2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Reach-SDP: Reachability Analysis of Closed-Loop Systems with Neural Network Controllers via Semidefinite Programming

Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George J. Pappas

Abstract— **There has been an increasing interest in using neural networks in closed-loop control systems to improve performance and reduce computational costs for on-line implementation. However, providing safety and stability guarantees for these systems is challenging due to the nonlinear and compositional structure of neural networks. In this paper, we propose a novel forward reachability analysis method for the safety verification of linear time-varying systems with neural networks in feedback interconnection. Our technical approach relies on abstracting the nonlinear activation functions by quadratic constraints, which leads to an outer-approximation of forward reachable sets of the closed-loop system. We show that we can compute these approximate reachable sets using semidefinite programming. We illustrate our method in a quadrotor example, in which we first approximate a nonlinear model predictive controller via a deep neural network and then apply our analysis tool to certify finite-time reachability and constraint satisfaction of the closed-loop system.**

I. Introduction

Deep neural networks (DNN) have seen renewed interest in recent years due to the proliferation of data and access to more computational power. In autonomous systems, DNNs are either used as feedback controllers [1], [2], motion planners [3], perception modules, or end-to-end controllers [4]. Despite their high performance, DNN-driven autonomous systems lack safety and stability guarantees. Indeed, recent studies show that DNNs can be vulnerable to small perturbations or adversarial attacks [5], [6]. This issue is more pronounced in closed-loop systems, as a small perturbation in the loop can dramatically change the behavior of the closed-loop system over time. Therefore, it is of utmost importance to develop tools for verification of DNN-driven control systems. The goal of this paper is to develop a methodology, based on semidefinite programming, for safety verification and reachability analysis of linear dynamical systems in feedback interconnection with DNNs.

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. Methods for reachability analysis can be categorized into exact (complete) or approximate (incomplete), which compute the reachable sets exactly and approximately, respectively. Verification of dynamical systems has been extensively studied in the past [7]–[9]. More recently, the problem of output range analysis of neural networks has been addressed in [10]–[16], mainly motivated by robustness analysis of DNNs against adversarial attacks [6]. Compared to these bodies of work, verification of closed-loop systems with neural network controllers has been less explored [17]–[20]. In [20] the authors extend the verification tools for neural networks to deep reinforcement learning tasks. In [18], a method for verification of sigmoid-based neural networks in feedback with a hybrid system is proposed, in which the neural network is transformed into a hybrid system and then a standard verification tool for hybrid systems is invoked. In [17], a new reachability analysis approach based on Bernstein polynomials is proposed that can verify neural-network-controlled systems with Lipschitz continuous activation functions. Dutta et al. [19] use a flow pipe construction scheme to over approximate the reachable sets. A piecewise polynomial model is used to provide an approximation of the input-output mapping of the controller and an error bound on the approximation. This approach, however, is only applicable to Rectified Linear Unit (ReLU) activation functions.

Contributions. In this paper, we propose a semidefinite program (SDP) for reachability analysis of linear time-varying dynamical systems in feedback interconnection with neural network controllers equipped with a projection operator, which projects the output of the neural network (the control action) onto a specified set of control inputs. Our technical approach relies on abstracting the nonlinear activation functions as well as the projection operator by quadratic constraints [14], which leads to an outer-approximation of forward reachable sets of the closed-loop system. We show that we can compute these approximate reachable sets using semidefinite programming. Our approach can be used to analyze control policies learned by neural networks in, for example, model predictive control (MPC) [21] and constrained reinforcement learning [22]. To the best of our knowledge, our result is the first convex-optimization-based method for reachability analysis of closed-loop systems with neural networks in the loop. We illustrate the utility of our approach in two numerical examples, in which we certify finite-time reachability and constraint satisfaction of a double integrator and a 6D quadrotor system.

A. Notation and Preliminaries

We denote the set of 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 n-dimensional identity matrix by I_n, and the set of n-by-n symmetric matrices by $\mathbb{S}^n$. For $A \in \mathbb{R}^{m\times n}$, the inequality $A \geq 0$ is component-wise. For $A \in \mathbb{S}^n$, the inequality $A \succeq 0$ means A is positive semidefinite.

Work supported by the NSF under grants DARPA Assured Autonomy and NSF CPS 1837210. The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania. Email: {haiminhu,mahyarfa,morari,pappasg}@seas.upenn.edu.

