neuron has a factorization in the form of (11) where $\lambda^{+,(k)}$, $\lambda^{-,(k)}$ are not necessarily non-negative (without loss of generality $\xi^{(k)}$ can be normalized to be non-negative). In contrast to the CP case, the constraint $\lambda^+ \geq 0$ only guarantees that $\sum_{i=1}^K \xi^{(k)} \lambda^{+,(k)} \geq 0$, and similarly for other inequalities. In terms of the pre/post-activation

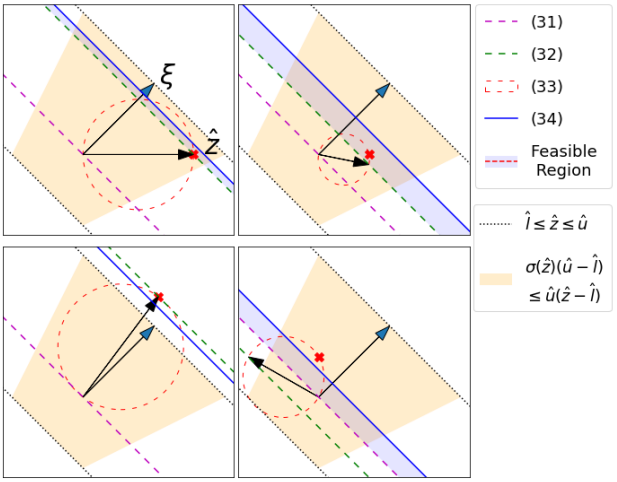


Figure 3: This figure depicts the geometry of the triangle cut. $\sum_k z^{(k)}(\hat{z}^{(k)} - z^{(k)}) = 0$ constrains z to the dotted red circle. $z \geq \hat{z}$ and $z \geq 0$ constrain z to the right of the dotted green and purple lines, respectively. The triangle cut (34) constrains z to the left of the solid blue line. The lower and upper bounds ($\hat{l} \leq \hat{z}$ and $\hat{z} \leq \hat{u}$) used to construct the triangle cut are demarcated by the dotted black lines. The red cross marks the factor-wise ReLU (as in Figure 1), and the shaded orange region indicates the region (in terms of $\hat{z}$) for which the factor-wise ReLU is feasible.

splittings, it is guaranteed that:

$$\sum_{i=1}^K \xi^{(k)} z^{(k)} \geq 0, \tag{31}$$

$$\sum_{i=1}^K \xi^{(k)} z^{(k)} \geq \sum_{i=1}^K \xi^{(k)} \hat{z}^{(k)}, \tag{32}$$

$$\sum_{i=1}^K z^{(k)} (\hat{z}^{(k)} - z^{(k)}) = 0. \tag{33}$$

These are the analogs of (12). The triangle cut provides the additional guarantee that

$$\sum_{i=1}^K \xi^{(k)} z^{(k)} \leq \frac{\hat{u}_{i,j}}{\hat{u}_{i,j} - \hat{l}_{i,j}} \left(\sum_{i=1}^K \xi^{(k)} \hat{z}^{(k)} - \hat{l}_{i,j} \right). \tag{34}$$

Figure (27) illustrates the geometry of these constraints when $K = 2$. The plots, from left to right and top to bottom, correspond to the cases where (34) is (i) non-redundant and includes $\sigma(\hat{z})$, (ii) redundant, (iii) infeasible, and (iv) non-redundant but eliminates $\sigma(\hat{z})$ given (31) - (33). The diagrams in the top row illustrate the effect of the triangle cut in restricting the feasible region of z given $\hat{z}$. In the first case, the triangle cut strictly reduces the feasible region, resulting in a strengthened relaxation. In the second case, the region defined by the intersection of (31) - (33) lies entirely within the region defined by (34), i.e., the triangle cut does not add any additional information.

The diagrams in the bottom row illustrate the triangle cut's role in enforcing "acceptable" values of $\hat{z}$. The third (infeasible) case occurs if and only if the pre-activation bounds are actually not satisfied $\sum_{i=1}^{K} \xi^{(k)}\hat{z}^{(k)} \notin [\hat{l}, \hat{u}]$, i.e., the triangle cut implicitly constrains $\hat{z} \in [\hat{l}, \hat{u}]$. In the fourth case, the triangle cut is feasible but counter-intuitively renders the factor-wise ReLU, $z^{(k)} = \sigma(\hat{z}^{(k)})$, infeasible. This is because the triangle cut is only valid for $\hat{z}$ that are convex combinations of

$$\hat{z}^{(k)} \in \left[\frac{\hat{l}}{\xi^{(k)}}, \frac{\hat{u}}{\xi^{(k)}}\right], \tag{35}$$

even if $\sum_{i=1}^{K} \xi^{(k)}\hat{z}^{(k)} \in [\hat{l}, \hat{u}]$. Condition (35) is satisfied by any $z^{(k)}$ corresponding to a forward pass of the network, but cannot be explicitly enforced in an SDP. The behavior of the triangle cut illustrates the contrast between the guarantees of the CPP formulation, which can be leveraged to (implicitly) impose constraints on individual factors, and SDP relaxations, which can only enforce constraints on the aggregate of factors.

5.2 Quadratic Constraints and the S-procedure

Fazlyab et al. (2022) proposes a novel framework based on the notion of quadratic constraints.[2] In contrast with existing work, the proposed framework deals with classes of constraints obtained by taking infinite combinations of "atomic" constraints. For example, the atomic constraints $x \geq 0$, $y \geq 0$ are subsumed by the infinite class of constraints $\{\alpha x+\beta y \geq 0\}$ parametrized by $\alpha \geq 0$, $\beta \geq 0$. Critically, $\alpha x + \beta y \geq 0$ for all $\alpha \geq 0$, $\beta \geq 0$ if and only if $x \geq 0$ and $y \geq 0$, so these constructions are equivalent. Similarly, for each of the infinite classes of quadratic constraints proposed, we will identify the equivalent set of atomic constraints, and rewrite the formulation of Fazlyab et al. (2022) in a form that parallels the (0-SOS) relaxation (14).

Polytopic Input Sets. Fazlyab et al. (2022) proposes encoding polytopic input sets, $\mathcal{X} = \{x \in \mathbb{R}^{k_0} \mid \overline{A}x \leq a\}$, via a quadratic constraint

$$\sum_{i,j} \Gamma_{i,j}(\underline{A}_{i,*}\lambda - a_i)(\underline{A}_{j,*}\lambda - a_j) \geq 0 \tag{36}$$

with parameters, $\Gamma_{i,j} \geq 0$, $\Gamma_{i,i} = 0$. The equivalent atomic constraints are

$$(\underline{A}_{i,*}\lambda - a_i)(\underline{A}_{j,*}\lambda - a_j) \geq 0, \quad \forall i \neq j. \tag{37}$$

ReLU Constraints. The ReLU constraints are defined through a quadratic constraint

$$\begin{aligned} 0 \geq &\sum_{(i,j)} \left[\rho_{(i,j)}\lambda^+_{i,j}\lambda^-_{i,j} - \nu_{i,j}\lambda^+_{i,j} - \eta_{i,j}\lambda^-_{i,j}\right] \\ &+ \sum_{(i,j)\neq(k,l)} \gamma_{(i,j),(k,l)}(\lambda^+_{i,j} - \lambda^+_{k,l})(\lambda^-_{i,j} - \lambda^-_{k,l}) \end{aligned} \tag{38}$$

for parameters $\rho_{(i,j)} \in \mathbb{R}$, $\gamma_{(i,j),(k,l)}$, $\nu_{i,j}$, $\eta_{i,j} \geq 0$. The equivalent atomic constraints are:

$$\lambda^+_{i,j} \geq 0, \qquad \lambda^-_{i,j} \geq 0, \quad \lambda^+_{i,j}\lambda^-_{i,j} = 0 \quad \forall (i,j) \tag{39}$$

$$\lambda^+_{i,j}\lambda^-_{k,l} \geq 0, \qquad \lambda^-_{i,j}\lambda^+_{k,l} \geq 0 \quad \forall (i,j) \neq (k,l) \tag{40}$$

