

Certified Training with Branch-and-Bound for Lyapunov-stable Neural Control

Zhouxing Shi

University of California, Riverside, CA 92521

ZHOUXING.SHI@UCR.EDU

Haoyu Li

University of Illinois Urbana-Champaign, IL 61801

HAOYULI5@ILLINOIS.EDU

Cho-Jui Hsieh

University of California, Los Angeles, CA 90095

CHOHSIEH@CS.UCLA.EDU

Huan Zhang

University of Illinois Urbana-Champaign, IL 61801

HUAN@HUAN-ZHANG.COM

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

We study the problem of learning verifiably Lyapunov-stable neural controllers that provably satisfy the Lyapunov asymptotic stability condition within a region-of-attraction (ROA). Unlike previous works that adopted counterexample-guided training without considering the computation of verification in training, we introduce Certified Training with Branch-and-Bound (CT-BaB), a new certified training framework that optimizes certified bounds, thereby reducing the discrepancy between training and test-time verification that also computes certified bounds. To achieve a relatively global guarantee on an entire input region-of-interest, we propose a training-time BaB technique that maintains a dynamic training dataset and adaptively splits hard input subregions into smaller ones, to tighten certified bounds and ease the training. Meanwhile, subregions created by the training-time BaB also inform test-time verification, for a more efficient training-aware verification. We demonstrate that CT-BaB yields verification-friendly models that can be more efficiently verified at test time while achieving stronger verifiable guarantees with larger ROA. On the largest output-feedback 2D Quadrotor system experimented, CT-BaB reduces verification time by over $11\times$ relative to the previous state-of-the-art baseline using Counterexample Guided Inductive Synthesis (CEGIS), while achieving $164\times$ larger ROA. Code is available at <https://github.com/shizhouxing/CT-BaB>.

Keywords: Lyapunov stability, certified training, neural network verification, branch-and-bound

1. Introduction

Deep learning with neural networks (NNs) has significantly advanced many domains, including control and robotic systems, where NN-based controllers are increasingly adopted. Despite their impressive capability, it remains challenging to obtain certified guarantees on the behaviors of NNs. However, these guarantees are critical for the trustworthy deployment of NNs in mission-critical domains (Ames et al., 2016; Bansal et al., 2017; Chang et al., 2019). In this work, we focus on learning NN-based controllers that are *formally verifiable* to be Lyapunov asymptotically stable in discrete-time nonlinear dynamical systems, which is usually approached by training the controller jointly with a Lyapunov function (Wu et al., 2023; Yang et al., 2024). Intuitively, a Lyapunov function provides an energy-like measure on the system state: it is positive definite and attains its global minimum at the equilibrium. The Lyapunov condition (Lyapunov, 1992) then requires that,

for every state within the region-of-attraction (ROA), the closed-loop update yields a strictly smaller Lyapunov function value, i.e., the system evolves toward lower “energy.” When these properties are verified, any trajectory initialized in the ROA is guaranteed to converge to the equilibrium, thereby establishing asymptotic stability.

Previous works (Wu et al., 2023; Yang et al., 2024) on this task were commonly based on the Counterexample Guided Inductive Synthesis (CEGIS) framework by iteratively searching for counterexamples and training models (both controllers and Lyapunov functions) on the counterexamples. The Lyapunov condition is subsequently verified by a formal verifier (Zhang et al., 2018; Xu et al., 2020) for the trained models. However, training on individual counterexamples does not consider the computation of verification required at test time (i.e., the offline formal verification after training is complete, which we call *test-time verification*), and thus the models are often not “verification-friendly”, leading to time-consuming verification. For instance, Yang et al. (2024) spent 8.9 hours to verify a controller for the 2D quadrotor system with output feedback. This difficulty also restricts the size of ROA that can be achieved.

In this paper, we propose a novel Certified Training framework enhanced with training-time Branch-and-Bound, namely **CT-BaB**. By “*certified training*” (Gowal et al., 2018; Mirman et al., 2019; Zhang et al., 2020; Shi et al., 2021), we optimize differentiable *certified bounds* on violations given *subregions* of inputs, rather than violations on *individual counterexamples* in CEGIS. Since test-time verification is also achieved by computing certified bounds, the training is now *verification-aware* to produce verification-friendly models. Unlike existing certified training works that primarily focused on the *local* robustness of NNs, we need relatively global guarantees on an entire input region-of-interest, which is too large for standard certified training to handle directly. To address this, we propose a *training-time branch-and-bound* (BaB) attached to certified training. We maintain a training dataset that consists of subregions within the region-of-interest, and we dynamically split hard subregions into smaller ones, which eases the training with tighter certified bounds. Although verification with more extensive BaB is still conducted at test time to fully verify models, subregions created during our training can be exported to bootstrap and significantly speed up test-time verification, making the verification *training-aware*.

We demonstrate CT-BaB on learning asymptotically Lyapunov-stable NN controllers, and to the best of our knowledge, this is the first certified training approach for this task. We show that CT-BaB produces models with much larger ROA, while enabling much faster test-time verification, thus producing verification-friendly models with stronger guarantees. Notably, compared to the previous state-of-the-art (Yang et al., 2024), on the largest 2D quadrotor system: for the state feedback setting, CT-BaB reduces the verification time from 1.1hrs to 49s (**80× faster**) while yielding a **16× larger ROA**; for the output feedback setting, CT-BaB reduces the verification time from 8.9hrs to 0.8hrs (**11× faster**) with a **164× larger ROA**.

2. Methodology

2.1. Background and Problem Settings

Lyapunov-stable neural control. We focus on learning verifiably stable neural controllers with Lyapunov asymptotic stability guarantees (Lyapunov, 1992), for a nonlinear discrete-time dynamical system with continuous control actions. We adopt the problem formulation from Yang et al. (2024). Suppose the input state has d dimensions, a nonlinear dynamical system is defined as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, u(\mathbf{x}_t)), \quad (1)$$

where \mathbf{x}_t is the state at the current time step t , a continuous control input $u(\mathbf{x}_t) \in \mathbb{R}^{n_u}$ is generated by a NN-based controller, and the system dynamics determines the next state \mathbf{x}_{t+1} . It is assumed that the system dynamics are known and there exists an equilibrium \mathbf{x}^* such that $\mathbf{x}^* = f(\mathbf{x}^*, u(\mathbf{x}^*))$. The analysis focuses on a region-of-interest (ROI) $\mathcal{B} \in \mathbb{R}^d$ containing the equilibrium, i.e., $\mathbf{x}^* \in \mathcal{B}$. We assume \mathcal{B} is an axis-aligned bounding box $\mathcal{B} = \{\mathbf{x} \mid \underline{\mathbf{b}} \leq \mathbf{x} \leq \bar{\mathbf{b}}, \mathbf{x} \in \mathbb{R}^d\}$ with boundary $\underline{\mathbf{b}}, \bar{\mathbf{b}} \in \mathbb{R}^d$ (“ \leq ” for vectors is elementwise here). Note that for simplicity, we show a state feedback setting here; but we have also considered output feedback settings following Yang et al. (2024), where a controller only has access to observed system outputs with state estimation by an observer, rather than the true state, and \mathbf{x}_t is augmented to account for state estimation errors (discussed further in Appendix A.1).

The goal is to guarantee that if the system starts from any state $\mathbf{x} \in \mathcal{S}$ within some region-of-attraction (ROA), $\mathcal{S} \subseteq \mathcal{B}$, it will converge to the equilibrium \mathbf{x}^* . To achieve this, a Lyapunov function $V(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ is to be learned jointly with the controller. The ROA is specified with a sublevel set within \mathcal{B} , as $\mathcal{S} := \{\mathbf{x} \in \mathcal{B} \mid V(\mathbf{x}) < \rho\}$ with threshold ρ . We aim to have the following property verified:

$$\forall \mathbf{x}_t \in \mathcal{B}, V(\mathbf{x}_t) < \rho \rightarrow \left(V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t) \leq -\kappa V(\mathbf{x}_t) \right) \wedge (\mathbf{x}_{t+1} \in \mathcal{B}), \quad (2)$$

where $\kappa > 0$ is a constant specifying the exponential stability convergence rate. When the positive definiteness of $V(\cdot)$, $V(\mathbf{x}^*) = 0$ and $V(\mathbf{x}) > 0$ ($\forall \mathbf{x} \neq \mathbf{x}^*$), is guaranteed by construction as discussed in Appendix A.1, and Eq. (2) is verified for the entire ROI ($\forall \mathbf{x}_t \in \mathcal{B}$), the Lyapunov asymptotic stability is guaranteed for ROA \mathcal{S} (Yang et al., 2024). To clarify, since all possible states within \mathcal{B} are considered when verifying Eq. (2), the verification can be independent from time step t and does not require reasoning over multi-step trajectories.

