Proceedings of Machine Learning Research vol xxx:1–13, 2021

Certifying Incremental Quadratic Constraints for Neural Networks via Convex Optimization

Navid Hashemi NAVID.HASHEMI@UTDALLAS.EDU
Mechanical Engineering, University of Texas at Dallas

Justin Ruths JRUTHS@UTDALLAS.EDU
Mechanical Engineering, University of Texas at Dallas

Mahyar Fazlyab MAHYARFAZLYAB@JHU.EDU
Mathematical Institute for Data Science, Johns Hopkins University

Abstract

Abstracting neural networks with constraints they impose on their inputs and outputs can be very useful in the analysis of neural network classifiers and to derive optimization-based algorithms for certification of stability and robustness of feedback systems involving neural networks. In this paper, we propose a convex program, in the form of a Linear Matrix Inequality (LMI), to certify incremental quadratic constraints on the map of neural networks over a region of interest. These certificates can capture several useful properties such as (local) Lipschitz continuity, one-sided Lipschitz continuity, invertibility, and contraction. We illustrate the utility of our approach in two different settings. First, we develop a semidefinite program to compute guaranteed and sharp upper bounds on the local Lipschitz constant of neural networks and illustrate the results on random networks as well as networks trained on MNIST. Second, we consider a linear time-invariant system in feedback with an approximate model predictive controller parameterized by a neural network. We then turn the stability analysis into a semidefinite feasibility program and estimate an ellipsoidal invariant set for the closed-loop system.

Keywords: Neural Networks, Convex Optimization, Linear Matrix Inequalities, Semidefinite Programming

1. Introduction

Due to their ability to capture complex dependencies, Deep Neural Networks (DNNs) have been tremendously successful at various learning tasks such as image classification and learning-based control. Despite this success, the complex structure of neural networks makes them hard to analyze and therefore, they are often used without formal guarantees. For instance, the fragility of deep neural networks to uncertainties and adversarial attacks has raised serious concerns about their adoption in safety-critical applications such as autonomous vehicles. In response, there has been a growing interest in defining an appropriate notion of robustness and building defenses to improve it. Among several measures of robustness is the Lipschitz constant of neural networks, which by definition quantify the sensitivity of the output of the neural network to input perturbations. Knowing this constant is instrumental in several applications, such as robustness certification of classifiers Weng et al. (2018b), stability and safety analysis of deep reinforcement learning controllers, deriving generalization bounds Bartlett et al. (2017); Bolcskei et al. (2019); Sokolić et al. (2017); Neyshabur

et al. (2017), and perception-based robust control Dean et al. (2019). However, an accurate estimation of this constant can be quite challenging and has spurred significant interest recently.

More generally, describing neural networks with constraints they impose on their inputs and outputs (e.g., Lipschitz continuity) can be very useful in the analysis of neural networks and to derive optimization-based algorithms for certification of stability and robustness of neural-network-driven feedback systems. In particular, quadratic constraints can be naturally incorporated into existing methods for analysis and design of feedback systems via matrix inequalities Boyd et al. (1994). Motivated by this vision, in this paper we propose an LMI to certify a class of quadratic constraints on the map of neural networks over a region of interest. These certificates can capture several useful properties such as (local) Lipschitz continuity, one-sided Lipschitz continuity, invertibility, contraction, etc. We illustrate the utility of our approach in two different settings. First, we develop a semidefinite program (SDP) to compute guaranteed and sharp upper bounds on the local Lipschitz constant of neural networks and illustrate the results on random networks as well as networks trained on MNIST. Comparisons with the existing methods to bound the local Lipschitz constant reveal that our method is more accurate and more scalable at the same time. Second, we consider a linear time-invariant system in feedback with a neural network that approximates an explicit model predictive control law and turn the stability analysis into an SDP. More specifically, we compute an ellipsoidal invariant set around the equilibrium point of the closed-loop system.

1.1. Related Work

The use of quadratic constraints has a rich history in robust control and leveraged as a tool to abstract nonlinearities, time variations, unmodeled dynamics, and uncertain parameters by the constraints they impose on their inputs and outputs Yakubovich (1992); Megretski and Rantzer (1997); Zames (1966). Recently, quadratic constraints have been used and adapted for safety verification of neural networks Fazlyab et al. (2019a); Raghunathan et al. (2018), estimation of their Lipschitz constants Fazlyab et al. (2019b), and reachability and stability analysis of feedback systems with neural network controllers Hu et al. (2020); Yin et al. (2020); Jin and Lavaei (2018).

Lipschitz constant esimtation of neural networks. The value of computing the local Lipschitz constant of a neural network is underscored by the variety of techniques that have been developed to approach the problem Weng et al. (2018a); Avant and Morgansen (2020); Weng et al. (2018b); Virmaux and Scaman (2018); Fazlyab et al. (2018); Latorre et al. (2020). Weng et al. (2018a) provide upper bounds (FastLip) by propagating interval-bounds using linear approximations of each neuron depending on whether they are active, inactive, or both over the local region. In Fazlyab et al. (2019b) the authors propose an SDP called LipSDP that computes guaranteed upper bounds on the *global* Lipschitz constant of deep neural networks. Latorre et al. (2020) proposed a polynomial optimization framework to bound the local Lipschitz constant (LiPopt) for sparse networks that employ smooth activation functions. Chen et al. (2020) extended this polynomial optimization approach (LipOpt) to handle ReLU networks by defining generalized derivatives using a set of semialgebraic constraints. Jordan and Dimakis (2020) presents a Mixed-Integer Programming formulation (LipMIP) that computes the exact local Lipschitz constant of a ReLU network and provides a direct linear relaxation (LipLP). Scalability of these methods or the structure imposed to provide scalability is a consistent challenge that competes with the conservatism of the local Lipschitz upper bounds - MIP optimizations become intractable quickly; the sparsity required for LiPopt and LipOpt enables reducing the size of the corresponding linear program. Many of these methods

exploit specific features of the activation functions (differentiability, piece-wise linearity, etc) or network topologies (sparsity, scalar-valued network), but limit the scope of networks for which they apply. LipSDP admits an approach that is broadly applicable to most activation functions and the semi-definite programs provide a computational tractability that is practical.