Notice that (39) is equivalent to $\lambda \geq 0$ and (13f), and (40) is equivalent to the non-negativity constraint $\Lambda[\lambda^+, \lambda^-] \geq 0$. In summary, the formulation proposed by Fazlyab et al. (2022) can be shown to be equivalent to

$$\begin{aligned} \min_{\lambda,\,\Lambda} \quad & c^\top \lambda \\ \text{s.t.} \quad & \left\langle \underline{A}^\top_{i,*}\underline{A}_{j,*},\, \Lambda \right\rangle - a_i\underline{A}_{j,*}\lambda - a_j\underline{A}_{i,*}\lambda \geq a_i a_j,\ i \neq j, \\ & W[i]_{j,*}\lambda = b[i]_j, \\ & \Lambda[\lambda^+_{i,j}, \lambda^-_{i,j}] = 0, \\ & \begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in S^+, \\ & \lambda \geq 0, \quad \Lambda[\lambda^+, (\lambda^-)^\top] \geq 0 \end{aligned} \tag{41}$$

In the context of (14), we see that (41) does not account for (13b), (13c), (13e), and only partially accounts for $\Lambda \in \mathcal{N}$. We also observe that (41) has included cross-quadratic input constraints that are redundant in (14).

Using the 0-SOS relaxation as a shared framework, we can thus unify many of the existing works on SDP-based verification. Our analysis clearly lays out the constraints that are included in or excluded from existing work, with Figure 2 summarizing these points.

6 EXPERIMENTS

In Sections 4 and 5, we have claimed that the "verification-defining" constraints are critical for accurately representing the neural network computation, and shown that a number of existing SDP-based relaxations omit various such constraints. In this section, we empirically evaluate some of the questions we have left unanswered. Chiefly, to what degree do the verification-defining constraints matter, i.e., is it possible to get a reasonable verification gap even without

[2] The techniques of Fazlyab et al. (2022) can be applied to activations other that ReLUs by abstracting non-ReLU activations with linear bounds characterizing various properties of activation functions (e.g., monotonicity, bounded slope, bounded values). In this section, we will focus on results pertaining to ReLU networks with polytopic input constraints, and halfspace output constraints, in order to draw a direct comparison to the other methods discussed in this paper. However, the analysis of linear and self-quadratic constraints applies to SDP encodings of the properties arising from general activation functions, even in the absence of exactness.

all of the verification-defining constraints? And, do proposed strengthening constraints sufficiently make up for the missing verification-defining constraints?

In this section, we consider a ReLU network with two inputs, one output, and one hidden layer with 10 neurons. We generated 100 network instances by selecting the weights and biases uniformly between -1.0 and 1.0. We consider a safety rule where the input set is defined as $\mathcal{X} := \{z_{0,*} \mid z_{0,*} \in [-1.0, 0.1]^2\}$ and the output set is defined as $\mathcal{Y} := \{z_{n,*} \mid z_{n,*} \geq 0\}$.

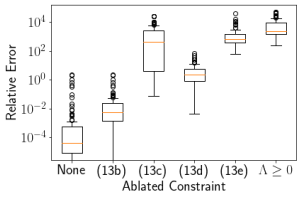
We computed the ground-truth minimum by formulating the verification problem as a MILP Tjeng et al. (2019), and solving to global optimality using Gurobi Optimization, LLC (2020), with a tolerance of 1×10^{-5}. We solve all SDPs using Splitting Conic Solver (SCS) O'Donoghue et al. (2016), also with a tolerance of 1×10^{-5}.

Ablating Verification-Defining Constraints. In Section 4, we have shown that removal of any of the verification-defining constraints fundamentally changes the underlying problem being relaxed, however we have left unanswered the degree to which removing individual constraints increases the relaxation gap. We evaluate this empirically by considering variants of (14) where constraints are removed individually from the problem (termed the ablated SDPs). We focus on ablating (13b)-(13e), and $\Lambda \geq 0$, because constraints (13f) and $\lambda \geq 0$, are not typically omitted in the literature. The relative errors of each of the ablated SDPs are plotted in Figure 4a.

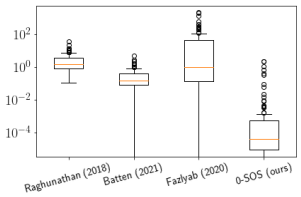
We find that removing the input linear constraints, (13b), has the smallest effect on the relaxation gap, even sometimes resulting in exact relaxations. On the other hand, removing the non-negativity constraint, $\Lambda \geq 0$, typically leads to bounds that are entirely uninformative, with relative errors ranging from 243 to 5.1×10^4. We also find that without the network self-quadratic constraints, (13e), the relative errors range from 63 to 4×10^4. The results of this ablation study are significant, because in the context of Figure (2), we see that there are no existing approaches that entirely encode (13b), (13e), and $\Lambda \geq 0$.

Comparison to the Literature. While we have shown that removing the verification-defining constraints typically introduces a significant relaxation gap, all existing work surveyed above also includes various strengthening constraints that imbue additional knowledge of the problem structure. In this experiment, we evaluate whether the strengthening constraints are sufficient to make up for the missing verification-defining constraints. Figure 4b plots the relative errors of the 0-SOS relaxation (14), and that of several relaxations proposed in the literature.

We find that (14) typically results in the tightest relaxations, and is often exact. Batten et al. (2021) and Fazlyab et al. (2022) are sometimes exact, while Raghunathan et al. (2018) never is. All exhibit median relaxation gaps that are orders of magnitudes larger than that of (14). Notably, all exhibit considerably smaller relaxation gaps than the ablated SDPs. This indicates that the gaps observed in the baselines are largely dependent on the strengthening constraints.



(a) This figure plots the relative error when constraints in (14) are ablated. While removing (13b) typically results in reasonable relaxation gaps, the removal of any other constraint renders the relaxation entirely uninformative with gaps typically on the order of 10^2.



(b) This figure shows the relative relaxation gap of the 0-SOS relaxation, (14), as well as those of several existing relaxations. The 0-SOS relaxation is often exact, with a median error that is orders of magnitude smaller than the other relaxations.

7 CONCLUSION

In this paper, we have shown that verification of ReLU networks can be formulated exactly as a complete positive program, and provided analysis showing that the formulation is minimal. Leveraging the similarity between the form of the CPP formulation and SDP-based relaxations, we use our formulation as the basis for a unifying perspective on existing approaches. We also provide empirical evaluation demonstrating (1) that the verification defining constraints are indispensable for accurately representing the verification problem, and (2) the 0-SOS relaxation of the proposed formulation exhibits relaxation gaps that are orders of magnitude smaller than existing methods, and is often exact.

While we have focused on the 0-SOS relaxation as a comparison to existing approaches, it is just one in a hierarchy of relaxations now at our disposal. Because higher order order SOS relaxations scale poorly, however, their application to solving verification requires further investigation. Fortunately, there is an abundance of literature devoted to systematically improving the tractability of SOS-based methods. In particular, we highlight the r-DSOS and r-SDSOS hierarchies proposed by Ahmadi and Majumdar (2019) as promising algorithmic paradigms for addressing the CPP formulation we have proposed. We believe the most valuable contribution of this work is the potential for a new *class* of verification methods that can systematically trade-off tightness and efficiency, with exactness as the limiting case.

Acknowledgements

This work was supported by NSF CCF (grant #1918549) and the NASA University Leadership Initiative (grant #80NSSC20M0163); this article solely reflects the opinions and conclusions of its authors and not any NSF or NASA entity.