Training problem. Both $u(\cdot)$ and $V(\cdot)$ are modeled as neural networks due to their expressive power, and we follow Yang et al. (2024) to use simple feedforward architectures as detailed in Appendix A.1. Suppose $g_\theta(\mathbf{x}_t)$ is a model that characterizes the violation of Eq. (2) on a given \mathbf{x}_t , parameterized by θ . $g_\theta(\mathbf{x}_t)$ depends on $u(\cdot)$ and $V(\cdot)$, and state \mathbf{x}_t is the input to the model. Given Eq. (2), we have:

$$g_\theta(\mathbf{x}_t) := \min \left\{ \rho - V(\mathbf{x}_t), \sigma(V(\mathbf{x}_{t+1}) - (1 - \kappa)V(\mathbf{x}_t)) + \sum_{1 \leq i \leq d} \sigma([\mathbf{x}_{t+1}]_i - \bar{\mathbf{b}}_i) + \sigma(\underline{\mathbf{b}}_i - [\mathbf{x}_{t+1}]_i) \right\}, \quad (3)$$

where $\sigma(x) := \max\{x, 0\}$ is known as ReLU. We aim to achieve a verifiable

$$\forall \mathbf{x}_t \in \mathcal{B}, \quad g_\theta(\mathbf{x}_t) \leq 0, \quad (4)$$

which can be written as a min-max optimization problem as $\arg \min_\theta \arg \max_{\mathbf{x}_t \in \mathcal{B}} g_\theta(\mathbf{x}_t)$, where θ denotes the parameters of both $u(\cdot)$ and $V(\cdot)$, jointly optimized during training. Note that Eq. (4) alone is insufficient: it could be trivially satisfied with a vacuously small ROA (e.g., by driving $V(\mathbf{x})$ to be large everywhere so that $V(\mathbf{x}) < \rho$ holds for almost no state). An additional loss term to encourage a large ROA is therefore necessary, as discussed in Section 2.4.

2.2. CT-BaB: Certified Training Framework with Training-time Branch-and-bound

We illustrate our proposed CT-BaB framework in Figure 1 and explain it below.

Certified upper bound for verification. We first explain the verification for Eq. (4). Recent computationally efficient methods typically compute a certified bound $\bar{g}_\theta(\mathcal{B})$ such that $\bar{g}_\theta(\mathcal{B}) \geq g_\theta(\mathbf{x}_t)$ ($\forall \mathbf{x}_t \in \mathcal{B}$) provably holds, and then $\bar{g}_\theta(\mathcal{B}) \leq 0$ implies a successful verification. State-of-the-art NN verifiers such as α, β -CROWN (Zhang et al., 2018; Xu et al., 2020, 2021; Wang et al., 2021;

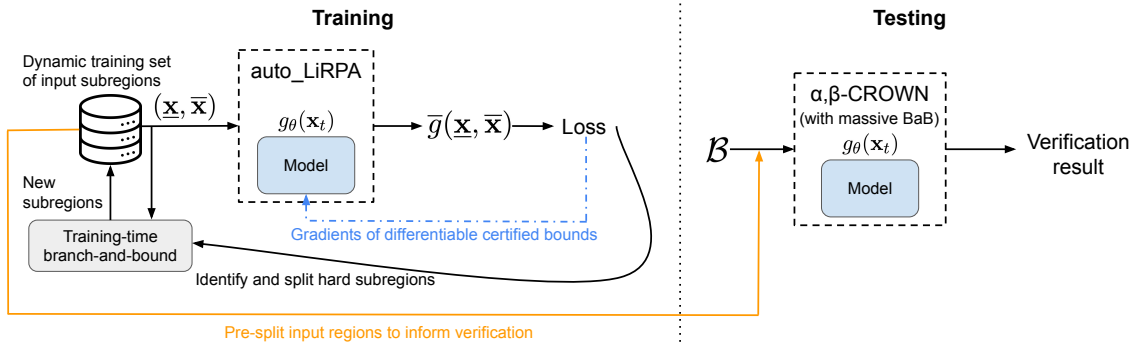


Figure 1: Illustration of our proposed CT-BaB framework.

Zhang et al., 2022; Shi et al., 2025) leverage a technique called *linear bound propagation* (Zhang et al., 2018; Wong and Kolter, 2018; Singh et al., 2019; Xu et al., 2020) for computing certified bounds, which relaxes nonlinear operators in the model by linear lower and upper bounds, and it then propagates linear bounds through the model to finally bound the output w.r.t. the input region. This technique has been generalized to support general computational graphs (i.e., general model architectures and problem specifications, including the problem here) in the auto_LiRPA framework (Xu et al., 2020) used by α, β -CROWN. Additionally, since initial certified bounds may not be tight enough for a successful verification, due to the over-approximation from linear relaxation, a branch-and-bound (BaB) technique is also commonly used by branching the verification problem into smaller subproblems and computing tighter certified bounds for smaller subproblems (Bunel et al., 2018, 2020; Wang et al., 2021). Since current works on this Lyapunov stability task commonly focus on systems with low-dimensional inputs, Yang et al. (2024) configured α, β -CROWN to run BaB by splitting the state (i.e., input \mathbf{x}_t to model $g_\theta(\mathbf{x}_t)$, not internal activations)¹.

Limitation of CEGIS. In previous works using CEGIS (Wu et al., 2023; Yang et al., 2024), each training iteration basically searches for counterexamples $\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \dots \in \mathcal{B}$ such that $g_\theta(\mathbf{x}_t^{(1)}), g_\theta(\mathbf{x}_t^{(2)}), \dots > 0$. Then it optimizes the model on a finite number of counterexamples as $\arg \min_\theta (g_\theta(\mathbf{x}_t^{(1)}) + g_\theta(\mathbf{x}_t^{(2)}) + \dots)$. While it often eliminates noticeable counterexamples during training, i.e., achieving $g_\theta(\mathbf{x}) \leq 0$ for concrete data points \mathbf{x} , they generally do not make certified bounds $\bar{g}(\mathcal{B})$ tight enough for efficient verification. This is because the computation of certified bounds is not considered in training, and there is a significant discrepancy between $g_\theta(\cdot)$ optimized during training and certified bounds $\bar{g}_\theta(\cdot)$ computed at test time. This limitation leads to extensive BaB required at test time to tighten certified bounds and thus costly verification, particularly given the difficulty of verification for NNs (Salman et al., 2019).

Certified training for verification-friendly models. In contrast, we propose to conduct certified training, where we incorporate the computation of verification into the training and optimize certified bounds. Specifically, we essentially optimize the model by $\arg \min_\theta \sigma(\bar{g}_\theta(\mathcal{B}))$, where $\bar{g}_\theta(\mathcal{B})$ is computed by auto_LiRPA (Xu et al., 2020) with linear bound propagation, thereby reducing the discrepancy between training and test-time verification and making trained models more verification-friendly. This training is possible because $\bar{g}_\theta(\mathcal{B})$ computed by linear bound propaga-

1. Although α, β -CROWN usually conducts BaB by splitting activations and did not explicitly discuss the use of input splitting in papers, input splitting has in fact been used to verify many low-dimensional systems.

tion is differentiable, as both linear relaxation and each linear bound propagation step consist of differentiable operations that can be used in training (Zhang et al., 2020; Xu et al., 2020).

