Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming

Mahyar Fazlyab[†], Manfred Morari, George J. Pappas

Abstract—Certifying the safety or robustness of neural networks against input uncertainties and adversarial attacks is an emerging challenge in the area of safe machine learning and control. To provide such a guarantee, one must be able to bound the output of neural networks when their input changes within a bounded set. In this paper, we propose a semidefinite programming (SDP) framework to address this problem for feed-forward neural networks with general activation functions and input uncertainty sets. Our main idea is to abstract various properties of activation functions (e.g., monotonicity, bounded slope, bounded values, and repetition across layers) with the formalism of quadratic constraints. We then analyze the safety properties of the abstracted network via the S-procedure and semidefinite programming. Our framework spans the trade-off between conservatism and computational efficiency and applies to problems beyond safety verification. We evaluate the performance of our approach via numerical problem instances of various sizes.

Index Terms—Deep Neural Networks, Robustness Analysis, Safety Verification, Convex Optimization, Semidefinite Programming.

I. Introduction

Neural networks have become increasingly effective at many difficult machine-learning tasks. However, the nonlinear and large-scale nature of neural networks makes them hard to analyze and, therefore, they are mostly used as black-box models without formal guarantees. In particular, neural networks are highly vulnerable to attacks, or more generally, uncertainty in their input. In the context of image classification, for example, neural networks can be easily deluded into changing their classification labels by slightly perturbing the input image [1]. Indeed, it has been shown that even imperceptible perturbations in the input of the state-of-the-art neural networks cause natural images to be misclassified with high probability [2]. Input perturbations can be either of an adversarial nature [3], or they could merely occur due to compression, resizing, and cropping [4]. These drawbacks limit the adoption of neural networks in safety-critical applications such as self-driving vehicles [5], aircraft collision avoidance procedures [6], speech recognition, and recognition of voice commands; see [7] for a survey.

Motivated by the serious consequences of the fragility of neural networks to input uncertainties or adversarial attacks, there has been an increasing effort in developing tools to measure or improve the robustness of neural networks. Many results focus on specific adversarial attacks and attempt to harden the network by, for example, crafting hard-to-classify examples [8]–[11]. Although these methods are scalable and work well in practice, they still suffer from false negatives. Safety-critical applications require provable robustness against any bounded variations in the input data. As a result, many tools have recently been used, adapted, or developed for this purpose, such as mixed-integer linear programming [12]–[15], convex relaxations and duality theory [16]–[18], Satisfiability Modulo Theory (SMT) [19], dynamical systems [20], [21], Abstract Interpretation [22], [23], interval-based methods [24]–[28]. All these works aim at bounding the worst-case value of a performance measure when their input is perturbed within a specified range.

Our contribution. In this paper, we develop a novel framework based on semidefinite programming (SDP) for safety verification and robustness analysis of neural networks against norm-bounded perturbations in their input. Our main idea is to abstract nonlinear activation functions of neural networks by the constraints they impose on the pre- and post- activation values. In particular, we describe various properties of activation functions using Quadratic Constraints (QCs), such as bounded slope, bounded values, monotonicity, and repetition across layers. Using this abstraction, any properties (e.g., safety or robustness) that we can guarantee for the abstracted network will automatically be satisfied by the original network as well. The quadratic form of these constraints allows us to formulate the verification problem as an SDP feasibility problem. Our main tool for developing the SDP is the $\mathcal{S}$-procedure from robust control [29], which allows us to reason about multiple quadratic constraints. Our framework has the following notable features:

- We use various forms of QCs to abstract any type of activation function.
- Our method can capture *the cross-coupling between neurons across different layers*, thereby reducing conservatism. This feature, which hinges on the assumption that the same activation function is used throughout the entire network (repetition across layers), becomes particularly effective for deep networks.
- We can control the trade-off between computational complexity and conservatism by systematically including or excluding different types of QCs.

In this paper, we focus on the neural network verification

problem (formally stated in §II-A) but the proposed framework (input-output characterization of neural networks via quadratic constraints) can be adapted to other problems such as sensitivity analysis of neural networks to input perturbations, output reachable set estimation, probabilistic verification, bounding the Lipschitz constant of neural networks, and closed-loop stability analysis.

A. Related Work

The performance of certification algorithms for neural networks can be measured along three axes. The first axis is the tightness of the certification bounds; the second axis is the computational complexity, and, the third axis is applicability across various models (e.g. different activation functions). These axes conflict. For instance, the conservatism of the verification algorithm is typically at odds with the computational complexity. The relative advantage of any of these algorithms is application-specific. For example, reachability analysis and safety verification applications call for less conservative algorithms, whereas in robust training, computationally fast algorithms are desirable [16], [24].

On the one hand, formal verification techniques such as Satisfiability Modulo (SMT) solvers [30]–[32], or integer programming approaches [14], [15] rely on combinatorial optimization to provide tight certification bounds for piecewise linear networks, whose complexity scales exponentially with the size of the network in the worst-case. A notable work to improve scalability is [15], where the authors do exact verification of piecewise-linear networks using mixed-integer programming with an order of magnitude reduction in computational cost via tight formulations for non-linearities and careful preprocessing.

On the other hand, certification algorithms based on continuous optimization are more scalable but less accurate. A notable work in this category is [16], in which the authors propose a linear-programming (LP) relaxation of piece-wise linear networks and provide upper bounds on the worst-case loss using weak duality. The main advantage of this work is that the proposed algorithm solely relies on forward- and back-propagation operations on a modified network, and thus is easily integrable into existing learning algorithms. In [33], the authors propose an SDP relaxation of one-layer sigmoid-based neural networks based on bounding the worst-case loss with a first-order Taylor expansion. The closest work to the present work is [34], in which the authors propose a semidefinite relaxation (SDR) for certifying robustness of piece-wise linear multi-layer neural networks. This relaxation is based on the so-called "lifting", where the original problem is embedded in a much larger space. This SDR approach provides tighter bounds than those of [16] but is less scalable. Finally, compared to the SDR method of [34], our SDP framework yield tighter bounds, especially for deep networks, and is not limited to ReLU networks. Parts of this work, specialized to probabilistic verification, have appeared in the conference paper [35].

The rest of the paper is organized as follows. In §II we formulate the safety verification problem and present the assumptions. In §III, we abstract the problem with Quadratic Constraints (QCs). In §IV we state our main results. In §V, we discuss further utilities of our framework beyond safety verification. In §VI we provide numerical experiments to evaluate the performance of our method and compare it with competing approaches. Finally, in §VII we draw conclusions.

B. Notation and Preliminaries

We denote the set of real numbers by $\mathbb{R}$, the set of nonnegative real numbers by $\mathbb{R}_+$, the set of real n-dimensional vectors by $\mathbb{R}^n$, the set of $m \times n$-dimensional matrices by $\mathbb{R}^{m \times n}$, the m-dimensional vector of all ones by 1_m, the $m \times n$-dimensional zero matrix by $0_{m \times n}$, and the n-dimensional identity matrix by I_n. We denote by $\mathbb{S}^n$, $\mathbb{S}^n_+$, and $\mathbb{S}^n_{++}$ the sets of n-by-n symmetric, positive semidefinite, and positive definite matrices, respectively. The p-norm ($p \geq 1$) is displayed by $\| \cdot \|_p \colon \mathbb{R}^n \to \mathbb{R}_+$. For $A \in \mathbb{R}^{m \times n}$, the inequality $A \geq 0$ is element-wise. For $A \in \mathbb{S}^n$, the inequality $A \succeq 0$ means A is positive semidefinite. For sets $\mathcal{I}$ and $\mathcal{J}$, we denote their Cartesian product by $\mathcal{I} \times \mathcal{J}$. The indicator function of a set $\mathcal{X}$ is defined as $\mathbf{1}_{\mathcal{X}}(x) = 1$ if $x \in \mathcal{X}$, and $\mathbf{1}_{\mathcal{X}}(x) = 0$ otherwise. For two matrices A, B of the same dimension, we denote their Hadamard product by $A \circ B$. A function $g \colon \mathbb{R}^n \to \mathbb{R}$ is α-convex ($0 \leq \alpha < \infty$) if $g - (\alpha/2)\| \cdot \|_2^2$ is convex. Furthermore, g is β-smooth ($0 \leq \beta < \infty$) if it is differentiable and $(\beta/2)\| \cdot \|_2^2 - g$ is convex. Finally, if g is α-convex and β-smooth, then

$$\frac{\alpha\beta}{\alpha + \beta}\|y - x\|_2^2 + \frac{1}{\alpha + \beta}\|\nabla g(y) - \nabla g(x)\|_2^2$$
$$\leq (\nabla g(y) - \nabla g(x))^\top (y - x),$$

for all $x, y \in \mathbb{R}^n$ [36, Theorem 2.1.12].

II. Safety Verification and Robustness Analysis of Neural Networks

A. Problem Statement

Consider the nonlinear vector-valued function $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ described by a multi-layer feed-forward neural network. Given a set $\mathcal{X} \subset \mathbb{R}^{n_x}$ of possible inputs (e.g., adversarial examples), the neural network maps $\mathcal{X}$ to an output set $\mathcal{Y}$ given by

$$\mathcal{Y} = f(\mathcal{X}) := \{y \in \mathbb{R}^{n_y} \mid y = f(x), \ x \in \mathcal{X}\}. \tag{1}$$

The desirable properties that we would like to verify can often be represented by a safety specification set $\mathcal{S}_y$ in the output space of the neural network. In this context, the network is safe if the output set lies within the safe region, i.e., if the inclusion $f(\mathcal{X}) \subseteq \mathcal{S}_y$ holds. Alternatively, we can define $\mathcal{S}_x := f^{-1}(\mathcal{S}_y)$ as the inverse image of $\mathcal{S}_y$ under f. Then, safety corresponds to the inclusion $\mathcal{X} \subseteq \mathcal{S}_x$.

Checking the condition $\mathcal{Y} \subseteq \mathcal{S}_y$, however, requires an exact computation of the nonconvex set $\mathcal{Y}$, which is very difficult. Instead, our interest is in finding a non-conservative over-approximation $\tilde{\mathcal{Y}}$ of $\mathcal{Y}$ and verifying the safety properties by checking the condition $\tilde{\mathcal{Y}} \subseteq \mathcal{S}_y$. This approach *detects all false negatives* but also produces false positives, whose rate depends on the tightness of the over-approximation–see Figure
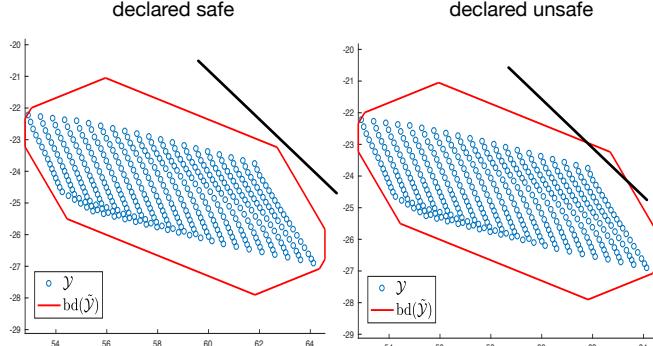
Fig. 1: The output set (in blue), the boundary of its over-approximation (in red), and the hyperplane characterizing the safe region (in black). Left: The network is deemed safe since $\tilde{\mathcal{Y}} \subseteq \mathcal{S}_y$. Right: The network is deemed unsafe since $\tilde{\mathcal{Y}} \not\subseteq \mathcal{S}_y$.

1. The goal of this paper is to solve this problem for a broad class of input uncertainties and safety specification sets using semidefinite programming.

1) Classification Example: Consider a data (e.g., image) classification problem with n_y classes, where a feed-forward neural network $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ takes as input a data point x and returns an n_y-dimensional vector of scores (or logits) – one for each class. The classification rule is based on assigning x to the class with the highest score. That is, the class of x is given by $C(x) = \mathrm{argmax}_{1 \leq i \leq n_y} f_i(x)$. To evaluate the local robustness of the neural network around a correctly-classified point $x^\star$, we consider a set $\mathcal{X}$, containing $x^\star$, that represents the set of all possible perturbations of $x^\star$. In image classification, a popular choice are perturbations in the ℓ_∞ norm, i.e., $\mathcal{X} = \{x : \|x - x^\star\|_\infty \leq \epsilon\}$, where ϵ is the maximum perturbation applied to each pixel. Then the classifier is locally robust at $x^\star$ if it assigns all the perturbed inputs to the same class as $x^\star$, i.e., if $C(x) = C(x^\star)$ for all $x \in \mathcal{X}$. For this problem, the safe set is the polytope $\mathcal{S}_y = \{y \in \mathbb{R}^{n_y} \mid y_{i^\star} \geq y_i \text{ for all } i \neq i^\star\}$, where $i^\star = \mathrm{argmax}_{1 \leq i \leq n_y} f_i(x^\star)$ is the class of $x^\star$.

