Journal of Machine Learning Research 26 (2025) 1-59 Submitted 1/21; Revised 9/24; Published 5/25

Towards Optimal Branching of Linear and Semidefinite Relaxations for Neural Network Robustness Certification

Brendon G. Anderson[*] BGANDERSON@BERKELEY.EDU
Ziye Ma[†] ZIYEMA@BERKELEY.EDU
Jingqi Li[†] JINGQILI@BERKELEY.EDU
Somayeh Sojoudi[†] SOJOUDI@BERKELEY.EDU
[*]*Department of Mechanical Engineering, University of California, Berkeley*
[†]*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley*

Editor: Miguel Carreira-Perpinan

Abstract

In this paper, we study certifying the robustness of ReLU neural networks against adversarial input perturbations. To diminish the relaxation error suffered by the popular linear programming (LP) and semidefinite programming (SDP) certification methods, we take a branch-and-bound approach to propose partitioning the input uncertainty set and solving the relaxations on each part separately. We show that this approach reduces relaxation error, and that the error is eliminated entirely upon performing an LP relaxation with a partition intelligently designed to exploit the nature of the ReLU activations. To scale this approach to large networks, we consider using a coarser partition whereby the number of parts in the partition is reduced. We prove that computing such a coarse partition that directly minimizes the LP relaxation error is NP-hard. By instead minimizing the worst-case LP relaxation error, we develop a closed-form branching scheme in the single-hidden layer case. We extend the analysis to the SDP, where the feasible set geometry is exploited to design a branching scheme that minimizes the worst-case SDP relaxation error. Experiments on MNIST, CIFAR-10, and Wisconsin breast cancer diagnosis classifiers demonstrate significant increases in the percentages of test samples certified. By independently increasing the input size and the number of layers, we empirically illustrate under which regimes the branched LP and branched SDP are best applied. Finally, we extend our LP branching method into a multi-layer branching heuristic, which attains comparable performance to prior state-of-the-art heuristics on large-scale, deep neural network certification benchmarks.

Keywords: ReLU neural networks, robustness certification, adversarial attacks, convex optimization, branch-and-bound

1. Introduction

It is evident that the data used in real-world engineering systems has uncertainty. This uncertainty takes many forms, including corruptions, random measurement noise, and adversarial attacks (Franceschi et al., 2018; Balunovic et al., 2019; Jin et al., 2020). Recently, researchers have shown that the performance of neural networks can be highly sensitive to these uncertainties in the input data (Szegedy et al., 2014; Fawzi et al., 2016; Su et al., 2019). Clearly, safety-critical systems, such as autonomous vehicles (Bojarski et al., 2016;

Wu et al., 2017) and the power grid (Kong et al., 2017; Muralitharan et al., 2018; Pan et al., 2019), must be robust against fluctuations and uncertainty in the inputs to their decision-making algorithms. This fact has motivated a massive influx of research studying methods to certify the robustness of neural networks against uncertainty in their input data (Wong and Kolter, 2018; Raghunathan et al., 2018; Xiang and Johnson, 2018; Weng et al., 2018; Zhang et al., 2018; Royo et al., 2019; Fazlyab et al., 2020; Anderson et al., 2020; Anderson and Sojoudi, 2022; Ma and Sojoudi, 2020; Jin et al., 2021).

The two primary settings taken in the robustness certification literature consider either random input uncertainty or adversarial uncertainty. In the former, the neural network input is assumed to be random and follow a known probability distribution. For instance, the works Weng et al. (2019); Anderson and Sojoudi (2022) derive high-probability guarantees that this randomness causes no misclassification or unsafe output. In the adversarial setting, the input is assumed to be unknown but contained in a prescribed input uncertainty set. The goal here is to certify that all possible inputs from the uncertainty set are mapped to outputs that the network operator deems as safe (Wong and Kolter, 2018; Royo et al., 2019). In this paper, we take the latter, worst-case perspective. We remark that this approach is more generally applicable than the techniques for random inputs. Indeed, certifying that a network is robust against adversarial inputs immediately also certifies it is robust against random inputs distributed over the same uncertainty set; if the worst-case input cannot cause a failure, then neither will randomly selected ones.

The problem of adversarial robustness certification amounts to proving that all possible outputs in the output set, i.e., the image of the input uncertainty set under the mapping of the network, are contained in the safe set. However, this output set is generally nonconvex, even when the input set is convex. Consequently, the certification decision problem has been shown to be NP-complete, and the optimization-based formulation for solving it is an NP-hard, nonconvex optimization problem (Katz et al., 2017; Weng et al., 2018). To make the problem more tractable, researchers have proposed various ways to over-approximate the nonconvex output set with a convex one. Performing the certification over the convex surrogate reduces the problem to an easy-to-solve convex relaxation, and if the relaxation issues a robustness certificate for the outer approximation, it also issues a certificate for the true set of outputs. Such convex relaxation-based certification algorithms are sound, but generally incomplete; the network may be robust even if the algorithm is unable to verify it. Thus, a line of works has been developed in order to increase the percentage of test inputs that convex relaxation-based methods are able to certify (Wong and Kolter, 2018; Raghunathan et al., 2018; Fazlyab et al., 2020; Tjandraatmadja et al., 2020; Müller et al., 2022b; Batten et al., 2021; Wang et al., 2021).

Perhaps the simplest and most popular outer approximation technique is based on a linear programming (LP) relaxation of the ReLU activation function (Ehlers, 2017; Wong and Kolter, 2018). However, this method has been shown to yield relatively loose outer approximations, making it possible for the approximating set to contain unsafe outputs, even if the true output set is entirely safe. If this occurs, the convex relaxation fails to issue a certificate of robustness, despite the fact the network is indeed robust. A semidefinite programming (SDP) relaxation was proposed in Raghunathan et al. (2018) and shown to yield tighter outer approximations when compared to the LP method. Other methods, such as the quadratically-constrained semidefinite program (Fazlyab et al., 2020), use sophisticated relaxations to tighten the approximation, but these SDP-based methods inevitably gain accuracy at the expense of enlarging the computational cost, and are still susceptible to relaxation error. The work Tjandraatmadja et al. (2020) introduces a joint ReLU LP relaxation that is tighter than the per-neuron approach, but may require an exponential number of linear inequalities and is weaker overall in comparison to the per-neuron relaxation when embedded into a branch-and-bound scheme (Wang et al., 2021). We follow the certification literature's emphasis on ReLU networks and focus our attention in this paper on such models, which, aside from their certifiable structure, are extremely popular for their non-vanishing gradient property and their quick training times (Xiang and Johnson, 2018). We remark that branch-and-bound methods that are derived for ReLU models may still be applied to models with non-ReLU activations, either by restricting the branching scheme to neurons with ReLU activations, or by naively applying the branching scheme to the non-ReLU neurons as if they had ReLU activations.

1.1 Branch-and-Bound Certification

As eluded to above, instead of resorting to exorbitantly costly relaxation techniques to lower the approximation error, e.g., using heuristic variants of Lasserre's hierarchy (Chen et al., 2020), state-of-the-art convex relaxation-based verifiers have turned to branch-and-bound methods, where the nonconvex optimization domain is recursively partitioned and outer-approximated by smaller convex sets. In fact, the verifier that won the 2021, 2022, and 2023 VNN certification competitions is α,β-CROWN (Bak et al., 2021; Müller et al., 2022a; Brix et al., 2023; Wang et al., 2021), which is a branch-and-bound method that uses linear programming relaxations in the bounding steps. α,β-CROWN uses α-CROWN to generate linear upper and lower bounds on the neural network output, together with the heuristic per-neuron branching methods BaBSR (Bunel et al., 2020) and filtered smart branching (FSB) (De Palma et al., 2021).

The success of α,β-CROWN is interesting, since the BaBSR and FSB branching strategies are designed in the context of Ehler's per-neuron "triangle" convex relaxations (Ehlers, 2017), which is a different and, when activation bounds are adequately chosen, tighter bounding approach than α-CROWN's linear bounding of the output, albeit computationally more expensive. FSB remains the state-of-the-art branching heuristic, outperforming BaBSR (Wang et al., 2021; De Palma et al., 2021), which in turn beat a state-of-the-art mixed-integer approach and all prior ReLU neuron splitting methods: Reluplex, Planet, and Neurify (Bunel et al., 2020; Katz et al., 2017; Ehlers, 2017; Wang et al., 2018). FSB works by assigning scores to each neuron that are computed by quickly estimating the improvement in the optimization objective upon splitting that neuron. In this paper, we propose two branch-and-bound certification methods, one utilizing the triangle relaxation of Ehlers (2017) as a bounding method and the other utilizing the SDP relaxation of Raghunathan et al. (2018), and for these bounding methods we derive novel branching schemes that minimize the worst-case relaxation error.

It is worth briefly mentioning here that branch-and-bound certification techniques can make use of parallel computing when solving independent subproblems arising from the feasible set partitioning (Lu and Kumar, 2020; Wu et al., 2020; Xu et al., 2021). This improves the scalability and efficiency of neural network certification. Our proposed branch-and-bound methods are naturally able to leverage such parallelizations.

1.2 Contributions

In this paper, we fully exploit the piecewise linear structure of ReLU networks in order to tighten convex relaxations for robustness certification. A condensed summary of our main contributions is as follows:

1. For the LP relaxation, we show that a methodically designed finite partition of the input set allows one to attain zero relaxation error. We then use the structure of this motivating partition to derive a closed-form branching scheme that minimizes worst-case relaxation error for single-hidden layer networks. This novel branching scheme makes theoretically principled strides towards optimal LP branching. We show how to extend our branching scheme into a multi-layer branching heuristic, applicable to deep neural networks.

2. We prove that computing the branching neuron that minimizes the actual LP relaxation error is NP-hard, in turn theoretically justifying our approach of minimizing the worst-case relaxation error.

3. For the SDP relaxation, we develop a geometrically interpretable measure for how far the solution is from being rank-1. We then show in the single-hidden layer case that, when branching along a given coordinate, this rank-1 gap is minimized by a uniform grid. Finally, we derive the branching coordinate that minimizes the worst-case relaxation error.

4. We embed our branching schemes into a branch-and-bound framework, and empirically validate the effectiveness of our approaches on the MNIST, CIFAR-10, and Wisconsin breast cancer diagnosis benchmarks. We experimentally show on these datasets that our LP branching scheme outperforms the state-of-the-art filtered smart branching, the branching heuristic used in α, β-CROWN to win the 2021, 2022, and 2023 VNN certification competitions. Our SDP branching scheme is found to yield even higher certified percentages. We also run experiments on large-scale deep neural network certification problems involving multiple layers with complex structures, and we find that our method yields performance comparable to that of these state-of-the-art methods. Overall, our experiments suggest that our theoretically principled single-hidden layer methods outperform the current methods on moderately-sized problems, and our heuristic extensions to deep networks achieve comparable performance on large-scale problems.

This paper is a major extension of the conference paper Anderson et al. (2020). Notably, the NP-hardness result, all of the SDP results, and all of the experiments of this paper (the second, third, and fourth major contributions listed above) are new additions. The contributions of this paper culminate into two theoretically principled branching-based convex relaxations for ReLU robustness certification. Our experiments demonstrate that the proposed approaches are effective and efficient, and from these results we develop general guidance for which network size and structure regimes the LP, branched LP, SDP, and branched SDP are best applied.

1.3 Outline

This paper is organized as follows. Section 2 introduces the ReLU robustness certification problem as well as its basic LP and SDP relaxations. In Section 3, we study the effect of partitioning the input uncertainty set for the LP relaxation. After developing formal guarantees for its effectiveness, we develop a branching strategy that optimally reduces the worst-case relaxation error for networks with a single hidden layer. We then extend our theoretically principled LP branching scheme to a multi-layer branching heuristic. Similarly, in Section 4 we study the partitioned SDP relaxation, bound its error, and propose a coordinate-wise branching scheme to minimize the bound. Section 6 demonstrates the developed methods on numerical examples and studies the effectiveness of branching on the LP and SDP as the network grows in width and depth. Finally, we conclude in Section 7.

1.4 Notations

For the reader's convenience, we list our commonly used symbols in Table 1, some of which will be more rigorously defined in the following sections.

For an index set $\mathcal{I} \subseteq \{1, 2, \dots, n\}$, the symbol $\mathbf{1}_{\mathcal{I}}$ is used to denote the n-vector with $(\mathbf{1}_{\mathcal{I}})_i = 1$ for $i \in \mathcal{I}$ and $(\mathbf{1}_{\mathcal{I}})_i = 0$ for $i \in \mathcal{I}^c = \{1, 2, \dots, n\} \setminus \mathcal{I}$. For a function $f \colon \mathbb{R}^n \to \mathbb{R}^m$ and the point $x \in \mathbb{R}^n$, we denote the i^{th} element of the m-vector $f(x)$ by $f_i(x)$. For a matrix $X \in \mathbb{R}^{m \times n}$, we use two indices to denote an element of X and one index to denote a column of X, unless otherwise stated. In particular, the (i, j) element and the i^{th} column of X are denoted by X_{ij} and X_i respectively, or, when necessary for clarity, by $(X)_{ij}$ and $(X)_i$ respectively. Furthermore, for a function $f \colon \mathbb{R} \to \mathbb{R}$, we define $f(X)$ to be an $m \times n$ matrix whose (i, j) element is equal to $f(X_{ij})$ for all $i \in \{1, 2, \dots, m\}$ and all $j \in \{1, 2, \dots, n\}$. If Z is a square $n \times n$ matrix, we use the notation $\operatorname{diag}(Z)$ to mean the n-vector $(Z_{11}, Z_{22}, \dots, Z_{nn})$ and $\operatorname{tr}(Z)$ to mean the trace of Z. When $Z \in \mathbb{S}^n$, we write $Z \succeq 0$ to mean that Z is positive semidefinite.

We use $\mathbb{I}$ to denote the indicator function, i.e., $\mathbb{I}(A) = 1$ if event A holds and $\mathbb{I}(A) = 0$ if event A does not hold. For a set $\mathcal{T} \subseteq \mathbb{R}$, we respectively denote its infimum and supremum by $\inf \mathcal{T}$ and $\sup \mathcal{T}$. For a function $f \colon \mathbb{R}^n \to \mathbb{R}$ and a set $\mathcal{X} \subseteq \mathbb{R}^n$, we write $\inf_{x \in \mathcal{X}} f(x)$ to mean $\inf\{f(x) : x \in \mathcal{X}\}$ and similarly for suprema. Finally, we assume that all infima and suprema throughout the paper are attained.

2. Problem Statement

2.1 Description of the Network and Uncertainty

In this paper, we consider a pre-trained K-layer ReLU neural network defined by

$$\begin{aligned} x^{[0]} &= x, \\ \hat{z}^{[k]} &= W^{[k-1]}x^{[k-1]} + b^{[k-1]}, \\ x^{[k]} &= \operatorname{ReLU}(\hat{z}^{[k]}), \\ z &= x^{[K]}, \end{aligned} \tag{1}$$

for all $k \in \{1, 2, \dots, K\}$. Here, the neural network input is $x \in \mathbb{R}^{n_x}$, the output is $z \in \mathbb{R}^{n_z}$, and the k^{th} layer's preactivation is $\hat{z}^{[k]} \in \mathbb{R}^{n_k}$. The parameters $W^{[k]} \in \mathbb{R}^{n_{k+1} \times n_k}$ and $b^{[k]} \in \mathbb{R}^{n_{k+1}}$ are the weight matrix and bias vector applied to the k^{th} layer's activation $x^{[k]} \in \mathbb{R}^{n_k}$, respectively. Without loss of generality,[1] we assume that the bias terms are accounted for in the activations $x^{[k]}$, thereby setting $b^{[k]} = 0$ for all layers k. We remark that, since we make no assumptions on the linear transformations defined by $W^{[k]}$, our work also applies to networks with linear convolutional layers. We let the function $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ denote the map $x \mapsto z$ defined by (1). In the case that f is a classification network, the output dimension n_z equals the number of classes. The problem at hand is to certify the robustness in the neural network output z when the input x is subject to uncertainty.

To model the input uncertainty, we assume that the network inputs are unknown but contained in a compact set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, called the *input uncertainty set*. We assume that the set $\mathcal{X}$ is a convex polytope, so that the condition $x \in \mathcal{X}$ can be written as a finite number of affine inequalities and equalities. In the adversarial robustness literature, the input uncertainty set is commonly modeled as $\mathcal{X} = \{x \in \mathbb{R}^{n_x} : \|x - \bar{x}\|_\infty \leq \epsilon\}$, where $\bar{x} \in \mathbb{R}^{n_x}$ is a nominal input to the network and $\epsilon > 0$ is an uncertainty radius (Wong and Kolter, 2018; Raghunathan et al., 2018). We remark that our generalized polytopic model for $\mathcal{X}$ includes this common case. The primary theme of this paper revolves around partitioning the input uncertainty set in order to strengthen convex robustness certification methods. Let us briefly recall the definition of a partition.

Definition 1 (Partition) *The collection $\{\mathcal{X}^{(j)} \subseteq \mathcal{X} : j \in \{1, 2, \dots, p\}\}$ is said to be a* partition *of the input uncertainty set $\mathcal{X}$ if $\mathcal{X} = \cup_{j=1}^{p} \mathcal{X}^{(j)}$ and $\mathcal{X}^{(j)} \cap \mathcal{X}^{(k)} = \emptyset$ for all $j \neq k$. The set $\mathcal{X}^{(j)}$ is called the j^{th}* input part*.*

We generally use the terminology *branching* to refer to partitioning with two parts, which is typically applied in a recursive fashion.

Now, in order to describe the robustness of the network (1), we need a notion of permissible outputs. For this, we consider a *safe set*, denoted $\mathcal{S} \subseteq \mathbb{R}^{n_z}$. If an output $z \in \mathbb{R}^{n_z}$ is an element of $\mathcal{S}$, we say that z is *safe*. It is common in the robustness certification literature

1. Note that $Wx + b = \begin{bmatrix} W & b \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} =: \tilde{W}\tilde{x}$, so the bias term b can be eliminated by appending a fixed value of 1 at the end of the activation x. This parameterization can be used throughout this paper by using matching lower and upper activation bounds of 1 in the last coordinate of each layer.

Table 1: List of commonly used symbols.

Symbol	Meaning
$\mathbb{R}^n$	Set of n-vectors with real elements
$\mathbb{S}^n$	Set of $n \times n$ matrices with real elements
e_i	i^{th} standard basis vector of $\mathbb{R}^n$
$\mathbf{1}_n$	n-vector of all ones
$X \leq Y$	The array X is element-wise less than or equal to the array Y
$X \odot Y$	Element-wise (Hadamard) product between arrays X and Y
$X \oslash Y$	Element-wise (Hadamard) division of array X by array Y
$\lvert\mathcal{S}\rvert$	Cardinality of the set $\mathcal{S}$
$\lVert \cdot \rVert_*$	Dual norm of the norm $\lVert \cdot \rVert$
$d_{\lVert\cdot\rVert}(\mathcal{X})$	Diameter of $\mathcal{X} \subseteq \mathbb{R}^n$ with respect to norm $\lVert \cdot \rVert$; $d_{\lVert\cdot\rVert}(\mathcal{X}) = \sup_{x,x' \in \mathcal{X}} \lVert x - x' \rVert$
f	Neural network
ReLU	Rectified linear unit; $\text{ReLU}(\cdot) = \max\{0, \cdot\}$
x, z	Neural network input and output, respectively
$x^{[k]}$, $\hat{z}^{[k]}$	Neural network activations and preactivations at layer k, respectively
$W^{[k]}$	Neural network weights at layer k
$l^{[k]}$, $u^{[k]}$	Neural network preactivation lower and upper bounds at layer k, respectively
$\mathcal{X}$	Neural network input uncertainty set
$f(\mathcal{X})$	Neural network output uncertainty set given the input uncertainty set $\mathcal{X}$
$\phi(\mathcal{X})$	Optimal value of certification problem over the input uncertainty set $\mathcal{X}$

to consider (possibly unbounded) polyhedral safe sets. We take this same perspective, and assume that $\mathcal{S}$ is defined by

$$\mathcal{S} = \{z \in \mathbb{R}^{n_z} : Cz \leq d\},$$

where $C \in \mathbb{R}^{n_{\mathcal{S}} \times n_z}$ and $d \in \mathbb{R}^{n_{\mathcal{S}}}$ are given. Note that, again, our model naturally includes the classification setting. In particular, suppose that $i^* \in \arg\max_{i \in \{1,2,\dots,n_z\}} f_i(\bar{x})$ is the true class of the nominal input $\bar{x}$. Then, define the i^{th} row of $C \in \mathbb{R}^{n_z \times n_z}$ to be

$$c_i^\top = e_i^\top - e_{i^*}^\top$$

and d to be the zero vector. Then it is immediately clear that an input x (which can be thought of as a perturbed version of $\bar{x}$) is safe if and only if $f_i(x) \leq f_{i^*}(x)$ for all i, i.e., the network classifies x into class i^*. From this classification perspective, the safe set represents the set of outputs without any misclassification (with respect to the nominal input $\bar{x}$). From here on, we consider f, $\mathcal{X}$, and $\mathcal{S}$ in their general forms—we do not restrict ourselves to the classification setting.

2.2 The Robustness Certification Problem

The fundamental goal of the robustness certification problem is to prove that all inputs in the input uncertainty set map to safe outputs, i.e., that $f(x) \in \mathcal{S}$ for all $x \in \mathcal{X}$. If this certificate holds, the network is said to be *certifiably robust*, which of course is a property that holds with respect to a particular input uncertainty set. The robustness condition can also be written as $f(\mathcal{X}) \subseteq \mathcal{S}$, or equivalently

$$\sup_{x\in\mathcal{X}} \left(c_i^\top f(x) - d_i\right) \leq 0 \text{ for all } i \in \{1,2,\dots,n_{\mathcal{S}}\},$$

where $c_i^\top$ is the i^{th} row of C. Under this formulation, the certification procedure amounts to solving $n_{\mathcal{S}}$ optimization problems. The methods we develop in this paper can be applied to each of these optimizations individually, and therefore in the sequel we focus on a single optimization problem by assuming that $n_{\mathcal{S}} = 1$, namely $\sup_{x\in\mathcal{X}} c^\top f(x)$. We also assume without loss of generality that $d = 0$. If d were nonzero, one may absorb d into the cost vector c and modify the network model by appending a fixed value of 1 at the end of the output vector $f(x)$. Under these formulations, we write the robustness certification problem as

$$\phi^*(\mathcal{X}) = \sup\{c^\top z : z = f(x),\ x \in \mathcal{X}\}, \tag{2}$$

and recall that we seek to certify that $\phi^*(\mathcal{X}) \leq 0$.

Since the function f is in general nonconvex, the nonlinear equality constraint $z = f(x)$ makes the optimization (2) a nonconvex problem and the set $f(\mathcal{X})$ a nonconvex set. Furthermore, since the intermediate activations and preactivations of the network generally have a large dimension in practice, the problem (2) is typically a high-dimensional problem. Therefore, computing an exact robustness certificate, as formulated in (2), is computationally intractable. Instead of directly maximizing $c^\top z$ over the nonconvex output set $f(\mathcal{X})$, one can overcome these hurdles by optimizing over a convex outer approximation $\hat{f}(\mathcal{X}) \supseteq f(\mathcal{X})$. Indeed, this new problem is a convex relaxation of the original problem, so it is generally easier and more efficient to solve. If the convex outer approximation is shown to be safe, i.e., $\hat{f}(\mathcal{X}) \subseteq \mathcal{S}$, then the true nonconvex set $f(\mathcal{X})$ is also known to be safe, implying that the robustness of the network is certified. Figure 1 illustrates this idea.

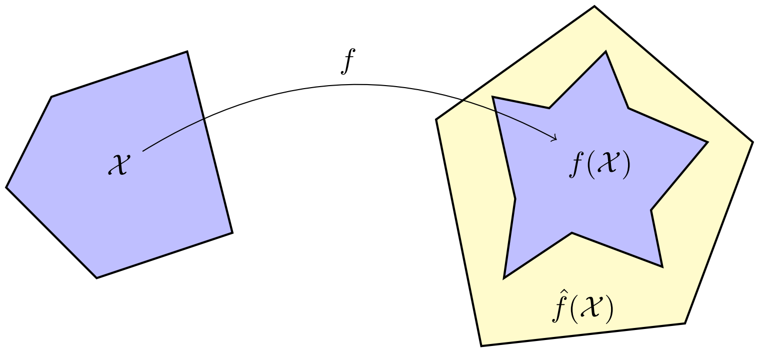


Figure 1: The set $\hat{f}(\mathcal{X})$ is a convex outer approximation of the nonconvex set $f(\mathcal{X})$. If the outer approximation is safe, i.e., $\hat{f}(\mathcal{X}) \subseteq \mathcal{S}$, then so is $f(\mathcal{X})$.

