SEEV: Synthesis with Efficient Exact Verification for ReLU Neural Barrier Functions

Hongchao Zhang*
Electrical & Systems Engineering
Washington University in St. Louis
hongchao@wustl.edu

Zhizhen Qin*
Computer Science & Engineering
University of California, San Diego
zhizhenqin@ucsd.edu

Sicun Gao
Computer Science & Engineering
University of California, San Diego
scungao@ucsd.edu

Andrew Clark
Electrical & Systems Engineering
Washington University in St. Louis
andrewclark@wustl.edu

Abstract

Neural Control Barrier Functions (NCBFs) have shown significant promise in enforcing safety constraints on nonlinear autonomous systems. State-of-the-art exact approaches to verifying safety of NCBF-based controllers exploit the piecewise-linear structure of ReLU neural networks, however, such approaches still rely on enumerating all of the activation regions of the network near the safety boundary, thus incurring high computation cost. In this paper, we propose a framework for Synthesis with Efficient Exact Verification (SEEV). Our framework consists of two components, namely (i) an NCBF synthesis algorithm that introduces a novel regularizer to reduce the number of activation regions at the safety boundary, and (ii) a verification algorithm that exploits tight over-approximations of the safety conditions to reduce the cost of verifying each piecewise-linear segment. Our simulations show that SEEV significantly improves verification efficiency while maintaining the CBF quality across various benchmark systems and neural network structures. Our code is available at https://github.com/HongchaoZhang-HZ/SEEV.

1 Introduction

Safety is a crucial property for autonomous systems that interact with humans and critical infrastructures in applications including medicine, energy, and robotics [1, 2], which has motivated recent research into safe control [3, 4, 5, 6, 7]. Control Barrier Functions (CBFs), which apply a constraint on the control input at each time in order to ensure that safety constraints are not violated, have attracted significant research attention due to their ease of implementation and compatibility with a variety of safety and performance criteria [8]. Recently, CBFs that are defined by neural networks, denoted as Neural Control Barrier Functions (NCBFs), have been proposed to leverage the expressiveness of NNs for safe control of nonlinear systems [9, 10, 11]. NCBFs have shown substantial promise in applications including robotic manipulation [10], navigation [12, 13], and flight control [14].

A key challenge in NCBF-based control is safety verification, which amounts to ensuring that the constraints on the control can be satisfied throughout the state space under actuation limits. The NCBF safety verification problem effectively combines two problems that are known to be difficult, namely, input-output verification of neural networks (VNN) [15, 16, 17, 18, 19, 20] and reachability verification of nonlinear systems. While sound and complete verifiers such as dReal can be applied to

*Equal contribution

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3214

NCBFs, they typically can only handle systems of dimension three or small neural networks [21, 22]. In [23], exact conditions for safety verification of NCBFs with ReLU activation functions were proposed that leverage the piecewise-linearity of ReLU-NNs to reduce verification time compared to dReal for general activation functions. The exact conditions, however, still require checking correctness of the NCBF by solving a nonlinear optimization problem along each piecewise-linear segment. Hence, the NCBF verification problem remains intractable for high-dimensional systems.

In this paper, we propose a framework for Synthesis with Efficient Exact Verification (SEEV) for piecewise-linear NCBFs. The main insight of SEEV is that the computational bottleneck of NCBF verification is the inherent requirement of verifying each linear segment of the neural network. We mitigate this bottleneck by (i) developing a training procedure that reduces the number of segments that must be verified and (ii) constructing verification algorithms that efficiently enumerate the linear segments at the safety boundary and exploit easily-checked sufficient conditions to reduce computation time. Towards (i), we introduce a regularizer to the loss function that penalizes the dissimilarity of activation patterns along the CBF boundary. Towards (ii), we propose a breadth-first search algorithm for efficiently enumerating the boundary segments, as well as tight linear over-approximations of the nonlinear optimization problems for verifying each segment. Moreover, we integrate the synthesis and verification components by incorporating safety counterexamples returned by the safety verifier into the training dataset. Our simulation results demonstrate significant improvements in verification efficiency and reliability across a range of benchmark systems.

Related Work: Neural control barrier functions have been proposed to describe complex safety sets to remain inside and certify safety of a controlled system [24, 21, 25, 26] or synthesize control input based on NCBFs to ensure safety [8, 9, 10, 27]. However, the synthesized NCBF may not ensure safety. Safety verification of NCBFs is required. Sum-of-squares (SOS) optimization [28, 29, 30, 31, 32] has been widely used for polynomial barrier functions, however, they are not applicable due to the non-polynomial and potentially non-differentiable activation functions of NCBFs. VNN [33, 34, 35] and methods for ReLU neural networks [36, 37] are also not directly applicable to NCBF verification. Nonlinear programming approach [23] provides another route for exact verification but is computationally intensive and relies on VNN tools. To synthesize neural networks with verifiable guarantees, Counterexample Guided Inductive Synthesis (CEGIS) has been applied using SMT-based techniques [21, 38, 39, 40, 41]. Other verification-in-the-loop approaches utilize reachability analysis [42] and branch-and-bound neural network verification tools [43]. However, existing works suffer from the difficulty of performing verification and generating counterexamples in a computationally efficient manner. Sampling-based approaches [41, 11] aim to prove safety using Lipschitz conditions, but they rely on dense sampling over the state space, which is computationally prohibitive. In this work, we present SEEV to integrate the synthesis and efficient verification by incorporating safety counterexamples from the exact verification.

Organization The remainder of the paper is organized as follows. Section 2 gives the system model and background on neural networks and the conditions of valid NCBFs. Section 3 presents the SEEV framework. Section 4 presents our efficient and exact verification. Section 5 contains simulation results. Section 6 concludes the paper.

2 Preliminaries

This section presents the system model, notations on neural networks, and exact conditions of safety.

2.1 System Model

We consider a system with state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ and input $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$, with initial condition $x(t_0) = x_0$ where x_0 lies in an initial set $\mathcal{I} \subseteq \mathcal{X}$. The continuous-time nonlinear control-affine system has the dynamics given by

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are known continuous functions.

We consider the case that the system is required to remain inside a given set of states, i.e., $x(t) \in \mathcal{C}$ for all time $t \geq t_0$. The set $\mathcal{C}$, referred to as the *safe set*, is defined as $\mathcal{C} = \{x : h(x) \geq 0\}$ by some given continuous function $h : \mathbb{R}^n \to \mathbb{R}$. The unsafe region is given by $\mathcal{X} \setminus \mathcal{C}$.

2.2 Neural Network Model and Notations

We let $\mathbf{W}$ and $\mathbf{r}$ denote the weight and bias of a neural network, and let θ be a parameter vector obtained by concatenating $\mathbf{W}$ and $\mathbf{r}$. We consider a θ-parameterized feedforward neural network $b_\theta : \mathbb{R}^n \to \mathbb{R}$ constructed as follows. The network consists of L layers, with each layer i consisting of M_i neurons. We let $(i, j) \in \{1, \ldots, L\} \times \{1, \ldots, M_i\}$ denote the j-th neuron at the i-th layer. We denote the pre-activation input as $z_j^{(i)}$, piecewise linear activation function σ and the post-activation output as $\hat{z}_j^{(i)} = \sigma(z_j^{(i)})$. Specifically, we assume that the NN has the Rectified Linear Unit (ReLU) activation function, defined by $\sigma(z) = z$ for $z \geq 0$ and $\sigma(z) = 0$ for $z < 0$. We define the neuron (i, j) as *active* if $z_j^{(i)} > 0$, *inactive* if $z_j^{(i)} < 0$ and *unstable* if $z_j^{(i)} = 0$. Let $\mathbf{S} = \tau_S(x) = \{(i, j) : z_j^{(i)} \geq 0\} \subseteq \{(i, j) : i = 1, \ldots, L, j = 1, \ldots, M_i\}$ denote the set of activated and unstable neurons, produced by state x and function τ_S. Let $\mathbf{T}(x) = \tau_T(x) = \{(i, j) : z_j^{(i)} = 0\}$ denote the set of unstable neurons produced by x and τ_T. $\mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)$ denote the set of unstable neurons produced by activation sets $\mathbf{S}_1, \ldots, \mathbf{S}_r$. The set of inactive neurons is given by $\mathbf{S}^c$, i.e., the complement of $\mathbf{S}$, and consists of neurons with negative pre-activation input. We define vectors $\overline{W}_{ij}(\mathbf{S}) \in \mathbb{R}^n$ and scalars $\overline{r}_{ij}(\mathbf{S}) \in \mathbb{R}$ such that $z_j^{(i)} = \overline{W}_{ij}(\mathbf{S})^T x + \overline{r}_{ij}(\mathbf{S})$ in the Appendix A.1. The symmetric difference between two sets $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathbf{A}\Delta\mathbf{B}$, is defined as $\mathbf{A}\Delta\mathbf{B} = (\mathbf{A} \setminus \mathbf{B}) \cup (\mathbf{B} \setminus \mathbf{A})$.

Finally, we define the terms *hyperplane* and *hinge*. For any $\mathbf{S} \subseteq \{1, \ldots, L\} \times \{1, \ldots, M_i\}$, we define $\overline{\mathcal{X}}(\mathbf{S}) := \{x : \mathbf{S}(x) = \mathbf{S}\}$. The collection of $\overline{\mathcal{X}}(\mathbf{S})$ for all $\mathbf{S}$ is the set of *hyperplanes* associated with the ReLU neural network. A hyperplane that intersects the set $\{x : b_\theta(x) = 0\}$ is a *boundary hyperplane*. The intersection of hyperplanes $\overline{\mathcal{X}}(\mathbf{S}_1), \ldots, \overline{\mathcal{X}}(\mathbf{S}_r)$ is called a *hinge*. A hinge that intersects the set $\{x : b_\theta(x) = 0\}$ is a *boundary hinge*.

2.3 Guaranteeing Safety via Control Barrier Functions

Barrier certificates [28] ensure the safety of a feedback-controlled system under policy $\mu(x \mid \lambda)$ by identifying a CBF to represent the invariant safe set. The barrier certificate defines an inner safe region $\mathcal{D} := \{x : b(x) \geq 0\}$ for some continuous function b. The verifiable invariance of $\mathcal{D}$ is obtained from the following result.