II. Problem Formulation

A. Neural Network Control System

We consider a discrete-time linear time-varying system

$$P: \; x_{t+1} = A_t x_t + B_t u_t + c_t, \tag{1}$$

where $x_t \in \mathbb{R}^{n_x}$, $u_t \in \mathbb{R}^{n_u}$ are the state and control vectors, and $c_t \in \mathbb{R}^{n_x}$ is an exogenous input. We assume that the system (1) is subject to input constraints,

$$u_t \in \mathcal{U}_t, \; t = 0, 1, \cdots, \tag{2}$$

which represent, for example, actuator limits that are naturally satisfied or hard constraints that must be satisfied by a control design specification. A specialization that we consider in this paper is input box constraint $\underline{u}_t \leq u_t \leq \bar{u}_t$. Furthermore, we assume a state-feedback controller $\pi(x_t) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ parameterized by a multi-layer feed-forward fully-connected neural network. The map $x \mapsto \pi(x)$ is described by the following equations,

$$\begin{aligned} x^0 &= x \\ x^{k+1} &= \phi(W^k x^k + b^k) \quad k = 0, \cdots, \ell - 1 \\ \pi(x) &= W^\ell x^\ell + b^\ell, \end{aligned} \tag{3}$$

where $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$, $b^k \in \mathbb{R}^{n_{k+1}}$ are the weight matrix and bias vector of the $(k+1)$-th layer. The nonlinear activation function $\phi(\cdot)$ is applied component-wise to the pre-activation vectors, i.e., $\phi(x) \coloneqq [\varphi(x_1) \cdots \varphi(x_{n_k})]^\top \; x \in \mathbb{R}^{n_k}$, where $\varphi : \mathbb{R} \to \mathbb{R}$ is the activation function of each individual neuron. Common choices include ReLU, sigmoid, tanh, leaky ReLU, etc. In this paper, we consider ReLU activation functions in our technical derivations but we can address other activation functions following the framework of [14]. To ensure that output of neural network respects the input constraint, we consider a projection operator in the loop and define the control input as

$$u_t = \mathrm{Proj}_{\mathcal{U}_t}(\pi(x_t)) = \min(\max(\pi(x_t), \underline{u}_t), \bar{u}_t). \tag{4}$$

We denote the closed-loop system with dynamics (1) and the projected neural network control policy (4) by

$$x_{t+1} = f_\pi(x_t) \coloneqq A_t x_t + B_t \, \mathrm{Proj}_{\mathcal{U}_t}(\pi(x_t)) + c_t, \tag{5}$$

which is a non-smooth nonlinear system because of nonlinear activation functions in the neural network as well the projection operator. For the closed-loop system (5), we denote by $\mathcal{R}_t(\mathcal{X}_0)$ the forward reachable set at time t from a given set of initial conditions $\mathcal{X}_0 \subseteq \mathbb{R}^{n_x}$, which is defined by the following recursion

$$\mathcal{R}_{t+1}(\mathcal{X}_0) \coloneqq f_\pi(\mathcal{R}_t(\mathcal{X}_0)), \; \mathcal{R}_0(\mathcal{X}_0) = \mathcal{X}_0. \tag{6}$$

See Figure 1 for an illustration.

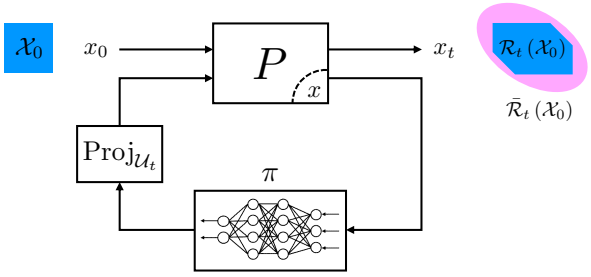

Fig. 1: An illustration of closed-loop reachability with the initial set $\mathcal{X}_0$, the t-step forward reachable set $\mathcal{R}_t(\mathcal{X}_0)$, and its over-approximation $\bar{\mathcal{R}}_t(\mathcal{X}_0)$ shown in magenta.