A fundamental drawback to the convex relaxation approach to robustness certification is as follows: if the outer approximation $\hat{f}(\mathcal{X})$ is too loose, then it may intersect with the unsafe region of the output space, meaning $\hat{f}(\mathcal{X}) \not\subseteq \mathcal{S}$, even in the case where the true output set is safe. In this scenario, the convex relaxation fails to issue a certificate of robustness, since the optimal value to the convex relaxation is positive, which incorrectly suggests the presence of unsafe network inputs within $\mathcal{X}$. This situation is illustrated in Figure 2.

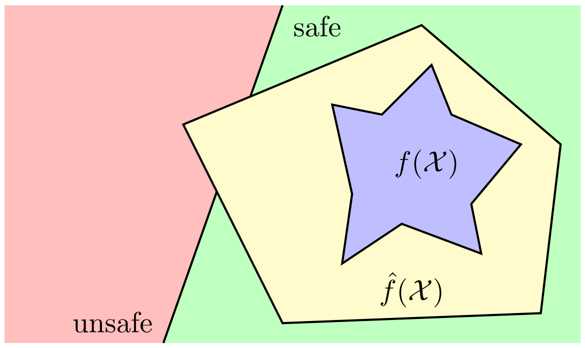


Figure 2: This scenario shows that if the convex outer approximation $\hat{f}(\mathcal{X})$ is too large, meaning the relaxation is too loose, then the convex approach fails to issue a certificate of robustness.

The fundamental goal of this paper is to develop convex relaxation methods for robustness certification such that the outer approximation tightly fits $f(\mathcal{X})$, in effect granting strong certificates while maintaining the computational and theoretical advantages of convex optimization. We focus on two popular types of convex relaxations, namely, LP (Wong and Kolter, 2018) and SDP (Raghunathan et al., 2018) relaxations. It has been shown that the SDP relaxation for ReLU networks is tighter than the LP relaxation, with the tradeoff of being computationally more demanding. Our general approach for both the LP and the SDP relaxations is based on partitioning the input uncertainty set and solving a convex relaxation on each input part separately. Throughout our theoretical development and experiments, we will show that this approach presents a valid, efficient, and effective way to tighten the relaxations of both LP and SDP certifications, and we derive principled branching strategies for both methods that minimize upper bounds on the relaxation errors. We now turn to mathematically formulating these relaxations.

2.3 LP Relaxation of the Network Constraints

We now introduce the LP relaxation. First, we remark that since $\mathcal{X}$ is bounded, the preactivations at each layer are bounded as well. That is, for every $k \in \{1, 2, \dots, K\}$, there exist preactivation bounds $l^{[k]}, u^{[k]} \in \mathbb{R}^{n_k}$ such that $l^{[k]} \leq \hat{z}^{[k]} \leq u^{[k]}$, where $\hat{z}^{[k]}$ is the k^{th} layer's preactivation in (1), for all $x \in \mathcal{X}$. We assume without loss of generality that $l^{[k]} \leq 0 \leq u^{[k]}$ for all k, since, if $l_i^{[k]} > 0$ for some i, the preactivation bound $l_i^{[k]} \leq \hat{z}_i^{[k]}$ can be replaced by $0 \leq \hat{z}_i^{[k]}$, and similarly for the case where $u_i^{[k]} < 0$. Although there exist various methods in the literature for efficiently approximating these preactivation bounds, we consider the bounds to be tight, i.e., $\hat{z}^{[k]} = l^{[k]}$ for some $x \in \mathcal{X}$, and similarly for the upper bound $u^{[k]}$. Tjeng et al. (2019) provides efficient methods for computing these bounds. Now, following the approach initially introduced in Wong and Kolter (2018), we can relax the k^{th} ReLU constraint in (1) to its convex upper envelope between the preactivation bounds. This is graphically shown in Figure 3.

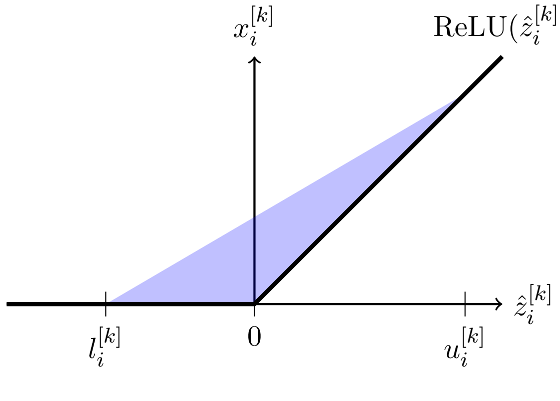


Figure 3: Relaxed ReLU constraint set $\mathcal{N}_{\text{LP}}^{[k]}$ at a single neuron i in layer k of the network.

We call the convex upper envelope associated with layer k the *relaxed ReLU constraint set*, and its mathematical definition is given by three linear inequalities:

$$\begin{aligned}\mathcal{N}_{\text{LP}}^{[k]} = \{(x^{[k-1]}, x^{[k]}) \in \mathbb{R}^{n_{k-1}} \times \mathbb{R}^{n_k} : x^{[k]} \leq u^{[k]} \odot (\hat{z}^{[k]} - l^{[k]}) \oslash (u^{[k]} - l^{[k]}),\\ x^{[k]} \geq 0,\ x^{[k]} \geq \hat{z}^{[k]},\ \hat{z}^{[k]} = W^{[k-1]} x^{[k-1]}\}.\end{aligned} \tag{3}$$

Next, we define the *relaxed network constraint set* to be

$$\mathcal{N}_{\text{LP}} = \{(x,z) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} : (x, x^{[1]}) \in \mathcal{N}_{\text{LP}}^{[1]},\ (x^{[1]}, x^{[2]}) \in \mathcal{N}_{\text{LP}}^{[2]}, \dots, (x^{[K-1]}, z) \in \mathcal{N}_{\text{LP}}^{[K]}\}. \tag{4}$$

In essence, $\mathcal{N}_{\text{LP}}$ is the set of all input-output pairs of the network that satisfy the relaxed ReLU constraints at every layer. Note that, since the bounds $l^{[k]}$ and $u^{[k]}$ are determined by the input uncertainty set $\mathcal{X}$, the set $\mathcal{N}_{\text{LP}}^{[k]}$ is also determined by $\mathcal{X}$ for all layers k.

Remark 2 *For networks with one hidden layer (i.e., $K = 1$), the single relaxed ReLU constraint set coincides with the relaxed network constraint set: $\mathcal{N}_{\text{LP}}^{[1]} = \mathcal{N}_{\text{LP}}$. Therefore, for $K = 1$ we drop the k-notation and simply write z, $\hat{z}$, x, W, l, u, and $\mathcal{N}_{\text{LP}}$.*

Finally, we introduce the LP relaxation of (2):

$$\hat{\phi}_{\text{LP}}^*(\mathcal{X}) = \sup\{c^\top z : (x,z) \in \mathcal{N}_{\text{LP}},\ x \in \mathcal{X}\}. \tag{5}$$

Notice that, if $x \in \mathcal{X}$ and $z = f(x)$, then $(x,z) \in \mathcal{N}_{\text{LP}}$ by the definition of $\mathcal{N}_{\text{LP}}$. Furthermore, since $\mathcal{X}$ is a bounded convex polytope and $\mathcal{N}_{\text{LP}}$ is defined by a system of linear constraints, we confirm that (5) is a linear program. Therefore, (5) is indeed an LP relaxation of (2), so it holds that

$$\phi^*(\mathcal{X}) \leq \hat{\phi}_{\text{LP}}^*(\mathcal{X}). \tag{6}$$

This analytically shows what Figures 1 and 2 illustrate: the condition that $\hat{\phi}_{\text{LP}}^*(\mathcal{X}) \leq 0$ is sufficient to conclude that the network is certifiably robust, but if $\hat{\phi}_{\text{LP}}^*(\mathcal{X}) > 0$, the relaxation fails to certify whether or not the network is robust, since it may still hold that $\phi^*(\mathcal{X}) \leq 0$.

2.4 SDP Relaxation of the Network Constraints

An alternative convex relaxation of the robustness certification problem can be formulated as an SDP. This method was first introduced in Raghunathan et al. (2018). Here, we will introduce the SDP relaxation for a network with a single hidden layer for notational convenience. The extension to multi-layer networks is straightforward and presented in Raghunathan et al. (2018). In this formulation, the optimization variable $(x, z) \in \mathbb{R}^{n_x+n_z}$ is lifted to a symmetric matrix

$$P = \begin{bmatrix} 1 \\ x \\ z \end{bmatrix} \begin{bmatrix} 1 & x^\top & z^\top \end{bmatrix} \in \mathbb{S}^{n_x+n_z+1}.$$

We use the following block-indexing for P:

$$P = \begin{bmatrix} P_1 & P_x^\top & P_z^\top \\ P_x & P_{xx} & P_{xz} \\ P_z & P_{zx} & P_{zz} \end{bmatrix},$$

where $P_1 \in \mathbb{R}$, $P_x \in \mathbb{R}^{n_x}$, $P_z \in \mathbb{R}^{n_z}$, $P_{xx} \in \mathbb{S}^{n_x}$, $P_{zz} \in \mathbb{S}^{n_z}$, $P_{xz} \in \mathbb{R}^{n_x \times n_z}$, and $P_{zx} = P_{xz}^\top$. This lifting procedure results in the optimization problem

$$\begin{aligned} \underset{P \in \mathbb{S}^{n_x+n_z+1}}{\text{maximize}} \quad & c^\top P_z \\ \text{subject to} \quad & P_z \geq 0, \\ & P_z \geq W P_x, \\ & \operatorname{diag}(P_{xx}) \leq (l+u) \odot P_x - l \odot u, \\ & \operatorname{diag}(P_{zz}) = \operatorname{diag}(W P_{xz}), \\ & P_1 = 1, \\ & P \succeq 0, \\ & \operatorname{rank}(P) = 1. \end{aligned}$$

Here, there are no preactivation bounds, unlike the LP relaxation. The vectors $l, u \in \mathbb{R}^{n_x}$ are lower and upper bounds on the input, which are determined by the input uncertainty set. For example, if $\mathcal{X} = \{x \in \mathbb{R}^{n_x} : \|x - \bar{x}\|_\infty \leq \epsilon\}$, then $l = \bar{x} - \epsilon \mathbf{1}_{n_x}$ and $u = \bar{x} + \epsilon \mathbf{1}_{n_x}$.

We remark that the above problem is equivalent to the original robustness certification problem; no relaxation has been made yet. The only nonconvex constraint in this formulation is the rank-1 constraint on P. Dropping this rank constraint, we obtain the SDP relaxed network constraint set:

$$\begin{aligned} \mathcal{N}_{\text{SDP}} = \{P \in \mathbb{S}^{n_x+n_z+1} : P_z \geq 0,\ P_z \geq W P_x,\ \operatorname{diag}(P_{zz}) = \operatorname{diag}(W P_{xz}), \\ \operatorname{diag}(P_{xx}) \leq (l+u) \odot P_x - l \odot u,\ P_1 = 1,\ P \succeq 0\}. \end{aligned} \tag{7}$$

Using this relaxed network constraint set as the feasible set for the optimization, we arrive at the SDP relaxation

$$\hat{\phi}^*_{\text{SDP}}(\mathcal{X}) = \sup\{c^\top P_z : P \in \mathcal{N}_{\text{SDP}},\ P_x \in \mathcal{X}\}. \tag{8}$$

It is clear that by dropping the rank constraint, we have enlarged the feasible set, so again we obtain a viable relaxation of the original problem (2): $\phi^*(\mathcal{X}) \leq \hat{\phi}^*_{\mathrm{SDP}}(\mathcal{X})$. In the case the solution P^* to (8) is rank-1, we can factorize it as

$$P^* = \begin{bmatrix} 1 \\ x^* \\ z^* \end{bmatrix} \begin{bmatrix} 1 & x^{*\top} & z^{*\top} \end{bmatrix}$$

and conclude that (x^*, z^*) solves the original nonconvex problem. However, it is generally the case that the SDP solution will be of higher rank, leading to relaxation error and the possibility of a void robustness certificate, similar to the LP relaxation. We now turn to building upon the LP and SDP convex relaxations via input partitioning in order to tighten their relaxations.

3. Partitioned LP Relaxation

3.1 Properties of Partitioned Relaxation

In this section, we investigate the properties and effectiveness of partitioning the input uncertainty set when solving the LP relaxation for robustness certification. We start by validating the approach, namely, by showing that solving the LP relaxation separately on each input part maintains a theoretically guaranteed upper bound on the optimal value of the unrelaxed problem (2). Afterwards, the approach is proven to yield a tighter upper bound than solving the LP relaxation without partitioning.

3.1.1 Partitioning Gives Valid Relaxation

Proposition 3 (Partitioned relaxation bound) *Let $\{\mathcal{X}^{(j)} \subseteq \mathcal{X} : j \in \{1, 2, \dots, p\}\}$ be a partition of $\mathcal{X}$. Then, it holds that*

$$\phi^*(\mathcal{X}) \leq \max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}). \tag{9}$$

Proof See Appendix A. ■

Despite the fact that Proposition 3 asserts an intuitively expected bound, we remark the importance for its formal statement and proof. In particular, the inequality (9) serves as the fundamental reason for why the partitioned LP relaxation can be used to certify that all inputs in $\mathcal{X}$ map to safe outputs in the safe set $\mathcal{S}$. Knowing that the partitioning approach is valid for robustness certification, we move on to studying the effectiveness of partitioning.

3.1.2 Tightening of the Relaxation

We now show that the bound (6) can always be tightened by partitioning the input uncertainty set. The result is given for networks with one hidden layer for simplicity. However, the conclusion naturally generalizes to multi-layer ReLU networks.

Proposition 4 (Improving the LP relaxation bound) *Consider a feedforward ReLU neural network with one hidden layer. Let $\{\mathcal{X}^{(j)} \subseteq \mathcal{X} : j \in \{1, 2, \dots, p\}\}$ be a partition of*

$\mathcal{X}$. For the jth input part $\mathcal{X}^{(j)}$, denote the corresponding preactivation bounds by $l^{(j)}$ and $u^{(j)}$, where $l \leq l^{(j)} \leq Wx \leq u^{(j)} \leq u$ for all $x \in \mathcal{X}^{(j)}$. Then, it holds that

$$\max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) \leq \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}). \tag{10}$$

Proof See Appendix B. ■

Propositions 3 and 4 together show that $\phi^*(\mathcal{X}) \leq \max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) \leq \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X})$, i.e., that the partitioned LP relaxation is theoretically guaranteed to perform at least as well as the unpartitioned LP when solving the robustness certification problem. The improvement in the partitioned LP relaxation is captured by the difference

$$\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}) - \max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}),$$

which is always nonnegative. We remark that it is possible for the improvement to be null in the sense that $\max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) = \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X})$. This may occur when the partition used is poorly chosen. An example of such a poor choice may be if one were to partition along a direction in the input space that, informally speaking, corresponds to directions near-orthogonal to the cost vector c in the output space. In this case, one would expect all improvements to be nullified, and for the partitioned relaxation to give the same optimal value as the unpartitioned relaxation. Consequently, the following important question arises: *what constitutes a good partition so that the improvement $\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}) - \max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)})$ is strictly greater than zero and maximal?* We address this question in Sections 3.2 and 3.3.

3.2 Motivating Partition

In this section, we begin to answer our earlier inquiry, namely, how to choose a partition in order to maximize the improvement $\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}) - \max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)})$ in the partitioned LP relaxation. Recall that this is equivalent to minimizing the relaxation error relative to the original unrelaxed problem, since $\phi^*(\mathcal{X}) \leq \max_{j \in \{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) \leq \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X})$. To this end, we construct a partition with finitely many parts, based on the parameters of the network, which is shown to exactly recover the optimal value of the original unrelaxed problem (2). For simplicity, we present the result for a single hidden layer, but the basic idea of partitioning at the "kinks" of the ReLUs in order to collapse the ReLU upper envelope onto the ReLU curve and eliminate relaxation error can be generalized to multi-layer settings. At this point, let us remark that in Proposition 5 below, we use a slight difference in notation for the partition. Namely, we use the set of all n_z-vectors with binary elements, $\mathcal{J} \coloneqq \{0,1\}^{n_z} = \{0,1\} \times \{0,1\} \times \cdots \times \{0,1\}$, to index the parts of the partition. Under this setting, the partition is composed of $p = 2^{n_z}$ parts, so that $\mathcal{X}^{(j)}$ is the part of the partition corresponding to the binary vector j, which is an element of the index set $\mathcal{J}$. This temporary change in notation is chosen to simplify the proof of Proposition 5.

Proposition 5 (Motivating partition) *Consider a feedforward ReLU neural network with one hidden layer and denote the ith row of W by $w_i^\top \in \mathbb{R}^{1 \times n_x}$ for all $i \in \{1,2,\dots,n_z\}$. Define $\mathcal{J} = \{0,1\}^{n_z}$ and take the partition of $\mathcal{X}$ to be indexed by $\mathcal{J}$, meaning that $\{\mathcal{X}^{(j)} \subseteq$*

$\mathcal{X} : j \in \mathcal{J}\}$, where for a given $j \in \mathcal{J}$ we define

$$\mathcal{X}^{(j)} = \{x \in \mathcal{X} : w_i^\top x \geq 0 \text{ for all } i \text{ such that } j_i = 1,\ w_i^\top x < 0 \text{ for all } i \text{ such that } j_i = 0\}. \tag{11}$$

Then, the partitioned relaxation is exact, i.e.,

$$\phi^*(\mathcal{X}) = \max_{j \in \mathcal{J}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}). \tag{12}$$

Proof We first show that $\{\mathcal{X}^{(j)} \subseteq \mathcal{X} : j \in \mathcal{J}\}$ is a valid partition. Since $\mathcal{X}^{(j)} \subseteq \mathcal{X}$ for all $j \in \mathcal{J}$, the relation $\cup_{j \in \mathcal{J}} \mathcal{X}^{(j)} \subseteq \mathcal{X}$ is satisfied. Now, suppose that $x \in \mathcal{X}$. Then, for all $i \in \{1, 2, \dots, n_z\}$, either $w_i^\top x \geq 0$ or $w_i^\top x < 0$ holds. Define $j \in \{0,1\}^{n_z}$ as follows:

$$j_i = \begin{cases} 1 & \text{if } w_i^\top x \geq 0, \\ 0 & \text{if } w_i^\top x < 0, \end{cases}$$

for all $i \in \{1, 2, \dots, n_z\}$. Then, by the definition of $\mathcal{X}^{(j)}$ in (11), it holds that $x \in \mathcal{X}^{(j)}$. Therefore, the relation $x \in \mathcal{X}$ implies that $x \in \mathcal{X}^{(j)}$ for some $j \in \{0,1\}^{n_z} = \mathcal{J}$. Hence, $\mathcal{X} \subseteq \cup_{j \in \mathcal{J}} \mathcal{X}^{(j)}$, and therefore $\cup_{j \in \mathcal{J}} \mathcal{X}^{(j)} = \mathcal{X}$.

We now show that $\mathcal{X}^{(j)} \cap \mathcal{X}^{(k)} = \emptyset$ for all $j \neq k$. Let $j, k \in \mathcal{J}$ with the property that $j \neq k$. Then, there exists $i \in \{1, 2, \dots, n_z\}$ such that $j_i \neq k_i$. Let $x \in \mathcal{X}^{(j)}$. In the case that $w_i^\top x \geq 0$, it holds that $j_i = 1$ and therefore $k_i = 0$. Hence, for all $y \in \mathcal{X}^{(k)}$, it holds that $w_i^\top y < 0$, and therefore $x \notin \mathcal{X}^{(k)}$. An analogous reasoning shows that $x \notin \mathcal{X}^{(k)}$ when $w_i^\top x < 0$. Therefore, one concludes that $x \in \mathcal{X}^{(j)}$ and $j \neq k$ implies that $x \notin \mathcal{X}^{(k)}$, i.e., that $\mathcal{X}^{(j)} \cap \mathcal{X}^{(k)} = \emptyset$. Hence, $\{\mathcal{X}^{(j)} \subseteq \mathcal{X} : j \in \mathcal{J}\}$ is a valid partition.

We now prove (12). Let $j \in \mathcal{J}$. Since $w_i^\top x \geq 0$ for all i such that $j_i = 1$, the preactivation lower bound becomes $l_i^{(j)} = 0$ for all such i. On the other hand, since $w_i^\top x < 0$ for all i such that $j_i = 0$, the preactivation upper bound becomes $u_i^{(j)} = 0$ for all such i. Therefore, the relaxed network constraint set (4) for the j^{th} input part reduces to

$$\begin{aligned} \mathcal{N}_{\mathrm{LP}}^{(j)} = \{(x,z) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} : z_i = 0 \text{ for all } i \text{ such that } j_i = 0, \\ z_i = w_i^\top x = (Wx)_i \text{ for all } i \text{ such that } j_i = 1\}. \end{aligned}$$

That is, the relaxed ReLU constraint envelope collapses to the exact ReLU constraint through the prior knowledge of each preactivation coordinate's sign. Therefore, we find that for all $x \in \mathcal{X}^{(j)}$ it holds that $(x,z) \in \mathcal{N}_{\mathrm{LP}}^{(j)}$ if and only if $z = \mathrm{ReLU}(Wx)$. Hence, the LP over the j^{th} input part yields that

$$\begin{aligned} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) &= \sup\{c^\top z : (x,z) \in \mathcal{N}_{\mathrm{LP}}^{(j)},\ x \in \mathcal{X}^{(j)}\} \\ &= \sup\{c^\top z : z = \mathrm{ReLU}(Wx),\ x \in \mathcal{X}^{(j)}\} \\ &\leq \sup\{c^\top z : z = \mathrm{ReLU}(Wx),\ x \in \mathcal{X}\} \\ &= \phi^*(\mathcal{X}). \end{aligned}$$

Since j was chosen arbitrarily, it holds that

$$\max_{j \in \mathcal{J}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) \leq \phi^*(\mathcal{X}).$$

Since $\phi^*(\mathcal{X}) \leq \max_{j\in\mathcal{J}} \hat{\phi}^*_{\text{LP}}(\mathcal{X}^{(j)})$ by the relaxation bound (9), the equality (12) holds, as desired. ■

Although the partition proposed in Proposition 5 completely eliminates relaxation error of the LP, using it in practice may be computationally intractable, as it requires solving 2^{n_z} separate linear programs. Despite this limitation, the result provides two major theoretical implications. First, our input partitioning approach is fundamentally shown to be a simple, yet very powerful method, as the robustness certification problem can be solved exactly via a finite number of linear program subproblems. Second, the partition proposed in Proposition 5 shows us the structure of an optimal partition, namely that the parts of the partition are defined by the half-spaces generated by the rows of W (see Figure 4). This result paves the way to develop a tractable branching scheme that incorporates the reduction in relaxation error endowed by the structure of this motivating partition. In the next section, we explore this idea further, and seek to answer the following question: *if we only partition along a single row of the weight matrix, which one is the best to choose?*

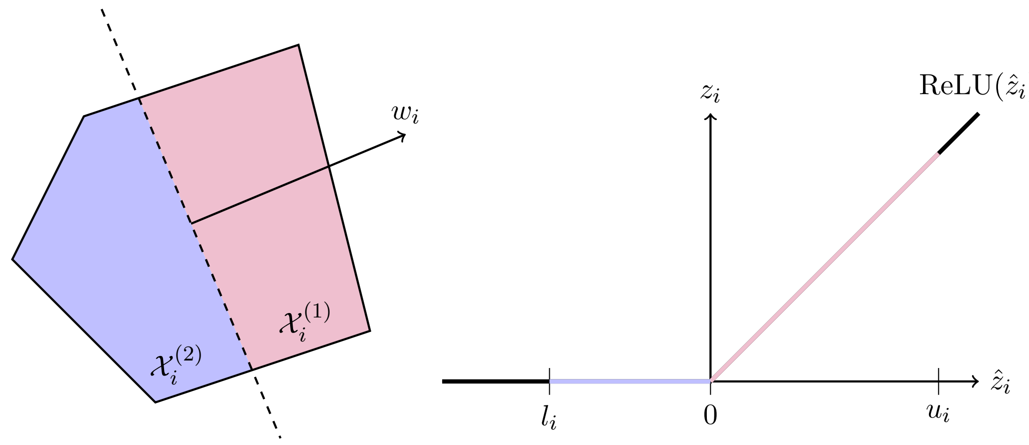


Figure 4: Partitioning based on row $w_i^\top$ of the weight matrix. This partition results in an exact ReLU constraint in coordinate i over the two resulting input parts $\mathcal{X}_i^{(1)} = \{x \in \mathcal{X} : w_i^\top x \geq 0\}$ and $\mathcal{X}_i^{(2)} = \mathcal{X} \setminus \mathcal{X}_i^{(1)}$.