Theorem 1 (Nagumo's Theorem [44], Section 4.2)**.** *A closed set $\mathcal{D}$ is controlled positive invariant if, whenever $x \in \partial\mathcal{D}$, where $\partial\mathcal{D}$ denotes the boundary of $\mathcal{D}$. we have*

$$(f(x) + g(x)u) \in \mathcal{A}_{\mathcal{D}}(x) \tag{2}$$

for some $u \in \mathcal{U}$ where $\mathcal{A}_{\mathcal{D}}(x)$ is the tangent cone to $\mathcal{D}$ at x.

We denote a state $\hat{x}_{ce}^c$ with $\hat{x}_{ce}^c \in \partial\mathcal{D}$ that violates (2) as a *safety counterexample*. In the case where b is continuously differentiable, (2) can be satisfied by selecting u to satisfy the condition $\frac{\partial b}{\partial x}(f(x(t)) + g(x(t))u(t)) \geq -\alpha(b(x(t)))$, where $\alpha : \mathbb{R} \to \mathbb{R}$ is a strictly increasing function with $\alpha(0) = 0$. When b is not continuously differentiable, as in a ReLU NCBFs, a modified condition is needed. Prior work [23] introduces exact conditions for safety verification of ReLU NCBFs, based on the following proposition. A collection of activation sets $\mathbf{S_1}, \ldots, \mathbf{S_r}$ is complete if, for any $\mathbf{S}' \notin \{\mathbf{S_1}, \ldots, \mathbf{S_r}\}$, we have $\overline{\mathcal{X}}(\mathbf{S_1}) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S_r}) \cap \overline{\mathcal{X}}(\mathbf{S}') = \emptyset$.

Proposition 1. *Suppose the function ReLU neural network-defined function b satisfies the following conditions:*

(i) For all activation sets $\mathbf{S}_1, \ldots, \mathbf{S}_r$ with $\{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ complete and any x satisfying $b(x) = 0$ and

$$x \in \left(\bigcap_{l=1}^{r} \overline{\mathcal{X}}(\mathbf{S}_l) \right), \tag{3}$$

there exist $l \in \{1, \ldots, r\}$ and $u \in \mathcal{U}$ such that

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T(f(x) + g(x)u) \geq 0 \quad \forall (i, j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \cap \mathbf{S}_l \tag{4}$$

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T(f(x) + g(x)u) \leq 0 \quad \forall (i, j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \setminus \mathbf{S}_l \tag{5}$$

$$\overline{W}(\mathbf{S}_l)^T(f(x) + g(x)u) \geq 0 \tag{6}$$

 https://doi.org/10.52202/079017-3214

(ii) For all activation sets $\mathbf{S}$*, we have*

$$(\overline{\mathcal{X}}(\mathbf{S}) \cap \mathcal{D}) \setminus \mathcal{C} = \emptyset \tag{7}$$

If $b(x(0)) \geq 0$*, then* $x(t) \in \mathcal{C}$ *for all* $t \geq 0$*.*

Any feedback control law $\mu : \mathcal{X} \to \mathcal{U}$ that satisfies (4)–(6) is guaranteed to ensure safety and is referred to as an NCBF control policy. Given a nominal control policy $\pi_{nom}(x)$, safe actions can be derived from a ReLU NCBF as a safety filter[29, 9] by solving the following optimization problem proposed in [23, Lemma 2]:

$$\min_{\mathbf{S} \in \mathbf{S}(x), u} ||u - \pi_{nom}(x)||_2^2 \quad \text{s.t.} \quad \overline{W}(\mathbf{S})^T (f(x) + g(x)u) \geq -\alpha(b(x)), u \in \mathcal{U}, (4) - (6) \tag{8}$$

The solution to this optimization problem provides a control u that minimally deviates from the nominal control $\pi_{nom}(x)$ while satisfying NCBF constraints derived in Proposition 1 ensuring that $\mathcal{D}$ is positive invariant and is contained in $\mathcal{C}$. Based on Proposition 1, we can define different types of safety counterexamples. Correctness counterexamples, denoted by $\hat{x}_{ce}^c$, refers to a state $\hat{x}_{ce}^c \in \mathcal{D} \cap (\mathcal{X} \setminus \mathcal{C})$. Hyperplane verification counterexamples refer to states $\hat{x}_{ce}^h$ that violate (6). Hinge verification counterexamples are states x with $\mathbf{T}(x) \neq \emptyset$ that violate (4)–(6).

3 Synthesis

In this section, we present the framework to synthesize NCBF $b_\theta(x)$ to ensure the safety of the system (1). The synthesis framework aims to train an NCBF and construct an NCBF-based safe control policy. We first formulate the problem and present an overview of the framework in 3.1. Then we demonstrate the design of the loss function in 3.2.

3.1 Overall Formulation

Our primary objective is to synthesize a ReLU Neural Control Barrier Function (ReLU-NCBF) for (1) and develop a safe control policy to ensure system safety.

Problem 1. *Given a system* (1)*, initial set* $\mathcal{I}$ *and a safety set* $\mathcal{C}$*, synthesize a ReLU NCBF* $b_\theta(x)$ *parameterized by* θ *such that the conditions in Proposition 1 are satisfied. Then construct a control policy to synthesize* u_t *such that the system remains positive invariant in* $\mathcal{D} := \{x : b_\theta(x) \geq 0\} \subseteq \mathcal{C}$*.*

We propose SEEV to address this problem with the synthesis framework demonstrated in Fig. 1. The training dataset $\mathcal{T}$ is initialized by uniform sampling over $\mathcal{X}$. The training framework consists of two loops. The inner loop attempts to choose parameter θ for $b_\theta(x)$ to satisfy the safety condition by minimizing the loss function over training data $\mathcal{T}$. The outer loop validates a given NCBF $b_\theta(x)$ by searching for safety counterexamples $\hat{x}_{ce}$ and updates the training dataset as $\mathcal{T} \cup \{\hat{x}_{ce}\}$.

To train the parameters of the NCBF to satisfy the conditions of Proposition 1, we propose a loss function that penalizes the NCBF for violating constraints (i) and (ii) at a collection of sample points. The loss function is a weighted sum of three terms. The first term is the correctness loss penalizing state $\hat{x} \in \mathcal{X} \setminus \mathcal{C}$ with $b_\theta(x) \geq 0$. The second term is verification loss that penalizes states $\hat{x}$ that $\nexists u$ such that (4)-(6) hold. The third term is a regularizer minimizing the number of hyperplanes and hinges along the boundary. However, minimizing the loss function is insufficient to ensure safety [9] because there may exist safety counterexamples outside of the training dataset. In order to guarantee safety, SEEV introduces an efficient exact verifier to certify whether $b_\theta(x)$ meets the safety conditions outlined in Proposition 1. The verifier either produces a proof of safety or generates a safety counterexample that can be added to the training dataset to improve the NCBF.

The integration of the verifier can improve safety by adding counterexamples to guide the training process, however, it may also introduce additional computation complexity. We propose a combined approach, leveraging two complementary methods to address this issue. First, the verification of SEEV introduces an efficient algorithm in Section 4 to mitigate the computational scalability challenges that arise as neural network depth and dimensionality increase. Second, SEEV introduces a regularizer to limit the number of boundary hyperplanes and hinges to be verified, addressing the complexity that arises as neural network size increases.

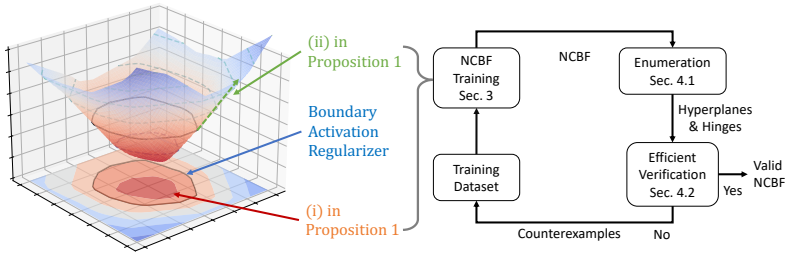


Figure 1: SEEV: Synthesis with Efficient Exact Verifier for ReLU NCBF

3.2 Loss Function Design and NCBF Training

The goal of the inner loop is to choose parameters θ so that the conditions of Proposition 1 are satisfied for all $\hat{x} \in \mathcal{T}$ and the computational cost of verifying safety is minimized. To achieve the latter objective, we observe (based on results presented in Table 2) that the computational complexity of verification grows with the cardinality of the collection of activation sets that intersect the safety boundary $\partial\mathcal{D}$. The collection is defined as $\mathcal{B} := \{\mathbf{S}_i : \partial\mathcal{D} \cap \overline{\mathcal{X}}(\mathbf{S}_i) \neq \emptyset\}$. Hence, we formulate the following unconstrained optimization problem to search for θ.

$$\min_{\theta} \quad \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}}(\mathcal{T}) + \lambda_f\mathcal{L}_f(\mathcal{T}) + \lambda_c\mathcal{L}_c(\mathcal{T}) \tag{9}$$

where $\mathcal{L}_{\mathcal{B}}(\mathcal{T})$ regularizer to minimize $|\mathcal{B}|$, $\mathcal{L}_f(\mathcal{T})$ is the loss penalizing the violations of constraint (4)-(6) ((i) of Proposition 1), $\mathcal{L}_c(\mathcal{T})$ penalizes the violations of constraint (7) ((ii) of Proposition 1), and $\lambda_{\mathcal{B}}$, λ_f and λ_c are non-negative coefficients defined as follows.