Notation. We denote 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 set of $m \times n$-dimensional matrices with non-negative components by $\mathbb{R}_+^{m\times n}$, 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. We denote the p-norm ($p \geq 1$) by $\|\cdot\|_p \colon \mathbb{R}^n \to \mathbb{R}_+$. We denote the quadratic norm, induced by $P \in \mathbb{S}_{++}^n$, with $\|x\|_P = \sqrt{x^\top P x}$. We denote the i-th unit vector in $\mathbb{R}^n$ by e_i. We write $\mathrm{diag}(a_1, ..., a_n)$ for a diagonal matrix whose diagonal entries starting in the upper left corner are $a_1, \cdots, a_n$. We denote the hadamard product between two matrices A, B by $A \circ B$.

2. Problem Statement

Consider a function $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_f}$ parameterized by a feed-forward neural network of the form

$$x_0 = x \quad x_{k+1} = \phi_k(W_k x_k + b_k) \quad k = 0, \cdots, \ell - 1 \quad f(x_0) = W_\ell x_\ell + b_\ell, \tag{1}$$

where $W_k \in \mathbb{R}^{n_{k+1}\times n_k}$, $b_k \in \mathbb{R}^{n_{k+1}}$ are the weight and bias of the k-th layer, $n_0 = n_x$, $n_f = n_{\ell+1}$ and $\phi_k \colon \mathbb{R}^{n_{k+1}} \to \mathbb{R}^{n_{k+1}}$ is the layer of activation functions. We denote by $n = \sum_{k=1}^{\ell} n_k$ the total number of neurons. Given two closed sets $\mathcal{X}_0, \mathcal{Y}_0 \subset \mathbb{R}^{n_0}$ in the input space and a symmetric and indefinite matrix $Q_f \in \mathbb{S}^{n_x+n_f}$, we would like to verify that

$$\begin{bmatrix} x_0 - y_0 \\ f(x_0) - f(y_0) \end{bmatrix}^\top Q_f \begin{bmatrix} x_0 - y_0 \\ f(x_0) - f(y_0) \end{bmatrix} \geq 0 \quad \forall x_0, y_0 \in \mathcal{X}_0 \times \mathcal{Y}_0. \tag{2}$$

In this paper, we develop an LMI whose feasibility leads to the certificate (2) for a given Q_f. We now provide two concrete applications in which the inequalities of the form (2) appear explicitly.

2.1. Robustness Certification of Neural Network Classifiers

Consider that f functions as a n_f-class classifier in which the data $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$ is assigned the class label $i^*(x) = \arg\max_i f_i(x)$, where f_i is the i-th component of f. The robustness of f can be quantified through the adversarial (worst-case) perturbation of minimum norm that is able to change the assigned class label of the point x, i.e., $\epsilon^*(x) = \{\inf_\epsilon \|\epsilon\| \text{ s.t. } i^*(x+\epsilon) \neq i^*(x)\}$ Fawzi et al. (2016); Peck et al. (2017). One technique to identify $\epsilon^*(x)$ is to identify the largest (ℓ_2) ball in the output space centered at $f(x)$ that maintains the same classification with radius $\rho = \min_{i\neq i^*} \frac{1}{\sqrt{2}} \left|(e_{i^*} - e_i)^\top f(x)\right|$ Fazlyab et al. (2019b). If f is locally Lipschitz with constant L_f, then it is possible project the ball with radius ρ back into the input space using the Lipschitz constant. This provides a lower bound on the adversarial perturbation $\epsilon^*(x) \geq \rho / L_f$.

Certifying quadratic inequalities in the form of (2) enables this analysis because the local Lipshitz continuity of f on $\mathcal{C}_0 \subset \mathbb{R}^{n_x}$ with Lipschitz constant L_f is equivalent to

$$\begin{bmatrix} x_0 - y_0 \\ f(x_0) - f(y_0) \end{bmatrix}^\top \begin{bmatrix} L_f^2 I_{n_x} & 0 \\ 0 & -I_{n_f} \end{bmatrix} \begin{bmatrix} x_0 - y_0 \\ f(x_0) - f(y_0) \end{bmatrix} \geq 0 \quad \forall x_0, y_0 \in \mathcal{C}_0 \times \mathcal{C}_0. \tag{3}$$

2.2. Stability of Neural Network Controlled Systems

Consider an LTI system in feedback with a neural network controller, $x^+ = Ax + Bf(x)$. Suppose $x^\star \in \mathbb{R}^{n_x}$ is an equilibrium of the closed loop system, that is $x^\star = Ax^\star + Bf(x^\star)$. By defining a quadratic Lyapunov function $V(x) = (x - x^\star)^\top P(x - x^\star)$ with $P \in \mathbb{S}^{n_x}_{++}$, local geometric stability of the closed-loop system on $\mathcal{D} \subset \mathbb{R}^{n_x}$ (which contains $x^\star$) is implied by the condition $V(x_+) \leq \rho V(x)$ for all $x \in \mathcal{D}$, where $\rho \in (0, 1)$ is the convergence rate Haddad and Chellaboina (2011). We can then write

$$V(x^+) - \rho V(x) = \begin{bmatrix} x - x^\star \\ f(x) - f(x^\star) \end{bmatrix}^\top \begin{bmatrix} A^\top P A - \rho P & PB \\ B^\top P & B^\top P B \end{bmatrix} \begin{bmatrix} x - x^\star \\ f(x) - f(x^\star) \end{bmatrix}. \tag{4}$$

Therefore, a sufficient condition for local geometric stability is that the right-hand side is non-positive for all $x \in \mathcal{D}$.

3. Canonical Representation of Nonlinearities

The building block of our method is local incremental quadratic constraints, or δQC for short, which aim to describe nonlinear functions/operators incrementally with respect two arbitrary inputs. A formal definition is as follows, which is inspired by the definition in Açıkmeşe and Corless (2011).

