Parallel Differentiable Reachability for Learning and Planning with Certified Neural Dynamics and Controllers

Keyi Shen and Glen Chou
Georgia Institute of Technology, Atlanta, GA 30308
Email: {kshen84, chou}@gatech.edu

Project Site and Video: (Link) | **Code (DiffReach)**: (Github) | **Code (DiffReach-Robotics)**: (Github)

***Abstract*—Neural network (NN) dynamics models and control policies achieve strong performance in robotics, but providing sound guarantees under uncertainty remains difficult, especially for closed-loop NN systems. Existing reachability tools provide formal over-approximations, yet are often non-differentiable, overly conservative, or too slow for modern learning and online planning pipelines. To address this, we present a parallelizable, differentiable reachability framework in JAX for continuous- and discrete-time systems with analytical and NN-based dynamics and controllers. Our framework combines Taylor-model flowpipe construction with CROWN-style linear bound propagation through a unified representation that preserves affine dependencies while supporting GPU-batched computation and automatic differentiation. Building on this reachability primitive, we develop (i) a certified training method that encourages reachability-friendly dynamics models and controllers, and (ii) a reachability-aware sampling-based MPC scheme with gradient-based refinement. Experiments on non-prehensile manipulation and quadrotor tasks, including hardware and higher-dimensional evaluations (up to 72D), demonstrate practical online planning while maintaining certified reachable-set over-approximations under bounded uncertainty.**

I. Introduction

Neural network (NN) dynamics models and controllers are increasingly used in robotics [33, 41, 63, 48], enabling operation in complex environments where analytical modeling is infeasible. Despite strong empirical performance, providing robustness and safety guarantees for NN-in-the-loop systems remains challenging, as NNs introduce high-dimensional nonlinear components that complicate worst-case analysis under disturbances and uncertainty [42]. Classical reachability analysis provides formal guarantees through reachable-set over-approximations [1, 11], but often scales poorly and struggles with NN components, leading to conservative bounds. Modern neural network verification (NNV) methods, such as CROWN [69, 64] and abstract interpretation [50], scale to large networks and provide efficient bound propagation, but are primarily designed for static verification and do not naturally handle continuous-time dynamics. Recent efforts combining reachability analysis and NNV [42] support NN-controlled systems, but remain largely offline, non-differentiable, and difficult to integrate into modern robot learning and online planning. As a result, reachability analysis is typically restricted to a *post-hoc* certification mechanism, and these limitations prevent verification from *directly informing* the training and deployment of certifiable models in robot learning. This can lead to costly cycles of non-certified training, failed certification, and retraining.

To address these challenges, we develop an efficient **parallel and differentiable reachability framework** that integrates certified analysis into robot learning and planning. Examples of reachability-aware planning rollouts are shown in Figure 1. Our key idea is to unify Taylor-model-based reachability

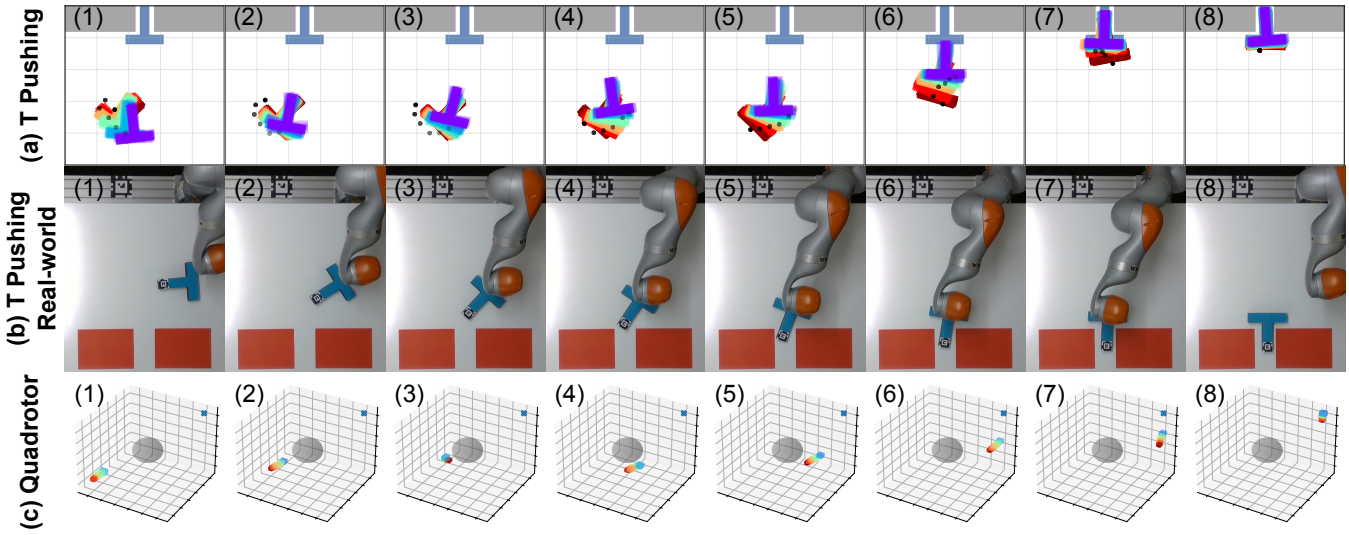

Fig. 1. **Reachability-guided planning**. We visualize three successful model predictive control (MPC) rollouts of our reachability-guided planner using certified, reachability-regularized models on **(a)** T-pushing, **(b)** T-pushing (Real-world) and **(c)** quadrotor. At each replanning step (indices shown), we plot the current state, the planned trajectory, and the certified reachable set under bounded disturbance (shaded footprints for T-pushing and 3D boxes for quadrotor). The results show that our planner achieves the task while maintaining tight, step-wise formal reachability guarantees under uncertainty. Overall, our parallel differentiable reachability tool can be efficiently integrated into model-training and planning pipelines in robotics, enabling verified-by-construction learning and planning with learned models.

for continuous-time dynamics with CROWN-style NN verification within a single JAX [5] computation graph. Our reachability tool, *DiffReach*, combines Taylor-model flowpipe construction for analytical dynamics with CROWN-style linear bound propagation for neural components through a *unified representation* that preserves affine dependencies *without intermediate interval relaxation*. To enable efficient GPU execution and automatic differentiation, reachability propagation is reformulated using *fixed-shape tensor operations* compatible with just-in-time (JIT) compilation and batched computation. The resulting framework supports continuous- and discrete-time systems with analytical and NN-based dynamics and controllers while exposing differentiable reachable-set objectives for downstream optimization. Building on this reachability primitive, we develop *DiffReach-Robotics*, which consists of 1) a **certified training** method for learning neural dynamics and controllers and 2) a **reachability-aware sampling-based model predictive control (MPC)** scheme with gradient-based refinement. Experiments on manipulation and quadrotor tasks, including hardware and higher-dimensional evaluations, demonstrate practical online planning while maintaining certified reachable-set over-approximations under bounded uncertainty. Our contributions are:

- A *parallel and differentiable reachability framework* combining Taylor-model flowpipes and CROWN-style bound propagation within a unified JAX-based representation for continuous- and discrete-time systems with analytical and NN-based dynamics and controllers.
- A *certified training method* that leverages differentiable reachable-set objectives to improve robustness of neural components under uncertainty.
- A *reachability-aware sampling-based MPC scheme* that integrates certified reachable sets into online planning with gradient-based refinement.
- Experimental evaluation on manipulation and quadrotor tasks, including hardware and higher-dimensional systems (up to 72D), demonstrating practical integration of certified reachability into robot learning and planning.