$\mathcal{L}_f$ **Regularizer:** For each sample $\hat{x} \in \mathcal{T}$ the safe control signal is calculated by

$$\min_{u,r} \quad ||u - \pi_{nom}(x)||_2^2 \qquad s.t. \quad \mathcal{W}(\mathbf{S}_l)^T(f(\hat{x}) + g(\hat{x})u) + r \geq 0 \tag{10}$$

where $\mathcal{W} = \overline{W}$ for differentiable points and $\mathcal{W} = \tilde{W}$ defined as the subgradient at non-differentiable points. The regularizer $\mathcal{L}_f$ enforces the satisfaction of the constraint by inserting a positive relaxation term r in the constraint and minimizing r with a large penalization in the objective function. We have the loss $\mathcal{L}_f$ defined as $\mathcal{L}_f = ||u - \pi_{nom}(x)||_2^2 + r$. We use [45] to make this procedure differentiable, allowing us to employ the relaxation loss into the NCBF loss function design.

$\mathcal{L}_c$ **Regularizer:** $\mathcal{L}_c$ regularizer enforces the correctness of the NCBF. In particular, it enforces the $b_\theta(x)$ of safe samples $x \in \mathcal{T}_{\mathcal{I}}$ to be positive, and $b_\theta(x)$ of unsafe samples $\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}$ to be negative. Define $N_{\text{safe}} = |\mathcal{T}_{\mathcal{I}}|$ and $N_{\text{unsafe}} = |\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}|$, and with a small positive tuning parameter $\epsilon > 0$, the loss term $\mathcal{L}_c$ can be defined as

$$\mathcal{L}_c = a_1 \frac{1}{N_{\text{safe}}} \sum_{x \in \mathcal{T}_{\mathcal{I}}} [\epsilon - b_\theta(x)]_+ + a_2 \frac{1}{N_{\text{unsafe}}} \sum_{x \in \mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}} [\epsilon + b_\theta(x)]_+ \tag{11}$$

where $[\cdot]_+ = max(\cdot, 0)$. a_1 and a_2 are positive parameters controlling penalization strength of the violations of safe and unsafe samples.

$\mathcal{L}_{\mathcal{B}}$ **Regularizer:** We propose a novel regularizer to limit the number of boundary hyperplanes and hinges by penalizing the dissimilarity, i.e., $\mathbf{S}(\hat{x}_i)\Delta\mathbf{S}(\hat{x}_j)$ of boundary activation sets $\mathbf{S}(\hat{x}) \in \mathcal{B}$. However, the dissimilarity measure of boundary activation sets is inherently nondifferentiable. To address this issue the regularizer introduces the generalized sigmoid function $\sigma_k(z) = \frac{1}{1+\exp(-k\cdot z)}$ to compute the vector of smoothed activation defined as $\phi_{\sigma_k}(x) := [\sigma_k(z_{i,j}), \forall i, j \in \{1, \ldots, L\} \times \{1, \ldots, M_i\}]$. The $\mathcal{L}_{\mathcal{B}}$ regularizer conducts the following two steps to penalize dissimilarity.

In the first step, the regularizer identifies the training data in the boundary hyperplanes and hinges denoted as $\hat{x} \in \mathcal{T}_{\partial\mathcal{D}}$. The set is defined as $\mathcal{T}_{\partial\mathcal{D}} := \{\hat{x} : \hat{x} \in \overline{\mathcal{X}}(\mathbf{S}), \forall \mathbf{S} \in \mathcal{B}\}$. To further improve the efficiency, the regularizer approximates $\mathcal{T}_{\partial\mathcal{D}}$ with a range-based threshold ϵ on the output of the NCBF, i.e., $|b_\theta(\hat{x})| \leq \epsilon$.

The second step is to penalize the dissimilarity of $\mathbf{S} \in \mathcal{B}$. To avoid the potential pitfalls of enforcing similarity across the entire boundary $\mathcal{B}$, the regularizer employs an unsupervised learning approach to

https://doi.org/10.52202/079017-3214

group the training data into $N_{\mathbf{B}}$ clusters. We define the collection of the activation set in each cluster as $\mathbf{B}_i \subseteq \mathcal{B}$ and the collection in each cluster as $\mathcal{T}_{\mathbf{B}_i} := \{\hat{x} : \hat{x} \in \bigcup_{\mathbf{S}\in\mathbf{B}_i} \overline{\mathcal{X}}(\mathbf{S})\}$. The $\mathcal{L}_{\mathcal{B}}$ is then defined as follows, with an inner sum over all pairs of $\hat{x} \in \mathcal{T}_{\mathbf{B}_i}$ and an outer sum over all clusters.

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{N_{\mathbf{B}}} \sum_{\mathbf{B}_i \in \mathcal{B}} \frac{1}{|\mathcal{T}_{\mathbf{B}_i}|^2} \sum_{\hat{x}_i, \hat{x}_j \in \mathcal{T}_{\mathbf{B}_i}} \|\phi_{\sigma_k}(x_i) - \phi_{\sigma_k}(x_j)\|_2^2, \tag{12}$$

4 Verification

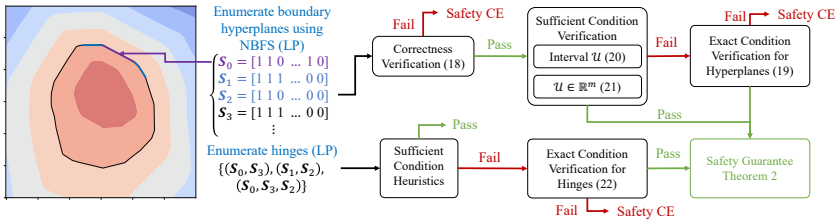


Figure 2: Overview of the Efficient Exact Verifier for ReLU NCBFs

In this section, we demonstrate the efficient exact verification of the given NCBF $b_\theta(x)$ to ensure the positive invariance of the set $\mathcal{D}$ under the NCBF-based control policy. In what follows, we propose an efficient enumeration and progressive verification moving from sufficient to exact conditions. The overview of the proposed approach is as shown in Fig. 2. SEEV decomposes the NCBF into hyperplanes and hinges and verifies each component hierarchically, with novel tractable sufficient conditions verified first and the more complex exact conditions checked only if the sufficient conditions fail. Given an NCBF $b_\theta(x)$, the verification of SEEV returns (i) a Boolean variable that is 'true' if the conditions of Proposition 1 are satisfied and 'false' otherwise, and (ii) a safety counterexample $\hat{x}$ that violates the conditions of Proposition 1 if the result is 'false'. Algorithm 1 presents an overview of the verification of SEEV. The algorithm consists of an enumeration stage to identify all boundary hyperplanes and hinges, and a verification stage to certify satisfaction of conditions in Proposition 1 for all hyperplanes and hinges.

Algorithm 1 Efficient Exact Verification

Input: $n, \mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$
Output: Verification Boolean result r, Categorized counterexample $\hat{x}_{ce}$
procedure EFFICIENT EXACT VERIFICATION($\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$)
    $\mathbf{S}_0 \leftarrow$ ENUMINIT($\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$) ▷ Initial Activation Set Identification, Section 4.1
    $\mathcal{S} \leftarrow$ NBFS($\mathbf{S}_0$) ▷ Activation Sets Enumeration, Section 4.1
    $r, \hat{x}_{ce}^{(c)} \leftarrow$ CorrectnessVerifier($\mathcal{S}$) ▷ Correctness Verification, Section 4.2
    $r, \hat{x}_{ce}^{(h)} \leftarrow$ HyperplaneVerifier($\mathcal{S}$) ▷ Hyperplane Verification, Section 4.2
    $\mathcal{V} \leftarrow$ HingeEnum($\mathcal{S}, n$) ▷ Hinges Enumeration, Section 4.1
    $r, \hat{x}_{ce}^{(g)} \leftarrow$ HingeVerifier($\mathcal{V}$) ▷ Hinge Verification, Section 4.2
    Return r, $\hat{x}_{ce}$

4.1 Enumeration of Hyperplanes and Hinges

The boundary enumeration identifies the boundary hyperplanes $\mathbf{S}_0 \in \mathcal{B}$. It initially identifies the initial boundary activation set $\mathbf{S}_0 \in \mathcal{B}$, enumerates all $\mathbf{S} \in \mathcal{B}$ by NBFS starting from $\mathbf{S}_0$, and finally enumerates all hinges consisting of the intersections of hyperplanes. The NBFS approach avoids over-approximation of the set of boundary hyperplanes that may be introduced by, e.g., interval propagation methods, and hence is particularly suited to deep neural networks.

In what follows, we assume that the unsafe region $\mathcal{X} \setminus \mathcal{C}$ and initial safe set $\mathcal{I}$ are connected, and use $\partial\mathcal{D}$ to refer to the connected component of the boundary of $\mathcal{D}$ that separates $\mathcal{I}$ and $\mathcal{X} \setminus \mathcal{C}$. This

assumption is without loss of generality since we can always repeat the following procedure for each connected component of $\mathcal{I}$ and $\mathcal{X} \setminus \mathcal{C}$.

Initial Activation Set Identification: First, we identify the initial boundary activation set $\mathbf{S}_0$. Given $\hat{x}_U \in \mathcal{X} \setminus \mathcal{C}$ and $\hat{x}_\mathcal{I} \in \mathcal{I}$, define a line segment

$$\tilde{L} = \left\{ x \in \mathbb{R}^n \mid x = (1-\lambda)\hat{x}_{\mathcal{X}\setminus\mathcal{C}} + \lambda \hat{x}_\mathcal{I},\ \lambda \in [0,1] \right\} \tag{13}$$

The following lemma shows the initial boundary activation set $\mathbf{S}_0$ can always be produced as $\mathbf{S}_0 = \mathbf{S}(\tilde{x})$ for some $\tilde{x} \in \tilde{L}$.

Lemma 1. *Given two sample points $\hat{x}_U$, such that $b(\hat{x}_U) < 0$, and $\hat{x}_\mathcal{I}$, such that $b(\hat{x}_U) > 0$, let $\tilde{L}$ denote the line segment connecting these two points. Then, there exists a point $\tilde{x} \in \tilde{L}$ with $b_\theta(\tilde{x}) = 0$.*

Lemma 1 follows from the intermediate value theorem and continuity of $b_\theta(x)$. In order to search for $\mathbf{S}_0$, we choose a sequence of $N_{\tilde{L}}$ points $x_1^0, \ldots, x_{N_{\tilde{L}}}^0 \in \tilde{L}$. For each x_i^0, we check to see if $\overline{\mathcal{X}}(\mathbf{S}(x_i^0)) \cap \partial\mathcal{D} \neq \emptyset$ by solving boundary linear program BoundaryLP($\mathbf{S}(x_i^0)$) (14) in Appendix A.4

Activation Sets Enumeration: We next describe Neural Breadth-First Search (NBFS) for enumerating all activation sets along the zero-level set, given an initial set $\mathbf{S}_0$. NBFS enumerates a collection of boundary hyperplanes denoted as $\mathcal{S}$ by repeating the following two steps.