B. Finite-Time Reach-Avoid Verification Problem

In this paper, we are interested in verifying the finite-time reach-avoid properties of the closed-loop system (5). 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 (5) 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 test if

$$\mathcal{R}_N(\mathcal{X}_0) \subseteq \mathcal{G} \tag{7a}$$

$$\mathcal{R}_t(\mathcal{X}_0) \cap \mathcal{A}_t = \emptyset \; \forall t = 0, \cdots, N, \tag{7b}$$

holds true for (5). There exist efficient methods [23] and software implementations [24] for testing set inclusion (7a) and set intersection (7b). However, computing exact reachable sets for the nonlinear closed-loop system (5) is, in general, computationally intractable. Therefore, we resort to finding outer-approximations of the closed-loop reachable sets, $\bar{\mathcal{R}}_t(\mathcal{X}_0) \supseteq \mathcal{R}_t(\mathcal{X}_0)$. To obtain useful certificates, we want the approximations $\bar{\mathcal{R}}_t(\mathcal{X}_0)$ to be as tight as possible. Thus our goal is to compute the tightest outer-approximations of the t-step reachable sets. This can be addressed, for example, by solving the following optimization problem,

$$\begin{aligned} &\text{minimize} \;\; \text{Volume}\left(\bar{\mathcal{R}}_t(\mathcal{X}_0)\right) \\ &\text{subject to } \mathcal{R}_t(\mathcal{X}_0) \subseteq \bar{\mathcal{R}}_t(\mathcal{X}_0). \end{aligned} \tag{8}$$

In the following sections, we will derive a convex relaxation to the optimization problem (8).

III. Problem Abstraction via Quadratic Constraints

In this section, we use the framework of Quadratic Constraints (QCs) to abstract the reachable set estimation problem. The main idea is to replace the original closed-loop system f_π in (5) with an abstracted system $\tilde{f}_\pi$ in the sense that $\tilde{f}_\pi$ over-approximates the output of the original system for any given initial set $\mathcal{X}_0$, i.e. $f_\pi(\mathcal{X}_0) \subseteq \tilde{f}_\pi(\mathcal{X}_0)$. Based on this abstracted system, we will then show that we can compute the reachable sets via semidefinite programming (SDP). In the following, we will develop such an abstraction.

A. Quadratic Constraints

We begin with a formal definition of QCs for sets and functions [14].

Definition 1 (Quadratic Constraints) *Let $\mathcal{X} \subset \mathbb{R}^n$ be a nonempty set and define*

$$\mathcal{P} = \{P \in \mathbb{S}^{n+1} \mid \begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \ \forall x \in \mathcal{X}\}. \tag{9}$$

Then we say $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}$. The vector $\begin{bmatrix} x^\top & 1 \end{bmatrix}^\top$ is called the basis of this QC.

For each fixed $P \in \mathcal{P}$, the set of x's satisfying the quadratic inequality in (9) is a superset of $\mathcal{X}$. Indeed, we have that

$$\mathcal{X} \subseteq \bigcap_{P \in \mathcal{P}} \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \right\}. \tag{10}$$

In this paper, we use polytopes and ellipsoids to represent sets. Nonetheless, as addressed by [14], other sets such as hyper-rectangles and zonotopes can also be used.

Proposition 1 (QC for polytope [14]) *The polytope $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Hx \leq h\}$ with $H \in \mathbb{R}^{m \times n_x}$, $h \in \mathbb{R}^m$ satisfies the QC defined by*

$$\mathcal{P} = \left\{ P \;\middle|\; P = \begin{bmatrix} H^\top \Gamma H & -H^\top \Gamma h \\ -h^\top \Gamma H & h^\top \Gamma h \end{bmatrix}, \Gamma \in \mathbb{S}^m, \Gamma \geq 0 \right\}. \tag{11}$$

Proposition 2 (QC for ellipsoid [14]) *The ellipsoid $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid \|Ax + b\|_2 \leq 1\}$ with $A \in \mathbb{S}^{n_x}$ and $b \in \mathbb{R}^{n_x}$ satisfies the QC defined by*

$$\mathcal{P} = \left\{ P \;\middle|\; P = \mu \begin{bmatrix} -A^\top A & -A^\top b \\ -b^\top A & 1 - b^\top b \end{bmatrix}, \ \mu \geq 0 \right\}. \tag{12}$$

In light of Definition 1, we define QCs for functions as quadratic constraints satisfied by their *graph*, $G(\phi) = \{(x, y) \mid y = \phi(x)\}$.

Definition 2 (QC for functions [14]) *Let $\phi \colon \mathbb{R}^n \to \mathbb{R}^n$ and suppose $\mathcal{Q} \subset \mathbb{S}^{2n+1}$ is the set of all symmetric indefinite matrices Q such that*

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \geq 0. \tag{13}$$

Then we say ϕ satisfies the QC defined by $\mathcal{Q}$.

