

# Interval Reachability of Nonlinear Dynamical Systems with Neural Network Controllers

**Saber Jafarpour\***

*Georgia Institute of Technology*

SABER@GATECH.EDU

**Akash Harapanahalli\***

*Georgia Institute of Technology*

AHARAPAN@GATECH.EDU

**Samuel Coogan**

*Georgia Institute of Technology*

SAM.COOGAN@GATECH.EDU

**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

**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., 2019; 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). Recently, several computationally tractable approaches including the ellipsoidal method (Kurzanski and Varaiya, 2000), the zonotope method (Girard, 2005), the mixed monotone reachability approach (Coogan and Arcak, 2015) have been developed for reachability analysis.

It is well-known that, for a closed-loop system, a direct combination of state-of-the-art neural network verification algorithms with the existing reachability analysis toolboxes can lead to overly-conservative estimates of reachable sets (Dutta et al., 2019, Section 2.1). For linear systems with neural network controllers, reachable set over-approximation has been studied using semi-definite programming (Hu et al., 2020) and linear programming (Everett et al., 2021b). Branch-and-Bound (Entesari et al., 2022) and suitable partitioning (Everett et al., 2021a; Xiang et al., 2021) approaches have also been used to improve the accuracy of reachable set estimations. For nonlinear systems with neural network controllers, Sidrane et al. (2022) establishes a mixed integer programming framework for reachable set over-approximation using polynomial bounds of the system dynamics. 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 structured zonotopes integrated into Taylor models (Schilling et al., 2022).

**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<sup>1</sup>. 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.

---

1. Due to the space constraints, full proof of the results of this paper is available in the following arXiv document <https://arxiv.org/pdf/2301.07912.pdf>.

## 2. Notation and Mathematical Preliminaries

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  $\leq_{\text{SE}}$  on  $\mathbb{R}^{2n}$  defined by  $\begin{bmatrix} x \\ \hat{x} \end{bmatrix} \leq_{\text{SE}} \begin{bmatrix} y \\ \hat{y} \end{bmatrix}$  if  $x \leq y$  and  $\hat{y} \leq \hat{x}$ . We define the subsets  $\mathcal{T}_{\geq 0}^{2n} = \{\begin{bmatrix} x \\ \hat{x} \end{bmatrix} \in \mathbb{R}^{2n} \mid x \leq \hat{x}\}$  and  $\mathcal{T}_{\leq 0}^{2n} = \{\begin{bmatrix} x \\ \hat{x} \end{bmatrix} \in \mathbb{R}^{2n} \mid x \geq \hat{x}\}$  and  $\mathcal{T}^{2n} = \mathcal{T}_{\geq 0}^{2n} \cup \mathcal{T}_{\leq 0}^{2n}$ . 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  $(v_{[i:w]})_j = \begin{cases} v_j & j \neq i \\ w_j & j = i. \end{cases}$  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  $\lceil A \rceil^{\text{Mzl}}$  and  $\lfloor A \rfloor^{\text{Mzl}}$ ,

respectively, where  $(\lceil A \rceil^{\text{Mzl}})_{ij} = \begin{cases} A_{ij} & A_{ij} \geq 0 \text{ or } i = j \\ 0 & \text{otherwise,} \end{cases}$  and  $\lfloor A \rfloor^{\text{Mzl}} = A - \lceil A \rceil^{\text{Mzl}}$ . Consider

a control system

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

with state  $x \in \mathbb{R}^n$ , measurable disturbance  $u : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^m$ . We denote the trajectory of the system with the disturbance  $u$  starting from  $x_0 \in \mathbb{R}^n$  at time  $t_0$  by  $t \mapsto \phi_f(t, t_0, x_0, u)$ . Given an initial state set  $\mathcal{X}_0 \subseteq \mathbb{R}^n$  and a disturbance set  $\mathcal{W} \subseteq \mathbb{R}^m$ , we denote the reachable set of (1) by

$$\mathcal{R}_f(t, 0, \mathcal{X}_0, \mathcal{U}) = \{\phi_f(t, 0, x_0, u) \mid x_0 \in \mathcal{X}_0, u : \mathbb{R}_{\geq 0} \rightarrow \mathcal{U} \text{ is piecewise cont.}\}$$

In general, computing the exact reachable sets of the system (1) is computationally intractable. Mixed monotone theory (Smith, 2008) has recently emerged as a computationally efficient framework for over-approximating the reachable sets of nonlinear systems (Coogan and Arcak, 2015). The control system (1) is mixed monotone if there exists a locally Lipschitz *decomposition function*  $F : \mathcal{T}^{2n} \times \mathcal{T}^{2m} \rightarrow \mathbb{R}^n$ , satisfying