B. Neural Network Model

For the model of the neural network, we consider an ℓ-layer feed-forward fully-connected neural network $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ described by the following recursive equations:

$$x^0 = x \tag{2}$$
$$x^{k+1} = \phi(W^k x^k + b^k) \quad k = 0, \cdots, \ell - 1$$
$$f(x) = W^\ell x^\ell + b^\ell,$$

where $x^0 = x \in \mathbb{R}^{n_0}(n_0 = n_x)$ is the input to the network and $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$, $b^k \in \mathbb{R}^{n_{k+1}}$ are the weight matrix and bias vector of the $(k + 1)$-th layer. Throughout, we denote by $n = \sum_{k=1}^\ell n_k$ the total number of neurons. The nonlinear activation function ϕ (ReLU[1], sigmoid, tanh, leaky ReLU, etc.) is applied coordinate-wise to the pre-activation vectors, i.e., it is of the form

$$\phi(x) := [\varphi(x_1) \ \cdots \ \varphi(x_{n_k})]^\top, \ x \in \mathbb{R}^{n_k}, \tag{3}$$

[1]Rectified Linear Unit.

where φ is the activation function of each neuron. The output $f(x)$ depends on the specific application we are considering. For example, in image classification with cross-entropy loss, $f(x)$ represents the logit input to the softmax function; or, in feedback control, x is the input to the neural network controller (e.g., tracking error) and $f(x)$ is the control input to the plant.

III. PROBLEM ABSTRACTION VIA QUADRATIC CONSTRAINTS

In this section, our goal is to provide an abstraction of the verification problem described in §II-A that can be converted into a semidefinite program. Our main tool is Quadratic Constraints (QCs), which were first developed in the context of robust control [37] for describing nonlinear, time-varying, or uncertain components of a system. We start with the abstraction of sets using quadratic constraints.

A. Input Set

We now provide a particular way of representing the input set $\mathcal{X}$ that will prove useful for developing the SDP.

Definition 1 *Let $\mathcal{X} \subset \mathbb{R}^{n_x}$ be a nonempty set. Suppose $\mathcal{P}_\mathcal{X}$ is the set of all symmetric indefinite matrices P such that*

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } x \in \mathcal{X}. \tag{4}$$

We then say that $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}_\mathcal{X}$.

Note that by definition, $\mathcal{P}_\mathcal{X}$ is a convex cone, i.e., if $P_1, P_2 \in \mathcal{P}_\mathcal{X}$ then $\theta_1 P_1 + \theta_2 P_2 \in \mathcal{P}_\mathcal{X}$ for all nonnegative scalars θ_1, θ_2. Furthermore, we can write

$$\mathcal{X} \subseteq \bigcap_{P \in \mathcal{P}_\mathcal{X}} \left\{ x \in \mathbb{R}^{n_x} : \begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \right\}. \tag{5}$$

In other words, we can over approximate $\mathcal{X}$ by the intersection of a possibly infinite number of sets defined by quadratic inequalities. We will see in §IV that the matrix $P \in \mathcal{P}_\mathcal{X}$ appears as a decision variable in the SDP. In this way, we can optimize the over-approximation of $\mathcal{X}$ to minimize the conservatism of the specific verification problem we want to solve.

Proposition 1 *(QC for hyper-rectangle) The hyper-rectangle $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid \underline{x} \leq x \leq \bar{x}\}$ satisfies the quadratic constraint defined by*

$$\mathcal{P}_\mathcal{X} = \left\{ P \mid P = \begin{bmatrix} -2\Gamma & \Gamma(\underline{x} + \bar{x}) \\ (\underline{x} + \bar{x})^\top \Gamma & -2\underline{x}^\top \Gamma \bar{x} \end{bmatrix} \right\}, \tag{6}$$

where $\Gamma \in \mathbb{R}^{n \times n}$ is diagonal and nonnegative. For this set, (5) holds with equality.

Proof 1 *See Appendix A.*

Our particular focus in this paper is on perturbations in the ℓ_∞ norm, $\mathcal{X} = \{x \mid \|x - x^\star\|_\infty \leq \epsilon\}$, which are a particular class of hyper-rectangles with $\underline{x} = x^\star - \epsilon 1$ and $\bar{x} = x^\star + \epsilon 1$. We can adapt the result of Proposition 1 to other sets such as polytopes, zonotopes, and ellipsoids, as outlined below. The derivation of the corresponding QCs can be found in Appendix B.

1) Polytopes: Let $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Hx \leq h\}$ be a polytope, where $H \in \mathbb{R}^{n_x \times m}, h \in \mathbb{R}^m$. Then $\mathcal{X}$ satisfies the QC defined by

$$\mathcal{P}_\mathcal{X} = \left\{ P \mid P = \begin{bmatrix} H^\top \Gamma H & -H^\top \Gamma h \\ -h^\top \Gamma H & h^\top \Gamma h \end{bmatrix} \right\}, \qquad (7)$$

where $\Gamma \in \mathbb{S}^m, \Gamma \geq 0, \Gamma_{ii} = 0$. Furthermore, if the set $\{x \in \mathbb{R}^{n_x} \mid Hx > h\}$ is empty, then (5) holds with equality.

2) Zonotopes: A zonotope is an affine transformation of the unit cube, $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid x = x_c + A\lambda, \quad \lambda \in [0,1]^m\}$, where $A \in \mathbb{R}^{n_x \times m}$ and $x_c \in \mathbb{R}^{n_x}$. Then any $P \in \mathcal{P}_\mathcal{X}$ satisfies

$$\begin{bmatrix} A & x_c \\ 0 & 1 \end{bmatrix}^\top P \begin{bmatrix} A & x_c \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 2\Gamma & -\Gamma 1_m \\ -1_m^\top \Gamma & 0 \end{bmatrix} \succeq 0, \quad (8)$$

for some diagonal and nonnegative $\Gamma \in \mathbb{R}^{m \times m}$.

3) Ellipsoids: Suppose the input set $\mathcal{X}$ is an ellipsoid defined by $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid \|Ax + b\|_2 \leq 1\}$, where $A \in \mathbb{S}^{n_x}$ and $b \in \mathbb{R}^{n_x}$. Then

$$\mathcal{P}_\mathcal{X} = \left\{ P \mid P = \mu \begin{bmatrix} -A^\top A & -A^\top b \\ -b^\top A & 1 - b^\top b \end{bmatrix}, \ \mu \geq 0 \right\}. \quad (9)$$

B. Safety Specification Set

As mentioned in the introduction, the safe set can be characterized either in the output space ($\mathcal{S}_y$) or in the input space ($\mathcal{S}_x$). In this paper, we consider the latter. Specifically, we assume $\mathcal{S}_x$ can be represented (or inner approximated) by the intersection of finitely many quadratic inequalities:

$$\mathcal{S}_x = \bigcap_{i=1}^m \left\{ x \in \mathbb{R}^{n_x} \mid \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix}^\top S_i \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix} \leq 0 \right\}, \quad (10)$$

where the $S_i \in \mathbb{S}^{n_x + n_y + 1}$ are given. In particular, this characterization includes ellipsoids and polytopes in the output space. For instance, for an output safety specification set described by the polytope $\mathcal{S}_y = \cap_{i=1}^m \{y \in \mathbb{R}^{n_y} \mid c_i^\top y - d_i \leq 0\}$, the S_i's are given by

$$S_i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & c_i \\ 0 & c_i^\top & -2d_i \end{bmatrix}, \ i = 1, \cdots, m.$$

C. Abstraction of Nonlinearities by Quadratic Constraints

One of the main difficulties in the analysis of neural networks is the composition of nonlinear activation functions. To simplify the analysis, instead of analyzing the network directly, our main idea is to remove the nonlinear activation functions from the network but retain the constraints they impose on the pre- and post-activation signals. Using this abstraction, any properties (e.g., safety or robustness) that we can guarantee for the "constrained" network will automatically be satisfied by the original network as well. In the following, we show how we can encode various properties of activation functions (e.g., monotonicity, bounded slope, and bounded values) using quadratic constraints. We first provide a formal definition below.

Definition 2 (QC for functions) *Let $\phi \colon \mathbb{R}^n \to \mathbb{R}^n$ and suppose $\mathcal{Q}_\phi \subset \mathbb{S}^{2n+1}$ is the set of all symmetric indefinite matrices Q such that the inequality*

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0, \qquad (11)$$

holds for all $x \in \mathbb{R}^n$. Then we say ϕ satisfies the quadratic constraint defined by $\mathcal{Q}_\phi$.

We remark that our definition of a quadratic constraint slightly differs from the one used in robust control [37], by including a constant in the vector surrounding the matrix Q, which allows us to incorporate affine constraints (e.g., bounded nonlinearities). In view of Definition 1, we can interpret the quadratic constraint satisfied by ϕ as a quadratic constraint satisfied by the graph of ϕ, $G(\phi) := \{(x,y) \mid y = \phi(x)\} \subset \mathbb{R}^{2n}$, i.e., $\mathcal{Q}_\phi = \mathcal{P}_{G(\phi)}$. Therefore, we can write

$$G(\phi) \subseteq \bigcap_{Q \in \mathcal{Q}_\phi} \left\{ (x,y) \in \mathbb{R}^{2n} \colon \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \geq 0 \right\}.$$

In other words, we over-approximate the graph of ϕ by a quadratically constrained set. The derivation of quadratic constraints is function specific but there are certain rules and heuristics that can be used for all of them which we describe below.

1) Sector-Bounded Nonlinearities: Consider the nonlinear function $\varphi \colon \mathbb{R} \to \mathbb{R}$ with $\varphi(0) = 0$. We say that φ is *sector-bounded* in the sector $[\alpha, \beta]$ ($\alpha \leq \beta < \infty$) if the following condition holds for all $x \in \mathbb{R}$,[2]

$$(\varphi(x) - \alpha x)(\varphi(x) - \beta x) \leq 0. \qquad (12)$$

Intuitively, this inequality means that the function $y = \varphi(x)$ lies in the sector formed by the lines $y = \alpha x$ and $y = \beta x$ (see Figure 2). As an example, the ReLU function $\max(0, x)$ belongs to the sector $[0, 1]$. In fact, it precisely lies on the boundary of the sector.

For the vector case, let $K_1, K_2 \in \mathbb{R}^{n \times n}$ be two matrices such that $K_2 - K_1$ is symmetric positive semidefinite. We say that $\phi \colon \mathbb{R}^n \to \mathbb{R}^n$ is sector-bounded in the sector $[K_1, K_2]$ if the following condition holds for all $x \in \mathbb{R}^n$ [38],

$$(\phi(x) - K_1 x)^\top (\phi(x) - K_2 x) \leq 0, \qquad (13)$$

or, equivalently,

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top \begin{bmatrix} -K_1^\top K_2 - K_2^\top K_1 & K_1^\top + K_2^\top & 0 \\ K_1 + K_2 & -2I_n & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0.$$

The sector condition does not impose any restriction on the slope of the function. This motivates a more accurate description of nonlinearities with bounded slope [39].

---

[2]For the case where $\alpha = -\infty$ or $\beta = +\infty$, we define the sector bound inequality as $x(\varphi(x) - \beta x) \leq 0$ and $x(\alpha x - \varphi(x)) \leq 0$, respectively.

$$\alpha \leq \frac{\varphi(x_2) - \varphi(x_1)}{x_2 - x_1} \leq \beta \qquad \alpha \leq \frac{\varphi(x)}{x} \leq \beta$$
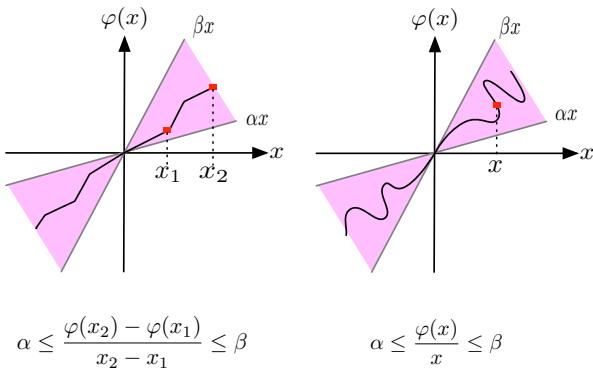
Fig. 2: A slope-restricted nonlinearity (left) and a sector-bounded nonlinearity (right).