Definition 1 (Local Incremental Quadratic Constraint) Let $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^n$ be two closed sets. We say the function $\phi\colon \mathbb{R}^n \to \mathbb{R}^n$ satisfies the local incremental quadratic constraint defined by $(\mathcal{X}, \mathcal{Y}, \mathcal{Q})$ if for any $Q \in \mathcal{Q} \subset \mathbb{S}^{2n}$ we have

$$\begin{bmatrix} x - y \\ \phi(x) - \phi(y) \end{bmatrix}^\top Q \begin{bmatrix} x - y \\ \phi(x) - \phi(y) \end{bmatrix} \geq 0 \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y}. \tag{5}$$

Note that $\mathcal{Q}$ is a convex set of all matrices that characterize ϕ incrementally with respect to two arbitrary points $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Further, if ϕ satisfies the δQC defined by $(\mathcal{X}, \mathcal{Y}, \mathcal{Q})$, it also satisfies the δQC defined by $(\bar{\mathcal{X}}, \bar{\mathcal{Y}}, \mathcal{Q})$ for any non-empty $\bar{\mathcal{X}} \subseteq \mathcal{X}$ and $\bar{\mathcal{Y}} \subseteq \mathcal{Y}$.

In the sequel, we elaborate on characterizing activation layers in neural networks via δQCs. For simplicity, we consider the case $\mathcal{X} = \mathcal{Y}$. Extensions to the more general case would be similar.

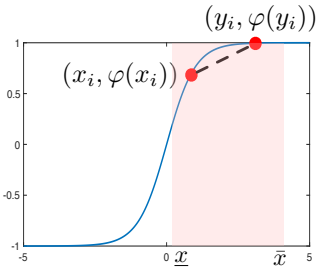
3.1. Smooth Activation Functions

We start with describing a single activation function over a bounded interval by δQCs. See Figure 1 for an illustration.

Lemma 2 *Let $\varphi\colon \mathbb{R} \to \mathbb{R}$ be continuous on $\mathcal{X} := [\underline{x},\ \bar{x}]$ and differentiable on $(\underline{x},\ \bar{x})$. Define $\alpha = \inf_{x \in (\underline{x},\ \bar{x})}\ \varphi'(x)$ and $\beta = \sup_{x \in (\underline{x},\ \bar{x})}\ \varphi'(x)$. Then φ satisfies the incremental quadratic constraint defined by $(\mathcal{X}, \mathcal{X}, \mathcal{Q})$ where*

$$\mathcal{Q} = \{Q \mid Q = \begin{bmatrix} -2\alpha\beta\lambda & (\alpha + \beta)\lambda \\ (\alpha + \beta)\lambda & -2\lambda \end{bmatrix},\ \lambda \geq 0\}. \tag{6}$$

Proof Using the mean-value theorem on the interval $[x, y] \subseteq [\underline{x}, \bar{x}]$, we have $\varphi'(c) = \frac{\varphi(y) - \varphi(x)}{y - x}$ for some $c \in (x, y)$. Therefore, we can write $\alpha \leq \inf_{c \in (x,y)}\ \varphi'(c) \leq \frac{\varphi(y) - \varphi(x)}{y - x} \leq \sup_{c \in (x,y)}\ \varphi'(c) =$



$$\alpha = \varphi'(\bar{x}) \leq \frac{\varphi(y_i) - \varphi(x_i)}{y_i - x_i} \leq \varphi'(\underline{x}) = \beta$$

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

Figure 1: Local incremental quadratic constraint for $\varphi(x)$ on $[\underline{x}, \bar{x}]$.

β. These inequalities can be written equivalently as $\lambda(\frac{\varphi(y)-\varphi(x)}{y-x} - \alpha)(\frac{\varphi(y)-\varphi(x)}{y-x} - \beta) \leq 0$, where $\lambda \geq 0$ is arbitrary. Therefore,

$$\begin{bmatrix} x - y \\ \varphi(x) - \varphi(y) \end{bmatrix}^\top \begin{bmatrix} -2\alpha\beta\lambda & (\alpha + \beta)\lambda \\ (\alpha + \beta)\lambda & -2\lambda \end{bmatrix} \begin{bmatrix} x - y \\ \varphi(x) - \varphi(y) \end{bmatrix} \geq 0 \quad \forall x, y \in [\underline{x}, \bar{x}].$$

■

Next, we extend the previous lemma to multi-variable nonlinearities.

Lemma 3 *Let $\phi(x) = (\varphi_1(x_1), \cdots, \varphi_n(x_n))$, $x \in \mathcal{X} \subseteq \mathbb{R}^n$, where all φ_i's are differentiable. Define $\alpha_i = \inf_{x \in \mathcal{X}} \varphi_i'(x_i)$ and $\beta_i = \sup_{x \in \mathcal{X}} \varphi_i'(x_i)$. Then ϕ satisfies the δQC defined by $(\mathcal{X}, \mathcal{X}, \mathcal{Q})$, where*

$$\mathcal{Q} = \{Q \mid Q = \begin{bmatrix} -2\operatorname{diag}(\alpha \circ \beta \circ \lambda) & \operatorname{diag}((\alpha + \beta) \circ \lambda) \\ \operatorname{diag}((\alpha + \beta) \circ \lambda) & -2\operatorname{diag}(\lambda) \end{bmatrix}, \ \lambda \in \mathbb{R}^n_+\}. \tag{7}$$

Proof By left- and right multiplying Q by $\begin{bmatrix} (x - y)^\top & (\phi(x) - \phi(y))^\top \end{bmatrix}^\top$ and its transpose, respectively, and using Lemma 2 we obtain

$$\begin{aligned} &\sum_{i=1}^n \lambda_i \left(-2\alpha_i\beta_i(x_i - y_i)^2 + 2(\alpha_i + \beta_i)(x_i - y_i)(\varphi_i(x_i) - \varphi_i(y_i)) - 2(\varphi_i(x_i) - \varphi_i(y_i))^2\right) \\ &= \sum_{i=1}^n \lambda_i \begin{bmatrix} x_i - y_i \\ \varphi_i(x_i) - \varphi_i(y_i) \end{bmatrix}^\top \begin{bmatrix} -2\alpha_i\beta_i & \alpha_i + \beta_i \\ \alpha_i + \beta_i & -2 \end{bmatrix} \begin{bmatrix} x_i - y_i \\ \varphi_i(x_i) - \varphi_i(y_i) \end{bmatrix} \geq 0. \end{aligned}$$

■

3.2. Piecewise Linear Activation Functions

Since the Rectified Linear Unit (ReLU) function is not differentiable, the result of Lemma 2 is not directly applicable. Therefore, we characterize (leaky) ReLU functions separately.