Step 1: Given a set $\mathbf{S}$, NBFS identifies a collection of neighbor activation sets denoted as $\tilde{\mathcal{B}}$ as follows. For each (i,j) with $i = 1, \ldots, L$ and $j = 1, \ldots, M_i$, construct $\mathbf{S}'$ as $\mathbf{S}' = \begin{cases} \mathbf{S} \setminus (i,j), & (i,j) \in \mathbf{S}' \\ \mathbf{S} \cup (i,j), & (i,j) \notin \mathbf{S}' \end{cases}$. We check whether $\mathbf{S}' \in \mathcal{B}$ by solving the linear program USLP($\mathbf{S}, (i,j)$) (15) in Appendix A.4.

If there exists such a state x, then $\mathbf{S}'$ is added to $\tilde{\mathcal{B}}$. To further improve efficiency, we employ the simplex algorithm to calculate the hypercube that overapproximates the boundary hyperplane, denoted as $\mathcal{H}(\mathbf{S}) \supseteq \overline{\mathcal{X}}(\mathbf{S})$, and relax the last constraint of (14) to $x \in \mathcal{H}(\mathbf{S})$.

Step 2: For each $\mathbf{S}' \in \tilde{\mathcal{B}}$, NBFS determines if the activation region $\overline{\mathcal{X}}(\mathbf{S}')$ is on the boundary (i.e., satisfies $\overline{\mathcal{X}}(\mathbf{S}') \cap \partial\mathcal{D} \neq \emptyset$) by checking the feasibility of $BoundaryLP(\mathbf{S}')$. If there exists such a state x, then $\mathbf{S}'$ is added to $\mathcal{S}$. This process continues in a breadth-first search manner until all such activation sets on the boundary have been enumerated. When this phase terminates, $\mathcal{S} = \mathcal{B}$. Detailed procedure is described as Algorithm 3 in Appendix A.3.

Hinge Enumeration: The verifier of SEEV enumerates a collection $\mathcal{V}$ of boundary hinges, where each boundary hinge $\mathbf{V} \in \mathcal{V}$ is a subset $\mathbf{S}_1, \ldots, \mathbf{S}_r$ of $\mathcal{B}$ with $\overline{\mathcal{X}}(\mathbf{S}_1) \cap \overline{\mathcal{X}}(\mathbf{S}_r) \cap \partial\mathcal{D} \neq \emptyset$.

Given $\mathcal{S}_i \subseteq \mathcal{B}$ and $\mathbf{S}$, hinge enumeration filter the set of neighbor activation sets of $\mathbf{S}$ defined as $\mathcal{N}_\mathbf{S}(\mathcal{S}_i) := \{\mathbf{S}' : \mathbf{S}' \Delta \mathbf{S} = 1, \mathbf{S}' \in \mathcal{S}_i\}$. Then, hinge enumeration identifies hinges $\mathbf{V}$ by solving linear program HingeLP($\mathcal{N}_\mathbf{S}^{(d)}(\mathcal{S}_i)$) (16) in Appendix A.4. If $\exists x$, hinge enumeration includes the hinge into the set $\mathcal{V} \cup \mathbf{V}$. The efficiency can be further improved by leveraging the sufficient condition verification proposed in Section 4.2. The following result describes the completeness guarantees of $\mathcal{S}$ and $\mathcal{V}$ enumerated in Line 5 and 8 of Algorithm 1.

Proposition 2. *Let $\mathcal{S}$ and $\mathcal{V}$ denote the output of Algorithm 1. Then the boundary $\partial\mathcal{D}$ satisfies $\partial\mathcal{D} \subseteq \bigcup_{\mathbf{S}\in\mathcal{S}} \overline{\mathcal{X}}(\mathbf{S})$. Furthermore, if $\mathcal{S}$ is complete and $\left(\bigcap_{i=1}^r \overline{\mathcal{X}}(\mathbf{S}_i)\right) \cap \{x : b(x) = 0\} \neq \emptyset$, then $\{\mathbf{S}_1, \ldots, \mathbf{S}_r\} \in \mathcal{V}$.*

The proof is omitted due to the space limit. A detailed proof is provided in Appendix A.2

4.2 Efficient Verification

The efficient verification component takes the sets of boundary hyperplanes $\mathcal{S}$ and boundary hinges $\mathcal{V}$ and checks that the conditions of Proposition 1 hold for each of them. As pointed out after Proposition 1, the problem of searching for safety counterexamples can be decomposed into searching for correctness, hyperplane, and hinge counterexamples. In order to maximize the efficiency of our approach, we first consider the least computationally burdensome verification task, namely, searching for correctness counterexamples. We then search for hyperplane counterexamples, followed by hinge counterexamples.

 https://doi.org/10.52202/079017-3214

Correctness Verification: The correctness condition ((7)) can be verified for boundary hyperplane $\overline{\mathcal{X}}(\mathbf{S})$ by solving the nonlinear program (17) in Appendix A.5. When $h(x)$ is convex, (17) can be solved efficiently. Otherwise, dReal can be used to check satisfiability of (17) in a tractable runtime.

Hyperplane Verification: Hyperplane counterexamples can be identified by solving the optimization problem (18) in Appendix A.5. Solving (18) can be made more efficient when additional structures on the input set $\mathcal{U}$ and dynamics f and g are present. Consider a case $\mathcal{U} = \{D\omega : ||\omega||_\infty \leq 1\}$. In this case, the problem reduces to the nonlinear program (19) in Appendix A.5. If $f(x)$ and $g(x)$ are linear in x, then the problem boils down to a linear program. If bounds on the Lipschitz coefficients of f and g are available, then they can be used to derive approximations of (19).

If $\mathcal{U} = \mathbb{R}^m$, then by [23, Corollary 1], the problem can be reduced to the nonlinear program (20) in Appendix A.5. If $g(x)$ is a constant matrix G, then safety is automatically guaranteed if $\overline{W}(\mathbf{S})^T G \neq 0$. If $f(x)$ is linear in x as well, then (20) is a linear program.

Hinge Verification: The hinge $\mathbf{V} = \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ can be certified by solving the nonlinear optimization problem (21) in Appendix A.5. In practice, simple heuristics are often sufficient to verify safety of hinges without resorting to solving (21). If $\overline{W}(\mathbf{S})^T f(x) > 0$ for all $x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S}_r)$, then the control input $u = \mathbf{0}$ suffices to ensure safety. Furthermore, if $\mathcal{U} = \mathbb{R}^m$ and there exists $i \in \{1, \ldots, m\}$ and $s \in \{0, 1\}$ such that $\text{sign}((\overline{W}(\mathbf{S}_l)^T g(x))_i) = s$ for all $x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \overline{\mathcal{X}}(\mathbf{S}_r)$ and $l = 1, \ldots, r$, then u can be chosen as $u_i = Ks$ for some sufficiently large $K > 0$ to ensure that the conditions of Proposition 1.

Safety Guarantee: The safety guarantees of our proposed approach are summarized in Theorem 2.

Theorem 2. *Given a NCBF $b_\theta(x)$, $b_\theta(x)$ is a valid NCBF if it passes the verification of Algorithm 1 using dReal to solve the optimization problems* (17)*,* (18)*, and* (21)*.*

The proof is derived from completeness of the enumeration in Algorithm 1 and dReal. A detailed proof can be found in Appendix A.6

5 Experiments

In this section, we evaluate the proposed SEEV in regularizer efficacy and verification efficiency. We also demonstrated the improved performance with counter-example guidance in the synthesis framework, whose results are detailed in B.2. We experiment on four systems, namely Darboux, obstacle avoidance, hi-ord$_8$, and spacecraft rendezvous. The experiments run on a workstation with an Intel i7-11700KF CPU, and an NVIDIA GeForce RTX 3080 GPU. We include experiment settings in Appendix B.1 and hyperparameters settings in Table 4 in Appendix B.3.

5.1 Experiment Setup

Darboux: We consider the Darboux system [46], a nonlinear open-loop polynomial system with detailed settings presented in Appendix B.1

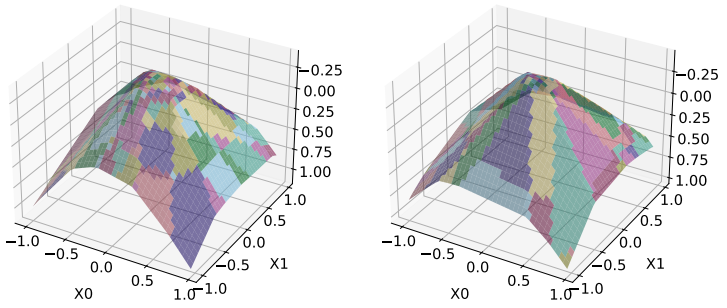
Obstacle Avoidance (OA): We evaluate our proposed method on a controlled system [47]. We consider Unmanned Aerial Vehicles (UAVs) avoiding collision with a tree trunk. The system state consists of a 2-D position and aircraft yaw rate $x := [x_1, x_2, \psi]^T$. The system is manipulated by the yaw rate input u with detailed settings presented in Appendix B.1.

Spacecraft Rendezvous (SR): We evaluate our approach on a spacecraft rendezvous problem from [48]. The system state is $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$ and control input is $u = [u_x, u_y, u_z]^T$ with with detailed settings presented in Appendix B.1.

hi-ord$_8$: We evaluate our approach on an eight-dimensional system that first appeared in [21] to evaluate the scalability of the proposed method. Detailed settings can be found in Appendix B.1

5.2 Regularizer Efficacy Evaluation

Table 2 and Figure 3 illustrates the impact of regularization on the CBF boundary's activation sets. Table 2 compares various configurations, where n denotes the input dimensions, L represents the number of layers, and M indicates the number of hidden units per layer. N_o and N_r are the number of



(a) $r = 0$ (b) $r = 50$

Figure 3: Effects of boundary regularization (r) on activation sets along the boundary. The figures show the results from a neural network with 4 layers of 8 hidden units, applied to the Spacecraft case. The surface represents the first two dimensions with the last four dimensions fixed at 0. Increasing r results in more organized boundary activation sets.