References

Ahmadi, A. and Majumdar, A. (2019). DSOS and SDSOS optimization: More tractable alternatives to sum of squares and semidefinite optimization. *SIAM Journal on Optimization*, 3(2):193–230.

Anderson, B., Ma, Z., Li, S., and Sojoudi, S. (2021). Partition-based convex relaxations for certifying the robustness of relu neural networks. *arXiv preprint arXiv:2004.00570*.

Batten, B., Kouvaros, P., Lomuscio, A., and Zheng, Y. (2021). Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *Int. Joint Conf. on Artificial Intelligence*.

Bunel, R., Turkaslan, I., Torr, P., Kohli, P., and Kumar, M. (2018). A unified view of piecewise linear neural network verification. In *Conf. on Neural Information Processing Systems*.

Burer, S. (2009). On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming*, 120(2):479–495.

Cheng, C., Nührenberg, G., and Ruess, H. (2017). Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*.

Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R., Shankar, S., Steinhardt, J., Goodfellow, I., Liang, P., and Kohli, P. (2020). Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Conf. on Neural Information Processing Systems*.

De Palma, A., Bunel, R., Desmaison, A., Dvijotham, K., Kohli, P. Torr, P., and P., K. (2021). Improved branch and bound for neural network verification via lagrangian decomposition. *arXiv preprint arXiv:2002.10410*.

Duŕ, M. (2010). Copositive programming–a survey. In Diehl, M., Glineur, F., Jarlebring, E., and Michiels, W., editors, *Recent Advances in Optimization and its Applications in Engineering*. Springer.

Dvijotham, K., Gowal, S., Stanforth, R., Arandjelovic, R., O'Donoghue, B., Uesato, J., and Kohli, P. (2018a). Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P. (2018b). A dual approach to scalable verification of deep networks. In *Proc. Conf. on Uncertainty in Artificial Intelligence*.

Dvijotham, K., Stanforth, R., Gowal, S., Qin, C., De, S., and Kohli, P. (2020). Efficient neural network verification with exactness characterization. In Adams, R. P. and Gogate, V., editors, *Proc. Conf. on Uncertainty in Artificial Intelligence*, volume 115 of *Proceedings of Machine Learning Research*, pages 497–507. PMLR.

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

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

Fischetti, M. and Jo, J. (2017). Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *arXiv preprint arXiv:1712.06174*.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. (2018). On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*.

Gurobi Optimization, LLC (2020). *Gurobi Optimizer Reference Manual*. Available at http://www.gurobi.com.

Jaeckle, F., Lu, J., and Kumar, M. (2021). Neural network branch-and-bound for neural network verification. *arXiv preprint arXiv:2107.12855*.

Katz, G., Barrett, C., Dill, D., Julian, K., and Kochenderfer, M. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. Int. Conf. Computer Aided Verification*.

Lin, C.-J. (2007). Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779.

Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., and Kochenderfer, M. (2021). Algorithms for verifying deep neural networks. *Foundations and Trends in Optimization*, 4(3-4):244–404.

Lomuscio, A. and Maganti, L. (2017). An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*.

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

Newton, M. and Papachristodoulou, A. (2021). Exploiting sparsity for neural network verification. In Jadbabaie, A., Lygeros, J., Pappas, G. J., A. Parrilo, P., Recht, B., Tomlin, C. J., and Zeilinger, M. N., editors, *Learning for Dynamics & Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 715–727. PMLR.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2016). Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory & Applications*, 169(3):1042–1068.

Parrilo, P. A. (2000). *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, Massachusetts Inst. of Technology.

Raghunathan, A., Steinhardt, J., and Liang, P. (2018). Semidefinite relaxations for certifying robustness to adversarial examples. In *Conf. on Neural Information Processing Systems*, pages 10900–10910.

Salman, H., Yang, G., Zhang, H.and Hsieh, C., and Zhang, P. (2019). A convex relaxation barrier to tight robustness verification of neural networks. In *Conf. on Neural Information Processing Systems*.

Scheibler, K., Winterer, L., Wimmer, R., and Becker, B. (2015). Towards verification of artificial neural networks. In *MBMV*.

Sherali, H. and Adams, W. P. (1998). Reformulation-linearization techniques for discrete optimization problems. In *Handbook of Combinatorial Optimization*. Springer.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. (2018). Fast and effective robustness certification. In *Conf. on Neural Information Processing Systems*.

Tjeng, V., Xiao, K., and Tedrake, R. (2019). Evaluating robustness of neural networks with mixed integer programming. In *Int. Conf. on Learning Representations*.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018a). Efficient formal safety analysis of neural networks. In *Conf. on Neural Information Processing Systems*.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018b). Formal security analysis of neural networks using symbolic intervals. In *Proc. USENIX Conf. on Security Symposium*.

Wong, E., Schmidt, F., Metzen, J., and Kolter, Z. (2018). Scaling provable adversarial defenses. In *Conf. on Neural Information Processing Systems*.

Zhang, R. (2020). On the tightness of semidefinite relaxations for certifying robustness to adversarial examples. In *Conf. on Neural Information Processing Systems*.

A APPENDIX

A.1 Illustration of expanded out matrices

In this paper, we have made a number of routine conversions between matrices that act on pre/post-activation variables corresponding to a single layer, to those that act on the collection of all positive/negative splittings of variables from all layers. In this section, we provide a concrete example of how these conversions are carried out. Specifically, we illustrate the relationship between the matrices that act on single layers, $\overline{M}$, those that act on all variables, including slack variables M, and those that act on all variables except slack variables $\underline{M}$.

Consider an input set defined as $\mathcal{X} = \{x \in \mathbb{R}^{h_0} \mid \overline{A}x \leq a\}$. The matrix $\overline{A} \in \mathbb{R}^{m \times h_0}$ only acts on the input vector, $z_{0,*}$. In this paper, we use the positive/negative splitting, so the input constraint is given by

$$\overline{A}(\lambda_{0,*}^{+} - \lambda_{0,*}^{-}) \leq a. \tag{42}$$

In addition, we concatenate all of the neurons and slack variables in a single vector

$$\lambda = \begin{pmatrix} \lambda_{0,*}^{+} \\ \lambda_{1,*}^{+} \\ \vdots \\ \lambda_{n,*}^{+} \\ \lambda_{0,*}^{-} \\ \lambda_{1,*}^{-} \\ \vdots \\ \lambda_{n,*}^{-} \\ s \end{pmatrix} \tag{43}$$

where $s \in \mathbb{R}^m$ is a vector of slack variables corresponding to the input inequality constraints. Let $N := \sum_{i=0}^{n} h_i$ be the number of neurons, $\lambda \in \mathbb{R}^{2N+m}$ (the factor of two is because there are two variables for every neuron). Then, the constraint $\overline{A}(\lambda_{0,*}^{+} - \lambda_{0,*}^{-}) \leq a$ may be written as

$$\begin{pmatrix} \overline{A} & 0 & \dots & 0 & -\overline{A} & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda_{0,*}^{+} \\ \lambda_{1,*}^{+} \\ \vdots \\ \lambda_{n,*}^{+} \\ \lambda_{0,*}^{-} \\ \lambda_{1,*}^{-} \\ \vdots \\ \lambda_{n,*}^{-} \\ s \end{pmatrix} \leq a. \tag{44}$$

We define $\underline{A} \in \mathbb{R}^{m \times (2N+m)}$, which acts on all variables but does not incorporate the effect of slack variables, as

$$\underline{A} := \begin{pmatrix} \overline{A} & 0 & \dots 0 & -\overline{A} & 0 & \dots 0 & 0 \end{pmatrix}. \tag{45}$$