Lemma 4 *Let $\phi(x) = \max(\alpha x, \beta x)$, $x \in \mathcal{X} \subseteq \mathbb{R}^n, 0 \leq \alpha \leq \beta < \infty$ 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 on $\mathcal{X}$, i.e., $\mathcal{I}^+ = \{i \mid x_i \geq 0 \text{ for all } x \in \mathcal{X}\}$, $\mathcal{I}^- = \{i \mid x_i < 0 \text{ for all } x \in \mathcal{X}\}$, and $\mathcal{I}^\pm = \{1, \cdots, n\} \setminus (\mathcal{I}^+ \cup \mathcal{I}^-)$. Define $\boldsymbol{\alpha} = [\alpha + (\beta - \alpha)\mathbf{1}_{\mathcal{I}^+}(1), \cdots, \alpha + (\beta - \alpha)\mathbf{1}_{\mathcal{I}^+}(n)]$ and $\boldsymbol{\beta} =$*

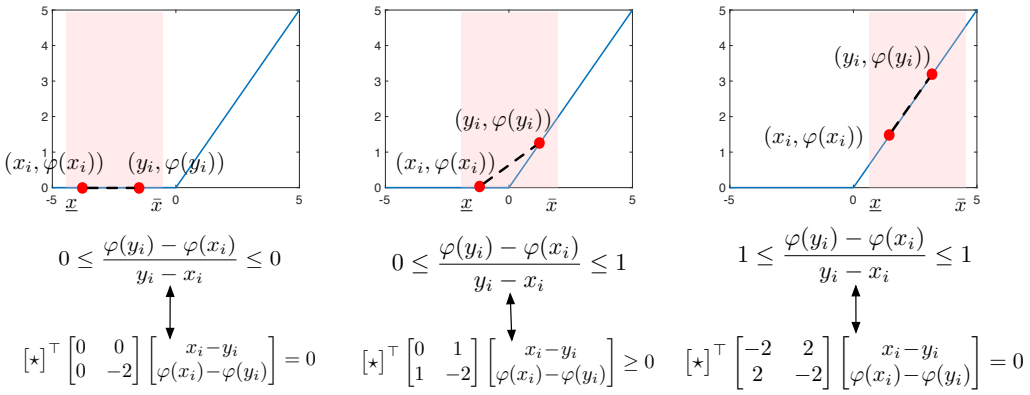


Figure 2: Local incremental quadratic constraint for $\varphi(x) = \max(0, x)$ defined on $[\underline{x}, \bar{x}]$.

$[\beta - (\beta - \alpha)\mathbf{1}_{\mathcal{I}^-}(1), \cdots, \beta - (\beta - \alpha)\mathbf{1}_{\mathcal{I}^-}(n)]$ *Then* ϕ *satisfies the* δQC *defined by* $(\mathcal{X}, \mathcal{X}, \mathcal{Q})$, *where*

$$\mathcal{Q} = \{Q \mid Q = \begin{bmatrix} -2\operatorname{diag}(\boldsymbol{\alpha} \circ \boldsymbol{\beta} \circ \lambda) & \operatorname{diag}((\boldsymbol{\alpha} + \boldsymbol{\beta}) \circ \lambda) \\ \operatorname{diag}((\boldsymbol{\alpha} + \boldsymbol{\beta}) \circ \lambda) & -2\operatorname{diag}(\lambda) \end{bmatrix}, \ \lambda_i \in \mathbb{R}_+ \ \textit{for} \ i \in \mathcal{I}^{\pm}\}. \tag{8}$$

Proof By left- and right multiplying Q by $\begin{bmatrix}(x-y)^\top & (\phi(x)-\phi(y))^\top\end{bmatrix}^\top$ and $\begin{bmatrix}(x-y)^\top & (\phi(x)-\phi(y))^\top\end{bmatrix}^\top$, we obtain

$$\begin{aligned}
&\sum_{i=1}^{n} \lambda_i \left(-2\alpha_i\beta_i(x_i - y_i)^2 + 2(\alpha_i + \beta_i)(x_i - y_i)(\varphi_i(x_i) - \varphi_i(y_i)) - 2(\varphi_i(x_i) - \varphi_i(y_i))^2\right) \\
&= \sum_{i \in \mathcal{I}^+} \lambda_i \underbrace{\left(-2\alpha_i\beta_i(x_i - y_i)^2 + 2(\alpha_i + \beta_i)(x_i - y_i)^2 - 2(x_i - y_i)^2\right)}_{=0 \text{ (since } \alpha_i = \beta_i = 1)} \\
&+ \sum_{i \in \mathcal{I}^-} \lambda_i \underbrace{\left(-2\alpha_i\beta_i(x_i - y_i)^2 + 2(\alpha_i + \beta_i)(x_i - y_i)(0 - 0) - 2(0 - 0)^2\right)}_{=0 \text{ (since } \alpha_i = \beta_i = 0)} \\
&+ \sum_{i \in \mathcal{I}^\pm} \lambda_i \underbrace{\begin{bmatrix} x_i - y_i \\ \phi_i(x_i) - \phi_i(y_i) \end{bmatrix}^\top \begin{bmatrix} -2\alpha_i\beta_i & \alpha_i + \beta_i \\ \alpha_i + \beta_i & -2 \end{bmatrix} \begin{bmatrix} x_i - y_i \\ \phi_i(x_i) - \phi_i(y_i) \end{bmatrix}}_{\ge 0},
\end{aligned}$$

where the last inequality follows from the fact that $\alpha_i \le \frac{\phi_i(x_i)-\phi_i(y_i)}{x_i - y_i} \le \beta_i$ when $i \in \mathcal{I}^\pm$. ■

4. Certifying Local Incremental Quadratic Constraints via Semidefinite Programming

In Theorem 5 we combine the δQCs of individual layers to derive an LMI for certifying a δQC for the entire neural network. Before stating the result, we define $\mathbf{x} = [x_0{}^\top \cdots x_\ell^\top]^\top \in \mathbb{R}^{n_0+n}$ and the entry selector matrices E_k such that $x_k = E_k\mathbf{x}$ for $k = 0, \cdots, \ell$.

