Two-Stage Learning of Stabilizing Neural Controllers via Zubov Sampling and Iterative Domain Expansion

Haoyu Li*   Xiangru Zhong*   Bin Hu   Huan Zhang
University of Illinois Urbana-Champaign
`haoyuli5@illinois.com, xiangru4@illinois.edu`
`binhu7@illinois.com, huan@huan-zhang.com`

Abstract

Learning-based neural network (NN) control policies have shown impressive empirical performance. However, obtaining stability guarantees and estimates of the region of attraction of these learned neural controllers is challenging due to the lack of stable and scalable training and verification algorithms. Although previous works in this area have achieved great success, much conservatism remains in their frameworks. In this work, we propose a novel two-stage training framework to jointly synthesize a controller and a Lyapunov function for continuous-time systems. By leveraging a Zubov-inspired region of attraction characterization to directly estimate stability boundaries, we propose a novel training-data sampling strategy and a domain-updating mechanism that significantly reduces the conservatism in training. Moreover, unlike existing works on continuous-time systems that rely on an SMT solver to formally verify the Lyapunov condition, we extend state-of-the-art neural network verifier α,β-CROWN with the capability of performing automatic bound propagation through the Jacobian of dynamical systems and a novel verification scheme that avoids expensive bisection. To demonstrate the effectiveness of our approach, we conduct numerical experiments by synthesizing and verifying controllers on several challenging nonlinear systems across multiple dimensions. We show that our training can yield region of attractions with volume $5 - 1.5 \cdot 10^5$ times larger compared to the baselines, and our verification on continuous systems can be up to $40 - 10{,}000$ times faster compared to the traditional SMT solver dReal. Our code is available at `https://github.com/Verified-Intelligence/Two-Stage_Neural_Controller_Training`.

1 Introduction

Learning-based neural network control policies have demonstrated great potential in complex systems due to their high expressiveness [19, 53]. However, the applications of these neural controllers in safety-critical real-world physical systems raise concerns since they typically lack stability guarantees. A promising method for quantifying safety is estimating the region of attraction (ROA), which refers to the set of initial states from which a system is guaranteed to converge to a desired equilibrium point. The computation of the ROA is often formulated as the search for an appropriate Lyapunov function satisfying some algebraic conditions over its sublevel sets [26, 41, 31]. Due to the highly complex and nonlinear nature of neural networks, searching for a Lyapunov function is not easy. Many existing works that are able to make stability guarantees about a system have some restrictions on the problem structure, for example, requiring linear or polynomial parameterizations of the system, where sum-of-squares technique [37, 47, 30, 7] can be applied. To enable certification for general nonlinear systems, many recent works explore data-driven methods to synthesize a neural certificate [8], such as

*Equal contribution.

Lyapunov functions [4, 6, 55, 44, 48, 35, 24, 34, 29, 23], barrier Functions [3, 32, 7, 10, 25, 9, 16, 17], and contraction metrics [38, 40, 39, 36, 22], to certify the system's stability.

In this work, we aim to synthesize neural controllers of continuous-time systems together with a Lyapunov certificate. Unlike typical machine learning tasks, this problem requires that the Lyapunov and boundary conditions hold over the *entire continuous domain of interest*, rather than statistically achieving high accuracy on a discrete test set. However, since at each iteration only a small number of points can be selected, the problem forces us to adopt specialized strategies both to pick the training domain and to sample training data from the domain. We argue that these two are the key factors for the conservativeness issue observed in many previous works. For selecting the training domain, many previous works simply select a fixed domain of interest and enforce the Lyapunov condition over it. While some previous works have proposed a new formulation to overcome the conservativeness by introducing an extra term to verify forward invariance and allow the ROA to touch the training domain nontrivially [48, 35], we argue that this does not fully solve the conservativeness issue and leaves the training domain selection a hard hyperparameter to tune.

In addition to the training domain, training data selection also plays an important role. Besides simple random sampling that typically does not suffice for certification, previous works dealing with similar problems [4, 6, 55, 48] typically achieve co-learning with a counterexample-guided (CEGIS) approach, which generates counterexamples of the Lyapunov condition as the training data, either using a verifier [4, 6] such as an SMT or MILP solver or a gradient-based approach [44, 48]. While these approaches have led to great success, such approaches can typically only refine upon already worked controller/Lyapunov function pair and thus rely on carefully tuned LQR-/RL-based initializations [4, 44, 48]. This limitation makes these works both conservative in their estimation of the ROA and also very difficult to apply to new systems without much domain-specific knowledge.

In our work, we propose a two-stage training framework with carefully designed curriculum domain expansion and training sample selection that can yield significantly less conservative ROA estimation inspired by Zubov theorem [56, 24, 18, 28], which provides a characterization of the true ROA with a PDE. During the first stage, we propose a Zubov-inspired training data sampling strategy by combining points from both inside and outside of the current ROA estimation to better guide the learning. Moreover, we eliminate the conservativeness introduced by the training domain by dynamically expanding it using trajectory information. This stage can provide us an almost-working controller and Lyapunov function that we can refine upon. The second stage then involves counterexample-guided learning (CEGIS) to eliminate all the counterexamples inside the ROA estimation and thus obtain a verifiable Lyapunov certificate. Importantly, our training also fully utilizes the physics-informed Zubov loss and can eliminate the conservatism introduced by the inner approximation nature of the normal Lyapunov function. As we shall demonstrate in our numerical experiments, our approach can robustly produce controllers with significantly larger ROAs compared to previous state-of-the-art approaches in all the benchmarks with various control input limits.

As we deal with continuous-time systems, verification over Jacobian vector products also presents a challenge. Previous works addressing the Lyapunov stability verification for continuous-time systems typically rely on SMT-, MIP-, or SDP-based solvers [4, 1, 24, 5, 49, 14, 5] such as dReal [15] or Z3 [11]. Inspired by the success in [48, 35], we also utilize the state-of-the-art neural network verification tool α,β-CROWN [52, 33, 45, 46, 43, 51] for formally verifying Lyapunov stability. The key challenge is to handle the Jacobian-related operators in dynamical systems, which have not been discussed or implemented in existing neural network verifiers. To ensure verification tightness, we design new linear relaxations for these Jacobian operators in continuous time system learning such as Tanh and Sigmoid. Furthermore, we introduce a novel verification scheme to dynamically adjust the specifications when noticing counterexamples without incurring much computational cost to avoid expensive bisections as in [48, 24].

To summarize, our main contributions include:

- We introduce a novel two-stage Zubov-based training framework with carefully designed curriculum training domain expansion and training sample selection that significantly reduces the conservatism during training.
- We customize and strengthen the advanced neural network verification tool α,β-CROWN with the ability of handling many commonly used Jacobian-related operators in continuous-time systems. Moreover, we propose a novel verification algorithm that can effectively avoid expensive bisections without sacrificing on soundness.

• We demonstrate through numerical experiments that our training can yield region of attraction with volume $5 - 1.5 \cdot 10^5$ times larger than the baselines. Moreover, across the systems under evaluation which we can formally verify, our verification achieves $40 - 10{,}000$ times speedup compared to the previous SMT solver dReal.

2 Background and Problem Statement

We consider a nonlinear continuous-time state-feedback control system

$$\dot{x} = g(x, \pi(x)) = f(x) \tag{1}$$

where x is the system state, g denotes the open-loop dynamics, u denotes the controller, and f denotes the closed-loop nonlinear dynamics. We denote the solution of the system starting with initial condition $x = x_0$ by $\phi(t, x_0)$, and denote the equilibrium state and the control input as x^*/u^* respectively. We denote the sublevelset $\{x : f(x) \leqslant c\}$ of some function f as $f^{\leqslant c}$ and correspondingly $f^{=c}$ for the levelset. Our objective is to jointly search for a neural network control policy u_θ together with a Lyapunov function V_θ that can certify the stability of the closed-loop dynamics (1) according to the Lyapunov theorem. The goal is to learn a controller that maximizes the region of attraction (ROA), which has the formal definition below.

Definition 2.1 (Region of Attraction (ROA) [21])**.** Region of attraction for the system (1) is the set $\mathcal{R}$ such that $\lim_{t\to\infty} \phi(t, x_0) = x^*$ for all $x_0 \in \mathcal{R}$.

Theorem 2.2 (Lyapunov Theorem [26, 21])**.** *Given a forward invariant set $\mathcal{S}$ containing the equilibrium, if there exists a function $V : \mathcal{S} \to \mathbb{R}$ such that we have*

$$\begin{aligned} V(x^*) &= 0 \\ V(x) &> 0 \;\; (\forall\, x \in \mathcal{S}\backslash\{x^*\}) \\ \nabla V(x)^\top f(x) &< 0 \;\; (\forall\, x \in \mathcal{S}) \end{aligned} \tag{2}$$

then $\mathcal{S}$ is a subset of the region of attraction.

Therefore, the objective of this work can be formalized as the optimization problem

$$\max_{\pi_\theta, V_\theta} \text{Vol}(\mathcal{S}), \quad \text{subject to Theorem (2.2)} \tag{3}$$

To maximize the possible ROA, we adopt the Zubov theorem, which states that a function that satisfies a stricter set of constraints than the Lyapunov condition can express the real underlying ROA $\mathcal{R}$.

Theorem 2.3 (Zubov Theorem [56, 24, 18])**.** *Consider the nonlinear dynamical system in* (1) *and we assume that $x^* = 0$ without loss of generality. Let $\mathcal{A} \subset \mathbb{R}^n$ and assume that there exists a continuously differentiable function $W : \mathcal{A} \to \mathbb{R}$ and a positive definite function $\Psi : \mathbb{R}^n \to \mathbb{R}$ satisfying the following conditions,*

$$W(0) = 0 \text{ and } 0 < W(x) < 1, \; \forall x \in \mathcal{A}\backslash\{0\} \tag{4}$$

$$W(x) \to 1 \text{ as } x \to \partial\mathcal{A} \text{ or } \|x\| \to \infty \tag{5}$$

$$\nabla W(x)^\top f(x) = -\Psi(x)(1 - W(x)), \quad \forall x \in \mathcal{A} \tag{6}$$

Then $x^* = 0$ is asymptotically stable with the region of attraction is exactly $\mathcal{A}$.

Here, equation (6) is called the Zubov equation and can be utilized as the physics-informed loss during our training. It is noticed that there exists a ground truth value for the Zubov function that can be obtained through the following theorem.

Theorem 2.4 (Liu et al. [24])**.** *We adopt the convention that divergence of integration yields the value ∞. Then for any $a, p > 0$, $W(x) = \tanh(a \int_0^\infty \|\phi(t, x)\|^p \, dt)$ solves the Zubov equation equation $\nabla W(x)^\top f(x) = -a(1 + W(x))(1 - W(x))\|x\|^p$.*

We refer the readers to [24] for more theoretical details. This theorem enables us to obtain the ground truth value of W during the training without solely depending on the PDE residual loss, making the training process more robust, and can provide better learning dynamics for the Lyapunov function.