We will focus on the ReLU function throughout the paper but other types of activation functions such as sigmoid and tanh can also be represented by QCs [14]. In the following lemma, we abstract ReLU functions via QCs.

Lemma 1 (QC for ReLU function [14]) *The ReLU activation function $\phi(x) = \max(0, x) : \mathbb{R}^n \to \mathbb{R}^n$ satisfies the QC defined by*

$$\mathcal{Q} = \left\{ Q \in \mathbb{S}^{2n+1} \;\middle|\; Q = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{12}^\top & Q_{22} & Q_{23} \\ Q_{13}^\top & Q_{23}^\top & Q_{33} \end{bmatrix} \right\}, \tag{14}$$

with basis $[x^\top \ \phi(x)^\top \ 1]^\top$. The submatrices are

$$\begin{aligned} &Q_{11} = 0_{n \times n}, \ Q_{12} = \operatorname{diag}(\lambda) + T, \\ &Q_{13} = -\nu, \ Q_{22} = -2(\operatorname{diag}(\lambda) + T), \\ &Q_{23} = \nu + \eta, \ Q_{33} = 0. \end{aligned} \tag{15}$$

Here, $\eta, \nu \geq 0$ and $T \in \mathbb{S}_+^n$ is given by

$$T = \sum_{i=1}^{n} \lambda_i e_i e_i^\top + \sum_{i=1}^{n-1} \sum_{j>i}^{n} \lambda_{ij} (e_i - e_j)(e_i - e_j)^\top, \tag{16}$$

where $\lambda_{ij} \geq 0$ and $e_i \in \mathbb{R}^n$ has 1 in the i-th entry and 0 everywhere else.

Proof: See [14]. ∎

As we will see in Section IV, the Q matrix in (14) will appear as a decision variable in the SDP. Note that the QC in (14) holds globally for the whole space $\mathbb{R}^n$. When restricted to a local region in the state space, these QCs can be tightened to provide tighter bounds [14]. This would require one to compute lower and upper bounds on the preactivation values in each layer, which can be done by using interval arithmetic [25] or linear programming [16] as a presolve step.

IV. Reach-SDP: Computing Reachable Sets via Semidefinite Programming

In this section, we propose Reach-SDP, an optimization-based approach that uses the QC abstraction developed in the previous section to recursively estimate the reachable sets,

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \text{Reach\_SDP}\left(\bar{\mathcal{R}}_t(\mathcal{X}_0)\right), \tag{17}$$

for $t = 0, \cdots, N-1$. To begin, consider the system (1) with the projected neural network controller (4):

$$x_{t+1} = A_t x_t + B_t u_t + c_t, \ x_t \in \bar{\mathcal{R}}_t(\mathcal{X}_0) \tag{18a}$$

$$\begin{aligned} x_t^0 &= x_t \\ x_t^{k+1} &= \max(W^k x_t^k + b^k, 0) \quad k = 0, \cdots, \ell - 1 \\ x_t^{\ell+1} &= \max(W^\ell x_t^\ell + b^\ell - \underline{u}_t, 0) \\ x_t^{\ell+2} &= \max(-x^{\ell+1} + \bar{u}_t - \underline{u}_t, 0) \\ u_t &= -x_t^{\ell+2} + \bar{u}_t, \end{aligned} \tag{18b}$$

where we have embedded the projection operator (4) as two additional ReLU layers into the neural network. The road map to developing the method is as follows. First we abstract the input set $\bar{\mathcal{R}}_t(\mathcal{X}_0)$ and the neural network by quadratic constraints, each with a different basis vector. Then, we will use congruence transformations to represent all these quadratic constraints with the same basis vector

$$[\mathbf{x}_t^\top \ 1]^\top := [x_t^{0\top} \ x_t^{1\top} \ \cdots \ x_t^{\ell+2\top} \ 1]^\top \in \mathbb{R}^{n_x + n_n + 2n_u + 1}, \tag{19}$$

i.e., the vector obtained by concatenating the state vector x_t^0 and all the activation values $x_t^1, \cdots, x_t^{\ell+2}$. Finally, we show that we can propagate the quadratic constraints through the abstracted neural network by the S-procedure in the form of a semidefinite program. In the sequel, we will go through the technical details of the overall procedure.