2) Slope-Restricted Nonlinearities: A nonlinear function $\phi\colon \mathbb{R}^n \to \mathbb{R}^n$ is slope-restricted in the sector $[\alpha, \beta]$ ($\alpha \leq \beta < \infty$), if for any $x, x^\star \in \mathbb{R}^n$,

$$(\phi(x) - \phi(x^\star) - \alpha(x - x^\star))^\top (\phi(x) - \phi(x^\star) - \beta(x - x^\star)) \leq 0. \tag{14}$$

For the one-dimensional case ($n = 1$), the slope restriction condition in (14) states that the chord connecting any two points on the curve of ϕ has a slope that is at least α and at most β:

$$\alpha \leq \frac{\phi(x) - \phi(x^\star)}{x - x^\star} \leq \beta \quad \forall x, x^\star \in \mathbb{R}.$$

Note that a slope-restricted nonlinearity with $\phi(0) = 0$ is also sector bounded. Furthermore, if ϕ is slope-restricted in $[\alpha, \beta]$, then the function $x \mapsto \phi(x + x^\star) - \phi(x^\star)$ belongs to the sector $[\alpha I_n, \beta I_n]$ for any $x^\star$. Finally, the gradient of an α-convex and β-smooth function is slope-restricted in $[\alpha, \beta]$.

To connect the results of the previous subsection to activation functions in neural networks, we recall the following result from convex analysis [36].

Lemma 1 (gradient of convex functions) *Consider a function* $g\colon \mathbb{R}^n \to \mathbb{R}$ *that is* α-convex *and* β-smooth. *Then the gradient function* $\nabla g\colon \mathbb{R}^n \to \mathbb{R}^n$ *is slope-restricted in the sector* $[\alpha, \beta]$.

Notably, all commonly-used activation functions for deep neural networks are gradients of convex functions. Therefore, they belong to the class of slope-restricted nonlinearities, according to Lemma 1. We have the following result.

Proposition 2 *The following statements hold true.*

(a) The ReLU function $\varphi(x) = \max(0, x)$, $x \in \mathbb{R}$ is slope-restricted and sector-bounded in $[0, 1]$.
(b) The sigmoid function, $\varphi(x) = \frac{1}{1 + e^{-x}}$, $x \in \mathbb{R}$ is slope-restricted in $[0, 1]$.
(c) The tanh function, $\varphi(x) = \tanh(x)$, $x \in \mathbb{R}$ is slope-restricted in $[0, 1]$.
(d) The leaky ReLU function, $\varphi(x) = ax\mathbb{I}(x < 0) + x\mathbb{I}(x \geq 0)$, $x \in \mathbb{R}$ with $a > 0$ is slope-restricted and sector-bounded in $[\min(a, 1), \max(a, 1)]$.

(e) The exponential linear function (ELU), $\varphi(x) = x\mathbb{I}(x \geq 0) + a(e^x - 1)\mathbb{I}(x < 0)$, $x \in \mathbb{R}$ with $a > 0$ is slope-restricted and sector-bounded in $[0, 1]$.
(f) The softmax function, $\phi(x) = [\frac{e^{x_1}}{\sum_{i=1}^d e^{x_i}}, \cdots, \frac{e^{x_n}}{\sum_{i=1}^d e^{x_i}}]^\top$, $x \in \mathbb{R}^n$ is slope-restricted in $[0, 1]$.

In the context of neural networks, our interest is in *repeated nonlinearities* of the form $\phi(x) = [\varphi(x_1) \cdots \varphi(x_n)]^\top$. Furthermore, the activation values might be bounded from below or above (e.g., the ReLU function which outputs a nonnegative value). The sector bound and slope restricted inequalities can become too conservative as they do not capture these properties. In the following, we discuss QCs for these properties.

3) Repeated Nonlinearities: Suppose $\varphi\colon \mathbb{R} \to \mathbb{R}$ is slope-restricted in $[\alpha, \beta]$ ($\alpha \leq \beta$) and let $\phi(x) = [\varphi(x_1) \cdots \varphi(x_n)]^\top$ be a vector-valued function constructed by component-wise repetition of φ. It is not hard to verify that ϕ is also slope-restricted in $[\alpha, \beta]$. Indeed, by summing the slope-restriction conditions

$$(\varphi(x_i) - \varphi(x_i^\star) - \alpha(x_i - x_i^\star))(\varphi(x_i) - \varphi(x_i^\star) - \beta(x_i - x_i^\star)) \leq 0.$$

over $i = 1, \cdots, n$, we obtain (14). However, this representation simply ignores the fact that all the nonlinearities that compose ϕ are the same. By taking advantage of this structure, we can refine the quadratic constraint that describes ϕ. To be specific, for an input-output pair $(x, \phi(x))$, $x \in \mathbb{R}^n$, we can write the inequality

$$(\varphi(x_i) - \varphi(x_j) - \alpha(x_i - x_j))(\varphi(x_i) - \varphi(x_j) - \beta(x_i - x_j)) \leq 0, \tag{15}$$

for all distinct $i, j = 1, \cdots, n$, $i \neq j$. This particular QC can considerably reduce conservatism, especially for deep networks, as it reasons about *the coupling between the neurons throughout the entire network*. By making an analogy to dynamical systems, we can interpret the neural network as a time-varying discrete-time dynamical system where the same nonlinearity is repeated for all "time" indexes k (the layer number). Then the QC in (15) couples all the possible neurons. In the following lemma, we characterize QCs for repeated nonlinearities.

Lemma 2 (QC for repeated nonlinearities) *Suppose* $\varphi\colon \mathbb{R} \to \mathbb{R}$ *is slope-restricted on* $[\alpha, \beta]$. *Then the vector-valued function* $\phi(x) = [\varphi(x_1) \; \varphi(x_2) \; \cdots \varphi(x_n)]^\top$ *satisfies the quadratic constraint*

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top \begin{bmatrix} -2\alpha\beta T & (\alpha + \beta)T & 0 \\ (\alpha + \beta)T & -2T & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0, \tag{16}$$

for all $x \in \mathbb{R}^n$, *where*

$$T = \sum_{1 \leq i < j \leq n} \lambda_{ij}(e_i - e_j)(e_i - e_j)^\top, \quad \lambda_{ij} \geq 0, \tag{17}$$

and $e_i \in \mathbb{R}^n$ *is the i-th unit vector.*

Proof 2 *By a conic combination of $\binom{n}{2}$ quadratic constraints of the form* (15)*, we obtain* (16)*. See Appendix C for a detailed proof.*

There are several results in the literature about repeated nonlinearities. For instance, in [40], [41], the authors derive QCs for repeated and odd nonlinearities (e.g. tanh function).

4) Bounded Nonlinearities: Finally, suppose the nonlinear function values are bounded, i.e., $\underline{\phi} \leq \phi(x) \leq \bar{\phi}$ for all $x \in \mathbb{R}^n$. Using Proposition 1, $\phi(x)$ satisfies the quadratic constraint

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2D & D(\underline{\phi}+\bar{\phi}) \\ 0 & (\underline{\phi}+\bar{\phi})^\top D & -2\underline{\phi}^\top D\bar{\phi} \end{bmatrix} \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0, \quad (18)$$

for all x, where $D \in \mathbb{R}^{n \times n}$ is diagonal and nonnegative. We can write a similar inequality when the pre- activation values are known to be bounded. More generally, if the graph of ϕ is known to satisfy $G(\phi) \subseteq \mathcal{G}$, then any quadratic constraint for $\mathcal{G}$ is also a valid quadratic constraint for ϕ.

We observe that the inequalities (15)-(18) are all quadratic in $(x, \phi(x), 1)$, and therefore can be encapsulated into QCs of the form (11). As we show in §IV, the matrix $Q \in \mathcal{Q}_\phi$ that abstracts the nonlinearity ϕ appears as a decision variable in the SDP.

Although the above rules can be used to guide the search for valid QCs for activation functions, a less conservative description of activation functions requires a case-by-case treatment to further exploit the structure of the nonlinearity. Furthermore, the QC definition in (18) is global as it holds for the whole space $\mathbb{R}^n$. When restricted to a local region in $\mathbb{R}^n$, we can refine the QC characterization of the activation functions. In the next subsection, we elaborate on QCs for ReLU activation functions.

D. Quadratic Constraints for ReLU Activation Function

The ReLU function precisely lies on the boundary of the sector $[0, 1]$. This observation can be used to refine the QC description of ReLU. Specifically, let $y = \max(\alpha x, \beta x)$, $x \in \mathbb{R}^n$ be the concatenation of n ReLU activation functions[3]. Then each individual activation function can be described by the following constraints [34]:

$$y_i = \max(\alpha x_i, \beta y_i) \iff \begin{cases} (y_i - \alpha x_i)(y_i - \beta x_i) = 0 \\ \beta x_i \leq y_i \\ \alpha x_i \leq y_i. \end{cases} \quad (19)$$

The first constraint is the boundary of the sector $[\alpha, \beta]$ and the other constraints simply prune these boundaries to recover the ReLU function. Furthermore, for any two distinct indices $i \neq j$, we can write the constraint (15):

$$(y_j - y_i - \alpha(x_j - x_i))(y_j - y_i - \beta(x_j - x_i)) \leq 0. \quad (20)$$

By adding a weighted combination of all these constraints (non-negative weights for inequalities), we find that the function $y = \max(\alpha x, \beta x)$ satisfies

$$\sum_{i=1}^n \{\lambda_i(y_i - \alpha x_i)(y_i - \beta x_i) - \nu_i(y_i - \beta x_i) - \eta_i(y_i - \alpha x_i)\} + \\ \sum_{i \neq j} \lambda_{ij}(y_j - y_i - \alpha(x_j - x_i))(y_j - y_i - \beta(x_j - x_i)) \leq 0. \quad (21)$$

[3]For ReLU, we have $\alpha = 0$ and $\beta = 1$.

for all $x \in \mathbb{R}^n$. In the following lemma, we provide a full QC characterization of the ReLU function.

Lemma 3 (Global QC for ReLU function) *The* ReLU *function,* $\phi(x) = \max(\alpha x, \beta x)$, $x \in \mathbb{R}^n$ *satisfies the QC defined by*

$$\mathcal{Q}_\phi = \left\{ Q \in \mathbb{S}^{2n+1} \mid Q = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{12}^\top & Q_{22} & Q_{23} \\ Q_{13}^\top & Q_{23}^\top & Q_{33} \end{bmatrix} \right\}, \quad (22)$$

where

$$Q_{11} = -2\alpha\beta(\text{diag}(\lambda) + T), \ Q_{12} = (\alpha + \beta)(\text{diag}(\lambda) + T),$$
$$Q_{13} = -\beta\nu - \alpha\eta, \ Q_{22} = -2(\text{diag}(\lambda) + T),$$
$$Q_{23} = \nu + \eta, \ Q_{33} = 0,$$

and T is given by (17).