Similarly, for appropriate $s \geq 0$, then $\overline{A}(\lambda_{0,*}^{+} - \lambda_{0,*}^{-}) \leq a$ may be written as

$$\begin{pmatrix} \overline{A} & 0 & \dots & 0 & -\overline{A} & 0 & \dots & 0 & I \end{pmatrix} \begin{pmatrix} \lambda_{0,*}^{+} \\ \lambda_{1,*}^{+} \\ \vdots \\ \lambda_{n,*}^{+} \\ \lambda_{0,*}^{-} \\ \lambda_{1,*}^{-} \\ \vdots \\ \lambda_{n,*}^{-} \\ s \end{pmatrix} = a. \tag{46}$$

We define $A \in \mathbb{R}^{m \times (2N+m)}$, which acts on all variables including slacks, as

$$A := \begin{pmatrix} \overline{A} & 0 & \dots 0 & -\overline{A} & 0 & \dots 0 & I \end{pmatrix}. \tag{47}$$

A.2 Proofs of Section 4

Theorem (4.1). *Problem (6) is equivalent to the following convex optimization problem:*

$$\begin{aligned} \texttt{OPT}_{\texttt{CPP=}} \quad \underset{\lambda,\, \Lambda}{\text{minimize}} \quad & c^\top \lambda \\ \text{subject to} \quad & A_{j,*}\lambda = a_j \quad , && \text{(13b)} \\ & \langle A_{j,*}^\top A_{j,*},\, \Lambda \rangle = a_j^2 \quad , && \text{(13c)} \\ & W[i]_{j,*}\lambda = b[i]_j \quad , && \text{(13d)} \\ & \langle W[i]_{j,*}^\top W[i]_{j,*},\, \Lambda \rangle = b[i]_j^2 \quad , && \text{(13e)} \\ & \Lambda[\lambda_{i,j}^+, \lambda_{i,j}^-] = 0 \quad , && \text{(13f)} \\ & \begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in \mathcal{C}^* && \text{(13g)} \end{aligned}$$

Exactness is defined by the following conditions:

1. $\texttt{OPT} = \texttt{OPT}_{\texttt{CPP}}$*: Problem (6) and Problem (13) have the same objective value.*
2. *If* $(\lambda^*, \Lambda^*) = \arg\min (13)$ *then* λ^* *is in the convex hull of optimal solutions for Problem (6).*

Proof. This follows as a direct application of Theorem 3.2 from Burer (2009). The two assumptions required are:

- The binary variables satisfy the *key assumption*, chiefly, boundedness. This assumption is automatically satisfied as formulation (6) does not have any binary variables.
- Each variable in a complementarity constraint is bounded. This follows from our assumption of a bounded input set. Explicit bounds on each neuron can be derived by forward propagating the input bounds.

□

Theorem (4.2). *Suppose there is a factorization*

$$\begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top. \tag{48}$$

Then, $V_{i,*}\lambda^{(k)} = \xi^{(k)} v_i$ *for all* k *if and only if*

$$V_{i,*}\lambda = v_i \tag{16}$$
$$\langle V_{i,*}^\top V_{i,*},\, \Lambda \rangle = v_i^2. \tag{17}$$

Proof. ($\Leftarrow$) We will first show that if (λ, Λ) satisfy (16) and (17), then $V_{i,*}\lambda^{(k)} = \xi^{(k)} v_i$. Expanding out the outer product,

$$\begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)}(\lambda^{(k)})^\top & \xi^{(k)}\lambda^{(k)} \\ \xi^{(k)}(\lambda^{(k)})^\top & (\xi^{(k)})^2 \end{pmatrix}. \tag{49}$$

Matching up terms, $\{\lambda^{(k)}, \xi^{(k)}\}$ must satisfy

$$\sum_k (\xi^{(k)})^2 = 1 \tag{50}$$
$$\sum_k \xi^{(k)}\lambda^{(k)} = \lambda \tag{51}$$
$$\sum_k \lambda^{(k)}(\lambda^{(k)})^\top = \Lambda \tag{52}$$

Substituting these summations into the constraints, we get that

$$V_{i,*}\lambda = \sum_k \xi^{(k)} V_{i,*}\lambda^{(k)} = v_i \tag{53}$$

$$\langle V_{i,*}^\top V_{i,*},\, \Lambda\rangle = \sum_k (V_{i,*}\lambda^{(k)})^2 = v_i^2 \tag{54}$$

By squaring (53) and using the fact that $\sum_k (\xi^{(k)})^2 = 1$, we get

$$\begin{aligned} v_i^2 &= (V_{i,*}\lambda)^2 && (55)\\ &= (\sum_k \xi^{(k)} V_{i,*}\lambda^{(k)})^2 && (56)\\ &\le (\sum_k (\xi^{(k)})^2) \sum_k (V_{i,*}\lambda^{(k)})^2 && (57)\\ &= \sum_k (V_{i,*}\lambda^{(k)})^2 && (58)\\ &= v_i^2 && (59) \end{aligned}$$

where (57) follows from Cauchy-Schwarz. More specifically, this shows that in (57) Cauchy-Schwarz holds with equality, which means that $[\xi^{(k)}]_k$ and $[V_{i,*}\lambda^{(k)}]_k$ are collinear. This means that there is a scalar $\alpha \in \mathbb{R}$ such that $\alpha\xi^{(k)} = V_{i,*}\lambda^{(k)}$ for all k. Substituting this into (53), we get

$$v_i = \sum_k \xi^{(k)} V_{i,*}\lambda^{(k)} = \sum_k (\xi^{(k)})^2 \alpha = \alpha. \tag{60}$$

This shows $V_{i,*}\lambda^{(k)} = \xi^{(k)} v_i$, thus concluding the proof of the reverse implication.
($\Rightarrow$) We now just have to show that if $V_{i,*}\lambda^{(k)} = \xi^{(k)} v_i$ for all k then $V_{i,*}\lambda = v_i$ and $\langle V_{i,*}^\top V_{i,*},\, \Lambda\rangle = v_i^2$. The former follows by substituting in the relation $\sum_k \xi^{(k)}\lambda^{(k)} = \lambda$

$$\begin{aligned} V_{i,*}\lambda &= \sum_k \xi^{(k)} V_{i,*}\lambda^{(k)} && (61)\\ &= \sum_k (\xi^{(k)})^2 v_i && (62)\\ &= v_i && (63) \end{aligned}$$

where the last line follows because $\sum_k (\xi^{(k)})^2 = 1$. The latter follows from substituting in the relationship $\langle V_{i,*}^\top V_{i,*},\, \Lambda\rangle = \sum_k (V_{i,*}\lambda^{(k)})^2$:

$$\begin{aligned} \langle V_{i,*}^\top V_{i,*},\, \Lambda\rangle &= \sum_k \langle V_{i,*}^\top V_{i,*},\, \lambda^k (\lambda^k)^\top\rangle && (64)\\ &= \sum_k (V_{i,*}\lambda^k)^2 && (65)\\ &= \sum_k (\xi^k)^2 v_i^2 && (66)\\ &= v_i^2. && (67) \end{aligned}$$

□

Corollary (4.2.1)**.** *Suppose there is a factorization,*

$$\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} = \sum_{k=1}^K \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top \tag{18}$$

and for all $i = \{1, \dots L\}$, $(\lambda,\, \Lambda)$ *satisfy*

$$V_{i,*}\lambda + s_i = v_i, \tag{19}$$
$$\left\langle V_{i,*}^\top V_{i,*},\, \Lambda[\lambda, \lambda^\top] \right\rangle + 2V_{i,*}\Lambda[\lambda, s_i] + \Lambda[s_i, s_i] = v_i^2. \tag{20}$$

Then we must have:

1. *For* $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in C^*$, *each factor individually satisfies the inequalities:*

$$V_{i,*}\lambda^{(k)} \le \xi^{(k)} v_i. \tag{68}$$

2. *For* $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in S^+$, *and* $s \ge 0$ *the weighted sum of factors satisfies the inequalities:*

$$V_{i,*}\lambda = \sum_k \xi^{(k)} V_{i,*}\lambda^{(k)} \le v_i. \tag{69}$$

Proof. Leveraging Theorem 4.2, we know that if

$$\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top \tag{18}$$

and (λ, Λ) satisfy (19) and (20), then $V_{i,*}\lambda^{(k)} + s_i^{(k)} = \xi^{(k)} v_i$ for all k.