3.3 LP Branching Scheme

In this section, we propose an explicit, computationally tractable, and effective LP branching scheme. The branching scheme is developed based on analyses for a single hidden layer. However, the resulting branching scheme is still applicable to multi-layer networks, and indeed will be shown to remain effective on two-layer networks in the experiments of Section 6. The development of the partition boils down to two main ideas. First, we restrict our attention to two-part partitions defined by rows of the weight matrix W, specifically, $\mathcal{X}_i^{(1)} = \{x \in \mathcal{X} : w_i^\top x \geq 0\}$ and $\mathcal{X}_i^{(2)} = \mathcal{X} \setminus \mathcal{X}_i^{(1)}$, as motivated in the previous section. Second, we seek which index $i \in \{1, 2, \dots, n_z\}$ gives the best partition, in the sense that the relaxation error of the resulting partitioned LP is minimized. As will be shown in Section 3.3.3, this second aspect is NP-hard to discern in general. Therefore, to find the optimal row to partition along, we instead seek to minimize a more tractable upper bound on the relaxation error. Since the actual relaxation error cannot exceed this upper bound even in the worst case, we refer to such a bound-minimizing partition as "worst-case optimal."

3.3.1 Worst-Case Relaxation Bound

We begin by bounding the relaxation error below.

Theorem 6 (Worst-case relaxation bound) *Consider a feedforward ReLU neural network with one hidden layer, with the input uncertainty set $\mathcal{X}$ and preactivation bounds $l, u \in \mathbb{R}^{n_z}$. Consider also the relaxation error $\Delta\phi^*(\mathcal{X}) \coloneqq \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}) - \phi^*(\mathcal{X})$. Let $(\tilde{x}^*, \tilde{z}^*)$ and (x^*, z^*) be optimal solutions for the relaxation $\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X})$ and the unrelaxed problem $\phi^*(\mathcal{X})$, respectively. Given an arbitrary norm $\|\cdot\|$ on $\mathbb{R}^{n_x}$, it holds that*

$$\Delta\phi^*(\mathcal{X}) \leq \sum_{i=1}^{n_z} \Bigg(\mathrm{ReLU}(c_i) \frac{u_i}{u_i - l_i} (\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} - l_i) + \mathrm{ReLU}(-c_i) \min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \Bigg), \tag{13}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$.

Proof First, recall that $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is assumed to be compact, and is therefore bounded, and hence $d_{\|\cdot\|}(\mathcal{X}) < \infty$. The definitions of $(\tilde{x}^*, \tilde{z}^*)$ and (x^*, z^*) give that

$$\Delta\phi^*(\mathcal{X}) = \sum_{i=1}^{n_z} c_i(\tilde{z}^*_i - z^*_i) \leq \sum_{i=1}^{n_z} \Delta\phi^*_i, \tag{14}$$

where

$$\Delta\phi^*_i = \sup\Bigg\{ c_i(\tilde{z}_i - z_i) : z_i = \mathrm{ReLU}(w_i^\top x),\ \tilde{z}_i \geq 0,\ \tilde{z}_i \geq w_i^\top \tilde{x}, \tilde{z}_i \leq \frac{u_i}{u_i - l_i}(w_i^\top \tilde{x} - l_i),\ x, \tilde{x} \in \mathcal{X} \Bigg\}$$

for all $i \in \{1, 2, \dots, n_z\}$. Note that

$$\Delta\phi^*_i = \sup\Bigg\{ c_i(\tilde{z}_i - z_i) : z_i = \mathrm{ReLU}(\hat{z}_i),\ \tilde{z}_i \geq 0,\ \tilde{z}_i \geq \hat{\tilde{z}}_i,\ \tilde{z}_i \leq \frac{u_i}{u_i - l_i}(\hat{\tilde{z}}_i - l_i), \hat{z} = Wx,\ \hat{\tilde{z}} = W\tilde{x},\ x, \tilde{x} \in \mathcal{X} \Bigg\}.$$

If $x, \tilde{x} \in \mathcal{X}$ and $\hat{z}, \hat{\tilde{z}}$ satisfy $\hat{z} = Wx$, $\hat{\tilde{z}} = W\tilde{x}$, then they satisfy $l \leq \hat{z}, \hat{\tilde{z}} \leq u$ and $|\hat{\tilde{z}}_k - \hat{z}_k| = |w_k^\top(\tilde{x} - x)| \leq \|w_k\|_* \|\tilde{x} - x\| \leq \|w_k\|_* d_{\|\cdot\|}(\mathcal{X})$ for all $k \in \{1, 2, \dots, n_z\}$ by the Cauchy-Schwarz inequality for dual norms. Therefore,

$$\begin{aligned}\Delta\phi_i^* \leq \sup\Big\{&c_i(\tilde{z}_i - z_i) : z_i = \mathrm{ReLU}(\hat{z}_i),\ \tilde{z}_i \geq 0,\ \tilde{z}_i \geq \hat{\tilde{z}}_i,\ \tilde{z}_i \leq \frac{u_i}{u_i - l_i}(\hat{\tilde{z}}_i - l_i),\\ &l \leq \hat{z}, \hat{\tilde{z}} \leq u,\ |\hat{\tilde{z}}_k - \hat{z}_k| \leq \|w_k\|_* d_{\|\cdot\|}(\mathcal{X}) \text{ for all } k \in \{1,2,\dots,n_z\},\ \hat{z}, \hat{\tilde{z}} \in \mathbb{R}^{n_z}\Big\}\\ = \sup\Big\{&c_i(\tilde{z}_i - z_i) : z_i = \mathrm{ReLU}(\hat{z}_i),\ \tilde{z}_i \geq 0,\ \tilde{z}_i \geq \hat{\tilde{z}}_i,\ \tilde{z}_i \leq \frac{u_i}{u_i - l_i}(\hat{\tilde{z}}_i - l_i),\\ &l_i \leq \hat{z}_i, \hat{\tilde{z}}_i \leq u_i,\ |\hat{\tilde{z}}_i - \hat{z}_i| \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X}),\ \hat{z}_i, \hat{\tilde{z}}_i \in \mathbb{R}\Big\}.\end{aligned}$$

For $c_i \geq 0$, the above inequality yields that

$$\begin{aligned}\Delta\phi_i^* \leq c_i \sup\Big\{&\tilde{z}_i - z_i : z_i = \mathrm{ReLU}(\hat{z}_i),\ \tilde{z}_i \geq 0,\ \tilde{z}_i \geq \hat{\tilde{z}}_i,\ \tilde{z}_i \leq \frac{u_i}{u_i - l_i}(\hat{\tilde{z}}_i - l_i),\\ &l_i \leq \hat{z}_i, \hat{\tilde{z}}_i \leq u_i,\ |\hat{\tilde{z}}_i - \hat{z}_i| \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X}),\ \hat{z}_i, \hat{\tilde{z}}_i \in \mathbb{R}\Big\}.\end{aligned}$$

The optimal solution to the above supremum is readily found by comparing the line $\tilde{z}_i = \frac{u_i}{u_i-l_i}(\hat{\tilde{z}}_i - l_i)$ to the function $z_i = \mathrm{ReLU}(\hat{z}_i)$ over $\hat{\tilde{z}}_i, \hat{z}_i \in [l_i, u_i]$. In particular, the maximum distance between $\tilde{z}_i$ and z_i on the above feasible set occurs when $z_i = \hat{z}_i = 0$, $\hat{\tilde{z}}_i = \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$, and $\tilde{z}_i = \frac{u_i}{u_i-l_i}(\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}) - l_i)$. Therefore, we find that

$$\Delta\phi_i^* \leq c_i \frac{u_i}{u_i - l_i}(\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}) - l_i), \tag{15}$$

for all $i \in \{1,2,\dots,n_z\}$ such that $c_i \geq 0$. We also note the trivial bound that $\tilde{z}_i - z_i \leq u_i$ on the feasible set of the above supremum, so that

$$\Delta\phi_i^* \leq c_i u_i = c_i \frac{u_i}{u_i - l_i}(u_i - l_i). \tag{16}$$

The inequalities (15) and (16) together imply that

$$\Delta\phi_i^* \leq c_i \frac{u_i}{u_i - l_i}(\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} - l_i) \tag{17}$$

for all $i \in \{1,2,\dots,n_z\}$ such that $c_i \geq 0$.

On the other hand, for all $i \in \{1,2,\dots,n_z\}$ such that $c_i < 0$, we have that

$$\begin{aligned}\Delta\phi_i^* \leq c_i \inf\Big\{&\tilde{z}_i - z_i : z_i = \mathrm{ReLU}(\hat{z}_i),\ \tilde{z}_i \geq 0,\ \tilde{z}_i \geq \hat{\tilde{z}}_i,\ \tilde{z}_i \leq \frac{u_i}{u_i - l_i}(\hat{\tilde{z}}_i - l_i),\\ &l_i \leq \hat{z}_i, \hat{\tilde{z}}_i \leq u_i,\ |\hat{\tilde{z}}_i - \hat{z}_i| \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X}),\ \hat{z}_i, \hat{\tilde{z}}_i \in \mathbb{R}\Big\}.\end{aligned}$$

The optimal solution to the above infimum is readily found by comparing the line $\tilde{z}_i = 0$ to the function $z_i = \mathrm{ReLU}(\hat{z}_i)$ over $\hat{\tilde{z}}_i, \hat{z}_i \in [l_i, u_i]$. In particular, the minimum value of $\tilde{z}_i - z_i$ on the above feasible set occurs when $\tilde{z}_i = \hat{\tilde{z}}_i = 0$ and $z_i = \hat{z}_i = \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$. Therefore, we find that

$$\Delta\phi_i^* \leq -c_i \|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), \tag{18}$$

for all $i \in \{1, 2, \dots, n_z\}$ such that $c_i < 0$. We also note the trivial bound that $\tilde{z}_i - z_i \geq -u_i$ on the feasible set of the above infimum, so that

$$\Delta\phi_i^* \leq -c_i u_i. \tag{19}$$

The inequalities (18) and (19) together imply that

$$\Delta\phi_i^* \leq -c_i \min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \tag{20}$$

for all $i \in \{1, 2, \dots, n_z\}$ such that $c_i < 0$. Substituting (17) and (20) into (14) gives the desired bound (13). ■

The value $\Delta\phi_i^*$ in the proof of Theorem 6 can be interpreted as the worst-case relaxation error in coordinate i. From this perspective, Theorem 6 gives an upper bound on the worst-case relaxation error of the overall network. Notice that, since $\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \leq u_i$, the bound (13) immediately gives rise to the simple bound that

$$\Delta\phi^*(\mathcal{X}) \leq \sum_{i=1}^{n_z} \left(\mathrm{ReLU}(c_i) + \mathrm{ReLU}(-c_i)\right) u_i = \sum_{i=1}^{n_z} |c_i| u_i,$$

the right-hand side of which equals the relaxation error incurred when, at every neuron, the activations of the relaxation solution and the original nonconvex solution are at opposite corners of the relaxed ReLU constraint set, i.e., the convex upper envelope illustrated in Figure 3. On the other hand, when $d_{\|\cdot\|}(\mathcal{X})$ is small, i.e., the input uncertainty set is small, one would expect the number of "kinks" in the graph of f over $\mathcal{X}$ to be small, and as a consequence the relaxation error to decrease. This intuition is captured by the bound (13), since, in this case, we find that

$$\begin{aligned}
\Delta\phi^*(\mathcal{X}) &\leq \sum_{i=1}^{n_z} \left(\mathrm{ReLU}(c_i) \frac{u_i}{u_i - l_i} (\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}) - l_i) + \mathrm{ReLU}(-c_i) \|w_i\|_* d_{\|\cdot\|}(\mathcal{X}) \right) \\
&\approx -\sum_{i=1}^{n_z} \mathrm{ReLU}(c_i) \frac{u_i l_i}{u_i - l_i} \\
&\leq \sum_{i=1}^{n_z} |c_i| u_i.
\end{aligned}$$

To continue our development of a branching scheme that is optimal with respect to the worst-case relaxation error, we use (13) to bound the relaxation error of the partitioned LP in terms of the row $w_i^\top$ that is used to define the partition. This bound is given in the following lemma.

Lemma 7 (Two-part bound) *Let $i \in \{1, 2, \dots, n_z\}$ and consider a two-part partition of $\mathcal{X}$ given by $\{\mathcal{X}_i^{(1)}, \mathcal{X}_i^{(2)}\}$, where $\mathcal{X}_i^{(1)} = \{x \in \mathcal{X} : w_i^\top x \geq 0\}$ and $\mathcal{X}_i^{(2)} = \mathcal{X} \setminus \mathcal{X}_i^{(1)}$. Consider also the partitioned relaxation error $\Delta\phi^*(\{\mathcal{X}_i^{(1)}, \mathcal{X}_i^{(2)}\}) \coloneqq \max_{j \in \{1,2\}} \hat{\phi}_{\mathrm{LP}}^*(\mathcal{X}_i^{(j)}) - \phi^*(\mathcal{X})$. It* *holds that*

$$\begin{aligned}\Delta\phi^*(\{\mathcal{X}_i^{(1)},\mathcal{X}_i^{(2)}\}) \le |c_i|\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \\ + \sum_{\substack{k=1\\k\neq i}}^{n_z}\Bigg(\mathrm{ReLU}(c_k)\frac{u_k}{u_k-l_k}(\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}-l_k) \\ + \mathrm{ReLU}(-c_k)\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}\Bigg).\end{aligned} \tag{21}$$

Proof Consider the relaxation solved over the first input part, $\mathcal{X}_i^{(1)}$, and denote by $l^{(1)}, u^{(1)} \in \mathbb{R}^{n_z}$ the corresponding preactivation bounds. Since $w_i^\top x \ge 0$ on this input part, the preactivation bounds for the first subproblem $\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}_i^{(1)})$ can be taken as

$$l^{(1)} = (l_1, l_2, \dots, l_{i-1}, 0, l_{i+1}, \dots, l_{n_z})$$

and $u^{(1)} = u$. Thus, from (13) and the fact that $d_{\|\cdot\|}(\mathcal{X}_i^{(j)}) \le d_{\|\cdot\|}(\mathcal{X})$ for $j \in \{1,2\}$, it follows that

$$\begin{aligned}\Delta\phi^*(\mathcal{X}_i^{(1)}) \le |c_i|\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \\ + \sum_{\substack{k=1\\k\neq i}}^{n_z}\Bigg(\mathrm{ReLU}(c_k)\frac{u_k}{u_k-l_k}(\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}-l_k) \\ + \mathrm{ReLU}(-c_k)\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}\Bigg).\end{aligned} \tag{22}$$

Similarly, over the second input part, $\mathcal{X}_i^{(2)}$, we have that $w_i^\top x < 0$, and so the preactivation bounds for the second subproblem $\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}_i^{(2)})$ can be taken as $l^{(2)} = l$ and

$$u^{(2)} = (u_1, u_2, \dots, u_{i-1}, 0, u_{i+1}, \dots, u_{n_z}),$$

resulting in a similar bound to (22):

$$\begin{aligned}\Delta\phi^*(\mathcal{X}_i^{(2)}) \le \sum_{\substack{k=1\\k\neq i}}^{n_z}\Bigg(\mathrm{ReLU}(c_k)\frac{u_k}{u_k-l_k}(\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}-l_k) \\ + \mathrm{ReLU}(-c_k)\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}\Bigg).\end{aligned} \tag{23}$$

Putting the two bounds (22) and (23) together and using the fact that $\phi^*(\mathcal{X}_i^{(j)}) \leq \phi^*(\mathcal{X})$ for all $j \in \{1,2\}$, we find that

$$\begin{aligned}\Delta\phi^*(\{\mathcal{X}_i^{(1)}, \mathcal{X}_i^{(2)}\}) &= \max_{j\in\{1,2\}} \left(\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}_i^{(j)}) - \phi^*(\mathcal{X})\right) \\ &\leq \max_{j\in\{1,2\}} \left(\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}_i^{(j)}) - \phi^*(\mathcal{X}_i^{(j)})\right) \\ &= \max_{j\in\{1,2\}} \Delta\phi^*(\mathcal{X}_i^{(j)}) \\ &\leq |c_i| \min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \\ &\qquad + \sum_{\substack{k=1\\k\neq i}}^{n_z} \Bigg(\mathrm{ReLU}(c_k)\frac{u_k}{u_k - l_k}(\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\} - l_k) \\ &\qquad\qquad + \mathrm{ReLU}(-c_k)\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}\Bigg),\end{aligned}$$

as desired. ∎

3.3.2 Proposed Branching Scheme

Lemma 7 bounds the worst-case relaxation error for each possible row-based partition. Therefore, our final step in the development of our branching scheme is to find which row minimizes the upper bound (21). This worst-case optimal branching scheme is now presented.

Theorem 8 (Worst-case optimal LP branching) *Consider the two-part partitions defined by the rows of W: $\{\mathcal{X}_i^{(1)}, \mathcal{X}_i^{(2)}\}$, where $\mathcal{X}_i^{(1)} = \{x \in \mathcal{X} : w_i^\top x \geq 0\}$ and $\mathcal{X}_i^{(2)} = \mathcal{X} \setminus \mathcal{X}_i^{(1)}$ for all $i \in \{1,2,\dots,n_z\} =: \mathcal{I}$. The optimal partition that minimizes the worst-case relaxation error bound in (21) is given by*

$$i^* \in \arg\min_{i\in\mathcal{I}} \mathrm{ReLU}(c_i)\frac{l_i}{u_i - l_i}\left(u_i - \min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\}\right). \tag{24}$$

Proof Denote the bound in (21) of Lemma 7 by

$$\begin{aligned}B(i) := |c_i| &\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \\ &+ \sum_{\substack{k=1\\k\neq i}}^{n_z} \Bigg(\mathrm{ReLU}(c_k)\frac{u_k}{u_k - l_k}(\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\} - l_k) \\ &\qquad + \mathrm{ReLU}(-c_k)\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}\Bigg),\end{aligned}$$

which is the quantity to be minimized over $i \in \mathcal{I}$. We may rewrite this as

$$\begin{aligned}
B(i) = \sum_{k=1}^{n_z} \Bigg(&\mathrm{ReLU}(c_k)\frac{u_k}{u_k - l_k}(\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\} - l_k) \\
&+ \mathrm{ReLU}(-c_k)\min\{\|w_k\|_* d_{\|\cdot\|}(\mathcal{X}), u_k\}\Bigg) \\
&+ |c_i|\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \\
&- \Bigg(\mathrm{ReLU}(c_i)\frac{u_i}{u_i - l_i}(\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} - l_i) \\
&+ \mathrm{ReLU}(-c_i)\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\}\Bigg).
\end{aligned}$$

Hence, using the fact that $|c_i| = \mathrm{ReLU}(c_i) + \mathrm{ReLU}(-c_i)$, we find that

$$\begin{aligned}
\min_{i\in\mathcal{I}} B(i) &= \min_{i\in\mathcal{I}} \Bigg(|c_i|\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} \\
&\quad - \Bigg(\mathrm{ReLU}(c_i)\frac{u_i}{u_i - l_i}(\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\} - l_i) \\
&\quad + \mathrm{ReLU}(-c_i)\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\}\Bigg)\Bigg) \\
&= \min_{i\in\mathcal{I}} \Bigg(\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\}\Bigg(|c_i| - \mathrm{ReLU}(c_i)\frac{u_i}{u_i - l_i} - \mathrm{ReLU}(-c_i)\Bigg) \\
&\quad + \mathrm{ReLU}(c_i)\frac{u_i l_i}{u_i - l_i}\Bigg) \\
&= \min_{i\in\mathcal{I}} \Bigg(\min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\}\Bigg(\mathrm{ReLU}(c_i) - \mathrm{ReLU}(c_i)\frac{u_i}{u_i - l_i}\Bigg) \\
&\quad + \mathrm{ReLU}(c_i)\frac{u_i l_i}{u_i - l_i}\Bigg) \\
&= \min_{i\in\mathcal{I}} \mathrm{ReLU}(c_i)\frac{l_i}{u_i - l_i}\left(u_i - \min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\}\right),
\end{aligned}$$

as desired. ■

Theorem 8 provides the branching scheme that optimally reduces the worst-case relaxation error that we seek. We remark its simplicity: to decide which row to partition along, it suffices to enumerate the values $\mathrm{ReLU}(c_i)\frac{l_i}{u_i-l_i}(u_i - \min\{\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}), u_i\})$ for $i \in \{1, 2, \ldots, n_z\}$, then choose the row corresponding to the minimum amongst these values. In practice (especially when using $\|\cdot\| = \|\cdot\|_\infty$), $d_{\|\cdot\|}(\mathcal{X})$ tends to be relatively small, making these values approximately equal to $\mathrm{ReLU}(c_i)\frac{u_i l_i}{u_i - l_i}$. In our experiments that follow, we find that using these simplified values to select the partition does not degrade performance. Note that this optimization over i scales linearly with the dimension n_z, and the resulting LP subproblems on each input part only require the addition of one extra linear constraint, meaning that this partitioning scheme is highly efficient.

We emphasize that our relaxation error bounds in Theorem 6 and Lemma 7, as well as our resulting worst-case optimal LP branching scheme of Theorem 8, are proven specifically for the case of single-hidden layer networks. Deriving worst-case optimal LP branching schemes for deep networks remains a challenging open theoretical problem. However, one may easily extend our theoretically principled single-layer LP branching scheme to a multi-layer branching heuristic as follows. First, consider a layer $k \in \{1, \dots, K\}$. We generate a surrogate "c"-vector of size $n_k \times 1$ so that the branching "scores" in (24) can be computed using this surrogate cost vector. To do this, treat the activation vector $x^{[k]}$ as the output and determine which coordinate $i_k \in \mathbb{R}^{n_k}$ of the nominal activation $\bar{x}^{[k]}$ is maximal. This means that i_k would be the class assigned to $\bar{x}$ if the output were after the k^{th} hidden layer. Then, we find the second-best coordinate $j_k \neq i_k$ so that $\bar{x}^{[k]}_{i_k} \geq \bar{x}^{[k]}_{j_k} \geq \bar{x}^{[k]}_{i}$ for all other i. Afterwards, the surrogate cost vector can be taken as $c^{[k]} := e_{j_k} - e_{i_k}$, meaning that it serves as a measure of whether the classification after the k^{th} hidden layer changes from i_k to j_k. Then, the branching score $s^{[k]}_i := \mathrm{ReLU}(c^{[k]}_i)\frac{u^{[k]}_i l^{[k]}_i}{u^{[k]}_i - l^{[k]}_i}$ from Theorem 8 corresponding to the i^{th} neuron in layer k can be computed. Of course, the surrogate vector $c^{[k]}$ is only used to compute the branching scores, and the full network and original cost vector c are used in the resultant branched LP. This multi-layer heuristic is summarized in Algorithm 1.

Algorithm 1 Heuristic extension of LP branching to multi-layer networks

```
Input: $\bar{x}$, $K$, $l^{[1]}, \dots, l^{[K]}$, $u^{[1]}, \dots, u^{[K]}$
1: for $k = 1, \dots, K$
2:     compute nominal activation vector $\bar{x}^{[k]}$
3:     compute nominal class $i_k \in \arg\max_{i \in \{1,\dots,n_k\}} \bar{x}^{[k]}_i$
4:     compute runner-up class $j_k \in \arg\max_{j \in \{1,\dots,n_k\} \setminus \{i_k\}} \bar{x}^{[k]}_j$
5:     assign surrogate "$c$"-vector $c^{[k]} \leftarrow e_{j_k} - e_{i_k}$
6:     assign branching scores $s^{[k]}_i \leftarrow \mathrm{ReLU}(c^{[k]}_i)\frac{u^{[k]}_i l^{[k]}_i}{u^{[k]}_i - l^{[k]}_i}$, $i \in \{1, \dots, n_k\}$
7: return branching scores $s^{[k]}_i$, $i \in \{1, \dots, n_k\}$, $k \in \{1, \dots, K\}$
```

We also note that Theorem 8 can be immediately extended to design multi-part partitions in two interesting ways. First, by ordering the values $\mathrm{ReLU}(c_i)\frac{u_i l_i}{u_i - l_i}$, we are ordering the optimality of the rows $w_i^\top$ to partition along. Therefore, by partitioning along the $n_p > 1$ rows corresponding to the smallest n_p of these values, Theorem 8 provides a strategy to design an effective 2^{n_p}-part partition, in the case one prefers to perform more than just a two-part partition. Second, Theorem 8 can be used directly in a branch-and-bound algorithm. See Section 5 for implementation details. In our experiments that follow, we find that this technique works particularly well; our partition yields a branching method for branch-and-bound that outperforms the state-of-the-art per-neuron branching technique on small-to-moderately-sized single-hidden layer networks, and achieves comparable performance on large-scale deep models.

3.3.3 Optimal Partitioning is NP-Hard

In this section, we show that finding a row-based partition that minimizes the actual LP relaxation error is an NP-hard problem. Recall that this approach is in contrast to our previous approach in the sense that our optimal partition in Theorem 8 minimizes the worst-case relaxation error. Consequently, the results of this section show that the partition given by Theorem 8 is in essence an optimal *tractable* LP partitioning scheme.