Proof 3 *See Appendix D.*

1) Tightening Relaxations: The QC of Lemma 3 holds globally for the whole space $\mathbb{R}^n$. When restricted to a local region $\mathcal{X}$, these QCs can be tightened. Specifically, suppose $y = \max(x, 0)$ and define $\mathcal{I}^+$, $\mathcal{I}^-$, and $\mathcal{I}^\pm$ as the set of activations that are known to be always active, always inactive, or unknown for all $x \in \mathcal{X} \subseteq \mathbb{R}^n$, i.e.,

$$\mathcal{I}^+ = \{i \mid x_i \geq 0 \text{ for all } x \in \mathcal{X}\} \quad (23)$$
$$\mathcal{I}^- = \{i \mid x_i < 0 \text{ for all } x \in \mathcal{X}\}$$
$$\mathcal{I}^\pm = \{1, \cdots, n\} \setminus (\mathcal{I}^+ \cup \mathcal{I}^-).$$

Then the function $y_i = \max(\alpha x_i, \beta x_i)$ belongs to the sector $[\alpha, \alpha]$, $[\alpha, \beta]$ and $[\beta, \beta]$ for inactive, unknown, and active neurons, respectively. Furthermore, since the constraint $y_i \geq \beta x_i$ holds with equality for active neurons, we can write $\nu_i \in \mathbb{R}$ if $i \in \mathcal{I}^+$, $\nu_i \geq 0$ otherwise. Similarly, the constraint $y_i \geq \alpha x_i$ holds with equality for inactive neurons. Therefore, we can write $\eta_i \in \mathbb{R}$ if $i \in \mathcal{I}^-$, $\eta_i \geq 0$ otherwise. Finally, the chord connecting the input-output pairs of always-active or always-inactive neurons has slope of α or β. Equivalently, for any $(i, j) \in (\mathcal{I}^+ \times \mathcal{I}^+) \cup (\mathcal{I}^- \times \mathcal{I}^-)$, we can write

$$(\frac{y_j - y_i}{x_j - x_i} - \alpha)(\frac{y_j - y_i}{x_j - x_i} - \beta) = 0.$$

Therefore, in (21), $\lambda_{ij} \in \mathbb{R}$ for $(i, j) \in (\mathcal{I}^+ \times \mathcal{I}^+) \cup (\mathcal{I}^- \times \mathcal{I}^-)$ and $\lambda_{ij} \geq 0$ otherwise. The above additional degrees of freedom on the multipliers can tighten the relaxation incurred in (21). In the following Lemma, we summarize the above observations.

Lemma 4 (*Local QC for* ReLU *function*) *Let* $\phi(x) = \max(\alpha x, \beta x)$, $x \in \mathcal{X} \subset \mathbb{R}^n$ *and define* $\mathcal{I}^+, \mathcal{I}^-$ *as in* (23). *Then ϕ satisfies the QC defined by*

$$\mathcal{Q}_\phi = \left\{ Q \in \mathbb{S}^{2n+1} \mid Q = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{12}^\top & Q_{22} & Q_{23} \\ Q_{13}^\top & Q_{23}^\top & Q_{33} \end{bmatrix} \right\}, \quad (24)$$

where

$$Q_{11} = -2\mathrm{diag}(\boldsymbol{\alpha} \circ \boldsymbol{\beta} \circ \lambda) - 2\alpha\beta T,$$
$$Q_{12} = \mathrm{diag}((\boldsymbol{\alpha} + \boldsymbol{\beta}) \circ \lambda) + (\alpha + \beta)T$$
$$Q_{13} = -\boldsymbol{\beta} \circ \nu - \boldsymbol{\alpha} \circ \eta, \ Q_{22} = -2T$$
$$Q_{23} = \nu + \eta, \ Q_{33} = 0,$$

with T given by (17)*, and*

$$\boldsymbol{\alpha} = [\alpha + (\beta - \alpha)\mathbf{1}_{\mathcal{I}^+}(1), \cdots, \alpha + (\beta - \alpha)\mathbf{1}_{\mathcal{I}^+}(n)]$$
$$\boldsymbol{\beta} = [\beta - (\beta - \alpha)\mathbf{1}_{\mathcal{I}^-}(1), \cdots, \beta - (\beta - \alpha)\mathbf{1}_{\mathcal{I}^-}(n)]$$
$$\nu_i \in \mathbb{R}_+ \ \text{for } i \notin \mathcal{I}^+$$
$$\eta_i \in \mathbb{R}_+ \ \text{for } i \notin \mathcal{I}^-$$
$$\lambda_{ij} \in \mathbb{R}_+ \ \text{for } \{i,j\} \notin (\mathcal{I}^+ \times \mathcal{I}^+) \cup (\mathcal{I}^- \times \mathcal{I}^-).$$

Proof 4 *See Appendix D.*

We do not know *a priori* which neurons are always active or always inactive. However, we can partially find them by computationally cheap presolve steps. Specifically, if x is known to satisfy $\underline{x} \le x \le \bar{x}$ (bounds on the pre-activation values), then we have $\mathcal{I}^+ = \{i \mid \underline{x}_i \ge 0\}$, $\mathcal{I}^- = \{i \mid \bar{x}_i < 0\}$, and $\mathcal{I}^\pm = \{i \mid \bar{x}_i \underline{x}_i \le 0\}$. These element-wise bounds can be found by, for example, interval bound propagation [42], [43] or the LP approach of [16]. Indeed, tighter bounds result in a less conservative description of the ReLU function outlined in Lemma 4.

E. Other Activation Functions

Deriving non-conservative QCs for the other functions (other than ReLU) is more complicated as they are not on the boundary of any sector. However, by bounding these functions at multiple points by sector bounds of the form (12), we can obtain a substantially better over-approximation. In Figure 3, we illustrate this idea for the $\tanh$ function.

A secondary approach is to use the element-wise bounds on the inputs to the activation functions to use a tighter sector bound condition in (12). For instance, suppose $x \in \mathbb{R}$ satisfies the bounds $\underline{x} \le x \le \bar{x}$. Then the $\tanh$ function satisfies the sector condition in (12), where α and β are given by

$$\alpha = \begin{cases} \tanh(\bar{x})/\bar{x} & \text{if } \underline{x}\bar{x} \ge 0 \\ \min(\tanh(\underline{x})/\underline{x}, \tanh(\bar{x})/\bar{x}) & \text{otherwise.} \end{cases}$$

$$\beta = \begin{cases} \tanh(\underline{x})/\underline{x} & \text{if } \underline{x}\bar{x} \ge 0 \\ 1 & \text{otherwise.} \end{cases}$$

IV. NEURAL NETWORK VERIFICATION VIA SEMIDEFINITE PROGRAMMING

In the previous section, we developed an abstraction of sets and nonlinearities using quadratic constraints. In this section, we use this abstraction to develop an LMI feasibility problem that can assert whether $f(\mathcal{X}) \subseteq \mathcal{S}_y$ (or $\mathcal{X} \subseteq \mathcal{S}_x = f^{-1}(\mathcal{S}_y)$). The crux of our idea in the development of the LMI is the $\mathcal{S}$-procedure [29], a technique to reason about multiple quadratic constraints, and is frequently used in robust control and optimization [44], [45].
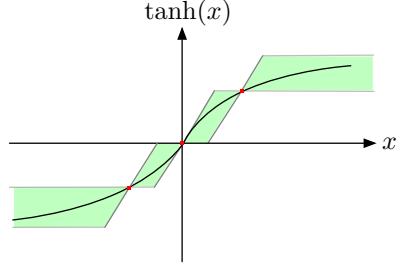

Fig. 3: The curve of the tanh function over-approximated by the intersection of three sectors.

A. Single-layer Neural Networks

For the sake of simplicity in the exposition, we start with the analysis of one-layer neural networks and then extend the results to the multi-layer case in §IV-B. We further assume that the safe set $\mathcal{S}_x$ in (10) is specified by a single quadratic form, i.e., $m = 1$. We state our main result in the following theorem.

Theorem 1 (SDP for one layer) *Consider a one-layer neural network $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ described by the equation*

$$f(x) = W^1 \phi(W^0 x + b^0) + b^1, \tag{25}$$

where ϕ satisfies the quadratic constraint defined by $\mathcal{Q}_\phi$, i.e., for any $Q \in \mathcal{Q}_\phi$,

$$\begin{bmatrix} z \\ \phi(z) \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} z \\ \phi(z) \\ 1 \end{bmatrix} \ge 0 \quad \text{for all } z \in \mathbb{R}^{n_1}. \tag{26}$$

Suppose $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$, where $\mathcal{X}$ satisfies the quadratic constraint defined by $\mathcal{P}_\mathcal{X}$, i.e., for any $P \in \mathcal{P}_\mathcal{X}$,

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \ge 0 \quad \text{for all } x \in \mathcal{X}. \tag{27}$$

Suppose the $\mathcal{S}_x$ is defined by

$$\mathcal{S}_x = \left\{ x \in \mathbb{R}^{n_x} \mid \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix}^\top S \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix} \le 0 \right\},$$

where $S \in \mathbb{S}^{n_x + n_y + 1}$ is given. Consider the following matrix inequality,

$$M_{\mathrm{in}}(P) + M_{\mathrm{mid}}(Q) + M_{\mathrm{out}}(S) \preceq 0, \tag{28}$$

where

$$M_{\mathrm{in}}(P) = \begin{bmatrix} I_{n_0} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} P \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{29a}$$

$$M_{\mathrm{mid}}(Q) = \begin{bmatrix} W^{0\top} & 0 & 0 \\ 0 & I_{n_1} & 0 \\ b^{0\top} & 0 & 1 \end{bmatrix} Q \begin{bmatrix} W^0 & 0 & b^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{29b}$$

$$M_{\mathrm{out}}(S) = \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & W^{1\top} & 0 \\ 0 & b^{1\top} & 1 \end{bmatrix} S \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & W^1 & b^1 \\ 0 & 0 & 1 \end{bmatrix}, \tag{29c}$$

If (28) *is feasible for some $P \in \mathcal{P}_\mathcal{X}$, $Q \in \mathcal{Q}_\phi$, then $\mathcal{X} \subseteq \mathcal{S}_x$.*

Proof 5 See Appendix F.

Theorem 1 states that if the matrix inequality (28) is feasible for some $(P, Q) \in \mathcal{P}_{\mathcal{X}} \times \mathcal{Q}_{\phi}$, then we can certify that the network $\mathcal{X} \subseteq \mathcal{S}_x$ or $f(\mathcal{X}) \subseteq \mathcal{S}_y$. Since $\mathcal{P}_{\mathcal{X}}$ and $\mathcal{Q}_{\phi}$ are both convex, (28) is a linear matrix inequality (LMI) feasibility problem and, hence, can be efficiently solved via interior-point method solvers for convex optimization.

Remark 1 (QC for neural network) It follows from the proof of Theorem 1 that, in view of Definition 2, the neural network in (25) satisfies the quadratic constraint defined by $\mathcal{Q}_f$, where

$$\mathcal{Q}_f = \{Q_f \,|\, \exists Q \in \mathcal{Q}_{\phi} \text{ s.t. } M_{\mathrm{mid}}(Q) \preceq M_{\mathrm{out}}(Q_f)\}. \quad (30)$$

In other words, for any $Q_f \in \mathcal{Q}_f$ we have

$$\begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix}^\top Q_f \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } x \in \mathbb{R}^{n_x}.$$

Note that the above QC for the neural network holds globally for the whole space $\mathbb{R}^{n_x}$.

B. Multi-layer Neural Networks

We now turn to multi-layer neural networks. Assuming that all the activation functions are the same across the layers (repetition across layers), we can concatenate all the pre- and post-activation signals together and form a more compact representation. To see this, we first introduce $\mathbf{x} = [x^{0\top} \cdots x^{\ell\top}]^\top \in \mathbb{R}^{n_0+n}$, where $\ell \geq 1$ is the number of hidden layers. We further define the entry selector matrices $\mathbf{E}^k \in \mathbb{R}^{n_k \times (n_0+n)}$ such that $x^k = \mathbf{E}^k \mathbf{x}$ for $k = 0, \cdots, \ell$. Then, we can write (2) compactly as

$$x = \mathbf{E}^0 \mathbf{x}, \ \mathbf{B}\mathbf{x} = \phi(\mathbf{A}\mathbf{x} + \mathbf{b}), \ f(x) = W^\ell \mathbf{E}^\ell \mathbf{x} + b^\ell, \quad (31a)$$

where

$$\mathbf{A} = \begin{bmatrix} W^0 & 0 & \cdots & 0 & 0 \\ 0 & W^1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & W^{\ell-1} & 0 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b^0 \\ b^1 \\ \vdots \\ b^{\ell-1} \end{bmatrix} \quad (31b)$$

$$\mathbf{B} = \begin{bmatrix} 0 & I_{n_1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_{n_{\ell-1}} & 0 \\ 0 & 0 & \cdots & 0 & I_{n_\ell} \end{bmatrix}.$$

In the following result, we develop the multi-layer counterpart of Theorem 1 for the multi-layer neural network in (31).