Theorem 5 *For the neural network in (1), define the reachable sets of the pre-activation vectors as* $\mathcal{D}_k = \{W_k x_k + b_k \mid x_i = \phi(W_{i-1}x_{i-1} + b_{i-1}),\ i = 1, \cdots, k,\ x_0 \in \mathcal{C}_0\}$, $k = 0, \cdots, \ell - 1$. *Suppose each nonlinear layer* ϕ_k *satisfies the local incremental quadratic constraint defined by* $(\mathcal{D}_k, \mathcal{D}_k, \mathcal{Q}_k)$. *For a given* $Q_f \in \mathbb{S}^{n_x+n_f}$, *consider the following LMI,*

$$M(Q_0, \cdots, Q_{\ell-1}, Q_f) = \sum_{k=0}^{\ell-1} \begin{bmatrix} W_k E_k \\ E_{k+1} \end{bmatrix}^\top Q_k \begin{bmatrix} W_k E_k \\ E_{k+1} \end{bmatrix} - \begin{bmatrix} E_0 \\ W_\ell E_\ell \end{bmatrix}^\top Q_f \begin{bmatrix} E_0 \\ W_\ell E_\ell \end{bmatrix}, \tag{9}$$

If $M(Q_0, \cdots, Q_{\ell-1}, Q_f) \preceq 0$ *for some* $(Q_0, \cdots, Q_{\ell-1}) \in \mathcal{Q}_0 \times \cdots \times \mathcal{Q}_{\ell-1}$, *then*

$$\begin{bmatrix} x - y \\ f(x) - f(y) \end{bmatrix}^\top Q_f \begin{bmatrix} x - y \\ f(x) - f(y) \end{bmatrix} \geq 0 \quad \forall x, y \in \mathcal{C}_0. \tag{10}$$

Proof Suppose $M(Q_0, \cdots, Q_{\ell-1}, Q_f) \preceq 0$ for some $(Q_0, \cdots, Q_{\ell-1}) \in \mathbb{R}_+ \times \mathcal{Q}_0 \times \cdots \times \mathcal{Q}_{\ell-1}$. By left- and right- multiplying both sides of (9) by $(\mathbf{x} - \mathbf{y})^\top$ and $(\mathbf{x} - \mathbf{y})$, respectively, we obtain

$$\sum_{k=0}^{\ell-1} \begin{bmatrix} W_k(x_k - y_k) \\ x_{k+1} - y_{k+1} \end{bmatrix}^\top Q_k \begin{bmatrix} W_k(x_k - y_k) \\ x_{k+1} - y_{k+1} \end{bmatrix} - \begin{bmatrix} x_0 - y_0 \\ f(x_0) - f(y_0) \end{bmatrix}^\top Q_f \begin{bmatrix} x_0 - y_0 \\ f(x_0) - f(y_0) \end{bmatrix} \leq 0.$$

By assumption, for each $k = 0, \cdots, \ell - 1$, the function $x \mapsto \phi_k \circ (W_k x + b_k)$ satisfies the δQC defined by $(\mathcal{D}_k, \mathcal{D}_k, \mathcal{Q}_k)$. Thus, all summands are non-negative, and as a result, we arrive at (9). ■

When specialized to certifying local Lipschitz continuity (see (3)), Theorem 5 essentially reduces to the local version of the main result of Fazlyab et al. (2019b). In Fazlyab et al. (2019b) it is assumed that $\mathcal{D}_k = \mathbb{R}^{n_{k+1}}$ and hence, the resulting bound is for the global Lipschitz constant. In contrast, we restrict the space of multipliers to the reachable sets $\mathcal{D}_k$ to find a bound on the local Lipschitz constant. We describe this procedure next.

4.1. Computation of Pre-activation Bounds

Instead of finding the reachable sets $\mathcal{D}_k$ exactly, which is computationally prohibitive, we over approximate them by hyper-rectangles, i.e, we seek to find $l^k, u^k \in \mathbb{R}^{n_{k+1}}$, $k = 0, 1, 2, \cdots \ell - 1$ such that $l^k \leq W_k x_k + b_k \leq u^k$. Here we provide an approach based on the idea presented in Zhang et al. (2018) to obtain the pre-activation bounds of the current layer given the pre-activation bounds of the previous layer. Specifically, given the reachable set $\mathcal{D}_{k-1}$ we truncate the activation functions on all the neurons and provide two optimal linear functions as the lower and upper bounds. We denote these linear functions by $h^k_{L,i}$, $h^k_{U,i}$, $i = 1, 2, \cdots, n_k$ which specify the lower and upper bounds on the activation function for the i-th neuron located in the k-th layer, as depicted in Fig. 3.

Table 2 in Zhang et al. (2018) provides a comprehensive characterization of the linear functions $h^k_{L,i}(x) = \alpha^k_{L,i}(x + \beta^k_{L,i})$ and $h^k_{U,i}(x) = \alpha^k_{U,i}(x + \beta^k_{U,i})$ based on the pre-activation interval $x \in [l^{k-1}_i, u^{k-1}_i]$. Concatenating the slopes and intercepts of the k-th layer $\alpha^k_{L,i}$, $\alpha^k_{U,i}$, $\beta^k_{L,i}$, and $\beta^k_{U,i}$ into the vectors $\alpha^k_L, \alpha^k_U, \beta^k_L$ and β^k_U, respectively, we then devise Algorithm 1 to iteratively compute the pre-activation bounds over all layers, with the following matrix definitions

$$\begin{aligned} \underline{C}_k &= W_k^+ \operatorname{diag}(\alpha^k_L) + W_k^- \operatorname{diag}(\alpha^k_U), \quad \overline{C}_k = W_k^+ \operatorname{diag}(\alpha^k_U) + W_k^- \operatorname{diag}(\alpha^k_L), \\ \underline{d}_k &= W_k^+(\alpha^k_L \circ \beta^k_L) + W_k^-(\alpha^k_U \circ \beta^k_U) + b_k, \quad \overline{d}_k = W_k^+(\alpha^k_U \circ \beta^k_U) + W_k^-(\alpha^k_L \circ \beta^k_L) + b_k. \end{aligned} \tag{11}$$

with $W_k^+ = \frac{W_k + |W_k|}{2}$ and $W_k^- = \frac{W_k - |W_k|}{2}$. We provide a detailed description of this approach in Appendix A, including the differences when compared with Zhang et al. (2018).