II. Related Work

Robust planning and control in robotics. Robotic tasks such as manipulation and agile flight involve complex nonlinear dynamics that make robust control under uncertainty challenging. A common approach separates *high-level planning* from *low-level feedback control* [51, 23, 67]. Existing methods compute invariant tubes or safety certificates around nominal trajectories using sums-of-squares [51, 40], Hamilton-Jacobi reachability [23], or contraction-based analysis [13, 29, 14, 52, 28]. While these approaches provide strong guarantees for nonlinear systems, they are typically designed for fixed analytical controllers and offline analysis. Extensions to NN controllers [28, 13, 54] similarly focus on post-hoc verification rather than integration into controller learning.

NN dynamics and controllers. NN dynamics models learned from simulation or real-world data are widely used in robotic manipulation and control [48, 59], including models based on visual observations [17, 15], latent representations [21, 63], and structured state representations [41, 47]. Despite strong empirical performance, most planning methods using NN dynamics lack formal safety guarantees, while certified approaches typically rely on slow offline verification [28, 13] decoupled from model training and planning. Learned NN controllers [30, 12] and heuristic safety filters [35, 68, 53] similarly lack optimization-aware certified objectives. In contrast, our framework directly incorporates differentiable reachable-set objectives into learning and planning.

Reachability and NNV. NNV provides formal guarantees on network outputs. Early methods [55, 7, 38] struggled to scale due to limited parallelization and GPU support [44, 72, 34]. Bound-propagation methods [16, 50, 57, 19], including CROWN [69], improve scalability by propagating certified affine bounds through network layers. Practical tools such as α,β-CROWN [65, 58, 71] achieve strong performance on large networks [6, 26]. NNV has also been applied to robotic control, including Lyapunov and Zubov neural control [66, 45, 32], neural contraction metrics [31], and branch-and-bound planning with neural dynamics [46], but these methods primarily certify stability certificates or planning objectives rather than differentiable reachable-set propagation for closed-loop NN systems.

Classical reachability tools address dynamical systems directly. CORA [1], JuliaReach [4], and NNV [56, 36] support reachability analysis for nonlinear and NN-controlled systems[37], while Flow* [10, 11, 9, 8] develops rigorous Taylor-model flowpipe construction for continuous-time systems [2]. Flow*-based NNCS tools such as POLAR [24, 60] and CROWN-Reach [73] further combine Taylor-model reachability with NN bound propagation for certified NN-in-the-loop analysis. Recent JAX-based tools, including immrax [22] and GoTube [20], improve compatibility with GPU acceleration and automatic differentiation, though immrax relies on conservative interval propagation and GoTube provides statistical rather than formal guarantees. While these methods provide strong foundations for certified NN system analysis, they are primarily designed for offline verification rather than differentiable reachability integrated into learning and online planning.

Certified training and verification-aware learning have also been explored in NN verification [70, 25] and neural control policies [49, 62]. In contrast, our framework unifies Taylor-model reachability and scalable NN bound propagation within a differentiable JAX-based framework, enabling reachable-set objectives to directly inform robot learning and online planning.

III. Problem Statement

We consider robotic systems that combine high-level planning with low-level feedback control under bounded uncertainty, where both dynamics models and controllers may contain NN components. Our goal is to compute sound reachable-set over-approximations over finite horizons for both discrete-time planning dynamics and continuous-time closed-loop execution, while exposing differentiable reachable-set objectives that can be integrated into downstream learning and online planning.

A. Planning under bounded uncertainty

We follow a modular planning-control framework, where a high-level planner optimizes a sequence of actions using a predictive discrete-time dynamics model. Let the planning state and action be $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^k$, respectively, and consider the predictive dynamics $\hat{x}_{t+1} = f_{\text{dt,dyn}}(\hat{x}_t, u_t)$, where $f_{\text{dt,dyn}}$ may be an analytical model or a neural network (NN). Starting from the current state $\hat{x}_{t_0} = x_{t_0}$, the planner rolls out $f_{\text{dt,dyn}}$ over a horizon H to obtain predicted future states $\{\hat{x}_t\}_{t=t_0}^{t_0+H}$.

In practice, the true state may deviate from the nominal estimate due to sensing errors, modeling errors, or external disturbances. We model the initial uncertainty as the bounded set $x_{t_0} \in \mathcal{B}_\epsilon(\hat{x}_{t_0}) = \{x \mid \hat{x}_{t_0} - \epsilon \le x \le \hat{x}_{t_0} + \epsilon\}$ where $\epsilon \in \mathbb{R}$ is the perturbation radius. Given a fixed action sequence $\boldsymbol{u} := \{u_t\}_{t=t_0}^{t_0+H}$, we denote by $\mathcal{R}_t(\boldsymbol{u}, \mathcal{B}_\epsilon(\hat{x}_{t_0}))$ the reachable set at time t, i.e., the set of all states reachable from initial states in $\mathcal{B}_\epsilon(\hat{x}_{t_0})$ under the dynamics $f_{\text{dt,dyn}}$ and action sequence $\boldsymbol{u}$. In our method, $\mathcal{R}_t$ is represented by a sound over-approximation that accounts for bounded uncertainty propagation through the dynamics. We consider the following uncertainty-aware finite-horizon trajectory optimization problem:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \textstyle\sum_{t=t_0}^{t_0+H} c(\hat{x}_t, u_t) \\ \text{s.t.} \quad & \hat{x}_{t+1} = f_{\text{dt,dyn}}(\hat{x}_t, u_t), \\ & G(\mathcal{R}_t(\boldsymbol{u}, \mathcal{B}_\epsilon(\hat{x}_{t_0}))) \ge 0, \quad t = t_0, \dots, t_0 + H, \end{aligned} \tag{1}$$

where c is a task-dependent stage cost, H is the planning horizon, and actions are box-constrained to $u_t \in \mathcal{U} := \{u \mid \underline{u} \le u \le \overline{u}\} \subset \mathbb{R}^k$. The generalized constraint functional $G(\cdot)$ enforces robustness and safety over reachable sets, e.g., collision avoidance for all states in $\mathcal{R}_t$ or penalties on reachable-set size.

The central challenge is to compute tight and sound reachable-set over-approximations efficiently enough for repeated use inside online MPC and trajectory optimization.

B. Low-level control under bounded uncertainty

We now consider low-level feedback control for continuous-time execution. Let $u_{\text{ctl}} = f_{\text{ct,ctl}}(\hat{x}, y_{\text{ref}}) \in \mathbb{R}^l$ denote a neural controller that takes as input the current state estimate $\hat{x}$ and a reference signal y_{ref} (e.g., desired states or actions from the high-level planner), and outputs a fine-grained control input applied to the continuous-time dynamics

$$\dot{x}(\tau) = f_{\text{ct,dyn}}(x(\tau), u_{\text{ctl}}). \tag{2}$$

We assume the dynamics $f_{\text{ct,dyn}}$ are available for reachability analysis, either as an analytical ODE (e.g., quadrotor dynamics) or as a learned neural ODE when analytical models are unavailable for complex manipulation systems.