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

Several different approaches have proposed in the literature for constructing the decomposition function for the control system (1) (Meyer et al., 2019; Abate et al., 2021; Abate and Coogan, 2021). Using a decomposition function  $F$ , we construct an embedding system associated to (1):

$$\frac{d}{dt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} = \begin{bmatrix} F(x, \hat{x}, u, \hat{u}) \\ F(\hat{x}, x, \hat{u}, u) \end{bmatrix} \quad (2)$$

The key result in mixed monotone reachability theory is as follows: if  $t \mapsto \begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix}$  is the trajectory of the embedding system (2) with disturbance  $\begin{bmatrix} u \\ \hat{u} \end{bmatrix} = \begin{bmatrix} u \\ \hat{u} \end{bmatrix}$  starting from  $\begin{bmatrix} x_0 \\ \hat{x}_0 \end{bmatrix}$ , then  $\mathcal{R}_f(t, 0, [x_0, \hat{x}_0], [u, \hat{u}]) \subseteq [x(t), \hat{x}(t)]$ , for every  $t \in \mathbb{R}_{\geq 0}$  (Meyer et al., 2019, Proposition 6). Given a map  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , the function  $\begin{bmatrix} \underline{G} \\ \overline{G} \end{bmatrix} : \mathcal{T}_{\geq 0}^{2n} \rightarrow \mathcal{T}_{\geq 0}^{2m}$  is an *inclusion function* for  $g$  if, for every  $x \leq \hat{x}$  and every  $z \in [x, \hat{x}]$ , we have  $\underline{G}(x, \hat{x}) \leq g(z) \leq \overline{G}(x, \hat{x})$ . 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

$$\dot{x} = f(x, u, w) \quad (3)$$

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  $f : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}^n$  is a parameterized vector field and the state feedback is parameterized by a  $k$ -layer feed-forward neural network controller  $\mathbf{N} : \mathbb{R}^n \rightarrow \mathbb{R}^p$  defined by:

$$\begin{aligned} \xi^{(i)}(x) &= \phi^{(i)}(W^{(i-1)}\xi^{(i-1)}(x) + b^{(i-1)}), \quad i \in \{1, \dots, k\} \\ x &= \xi^{(0)} \quad u = W^{(k)}\xi^{(k)}(x) + b^{(k)} := \mathbf{N}(x), \end{aligned} \quad (4)$$

where  $n_i$  is the number of neurons in the  $i$ th layer,  $W^{(i-1)} \in \mathbb{R}^{n_i \times n_{i-1}}$  is the weight matrix of the  $i$ th layer,  $b^{(i-1)} \in \mathbb{R}^{n_i}$  is the bias vector of the  $i$ th layer,  $\xi^{(i)}(y) \in \mathbb{R}^{n_i}$  is the  $i$ -th layer hidden variable, and  $\phi^{(i)} : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_i}$  is  $i$ th layer diagonal activation function satisfying  $0 \leq \frac{\phi_j^{(i)}(x) - \phi_j^{(i)}(y)}{x - y} \leq 1$ , for every  $j \in \{1, \dots, n_i\}$ . One can show that a large class of activation function including, but not restricted to, ReLU, leaky ReLU, sigmoid, and tanh satisfies this condition. We assume that, the neural network (4) 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 verify the safety of the closed-loop system given by:

$$\dot{x} = f(x, \mathbf{N}(x), w) := f^{\text{cl}}(x, w). \quad (5)$$

i.e., to ensure that the closed-loop system avoids the unsafe domain  $\mathbb{R}^n / \mathcal{S}$ . Given an initial state set  $\mathcal{X}_0 \subseteq \mathbb{R}^n$ , our goal is to check if  $\mathcal{R}_{f^{\text{cl}}}(t, 0, \mathcal{X}_0, \mathcal{W}) \subseteq \mathcal{S}$  holds for every  $t \in \mathbb{R}_{\geq 0}$ . Our approach is based on constructing a computationally efficient over-approximation  $\bar{\mathcal{R}}_{f^{\text{cl}}}(t, 0, \mathcal{X}_0, \mathcal{W})$  of the reachable set of the closed-loop system. Then, avoiding the unsafe set  $\mathbb{R}^n / \mathcal{S}$  is guaranteed when  $\bar{\mathcal{R}}_{f^{\text{cl}}}(t, 0, \mathcal{X}_0, \mathcal{W}) \subseteq \mathcal{S}$ , 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 (4) and let the perturbed input vector  $x$  be in the interval  $[\underline{x}, \bar{x}]$ . For every  $i \in \{1, \dots, k\}$ , we define the pre-activation input to  $i$ th layer as  $z^{(i)} = W^{(i)}\xi^{(i-1)}(x) + b^{(i)}$ . When  $x$  is perturbed in the interval  $[\underline{x}, \bar{x}]$ , we assume that  $L^{(i)}, U^{(i)} \in \mathbb{R}^{n_i}$  are such that  $L^{(i)} \leq z^{(i)} \leq U^{(i)}$ . These bounds can be obtained using a fast neural network verification algorithm such as the Interval Bound Propagation (IBP) method (Gowal et al., 2019, Equations (6) and (7)). In this case, for the  $j$ th neuron in the  $i$ th layer, there exist  $\alpha_{U,j}^{(i)}, \beta_{U,j}^{(i)}, \alpha_{L,j}^{(i)}, \beta_{L,j}^{(i)}$  such that

