2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

An Introduction to Neural Network Analysis via Semidefinite Programming

Mahyar Fazlyab[1], Manfred Morari[2], George J. Pappas[2]

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683096

***Abstract*— 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. This issue becomes even more complicated when neural networks are used in learning-enabled closed-loop systems, where a small perturbation can substantially impact the system being controlled. Therefore, it is of utmost importance to develop tools that can provide useful certificates of stability, safety, and robustness for neural network-driven systems.**

In this overview, we present a convex optimization framework for the analysis of neural networks. The main idea is to abstract hard-to-analyze components of a neural network (e.g., the nonlinear activation functions) with the formalism of quadratic constraints. This abstraction allows us to reason about various properties of neural networks (safety, robustness, generalization, stability in closed-loop settings, etc.) via semidefinite programming.

I. Introduction

Due to their ability to capture complex dependencies, neural networks have been tremendously successful at various complex tasks. Despite this success, the complex and large-scale structure of neural networks makes them hard to analyze and therefore, they are often used without formal guarantees. In particular, neural networks can be highly vulnerable to uncertainties in their input. This fragility becomes even worse when neural networks are used in feedback loops, where a small perturbation can substantially impact the closed-loop system over time. Therefore, a successful adoption of neural networks in safety-critical systems will require strict guarantees of stability, robustness, and safety.

Motivated by the lack of formal guarantees on the performance of neural networks, there has been an increasing effort to develop tools to verify desirable properties of neural networks. For example, in the context of image classification using deep networks, it is desirable to verify that all points in the vicinity of a correctly-classified example would be classified correctly (local robustness). Similarly, in deep reinforcement learning tasks, it is desirable to choose a robust action under a worst-case deviation in the input of the network due to possible adversaries or noise in the observed states [1].

The goal of this tutorial is to present a modular framework inspired by robust control and based on convex optimization for analyzing various properties of neural networks. The main idea is to abstract the nonlinearities in the neural network by the constraints they impose on all instances of their inputs and outputs. Then any property that we can guarantee for the abstracted network would hold for the original network as well. Notably, we show that we can capture various properties of nonlinear functions using Quadratic Constraints (QCs), such as bounded slope, bounded values, monotonicity, repeated nonlinearity across neurons, etc. Compared to linear constraints, quadratic constraints can reduce conservatism in the description of nonlinearities and can be naturally incorporated into existing methods for analysis and design of feedback systems via matrix inequalities [2].

The rest of the tutorial is organized as follows. In §II we discuss several problems surrounding neural networks, from verification of neural networks in isolation, to reachability analysis of neural network-controlled systems. In §III, we will provide a comprehensive introduction to Quadratic Constraints. In §IV we will use QCs to abstract neural networks. In §V we will show how we can tackle the problems discussed in §II via semidefinite programming.

This tutorial paper is based on the material developed in [3]–[7].

A. Notation and Preliminaries

We denote the set of real numbers by $\mathbb{R}$, the set of non-negative 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 $\mathcal{I} \times \mathcal{J}$. We denote ellipsoids by $\mathcal{E}(x_c, P) = \{x \mid (x - x_c)^\top P(x - x_c) \leq 1\}$, where $P \in \mathbb{S}^n_{++}$ and $x_c \in \mathbb{R}^n$ is the center of the ellipsoid.

II. Analysis of Neural Networks

A. Neural Network Verification

Consider the nonlinear vector-valued function $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_f}$ described by a multi-layer feed-forward neural network. Suppose a set $\mathcal{X} \subset \mathbb{R}^{n_x}$ of inputs is mapped by the neural network f to

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

The desirable properties that we would like to verify can often be represented by a safety specification set $\mathcal{S}_y$ in

[1] Department of Electrical and Computer Engineering, Johns Hopkins University. Email: mahyarfazlyab@jhu.edu
[2] Department of Electrical and Systems Engineering, University of Pennsylvania

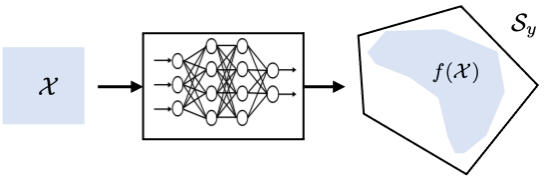


Fig. 1: Illustration of verifying whether $f(\mathcal{X}) \subseteq \mathcal{S}_y$.

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. This geometric description of verification can be generalized and connected to mathematical optimization. Specifically, given a function $J\colon \mathbb{R}^{n_f} \to \mathbb{R}$, we say that f satisfies the specification J if

$$J^\star := \sup_{x\in\mathcal{X}} \; J(f(x)) = \sup_{y\in\mathcal{Y}} \; J(y) \leq 0. \tag{1}$$

Here, the safety specification set $\mathcal{S}_y$ is the zero sublevel set of J, i.e., $\mathcal{S}_y = \{y \in \mathbb{R}^{n_f} \mid J(y) \leq 0\}$. From this perspective, one could consider more general specification functions that, for example, depend on both the network input x and the output $f(x)$ [8]. We will consider this scenario in §V-C, where we study reach-avoid verification problems for neural network-controlled systems.

To verify the inequality in (1), one must maximize the non-convex function $J(f(\cdot))$ over $\mathcal{X}$, or equivalently, maximize J over $\mathcal{Y}$, for its global solution $x^\star$ and verify whether the corresponding optimal value $J^\star$ is non-positive. Geometrically, this would require an exact computation of the non-convex set $\mathcal{Y}$. These methods are referred to as complete or exact verification, in which the verification method either verifies the specification ($J^\star \leq 0$) or finds a counterexample x that falsifies it ($J(x) > 0$).

For ReLU networks and affine J, (1) can be solved globally with Mixed-Integer Linear Programming (MILP) [9]–[12] or Satisfiability Modulo Theories (SMT) solvers [13]. While we do not expect these approaches to scale to large problems, for small neural networks and small input sets, they can still be practical.

To improve computational tractability, one can settle for incomplete or inexact verification, in which we find an upper bound $\bar{J}^\star$ on $J^\star$ and then we verify whether $\bar{J}^\star \leq 0$. Geometrically, this corresponds to overapproximating the output reachable set $\mathcal{Y}$ by a set $\tilde{\mathcal{Y}}$ and verifying the safety properties by checking the condition $\tilde{\mathcal{Y}} \subseteq \mathcal{S}_y$ instead. Indeed, incomplete verification methods may fail to verify a specification even if that specification is satisfied, i.e., we might have $J^\star \leq 0 < \bar{J}^\star$. However, we obtain computational tractability in return.

Incomplete verification methods are predominantly based on convex relaxations of (1), which can then be solved for their global solution via iterative methods for convex optimization [14]–[18]. Linear Programming (LP)-based relaxations are the most tractable form of convex relaxations that are suitable for large problem [16]–[18]. However, even solving LPs can become computationally prohibitive for small convolutional networks [18]. Furthermore, the LP-based upper bounds become loose for large input sets or large neural networks. This has motivated tighter convex relaxations at the expense of less scalability. In this tutorial, we are primarily interested in semidefinite relaxations [3]–[5], [14], [15], which are more computationally expensive but yield tighter relaxations.