Theorem 2 (SDP for multiple layers) *Consider the multi-layer neural network described by* (31). *Suppose* $\mathcal{X} \subset \mathbb{R}^{n_x}$ *and* ϕ *satisfy the quadratic constraints defined by* $\mathcal{P}_{\mathcal{X}}$ *and* $\mathcal{Q}_{\phi}$, *respectively, as in* (27) *and* (26). *Consider the following LMI.*

$$M_{\mathrm{in}}(P) + M_{\mathrm{mid}}(Q) + M_{\mathrm{out}}(S) \preceq 0, \quad (32)$$

where

$$M_{\mathrm{in}}(P) = \begin{bmatrix} \mathbf{E}^0 & 0 \\ 0 & 1 \end{bmatrix}^\top P \begin{bmatrix} \mathbf{E}^0 & 0 \\ 0 & 1 \end{bmatrix} \quad (33a)$$

$$M_{\mathrm{mid}}(Q) = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{B} & 0 \\ 0 & 1 \end{bmatrix}^\top Q \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{B} & 0 \\ 0 & 1 \end{bmatrix} \quad (33b)$$

$$M_{\mathrm{out}}(S) = \begin{bmatrix} \mathbf{E}^0 & 0 \\ W^\ell \mathbf{E}^\ell & b^\ell \\ 0 & 1 \end{bmatrix}^\top S \begin{bmatrix} \mathbf{E}^0 & 0 \\ W^\ell \mathbf{E}^\ell & b^\ell \\ 0 & 1 \end{bmatrix}, \quad (33c)$$

and $S \in \mathbb{S}^{n_x+n_y+1}$ *is a given symmetric matrix. If* (32) *is feasible for some* $(P, Q) \in \mathcal{P}_{\mathcal{X}} \times \mathcal{Q}_{\phi}$, *then*

$$\begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix}^\top S \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix} \leq 0 \quad \text{for all } x \in \mathcal{X}. \quad (34)$$

Proof 6 See Appendix G.

Remark 2 For the case that the safe set is characterized by more than one quadratic inequality, i.e., when $m > 1$ in (10), then $\mathcal{X} \subseteq \mathcal{S}_x$ if the following LMIs,

$$M_{\mathrm{in}}(P_i) + M_{\mathrm{mid}}(Q_i) + M_{\mathrm{out}}(S_i) \preceq 0 \ i = 1, \cdots, m, \quad (35)$$

hold for some $P_i \in \mathcal{P}_{\mathcal{X}}$ and $Q_i \in \mathcal{Q}_{\phi}$.

V. OPTIMIZATION OVER THE ABSTRACTED NETWORK

In the previous section, we developed an LMI feasibility problem as a sufficient to verify the safety of the neural network. We can incorporate this LMI as a constraint of an optimization problem to solve problems beyond safety verification. More specifically, we can define the following SDP,

$$\begin{aligned} \text{minimize} \quad & g(P, Q, S) \quad (36) \\ \text{subject to} \quad & M_{\mathrm{in}}(P) + M_{\mathrm{mid}}(Q) + M_{\mathrm{out}}(S) \preceq 0 \\ & (P, Q, S) \in \mathcal{P}_{\mathcal{X}} \times \mathcal{Q}_{\phi} \times \mathcal{S}, \end{aligned}$$

where $g(P, Q, S)$ is a convex function of P, Q, S, and $\mathcal{S}$ is a convex set in $\mathbb{S}^{n_x+n_y+1}$. In the following subsections, we allude to some utilities of the SDP (36), which we call DeepSDP.

A. Reachable Set Estimation

In Theorem 1, we developed a feasibility problem to assert whether the input set $\mathcal{X}$ is enclosed in the safe set $\mathcal{S}_x$. In particular, suppose $\mathcal{S}_x$ is described by $\mathcal{S}_x = \{x \mid c^\top f(x) - d \leq 0\}$ with a given c and d. By defining

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & c \\ 0 & c^\top & -2d \end{bmatrix}, \quad (37)$$

the feasibility of (32) for some $(P, Q) \in \mathcal{P}_{\mathcal{X}} \times \mathcal{Q}_{\phi}$ implies $c^\top f(x) \leq d$ for all $x \in \mathcal{X}$. In other words, d is a certified upper bound on the optimal value of the optimization problem

$$\text{maximize } c^\top f(x) \quad \text{subject to } x \in \mathcal{X}. \quad (38)$$

Now if we treat $d \in \mathbb{R}$ as a decision variable, we can minimize this bound by solving (36) with $g(P, Q, S) = d$. This is particularly useful for over approximating the reachable set $f(\mathcal{X})$ by a polyhedron of the form $\mathcal{S}_y = \cap_i \{y \in \mathbb{R}^{n_y} \mid c_i^\top y - d_i \leq 0\}$, where c_i are given and the goal is to find the smallest value of d_i, for each i, such that $f(\mathcal{X}) \subseteq \mathcal{S}_y$.

More generally, we can use the SDP in (36) to find the "smallest" over-approximation of $f(\mathcal{X})$. Specifically, define

$$ S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & A_y^2 & A_y b_y \\ 0 & b_y^\top A_y & b_y^\top b_y - 1 \end{bmatrix}. $$

Then, the inclusion $f(\mathcal{X}) \subseteq \mathcal{S}_y = f(\mathcal{S}_x)$ implies that $f(\mathcal{X})$ is enclosed by the ellipsoid $\mathcal{S}_y = \{y \in \mathbb{R}^{n_y} \mid \|A_y y + b_y\|_2 \leq 1\}$. Therefore, finding the minimum-volume ellipsoid enclosing $f(\mathcal{X})$ amounts to the optimization problem

$$
\begin{aligned}
\text{minimize} \quad & \log \det(A_y^{-1}) & (39) \\
\text{subject to} \quad & M_{\text{in}}(P) + M_{\text{mid}}(Q) + M_{\text{out}}(S(A_y, b_y)) \preceq 0 \\
& (P, Q, A_y, b_y) \in \mathcal{P}_\mathcal{X} \times \mathcal{Q} \times \mathbb{S}^{n_y} \times \mathbb{R}^{n_y}.
\end{aligned}
$$

Note that this problem is not convex in (A_y, b_y) due to the non-affine dependence of S on these variables. However, by using Schur complements, we can formulate an equivalent convex program. We skip the details for the sake of space and refer the reader to [35].

VI. DISCUSSION AND NUMERICAL EXPERIMENTS

In this section, we discuss the numerical aspects of our approach. For solving the SDP, we used MOSEK [46] with CVX [47] on a 5-core personal computer with 8GB of RAM. For all experiments, we used ReLU activation functions and did Interval Bound Propagation as a presolve step to determine the element-wise bounds on the activation functions. All code, data, and experiments for this paper are available at https://github.com/mahyarfazlyab/DeepSDP. We start with the computational complexity of the proposed SDP.

A. Computational Complexity

1) Input Set: The number of decision variables for the input set depends on the set representation. The quadratically constrained set that over-approximates hyperectangles is indexed by n_x decision variables, where n_x is the input dimension (see Proposition 1). Note that for hyper-rectangles, we can include additional quadratic constraints. Indeed, any x satisfying $\underline{x} \leq x \leq \bar{x}$ satisfies $2n_x^2 - n_x$ quadratic constraints of the form $(x_i - \underline{x}_i)(\bar{x}_j - x_i) \geq 0$, $(x_i - \underline{x}_i)(x_j - \underline{x}_j) \geq 0$ $i \neq j$, $(x_i - \bar{x}_i)(x_j - \bar{x}_j) \geq 0$ $i \neq j$. However, one can precisely characterize a hyper-rectangle with only n_x of these quadratic constraints, namely, $(x_i - \underline{x}_i)(\bar{x}_i - x_i) \geq 0$. Our numerical computations reveal that adding the remaining QCs would not tighten the relaxation.

For polytopes, the maximum number of decision variables is $\binom{m}{2}$, where m is the number of half-spaces defining the polytope. However, we can use some heuristics to remove quadratic constraints that are not "tight". For instance, for the polytope $\mathcal{X} = \{x \mid Hx \leq h\}$, we can write $\binom{m}{2}$ sector

bounds of the form $(H_i^\top x - h_i)(H_j^\top x - h_j) \geq 0$. Now if the intersection of these hyperplanes belongs to $\mathcal{X}$, then the sector would be tight (see Figure 5). We can verify this by checking the feasibility of

$$ H_i^\top x - h_i = H_j^\top x - h_j = 0, \ H_k^\top x - h_k \leq 0, \ k \neq i, j. $$

Finally, for the case of ellipsoids, we only have one decision variable, the parameter μ in (9).

2) Activation Functions: For a network with n hidden neurons, if we use all possible quadratic constraints, the number of decision variables will be $\mathcal{O}(n + n^2)$. If we ignore repeated nonlinearities, we will arrive at $\mathcal{O}(n)$ decision variables. In our numerical experiments, we did not observe any additional conservatism after removing repeated nonlinearities across the neurons of the same layer. However, accounting for repeated nonlinearities was very effective for the case of multiple layers.

3) Safety Specification Set: The number of decision variables for the safety specification set depends on how we would like to bound the output set. For instance, for finding a single hyperplane, we have only one decision variable. For the case of ellipsoids, there will be $\mathcal{O}(n_y^2)$ decision variables.

B. Synthetic Examples

1) Number of Hidden Layers: As the first experiment, we consider finding over-approximations of the reachable set of a neural network with a varying number of layers, for a given input set. Specifically, we consider randomly-generated neural networks with $n_x = 2$ inputs, $n_y = 2$ outputs, and $\ell = \{1, 2, 3, 4\}$ hidden layers, each having $n_k = 100$ neurons per layer. For the input set, we consider ℓ_∞ balls with center $x^\star = (1, 1)$ and radius $\epsilon = 0.1$. We use DeepSDP to find over-approximations of $f(\mathcal{X})$ in the form of polytopes (see §V-A). In Figure 4, we compare the output set $f(\mathcal{X})$ (using exhaustive search over $\mathcal{X}$) with two over-approximations: the red polytope is obtained by solving DeepSDP. The dashed black polytope is obtained by the semidefinite relaxation (SDR) approach of [34]. We observe that the bounds obtained by DeepSDP are relatively tighter, especially for deeper networks. In Appendix H, we provide more visualizations.

2) Repeated Nonlinearities: As the second experiment, we study the effect of including repeated nonlinearities on the tightness of the bounds. Specifically, we bound the output of a randomly-generated neural network with $n_x = 2$ inputs, $n_y = 2$ output, and $n_k = 10$ neurons per layer by a polytope with 6 facets. For the input set we consider ℓ_∞ ball with center $x^\star = (1, 1)$ and radius $\epsilon = 0.1$. In Figure 6, we plot the output set, and it over-approximation by DeepSDP before and after including repeated nonlinearities. We observe that by including repeated nonlinearities, the bounds become tighter, especially for deep networks.

3) Comparison with Other Methods: As the third experiment, we consider the following optimization problem,

$$ f^\star = \sup_{\|x - x^\star\|_\infty \leq \epsilon} c^\top f(x). \quad (40) $$
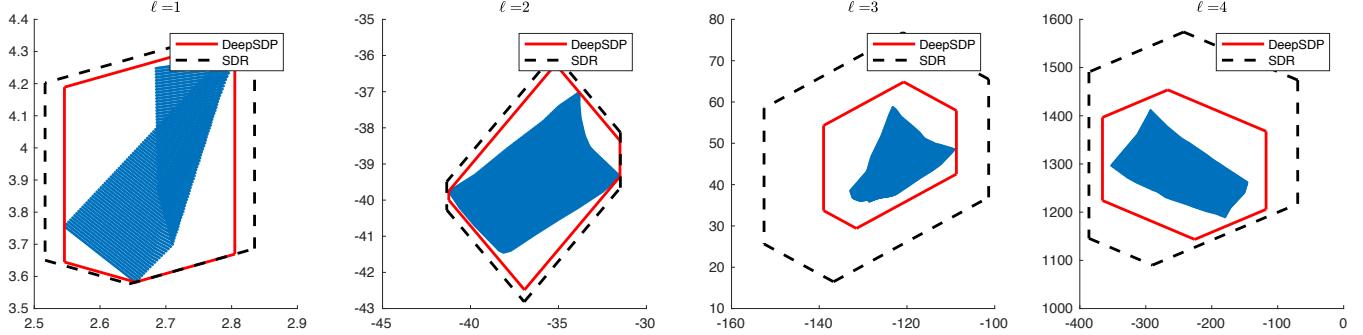
Fig. 4: Illustrations of the output set (blue), the polytope obtained from the results of this paper (red), and the polytope obtained by the semidefinite relaxation of [34] (dashed black). The number of neurons per layer is 100, and the input set is the ℓ_∞ ball with center $x^\star = (1,1)$ and radius $\epsilon = 0.1$. The weights of the neural networks are drawn according to the Gaussian distribution $\mathcal{N}(0, 1/\sqrt{n_x})$. From the left to right, the number of hidden layers is $1, 2, 3$, and 4 (the activation function is ReLU).