Training-time branch-and-bound. Previous certified training works (Gowal et al., 2018; Mirman et al., 2018; Shi et al., 2021; Wang et al., 2022; Müller et al., 2023; De Palma et al., 2022; Mao et al., 2023) commonly focused on adversarial robustness within small local neighborhoods, e.g., $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_0\| \leq \epsilon\}$ around each individual example $\mathbf{x}_0 \in \mathcal{B}$ (suppose \mathcal{B} is still the set of all possible inputs), with a small radius ϵ (Goodfellow et al., 2015). For them, each certified bound computation only needs to handle a small local region within \mathcal{B} . However, unlike those previous works, we desire guarantees on the entire \mathcal{B} (as Eq. (4)). Since \mathcal{B} covers all the states that the model cares about, it is large and relatively global, and directly computing $\bar{g}_\theta(\mathcal{B})$ leads to extremely loose certified bounds. Naively, one may split \mathcal{B} into many small subregions in the beginning. However, this would be inefficient, as the difficulty of computing relatively tight certified bounds varies for different subregions, and uniformly splitting \mathcal{B} in the beginning cannot identify and spend much more splitting on the hardest subregions. To address this, we propose a novel *training-time branch-and-bound* technique. Throughout the training, we dynamically split \mathcal{B} into smaller subregions, and we maintain a dataset with n examples: $\mathbb{D} = \{(\underline{\mathbf{x}}^{(1)}, \bar{\mathbf{x}}^{(1)}), (\underline{\mathbf{x}}^{(2)}, \bar{\mathbf{x}}^{(2)}), \dots, (\underline{\mathbf{x}}^{(n)}, \bar{\mathbf{x}}^{(n)})\}$, where each example $(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)})$ ($1 \leq k \leq n$) is a subregion in \mathcal{B} , defined as a bounding box $\{\mathbf{x} : \mathbf{x} \in \mathbb{R}^d, \underline{\mathbf{x}}^{(k)} \leq \mathbf{x} \leq \bar{\mathbf{x}}^{(k)}\}$. All these examples are non-overlapping and cover \mathcal{B} . We dynamically update and expand the dataset during training, by splitting hard examples into more examples with even smaller subregions, as detailed in Section 2.3.

Empirical training objective. In addition to the main training objective with certified bounds, we also add a term where we try to empirically find the worst-case violation of Eq. (4) by projected gradient descent (PGD) (Madry et al., 2018). It can make the training more quickly reach a solution with most counterexamples eliminated, for the certified training to focus on making it verifiable. It also helps to achieve that at least no counterexample can be empirically found, even when not all the subregions in \mathbb{D} can be verified yet, as we may fully verify Eq. (4) using α, β -CROWN with more extensive BaB at test time. Specifically, we denote this objective as $\bar{g}^A(\underline{\mathbf{x}}, \bar{\mathbf{x}}) := g(A(\underline{\mathbf{x}}, \bar{\mathbf{x}}))$, where $A(\underline{\mathbf{x}}, \bar{\mathbf{x}})$ ($\underline{\mathbf{x}} \leq A(\underline{\mathbf{x}}, \bar{\mathbf{x}}) \leq \bar{\mathbf{x}}$) is a data point empirically found by PGD to maximize $g(A(\underline{\mathbf{x}}, \bar{\mathbf{x}}))$, as $\arg \max_{\mathbf{x} \in \mathbb{R}^d} (\underline{\mathbf{x}} \leq \mathbf{x} \leq \bar{\mathbf{x}}) g(\mathbf{x})$, but this maximization by PGD is not guaranteed to be optimal.

Overall training objective. Overall, we optimize for the following loss function:

$$L = \frac{1}{|\mathbb{D}|} \sum_{(\underline{\mathbf{x}}, \bar{\mathbf{x}}) \in \mathbb{D}} \left(\sigma(\bar{g}(\underline{\mathbf{x}}, \bar{\mathbf{x}}) + \epsilon) + \lambda \sigma(\bar{g}^A(\underline{\mathbf{x}}, \bar{\mathbf{x}}) + \epsilon) \right) + L_{\text{extra}}, \quad \text{where } \bigcup_{(\underline{\mathbf{x}}, \bar{\mathbf{x}}) \in \mathbb{D}} = \mathcal{B}, \quad (5)$$

ϵ is small value for ideally eliminating the violation with a margin, λ is a coefficient for weighting the PGD term, and L_{extra} is an extra loss term for controlling additional properties as discussed in Section 2.4. We monitor training convergence by tracking the proportion of verified subregions in \mathbb{D} (i.e., those with $\bar{g}(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)}) \leq 0$). After the training, we use α, β -CROWN to verify Eq. (4), and thus the soundness of the trained models is guaranteed as long as the final verification succeeds. It is worth noting that CT-BaB is generally formulated, as $g(\cdot)$ can potentially model other properties from various applications, but we focus on Lyapunov asymptotic stability in this work.

2.3. Training-Time Branch-and-Bound

As discussed in Section 2.2, our training-time BaB aims to address challenges in obtaining relatively tight certified bounds when we require guarantees on the entire \mathcal{B} . We introduce it in more detail.

Initial splits. We initialize \mathbb{D} by splitting \mathcal{B} into grids along each of its d dimensions, respectively. We control the maximum size of the initial regions with a threshold l that denotes the maximum length of each input dimension. For each input dimension i ($1 \leq i \leq d$), we uniformly split the input range $[\underline{b}_i, \bar{b}_i]$ into $m_i = \lceil \frac{\bar{b}_i - \underline{b}_i}{l} \rceil$ segments in the initial split, such that the length of each segment is no larger than the threshold l . We thereby create $\prod_{i=1}^d m_i$ regions to initialize \mathbb{D} . We set l such that the number of initial examples is close to the batch size.

Dynamic splits in training. During training, we dynamically split hard regions into even smaller subregions after every epoch. For each training batch, we take each $(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)})$ with $\bar{g}(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)}) > 0$, i.e., we have not been able to verify that $g(\mathbf{x}) \leq 0$ for region $[\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)}]$. We then uniformly split the region into two subregions along a chosen input dimension i ($1 \leq i \leq d$), and we replace the original region with the two new subregions in the dataset. To maximize the benefit of splitting an example, we choose the input dimension by considering each input dimension j ($1 \leq j \leq d$) and computing the total loss of the two new subregions when dimension j is split. Suppose $L(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)})$ is the contribution of an example $(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)})$ to the loss function in Eq. (5). We take the dimension j that leads to the lowest loss value for the new examples to split:

$$\arg \min_{1 \leq j \leq d} L(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k,j)}) + L(\underline{\mathbf{x}}^{(k,j)}, \bar{\mathbf{x}}^{(k)}), \quad \text{where } \underline{\mathbf{x}}_j^{(k,j)} = \bar{\mathbf{x}}_j^{(k,j)} = \frac{\underline{\mathbf{x}}_j^{(k)} + \bar{\mathbf{x}}_j^{(k)}}{2}, \quad (6)$$

and $\underline{\mathbf{x}}_i^{(k,j)} = \underline{\mathbf{x}}_i^{(k)}$, $\bar{\mathbf{x}}_i^{(k,j)} = \bar{\mathbf{x}}_i^{(k)}$ keep unchanged for other dimensions $i \neq j$. This only mildly affects the total training cost. As training progresses, most subregions are verified and thus no longer require splitting, and different regions to split and input dimensions to consider can be handled in parallel on GPUs, though further training may introduce new counterexamples in previously verified subregions, requiring those subregions to be split again. The splitting may terminate primarily when the dataset \mathbb{D} reaches a maximum size, as the system memory can be limited (in our experiments, the limit is never reached). Optionally, splitting can also be stopped after a fixed number of training steps, which is not required in our experiments.

Training-aware verification. As mentioned in Section 2.2, α, β -CROWN with more extensive BaB is used at test time to finally verify the model. When the training converges, CT-BaB can leave a small proportion of training subregions unverified (e.g., up to 6% in our results), and these subregions require further splitting. By default, α, β -CROWN starts splitting from the original \mathcal{B} and is likely to take different splitting paths compared to our training-time BaB. To further reduce the discrepancy between training and testing, we propose *training-aware verification* by exporting training subregions to inform test-time verification. We modify the α, β -CROWN verifier to allow loading so-called ‘‘pre-split’’ input regions, i.e., input regions from our final \mathbb{D} that have already been split from \mathcal{B} during training prior to verification. Thereby, our training-time BaB can bootstrap and further speed-up the test-time verification.