3 Methods

3.1 Learning of Neural Controller and Lyapunov Function

In this section, we introduce our training pipeline for synthesizing the neural controller and a corresponding Lyapunov function. As we have discussed, the selection of the training domain and training samples are the most crucial factors for successful training. We shall see that the Zubov theorem provides us not only the physics-inspired loss, but also a principled way of data selection. During training, we parameterize the controller and Lyapunov function as

$$u_\theta(x) = c \cdot \tanh(\mathrm{NN}_c(x) - \mathrm{NN}_c(x^*) + \tanh^{-1}(\frac{u^*}{c})), \quad V_\theta(x) = \mathrm{sigmoid}(\mathrm{NN}_v(x)) \tag{7}$$

This parametrization ensures that we have $0 \leqslant V_\theta \leqslant 1$ and the crucial property $u_\theta(x^*) = u^*$, which ensures that the origin is also an equilibrium for the closed-loop system. Moreover, this parametrization constrains the control input lie in the range $[-c, c]$. Sometimes, if a non-symmetric limit $[0, c]$ is more natural, we append another ReLU layer to u_θ. This operation can also be applied to each coordinate separately to support mixed control-input limits.

3.1.1 ROA Estimation Stage

The first stage of colearning which we call ROA estimation stage, aims to set up a mostly working controller and an estimate of the Lyapunov function that can be further refined. As we have discussed, in this stage the most important aspect is the selection of training data and training domain.

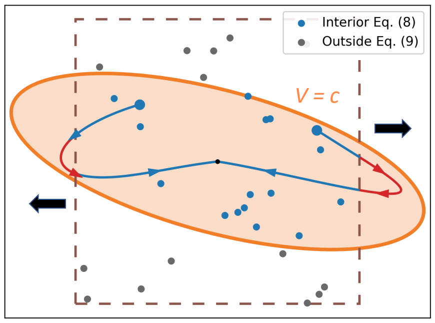


Figure 1: An illustration of the key idea used in the ROA estimation stage. Notice the balanced training sample selection from both in and outside of the ROA estimation $V^{\leqslant c}$. The training domain is also dynamically expanded to include all the states along a convergent trajectory.

Zubov Guided Sampling: Intuitively, a good training data selection scheme should mix the data points inside and outside of the running region of attraction. Selecting only "outside" points risks collapsing the sublevel set back toward the origin, while sampling exclusively "inside" offers no signal to enlarge it. In low dimensions, a well-chosen training box plus random sampling can approximate this mix, but as dimensionality grows, the ROA occupies a vanishing fraction of the domain, and naive draws will almost always only select the points outside of the ROA. For example, in the cartpole system, the ROA we found at the end of training only occupies less than 1% of the full training domain, and this will only be more pronounced in even higher-dimensional systems, and therefore naive sampling is infeasible.

In the traditional Lyapunov function framework, it is difficult to obtain this running estimation of the ROA. Zubov theorem, on the contrary, characterizes the exact region of attraction by the sublevel set $V^{\leqslant 1}$. Intuitively, in our training pipeline, since our parametrization already deems that V cannot exactly equal 1, we propose picking the training data by selecting a batch of points lying in $V_\theta^{\leqslant c}$ and another batch with $V_\theta(x)$ as close to 1 as possible, with $c \approx 1$. To achieve this, we select data by performing projected gradient descent (PGD) [27] with the following two objectives

$$L_{\text{interior}}(\theta) = \mathrm{ReLU}(V_\theta(x) - c) \tag{8}$$
$$L_{\text{outside}}(\theta) = |V_\theta(x) - 1|. \tag{9}$$

The first interior loss will push points to the sublevel set $V_\theta^{\leqslant c}$ and the outside loss will push points further away from the region of attraction as shown in Figure 1. A detailed discussion on how to pick c is presented in Appendix A.4.

Dynamic Training Domain Expansion: While previous works attempt to resolve the conservativeness introduced by a fixed training domain by allowing the ROA to touch non-trivially with it, we argue that the best way is instead to dynamically expand the domain during training. We thus propose to utilize the trajectory information to dynamically update the training box. We start from a relatively small training region and periodically perform a trajectory simulation starting from the points inside the current ROA estimation $V_\theta^{\leqslant c}$ with the same c in the data selection. We then expand the training region to include the farthest states that the converged trajectories can reach. To also benefit simple systems where all trajectories converge to zero in a monotone fashion, we further enlarge the box uniformly for the final training domain. If there is no convergent trajectory for many iterations, we manually enlarge the box for further exploration. This approach simultaneously resolves conservativeness in ROA posed by fixed domain and also resolves the challenging forward invariance issue. A visualization of our approach can be found in Figure 1, where the original blue training domain is expanded further to incorporate the states along the convergent trajectories. The domain updating algorithm can be found in Algorithm 2 in appendix.

Loss Function: The training loss can be easily adapted from the Zubov theorem. The objective of the training is to train the Lyapunov function and controller to encourage V_θ to behave more like a Zubov function. Therefore, naturally we could minimize $V_\theta(0)$ and train on the PDE residual of (6) and the data loss (2.4). In the handling of data loss, previous works such as [42] often simulates the trajectory till convergence to compute $\int_0^\infty \|\phi(t,x)\|^p\, dt$. This, however, would lead to very lengthy training and add difficulty in the implementation of efficient batched operations since different trajectories have different convergence speeds. We thus propose to replace this term using a Bellman-type dynamic programming approach. We note that the following rewrite is possible:

$$\begin{aligned} W(x) &= \tanh(a \int_0^\infty \|\phi(t,x)\|^p\, dt) \\ &= \tanh(a \int_0^T \|\phi(t,x)\|^p\, dt + \tanh^{-1}(W(\phi(T,x)))). \end{aligned} \tag{10}$$

With this approach, only simulation to time T is required. In practice, we found that this term can benefit training even if a very small T like $T = 0.01s$ is adopted. We also utilized a similar strategy as in [42] to decouple the learning of the Lyapunov function and controller using the actor-critic framework and impose the boundary condition $V_\theta(y) \approx 1$ for $y \in \partial(2\Omega)$ to guide the early phase learning. Moreover, to ensure that the second CEGIS stage is numerically stable, we minimize $V_\theta(0)$ only with a hinge-type loss. Let $\epsilon > 0$ be a small positive number. The final loss function is a weighted combination of the following components:

$$L_{\text{zero}}(\theta) = \text{ReLU}\left(V_\theta^2(0) - \epsilon\right), \tag{11}$$

$$L_{\text{pde}}(\theta) = \frac{1}{k_1} \sum_{i=1}^{k_1} \left(\nabla V_\theta(x_i)^\top f(x_i, \text{detach}(u(x_i))) + a(1+V_\theta(x_i))(1-V_\theta(x_i))\|x_i\|^p\right)^2, \tag{12}$$

$$L_{\text{data}}(\theta) = \frac{1}{k_2} \sum_{i=1}^{k_2} \left(V_\theta(x_i) - \tanh(a \int_0^T \|\phi(t,x_i)\|^p\, dt) - \tanh^{-1}(V_\theta(\phi(T,x_i)))\right)^2, \tag{13}$$

$$L_{\text{controller}}(\theta) = \frac{1}{k_1} \sum_{i=1}^{k_1} \text{detach}(\nabla V_\theta(x_i))^\top f(x_i, u(x_i)), \tag{14}$$

$$L_{\text{boundary}}(\theta) = \frac{1}{k_3} \sum_{i=1}^{k_3} \left(V_\theta(y_i) - 1\right)^2, \tag{15}$$

where $k_1, k_2, k_3 > 0$ represents the number of data points. Here, the PDE loss minimizes the PDE residual (6) by optimizing the Lyapunov function, whereas the controller loss optimizes the controller akin to a Sontag-style formula. Here x_i are training data sampled by our novel sampling scheme, and y_i are the boundary points from $\partial(2\Omega)$. More details about the loss function design can be found in appendix A.3.

3.1.2 CEGIS-based Refinement

Many previous works [4, 44, 48] start their training with a counterexample-guided approach (CEGIS [2]). However, without a good initialization, the counterexamples generated cannot be easily eliminated and will accumulate through the training and cause failure eventually, which is shown in the ablation studies Table 2. We found that our first stage can yield a great initialization for CEGIS-based training. The controller after the pretraining stage is already almost working, meaning that almost all trajectories starting within $V_\theta^{\leqslant c}$ will successfully converge to the equilibrium. However, as we have discussed, no certification can be given at this time since random samples typically are not sufficient to guarantee the condition holds over the entire continuous domain. Therefore, another stage of CEGIS-based finetuning is performed to ensure the certifiability of the system. To find counterexamples, we follow the previous works as in [48, 44] to use a cheap gradient-based PGD attack with the objective

$$L_{\text{cex}}(x) = \min(\nabla V_\theta(x) f(x), V_\theta(x) - c', c - V_\theta(x)) \tag{16}$$

Maximizing this objective leads to potential x_{cex} that violates the Lyapunov condition in the band $V_\theta^{\leqslant c} \backslash V_\theta^{\leqslant c'}$. Here we require a lower c' since there is no guarantee that $V_\theta(0) = 0$, and so near the origin there must exist some small violations. However, we can choose this c' to be just slightly larger than $V_\theta(0)$. These counterexamples then serve as the dataset for the CEGIS training. We then eliminate these counterexamples with the loss function

$$L_{\text{cegis}}(\theta; x_{\text{cex}}) = \text{ReLU}(\nabla V_\theta(x_{\text{cex}}) f(x_{\text{cex}})) \tag{17}$$

where it enforces the Lyapunov condition. Moreover, the overall shaping of the Lyapunov function should mostly be formed at the ROA estimation stage due to the presence of Zubov-inspired loss and should thus be only slightly refined at the CEGIS stage. Therefore, we denote v_0 as the value of $V_\theta(0)$ prior to the CEGIS finetuning, and apply the regularization term

$$L_{\text{reg}}(\theta) = \max(\text{ReLU}(v_0 - V_\theta(0)), \text{ReLU}(V_\theta(0) - c')) + \sum_{x \in \partial\Omega} |V_\theta(x) - 1| \tag{18}$$

with the aim of preventing the levelset of V_θ from changing drastically during CEGIS. The first term thus stabilizes the levelset structure near the origin by restricting $V_\theta(0)$ to stay close to the original value, whereas the second term stabilizes the levelset near the boundary of the training domain and encourages the ROA to stay away from this boundary to prevent trajectories from escaping from it.