$$\alpha_{L,j}^{(i)}(z + \beta_{L,j}^{(i)}) \leq \phi_i(z) \leq \alpha_{U,j}^{(i)}(z + \beta_{U,j}^{(i)}), \quad \text{for every } L_j^{(i)} \leq z \leq U_j^{(i)}. \quad (6)$$

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

$$\underline{A}(\underline{x}, \bar{x})x + \underline{b}(\underline{x}, \bar{x}) \leq \mathbf{N}(x) \leq \bar{A}(\underline{x}, \bar{x})x + \bar{b}(\underline{x}, \bar{x}), \quad (7)$$

where functions  $\underline{A}, \bar{A} : \mathcal{T}_{\geq 0}^{2n} \rightarrow \mathbb{R}^{p \times n}$  and  $\underline{b}, \bar{b} : \mathcal{T}_{\geq 0}^{2n} \rightarrow \mathbb{R}^p$  are defined as follows:

$$\begin{aligned} \bar{A}(\underline{x}, \bar{x}) &= \Lambda^{(0)}, & \bar{b}(\underline{x}, \bar{x}) &= \sum_{i=1}^k \Lambda^{(i)} \bar{b}^{(i)} + \text{diag}(\Lambda^{(i)} \Delta^{(i)}), \\ \underline{A}(\underline{x}, \bar{x}) &= \Omega^{(0)}, & \underline{b}(\underline{x}, \bar{x}) &= \sum_{i=1}^k \Omega^{(i)} \underline{b}^{(i)} + \text{diag}(\Omega^{(i)} \Theta^{(i)}) \end{aligned} \quad (8)$$

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

**Theorem 1 (Inclusion functions for Neural Networks)** *Consider the  $k$ -layer feed-forward neural network  $u = \mathbf{N}(x)$  given by (4). Then,*

(i) *for every input perturbation interval  $[\underline{x}, \bar{x}]$ , the mapping  $\begin{bmatrix} \underline{G}_{[\underline{x}, \bar{x}]} \\ \bar{G}_{[\underline{x}, \bar{x}]} \end{bmatrix} : \mathcal{T}_{\geq 0}^{2n} \rightarrow \mathcal{T}_{\geq 0}^{2p}$  defined by*

$$\begin{aligned} \underline{G}_{[\underline{x}, \bar{x}]}(x, \hat{x}) &= [\underline{A}(\underline{x}, \bar{x})]^+ x + [\underline{A}(\underline{x}, \bar{x})]^- \hat{x} + \underline{b}(\underline{x}, \bar{x}), \\ \bar{G}_{[\underline{x}, \bar{x}]}(x, \hat{x}) &= [\bar{A}(\underline{x}, \bar{x})]^+ \hat{x} + [\bar{A}(\underline{x}, \bar{x})]^- x + \bar{b}(\underline{x}, \bar{x}) \end{aligned} \quad (9)$$

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

(ii) *the mapping  $\begin{bmatrix} \underline{H} \\ \bar{H} \end{bmatrix} : \mathcal{T}_{\geq 0}^{2n} \rightarrow \mathcal{T}_{\geq 0}^{2p}$  defined by  $\underline{H}(x, \hat{x}) = \underline{G}_{[\underline{x}, \bar{x}]}(x, \hat{x})$  and  $\bar{H}(x, \hat{x}) = \bar{G}_{[\underline{x}, \bar{x}]}(x, \hat{x})$  is an inclusion function for the neural network  $\mathbf{N}$  on  $\mathbb{R}^n$ .*

## 5. Interval reachability of closed-loop system

In this section, inspired by mixed monotone reachability, we present a system-level approach for over-approximating the reachable set of the closed-loop system (5) with the neural network controller (4). The key idea is to design a suitable function  $E = \begin{bmatrix} \underline{E} \\ \bar{E} \end{bmatrix} : \mathcal{T}_{\geq 0}^{2n} \times \mathcal{T}_{\geq 0}^{2q} \rightarrow \mathbb{R}^n$  and use it to construct the following embedding system associated to the closed-loop system (5):

$$\frac{d}{dt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} = \begin{bmatrix} \underline{E}(x, \hat{x}, w, \hat{w}) \\ \bar{E}(\hat{x}, x, \hat{w}, w) \end{bmatrix} \quad (10)$$

