Interval Reachability of Nonlinear Dynamical Systems with Neural Network Controllers

Saber Jafarpour[*] SABER@GATECH.EDU
Georgia Institute of Technology

Akash Harapanahalli[*] AHARAPAN@GATECH.EDU
Georgia Institute of Technology

Samuel Coogan SAM.COOGAN@GATECH.EDU
Georgia Institute of Technology

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

This paper proposes a computationally efficient framework, based on interval analysis, for rigorous verification of nonlinear continuous-time dynamical systems with neural network controllers. Given a neural network, we use an existing verification algorithm to construct inclusion functions for its input-output behavior. Inspired by mixed monotone theory, we embed the closed-loop dynamics into a larger system using an inclusion function of the neural network and a decomposition function of the open-loop system. This embedding provides a scalable approach for safety analysis of the neural control loop while preserving the nonlinear structure of the system.

We show that one can efficiently compute hyper-rectangular over-approximations of the reachable sets using a single trajectory of the embedding system. We design an algorithm to leverage this computational advantage through partitioning strategies, improving our reachable set estimates while balancing its runtime with tunable parameters. We demonstrate the performance of this algorithm through two case studies. First, we demonstrate this method's strength in complex nonlinear environments. Then, we show that our approach matches the performance of the state-of-the art verification algorithm for linear discretized systems.

Keywords: Reachability analysis, Neural feedback loop, Verification of neural networks, Safety verification.

1. Introduction

Neural network components are increasingly deployed as controllers in safety-critical applications such as self-driving vehicles and robotic systems. For example, they may be designed based on reinforcement learning algorithms (Zhang et al., 2016) or trained to approximate some dynamic optimization-based controllers (Chen et al., 2018). However, neural networks are known to be vulnerable to small input perturbations (Szegedy et al., 2014); a slight disturbance in their input can lead to a large change in their output. In many applications, these learning-based components are interconnected with nonlinear and time-varying systems. Moreover, they are usually trained with no safety and robustness guarantees. Their learned control policies can therefore suffer from a significant degradation in performance when presented with uncertainties not accounted for during training.

[*] These authors contributed equally

© 2023 S. Jafarpour, A. Harapanahalli & S. Coogan.

Related works. There is extensive literature on verification of isolated neural networks. Rigorous verification approaches generally fall into three different categories: (i) reachability-based methods which focus on layer-by-layer estimation of reachable sets using interval bound propagation (Mirman et al., 2018; Gowal et al., 2018; Wang et al., 2018), activation function relaxation (Zhang et al., 2018), and symbolic interval analysis (Wang et al., 2018); (ii) optimization-based methods which use linear programming (Wong and Kolter, 2018), semi-definite programming (Fazlyab et al., 2019), or search and optimization (Katz et al., 2017) to estimate the input-output behavior of the neural networks; and (iii) probabilistic methods (Cohen et al., 2019; Li et al., 2019). We refer to the survey (Liu et al., 2021) for a review of neural network verification algorithms. Reachability of nonlinear dynamical systems has been studied using optimization-based methods such as the Hamilton-Jacobi approach (Bansal et al., 2017) and the level set approach (Mitchell and Tomlin, 2000). Several computationally tractable approaches including ellipsoidal (Kurzhanski and Varaiya, 2000) and zonotope methods (Girard, 2005) have been developed for reachability analysis of linear systems. Mixed monotone theory has emerged as a computationally efficient framework for over-approximating the reachable sets of nonlinear systems (Coogan and Arcak, 2015).

Recently, there has been an increased interest in verification methods for closed-loop systems with neural network controllers. However, a direct combination of state-of-the art neural network verification algorithms with the existing reachability analysis toolboxes and can lead to overly-conservative estimates of reachable sets (Dutta et al., 2019, Section 2.1). For linear discrete-time systems, reachable set over-approximation has been studied using semi-definite programming (Hu et al., 2020) and linear programming (Everett et al., 2021b,a). For nonlinear discrete-time systems, Sidrane et al. (2022) establishes a mixed integer programming framework for reachable set over-approximation using polynomial bounds of the system dynamics. For nonlinear continuous-time systems, Dutta et al. (2019) uses polynomial bounding on the dynamics as well as the neural network to over-approximate reachable sets. Other rigorous approaches for verification of closed-loop neural network controllers include using finite-state abstractions (Sun et al., 2019) and simulation-guided interval analysis (Xiang et al., 2021).

Contributions. In this paper, we use elements of mixed monotone system theory to develop a flexible framework for safety verification of nonlinear continuous-time systems with neural network controllers. First, we employ CROWN—a well-established isolated neural network verification framework (Zhang et al., 2018)—to construct an inclusion function for a pre-trained neural network's output. Then, we use this inclusion function and a decomposition function of the open-loop system to construct suitable embedding systems for the closed-loop system using twice the number of states. Finally, we simulate trajectories of these embedding systems to compute hyper-rectangular over-approximations of the closed-loop reachable sets. Our framework has several advantageous features. It is agnostic to the neural network verifier, only requiring an inclusion function for the neural network. Additionally, our approach is fast and scalable, as only a single simulation of the embedding system is required. This feature, combined with a clever partitioning of the state space, is used to improve the accuracy of our reachable set over-approximations while retaining computational viability. Through several numerical experiments, we study the efficiency of our approach for obstacle avoidance in a simple vehicle model and for perturbation analysis in a linear quadrotor model.

2. Notation

For every $x \in \mathbb{R}^n$ and every $r \in \mathbb{R}_{\geq 0}$ we define the closed ball $B_\infty(x,r) = \{y \in \mathbb{R}^n \mid \|y-x\|_\infty \leq r\}$. The partial order $\leq$ on $\mathbb{R}^n$ is defined by $x \leq y$ if and only if $x_i \leq y_i$, for every $i \in \{1,\dots,n\}$. For every $x \leq y$, we can define the interval $[x,y] = \{z \in \mathbb{R}^n \mid x \leq z \leq y\}$. The partial order $\leq$ on $\mathbb{R}^n$ induces the southeast partial order $\leq_{\mathrm{SE}}$ on $\mathbb{R}^{2n}$ defined by $\left[\begin{smallmatrix} x \\ \widehat{x} \end{smallmatrix}\right] \leq_{\mathrm{SE}} \left[\begin{smallmatrix} y \\ \widehat{y} \end{smallmatrix}\right]$ if $x \leq y$ and $\widehat{y} \leq \widehat{x}$. We define the subsets $\mathcal{T}^{2n}_{\geq 0}, \mathcal{T}^{2n}_{\leq 0}, \mathcal{T}^{2n} \subseteq \mathbb{R}^{2n}$ as follows:

$$\mathcal{T}^{2n}_{\geq 0} = \{\left[\begin{smallmatrix} x \\ \widehat{x} \end{smallmatrix}\right] \in \mathbb{R}^{2n} \mid x \leq \widehat{x}\}, \qquad \mathcal{T}^{2n}_{\leq 0} = \{\left[\begin{smallmatrix} x \\ \widehat{x} \end{smallmatrix}\right] \in \mathbb{R}^{2n} \mid x \geq \widehat{x}\}, \qquad \mathcal{T}^{2n} = \mathcal{T}^{2n}_{\geq 0} \cup \mathcal{T}^{2n}_{\leq 0}.$$

For every two vectors $v, w \in \mathbb{R}^n$ and every $i \in \{1,\dots,n\}$, we define the vector $v_{[i:w]} \in \mathbb{R}^n$ by $\left(v_{[i:w]}\right)_j = \begin{cases} v_j & j \neq i \\ w_j & j = i. \end{cases}$, for every $j \in \{1,\dots,n\}$. Given a matrix $B \in \mathbb{R}^{n \times m}$, we denote the non-negative part of B by $[B]^+ = \max(B,0)$ and the nonpositive part of B by $[B]^- = \min(B,0)$. The Metzler and non-Metzler part of square matrix $A \in \mathbb{R}^{n\times n}$ are denoted by $\lceil A \rceil^{\mathrm{Mzl}}$ and $\lfloor A \rfloor^{\mathrm{Mzl}}$, respectively, where $(\lceil A \rceil^{\mathrm{Mzl}})_{ij} = \begin{cases} A_{ij} & A_{ij} \geq 0 \text{ or } i = j \\ 0 & \text{otherwise,} \end{cases}$ and $\lfloor A \rfloor^{\mathrm{Mzl}} = A - \lceil A \rceil^{\mathrm{Mzl}}$. Consider a control system $\dot{x} = f(x,u)$ on $\mathbb{R}^n$ with a measurable input $\mathbf{u} : \mathbb{R}_{\geq 0} \to \mathbb{R}^p$. We denote the trajectory of the system with the control input $\mathbf{u}$ starting from $x_0 \in \mathbb{R}^n$ at time t_0 by $t \mapsto \phi_f(t,t_0,x_0,u)$. Given a map $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and a Lipschitz function $\mathsf{F} : \mathcal{T}^{2n} \times \mathcal{T}^{2m} \to \mathbb{R}^n$, we say that F is a *decomposition function for f* if

(i) $\mathsf{F}_i(x,x,u,u) = f_i(x,u)$, for every $x \in \mathbb{R}^n$ and every $u \in \mathbb{R}^p$;
(ii) $\mathsf{F}_i(x,\widehat{x},u,\widehat{u}) \leq \mathsf{F}_i(y,\widehat{y},u,\widehat{u})$, for every $x \leq y$ such that $x_i = y_i$, and every $\widehat{y} \leq \widehat{x}$;
(iii) $\mathsf{F}_i(x,\widehat{x},u,\widehat{u}) \leq \mathsf{F}_i(x,\widehat{x},v,\widehat{v})$, for every $u \leq v$ and every $\widehat{v} \leq \widehat{u}$.