B. Probabilistic Verification

In a deterministic setting, neural network verification is a yes/no problem whose answer does not quantify the proportion of inputs for which a property is violated. If the input perturbation is random and potentially unbounded, the output $f(x)$ would satisfy the desired specification only with a certain probability. More precisely, given a safe region $\mathcal{S}_y$ in the output space of the neural network, we are interested in finding the probability that the neural network maps a random input X to the safe region $\mathcal{S}_y$, $\mathbf{Pr}(f(X) \in \mathcal{S}_y)$. Since f is a nonlinear function, computing the distribution of $f(X)$ given the distribution of X is prohibitive, except for special cases. As a result, we settle for providing a *lower bound* $p \in (0, 1)$ on the desired probability,

$$\mathbf{Pr}(f(X) \in \mathcal{S}_y) \geq p.$$

To compute the lower bound, we can adopt a geometrical approach, in which we verify whether the reachable set of a *confidence region* of the input lies entirely in $\mathcal{S}_y$.

Definition 1 (Confidence region) *The p-level ($p \in [0, 1]$) confidence region of a vector random variable $X \in \mathbb{R}^n$ is defined as any set $\mathcal{E}_p \subseteq \mathbb{R}^n$ for which $\mathbf{Pr}(X \in \mathcal{E}_p) \geq p$.*

Let $\mathcal{E}_p$ be a confidence region of a random variable X. If $f(\mathcal{E}_p) \subseteq \mathcal{S}_y$, then $\mathcal{S}_y$ is a p-level confidence region for the random variable $f(X)$, i.e., $\mathbf{Pr}(f(X) \in \mathcal{S}_y) \geq p$. Therefore, if we can certify that the output reachable set $f(\mathcal{E}_p)$ lies entirely in the safe set $\mathcal{S}_y$ for some $p \in (0, 1)$, then the network is safe with probability at least p. In particular, finding the best lower bound corresponds to the optimization problem,

$$\text{maximize} \quad p \quad \text{subject to } f(\mathcal{E}_p) \subseteq \mathcal{S}_y,$$

with decision variable $p \in [0, 1)$.

A closely related problem to probabilistic safety verification is confidence propagation. Explicitly, given a p-level confidence region $\mathcal{E}_p$ of the input of a neural network, our goal is to find a p-level confidence region for the output. Of course, there is an infinite number of such possible confidence regions. Our goal is find the "best" confidence region with respect to some metric. Using the volume of the region as an optimization criterion, the best confidence region amounts to solving the problem

$$\text{minimize Volume}(\mathcal{S}) \quad \text{subject to } f(\mathcal{E}_p) \subseteq \mathcal{S}.$$

The solution to the above problem provides the p-level confidence region with the minimum volume.

C. Local Robustness Certification of Classifiers

Consider a data classification problem with n_f classes, where $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_f}$ classifies a data point x as $C(x) = \operatorname{argmax}_i f_i(x)$. To evaluate the local robustness of the neural network around a correctly-classified point $x^\star$ with label $i^\star = C(x^\star)$, 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 is the ℓ_∞ ball, $\mathcal{X} = \{x\colon \|x - x^\star\|_\infty \le \epsilon\}$, 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 $C(x) = C(x^\star) = i^\star$ for all $x \in \mathcal{X}$. For this problem, the safe set is the polytope

$$\mathcal{S}_y = \{y \in \mathbb{R}^{n_f} \mid y_{i^\star} > y_i \text{ for all } i \neq i^\star\},$$

Verifying the local robustness at $x^\star$ then amounts to verifying $f(\mathcal{X}) \subseteq \mathcal{S}_y$ or, equivalently, verifying that

$$\sup_{x\in\mathcal{X}} \{J_i(x) := f_i(x) - f_{i^\star}(x)\} < 0, \tag{2}$$

for all $i \neq i^\star$, i.e., the score of the class $i^\star$ must remain the largest for all input perturbations.

One way to verify (2) is by falsification, i.e., searching for an x that violates (2) by, for example, performing projected gradient method [19],

$$x \leftarrow \operatorname{Proj}_{\mathcal{X}}(x + \eta \nabla J_i(x)).$$

Since J is non-convex, we can only hope to find a local maximum. If this local solution does not falsify the specification, then we do not have a guarantee on the satisfaction of the specification. Therefore, we must either solve (2) globally or compute an upper bound $\bar{J}_i^\star$ on its optimal value. Then the local robustness property is verified when $\bar{J}_i^\star \le 0$ for all $i \neq i^\star$. In §V-A we will show how we can tackle this problem using semidefinite programming.

D. Robustness Certification via Smoothness

Defining an appropriate notion of robustness for neural networks is still an open question. Among several measures of robustness is the Lipschitz constant of neural networks, which by definition quantifies the sensitivity of the output of the neural network to input perturbations. Formally, a function $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_f}$ is said to be Lipschitz continuous on $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ if there exists a non-negative constant $L_f \ge 0$ such that

$$\|f(x) - f(\tilde{x})\| \le L_f \|x - \tilde{x}\| \quad \text{for all } x, \tilde{x} \in \mathcal{X}. \tag{3}$$

The smallest such L_f is called the (local) Lipschitz constant of f. When f is characterized by a neural network, tight bounds on its Lipschitz constant can be extremely useful in a variety of applications, such as robustness certification of classifiers [20], stability analysis of deep reinforcement learning controllers [21] and derivation of generalization bounds [22]. In these applications and many others, it is essential to have tight bounds on the Lipschitz constant of the neural network. Therefore, this problem has received significant attention recently [5], [20], [22]–[33].

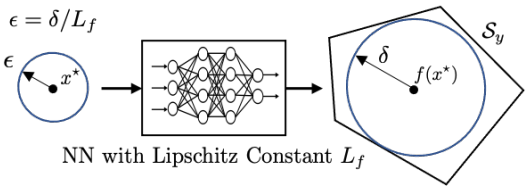


Fig. 2: Using the Lipschitz constant to certify local robustness for a classifier.

To see the utility of Lipschitz constant in robustness certification of classifiers, consider the classification example of §II-C. Suppose that $x^\star$ is an instance that is classified correctly by the neural network f with label $C(x^\star)$. The adversarial robustness of f can be quantified through the adversarial perturbation of minimum norm that is able to change the assigned class label of the point $x^\star$, i.e., $\epsilon^\star(x^\star) = \{\inf_\epsilon \|\epsilon\|_2 \text{ s.t. } C(x^\star + \epsilon) \neq C(x^\star)\}$ [34], [35]. One technique to identify $\epsilon^\star(x^\star)$ is to first compute the largest ball in the *output* space centered at $f(x^\star)$ that maintains the same classification. The radius of this ball is $\delta = \min_{i\neq i^\star} \frac{1}{\sqrt{2}} \left|(e_{i^\star} - e_i)^\top f(x^\star)\right|$. Using the Lipschitz constant, it is possible to project this ball back into the input space. This provides a lower bound on the adversarial perturbation $\epsilon^\star(x) \ge \delta/L_f$. See Fig. 2 for an illustration of the concept. In §V-D, we will show how we can find guaranteed upper bounds on the Lipschitz constant of neural networks using semidefinite programming.