Then, one can use the embedding system (10) to study the propagation of the state and disturbance bounds with time. Suppose that the initial condition set is given by  $\mathcal{X}_0 \subseteq [\underline{x}_0, \bar{x}_0]$  and the disturbance set is given by  $\mathcal{W} \subseteq [\underline{w}, \bar{w}]$ . Let  $F$  be a decomposition function for the open-loop system (3) and  $\begin{bmatrix} \underline{G} \\ \bar{G} \end{bmatrix}$  and  $\begin{bmatrix} \underline{H} \\ \bar{H} \end{bmatrix}$  be the inclusion functions of the neural network (4) established in Theorem 1. We introduce ‘‘global’’ function  $E^G$ , ‘‘hybrid’’ function  $E^H$ , and ‘‘local’’ function  $E^L$  as follows:

$$\begin{aligned} \begin{bmatrix} \underline{E}_i^G(x, \hat{x}, w, \hat{w}) \\ \bar{E}_i^G(x, \hat{x}, w, \hat{w}) \end{bmatrix} &= \begin{bmatrix} F_i(x, \hat{x}, \underline{H}(x, \hat{x}), \bar{H}(x, \hat{x}), w, \hat{w}) \\ F_i(\hat{x}, x, \bar{H}(x, \hat{x}), \underline{H}(x, \hat{x}), \hat{w}, w) \end{bmatrix} \\ \begin{bmatrix} \underline{E}_i^H(x, \hat{x}, w, \hat{w}) \\ \bar{E}_i^H(x, \hat{x}, w, \hat{w}) \end{bmatrix} &= \begin{bmatrix} F_i(x, \hat{x}, \underline{G}_{[\underline{x}, \bar{x}]}(x, \hat{x}_{[i:x]}), \bar{G}_{[\underline{x}, \bar{x}]}(x, \hat{x}_{[i:x]}), w, \hat{w}) \\ F_i(\hat{x}, x, \bar{G}_{[\underline{x}, \bar{x}]}(x_{[i:\hat{x}]}, \hat{x}), \underline{G}_{[\underline{x}, \bar{x}]}(x_{[i:\hat{x}]}, \hat{x}), \hat{w}, w) \end{bmatrix} \\ \begin{bmatrix} \underline{E}_i^L(x, \hat{x}, w, \hat{w}) \\ \bar{E}_i^L(x, \hat{x}, w, \hat{w}) \end{bmatrix} &= \begin{bmatrix} F_i(x, \hat{x}, \underline{H}(x, \hat{x}_{[i:x]}), \bar{H}(x, \hat{x}_{[i:x]}), w, \hat{w}) \\ F_i(\hat{x}, x, \bar{H}(x_{[i:\hat{x}]}, \hat{x}), \underline{H}(x_{[i:\hat{x}]}, \hat{x}), \hat{w}, w) \end{bmatrix}, \end{aligned} \quad (11)$$

for every  $i \in \{1, \dots, n\}$ .

**Theorem 2 (Interval over-approximation of reachable sets)** Consider the control system (3) with the neural network controller (4). Suppose that the initial condition  $x_0$  belongs to  $\mathcal{X}_0 \subseteq [\underline{x}_0, \bar{x}_0]$  and the disturbance  $w$  belongs to  $\mathcal{W} \subseteq [\underline{w}, \bar{w}]$ . Let  $F : \mathcal{T}^{2n} \times \mathcal{T}^{2p} \times \mathcal{T}^{2q} \rightarrow \mathbb{R}^n$  be a decomposition function for the open-loop system (3). Given  $s \in \{G, L, H\}$ , consider  $E^G, E^H, E^L$  as defined in (11) and let  $t \mapsto \begin{bmatrix} \underline{x}^s(t) \\ \bar{x}^s(t) \end{bmatrix}$  be the trajectory of the embedding system (10) with  $E = E^s$  and disturbance  $[\frac{w}{\hat{w}}] = [\frac{\underline{w}}{\bar{w}}]$  starting from  $[\frac{x_0}{\bar{x}_0}]$ . Then, the following statements hold:

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

$$\mathcal{R}_{fcl}(t, 0, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}^s(t), \bar{x}^s(t)],$$

(ii) If all the activation functions of the neural network (4) are ReLU, then, for every  $t \in \mathbb{R}_{\geq 0}$ ,

$$\mathcal{R}_{fcl}(t, 0, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}^L(t), \bar{x}^L(t)] \subseteq [\underline{x}^H(t), \bar{x}^H(t)] \subseteq [\underline{x}^G(t), \bar{x}^G(t)].$$