The controller operates at a fixed control frequency $q_{\text{ctl}} = 1/\delta$, typically higher than the planning frequency. Over each control interval $\tau \in [i\delta, (i+1)\delta]$, we apply a zero-order hold policy: the control input u_{ctl} and reference y_{ref} are treated as constant during the interval, yielding a closed-loop trajectory determined by the ODE solution with initial condition $x(i\delta)$.

As in the planning setting, the true state may deviate from its nominal estimate due to bounded uncertainty, i.e., $x(0) \in \mathcal{B}_\epsilon(\hat{x}(0))$. Given a reference sequence $\boldsymbol{y}_{\text{ref}}$, we seek a sound over-approximation of the closed-loop reachable set

$$\mathcal{R}(\tau\,; \boldsymbol{y}_{\text{ref}}, \mathcal{B}_\epsilon(\hat{x}(0))), \tag{3}$$

i.e., the set of all states reachable under the initial uncertainty and closed-loop execution of $f_{\text{ct,ctl}}$ conditioned on $\boldsymbol{y}_{\text{ref}}$.

Beyond post-hoc verification, we use reachable-set information directly during learning. In particular, differentiable summaries of $\mathcal{R}(\tau\,; \cdot)$ are incorporated as regularizers or constraints when training controllers and, when applicable, continuous-time dynamics models. This requires the reachability computation to support repeated invocation during optimization while exposing differentiable objectives that can guide parameter updates.

In summary, we target three problems:

Problem 1. ***(Reachability-aware planning)*** *Given a predictive dynamics model $f_{dt,dyn}$, stage cost c, robustness/safety functional G, nominal initial state $\hat{x}_0$, disturbance radius ϵ, and planning horizon H, compute an action sequence $\boldsymbol{u}$ that achieves the task objective while maintaining sound reachable-set over-approximations $\{\mathcal{R}_t\}_{t=0}^{H}$ under the initial uncertainty $x_0 \in \mathcal{B}_\epsilon(\hat{x}_0)$. The reachable-set computation should be efficient to support repeated online replanning within MPC.*

Problem 2. ***(Certified learning of dynamics and control)*** *Given a dataset $\mathcal{D}$ of N state–action trajectories and a disturbance radius ϵ, learn model components (e.g., $f_{dt,dyn}$, $f_{ct,ctl}$, and $f_{ct,dyn}$ when applicable) that achieve strong nominal performance while remaining robust to bounded uncertainty, as quantified by reachable-set-based objectives and constraints.*

Problem 3. ***(Differentiable reachability engine)*** *Design a reachability engine that supports the planning and learning problems in Problems 1 and 2. The engine should handle both continuous- and discrete-time systems with NN-based dynamics and controllers, while supporting exogenous reference inputs such as planned trajectories and tracking references. It should provide sound reachable-set over-approximations efficiently enough for repeated use in online planning and iterative learning, while exposing differentiable reachable-set objectives for downstream optimization.*

IV. DiffReach: A GPU-Parallelized and Differentiable Reachability Engine

In this section, we present *DiffReach*: a parallel, differentiable reachability pipeline for NN-in-the-loop systems (Figure 2(a)). Section IV-A introduces a fixed-shape Taylor model (TM) representation that enables efficient batched computation. Section IV-B develops TM flowpipe propagation for continuous-time (CT) analytical dynamics, while Section IV-C incorporates neural components via CROWN through a unified TM–CROWN interface. Building on this interface, Section IV-D presents closed-loop reachability for NN feedback systems by alternating controller certification and TM propagation. Finally, Section IV-E extends the framework to discrete-time (DT) systems via stepwise propagation. Overall, this unifies analytical and NN components under a shared affine representation and enables GPU-parallel reachability for learning and planning.

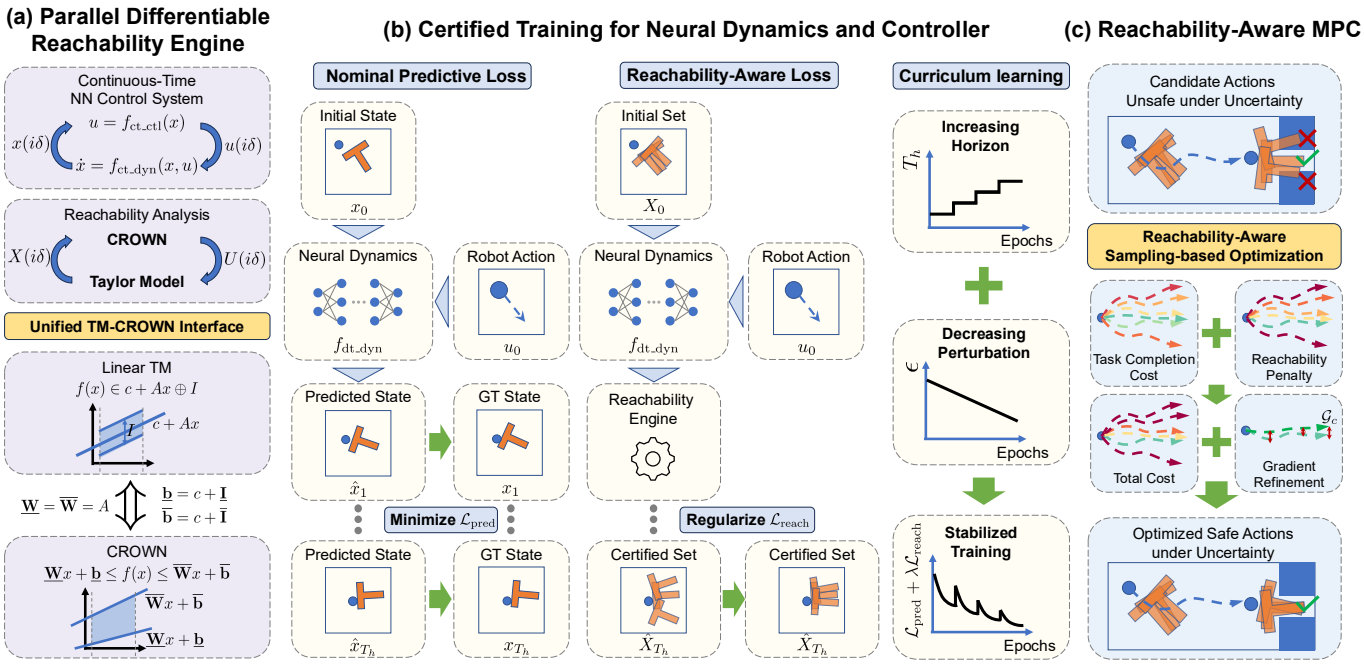

Fig. 2. **Overview of the proposed framework. (a)** Our parallel differentiable reachability engine unifies Taylor-model (TM) flowpipe propagation for analytical dynamics with CROWN-based neural bound propagation through a shared TM–CROWN interface, enabling GPU-batched and differentiable reachability analysis for continuous- and discrete-time systems. **(b)** The differentiable reachable sets are used to enable certified training of neural dynamics and controllers. Standard prediction or tracking losses are augmented with reachability-aware regularization that penalizes large certified reachable sets under bounded uncertainty, while curriculum learning stabilizes optimization over increasing horizons and perturbations. **(c)** The same reachability engine is integrated into sampling-based MPC, where candidate trajectories are evaluated using both task objectives and reachable-set penalties. Parallel reachability evaluation and gradient-based refinement enable uncertainty-aware online planning with certified reachable-set over-approximations.