If $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in C^*$ then in particular $s^{(k)} \ge 0$ so $V_{i,*}\lambda^{(k)} \le \xi^{(k)} v_i$.

In the case where $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in S^+$, and $s \ge 0$, the result follows directly from the constraints $V_{i,*}\lambda + s_i = v_i$ and $s_i \ge 0$. □

Corollary (4.2.2)**.** *If* $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in S^+ \cap \mathcal{N}$, *and the following hold:*

1. *There is a factorization*

$$\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} \lambda^{(k)} \\ s^{(k)} \\ \xi^{(k)} \end{pmatrix}^\top \tag{18}$$

2.
$$V_{i,*}\lambda + s_i = v_i \tag{19}$$

3.
$$\left\langle V_{i,*}^\top V_{i,*},\, \Lambda[\lambda, \lambda^\top] \right\rangle + 2V_{i,*}\Lambda[\lambda, s_i] + \Lambda[s_i, s_i] = v_i^2 \tag{20}$$

then the following inequality also holds:

$$v_i v_j - v_i V_{j,*}\lambda - v_j V_{i,*}\lambda + \left\langle V_{i,*}^\top V_{j,*},\, \Lambda[\lambda, \lambda^\top] \right\rangle \ge 0 \tag{28}$$

Proof. From Theorem 4.2, $V_{i,*}\lambda^{(k)} + s_i^{(k)} = \xi^{(k)} v_i$ holds. Substituting we have

$$
\begin{aligned}
& v_i v_j - v_i V_{j,*}\lambda - v_j V_{i,*}\lambda + \left\langle V_{i,*}^\top V_{j,*},\, \Lambda[\lambda, \lambda^\top] \right\rangle && (70)\\
&= v_i v_j + \sum_k \left(-v_i V_{j,*}(\xi^{(k)}\lambda^{(k)}) - v_j V_{i,*}(\xi^{(k)}\lambda^{(k)}) + \left\langle V_{i,*}^\top V_{j,*},\, \lambda^{(k)}(\lambda^{(k)})^\top \right\rangle \right) && (71)\\
&= \sum_k (\xi^{(k)})^2 v_i v_j - \xi^{(k)} v_i V_{j,*}\lambda^{(k)} - \xi^{(k)} v_j V_{i,*}\lambda^{(k)} + V_{i,*}\lambda^{(k)} V_{j,*}\lambda^{(k)} && (72)\\
&= \sum_k (\xi^{(k)} v_i - V_{i,*}\lambda^{(k)})(\xi^{(k)} v_j - V_{j,*}\lambda^{(k)}) && (73)\\
&= \sum_k s_i^{(k)} s_j^{(k)} && (74)\\
&\geq 0. && (75)
\end{aligned}
$$

The last line follows from the equality

$$\Lambda[s_i, s_j] = \sum_k s_i^{(k)} s_j^{(k)} \tag{76}$$

and the entrywise non-negativity of $\begin{pmatrix} \Lambda[\lambda,\lambda^\top] & \Lambda[\lambda,s^\top] & \lambda \\ \Lambda[s,\lambda^\top] & \Lambda[s,s^\top] & s \\ \lambda^\top & s^\top & 1 \end{pmatrix} \in \mathcal{N}$. □

A.3 Reformulation of Raghunathan et al. (2018)

Raghunathan et al. (2018) proposed the following SDP relaxation:

$$
\begin{aligned}
\min_{z,\, Z} \quad & \overline{c}^\top z_{n,*} && (29)\\
\text{s.t.} \quad & z_{i,*} \geq 0,\ z_{i,*} \geq \overline{W}[i-1] z_{i-1,*}, \\
& Z[z_{i,j}, z_{i,j}] = \overline{W}[i-1]_{j,*} Z[z_{i-1,*}, z_{i,j}], \\
& -Z[z_{i,j}, z_{i,j}] + (u_{i,j} + l_{i,j}) z_{i,j} - l_{i,j} u_{i,j} \geq 0,\ \forall i, j, \\
& \begin{pmatrix} Z & z \\ z^\top & 1 \end{pmatrix} \in \mathcal{S}^+
\end{aligned}
$$

The proposed SDP relaxation is based on directly transcribing the constraints in the following quadratically constrained quadratic program:

$$
\begin{aligned}
\min_{z} \quad & \overline{c}^\top z_{n,*} \\
\text{s.t.} \quad & z_{i,*} \geq 0,\ z_{i,*} \geq \overline{W}[i-1] z_{i-1,*}, \\
& z_{i,j}^2 = z_{i,j} \overline{W}[i-1]_{j,*} z_{i-1,*}, \\
& (z_{i,j} - l_{i,j})(u_{i,j} - z_{i,j}) \geq 0,\ \forall i, j
\end{aligned} \tag{77}
$$

To clarify the relationship with our proposed framework, we include the biases, introduce an additional equality constraint $\hat{z}_{i+1,*} = W[i] z_{i,*} + b[i]$ and the change the inequality from $z_{i+1,*} \geq W[i] z_{i,*} + b[i]$ to $z_{i,*} \geq \hat{z}_{i,*}$. Without loss of generality, we will assume that $\hat{l}_{i,j} < 0 < \hat{u}_{i,j}$ for all $i \geq 1$, otherwise the neuron (i,j) could be treated as the identity, if $\hat{l}_{i,j} \geq 0$, or zero, if $\hat{u}_{i,j} \leq 0$. The post-activation bounds are then $l_{i,j} = 0$ and $u_{i,j} = \hat{u}_{i,j}$, and the constraint $(z_{i,j} - l_{i,j})(u_{i,j} - z_{i,j}) \geq 0$ becomes $z_{i,j}(\hat{u}_{i,j} - z_{i,j}) \geq 0$. With a change of variables to the positive/negative splitting, (77) is equivalent to:

$$
\begin{aligned}
\min_{\lambda} \quad & \overline{c}^\top (\lambda_{n,*}^+ - \lambda_{n,*}^-) \\
\text{s.t.} \quad & \lambda_{i,*}^+ \geq 0,\ \lambda_{i,*}^- \geq 0,\ \lambda_{i+1,*}^+ - \lambda_{i+1,*}^- = \overline{W}[i] \lambda_{i,*}^+ + b[i], \\
& \lambda_{i,*}^+ \lambda_{i,*}^- = 0, \\
& \lambda_{i,j}^+ (\hat{u}_{i,j} - \lambda_{i,j}^+) \geq 0,\ i \geq 1, \\
& ((\lambda_{0,*}^+ - \lambda_{0,*}^-) - l_{0,*})(u_{0,*} - (\lambda_{0,*}^+ - \lambda_{0,*}^-)) \geq 0
\end{aligned} \tag{78}
$$

Directly transcribing the constraints of (78) results in the following SDP:

$$\begin{aligned} \min_{\lambda,\,\Lambda} \quad & c^\top \lambda \\ \text{s.t.} \quad & (l_{0,i} + u_{0,i}) U_{i,*}\lambda - \left\langle U_{i,*}^\top U_{i,*},\, \Lambda \right\rangle - u_i l_i \geq 0, \\ & \hat{u}_{i,j}\lambda_{i,j}^+ - \Lambda[\lambda_{i,j}^+, \lambda_{i,j}^+] \geq 0,\ i \geq 1, \\ & W[i]_{j,*}\lambda = b[i]_j, \\ & \Lambda[\lambda_{i,j}^+, \lambda_{i,j}^-] = 0, \\ & \begin{pmatrix} \Lambda & \lambda \\ \lambda^\top & 1 \end{pmatrix} \in S^+, \\ & \lambda \geq 0 \end{aligned} \tag{30}$$

where U is defined as $U_{i,*}\lambda = \lambda_{0,i}^+ - \lambda_{0,i}^-$.