E. Certified Robustness for Deep Reinforcement Learning

Another domain where evaluating robustness would be useful is robust deep reinforcement learning (RL). In one formulation, [36] assumes that a state-action value function, $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ was learned ahead of time and stored in a deep neural network (e.g., using the DQN algorithm [37]). Then during execution, at each time step the agent receives a perturbed observation, $\mathbf{s}_{\text{adv}}$, which is assumed to lie within an ϵ-ball of the true state, $\mathbf{s}^\star$, $\|\mathbf{s}_{\text{adv}} - \mathbf{s}^\star\| \le \epsilon$. To select an action robustly, rather than choosing the action that maximizes the value function for the observed state (i.e., $a = \arg\max_{a\in\mathcal{A}} Q(\mathbf{s}_{\text{adv}}, a)$), the robust agent chooses the action with the highest worst-case value,

$$a = \arg\max_{a\in\mathcal{A}} \min_{\mathbf{s}\in\mathcal{B}(\mathbf{s}_{\text{adv}},\epsilon)} Q(\mathbf{s}, a),$$

where $\mathcal{B}$ represents the ϵ-ball. The inner minimization is an instance of the robustness analysis problems from above, as it requires minimizing the output of a NN given an input set. [36] computes a lower bound using Fast-Lin [38] but a lower bound could also be computed with SDP or other methods. Experiments in [36] suggest that bringing formal robustness analysis into reinforcement learning can recover much of the performance lost due to sensor noise or adversarial attacks.

F. Stability Analysis of Neural Network-Controlled Systems

In this subsection as well as the next one, we consider neural networks in feedback loops. Specifically, we consider

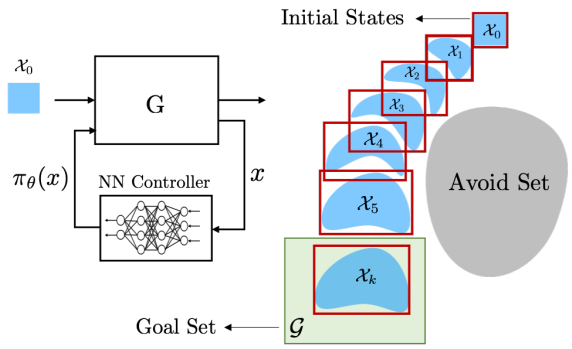


Fig. 3: An illustration of closed-loop reachability with the initial set $\mathcal{X}_0$, the t-step forward reachable set $\mathcal{X}_t$, and its over-approximation $\bar{\mathcal{X}}_t$.

the feedback interconnection of a linear time-invariant (LTI) system and a neural network in discrete time.

$$x_{t+1} = f_{cl}(x_t) = Ax_t + Bf(x_t). \tag{4}$$

Within this setup, the neural network can represent a controller that, for example, results from a reinforcement learning algorithm or is trained to approximate a computationally more expensive model predictive controller (MPC). In this case, we would like to verify various properties such as constraint satisfaction, stability, or safety. This problem has been addressed recently by several papers [3], [7], [39]. A useful property to verify is local stability of the equilibrium and to compute a positive invariant set. A positive invariant set is an ultimate safety certification for dynamical systems since it contains initial states whose trajectories will stay in the set *forever* and never violate the system constraints.

Definition 2 (Positive Invariant Set) Consider the dynamical system $x_{k+1} = f_{cl}(x_k)$. A set $\mathcal{O} \subseteq \mathbb{R}^{n_x}$ of states is a positively invariant set with respect to f_{cl}, if $x_0 \in \mathcal{X}$ implies $x_k \in \mathcal{O}$ for all $k \geq 0$.

Verifying whether a candidate set $\mathcal{O}$ is positively invariant amounts to verifying that $f_{cl}(\mathcal{O}) \subseteq \mathcal{O}$.. In §V-B we will describe a method based on Lyapunov functions and semidefinite programming to compute a positively invariant set for (4).

G. Safety Verification of Neural Network-Controlled Systems

Safety verification or reachability analysis aims to show that starting from some initial conditions, a dynamical system cannot evolve to some unsafe region in the state space. For the closed-loop system (4), we denote by $\mathcal{X}_t$ the *forward reachable set* (FRS) at time $t \geq 0$ from a given set of initial conditions $\mathcal{X}_0 \subset \mathbb{R}^{n_x}$, which is defined by the following recursion

$$\mathcal{X}_{t+1} := f_{cl}(\mathcal{X}_t),$$

We are interested in verifying the finite-time reach-avoid properties of the closed-loop system (4). More specifically, given a goal set $\mathcal{G} \subseteq \mathbb{R}^{n_x}$ and a sequence of avoid sets $\mathcal{A}_t \subseteq \mathbb{R}^{n_x}$, we would like to test if all initial states $x_0 \in \mathcal{X}_0$ of the closed-loop system (4) can reach $\mathcal{G}$ in a finite time horizon $N \geq 0$, while avoiding $\mathcal{A}_t$ for all $t = 0, \cdots, N$. This is equivalent to verify if

$$\mathcal{X}_N \subseteq \mathcal{G} \tag{5a}$$
$$\mathcal{X}_t \cap \mathcal{A}_t = \emptyset \ \forall t = 0, \cdots, N \tag{5b}$$

hold true for (4). See Fig. 3 for an illustration. There exist efficient methods [40] and software implementations [41] for testing the set inclusion (5a) and the set intersection (5b). However, computing exact reachable sets for the nonlinear closed-loop system (4) is, in general, computationally intractable. Therefore, we resort to finding outer-approximations of the closed-loop reachable sets, $\bar{\mathcal{X}}_t \supseteq \mathcal{X}_t$. To obtain useful certificates, our goal is to compute the tightest outer-approximations of the t-step reachable sets. In §V-C we will show how we can achieve this task using semidefinite programming.

III. An Introduction to Quadratic Constraints

Quadratic constraints have a rich history in the robust control literature and have been used as a tool to asbtract nonlinearities, time variations, and uncertain parameters by the constraints they impose on their inputs and outputs. More recently, quadratic constraints have been used and adapted to analyze neural networks [3]–[6], [39], [42]. In this section, we provide an overview of quadratic constraints and their geometric interpretation in describing sets and functions. In §IV we will make use of these QCs to abstract the map of neural networks.

A. Set Description via Quadratic Constraints

We begin by describing sets using quadratic constraints.

Definition 3 *Let $\mathcal{X} \subset \mathbb{R}^{n_x}$ be a nonempty set. Suppose $\mathcal{P}_\mathcal{X}$ is a set of symmetric indefinite matrices P such that*

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \quad \textit{for all } x \in \mathcal{X}. \tag{6}$$

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

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

In other words, the set $\mathcal{X}$ is over approximated by the intersection of a possibly infinite number of sets defined by quadratic inequalities. Finally, we note that (6) is trivial for positive definite P and hence, we omit it from consideration.

A useful concept for describing sets via QCs is a sector. Formally, we define a sector as a set defined by exclusive-disjunction of two affine inequalities,

$$S := \{x \in \mathbb{R}^{n_x} \mid (a_i^\top x - b_i)(a_j^\top x - b_j) \leq 0\}. \tag{7}$$

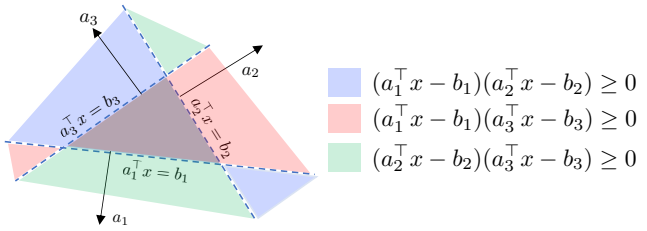


Fig. 4: Describing a polytope by intersecting three sectors. The sectors are formed by the hyper-planes that define the polytope.

Using this definition, we can describe several useful geometries such as polytopes or even non-convex geometries by intersecting different sectors. We have illustrated this idea in Fig. 4 for a two-dimensional polytope.