A. Representation of Reachable Sets

We represent reachable sets using *Taylor models* (TMs), which provide a certified representation of nonlinear functions over bounded domains:

$$f(x) = p_k(x) \oplus I, \quad x \in X, \tag{4}$$

where p_k is a k-th order polynomial approximation and $I = [\underline{I}, \overline{I}]$ soundly bounds truncation and higher-order errors. This "polynomial + interval" form enables guaranteed set propagation, since bounding p_k over X and adding I yields a sound enclosure of $f(X)$. Taylor models are also closed under common operations such as addition, multiplication, and composition, making them suitable for nonlinear reachability analysis.

Classical Taylor models use variable-size polynomial representations whose complexity depends on the number of monomials, which grows combinatorially with state dimension and polynomial order. While effective in CPU-based tools, these representations are not well suited for GPU execution, which prefer static tensor shapes and dense matrix operations.

To address this, we adopt a *fixed-shape* TM representation by restricting the polynomial component to linear or quasi-quadratic forms. For an n-dim state, we represent each TM as

$$F(z) = c + Az \oplus I, \quad \text{or} \quad F(z) = c + Az + B\,\tau z \oplus I, \tag{5}$$

where $z \in \mathbb{R}^{n+1}$ augments the state with time, $c \in \mathbb{R}^n$, $A, B \in \mathbb{R}^{n\times(n+1)}$, and $I \subset \mathbb{R}^n$ is the interval remainder. This yields a fixed $O(n^2)$ representation with static tensor shapes.

The affine term Az preserves first-order correlations, while the quasi-quadratic term $B\,\tau z$ captures limited curvature without incurring the $O(n^3)$ cost of full quadratic parameterizations. As a result, TM operations such as composition and bounding reduce to batched matrix operations that are compatible with GPU acceleration and automatic differentiation.

B. Propagation under Continuous-Time Analytical Dynamics

Given the TM representation above, we now describe reachable-set propagation for continuous-time analytical dynamics. Consider the uncontrolled system $\dot{x}(\tau) = f_{\text{ct,dyn}}(x(\tau))$, with uncertain initial state $x(0) \in X_0$. The goal is to compute a sound enclosure of all states reachable over a finite horizon t.

We adopt *flowpipe construction*, which propagates the initial set through a sequence of Taylor models. Over a small interval of length h, we compute a TM enclosure of the solution using truncated Picard iteration, summarized in Theorem IV.1.

Theorem IV.1 (TM flowpipe step via Picard iteration [10])**.** *Consider the ODE $\dot{x}(\tau) = f_{ct,dyn}(x(\tau))$ with initial set $x(0) \in \mathcal{X}_i$, and fix a step size $h > 0$. The $(i+1)$-th TM flowpipe over $\tau \in [0, h]$ is given by*

$$X^{(i+1)}(x_0, \tau) = p_k(x_0, \tau) \oplus I, \quad x_0 \in \mathcal{X}_i,$$

where p_k is obtained via truncated k-order Picard iteration:

$$g_{j+1}(x_0, \tau) = \mathit{Trunc}_j\left(x_0 + \textstyle\int_0^\tau f_{ct,dyn}\big(g_j(x_0, s)\big)\, ds\right), \tag{6}$$

for $j = 0, \ldots, k-1$, with $g_0(x_0, \tau)$ initialized as the polynomial part of $X^{(i)}(x_0, h)$.

The interval remainder I is obtained via Picard-based validation and refinement. Starting from a conservative estimate I_0, we replay the Picard iteration with

$$g_0(x_0, \tau) = p_k(x_0, \tau) \oplus I_0.$$

Let $I_1 := g_1(x_0, \tau) - p_k(x_0, \tau)$ denote the induced remainder after one iteration. If the iteration is contractive, i.e., $I_1 \subseteq I_0$, we continue refinement to obtain the minimal contractive interval I subject to termination conditions. Otherwise, I_0 is enlarged until a contractive remainder is achieved.

Each step constructs a local polynomial approximation of the trajectory over $[0, h]$ while rigorously bounding truncation and higher-order errors through the interval remainder. Iterating this procedure over $N := \lceil \frac{t}{h} \rceil$ steps yields a certified reachable tube over horizon t. In practice, repeated propagation can accumulate over-approximation error over long horizons. Classical reachability tools mitigate this effect through affine normalization, reparameterization, and wrapping control; we defer these details to Appendix A and refer to [8, 9].

C. Neural Components via CROWN

The flowpipe construction above applies directly to analytical dynamics. However, when dynamics or controllers are represented by neural networks, directly propagating Taylor models (TMs) through the network is often either overly conservative for linear TMs or computationally prohibitive for higher-order TMs. Linear TM propagation corresponds to forward bound propagation through the network, where local relaxations are constructed independently at each nonlinear layer, leading to accumulated over-approximation. Higher-order TMs can improve tightness, but are difficult to implement efficiently with matrix operations and scale poorly to large networks.

In contrast, *CROWN* (Theorem IV.2) computes affine bounds via backward bound propagation that accounts for downstream layers, often yielding significantly tighter bounds at moderate additional cost. We therefore leverage CROWN (through its JAX implementation in `jax_verify` [18]) to certify neural components within the TM-based pipeline.

Theorem IV.2 (CROWN linear bounds of NN [69, 64])**.** *Given an NN f_{NN} with input dimension n_i and output dimension n_o, and a bounded input $x \in \mathcal{X}$ where $\mathcal{X}$ is a box set, there exist provable linear lower and upper bounds on $f_{NN}(x)$ such that*

$$\underline{\mathbf{W}} x + \underline{\mathbf{b}} \leq f_{NN}(x) \leq \overline{\mathbf{W}} x + \overline{\mathbf{b}}, \tag{7}$$

where $\underline{\mathbf{W}}, \overline{\mathbf{W}} \in \mathbb{R}^{n_o \times n_i}$ and $\underline{\mathbf{b}}, \overline{\mathbf{b}} \in \mathbb{R}^{n_o}$.

A key observation is that a *linear TM* $p(z) \oplus I = c + Az \oplus I$ is equivalent to a pair of affine bounds with shared slope:

$$\underline{\mathbf{W}} = \overline{\mathbf{W}} = A, \quad \underline{\mathbf{b}} = c + \underline{I}, \quad \overline{\mathbf{b}} = c + \overline{I}.$$

This allows analytical and neural components to share a common representation. To preserve this structure, we enforce a *shared-slope constraint* in CROWN so that the lower and upper bounds use the same linear term, allowing neural network outputs to be represented again as linear TMs.