Algorithm 1 Training of Neural Controller and Lyapunov Function

```
Require: Network Parameters $\theta$, Starting Domain $\Omega$, Maximum Iterations $M_1, M_2$, Integration
    Time $T$ and Discretizations Timesteps $dt$, Learning rate $\alpha$, Number of Training Points $N$,
    Domain Update Frequency $\gamma$, Number of Trajectories $N_T$, Lyapunov Upper Threshold $c$, Domain
    Expansion Factor $\beta$. PGD steps $P$ and step size $\alpha_1$, Finetuning learning rate $\alpha_2$ and epochs $K$
1: for $i = 1$ to $M$ do                                            ▷ Stage 1: ROA Estimation Stage
2:     if $\gamma \mid i$ then
3:         $\Omega \leftarrow$ UPDATEDOMAIN($\theta$, $\Omega$, $T$, $dt$, $N_T$, $c$, $\beta$)          ▷ See algorithm 2
4:     $x \leftarrow N$ data points from (8) and (9)
5:     $x_{\text{boundary}} \leftarrow N$ points from $\partial(2\Omega)$
6:     $\theta \leftarrow \theta - \alpha\nabla L(\theta, x, x_{\text{boundary}}, T, dt)$          ▷ See loss function section.

7: Dataset $\leftarrow \varnothing$                                  ▷ Stage 2: CEGIS Refinement
8: for $i = 1$ to $M$ do
9:     $x \leftarrow P$ points from $\Omega$
10:    for $j = 1$ to $P$ do
11:        $x \leftarrow x + \alpha_1 \nabla L_{\text{cex}}(x)$                        ▷ See equation (16)
12:    Dataset $\leftarrow \{\text{Dataset}, x\}$            ▷ Buffer has length limit. See appendix .
13:    for $j = 1$ to $K$ do
14:        $\theta \leftarrow \theta - \alpha_2 \nabla(\beta_1 L_{\text{cegis}}(\theta; \text{Dataset}) + \beta_2 L_{\text{reg}}(\theta))$     ▷ See section 3.1.2
15:    if No CEX in dataset for at least $n$ steps then
16:        break
17: return Trained parameters $\theta$, Final Training Domain $\Omega$
```

3.2 Verification

Criterion: Since there is no guarantee that $V_\theta(0) = 0$, counterexamples near the origin are unavoidable. As the Lyapunov condition enforces the Lyapunov function value to decrease along trajectories, we propose to exclude an unverifiable region defined as a sublevel set of the Lyapunov function. If the excluded set is small enough, this verification suffices for most practical needs. Formally, we propose the verification criterion

Theorem 3.1. *Let $0 < c_1 < c_2$. Suppose we have*

$$x \in (V^{\leqslant c_2} \backslash V^{\leqslant c_1}) \cap \Omega \;\longrightarrow\; \nabla V(x) \cdot g\big(x, \pi(x)\big) \;<\; 0, \tag{19}$$

$$x \in \partial\Omega \cap V^{\leqslant c_2} \;\longrightarrow\; g\big(x, \pi(x)\big) \cdot \vec{n}(x) \;<\; 0, \tag{20}$$

where $\vec{n}(x)$ is the outer normal vector of $\partial\Omega$ at x. Then both $V^{\leqslant c_1}$ and $V^{\leqslant c_2}$ are forward invariant*, and every trajectory initialized in $V^{\leqslant c_2}$ enters the smaller set $V^{\leqslant c_1}$ in finite time.*

The proof of this theorem can be found in appendix section B.1. Here equation (19) states the normal Lyapunov condition has to hold in the band $\{x : c_1 \leqslant V(x) \leqslant c_2\}$ whereas equation (20) states that the trajectory cannot escape through the boundary. Compared to previous works that simply exclude a rectangle near the origin [48], our verification scheme is more principled since it also provides some guarantees inside the unverifiable region. In systems where formal certification over the full domain is challenging, verifying the condition within a thin band where $c_2 - c_1 < \epsilon$ for arbitrarily small ϵ still allows one to certify the forward invariance of the sublevel set $V^{\leqslant c_1}$.

Formal Verification: For formal verification, instead of the commonly used SMT solver dReal in previous works [15, 4, 24], we hope to use a state-of-the-art neural network verification framework, such as α,β-CROWN [52, 33, 45, 46, 43, 51, 54], to rigorously verify the Lyapunov condition through the tool's efficient linear bound propagation and branch-and-bound procedures. However, several **key challenges remain** for the application of α,β-CROWN. First, since the continuous-time Lyapunov condition involves Jacobian-vector products, it is essential to support efficient and tight bound propagation for the Jacobian of various dynamical system operators, which were previously not supported in any NN verifiers. Second, even after applying a CEGIS procedure, small counterexamples can still arise near the levelset boundaries at c_1 and c_2. To identify the tightest verifiable values of c_1 and c_2, a lengthy and expensive bisection is typically required, as commonly done in prior work [48, 24].

In our work, we extend α,β-CROWN with the capability to efficiently and tightly bound the Jacobian of many commonly used operators such as Tanh, Sigmoid. We implemented new linear relaxations that are much tighter compared to naively treating their Jacobian as composite functions of already supported operators, where the details can be found in the appendix section B.3. Furthermore, we enable our verifier to dynamically update c_1, c_2 during verification to avoid expensive bisection while remain sound. Intuitively, during the branch-and-bound process, we pick out those domains currently with the worst bound and try to find counterexamples in those domains. If counterexamples are found, we update c_1 and c_2 to exclude the counterexamples. Notice that after this update, all the verified subproblems during branch and bound still remain sound and do not need to be verified again. This effectively saves the time needed for the repetitive verification needed if c_1 and c_2 are to be found by bisection. The full verification algorithm can be found in appendix section B.4.

Evaluation Schemes: Compared to discrete-time systems, where the final verification is performed on the discrete Lyapunov condition $\Delta V(x) = V(g(x, u(x)) - V(x) < 0$ through domain. Continuous-time systems require formally verifying the Jacobian vector product (19). The large number of multiplications is challenging for current verifiers since it is hard to bound multiplication well with a linear relaxation. Therefore, even with the strongest verification tool, verification is hard to scale to higher dimensions due to bound tightness issues. We thus define three evaluation schemes we considered. As system dimensionality increases, we must rely increasingly more on empirical checks.

1. **Formal Verification:** This scheme implies the condition in theorem (3.1) is formally verified throughout the domain with a verifier.

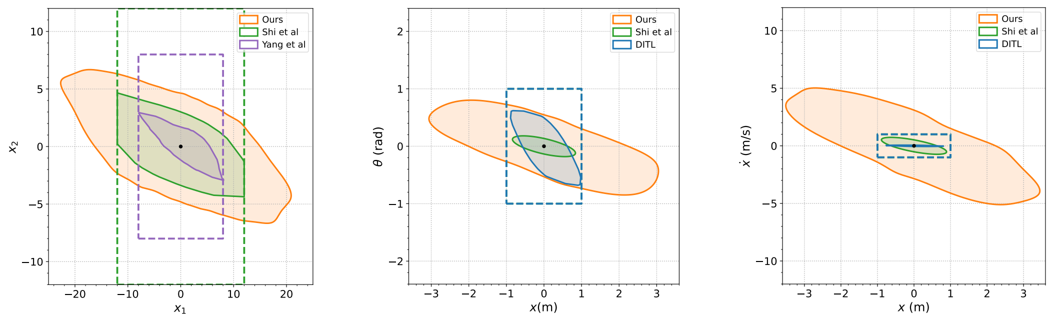
2. **PGD Evaluation:** In this scheme, we use a strong PGD attack to attempt finding counterexamples of the condition in theorem (3.1). If counterexamples are not found, we say the empirical PGD evaluation succeeds.

3. **Trajectory Evaluation:** In this scheme, we randomly sample trajectories from the sublevel set $V^{\leqslant c_2}$ and determine whether the trajectories converge to the desired equilibrium.

We can notice that these schemes are more and more empirical but are also more and more scalable to higher dimensional systems. The formal verification scheme, even though it is the strongest verification and can imply the other two schemes, can be hard to scale up. The empirical PGD evaluation can scale up better, but still fails on our hardest 12-dimensional system. However, the trajectory based evaluation can be easily scaled up to any dimension. While this sampling approach provides no formal guarantees, several previous works utilize this technique to validate the learned controller [50, 10, 47].

4 Experiments

Setups: We demonstrate the effectiveness of our methods in several challenging benchmarks including four 2D systems (Van-Der-Pol, Double-Integrator, Inverted Pendulum, Path Tracking), one 4D system (Cartpole), three 6D systems (PVTOL, 2D Quadrotor, Ducted Fan), and a 12D system (3D Quadrotor). System details follow prior works [44, 48, 35] and are also provided in appendix C.2. We compare our training framework against baselines from DITL [44], Yang et al. [48], and Shi et al. [35]. We note that all of them work with discrete-time systems. We focus on comparing to these baselines due to limitations in the existing continuous-time methods. The works [4, 42, 13] incorrectly handle the condition $u(x^*) = u^*$, so that the origin fails to be the equilibrium for most systems. Further discussions and examples are included in the appendix C.1. However, to ensure fair comparison, we still run and present the comparison with Fossil 2.0 [13] for those systems that their method handle correctly. For each method, we run the training/verification pipeline five times with different seeds. We compare training success rates and ROA volumes, where the latter is estimated following previous works [44, 35] as the fraction of points within the ROA sublevel set among a dense sample from the full domain. All models undergo the three evaluation steps from section 3.2 in reverse order. More experimental details can be found in appendix C.

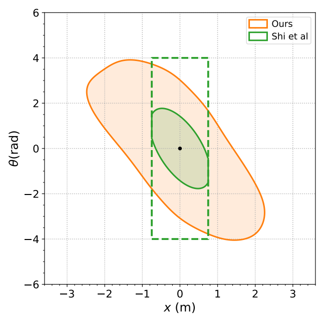


(a) Double Integrator. **(b)** Cartpole (x, θ) slice. **(c)** Cartpole $(x, \dot{x})$ slice.

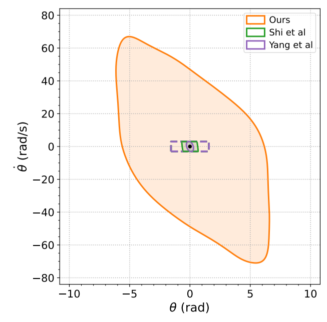
Figure 2: Region-of-Attraction (ROA) for Double Integrator and Cartpole compared to baselines. The dotted box shows the baseline's training box being used. For the double integrator system, it is clear that the initial training box is limiting the baseline's potential but is not constraining us.

Two and Four Dimensional Systems: On all the 2D systems, our methods robustly produce much larger ROAs compared to existing baselines. On both the path tracking and the inverted pendulum system, our training framework yields **5** times bigger ROA compared to the baselines as shown in table 1. Moreover, on the more challenging systems such as inverted pendulum with small torque, our method is the only one that can get perfect success rate. Visualizations of the learned ROA for Double Integrator can be seen in the left of Figure 2. We also test our method on a more challenging Cartpole system with four dimensions. Compared to the baselines, we obtain an ROA with more than $\mathbf{330}\times$ larger than the previous state of the art, as can be seen in Table 1. The visualizations for different projections of the ROA for Cartpole can be found on the right of Figure 2. The rest visualizations and details can be found in the Appendix C.6.