Table 1: Comparison of N the number of boundary hyperplanes and C coverage of the safe region $\mathcal{D}$ of NCBF trained with and without boundary hyperplane regularizer denoted with subscripts $_r$ and $_o$.

Case	n	L	M	N_o	C_o	$N_{r=1}$	$\rho_{r=1}$	$N_{r=10}$	$\rho_{r=10}$	$N_{r=50}$	$\rho_{r=50}$
OA	3	2	8	26	89.46%	25	0.996	23.3	0.994	**13.3**	1.006
	3	2	16	116	83.74%	119	1.012	111	1.005	**98**	1.055
	3	4	8	40	91.94%	38	0.988	36	0.993	**13**	0.937
	3	4	16	156	87.81%	170	0.971	147	1.003	**64**	1.038
SR	6	2	8	2868	98.58%	2753	1	1559	1	**418**	1
	6	4	8	6371	98.64%	6218	1	3055	1	**627**	1
	6	2	16	N/A	N/A	204175	N/A	68783	N/A	**13930**	N/A

hyperplanes along the zero-level boundary of the CBF without and with regularization, respectively, with r indicating the regularization strength. C_o captures the CBF's coverage of the safe region, while $\rho_{(\cdot)} = C_{(\cdot)}/C_o$ represents the safety coverage ratio relative to the unregularized CBF. Notably, "N/A" entries indicate configurations where training a fully verifiable network was infeasible due to the excessive number of boundary hyperplanes, which leads the verification process to time out.

The results demonstrate that regularization effectively reduces the number of activation sets along the CBF boundary without compromising the coverage of the safe region. The efficiency is especially improved in cases with a greater number of hidden layers, where the unregularized model results in a significantly higher number of hyperplanes. For instance, in the SR case with $n = 6$, $L = 4$, and $M = 8$, the regularization reduces $N_{r=50}$ to 627 from $N_o = 6218$, maintaining the same safety coverage ($\rho_{r=50} = 1$). See Appendix B.4 for hyperparameter sensitivity analysis.

Figure 3 illustrates the level sets of two CBFs trained for the SR case with $n = 6$, $L = 4$, and $M = 8$. These level sets are extracted from the first two dimensions with the rest set to zero. Each colored patch represents an activation pattern. The regularizer organizes the activation sets around the boundary, reducing unnecessary rapid changes and thereby enhancing the verification efficiency.

5.3 Efficient Verification Evaluation

The results presented in Table 2 illustrate a significant improvement in verification efficiency for Neural Control Barrier Functions NCBFs using the proposed method. In the table t_h represents the time spent searching for hyperplanes containing the CBF boundary and verifying CBF sufficient conditions on these boundaries, and t_g represents the time spent in hinge enumeration and verification. The total time, T, is the sum of t_h and t_g. We compare our approach with three baselines, exact verification [23], SMT-based verification [21] with dReal and Z3. Baseline methods' run times are represented by Baseline [23], dReal, and Z3.

In the Darboux cases, our method achieves verification in 2.5 seconds and 3.3 seconds for $M = 256$ and $M = 512$ respectively, whereas baseline methods take substantially longer, with Baseline [23]

 https://doi.org/10.52202/079017-3214

Table 2: Comparison of verification run-time of NCBF in seconds. We denote the run-time as 'UTD' when the method is unable to be directly used for verification.

Case	n	L	M	N	t_h	t_g	SEEV	Baseline [23]	dReal	Z3
Darboux	2	2	256	15	2.5s	0	2.5s	315s	>3h	>3h
	2	2	512	15	3.3s	0	3.3s	631s	>3h	>3h
OA	3	2	16	86	0.41s	0	0.41s	16.0s	>3h	>3h
	3	4	8	15	0.39	0	0.39	16.1s	>3h	>3h
	3	4	16	136	0.65s	0	0.65s	36.7s	>3h	>3h
	3	1	128	5778	20.6s	0	20.6s	207s	>3h	>3h
hi-ord$_8$	8	2	8	73	0.54s	0	0.54s	>3h	>3h	>3h
	8	2	16	3048	11.8s	0	11.8s	>3h	>3h	>3h
	8	4	16	3984	22.4s	0	22.4s	>3h	>3h	>3h
SR	6	2	8	2200	7.1s	2.7s	9.8s	179s	UTD	UTD
	6	4	8	4918	45.8s	14.3s	60.1s	298.7s	UTD	UTD

taking 315 seconds and 631 seconds, and both dReal and Z3 taking more than 3 hours. Similarly, in the OA cases, our method's run times range from 0.39 seconds to 20.6 seconds, faster than the baseline methods. In the more higher dimensional systems high-ord$_8$ and SR, our method significantly outperforms Baseline [23]. Specifically, in high-ord$_8$ our methods finishes within 22.4 seconds while Baseline [23], dReal and Z3 times out, due to the need to enumerate the 8-dimensional input space. For the SR case, SEEV's run time are 9.8 seconds and 60.1 seconds, beating Baseline [23] which takes 179 seconds and 298.7 seconds respectively. Neural barrier certificate based dReal and Z3 are able to directly applicable since they require an explicit expression of the controlled feedback system. However, the SR system is manipulated by an NCBF-based safe controller that is nontrivial to derive an explicit expression.

Note that hinge enumeration and certification may be time-consuming procedure, since they involve enumerating all combinations of hyperplanes. However, the results from Table 2 show that the certification can be completed on most hyperplanes with sufficient condition verification in Section 4.2, greatly improving the overall run time.

6 Conclusion

This paper considered the problem of synthesizing and verifying NCBFs with ReLU activation function in an efficient manner. Our approach is guided by the fact that the main contribution to the computational cost of verifying NCBFs is enumerating and verifying safety of each piecewise-linear activation region at the safety boundary. We proposed Synthesis with Efficient Exact Verification (SEEV), which co-designs the synthesis and verification components to enhance scalability of the verification process. We augment the NCBF synthesis with a regularizer that reduces the number of piecewise-linear segments at the boundary, and hence reduces the total workload of the verification. We then propose a verification approach that efficiently enumerates the linear segments at the boundary and exploits tractable sufficient conditions for safety.

Limitations: The method proposed in this paper mitigated the scalability issue. However, the synthesis and verification of NCBFs for higher-dimensional systems is challenging. Exact verification of non-ReLU NCBFs, which lack ReLU's simple piecewise linearity, remains an open problem.

Acknowledgements and Disclosure of Funding

This research was partially supported by NSF grant CNS-1941670, CMMI-2418806, AFOSR grant FA9550-22-1-0054, NSF Career CCF 2047034, NSF CCF DASS 2217723, NSF AI Institute CCF 2112665, and Amazon Research Award.

References

[1] Shao-Chen Hsu, Xiangru Xu, and Aaron D Ames. Control barrier function based quadratic programs with application to bipedal robotic walking. In *2015 American Control Conference (ACC)*, pages 4542–4548. IEEE, 2015.

[2] Devansh R Agrawal and Dimitra Panagou. Safe control synthesis via input constrained control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6113–6118. IEEE, 2021.

[3] Xiangru Xu, Jessy W Grizzle, Paulo Tabuada, and Aaron D Ames. Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, 15(3):1216–1229, 2017.

[4] Joseph Breeden and Dimitra Panagou. High relative degree control barrier functions under input constraints. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6119–6124. IEEE, 2021.

[5] Hongkai Dai and Frank Permenter. Convex synthesis and verification of control-lyapunov and barrier functions with input constraints. In *2023 American Control Conference (ACC)*, pages 4116–4123. IEEE, 2023.

[6] Shucheng Kang, Yuxiao Chen, Heng Yang, and Marco Pavone. Verification and synthesis of robust control barrier functions: Multilevel polynomial optimization and semidefinite relaxation, 2023.

[7] Nicholas Rober, Michael Everett, Songan Zhang, and Jonathan P How. A hybrid partitioning strategy for backward reachability of neural feedback loops. In *2023 American Control Conference (ACC)*, pages 3523–3528. IEEE, 2023.

[8] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

[9] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. *arXiv preprint arXiv:2310.15478*, 2023.

[10] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural Lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[11] Manan Tayal, Hongchao Zhang, Pushpak Jagtap, Andrew Clark, and Shishir Kolathaya. Learning a formally verified control barrier function in stochastic environment. *arXiv preprint arXiv:2403.19332*, 2024.

[12] Kehan Long, Cheng Qian, Jorge Cortés, and Nikolay Atanasov. Learning barrier functions with memory for robust safe navigation. *IEEE Robotics and Automation Letters*, 6(3):4931–4938, 2021.

[13] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Makram Chahine, Alexander Amini, Xiao Li, and Daniela Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 2023.

[14] Hengjun Zhao, Xia Zeng, Taolue Chen, Zhiming Liu, and Jim Woodcock. Learning safe neural network controllers with barrier certificates. *Formal Aspects of Computing*, 33:437–455, 2021.

[15] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31:4939–4948, 2018.

[16] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.

 https://doi.org/10.52202/079017-3214

[17] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32:9835–9846, 2019.

[18] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021.

[19] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.

[20] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of ReLU networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 26591–26604, 2022.

[21] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. Fossil: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[22] Alec Edwards, Andrea Peruffo, and Alessandro Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. *arXiv preprint arXiv:2311.09793*, 2023.

[23] Hongchao Zhang, Junlin Wu, Yevgeniy Vorobeychik, and Andrew Clark. Exact verification of relu neural control barrier functions. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in Neural Information Processing Systems*, 30, 2017.

[25] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*, 2021.

[26] Zhizhen Qin, Tsui-Wei Weng, and Sicun Gao. Quantifying safety of learning-based self-driving control using almost-barrier functions. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12903–12910. IEEE, 2022.