When the open-loop system (3) is linear with  $f(x, u, w) = Ax + Bu + Cw$  where  $A \in \mathbb{R}^{n \times n}$ ,  $B \in \mathbb{R}^{n \times p}$ , and  $C \in \mathbb{R}^{n \times q}$ , one can find another embedding function for the closed-loop system (5):

$$E^{\text{Lin}}(x, \hat{x}, w, \hat{w}) = \begin{bmatrix} [A+R(x, \hat{x})]^{Mzl} & [A+R(x, \hat{x})]^{Mzl} \\ [A+S(x, \hat{x})]^{Mzl} & [A+S(x, \hat{x})]^{Mzl} \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} C^+ & C^- \\ C^- & C^+ \end{bmatrix} \begin{bmatrix} w \\ \hat{w} \end{bmatrix}, \quad (12)$$

where  $R(x, \hat{x}) = B^+[A(x, \hat{x})] + B^-[A(x, \hat{x})]$  and  $S(x, \hat{x}) = B^+[A(x, \hat{x})] + B^-[A(x, \hat{x})]$ . The next theorem compares reachable set over-approximation using  $E^{\text{Lin}}$  and hybrid function  $E^H$ .

**Theorem 3 (Linear systems)** Consider the control system (3) 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 (4). Suppose that the initial set satisfies  $\mathcal{X}_0 \subseteq [\underline{x}_0, \bar{x}_0]$  and the disturbance set satisfies  $\mathcal{W} \subseteq [\underline{w}, \bar{w}]$ . Then,

(i) We recover all the results of Theorem (2) with the following ‘‘hybrid’’ function:

$$E^H(x, \hat{x}, w, \hat{w}) = \begin{bmatrix} [A]^{Mzl} + [R(x, \hat{x})]^{Mzl} & [A]^{Mzl} + [R(x, \hat{x})]^{Mzl} \\ [A]^{Mzl} + [S(x, \hat{x})]^{Mzl} & [A]^{Mzl} + [S(x, \hat{x})]^{Mzl} \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} C^+ & C^- \\ C^- & C^+ \end{bmatrix} \begin{bmatrix} w \\ \hat{w} \end{bmatrix}.$$

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

(ii) if  $t \mapsto \begin{bmatrix} \underline{x}^{\text{Lin}}(t) \\ \bar{x}^{\text{Lin}}(t) \end{bmatrix}$  is the trajectory of the embedding system (12) with disturbance  $[\frac{w}{\hat{w}}] = [\frac{\underline{w}}{\bar{w}}]$  starting from  $[\frac{x_0}{\bar{x}_0}]$  then, for every  $t \in \mathbb{R}_{\geq 0}$ ,

$$\mathcal{R}_{fcl}(t, 0, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}^{\text{Lin}}(t), \bar{x}^{\text{Lin}}(t)] \subseteq [\underline{x}^H(t), \bar{x}^H(t)].$$

**Remark 4** The following remarks are in order.

(i) (**Comparison with the literature**) Theorem 3 can be used to show that the forward Euler integration of the embedding system (10) with  $E = E^{\text{Lin}}$  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.



- (ii) **(Generality of the approach)** Theorem 2 proposes a general embedding-based framework for verification of the closed-loop system (5) 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 which are heavily dependent on the neural network verification algorithm (see for instance (Everett et al., 2021b) and (Sidrane et al., 2022)).
- (iii) **(Computational complexity)** From a computational perspective, the framework presented in Theorem 2 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 (10). For a neural network with  $k$ -layer and  $N$  neuron per layer, the complexity of CROWN is  $\mathcal{O}(k^2 N^3)$  (Zhang et al., 2018). Moreover, the functions  $E^G$  and  $E^H$  call CROWN once per integration step, while the function  $E^L$  calls CROWN  $2n$  times per integration step. The run time of the integration process depends on 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 (2) to obtain a computationally efficient algorithm for generating reachable set over-approximations of the closed-loop system (5). Our partitioning strategy consists of two main components: (i) a uniform division of the state space of the embedding system (10) to compute the neural network inclusion functions using CROWN, and (ii) a uniform subdivision to implement the integration on the embedding system (10). We first pick an  $s \in \{G, H, L\}$  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  $(\bar{x}_0)_j = \max_{x \in \mathcal{X}_0} x_j$  and  $(\underline{x}_0)_j = \min_{x \in \mathcal{X}_0} x_j$ , for every  $i \in \{1, \dots, n\}$ . We then divide the hyper-rectangle  $[\underline{x}_0, \bar{x}_0]$  into  $D_a$  partitions and obtain the set  $\{[\underline{x}^1, \bar{x}^1], \dots, [\underline{x}^{D_a}, \bar{x}^{D_a}]\}$ . For every  $k \in \{1, \dots, D_a\}$ , we compute the trajectory of the embedding system (10) with  $E = E^s$  and the initial condition  $\begin{bmatrix} \underline{x}^k \\ \bar{x}^k \end{bmatrix}$  at time  $\Delta t$  as in Theorem 2. For every  $k \in \{1, \dots, D_a\}$ , we divide each states of the hyper-rectangle  $[\underline{x}^k, \bar{x}^k]$  into  $D_s$  partitions, obtaining the subpartitions  $\{[\underline{x}^{k,1}, \bar{x}^{k,1}], \dots, [\underline{x}^{k,D_s}, \bar{x}^{k,D_s}]\}$ . For every  $k \in \{1, \dots, D_a\}$  and  $l \in \{1, \dots, D_s\}$ , we compute the trajectory of the embedding system (10) with  $E = E^{\text{Bs}}$  defined by