B. Function Description via Quadratic Constraints

To describe functions with QCs, we simply describe their set representation, i.e., their graph. Specifically, consider a nonlinear function $\phi\colon \mathbb{R}^n \to \mathbb{R}^m$ mapping $\mathcal{X} \subset \mathbb{R}^n$ to $\mathcal{Y} \subset \mathbb{R}^m$. The graph of ϕ is a subset of $\mathcal{X} \times \mathcal{Y}$ defined as $\mathcal{G}(\phi) = \{(x, \phi(x)) \mid x \in \mathcal{X}\}$. We then apply Definition 3 to the set $\mathcal{G}(\phi)$, which leads to the following definition.

Definition 4 (QC for functions) *We say $\phi\colon \mathcal{X} \to \mathcal{Y}$ satisfies the QC defined by $\mathcal{Q}_\phi \subset \mathbb{S}^{2n+1}$ if for any $Q \in \mathcal{Q}_\phi$ we have*

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0 \quad \textit{for all } x \in \mathcal{X}, \tag{8}$$

1) Non-Smooth Activation Functions: We consider the ReLU function $\varphi(x) = \max(0, x)$ over the interval $[\underline{x}, \bar{x}]$. We first consider the case $\underline{x} \leq 0 \leq \bar{x}$, i.e., when it is not clear if the function is active or not. In this case, we can precisely describe the graph of φ by intersecting three sectors:

$$\begin{aligned} &y(y - x) = 0 \\ &y(y - \frac{\bar{x}}{\bar{x} - \underline{x}}(x - \underline{x})) \leq 0 \\ &(y - x)(y - \frac{\bar{x}}{\bar{x} - \underline{x}}(x - \underline{x})) \leq 0 \end{aligned} \tag{9}$$

We illustrate these sector bounds geometrically in Fig. 5.

By multiplying the above inequalities by the multipliers $\lambda \in \mathbb{R}$, $\mu \geq 0$, $\nu \geq 0$ and summing them up, we can arrive at a QC of the form

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^\top Q(\lambda, \mu, \nu) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } \underline{x} \leq 0 \leq \bar{x},$$

where Q is linear in its arguments. Note that we could have alternatively described the graph of φ using the following four inequalities

$$y(y - x) = 0 \quad y \geq 0 \quad y \geq x \quad (x - \underline{x})(x - \bar{x}) \leq 0.$$

Nevertheless, the description in (9) is preferable as , without incurring conservatism, it has fewer constraints, all of which are quadratic.

We now consider the case when the ReLU function is active, i.e, when $\underline{x} \geq 0$. For this case we have $\varphi(x) = x$ and therefore, we can describe its graph by intersecting the following three sectors:

$$(y - x)(x - \underline{x}) = 0, \ (y - x)(x - \bar{x}) = 0, \ (x - \underline{x})(x - \bar{x}) \leq 0.$$

Finally, for the case when the ReLU function is inactive, i.e., when $\bar{x} \leq 0$, we can describe the graph of φ by

$$y(x - \underline{x}) = 0, \ y(x - \bar{x}) = 0, \ (x - \underline{x})(x - \bar{x}) \leq 0.$$

2) Smooth Activation Functions: We now turn to smooth activation functions. For simplicity, we only consider $\tanh$ function. Let $\varphi(x) = \tanh(x)$ over the interval $[\underline{x}, \bar{x}]$. When $\underline{x} \leq 0 \leq \bar{x}$, then the graph of $\tanh$ is contained in the sector described by

$$S := \{x \in \mathbb{R}^n \mid (y - \alpha x)(y - \beta x) \leq 0\}.$$

where $\alpha = \min(\tanh(\underline{x})/\underline{x}, \tanh(\bar{x})/\bar{x})$ and $\beta = 1$; see Fig. 6.a. When $\underline{x} \cdot \bar{x} > 0$, then we can over approximate the graph of φ by a polytope as illustrated in Fig. 6.b.

C. Incremental Quadratic Constraints

Incremental Quadratic Constraints, or δQC for short, aim to describe nonlinear functions/operators incrementally with respect to two inputs. Typically, one of these points is fixed, e.g., the equilibrium of the controlled system (4). Nevertheless, we consider the more general case where these two points may belong to different subsets of $\mathbb{R}^n$. We formalize this in the next definition [6].

Definition 5 (Local Incremental Quadratic Constraint) Let $\mathcal{X}, \tilde{\mathcal{X}} \subseteq \mathbb{R}^n$ be two closed sets. We say the function $\phi\colon \mathbb{R}^n \to \mathbb{R}^m$ satisfies the local incremental quadratic constraint defined by $\mathcal{Q}$ on $\mathcal{X} \times \tilde{\mathcal{X}}$ if for any $Q \in \mathcal{Q} \subset \mathbb{S}^{m+n}$ we have

$$\begin{bmatrix} x - \tilde{x} \\ \phi(x) - \phi(\tilde{x}) \end{bmatrix}^\top Q \begin{bmatrix} x - \tilde{x} \\ \phi(x) - \phi(\tilde{x}) \end{bmatrix} \geq 0 \quad \forall (x, \tilde{x}) \in \mathcal{X} \times \tilde{\mathcal{X}}. \tag{10}$$

In the sequel, we elaborate on characterizing nonlinear functions via δQCs. For simplicity in exposition, we consider the simpler case of $\mathcal{X} = \tilde{\mathcal{X}}$. Extensions to the more general case would be similar.

1) Smooth Activation Functions: Let $\varphi\colon \mathbb{R} \to \mathbb{R}$ be continuous on $[\underline{x}, \ \bar{x}]$ and differentiable on $(\underline{x}, \ \bar{x})$. Using the mean-value theorem on the interval $[x, \tilde{x}] \subseteq [\underline{x}, \bar{x}]$, we have

$$\varphi'(c) = \frac{\varphi(x) - \varphi(\tilde{x})}{x - \tilde{x}} \text{ for some } c \in (x, \tilde{x}).$$

By defining $\alpha = \inf_{c \in (\underline{x}, \bar{x})} \ \varphi'(c)$ and $\beta = \sup_{c \in (\underline{x}, \bar{x})} \ \varphi'(c)$, we see that

$$\alpha \leq \frac{\varphi(x) - \varphi(\tilde{x})}{x - \tilde{x}} \leq \beta \quad \text{for all } x, \tilde{x} \in [\underline{x}, \bar{x}].$$

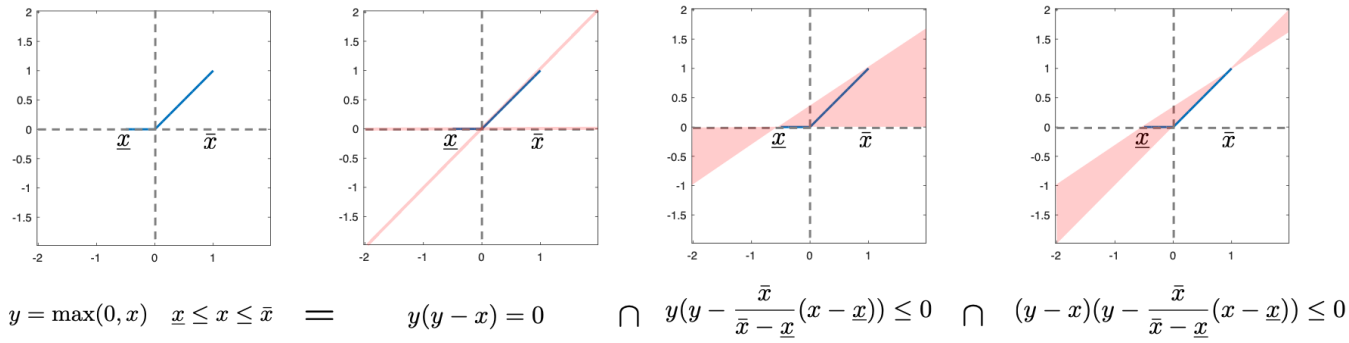


Fig. 5: Describing the ReLU function over an interval $[\underline{x}, \bar{x}]$ $(\underline{x} \le 0 \le \bar{x})$ by intersecting three sectors.