In our setting, neural networks are evaluated on set-valued TM inputs. Since standard CROWN assumes box-set inputs, we apply input reparameterization and certify an equivalent network, as summarized in Theorem IV.3. We defer the proof to Appendix B.

Theorem IV.3 (CROWN bounds of NN with linear TM input)**.** *Given a neural network f_{NN} with input dimension n_i and output dimension n_o, and a bounded input x parameterized by a linear TM $g(z) = c_g + A_g z \oplus I_g$, where $z \in \mathcal{Z} \subset \mathbb{R}^{n_z}$, $A \in \mathbb{R}^{n_i \times n_z}$, and $I_g = [\underline{I}_g, \overline{I}_g] \subset \mathbb{R}^{n_i}$, we can construct a linear TM $p(z) \oplus I = c + Pz \oplus I$ that soundly over-approximates $f_{NN}(g(z))$, where $c \in \mathbb{R}^{n_o}$, $P \in \mathbb{R}^{n_o \times n_z}$ and $I = [\underline{I}, \overline{I}] \subset \mathbb{R}^{n_o}$. In particular, P and I can be obtained by applying CROWN (with the enforced constraint $\underline{\mathbf{W}} = \overline{\mathbf{W}}$) to the equivalent network $f_{NN}(c + Az + r)$ with inputs $z \in \mathcal{Z}$ and $r \in I_g$.*

Remark IV.4 (Quasi-quadratic TM inputs)**.** *For quasi-quadratic TMs, we bound the higher-order term as an interval and reduce the input to a linear TM before applying CROWN. Since the step size h is small, the resulting conservatism is typically modest in practice.*

CROWN therefore serves as a drop-in module for neural components within the TM-based reachability pipeline. Analytical dynamics are propagated with TM arithmetic (Section IV-B), while neural components are certified through CROWN under the same linear-TM interface. This unified representation avoids intermediate interval conversion, preserves cross-state correlations, and maintains end-to-end differentiability.

D. Closed-Loop Reachability for Continuous-Time Systems

We now consider reachability of continuous-time systems under neural feedback control. Our key idea is to compute reachable sets of the closed-loop system by *alternating* between (i) certifying the neural controller using CROWN and (ii) propagating the system dynamics using TM flowpipes under zero-order hold. This yields a unified pipeline in which both controller and dynamics share the same TM-based representation.

We consider a continuous-time control system $\dot{x}(\tau) = f_{\mathrm{ct,dyn}}(x(\tau), u_{\mathrm{ctl}})$, where the control input is given by a neural controller $u_{\mathrm{ctl}} = f_{\mathrm{ct,ctl}}(x(i\delta))$ and held constant over each control interval $\tau \in [i\delta, (i+1)\delta]$ (zero-order hold). The initial state satisfies $x(0) \in X_0$. For clarity, we omit additional controller inputs such as reference trajectories $(x_{\mathrm{ref}}, u_{\mathrm{ref}})$, which can be treated as known exogenous inputs to $f_{\mathrm{ct,ctl}}$.

To compute a certified reachable tube over a finite horizon, we decompose each control interval into two steps:

(A) Controller certification. At $\tau = i\delta$, we apply the TM–CROWN interface from Section IV-C to the TM enclosure of $x(i\delta)$ and obtain a certified linear-TM enclosure of the controller output $u_{\mathrm{ctl}} = f_{\mathrm{ct,ctl}}(x(i\delta))$.

(B) Dynamics propagation. Given the certified control enclosure, we propagate the dynamics over $\tau \in [i\delta, (i+1)\delta]$ using the TM flowpipe method from Section IV-B. The control input is treated as constant by augmenting the state with u_{ctl} and enforcing $\dot{u}_{\mathrm{ctl}} = 0$. Each control interval is discretized

Algorithm 1 Reachability of CT NN Control Systems.

Inputs: dynamics $f_{\text{ct,dyn}}(x,u)$ (analytical or NN), NN controller $f_{\text{ct,ctl}}$, initial box $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$, total control steps N_{ctl}, control interval $\delta = Kh$, atomic step interval h, TM-flowpipe params θ_{fp}
Outputs: certified reachable set $\{\mathcal{X}_j\}_{j=0}^{N} (N = N_{\text{ctl}}K)$

Precompute step-local boxes and initialize TM state
$X_{\text{pre}} \leftarrow [\texttt{BuildLinearTM}(\mathcal{X}_0),\ \mathbf{0}]$ ▷ affine TM for $\tilde{x}$
$\tilde{f} \leftarrow [f_{\text{ct,ctl}},\ \mathbf{0}]$ ▷ define augmented dynamics $\tilde{f}(x,u)$
# Outer loop over control updates (zero-order hold)
for $i = 0, \ldots, N_{\text{ctl}} - 1$ **do**
  # (A) Impose NN control at the boundary using CROWN–TM
  $X_{iK} \leftarrow X_{\text{pre}}(h)$ ▷ TM at the control boundary $i\delta$
  $X_{iK} \leftarrow \texttt{Ctl}_{\texttt{CROWN}}\big(X_{iK}, f_{\text{ct,ctl}}\big)$ ▷ get affine TM $u = f_{\text{ct,ctl}}(x)$
  # (B) Inner loop: propagate augmented TM under $\tilde{f}$ for δ
  $X_{\text{in}} = X_{iK}$
  for $j = 1, \ldots, K$ **do**
    $(X_{\text{in}},\ \mathcal{X}_{iK+j}) \leftarrow \texttt{ReachStep}\big(\tilde{f}, X_{\text{in}}; h, \theta_{\text{fp}}\big)$
  $X_{\text{pre}} \leftarrow X_{\text{in}}$ ▷ TM for next control step
return $\{\mathcal{X}_j\}_{j=0}^{N}$

into smaller steps of size $h \ll \delta$ to obtain a certified enclosure of the reachable tube and the next boundary state $x((i+1)\delta)$.

Iterating these two steps over all control intervals yields a sound over-approximation of the reachable set of the closed-loop system. Algorithm 1 summarizes the alternating procedure. The outer loop performs controller certification at frequency $1/\delta$, while the inner loop applies TM flowpipe propagation with atomic step size h. Reachable sets are represented as TM-valued states $X(\tau)$, where evaluating a TM at a time point yields a box enclosure. The operation $\texttt{Ctl}_{\texttt{CROWN}}()$ implements controller certification via the TM–CROWN interface, while $\texttt{ReachStep}()$ performs one TM flowpipe step under the augmented zero-order-hold dynamics.

Closely related templates combining TM-based flowpipe propagation with CROWN-style certification have appeared in prior work [24, 73]. However, these approaches typically connect the two modules through interval-only interfaces, e.g., converting TMs to intervals before applying CROWN. This discards affine dependencies captured by TM representations and can amplify conservatism over long horizons. In contrast, our method maintains a unified linear-TM representation throughout the pipeline, avoiding intermediate intervalization while preserving dependencies across states and time steps. This unified interface further enables the fully neural setting where both $f_{\text{ct,dyn}}$ and $f_{\text{ct,ctl}}$ are neural networks.