[27] Simin Liu, Changliu Liu, and John Dolan. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, pages 1970–1980. PMLR, 2023.

[28] Stephen Prajna, Ali Jadbabaie, and George J Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[29] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[30] Weiye Zhao, Tairan He, Tianhao Wei, Simin Liu, and Changliu Liu. Safety index synthesis via sum-of-squares programming. In *2023 American Control Conference (ACC)*, pages 732–737. IEEE, 2023.

[31] Michael Schneeberger, Florian Dörfler, and Silvia Mastellone. SOS construction of compatible control Lyapunov and barrier functions. *arXiv preprint arXiv:2305.01222*, 2023.

[32] Andrew Clark. Verification and synthesis of control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6105–6112. IEEE, 2021.

[33] Claudio Ferrari, Mark Niklas Muller, Nikola Jovanovic, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*, 2022.

[34] Patrick Henriksen and Alessio Lomuscio. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In *IJCAI*, pages 2549–2555, 2021.

[35] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems*, 35:1656–1670, 2022.

[36] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

[37] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The Marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*, pages 443–452. Springer, 2019.

[38] Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pages 1–11, 2020.

[39] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

[40] Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 370–388. Springer International Publishing, 2021.

[41] Mahathi Anand and Majid Zamani. Formally verified neural network control barrier certificates for unknown systems. *IFAC-PapersOnLine*, 56(2):2431–2436, 2023.

[42] Yixuan Wang, Chao Huang, Zhaoran Wang, Zhilu Wang, and Qi Zhu. Design-while-verify: correct-by-construction control learning with verification in the loop. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 925–930, 2022.

[43] Xinyu Wang, Luzia Knoedler, Frederik Baymler Mathiesen, and Javier Alonso-Mora. Simultaneous synthesis and verification of neural control barrier functions through branch-and-bound verification-in-the-loop training. In *2024 European Control Conference (ECC)*, pages 571–578. IEEE, 2024.

[44] Franco Blanchini and Stefano Miani. *Set-Theoretic Methods in Control*, volume 78. Springer, 2008.

[45] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

[46] Xia Zeng, Wang Lin, Zhengfeng Yang, Xin Chen, and Lilei Wang. Darboux-type barrier certificates for safety verification of nonlinear hybrid systems. In *Proceedings of the 13th International Conference on Embedded Software*, pages 1–10, 2016.

[47] Andrew J Barry, Anirudha Majumdar, and Russ Tedrake. Safety verification of reactive controllers for uav flight in cluttered environments using barrier certificates. In *2012 IEEE International Conference on Robotics and Automation*, pages 484–490. IEEE, 2012.

[48] Christopher Jewison and R Scott Erwin. A spacecraft benchmark problem for hybrid control and estimation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3300–3305. IEEE, 2016.

 https://doi.org/10.52202/079017-3214

A Supplement

A.1 Definition of $\overline{W}_{ij}(\mathbf{S})$ and $\overline{r}_{ij}(\mathbf{S})$

In what follows, we define vectors $\overline{W}_{ij}(\mathbf{S}) \in \mathbb{R}^n$ and scalars $\overline{r}_{ij}(\mathbf{S}) \in \mathbb{R}$ such that $z_j^{(i)} = \overline{W}_{ij}(\mathbf{S})^T x + \overline{r}_{ij}(\mathbf{S})$. The weight and bias of the input layer is defined by

$$\overline{W}_{1j}(\mathbf{S}) = \begin{cases} W_{1j}, & (1,j) \in \mathbf{S} \\ 0, & \text{else} \end{cases} \qquad \overline{r}_{1j}(\mathbf{S}) = \begin{cases} r_{1j}, & (1,j) \in \mathbf{S} \\ 0, & \text{else} \end{cases}$$

Proceeding inductively, define $\overline{W}_i(\mathbf{S}) \in \mathbb{R}^{M_i \times n}$ to be a matrix with columns $\overline{W}_{ij}(\mathbf{S})$ for $j = 1, \ldots, M_i$ and $\overline{r}_i(\mathbf{S}) \in \mathbb{R}^{M_i}$ to be a vector with elements $\overline{r}_{ij}(\mathbf{S})$. We then define

$$\overline{W}_{ij}(\mathbf{S}) = \begin{cases} \overline{W}_{i-1}(\mathbf{S}) W_{ij}, & (i,j) \in \mathbf{S} \\ 0, & \text{else} \end{cases} \qquad \overline{r}_{ij}(\mathbf{S}) = \begin{cases} W_{ij}^T \overline{r}_{i-1}(\mathbf{S}) + r_{ij}, & (i,j) \in \mathbf{S} \\ 0, & \text{else} \end{cases}$$

At the last layer, let $\overline{W}(\mathbf{S}) = W_L(\mathbf{S})\Omega$ and $\overline{r}(\mathbf{S}) = \Omega^T r_L(\mathbf{S}) + \psi$, so that $y = \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S})$ if $\mathbf{S} = \{(i,j) : z_j^{(i)} \geq 0\}$.

A.2 Proof of Proposition 2

Proof. Suppose that $x' \in \partial\mathcal{D} \setminus \left(\bigcup_{\mathbf{S} \in \mathcal{S}} \overline{\mathcal{X}}(\mathbf{S})\right)$, and let x denote the state on $\partial\mathcal{D}$ found by Line 5 of Algorithm 1. Since $\partial\mathcal{D}$ is connected, there exists a path γ with $\gamma(0) = x$ and $\gamma(1) = x'$ contained in $\partial\mathcal{D}$. Let $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_K$ denote a sequence of activation sets with $\mathbf{S}_0 \in \mathbf{S}(x)$, $\mathbf{S}_K \in \mathbf{S}(x')$, and $\mathbf{S}_0$ equal to the set computed at Line 5 of Algorithm 1. Then there exists $i \geq 2$ such that $\mathbf{S}_{i-1} \in \mathcal{S}$ and $\mathbf{S}_i \notin \mathcal{S}$, and there exists t' such that $\gamma(t') \in \overline{\mathcal{X}}(\mathbf{S}_{i-1}) \cap \overline{\mathcal{X}}(\mathbf{S}_i) \cap \{x : b(x) = 0\}$. We then have that $\mathbf{T}(\mathbf{S}_{i-1}, \mathbf{S}_i)$ is a subset of the set $\mathbf{T}$, and hence $\mathbf{S}_i$ will be identified and added to $\mathcal{S}$ at Line 5 of Algorithm 1.

Now, suppose that $\mathbf{S}_1, \ldots, \mathbf{S}_r \in \mathcal{S}$ is complete and $\left(\bigcap_{i=1}^r \overline{\mathcal{X}}_0(\mathbf{S}_i)\right) \cap \{x : b(x) = 0\} \neq \emptyset$. Let $\mathbf{T} = \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)$. Then $\mathbf{T}$ is a subset of the sets $\mathbf{S}_1, \ldots, \mathbf{S}_r \in \mathcal{S}$. Since the intersection of the $\overline{\mathcal{X}}(\mathbf{S})$ sets with $\{x : b(x) = 0\}$ is nonempty, $\{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ is added to $\mathcal{V}$. □

A.3 Algorithm for Verification

SEEV identifies the initial activation set by conducting a BoundaryLP-based binary search as shown in Algorithm. 2. The algorithm presents a procedure to identify a hyperplane characterized by an initial activation set $\mathbf{S}_0$ that may contain the boundary $\partial\mathcal{D}$. It iterates over pairs of sample states from the unsafe training set $\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}$ and the safe training set $\mathcal{T}_{\mathcal{I}}$. For each pair $(\hat{x}_{\mathcal{X}\setminus\mathcal{C}}, \hat{x}_{\mathcal{I}})$, the algorithm initializes the left and right points of a search interval. It then performs a binary search by repeatedly computing the midpoint x_{mid} and checking the feasibility of the boundary linear program BoundaryLP(x_{mid}). The algorithm terminates when BoundaryLP(x_{mid}) is feasible, returning the activation set $\mathbf{S}(x_{\text{mid}})$ as $\mathbf{S}_0$.

SEEV utilizes NBFS for enumerating all activation sets along the zero-level set in a breadth-first search manner, starting from an initial set $\mathbf{S}_0$. The algorithm initializes a queue $\mathcal{Q}$ and a set $\mathcal{S}$ with $\mathbf{S}_0$. While $\mathcal{Q}$ is not empty, it dequeues an activation set $\mathbf{S}$ and checks if the set $\mathbf{S} \in \mathcal{B}$ by solving the boundary linear program BoundaryLP($\mathbf{S}$). If so, $\mathbf{S}$ is identified and added to $\mathcal{S}$. The algorithm then explores its neighboring activation sets by flipping each neuron activation (i,j) in $\mathbf{S}$. For each flip, it solves the unstable neuron linear program USLP($\mathbf{S}, (i,j)$). If USLP is feasible, the new activation set $\mathbf{S}'$ obtained by flipping (i,j) is added to $\mathcal{Q}$ for further search on its neighbors. This process continues until all relevant activation sets are explored, resulting in a set $\mathcal{S}$ that contains activation sets potentially on the boundary.