A.3.1 Extensions to Raghunathan et al. (2018)

Direct Extensions. The following works have focused on improving some of the computational aspects of Raghunathan et al. (2018) (e.g., time to solution, memory usage) without aiming to improve the relaxation gap of Raghunathan et al. (2018).

The construction in Dvijotham et al. (2020) is based on the diagonally dominant SOS hierarchy (0-DSOS) relaxation. It approximates S^+ with the cone of diagonally dominant matrices, ultimately resulting in an LP. This is a further relaxation of (30), and reduces computation time at the expense of a larger relaxation gap.

Based on the fact that memory usage, rather than compute, is the main barrier for scaling of SDPs, Dathathri et al. (2020) proposes solving (30) using first order methods. This comes at the expense of increased time to solution.

SDP-based Branching. Previously, Zhang (2020) defined an SDP relaxation to be tight if it has a unique rank-one solution. Based on this, Anderson et al. (2021) proposed a metric measuring how far the solution is from being rank one and shows that a uniform partitioning of the input (before neuron bound propagation) leads to the greatest reduction of an upper bound of this metric. This approach can be thought of as branching on the input.

Critically, Theorem 4.1 states that the CPP (13) is always exact, even if the optimal solution is not rank one. If an optimizer of the (0-SOS) relaxation (14) happens to be completely positive, it is exact even if it is not rank one. This means that the rank-one condition is a sufficient but not necessary condition for exactness. For this reason, we propose a less-restrictive definition of tightness, allowing exact but non-rank-one optimizers; these points are further clarified in Section A.5.

Ma and Sojoudi (2020) work directly with (30), and propose a method equivalent to spatial branch and bound. In particular, they choose a basis $\{\phi_i\}$ for $\mathbb{R}^N$, and partition the search space along the axes aligned with $\{\phi_i\}$. To illustrate how their approach works, consider $\{\phi_i\}$ given by the standard basis vectors, and a partitioning of $\lambda_{i,j}^+$. Their approach generates M sub-problems that are determined by points $l_{i,j} = \gamma_0 < \gamma_1 < \ldots < \gamma_M = u_{i,j}$, where the mth sub-problem is defined by the following additional constraints:

$$\lambda_{i,j}^+ \leq \gamma_m \tag{79}$$

$$\lambda_{i,j}^+ \geq \gamma_{m-1} \tag{80}$$

$$\gamma_m \lambda_{i,j}^+ - \gamma_m\gamma_{m-1} - \Lambda[\lambda_{i,j}^+, \lambda_{i,j}^+] + \gamma_m \lambda_{i,j}^+ \geq 0 \tag{81}$$

Notice that equation (81) is the cross-quadratic constraint derived from $\lambda_{i,j}^+ \leq \gamma_m$ and $\lambda_{i,j}^+ \geq \gamma_{m-1}$.

A.4 Reformulation of Fazlyab et al. (2022)

Fazlyab et al. (2022) develop a method where they abstract the definition of particular constraint sets as quadratic constraints. In contrast to the quadratic constraints presented thus far in this paper, the quadratic constraints in Fazlyab et al. (2022) are defined by (potentially infinite) matrix cones.

A.4.1 Background on quadratic constraints

For example, if $\mathcal{X}$ is the input set, then the matrix cone describing $\mathcal{X}$ is defined as

$$\mathcal{P}_{\mathcal{X}} := \left\{ P \mid \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0,\ \forall \lambda \in \mathcal{X} \right\} \tag{82}$$

In the converse, $\mathcal{P}_{\mathcal{X}}$ can be used to over-approximate $\mathcal{X}$ via an infinite set of constraints:

$$\mathcal{X} \subseteq \bigcap_{P \in \mathcal{P}_{\mathcal{X}}} \left\{ \lambda \mid \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0 \right\} \tag{83}$$

The quadratic constraints are related to the SDP/CPP formulations of verification through the following equality:

$$\begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} = \left\langle P, \begin{pmatrix}\lambda \\ 1\end{pmatrix}\begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top \right\rangle. \tag{84}$$

The premise of the SDP/CPP formulations is relaxing the rank-one outer product $\begin{pmatrix}\lambda \\ 1\end{pmatrix}\begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top$ to $\begin{pmatrix}\Lambda & \lambda \\ \lambda^\top & 1\end{pmatrix} \in \mathcal{S}^+$.

A.4.2 Equivalence between atomic and infinite quadratic constraints

Our analysis is based on identifying an equivalent, finite set of quadratic constraints ("atomic constraints") for each infinite set of quadratic constraints proposed in Fazlyab et al. (2022)—these correspond with the extreme rays of $\mathcal{P}$. In other words, for each quadratic constraint set $\mathcal{P}$ with infinite cardinality, we will find a finite set $\hat{\mathcal{P}}$ such that

$$\bigcap_{P \in \mathcal{P}} \left\{ \lambda \mid \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0 \right\} = \bigcap_{P \in \hat{\mathcal{P}}} \left\{ \lambda \mid \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0 \right\}. \tag{85}$$

In particular, if

$$\mathcal{P} = \left\{ \sum_{i=1}^{N_{\text{ineq}}} \alpha_i P_{\text{ineq}}^{(i)} + \sum_{i=1}^{N_{\text{eq}}} \beta_i P_{\text{eq}}^{(i)} \mid \alpha_i \in \mathbb{R}_{\geq 0},\ \beta_i \in \mathbb{R} \right\} \tag{86}$$

then

$$\begin{aligned} \bigcap_{P \in \mathcal{P}} \left\{ \lambda \mid \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0 \right\} = \Bigg\{ \lambda \mid & \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P_{\text{ineq}}^{(i)} \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0, i = 1, \ldots, N_{\text{ineq}}, \\ & \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P_{\text{eq}}^{(j)} \begin{pmatrix}\lambda \\ 1\end{pmatrix} = 0, j = 1, \ldots, N_{\text{eq}} \Bigg\}. \end{aligned} \tag{87}$$

The equivalence can be shown as follows:

- If λ satisfies

$$\begin{aligned} &\begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P_{\text{ineq}}^{(i)} \begin{pmatrix}\lambda \\ 1\end{pmatrix} \geq 0, i = 1, \ldots, N_{\text{ineq}}, \\ &\begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P_{\text{eq}}^{(j)} \begin{pmatrix}\lambda \\ 1\end{pmatrix} = 0, j = 1, \ldots, N_{\text{eq}}, \end{aligned} \tag{88}$$

  then for any $P \in \mathcal{P}$ using the decomposition $P = \sum_{i=1}^{N_{\text{ineq}}} \alpha_i P_{\text{ineq}}^{(i)} + \sum_{i=1}^{N_{\text{eq}}} \beta_i P_{\text{eq}}^{(i)}$ with $\alpha_i \geq 0$,

$$\begin{aligned} \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P \begin{pmatrix}\lambda \\ 1\end{pmatrix} &= \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top \left(\sum_{i=1}^{N_{\text{ineq}}} \alpha_i P_{\text{ineq}}^{(i)} + \sum_{i=1}^{N_{\text{eq}}} \beta_i P_{\text{eq}}^{(i)} \right) \begin{pmatrix}\lambda \\ 1\end{pmatrix} && (89) \\ &= \sum_{i=1}^{N_{\text{ineq}}} \alpha_i \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P_{\text{ineq}}^{(i)} \begin{pmatrix}\lambda \\ 1\end{pmatrix} + \sum_{i=1}^{N_{\text{eq}}} \beta_i \begin{pmatrix}\lambda \\ 1\end{pmatrix}^\top P_{\text{eq}}^{(i)} \begin{pmatrix}\lambda \\ 1\end{pmatrix} && (90) \\ &\geq 0. && (91) \end{aligned}$$