A decomposition function F is *tight*, if for every other decomposition function d, we have

$$\begin{bmatrix} d(x,\widehat{x},u,\widehat{u}) \\ d(\widehat{x},x,\widehat{u},u) \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \mathsf{F}(x,\widehat{x},u,\widehat{u}) \\ \mathsf{F}(\widehat{x},x,\widehat{u},u) \end{bmatrix}, \qquad \text{for every } x \leq \widehat{x},\ w \leq \widehat{w}.$$

Given a map $g : \mathbb{R}^n \to \mathbb{R}^m$, the function $\begin{bmatrix} \underline{\mathsf{G}} \\ \overline{\mathsf{G}} \end{bmatrix} : \mathcal{T}^{2n}_{\geq 0} \to \mathcal{T}^{2m}_{\geq 0}$ is *an inclusion function for g* if, for every $x \leq \widehat{x}$ and every $z \in [x,\widehat{x}]$, we have $\underline{\mathsf{G}}(x,\widehat{x}) \leq g(z) \leq \overline{\mathsf{G}}(x,\widehat{x})$. Note that our definition of inclusion function is closely-related to the notion of *inclusion interval function* in the interval analysis (Jaulin et al., 2001, Section 2.4.1).

3. Problem Statement

We consider a nonlinear plant of the form

$$\dot{x} = f(x,u,w) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^p$ is the control input, and $w \in \mathcal{W} \subseteq \mathbb{R}^q$ is the disturbance. We assume that $f : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^n$ is a parameterized vector field and the state feedback is parameterized by a k-layer feed-forward neural network controller $\mathsf{N} : \mathbb{R}^n \to \mathbb{R}^p$ defined by:

$$\begin{aligned} \xi^{(i)}(x) &= \phi_{i-1}(W^{(i-1)}\xi^{(i-1)}(x) + b^{(i-1)}), \quad i \in \{1,\dots,k\} \\ x = \xi^{(0)} \qquad u &= W^{(k)}\xi^{(k)}(x) + b^{(k)} := \mathsf{N}(x), \end{aligned} \tag{2}$$

where n_i is the number of neurons in the ith layer, $W^{(i-1)} \in \mathbb{R}^{n_i \times n_{i-1}}$ is the weight matrix of the ith layer, $b^{(i-1)} \in \mathbb{R}^{n_i}$ is the bias vector of the ith layer, and $\xi^{(i)}(y) \in \mathbb{R}^{n_i}$ is the i-th layer hidden variable and the activation function ϕ_i satisfies $0 \leq \frac{\phi_i(x)-\phi_i(y)}{x-y} \leq 1$. One can show that a large class of activation function including, but not restricted to, ReLU, leaky ReLU, sigmoid, and tanh satisfies this condition (after a possibly re-scaling of their co-domain). We assume that, the neural network (2) is trained to approximate an offline controller with the auxiliary objective of achieving a goal set $\mathcal{G} \subseteq \mathbb{R}^n$ while remaining in a safe set $\mathcal{S} \subseteq \mathbb{R}^n$. Our aim is to consider the closed-loop system given by:

$$\dot{x} = f(x, \mathsf{N}(x), w) := f^{\mathrm{cl}}(x, w). \tag{3}$$

and to verify its safety, i.e., to ensure that the closed-loop system avoids the unsafe domain $\mathbb{R}^n/\mathcal{S}$. We define the reachable set of the closed-loop system (3) by:

$$\mathcal{R}(t, \mathcal{X}_0, \mathcal{W}) = \{\phi_{f^{\mathrm{cl}}}(t, 0, x_0, \mathbf{w}) \mid x_0 \in \mathcal{X}_0, \quad \mathbf{w} : \mathbb{R}_{\geq 0} \to \mathcal{W} \quad \text{is piecewise continuous}\}$$

Thus, our goal is to check if $\mathcal{R}(t, \mathcal{X}_0, \mathcal{W}) \subseteq \mathcal{S}$ holds for every $t \in \mathbb{R}_{\geq 0}$. In general, computing the exact reachable sets of the closed-loop system (3) is not computationally tractable. Our approach is based on constructing a computationally efficient over-approximation $\overline{\mathcal{R}}(t, \mathcal{X}_0, \mathcal{W})$ of the reachable set of the closed-loop system. Then, avoiding the unsafe set $\mathbb{R}^n/\mathcal{S}$ is guaranteed when $\overline{\mathcal{R}}(t, \mathcal{X}_0, \mathcal{W}) \subseteq \mathcal{S}$, for every $t \in \mathbb{R}_{\geq 0}$.

4. Input-output reachability of neural networks

In order to estimate the input-output behavior of the neural network controller, we use the verification algorithm called CROWN (Zhang et al., 2018). We first use CROWN to obtain an inclusion function for the input-output map of the neural network. Consider the neural network (2) and let the perturbed input vector x be in the interval $[\underline{x}, \overline{x}]$. For every $i \in \{1, \ldots, k\}$, we define the pre-activation input to ith layer as $z^{(i)} = W^{(i)}\xi^{(i-1)}(x) + b^{(i)}$. When x is perturbed in the interval $[\underline{x}, \overline{x}]$, we assume that $L^{(i)}, U^{(i)} \in \mathbb{R}^{n_i}$ are such that $L^{(i)} \leq z^{(i)} \leq U^{(i)}$. In this case, for the jth neuron in the ith layer, there exist $\alpha^{(i)}_{U,j}, \beta^{(i)}_{U,j}, \alpha^{(i)}_{L,j}, \beta^{(i)}_{L,j}$ such that

$$\alpha^{(i)}_{L,j}(z + \beta^{(i)}_{L,j}) \leq \phi_i(z) \leq \alpha^{(i)}_{U,j}(z + \beta^{(i)}_{U,j}), \qquad \text{for every } L^{(i)}_j \leq z \leq U^i_j. \tag{4}$$

For every $y \in [\underline{y}, \overline{y}]$, the output of the neural network is bounded as below:

$$\underline{A}(\underline{x}, \overline{x})x + \underline{b}(\underline{x}, \overline{x}) \leq \mathsf{N}(x) \leq \overline{A}(\underline{x}, \overline{x})x + \overline{b}(\underline{x}, \overline{x}), \tag{5}$$

where matrix-valued functions $\underline{A}, \overline{A} : \mathcal{T}^{2n}_{\geq 0} \to \mathbb{R}^{p \times n}$ and two vector-valued functions $\underline{b}, \overline{b} : \mathcal{T}^{2n}_{\geq 0} \to \mathbb{R}^p$ are defined as follows:

$$\begin{aligned} \overline{A}(\underline{x}, \overline{x}) &= \Lambda^{(0)}, & \overline{b}(\underline{x}, \overline{x}) &= \sum\nolimits_{i=1}^{k} \Lambda^{(i)} b^{(i)} + \operatorname{diag}(\Lambda^{(i)}\Delta^{(i)}), \\ \underline{A}(\underline{x}, \overline{x}) &= \Omega^{(0)}, & \underline{b}(\underline{x}, \overline{x}) &= \sum\nolimits_{i=1}^{k} \Omega^{(i)} b^{(i)} + \operatorname{diag}(\Omega^{(i)}\Theta^{(i)}) \end{aligned} \tag{6}$$

where, for every $i \in \{1, \ldots, k\}$, $\Lambda^{(i)}, \Delta^{(i)}, \Omega^{(i)}, \Theta^{(i)} \in \mathbb{R}^{n_k \times n_i}$ are as defined in (Zhang et al., 2018, Theorem 3.2) for the input perturbation set $B_\infty(\frac{\underline{x}+\overline{x}}{2}, \frac{\overline{x}-\underline{x}}{2})$.

Theorem 1 (Inclusion functions for Neural Networks) *Consider the k-layer feed-forward neural network $u = \mathsf{N}(x)$ given by (2). Then,*

(i) for every input perturbation interval $[\underline{x}, \overline{x}]$, the mapping $\left[\begin{smallmatrix}\underline{\mathsf{G}}_{[\underline{x},\overline{x}]}\\ \overline{\mathsf{G}}_{[\underline{x},\overline{x}]}\end{smallmatrix}\right] : \mathcal{T}^{2n}_{\geq 0} \to \mathcal{T}^{2p}_{\geq 0}$ defined by

$$\begin{aligned}\underline{\mathsf{G}}_{[\underline{x},\overline{x}]}(x, \widehat{x}) &= [\underline{A}(\underline{x}, \overline{x})]^{+}x + [\underline{A}(\underline{x}, \overline{x})]^{-}\widehat{x} + \underline{b}(\underline{x}, \overline{x}),\\ \overline{\mathsf{G}}_{[\underline{x},\overline{x}]}(x, \widehat{x}) &= [\overline{A}(\underline{x}, \overline{x})]^{+}\widehat{x} + [\overline{A}(\underline{x}, \overline{x})]^{-}x + \overline{b}(\underline{x}, \overline{x}).\end{aligned} \tag{7}$$

is an inclusion function for the neural network N on $[\underline{x}, \overline{x}]$;

(ii) the mapping $\left[\begin{smallmatrix}\underline{\mathsf{H}}\\ \overline{\mathsf{H}}\end{smallmatrix}\right] : \mathcal{T}^{2n}_{\geq 0} \to \mathcal{T}^{2p}_{\geq 0}$ defined by $\underline{\mathsf{H}}(x, \widehat{x}) = \underline{\mathsf{G}}_{[x,\widehat{x}]}(x, \widehat{x})$ and $\overline{\mathsf{H}}(x, \widehat{x}) = \overline{\mathsf{G}}_{[x,\widehat{x}]}(x, \widehat{x})$ is an inclusion function for the neural network N on $\mathbb{R}^n$.