A. Input Set

Proposition 3 *Suppose the set $\bar{\mathcal{R}}_t(\mathcal{X}_0)$ satisfies the QC defined by $\mathcal{P}$. Then, for any $P \in \mathcal{P}$, we have*

$$\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top M_{\text{in}}(P)\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} \geq 0 \;\; \forall\, x_t \in \bar{\mathcal{R}}_t(\mathcal{X}_0), \tag{20}$$

where $M_{\text{in}}(P) = E_{\text{in}}^\top P E_{\text{in}}$. The change-of-basis matrix is

$$E_{\text{in}} = \begin{bmatrix} I_{n_x} & 0 & \cdots & 0 & 0\\ 0 & 0 & \cdots & 0 & 1\end{bmatrix}.$$

B. Reachable Set

In our framework, we assume the candidate set $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$ that over-approximates $\mathcal{R}_{t+1}(\mathcal{X}_0)$ is represented by the intersection of finitely many *quadratic inequalities*,

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \bigcap_{i=1}^{m} \left\{ x_{t+1} \in \mathbb{R}^{n_x} \;\middle|\; \begin{bmatrix}x_{t+1}\\1\end{bmatrix}^\top S_i \begin{bmatrix}x_{t+1}\\1\end{bmatrix} \leq 0 \right\}, \tag{21}$$

where the matrices $S_i \in \mathbb{S}^{n_x+1}$ capture the shape and volume of the set and will appear as decision variables in the SDP. Typically, (21) is able to describe polytopic and ellipsoidal sets, which we discuss in detail now.

1) Polytopic reachable set: If the reachable set is to be over-approximated by a polytope, i.e. $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \{x \in \mathbb{R}^{n_x} \mid Hx \leq h\}$, then,

$$S_i = \begin{bmatrix} 0 & H_i \\ H_i^\top & -2h_i \end{bmatrix}, \tag{22}$$

where $H_i^\top \in \mathbb{R}^{1\times n_x}$ is the i-th row of $H \in \mathbb{R}^{m\times n_x}$ and $h_i \in \mathbb{R}$ is the i-th entry of $h \in \mathbb{R}^m$. Here, we require the H matrix, which determines the orientation of each facet of the polytope, to be given while we leave h_i's as decision variables in the SDP.

2) Ellipsoidal reachable set: If the reachable set is to be over-approximated by the ellipsoid $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \{x \in \mathbb{R}^{n_x} \mid \|Ax+b\|_2 \leq 1\}$, then,

$$S_i = S = \begin{bmatrix} A^\top A & A^\top b \\ b^\top A & b^\top b - 1 \end{bmatrix}, \tag{23}$$

where $A \in \mathbb{S}^{n_x}$ and $b \in \mathbb{R}^{n_x}$ are decision variables describing the center, orientation, and volume of the ellipsoid.

Proposition 4 *Let $\mathbf{x}_{t+1}$ be defined as in* (19) *and $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$ as in* (21)*. Then*

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \bigcap_{i=1}^{m} \left\{ \mathbf{x}_t \in \mathbb{R}^{n_x} \;\middle|\; \begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top S_i \begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} \leq 0 \right\}, \tag{24}$$

where $M_{\text{out}}(S_i) = E_{\text{out}}^\top S_i E_{\text{out}}$. The change-of-basis matrix is

$$E_{\text{out}} = \begin{bmatrix} A_t & 0 & \cdots & 0 & -B_t & B_t\bar{u}_t + c_t \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

Proposition 5 *Consider the projected neural network* (18b) *and let $\mathcal{Q}$ be defined as in* (14)*. Then for any $Q \in \mathcal{Q}$ we have*

$$\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top M_{\text{mid}}(Q)\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} \geq 0 \quad \forall\, x_t \in \mathbb{R}^{n_x}, \tag{25}$$

where $M_{\text{mid}}(Q) = E_{\text{mid}}^\top Q E_{\text{mid}}$. The change-of-basis matrix is

$$E_{\text{mid}} = \begin{bmatrix} E_1 & e_1 \\ E_2 & 0 \\ 0 & 1 \end{bmatrix}, \tag{26}$$

where

$$E_1 = \begin{bmatrix} W^0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & W^\ell & 0 & 0 \\ 0 & \cdots & 0 & -I_{n_u} & 0 \end{bmatrix} e_1 = \begin{bmatrix} b^0 \\ \vdots \\ b^{\ell-1} \\ b^\ell - \underline{u}_t \\ \bar{u}_t - \underline{u}_t \end{bmatrix}$$
$$E_2 = \begin{bmatrix} 0 & I_{n_1} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & 0 \\ 0 & 0 & \cdots & I_{n_\ell} & 0 & 0 \\ 0 & 0 & \cdots & 0 & I_{n_u} & 0 \\ 0 & 0 & \cdots & 0 & 0 & I_{n_u} \end{bmatrix}. \tag{27}$$

We are now ready to state our main result for over-approximating the reachable set $\mathcal{R}_{t+1}(\mathcal{X}_0)$ for the closed-loop system (18).