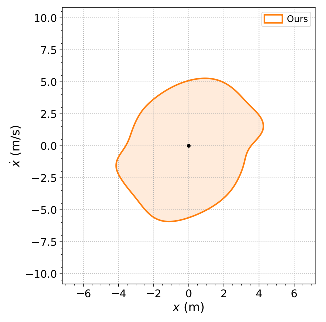
Six Dimensional Systems: We also implement our method on three higher dimensional systems: PVTOL, 2D Quadrotor, and Ducted Fan. For each system, our training pipeline yields a much larger ROA robustly. Compared to the baselines, we obtain an ROA with more than $\mathbf{150000}\times$ larger than the previous state of the art on average for 2D quadrotor and more than $\mathbf{380}\times$ larger for the PVTOL system as can be seen in Table 1. In the ducted fan case, we are the only method that can achieve an empirically verifiable ROA. We choose one projection to visualize for each of the system given in Figure 4. More visualizations can be found in the Appendix C.6. Due to the difficulty of continuous-time system verification, formal verification for these three systems is challenging.



(a) PVTOL



(b) 2D Quadrotor



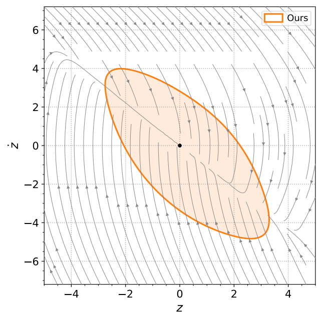
(c) Ducted Fan

Figure 3: Slice of the ROA for the 6D benchmark systems compared to baselines.

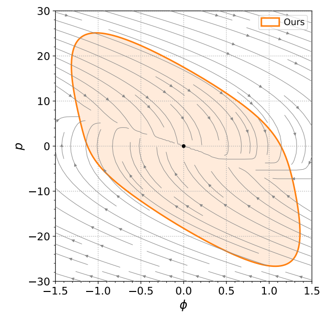
Table 1: Success rates (Succ) and ROA volume (ROA) across methods. Dashes (–) denote systems that are not already covered by the baseline and we were unable to obtain successful runs. The "Scheme" column indicates the evaluation scheme for each system as described in section 3.2: Formal = formal verification; PGD = empirical PGD evaluation; Trajectory = trajectory evaluation. "B" in a system implies the Big-torque setting, and "S" implies Small-torque.

System	Eval Scheme	Fossil [13]		DITL [44]		Yang et al. [48]		Shi et al. [35]		Ours	
		ROA	Succ	ROA	Succ	ROA	Succ	ROA	Succ	ROA	Succ
Van-der-Pol	Formal	3.05 ± 0.58	60%	–	–	1.36 ± 1.57	80%	20.2 ± 18.3	80%	$\mathbf{57.6 \pm 3.4}$	100%
Double Int	Formal	258.1 ± 36.3	80%	–	–	18.18 ± 16.19	60%	130.3 ± 3.9	60%	$\mathbf{302.5 \pm 10.7}$	100%
Pendulum B	Formal	106.5 ± 23.5	80%	61 ± 31	100%	70.6 ± 12.2	100%	487.5 ± 58.5	80%	$\mathbf{2946.5 \pm 149.1}$	100%
Pendulum S	Formal	285.2 ± 53.3	80%	–	–	217.34 ± 6.07	60%	306.3 ± 48.7	40%	$\mathbf{1169.2 \pm 124.5}$	100%
Path Tracking B	Formal	–	–	9 ± 3.5	100%	24.06 ± 0.29	100%	15.3 ± 8.9	60%	$\mathbf{122.0 \pm 3.7}$	100%
Path Tracking S	Formal	–	–	–	–	14.86 ± 0.18	100%	12.5 ± 6.5	100%	$\mathbf{73.8 \pm 12.5}$	100%
Cartpole	Formal	–	–	0.021 ± 0.012	100%	–	–	0.9266	20%	$\mathbf{306.1 \pm 54.2}$	100%
PVTOL	PGD	–	–	–	–	–	–	49.87 ± 3.91	80%	$(\mathbf{1.91 \pm 0.23}) \cdot \mathbf{10^4}$	100%
2D Quadrotor	PGD	–	–	–	–	2.33 ± 0.47	100%	44.53 ± 18.38	80%	$(\mathbf{6.64 \pm 4.67}) \cdot \mathbf{10^6}$	100%
Ducted Fan	PGD	–	–	–	–	–	–	–	–	$(\mathbf{4.31 \pm 1.86}) \cdot \mathbf{10^4}$	100%
3D Quadrotor	Trajectory	–	–	–	–	–	–	–	–	$(\mathbf{1.17 \pm 0.64}) \cdot \mathbf{10^9}$	100%

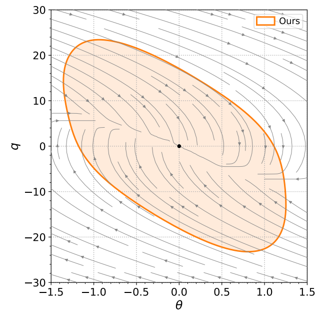
However, empirical verification is confirmed for these systems, and in 2D Quadrotor, we can formally verify the forward invariance of the sublevelset $V^{\leqslant 0.2}$ using Theorem 3.1.



(a) Projection on $(z, \dot{z})$



(b) Projection on (ϕ, p)



(c) Projections on (θ, q)

Figure 4: ROA estimation for 3D quadrotor. As we can see from the flow map, all the trajectories in our ROA estimation converges to the origin.

Twelve Dimensional System: Finally, we test our training pipeline in a 12 dimension 3D quadrotor system, in which none of the previous methods successfully give any ROA estimation. In this high dimensional case, we found that our pretraining framework works smoothly and can consistently yield a working controller. However, we found that CEGIS in higher dimensions does not work perfectly and we oftentimes found that counterexamples can still be located using longer PGD with more restarts even if a prolonged CEGIS is already performed. However, the empirical success of our approach is confirmed by sampling over 10^6 trajectories inside the levelset $V^{\leqslant c_2}$ and confirm that they all converge. The flow map in Figure 4 also confirms this.

Table 2: Success rates of different training schemes under three verification regimes.

System	Training scheme	Evaluation scheme (% success)		
		Formal	Empirical-PGD	Trajectory
Cartpole	Two-stage with Random sampling	0	0	0
	ROA-Estimation stage only (Zubov + Domain Expansion)	0	0	100%
	CEGIS only	0	0	0
	Ours (full pipeline)	100%	100%	100%
2-D Quadrotor	Two-stage with Random sampling	0	0	0
	ROA-Estimation stage only (Zubov + Domain Expansion)	0	0	100%
	CEGIS only	0	0	0
	Ours (full pipeline)	–	100%	100%

Verification Time: As we have discussed, our verifier struggles to verify higher-dimensional systems due to the bound tightness issue over Jacobian vector product. However, we shall note that our approach is already much faster compared to the previous state-of-the-art verifier dReal [15] used by many previous work [4, 24] for continuous time systems. In the 2D systems verification, our method yields $40 - 10000\times$ faster verification compared to dReal. And for the cartpole system, the dReal fails to terminate. The full verification time comparison can be found in Table 3.

Ablation Studies: We also investigate the effectiveness of our training design choices. We mainly focus on the effectiveness of our training sample selection scheme and the two-stage training by considering three training settings:

1. Two Stage training with domain expansion but with *random sampling*.
2. *Only the first ROA estimation stage* with Zubov inspired sampling and domain expansion.
3. *Only CEGIS* and domain expansion, no ROA estimation stage.

We then investigate the success rate of these three training schemes under different evaluations with two higher dimensional systems Cartpole and 2D quadrotor. The overall result can be found in Table 2. We can found that without our careful design choices, neither training Scheme 1 or Scheme 3 can yield any verifiable result. However, with our designed ROA estimation stage, even without CEGIS our trained controller and Lyapunov function can robustly pass the trajectory-based verification. Then together with CEGIS stage, our full training pipeline consistently yields controller/Lyapunov function that can be either formally verified or pass the PGD attack evaluation.

Table 3: Example verification time for each solver across our benchmark systems with formal verification. For 2D quadrotor, it is marked with a star due to only forward invariance of the set $V^{\leqslant 0.2}$ is verified by involving theorem 3.1 with $c_1 = 0.2, c_2 = 0.21$.

Method	Van der Pol	Double Integrator	Pendulum B	Pendulum S	Path Tracking B	Path Tracking S	Cartpole	2DQuad*
dReal	39265.21 s	359.27 s	1479.55 s	1709.19 s	113.72 s	142.20 s	–	–
α,β-CROWN	3.94 s	3.00 s	3.64 s	3.94 s	3.77 s	3.67 s	76443 s	104614 s

5 Conclusion

In this work, we propose a two stage training pipeline with novel training domain and training data selection for obtaining stabilizing controller with large region of attraction. Through extensive numerical experiments across various dimensions, we demonstrate that our training pipeline can achieve ROA volumes that are $5 - 1.5 \cdot 10^5$ bigger than the baselines. We also extend the state-of-the-art verifier α,β-CROWN with the capability of performing bound propagation through many Jacobian operators and introduce novel verification schemes to avoid expensive bisection. We demonstrated that our verification can yield a $40 - 10{,}000$ times speedup compared to the previously most commonly used verifier dReal. Although our method achieves promising empirical performance regarding the ROA volume and dimensionality compared to existing baselines, formal verification is still challenging for higher dimensional systems. As future works, it will be interesting to explore more verifier-friendly training strategies and to further tailor and improve the verifier itself for this specific problem. Moreover, it would be interesting to extend the framework to more general control system settings, such as discrete-time or hybrid systems, slowly time-varying systems, or to synthesize and verify robust controller for systems with perturbations.

Acknowledgment

H. Zhang is supported in part by the AI2050 program at Schmidt Sciences (AI2050 Early Career Fellowship) and NSF (IIS-2331967). B. Hu is generously supported by the AFOSR award FA9550-23-1-0732.

References

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

[2] Alessandro Abate, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. Counterexample guided inductive synthesis modulo theories. In *International Conference on Computer Aided Verification*, pages 270–288. Springer, 2018.

[3] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[4] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

[5] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. Learning lyapunov functions for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[6] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

[7] 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.

[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*, 39(3):1749–1767, 2023.

[9] Charles Dawson, Bethany Lowenkamp, Dylan Goff, and Chuchu Fan. Learning safe, generalizable perception-based hybrid control with certificates. *IEEE Robotics and Automation Letters*, 7(2):1904–1911, 2022.