Proof Regarding part (i), suppose that $\eta \leq \widehat{\eta} \in [\underline{x}, \overline{x}]$ and $z \in [\eta, \widehat{\eta}]$. Then,

$$\begin{aligned}\overline{\mathsf{G}}_{[\underline{x},\overline{y}]}(\eta, \widehat{\eta}) &= [\overline{A}(\underline{x}, \overline{x})]^{+}\widehat{\eta} + [\overline{A}(\underline{x}, \overline{x})]^{-}\eta + \overline{b}(\overline{x}, \overline{x}) \geq [\overline{A}(\underline{x}, \overline{x})]^{+}z + [\overline{A}(\underline{x}, \overline{x})]^{-}z + \overline{b}(\overline{x}, \overline{x})\\ &= [\overline{A}(\underline{x}, \overline{x})]z + \overline{b}(\overline{x}, \overline{x}) \geq \mathsf{N}(z),\end{aligned}$$

where the first inequality holds because $z \in [\eta, \widehat{\eta}]$, the matrix $[\overline{A}(\underline{x}, \overline{x})]^{+}$ is non-negative, and the matrix $[\overline{A}(\underline{x}, \overline{x})]^{+}$ is non-positive. The second inequality holds by noting the fact that $[\overline{A}(\underline{x}, \overline{x})]^{+} + [\overline{A}(\underline{x}, \overline{x})]^{-} = [\overline{A}(\underline{x}, \overline{x})]$, for every $\underline{x} \leq \overline{x}$. Finally, the last inequality holds by (5). Similarly, one can show that $\underline{\mathsf{G}}_{[\underline{x},\overline{x}]}(\widehat{\eta}, \eta) \leq \mathsf{N}(z)$. Regarding part (ii), suppose that $x \leq \widehat{x}$, for every $z \in [x, \widehat{x}]$, one can use a similar argument as part (i) to show that

$$\begin{aligned}\overline{\mathsf{G}}_{[x,\widehat{x}]}(x, \widehat{x}) &= [\overline{A}(x, \widehat{x})]^{+}\widehat{x} + [\overline{A}(x, \widehat{x})]^{-}x + \overline{b}(x, \widehat{x}) \geq [\overline{A}(x, \widehat{x})]^{+}z + [\overline{A}(x, \widehat{x})]^{-}z + \overline{b}(x, \widehat{x})\\ &= [\overline{A}(x, \widehat{x})]z + \overline{b}(x, \widehat{x}) \geq \mathsf{N}(z).\end{aligned}$$

One can similarly show that $\underline{\mathsf{G}}_{[x,\widehat{x}]}(\widehat{x}, x) \leq \mathsf{N}(z)$, for every $z \in [x, \widehat{x}]$. Moreover, suppose that $\underline{x} = \overline{x} = z$. In this case, both inequalities in (4) are equality with $\alpha^{(i)}_{L} = \alpha^{(i)}_{U}$ and $\beta^{(i)}_{L} = \beta^{(i)}_{U}$. Thus, using the equation (6), we get that $\underline{A}(z, z) = \overline{A}(z, z)$ and $\underline{b}(z, z) = \overline{b}(z, z)$. This implies that $\mathsf{N}(z) = \underline{A}(z, z)z + \underline{b}(z, z) = \underline{\mathsf{G}}_{[z,z]}(z, z)$. Thus, $\left[\begin{smallmatrix}\underline{\mathsf{H}}\\ \overline{\mathsf{H}}\end{smallmatrix}\right]$ is an inclusion function for N on the $\mathbb{R}^n$. ■

Remark 2 *The following remarks are in order.*

(i) For each layer $i \in \{1, \ldots, k\}$, the intermediate pre-activation bounds U^i, L^i can be computed using either the Interval Bound Propagation (IBP) (Gowal et al., 2018) or using the CROWN itself (Zhang et al., 2018). In the former case, the IBP can be considered as a special case of CROWN by choosing $\alpha^{(i)}_{U} = \alpha^{(i)}_{L} = 0$ and $\beta^{(i)}_{L} = L^{(i)}$ and $\beta^{(i)}_{U} = U^{(i)}$, for every $i \in \{1, \ldots, k\}$.

(ii) The inclusion function $\left[\begin{smallmatrix}\underline{\mathsf{G}}_{[\underline{x},\overline{x}]}\\ \overline{\mathsf{G}}_{[\underline{x},\overline{x}]}\end{smallmatrix}\right]$ defined in Theorem 1(i) is linear but it depends on the input perturbation interval $[\underline{x}, \overline{x}]$. On the other hand, the inclusion function $\left[\begin{smallmatrix}\underline{\mathsf{H}}\\ \overline{\mathsf{H}}\end{smallmatrix}\right]$ defined in Theorem 1(ii) is nonlinear but it is independent of the input perturbation set. For both inclusion functions, it can be shown that the smaller the input perturbation interval, the tighter the relaxation bounds in equation (4) are and, thus, the tighter the over- and the under-approximations of the neural network in equation (5) are.

5. Interval reachability of closed-loop system

In this section, we present a system-level approach for over-approximating the reachable set of the closed-loop system (3) with the neural network controller (2). The key idea is to design a suitable function $d : \mathcal{T}^{2n} \times \mathcal{T}^{2q} \to \mathbb{R}^n$ and use it to construct the following embedding system associated to the closed-loop system (3):

$$\frac{d}{dt}\begin{bmatrix} x \\ \widehat{x} \end{bmatrix} = \begin{bmatrix} d(x, \widehat{x}, w, \widehat{w}) \\ d(\widehat{x}, x, \widehat{w}, w) \end{bmatrix} \tag{8}$$

Then, one can use the embedding system (8) to study the propagation of the state and disturbance bounds with time. Suppose that the initial condition set is given by $\mathcal{X}_0 \subseteq [\underline{x}_0, \overline{x}_0]$ and the disturbance set is given by $\mathcal{W} \subseteq [\underline{w}, \overline{w}]$. Let F be a decomposition function for the open-loop system (1) and $\left[\begin{smallmatrix} \underline{\mathsf{G}} \\ \overline{\mathsf{G}} \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} \underline{\mathsf{H}} \\ \overline{\mathsf{H}} \end{smallmatrix}\right]$ be the inclusion functions of the neural network (2) established in Theorem 1. We introduce "global" function d^{G}, "hybrid" function d^{H}, and "local" function d^{L}, for every $i \in \{1, \dots, n\}$, as follows:

$$d_i^{\mathrm{G}}(x, \widehat{x}, w, \widehat{w}) = \begin{cases} \mathsf{F}_i(x, \widehat{x}, \underline{\mathsf{H}}(x, \widehat{x}), \overline{\mathsf{H}}(x, \widehat{x}), w, \widehat{w}), & x \leq \widehat{x}, w \leq \widehat{w}, \\ \mathsf{F}_i(\widehat{x}, x, \overline{\mathsf{H}}(x, \widehat{x}), \underline{\mathsf{H}}(x, \widehat{x}), \widehat{w}, w), & \widehat{x} \leq x, \widehat{w} \leq w, \end{cases}$$

$$d_i^{\mathrm{H}}(x, \widehat{x}, w, \widehat{w}) = \begin{cases} \mathsf{F}_i(x, \widehat{x}, \underline{\mathsf{G}}_{[x,\widehat{x}]}(x, \widehat{x}_{[i:x]}), \overline{\mathsf{G}}_{[x,\widehat{x}]}(x, \widehat{x}_{[i:x]}), w, \widehat{w}), & x \leq \widehat{x}, w \leq \widehat{w}, \\ \mathsf{F}_i(\widehat{x}, x, \overline{\mathsf{G}}_{[x,\widehat{x}]}(x_{[i:\widehat{x}]}, \widehat{x}), \underline{\mathsf{G}}_{[x,\widehat{x}]}(x_{[i:\widehat{x}]}, \widehat{x}), \widehat{w}, w), & \widehat{x} \leq x, \widehat{w} \leq w, \end{cases}$$

$$d_i^{\mathrm{L}}(x, \widehat{x}, w, \widehat{w}) = \begin{cases} \mathsf{F}_i(x, \widehat{x}, \underline{\mathsf{H}}(x, \widehat{x}_{[i:x]}), \overline{\mathsf{H}}(x, \widehat{x}_{[i:x]}), w, \widehat{w}) & x \leq \widehat{x}, w \leq \widehat{w} \\ \mathsf{F}_i(\widehat{x}, x, \overline{\mathsf{H}}(x_{[i:\widehat{x}]}, \widehat{x}), \underline{\mathsf{H}}(x_{[i:\widehat{x}]}, \widehat{x}), \widehat{w}, w) & \widehat{x} \leq x, \widehat{w} \leq w. \end{cases}$$

For every $s \in \{\mathrm{G}, \mathrm{L}, \mathrm{H}\}$, the trajectory of the embedding system (8) with $d = d^s$ and disturbance $\left[\begin{smallmatrix} w \\ \widehat{w} \end{smallmatrix}\right] = \left[\begin{smallmatrix} \underline{w} \\ \overline{w} \end{smallmatrix}\right]$ starting from $\left[\begin{smallmatrix} \underline{x}_0 \\ \overline{x}_0 \end{smallmatrix}\right]$ is denoted by $t \mapsto \left[\begin{smallmatrix} \underline{x}^s(t) \\ \overline{x}^s(t) \end{smallmatrix}\right]$.