Theorem 1 (SDP for one-step reachable set) *Consider the closed-loop system* (18)*. Suppose the set $\bar{\mathcal{R}}_t(\mathcal{X}_0)$ satisfies the QC defined by $\mathcal{P}$. Let $\mathcal{Q}$ be defined as in* (14)*. Define $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$ as in* (21)*. Consider the LMI's*

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

for $i = 1, \cdots, m$. If there exists matrices $(P_i, Q_i) \in \mathcal{P}\times\mathcal{Q}$ that satisfy (28)*, then $\mathcal{R}_{t+1}(\mathcal{X}_0) \subseteq \bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$.*

Proof: Suppose (28) holds for some matrices $(P_i, Q_i) \in \mathcal{P}\times\mathcal{Q}$. By left- and right-multiplying both sides of (28) by $[\mathbf{x}_t^\top\ 1]$ and $[\mathbf{x}_t^\top\ 1]^\top$, the basis vector in (19), we have

$$\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top M_{\text{in}}\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top M_{\text{mid}}\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top M_{\text{out}}\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} \leq 0.$$

The first two quadratic terms are nonnegative for all $x_t \in \bar{\mathcal{R}}_t(\mathcal{X}_0)$ by Proposition 3 and 5, respectively. Consequently, the last quadratic term must be nonpositive for all $x_t \in \bar{\mathcal{R}}_t(\mathcal{X}_0)$,

$$\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix}^\top M_{\text{out}}(S)\begin{bmatrix}\mathbf{x}_t\\1\end{bmatrix} \leq 0 \;\forall x_t \in \bar{\mathcal{R}}_t(\mathcal{X}_0). \tag{29}$$

By Proposition 4, the above condition is equivalent to $x_{t+1} \in \bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$. ∎

C. Minimum-Volume Approximate Reachable Set

Theorem 1 provides a sufficient condition for over-approximating the reachable set $\mathcal{R}_{t+1}(\mathcal{X}_0)$. Now we can use this result to reformulate problem (8), which finds a minimum-volume approximate reachable set $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$.

If the approximate reachable set is parametrized by a polytope as in (22), it is difficult to find a minimum-volume polytope directly. However, given a matrix $H \in \mathbb{R}^{m\times n_x}$

that describes the facets of the polytope, we can solve the following SDP problem,

$$\underset{P\in\mathcal{P},\ Q\in\mathcal{Q},\ h_i\in\mathbb{R}}{\text{minimize}} \ h_i \quad \text{subject to (28)}, \tag{30}$$

for all $i = 1, \cdots, m$. For each fixed facet $H_i \in \mathbb{R}^{n_x}$ in H, SDP (30) finds the tightest halfspace $\{y \mid H_i^\top y \leq h_i\}$ that contains $\mathcal{R}_{t+1}(\mathcal{X}_0)$. Finally, the polytopic approximate reachable set is given by the intersection of those halfspaces, i.e. $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \{y \mid Hy \leq h\}$, as in Section IV-B.

If the approximate reachable set is parametrized by an ellipsoid as in (23), we can easily obtain a minimum-volume ellipsoid that encloses $\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0)$ by solving,

$$\underset{\substack{P\in\mathcal{P},\ Q\in\mathcal{Q},\\ A\in\mathbb{S}^{n_x},\ b\in\mathbb{R}^{n_x}}}{\text{minimize}} \ -\log\det(A) \quad \text{subject to (28)}. \tag{31}$$

Note that (28) is not convex in A and b. Nonetheless we can find a convex constraint equivalent to (28) using Schur complements. See [26] for details.

V. Numerical Experiments

In this section, we demonstrate our approach with two application examples. The controllers used to generate training data were implemented in YALMIP [27]. All neural network controllers were trained with ReLU activation functions and the Adam algorithm in PyTorch. We used MATLAB, CVX [28] and Mosek [29] to solve the semidefinite programs.