SEEV enumerates all hinges $\mathbf{V} \in \mathcal{V}$ with Algorithm 4. Algorithm 4 outlines a method to enumerate all feasible hinge hyperplanes formed by combinations of activation sets up to size n. The algorithm takes as input a set of activation sets $\mathcal{S}$ and a maximum combination size n. It initializes an empty list to store feasible hinges. For each combination size k from 2 to n, the algorithm iterates over all activation sets in $\mathcal{S}$. For each activation set $\mathbf{S}$, it generates candidate combinations $\mathcal{V}_c$ based on adjacency—either the set of adjacent activation sets when $k = 2$, or the feasible combinations

Algorithm 2 Binary Search for $\mathbf{S}_0$

Input: $\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$
Output: initial activation set $\mathbf{S}_0$
procedure ENUMINIT($\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$)
    for $\hat{x}_{\mathcal{X}\setminus\mathcal{C}} \in \mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}$ and $\hat{x}_{\mathcal{I}} \in \mathcal{T}_{\mathcal{I}}$ **do** ▷ Loop over all pairs of samples
        $x_{left}, x_{right} \leftarrow \hat{x}_{\mathcal{X}\setminus\mathcal{C}}, \hat{x}_{\mathcal{I}}$
        $x_{mid} = 0.5(x_{left} + x_{right})$
        while $x_{left} < x_{right}$ **do**
            if BoundaryLP(x_{mid}) **then** ▷ Check if the BoundaryLP is feasible
                Return $\mathbf{S}(x_{mid})$ ▷ Return $\mathbf{S}(x_{mid})$ as $\mathbf{S}_0$
            if $\overline{W}(\mathbf{S}(x'))x_{mid} + \overline{r}(\mathbf{S}(x')) \geq 0$ **then** ▷ Check if $b(x_{mid}) > 0$
                $x_{right} \leftarrow x_{mid}$ ▷ Update the right point
            else
                $x_{left} \leftarrow x_{mid}$ ▷ Update the left point

Algorithm 3 Enumerate Activated Sets

Input: $\mathbf{S}_0$
Output: Set of activation sets $\mathcal{S}$
procedure NBFS($\mathbf{S}_0$)
    Initialize queue $\mathcal{Q}$ with initial activation set $\mathbf{S}_0$
    Initialize sets $\mathcal{S}$ with $\mathbf{S}_0$
    while queue $\mathcal{Q}$ is not empty **do**
        Dequeue $\mathbf{S}$ from $\mathcal{Q}$ ▷ Dequeue the first activation set from the queue
        if BoundaryLP($\mathbf{S}$) is feasible **then** ▷ Check if $\mathbf{S}$ is on a boundary activation set
            if $\mathbf{S} \notin \mathcal{S}$ **then** ▷ If $\mathbf{S}$ is not already in $\mathcal{S}$
                Add $\mathbf{S}$ to $\mathcal{S}$
            for $(i, j) \in \{1, \ldots, L\} \times \{1, \ldots, M_i\}$ **do** ▷ For activation (i, j) of each neurons
                if USLP($\mathbf{S}, (i, j)$) **then** ▷ Check if the neighbor may contain zero-level set
                    Create $\mathbf{S}'$ by flipping activation (i, j)
                    Add $\mathbf{S}'$ to $\mathcal{Q}$ ▷ Add adjacent activation set $\mathbf{S}'$ to the queue
    Return $\mathcal{S}$

from the previous iteration when $k > 2$. It then checks each candidate combination $\mathcal{S}_c$ by verifying adjacency and solving the hinge linear program HingeLP($\mathcal{S}_c \cup \{\mathbf{S}\}$). If the HingeLP is feasible, the combination is added to the list of feasible hinges $\mathcal{V}$. This process continues until all combinations up to size n have been examined, resulting in a comprehensive list of feasible hinge hyperplanes that are essential for understanding the intersections of activation regions.

A.4 Linear Programs for Enumeration

Given a state x_i^0, we define the activation set is $\mathbf{S} = \tau_S(x_i^0)$. To determine if the activation set $\mathbf{S} \in \mathcal{B}$, we solve a linear program referred to as the boundary linear program. The program checks the existence of a state $x \in \overline{\mathcal{X}}(\mathbf{S}(x_i^0))$ that satisfies $\overline{W}(\mathbf{S}(x_i^0))^T x + \overline{r}(\mathbf{S}(x_i^0)) = 0$. The boundary linear program (BoundaryLP($\mathbf{S}(x_i^0)$)) is defined as follows.

$$\text{BoundaryLP}(\mathbf{S}(x_i^0)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\mathbf{S}(x_i^0))^T x + \overline{r}(\mathbf{S}(x_i^0)) = 0 \\ & x \in \overline{\mathcal{X}}(\mathbf{S}(x_i^0)) \end{cases} \tag{14}$$

NBFS conducts its search in a breadth-first manner. To determine if a neighboring activation set, resulting from a flip in its (i, j) neuron, may contain the boundary, NBFS solves a linear program, referred to as the unstable neuron linear program of USLP($\mathbf{S}, (i, j)$). This linear program checks the existence of a state $x \in \overline{\mathcal{X}}(\mathbf{S}) \cap \{x : W_{ij}x + r_{ij} = 0\}$ that satisfies $\overline{W}(\mathbf{S})x + \overline{r}(\mathbf{S}) = 0$. The

https://doi.org/10.52202/079017-3214

Algorithm 4 Enumerate Hinges

Input: $\mathcal{S}, n$
Output: $\mathcal{V}$
procedure HINGEENUM($\mathcal{S}, n$)
    Initialize hinge_list as an empty list
    for k from 2 to n **do** ▷ Outer loop to iterate over combination sizes
        for $\mathbf{S} \in \mathcal{S}$ **do** ▷ Iterate over all activation set
            if $k = 2$ **then**
                $\mathcal{V}_c \leftarrow \mathcal{N}_{\mathbf{S}}(\mathcal{S})$
            else
                $\mathcal{V}_c \leftarrow \mathcal{V}^{(k-1)}$
            for each combination $\mathcal{S}_c \in \mathcal{V}_c$ and $\mathbf{S} \notin \mathcal{S}_c$ **do** ▷ Iterate over combinations
                if $\mathbf{S}$ and $\mathcal{S}_c$ are not adjacent **then**
                    Continue ▷ skip non-adjacent combinations
                if HingeLP($\mathcal{S}_c \cup \{\mathbf{S}\}$) **then**
                    Add $\mathbf{V} \leftarrow \mathcal{S}_c \cup \{\mathbf{S}\}$ to $\mathcal{V}^{(k)}$ ▷ Add to set of corresponding k
        Add $\mathcal{V}^{(k)}$ to $\mathcal{V}$
    return hinge_list

unstable neuron linear program is defined as follows.

$$\text{USLP}(\mathbf{S}, (i,j)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0 \\ & W_{ij}(\mathbf{S})^T x + r_{ij}(\mathbf{S}) = 0 \\ & x \in \overline{\mathcal{X}}(\mathbf{S}) \end{cases} \tag{15}$$

Finally, we enumerate all hinges by solving the hinge linger program HingeLP($\mathcal{S}_i$), defined as follows.

$$\text{HingeLP}(\mathcal{N}_{\mathbf{S}}^{(d)}(\mathcal{S}_i)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, \quad \forall \mathbf{S} \in \mathcal{N}_{\mathbf{S}}(\mathcal{S}_i) \\ & x \in \mathcal{X}(\mathbf{S}), \quad \forall \mathbf{S} \in \mathcal{N}_{\mathbf{S}}(\mathcal{S}_i) \end{cases} \tag{16}$$

A.5 Nonlinear Programs for Verification

The correctness condition ((7)) can be verified for boundary hyperplane $\overline{\mathcal{X}}(\mathbf{S})$ by solving the nonlinear program

$$\begin{array}{ll} \min_x & h(x) \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{17}$$

If we found some state x such that $h(x) < 0$ and $\overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0$, it indicates that the state lies on the boundary of the set $\mathcal{D}$ but not inside the set $\mathcal{C}$, i.e., $x \in \partial\mathcal{D} \notin \mathcal{C}$. This implies a violation of the condition $\mathcal{D} \subseteq \mathcal{C}$.

Hyperplane counterexamples can be identified by solving the optimization problem

$$\begin{array}{ll} \min_x & \max\{\overline{W}(\mathbf{S})^T(f(x) + g(x)u) : u \in \mathcal{U}\} \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{18}$$

Consider a case $\mathcal{U} = \{D\omega : ||\omega||_\infty \leq 1\}$. The bounded input set $\mathcal{U}$ allows us to replace the maximization over u with L-1 norm term $\|\overline{W}(\mathbf{S})^T g(x) D\|_1$, which simplifies the computational complexity. In this case, the problem reduces to the nonlinear program

$$\begin{array}{ll} \min\limits_x & \overline{W}(\mathbf{S})^T f(x) + \|\overline{W}(\mathbf{S})^T g(x) D\|_1 \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{19}$$

If $\mathcal{U} = \mathbb{R}^m$, then by [23, Corollary 1], the problem can be reduced to the nonlinear program

$$\begin{array}{ll} \min_x & \overline{W}(\mathbf{S})^T f(x) \\ \text{s.t.} & \overline{W}(\mathbf{S})^T g(x) = 0, \overline{W}(\mathbf{S}) x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{20}$$

The hinge $\mathbf{V} = \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ can be certified by solving the nonlinear optimization problem

$$\begin{array}{ll} \min_x & \max\{\overline{W}(\mathbf{S}_l)^T(f(x)+g(x)u) : l = 1, \ldots, r, u \text{ satisfies (4)–(5)}\} \\ \text{s.t.} & x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S}_r), \overline{W}(\mathbf{S}_1)^T x + \overline{r}(\mathbf{S}_1) = 0 \end{array} \tag{21}$$

A.6 Proof of Theorem 2

Proof. By Lemma 1, the initial boundary activation set $\mathbf{S}_0$ is ensured to be identified by the verification of SEEV in Line 4 Algorithm. 1. Given $\mathbf{S}_0$ and the enumeration in Line 5 and 6 Algorithm. 1 being complete, the completeness of $\mathcal{S}$ and $\mathcal{V}$ is guaranteed by Proposition 2. By dReal solving the equivalent NLPs, the conditions of Proposition 1, are satisfied. Therefore, $b_\theta(x)$ is a valid NCBF $\square$

B Experiments

B.1 Experiment Settings

Experiment Settings of Darboux: The dynamic model of Darboux is given as follows.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 + 2x_1x_2 \\ -x_1 + 2x_1^2 - x_2^2 \end{bmatrix}. \tag{22}$$

We define state space, initial region, and safe region as $\mathcal{X} : \{\mathbf{x} \in \mathbb{R}^2 : x \in [-2, 2] \times [-2, 2]\}$, $\mathcal{I} : \{\mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le 1, 1 \le x_2 \le 2\}$ and $\mathcal{C} : \{\mathbf{x} \in \mathbb{R}^2 : x_1 + x_2^2 \ge 0\}$ respectively.