Theorem 3 (Interval over-approximation of reachable sets) *Consider the control system (1) with the neural network controller (2). Suppose that the initial condition x_0 belongs to $\mathcal{X}_0 \subseteq [\underline{x}_0, \overline{x}_0]$ and the disturbance w belongs to $\mathcal{W} \subseteq [\underline{w}, \overline{w}]$. Let $\mathsf{F} : \mathcal{T}^{2n} \times \mathcal{T}^{2p} \times \mathcal{T}^{2q} \to \mathbb{R}^n$ be a decomposition function for the open-loop system (1). Then the following statement holds:*

(i) for every $s \in \{\mathrm{G}, \mathrm{L}, \mathrm{H}\}$ and every $t \in \mathbb{R}_{\geq 0}$, we have

$$\mathcal{R}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}^s(t), \overline{x}^s(t)].$$

If all the activation functions of the neural network $u = \mathsf{N}(x)$ are ReLU, then

(ii) for every $t \in \mathbb{R}_{\geq 0}$, we have

$$\mathcal{R}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}^{\mathrm{L}}(t), \overline{x}^{\mathrm{L}}(t)] \subseteq [\underline{x}^{\mathrm{H}}(t), \overline{x}^{\mathrm{H}}(t)] \subseteq [\underline{x}^{\mathrm{G}}(t), \overline{x}^{\mathrm{G}}(t)].$$

Proof Regarding part (i), we provide the proof for $s = \mathrm{H}$. The proof of other cases are mutadis mutandis with $s = \mathrm{H}$. Note that, by (Abate et al., 2021, Theorem 1), the tight decomposition function d^c for the closed-loop system f^{cl} and the tight decomposition function d^o for the open loop system f can be computed as follows:

$$d^c_i(x,\widehat{x},w,\widehat{w}) = \begin{cases} \min_{\substack{z\in[x,\widehat{x}],\xi\in[w,\widehat{w}]\\ z_i=x_i}} f_i(z,\mathsf{N}(z),\xi), & x\le\widehat{x}, w\le\widehat{w} \\ \max_{\substack{z\in[\widehat{x},x],\xi\in[w,\widehat{w}]\\ z_i=\widehat{x}_i}} f_i(z,\mathsf{N}(z),\xi), & \widehat{x}\le x, \widehat{w}\le w. \end{cases} \tag{9}$$

$$d^o_i(x,\widehat{x},u,\widehat{u},w,\widehat{w}) = \begin{cases} \min_{\substack{z\in[x,\widehat{x}],\xi\in[w,\widehat{w}]\\ z_i=x_i,\eta\in[u,\widehat{u}]}} f_i(z,\eta,\xi), & x\le\widehat{x}, u\le\widehat{u}, w\le\widehat{w}, \\ \max_{\substack{z\in[\widehat{x},x],\xi\in[w,\widehat{w}]\\ z_i=\widehat{x}_i,\eta\in[u,\widehat{u}]}} f_i(z,\eta,\xi), & \widehat{x}\le x, \widehat{u}\le u, \widehat{w}\le w. \end{cases} \tag{10}$$

The trajectory of the embedding system (8) with $d = d^{\mathrm{c}}$ and disturbance $\left[\begin{smallmatrix} w\\ \widehat{w}\end{smallmatrix}\right] = \left[\begin{smallmatrix} \underline{w}\\ \overline{w}\end{smallmatrix}\right]$ starting from $\left[\begin{smallmatrix} \underline{x}_0\\ \overline{x}_0\end{smallmatrix}\right]$ is denoted by $t\mapsto \begin{bmatrix} \underline{x}^c(t)\\ \overline{x}^c(t)\end{bmatrix}$. Since F is a decomposition function for the open-loop system f, for every $x\le\widehat{x}$, $u\le\widehat{u}$, $w\le\widehat{w}$, and $i\in\{1,\dots,n\}$, we get

$$\mathsf{F}_i(x,\widehat{x},u,\widehat{u},w,\widehat{w}) \le d^o_i(x,\widehat{x},u,\widehat{u},w,\widehat{w}) = \min_{\substack{z\in[x,\widehat{x}],z_i=x_i\\ \xi\in[w,\widehat{w}],\eta\in[u,\widehat{u}]}} f_i(z,\eta,\xi). \tag{11}$$

Suppose that $x\le\widehat{x}$ and let $i\in\{1,\dots,n\}$ and $y\in[x,\widehat{x}]$ be such that $y_i = x_i$. By Theorem 1(i), $\begin{bmatrix} \underline{\mathsf{G}}_{[x,\widehat{x}]}\\ \overline{\mathsf{G}}_{[x,\widehat{x}]}\end{bmatrix}$ is an inclusion function for N on $[x,\widehat{x}]$. Moreover, $y\in[x,\widehat{x}_{[i:x]}]\subseteq[x,\widehat{x}]$ and thus

$$\underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}) \le \mathsf{N}(y) \le \overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}). \tag{12}$$

Therefore, for every $i\in\{1,\dots,n\}$, we get

$$\begin{aligned} d^c_i(x,\widehat{x},w,\widehat{w}) &= \min_{\substack{z\in[x,\widehat{x}],\xi\in[w,\widehat{w}]\\ z_i=x_i}} f_i(z,\mathsf{N}(z),\xi) \ge \min_{\substack{z\in[x,\widehat{x}],\xi\in[w,\widehat{w}],z_i=x_i\\ u\in[\underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}),\overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]})]}} f_i(z,u,\xi) \\ &\ge \mathsf{F}_i(x,\widehat{x},\underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}),\overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}),w,\widehat{w}) = d^{\mathrm{H}}_i(x,\widehat{x},w,\widehat{w}). \end{aligned} \tag{13}$$

where the first equality holds by definition of d^c, the second inequality holds by equation (12), the third inequality holds by equation (11), and the fourth inequality holds by definition of d^{H}. Similarly, one can show that $d^c_i(\widehat{x},x,\widehat{w},w) \le d^{\mathrm{H}}_i(\widehat{x},x,\widehat{w},w)$, for every $i\in\{1,\dots,n\}$. This implies that $\begin{bmatrix} d^{\mathrm{H}}(x,\widehat{x},w,\widehat{w})\\ d^{\mathrm{H}}(\widehat{x},x,\widehat{w},w)\end{bmatrix} \le_{\mathrm{SE}} \begin{bmatrix} d^c(x,\widehat{x},w,\widehat{w})\\ d^c(\widehat{x},x,\widehat{w},w)\end{bmatrix}$, for every $x\le\widehat{x}$ and every $w\le\widehat{w}$. Note that, by (Abate et al., 2021, Theorem 1), the vector field $\begin{bmatrix} d^c(x,\widehat{x},w,\widehat{w})\\ d^c(\widehat{x},x,\widehat{w},w)\end{bmatrix}$ is monotone with respect to the southeast order $\le_{\mathrm{SE}}$ on $\mathbb{R}^{2n}$. Now, we can use (Michel et al., 2008, Theorem 3.8.1), to deduce that $\begin{bmatrix} \underline{x}^{\mathrm{H}}(t)\\ \overline{x}^{\mathrm{H}}(t)\end{bmatrix} \le \begin{bmatrix} \underline{x}^c(t)\\ \overline{x}^c(t)\end{bmatrix}$, for every $t\in\mathbb{R}_{\ge0}$. This implies that $[\underline{x}^c(t),\overline{x}^c(t)]\subseteq[\underline{x}^{\mathrm{H}}(t),\overline{x}^{\mathrm{H}}(t)]$. On the other hand, by (Abate et al., 2021, Theorem 2), we know that $\mathcal{R}(t,\mathcal{X}_0,\mathcal{W})\subseteq[\underline{x}^c(t),\overline{x}^c(t)]$, for every $t\in\mathbb{R}_{\ge0}$. This lead to $\mathcal{R}(t,\mathcal{X}_0,\mathcal{W})\subseteq[\underline{x}^{\mathrm{H}}(t),\overline{x}^{\mathrm{H}}(t)]$, for every $t\in\mathbb{R}_{\ge0}$.