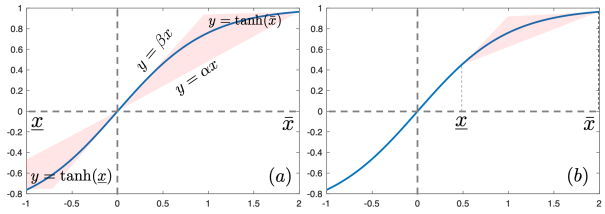


Fig. 6: Describing the tanh function over an interval $[\underline{x}, \bar{x}]$; (a) is for the case $(\underline{x} \le 0 \le \bar{x})$ and (b) is for the case $\bar{x} \ge \underline{x} \ge 0$.

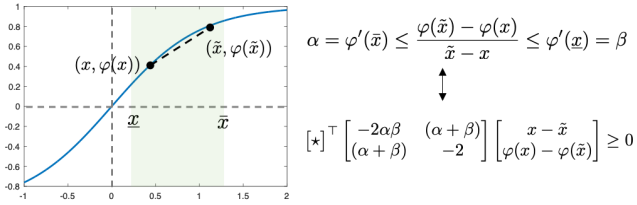


Fig. 7: Local incremental quadratic constraint for $\varphi(x) = \tanh(x)$ on $[\underline{x}, \bar{x}]$.

These two inequalities can be written equivalently as

$$\begin{bmatrix} x - \tilde{x} \\ \varphi(x) - \varphi(\tilde{x}) \end{bmatrix}^\top \begin{bmatrix} -2\alpha\beta & (\alpha+\beta) \\ (\alpha+\beta) & -2 \end{bmatrix} \begin{bmatrix} x - \tilde{x} \\ \varphi(x) - \varphi(\tilde{x}) \end{bmatrix} \ge 0.$$

See Fig. 7 for an illustration.

2) Non-smooth Activation Functions: Since the ReLU function is not differentiable, the mean value theorem is not directly applicable but we can still use the slope restriction condition to derive incremental quadratic constraints. Specifically, for the ReLU function we can write slope restriction condition

$$\alpha \le \frac{\varphi(x) - \varphi(\tilde{x})}{x - \tilde{x}} \le \beta.$$

where $\alpha = \beta = 1$ when the ReLU is active, $\alpha = \beta = 0$ when the ReLU is inactive, and $\alpha = 0, \beta = 1$ when the ReLU is uncertain. See Fig. 8 for an illustration.

D. S-Procedure

The original application of the S-lemma is to decide whether a quadratic (in)equality is satisfied over a domain. When this domain is defined by quadratic inequalities, then the question is when a quadratic inequality is a consequence of other quadratic inequalities. This idea can be formalized as follows. Let $g_i \colon \mathbb{R}^n \to \mathbb{R}$, $i = 0, \cdots, m$ be quadratic of the form $g_i(x) = \begin{bmatrix} x^\top & 1 \end{bmatrix} P_i \begin{bmatrix} x^\top & 1 \end{bmatrix}^\top$. Then, the implication

$$g_i(x) \ge 0 \quad i = 1, \cdots, m \implies g_0(x) \ge 0.$$

holds if there exists non-negative $\tau_1, \tau_2, \cdots, \tau_m$ (called *multipliers*) such that the following matrix inequality holds.

$$\sum_{k=1}^{m} \tau_k P_k \preceq P_0 \tag{11}$$

To see this, we just need to right- and left-multiply both sides of (11) by the "basis vector" $\begin{bmatrix} x^\top & 1 \end{bmatrix}^\top$ and its transpose, respectively. Notably, when $m = 1$, this condition is both necessary and sufficient [43]. Furthermore, (11) is linear in the multipliers, resulting in a Linear Matrix Inequality (LMI).

IV. Neural Network Abstraction with Quadratic Constraints

Having characterized sets and functions with various forms of quadratic constraints, in this section, our goal is to abstract nonlinearities within a neural network with these quadratic constraints and then invoke the S-procedure to translate them into end-to-end quadratic constraints for the entire network. We will then show that how the analysis of these abstracted networks lends itself to semidefinite programming.

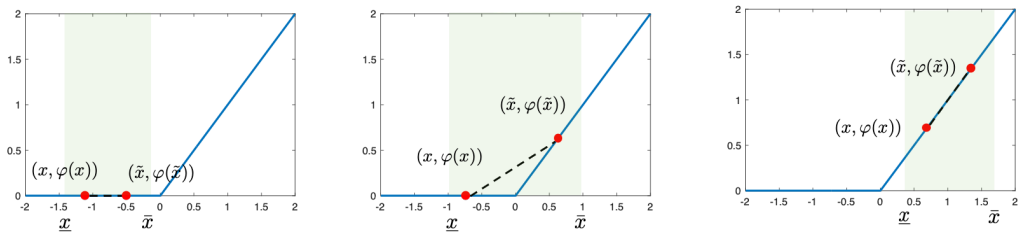
A. Neural Network Model

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

$$\begin{aligned} x^{k+1} &= \phi(W^k x^k + b^k) \quad k = 0, \cdots, \ell - 1 \\ f(x^0) &= W^\ell x^\ell + b^\ell. \end{aligned} \tag{12}$$

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. We denote by $n = \sum_{k=1}^{\ell} n_k$ the total number of neurons. The activation layer ϕ is of the form

$$\phi(x) := [\varphi(x_1) \cdots \varphi(x_d)]^\top, \ x \in \mathbb{R}^d, \ k = 1, \cdots, \ell - 1,$$



$$0 \le \frac{\varphi(x)-\varphi(\tilde{x})}{x-\tilde{x}} \le 0 \qquad 0 \le \frac{\varphi(x)-\varphi(\tilde{x})}{x-\tilde{x}} \le 1 \qquad 1 \le \frac{\varphi(x)-\varphi(\tilde{x})}{x-\tilde{x}} \le 1$$

$$[\star]^\top \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x-\tilde{x} \\ \varphi(x)-\varphi(\tilde{x}) \end{bmatrix} = 0 \qquad [\star]^\top \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x-\tilde{x} \\ \varphi(x)-\varphi(\tilde{x}) \end{bmatrix} \ge 0 \qquad [\star]^\top \begin{bmatrix} -2 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} x-\tilde{x} \\ \varphi(x)-\varphi(\tilde{x}) \end{bmatrix} = 0$$

Fig. 8: Local incremental quadratic constraint for $\varphi(x) = \max(0, x)$ defined on $[\underline{x}, \bar{x}]$, where $\star$ denotes the repeated part of the vector surrounding the middle matrix.

where φ is the activation function of each neuron (ReLU[1], sigmoid, tanh, etc.).