E. Discrete-Time Systems

We now consider discrete-time systems, where reachability reduces to stepwise propagation without flowpipe construction. Consider the system $x_{k+1} = f_{\text{dt,dyn}}(x_k, u_k)$, with uncertain initial state $x_0 \in X_0$ and planned or exogenous inputs $\{u_k\}_{k=0}^{H-1}$. Our goal is to compute a sequence of reachable sets $\{X_k\}$ over a finite horizon.

At each time step, we apply the TM–CROWN interface (Section IV-C) to certify the one-step map and obtain an affine TM enclosure of the next reachable set:

$$X_{k+1} = f_{\text{dt,dyn}}(X_k, u_k).$$

Algorithm 2 Reachability of DT Dynamics.

Inputs: DT model $f_{\text{dt,dyn}}$, initial box $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$, planned inputs $\{u_k\}_{k=0}^{H-1}$, horizon H
Outputs: reachable sets $\{\mathcal{X}_t\}_{t=0}^{H}$

$X_0 \leftarrow \texttt{BuildLinearTM}(\mathcal{X}_0)$
for $k = 0, \ldots, H-1$ **do**
  # Certify one-step map $x_{k+1} = f_{\text{dt,dyn}}(x_k, u_k)$
  $X_{k+1} \leftarrow \texttt{CROWN}\big(\lambda x.\, f_{\text{dt,dyn}}(x, u_k),\ X_k\big)$ ▷ affine bounds
  $\mathcal{X}_{k+1} \leftarrow \texttt{EvalInt}\big(X_{k+1}\big)$ ▷ box enclosure of next state
return $\{\mathcal{X}_k\}_{k=0}^{H}$

All operations are performed within the same linear-TM representation. Algorithm 2 summarizes this stepwise procedure, where each iteration applies CROWN-based certification followed by TM evaluation.

Compared to the continuous-time setting, this formulation removes inner-loop integration and yields a simpler, more efficient propagation scheme. It is particularly suitable for planning and control tasks requiring batched rollout over finite horizons, and integrates naturally with reachability-aware MPC and trajectory optimization. The same TM–CROWN interface can also support discrete-time neural feedback controllers analogously to the continuous-time case.

V. DiffReach-Robotics: Reachability-Aware Model Learning and Planning

Our reachability engine, DiffReach (Section IV), provides differentiable reachable-set objectives that can be embedded into downstream optimization. We use this primitive to regularize the training of NN components in Section V-A, and to incorporate reachable-set penalties into sampling-based MPC in Section V-B. We refer to these robotics-focused methods collectively as *DiffReach-Robotics* (Figure 2(b,c)).

A. Certified training of neural dynamics and controllers

We train neural dynamics and controllers with standard prediction or tracking losses augmented by reachability regularization: for rollouts under bounded initial uncertainty, we penalize the growth of certified reachable sets computed by the corresponding DT or CT reachability engine.

Training DT neural dynamics. We learn a discrete-time neural dynamics model $f_{\text{dt,dyn}}$ that is accurate over multi-step rollouts and yields compact certified reachable sets. From random-interaction data $\mathcal{D} = \{(x_t^m, u_t^m) \mid t = 0, \ldots, T-1,\ m = 0, \ldots, M-1\}$, we form M' episodes of length T_h+1 and train $f_{\text{dt,dyn}}$ with the autoregressive prediction loss

$$\mathcal{L}_{\text{pred}} = \frac{1}{M'T_h} \sum_{m=0}^{M'-1} \sum_{t=0}^{T_h-1} w_t \left\| x_{t+1}^m - \hat{x}_{t+1}^m \right\|_2^2, \quad (8)$$

where $\hat{x}_{t+1}^m = f_{\text{dt,dyn}}(\hat{x}_t^m, u_t^m)$, $\hat{x}_0^m = x_0^m$, and time-increasing weights $\{w_t\}$ emphasize long-horizon consistency.

Reachability-aware loss. Prediction accuracy alone does not control uncertainty growth, so we regularize the DT reachable tube from $\mathcal{X}_0^m = \mathcal{B}_\epsilon(x_0^m)$ under actions $u_{0:T_h-1}^m$. Let $\{\mathcal{R}_t(u_{0:T_h-1}^m, \mathcal{X}_0^m)\}_{t=1}^{T_h} = \{\mathcal{X}_t^m\}_{t=1}^{T_h}$ denote the certified reachable sets computed by Section IV-E. We define

$$\mathcal{L}_{\text{reach}} = \frac{1}{M'} \sum_{m=0}^{M'-1} \log\big(1 + V_{\mathcal{R},\epsilon}^m\big), \quad (9)$$

where $V^m_{\mathcal{R},\epsilon} = \sum_{t=1}^{T_h} \sum_{j=0}^{n-1} \texttt{width}(\mathcal{X}^m_{t,j})$ is a tube-volume proxy. The width sum and log term stabilize gradients when volumes vary widely. The final objective is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \lambda\, \mathcal{L}_{\text{reach}}$, with regularization weight λ.

Curriculum learning with horizon and perturbation scheduling. Joint optimization can become unstable for large rollout horizons T_h and perturbation radii ϵ, since early training may produce excessively large reachable sets and poorly conditioned reachability losses. To mitigate this, we use a curriculum that progressively increases T_h from one-step prediction to the target horizon using a logarithmic schedule, while simultaneously decreasing ϵ from a larger initial radius to the desired value. This stabilizes early training while encouraging reachability-friendly dynamics over long horizons. The full procedure is given in Algorithm 4 in Appendix C.

Training CT neural controllers. We train a CT neural controller $f_{\text{ct,ctl}}$ to track reference signals while producing compact certified reachable tubes. We collect a dataset $\mathcal{D} = \{(x^m_t, u^m_t, y^m_{\text{ref},t}) \mid t = 0, \dots, T-1,\ m = 0, \dots, M-1\}$ at control frequency $q_{\text{ctl}} = 1/\delta$ by executing a nominal controller under randomized references, and reformat it into M' episodes of length $T_t + 1$. Using a differentiable CT dynamics model $f_{\text{ct,dyn}}$ (analytical ODE or neural ODE) as a rollout proxy, we train $f_{\text{ct,ctl}}$ with

$$\mathcal{L}_{\text{track}} = \frac{1}{M' T_t} \sum_{m=0}^{M'-1} \sum_{t=0}^{T_t-1} w_t \Big(\|u^m_t - \hat{u}^m_t\|_2^2 + \gamma \|x^m_{t+1} - \hat{x}^m_{t+1}\|_2^2 \Big), \tag{10}$$

where $\hat{u}^m_t = f_{\text{ct,ctl}}(\hat{x}^m_t, y^m_{\text{ref},t})$ is held constant over $\tau \in [t\delta, (t+1)\delta]$, and $\hat{x}^m_{t+1}$ is obtained by integrating $\dot{\hat{x}}^m(\tau) = f_{\text{ct,dyn}}(\hat{x}^m(\tau), \hat{u}^m_t)$ using the JAX-based ODE solver `diffrax` [27]. The second term encourages consistency between predicted actions and their induced state transitions.