Regarding part (ii), suppose that all the activation functions are ReLU and let $L^{(i)}_{[x,\widehat{x}]}$ and $U^{(i)}_{[x,\widehat{x}]}$ are the intermediate bounds associated to the input perturbation $[x,\widehat{x}]$. Then, for every $\left[\begin{smallmatrix} x\\ \widehat{x}\end{smallmatrix}\right] \le_{\mathrm{SE}} \left[\begin{smallmatrix} y\\ \widehat{y}\end{smallmatrix}\right] \in \mathcal{T}^{2n}_{\ge0}$, the intermediate bounds in CROWN satisfy:

$$L^{(i)}_{[x,\widehat{x}]} \le L^{(i)}_{[y,\widehat{y}]}, \qquad U^{(i)}_{[y,\widehat{y}]} \le U^{(i)}_{[x,\widehat{x}]}, \qquad \text{for every } i\in\{1,\dots,k\}.$$

Since all the activation functions are ReLU, we can use (Zhang et al., 2018, Table 1 and Table 2) to show that

$$\begin{aligned} &\alpha^{(i)}_{L,[x,\widehat{x}]} \leq \alpha^{(i)}_{L,[y,\widehat{y}]}, \qquad \alpha^{(i)}_{U,[y,\widehat{y}]} \leq \alpha^{(i)}_{U,[x,\widehat{x}]}, \\ &\beta^{(i)}_{L,[x,\widehat{x}]} \leq \beta^{(i)}_{L,[y,\widehat{y}]}, \qquad \beta^{(i)}_{U,[y,\widehat{y}]} \leq \beta^{(i)}_{U,[x,\widehat{x}]}, \qquad \text{for every } i \in \{1, \dots, k\} \end{aligned}$$

Therefore, using the equations (4) and (6), and the formula in (Zhang et al., 2018, Theorem 3.2) for $\Lambda^{(i)}, \Delta^{(i)}, \Omega^{(i)}, \Theta^{(i)}$, we get

$$\begin{aligned} \underline{A}(x,\widehat{x}) \leq \underline{A}(y,\widehat{y}), \qquad \overline{A}(y,\widehat{y}) \leq \overline{A}(x,\widehat{x}), \\ \underline{b}(x,\widehat{x}) \leq \underline{b}(y,\widehat{y}), \qquad \overline{b}(y,\widehat{y}) \leq \overline{b}(x,\widehat{x}), \end{aligned}$$

This implies that

$$\begin{aligned} \underline{\mathsf{G}}_{[x,\widehat{x}]}(w,\widehat{w}) &= [\underline{A}(x,\widehat{x})]^{+}w + [\underline{A}(x,\widehat{x})]^{-}\widehat{w} \leq [\underline{A}(y,\widehat{y})]^{+}w + [\underline{A}(y,\widehat{y})]^{-}\widehat{w} \\ &\leq [\underline{A}(y,\widehat{y})]^{+}v + [\underline{A}(y,\widehat{y})]^{-}\widehat{v} = \underline{\mathsf{G}}_{[y,\widehat{y}]}(v,\widehat{v}). \end{aligned}$$

Similarly, we can show that $\overline{\mathsf{G}}_{[y,\widehat{y}]}(v,\widehat{v}) \leq \overline{\mathsf{G}}_{[x,\widehat{x}]}(w,\widehat{w})$. As a result, for every $\left[\begin{smallmatrix} x \\ \widehat{x} \end{smallmatrix}\right] \leq_{\mathrm{SE}} \left[\begin{smallmatrix} y \\ \widehat{y} \end{smallmatrix}\right] \in \mathcal{T}^{2n}_{\geq 0}$ and $\left[\begin{smallmatrix} w \\ \widehat{w} \end{smallmatrix}\right] \leq_{\mathrm{SE}} \left[\begin{smallmatrix} v \\ \widehat{v} \end{smallmatrix}\right] \in \mathcal{T}^{2n}_{\geq 0}$,

$$\begin{bmatrix} \underline{\mathsf{G}}_{[x,\widehat{x}]}(w,\widehat{w}) \\ \overline{\mathsf{G}}_{[x,\widehat{x}]}(w,\widehat{w}) \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{\mathsf{G}}_{[y,\widehat{y}]}(v,\widehat{v}) \\ \overline{\mathsf{G}}_{[y,\widehat{y}]}(v,\widehat{v}) \end{bmatrix} \tag{14}$$

Note that, using property (14) and definition of decomposition function, one can easily check that $d^{\mathrm{G}}, d^{\mathrm{L}}, d^{\mathrm{H}}$ are decomposition functions for f^{cl} and thus, they are monotone with respect to the southeast order $\leq_{\mathrm{SE}}$ on $\mathbb{R}^{2n}$. On the other hand, we have $\widehat{x}_{[i:x]} \leq \widehat{x}$. Therefore, using (14),

$$\begin{bmatrix} \underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{i:x}) \\ \overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}) \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}) \\ \overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}) \end{bmatrix}, \qquad \text{for every } x \leq \widehat{x} \tag{15}$$

This implies that, for every $x \leq \widehat{x}$ and every $w \leq \widehat{w}$,

$$\begin{aligned} d^{\mathrm{H}}(x,\widehat{x},w,\widehat{w}) &= \mathsf{F}_i(x,\widehat{x},\underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}),\overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}_{[i:x]}),w,\widehat{w}) \\ &\leq \mathsf{F}_i(x,\widehat{x},\underline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}),\overline{\mathsf{G}}_{[x,\widehat{x}]}(x,\widehat{x}),w,\widehat{w}) = d^{\mathrm{G}}(x,\widehat{x},w,\widehat{w}) \end{aligned}$$

where the inequality holds using equation 15 and the fact that F is a decomposition function for f^{cl}. Similarly, we can show that $d^{\mathrm{H}}(\widehat{x},x,\widehat{w},w) \leq d^{\mathrm{G}}(\widehat{x},x,\widehat{w},w)$, for every $x \leq \widehat{x}$ and every $w \leq \widehat{w}$. This implies that $\left[\begin{smallmatrix} d^{\mathrm{G}}(x,\widehat{x},w,\widehat{w}) \\ d^{\mathrm{G}}(\widehat{x},x,\widehat{w},w) \end{smallmatrix}\right] \leq_{\mathrm{SE}} \left[\begin{smallmatrix} d^{\mathrm{H}}(x,\widehat{x},w,\widehat{w}) \\ d^{\mathrm{H}}(\widehat{x},x,\widehat{w},w) \end{smallmatrix}\right]$, for every $x \leq \widehat{x}$ and every $w \leq \widehat{w}$. Note that d^{G} is a decomposition function for f^{cl} and thus the vector field $\left[\begin{smallmatrix} d^{G}(x,\widehat{x},w,\widehat{w}) \\ d^{G}(\widehat{x},x,\widehat{w},w) \end{smallmatrix}\right]$ is monotone with respect to the southeast order $\leq_{\mathrm{SE}}$ on $\mathbb{R}^{2n}$. Now, we can use (Michel et al., 2008, Theorem 3.8.1), to deduce that $\left[\begin{smallmatrix} \underline{x}^{\mathrm{G}}(t) \\ \overline{x}^{\mathrm{G}}(t) \end{smallmatrix}\right] \leq \left[\begin{smallmatrix} \underline{x}^{\mathrm{H}}(t) \\ \overline{x}^{\mathrm{H}}(t) \end{smallmatrix}\right]$, for every $t \in \mathbb{R}_{\geq 0}$. This implies that $[\underline{x}^{\mathrm{H}}(t), \overline{x}^{\mathrm{H}}(t)] \subseteq [\underline{x}^{\mathrm{G}}(t), \overline{x}^{\mathrm{G}}(t)]$. The proofs for the other inclusions are similar and we remove it for the sake of brevity. ■

For the special case when the open-loop system (1) is linear in state and affine in control, one can find closed-from expressions for d^{G}, d^{H}, and d^{L}.

Corollary 4 (Linear systems) *Consider the control system (1) with $f(x,u,w) = Ax + Bu + Cw$ with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{n\times q}$ and with the neural network controller (2). Then we recover all the results of Theorem (3) with the following "global", "hybrid", and "local" functions:*

$$d^{\mathrm{G}}(x,\widehat{x},w,\widehat{w}) = \Big(\lceil A\rceil^{\mathrm{Mzl}} + [R(x,\widehat{x})]^{+}\Big)\,x + \Big(\lfloor A\rfloor^{\mathrm{Mzl}} + [S(x,\widehat{x})]^{-}\Big)\,\widehat{x} + C^{+}w + C^{-}\widehat{w}$$

$$d^{\mathrm{H}}(x,\widehat{x},w,\widehat{w}) = \lceil A + R(x,\widehat{x})\rceil^{\mathrm{Mzl}}x + \lfloor A + S(x,\widehat{x})\rfloor^{\mathrm{Mzl}}\widehat{x} + C^{+}w + C^{-}\widehat{w}$$

$$d^{\mathrm{L}}_i(x,\widehat{x},w,\widehat{w}) = \lceil A + R(x,\widehat{x}_{[i:x]})\rceil^{\mathrm{Mzl}}x + \lfloor A + S(x,\widehat{x}_{[i:x]})\rfloor^{\mathrm{Mzl}}\widehat{x} + C^{+}w + C^{-}\widehat{w}, \quad \forall i \in \{1,\dots,n\},$$

where $R(x,\widehat{x}) = B^{+}[\underline{A}(x,\widehat{x})] + B^{-}[\overline{A}(x,\widehat{x})]$ and $S(x,\widehat{x}) = B^{+}[\overline{A}(x,\widehat{x})] + B^{-}[\underline{A}(x,\widehat{x})]$.