[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] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[12] Alec Edwards, Andrea Peruffo, and Alessandro Abate. A general framework for verification and control of dynamical models via certificate synthesis. *arXiv preprint arXiv:2309.06090*, 2023.

[13] Alec Edwards, Andrea Peruffo, and Alessandro Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2024.

[14] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 67(1):1–15, 2020.

[15] Sicun Gao, Soonho Kong, and Edmund M Clarke. dreal: An smt solver for nonlinear theories over the reals. In *International conference on automated deduction*, pages 208–214. Springer, 2013.

[16] Pushpak Jagtap, George J Pappas, and Majid Zamani. Control barrier functions for unknown nonlinear systems using gaussian processes. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3699–3704. IEEE, 2020.

[17] Pushpak Jagtap, Sadegh Soudjani, and Majid Zamani. Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*, 66(7):3097–3110, 2020.

[18] Wei Kang, Kai Sun, and Liang Xu. Data-driven computational methods for the domain of attraction and zubov's equation. *IEEE Transactions on Automatic Control*, 69(3):1600–1611, 2023.

[19] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[20] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[21] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[22] Haoyu Li, Xiangru Zhong, Bin Hu, and Huan Zhang. Neural contraction metrics with formal guarantees for discrete-time nonlinear dynamical systems. In *7th Annual Learning for Dynamics & Control Conference*, pages 1447–1459. PMLR, 2025.

[23] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Tool lyznet: A lightweight python tool for learning and verifying neural lyapunov functions and regions of attraction. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–8, 2024.

[24] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Physics-informed neural network lyapunov functions: Pde characterization, learning, and verification. *Automatica*, 175:112193, 2025.

[25] 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.

[26] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[28] Yiming Meng, Ruikun Zhou, and Jun Liu. Zubov-koopman learning of maximal lyapunov functions. In *2024 American Control Conference (ACC)*, pages 4020–4025. IEEE, 2024.

[29] Yiming Meng, Ruikun Zhou, and Jun Liu. Learning regions of attraction in unknown dynamical systems via zubov-koopman lifting: Regularities and convergence. *IEEE Transactions on Automatic Control*, 2025.

[30] Antonis Papachristodoulou and Stephen Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 3482–3487. IEEE, 2002.

[31] Stefan Ratschan and Zhikun She. Providing a basin of attraction to a target region of polynomial systems by computation of lyapunov-like functions. *SIAM Journal on Control and Optimization*, 48(7):4377–4394, 2010.

[32] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. Ieee, 2020.

[33] 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.

[34] Mohamed Serry, Haoyu Li, Ruikun Zhou, Huan Zhang, and Jun Liu. Safe domains of attraction for discrete-time nonlinear systems: Characterization and verifiable neural network estimation. *arXiv preprint arXiv:2506.13961*, 2025.

[35] Zhouxing Shi, Cho-Jui Hsieh, and Huan Zhang. Certified training with branch-and-bound: A case study on lyapunov-stable neural control. *arXiv preprint arXiv:2411.18235*, 2024.

[36] Dawei Sun, Susmit Jha, and Chuchu Fan. Learning certified control using contraction metric. In *Conference on Robot Learning*, pages 1519–1539. PMLR, 2021.

[37] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[38] Hiroyasu Tsukamoto and Soon-Jo Chung. Neural contraction metrics for robust estimation and control: A convex optimization approach. *IEEE Control Systems Letters*, 5(1):211–216, 2020.

[39] Hiroyasu Tsukamoto, Soon-Jo Chung, Jean-Jacques Slotine, and Chuchu Fan. A theoretical overview of neural contraction metrics for learning-based control with guaranteed stability. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2949–2954. IEEE, 2021.

[40] Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jacques E Slotine. Neural stochastic contraction metrics for learning-based control and estimation. *IEEE Control Systems Letters*, 5(5):1825–1830, 2020.

[41] Giorgio Valmorbida and James Anderson. Region of attraction estimation using invariant sets and rational lyapunov functions. *Automatica*, 75:37–45, 2017.

[42] Jiarui Wang and Mahyar Fazlyab. Actor-critic physics-informed neural lyapunov control. *IEEE Control Systems Letters*, 2024.

[43] 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.

[44] Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural lyapunov control for discrete-time systems. *Advances in neural information processing systems*, 36:2939–2955, 2023.

[45] 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.

[46] 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.

[47] Lujie Yang, Hongkai Dai, Alexandre Amice, and Russ Tedrake. Approximate optimal controller synthesis for cart-poles and quadrotors via sums-of-squares. *IEEE Robotics and Automation Letters*, 8(11):7376–7383, 2023.

[48] Lujie Yang, Hongkai Dai, Zhouxing Shi, Cho-Jui Hsieh, Russ Tedrake, and Huan Zhang. Lyapunov-stable neural control for state and output feedback: A novel formulation. *arXiv preprint arXiv:2404.07956*, 2024.

[49] He Yin, Peter Seiler, and Murat Arcak. Stability analysis using quadratic constraints for systems with neural network controllers. *IEEE Transactions on Automatic Control*, 67(4):1980–1987, 2021.

[50] Zhexuan Zeng, Ruikun Zhou, Yiming Meng, and Jun Liu. Data-driven optimal control of unknown nonlinear dynamical systems using the koopman operator. *arXiv preprint arXiv:2412.01085*, 2024.

[51] 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*, 2022.

[52] 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.

[53] Xiaoning Zhao, Yougang Sun, Yanmin Li, Ning Jia, and Junqi Xu. Applications of machine learning in real-time control systems: a review. *Measurement Science and Technology*, 2024.

[54] Duo Zhou, Jorge Chavez, Hesun Chen, Grani A Hanasusanto, and Huan Zhang. Clip-and-verify: Linear constraint-driven domain clipping for accelerating neural network verification. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[55] Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. *Advances in Neural Information Processing Systems*, 35:29113–29125, 2022.

[56] Vladimir Ivanovich Zubov. *Methods of AM Lyapunov and their application*, volume 4439. US Atomic Energy Commission, 1961.

A Training Details

A.1 Domain Update Algorithm

The full domain update algorithm can be found in algorithm 2. It first samples a batch of points from the sublevelset $V^{\leqslant c}$ by PGD with the objective (8), and then performs trajectory simulation starting from these points. The algorithm then pick out those convergent trajectories and update the training domain to include the farthest state possible these convergent trajectories can reach.

Algorithm 2 UpdateDomain

```
Require: Controller $u_\theta$, Lyapunov Function $V_\theta$, Domain $\Omega$, Integration Time $T$ and Discretizations
    Timesteps $dt$, Number of Trajectories $N_T$, Lyapunov Upper Threshold $c$, Domain Expansion
    Factor $\beta$
1: $x_{\text{init}} \leftarrow N_T$ samples from the sublevel set $0 < V_\theta < c$          ▷ Data from (8)
2: $\mathcal{C} \leftarrow$ Set of indices $j$ where trajectory $\phi(t, x^j_{\text{init}})$ converges
3: for each dimension $d$ do
4:     $d_{\min} \leftarrow \min_{j\in\mathcal{C}, t\in[0,T]}[\phi(t, x^j_{\text{init}})]_d$
5:     $d_{\max} \leftarrow \max_{j\in\mathcal{C}, t\in[0,T]}[\phi(t, x^j_{\text{init}})]_d$
6:     $\Omega_d \leftarrow [d_{\min}, d_{\max}]$
7: $\Omega \leftarrow \beta \times \Omega$          ▷ To handle monotonic trajectories.
8: return $\Omega$
```

A.2 CEGIS Dataset

The dataset from line 12 of algorithm 1 will not grow without limitations. In practice, we fix a maximal size k allowed for this dataset. After each iteration, we sort the dataset in descending order according to the violation of Lyapunov condition, and only keep the top k samples.

A.3 Loss Function

As we have discussed in the main text, the training loss is simply an adaptation of the Zubov theorem. The goal is to guide V_θ to behave more like the Zubov function through training. We adopt a similar loss function as in [42] and modify the data loss using the novel derivation (10). Moreover, to ensure that the second CEGIS stage is numerically stable, we minimize $V_\theta(0)$ only with a hinge-type loss. In the loss functions, x_i are training data from our novel data selection (8) and (9). y_i are boundary points from $\partial(2\Omega)$ and this boundary loss would help during the initial phase of learning by ensuring that the controller does not drive the trajectory to escape from at least two times the training domain. To ensure a balanced weight of the loss term and ease the hyperparameter tuning, we adopt loss weights that are learnable using the techniques presented in [20].

A.4 Hyperparameters

Compared to previous works, our training framework significantly reduces the difficulty of hyperparameter tuning when adapting to new systems. For example, the training domain now does not require too much tuning, and the training almost always works when starting from a small domain. However, several key hyperparameters still require careful adjustment.

Firstly, the weights for each loss term must be chosen with care. In practice, we have found that the boundary loss and the zero loss benefit from higher weights compared to the other three loss terms. Conversely, assigning a relatively small weight to the PDE residual loss tends to yield better results. We hypothesize that this is because an overly dominant PDE loss introduces a complex optimization landscape, which can impede the learning of other critical components. Secondly, the integration involved in the data loss must be approximated using a numerical scheme. Both the discretization timestep and the number of simulation steps play an important role in ensuring stable and efficient learning. We found that a simple forward Euler scheme, with a small timestep such as $dt = 0.001\,\text{s}$ and approximately 50 simulation steps, is often sufficient. In this context, as long as dt is not excessively small, a smaller dt generally performs better than a larger one, since the dynamics can exhibit stiffness when the controller is only partially learned. Moreover, during trajectory simulations used for domain updates, a smaller dt and longer steps are recommended. Using a large timestep in this setting can cause otherwise convergent trajectories to be recognized as non-convergent. Finally, in the training data selection, the choice of c is crucial. We typically fix it to be very close to 1, like $c = 0.95$. However, as the dimension of the system goes up, the Zubov equation might not be fitted well, so we reduce c to be around 0.9 for better training stability.