2.4. Controlling the ROA

As shown in Eq. (5), an additional term L_{extra} can be added to control additional properties. We use this term to control the size of the region-of-attraction (ROA). We aim to have a good proportion of data points $\mathbf{x} \in \mathcal{B}$ that are within the ROA as $V(\mathbf{x}) < \rho$. We randomly draw n_ρ samples within \mathcal{B} , as $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_{n_\rho} \in \mathcal{B}$ (these samples are independent from $(\underline{\mathbf{x}}^{(k)}, \bar{\mathbf{x}}^{(k)})$ used for other terms in Eq. (5)), and we define L_{extra} as:

$$L_{\text{extra}} = \mathbb{I} \left(\frac{1}{n_\rho} \sum_{i=1}^{n_\rho} \mathbb{I}(V(\tilde{\mathbf{x}}_i) < \rho) < \rho_{\text{ratio}} \right) \frac{\lambda_\rho}{n_\rho} \sum_{i=1}^{n_\rho} \sigma(V(\tilde{\mathbf{x}}_i) + \rho - \epsilon), \quad (7)$$

which penalizes samples with $V(\tilde{\mathbf{x}}_i) > \rho - \epsilon$ when the ratio of samples within the sublevel set is below the threshold ρ_{ratio} , thereby encouraging more samples to be within the ROA. Here ϵ is a small value for the margin as similarly used in Eq. (5) and λ_ρ is for weighting the L_{extra} term in Eq. (5). In our implementation, we simply fix $\rho = 1$ and make n_ρ equal to the batch size. The threshold ρ_{ratio} and the weight λ_ρ can be set to reach the desired ROA size, although setting a stricter requirement on ROA naturally tends to increase the difficulty of training. Compared to the loss term for ROA in Yang et al. (2024) which required carefully selecting candidate states that are desired to be within the ROA by referring to classical LQR solutions, ours is self-contained and does not refer to any other solution, and thus can reduce the burden of applying our method. Note that this term is a necessary component of the training: without it, the Lyapunov condition Eq. (2) could be trivially satisfied by collapsing the ROA to a negligibly small region.

3. Experiments

Additional experimental details are included in Appendix A.

Table 1: Dynamical systems in the experiments, all following previous works (Yang et al., 2024; Wu et al., 2023). d and n_u represent the dimension of input state and control input, respectively. There is a limit on u which is clamped according to the limit, where some symbols are from the system dynamics: m for mass, g for gravity, l for length, and v for velocity. Size of the region-of-interest is denoted by the upper boundary $\bar{\mathbf{b}}$ (with $\underline{\mathbf{b}} = -\bar{\mathbf{b}}$). Equilibrium state of all the systems is $\mathbf{x}^* = \mathbf{0}$. Systems with “(output)” are output feedback settings, and others are state feedback settings.

System	d	n_u	Limit on u	Region-of-interest (denoted as upper bound of box)
Inverted Pendulum	2	1	$ u \leq 8.15 \cdot mgl$ (large torque) $ u \leq 1.02 \cdot mgl$ (small torque)	[12, 12]
Inverted Pendulum (output)	4	1	$ u \leq \frac{mgl}{3}$	[0.7 π , 0.7 π , 0.175 π , 0.175 π]
Path Tracking	2	1	$ u \leq 1.68 \cdot l/v$ (large torque) $ u \leq l/v$ (small torque)	[3, 3]
Cart-Pole	4	1	$ u \leq 30$	[1, 1, 1, 1]
PVTOL	6	2	$\ u\ _\infty \leq mg$	[0.75, 0.75, π , 4, 4, 3]
2D Quadrotor	6	2	$\ u\ _\infty \leq 1.25 \cdot mg$	[0.75, 0.75, π , 4, 4, 3]
2D Quadrotor (output)	8	2	$\ u\ _\infty \leq 1.5 \cdot mg$	[0.1, 0.2 π , 0.2, 0.2 π , 0.05, 0.1 π , 0.1, 0.1 π]

Dynamical systems. We experiment on several dynamical systems following Wu et al. (2023); Yang et al. (2024), as listed in Table 1: *Inverted Pendulum* is about swinging up a pendulum to the upright equilibrium; *Path Tracking* is about tracking the path for a planar vehicle; *Cart-Pole* is about balancing an Inverted Pendulum mounted on a Cart-Pole; *PVTOL* is about drone planar vertical takeoff and landing; and *2D Quadrotor* is about hovering a planar quadrotor. For Inverted Pendulum and Path Tracking, there are two different limits on the maximum allowed torque for the controller, where the setting is more challenging with a smaller torque limit. Unless otherwise noted, state feedback is considered for all these systems. Output feedback is additionally considered for Inverted Pendulum and 2D Quadrotor, following Yang et al. (2024). Their system dynamics are attached at Appendix C.

Baselines and Metrics. We compare with previous works on the same task, i.e., learning NN-based controllers with verified Lyapunov asymptotic stability for discrete-time dynamical systems (Wu

Table 2: Comparison on the verification time cost and the size of ROA. Model checkpoints for Wu et al. (2023) are obtained from the source code of Yang et al. (2024) and the same models have been used for comparison in Yang et al. (2024). “-” denotes that the models for Wu et al. (2023); Yang et al. (2024) are not available on some of the systems (although Wu et al. (2023) originally had models for PVTOL, an issue was found by Yang et al. (2024) and the training could not work after the issue was fixed). “+Loading” indicates that we load training subregions into α, β -CROWN for a training-aware verification, as discussed in Section 2.3.

System	Wu et al. (2023)		CEGIS (Yang et al., 2024)		CT-BaB (Ours)		
	Time	ROA	Time	ROA	Time (+Loading)	ROA	
Inverted Pendulum (large torque)	11.3s	53.28	33s	239.04	12.4s	1.8s	505.4
Inverted Pendulum (small torque)	-	-	25s	187.20	12.0s	2.2s	457.9
Inverted Pendulum (output)	-	-	94s	6.26	48.8s	5.3s	12.23
Path Tracking (large torque)	11.7s	14.38	39s	18.27	19.2s	3.0s	20.79
Path Tracking (small torque)	-	-	34s	10.53	17.2s	3.6s	14.03
Cart-Pole	2.2min	0.037	-	-	68s	2.7s	1.35
PVTOL	-	-	-	-	109s	16s	46.95
2D Quadrotor	-	-	1.1hrs	3.29	144s	49s	54.39
2D Quadrotor (output)	-	-	8.9hrs	6.7e-9	>12hrs	0.8hrs	1.1e-6

et al., 2023; Yang et al., 2024). Among them, Yang et al. (2024) is the previous state-of-the-art, and they both outperform classical non-learning methods such as LQR (Tedrake et al., 2010). For comparison, we mainly use the verification time cost and size of ROA as the metrics. We follow Wu et al. (2023) to estimate the size of ROA. We consider grid points in the region-of-interest \mathcal{B} and count the proportion of grid points within the verified \mathcal{S} , multiplied by the volume of \mathcal{B} . Hyperparameters for all the model architectures follow Yang et al. (2024).

Main Results. We show the main results in Table 2. On all the settings where any baseline is applicable, CT-BaB yields much larger ROA and much smaller verification time cost. Among the experimented systems, Inverted Pendulum, Path Tracking, and Cart-Pole are relatively small in terms of the number of input states (up to 4), where test-time verification for our models completes within 4s. For PVTOL, previously only Wu et al. (2023) trained a model but their training could not work after an implementation issue was found and fixed by Yang et al. (2024); our training works successfully on PVTOL. On the two largest systems, 2D Quadrotor with state feedback and output feedback, respectively, our improvement over the previous state-of-the-art Yang et al. (2024) is particularly significant. Specifically, for the state feedback setting, we reduce the verification time from 1.1hrs to 49s, while enlarging the ROA by $16\times$; for the output feedback setting, we reduce the verification time from 8.9hrs to 0.8hrs, while enlarging the ROA by $164\times$. We provide a visualization of ROA in Appendix B.2. These results demonstrate the significant advantage of our CT-BaB method for producing verification-friendly models (i.e., models that can be verified in much shorter time) with stronger verifiable guarantees (i.e., larger ROA where the Lyapunov stability is guaranteed).