B. Quadratic Constraints for Neural Networks

Consider the neural network model in (12). Suppose $x^0 \in \mathcal{X}^0$, where $\mathcal{X}^0$ is a region of interest on which we wish to analyze the neural network. We denote by $\mathcal{X}^k$ as the reachable sets of x^k, which are characterized by the recursions,

$$\mathcal{X}^{k+1} = \phi^k(W^k\mathcal{X}^k + b^k) \quad k = 0, \cdots, \ell-1.$$

Furthermore, define $\mathbf{x}^\top = [x^{0\top} \cdots x^{\ell\top}]$ and

$$\boldsymbol{\mathcal{X}} = \{\mathbf{x} \mid (12) \text{ holds and } x^0 \in \mathcal{X}^0\}. \tag{13}$$

Suppose the input set $\mathcal{X}^0$ satisfies the QC defined by $\mathcal{P}$, i.e., for each $P \in \mathcal{P}$ we have (see Definition 3)

$$\begin{bmatrix} x^0 \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x^0 \\ 1 \end{bmatrix} \ge 0 \ \forall\ x \in \mathcal{X}^0. \tag{14a}$$

Furthermore, suppose ϕ^k satisfies the local QC defined by $\mathcal{Q}^k$ on the set $W^k\mathcal{X}^k + b^k$, i.e.,

$$\begin{bmatrix} W^k x^k + b^k \\ x^{k+1} \\ 1 \end{bmatrix}^\top Q^k \begin{bmatrix} W^k x^k + b^k \\ x^{k+1} \\ 1 \end{bmatrix} \ge 0 \quad \forall x^k \in \mathcal{X}^k. \tag{14b}$$

These QCs are coupled together layer by layer. We wish to use the chain of QCs in (14a) and (14b) to infer a QC for the end-to-end map f. That is, given a symmetric matrix Q_f, we would like to verify that

$$\begin{bmatrix} x^0 \\ f(x^0) \\ 1 \end{bmatrix}^\top Q^f \begin{bmatrix} x^0 \\ f(x^0) \\ 1 \end{bmatrix} \ge 0, \quad \forall x^0 \in \mathcal{X}^0. \tag{14c}$$

How can conclude (14c) from (14a) and (14b)? Recalling §III-D, we need to invoke the S-procedure, after expressing all the QCs in *the same basis*. To this end, we define the entry selector matrices E^k such that $x^k = E^k\mathbf{x}$. We see that (14a), (14b) and (14c) are equivalent to (15a), (15b) and (15c), respectively.

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} E^0 & 0 \\ 0 & 1 \end{bmatrix} P \begin{bmatrix} E^0 & 0 \\ 0 & 1 \end{bmatrix}}_{:=M_{in}(P)} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \ge 0 \ \forall\ \mathbf{x} \in \boldsymbol{\mathcal{X}}. \tag{15a}$$

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} W^k E^k & b^k \\ E^{k+1} & 0 \\ 0 & 1 \end{bmatrix}^\top Q^k \begin{bmatrix} W^k E^k & b^k \\ E^{k+1} & 0 \\ 0 & 1 \end{bmatrix}}_{:=M^k_{mid}(Q^k)} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \ge 0 \ \forall \mathbf{x} \in \boldsymbol{\mathcal{X}}. \tag{15b}$$

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \underbrace{\begin{bmatrix} E^0 & 0 \\ W^\ell E^\ell & b^\ell \\ 0 & 1 \end{bmatrix}^\top Q^f \begin{bmatrix} E^0 & 0 \\ W^\ell E^\ell & b^\ell \\ 0 & 1 \end{bmatrix}}_{:=M_{out}(Q^f)} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \ge 0, \ \forall \mathbf{x} \in \boldsymbol{\mathcal{X}}. \tag{15c}$$

Then, if the following matrix inequality,

$$M_{in}(P) + \sum_{k=0}^{\ell-1} M_{mid}(Q^k) \preceq M_{out}(Q^f) \tag{16}$$

holds for some $Q^k \in \mathcal{Q}^k$ $k = 0, \cdots, \ell-1$, then the neural network satisfies the QC in (14c). We note that the multipliers in the S-procedure (see (11)) are lumped into the matrices $\mathcal{Q}^k$ that characterize the QCs for activation layers.

We highlight that the condition in (16) is equivalent to an LMI feasibility problem since the sets $\mathcal{Q}^k$ are convex and the constraint in (16) is linear in all Q^k. We can use this LMI as a constraint of a convex optimization problem to further optimize some function of the decision variables.

[1]Rectified Linear Unit.

Specifically, we can define the following SDP,

$$\begin{aligned} \text{minimize} \quad & g(P, Q^0, \cdots, Q^{\ell-1}, Q^f) \\ \text{subject to} \quad & M_{in}(P) + \sum_{k=0}^{\ell-1} M_{mid}(Q^k) \preceq M_{out}(Q^f) \\ & P \in \mathcal{P}, Q^k \in \mathcal{Q}^k, Q^f \in \mathcal{Q}^f \end{aligned} \tag{17}$$

where $g(P, Q^0, \cdots, Q^{\ell-1}, Q^f)$ is convex and $\mathcal{Q}^f$ is a convex subset of $\mathbb{S}^{n_x+n_f+1}$.

Remark 1 (Pre-activation bounds) To obtain local QCs for activation functions in (14b), we need to compute bounds on the pre-activation values, i.e., for each layer k, we must find ℓ^k, u^k such that $\ell^k \leq W^k x^k + b^k \leq u^k$ for all k. There are a variety of methods for computing the pre-activation bounds, such as interval arithmetic and the dual-based relaxation technique of [44].

C. Incremental Quadratic Constraints for Neural Networks

Consider the neural network model in (12). Suppose $x^0 \in \mathcal{X}^0$ and $\tilde{x}^0 \in \tilde{\mathcal{X}}^0$. For a given symmetric matrix $Q^f \in \mathbb{R}^{n_x+n_f}$ our goal is to verify that the neural network satisfies the incremental quadratic constraint

$$\begin{bmatrix} x^0 - \tilde{x}^0 \\ f(x^0) - f(\tilde{x}^0) \end{bmatrix}^\top Q^f \begin{bmatrix} x^0 - \tilde{x}^0 \\ f(x^0) - f(\tilde{x}^0) \end{bmatrix} \geq 0 \tag{18}$$

for all $(x^0, \tilde{x}^0) \in \mathcal{X}^0 \times \tilde{\mathcal{X}}^0$. We denote by $\mathcal{X}^k$ and $\tilde{\mathcal{X}}^k$ as the reachable sets of x^k from $\mathcal{X}^0$ and $\tilde{\mathcal{X}}^0$, respectively, which are characterized by the recursions,

$$\begin{aligned} \mathcal{X}^{k+1} &= \phi^k(W^k \mathcal{X}^k + b^k) \quad k = 0, \cdots, \ell-1. \\ \tilde{\mathcal{X}}^{k+1} &= \phi^k(W^k \tilde{\mathcal{X}}^k + b^k) \quad k = 0, \cdots, \ell-1. \end{aligned}$$

Furthermore, define $\boldsymbol{\mathcal{X}}$ and $\tilde{\boldsymbol{\mathcal{X}}}$ akin to (13). Suppose ϕ^k satisfies the local incremental QC defined by $\mathcal{Q}^k$, i.e.,

$$\begin{bmatrix} \star \end{bmatrix}^\top Q^k \begin{bmatrix} W^k(x^k - \tilde{x}^k) \\ x^{k+1} - \tilde{x}^{k+1} \end{bmatrix} \geq 0 \quad \forall (x^k, \tilde{x}^k) \in \mathcal{X}^k \times \tilde{\mathcal{X}}^k. \tag{19}$$