Proof Note that a decomposition function for the open-loop system is given by $\mathsf{F}(x,\widehat{x},u,\widehat{u},w,\widehat{w}) = \lceil A\rceil^{\mathrm{Mzl}}x + \lfloor A\rfloor^{\mathrm{Mzl}}\widehat{x} + B^{+}u + B^{-}\widehat{u} + C^{+}w + C^{-}\widehat{w}$ (Coogan, 2020, Example 3). Using simple algebraic manipulations, one can show that for every $i \in \{1,\dots,n\}$ and every $M \in \mathbb{R}^{n\times n}$,

$$\big(M^{+}x + M^{-}\widehat{x}_{[i:x]}\big)_i = \Big(\lceil M\rceil^{\mathrm{Mzl}}x + \lfloor M\rfloor^{\mathrm{Mzl}}\widehat{x}\Big)_i. \tag{16}$$

Thus, using Theorem 3 and the identity (16), for every $i \in \{1,\dots,n\}$, we have

$$\begin{aligned}
d^{\mathrm{H}}_i(x,\widehat{x},w,\widehat{w}) &= \Big(\lceil A\rceil^{\mathrm{Mzl}}x + \lfloor A\rfloor^{\mathrm{Mzl}}\widehat{x} + B^{+}[\underline{A}]^{+}(x,\widehat{x})x + B^{+}[\underline{A}]^{-}(x,\widehat{x})\widehat{x}_{[i:x]}\Big)_i \\
&\quad + \big(B^{-}[\overline{A}]^{+}(x,\widehat{x})x + B^{-}[\overline{A}]^{-}(x,\widehat{x})\widehat{x}_{[i:x]} + C^{+}w + C^{-}\widehat{w}\big)_i \\
&= \Big(\lceil A\rceil^{\mathrm{Mzl}}x + \lfloor A\rfloor^{\mathrm{Mzl}}\widehat{x} + \lceil B^{+}\underline{A}(x,\widehat{x})\rceil^{\mathrm{Mzl}}x + \lfloor B^{+}\underline{A}(x,\widehat{x})\rfloor^{\mathrm{Mzl}}\widehat{x}\Big)_i \\
&\quad + \Big(\lfloor B^{-}\overline{A}(x,\widehat{x})\rfloor^{\mathrm{Mzl}}x + \lceil B^{-}\overline{A}(x,\widehat{x})\rceil^{\mathrm{Mzl}}\widehat{x} + C^{+}w + C^{-}\widehat{w}\Big)_i \\
&= \Big(\lceil A + B^{+}\underline{A}(x,\widehat{x}) + B^{-}\overline{A}(x,\widehat{x})\rceil^{\mathrm{Mzl}} + \lfloor A + B^{+}[\overline{A}(x,\widehat{x})] + B^{-}[\underline{A}(x,\widehat{x})]\rfloor^{\mathrm{Mzl}}\Big)_i \\
&\quad + \big(C^{+}w + C^{-}\widehat{w}\big)_i \\
&= \Big(\lceil A + R(x,\widehat{x})\rceil^{\mathrm{Mzl}}x + \lfloor A + S(x,\widehat{x})\rfloor^{\mathrm{Mzl}}\widehat{x} + C^{+}w + C^{-}\widehat{w}\Big)_i.
\end{aligned}$$

This completes the proof for $s = \mathrm{H}$. The proofs of the other cases follow similarly. ■

Remark 5 *The following remarks are in order.*

(i) ***(Tight decomposition function)*** *One can use the tight decomposition function of the closed loop system (3) in the embedding system (8) to obtain hyper-rectangular over-approximations for reachable set of the system. However, finding the tight decomposition function requires solving a computationally complicated optimization problem and generally difficult. In this context, Theorem 3 uses the decomposition function of the open-loop system (1) and the inclusion function of the neural network to establish three under-approximation the tight decomposition function of the closed-loop system (8).*

(ii) ***(Comparison with the literature)*** *For linear systems, Corollary 4 can be used to show that the forward Euler integration of the embedding system (8) with $d = d^{\mathrm{H}}$ and with a small enough time-step will lead to the identical over-approximation sets as (Everett et al., 2021b, Lemma IV.3) applied to the forward Euler discretization of the linear system.*

(iii) ***(Generality of the approach)*** *Theorem 3 proposes a general embedding-based framework for verification of the closed-loop system* (3) *which is based on combining mixed monotone reachability of the open-loop system with interval analysis of the neural network. Using this perspective, our approach can be applied to arbitrary neural network verification algorithms as long as one can construct an inclusion function for the neural network. This is in contrast with most of the existing approaches for neural network closed-loop reachability analysis, which are heavily dependent on the neural network verification algorithm (see for instance (Everett et al., 2021b) and (Sidrane et al., 2022)).*

(iv) ***(Computational complexity)*** *From a computational perspective, the framework presented in Theorem 3 consists of two main ingredients: (i) evaluating CROWN to compute the inclusion function of the neural network as in Theorem 1, and (ii) integrating the embedding dynamical system* (8)*. For a neural network with k-layer and N neuron per layer, the complexity of CROWN is $\mathcal{O}(k^2N^3)$ (Zhang et al., 2018). Moreover, the functions d^{G} and d^{H} call CROWN once per integration step, while the function d^{L} calls CROWN n times per integration step. The run time of the integration process depends on the form of the open-loop decomposition function* F.

6. Efficient reachability analysis via partitioning

In this section, we develop a suitable partitioning of the state space and combine it with Theorem (3) to obtain a computationally efficient algorithm for generating reachable set over-approximations of the closed-loop system. Interval methods are known to suffer from large over-approximation error due to the wrapping effect (Jaulin et al., 2001, Section 2.2.4). However, their ease of computation allows for combining them efficiently with partitioning strategies. Our partitioning strategy consists of two main components: (i) a uniform division of the state space of the embedding system (8) to compute the neural network inclusion functions using CROWN, and (ii) a uniform subdivision to implement the integration on the embedding system (8). We first pick an $s \in \{\mathrm{G}, \mathrm{H}, \mathrm{L}\}$ and start with the initial perturbation set $\mathcal{X}_0$. In the first step, we find the smallest hyper-rectangle containing $\mathcal{X}_0$ by computing $(\overline{x}_0)_j = \max_{x \in \mathcal{X}_0} x_j$ and $(\underline{x}_0)_j = \min_{x \in \mathcal{X}_0} x_j$, for every $i \in \{1, \ldots, n\}$. We then divide the hyper-rectangle $[\underline{x}_0, \overline{x}_0]$ into D_a partitions and obtain the set $\{[\underline{x}^1, \overline{x}^1], \ldots, [\underline{x}^{D_a}, \overline{x}^{D_a}]\}$. For every $k \in \{1, \ldots, D_a\}$, we compute the trajectory of the embedding system (8) with $d = d^{\mathrm{s}}$ and the initial condition $\begin{bmatrix} \underline{x}^k \\ \overline{x}^k \end{bmatrix}$ at time Δt as in Theorem 3. For every $k \in \{1, \ldots, D_a\}$, we divide each states of the hyper-rectangle $[\underline{x}^1, \overline{x}^1]$ into D_s partitions, obtaining the subpartitions $\{[\underline{x}^{k,1}, \overline{x}^{k,1}], \ldots, [\underline{x}^{k,D_s}, \overline{x}^{k,D_s}]\}$. For every $k \in \{1, \ldots, D_a\}$ and $l \in \{1, \ldots, D_s\}$, we compute the trajectory of the embedding system (8) with $d = d^{\mathrm{Bs}}$, where, for every $i \in \{1, \ldots, n\}$,

$$d_i^{\mathrm{Bs}}(x, \widehat{x}, w, \widehat{w}) = \mathsf{F}_i(x, \widehat{x}, \underline{\mathsf{G}}_{[\underline{x}^k, \overline{x}^k]}(x, \widehat{x}_{[i:x]}), \overline{\mathsf{G}}_{[\underline{x}^k, \overline{x}^k]}(\widehat{x}_{[i:x]}, x), w, \widehat{w}) \tag{17}$$

and the initial condition $\begin{bmatrix} \underline{x}^{k,l} \\ \overline{x}^{k,l} \end{bmatrix}$ at time Δt as in Theorem 3. We then set $\mathcal{X}_1 = \bigcup_{k=1}^{D_a} \bigcup_{l=1}^{D_s} [\underline{x}^{l,k}(\Delta t), \overline{x}^{l,k}(\Delta t)]$ and repeat this procedure on the initial set $\mathcal{X}_1$. We keep repeating this algorithm until we get to the final time T. Note that our partitioning approach is different from (Everett et al., 2021a) and (Xiang et al., 2021) in that we re-partitioning the state-space at every time step. A summary of the above procedure is presented in Algorithm 1.