Analogously to DT dynamics training, we regularize $f_{\text{ct,ctl}}$ using the CT NN-control reachability engine (Section IV-D) and the same reachable-tube loss $\mathcal{L}_{\text{reach}}$ in Eq. 9, initialized from $\mathcal{B}_\epsilon(x^m_0)$. We use the same curriculum strategy to stabilize training and obtain controllers that are both performant and reachability-friendly. The same procedure also applies to CT dynamics training.

B. Reachability-aware MPC

Using the differentiable reachability engine and reachability-regularized neural models, we develop a reachability-aware MPC that jointly optimizes task cost and reachability penalties.

Planning. Solving Eq. 1 with neural dynamics in real time is challenging, as direct optimization and classical reachability-based methods are computationally expensive, while sampling-based planners such as MPPI [61] and CEM [43] must evaluate many candidate trajectories under uncertainty. We integrate the DT reachability engine (Section IV-E) directly into this process: for each sampled action sequence, we roll out the nominal dynamics, compute the corresponding reachable sets, and evaluate

$$O = \sum_{t=t_0}^{t_0+H} c(\hat{x}_t, u_t) + C \max(0, -G(\mathcal{R}_t(\boldsymbol{u}, \mathcal{B}_\epsilon(\hat{x}_{t_0})))) \tag{11}$$

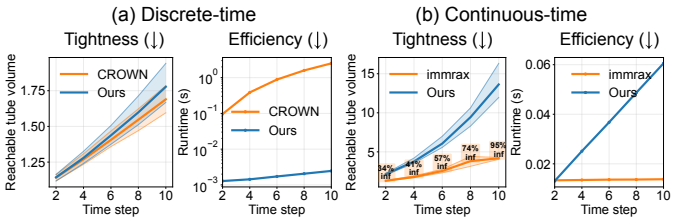

Fig. 3. **Benchmarking our JAX reachability tool. Top**: Compared with DT CROWN reachability, our method achieves similar reachable-set volumes while being ≈100× faster. **Bottom**: Compared with CT immrax reachability, our method produces substantially tighter reachable sets with comparable runtime.

where $\max(0, -G(\cdot))$ penalizes unsafe trajectories or large reachable sets, and C controls the penalty strength. Parallelization enables efficient evaluation across many samples, while gradient-based refinement is applied only to the top candidate at the final iteration to further reduce reachable-set size.

MPC. The planner is embedded in a receding-horizon MPC loop with online replanning. After each update, the first few planned actions and predicted states are passed as references to the neural controller $f_{\text{ct,ctl}}$ for low-level execution under $f_{\text{ct,dyn}}$. Guarantees are with respect to the learned dynamics; in practice, calibrated model-error bounds (e.g., via conformal prediction) can be incorporated into the reachable sets.

VI. Results

We evaluate the proposed framework along four directions: (i) runtime and tightness of the reachability engine against DT and CT baselines (Sec. VI-A); (ii) scalability and conservativeness on higher-dimensional systems (Sec. VI-B); (iii) certified training of neural dynamics and controllers (Sec. VI-C); and (iv) robustness of reachability-aware MPC in simulation and hardware (Sec. VI-D). All experiments were run on an RTX 4090 GPU.

A. Reachability Tool Evaluations

We benchmark the reachability engine against state-of-the-art DT (CROWN [69]) and CT (immrax [22]) baselines with GPU support in JAX in terms of runtime and reachable-set conservativeness (Fig. 3), before integrating the method into downstream learning and planning.

We perform reachability over a horizon of 10 for 128 randomly sampled $\mathcal{X}_0$. In the DT setting (Fig. 3, top), we use a randomly initialized neural dynamics model $f_{\text{dt,dyn}}$ with three 96-unit hidden layers. CROWN produces slightly tighter tubes because it propagates bounds jointly across time steps, unlike our truncated propagation (Alg. 4). However, our JAX implementation is ≈100× faster while maintaining comparable tightness.

For the CT setting (Fig. 3, bottom), we evaluate analytical quadrotor dynamics. immrax is slightly faster due to lightweight interval propagation, but produces substantially looser reachable sets and frequently returns infinite-volume tubes (up to 95%) from numerical overflow. Even after partitioning $\mathcal{X}_0$ into 64 subsets and taking the union of the resulting reachable sets, conservativeness remains high. We therefore report average tube volume over successful trials, and separately report the failure rate ("inf") for immrax. Overall, our method achieves comparable runtime with substantially tighter over-approximations.

Beyond forward reachability, the differentiable GPU-parallel implementation also enables practical refinement strategies for reducing conservativeness (Table I). Parallelized input splitting significantly reduces reachable-set volume with modest runtime increase, while gradient-based refinement further improves tightness without modifying the underlying algorithm. These results demonstrate that differentiable reachability can serve not only as a verification tool, but also as an optimization primitive for downstream learning and planning tasks.

TABLE I
REFINEMENT STRATEGIES.

Method	Volume	Runtime (s)
Baseline	91.94	0.0021
Input splitting (8)	27.56	0.0048
Gradient (20 iters)	18.18	0.1084

TABLE II
RUNTIME COMPARISON.

Method	1 Quad (12D)	6 Quad (72D)
Sampling	0.2601	2.2455
Ours	0.1399	0.1976

B. Scalability on Higher-Dimensional Systems

We further evaluate scalability and reachable-set conservativeness on moderately higher-dimensional robotic systems beyond the base T-pushing and single-quadrotor tasks.

We first evaluate a coupled 6-quadrotor swarm with *72D state* and *18D action*, where quadrotors move with spring-like coupling to the swarm center (Fig. 4). We apply 0.05 perturbations to position and velocity and perform reachability over 100 steps with $\Delta t = 0.02$. Compared with dense sampling (100k samples, under-approximation), our method produces reasonable over-approximations with average $1.37\times$ expansion on position dimensions while remaining efficient. As shown in Table II, runtime remains close to the single-quadrotor case despite the substantially larger state dimension. Although memory scales as $\mathcal{O}(n^2)$ under the affine representation, runtime scales more favorably in practice due to GPU-parallel matrix operations.

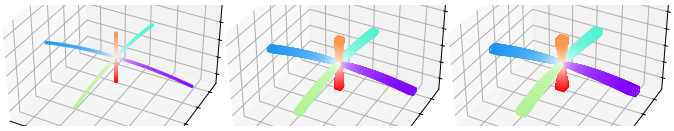

Fig. 4. **Reachability on a coupled 6-quadrotor swarm**. **(a)** Nominal trajectory. Quadrotors fly toward 6 different directions while coupled by spring-like attraction to the swarm center. **(b)** Dense sampled reachable set (under-approximation) under uncertainty. **(c)** Certified reachable set computed by our method. Our method produces a reasonable certified over-approximation.

We also evaluate an acceleration-controlled 10-joint planar arm with *40D state* and *10D action*. We apply ± 0.1 rad/s perturbations to joint velocities and perform reachability over 100 steps with $\Delta t = 0.01$. Table III compares against dense sampling (100k samples). Without splitting, reachable sets become noticeably more conservative on this articulated system. However, parallelized splitting substantially improves end-effector reachable widths while maintaining practical runtime. Overall, these results suggest that the framework can scale to moderately high-dimensional robotic systems, although tightness increasingly depends on system structure and refinement strategies such as splitting.