$$\begin{bmatrix} \underline{E}_i^{\text{Bs}}(x, \hat{x}, w, \hat{w}) \\ \bar{E}_i^{\text{Bs}}(x, \hat{x}, w, \hat{w}) \end{bmatrix} = \begin{bmatrix} F_i(x, \hat{x}, \underline{G}_{[\underline{x}^k, \bar{x}^k]}(x, \hat{x}_{[i:x]}), \bar{G}_{[\underline{x}^k, \bar{x}^k]}(\hat{x}_{[i:x]}, x), w, \hat{w}) \\ F_i(\hat{x}, x, \underline{G}_{[\underline{x}^k, \bar{x}^k]}(x_{[i:\hat{x}]}, \hat{x}), \bar{G}_{[\underline{x}^k, \bar{x}^k]}(\hat{x}, x_{[i:\hat{x}]}) \hat{w}, w) \end{bmatrix}, \quad (13)$$

and with the initial condition  $\begin{bmatrix} \underline{x}^{k,l} \\ \bar{x}^{k,l} \end{bmatrix}$  at time  $\Delta t$  as in Theorem 2. We then set  $\mathcal{X}_1 = \bigcup_{k=1}^{D_a} \bigcup_{l=1}^{D_s} [\underline{x}^{k,l}(\Delta t), \bar{x}^{k,l}(\Delta t)]$  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-partition 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 (5)
 