A. Double Integrator

We first consider a double integrator system

$$x_{t+1} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{A} x_t + \underbrace{\begin{bmatrix} 0.5 \\ 1 \end{bmatrix}}_{B} u_t, \tag{32}$$

discretized with sampling time $t_s = 1$s and subject to state and input constraints, $\mathcal{A}^{\complement} = [-5, 5] \times [-5, 5]$ and $\mathcal{U} = [-1, 1]$, respectively. We implemented a standard linear MPC following [30] with a prediction horizon $N_{\text{MPC}} = 10$, weighting matrices $Q = I_2$, $R = 1$, the terminal region $\mathcal{O}_\infty^{LQR}$ and the terminal weighting matrix P_∞ synthesized from the discrete-time algebraic Riccati equation. The MPC is designed as a stabilizing controller which steers the system to the origin while satisfying the constraints. We then use the MPC controller to generate 2420 samples of state and input pairs $(x, \pi_{\text{MPC}}(x))$ for learning. The neural network has 4 hidden layers each with 10 neurons and ReLU activation function. Our goal is to verify if all initial states in $\mathcal{X}_0 = [3, 4] \times [0.5, 1.5]$ can reach the set $\mathcal{G} = [-0.5, 0.5] \times [-0.5, 0.5]$, a region near the origin, in $N = 6$ steps while avoiding $\mathcal{A}$ at all times. We computed the polytopic approximate reachable sets $\bar{\mathcal{R}}_1(\mathcal{X}_0), \cdots, \bar{\mathcal{R}}_6(\mathcal{X}_0)$ using the Reach-SDP introduced in Section IV. As shown in Figure 2, our approach yielded a tight outer-approximation of the reachable sets and successfully verified the safety properties sought.

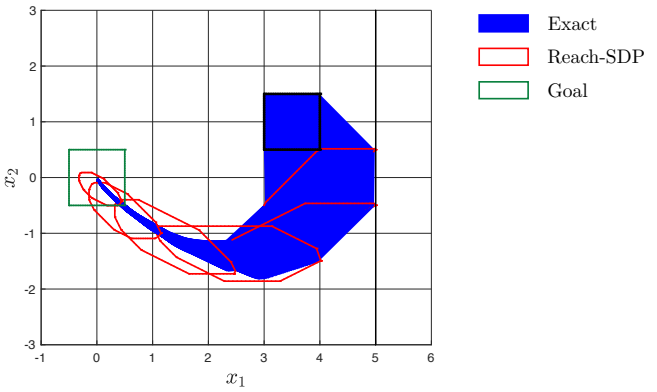

Fig. 2: Illustration of the initial set $\mathcal{X}_0$ (black), exact reachable sets $\mathcal{R}_t(\mathcal{X}_0)$ (blue), approximate reachable sets $\bar{\mathcal{R}}_t(\mathcal{X}_0)$ (red) given by Reach-SDP and the goal set $\mathcal{G}$ (green). The solid black line represents the state constraint $x_1 \leq 5$.

B. 6D Quadrotor

In the second example, we apply Reach-SDP to a 6D quadrotor model in [31]. The dynamics are given as follows,

$$\dot{x} = \underbrace{\begin{bmatrix} 0_{3\times3} & I_3 \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}}_{A} x + \underbrace{\begin{bmatrix} & 0_{3\times3} & \\ g & 0 & 0 \\ 0 & -g & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} \tan(\theta) \\ \tan(\phi) \\ \tau \end{bmatrix}}_{u} + \underbrace{\begin{bmatrix} 0_{5\times1} \\ -g \end{bmatrix}}_{c} \tag{33}$$