To start, recall the robustness certification problem for a K-layer ReLU neural network:

$$\begin{aligned} \text{maximize} \quad & c^\top x^{[K]} \\ \text{subject to} \quad & x^{[0]} \in \mathcal{X}, \\ & x^{[k+1]} = \operatorname{ReLU}(W^{[k]} x^{[k]}), \quad k \in \{0,1,\dots,K-1\}, \end{aligned} \tag{25}$$

where the optimal value of (25) is denoted by $\phi^*(\mathcal{X})$. Moreover, recall the LP relaxation of (25):

$$\begin{aligned} \text{maximize} \quad & c^\top x^{[K]} \\ \text{subject to} \quad & x^{[0]} \in \mathcal{X}, \\ & x^{[k+1]} \geq W^{[k]} x^{[k]}, && k \in \{0,1,\dots,K-1\}, \\ & x^{[k+1]} \geq 0, && k \in \{0,1,\dots,K-1\}, \\ & x^{[k+1]} \leq u^{[k+1]} \odot (W^{[k]} x^{[k]} - l^{[k+1]}) \oslash (u^{[k+1]} - l^{[k+1]}), && k \in \{0,1,\dots,K-1\}. \end{aligned} \tag{26}$$

As suggested by the motivating partition of Proposition 5, consider partitioning the input uncertainty set into 2^{n_p} parts based on n_p preactivation decision boundaries corresponding to activation functions in the first layer. In particular, for each $j \in \mathcal{J}_p := \{j_1, j_2, \dots, j_{n_p}\} \subseteq \{1,2,\dots,n_1\}$ we partition the input uncertainty set along the hyperplane $w_j^{[0]\top} x^{[0]} = 0$, giving rise to the partition $\{\mathcal{X}^{(1)}, \mathcal{X}^{(2)}, \dots, \mathcal{X}^{(2^{n_p})}\}$. Note that, for all $j \in \mathcal{J}_p$, the partition implies that the j^{th} coordinate of the first layer's ReLU equality constraint becomes linear and exact on each part of the partition. Therefore, for all $j' \in \{1,2,\dots,2^{n_p}\}$, we may write the LP relaxation over part $\mathcal{X}^{(j')}$ as

$$\begin{aligned} \text{maximize} \quad & c^\top x^{[K]} \\ \text{subject to} \quad & x^{[0]} \in \mathcal{X}^{(j')}, \\ & x^{[k+1]} \geq W^{[k]} x^{[k]}, && k \in \{0,1,\dots,K-1\}, \\ & x^{[k+1]} \geq 0, && k \in \{0,1,\dots,K-1\}, \\ & x^{[k+1]} \leq u^{[k+1]} \odot (W^{[k]} x^{[k]} - l^{[k+1]}) \oslash (u^{[k+1]} - l^{[k+1]}), && k \in \{0,1,\dots,K-1\}, \\ & x_j^{[1]} = \operatorname{ReLU}(w_j^{[0]\top} x^{[0]}), && j \in \mathcal{J}_p, \end{aligned} \tag{27}$$

with optimal objective value denoted by $\hat{\phi}^*_{\text{LP}}(\mathcal{X}^{(j')})$, and thus the partitioned LP relaxation becomes

$$\phi^*_{\mathcal{J}_p}(\mathcal{X}) := \max_{j' \in \{1,2,\dots,2^{n_p}\}} \hat{\phi}^*_{\text{LP}}(\mathcal{X}^{(j')}). \tag{28}$$

To reiterate, the final equality constraint in (27) is linear over the restricted feasible set $\mathcal{X}^{(j')}$, which makes the problem (28) a partitioned linear program.

If we now allow the indices used to define the partition, namely $\mathcal{J}_p$, to act as a variable, we can search for the optimal n_p rows of the first layer that result in the tightest partitioned LP relaxation. To this end, the problem of optimal partitioning in the first layer is formulated as

$$\begin{aligned} \underset{\mathcal{J}_p \subseteq \{1,2,\dots,n_1\}}{\text{minimize}} \quad & f^*_{\mathcal{J}_p}(\mathcal{X}) \\ \text{subject to} \quad & |\mathcal{J}_p| = n_p. \end{aligned} \tag{29}$$

In what follows, we prove the NP-hardness of the optimal partitioning problem (29), thereby supporting the use of the worst-case sense optimal partition developed in Theorem 8. To show the hardness of (29), we reduce an arbitrary instance of an NP-hard problem, the Min-$\mathcal{K}$-Union problem, to an instance of (29). The reduction will show that the Min-$\mathcal{K}$-Union problem can be solved by solving an optimal partitioning problem. Before we proceed, we first recall the definition of the Min-$\mathcal{K}$-Union problem.

Definition 9 (Min-$\mathcal{K}$-Union problem (Hochbaum, 1996)) *Consider a collection of n sets $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n\}$, where $\mathcal{S}_j$ is finite for all $j \in \{1, 2, \dots, n\}$, and a positive integer $\mathcal{K} \leq n$. Find $\mathcal{K}$ sets in the collection whose union has minimum cardinality, i.e., find a solution $\mathcal{J}^*$ of the following optimization problem:*

$$\begin{aligned} \underset{\mathcal{J} \subseteq \{1,2,\dots,n\}}{\text{minimize}} \quad & \left| \bigcup_{j \in \mathcal{J}} \mathcal{S}_j \right| \\ \text{subject to} \quad & |\mathcal{J}| = \mathcal{K}. \end{aligned} \tag{30}$$

Remark the similarities between the optimal partitioning problem and the Min-$\mathcal{K}$-Union problem. In particular, if we think of the convex upper envelopes of the relaxed ReLU constraints as a collection of sets, then the goal of finding the optimal n_p input coordinates to partition along is intuitively equivalent to searching for the $\mathcal{K} = n_1 - n_p$ convex upper envelopes with minimum size, i.e., those with the least amount of relaxation. This perspective shows that the optimal partitioning problem is essentially a Min-$\mathcal{K}$-Union problem over the collection of relaxed ReLU constraint sets. Since the Min-$\mathcal{K}$-Union problem is NP-hard in general, it is not surprising that the optimal partitioning problem is also NP-hard. Indeed, this result is formalized in the following proposition.

Proposition 10 (NP-hardness of optimal partition) *Consider the partitioned LP relaxation* (28) *of the K-layer ReLU neural network certification problem. The optimal partitioning problem in the first-layer, as formulated in* (29)*, is NP-hard.*

Proof See Appendix C. ■

This concludes our development and analysis for partitioning the LP relaxation. In the next section, we follow a similar line of reasoning to develop a branching scheme for the other popular convex robustness certification technique, i.e., the SDP relaxation. Despite approaching this relaxation from the same partitioning perspective as the LP, the vastly different geometries of the LP and SDP feasible sets make the branching procedures quite distinct.

4. Partitioned SDP Relaxation

4.1 Tightening of the Relaxation

As with the LP relaxation, we begin by showing that the SDP relaxation error is decreased when the input uncertainty set is partitioned. This proposition is formalized below.

Proposition 11 (Improving the SDP relaxation bound) *Consider a neural network with one hidden ReLU layer. Let $\{\mathcal{X}^{(j)} : j \in \{1, 2, \dots, p\}\}$ be a partition of $\mathcal{X}$. For the j^{th} input part $\mathcal{X}^{(j)}$, denote the corresponding input bounds by $l \leq l^{(j)} \leq x \leq u^{(j)} \leq u$, where $x \in \mathcal{X}^{(j)}$. Then, it holds that*

$$\max_{j \in \{1,2,\dots,p\}} \hat{\phi}_{\mathrm{SDP}}(\mathcal{X}^{(j)}) \leq \hat{\phi}_{\mathrm{SDP}}(\mathcal{X}). \tag{31}$$

Proof See Appendix D. ■

Proposition 11 guarantees that partitioning yields a tighter SDP relaxation. However, it is not immediately clear how to design the partition in order to maximally reduce the relaxation error. Indeed, a poorly designed partition may even yield an equality in the bound (31). One notable challenge in designing the SDP partition relates to an inherent difference between the SDP relaxation and the LP relaxation. With the LP relaxation, the effect of partitioning can be visualized by how the geometry of the feasible set changes; see Figure 4. However, with the SDP, the relaxation comes from dropping the nonconvex rank constraint, the geometry of which is more abstract and harder to exploit.

In the next section, we develop a bound measuring how far the SDP solution is from being rank-1, which corresponds to an exact relaxation, where the improvement in (31) is as good as possible. By studying the geometry of the SDP feasible set through this more tractable bound, we find that the partition design for the SDP naturally reduces to a uniform partition along the coordinate axes of the input set.

4.2 Motivating Partition

In this section, we seek the form of a partition that best reduces the SDP relaxation error. By restricting our focus to ReLU networks with one hidden layer, we develop a simple necessary condition for the SDP relaxation to be exact, i.e., for the matrix P to be rank-1. We then work on the violation of this condition to define a measure of how close P is to being rank-1 in the case it has higher rank. Next, we develop a tractable upper bound on this rank-1 gap. Finally, we formulate an optimization problem in which we search for a coordinate-wise partition of the input uncertainty set that minimizes our upper bound. Since this partition robustly minimizes our rank-1 gap-based upper bound on the relaxation error, we refer to it as "worst-case optimal," similarly to our previous worst-case optimal LP partition. We show that such a coordinate-wise worst-case optimal SDP partition takes the form of a uniform division of the input set. The result motivates the use of uniform partitions of the input uncertainty set, and in Section 4.3, we answer the question of which coordinate is best to uniformly partition along. Note that, despite the motivating partition being derived for networks with one hidden layer, the relaxation tightening in Proposition 11 still holds for multi-layer networks. Indeed, the experiments in Section 6 will show that the resulting SDP branching scheme design maintains a relatively constant efficacy as the number of layers increases.

Proposition 12 (Necessary condition for exact SDP) *Let $P^* \in \mathbb{S}^{n_x+n_z+1}$ denote a solution to the semidefinite programming relaxation* (8)*. If the relaxation is exact, meaning that* $\operatorname{rank}(P^*) = 1$*, then the following conditions hold:*

$$\operatorname{tr}(P^*_{xx}) = \|P^*_x\|_2^2, \qquad \operatorname{tr}(P^*_{zz}) = \|P^*_z\|_2^2. \tag{32}$$

Proof Since the SDP relaxation is exact, it holds that $\operatorname{rank}(P^*) = 1$. Therefore, P^* can be expressed as

$$P^* = \begin{bmatrix} 1 \\ v \\ w \end{bmatrix} \begin{bmatrix} 1 & v^\top & w^\top \end{bmatrix} = \begin{bmatrix} 1 & v^\top & w^\top \\ v & vv^\top & vw^\top \\ w & wv^\top & ww^\top \end{bmatrix}$$

for some vectors $v \in \mathbb{R}^{n_x}$ and $w \in \mathbb{R}^{n_z}$. Recall the block decomposition of P^*:

$$P^* = \begin{bmatrix} P^*_1 & P^{*\top}_x & P^{*\top}_z \\ P^*_x & P^*_{xx} & P^*_{xz} \\ P^*_z & P^*_{zx} & P^*_{zz} \end{bmatrix}.$$

Equating coefficients, we find that $P^*_{xx} = vv^\top = P^*_x P^{*\top}_x$ and $P^*_{zz} = ww^\top = P^*_z P^{*\top}_z$. Therefore,

$$\operatorname{tr}(P^*_{xx}) = \operatorname{tr}(P^*_x P^{*\top}_x) = \operatorname{tr}(P^{*\top}_x P^*_x) = \|P^*_x\|_2^2,$$

proving the first condition in (32). The second condition follows in the same way. ■

Enforcing the conditions (32) as constraints in the SDP relaxation may assist in pushing the optimization variable P towards a rank-1 solution. However, because the conditions in (32) are nonlinear equality constraints in the variable P, we cannot impose them directly on the SDP without making the problem nonconvex. Instead, we will develop a convex method based on the rank-1 conditions (32) that can be used to motivate the SDP solution to have a lower rank.

In the general case that $\operatorname{rank}(P) = r \geq 1$, P may be written as $P = VV^\top$, where

$$V = \begin{bmatrix} e^\top \\ X \\ Z \end{bmatrix}, \; e \in \mathbb{R}^r, \; X \in \mathbb{R}^{n_x \times r}, \; Z \in \mathbb{R}^{n_z \times r},$$

and where the vector e satisfies the equation $e^\top e = \|e\|_2^2 = 1$. The i^{th} row of X (respectively, Z) is denoted by $X_i^\top \in \mathbb{R}^{1\times r}$ (respectively, $Z_i^\top \in \mathbb{R}^{1\times r}$). Under this expansion, we find that

$P_x = Xe$, $P_z = Ze$, $P_{xx} = XX^\top$, and $P_{zz} = ZZ^\top$. Therefore, the conditions (32) can be written as

$$\operatorname{tr}(XX^\top) = \|Xe\|_2^2, \qquad \operatorname{tr}(ZZ^\top) = \|Ze\|_2^2.$$

To simplify the subsequent analysis, we will restrict our attention to the first of these two necessary conditions for P to be rank-1. As the simulation results in Section 6 show, this restriction still yields significant reduction in relaxation error. Now, note that

$$\operatorname{tr}(XX^\top) = \sum_{i=1}^{n_x} (XX^\top)_{ii} = \sum_{i=1}^{n_x} \|X_i\|_2^2,$$

where $(XX^\top)_{ii}$ is the (i,i) element of the matrix $XX^\top$, and also that

$$\|Xe\|_2^2 = \sum_{i=1}^{n_x} (Xe)_i^2 = \sum_{i=1}^{n_x} (X_i^\top e)^2,$$

where $(Xe)_i$ is the i^{th} element of the vector Xe. Therefore, the rank-1 necessary condition is equivalently written as

$$g(P) \coloneqq \sum_{i=1}^{n_x} (\|X_i\|_2^2 - (X_i^\top e)^2) = 0,$$

where $g(P)$ serves as a measure of the rank-1 gap. Note that $g(P)$ is solely determined by $P = VV^\top$, even though it is written in terms of X and e, which are blocks of V. In general, $g(P) \geq 0$ when $\operatorname{rank}(P) \geq 1$.

Lemma 13 (Rank-1 gap) *Let $P \in \mathbb{S}^{n_x+n_z+1}$ be an arbitrary feasible point for the SDP relaxation* (8)*. The rank-1 gap $g(P)$ is nonnegative, and is zero if P is rank-1.*

Proof By the Cauchy-Schwarz inequality, we have that $|X_i^\top e| \leq \|X_i\|_2 \|e\|_2$ for all $i \in \{1, 2, \dots, n_x\}$. Since P is feasible for (8) we also have that $\|e\|_2 = 1$, so squaring both sides of the inequality gives that $(X_i^\top e)^2 \leq \|X_i\|_2^2$. Summing these inequalities over i gives

$$g(P) = \sum_{i=1}^{n_x} (\|X_i\|_2^2 - (X_i^\top e)^2) \geq 0.$$

If P is rank-1, then the dimension r of the vectors e and X_i is equal to 1. That is, $e, X_i \in \mathbb{R}$. Hence, $\|X_i\|_2 = |X_i|$ and $|e| = \|e\|_2 = 1$, yielding $\|X_i\|_2^2 - (X_i^\top e)^2 = X_i^2 - X_i^2 e^2 = 0$. Therefore, $g(P) = 0$ in the case that $\operatorname{rank}(P) = 1$. ■

Since $g(P) = 0$ is necessary for P to be rank-1 and $g(P) \geq 0$, it is desirable to make $g(P^*)$ as small as possible at the optimal solution P^* of the partitioned SDP relaxation. Indeed, this is our partitioning motivation: we seek to partition the input uncertainty set to minimize $g(P^*)$, in order to influence P^* to be of low rank. However, there is a notable hurdle with this approach. In particular, the optimal solution P^* depends on the partition we choose, and finding a partition to minimize $g(P^*)$ in turn depends on P^* itself. To overcome this cyclic dependence, we propose first bounding $g(P^*)$ by a worst-case upper bound, and then choosing an optimal partition to minimize the upper bound. This will make the partition design tractable, resulting in a closed-form solution.

To derive the upper bound on the rank-1 gap at optimality, let $\{\mathcal{X}^{(j)} : j \in \{1, 2, \dots, p\}\}$ denote the partition of $\mathcal{X}$. For the jth input part $\mathcal{X}^{(j)}$, denote the corresponding input bounds by $l^{(j)}, u^{(j)}$. The upper bound is derived below.

Lemma 14 (Rank-1 gap upper bound) *The rank-1 gap at the solution P^* of the partitioned SDP satisfies*

$$0 \leq g(P^*) \leq \frac{1}{4} \sum_{i=1}^{n_x} \max_{j \in \{1,2,\dots,p\}} (u_i^{(j)} - l_i^{(j)})^2. \tag{33}$$

Proof The left inequality is a direct result of Lemma 13. For the right inequality, note that

$$g(P^*) \leq \max_{j \in \{1,2,\dots,p\}} \sup_{P \in \mathcal{N}_{\text{SDP}}^{(j)},\ P_x \in \mathcal{X}^{(j)}} g(P) \leq \sum_{i=1}^{n_x} \max_{j \in \{1,2,\dots,p\}} \sup_{P \in \mathcal{N}_{\text{SDP}}^{(j)},\ P_x \in \mathcal{X}^{(j)}} (\|X_i\|_2^2 - (X_i^\top e)^2). \tag{34}$$

Let us focus on the optimization over the jth part of the partition, namely,

$$\sup_{P \in \mathcal{N}_{\text{SDP}}^{(j)},\ P_x \in \mathcal{X}^{(j)}} (\|X_i\|_2^2 - (X_i^\top e)^2).$$

To bound this quantity, we analyze the geometry of the SDP relaxation over part j, following the methodology of Raghunathan et al. (2018); see Figure 5.

The shaded circle represents the set of feasible X_i over part j of the partition, namely, those satisfying the ith coordinate of the constraint $\operatorname{diag}(P_{xx}) \leq (l^{(j)} + u^{(j)}) \odot P_x - l^{(j)} \odot u^{(j)}$. To understand this, note that the constraint is equivalent to $\|X_i\|_2^2 \leq (l_i^{(j)} + u_i^{(j)}) X_i^\top e - l_i^{(j)} u_i^{(j)}$, or, more geometrically written, that $\|X_i - \frac{1}{2}(u_i^{(j)} + l_i^{(j)}) e\|_2^2 \leq \left(\frac{1}{2}(u_i^{(j)} - l_i^{(j)})\right)^2$. This shows that X_i is constrained to a 2-norm ball of radius $r_i^{(j)} = \frac{1}{2}(u_i^{(j)} - l_i^{(j)})$ centered at $\frac{1}{2}(u_i^{(j)} + l_i^{(j)}) e$, as shown in Figure 5.

The geometry of Figure 5 immediately shows that $\|X_i\|_2^2 = a^2 + (X_i^\top e)^2$ and $r_i^{(j)2} = (a+b)^2 + (X_i^\top e - \frac{1}{2}(u_i^{(j)} + l_i^{(j)}))^2$, and therefore

$$\|X_i\|_2^2 - (X_i^\top e)^2 = a^2 = r_i^{(j)2} - (X_i^\top e - \tfrac{1}{2}(u_i^{(j)} + l_i^{(j)}))^2 - 2ab - b^2.$$

Since a and b are nonnegative,

$$\begin{aligned} &\sup_{P \in \mathcal{N}_{\text{SDP}}^{(j)},\ P_x \in \mathcal{X}^{(j)}} \|X_i\|_2^2 - (X_i^\top e)^2 \\ &= \sup_{P \in \mathcal{N}_{\text{SDP}}^{(j)},\ P_x \in \mathcal{X}^{(j)}} (r_i^{(j)2} - (X_i^\top e - \tfrac{1}{2}(u_i^{(j)} + l_i^{(j)}))^2 - 2ab - b^2) \\ &\leq \sup_{P \in \mathcal{N}_{\text{SDP}}^{(j)},\ P_x \in \mathcal{X}^{(j)}} (r_i^{(j)2} - (X_i^\top e - \tfrac{1}{2}(u_i^{(j)} + l_i^{(j)}))^2) \leq r_i^{(j)2} = \frac{1}{4}(u_i^{(j)} - l_i^{(j)})^2. \end{aligned}$$

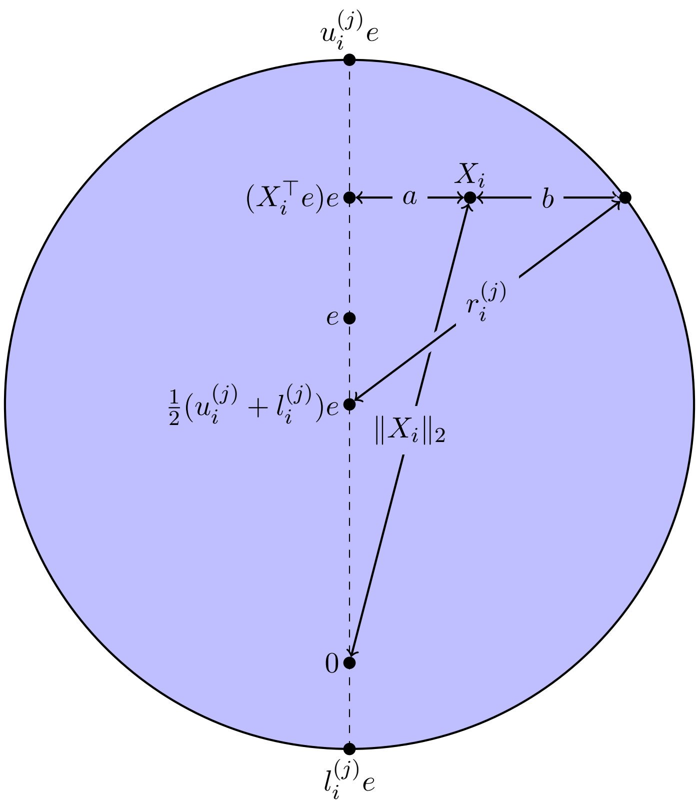


Figure 5: Geometry of the SDP relaxation in coordinate i over part j of the partition. The shaded region shows the feasible X_i satisfying the input constraint (Raghunathan et al., 2018).