Algorithm 1 Over-approximation of reachable sets of (3)

```
Input: $s \in \{\mathrm{G}, \mathrm{H}, \mathrm{L}\}$, the initial set $\mathcal{X}_0$, the final time $T$, the actuation step $\Delta t$, divisions $D_a$, subdivision $D_s$
Output: the over-approximation of the reachable set $\overline{\mathcal{R}}(T, 0, \mathcal{X}_0, \mathcal{W})$
$j \leftarrow 0$
while $j < \lfloor \frac{T}{\Delta t} \rfloor$ do
    $\underline{x}_i \leftarrow \min_{x \in \mathcal{X}_j} x_i$, for every $i \in \{1, \dots, n\}$
    $\overline{x}_i \leftarrow \max_{x \in \mathcal{X}_j} x_i$, for every $i \in \{1, \dots, n\}$
    $\{[\underline{x}^1, \overline{x}^1], \dots, [\underline{x}^{D_a}, \overline{x}^{D_a}]\} \leftarrow \text{uniform\_partition}([\underline{x}, \overline{x}], D_a)$
    for $k = \{1, \dots, D_a\}$ do
        Compute $\underline{A}(\underline{x}^k, \overline{x}^k)$, $\overline{A}(\underline{x}^k, \overline{x}^i)$, $\underline{b}(\underline{x}^k, \overline{x}^k)$, and $\overline{b}(\underline{x}^k, \overline{x}^k)$ using CROWN and (6).
        $\{[\underline{x}^{k,1}, \overline{x}^{k,1}], \dots, [\underline{x}^{k,D_s}, \overline{x}^{k,D_s}]\} \leftarrow \text{uniform\_partition}([\underline{x}^k, \overline{x}^k], D_s)$
        for $l = \{1, \dots, D_s\}$ do
            Compute $\begin{bmatrix} \underline{x}^{k,l}(\Delta t) \\ \overline{x}^{k,l}(\Delta t) \end{bmatrix}$ for system (8) with $d = d^{\mathrm{Bs}}$ and initial condition $\begin{bmatrix} \underline{x}^{k,l} \\ \overline{x}^{k,l} \end{bmatrix}$.
        end for
    end for
    $\mathcal{X}_{j+1} = \bigcup_{k=1}^{D_a} \bigcup_{l=1}^{D_s} [\underline{x}^{l,k}(\Delta t), \overline{x}^{l,k}(\Delta t)]$
end while
return $\overline{\mathcal{R}}(T, 0, \mathcal{X}_0, \mathcal{W}) \leftarrow \mathcal{X}_{\lfloor \frac{T}{\Delta t} \rfloor}$
```

7. Numerical Simulations

In this section, we show the efficiency of our reachability analysis using numerical experiments on a nonlinear vehicle model and a linear quadrotor model[1].

7.1. Nonlinear Vehicle model

We consider the dynamics of a vehicle adopted from (Polack et al., 2017) satisfying the following nonlinear ordinary differential equation:

$$\dot{p_x} = v\cos(\phi + \beta(u_2)), \quad \dot{\phi} = \frac{v}{\ell_r}\sin(\beta(u_2)), \quad \dot{p_y} = v\sin(\phi + \beta(u_2)), \quad \dot{v} = u_1 + w, \tag{18}$$

where $[p_x, p_y]^\top \in \mathbb{R}^2$ is the displacement of the center of mass, $\phi \in [-\pi, \pi)$ is the heading angle in the plane, $v \in \mathbb{R}^+$ is the speed of the center of mass. Control input u_1 is the applied force subject to disturbance w, input u_2 is the angle of the front wheels, and $\beta(u_2) = \arctan\left(\frac{\ell_f}{\ell_f + \ell_r}\tan(u_2)\right)$ is the slip slide angle. We set $x = [p_x, p_y, \phi, v]^\top$ and $u = [u_1, u_2]^\top$. We use the following tight decomposition function for the open-loop system:

$$\mathsf{F}(x, \widehat{x}, u, \widehat{u}, w, \widehat{w}) = \begin{bmatrix} d^{b_1 b_2}\left([v, d^{\cos}(\phi + \beta(u_2), \widehat{\phi} + \beta(\widehat{u}_2))]^\top, [\widehat{v}, d^{\cos}(\widehat{\phi} + \beta(\widehat{u}_2), \phi + \beta(u_2))]^\top\right) \\ d^{b_1 b_2}\left([v, d^{\sin}(\phi + \beta(u_2), \widehat{\phi} + \beta(\widehat{u}_2))]^\top, [\widehat{v}, d^{\sin}(\widehat{\phi} + \beta(\widehat{u}_2), \phi + \beta(u_2))]^\top\right) \\ d^{b_1 b_2}\left([v, d^{\sin}(\beta(u_2), \beta(\widehat{u}_2))]^\top, [\widehat{v}, d^{\sin}(\beta(\widehat{u}_2), \beta(u_2))]^\top\right) \\ u_1 + w \end{bmatrix},$$

[1] All the code is available at https://github.com/gtfactslab/L4DC2023_NNControllerReachability

where $d^{b_1 b_2}$, $d^{\cos}$, and $d^{\sin}$ are defined in (Cao et al., 2022). We designed an offline nonlinear model predictive controller in Python using Casadi (Andersson et al., 2019) to steer the vehicle to the origin while avoiding obstacles. We use a fixed horizon of 20 with an actuation step of 0.25 seconds, and a quadratic cost function with $Q = \mathrm{diag}(1, 1, 0, 0)$, $Q_{hor} = \mathrm{diag}(100, 100, 0, 1)$, and other regularizing terms tuned empirically. Additionally, we add circular obstacles with 25% padding as hard constraints with slack variables; in Figures 1 and 2, we consider one centered at $(4, 4)$ with a radius of 2.4. We simulated 65000 real trajectories (5s, 20 control actions) with initial conditions uniformly sampled from a specified region, and aggregated the data into a set of $1.3M$ training pairs $(x, u) \in \mathbb{R}^4 \times \mathbb{R}^2$. A neural network $u = \mathsf{N}(x)$ with 2 hidden layers with 100 neurons per layer and ReLU activation was trained in Pytorch to approximate the model predictive controller under a scaled Mean Squared Error loss. We use Algorithm 1 with $D_a = 16$ and $D_s = 1$ to provide over-approximations for the reachable sets of the vehicle model (18), comparing functions d^{G}, d^{L}, and d^{H} from Theorem 3. Algorithm 1 line 7 is computed using auto_LiRPA (Xu et al., 2020). The results are shown in Figure 1.

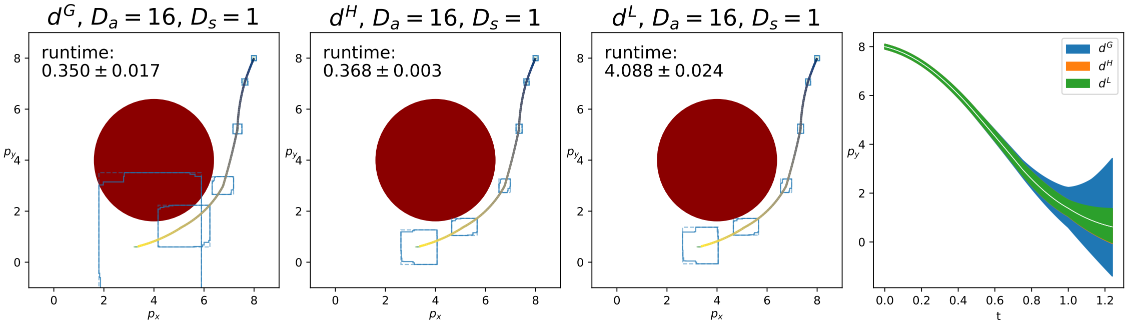

Figure 1: Performance of the three different functions d^{G}, d^{H}, and d^{L} in Theorem (3) for over-approximation of the reachable set of the system (18) with neural network $u = \mathsf{N}(x)$ trained to approximate an offline model predictive controller. The $p_x - p_y$ plot of the motion of the vehicle is shown starting from an initial set $[7.9, 8.1]^2 \times [-\frac{2\pi}{3} - 0.01, -\frac{2\pi}{3} + 0.01] \times [1.99, 2.01]$. In terms of accuracy, the size of the over-approximations obtained using the functions d^{H} and d^{L} are close to each other, but are much smaller than the size of the over-approximations obtained using the function d^{G}. On the other hand, finding the over-approximations using the functions d^{G} and d^{H} take about the same amount of time and are much faster than finding the over-approximations using the functions d^{L}. The runtimes are averaged over 10 instances and mean and standard deviation are reported.