	Bounds				Running Time (Sec)			
ℓ	MILP	DeepSDP	SDR	LP	MILP	DeepSDP	SDR	LP
1	1.07	1.12	1.13	1.81	0.04	0.82	0.55	
2	2.04	2.52	2.74	7.62	25.96	8.26	4.71	
3	-	11.08	12.21	50.60	-	34.18	31.20	
4	-	47.74	54.15	368.65	-	78.95	94.74	
5	-	218.8	266.3	3004.9	-	164.63	207.77	

TABLE I: Average values (over 100 runs) of different upper bounds for the problem $\sup_{x \in \mathcal{X}} f(x)$ with $\mathcal{X} = \|x - x_\star\|_\infty \leq \epsilon$, $x_\star = 1_{n_x}$ and $\epsilon = 0.2$.
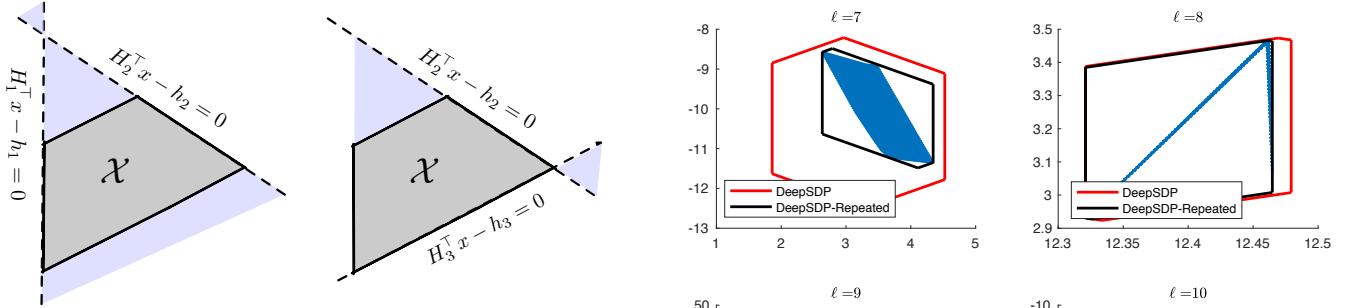

Fig. 5: A sector bound that is not tight (Left); and a sector bound that is tight (Right).

To evaluate the tightness of our bounds, we compare DeepSDP with the MILP formulation of [13], the semidefinite relaxation (SDR) of [34], and the LP relaxation of [16]. For the problem data, we generated random instances of neural networks with $n_x = 10$ inputs, $n_y = 1$ output and $\ell \in \{1, \cdots, 5\}$ hidden layers; for each layer size, we generated 100 random neural networks with their weights and biases chosen independently from the normal distribution $\mathcal{N}(0, 1/\sqrt{n_x})$. For the input set, we consider $x^\star = 1_{n_x}$ and $\epsilon = 0.2$. In Table I, we report the comparisons of bounds and running times. The MILP formulation finds the global solution but the running time grows quickly as the number of neurons increases. Compared to SDR, the bounds of DeepSDP are relatively tighter, especially for deeper networks. Finally, the LP relaxation bounds are considerably looser but the running time is negligible. In Figure 7, we plot the histograms of the normalized gap between the optimal value $f^\star$ and the upper bound f^{SDP} for layer sizes $\ell = 1, 2$.
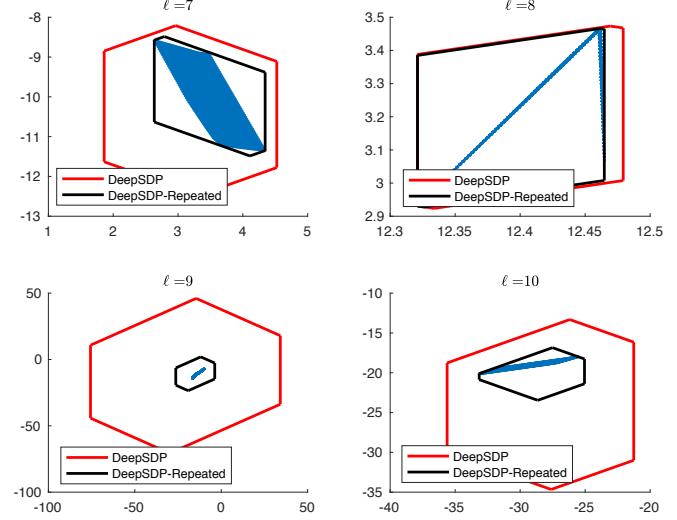

Fig. 6: Plots of the output set (blue), the polytope obtained from DeepSDP without repeated nonlinearities (red), and the polytope obtained by DeepSDP after including repeated nonlinearities (black).

C. Verification of Approximate Model Predictive Control

Consider an LTI system

$$x_{k+1} = Ax_k + Bu_k, \quad x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad (41)$$

where $x_k \in \mathbb{R}^{n_x}$ is the state at time k, $u_k \in \mathbb{R}^{n_u}$ is the control input, and A, B are matrices of appropriate size. The state and control input are subject to the box constraints $\mathcal{X} = \{x \mid \underline{x} \leq x \leq \bar{x}\}$ and $\mathcal{U} = \{u \mid \underline{u} \leq u \leq \bar{u}\}$.

Suppose the control policy is parameterized by a multi-layer fully-connected feed-forward network f that is trained off-line to approximate a model predictive control (MPC) law
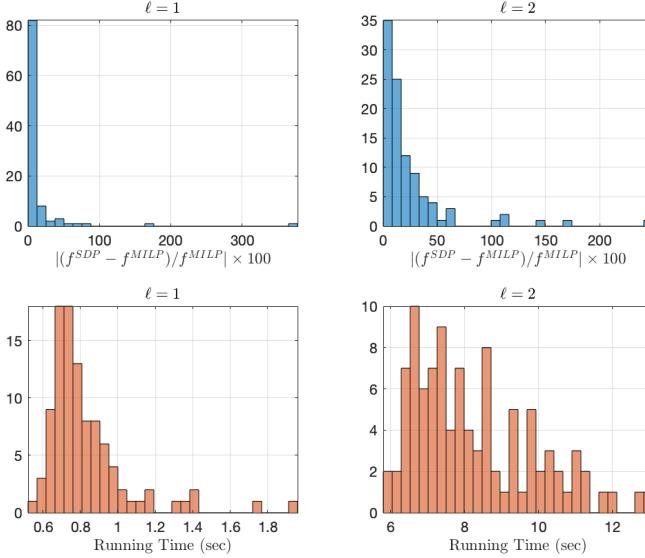
Fig. 7: (Top) Histograms of the normalized gap between the optimal values and their corresponding bounds obtained by DeepSDP. (Bottom) Histograms of solve times in seconds.

$\mu^\star(x)$. The motivation is to reduce the computational burden of solving an optimization problem online to determine the MPC control action. The trained neural network, however, does not necessarily satisfy the specifications of the MPC control law such as state and control constraint satisfaction. To ensure input constraint satisfaction, we project the neural network output onto $\mathcal{U}$, resulting in the closed-loop system,

$$x_{k+1} = f_{cl}(x_k) := Ax_k + B\text{Proj}_{\mathcal{U}}(f(x_k)). \quad (42)$$

Note that for input box constraints, $\mathcal{U} = \{u \mid \underline{u} \leq u \leq \bar{u}\}$, we can embed the projection operator as two additional layers with a specific choice of weights and biases. Indeed, for an ℓ-layer f, we can describe $f_p(x) = \text{Proj}_{\mathcal{U}}(f(x_k))$ via the $(\ell + 2)$-layer ReLU network,

$$
\begin{aligned}
x^0 &= x \quad (43) \\
x^{k+1} &= \max(W^k x^k + b^k, 0) \quad k = 0, \cdots, \ell - 1 \\
x^{\ell+1} &= \max(W^\ell x^\ell + b^\ell - \underline{u}, 0) \\
x^{\ell+2} &= \max(-x^{\ell+1} + \bar{u} - \underline{u}, 0) \\
f_p(x) &= -x^{\ell+2} + \bar{u}.
\end{aligned}
$$

To validate state constraint satisfaction, we must ensure that there is a set of initial states $\mathcal{E} \subseteq \mathcal{X}$ whose trajectories would always satisfy the state constraints. One such set is a positive invariant set. By definition, a set $\mathcal{E}$ is positively invariant with respect to f_{cl}, if and only if $x_0 \in \mathcal{E}$ implies $x_k \in \mathcal{E}$ for all $k \geq 1$. Equivalently, $\mathcal{E}$ is positively invariant if $f_{cl}(\mathcal{E}) \subseteq \mathcal{E}$. We now show that how we can compute a positive invariant set for (42) using semidefinite programming.

To find a positive invariant set for the closed-loop system, we consider the candidate set $\mathcal{E} = \{x \mid \|x\|_\infty \leq \epsilon\}$. We first over approximate the one-step reachable set $f_{cl}(\mathcal{E})$ by the polytope $\mathcal{P} = \{x \mid Hx \leq h\}$, $H \in \mathbb{R}^{m \times n_x}, h \in \mathbb{R}^m$. To

do this, we form the following m SDPs

$$
\begin{aligned}
\text{minimize} \quad & h_i \quad (44) \\
\text{subject to} \quad & M_{\text{in}}(P) + M_{\text{mid}}(Q) + M_{\text{out}}(S_i) \preceq 0 \\
& (P, Q, h_i) \in \mathcal{P}_{\mathcal{E}} \times \mathcal{Q}_\phi \times \mathbb{R},
\end{aligned}
$$

where

$$
S_i = \begin{bmatrix} 0 & 0 & A^\top H^\top e_i \\ 0 & 0 & B^\top H^\top e_i \\ e_i^\top HA & e_i^\top HB & -2e_i^\top h \end{bmatrix} \quad i = 1, \cdots, m.
$$

With this choice of S_i, it is not difficult to show that the feasibility of the LMIs in (44) implies $f_{cl}(\mathcal{E}) \subseteq \mathcal{P}$, and therefore, (44) finds the smallest $\mathcal{P}$ that encloses $f_{cl}(\mathcal{E})$. Then, $\mathcal{E}$ is positively invariant if $\mathcal{P} \subseteq \mathcal{E}$.

For the numerical experiment, we first consider a 2D system

$$x_{t+1} = 1.2 \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u_t, \quad (45)$$

subject to the state and input constraints $x_t \in \mathcal{X} = \{x \mid \|x\|_\infty \leq 5\}$ and $u \in \mathcal{U} = \{u \mid \|u\|_\infty \leq 1\}$. We are interested in stabilizing the system by solving the finite horizon problem

$$
\begin{aligned}
\text{minimize} \sum_{t=0}^{T} &\|x_t\|_2^2 + u_t^2 \quad (46) \\
\text{s.t.} \quad & (x_t, u_t) \in \mathcal{X} \times \mathcal{U} \quad t = 0, \cdots, T, \ x_0 = x,
\end{aligned}
$$

and choosing the control law as $\mu_{MPC}(x) = u_0^\star$. For generating the training data, we compute $\mu_{MPC}(x)$ at 6284 uniformly chosen random points from the control invariant set. We then train a neural network with two inputs, one output, and two hidden layers with 32 and 16 neurons, respectively using the mean-squared loss. In Figure 9, we plot the explicit MPC control law as well as its approximation by the neural network.

In Figure 8, we plot the largest invariant set $\mathcal{E}$ that we could find, which is $\mathcal{E} = \{x \mid \|x\|_\infty \leq 0.65\}$. In this figure, we also plot the output reachable sets for the first four time steps, starting from the initial set $\mathcal{E}$, as well as their over-approximations by DeepSDP.

VII. CONCLUSIONS

We proposed a semidefinite programming framework for robustness analysis and safety verification of feed-forward fully-connected neural networks with general activation functions. Our main idea is to abstract the nonlinear activation functions by quadratic constraints that are known to be satisfied by all possible input-output instances of the activation functions. We then showed that we can analyze the abstracted network via semidefinite programming. We conclude this paper with several future directions.