Thus, (34) gives

$$g(P^*) \leq \frac{1}{4}\sum_{i=1}^{n_x} \max_{j\in\{1,2,\dots,p\}} (u_i^{(j)} - l_i^{(j)})^2,$$

as desired. ■

Remark 15 *The upper bound in* (33) *may not be tight in general, since, if the solution P^* to the partitioned SDP is rank-1—which occurs under some technical conditions (see Zhang (2020))—then $g(P^*) = 0$, whereas the upper bound will be strictly positive whenever the input bounds are such that $l_i^{(j)} < u_i^{(j)}$ for some i, j.*

With Lemma 14 in place, we now have an upper bound on the rank-1 gap at optimality, in terms of the input bounds $\{l^{(j)}, u^{(j)}\}_{j=1}^p$ associated with the partition. At this point, we turn to minimizing the upper bound over all valid choices of p-part partitions of the input uncertainty set along a given coordinate. Note that, in order for $\{l^{(j)}, u^{(j)}\}_{j=1}^p$ to define valid input bounds for a p-part partition, it must be that the union of the input parts cover the input uncertainty set. In terms of the input bounds, this leads to the constraint that $[l,u] = \cup_{j=1}^{p}[l^{(j)},u^{(j)}]$, where $[l,u] \coloneqq [l_1,u_1]\times[l_2,u_2]\times\cdots\times[l_{n_x},u_{n_x}]$, and similarly for $[l^{(j)},u^{(j)}]$. Since we consider the partition along a single coordinate k, this constraint becomes equivalent to $\cup_{j=1}^{p}[l_k^{(j)},u_k^{(j)}] = [l_k,u_k]$, because all other coordinates $i\neq k$ satisfy $l_i^{(j)} = l_i$ and $u_i^{(j)} = u_i$ for all j by assumption. We now give the coordinate-wise worst-case optimal partitioning scheme for the SDP that minimizes the upper bound in Lemma 14, which turns out to be a uniform split:

Theorem 16 (Uniform coordinate-wise SDP partition) *Let $k\in\{1,\dots,n_x\}$ and define $\mathcal{I}_k = \{1,2,\dots,n_x\}\setminus\{k\}$. Consider the optimization problem of finding the partition to minimize the upper bound* (33)*, namely*

$$\begin{array}{lll}
\underset{\mathcal{P}=\{l^{(j)},u^{(j)}\}_{j=1}^{p}\subseteq\mathbb{R}^{n_x}}{\text{minimize}} & h(\mathcal{P}) = \displaystyle\sum_{i=1}^{n_x}\max_{j\in\{1,2,\dots,p\}}(u_i^{(j)}-l_i^{(j)})^2 & \\
\text{subject to} & \displaystyle\bigcup_{j=1}^{p}[l_k^{(j)},u_k^{(j)}] = [l_k,u_k], & i\in\mathcal{I}_k,\ j\in\{1,2,\dots,p\}, \\
 & l_i^{(j)} = l_i, & i\in\mathcal{I}_k,\ j\in\{1,2,\dots,p\}, \\
 & u_i^{(j)} = u_i, & i\in\mathcal{I}_k,\ j\in\{1,2,\dots,p\},
\end{array} \tag{35}$$

Consider also the uniform partition defined by $\bar{\mathcal{P}} = \{\bar{l}^{(j)},\bar{u}^{(j)}\}_{j=1}^{p}\subseteq\mathbb{R}^{n_x}$, where

$$\bar{l}_i^{(j)} = \begin{cases}\frac{j-1}{p}(u_i-l_i)+l_i & \text{if } i=k,\\ l_i & \text{otherwise,}\end{cases}$$

$$\bar{u}_i^{(j)} = \begin{cases}\frac{j}{p}(u_i-l_i)+l_i & \text{if } i=k,\\ u_i & \text{otherwise,}\end{cases}$$

for all $j\in\{1,2,\dots,p\}$. It holds that $\bar{\mathcal{P}}$ is a solution to (35)*.*

Proof To prove the result, we show that the proposed $\bar{\mathcal{P}}$ is feasible for the optimization, and that $h(\bar{\mathcal{P}})\leq h(\mathcal{P})$ for all feasible $\mathcal{P}$. First, note that it is obvious by the definition of $\bar{\mathcal{P}}$ that $\bar{l}_i^{(j)} = l_i$ and $\bar{u}_i^{(j)} = u_i$ for all $i\in\{1,2,\dots,n_x\}\setminus\{k\}$ and all $j\in\{1,2,\dots,p\}$. Therefore, to prove that $\bar{\mathcal{P}}$ is feasible, it suffices to show that $\cup_{j=1}^{p}[\bar{l}_k^{(j)},\bar{u}_k^{(j)}] = [l_k,u_k]$. Indeed, since

$$\bar{u}_k^{(j)} = \frac{j}{p}(u_k-l_k)+l_k = \frac{(j+1)-1}{p}(u_k-l_k)+l_k = \bar{l}_k^{(j+1)}$$

for all $j\in\{1,2,\dots,p-1\}$,

$$\bar{l}_k^{(1)} = \frac{1-1}{p}(u_k-l_k)+l_k = l_k,$$

and

$$\bar{u}_k^{(p)} = \frac{p}{p}(u_k-l_k)+l_k = u_k,$$

we have that

$$\bigcup_{j=1}^{p}[\bar{l}_k^{(j)},\bar{u}_k^{(j)}] = [\bar{l}_k^{(1)},\bar{u}_k^{(1)}] \cup [\bar{l}_k^{(2)},\bar{u}_k^{(2)}] \cup \cdots \cup [\bar{l}_k^{(p)},\bar{u}_k^{(p)}] = [\bar{l}_k^{(1)},\bar{u}_k^{(p)}] = [l_k,u_k].$$

Hence, $\bar{\mathcal{P}} = \{\bar{l}^{(j)},\bar{u}^{(j)}\}_{j=1}^{p}$ is feasible.

The objective at the proposed feasible point can be computed as

$$\begin{aligned} h(\bar{\mathcal{P}}) &= \sum_{i=1}^{n_x} \max_{j\in\{1,2,\dots,p\}} (\bar{u}_i^{(j)} - \bar{l}_i^{(j)})^2 = \sum_{\substack{i=1\\ i\neq k}}^{n_x} \max_{j\in\{1,2,\dots,p\}} (\bar{u}_i^{(j)} - \bar{l}_i^{(j)})^2 + \max_{j\in\{1,2,\dots,p\}} (\bar{u}_k^{(j)} - \bar{l}_k^{(j)})^2 \\ &= \sum_{\substack{i=1\\ i\neq k}}^{n_x} \max_{j\in\{1,2,\dots,p\}} (u_i - l_i)^2 + \max_{j\in\{1,2,\dots,p\}} \left(\frac{j}{p}(u_k-l_k) + l_k - \frac{j-1}{p}(u_k-l_k) - l_k\right)^2 \\ &= \sum_{\substack{i=1\\ i\neq k}}^{n_x} (u_i - l_i)^2 + \max_{j\in\{1,2,\dots,p\}} \left(\frac{1}{p}(u_k-l_k)\right)^2 = C + \frac{1}{p^2}(u_k-l_k)^2, \end{aligned}$$

where $C \coloneqq \sum_{\substack{i=1\\ i\neq k}}^{n_x}(u_i-l_i)^2$. Now, let $\mathcal{P} = \{l^{(j)},u^{(j)}\}_{j=1}^{p}$ be an arbitrary feasible point for the optimization (35). Then by a similar analysis as above, the objective value at $\mathcal{P}$ satisfies

$$\begin{aligned} h(\mathcal{P}) &= \sum_{i=1}^{n_x} \max_{j\in\{1,2,\dots,p\}} (u_i^{(j)} - l_i^{(j)})^2 = \sum_{\substack{i=1\\ i\neq k}}^{n_x} \max_{j\in\{1,2,\dots,p\}} (u_i^{(j)} - l_i^{(j)})^2 + \max_{j\in\{1,2,\dots,p\}} (u_k^{(j)} - l_k^{(j)})^2 \\ &= C + \max_{j\in\{1,2,\dots,p\}} (u_k^{(j)} - l_k^{(j)})^2 = C + \left(\max_{j\in\{1,2,\dots,p\}} (u_k^{(j)} - l_k^{(j)})\right)^2 \\ &\geq C + \left(\frac{1}{p}\sum_{j=1}^{p}(u_k^{(j)} - l_k^{(j)})\right)^2 = C + \frac{1}{p^2}\left(\sum_{j=1}^{p}(u_k^{(j)} - l_k^{(j)})\right)^2. \end{aligned}$$

Since $\mathcal{P}$ is feasible, it holds that $[l_k,u_k] = \bigcup_{j=1}^{p}[l_k^{(j)},u_k^{(j)}]$. Therefore, by subadditivity of Lebesgue measure μ on the Borel σ-algebra of $\mathbb{R}$, we have that

$$u_k - l_k = \mu([l_k,u_k]) = \mu\left(\bigcup_{j=1}^{p}[l_k^{(j)},u_k^{(j)}]\right) \leq \sum_{j=1}^{p}\mu([l_k^{(j)},u_k^{(j)}]) = \sum_{j=1}^{p}(u_k^{(j)} - l_k^{(j)}).$$

Substituting this into our above expressions, we conclude that

$$h(\bar{\mathcal{P}}) = C + \frac{1}{p^2}(u_k-l_k)^2 \leq C + \frac{1}{p^2}\left(\sum_{j=1}^{p}(u_k^{(j)} - l_k^{(j)})\right)^2 \leq h(\mathcal{P}).$$

Since $\mathcal{P}$ was an arbitrary feasible point for the optimization, this implies that $\bar{\mathcal{P}}$ is a solution to the optimization. ■

It is important to note that, in general, an optimal partition of the input uncertainty set that minimizes the actual SDP relaxation error may not be along a coordinate axis. However, finding such an optimal general-form partition poses a significantly challenging open problem. Our Theorem 16 shows that, by restricting ourselves to coordinate-wise partitions, the analysis becomes tractable, and we find that the partition that uniformly splits the input set along the given coordinate is an optimal one (with respect to our worst-case upper bound (33)). This gives a well-motivated, yet simple way to design a partition of the input uncertainty set in order to push the SDP relaxation towards being rank-1, thereby reducing relaxation error.

4.3 SDP Branching Scheme

With the motivating partition of Section 4.2 now established, we turn our attention from the *form* of a coordinate-wise worst-case optimal SDP partition to the *coordinate* of such a partition. In particular, we seek to find the best branching scheme to minimize relaxation error of the SDP. The results of Section 4.2 suggest using a uniform partition, and in this section we seek to find which coordinate to apply the partitioning to. Similar to the LP relaxation, we derive an optimal branching scheme by first bounding the relaxation error in the worst-case sense.

4.3.1 Worst-Case Relaxation Bound

In the worst-case relaxation bound of Theorem 19 below, and the subsequent coordinate-wise worst-case optimal SDP branching scheme proposed in Theorem 20, we restrict our attention to a single hidden ReLU layer and make the following assumption on the weight matrix.

Assumption 17 (Normalized rows) *The rows of the weight matrix are assumed to be normalized with respect to the ℓ_1-norm, i.e., that $\|w_i\|_1 = 1$ for all $i \in \{1, 2, \dots, n_z\}$.*

We briefly remark that Assumption 17 imposes no loss of generality, as it can be made to hold for any network by a simple rescaling. In particular, if the assumption does not hold, the network architecture can be rescaled as follows:

$$z = \operatorname{ReLU}(Wx) = \operatorname{ReLU}\left(\begin{bmatrix} w_1^\top \\ w_2^\top \\ \vdots \\ w_{n_z}^\top \end{bmatrix} x\right)$$

$$= \operatorname{ReLU}\left(\operatorname{diag}(\|w_1\|_1, \|w_2\|_1, \dots, \|w_{n_z}\|_1) \begin{bmatrix} \frac{w_1^\top}{\|w_1\|_1} \\ \frac{w_2^\top}{\|w_2\|_1} \\ \vdots \\ \frac{w_{n_z}^\top}{\|w_{n_z}\|_1} \end{bmatrix} x\right) = W_{\text{scale}} \operatorname{ReLU}(W_{\text{norm}} x),$$

where $W_{\text{scale}} = \operatorname{diag}(\|w_1\|_1, \|w_2\|_1, \dots, \|w_{n_z}\|_1) \in \mathbb{R}^{n_z \times n_z}$ and

$$W_{\text{norm}} = \begin{bmatrix} \frac{w_1^\top}{\|w_1\|_1} \\ \frac{w_2^\top}{\|w_2\|_1} \\ \vdots \\ \frac{w_{n_z}^\top}{\|w_{n_z}\|_1} \end{bmatrix} \in \mathbb{R}^{n_z \times n_x}$$

are the scaling and normalized factors of the weight matrix W, respectively. The scaling factor can therefore be absorbed into the optimization cost vector c, yielding a problem with normalized rows as desired.

Before introducing the worst-case relaxation bound of Theorem 19, we state a short lemma that will be used in proving the relaxation bound.

Lemma 18 (Bound on elements of PSD matrices) *Let $P \in \mathbb{S}^n$ be a positive semidefinite matrix. Then $|P_{ij}| \leq \frac{1}{2}(P_{ii} + P_{jj})$ for all $i, j \in \{1, 2, \dots, n\}$.*

Proof See Appendix E. ■

Theorem 19 (Worst-case relaxation bound for SDP) *Consider a feedforward ReLU neural network with one hidden layer, and with the input uncertainty set $\mathcal{X}$. Let the network have input bounds $l, u \in \mathbb{R}^{n_x}$ and preactivation bounds $\hat{l}, \hat{u} \in \mathbb{R}^{n_z}$. Consider also the relaxation error $\Delta\phi^*_{\text{SDP}}(\mathcal{X}) \coloneqq \hat{\phi}_{\text{SDP}}(\mathcal{X}) - \phi^*(\mathcal{X})$. Let P^* and (x^*, z^*) be optimal solutions for the relaxation $\hat{\phi}_{\text{SDP}}(\mathcal{X})$ and the unrelaxed problem $\phi^*(\mathcal{X})$, respectively. Given an arbitrary norm $\|\cdot\|$ on $\mathbb{R}^{n_x}$, it holds that*

$$\Delta\phi^*_{\text{SDP}}(\mathcal{X}) \leq \sum_{i=1}^{n_z} \Big(\operatorname{ReLU}(c_i) q(l, u) + \operatorname{ReLU}(-c_i) \min\{\hat{u}_i, \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})\} \Big), \tag{36}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$, and where

$$q(l, u) = \max_{k \in \{1,2,\dots,n_x\}} \max\{|l_k|, |u_k|\}.$$

Proof First, recall that $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is assumed to be compact, and is therefore bounded, and hence $d_{\|\cdot\|}(\mathcal{X}) < \infty$. The definitions of P^*_x and (x^*, z^*) give that

$$\Delta\phi^*_{\text{SDP}}(\mathcal{X}) = \sum_{i=1}^{n_z} c_i((P^*_z)_i - z^*_i) \leq \sum_{i=1}^{n_z} \Delta\phi^*_i, \tag{37}$$

where

$$\Delta\phi^*_i = \sup \Big\{ c_i((P_z)_i - z_i) : z_i = \operatorname{ReLU}(w_i^\top x),\ P_z \geq 0,\ P_z \geq W P_x,$$
$$\operatorname{diag}(P_{zz}) = \operatorname{diag}(W P_{xz}),\ P_1 = 1,\ P \succeq 0,\ x, P_x \in \mathcal{X} \Big\}$$

for all $i \in \{1, 2, \dots, n_z\}$. Defining the auxiliary variables $P_{\hat{z}} = WP_x$ and $\hat{z} = Wx$, this is equivalent to

$$\Delta\phi_i^* = \sup \Bigg\{ c_i((P_z)_i - z_i) : z_i = \mathrm{ReLU}(\hat{z}_i),\ P_z \geq 0,\ P_z \geq P_{\hat{z}},\ \mathrm{diag}(P_{zz}) = \mathrm{diag}(WP_{xz}),$$
$$P_1 = 1,\ P \succeq 0,\ P_{\hat{z}} = WP_x,\ \hat{z}_i = w_i^\top x,\ x, P_x \in \mathcal{X} \Bigg\}.$$

If $x, P_x \in \mathcal{X}$ and $\hat{z}, P_{\hat{z}}$ satisfy $\hat{z} = Wx$ and $P_{\hat{z}} = WP_x$, then $|(P_{\hat{z}})_i - \hat{z}_i| = |w_i^\top (P_x - x)| \leq \|w_i\|_* \|P_x - x\| \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$ for all $i \in \{1, 2, \dots, n_z\}$ by the Cauchy-Schwarz inequality for dual norms. Therefore,

$$\Delta\phi_i^* \leq \sup \Bigg\{ c_i((P_z)_i - z_i) : z_i = \mathrm{ReLU}(\hat{z}_i),\ P_z \geq 0,\ P_z \geq P_{\hat{z}},$$
$$\mathrm{diag}(P_{zz}) = \mathrm{diag}(WP_{xz}),\ P_1 = 1,\ P \succeq 0,\ \hat{l} \leq \hat{z}, P_{\hat{z}} \leq \hat{u},$$
$$|(P_{\hat{z}})_k - \hat{z}_k| \leq \|w_k\|_* d_{\|\cdot\|}(\mathcal{X}) \text{ for all } k \in \{1, 2, \dots, n_z\},\ \hat{z}, P_{\hat{z}} \in \mathbb{R}^{n_z} \Bigg\}.$$

We now translate the optimization variables in the above problem from $\hat{z} \in \mathbb{R}^{n_z}$ and $P \in \mathbb{S}^{1+n_x+n_z}$ to the scalars $\hat{z}_i, (P_{\hat{z}})_i \in \mathbb{R}$. To this end, we note that if P is feasible for the above supremum, then

$$\mathrm{diag}(P_{zz})_i = \mathrm{diag}(WP_{xz})_i = w_i^\top (P_{xz})_i \leq \|(P_{xz})_i\|_\infty \|w_i\|_1,$$

where $(P_{xz})_i$ is the i^{th} column of the matrix P_{xz}, and the inequality again comes from Cauchy-Schwarz. By the weight matrix scaling assumption, this yields

$$\mathrm{diag}(P_{zz})_i \leq \|(P_{xz})_i\|_\infty.$$

Now, since P is positive semidefinite, Lemma 18 gives that

$$\|(P_{xz})_i\|_\infty = \max_{k \in \{1,2,\dots,n_x\}} |(P_{xz})_i|_k = \max_{k \in \{1,2,\dots,n_x\}} |(P_{xz})_{ki}|$$
$$\leq \max_{k \in \{1,2,\dots,n_z\}} \frac{1}{2}\Big((P_{xx})_{kk} + (P_{zz})_{ii} \Big) = \frac{1}{2}(P_{zz})_{ii} + \frac{1}{2} \max_{k \in \{1,2,\dots,n_x\}} (P_{xx})_{kk}.$$

Noting that $(P_{zz})_{ii} = \mathrm{diag}(P_{zz})_i$, the bound of interest becomes

$$\mathrm{diag}(P_{zz})_i \leq \max_{k \in \{1,2,\dots,n_z\}} (P_{xx})_{kk}.$$

We now seek to bound $(P_{xx})_{kk}$. Recall that $(P_{xx})_{kk} = \mathrm{diag}(P_{xx})_k \leq (l_k + u_k)(P_x)_k - l_k u_k$. If $(l_k + u_k) \geq 0$, then $(P_x)_k \leq u_k$ implies that $(l_k + u_k)(P_x)_k \leq (l_k + u_k)u_k$, and therefore $(P_{xx})_{kk} \leq (l_k + u_k)u_k - l_k u_k = u_k^2$. On the other hand, if $(l_k + u_k) < 0$, then $(P_x)_k \geq l_k$ implies that $(l_k + u_k)(P_x)_k \leq (l_k + u_k)l_k$, and therefore $(P_{xx})_{kk} \leq (l_k + u_k)l_k - l_k u_k = l_k^2$. Hence, in all cases, it holds that

$$(P_{xx})_{kk} \leq \mathbb{I}(l_k + u_k \geq 0)u_k^2 + \mathbb{I}(l_k + u_k < 0)l_k^2.$$

We can further simplify this bound as follows. If $l_k + u_k \geq 0$, then $u_k \geq -l_k$ and $u_k \geq l_k$, implying $|l_k| \leq u_k$, so $l_k^2 \leq u_k^2$ and therefore $u_k^2 = \max\{l_k^2, u_k^2\}$. On the other hand, if $l_k + u_k < 0$, then an analogous argument shows that $l_k^2 = \max\{l_k^2, u_k^2\}$. Hence, we conclude that the above bound on $(P_{xx})_{kk}$ can be rewritten as

$$(P_{xx})_{kk} \leq \max\{l_k^2, u_k^2\}.$$

Therefore, returning to the bound on $(P_{zz})_i$, we find that

$$\operatorname{diag}(P_{zz})_i \leq \max_{k \in \{1,2,\dots,n_x\}} \max\{l_k^2, u_k^2\},$$

for all $i \in \{1, 2, \dots, n_z\}$. Now, note that since $P \succeq 0$, the Schur complement gives that

$$\begin{bmatrix} P_{xx} - P_x P_x^\top & P_{xz} - P_x P_z^\top \\ P_{xz}^\top - P_z P_x^\top & P_{zz} - P_z P_z^\top \end{bmatrix} \succeq 0,$$

which implies that

$$\operatorname{diag}(P_{zz}) \geq \operatorname{diag}(P_z P_z^\top) = P_z \odot P_z.$$

Therefore, our upper bound on the diagonal elements of P_{zz} yields that

$$(P_z)_i \leq \max_{k \in \{1,2,\dots,n_x\}} \max\{|l_k|, |u_k|\} = q(l, u).$$

Hence, we have derived a condition on the component $(P_z)_i$ that all feasible P must satisfy. The supremum of interest may now be further upper bounded giving rise to

$$\begin{aligned}\Delta\phi_i^* \leq \sup \Big\{ & c_i((P_z)_i - z_i) : z_i = \operatorname{ReLU}(\hat{z}_i),\ (P_z)_i \geq 0,\ (P_z)_i \geq (P_{\hat{z}})_i,\ (P_z)_i \leq q(l, u), \\ & \hat{l}_i \leq \hat{z}_i, (P_{\hat{z}})_i \leq \hat{u}_i,\ |(P_{\hat{z}})_i - \hat{z}_i| \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X}),\ \hat{z}_i, (P_{\hat{z}})_i \in \mathbb{R} \Big\},\end{aligned} \tag{38}$$

which is now in terms of the scalar optimization variables $\hat{z}_i$ and $(P_{\hat{z}})_i$, as we desired. This reformulation makes it tractable to compute the supremum in (38) in closed-form, which we now turn to do.

First, consider the case that $c_i \geq 0$. Then we seek to maximize the difference $(P_z)_i - z_i$ subject to the given constraints. Noting that $(P_z)_i \leq q(l, u)$ and $z_i \geq 0$ on the above feasible set, we remark that the objective is upper bounded as $c_i((P_z)_i - z_i) \leq c_i q(l, u)$. Indeed, this upper bound is attained at the feasible point defined by $z_i = \hat{z}_i = (P_{\hat{z}})_i = 0$ and $(P_z)_i = q(l, u)$. Hence, we conclude that for all $i \in \{1, 2, \dots, n_z\}$ such that $c_i \geq 0$, it holds that

$$\Delta\phi_i^* \leq c_i q(l, u). \tag{39}$$

Now consider the case that $c_i < 0$. Then we seek to minimize the difference $(P_z)_i - z_i$ subject to the given constraints. In this case, the optimal objective value depends on the relative sizes of $\hat{u}_i$ and $\|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$. In particular, when $\hat{u}_i \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$, the constraint $\hat{z}_i \leq \hat{u}_i$ becomes active at optimum, yielding a supremum value of $-c_i u_i$. Alternatively, when $\|w_i\|_* d_{\|\cdot\|}(\mathcal{X}) \leq \hat{u}_i$, the constraint $|(P_{\hat{z}})_i - \hat{z}_i| \leq \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$ becomes active at optimum, yielding the supremum value of $-c_i\|w_i\|_* d_{\|\cdot\|}(\mathcal{X})$. Therefore, we conclude that for all $i \in \{1,2,\dots,n_z\}$ such that $c_i < 0$, it holds that

$$\Delta\phi_i^* \leq -c_i \min\{\hat{u}_i, \|w_i\|_* d_{\|\cdot\|}(\mathcal{X})\}. \tag{40}$$

Substituting (39) and (40) into (37) gives the desired bound. ■

When the x-block P_x^* of the SDP relaxation stays close to the true solution x^*, the bound (36) shows that the worst-case relaxation error scales with the loosest input bound, i.e., the maximum value amongst the limits $|l_k|$ and $|u_k|$. This fact allows us to choose which coordinate to partition along in order to maximally reduce the relaxation bound on the individual parts of the partition. We state our proposed SDP branching scheme next.

4.3.2 Proposed Branching Scheme

We now focus on developing a coordinate-wise worst-case optimal branching scheme based on the relaxation bound of Theorem 19. Similar to the partitioned LP relaxation, the diameter $d_{\|\cdot\|}(\mathcal{X})$ tends to be small in practical settings, making the terms being summed in (36) approximately equal to $\mathrm{ReLU}(c_i)q(l,u)$. We restrict ourselves to this form in order to simplify the subsequent analysis. The objective to optimize therefore takes the form

$$q(l,u)\sum_{i=1}^{n_z}\mathrm{ReLU}(c_i), \tag{41}$$

where

$$q(l,u) = \max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\}.$$

Since the design of the partition amounts to choosing input bounds l and u for the input parts, the input bounds serve as our optimization variables in minimizing the above relaxation bound. By restricting the form of our partition to the uniform division motivated in Theorem 16, it follows from the form of q that the best coordinate to partition along is that with the loosest input bound, i.e., along coordinate $i^* \in \arg\max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\}$. This observation is formalized below.

Theorem 20 (Coordinate-wise worst-case optimal SDP branching) *Consider the two-part partitions defined by dividing $\mathcal{X}$ uniformly along the coordinate axes: $\{\mathcal{X}_i^{(1)},\mathcal{X}_i^{(2)}\}$, with $\mathcal{X}_i^{(1)} = \{x\in\mathcal{X} : l_i^{(1)} \leq x \leq u_i^{(1)}\}$ and $\mathcal{X}_i^{(2)} = \{x\in\mathcal{X} : l_i^{(2)} \leq x \leq u_i^{(2)}\}$, where $l_i^{(1)} = l$, $u_i^{(1)} = (u_1,u_2,\dots,u_{i-1},\frac{1}{2}(l_i+u_i),u_{i+1},\dots,u_{n_x})$, $l_i^{(2)} = (l_1,l_2,\dots,l_{i-1},\frac{1}{2}(l_i+u_i),l_{i+1},\dots,l_{n_x})$, and $u_i^{(2)} = u$, for all $i\in\{1,2,\dots,n_x\} =: \mathcal{I}$. Let*

$$i^* \in \operatorname*{arg\,max}_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\}, \tag{42}$$

and assume that $|l_{i^*}| \neq |u_{i^*}|$. Then the partition $\{\mathcal{X}_{i^*}^{(1)},\mathcal{X}_{i^*}^{(2)}\}$ is optimal in the sense that the upper bound factor $q(l_{i^*}^{(j)},u_{i^*}^{(j)})$ in (41) equals the unpartitioned upper bound $q(l,u)$ on one part j of the partition, is strictly less than $q(l,u)$ on the other part, and $q(l_i^{(j)},u_i^{(j)}) = q(l,u)$ for both $j\in\{1,2\}$ for all other $i \notin \arg\max_{k\in\{1,2,\dots,n_x\}}\max\{|l_k|,|u_k|\}$.