Advantage of training-aware verification. As proposed in Section 2.3, our training-aware verification is informed by training-time BaB and loads training subregions. In Table 2, the results show that loading training subregions significantly reduces the verification time cost on all the settings. On the largest 2D Quadrotor with output feedback, the original verification times out after trying

verification for 12 hours, which is reasonable considering that our ROA is $164\times$ larger than the one in Yang et al. (2024) verified with 8.9 hours; in contrast, by loading our training subregions, the verification completes within 0.8 hours. On other systems, even without loading training subregions, our verification time cost is still much smaller compared to Yang et al. (2024) while our ROA is much larger, which similarly holds for Cart-Pole where only results from Wu et al. (2023) are available. Although Wu et al. (2023) only took approximately 11s to verify systems on Inverted Pendulum and Path Tracking (large torque), their ROA is much smaller compared to Yang et al. (2024) and ours. Overall, our results demonstrate that CT-BaB not only produces verification-friendly models but also enables training-aware verification to further speed up the verification.

Table 3: Runtime of training, the size of the training dataset, the ratio of training subregions verifiable without further BaB at the end of the training, and the number of BaB domains visited by α, β -CROWN during verification. All the models are fully verified by α, β -CROWN with more BaB.

System	Runtime	Dataset size		W/o more BaB	Verified	
		Initial	Final		By α, β -CROWN	BaB domains
Inverted Pendulum (large torque)	1.7min	58,080	68,686	99.9993%	100%	12
Inverted Pendulum (small torque)	11.0min	58,080	2,286,676	99.91%	100%	14,196
Inverted Pendulum (output)	11.8min	19,200	3,411,802	99.7%	100%	134,187
Path Tracking (large torque)	3.3min	40,400	596,100	98.4%	100%	60,079
Path Tracking (small torque)	14.9min	40,400	1,325,095	99.6%	100%	78,144
Cart-Pole	48.7min	12,480	3,795,905	99.95%	100%	17,940
PVTOL	1hrs	46,336	18,977,973	95.6%	100%	40,603,447
2D Quadrotor	1.5hrs	46,336	29,015,573	95.6%	100%	154,390,951
2D Quadrotor (output)	2.0hrs	44,032	20,448,758	94.0%	100%	2,745,392,253

Training time and data. In Table 3, we show more information about the training, including the time cost of training and size of the dynamic training dataset. Our training dataset is dynamically maintained as described in Section 2.3, and the dataset size grows from the “initial” size to the “final” size. At the end of the training, most training subregions (at least 94.0%) can be verified without further BaB, and all the models can be fully verified by α, β -CROWN with more extensive BaB at test time. A detailed comparison of runtime against Yang et al. (2024) is provided in Table 4; although the training time of CT-BaB is often higher than Yang et al. (2024), CT-BaB achieves the lowest total time cost on the largest 2D Quadrotor with output feedback setting, demonstrating its potential for scaling up.

4. Related Work

Lyapunov-stable controllers. Some works on Lyapunov stability focused constructing Lyapunov functions for a given dynamical system, without jointly learning a controller (Alfarano et al., 2023; Zou et al., 2025; Liu et al., 2025; Giesl et al., 2026). Classical methods such as linear quadratic regulator (LQR) (Tedrake et al., 2010) and sum-of-squares (SOS) (Parrilo, 2000; Majumdar et al., 2013; Yang et al., 2023; Dai and Permenter, 2023) can synthesize stable controllers with guarantees but are oftentimes limited to linear or polynomial controllers. In contrast, learning NN-based controllers has shown great potential for more complicated systems and larger ROA. Many works only used sampling without formal guarantees (Jin et al., 2020; Sun and Wu, 2021; Dawson et al., 2022; Liu et al., 2023; Min et al., 2023). Although some works such as Jin et al. (2020) theoretically

considered verification, they assumed the existence of a Lipschitz constant that was not practically computed.

To achieve formal verification for NN-based controllers, previous works commonly adopted a Counter-Example Guided Inductive Synthesis (CEGIS) approach by iteratively searching for counterexamples and optimizing the models to eliminate the counterexamples. The counterexamples can be generated by Satisfiable Modulo Theories (SMT) solver (Gao et al., 2013; De Moura and Bjørner, 2008; Chang et al., 2019; Abate et al., 2020), Mixed Integer Programming (MILP) solver (Dai et al., 2021; Chen et al., 2021; Wu et al., 2023), or more efficiently, projected gradient descent (PGD) (Madry et al., 2018; Wu et al., 2023; Yang et al., 2024). Among them, Wu et al. (2023) used a verifier (Xu et al., 2020) but only to guarantee the positive definiteness of the Lyapunov function (which can also be directly achieved by construction, as done in Yang et al. (2024)) and not the main Lyapunov condition. To our knowledge, we are the first to incorporate certified bounds for the violation of the Lyapunov condition during training, to produce verification-friendly models with stronger guarantees, while informing test-time verification for a training-aware verification.

Safety verification beyond stability. Apart from Lyapunov stability, there are also many works on verifying other safety properties of neural controllers, such as reachability (Althoff and Kochdumper, 2016; Tran et al., 2020; Hu et al., 2020; Everett et al., 2021; Ivanov et al., 2021; Knuth et al., 2021; Huang et al., 2022; Wang et al., 2023b; Schilling et al., 2022; Chatterjee et al., 2023; Kochdumper et al., 2023; Jafarpour et al., 2023, 2024; Teuber et al., 2024; Badings et al., 2025), forward invariance (Ames et al., 2016; Zhao et al., 2021; Wang et al., 2023a; Huang et al., 2023; Harapanahalli and Coogan, 2024; Hu et al., 2024; Wang et al., 2024), and contraction (Tsukamoto and Chung, 2020; Fitzsimmons and Liu, 2024; Li et al., 2025). The Lyapunov asymptotic stability we study is a relatively strong guarantee, as it ensures convergence towards an equilibrium.

5. Conclusion

To conclude, we propose CT-BaB for training Lyapunov asymptotically stable NN-based controllers and achieving verifiable guarantees on an entire input region-of-interest. CT-BaB is verification-aware by optimizing certified bounds during training, and it handles a relatively large region-of-interest by a new training-time BaB. It also informs test-time verification by the dynamic training dataset created during training, for a more efficient training-aware verification. We have demonstrated that CT-BaB consistently produces more verification-friendly models with stronger guarantees, i.e., much smaller verification cost at test time with much larger ROA.

A limitation of this work is that only low-dimensional dynamical systems have been considered, which is also a common limitation of previous works on Lyapunov stability (Chang et al., 2019; Wu et al., 2023; Yang et al., 2024). Even after the training, the verification may still fail. In this case, like prior works, we may continue training, as further optimization of the certified bounds may yield a more verification-friendly model. However, there may be fundamental challenges that prevent successful verification, for instance when the system dimensionality is too large. Future works may try to scale up our method to higher-dimensional systems, by potentially conducting training-time BaB on learned internal activations, not only inputs. Additionally, although we focus on Lyapunov stability in this work, our CT-BaB is generally formulated, and thus we envision that CT-BaB can potentially be applied to other types of safety properties in future work.

Acknowledgments

This project is supported in part by NSF 2048280, 2331966 and ONR N00014-23-1-2300:P00001. Huan Zhang is supported in part by the AI2050 program at Schmidt Sciences (AI2050 Early Career Fellowship) and NSF (IIS-2331967).