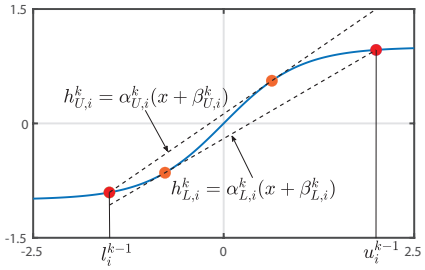


Figure 3: The activation function of neuron i in layer k is lower- and upper-bounded by the linear functions $h^k_{L,i}$ and $h^k_{U,i}$, respectively, over the pre-activation interval $[l_i^{k-1}, u_i^{k-1}]$.

Algorithm 1 : Computing pre-activation bounds

Input: $\mathcal{C}_0 := [\underline{x}_0, \overline{x}_0]$

Result: $\mathcal{D}_k := \left[l^k,\ u^k\right], k = 0, \cdots, \ell - 1.$

$l^0 = -|W_0|\frac{\overline{x}_0 - \underline{x}_0}{2} + W_0\frac{\overline{x}_0 + \underline{x}_0}{2} + b_0$

$u^0 = |W_0|\frac{\overline{x}_0 - \underline{x}_0}{2} + W_0\frac{\overline{x}_0 + \underline{x}_0}{2} + b_0$

for $k \in \{1, 2, \cdots, \ell - 1\}$ **do**

$\quad l^k = -|\underline{C}_k|\frac{u^{k-1} - l^{k-1}}{2} + \underline{C}_k\frac{u^{k-1} + l^{k-1}}{2} + \underline{d}_k$

$\quad u^k = |\overline{C}_k|\frac{u^{k-1} - l^{k-1}}{2} + \overline{C}_k\frac{u^{k-1} + l^{k-1}}{2} + \overline{d}_k$

end

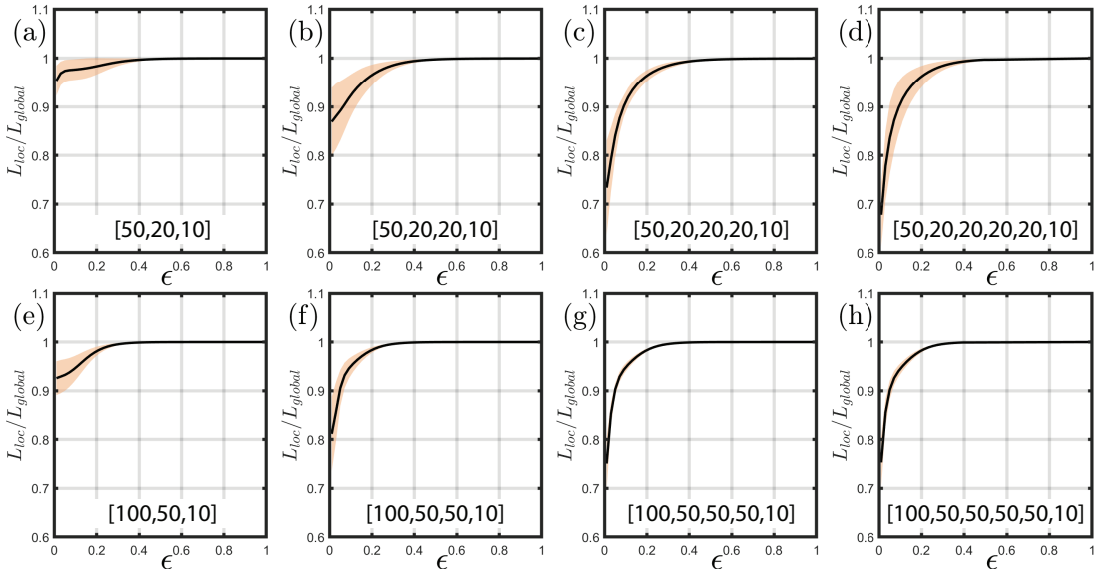


Figure 4: The local Lipschitz constant approaches the global value as the input set expands in random neural networks with various widths and depths (sizes inset). Black lines demarcate the average and shaded region the standard deviation over 10 different realizations.

5. Numerical Experiments

Local vs. Global Lipschitz Constants. We consider ℓ_∞ ball input sets $\mathcal{C}_0 = \{x_0 \mid \|x_0\|_\infty \leq \epsilon\}$ parameterized by ϵ. We then compute the ratio between the local and global Lipschitz constants as a function of increasing ϵ. In Figure 4, we plot the results for randomly generated (weights pulled from the normal distribution) hyperbolic tangent neural networks of various depths and widths.

MNIST Classification Robustness. Following the discussion in Section 2, we analyze the robustness of a neural network classifier trained on the MNIST Dataset. Calculating a local Lipschitz constant L_{loc} over a sufficiently large perturbation ϵ, the maximum input perturbation ϵ^* of the input data x (in ℓ_∞ norm) which does not change the classification is given by $\epsilon^* = \frac{\rho}{\sqrt{n_0}L_{\text{loc}}}$, where

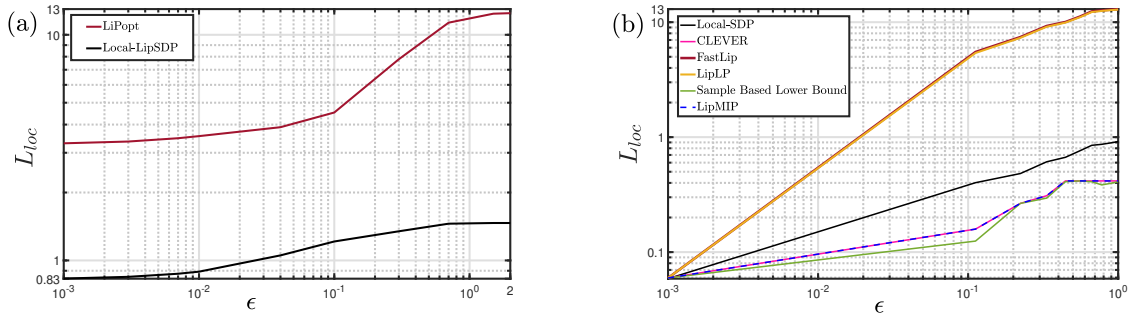


Figure 5: Local-LipSDP outperforms current methods on random tanh network $[20, 20, 20, 1]$ in (a) and ReLU network $[2, 100, 100, 2]$ in (b), chosen relatively small to avoid the scalability challenges and technical limitations of several of the other methods.