7.2. Linear 6D quadrotor model

We use the linear 6D quadrotor model adopted from (Ivanov et al., 2019) with the dynamics $\dot{x} = Ax + Bu + c$ where A, B, c, u are as define in (Everett et al., 2021b) and $x = [p_x, p_y, p_z, v_x, x_y, v_z]^\top$ is the state of the system with p_x, p_y, and p_y being the linear displacement in the x, y, and z directions, respectively and v_x, v_y, and v_z the velocity in the x, y, and z directions respectively. The neural network controller $u = \mathsf{N}(x)$ consists of 2 layers with 32 neurons in each layer and is identical to (Everett et al., 2021b, Section H). One can compute the following tight decomposition function for the open-loop system $\mathsf{F}(x, \widehat{x}, u, \widehat{u}, w, \widehat{w}) = Ax + B^+u + B^-\widehat{u} + c$. We compare the

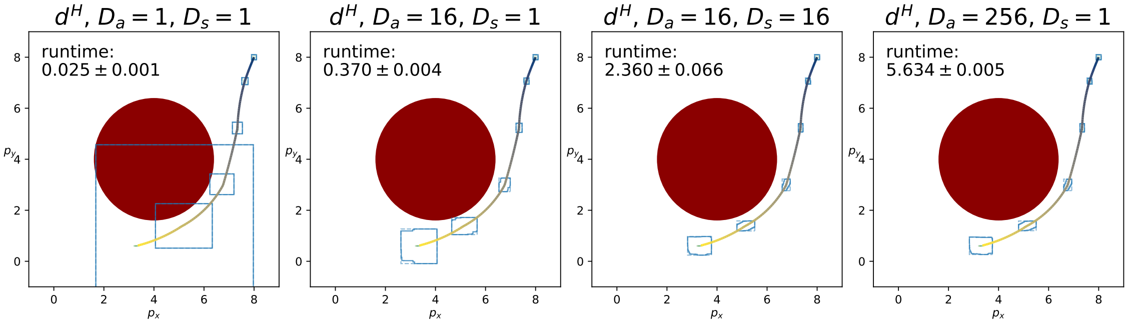

Figure 2: Performance of the Algorithm (1) with $s = \mathrm{H}$ for different partitions D_s and subpartitions D_a for over-approximation of the reachable set of the system (18) with neural network $u = \mathsf{N}(x)$ trained to approximate an offline model predictive controller. All four figures show the $p_x - p_y$ plot of the motion of the vehicle. The runtimes are averaged over 10 instances and the mean and standard deviation are reported.

over-approximation of the reachable sets using different integration techniques for Algorithm 1. The results are shown in Figure 3. Note that, our Algorithm 1 with the forward Euler method yields identical results as the linear programming approach in (Everett et al., 2021b).

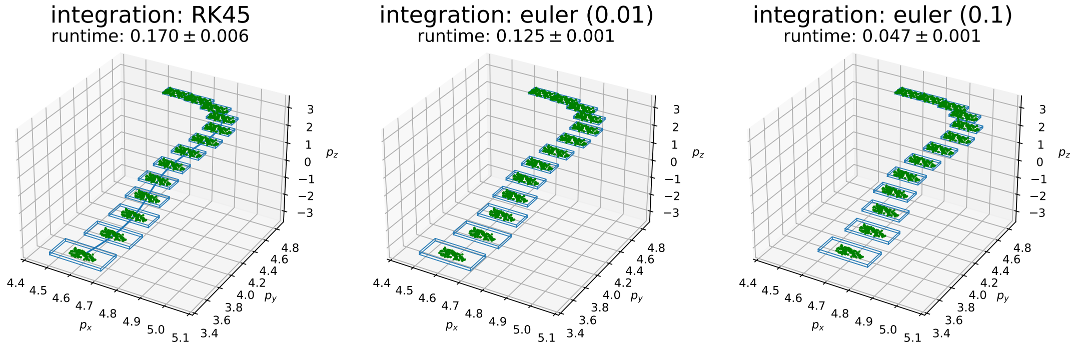

Figure 3: The time evolution of the reachable sets of the 6D quadrotor with the neural network controller $u = \mathsf{N}(x)$ in p_x, p_y, p_z coordinate. The blue hyper-rectangles are the over-approximations of the reachable sets of the system computed using the function d^{H} in Theorem 3. **Left plot**: The ODEs are integrated using a Runge-Kutta method. **Middle plot**: The ODEs are integrated using forward Euler method with time step 0.01. **Right plot**: The ODEs are integrated using forward Euler method with time step 0.1 to match the actuation time step. Note that the reachable set estimates of this plot are identical to those in (Everett et al., 2021b, Figure 10(a)) where the over-approximations are computed using a linear program. We observed that, on the same computer, this linear program takes more than twice as long, 0.113 ± 0.004 seconds. The runtimes are averaged over 10 instances and mean and standard deviation are reported.

8. Conclusions

We presented a fast and scalable method, based on mixed monotone theory, to over-approximate the reachable sets of nonlinear systems coupled with neural network controllers. We introduce three methods to intertwine neural network inclusion functions from CROWN (Zhang et al., 2018) and open-loop decomposition functions with varying empirical results. For future research, we plan to study the role of the neural network verification algorithms in the tightness of our approximations.

References

M. Abate, M. Dutreix, and S. Coogan. Tight decomposition functions for continuous-time mixed-monotone systems with disturbances. *IEEE Control Systems Letters*, 5(1):139–144, 2021. doi:10.1109/LCSYS.2020.3001085.

J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi:10.1007/s12532-018-0139-4.

S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, 2017. doi:10.1109/CDC.2017.8263977.

M. E. Cao, M. Bloch, and S. Coogan. Efficient learning of hyperrectangular invariant sets using gaussian processes. *IEEE Open Journal of Control Systems*, 1:223–236, 2022. doi:10.1109/OJCSYS.2022.3206083.

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)*, pages 1520–1527, 2018. doi:10.23919/ACC.2018.8431275.

J. Cohen, E. Rosenfeld, and J. Z. Kolter. Certified adversarial robustness via randomized smoothing. pages 1310–1320, 2019. URL `https://arxiv.org/abs/1902.02918`.

S. Coogan. Mixed monotonicity for reachability and safety in dynamical systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5074–5085, 2020. doi:10.1109/CDC42340.2020.9304391.

S. Coogan and M. Arcak. Efficient finite abstraction of mixed monotone systems. In *Hybrid Systems: Computation and Control*, pages 58–67, April 2015. doi:10.1145/2728606.2728607.

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*, page 157–168. Association for Computing Machinery, 2019. ISBN 9781450362825. doi:10.1145/3302504.3311807.

M. Everett, G. Habibi, and J. P. How. Robustness analysis of neural networks via efficient partitioning with applications in control systems. *IEEE Control Systems Letters*, 5(6):2114–2119, 2021a. doi:10.1109/LCSYS.2020.3045323.

M. Everett, G. Habibi, C. Sun, and J. P. How. Reachability analysis of neural feedback loops. *IEEE Access*, 9:163938–163953, 2021b. doi:10.1109/ACCESS.2021.3133370.

M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas. Efficient and accurate estimation of Lipschitz constants for deep neural networks. 2019. URL https://arxiv.org/abs/1906.04893.

A. Girard. Reachability of uncertain linear systems using zonotopes. In *Proceedings of the 8th International Conference on Hybrid Systems: Computation and Control*, HSCC'05, page 291–305, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/978-3-540-31954-2_19.

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

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. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5929–5934, 2020. doi:10.1109/CDC42340.2020.9304296.

R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, page 169–178, 2019. doi:10.1145/3302504.3311806.

L. Jaulin, M. Kieffer, O. Didrit, and É. Walter. *Applied Interval Analysis*. Springer London, 2001. doi:10.1007/978-1-4471-0249-6.

G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification*, pages 97–117, Cham, 2017. Springer International Publishing. URL https://arxiv.org/abs/1702.01135.

A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In Nancy Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control*, pages 202–214, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. doi:10.1007/3-540-46430-1_19.

B. Li, C. Chen, W. Wang, and L. Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1809.03113.

C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M. J. Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404, 2021.

A. N. Michel, L. Hou, and D. Liu. *Stability of Dynamical Systems: Continuous, Discontinuous, and Discrete Systems*. Stability of Dynamical Systems: Continuous, Discontinuous, and Discrete Systems. Birkhäuser Boston, 2008. URL https://books.google.com/books?id=s4mq1h0elUkC.

M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3578–3586. PMLR, 10-15 Jul 2018. URL https://proceedings.mlr.press/v80/mirman18b.html.

I. Mitchell and C. J. Tomlin. Level set methods for computation in hybrid systems. In *Hybrid Systems: Computation and Control*, pages 310–323, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. doi:10.1007/3-540-46430-1_27.

P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017. doi:10.1109/IVS.2017.7995816.

C. Sidrane, A. Maleki, A. Irfan, and M. J. Kochenderfer. Overt: An algorithm for safety verification of neural network control policies for nonlinear systems. *Journal of Machine Learning Research*, 23(117):1–45, 2022.

X. Sun, H. Khedr, and Y. Shoukry. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, page 147–156. Association for Computing Machinery, 2019. doi:10.1145/3302504.3311802.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Efficient formal safety analysis of neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6369–6379, 2018. URL https://arxiv.org/abs/1809.08098.

E. Wong and J. Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295, 2018. URL http://proceedings.mlr.press/v80/wong18a.html.

W. Xiang, H-D. Tran, X. Yang, and T. T. Johnson. Reachable set estimation for neural network control systems: A simulation-guided approach. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1821–1830, 2021. doi:10.1109/TNNLS.2020.2991090.

K. Xu, Z. Shi, H. Zhang, Y. Wang, K-W Chang, M. Huang, B. Kailkhura, X. Lin, and C-J. Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.

H. Zhang, T-W. Weng, P-Y. Chen, C-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, volume 31, page 4944–4953, 2018. URL https://proceedings.neurips.cc/paper/2018/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf.

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)*, page 528–535, 2016. doi:10.1109/ICRA.2016.7487175.