References

- Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.
- Alberto Alfarano, François Charton, Amaury Hayat, and CERMICS-Ecole des Ponts Paristech. Discovering lyapunov functions with transformers. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS*, volume 23, 2023.
- Matthias Althoff and Niklas Kochdumper. Cora 2016 manual. *TU Munich*, 85748, 2016.
- 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.
- Thom Badings, Wietze Koops, Sebastian Junges, and Nils Jansen. Policy verification in stochastic dynamical systems using logarithmic neural certificates. In *International Conference on Computer Aided Verification*, pages 349–375. Springer, 2025.
- Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, 2017.
- Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems*, pages 4795–4804, 2018.
- Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Branch and bound for piecewise linear neural network verification. *J. Mach. Learn. Res.*, 21:42:1–42:39, 2020.
- Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, pages 3240–3249, 2019.
- Krishnendu Chatterjee, Thomas A Henzinger, Mathias Lechner, and Dorde Zikelic. A learner-verifier framework for neural network controllers and certificates of stochastic systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 3–25. Springer, 2023.
- 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.

- Hongkai Dai and Frank Permenter. Convex synthesis and verification of control-lyapunov and barrier functions with input constraints. In *IEEE American Control Conference (ACC)*, 2023.
- Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.
- Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, 2022.
- 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*, 2008.
- Alessandro De Palma, Rudy Bunel, Krishnamurthy Dvijotham, M Pawan Kumar, and Robert Stanforth. Ibp regularization for verified adversarial robustness via branch-and-bound. *arXiv preprint arXiv:2206.14772*, 2022.
- Michael Everett, Golnaz Habibi, Chuangchuang Sun, and Jonathan P How. Reachability analysis of neural feedback loops. *IEEE Access*, 2021.
- Maxwell Fitzsimmons and Jun Liu. Computation and formal verification of neural network contraction metrics. *IEEE Control Systems Letters*, 2024.
- 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, 2013.
- P Giesl, S Hafstein, B Hamzi, J Lee, H Owhadi, G Santin, and U Vaidya. Kernel methods for the construction of certified lyapunov functions via approximate koopman eigenfunctions. *arXiv preprint arXiv:2602.21767*, 2026.
- Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.
- Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.
- Akash Harapanahalli and Samuel Coogan. Certified robust invariant polytope training in neural controlled odes. *arXiv preprint arXiv:2408.01273*, 2024.
- Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George J Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In *2020 59th IEEE conference on decision and control (CDC)*, pages 5929–5934, 2020.
- Hanjiang Hu, Yujie Yang, Tianhao Wei, and Changliu Liu. Verification of neural control barrier functions with symbolic derivative bounds propagation. In *8th Annual Conference on Robot Learning*, 2024.
- Chao Huang, Jiameng Fan, Xin Chen, Wenchao Li, and Qi Zhu. Polar: A polynomial arithmetic framework for verifying neural-network controlled systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 414–430, 2022.

- Yujia Huang, Ivan Dario Jimenez Rodriguez, Huan Zhang, Yuanyuan Shi, and Yisong Yue. Fi-ode: Certified and robust forward invariance in neural odes. *arXiv*, 2023.
- Radoslav Ivanov, Taylor Carpenter, James Weimer, Rajeev Alur, George Pappas, and Insup Lee. Verisig 2.0: Verification of neural network controllers using taylor model preconditioning. In *International Conference on Computer Aided Verification*, pages 249–262, 2021.
- Saber Jafarpour, Akash Harapanahalli, and Samuel Coogan. Interval reachability of nonlinear dynamical systems with neural network controllers. In *Learning for Dynamics and Control Conference*, pages 12–25, 2023.
- Saber Jafarpour, Akash Harapanahalli, and Samuel Coogan. Efficient interaction-aware interval analysis of neural network feedback loops. *IEEE Transactions on Automatic Control*, 2024.
- Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.
- Craig Knuth, Glen Chou, Necmiye Ozay, and Dmitry Berenson. Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants. *IEEE Robotics and Automation Letters*, 6(3):5129–5136, 2021.
- Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open-and closed-loop neural network verification using polynomial zonotopes. In *NASA Formal Methods Symposium*, pages 16–36, 2023.
- 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.
- Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Physics-informed neural network lyapunov functions: Pde characterization, learning, and verification. *Automatica*, 175:112193, 2025.
- Simin Liu, Changliu Liu, and John Dolan. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, 2023.
- Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.
- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.
- Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061, 2013.
- Yuhao Mao, Mark Niklas Müller, Marc Fischer, and Martin T. Vechev. Connecting certified and adversarial training. In *Advances in Neural Information Processing Systems*, 2023.

- Youngjae Min, Spencer M Richards, and Navid Azizan. Data-driven control with inherent lyapunov stability. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 6032–6037. IEEE, 2023.
- Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, volume 80, pages 3575–3583, 2018.
- Matthew Mirman, Gagandeep Singh, and Martin Vechev. A provable defense for deep residual networks. *arXiv preprint arXiv:1903.12519*, 2019.
- Mark Niklas Müller, Franziska Eckert, Marc Fischer, and Martin T. Vechev. Certified training: Small boxes are all you need. In *International Conference on Learning Representations*, 2023.
- Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. 2000.
- Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems*, pages 9832–9842, 2019.
- Christian Schilling, Marcelo Forets, and Sebastián Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 8169–8177, 2022.
- Zhouxing Shi, Yihan Wang, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Fast certified robust training with short warmup. In *Advances in Neural Information Processing Systems*, pages 18335–18349, 2021.
- Zhouxing Shi, Qirui Jin, Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. Neural network verification with branch-and-bound for general nonlinearities. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 15696, pages 315–335, 2025. doi: 10.1007/978-3-031-90643-5_17.
- Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):41, 2019.
- Wei Sun and Tianfu Wu. Learning layout and style reconfigurable gans for controllable image synthesis. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5070–5087, 2021.
- 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*, 2010.
- Samuel Teuber, Stefan Mitsch, and André Platzer. Provably safe neural network controllers via differential dynamic logic. In *Advances in Neural Information Processing Systems*, 2024.

- Hoang-Dung Tran, Xiaodong Yang, Diego Manzananas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T Johnson. Nnv: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, pages 3–17, 2020.
- 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.
- 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 neural network robustness verification. In *Advances in Neural Information Processing Systems*, pages 29909–29921, 2021.
- Xinyu Wang, Luzia Knoedler, Frederik Baymler Mathiesen, and Javier Alonso-Mora. Simultaneous synthesis and verification of neural control barrier functions through branch-and-bound verification-in-the-loop training. In *2024 European Control Conference (ECC)*, pages 571–578, 2024.
- Yihan Wang, Zhouxing Shi, Quanquan Gu, and Cho-Jui Hsieh. On the convergence of certified robust training with interval bound propagation. In *International Conference on Learning Representations*, 2022.
- Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*, volume 202, pages 36593–36604, 2023a.
- Yixuan Wang, Weichao Zhou, Jiameng Fan, Zhilu Wang, Jiajun Li, Xin Chen, Chao Huang, Wen-chao Li, and Qi Zhu. Polar-express: Efficient and precise formal reachability analysis of neural-network controlled systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023b.
- Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, volume 80, pages 5283–5292, 2018.
- Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural lyapunov control for discrete-time systems. In *Advances in Neural Information Processing Systems*, 2023.
- 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. In *Advances in Neural Information Processing Systems*, 2020.
- 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.
- 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*, 2023.

- 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. In *International Conference on Machine Learning*, 2024.
- Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, pages 4944–4953, 2018.
- Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.
- 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. In *Advances in Neural Information Processing Systems*, 2022.
- Hengjun Zhao, Xia Zeng, Taolue Chen, Zhiming Liu, and Jim Woodcock. Learning safe neural network controllers with barrier certificates. *Formal Aspects of Computing*, 33:437–455, 2021.
- Haohan Zou, Jie Feng, Hao Zhao, and Yuanyuan Shi. Analytical lyapunov function discovery: An rl-based generative approach. In *International Conference on Machine Learning*, pages 80776–80804. PMLR, 2025.

Appendix A. Details of the Implementation and Experiments

A.1. Modeling

Since our focus is on a new training framework, we follow model architectures in Yang et al. (2024), and we follow their source code which has some minor difference with details mentioned in their paper.

Controller. We use a fully-connected NN for the controller $u(\mathbf{x})$. There are 8 hidden neurons in each hidden layer. For Inverted Pendulum and Path tracking, there are 4 layers, and other systems, there are 2 layers. ReLU is used as the activation function.