Proof First, consider partitioning along coordinate $i \notin \arg\max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\}$. Then

$$q(l_i^{(1)},u_i^{(1)}) = \max_{k\in\{1,2,\dots,n_x\}} \max\{|(l_i^{(1)})_k|,|(u_i^{(1)})_k|\}$$
$$= \max\left\{|l_1|,\dots,|l_{n_x}|,|u_1|,\dots,\left|\frac{l_i+u_i}{2}\right|,\dots,|u_{n_x}|\right\} = \max\{|l_{i^*}|,|u_{i^*}|\} = q(l,u),$$

since $\left|\frac{l_i+u_i}{2}\right| \le \frac{|l_i|+|u_i|}{2} < \max\{|l_{i^*}|,|u_{i^*}|\}$ and $i \neq i^*$ implies that

$$\max\{|l_{i^*}|,|u_{i^*}|\} \in \left\{|l_1|,\dots,|l_{n_x}|,|u_1|,\dots,\left|\frac{l_i+u_i}{2}\right|,\dots,|u_{n_x}|\right\}.$$

In an analogous fashion, it follows that

$$q(l_i^{(2)},u_i^{(2)}) = q(l,u).$$

Now, consider partitioning along coordinate i^*. Note that either

$$\max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\} = |l_{i^*}|, \text{ or } \max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\} = |u_{i^*}|.$$

Suppose that the first case holds true. Then

$$q(l_{i^*}^{(1)},u_{i^*}^{(1)}) = \max_{k\in\{1,2,\dots,n_x\}} \max\{|(l_{i^*}^{(1)})_k|,|(u_{i^*}^{(1)})_k|\}$$
$$= \max\left\{|l_1|,\dots,|l_{n_x}|,|u_1|,\dots,\left|\frac{l_{i^*}+u_{i^*}}{2}\right|,\dots,|u_{n_x}|\right\} = |l_{i^*}| = q(l,u),$$

since $|(l_{i^*}+u_{i^*})/2| \le (|l_i^*|+|u_i^*|)/2 < |l_i^*|$ and

$$|l_{i^*}| \in \max\left\{|l_1|,\dots,|l_{n_x}|,|u_1|,\dots,\left|\frac{l_{i^*}+u_{i^*}}{2}\right|,\dots,|u_{n_x}|\right\}.$$

Over the second part of the partition,

$$q(l_{i^*}^{(2)},u_{i^*}^{(2)}) = \max_{k\in\{1,2,\dots,n_x\}} \max\{|(l_{i^*}^{(2)})_k|,|(u_{i^*}^{(2)})_k|\}$$
$$= \max\left\{|l_1|,\dots,\left|\frac{l_{i^*}+u_{i^*}}{2}\right|,\dots,|l_{n_x}|,|u_1|,\dots,|u_{n_x}|\right\} < |l_{i^*}| = q(l,u),$$

since $|(l_{i^*}+u_{i^*})/2| < |l_{i^*}|$ and

$$|l_i^*| \notin \left\{|l_1|,\dots,\left|\frac{l_{i^*}+u_{i^*}}{2}\right|,\dots,|l_{n_x}|,|u_1|,\dots,|u_{n_x}|\right\}$$

since $|l_{i^*}| \neq |u_{i^*}|$. In the other case that $\max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\} = |u_{i^*}|$, it follows via the same argument that $q(l_{i^*}^{(1)},u_{i^*}^{(1)}) < q(l,u)$ and $q(l_{i^*}^{(2)},u_{i^*}^{(2)}) = q(l,u)$. Since partitioning along any other coordinate $i \notin \arg\max_{k\in\{1,2,\dots,n_x\}} \max\{|l_k|,|u_k|\}$ was shown to yield $q(l_i^{(1)},u_i^{(1)}) = q(l_i^{(2)},u_i^{(2)}) = q(l,u)$, the coordinate i^* is optimal in the sense proposed. ■

Algorithm 2 LP branch-and-bound procedure for certification

```
Input: $f$, $\mathcal{X}$, $n_{\text{branches}}$
 1: for $n = 1, \dots, n_{\text{branches}}$
 2:     compute $\mathcal{X}_{i^*}^{(1)}, \mathcal{X}_{i^*}^{(2)}$ according to Theorem 8
 3:     compute $\hat{\phi}_{\text{LP}}(\mathcal{X}_{i^*}^{(1)}), \hat{\phi}_{\text{LP}}(\mathcal{X}_{i^*}^{(2)})$ according to (5)
 4:     if $\hat{\phi}_{\text{LP}}^*(\mathcal{X}_{i^*}^{(1)}) \geq \hat{\phi}_{\text{LP}}^*(\mathcal{X}_{i^*}^{(2)})$
 5:         assign $\mathcal{X} \leftarrow \mathcal{X}_{i^*}^{(1)}$
 6:     else
 7:         assign $\mathcal{X} \leftarrow \mathcal{X}_{i^*}^{(2)}$
 8: if $\max_{j \in \{1,2\}} \hat{\phi}_{\text{LP}}^*(\mathcal{X}_{i^*}^{(j)}) \leq 0$
 9:     assign is_certified ← True
10: else
11:     assign is_certified ← False
12: return is_certified
```

Intuitively, the branching scheme defined in Theorem 20 is the worst-case optimal coordinate-wise method, because any other uniform partition along a coordinate axis cannot tighten the relaxation error bound (41). On the other hand, Theorem 20 guarantees that using the partition coordinate in (42) results in a strict tightening of the relaxation error upper bound factor q on at least one part of the partition.

As was the case with our LP branching scheme, we emphasize that our theoretical results for the SDP (namely, Theorems 16, 19, and 20) are proven under the assumption of a single hidden layer. However, we may extend our SDP branching scheme (42) into a branching heuristic for deep multi-layer networks, as we did with the LP branching scheme. Unlike the LP, where a surrogate "c"-vector was defined in order to apply our LP branching score to neurons in layers other than the last, our SDP branching score values can be directly computed in the multi-layer setting at neuron i in layer k as $\max\{|l_i^{[k]}|, |u_i^{[k]}|\}$, since such scores only depend on the k^{th}-layer activation bounds $l^{[k]}, u^{[k]}$ satisfying $l^{[k]} \leq x^{[k]} \leq u^{[k]}$ for all $x \in \mathcal{X}$.

5. Implementing the Branching Schemes

For the reader's convenience, we give a pseudocode in Algorithm 2 to embed our branching schemes (Theorems 8 and 20) into an overall branch-and-bound algorithm, which is what we use to compute the simulation results in Section 6 below. The pseudocode for the SDP branch-and-bound procedure is the same as in Algorithm 2 albeit with lines 2 and 3 modified to use Theorem 20 and (8), respectively. Solving the two subproblems in line 3 can be done independently, and in a parallel fashion, to enhance computational efficiency and scalability.

6. Simulation Results

In this section, we experimentally corroborate the effectiveness of our proposed certification methods. We first perform single-hidden layer LP branch-and-bound on moderately-sized benchmark datasets and find that our derived branching scheme beats the current state-of-the-art. We then compute the SDP and branched SDP, and compare to the LP results. Next, we explore the effectiveness of branching on the LP and SDP as networks grow in size, namely, as the number of inputs and the number of layers independently increase. Finally, we compare our multi-layer LP branching heuristic to the state-of-the-art on large-scale certification problems from the α, β-CROWN benchmarks and VNN competition benchmarks (Wang et al., 2021; Bak et al., 2021). The experiments in Sections 6.1, 6.2, and 6.3 are performed on a standard laptop computer using Tensorflow 2.5 in Python 3.9. Training of networks is done using the Adam optimizer (Kingma and Ba, 2015), and certifications are performed using MOSEK in CVXPY (Diamond and Boyd, 2016). The experiments in Section 6.4 are conducted on a desktop computer equipped with an Nvidia 4080 GPU, using PyTorch 2.0.1 and Python 3.11.

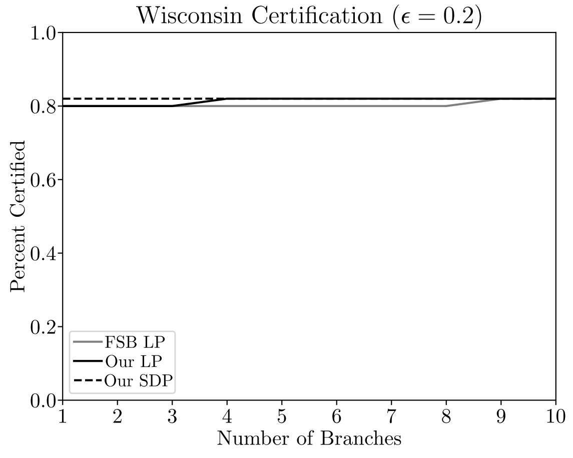
6.1 Single-Hidden Layer LP Results

In this experiment, we consider classification networks trained on three datasets: the Wisconsin breast cancer diagnosis dataset with $(n_x, n_z) = (30, 2)$ (Dua and Graff, 2017), the MNIST handwritten digit dataset with $(n_x, n_z) = (784, 10)$ (LeCun, 1998), and the CIFAR-10 image classification dataset with $(n_x, n_z) = (3072, 10)$ (Krizhevsky and Hinton, 2009). The Wisconsin breast cancer dataset is characteristic of a real-world machine learning setting in which robustness guarantees are crucial for safety; a misdiagnosis of breast cancer may result in grave consequences for the patient under concern. Each neural network is composed of an affine layer followed by a ReLU hidden layer followed by another affine layer. For each network, we consider 15 different nominal inputs $\bar{x}$ and corresponding uncertainty sets $\mathcal{X} = \{x \in \mathbb{R}^{n_x} : \|x - \bar{x}\|_\infty \leq \epsilon\}$, where we choose a range of attack radii ϵ. We also perform a sweep over the number of branching steps to use in the branch-and-bound certification scheme. Recall that a negative optimal objective value of the robustness certification problem proves that no perturbation of $\bar{x}$ within $\mathcal{X}$ results in misclassification.

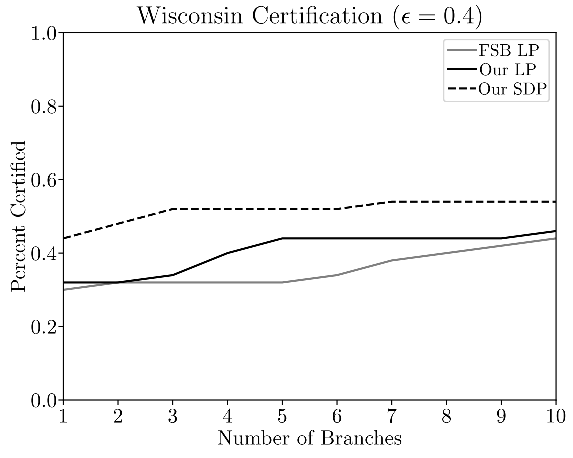
Figures 6, 7, and 8 display the percent of test inputs certified using the LP relaxation with our branching method, as well as those certified using the state-of-the-art LP branching method, filtered smart branching (FSB) (De Palma et al., 2021). FSB was used in the α, β-CROWN branch-and-bound scheme to win the 2021, 2022, and 2023 VNN-COMP certification competitions (Wang et al., 2021; Bak et al., 2021; Müller et al., 2022a; Brix et al., 2023). We see that, for all three benchmarks and at all considered attack radii ϵ and number of branching steps, our LP branching method meets or exceeds the performance of FSB; our method attains higher certification percentages in fewer branching steps and at larger radii than FSB. At some radius-number of branches settings, we even see that our LP branching scheme substantially increases the percentage certified by approximately 10% (Wisconsin), 20% (MNIST), and 20% (CIFAR-10).

6.2 Single-Hidden Layer SDP Results

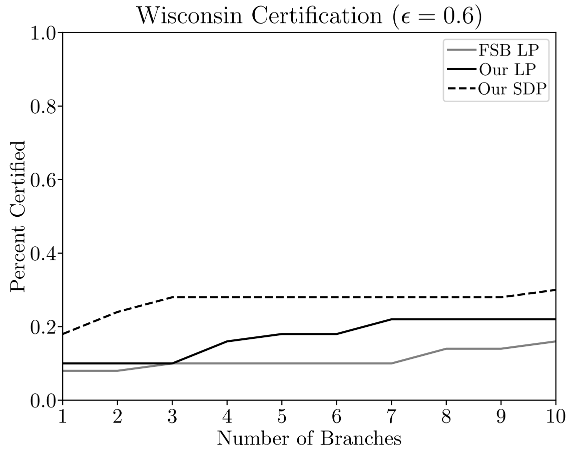
In this experiment, we consider the same single-hidden layer ReLU neural network trained on the Wisconsin breast cancer dataset as used in Section 6.1. We solve the branched SDP using our proposed branching scheme from Theorem 20, where the same test data and simulation parameters are used as with the LP in Section 6.1. The results are shown



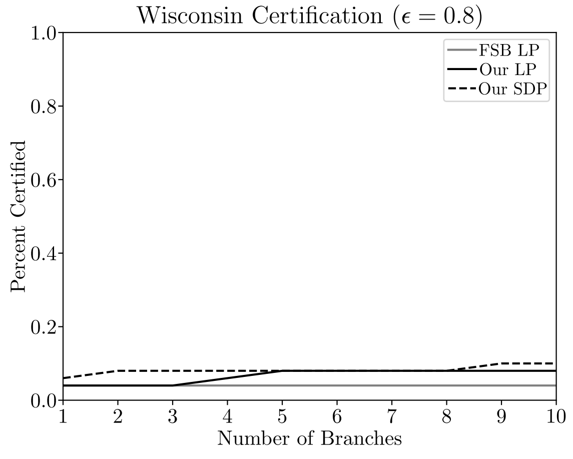
(a) Attack radius $\epsilon = 0.2$.



(b) Attack radius $\epsilon = 0.4$.



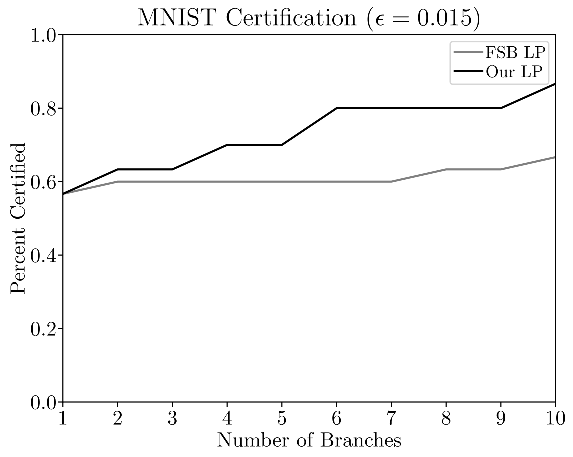
(c) Attack radius $\epsilon = 0.6$.



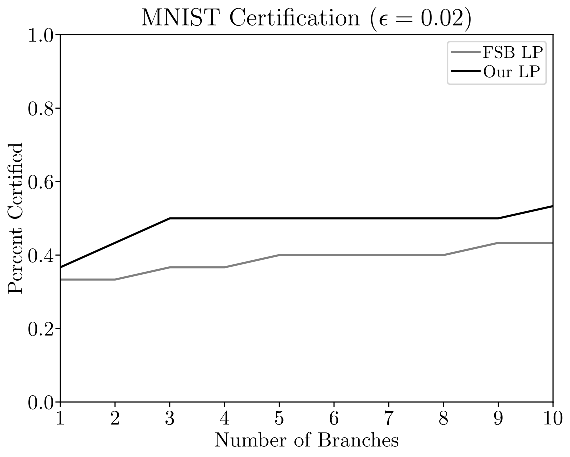
(d) Attack radius $\epsilon = 0.8$.

Figure 6: Percent certified on Wisconsin breast cancer diagnosis dataset using branching on LP and SDP relaxations of single-hidden layer network.

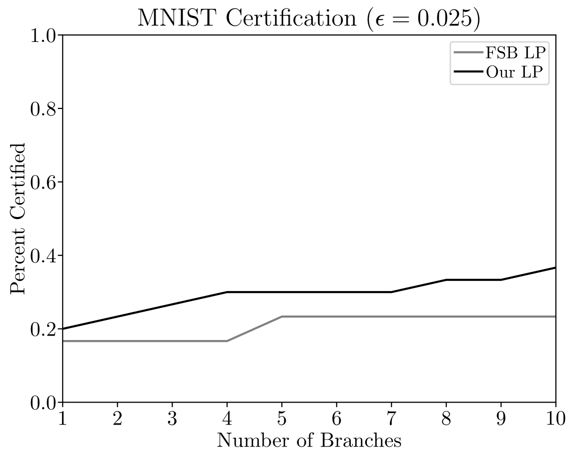
in Figure 6. In comparing the SDP results to the LP results in Figure 6, we see that our branched SDP achieves better certification percentages compared to both of the branched LP approaches, up to around 20% better in some radius-number of branches settings. We remark that Zhang (2020) provides theoretical guarantees for the tightness of the SDP relaxation under some technical conditions. However, since we find a strict increase in the certified percentages upon branching on the SDP, we conclude that such conditions for exactness may not be satisfied in general by practical networks, indicating that branching on the SDP may in fact be necessary in settings where high accuracy is the primary concern at hand. As we will see in the next experiment, this certification enhancement via SDP branching becomes even more substantial as the network depth increases. We now move



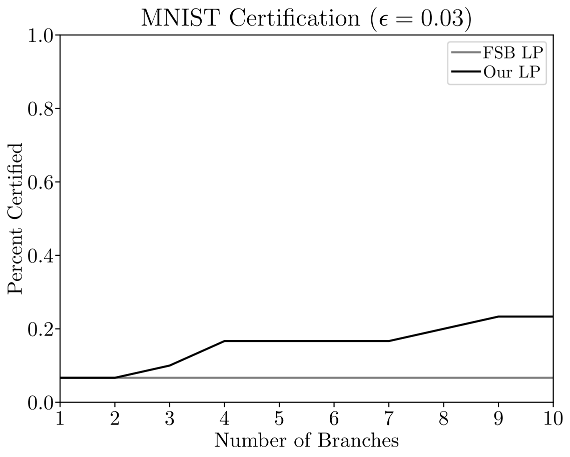
(a) Attack radius $\epsilon = 0.015$.



(b) Attack radius $\epsilon = 0.02$.



(c) Attack radius $\epsilon = 0.025$.



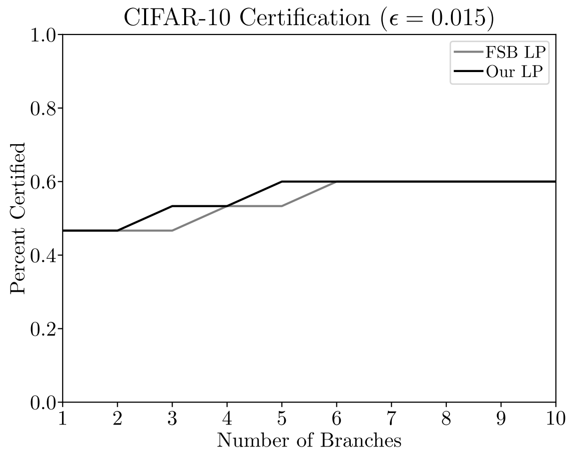
(d) Attack radius $\epsilon = 0.03$.

Figure 7: Percent certified on MNIST dataset using branching on LP relaxation of single-hidden layer network.

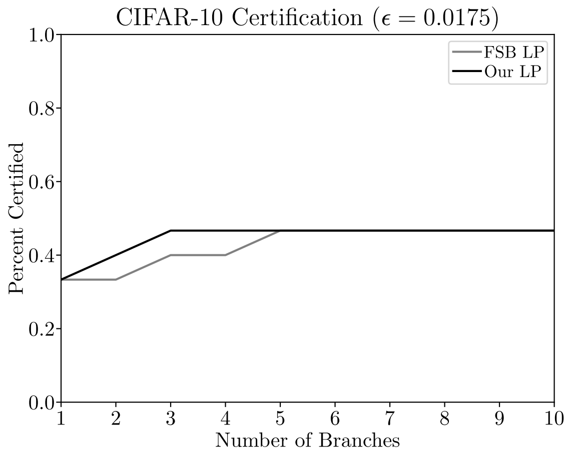
to this experiment, and propose a general rule of thumb for when LP, SDP, and their branch-and-bound variants are best applied based on depth and width of the network.

6.3 Effectiveness as Network Grows

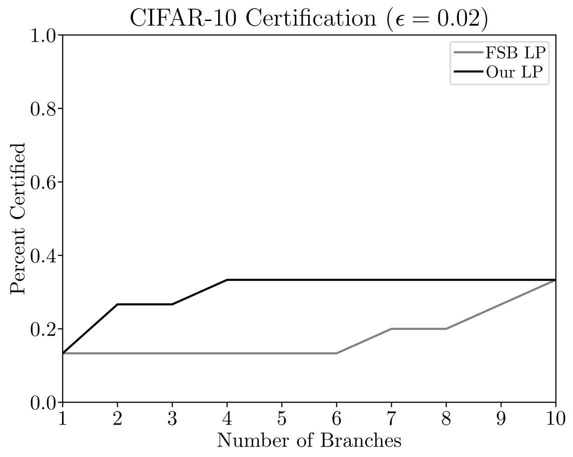
In this section, we perform two experiments to test the effectiveness of branching as the size and structure of the network changes. First, we consider two-layer networks of structure $n_x \times 100 \times 5$, where n_x is the input dimension. For each input size $n_x \in \{5, 10, 20, 40, 80, 100\}$, we generate one network with standard normal random weights, and another network with uniformly distributed weights (where each element is distributed uniformly on the interval $[0, 1]$). The weights are normalized according to Assumption 17. For each network being tested, we compute the LP, branched LP, SDP, and branched SDP relaxations at a fixed



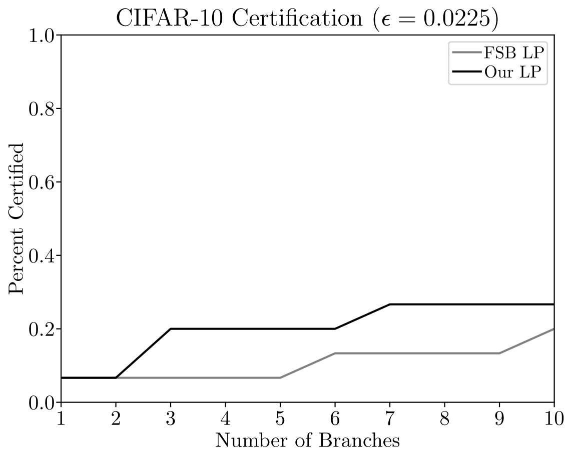
(a) Attack radius $\epsilon = 0.015$.



(b) Attack radius $\epsilon = 0.0175$.



(c) Attack radius $\epsilon = 0.02$.



(d) Attack radius $\epsilon = 0.0225$.

Figure 8: Percent certified on CIFAR-10 dataset using branching on LP relaxation of single-hidden layer network.

nominal input $\bar{x}$ using the input uncertainty set $\mathcal{X} = \{x \in \mathbb{R}^{n_x} : \|x - \bar{x}\|_\infty \leq \epsilon\}$ with $\epsilon = 0.5$. The optimal values, corresponding computation times, and percentage improvements induced by branching are reported in Table 2. The effectiveness of branching for the LP remains relatively constant between 5 and 10 percent improvement, whereas the branching appears to lose its efficacy on the SDP as the input size grows. As expected, the two-part partitioned convex relaxations take twice as long to solve as their unpartitioned counterparts. Note that, despite the fact that branching works better for the LP with wide networks, the actual optimal value of the SDP-based certificates are always lower (tighter) than the LP-based ones. This matches what is known in the literature: the SDP is a tighter relaxation technique than the LP (Raghunathan et al., 2018). However, the computation times of the SDP and branched SDP quickly increase as the network size increases, whereas the LP and branched LP computation times are seen to slowly increase. Indeed, SDP-based certification is known to suffer from scalability issues in high-dimensional settings (Chiu and Zhang, 2023). This limitation also applies to our branched SDP method as well. All of this suggests the following: in the regime of shallow (i.e., one or two hidden layers) but very wide networks, the branched LP should be used, since the branching remains effective in tightening the relaxation, yet the method is scalable to large networks where the SDP cannot be feasibly applied.