First, a notable advantage of the proposed SDP compared to other convex relaxations is the relative tightness of the bounds. In particular, coupling all pairs of neurons in the network (repeated nonlinearities) can considerably reduce conservatism. However, coupling all neurons is not feasible for even medium-sized networks as the number of decision variables would scale quadratically with the number of neurons. Nevertheless, our numerical experiments show that most of
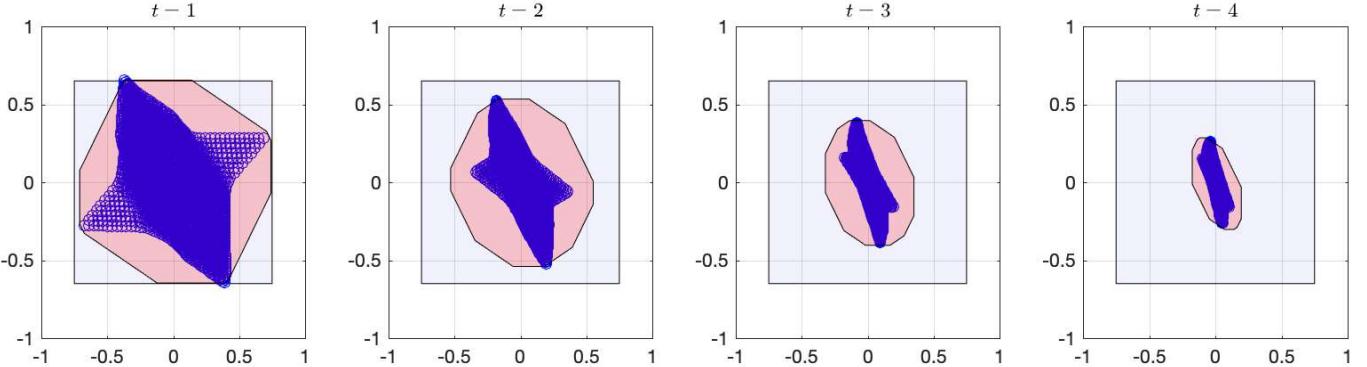
Fig. 8: Illustration of the invariant set $\mathcal{E}$ (light blue), the output reachable sets (dark blue) and their over-approximations (light red) for the system described in § VI-C. To over approximate the reachable set at each time step t, we use the over-approximation of the reachable set computed by DeepSDP at $t-1$ as the initial set.
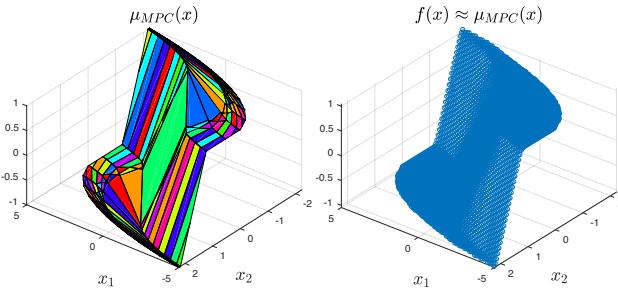

Fig. 9: The explicit MPC control law for the system described in §VI-C (left), and its approximation by a neural network (right).

these pair-wise couplings of neurons are redundant and do not tighten the bounds. It would be interesting to develop a method that can decide *a priori* that coupling which pairs of neurons would tighten the relaxation. Second, one of the drawbacks of SDPs is their limited scalability in general. Exploiting the structure of the problem to reduce the computational complexity would be an important future direction. Third, we have only considered fully-connected networks in this paper. It would be interesting to extend the results to other architectures. Finally, incorporating the proposed framework in training neural networks with desired robustness properties would be another important future direction.

APPENDIX

A. Proof of Proposition 1

The inequality $\underline{x} \leq x \leq \bar{x}$ is equivalent to n_x quadratic inequalities of the form $(x_i - \underline{x}_i)(\bar{x}_i - x_i) \geq 0 \quad i = 1, \cdots, n_x$. Multiplying both sides of with $\gamma_i \geq 0$ and summing over $i = 1, \cdots, n_x$ yields the claimed inequality. $\square$

B. QCs for Polytopes, Zonotopes, and Ellipsoids

1) Polytopes: For every vector x satisfying $Hx \leq h$, we have $(H_i^\top x - h_i)(H_j^\top x - h_j) \geq 0$, $i \neq j$, where $H_i^\top$ is the i-th row of H. These inequalities imply

$$\sum_{1 \leq i,j \leq m} \Gamma_{ij}(H_i^\top x - h_i)(H_j^\top x - h_j) \geq 0,$$

where $\Gamma_{ij} = \Gamma_{ji}$, $i \neq j$, $\Gamma_{ii} = 0$, and $\gamma_i \geq 0$. Equivalently, $(Hx - h)^\top \Gamma (Hx - h) \geq 0$. The preceding inequality is equivalent to (7). Suppose now the set $\{x \mid Hx > h\}$ is empty. Then

$$\mathcal{X} = \{x \mid (H_i^\top x - h_i)^\top (H_j^\top x - h_j) \geq 0, \ i \neq j\}.$$

To show this set equality define $\mathcal{X}_Q$ as the set on the right-hand side. We have $\mathcal{X} \subset \mathcal{X}_Q$. To show $\mathcal{X}_Q \subset \mathcal{X}$, suppose $x \notin \mathcal{X}$, meaning that there exists an $1 \leq r \leq m$ such that

$$H_{i_k}^\top x - h_{i_k} > 0 \quad k = 1, \cdots, r$$
$$H_{i_k}^\top x - h_{i_k} \leq 0 \quad k = r+1, \cdots, m$$

Therefore $x \notin \mathcal{X}_Q$ unless $r = m$, which cannot happen as the set $\{x \mid Hx > h\}$ is assumed to be empty. Therefore $\mathcal{X}_Q \subset \mathcal{X}$, and hence, $\mathcal{X} = \mathcal{X}_Q$.

2) Zonotopes: By multiplying both sides of (8) by $[\lambda^\top \ 1]$ and $[\lambda^\top \ 1]^\top$, respectively, and noting that $x = x_c + A\lambda$ we obtain

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq \begin{bmatrix} \lambda \\ 1 \end{bmatrix}^\top \begin{bmatrix} -2\Gamma & \Gamma 1_m \\ -1_m^\top \Gamma & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ 1 \end{bmatrix} \geq 0,$$

where the second inequality follows from the fact that $\lambda \in [0,1]^m$, hence satisfying the QC of Proposition 1.

3) Ellipsoids: Any $x \in \mathcal{X}$ satisfies $\mu(1 - (Ax+b)^\top(Ax+b)) \geq 0$ for $\mu \geq 0$. The latter inequality is equivalent to (9). $\square$

C. Proof of Lemma 2

For any distinct pairs $(x_i, \varphi(x_i))$ and $(x_j, \varphi(x_j))$, $1 \leq i < j \leq n$, we can write the slope restriction inequality in (14) as

$$\begin{bmatrix} x_i - x_j \\ \varphi(x_i) - \varphi(x_j) \end{bmatrix}^\top \begin{bmatrix} -2\alpha\beta & \alpha+\beta \\ \alpha+\beta & -2 \end{bmatrix} \begin{bmatrix} x_i - x_j \\ \varphi(x_i) - \varphi(x_j) \end{bmatrix} \geq 0.$$

By multiplying both sides by $\lambda_{ij} \geq 0$, we obtain

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top \begin{bmatrix} -2\alpha\beta E_{ij}\lambda_{ij} & (\alpha+\beta)E_{ij}\lambda_{ij} & 0 \\ (\alpha+\beta)E_{ij}\lambda_{ij} & -2E_{ij}\lambda_{ij} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0,$$

where $E_{ij} = (e_i - e_j)(e_i - e_j)^\top$ and $e_i \in \mathbb{R}^n$ is the i-th unit vector in $\mathbb{R}^n$. Summing over all $1 \leq i < j \leq n$ will yield the desired result.

D. Proof of Lemma 3

Consider the equivalence in (19) for the i-th coordinate of $y = \max(\alpha x, \beta x)$, $x \in \mathbb{R}^n$:

$$(y_i - \alpha x_i)(y_i - \beta x_i) = 0, \ y_i \geq \beta x_i, \quad y_i \geq \alpha x_i.$$

Multiplying these constraints by $\lambda_i \in \mathbb{R}$, $\nu_i \in \mathbb{R}_+$, and $\eta_i \in \mathbb{R}_+$, respectively, and adding them together, we obtain

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}^\top \begin{bmatrix} 0 & (\alpha+\beta)\lambda_i & -\beta\nu_i - \alpha\eta_i \\ (\alpha+\beta)\lambda_i & -2\lambda_i & \nu_i + \eta_i \\ -\beta\nu_i - \alpha\eta_i & \nu_i + \eta_i & 0 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \geq 0.$$

Substituting $x_i = e_i^\top x$ and $y_i = e_i^\top y$, where e_i is the i-th unit vector in $\mathbb{R}^n$, and rearranging terms, we get

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^\top Q_i \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \geq 0, \ i = 1, \cdots, n, \tag{47}$$

where

$$Q_i = \begin{bmatrix} 0 & (\alpha+\beta)\lambda_i e_i^\top & (-\beta\nu_i - \alpha\eta_i)e_i \\ (\alpha+\beta)\lambda_i e_i & -2\lambda_i e_i & (\nu_i + \eta_i)e_i \\ (-\beta\nu_i - \alpha\eta_i)e_i & (\nu_i + \eta_i)e_i & 0 \end{bmatrix}.$$

Furthermore, since $y_i = \max(\alpha x_i, \beta x_i)$ is slope-restricted in $[\alpha, \beta]$, by Lemma 2 we can write

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^\top \begin{bmatrix} -2\alpha\beta T & (\alpha+\beta)T & 0 \\ (\alpha+\beta)T & -2T & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \geq 0. \tag{48}$$

Summing (47) over all $i = 1, \cdots, n$ and adding the result to (48) would yield (22). $\square$

E. Proof of Lemma 4

Consider the relation $y = \max(\alpha x, \beta x)$. For active neurons, $i \in \mathcal{I}^+$, we can write

$$(y_i - \beta x_i)(y_i - \beta x_i) = 0, \ y_i = \beta x_i, \quad y_i \geq \alpha x_i.$$

Similarly, for inactive neurons, $i \in \mathcal{I}^-$, we can write

$$(y_i - \alpha x_i)(y_i - \alpha x_i) = 0, \ y_i \geq \beta x_i, \quad y_i = \alpha x_i.$$

Finally, for unknown neurons, $i \in \mathcal{I}^\pm$, we can write

$$(y_i - \alpha x_i)(y_i - \beta x_i) = 0, \ y_i \geq \beta x_i, \quad y_i \geq \alpha x_i.$$

A weighted combination of the above constraints yields

$$\sum_{i=1}^n \lambda_i(y_i - \alpha_i x_i)(y_i - \beta_i x_i) + \nu_i(y_i - \beta_i x_i) + \eta_i(y_i - \alpha_i x_i) \geq 0 \tag{49}$$

where $\alpha_i = \alpha + (\beta - \alpha)\mathbf{1}_{\mathcal{I}^+}(i)$, $\beta_i = \beta - (\beta - \alpha)\mathbf{1}_{\mathcal{I}^-}(i)$, $\nu_i \in \mathbb{R}_+$ for $i \notin \mathcal{I}^+$ and $\eta_i \in \mathbb{R}_+$ for $i \notin \mathcal{I}^-$. Furthermore, since $y_i = \max(\alpha x_i, \beta x_i)$ is slope-restricted on $[\alpha, \beta]$, we can write

$$\sum_{i \neq j} \lambda_{ij}(y_j - y_i - \alpha(x_j - x_i))(y_j - y_i - \beta(x_j - x_i)) \geq 0. \tag{50}$$

Adding (49) and (50) and rearranging terms would yield the desired inequality. $\square$