B Verification Details

B.1 Proof of theorem 3.1

Proof. Notice that if the trajectory is in $(V^{\leqslant c_2} \backslash V^{\leqslant c_1}) \cap \Omega$, the Lyapunov function value will keep decreasing along the trajectory. Indeed, by the mean value theorem, we have

$$V(x(t)) - V(x(0)) = t\dot{V}(x(\tau)) < 0 \tag{21}$$

First, we shall show that $V^{\leqslant c_2} \cap \Omega$ is forward invariant. Assume by contradiction that there exists $x \in V^{\leqslant c_2} \cap \Omega$ such that $\phi(T, x) \notin V^{\leqslant c_2} \cap \Omega$ for some $T > 0$. We define the first exit time as $T^* = \inf\{t : \phi(t, x) \notin V^{\leqslant c_2} \cap \Omega\}$. By continuity, we have $\phi(T^*, x) \in \partial(S_2 \cap \Omega)$, which satisfies

$$\phi(T^*, x) \in \partial(V^{\leqslant c_2} \cap \Omega) \subset (\partial\Omega \cap V^{\leqslant c_2}) \cup (\Omega \cap V^{=c_2})$$

If $\phi(T^*, x) \in (\partial\Omega \cap V^{\leqslant c_2})$, condition (20) says that it cannot exit through the boundary of Ω. Otherwise, suppose $\phi(T^*, x) \in \Omega \cap V^{=c_2}$. We know by definition and continuity that on a small time interval $(T^* - \delta, T^*)$ we have $V(x(t)) < c_2$, which contradicts the decrease on Lyapunov function. Next, we show that on finite time the trajectory will enter $\Omega \cap V^{\leqslant c_1}$. Indeed, suppose for the contrary that the trajectory stays within $(V^{\leqslant c_2} \backslash V^{\leqslant c_1}) \cap \Omega$, the Lyapunov value will decrease indefinitely as we know by the above derivation $V(x(t_1)) < V(x(t_2))$ for $t_1 > t_2$, which cannot be possible by the definition of the set. Finally, we show that S_1 is forward invariant. With the same proof, it is easy to show that for any $0 < \epsilon < c_2 - c_1$ the set $V^{\leqslant c_1 + \epsilon}$ is forward invariant. Now if S_1 is not forward invariant, we know by definition that there exists $x \in V^{\leqslant c_1}$ and $t > 0$ such that $V(\phi(x, t)) = c_1 + \epsilon'$. This violates the forward invariance of $V^{\leqslant c_1 + \epsilon}$ for any $\epsilon < \epsilon'$, leading to a contradiction. □

B.2 The α,β-CROWN verifier

α,β-CROWN has been developed to be the state-of-the-art neural network verifier, where formally, the toolbox aims to certify

$$F(x) \leqslant 0 \text{ for all } x \in \mathcal{B} \tag{22}$$

with F being a general computation graph and B being a hyper-rectangle.

High Level Intuition: α,β-CROWN couples fast *symbolic linear bound propagation* with *input-space branch-and-bound*. For a box $\mathcal{B}$, it constructs an affine upper bound of F and provide certification based on the maximum value of this affine bound. If inconclusive, it *splits* $\mathcal{B}$ into sub-boxes and repeats. As boxes shrink, local linear relaxations tighten and nonlinearities stabilize, so on small sub-boxes the affine bound is often already exact. The full process is very efficient, as each bound computation essentially costs a backward pass; thus many sub-boxes can be processed in parallel (tens of thousands), enabling rapid pruning of large regions and certification with only coarse bounds on tiny boxes. Moreover, thanks to the tensorized nature of bound propagation and the flexibility of the branch-and-bound procedure, α,β-CROWN can naturally handle general specifications with "or" and "and". In principle, α,β-CROWN can handle all cases a traditional SMT solver, such as dReal, can handle, and in a much more scalable fashion. We refer the readers to the following papers for more details [52, 45, 45, 46, 43, 51].

For our application, since the condition in theorem 3.1 can be effectively rewritten as

$$(\nabla V(x) \cdot g(x, u(x)) < 0) \vee (V(x) > c_2) \vee (V(x) < c_2) \tag{23}$$

$$(g(x, u(x)) \cdot \vec{n}(x) < 0) \vee (x \notin \partial\mathcal{B}) \vee (V(x) > c_2), \tag{24}$$

and each term is a computation graph, α,β-CROWN can efficiently verify this condition.

B.3 Linear Relaxation of $\frac{\mathrm{d}}{\mathrm{d}x}\tanh(x)$ and $\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}(x)$

To compute the Jacobian of a neural Lyapunov function involving $\tanh(x)$ and $\mathrm{sigmoid}(x)$, by the chain rule, we need to propagate the gradient through the operators $\frac{\mathrm{d}}{\mathrm{d}x}\tanh$ and $\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}$. Naturally, to use CROWN [52] to compute bounds for the Jacobian of the network, we need to linearly relax these operators so that we can propagate linear bounds through them. Given

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}\tanh(x) &= 1 - \tanh^2(x), \\ \frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}(x) &= \mathrm{sigmoid}(x)(1-\mathrm{sigmoid}(x)), \end{aligned} \tag{25}$$

Finding linear relaxations of $\frac{\mathrm{d}}{\mathrm{d}x}\tanh$ and $\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}$ can be reduced to computing the composition of the linear relaxations of $\tanh(x)$, x^2, $\mathrm{sigmoid}(x)$, and $x \cdot y$. The linear relaxations of these operators are already supported by auto_LiRPA [45]. However, in practice, this results in overly conservative relaxations. Thus, in our work, we directly derive a novel linear relaxation for these two gradient operators.

$\frac{\mathrm{d}}{\mathrm{d}x}\tanh(x)$ and $\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}(x)$ are "bell-shaped" functions having similar monotonicity and convexity/concavity properties, as shown in figure 5. Thus, we apply similar strategies finding linear relaxations for both of them. Suppose the function to be bounded is denoted as $\sigma(x)$, with the bounds of x as $x \in [l, u]$. We aim to find a lower bound $\alpha^L x + \beta^L$ and an upper bound $\alpha^U x + \beta^U$, such that

$$\alpha^L x + \beta^L \leqslant \sigma(x) \leqslant \alpha^U x + \beta^U, \quad \forall x \in [l, u], \tag{26}$$

where parameters α^L, β^L, α^U, β^U are dependent on l and u. In our following discussion, for a "bell-shaped" functions $\sigma(x)$, we denote $\pm z_\sigma$ as the two inflection points (i.e. the solution to $\sigma''(x) = 0$), and we assume that $l + u \geqslant 0$ without loss of generality. In many cases, the linear bound is chosen to be a tangent line of σ, and thus is solely determined by the tangent point d^L or d^U.

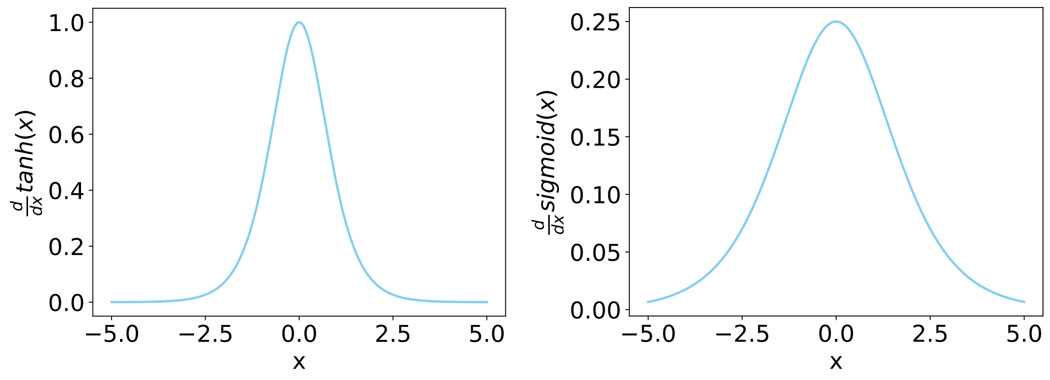


Figure 5: $\frac{\mathrm{d}}{\mathrm{d}x}\tanh(x)$ (left) and $\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}(x)$ (right) have similar monotonicity and convexity/concavity properties.

The segment to be bounded is concave if $-z_\sigma \leqslant l \leqslant u \leqslant z_\sigma$. In this case, the lower bound is chosen to be the line directly connecting two endpoints $(l, \sigma(l))$ and $(u, \sigma(u))$. For the upper bound, d^U can be any point between l and u (e.g. the middle point). See **(a)** of figure 6 (a).

The segment to be bounded is convex if $z_\sigma \leqslant l \leqslant u$. In this case, the upper bound is chosen to be the line directly connecting two endpoints. For the lower bound, d^L can be any point between l and u (e.g. the middle point). See figure 6 (b).

The segment is bounded similarly to an "S-shaped" function if $l < z_\sigma < u$. See figure 6 (c).

- For the lower bound, let d_l be the tangent point of the tangent line that passes through $(l, \sigma(l))$. If $u \leqslant d_l$, the lower bound is chosen to be the line directly connecting two endpoints. Otherwise, d^L can be any point between d_l and u (e.g. the middle point).
- For the upper bound, let d_u be the tangent point of the tangent line that passes through $(u, \sigma(u))$. If $l \geqslant d_u$, the upper bound is chosen to be the line directly connecting two

endpoints. Otherwise, d^U can be any point between 0 and d_u (e.g. linearly interpolated given l ranging from $-u$ to d_u).

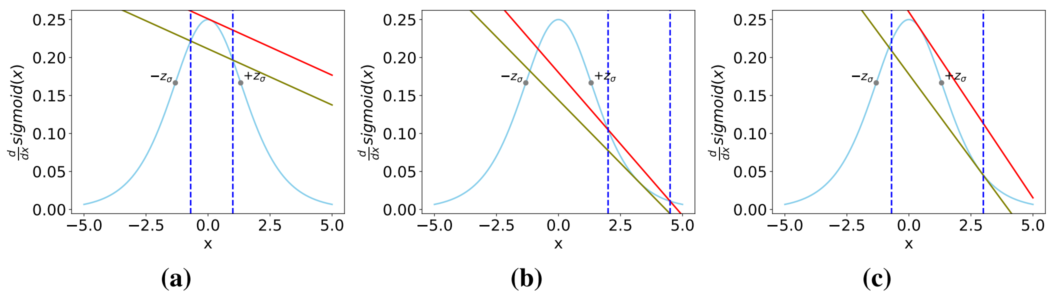


Figure 6: Illustration of linear relaxations of a "bell-shaped" function (sigmoid). **(a)** $-z_\sigma \leqslant l \leqslant u \leqslant z_\sigma$. The segment to be bounded is concave. **(b)** $z_\sigma \leqslant l \leqslant u$. The segment to be bounded is convex. **(c)** $l < z_\sigma < u$. The segment is bounded similarly to an "S-shaped" function.

B.4 Verification Algorithm

Although a CEGIS procedure has been applied during training, small counterexamples can still exist. Previous works use bisection to identify the optimal boundaries of the region where the Lyapunov condition can be verified. This requires test running the entire verification procedure every time a new candidate boundary is set, which is very inefficient. In contrast, our work, thanks to the formulation of the region to verify as a levelset defined by c_1 and c_2 in 19, and the input-split branch-and-bound pipeline, can adaptively adjust the values of c_1 and c_2 *in a single verification run*, avoiding the tedious and time-consuming bisection process.