Similar to §IV-B, we can use the S-procedure after a change of basis to find an LMI that allows us to conclude the end-to-end quadratic constraint (18) from layer-wise incremental quadratic constraints (19). To see this, we first express (18) and (19) in the same basis,

$$(\mathbf{x} - \tilde{\mathbf{x}})^\top \underbrace{\begin{bmatrix} \star \end{bmatrix}^\top Q^f \begin{bmatrix} E^0 \\ W^\ell E^\ell \end{bmatrix}}_{= M_{out}(Q^f)} (\mathbf{x} - \tilde{\mathbf{x}}) \geq 0 \ \forall (\mathbf{x}, \tilde{\mathbf{x}}) \in \boldsymbol{\mathcal{X}} \times \tilde{\boldsymbol{\mathcal{X}}} \tag{20}$$

$$(\mathbf{x} - \tilde{\mathbf{x}})^\top \underbrace{\begin{bmatrix} \star \end{bmatrix}^\top Q^k \begin{bmatrix} W^k E^k \\ E^{k+1} \end{bmatrix}}_{M_{mid}(Q^k)} (\mathbf{x} - \tilde{\mathbf{x}}) \geq 0 \ \forall (\mathbf{x}, \tilde{\mathbf{x}}) \in \boldsymbol{\mathcal{X}} \times \tilde{\boldsymbol{\mathcal{X}}} \tag{21}$$

Using the S-procedure, the neural network satisfies the incremental QC (18) if the matrix inequality

$$\sum_{k=0}^{\ell-1} M_{mid}(Q^k) \preceq M_{out}(Q^f). \tag{22}$$

holds for some $Q^k \in \mathcal{Q}^k$ $k = 0, \cdots, \ell-1$. We note that (22) is linear in Q^k and Q^f.

V. Analysis of Neural Networks via Semidefinite Programming

In this section, we revisit the problems discussed in §II. Specifically, we show how we can use the LMI condition (16) and (22) to verify a certain property about the neural network.

A. Neural Network Verification

We begin with considering the local robustness certification task in §II-C. We note that the verification of the property in (2), amounts to verifying that the neural network satisfies the quadratic constraints

$$\begin{bmatrix} \star \end{bmatrix}^\top \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & (e_{i^\star} - e_i) \\ 0 & (e_{i^\star} - e_i)^\top & 0 \end{bmatrix}}_{Q^f} \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix} \geq 0 \ \forall x \in \mathcal{X}.$$

for all classes $i \neq i^\star$. Therefore, for a classifier with n_f number of classes, we need to solve $n_f - 1$ LMI feasibility problems of the form (16) to verify local robustness around a specific point.

Rather than solving $n_f - 1$ distinct LMI feasibility problems for a specific input data point, an alternative approach is to verify that the output of the neural network lies inside the largest ellipsoid inscribed in the safe polytope $\mathcal{S}_y$ and centered at $f(x^\star)$. We can find such an ellipsoid using convex optimization [43]. See Fig. 9 for an illustration. Let us denote such an ellipsoid by $\mathcal{E}(f(x^\star), P)$. Then verifying that the output of the neural network is enclosed by the ellipsoid for all $x \in \mathcal{X}$ amounts to verifying that (f(x)-f(x*))$^\top P(f(x) - f(x^\star)) \leq 1$ for all $x \in \mathcal{X}$,

which is equivalent to the quadratic constraint

$$\begin{bmatrix} \star \end{bmatrix}^\top \begin{bmatrix} 0 & 0 & 0 \\ 0 & -P & Pf(x^\star) \\ 0 & f(x^\star)^\top P & 1 - f(x^\star)^\top P f(x^\star) \end{bmatrix} \begin{bmatrix} x \\ f(x) \\ 1 \end{bmatrix} \geq 0,$$

B. Stability Analysis of Neural Network-Controlled Systems

In this subsection, we analyze the stability of (4). Suppose $x_\star$ is the equilibrium point of (4), which is typically the origin. To analyze local stability of $x_\star$ and find a positive invariant set, we consider the following quadratic Lyapunov function candidate,

$$V(x) = (x - x_\star)^\top P (x - x_\star),$$

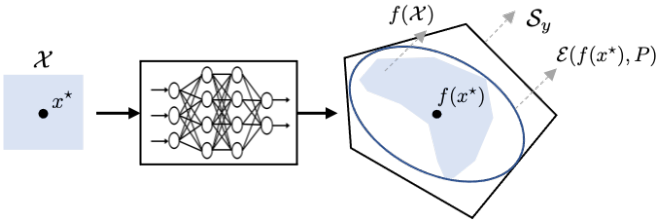


Fig. 9: Verifying $f(\mathcal{X}) \subset \mathcal{S}_y$ by verifying whether $f(\mathcal{X}) \subset \mathcal{E}(f(x^\star), P)$.

where $P \in \mathbb{S}^{n_x}_{++}$ is to be determined, local geometric stability of the closed-loop system on an open set $\mathcal{D}$ (which contains $x_\star$) is implied by the condition

$$V(Ax + Bf(x)) \leq \rho V(x) \quad \forall x \in \mathcal{D}, \tag{23}$$

where $\rho \in (0, 1)$ is the convergence rate. This condition can be equivalently expressed as

$$\begin{bmatrix}\star\end{bmatrix}^\top \begin{bmatrix} A^\top P A - \rho P & PB \\ B^\top P & B^\top P B \end{bmatrix} \begin{bmatrix} x - x_\star \\ f(x) - f(x_\star) \end{bmatrix} \leq 0 \ \forall x \in \mathcal{D}. \tag{24}$$

If there exists a $P \in \mathbb{S}_{++}$ that satisfies (24), then $x_\star$ is locally geometrically stable.

We see that (24) is an incremental quadratic constraint for f. To verify (24) for a fixed $P \succ 0$, we can use the LMI in (22) with Q^f defined as the middle matrix in (24). Then if the LMI is feasible for some $Q^k \in \mathcal{Q}^k$ $k = 0, \cdots, \ell - 1$ and $P \succ 0$, then the closed-loop system is geometrically stable with rate ρ.

When $\rho = 1$, then the stability condition in (23) will imply that the largest sublevel set of V contained in $\mathcal{D}$ is a positive invariant set, i.e., the set

$$\mathcal{O} = \{x \in \mathbb{R}^{n_x} \mid V(x) \leq c\},$$

where c is the largest positive constant such that $\mathcal{O} \subseteq \mathcal{D}$. To see this, suppose $x_k \in \mathcal{O}$ for some k. Then by (23), we must have $V(x_{k+1}) \leq c$, or equivalently, $x_{k+1} \in \mathcal{O}$.

C. Reachability Analysis of Neural Network-Controlled Systems

We now revisit the finite-time reach-avoid problem discussed in §II-G. Specifically, our goal is to recursively overapproximate the reachable sets from a set of initial conditions. To illustrate the procedure, consider that $\bar{\mathcal{X}}_t \supseteq \mathcal{X}_t$ has already been computed by a template polytope of the form

$$\bar{\mathcal{X}}_t = \{x \in \mathbb{R}^{n_x} \mid Hx \leq h_t\},$$

where $H \in \mathbb{R}^{m \times n}$. The goal is to compute $\bar{\mathcal{X}}_{t+1}$ using the same template polytope

$$\bar{\mathcal{X}}_{t+1} = \{x \in \mathbb{R}^{n_x} \mid Hx \leq h_{t+1}\},$$

such that $\mathcal{X}_{t+1} \subseteq \bar{\mathcal{X}}_{t+1}$. For this inclusion to hold, we must have that

$$H(Ax_t + Bf(x_t)) \leq h_{t+1} \quad \forall x_t \in \bar{\mathcal{X}}_t.$$

These m inequalities can be equivalently written as m quadratic constraints

$$\begin{bmatrix}\star\end{bmatrix}^\top \begin{bmatrix} 0 & 0 & -A^\top H^\top e_i \\ 0 & 0 & -B^\top H^\top e_i \\ -e_i^\top HA & -e_i^\top HB & 2e_i^\top h_{t+1} \end{bmatrix} \begin{bmatrix} x_t \\ f(x_t) \\ 1 \end{bmatrix} \geq 0,$$

for $i = 1, \cdots, m$, where $e_i \in \mathbb{R}^m$ is the i-th unit vector. For given h_{t+1}, these quadratic constraints can be verified using the LMI in (16). To find the smallest inscribing polytope, we can formulate m SDPs with objective functions $e_i^\top h_{t+1}$, $i = 1, \cdots, m$. After solving these SDPs, we will arrive at the polytope $\bar{\mathcal{X}}_{t+1}$. We can continue this procedure to compute all the forward reachable sets.