F. Proof of Theorem 1

Consider the identity $x^1 = \phi(W^0 x^0 + b^0)$. Using the assumption that ϕ satisfies the quadaratic constraint defined by $\mathcal{Q}_\phi$, x^0, x^1 satisfy the QC

$$\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} W^0 & 0 & b^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top Q \begin{bmatrix} W^0 & 0 & b^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{M_{\text{mid}}(Q)} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix} \geq 0, \tag{51}$$

for any $Q \in \mathcal{Q}_\phi$. By assumption $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}_\mathcal{X}$, implying that for any $P \in \mathcal{P}_\mathcal{X}$,

$$\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} I_{n_0} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^\top P \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{M_{\text{in}}(P)} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix} \geq 0, \tag{52}$$

for all $x^0 \in \mathcal{X}$. Suppose (28) holds for some $(P, Q) \in \mathcal{P}_\mathcal{X} \times \mathcal{Q}_\phi$. By left- and right- multiplying both sides of (28) by $[x^{0\top} \ x^{1\top} \ 1]$ and $[x^{0\top} \ x^{1\top} \ 1]^\top$, respectively, we obtain

$$\underbrace{\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^\top M_{\text{in}}(P) \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}}_{\geq 0 \text{ for all } x^0 \in \mathcal{X} \text{ by (52)}} + \underbrace{\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^\top M_{\text{mid}}(Q) \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}}_{\geq 0 \text{ for all } x^0 \in \mathbb{R}^{n_x} \text{ by (51)}}$$

$$+ \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^\top M_{\text{out}}(S) \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix} \leq 0.$$

Therefore, the last term on the left-hand side must be nonpositive for all $x^0 \in \mathcal{X}$, $x^1 = \phi(W^0 x^0 + b^0)$, or, equivalently,

$$\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^\top \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & W^{1\top} & 0 \\ 0 & b^{1\top} & 1 \end{bmatrix} S \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & W^1 & b^1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix} \leq 0.$$

Using the relations $x^0 = x$ and $f(x) = W^1 x^1 + b^1$, the above inequality is the desired inequality in (34). $\square$

G. Proof of Theorem 2

Since ϕ satisfies the QC defined by $\mathcal{Q}_\phi$, for all $\mathbf{x}$ such that $\mathbf{B}\mathbf{x} = \phi(\mathbf{A}\mathbf{x} + \mathbf{b})$ we have

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{B} & 0 \\ 0 & 1 \end{bmatrix}^\top Q \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{B} & 0 \\ 0 & 1 \end{bmatrix}}_{M_{\text{mid}}(Q)} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \geq 0. \tag{53}$$

for any $Q \in \mathcal{Q}_\phi$. By assumption $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}_\mathcal{X}$. Using the relation $x^0 = \mathbf{E}^0 \mathbf{x}$, this QC can be written as

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} \mathbf{E}^0 & 0 \\ 0 & 1 \end{bmatrix}^\top P \begin{bmatrix} \mathbf{E}^0 & 0 \\ 0 & 1 \end{bmatrix}}_{M_{\text{in}}(P)} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \geq 0, \tag{54}$$

which hold for for any $P \in \mathcal{P}_\mathcal{X}$. Suppose the LMI in (32) holds for some $(P, Q) \in \mathcal{P}_\mathcal{X} \times \mathcal{Q}_\phi$. By left- and right-
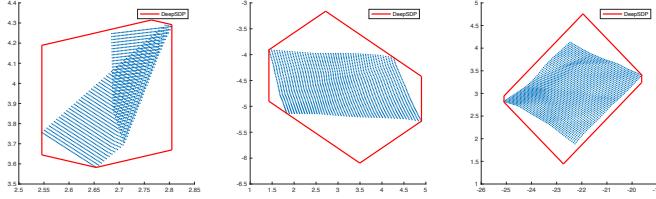
Fig. 10: The effect of the number of hidden neurons on the over-approximation quality of the SDP for a one-layer neural network with 100 (left), 500 (middle), and 1000 hidden nuerons (right). The activation function is ReLU. Quadratic constraints for repeated nonlinearity are not included.
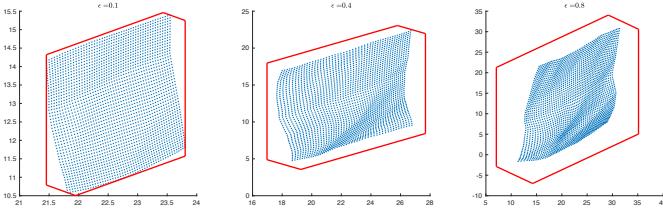

Fig. 11: The effect of ϵ (the ℓ_∞ norm of the input set) on the over-approximation quality of the SDP for $\epsilon = 0.1$ (left), $\epsilon = 0.4$ (middle), and $\epsilon = 0.8$ (right). The network architecture is 2-500-2 with ReLU activation functions. Quadratic constraints for repeated nonlinearity are not included.

multiplying both sides of (26) by $[\mathbf{x}^\top \ 1]$ and $[\mathbf{x}^\top \ 1]^\top$, respectively, we obtain

$$\underbrace{\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_{\mathrm{in}}(P) \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}}_{\geq 0 \text{ by (54)}} + \underbrace{\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_{\mathrm{mid}}(Q) \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}}_{\geq 0 \text{ by (53)}}$$
$$+ \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top M_{\mathrm{out}}(S) \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \leq 0.$$

Therefore, the last quadratic term must be nonpositive for all $x^0 \in \mathcal{X}$, from where we can write

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{E}^0 & 0 \\ W^\ell \mathbf{E}^\ell & b^\ell \\ 0 & 1 \end{bmatrix}^\top S \begin{bmatrix} \mathbf{E}^0 & 0 \\ W^\ell \mathbf{E}^\ell & b^\ell \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \leq 0,$$

for all $x^0 \in \mathcal{X}$. Using the relations $x^0 = \mathbf{E}^0 \mathbf{x}$ and $f(x) = W^\ell \mathbf{E}^\ell \mathbf{x} + b^\ell$ from (31), the above inequality can be written as

$$\begin{bmatrix} x^0 \\ f(x^0) \\ 1 \end{bmatrix}^\top S \begin{bmatrix} x^0 \\ f(x^0) \\ 1 \end{bmatrix} \leq 0, \text{ for all } x^0 \in \mathcal{X}.$$

$\square$

H. More Visualizations

In Figure 10, we show the effect of the number of hidden neurons on the quality of approximation for a single-layer network, and in Figure 11, we change the perturbation size.

REFERENCES

[1] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[2] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[4] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 4480–4488, 2016.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[6] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pp. 1–10, IEEE, 2016.

[7] W. Xiang, P. Musau, A. A. Wild, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson, "Verification for machine learning, autonomy, and neural networks survey," *arXiv preprint arXiv:1810.01989*, 2018.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples (2014)," *arXiv preprint arXiv:1412.6572*.

[9] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[10] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, 2016.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

[12] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," in *Advances in neural information processing systems*, pp. 2613–2621, 2016.

[13] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods Symposium*, pp. 121–138, Springer, 2018.

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

[15] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.

[16] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.

[17] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks," in *UAI*, 2018.

[18] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," in *Advances in Neural Information Processing Systems*, pp. 9832–9842, 2019.

[19] L. Pulina and A. Tacchella, "Challenging smt solvers to verify neural networks," *AI Communications*, vol. 25, no. 2, pp. 117–135, 2012.

[20] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: Verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC 19, (New York, NY, USA), p. 169178, Association for Computing Machinery, 2019.

[21] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE transactions on neural networks and learning systems*, no. 99, pp. 1–7, 2018.

[22] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *International Conference on Machine Learning*, pp. 3575–3583, 2018.

[23] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, IEEE, 2018.

[24] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel, "Towards fast computation of certified robustness for relu networks," *arXiv preprint arXiv:1804.09699*, 2018.

[25] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in neural information processing systems*, pp. 4939–4948, 2018.

[26] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, pp. 2266–2276, 2017.

[27] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Advances in Neural Information Processing Systems*, pp. 6367–6377, 2018.

[28] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," in *27th USENIX Security Symposium (USENIX Security 18)*, (Baltimore, MD), pp. 1599–1614, USENIX Association, Aug. 2018.

[29] V. Yakubovich, "S-procedure in nonlinear control theory," *Vestnick Leningrad Univ. Math.*, vol. 4, pp. 73–93, 1997.

[30] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286, Springer, 2017.

[31] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *International Conference on Computer Aided Verification*, pp. 3–29, Springer, 2017.

[32] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*, pp. 97–117, Springer, 2017.

[33] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[34] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *Advances in Neural Information Processing Systems*, pp. 10900–10910, 2018.

[35] M. Fazlyab, M. Morari, and G. J. Pappas, "Probabilistic verification and reachability analysis of neural networks: Convex relaxations," in *2019 IEEE Conference on Decision and Control (CDC)*, IEEE, 2019.

[36] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87. Springer Science & Business Media, 2013.

[37] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, 1997.

[38] H. K. Khalil and J. W. Grizzle, *Nonlinear systems*, vol. 3. Prentice hall Upper Saddle River, NJ, 2002.

[39] G. Zames and P. Falb, "Stability conditions for systems with monotone and slope-restricted nonlinearities," *SIAM Journal on Control*, vol. 6, no. 1, pp. 89–108, 1968.

[40] F. D'amato, M. A. Rotea, A. Megretski, and U. Jönsson, "New results for analysis of systems with repeated nonlinearities," *Automatica*, vol. 37, no. 5, pp. 739–747, 2001.

[41] V. V. Kulkarni and M. G. Safonov, "All multipliers for repeated monotone nonlinearities," *IEEE Transactions on Automatic Control*, vol. 47, no. 7, pp. 1209–1212, 2002.

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

[43] C.-H. Cheng, G. Nührenberg, and H. Ruess, "Maximum resilience of artificial neural networks," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 251–268, Springer, 2017.

[44] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, vol. 15. Siam, 1994.

[45] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, vol. 28. Princeton University Press, 2009.

[46] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

[47] I. CVX Research, "CVX: Matlab software for disciplined convex programming, version 2.0." http://cvxr.com/cvx, Aug. 2012.

Mahyar Fazlyab (S'13) received his Ph.D. in Electrical and Systems Engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2018. Currently, he is a Postdoctoral Researcher at Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia. His research interests are at the intersection of optimization, control, and machine learning. Dr. Fazlyab won the Joseph and Rosaline Wolf Best Doctoral Dissertation Award in 2019, awarded by the Department of Electrical and Systems Engineering at the University of Pennsylvania.

Manfred Morari (F'05) received the Diploma degree in chemical engineering from ETH Zürich, Zürich, Switzerland, and the Ph.D. degree in chemical engineering from the University of Minnesota, Minneapolis, MN, USA. He was a Professor and the Head of the Department of Information Technology and Electrical Engineering, ETH Zürich. He was the McCollumCorcoran Professor of chemical engineering and the Executive Officer of control and dynamical systems with the California Institute of Technology (Caltech), Pasadena, CA, USA. He was a Professor at the University of Wisconsin, Madison, WI, USA. He is currently with the University of Pennsylvania, Philadelphia, PA, USA. He supervised more than 80 Ph.D. students.

Dr. Morari is a fellow of AIChE, IFAC, and the U.K. Royal Academy of Engineering. He is a member of the U.S. National Academy of Engineering. He was a recipient of numerous awards, including Eckman, Ragazzini, and Bellman Awards from the American Automatic Control Council (AACC); Colburn, Professional Progress, and CAST Division Awards from the American Institute of Chemical Engineers (AIChE); Control Systems Award and Bode Lecture Prize from IEEE; Nyquist Lectureship and Oldenburger Medal from the American Society of Mechanical Engineers (ASME); and the IFAC High Impact Paper Award. He was the President of the European Control Association. He served on the technical advisory boards of several major corporations.

George J. Pappas (S'90–M'91–SM'04–F'09) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1998. He is currently the Joseph Moore Professor and Chair of the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He also holds a secondary appointment with the Department of Computer and Information Sciences and the Department of Mechanical Engineering and Applied Mechanics. He is a member of the GRASP Lab and the PRECISE Center. He had previously served as the Deputy Dean for Research with the School of Engineering and Applied Science. His research interests include control theory and, in particular, hybrid systems, embedded systems, cyberphysical systems, and hierarchical and distributed control systems, with applications to unmanned aerial vehicles, distributed robotics, green buildings, and biomolecular networks. Dr. Pappas has received various awards, such as the Antonio Ruberti Young Researcher Prize, the George S. Axelby Award, the Hugo Schuck Best Paper Award, the George H. Heilmeier Award, the National Science Foundation PECASE award, and numerous best student papers awards at ACC, CDC, and ICCPS.