where g is the gravitational acceleration, the state vector $x = [p_x, p_y, p_z, v_x, v_y, v_z]^\top$ include positions and velocities of the quadrotor in the 3D space and the control vector u is a function of θ (pitch), ϕ (roll) and τ (thrust), which are the control inputs of the original model. The control task is to steer the quadrotor to the origin while respecting the state constraints $\mathcal{A}^{\complement} = [-5, 5] \times [-5, 5] \times [-5, 5] \times [-1, 1] \times [-1, 1] \times [-1, 1]$ and the actuator constraints $[\theta, \phi, \tau]^\top \in [-\pi/9, \pi/9] \times [-\pi/9, \pi/9] \times [0, 2g]$. We implemented a nonlinear MPC with a prediction horizon $N_{\text{MPC}} = 30$, weighting matrices $Q = I_6$, $R = I_4$ and the terminal constraint $x_{N_{\text{MPC}}} = 0$. The continuous-time model in (33) is discretized with a sampling time $t_s = 0.1$s using the RungeKutta 4th order method. A total number of 4531 feasible samples of state and input pairs $(x, \pi_{\text{MPC}}(x))$ were generated and used to train a neural network with 2 hidden layers and 32 neurons in each layer. Note that the actual control input θ and ϕ fed into the quadrotor can be effectively obtained by applying $\arctan$ to the first two entries in $\pi_{\text{MPC}}(x)$. The initial set is given as an ellipsoid $\mathcal{X}_0 = \mathcal{E}(q_0, Q_0)$, where $q_0 = [4.7\ 4.7\ 3\ 0.95\ 0\ 0]^\top$ is the center and $Q_0 = \text{diag}(0.05^2, 0.05^2, 0.05^2, 0.01^2, 0.01^2, 0.01^2)$ is the shape matrix. Here, we want to verify if all initial states in $\mathcal{X}_0$ can reach the set $\mathcal{G} = [3.7, 4.1] \times [2.5, 3.5] \times [1.2, 2.6]$, which is defined in the (p_x, p_y, p_z)-space, in $t = 1$ second subject to the state and input constraints. We approximated the ellipsoidal forward reachable sets $\bar{\mathcal{R}}_1(\mathcal{X}_0), \cdots, \bar{\mathcal{R}}_{10}(\mathcal{X}_0)$ using the Reach-SDP. The resulting approximate reachable sets are plotted in Figure 3 which shows that our method is

able to verify the given reach-avoid specifications.

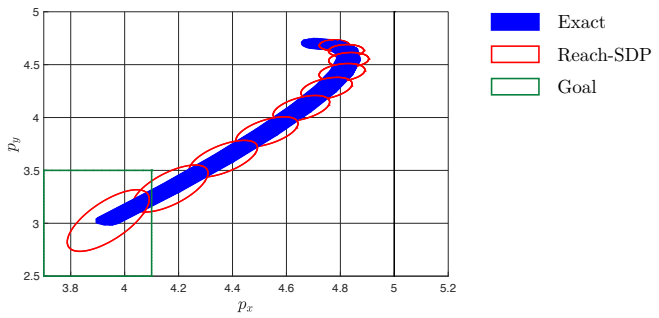

Fig. 3: Illustration of the exact and approximate reachable sets of the quadrotor system in the (p_x, p_y)-space. The solid black line represents the state constraint $p_x \leq 5$.

VI. Conclusions

In this paper, we propose a novel convex-optimization-based reachability analysis method for linear systems in feedback interconnection with neural network controllers. Our approach relies on abstracting the nonlinear components of the closed-loop system by quadratic constraints. Then we show that we can compute a guaranteed outer-approximation of the reachable sets via semidefinite programming. Future work includes extending the current approach to incorporate nonlinear dynamics and to approximate backward reachable sets, which is useful for certifying invariance properties.

References

[1] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 528–535, IEEE, 2016.

[2] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[3] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124, IEEE, 2019.

[4] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

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

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

[7] A. Bemporad and M. Morari, "Verification of hybrid systems via mathematical programming," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 31–45, Springer, 1999.

[8] A. Bemporad, F. D. Torrisi, and M. Morari, "Optimization-based verification and stability characterization of piecewise affine and hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 45–58, Springer, 2000.

[9] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 986–1001, 2003.

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

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

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

[13] 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.

[14] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *arXiv preprint arXiv:1903.01287*, 2019.

[15] 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.

[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] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, "Reachnn: Reachability analysis of neural-network controlled systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–22, 2019.

[18] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 169–178, 2019.

[19] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 157–168, 2019.

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

[21] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *2018 Annual American control conference (ACC)*, pp. 1520–1527, IEEE, 2018.

[22] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 7450–7460, 2018.

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

[24] 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.

[25] 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.

[26] 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.

[27] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pp. 284–289, IEEE, 2004.

[28] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2009.

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

[30] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[31] D. M. Lopez, P. Musau, H.-D. Tran, and T. T. Johnson, "Verification of closed-loop systems with neural network controllers," in *6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, vol. 61, pp. 201–210, 2019.