Lyapunov function. For the Lyapunov function $V(\mathbf{x})$, we either use a model based on a fully-connected NN $\phi(\mathbf{x})$, as $V(\mathbf{x}) = |\phi(\mathbf{x}) - \phi(\mathbf{x}^*)| + \|(\epsilon_V \mathbf{I} + \mathbf{R}^\top \mathbf{R})(\mathbf{x} - \mathbf{x}^*)\|_1$, or a quadratic function as $V(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^\top (\epsilon_V \mathbf{I} + \mathbf{R}^\top \mathbf{R})(\mathbf{x} - \mathbf{x}^*)$, where $\mathbf{R} \in \mathbb{R}^{n_r \times n_r}$ is an optimizable matrix parameter, and $\epsilon_V > 0$ is a small value to guarantee positive definiteness. This construction automatically guarantees $V(\mathbf{x}^*) = 0$ and $V(\mathbf{x}) > 0$ ($\forall \mathbf{x} \neq \mathbf{x}^*$) (Yang et al., 2024). A NN-based Lyapunov function is used for Inverted Pendulum and Path tracking, where the NN is a fully-connected NN with 4 layers, and the number of hidden neurons is 16, 16, and 8 for the three hidden layers, respectively. Leaky ReLU is used as the activation function for NN-based Lyapunov functions. A quadratic Lyapunov function with $n_r = d$ is used for other systems. Our framework is general w.r.t. the formulation of the Lyapunov function. Regardless of the formulation of $V(\mathbf{x})$, it can be implemented as a module in PyTorch with the corresponding parameters, and jointly trained with the controller by optimizing Eq. (5), where the Lyapunov function is simply part of the PyTorch computational graph specified by Eq. (5).

Output feedback system and observer. There are two different settings considered – state feedback control and output feedback control. The state feedback setting is straightforward, and the true state is directly observed as \mathbf{x}_t . In the output feedback setting, the controller is accompanied by an NN-based observer taking the observable output of the system \mathbf{y}_t and predicts an estimated state $\hat{\mathbf{x}}_t$, and the controller then only takes the estimated state $\hat{\mathbf{x}}_t$ and system output y_t rather than the true state $\tilde{\mathbf{x}}_t$; and the input state $\mathbf{x}_t = [\tilde{\mathbf{x}}_t, \mathbf{e}_t]$ of the problem models the true state $\tilde{\mathbf{x}}_t$ and the state estimation error $\mathbf{e}_t = \hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_t$. Thus, output feedback settings contain more input dimensions and an additional observer NN than their state feedback counterparts. The observer is a fully-connected NN with 3 layers for Inverted Pendulum and 2 layers for 2D Quadrotor, with 8 neurons in each hidden layer. System dynamics for output feedback systems are included in Appendix C.2, and we refer readers to Section 2 and Section 4.3 in Yang et al. (2024) for further details.

A.2. Training

We use a batch size of 30000 for all the training, with a learning rate of 5×10^{-3} . We train models for up to 15,000 steps, but training for relatively smaller systems converges much faster. In the loss function, we set λ to 10^{-4} , λ_p to 0.1, and ϵ to 10^{-4} . The only hyperparameter we tune for individual systems is ρ_{ratio} : we try to make it as large as possible for individual systems to maximize ROA, as long as the training works, which is principled. For Inverted Pendulum and Path tracking, the range of ρ_{ratio} is between 0.6 and 0.9; for other systems, we set $\rho_{\text{ratio}} = 0.1$. We start our dynamic splits after 100 initial training steps. We use auto_LiRPA (Xu et al., 2020) to compute certified bounds, and we configure it to mainly use the backward bound propagation algorithm (a.k.a., CROWN (Zhang

et al., 2018) generalized to general computational graphs), while we use Interval Bound Propagation (IBP) (Gowal et al., 2018; Mirman et al., 2018) for bounding hidden layers of NNs required for the linear relaxation in CROWN, following Zhang et al. (2020); Xu et al. (2020). For PGD, we use 10 steps and a step size of 0.25 relative to the size of the subregion.

Appendix B. Additional Results

B.1. Detailed Comparison of Runtime

Table 4: Comparison of training time, verification time, and total time between CT-BaB (ours) and CEGIS (Yang et al., 2024). Training times for Yang et al. (2024) are obtained by running their released code and are only available for the listed settings.

System	CEGIS (Yang et al., 2024)			CT-BaB (Ours)		
	Training	Verification	Total	Training	Verification	Total
Inverted Pendulum	8.4min	33s	8.9min	1.7min	1.8s	1.7min
Inverted Pendulum (output)	7.3min	94s	8.8min	11.8min	5.3s	11.9min
Path Tracking	12.4min	39s	13.1min	3.3min	3.0s	3.4min
2D Quadrotor	8.1min	1.1hrs	1.24hrs	1.5hrs	49s	1.51hrs
2D Quadrotor (output)	7.5min	8.9hrs	9.02hrs	2.0hrs	0.8hrs	2.8hrs

In Table 4, we provide a detailed runtime comparison with Yang et al. (2024). Although our verification time is consistently lower than Yang et al. (2024), they sometimes achieve lower training time and total time. However, on the largest system of 2D Quadrotor with output feedback, our total time is also much less than Yang et al. (2024) due to a large improvement on verification time. This demonstrates the potential of our work for further scaling up to larger systems.

B.2. Visualization of ROA

In Figure 2, we visualize the ROA on the 2D Quadrotor with output feedback, demonstrating much larger ROA compared to Yang et al. (2024)

Appendix C. System Dynamics

We adopt system dynamics from existing works (Wu et al., 2023; Yang et al., 2024), and we attach them here for completeness. The systems are originally described in continuous time, and since we study discrete-time systems, following Wu et al. (2023); Yang et al. (2024), we set a time interval Δt with $\Delta t = 0.05$ for most systems except for $\Delta t = 0.01$ for 2D Quadrotor and output feedback settings.

C.1. State-Feedback Systems

Inverted Pendulum. The state has 2 dimensions as $\mathbf{x}_t = [\theta_t, \dot{\theta}_t]$, the control input has only 1 dimension as u_t , and the dynamics are:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \left[\dot{\theta}_t, \frac{-b\dot{\theta}_t + u_t}{ml^2} + \frac{g \sin \theta_t}{l} \right] \Delta t,$$

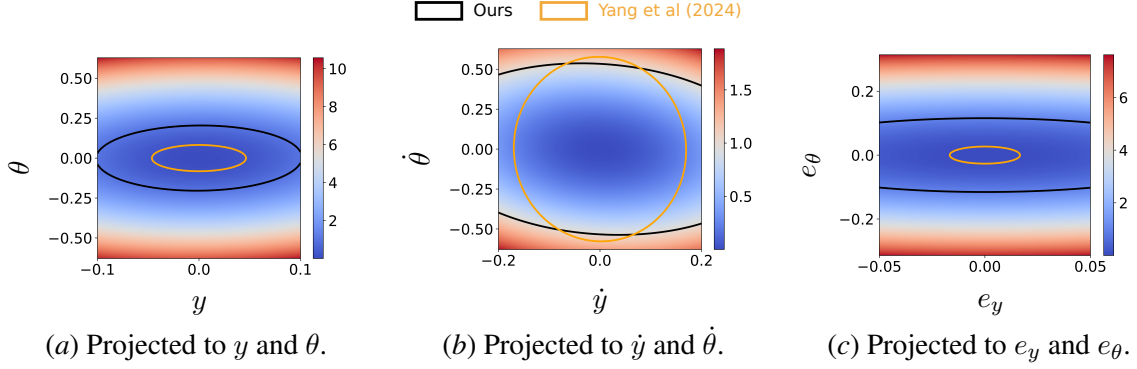


Figure 2: Visualization of the Lyapunov function (color plots) and ROA (contours) compared to Yang et al. (2024), on the 2D Quadrotor with output feedback, with three different 2D views. The system contains 8 states denoted as $\mathbf{x} = [y, \theta, \dot{y}, \dot{\theta}, e_y, e_\theta, e_{\dot{y}}, e_{\dot{\theta}}]$ (detailed in Appendix C.2). Our method demonstrates a $164\times$ larger ROA (in terms of the volume of ROA on the 8-dimensional input space) compared to Yang et al. (2024).

where mass is $m = 0.15$, length is $l = 0.5$, gravity is $g = 9.81$, and friction is $b = 0.1$.