---

**Input:**  $s \in \{G, H, L\}$ , 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  $\overline{\mathcal{R}}(T, 0, \mathcal{X}_0, \mathcal{W})$

```

1:  $j \leftarrow 0$ 
2: while  $j < \lfloor \frac{T}{\Delta t} \rfloor$  do
3:    $\underline{x}_j \leftarrow \min_{x \in \mathcal{X}_j} x_i$ , for every  $i \in \{1, \dots, n\}$ 
4:    $\overline{x}_j \leftarrow \max_{x \in \mathcal{X}_j} x_i$ , for every  $i \in \{1, \dots, n\}$ 
5:    $\{[\underline{x}^1, \overline{x}^1], \dots, [\underline{x}^{D_a}, \overline{x}^{D_a}]\} \leftarrow \text{uniform\_partition}([\underline{x}, \overline{x}], D_a)$ 
6:   for  $k = \{1, \dots, D_a\}$  do
7:     Compute  $\underline{A}(\underline{x}^k, \overline{x}^k)$ ,  $\overline{A}(\underline{x}^k, \overline{x}^k)$ ,  $\underline{b}(\underline{x}^k, \overline{x}^k)$ , and  $\overline{b}(\underline{x}^k, \overline{x}^k)$  using CROWN and (8).
8:      $\{[\underline{x}^{k,1}, \overline{x}^{k,1}], \dots, [\underline{x}^{k,D_s}, \overline{x}^{k,D_s}]\} \leftarrow \text{uniform\_partition}([\underline{x}^k, \overline{x}^k], D_s)$ 
9:     for  $l = \{1, \dots, D_s\}$  do
10:      Compute  $\begin{bmatrix} \underline{x}^{k,l}(\Delta t) \\ \overline{x}^{k,l}(\Delta t) \end{bmatrix}$  for system (10) with  $E = E^{\text{Bs}}$  and initial condition  $\begin{bmatrix} \underline{x}^{k,l} \\ \overline{x}^{k,l} \end{bmatrix}$ .
11:    end for
12:  end for
13:   $\mathcal{X}_{j+1} = \bigcup_{k=1}^{D_a} \bigcup_{l=1}^{D_s} [\underline{x}^{k,l}(\Delta t), \overline{x}^{k,l}(\Delta t)]$ 
14: end while
15: 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<sup>2</sup>.

### 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:

$$\dot{p}_x = v \cos(\phi + \beta(u_2)), \quad \dot{\phi} = \frac{v}{\ell_r} \sin(\beta(u_2)), \quad \dot{p}_y = v \sin(\phi + \beta(u_2)), \quad \dot{v} = u_1 + w, \quad (14)$$

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 decomposition function for the open-loop system:

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

---

2. All the code is available at

[https://github.com/gtfactslab/L4DC2023\\_NNControllerReachability](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

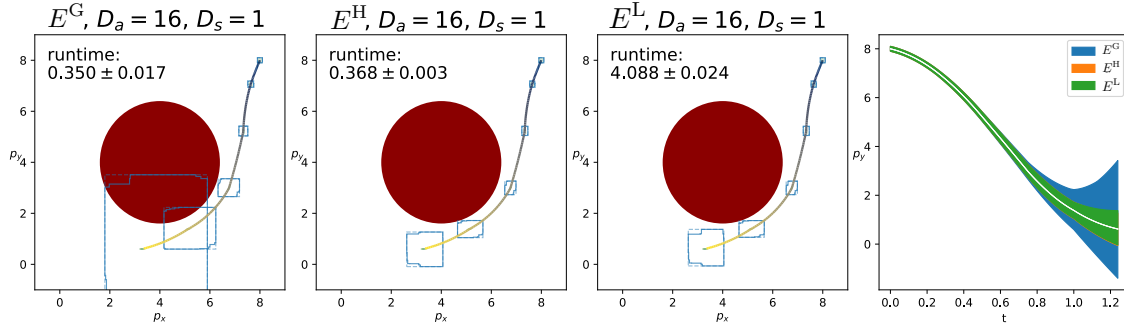


Figure 1: Performance of the three different functions  $E^G$ ,  $E^H$ , and  $E^L$  in Theorem (2) for over-approximation of the reachable set of the system (14) with neural network  $u = N(x)$  trained to approximate an offline model predictive controller. The  $p_x - p_y$  plot of the motion of the vehicle is shown starting from an initial set  $[7.9, 8.1]^2 \times [-\frac{2\pi}{3} - 0.01, -\frac{2\pi}{3} + 0.01] \times [1.99, 2.01]$ . The size of the over-approximations obtained using the functions  $E^H$  and  $E^L$  are close to each other, but are much smaller than the size of the over-approximations obtained using the function  $E^G$ . On the other hand, finding the over-approximations using the functions  $E^G$  and  $E^H$  take about the same amount of time and are much faster than finding the over-approximations using the functions  $E^L$ . The runtimes are averaged over 10 instances and mean and standard deviation are reported.

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  $Q = \text{diag}(1, 1, 0, 0)$ ,  $Q_{hor} = \text{diag}(100, 100, 0, 1)$ , 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  $u = N(x)$  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 (14), comparing functions  $E^G$ ,  $E^L$ , and  $E^H$  from Theorem 2. Algorithm 1 line 7 is computed using auto\_LiRPA (Xu et al., 2020). The results are shown in Figure 1.

## 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  $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$  is the state of the system with  $p_x, p_y$ , and  $p_z$  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  $u = N(x)$  consists of 2 layers with 32 neurons in each layer and is identical to (Everett et al., 2021b, Section H). We use Theorem 2 with the embedding function  $E^H$  to over-approximate the reachable sets of the closed-loop system. The results are shown in Figure 3.

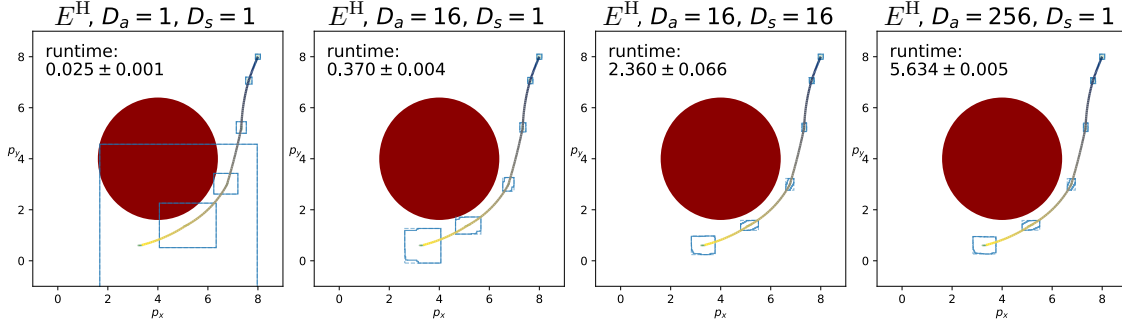