In the second simulation of this section, we analyze the effectiveness of branching as the depth of the network increases. In particular, we consider networks with normal random weights having 5 inputs and 5 outputs, and each intermediate layer having 10 neurons. We run the experiment on networks having 1 through 6 such intermediate layers. Recall that our LP branching scheme was derived for single-hidden layer networks. Therefore, we heuristically extend our LP branching scheme to multi-layer networks using the technique proposed in Algorithm 1. Unlike the branched LP, the SDP branching scheme given in Theorem 20 can directly be applied to deep networks, without the need to use surrogate parameters or other intermediate steps to compute the partition. We compute the LP, branched LP, SDP, and branched SDP on the networks at hand and report the objective values and computation times in Table 3. In this simulation, we see a stark contrast to the results in Table 2. Specifically, the percentage improvement induced by branching on the LP reduces quickly to nearly zero percent for networks with 3 or more intermediate 10-neuron hidden layers. Indeed, this is one fundamental drawback behind the LP relaxation: the convex upper envelope is used independently at every neuron, so the relaxation error quickly compounds as the network becomes deeper. On the other hand, the SDP relaxation takes into account the coupling between the layers of the network. This theoretical advantage is demonstrated empirically, as the percentage improvement gained by the branched SDP hovers around 10% even for the deep networks tested here. Moreover, note how the SDP computation time remains relatively close to that of the LP, unlike the rapid increase in computation time seen when increasing the input size. This behavior suggests the following: in the regime of deep but relatively narrow networks, the branched SDP should be used, since the branching is effective in tightening the relaxation, yet the computational cost grows relatively slowly as more layers are added (compared to the case where more inputs are added).

6.4 Large-Scale, Multi-Layer Certification Benchmarks

In this section, we compare LP branch-and-bound with our multi-layer branching heuristic (Algorithm 1) to the state-of-the-art deep neural network verifier, α, β-CROWN with k-FSB branching (a slight variant of filtered smart branching, introduced in De Palma et al. (2021)). To implement our method, we embed our branching heuristic into the open-source α, β-CROWN branch-and-bound verifier (Wang et al., 2021). We consider two image classification certification benchmarks: 1) `cifar10_resnet`, a benchmark from the α, β-CROWN (Wang et al., 2021) repository that was used in VNN-COMP 2021 (Bak et al., 2021), and 2) the `cifar2020` benchmark from VNN-COMP 2022 (Müller et al., 2022a). These large-scale problems involve deep neural networks with multiple layers and complex structures.

Table 2: Varying input size n_x for $n_x \times 100 \times 5$ ReLU network. Optimal values and corresponding computation times reported. B-LP and B-SDP correspond to branched LP and branched SDP, respectively. %-LP and %-SDP represent the percentage tightening of the optimal values obtained from branching.

(a) Normally distributed network weights.

Input size	LP	B-LP	%-LP	SDP	B-SDP	%-SDP
5	126.93	117.92	**7.10%**	16.82	14.83	**11.85%**
	0.71 s	1.46 s	104.18%	1.66 s	3.33 s	101.33%
10	187.57	176.19	**6.07%**	33.62	32.96	**1.98%**
	0.77 s	1.36 s	76.13%	1.54 s	3.16 s	105.57%
20	386.49	364.53	**5.68%**	54.02	54.01	**0.02%**
	0.71 s	1.42 s	100.49%	1.85 s	4.31 s	132.94%
40	874.70	864.56	**1.16%**	104.90	104.38	**0.49%**
	1.27 s	2.68 s	110.93%	4.79 s	9.33 s	95.01%
80	1591.41	1496.23	**5.98%**	310.37	310.31	**0.02%**
	1.76 s	2.97 s	69.00%	9.81 s	17.87 s	82.11%
100	2184.94	2175.87	**0.42%**	383.63	383.50	**0.03%**
	0.78 s	1.84 s	136.93%	5.02 s	10.52 s	109.46%

(b) Uniformly distributed network weights.

Input size	LP	B-LP	%-LP	SDP	B-SDP	%-SDP
5	11.65	10.69	**8.31%**	5.95	5.74	**3.44%**
	0.65 s	1.36 s	109.54%	1.39 s	2.20 s	58.32%
10	34.13	34.13	**0.00%**	12.61	11.92	**5.47%**
	0.68 s	1.36 s	101.35%	1.32 s	2.48 s	87.45%
20	83.74	83.02	**0.86%**	19.20	19.00	**1.06%**
	0.67 s	1.40 s	106.88%	1.31 s	2.85 s	118.39%
40	141.37	133.30	**5.71%**	25.89	25.69	**0.74%**
	0.69 s	1.43 s	106.67%	1.63 s	3.23 s	97.62%
80	260.80	242.19	**7.14%**	21.86	21.68	**0.84%**
	0.71 s	1.42 s	99.25%	2.82 s	5.44 s	92.97%
100	400.73	387.24	**3.37%**	102.87	102.35	**0.51%**
	0.74 s	1.56 s	111.10%	3.33 s	6.89 s	106.64%

Table 4 shows that our method yields performance comparable to that of the baseline α, β-CROWN with k-FSB branching. We remark that the differences in computation time are negligible. Indeed, the implementation of either branching heuristic only requires the computation of the preactivation bounds $l^{[k]}, u^{[k]}$ and an evaluation of a simple branching score function at each neuron (with k-FSB depending on one additional backwards pass through an auxiliary variable that our method does not require). Overall, our experiments

Table 3: Varying number of hidden layers for a $5 \times 10 \times 10 \times \cdots \times 10 \times 5$ ReLU network with normal random weights. Optimal values and corresponding computation times reported. B-LP and B-SDP correspond to branched LP and branched SDP, respectively. %-LP and %-SDP represent the percentage tightening of the optimal values obtained from branching.

Layers	LP	B-LP	%-LP	SDP	B-SDP	%-SDP
1	10.16	7.03	**30.79**%	4.70	4.65	**1.12**%
	0.59 s	1.21 s	105.06%	0.68 s	1.29 s	91.17%
2	46.29	44.89	**3.03**%	2.42	1.94	**19.94**%
	0.62 s	1.21 s	93.07%	0.71 s	1.49 s	108.81%
3	626.96	626.96	**0.00**%	36.29	34.36	**5.31**%
	0.61 s	1.29 s	110.86%	0.72 s	1.47 s	103.32%
4	5229.32	5229.32	**0.00**%	179.79	167.34	**6.93**%
	0.65 s	1.29 s	97.47%	0.99 s	1.88 s	89.81%
5	37625.91	37625.86	**0.00**%	628.78	561.60	**10.68**%
	0.69 s	1.34 s	94.59%	1.13 s	2.04 s	80.35%
6	326743.55	326743.34	**0.00**%	3245.41	3050.69	**6.00**%
	0.75 s	1.35 s	79.44%	1.19 s	2.35 s	98.01%

suggest that LP branching scheme outperforms the state-of-the-art on small-to-moderately-sized single-hidden layer certification problems, and achieves comparable performance in large-scale deep neural network certification problems.

Table 4: Our multi-layer branching heuristic versus k-FSB branching in α, β-CROWN branch-and-bound verifier. Benchmarks include (a) `cifar10_resnet`, a residual neural network with 2 residual blocks, 5 convolutional layers, and 2 fully connected affine layers, and (b) `cifar2020`, a 4-layer ReLU network with 2 convolutional layers and 2 fully connected affine layers. The reported "Accuracy" is the ratio between the number of test inputs that are successfully certified and the total number of test inputs. "Time" is the average certification time per test input.

(a) `cifar10_resnet`

Method	Accuracy	Time (seconds)
Our LP	30/56	56.99
k-FSB	30/56	53.12

(b) `cifar2020`

Method	Accuracy	Time (seconds)
Our LP	60/72	28.40
k-FSB	61/72	33.11

7. Conclusions

In this paper, we propose intelligently designed closed-form branching schemes for linear programming (LP) and semidefinite programming (SDP) robustness certification methods of ReLU neural networks. The branching schemes are derived by minimizing the worst-case error induced by the corresponding convex relaxations on single-hidden layer networks, which is theoretically justified by showing that minimizing the true relaxation error is NP-hard. The proposed techniques are experimentally substantiated on benchmark datasets by demonstrating significant reduction in relaxation error for moderately-sized single-hidden layer models, and performance on par with the state-of-the-art on large-scale, deep models.

Acknowledgments

This work was supported in part by U.S. Army Research Laboratory and the U.S. Army Research Office under Grant W911NF2010219, and in part by ONR and NSF.

Appendix A. Proof of Proposition 3

Proof Assume that $\phi^*(\mathcal{X}) > \max_{j\in\{1,2,\dots,p\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)})$. Then,

$$\phi^*(\mathcal{X}) > \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j)}) \text{ for all } j \in \{1,2,\dots,p\}. \tag{43}$$

Let (x^*, z^*) denote an optimal solution to the unrelaxed problem (2), i.e., $x^* \in \mathcal{X}$, $z^* = f(x^*)$, and

$$c^\top z^* = \phi^*(\mathcal{X}). \tag{44}$$

Since $\cup_{j=1}^p \mathcal{X}^{(j)} = \mathcal{X}$, there exists $j^* \in \{1,2,\dots,p\}$ such that $x^* \in \mathcal{X}^{(j^*)}$. Since $x^* \in \mathcal{X}^{(j^*)}$ and $z^* = f(x^*)$, it holds that $(x^*, z^*) \in \mathcal{N}^{(j^*)}_{\mathrm{LP}}$, where $\mathcal{N}^{(j^*)}_{\mathrm{LP}}$ is the relaxed network constraint set defined by $\mathcal{X}^{(j^*)}$. Therefore,

$$c^\top z^* \leq \sup\{c^\top z : x \in \mathcal{X}^{(j^*)},\ (x,z) \in \mathcal{N}^{(j^*)}_{\mathrm{LP}}\} = \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j^*)}) < \phi^*(\mathcal{X}),$$

where the first inequality comes from the feasibility of (x^*, z^*) over the $j^{*\text{th}}$ subproblem and the final inequality is due to (43). This contradicts the optimality of (x^*, z^*) given in (44). Hence, (9) must hold. ■

Appendix B. Proof of Proposition 4

Proof Let $j \in \{1,2,\dots,p\}$. It will be shown that $\mathcal{N}^{(j)}_{\mathrm{LP}} \subseteq \mathcal{N}_{\mathrm{LP}}$. Let $(x,z) \in \mathcal{N}^{(j)}_{\mathrm{LP}}$. Define $u' = u^{(j)}$, $l' = l^{(j)}$, and

$$g(x) = u \odot (Wx - l) \oslash (u - l),$$
$$g'(x) = u' \odot (Wx - l') \oslash (u' - l').$$

Then, by letting $\Delta g(x) = g(x) - g'(x) = a \odot (Wx) + b$, where

$$\begin{aligned} a &= u \oslash (u-l) - u' \oslash (u'-l'), \\ b &= u' \odot l' \oslash (u'-l') - u \odot l \oslash (u-l), \end{aligned}$$

the following relations are derived for all $i \in \{1, 2, \dots, n_z\}$:

$$\begin{aligned} g_i^* &:= \inf_{\{x : l' \leq Wx \leq u'\}} (\Delta g(x))_i \geq \inf_{\{\hat{z} : l' \leq \hat{z} \leq u'\}} (a \odot \hat{z} + b)_i \\ &= \inf_{\{\hat{z}_i : l'_i \leq \hat{z}_i \leq u'_i\}} (a_i \hat{z}_i + b_i) = \begin{cases} a_i l'_i + b_i & \text{if } a_i \geq 0, \\ a_i u'_i + b_i & \text{if } a_i < 0. \end{cases} \end{aligned}$$

In the case that $a_i \geq 0$, we have that

$$g_i^* \geq a_i l'_i + b_i = \left(\frac{u_i}{u_i - l_i} - \frac{u'_i}{u'_i - l'_i} \right) l'_i + \left(\frac{u'_i l'_i}{u'_i - l'_i} - \frac{u_i l_i}{u_i - l_i} \right) = \frac{u_i}{u_i - l_i} (l'_i - l_i) \geq 0,$$

where the final inequality comes from the fact that $u \geq 0$, $l' \geq l$, and $u > l$. On the other hand, if $a_i < 0$, it holds that

$$\begin{aligned} g_i^* \geq a_i u'_i + b_i &= \left(\frac{u_i}{u_i - l_i} - \frac{u'_i}{u'_i - l'_i} \right) u'_i + \left(\frac{u'_i l'_i}{u'_i - l'_i} - \frac{u_i l_i}{u_i - l_i} \right) \\ &= \frac{u_i}{u_i - l_i} (u'_i - l_i) - u'_i = \frac{u'_i - u_i}{u_i - l_i} l_i \geq 0, \end{aligned}$$

where the final inequality comes from the fact that $u' \leq u$, $l \leq 0$, and $u > l$. Therefore,

$$g^* = (g_1^*, g_2^*, \dots, g_{n_z}^*) \geq 0,$$

which implies that $\Delta g(x) = g(x) - g'(x) \geq 0$ for all x such that $l^{(j)} = l' \leq Wx \leq u' = u^{(j)}$. Hence, since $(x, z) \in \mathcal{N}_{\text{LP}}^{(j)}$, it holds that $z \geq 0$, $z \geq Wx$, and

$$z \leq g'(x) \leq g(x) = u \odot (Wx - l) \oslash (u - l).$$

Therefore, we have that $(x, z) \in \mathcal{N}_{\text{LP}}$.

Since $\mathcal{X}^{(j)} \subseteq \mathcal{X}$ (by definition) and $\mathcal{N}_{\text{LP}}^{(j)} \subseteq \mathcal{N}_{\text{LP}}$, it holds that the solution to the problem over the smaller feasible set gives a lower bound to the original solution: $\hat{\phi}_{\text{LP}}^*(\mathcal{X}^{(j)}) \leq \hat{\phi}_{\text{LP}}^*(\mathcal{X})$. Finally, since j was chosen arbitrarily, this implies the desired inequality (10). ■

Appendix C. Proof of Proposition 10

Proof We prove the result by reducing an arbitrary instance of the Min-$\mathcal{K}$-Union problem to an instance of the optimal partitioning problem (29). The proof is broken down into steps. In Step 1, we introduce the Min-$\mathcal{K}$-Union problem. We then construct a specific neural network based on the parameters of the Min-$\mathcal{K}$-Union problem in Step 2. In Step 3, we construct the solution to the partitioned LP relaxation for our neural network in the case that the partition is performed along all input coordinates. In Step 4, we construct the solution to the partitioned LP relaxation in the case that only a subset of the input coordinates are partitioned. Finally, in Step 5, we show that the solution to the Min-$\mathcal{K}$-Union problem can be constructed from the solution to the optimal partitioning problem, i.e., by finding the best subset of coordinates to partition along in the fourth step. As a consequence, we show that optimal partitioning is NP-hard.

Step 1: Arbitrary Min-$\mathcal{K}$-Union Problem. Suppose that we are given an arbitrary instance of the Min-$\mathcal{K}$-Union problem, i.e., a finite number of finite sets $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n$ and a positive integer $\mathcal{K} \leq n$. Since each set $\mathcal{S}_j$ is finite, the set $\bigcup_{j=1}^n \mathcal{S}_j$ is finite with cardinality $m := \left|\bigcup_{i=j}^n \mathcal{S}_j\right| \in \mathbb{N}$. Therefore, there exists a bijection between the elements of $\bigcup_{j=1}^n \mathcal{S}_j$ and the set $\{1, 2, \dots, m\}$. Hence, without loss of generality, we assume $\mathcal{S}_j \subseteq \mathbb{N}$ for all $j \in \{1, 2, \dots, n\}$ such that $\bigcup_{j=1}^n \mathcal{S}_j = \{1, 2, \dots, m\}$. In this Min-$\mathcal{K}$-Union problem, the objective is to find $\mathcal{K}$ sets $\mathcal{S}_{j_1}, \mathcal{S}_{j_2}, \dots, \mathcal{S}_{j_\mathcal{K}}$ among the collection of n given sets such that $\left|\bigcup_{i=1}^{\mathcal{K}} \mathcal{S}_{j_i}\right|$ is minimized over all choices of $\mathcal{K}$ sets. In what follows, we show that the solution to this problem can be computed by solving a particular instance of the optimal partitioning problem (29).

Step 2: Neural Network Construction. Consider a 3-layer ReLU network, where $x^{[0]}, x^{[1]} \in \mathbb{R}^n$ and $x^{[2]}, x^{[3]} \in \mathbb{R}^m$. Let the weight vector on the output be $c = \mathbf{1}_m$. Take the input uncertainty set to be $\mathcal{X} = [-1, 1]^n$. Let $W^{[0]} = I_n$ and $W^{[2]} = I_m$. In addition, construct the weight matrix on the first layer to be $W^{[1]} \in \mathbb{R}^{m \times n}$ such that

$$W_{ij}^{[1]} = \begin{cases} 1 & \text{if } i \in \mathcal{S}_j, \\ 0 & \text{otherwise.} \end{cases}$$

We remark that, since all entries of $c = \mathbf{1}_m$, $W^{[0]} = I_n$, $W^{[1]}$, and $W^{[2]} = I_m$ are nonnegative, the optimal value of the unrelaxed certification problem (25) is $\phi^*(\mathcal{X}) = \mathbf{1}_m^\top W^{[1]} \mathbf{1}_n$.

To finish defining the network and its associated LP relaxations, we must specify the preactivation bounds at each layer. Since all weights of the neural network are nonnegative, the largest preactivation at each layer is attained when the input is $x^{[0]} = \mathbf{1}_n$, the element-wise maximum vector in $\mathcal{X}$. The preactivations corresponding to this input are $\hat{z}^{[1]} = \mathbf{1}_n$, $\hat{z}^{[2]} = W^{[1]}\mathbf{1}_n$, and $\hat{z}^{[3]} = W^{[1]}\mathbf{1}_n$. Therefore, setting

$$\begin{aligned} u^{[1]} &= 2\mathbf{1}_n, \\ u^{[2]} &= \frac{3}{2} W^{[1]} \mathbf{1}_n, \\ u^{[3]} &= \frac{5}{4} W^{[1]} \mathbf{1}_n + \frac{1}{8} \mathbf{1}_m, \end{aligned}$$

we obtain valid preactivation upper bounds. Similarly, taking

$$l^{[k]} = -u^{[k]}$$

for all $k \in \{1, 2, 3\}$ defines valid preactivation lower bounds.

Step 3: Densely Partitioned LP Relaxation. With the network parameters defined, we now consider the first variant of our partitioned LP relaxation. In particular, we consider the relaxation (28) where all coordinates of the first layer are partitioned. We denote by $\bar{\phi}(\mathcal{X})$ the optimal objective value of this problem:

$$\bar{\phi}(\mathcal{X}) = \max_{j' \in \{1,2,\dots,2^n\}} \hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j')}), \tag{45}$$

where $\hat{\phi}^*_{\mathrm{LP}}(\mathcal{X}^{(j')})$ denotes the optimal objective value of

$$\begin{aligned} \text{maximize} \quad & c^\top x^{[3]} \\ \text{subject to} \quad & x^{[0]} \in \mathcal{X}^{(j')}, \\ & x^{[k+1]} \geq W^{[k]} x^{[k]}, && k \in \{0,1,2\}, \\ & x^{[k+1]} \geq 0, && k \in \{0,1,2\}, \\ & x^{[k+1]} \leq u^{[k+1]} \odot (W^{[k]} x^{[k]} - l^{[k+1]}) \oslash (u^{[k+1]} - l^{[k+1]}), && k \in \{0,1,2\}, \\ & x^{[1]} = \mathrm{ReLU}(W^{[0]} x^{[0]}). \end{aligned} \tag{46}$$

This problem serves as a baseline; this is the tightest LP relaxation of the certification problem among all those with partitioning along the input coordinates. (Recall that the final equality constraint in (46) is linear over the restricted feasible set $\mathcal{X}^{(j')}$.)

We will now show that an optimal solution $\bar{x} = (\bar{x}^{[0]}, \bar{x}^{[1]}, \bar{x}^{[2]}, \bar{x}^{[3]})$ of the partitioned LP defined by (45) and (46) can be taken to satisfy

$$\bar{x}^{[3]} = \frac{5}{4} W^{[1]} \mathbf{1}_n + \frac{1}{16} \mathbf{1}_m.$$

To see this, note that since all weights of the network and optimization (45) are nonnegative, the optimal activations $\bar{x}$ will be as large as possible in all coordinates and at all layers. Therefore, since the input is constrained to $\mathcal{X} = [-1,1]^n$, the optimal input for (45) is $\bar{x}^{[0]} = \mathbf{1}_n$. Since the ReLU constraint in (46) is exact, this implies that the optimal activation over this part at the first layer is

$$\bar{x}^{[1]} = \mathrm{ReLU}(W^{[0]} \bar{x}^{[0]}) = \mathrm{ReLU}(\mathbf{1}_n) = \mathbf{1}_n.$$

Now, for the second layer, the activation attains its upper bound. Since $u^{[2]} = -l^{[2]} = \frac{3}{2} W^{[1]} \mathbf{1}_n$, this implies that

$$\begin{aligned} \bar{x}^{[2]} &= u^{[2]} \odot (W^{[1]} \bar{x}^{[1]} - l^{[2]}) \oslash (u^{[2]} - l^{[2]}) \\ &= u^{[2]} \odot (W^{[1]} \bar{x}^{[1]} + u^{[2]}) \oslash (2u^{[2]}) \\ &= \frac{1}{2} (W^{[1]} \bar{x}^{[1]} + u^{[2]}) \\ &= \frac{1}{2} \left(W^{[1]} \mathbf{1}_n + \frac{3}{2} W^{[1]} \mathbf{1}_n \right) \\ &= \frac{5}{4} W^{[1]} \mathbf{1}_n. \end{aligned}$$

Similarly, for the third layer, we find that the optimal activation attains its upper bound as well. Since $u^{[3]} = -l^{[3]} = \frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{1}{8}\mathbf{1}_m$ and $W^{[2]} = I_m$ this gives that

$$\begin{aligned}
\bar{x}^{[3]} &= u^{[3]} \odot (W^{[2]}\bar{x}^{[2]} - l^{[3]}) \oslash (u^{[3]} - l^{[3]}) \\
&= u^{[3]} \odot (\bar{x}^{[2]} + u^{[3]}) \oslash (2u^{[3]}) \\
&= \frac{1}{2}(\bar{x}^{[2]} + u^{[3]}) \\
&= \frac{1}{2}\left(\frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{1}{8}\mathbf{1}_m\right) \\
&= \frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{1}{16}\mathbf{1}_m,
\end{aligned}$$

as claimed in (3). It is easily verified that $\bar{x}$ as computed above satisfies all constraints of the problem (46) over the part of the partition containing $\bar{x}^{[0]} = \mathbf{1}_n$.

Step 4: Sparsely Partitioned LP Relaxation. We now introduce the second variant of the partitioned LP relaxation. In particular, let $\mathcal{J}_p \subseteq \{1, 2, \dots, n\}$ be an index set such that $|\mathcal{J}_p| = n_p = n - \mathcal{K}$. Denote the complement of $\mathcal{J}_p$ by $\mathcal{J}_p^c = \{1, 2, \dots, n\} \setminus \mathcal{J}_p$. We consider the partitioned LP defined in (28), which partitions along each coordinate in the index set $\mathcal{J}_p$. The optimal value of this problem is denoted by $\phi^*_{\mathcal{J}_p}(\mathcal{X})$, and we denote an optimal solution by $\hat{x} = (\hat{x}^{[0]}, \hat{x}^{[1]}, \hat{x}^{[2]}, \hat{x}^{[3]})$. We will compute $\hat{x}$ in three steps.

Step 4.1: Upper Bounding the Solution. We start by upper bounding the final layer activation of the solution. In particular, we claim that the optimal solution $\hat{x}$ satisfies

$$\hat{x}^{[3]} \leq t := u^{[3]} - \frac{1}{16}\mathbf{1}_{\mathcal{I}^c}. \tag{47}$$

where $\mathcal{I} = \bigcup_{j \in \mathcal{J}_p^c} S_j \subseteq \{1, 2, \dots, m\}$ and $\mathcal{I}^c = \{1, 2, \dots, m\} \setminus \mathcal{I}$. Since $\bar{x}^{[3]} = u^{[3]} - \frac{1}{16}\mathbf{1}_m$, the bound (47) is equivalent to

$$\hat{x}_i^{[3]} \leq t_i = \begin{cases} u_i^{[3]} & \text{if } i \in \mathcal{I}, \\ \bar{x}_i^{[3]} & \text{if } i \in \mathcal{I}^c, \end{cases} \tag{48}$$

for all $i \in \{1, 2, \dots, m\}$. We now prove the element-wise representation of the bound, (48).