D. Estimation of Lipschitz Constant

In this subsection, we show how we can compute a bound on the local Lipschitz constant of a neural network over a region $\mathcal{X}^0 \subseteq \mathbb{R}^{n_x}$ (for the case of set equality, the bound will be on the global Lipschitz constant). To begin, note that Lipschitz continuity of f over $\mathcal{X}^0$ with Lipschitz constant $\sqrt{\rho}$ is equivalent to the following incremental quadratic constraint on f,

$$\begin{bmatrix} x^0 - \tilde{x}^0 \\ f(x^0) - f(\tilde{x}^0) \end{bmatrix}^\top \begin{bmatrix} \rho I_{n_x} & 0 \\ 0 & -I_{n_f} \end{bmatrix} \begin{bmatrix} x^0 - \tilde{x}^0 \\ f(x^0) - f(\tilde{x}^0) \end{bmatrix} \geq 0, \tag{25}$$

for all $x^0, \tilde{x}^0 \in \mathcal{X}^0$. To verify (25) for a fixed $\rho > 0$, we can invoke the LMI in (22) with $Q_f = \mathrm{blkdiag}(\rho I_{n_x}, -I_{n_f})$. If the LMI is feasible, then $\sqrt{\rho}$ is a guaranteed upper bound on the Lipschitz constant. Since this LMI is linear in ρ, the best upper bound on the (local) Lipschitz constant can be obtained by minimizing ρ subject to the constraint defined by the LMI. See [5], [6] for a detailed account of this.

VI. Conclusion

The use of quadratic constraints has a rich history in robust control and has been leveraged as a tool to abstract nonlinearities, time variations, unmodeled dynamics, and uncertain parameters by the constraints they impose on their inputs and outputs. In particulaar, abstracting neural networks with the quadratic 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 tutorial paper, we provide a framework based on quadratic constraints to analyze neural networks. The main idea is to abstract the nonlinearities of a neural network (e.g., activation functions) by quadratic constraints. We then showed that we can analyze the abstracted network via semidefinite programming.

In principle, there is a trade-off between conservatism, run time, and memory requirements for solving these convex programs. Developing numerical algorithms that can span this trade-off is a future research direction. For instance, due to the sequential structure of neural networks, there is an inherent chordal sparsity pattern in these semidefinite

programs that can be exploited to improve scalability. Moreover, we have only considered fully-connected networks so far. 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.

REFERENCES

[1] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *Conference on Robot Learning*, pp. 1328–1337, PMLR, 2020.

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

[3] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *IEEE Transactions on Automatic Control*, 2020.

[4] M. Fazlyab, M. Morari, and G. J. Pappas, "Probabilistic verification and reachability analysis of neural networks via semidefinite programming," *arXiv preprint arXiv:1910.04249*, 2019.

[5] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 11423–11434, 2019.

[6] N. Hashemi, J. Ruths, and M. Fazlyab, "Certifying incremental quadratic constraints for neural networks via convex optimization," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control* (A. Jadbabaie, J. Lygeros, G. J. Pappas, P. A.nbsp;Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger, eds.), vol. 144 of *Proceedings of Machine Learning Research*, pp. 842–853, PMLR, 07 – 08 June 2021.

[7] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, "Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming," *arXiv preprint arXiv:2004.07876*, 2020.

[8] H. Qian and M. N. Wegman, "L2-nonexpansive neural networks," *arXiv preprint arXiv:1802.07896*, 2018.

[9] 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.

[10] 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.

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

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

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

[14] 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.

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

[16] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, vol. 1, no. 2, p. 3, 2017.

[17] K. Dvijotham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks," *arXiv preprint arXiv:1803.06567*, 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] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[20] 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.

[21] M. Jin and J. Lavaei, "Stability-certified reinforcement learning: A control-theoretic perspective," *arXiv preprint arXiv:1810.11505*, 2018.

[22] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

[23] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," in *Advances in Neural Information Processing Systems*, pp. 3835–3844, 2018.

[24] R. Balan, M. Singh, and D. Zou, "Lipschitz properties for deep convolutional networks," *arXiv preprint arXiv:1701.05217*, 2017.

[25] D. Zou, R. Balan, and M. Singh, "On lipschitz bounds of general convolutional neural networks," *arXiv preprint arXiv:1808.01415*, 2018.

[26] P. L. Combettes and J.-C. Pesquet, "Lipschitz certificates for neural network structures driven by averaged activation operators," *arXiv preprint arXiv:1903.01014*, 2019.

[27] A. Araujo, B. Negrevergne, Y. Chevaleyre, and J. Atif, "On lipschitz regularization of convolutional layers using toeplitz matrix theory," 2021.

[28] 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.

[29] T. Avant and K. A. Morgansen, "Analytical bounds on the local lipschitz constants of affine-relu functions," *arXiv preprint arXiv:2008.06141*, 2020.

[30] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.

[31] F. Latorre, P. Rolland, and V. Cevher, "Lipschitz constant estimation of neural networks via sparse polynomial optimization," in *International Conference on Learning Representations*, 2020.

[32] M. Jordan, J. Lewis, and A. G. Dimakis, "Provable certificates for adversarial examples: Fitting a ball in the union of polytopes," *arXiv preprint arXiv:1903.08778*, 2019.

[33] T. Chen, J. B. Lasserre, V. Magron, and E. Pauwels, "Semialgebraic optimization for lipschitz constants of relu networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: from adversarial to random noise," *Advances in Neural Information Processing Systems*, vol. 29, pp. 1632–1640, 2016.

[35] J. Peck, J. Roels, B. Goossens, and Y. Saeys, "Lower bounds on the robustness to adversarial perturbations," in *Advances in Neural Information Processing Systems*, pp. 804–813, 2017.

[36] M. Everett, B. Lutjens, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," *arXiv preprint arXiv:2004.06496*, 2020.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," in *Nature*, vol. 518, Nature Publishing Group, a division of Macmillan Publishers Limited., 2015.

[38] T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning (ICML)*, 2018.

[39] H. Yin, P. Seiler, and M. Arcak, "Stability analysis using quadratic constraints for systems with neural network controllers," *arXiv preprint arXiv:2006.07579*, 2020.

[40] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Springer, 2008.

[41] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, (Zürich, Switzerland), pp. 502–510, July 17–19 2013.

[42] P. Pauli, D. Gramlich, J. Berberich, and F. Allgöwer, "Linear systems with neural network nonlinearities: Improved stability analysis via acausal zames-falb multipliers," *arXiv preprint arXiv:2103.17106*, 2021.

[43] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[44] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, pp. 5286–5295, PMLR, 2018.