Specifically, during the branch-and-bound procedure, for each batch of subdomains, apart from identifying each of them as VERIFIED or UNKNOWN based on its certified bounds, we additionally empirically search for counterexamples on some of them (usually the one with the worst certified bound). We shall note that searching for counterexamples in a subdomain by either random or PGD attack is much easier than doing so in the whole initial domain, and thus is more likely to find counterexamples that were missed out in the CEGIS based training stage. We then shrink the levelset by increasing c_1 or decreasing c_2 to exclude all the counterexamples. The key intuition here is although the specification to verify is updated, the subdomains that has been verified remain verified, so we don't need to rerun the whole algorithm from the beginning again. The full verification algorithm is shown in algorithm 3.

C Experiment Details

C.1 Continuous Time Baselines

In the main text, we discussed the limitations of the continuous-time baselines [4, 12, 42]. A critical condition for theoretical guarantees is that the controller satisfies $u(x^*) = u^*$; otherwise, x^* is no longer an equilibrium point, rendering the Lyapunov-based guarantee vacuous. In NLC [4], we observed that the provided code mistakenly uses a bias-free controller for both the inverted pendulum and the path-following dynamics. While this is correct for the inverted pendulum, where $u^* = 0$, it is incorrect for the path-following dynamics, as the chosen parametrization still enforces $u(0) = 0$, which does not hold in this case. This exact issue has also been noted in an open issue on NLC's official GitHub repository: `https://github.com/YaChienChang/Neural-Lyapunov-Control/issues/14`. For the method proposed by Wang et al. [42], we found no special treatment to enforce this condition. In fact, when testing the publicly released checkpoints from their GitHub repository, we found that none of them satisfied $u(x^*) = u^*$. For example, in the Double Integrator system, $u(0) = 0.3733$, and in the Van der Pol system, $u(0) = -0.0633$, despite both systems having $u^* = 0$ as their correct equilibrium input. Moreover, we believe there are some subtle issues in the verification formulation of Fossil 2.0. In [12], which underpins Fossil 2.0, we notice that their Certificate 2 misses the boundary condition, which should supposedly ensure either $\{x : V(x) \leqslant \beta\} \cap \mathcal{X}$ is empty or any trajectory starting from this intersection will remain in $\mathcal{X}$. We have observed that their method can

Algorithm 3 Adaptive Branch-and-Bound Verification with Levelset Adjustment

```
Require: Neural network $f$, specification functions $V$ and $\dot{V} = \nabla V \cdot f$, initial domain $\Omega$, initial
    levelset thresholds $c_1 < c_2$, batch size $B$, boundary update margin $\epsilon$
 1: $\mathcal{S} \leftarrow$ stack([$\Omega$])                          ▷ stack of unverified subdomains
 2: while $\mathcal{S} \neq \varnothing$ and $c_1 < c_2$ do
 3:     $\mathcal{B} \leftarrow$ PopBatch($\mathcal{S}, B$)                     ▷ pop up to $B$ subdomains
 4:     $\mathcal{U} \leftarrow \varnothing$                                ▷ unverified subdomains in this batch
 5:     for each $D$ in $\mathcal{B}$ do
 6:         Compute certified bounds of $V(x)$ and $\dot{V}(x)$ over $D$
 7:         if condition 19 is *not* verified to satisfy then          ▷ $c_1$ and $c_2$ are used here
 8:             add $D$ to $\mathcal{U}$
 9:     if $\mathcal{U} \neq \varnothing$ then
10:         $D_{\text{worst}} \leftarrow$ WorstDomain($\mathcal{U}$)     ▷ greatest positive upper bound of $\dot{V}(x)$
11:         Empirically search $D_{\text{worst}}$ for counterexample $x_{\text{ce}}$
12:         if counterexample found then
13:             UpdateThresholds($x_{\text{ce}}, c_1, c_2, \epsilon$)
14:         for each $D$ in $\mathcal{U}$ do
15:             Split $D$ into $D_1, D_2$; Push($\mathcal{S}, D_1, D_2$)          ▷ branching
16: if $\mathcal{S} = \varnothing$ then
17:     return Verified                               ▷ all subdomains have been verified
18: else
19:     return Falsified                ▷ $c_1 = c_2$, the levelset is completely excluded

20: function UpdateThresholds($x_{\text{ce}}, c_1, c_2, \epsilon$)
21:     $v \leftarrow V(x_{\text{ce}})$
22:     if $|v - c_1| < |v - c_2|$ then
23:         $c_1 \leftarrow v + \epsilon$                            ▷ counterexample near lower bound
24:     else
25:         $c_2 \leftarrow v - \epsilon$                            ▷ counterexample near upper bound
```

occasionally return false positives, i.e., a controller and Lyapunov function are claimed to be found while the controller cannot stabilize any trajectory. Therefore, on top of their returned results, we ran an additional check with trajectory simulations, and recorded a negative for a run if the controller does not work.

C.2 Dynamical Systems

Van Der Pol: The dynamics is given by

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= -x_1 + \mu(1 - x_1^2)x_2 + u\end{aligned}$$

with $\mu = 1$. We choose the control input limit as $|u| \leqslant 1$ for all the baselines.

Double Integrator: The dynamics is given by

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= u\end{aligned}$$

and constrain the torque limit as $|u| \leqslant 1$.

Inverted Pendulum: The dynamics is given as

$$\begin{aligned}\dot{\theta}_1 &= \theta_2\\ \dot{\theta}_2 &- \frac{\beta}{ml^2}\theta_2 + \frac{g}{l}\sin(\theta_1) + \frac{1}{ml^2}u.\end{aligned}$$

We consider two set of torque limits as in [35]. The large torque case sets $|u| \leqslant 8.15mgl$ and the small torque case presents the more challenging limit $|u| \leqslant 1.02mgl$.

Path Tracking: The dynamics is given as

$$\begin{aligned}\dot{d}_e &= v\sin(\theta_e)\\ \dot{\theta}_e &= \frac{v}{l}u - \frac{\cos(\theta_e)}{\frac{r}{v} - \sin(\theta_e)}.\end{aligned}$$

We also consider two set of torque limits for this dynamics. We set $|u| \leqslant 1.68\frac{l}{v}$ in the large torque case and $|u| \leqslant \frac{l}{v}$ in the small torque case following [35].

Cartpole: The dynamics is given by

$$\begin{aligned}\ddot{x} &= \frac{1}{m_c + m_p\sin^2\theta}(u + m_p\sin\theta(l\dot{\theta}^2 - g\cos\theta))\\ \ddot{\theta} &= \frac{1}{l(m_c + m_p\sin^2\theta)}(-u\cos\theta - m_p l\dot{\theta}^2\cos\theta\sin\theta + (m_c + m_p)g\sin\theta).\end{aligned}$$

Following DITL [44], we set $m_c = 1.0, m_p = 0.1, l = 1.0, g = 9.81$ and $|u| \leqslant 30$.

2D Quadrotor: The dynamics is given by

$$\begin{aligned}\ddot{x} &= -\frac{1}{m}\sin(\theta)(u_1 + u_2)\\ \ddot{y} &= \frac{1}{m}\cos(\theta)(u_1 + u_2) - g\\ \ddot{\theta} &= \frac{l}{I}(u_1 - u_2)\end{aligned}$$

with $m = 0.486, l = 0.25, I = 0.00383, g = 9.81$. We also set the control input $u \in \mathbb{R}^2_{\geqslant 0}$ to have the constraints $\|u\|_\infty \leqslant 1.25mg$ as in [35].

PVTOL: The dynamics is given by

$$\dot{x}(t) = \begin{bmatrix} v_x\cos\phi - v_z\sin\phi \\ v_x\sin\phi + v_z\cos\phi \\ \dot{\phi} \\ v_z\dot{\phi} - g\sin\phi \\ -v_x\dot{\phi} - g\cos\phi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & \frac{1}{m} \\ \frac{l}{J} & -\frac{l}{J} \end{bmatrix} u \tag{27}$$

where $g = 9.8, m = 4.0, l = 0.25, J = 0.0475$. We set the control input $u \in \mathbb{R}^2_{\geqslant 0}$ to have the limit $\|u\|_\infty \leqslant 39.2$ as in [44].

Ducted Fan: The dynamics is given by

$$\begin{aligned}\ddot{x} &= \frac{1}{m}\left(-d\dot{x} + u_0\cos\theta - u_1\sin\theta\right)\\ \ddot{y} &= \frac{1}{m}\left(-d\dot{y} + u_0\sin\theta + u_1\cos\theta\right) - g\\ \ddot{\theta} &= \frac{r}{I}u_0\end{aligned}$$

with $m = 11.2$, $r = 0.156$, $I = 0.0462$, $d = 0.1$, and $g = 0.28$. We set the control input $u = [u_0, u_1]^\top \in \mathbb{R}^2$ to be $u_0 \in [-10, 10]$, $u_1 \in [0, 10]$.

3D Quadrotor:

$$\begin{aligned}\ddot{\mathbf{p}} &= \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m}R_{\text{col}}(\phi, \theta, \psi)\cdot T\\ \dot{\phi} &= \omega_x + \tan(\theta)(\sin(\phi)\omega_y + \cos(\phi)\omega_z)\\ \dot{\theta} &= \cos(\phi)\omega_y - \sin(\phi)\omega_z\\ \dot{\psi} &= \frac{\sin(\phi)\omega_y + \cos(\phi)\omega_z}{\cos(\theta)}\\ \dot{\boldsymbol{\omega}} &= \mathbf{I}^{-1}\left(-\boldsymbol{\omega}\times(\mathbf{I}\boldsymbol{\omega}) + \boldsymbol{\tau}\right)\end{aligned}$$

where

$$R_{\text{col}}(\phi, \theta, \psi) = \begin{bmatrix} \sin\psi \sin\phi + \cos\psi \sin\theta \cos\phi \\ -\cos\psi \sin\phi + \sin\psi \sin\theta \cos\phi \\ \cos\theta \cos\phi \end{bmatrix}$$

The control input $u \in \mathbb{R}^4$ represents the four rotor thrusts. These are mapped to generalized forces via

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ c & -c & c & -c \end{bmatrix} u$$

The hyperparameters are specified with $m = 0.486, l = 0.225, g = 9.81, \mathbf{I} = \text{diag}(I_x, I_y, I_z) = \text{diag}(0.0049,\ 0.0049,\ 0.0088), c = \frac{1.1}{29} \approx 0.03793, u^* = \frac{mg}{4} = \frac{0.486 \times 9.81}{4} = 1.190325$. The torque limit of $u \in \mathbb{R}^4_{\geqslant 0}$ is given by $|u| \leqslant 3.6$.

C.3 Examples of Domain Update

We present some examples on the effect of the dynamic domain expansion mechanism and demonstrate its effectiveness. The initial domain for 2D Quadrotor is picked by 0.4 multiplied the original domain from [48, 35], and the other ones are designed to be small but overall are picked quite arbitrarily without much tuning. A comparison between the initial training domain and the one after the first stage training is presented in Table 4. We can notice that our domain update mechanism is able to enlarge the training domain and thus the ROA by a large amount, resulting in much bigger and much more less conservative ROA estimation compared to the baselines.