So

$$\bigcap_{P\in\mathcal{P}}\left\{\lambda \mid \begin{pmatrix}\lambda\\1\end{pmatrix}^\top P \begin{pmatrix}\lambda\\1\end{pmatrix} \geq 0\right\} \supseteq \left\{\lambda \mid \begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{ineq}}^{(i)} \begin{pmatrix}\lambda\\1\end{pmatrix} \geq 0, i=1,\dots,N_{\text{ineq}}, \begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{eq}}^{(j)} \begin{pmatrix}\lambda\\1\end{pmatrix} = 0, j=1,\dots,N_{\text{eq}}\right\}. \tag{92}$$

- Given λ, suppose there exists i^* such that $\begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{ineq}}^{(i^*)} \begin{pmatrix}\lambda\\1\end{pmatrix} < 0$. Then, letting $\alpha_{i^*} = 1$, $\alpha_i = 0$ for $i \neq 0$ and $\beta_j = 0$, gives an example of $P = \sum_{i=1}^{N_{\text{ineq}}} \alpha_i P_{\text{ineq}}^{(i)} + \sum_{i=1}^{N_{\text{eq}}} \beta_i P_{\text{eq}}^{(i)} \in \mathcal{P}$ with

$$\begin{pmatrix}\lambda\\1\end{pmatrix}^\top P \begin{pmatrix}\lambda\\1\end{pmatrix} < 0 \tag{93}$$

  An analogous argument holds if there exists i^* such that $\begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{eq}}^{(i^*)} \begin{pmatrix}\lambda\\1\end{pmatrix} \neq 0$ with $\beta_{i^*} = -\text{sign}\left(\begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{eq}}^{(i^*)} \begin{pmatrix}\lambda\\1\end{pmatrix}\right)$.

  This shows that

$$\bigcap_{P\in\mathcal{P}}\left\{\lambda \mid \begin{pmatrix}\lambda\\1\end{pmatrix}^\top P \begin{pmatrix}\lambda\\1\end{pmatrix} \geq 0\right\} \subseteq \left\{\lambda \mid \begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{ineq}}^{(i)} \begin{pmatrix}\lambda\\1\end{pmatrix} \geq 0, i=1,\dots,N_{\text{ineq}}, \begin{pmatrix}\lambda\\1\end{pmatrix}^\top P_{\text{eq}}^{(j)} \begin{pmatrix}\lambda\\1\end{pmatrix} = 0, j=1,\dots,N_{\text{eq}}\right\}. \tag{94}$$

Combining the inclusions in both directions shows that the two constraints are equal.

A.4.3 Polytopic input constraints

To encode polytopic input sets, $\mathcal{X} = \{x \in \mathbb{R}^{k_0} \mid \overline{A}x \leq a\}$, Fazlyab et al. (2022) use the following quadratic constraint set:

$$\mathcal{P}_{\mathcal{X}} = \left\{P \mid P = \begin{bmatrix}\underline{A}^\top\Gamma\underline{A} & -\underline{A}^\top\Gamma a\\ -a^\top\Gamma\underline{A} & a^\top\Gamma a\end{bmatrix}\right\} \tag{95}$$

where $\Gamma \geq 0$, $\Gamma_{i,i} = 0$ and $\Gamma = \Gamma^\top$. Each $P \in \mathcal{P}_{\mathcal{X}}$ can be decomposed as

$$P = \sum_{i\neq j} \Gamma_{i,j} \begin{bmatrix}\underline{A}_{i,*}^\top\underline{A}_{j,*} & \frac{-a_j\underline{A}_{i,*}^\top - a_i\underline{A}_{j,*}^\top}{2}\\ \frac{-a_j\underline{A}_{i,*} - a_i\underline{A}_{j,*}}{2} & a_ia_j\end{bmatrix} \tag{96}$$

As a consequence, $\mathcal{P}_{\mathcal{X}}$ defines constraints of the form:

$$\sum_{i\neq j} \Gamma_{i,j}(\underline{A}_{i,*}\lambda - a_i)(\underline{A}_{j,*}\lambda - a_j) \geq 0 \tag{97}$$

with parameters $\Gamma_{i,j} \geq 0$, $\Gamma_{i,i} = 0$. The equivalent atomic constraints are

$$(\underline{A}_{i,*}\lambda - a_i)(\underline{A}_{j,*}\lambda - a_j) \geq 0, \quad \forall i \neq j. \tag{98}$$

A.4.4 ReLU constraints

Fazlyab et al. (2022) defined a global quadratic constraint for ReLUs, acting on $\begin{pmatrix}z\\\hat{z}\\1\end{pmatrix}$, through a constraint set of the form

$$\mathcal{P} = \begin{bmatrix}P_{1,1} & P_{1,2} & P_{1,3}\\ P_{2,1} & P_{2,2} & P_{2,3}\\ P_{3,1} & P_{3,2} & P_{3,3}\end{bmatrix} \tag{99}$$

where

$$P_{1,1} = 0 \in \mathbb{R}^{N\times N} \tag{100}$$
$$P_{1,2} = \mathrm{diag}(\rho) + T \in \mathbb{R}^{N\times N} \tag{101}$$
$$P_{1,3} = -\nu \in \mathbb{R}^{N} \tag{102}$$
$$P_{2,2} = -2(\mathrm{diag}(\rho) + T) \in \mathbb{R}^{N\times N} \tag{103}$$
$$P_{2,3} = \nu + \eta \in \mathbb{R}^{N} \tag{104}$$
$$P_{3,3} = 0 \in \mathbb{R} \tag{105}$$

where $\eta, \nu \geq 0$, ρ is unconstrained, and the matrix T is defined as

$$T := \sum_{1\leq i<j\leq N} \gamma_{i,j}(e_i - e_j)(e_i - e_j)^\top \tag{106}$$

where e_i is the ith basis vector, and $\gamma_{i,j} \geq 0$. Each quadratic constraint, P, acting on $\begin{pmatrix} z \\ \hat{z} \\ 1 \end{pmatrix}$ can be converted to an equivalent quadratic constraint acting on $\begin{pmatrix} \lambda^+ \\ \lambda^- \\ 1 \end{pmatrix}$ by pre- and post-multiplying by $\begin{pmatrix} I & 0 & 0 \\ I & -I & 0 \\ 0 & 0 & 1 \end{pmatrix}$

We have taken the liberty to substitute in $\alpha = 0$, $\beta = 1$ in $z_{i,j} = \max(\alpha\hat{z}_{i,j}, \beta\hat{z}_{i,j})$. We have also renamed constraints to prevent notational clash with our notation.