the rescaling factor accommodates changing from a ℓ_2 norm to an ℓ_∞ norm, motivated by pixel-wise perturbations. We train a ReLU network using the linear programming-based method of Wong and Kolter (2018) (with ℓ_∞ perturbation radius 0.05) with dimensions $[784, 100, 50, 50, 10]$. The input datapoint x is labeled as $i^* = 1$ and $\rho = 11.1427$. Because the local Lipschitz constant is a function of the input perturbation ϵ, we use a bisection algorithm to determine the ϵ for which $\sqrt{n_0} L_{\text{loc}} \epsilon = \rho$. This results in $\epsilon^* = 0.05892$ and $L_{\text{loc}} = 6.7529$. If we were to instead use the global Lipschitz constant, $L_{\text{global}} = 14.3764$, the certification radius becomes $\epsilon^* = 0.0277$ which is 53% smaller and underscores the enhancement possible using the local Lipschitz constant.

Comparison with Other Methods. Figure 5 provides a comparison of Local-LipSDP with various current methods for estimating the local Lipschitz constant over an expanding input set. The scalability and technical limitations of some of these methods restrict the sizes of networks we can consider. Nonetheless, the effectiveness of our methods is clear even on these relatively simple networks. In Figure 5a, Local-LipSDP compares well against LiPopt on a random tanh network with layer sizes $[20, 20, 20, 1]$ (LiPopt requires a single output, hence why we compare separately). In Figure 5, Local-LipSDP out performs FastLip and LipLP on a random ReLU network with layer sizes $[2, 100, 100, 2]$. LipMIP provides the exact value at the cost of solving a mixed-integer non-convex optimization, the scalability of which limits the sizes that can be considered. In this case the sampling approaches (naive lower bound, CLEVER) perform well because the network is small with low-dimension inputs and outputs. The performance of these sample-based methods degrades as the network complexity grows. The experiments in Figure 5 highlight the effectiveness of Local-LipSDP and its capability to handle the full-scale MNIST classifier in the previous experiments demonstrates its scalability and lack of technical limitations compared with other methods.

Stability analysis of an approximate MPC controller. Consider a double integrator system

$$x_{t+1} = Ax_t + Bu_t \quad A = 1.2 \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, \tag{12}$$

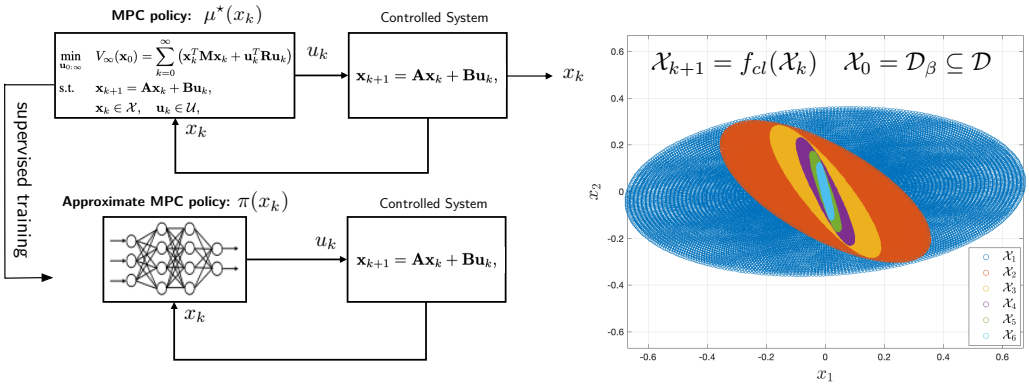


Figure 6: Illustration of an ellipsoidal invariant set computation for the approximate MPC example.

subject to the state and input constraints $x_t \in \mathcal{X} = \{x \mid \|x\|_\infty \leq 5\}$ and $u \in \mathcal{U} = \{u \mid \|u\|_\infty \leq 1\}$. Consider the finite horizon problem

$$\text{minimize} \quad \sum_{t=0}^{T} \|x_t\|_2^2 + u_t^2 \quad \text{s.t. } (x_t, u_t) \in \mathcal{X} \times \mathcal{U} \quad t = 0, \cdots, T, \; x_0 = x, \tag{13}$$

and the control law $\mu_{MPC}(x) = u_0^\star$. Instead of implementing this control policy, we implement $u_t = \pi(x_t) \approx \mu_{MPC}(x_t)$, where π is a ReLU network with architecture $[2, 32, 32, 1]$ that is trained off-line to approximate $\mu_{MPC}(x)$. For generating the training data, we compute $\mu_{MPC}(x)$ at 6284 uniformly chosen random points from the control invariant set. Our goal is to find the largest ellipsoidal invariant set inside the region of interest $\mathcal{D} = \{x \mid \|x\|_\infty \leq \epsilon\}$ containing the equilibrium point $x^\star = 0$. We first consider the quadratic Lyapunov function $V(x) = x^\top P x$ with $P \succ 0$ and a candidate invariant set $\mathcal{D}_\beta = \{x \in \mathcal{D} \mid V(x) \leq \beta\}$. Now if $V(Ax + B\pi(x)) \leq V(x)$ for all $x \in \mathcal{D}_\beta$, then every trajectory starting in $\mathcal{D}_\beta$ will remain inside $\mathcal{D}_\beta$. Therefore, the maximum value of β such that $\mathcal{D}_\beta \subseteq \mathcal{D}$ yields the maximum inner estimate of the positive invariant set. To find such a set, we use bisection to find the largest ϵ such that the right-hand side of (4) is non-positive for $\rho = 1$ and some positive definite P. We then find the maximum β such that $\mathcal{D}_\beta \subseteq \mathcal{D}$. For our problem data, we find $\epsilon^\star = 0.669$. In Figure 6, we plot the largest ellipsoidal invariant set (in blue) inside $\mathcal{D}$. We also plot the reachable sets $\mathcal{X}_k$ of the closed loop system from the initial set $\mathcal{X}_0 = \mathcal{D}_\beta$ to visualize the invariance of $\mathcal{D}_\beta$.