Path tracking. The state has 2 dimensions as $\mathbf{x}_t = [e_t, \theta_{et}]$, the control input has only 1 dimension as u_t , and the dynamics are:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \left[\sin(\theta_{et}), \frac{u_t}{l} - \frac{\cos(\theta_{et})}{r - s \sin(\theta_{et})} \right] \cdot s \cdot \Delta t,$$

where speed is $s = 2.0$, radius is $r = 10$, and length is $l = 1.0$.

Cart-Pole. The state has 4 dimensions as $\mathbf{x}_t = [x_t, \dot{x}_t, \theta_t, \dot{\theta}_t]$, the control input has only 1 dimension as u_t , and the dynamics are:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{bmatrix} \dot{x}_t \\ \frac{u_t + m_p \sin \theta_t \cdot (l \cdot \dot{\theta}_t^2 - g \cos \theta_t)}{m_c + m_p \sin^2 \theta_t} \\ \dot{\theta}_t \\ \frac{-u_t \cos \theta_t - m_p l \dot{\theta}_t^2 \cdot \cos \theta_t \sin \theta_t + (m_p + m_c) g \sin \theta_t}{l(m_c + m_p \sin^2 \theta_t)} \end{bmatrix} \Delta t,$$

where length is $l = 1.0$, gravity is $g = 9.8$, mass for the cart is $m_c = 1.0$, and mass for the pole is $m_p = 0.1$.

PVTOL. The state has 6 dimensions as $\mathbf{x}_t = [x_t, y_t, \theta_t, \dot{x}_t, \dot{y}_t, \dot{\theta}_t]$, the control input has 2 dimensions as \mathbf{u}_t , and the dynamics are:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \left(\begin{bmatrix} \dot{x}_t \cos \theta_t - \dot{y}_t \sin \theta_t \\ \dot{x}_t \sin \theta_t + \dot{y}_t \cos \theta_t \\ \dot{\theta}_t \\ \dot{y}_t \dot{\theta}_t - g \sin \theta_t \\ -\dot{x}_t \dot{\theta}_t - g \cos \theta_t \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ (1/m) & (1/m) \\ l/J & (-l/J) \end{bmatrix} \mathbf{u}_t \right) \Delta t,$$

where $m = 4.0$, $l = 0.25$, $J = 0.0475$, and $g = 9.8$.

2D Quadrotor. The state has 6 dimensions as $\mathbf{x}_t = [x_t, y_t, \theta_t, \dot{x}_t, \dot{y}_t, \dot{\theta}_t]$, the control input has 2 dimensions as $\mathbf{u}_t = [u_{t1}, u_{t2}]$, and the dynamics are:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \left[\dot{x}_t \Delta t + \frac{\ddot{x}_t (\Delta t)^2}{2}, \dot{y}_t \Delta t + \frac{\ddot{y}_t (\Delta t)^2}{2}, \dot{\theta}_t \Delta t + \frac{\ddot{\theta}_t (\Delta t)^2}{2}, \ddot{x}_t \Delta t, \ddot{y}_t \Delta t, \ddot{\theta}_t \Delta t \right],$$

$$\ddot{x}_t = \frac{-\sin \theta_t (u_{t1} + u_{t2})}{m},$$

$$\ddot{y}_t = \frac{\cos \theta_t (u_{t1} + u_{t2}) - g}{m},$$

$$\ddot{\theta}_t = \frac{u_{t1} - u_{t2}}{J},$$

where $m = 0.486$, $l = 0.25$, $J = 0.00383$, and $g = 9.81$.

C.2. Output-Feedback Systems

For output-feedback systems, we follow the settings in Yang et al. (2024) as follows. The state \mathbf{x}_t considered in the verification problem is not simply the true state which we denote by $\tilde{\mathbf{x}}_t$. Suppose we have the same dynamics with a system output $\mathbf{y}_t \in \mathbb{R}^{n_y}$ given by an output map $h(\tilde{\mathbf{x}})$ as

$$\begin{aligned} \tilde{\mathbf{x}}_{t+1} &= f(\tilde{\mathbf{x}}_t, \mathbf{u}_t), \\ \mathbf{y}_t &= h(\tilde{\mathbf{x}}_t). \end{aligned}$$

We here consider the setting where the controller does not have access to the true state $\tilde{\mathbf{x}}_t$, but rather only have the information about the output \mathbf{y}_t , which could be a subset of $\tilde{\mathbf{x}}_t$ or in general any nonlinear function of $\tilde{\mathbf{x}}_t$. The state $\hat{\mathbf{x}}_t$ is estimated with a dynamic state-observer using a NN $\phi_{\text{obs}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_x}$ as

$$\hat{\mathbf{x}}_{t+1} = f(\hat{\mathbf{x}}_t, \mathbf{u}_t) + \phi_{\text{obs}}(\hat{\mathbf{x}}_t, \mathbf{y}_t - h(\hat{\mathbf{x}}_t)) - \phi_{\text{obs}}(\hat{\mathbf{x}}_t, \mathbf{0}_{n_y}).$$

The control actions are obtained by $\mathbf{u}_t = \pi(\hat{\mathbf{x}}_t, \mathbf{y}_t)$ where $\pi(\cdot, \cdot)$ is a NN-based controller. Now by considering the augmented system with augmented state $\mathbf{x}_t = [\hat{\mathbf{x}}_t, \mathbf{e}_t]^\top$ where $\mathbf{e}_t = \hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_t$, the output-feedback system is the following closed-loop dynamics $\mathbf{x}_{t+1} = f_{\text{cl}}(\mathbf{x}_t)$ with f_{cl} being defined as:

$$f_{\text{cl}}(\mathbf{x}_t) = \begin{bmatrix} f(\tilde{\mathbf{x}}_t, \pi(\hat{\mathbf{x}}_t, h(\tilde{\mathbf{x}}_t))) \\ f(\hat{\mathbf{x}}_t, \pi(\hat{\mathbf{x}}_t, h(\tilde{\mathbf{x}}_t))) + g(\tilde{\mathbf{x}}_t, \hat{\mathbf{x}}_t) - \tilde{\mathbf{x}}_t \end{bmatrix},$$

where $g(\tilde{\mathbf{x}}_t, \hat{\mathbf{x}}_t) = \phi_{\text{obs}}(\hat{\mathbf{x}}_t, h(\tilde{\mathbf{x}}_t) - h(\hat{\mathbf{x}}_t)) - \phi_{\text{obs}}(\hat{\mathbf{x}}_t, \mathbf{0}_{n_y})$.

We follow Yang et al. (2024) to consider the following two output-feedback dynamics.

Inverted Pendulum with output feedback. The system dynamics are the same as the state-feedback setting with output map $h(\mathbf{x}_t) = \theta_t$.

2D Quadrotor with output feedback. There is a 4-dimensional state $\tilde{\mathbf{x}} = [p, \theta, \dot{p}, \dot{\theta}]$ with two control inputs $\mathbf{u} = [u_1, u_2]$:

$$\tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_t + \begin{bmatrix} \dot{p}_t \\ \dot{\theta}_t \\ \frac{1}{m} \cos(\theta_t)(u_1 + u_2) - g \\ \frac{l}{I}(u_1 - u_2) \end{bmatrix} \Delta_t$$

where $m = 0.486, l = 0.25, I = 0.00383, g = 9.81, \Delta_t = 0.01$. This system obtains observations from a lidar sensor. Let $\alpha_{\max} = 0.149\pi$ and α_i be 4 angles evenly spaced in $[-\alpha_{\max}, \alpha_{\max}]$. Let $H = 5$ and $h_0 = 1$, then we define

$$h(x) = [h_1(x), \dots, h_4(x)]^\top \quad \text{s.t.} \quad h_i(x) = \text{clamp}\left(\frac{p + h_0}{\cos(\theta - \alpha_i)}, 0, H\right),$$

as the output map.