First, by the feasibility of $\hat{x}$ and the definitions of $u^{[3]}, l^{[3]}$, it must hold for all $i \in \{1, 2, \dots, m\}$ that

$$\hat{x}_i^{[3]} \leq \frac{u_i^{[3]}}{u_i^{[3]} - l_i^{[3]}}(w_i^{[2]\top}\hat{x}^{[2]} - l_i^{[3]}) = \frac{1}{2}(w_i^{[2]\top}\hat{x}^{[2]} + u_i^{[3]}),$$

and also that

$$\hat{x}_i^{[3]} \geq w_i^{[2]\top}\hat{x}^{[2]}.$$

Combining these inequalities, we find that $\hat{x}_i^{[3]} \leq \frac{1}{2}(\hat{x}_i^{[3]} + u_i^{[3]})$, or, equivalently, that

$$\hat{x}_i^{[3]} \leq u_i^{[3]}.$$

This bound holds for all $i \in \{1, 2, \dots, m\}$, and therefore it also holds for $i \in \mathcal{I}$. This proves the first case in the bound (48).

We now prove the second case of the claimed upper bound. For this case, suppose $i \notin \mathcal{I}$. Then $i \notin S_j$ for all $j \in \mathcal{J}_p^c$, which implies that

$$W_{ij}^{[1]} = 0 \text{ for all } j \in \mathcal{J}_p^c,$$

by the definition of $W^{[1]}$. Therefore,

$$w_i^{[1]\top} \hat{x}^{[1]} = \sum_{j=1}^{n} W_{ij}^{[1]} \hat{x}_j^{[1]} = \sum_{j \in \mathcal{J}_p^c} W_{ij}^{[1]} \hat{x}_j^{[1]} + \sum_{j \in \mathcal{J}_p} W_{ij}^{[1]} \hat{x}_j^{[1]} = \sum_{j \in \mathcal{J}_p} W_{ij}^{[1]} \hat{x}_j^{[1]}.$$

Now, note that for $j \in \mathcal{J}_p$, the jth coordinate of the input is being partitioned, and therefore the optimal solution must satisfy

$$\hat{x}_j^{[1]} = \text{ReLU}(w_j^{[0]\top} \hat{x}^{[0]}) = \text{ReLU}(e_j^\top \hat{x}^{[0]}) = \text{ReLU}(\hat{x}_j^{[0]}) \leq 1,$$

since $\hat{x}_j^{[0]} \in [-1, 1]$. Therefore,

$$w_i^{[1]\top} \hat{x}^{[1]} \leq \sum_{j \in \mathcal{J}_p^c} W_{ij}^{[1]} \leq \sum_{j=1}^{n} W_{ij}^{[1]} = w_i^{[1]\top} \mathbf{1}_n.$$

It follows from the feasibility of $\hat{x}$ and the definitions of $u^{[2]}, l^{[2]}$ that

$$\begin{aligned}
\hat{x}_i^{[2]} &\leq \frac{u_i^{[2]}}{u_i^{[2]} - l_i^{[2]}} (w_i^{[1]\top} \hat{x}^{[1]} - l_i^{[2]}) = \frac{1}{2} (w_i^{[1]\top} \hat{x}^{[1]} + u_i^{[2]}) \\
&\leq \frac{1}{2} \left(w_i^{[1]\top} \mathbf{1}_n + \frac{3}{2} w_i^{[1]\top} \mathbf{1}_n \right) = \frac{5}{4} w_i^{[1]\top} \mathbf{1}_n = \bar{x}_i^{[2]},
\end{aligned}$$

where $\bar{x}$ is the solution computed for the densely partitioned LP relaxation in Step 3. Therefore, we conclude that for all $i \notin \mathcal{I}$, it holds that

$$\hat{x}_i^{[3]} \leq \frac{u_i^{[3]}}{u_i^{[3]} - l_i^{[3]}} (w_i^{[2]\top} \hat{x}^{[2]} - l_i^{[3]}) \leq \frac{u_i^{[3]}}{u_i^{[3]} - l_i^{[3]}} (w_i^{[2]\top} \bar{x}^{[2]} - l_i^{[3]}) = \bar{x}_i^{[3]},$$

by our previous construction of $\bar{x}^{[3]}$. Thus, we have proven the second case in (48) holds. Hence, the claimed bound (47) holds.

Step 4.2: Feasibility of Upper Bound. Let us define $x = (x^{[0]}, x^{[1]}, x^{[2]}, x^{[3]})$, a point in $\mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$, by

$$x^{[0]} = \mathbf{1}_n, \qquad x^{[1]} = \mathbf{1}_{\mathcal{J}_p} + \frac{5}{4} \mathbf{1}_{\mathcal{J}_p^c}, \qquad x^{[2]} = u^{[3]} - \frac{1}{8} \mathbf{1}_{\mathcal{I}^c}, \qquad x^{[3]} = u^{[3]} - \frac{1}{16} \mathbf{1}_{\mathcal{I}^c}.$$

Note that $x^{[3]}$ equals the upper bound $t = (t_1, t_2, \dots, t_m)$. We now show that x is feasible for the partitioned LP defined by (27) and (28).

First, the input uncertainty constraint is satisfied, since $x^{[0]} = \mathbf{1}_n \in \mathcal{X}^{(j')} \subseteq \mathcal{X}$ for some part $\mathcal{X}^{(j')}$. Next, the relaxed ReLU constraints at the first layer are satisfied, since

$$x^{[1]} = \mathbf{1}_{\mathcal{J}_p} + \frac{5}{4}\mathbf{1}_{\mathcal{J}_p^c} \geq 0, \qquad \text{(Layer 1 lower bound.)}$$

$$x^{[1]} - W^{[0]}x^{[0]} = \frac{1}{4}\mathbf{1}_{\mathcal{J}_p^c} \geq 0, \qquad \text{(Layer 1 lower bound.)}$$

$$x^{[1]} - u^{[1]} \odot (W^{[0]}x^{[0]} - l^{[1]}) \oslash (u^{[1]} - l^{[1]}) = -\frac{1}{2}\mathbf{1}_{\mathcal{J}_p} - \frac{1}{4}\mathbf{1}_{\mathcal{J}_p^c} \leq 0. \quad \text{(Layer 1 upper bound.)}$$

The relaxed ReLU constraints are also satisfied in the second layer, since

$$x^{[2]} = \frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{1}{8}\mathbf{1}_{\mathcal{I}} \geq 0, \qquad \text{(Layer 2 lower bound.)}$$

$$x^{[2]} - W^{[1]}x^{[1]} = \frac{1}{4}W^{[1]}\mathbf{1}_{\mathcal{J}_p} + \frac{1}{8}\mathbf{1}_{\mathcal{I}} \geq 0, \qquad \text{(Layer 2 lower bound.)}$$

$$x^{[2]} - u^{[2]} \odot (W^{[1]}x^{[1]} - l^{[2]}) \oslash (u^{[2]} - l^{[2]}) = \frac{1}{8}(\mathbf{1}_{\mathcal{I}} - W^{[1]}\mathbf{1}_{\mathcal{J}_p^c}) \leq 0.$$
$$\text{(Layer 2 upper bound.)}$$

The final inequality above follows from the fact that either $(\mathbf{1}_{\mathcal{I}})_i = 0$ or $(\mathbf{1}_{\mathcal{I}})_i = 1$. For coordinates i such that $(\mathbf{1}_{\mathcal{I}})_i = 0$, the inequality obviously holds. For coordinates i such that $(\mathbf{1}_{\mathcal{I}})_i = 1$, we know that $i \in \mathcal{I}$, implying that $i \in \mathcal{S}_j$ for some $j \in \mathcal{J}_p^c$. This in turn implies that $W_{ij}^{[1]} = 1$ for some $j \in \mathcal{J}_p^c$, and therefore $w_i^{[1]\top}\mathbf{1}_{\mathcal{J}_p^c} = \sum_{j \in \mathcal{J}_p^c} W_{ij}^{[1]} \geq 1 = (\mathbf{1}_{\mathcal{I}})_i$.

Continuing to check feasibility of x, the relaxed ReLU constraints in the final layer are also satisfied:

$$x^{[3]} = \frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{1}{16}\mathbf{1}_m + \frac{1}{16}\mathbf{1}_{\mathcal{I}} \geq 0, \qquad \text{(Layer 3 lower bound.)}$$

$$x^{[3]} - W^{[2]}x^{[2]} = \frac{1}{16}\mathbf{1}_{\mathcal{I}^c} \geq 0, \qquad \text{(Layer 3 lower bound.)}$$

$$x^{[3]} - u^{[3]} \odot (W^{[2]}x^{[2]} - l^{[3]}) \oslash (u^{[3]} - l^{[3]}) = 0 \leq 0. \qquad \text{(Layer 3 upper bound.)}$$

Hence, the relaxed ReLU constraints are satisfied at all layers. The only remaining constraints to verify are the exact ReLU constraints for the partitioned input indices $\mathcal{J}_p$. Indeed, for all $j \in \mathcal{J}_p$, we have that

$$x_j^{[1]} - \text{ReLU}(W^{[0]}x^{[0]})_j = (\mathbf{1}_{\mathcal{J}_p})_j + \frac{5}{4}(\mathbf{1}_{\mathcal{J}_p^c})_j - \text{ReLU}(\mathbf{1}_n)_j = 1 + 0 - 1 = 0.$$

Hence, the ReLU equality constraint is satisfied for all input coordinates in $\mathcal{J}_p$. Therefore, our proposed point x is feasible for (28).

Step 4.3: Solution to Sparsely Partitioned LP. As shown in the previous step, the proposed point $x = (x^{[0]}, x^{[1]}, x^{[2]}, x^{[3]})$ is feasible for the partitioned LP defined by (27) and (28). Recall from the upper bound (47) that our solution $\hat{x} = (\hat{x}^{[0]}, \hat{x}^{[1]}, \hat{x}^{[2]}, \hat{x}^{[3]})$ satisfies $\hat{x}^{[3]} \leq t$. The objective value of the feasible point x gives that

$$c^\top x^{[3]} = \sum_{i=1}^m x^{[3]} = \sum_{i=1}^m t_i \geq \sum_{i=1}^m \hat{x}_i^{[3]} = \phi_{\mathcal{J}_p}^*(\mathcal{X}).$$

Since $\phi^*_{\mathcal{J}_p}(\mathcal{X})$ is the maximum value of the objective for all feasible points, it must be that $c^\top x^{[3]} = \phi^*_{\mathcal{J}_p}(\mathcal{X})$. Hence, the point x is an optimal solution to (28). Therefore, we can write the final activation of our optimal solution $\hat{x}$ to (28) as

$$\hat{x}^{[3]} = x^{[3]} = t = u^{[3]} - \frac{1}{16}\mathbf{1}_{\mathcal{I}^c}. \tag{49}$$

Step 5: Min-$\mathcal{K}$-Union from Optimal Partition. With the solutions constructed in Steps 3 and 4, we compute the difference in the objective values between the two partitioned LP relaxations:

$$\begin{aligned}\phi^*_{\mathcal{J}_p}(\mathcal{X}) - \bar{\phi}(\mathcal{X}) &= c^\top \hat{x}^{[3]} - c^\top \bar{x}^{[3]} = c^\top \left(u^{[3]} - \frac{1}{16}\mathbf{1}_{\mathcal{I}^c} - \frac{5}{4}W^{[1]}\mathbf{1}_n - \frac{1}{16}\mathbf{1}_m\right) \\ &= c^\top \left(\frac{5}{4}W^{[1]}\mathbf{1}_n + \frac{1}{8}\mathbf{1}_m - \frac{1}{16}\mathbf{1}_{\mathcal{I}^c} - \frac{5}{4}W^{[1]}\mathbf{1}_n - \frac{1}{16}\mathbf{1}_m\right) = \frac{1}{16}c^\top(\mathbf{1}_m - \mathbf{1}_{\mathcal{I}^c}) \\ &= \frac{1}{16}c^\top \mathbf{1}_{\mathcal{I}} = \frac{1}{16}\sum_{i\in\mathcal{I}} 1 = \frac{1}{16}|\mathcal{I}| = \frac{1}{16}\left|\bigcup_{j\in\mathcal{J}_p^c} S_j\right|.\end{aligned}$$

Therefore,

$$\left|\bigcup_{j\in\mathcal{J}_p^c} \mathcal{S}_j\right| = 16\left(\phi^*_{\mathcal{J}_p}(\mathcal{X}) - \bar{\phi}(\mathcal{X})\right), \tag{50}$$

which holds for all partition index sets $\mathcal{J}_p \subseteq \{1, 2, \dots, n\}$ such that $|\mathcal{J}_p| = n_p = n - \mathcal{K}$.

Now, let $\mathcal{J}_p^*$ be an optimal partition, i.e., a solution to (29) with our specified neural network parameters. Then, by (50), we have $\left|\bigcup_{j\in(\mathcal{J}_p^*)^c} \mathcal{S}_j\right| = 16\left(\phi^*_{\mathcal{J}_p^*}(\mathcal{X}) - \bar{\phi}(\mathcal{X})\right) \leq 16\left(\phi^*_{\mathcal{J}_p}(\mathcal{X}) - \bar{\phi}(\mathcal{X})\right) = \left|\bigcup_{j\in\mathcal{J}_p^c} \mathcal{S}_j\right|$ for all $\mathcal{J}_p$ with $|\mathcal{J}_p| = n_p$. Since this holds for all $\mathcal{J}_p^c$ with $|\mathcal{J}_p^c| = n - n_p = \mathcal{K}$, this shows that the set $(\mathcal{J}_p^*)^c$ is an optimal solution to the Min-$\mathcal{K}$-Union problem specified at the beginning of the proof. Now, suppose the optimal partitioning problem in (29) could be solved for $\mathcal{J}_p^*$ in polynomial time. Then the optimal solution $(\mathcal{J}_p^*)^c$ to the Min-$\mathcal{K}$-Union problem is also computable in polynomial time. Since this holds for an arbitrary instance of the Min-$\mathcal{K}$-Union problem, this implies that the Min-$\mathcal{K}$-Union problem is polynomially solvable in general, which is a contradiction. Therefore, the problem (29) is NP-hard in general. ■

Appendix D. Proof of Proposition 11

Proof Let $j \in \{1, 2, \dots, p\}$. From the definition of the partition, it holds that $\mathcal{X}^{(j)} \subseteq \mathcal{X}$. What remains to be shown is that $\mathcal{N}^{(j)}_{\mathrm{SDP}} \subseteq \mathcal{N}_{\mathrm{SDP}}$.

Let $P \in \mathcal{N}_{\mathrm{SDP}}^{(j)}$. Define $u' = u^{(j)}$ and $l' = l^{(j)}$. Since $P \in \mathcal{N}_{\mathrm{SDP}}^{(j)}$, it follows that

$$\begin{aligned}
P_z &\geq 0, \\
P_z &\geq W P_x, \\
\operatorname{diag}(P_{zz}) &= \operatorname{diag}(W P_{xz}), \\
\operatorname{diag}(P_{xx}) &\leq (l' + u') \odot P_x - l' \odot u', \\
P_1 &= 1, \\
P &\succeq 0.
\end{aligned}$$

To show that $P \in \mathcal{N}_{\mathrm{SDP}}$, we should show that the above expressions imply that $\operatorname{diag}(P_{xx}) \leq (l+u) \odot P_x - l \odot u$. To do so, define $\Delta l_i \geq 0$ and $\Delta u_i \geq 0$ such that $l'_i = l_i + \Delta l_i$ and $u'_i = u_i - \Delta u_i$ for all $i \in \{1, 2, \dots, n_x\}$. Then we find that

$$\begin{aligned}
((l' + u') \odot P_x - l' \odot u')_i &= (l'_i + u'_i)(P_x)_i - l'_i u'_i \\
&= (l_i + u_i)(P_x)_i - l_i u_i + (\Delta l_i - \Delta u_i)(P_x)_i \\
&\quad - (\Delta l_i u_i - \Delta u_i l_i - \Delta u_i \Delta l_i) \\
&= ((l + u) \odot P_x - l \odot u)_i + (\Delta l_i - \Delta u_i)(P_x)_i \\
&\quad - (\Delta l_i u_i - \Delta u_i l_i - \Delta u_i \Delta l_i) \\
&= ((l + u) \odot P_x - l \odot u)_i + \Delta_i,
\end{aligned}$$

where $\Delta_i \coloneqq (\Delta l_i - \Delta u_i)(P_x)_i - (\Delta l_i u_i - \Delta u_i l_i - \Delta u_i \Delta l_i)$. Therefore, it suffices to prove that $\Delta_i \leq 0$ for all i. Since $-l_i u_i \geq -l'_i u'_i$ by definition, it holds that $\Delta l_i u_i - \Delta u_i l_i - \Delta u_i \Delta l_i \geq 0$. Thus, when $(\Delta l_i - \Delta u_i)(P_x)_i \leq 0$, it holds that $\Delta_i \leq 0$, as desired. On the other hand, suppose that $(\Delta l_i - \Delta u_i)(P_x)_i \geq 0$. Then we find two cases:

1. $(\Delta l_i - \Delta u_i) \geq 0$ and $(P_x)_i \geq 0$. In this case, the maximum value of $(\Delta l_i - \Delta u_i)(P_x)_i$ is $(\Delta l_i - \Delta u_i)u'_i$. Therefore, the maximum value of Δ_i is

$$\begin{aligned}
\Delta_i &= (\Delta l_i - \Delta u_i)u'_i - (\Delta l_i u_i - \Delta u_i l_i - \Delta u_i \Delta l_i) \\
&= \Delta l_i (u'_i - u_i) - \Delta u_i u'_i + \Delta u_i l_i + \Delta u_i \Delta l_i \\
&= \Delta l_i (-\Delta u_i) + \Delta u_i \Delta l_i - \Delta u_i u'_i + \Delta u_i l_i \\
&= -\Delta u_i u'_i + \Delta u_i l_i.
\end{aligned}$$

   Both of the two final terms are nonpositive, and therefore $\Delta_i \leq 0$.

2. $(\Delta l_i - \Delta u_i) \leq 0$ and $(P_x)_i \leq 0$. In this case, the maximum value of $(\Delta l_i - \Delta u_i)(P_x)_i$ is $(\Delta l_i - \Delta u_i)l'_i$. Therefore, the maximum value of Δ_i is

$$\begin{aligned}
\Delta_i &= (\Delta l_i - \Delta u_i)l'_i - (\Delta l_i u_i - \Delta u_i l_i - \Delta u_i \Delta l_i) \\
&= -\Delta u_i \Delta l_i + \Delta u_i \Delta l_i + \Delta l_i l'_i - \Delta l_i u_i \\
&= \Delta l_i l'_i - \Delta l_i u_i.
\end{aligned}$$

   Both of the two final terms are nonpositive, and therefore $\Delta_i \leq 0$.

Hence, we find that $(l' + u') \odot P_x - l' \odot u' \leq (l + u) \odot P_x - l \odot u$ for all $P \in \mathcal{N}_{\mathrm{SDP}}^{(j)}$, proving that $P \in \mathcal{N}_{\mathrm{SDP}}$, and therefore $\mathcal{N}_{\mathrm{SDP}}^{(j)} \subseteq \mathcal{N}_{\mathrm{SDP}}$.

Since $\mathcal{X}^{(j)} \subseteq \mathcal{X}$ and $\mathcal{N}_{\mathrm{SDP}}^{(j)} \subseteq \mathcal{N}_{\mathrm{SDP}}$, it holds that the solution to the problem over the smaller feasible set lower bounds the original solution: $\hat{\phi}^*_{\mathrm{SDP}}(\mathcal{X}^{(j)}) \leq \hat{\phi}^*_{\mathrm{SDP}}(\mathcal{X})$. Finally, since j was chosen arbitrarily, this implies the desired inequality (31). ■

Appendix E. Proof of Lemma 18

Proof Let $i, j \in \{1, 2, \dots, n\}$. Since P is positive semidefinite, the 2nd-order principal minor $P_{ii}P_{jj} - P_{ij}^2$ is nonnegative, and therefore

$$|P_{ij}| \leq \sqrt{P_{ii}P_{jj}}. \tag{51}$$

Furthermore, by the basic inequality that $2ab \leq a^2 + b^2$ for all $a, b \in \mathbb{R}$, we have that $\sqrt{P_{ii}P_{jj}} \leq \frac{1}{2}(P_{ii} + P_{jj})$. Substituting this inequality into (51) gives the desired bound. ■

References

Brendon G. Anderson and Somayeh Sojoudi. Data-driven certification of neural networks with random input noise. *IEEE Transactions on Control of Network Systems*, 10(1): 249–260, 2022.

Brendon G. Anderson, Ziye Ma, Jingqi Li, and Somayeh Sojoudi. Tightened convex relaxations for neural network robustness certification. In *IEEE Conference on Decision and Control (CDC)*, pages 2190–2197, 2020.

Stanley Bak, Changliu Liu, and Taylor T. Johnson. The second international verification of neural networks competition (VNN-COMP 2021): Summary and results. *arXiv preprint arXiv:2109.00498*, 2021.

Mislav Balunovic, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin Vechev. Certifying geometric robustness of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15287–15297, 2019.

Ben Batten, Panagiotis Kouvaros, Alessio Lomuscio, and Yang Zheng. Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2184–2190, 2021.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The fourth international verification of neural networks competition (VNN-COMP 2023): Summary and results. *arXiv preprint arXiv:2312.16760*, 2023.

Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H.S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020.

Tong Chen, Jean B. Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic optimization for Lipschitz constants of ReLU networks. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19189–19200, 2020.

Hong-Ming Chiu and Richard Y. Zhang. Tight certification of adversarially trained neural networks via nonconvex low-rank semidefinite relaxations. In *International Conference on Machine Learning (ICML)*, pages 5631–5660, 2023.

Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip H.S. Torr, and M. Pawan Kumar. Improved branch and bound for neural network verification via Lagrangian decomposition. *arXiv preprint arXiv:2104.06718*, 2021.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic`.

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

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1632–1640, 2016.

Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 67(1):1–15, 2020.

Jean-Yves Franceschi, Alhussein Fawzi, and Omar Fawzi. Robustness of classifiers to uniform ℓ_p and Gaussian noise. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1280–1288, 2018.

Dorit S. Hochbaum. *Approximating Covering and Packing Problems: Set Cover, Vertex Cover, Independent Set, and Related Problems*, pages 94–143. PWS Publishing Co., USA, 1996. ISBN 0534949681.

Ming Jin, Javad Lavaei, Somayeh Sojoudi, and Ross Baldick. Boundary defense against cyber threat for power system state estimation. *IEEE Transactions on Information Forensics and Security*, 16:1752–1767, 2020.

Ming Jin, Heng Chang, Wenwu Zhu, and Somayeh Sojoudi. Power up! Robust graph convolutional network via graph powering. In *AAAI Conference on Artificial Intelligence*, pages 8004–8012, 2021.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Weicong Kong, Zhao Yang Dong, Youwei Jia, David J. Hill, Yan Xu, and Yuan Zhang. Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Transactions on Smart Grid*, 10(1):841–851, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Yann LeCun. The MNIST database of handwritten digits, 1998. URL `http://yann.lecun.com/exdb/mnist/`.

Jingyue Lu and M. Pawan Kumar. Neural network branching for neural network verification. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The third international verification of neural networks competition (VNN-COMP 2022): Summary and results. *arXiv preprint arXiv:2212.10376*, 2022a.

Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. Prima: General and precise neural network certification via scalable convex hull approximations. *Proceedings of the ACM on Programming Languages*, 6(43):1–33, 2022b.

K. Muralitharan, Rathinasamy Sakthivel, and R. Vishnuvarthan. Neural network based optimization approach for energy demand prediction in smart grid. *Neurocomputing*, 273:199–208, 2018.

Xiang Pan, Tianyu Zhao, and Minghua Chen. DeepOPF: Deep neural network for DC optimal power flow. In *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2019.

Aditi Raghunathan, Jacob Steinhardt, and Percy S. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10877–10887, 2018.

Vicenc Rubies Royo, Roberto Calandra, Dusan M. Stipanovic, and Claire Tomlin. Fast neural network verification via shadow prices. *arXiv preprint arXiv:1902.07247*, 2019.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Christian Tjandraatmadja, Ross Anderson, Joey Huchette, Will Ma, Krunal Kishor Patel, and Juan Pablo Vielma. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21675–21686, 2020.

Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In Jennifer Dy and Andreas Krause, editors, *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 5276–5285, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR.

Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In *International Conference on Machine Learning (ICML)*, pages 6727–6736, 2019.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, pages 5286–5295, 2018.

Bichen Wu, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 129–137, 2017.

Haoze Wu, Alex Ozdemir, Aleksandar Zeljic, Kyle Julian, Ahmed Irfan, Divya Gopinath, Sadjad Fouladi, Guy Katz, Corina Pasareanu, and Clark Barrett. Parallelization techniques for verifying neural networks. In *International Conference on Formal Methods In Computer-Aided Design*, 2020.

Weiming Xiang and Taylor T. Johnson. Reachability analysis and safety verification for neural network control systems. *arXiv preprint arXiv:1805.09944*, 2018.

Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations (ICLR)*, 2021.

Huan Zhang, Lily Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4939–4948, 2018.

Richard Y. Zhang. On the tightness of semidefinite relaxations for certifying robustness to adversarial examples. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.