Experiment Settings of the Obstacle Avoidance: The dynamic model of obstacle avoidance is given as follows.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \sin\psi \\ v \cos\psi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix}. \tag{23}$$

We define the state space, initial region and safe region as $\mathcal{X}$, $\mathcal{I}$ and $\mathcal{C}$, respectively as

$$\begin{aligned} \mathcal{X} &: \{\mathbf{x} \in \mathbb{R}^3 : x_1, x_2, \psi \in [-2, 2] \times [-2, 2] \times [-2, 2]\} \\ \mathcal{I} &: \{\mathbf{x} \in \mathbb{R}^3 : -0.1 \le x_1 \le 0.1, -2 \le x_2 \le -1.8,\ -\pi/6 < \psi < \pi/6\} \\ \mathcal{C} &: \{\mathbf{x} \in \mathbb{R}^3 : x_1^2 + x_2^2 \ge 0.04\} \end{aligned} \tag{24}$$

Experiment Settings of the Spacecraft Rendezvous: The state of the chaser is expressed relative to the target using linearized Clohessy–Wiltshire–Hill equations, with state $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$, control input $u = [u_x, u_y, u_z]^T$ and dynamics defined as follows.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}. \tag{25}$$

We define the state space and safe region as $\mathcal{X}$, initial safe region $\mathcal{X}_\mathcal{I}$ and $\mathcal{C}$, respectively as

$$\begin{aligned} \mathcal{X} &: \{\mathbf{x} \in \mathbb{R}^3 : p, v, \in [-5, 5] \times [-1, 1]\} \\ \mathcal{I} &: \left\{r \ge 0.75, \text{where } r = \sqrt{p_x^2 + p_y^2 + p_z^2}\right\} \\ \mathcal{C} &: \left\{r \ge 0.25, \text{where } r = \sqrt{p_x^2 + p_y^2 + p_z^2}\right\} \end{aligned} \tag{26}$$

hi-ord$_8$: The dynamic model of hi-ord$_8$ is given as follows.

$$x^{(8)} + 20x^{(7)} + 170x^{(6)} + 800x^{(5)} + 2273x^{(4)} + 3980x^{(3)} + 4180x^{(2)} + 2400x^{(1)} + 576 = 0 \tag{27}$$

where we denote the i-th derivative of variable x by $x^{(i)}$. We define the state space $\mathcal{X}$, initial region $\mathcal{X}_\mathcal{I}$ and safe region $\mathcal{C}$, respectively as

$$\begin{aligned} \mathcal{X} &: \{x_1^2 + \ldots + x_8^2 \le 4\} \\ \mathcal{I} &: \{(x_1 - 1)^2 + \ldots + (x_8 - 1)^2 \le 1\} \\ \mathcal{C} &: \{(x_1 + 2)^2 + \ldots + (x_8 + 2)^2 \ge 3\} \end{aligned} \tag{28}$$

https://doi.org/10.52202/079017-3214

B.2 Synthesis Framework Evaluation

The experimental results presented in Table 3 demonstrate the effectiveness of Counter Example (CE) guided training on Darboux and hi-ord$_8$ system. In this method, after each training epoch, we calculate the Control Barrier Function (CBF) outputs on representative samples. If the CBF correctly categorizes the samples into safe and unsafe regions, the certification procedure is initiated. If the CBF fails certification, the counter example is added to the training dataset for retraining. Otherwise, training is stopped early.

We capped the maximum training epochs at 50 and conducted three rounds of training for each network structure and system using different random seeds. The results indicate that without CE, the training process could basrely generate a CBF that passes certification. In contrast, with CE enabled, there was a success rate of at least 1/3 for most network structure, with verifiable policies generated in as few as 10 epochs. This highlights the improvement in training efficiency and reliability with the incorporation of CEs.

Case	L	M	No CE	With CE	
			sr	sr	min epoch
Darboux	2	8	0/3	3/3	38
	2	16	0/3	1/3	10
	4	8	0/3	1/3	43
	4	16	0/3	2/3	26
hi-ord$_8$	2	8	1/3	1/3	15
	2	16	0/3	2/3	19
	4	8	0/3	0/3	-
	4	16	0/3	2/3	13

Table 3: Success rates (sr) and minimum epochs required for certification with and without Counter Example (CE) guided training for different network structures on Darboux and hi-ord$_8$ systems.

B.3 Hyperparameters

Table 4 shows values the following hyperparameters used during CBF synthesis:

- N_{data}: number of samples to train CBF on.
- a_1: weight penalizing incorrect classification of safe samples in Equation 11.
- a_2: weight penalizing incorrect classification of unsafe samples Equation 11.
- λ_f: weight penalizing violation of Lie derivative condition of CBF in Equation 9.
- λ_c: weight penalizing correct loss for in Equation 9.
- n_{cluster}: number of clusters in $\mathcal{L}_{\mathcal{B}}$ regularization.
- k_σ: value of k used in generalized sigmoid function to perform differentiable activation pattern approximation.
- $\epsilon_{\text{boundary}}$: the threshold for range-based approximation of CBF boundary.

Case	N_{data}	a_1	a_2	λ_f	λ_c	n_{cluster}	k_σ	$\epsilon_{\text{boundary}}$
Darboux	5000	100	100	4.0	1.0	N/A	N/A	N/A
hi-ord$_8$	50000	100	200	1.0	1.0	N/A	N/A	N/A
OA	10000	100	100	2.0	1.0	5	4	1.0
SR	10000	100	100	2.0	1.0	5	4	1.0

Table 4: Hyperparameters of CBF synthesis.

B.4 Sensitivity Analysis of Hyperparameters

Next, we performed a sensitivity analysis of the hyperparameters. We chose the case study of Spacecraft Rendezvous with the number of layers $L = 4$ and the number of hidden units per layer

λ_c	SR	ME	N
1	0/3	x	x
10	0/3	x	x
100	**3/3**	**17**	**3265**
200	3/3	17.7	2922

(a) Ablation study on λ_c

λ_f	SR	ME	N
1	3/3	16.3	3254
2	**3/3**	**17**	**3265**
4	3/3	17	3352
8	2/3	29.5	3419.5

(b) Ablation study on λ_f

k	SR	ME	N
1	3/3	18	3842
2	3/3	15.7	3523
4	**3/3**	**17**	**3265**
8	1/3	10	1984

(c) Ablation study on k

Table 5: Ablation study for training hyperparameters. In each table, the bold lines indicate the baseline setting. **SR**: the success rate among runs with three random seeds. **ME**: the average first training epoch when a valid NCBF is obtained. **N**: the average number of boundary hyperplanes.

$N = 8$. We studied the sensitivity of the training performance to the hyperparameters $\lambda_{\mathcal{B}}$, λ_f, and λ_c in Equation 8, corresponding to the weightings for regularizing the number of boundary hyperplanes, NCBF value violation, and NCBF Lie derivative violation, respectively. We also studied the sensitivity to the hyperparameter k employed in the modified sigmoid function $\sigma_k(z) = \frac{1}{1+\exp(-k\cdot z)}$ to approximate the regularization pattern. We compared against the settings used in the original paper: $\lambda_{\mathcal{B}} = 10$, $\lambda_f = 2$, $\lambda_c = 100$, and $k = 4$. For each hyperparameter, we chose four values to perform the ablation study: $\lambda_{\mathcal{B}} \in \{0, 1, 10, 50\}$, $\lambda_f \in \{1, 2, 4, 8\}$, $\lambda_c \in \{1, 10, 100, 200\}$, and $k \in \{1, 2, 4, 8\}$. For each setting, we performed three runs with different random seeds. We measured the results by **Success Rate (SR)**, **Min Epoch (ME)**, and **N**, as described in the caption of Table 5.

λ_c regularizes the shape of the NCBF by penalizing incorrectly categorized samples. Table 5a indicates that when λ_c is too small, the training procedure fails to train an NCBF that correctly separates the safe and unsafe regions, resulting in failure of certification. Meanwhile, a larger weight delivers similarly good performance.

λ_f penalizes violations of Lie derivative conditions. Table 5b shows that the result is not sensitive to this hyperparameter, as this term quickly goes down to 0 when the Lie derivative condition is satisfied. We note that over-penalizing this condition should be avoided since the NCBF would otherwise learn an unrecoverable incorrect shape, as demonstrated by the failure case when $\lambda_f = 8$.

$\lambda_{\mathcal{B}}$ has been studied in the original paper, with detailed analysis in Section 5.2. The boundary regularization term reduces the number of boundary hyperplanes and benefits convergence.

Table 5c shows the importance of the term k used in the modified sigmoid function. Since this term appears in the exponential part of the sigmoid function, when it is too large, it leads to gradient explosions during backpropagation, which crashes the training process. Conversely, a reasonably larger k better approximates the activation pattern, leading to a reduced number of boundary hyperplanes.

In summary, balancing the hyperparameters is relatively straightforward, as the training performance remains robust across a wide range of hyperparameter values. When training failures do occur, we can systematically identify the cause from observation. This enables proper guidance in choosing and adjusting the appropriate hyperparameters.

https://doi.org/10.52202/079017-3214

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly states the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We point out the limitation on dynamical models in Section 2.1 and limitations on the type of activation functions throughout the paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We present assumptions in Section 2, and theoretical contributions in Section 4 with proof in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We specified random seeds for reproducibility. The code is uploaded with all the commands containing random seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

https://doi.org/10.52202/079017-3214

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is submitted with the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the detailed experiment settings with hyperparameters in the appendix. We present how to generate training data in Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All verification experiments are conducted in a deterministic manner. We also present the success rate of training with random seeds in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We present the details of the machine we experiment on in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

https://doi.org/10.52202/079017-3214

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: the research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper focuses on the safe control problem of dynamical systems. Therefore it would not harm or raise technology to harm the society.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: the paper poses no such risks. The experiments we conduct only generate simulation-based data of dynamic systems with no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cited all references in the both the paper and the code Readme file.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We upload our code with the submission.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

https://doi.org/10.52202/079017-3214

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

the paper does not involve crowdsourcing nor research with human subjects

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.