Figure 2: Performance of the Algorithm (1) with  $s = H$  for different partitions  $D_s$  and subpartitions  $D_a$  for over-approximation of the reachable set of the system (14) with neural network  $u = N(x)$  trained to approximate an offline model predictive controller. All four figures show the  $p_x - p_y$  plot of the motion of the vehicle. The runtimes are averaged over 10 instances and the mean and standard deviation are reported.

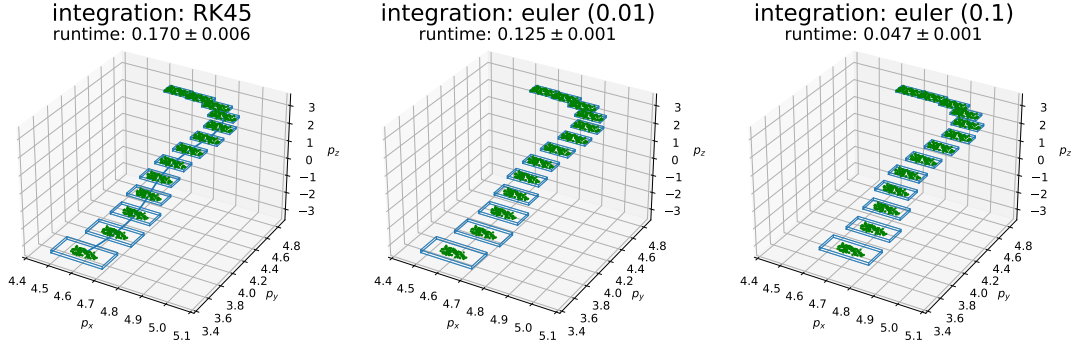


Figure 3: The time evolution of the reachable sets of the 6D quadrotor with the neural network controller  $u = N(x)$  is shown in  $p_x, p_y, p_z$  coordinate starting from the initial set  $[4.65, 4.75]^2 \times [2.95, 3.05] \times [0.94, 0.96] \times [-0.01, 0.01]^2$ . The blue hyper-rectangles are the over-approximations of the reachable sets of the system computed using the function  $E^{\text{Lin}}$  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 very similar to those in (Everett et al., 2021b, Figure 10(a)), which takes  $0.113 \pm 0.004$  seconds on the same computer. 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.

## Acknowledgments

This work was supported in part by Cisco Systems and the National Science Foundation under awards #1931980 and #2219755.

## References

- M. Abate and S. Coogan. Improving the fidelity of mixed-monotone reachable set approximations via state transformations. In *2021 American Control Conference (ACC)*, pages 4674–4679, 2021. doi:[10.23919/ACC50511.2021.9483264](https://doi.org/10.23919/ACC50511.2021.9483264).
- 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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.23919/ACC.2018.8431275).
- J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09-15 Jun 2019. URL <https://proceedings.mlr.press/v97/cohen19c.html>.
- 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](https://doi.org/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](https://doi.org/10.1145/3302504.3311807).
- T. Entesari, S. Sharifi, and M. Fazlyab. ReachLipBnB: A branch-and-bound method for reachability analysis of neural autonomous systems using lipschitz bounds. *arXiv Preprint*, 2022. URL <https://arxiv.org/abs/2211.00608>.

- 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](https://doi.org/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](https://doi.org/10.1109/ACCESS.2021.3133370).
- M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.
- 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](https://doi.org/10.1007/978-3-540-31954-2_19).
- S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. A. Mann, and P. Kohli. Scalable verified training for provably robust image classification. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4841–4850, 2019. doi:[10.1109/ICCV.2019.00494](https://doi.org/10.1109/ICCV.2019.00494).
- 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](https://doi.org/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 2019, page 169–178, 2019. doi:[10.1145/3302504.3311806](https://doi.org/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](https://doi.org/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](https://doi.org/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, and M. J. Kochenderfer. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404, 2021. doi:[10.1561/24000000035](https://doi.org/10.1561/24000000035).

- P.-J. Meyer, A. Devonport, and M. Arcak. Tira: Toolbox for interval reachability analysis. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 224–229, 2019. doi:[10.1145/3302504.3311808](https://doi.org/10.1145/3302504.3311808).
- 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](https://doi.org/10.1007/3-540-46430-1_27).
- P. Polack, F. Alché, 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](https://doi.org/10.1109/IVS.2017.7995816).
- C. Schilling, M. Forets, and S. Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8169–8177, 2022.
- 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.
- H. L. Smith. Global stability for mixed monotone systems. *Journal of Difference Equations and Applications*, 14(10-11):1159–1164, 2008. doi:[10.1080/10236190802332126](https://doi.org/10.1080/10236190802332126).
- 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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1109/ICRA.2016.7487175).