Table 4: Example start and end domain for each dynamical system. Lower/Upper bound of the domain is given.

System	Start Domain	End Domain
Double Integrator	$\pm[1, 1]$	$\pm[27.6, 9.6]$
Van der Pol	$\pm[1, 1]$	$\pm[6, 8.5]$
Pendulum (small torque)	$\pm[1, 2]$	$\pm[27.6, 69.6]$
Pendulum (large torque)	$\pm[1, 2]$	$\pm[30, 120]$
Path Tracking (small torque)	$\pm[2, 2]$	$\pm[9.6, 7.2]$
Path Tracking (large torque)	$\pm[2, 2]$	$\pm[12, 12]$
Cartpole	$\pm[0.4, 0.4, 0.4, 0.4]$	$\pm[4.8, 3.6, 13.2, 13.2]$
2D Quadrotor	$\pm[0.3, 0.3, 0.2\pi, 1.6, 1.6, 1.2]$	$\pm[12, 13.2, 12, 19.2, 20.4, 88.8]$
PVTOL	$\pm[0.4, 0.4, 0.4, 0.4, 0.4, 0.4]$	$\pm[3.6, 3.6, 3.6, 6, 8.4, 33.6]$
Ducted Fan	$\pm[0.4, 0.4, 0.4, 0.4, 0.4, 0.4]$	$\pm[6, 7.2, 10.8, 4.8, 3.6, 25.2]$

C.4 Examples of verifiable c_1 and c_2

Table 5 gives some examples of the final verifiable c_1, c_2 thresholds. To demonstrate that the unverifiable hole is small, we also report the volume of the sublevelset $V^{\leqslant c_1}$ and the volume it occupies the full ROA. We can see that for all the systems the unverifiable region occupies only less than 0.1% of the full ROA. For the higher-dimensional (6-D) systems, none of the 5×10^7 samples fell into the small unverifiable region.

C.5 Training Time

Table 6 provides the training time for our method. The time needed for the ROA estimation stage is relatively stable since it is trained to a fixed epoch. However, the time taken by CEGIS will depend on the c_1 chosen and the hardness of counterexamples near c_1. In practice, we found that even though the CEGIS time varies, all training processes complete within reasonable time.

C.6 More ROA Visualizations

We also provide more visualizations on the final ROA obtained through our methods in figure 7. Across all visualized slices, our method yield much larger ROA estimation compared to the baselines.

Table 5: Examples of verifiable thresholds c_1, c_2, and the volume of the unverifiable sublevel set $\{x : V(x) \leqslant c_1\}$ with its proportion of the full ROA. The ROA is estimated in a Monte-Carlo fashion with 50000000 points.

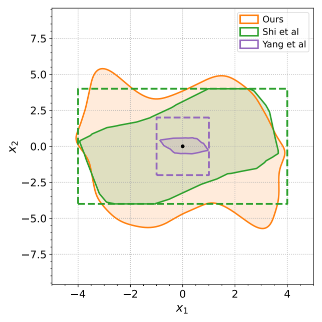
System	c_1	c_2	Vol$\{V(x) \leqslant c_1\}$	Proportion to Full ROA
Double Integrator	0.01	0.99	0.038	0.014%
Van der Pol	0.0106	0.99	0.015	0.025%
Pendulum (small torque)	0.0103	0.98	0.17	0.016%
Pendulum (large torque)	0.022	0.98	0.36	0.012%
Path Tracking (small torque)	0.0108	0.99	0.057	0.08%
Path Tracking (large torque)	0.0105	0.99	0.040	0.03%
Cartpole	0.0068	0.95	0.0154	0.003%
2D Quadrotor	0.096	0.89	0	0%
PVTOL	0.003	0.90	0	0%
Ducted Fan	0.0058	0.89	0	0%

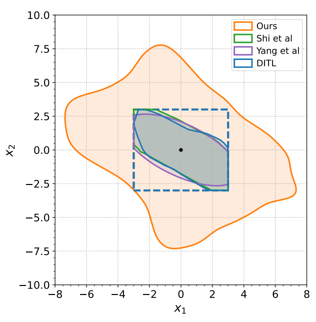
Table 6: Training time (in seconds) for each system. We report separately for the ROA Estimation stage, CEGIS stage, and a combined overall time in seconds.

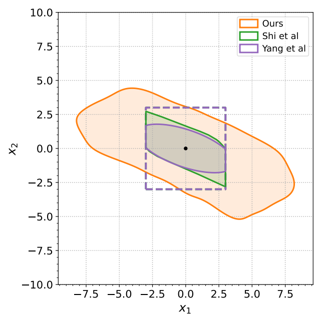
System	ROA Estimation	CEGIS	Full
Double Integrator	1173.20 ± 30.83	162.39 ± 97.61	1335.6 ± 123.80
Van der Pol	1201.10 ± 25.35	187.53 ± 28.14	1388.64 ± 44.38
Pendulum (small torque)	1169.27 ± 139.29	225.14 ± 64.17	1394.41 ± 134.53
Pendulum (large torque)	2946.59 ± 166.73	98.25 ± 44.97	3044.85 ± 153.15
Path Tracking (small torque)	789.30 ± 29.64	131.55 ± 54.61	920.85 ± 62.67
Path Tracking (large torque)	820.67 ± 42.41	161.67 ± 48.19	982.35 ± 81.07
Cartpole	4421.06 ± 78.97	3182.58 ± 879.25	7603.65 ± 881.36
2D Quadrotor	5704.92 ± 251.83	12050.59 ± 7258.85	17755.52 ± 7110.40
PVTOL	7100.86 ± 192.40	10240.46 ± 1554.67	17341.33 ± 1570.56
Ducted Fan	6132.89 ± 408.35	12129.13 ± 3419.57	18262.03 ± 3187.87
3D Quadrotor	4721.13 ± 306.55	–	4721.13 ± 306.55

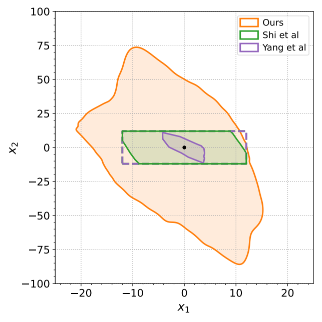
C.7 Computation Resource

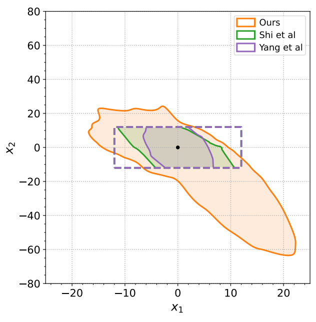
The training was performed on NVIDIA V100 gpus, each with 16GB memory. The verification was performed on NVIDIA 5090 gpus with 32GB memory.

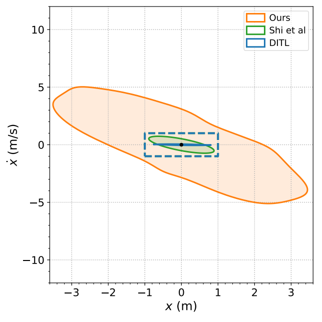

(a) Van Der Pol.

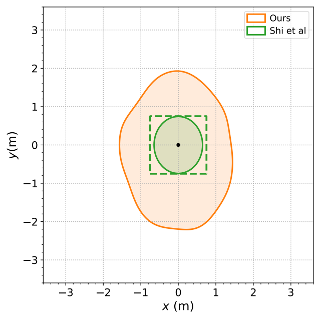

(b) Path Tracking large.

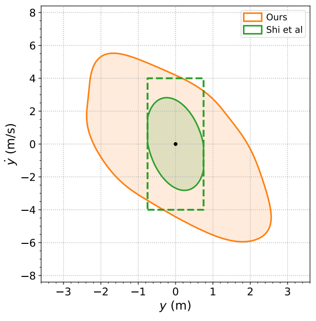

(c) Path Tracking small.

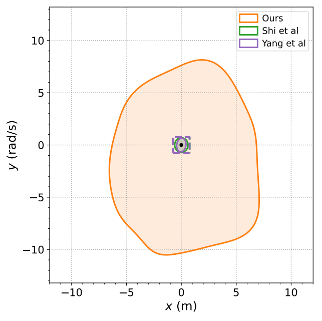

(d) Pendulum big.

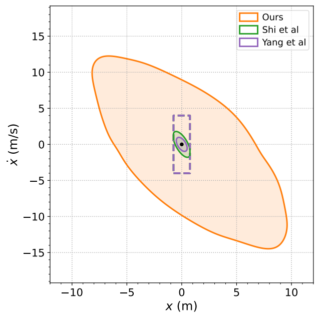

(e) Pendulum small.

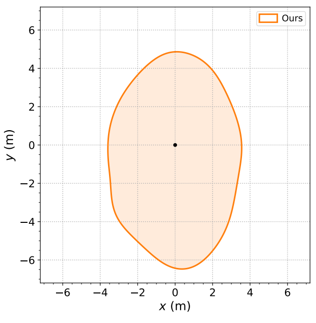

(f) Cartpole $(x, \dot{x})$ slice.


(g) PVTOL (x, y) slice.


(h) PVTOL $(y, \dot{y})$ slice.


(i) 2D Quadrotor (x, y) slice.


(j) 2D Quadrotor $(x, \dot{x})$ slice.


(k) Ducted fan (x, y) slice.

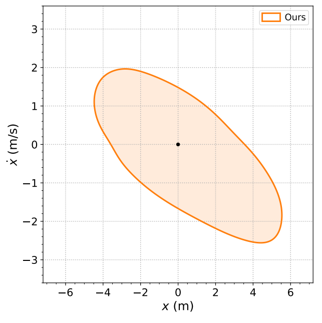

(l) Ducted fan $(x, \dot{x})$ slice.

Figure 7: More ROA visualizations.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Please see method and experiments section.

   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 clearly discussed the limitations in the conclusion section.

   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: One theorem about verification condition is included in theorem 3.1. Assumption is fully discussed, proof can be found in 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: Complete algorithms are exactly discussed. Hyperparameters are discussed in appendix, and a GitHub link containing codes we used can be found in abstract.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - 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 codes are open-sourced. All data used is generated along the way (details in code), no additional data source is needed.

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: All parameters can be found in codes. Hyperparameters are discussed also in appendix.

   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: Each method is trained 5 times with different seeds, and success rate is reported. Variance of the result is also reported.

   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: The machine being used is discussed in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - 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 conforms 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: There is no particular societal impact of the work that the authors can foresee that are important to discuss.

    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: No such risks are presented as far as the authors concern.

    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: [NA]

    Justification: Code is original, no other data is used.

    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: All codes are clearly documented.

    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**

    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]

    Justification: There are no such experiments.

    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: There are no such experiments.

    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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are only used for polishing the writing.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.