6. Conclusion

In this paper, we developed a convex program to certify incremental quadratic constraints on the map of neural networks over a region of interest. We illustrated the utility of our method in sharp estimation of local Lipschitz constant of neural networks as well as stability analysis of neural network controlled feedback systems via semidefinite programming. In principle, there is a trade-off between the conservatism, the run time, and the memory requirements of solving these convex programs: certifying tighter bounds typically requires more time and/or memory. Developing numerical algorithms that can span this trade-off is a future research direction.

References

Behçet Açıkmeşe and Martin Corless. Observers for systems with nonlinearities satisfying incremental quadratic constraints. *Automatica*, 47(7):1339–1348, 2011.

Trevor Avant and Kristi A Morgansen. Analytical bounds on the local lipschitz constants of affine-relu functions. *arXiv preprint arXiv:2008.06141*, 2020.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

Helmut Bolcskei, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1): 8–45, 2019.

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

Tong Chen, Jean B Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic optimization for lipschitz constants of relu networks. *Advances in Neural Information Processing Systems*, 33, 2020.

Sarah Dean, Nikolai Matni, Benjamin Recht, and Vickie Ye. Robust guarantees for perception-based control. *arXiv preprint arXiv:1907.03680*, 2019.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *Advances in Neural Information Processing Systems*, 29:1632–1640, 2016.

Mahyar Fazlyab, Alejandro Ribeiro, Manfred Morari, and Victor M Preciado. Analysis of optimization algorithms via integral quadratic constraints: Nonstrongly convex problems. *SIAM Journal on Optimization*, 28(3):2654–2689, 2018.

Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *arXiv preprint arXiv:1903.01287*, 2019a.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 11427–11438, 2019b.

Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton university press, 2011.

Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George J Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. *arXiv preprint arXiv:2004.07876*, 2020.

Ming Jin and Javad Lavaei. Stability-certified reinforcement learning: A control-theoretic perspective. *arXiv preprint arXiv:1810.11505*, 2018.

Matt Jordan and Alexandros G Dimakis. Exactly computing the local lipschitz constant of relu networks. *arXiv preprint arXiv:2003.01219*, 2020.

Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations*, 2020.

Alexandre Megretski and Anders Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, 1997.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

Jonathan Peck, Joris Roels, Bart Goossens, and Yvan Saeys. Lower bounds on the robustness to adversarial perturbations. In *Advances in Neural Information Processing Systems*, pages 804–813, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10900–10910, 2018.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844, 2018.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018a.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018b.

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

VA Yakubovich. Nonconvex optimization problem: The infinite-horizon linear-quadratic control problem with quadratic constraints. *Systems & Control Letters*, 19(1):13–22, 1992.

He Yin, Peter Seiler, and Murat Arcak. Stability analysis using quadratic constraints for systems with neural network controllers. *arXiv preprint arXiv:2006.07579*, 2020.

George Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE transactions on automatic control*, 11(2):228–238, 1966.

Huan Zhang, Tsui-Wei 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*, pages 4939–4948, 2018.

Appendix A. Details of Algorithm 1

By defining $z_k = W_k x_k + b_k$ as the input of the k-th hidden layer, the iterations of the neural network can be written as

$$z_k = W_k \phi(z_{k-1}) + b_k \tag{14}$$

Denote $W_k^+ = \frac{W_k + |W_k|}{2}$ which contains only the positive elements of W_k and $W_k^- = \frac{W_k - |W_k|}{2}$ where contains the negative elements. Therefore,

$$z_k = W_k^+ \phi(z_{k-1}) + W_k^- \phi(z_{k-1}) + b_k. \tag{15}$$

On the other hand,

$$\mathrm{diag}(\alpha_L^k) z_{k-1} + \alpha_L^k \circ \beta_L^k \leq \phi(z_{k-1}) \leq \mathrm{diag}(\alpha_U^k) z_{k-1} + \alpha_U^k \circ \beta_U^k, \tag{16}$$

which implies

$$\begin{aligned} W_k^+ \left(\mathrm{diag}(\alpha_L^k) z_{k-1} + \alpha_L^k \circ \beta_L^k\right) &\leq W_k^+ \phi(z_{k-1}) \leq W_k^+ \left(\mathrm{diag}(\alpha_U^k) z_{k-1} + \alpha_U^k \circ \beta_U^k\right), \\ W_k^- \left(\mathrm{diag}(\alpha_U^k) z_{k-1} + \alpha_U^k \circ \beta_U^k\right) &\leq W_k^- \phi(z_{k-1}) \leq W_k^- \left(\mathrm{diag}(\alpha_L^k) z_{k-1} + \alpha_L^k \circ \beta_L^k\right). \end{aligned} \tag{17}$$

If we define $\underline{C}_k, \underline{d}_k, \overline{C}_k$ and $\overline{d}_k$ as (11) then we conclude,

$$\underline{C}_k z_{k-1} + \underline{d}_k \leq z_k \leq \overline{C}_k z_{k-1} + \overline{d}_k. \tag{18}$$

Therefore to find u^k and l^k we need to solve two different linear programs, which respectively maximize and minimize a linear objective function $f^\top z_{k-1}$ such that $z_{k-1} \in [l^{k-1},\ u^{k-1}]$. This linear program has analytical solutions provided in the **for** loop embedded inside Algorithm 1. For the first hidden layer of the neural network, there is no activation function on the input layer so to find the reachable set of inputs in first layer, $\mathcal{D}_0$, we assume linear activation function for input layer, such that, $x_0 \in \mathcal{C}_0$ is considered as its output and compute $\mathcal{D}_0$. In this case $\alpha_L^0 = \alpha_U^0 = \mathbf{1}_{n_0}$ and $\beta_L^0 = \beta_U^0 = \mathbf{0}_{n_0}$.