The quadratic constraint set, $\mathcal{P}$, thus enforces quadratic constraints of the form (Fazlyab et al., 2022, Lemma 3 and Appendix D):

$$\begin{aligned} 0 \geq &\sum_{(i,j)} \left[\rho_{(i,j)}\lambda^+_{i,j}\lambda^-_{i,j} - \nu_{i,j}\lambda^+_{i,j} - \eta_{i,j}\lambda^-_{i,j}\right] \\ &+ \sum_{(i,j)\neq(k,l)} \gamma_{(i,j),(k,l)}(\lambda^+_{i,j} - \lambda^+_{k,l})(\lambda^-_{i,j} - \lambda^-_{k,l}) \end{aligned} \tag{107}$$

Each term in the sum decomposes into the atomic constraints (together, equivalent to the full sum)

$$\lambda^+_{i,j} \geq 0 \tag{108}$$
$$\lambda^-_{i,j} \geq 0 \tag{109}$$
$$\lambda^+_{i,j}\lambda^-_{i,j} = 0 \tag{110}$$
$$(\lambda^+_{i,j} - \lambda^+_{k,l})(\lambda^-_{i,j} - \lambda^-_{k,l}) \leq 0 \tag{111}$$

Expanding out (111),

$$0 \geq (\lambda^+_{i,j} - \lambda^+_{k,l})(\lambda^-_{i,j} - \lambda^-_{k,l}) \tag{112}$$
$$= \lambda^+_{i,j}\lambda^-_{i,j} - \lambda^+_{i,j}\lambda^-_{k,l} - \lambda^+_{k,l}\lambda^-_{i,j} + \lambda^+_{k,l}\lambda^-_{k,l} \tag{113}$$
$$= -\lambda^+_{i,j}\lambda^-_{k,l} - \lambda^+_{k,l}\lambda^-_{i,j} \tag{114}$$

Since either $\lambda^+_{i,j} = 0$ or $\lambda^-_{i,j} = 0$, the constraint $\lambda^+_{i,j}\lambda^-_{k,l} + \lambda^+_{k,l}\lambda^-_{i,j} \geq 0$ can be decomposed into two separate constraints $\lambda^+_{i,j}\lambda^-_{k,l} \geq 0$ and $\lambda^+_{k,l}\lambda^-_{i,j} \geq 0$.

In summary, the equivalent atomic constraints are:

$$\lambda^+_{i,j} \geq 0, \qquad \lambda^-_{i,j} \geq 0, \quad \lambda^+_{i,j}\lambda^-_{i,j} = 0 \quad \forall(i,j) \tag{115}$$
$$\lambda^+_{i,j}\lambda^-_{k,l} \geq 0, \qquad \lambda^-_{i,j}\lambda^+_{k,l} \geq 0 \quad \forall(i,j) \neq (k,l). \tag{116}$$

A.4.5 Extensions to Fazlyab et al. (2022)

Direct Extensions. Newton and Papachristodoulou (2021) showed that the formulation proposed in Fazlyab et al. (2022) exhibits chordal sparsity. They exploit this insight by using an off-the-shelf solver, SparseCoLo, capable of taking advantage of chordal sparsity. This approach offers an advantage for deep networks. Because the formulation is the same as Fazlyab et al. (2022), it exhibits the same relaxation gap.

A.5 Rank and Exactness

Zhang (2020) defined an SDP relaxation to be tight if it has a unique rank-one solution. The rationale for such a definition is because if an interior-point method converges to a maximum rank solution, the optimal solution will not be rank-one unless it is unique.

Based on Theorem 4.1, however, we propose a less restrictive definition of tightness for SDP-based relaxations. Specifically, we say that an SDP-relaxation is tight if the optimal value is equal to that of (6). This definition is motivated by the fact that Theorem 4.1 does not preclude high rank solutions. As a consequence, if the optimal solution for Problem (14)(0-SOS) is feasible for Problem (13) (CPP), then it is exact, regardless of its rank.

In this section, we show that the revised definition is not vacuous, and there are in fact verification instances where the optimal solution of the SDP-relaxation is exact despite not being rank-one, and show how the optimal solutions of (6) can be recovered from the SDP solution.

Consider the ReLU network given by the following weights and biases,

$$W[0] = \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix}, \qquad b[0] = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{117}$$

$$W[1] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}, \qquad b[1] = \begin{pmatrix} 2.1 \\ 0 \\ 2.1 \\ 0 \end{pmatrix} \tag{118}$$

$$W[2] = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 \end{pmatrix}, \qquad b[2] = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{119}$$

$$W[3] = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \qquad b[3] = \begin{pmatrix} 2.1 \end{pmatrix} \tag{120}$$

and the safety rule to be verified defined by:

$$\mathcal{X} := \{z_{0,*} \mid -1 \leq z_{0,*} \leq 1\} \subset \mathbb{R}^2 \tag{121}$$

$$\mathcal{Y} := \{z_{3,*} \mid z_{3,*} \geq -2.1\} \subset \mathbb{R}. \tag{122}$$

We form the 0-SOS relaxation, (14), and solve the SDP using SCS; this had an optimal value of -2.0002017.

We then factored the optimal solution using the using the Alternate Least Square Using Projected Gradient Descent algorithm—Lin (2007) (implemented in the NMF.jl package). We found a rank-four factorization with a maximum entry-wise error of 0.0001158, and objective value of -2.0002022. The corresponding values of $\lambda_{0,*}^{\pm,(k)}$ and $\xi^{(k)}$ are as follows:

$$k = 1 \qquad \lambda_{0,*}^{+,(1)} = \begin{pmatrix} 0.491844 \\ 3.32366 \times 10^{-5} \end{pmatrix}, \quad \lambda_{0,*}^{-,(1)} = \begin{pmatrix} 0.0 \\ 0.491843 \end{pmatrix} \quad \xi^{(1)} = 0.491849 \tag{123}$$

$$k = 2 \qquad \lambda_{0,*}^{+,(2)} = \begin{pmatrix} 4.76418 \times 10^{-5} \\ 0.504452 \end{pmatrix}, \quad \lambda_{0,*}^{-,(2)} = \begin{pmatrix} 0.504447 \\ 0.0 \end{pmatrix} \quad \xi^{(2)} = 0.504478 \tag{124}$$

$$k = 3 \qquad \lambda_{0,*}^{+,(3)} = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}, \quad \lambda_{0,*}^{-,(3)} = \begin{pmatrix} 0.522856 \\ 0.522838 \end{pmatrix} \quad \xi^{(3)} = 0.522831 \tag{125}$$

$$k = 4 \qquad \lambda_{0,*}^{+,(4)} = \begin{pmatrix} 0.411613 \\ 0.411621 \end{pmatrix}, \quad \lambda_{0,*}^{-,(3)} = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \quad \xi^{(4)} = 0.411569 \tag{126}$$

It can be seen that each solution input corresponds to a corner of $\mathcal{X}$. Each of these solutions are plotted alongside the level sets of the network in Figure 5. In this example, the optimal SDP-relaxation had a non-negative factorization, but this is not generally guaranteed. This does suggest, however, that one should attempt to find a non-negative factorization of the 0-SOS relaxation as a certificate of optimality.

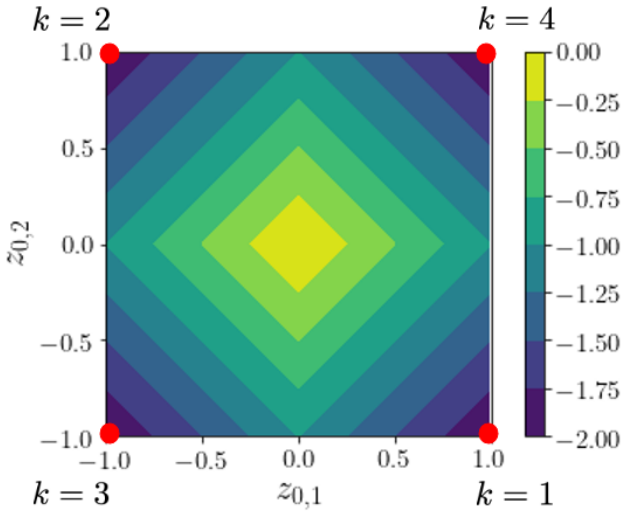


Figure 5: This figure plots the level set of the network defined by the weights and biases in Equations (117)-(120), as well as the inputs, (123)-(126), corresponding to individual factors in the SDP solution.