TABLE III
REACHABILITY ON A 10-JOINT ARM.

Method	EEF X width	EEF Y width	Runtime (s)
Sampling (100k)	0.4799	0.6256	1.0944
Ours (no split)	0.8414	1.2264	0.0531
Ours (64 splits)	0.8118	0.9756	0.1186
Ours (512 splits)	0.7353	0.8999	0.3854

C. Certified Training

We evaluate certified training of CT and DT dynamics models and controllers on T-pushing and quadrotor tasks (Fig. 5). Using 1,000 random initial sets $\mathcal{X}_0$ and test trajectories, we measure reachable-set tightness together with nominal prediction or tracking performance. Certified training achieves performance comparable to standard training with only slight degradation, while producing substantially tighter reachable sets. In contrast, models trained without reachability regularization yield significantly looser and less verifiable predictions.

To assess conservativeness, we estimate a lower bound on reachable-set volume (green, "sampled") by propagating 64 sampled initial states from each $\mathcal{X}_0$ through the true dynamics and measuring the axis-aligned bounding-box volume at each step. Across tasks, certified training remains substantially closer to these sampled lower-bound trends than regular training. Fig. 6 further illustrates this behavior: realized trajectories closely align with the certified reachable regions, demonstrating the fidelity of the resulting certificates.

We also study long-horizon reachable-set growth on the quadrotor task. Table IV reports the median reachable-volume ratio against dense sampling on valid trajectories with finite reachable-set volumes for DT Dyn and CT Ctl rollouts up to $5\times$ longer than those in Fig. 5. DT reachability remains relatively stable under stepwise propagation, while CT conservativeness grows substantially over long horizons due to trajectories with infinite-volume reachable sets. Parallelized splitting on critical roll-pitch-yaw states improves tightness and reduces the fraction of unstable CT trajectories from 42% to 24%. Since the reported ratios are computed only over valid trajectories, intermediate horizons may not monotonically reflect this improvement. Nevertheless, splitting substantially improves long-horizon behavior, reducing the $5\times$ horizon ratio from 18.6154 to 6.2407. These results suggest that although long-horizon CT reachability remains challenging, splitting provides a practical mechanism for mitigating conservativeness growth.

TABLE IV
TIGHTNESS OVER LONG HORIZONS ON QUADROTOR.

Horizon	$1\times$	$2\times$	$3\times$	$4\times$	$5\times$
DT Dyn (no split)	1.3221	1.3227	1.8782	1.8872	1.8898
CT Ctl (no split)	1.8648	2.1591	3.8261	3.8737	18.6154
CT Ctl (8 splits, rpy)	1.9171	2.9817	3.8261	3.0396	6.2407

D. Reachability-Aware MPC

Finally, we evaluate reachability-aware MPC under dynamics and control disturbances. For T-pushing, we apply bounded perturbations of 0.05 and 0.015 at the planning and low-level control layers; for quadrotor control, the corresponding bounds are 0.3 and 0.1. In Fig. 1, MPC plans toward goal states while enforcing state constraints with tight reachable sets under

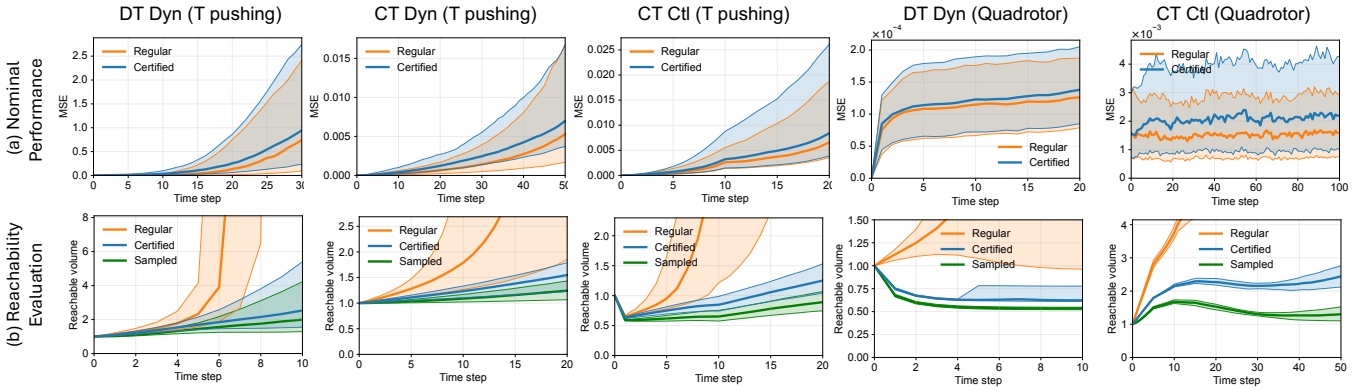

Fig. 5. **Benchmarking certified training. Top**: Across CT/DT systems and tasks, certified training maintains nominal performance comparable to naïve training. **Bottom**: Certified training produces substantially tighter reachable sets that more closely match sampled lower-bound volumes.

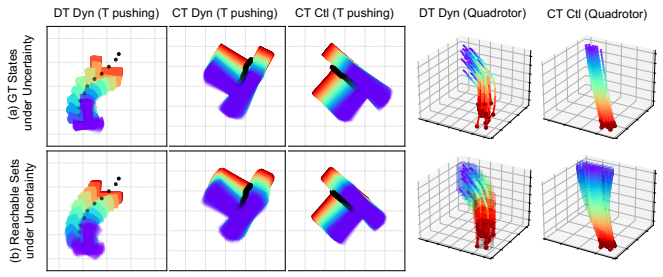

Fig. 6. **Certified training visualizations. Top**: Rollouts from sampled initial states in $\mathcal{X}_0$ under the true dynamics or controller. **Bottom**: States sampled from the certified reachable sets closely overlap with the realizable trajectories.

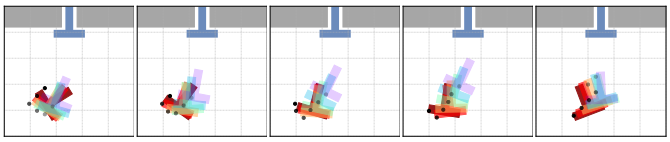

Fig. 7. **Failure case of vanilla MPC with non-certified dynamics.** The planner places the pusher too close to the T-shaped object, leading to unintended contact and failure to reach the goal.

uncertainty. The T-shaped object must avoid the gray unsafe region, while the quadrotor must avoid a spherical obstacle.

Fig. 1 visualizes MPC trajectories generated using the learned dynamics model and Eq. 1. For T-pushing, we sample configurations from the reachable sets to illustrate uncertainty propagation; for quadrotor navigation, we visualize reachable tubes projected into position space. Across both tasks, the planner satisfies constraints while remaining robust to bounded disturbances.

We further compare against vanilla MPC without reachability constraints using non-certified dynamics (Fig. 7). Here, the planner positions the pusher too close to the T-shaped object, and unintended contact perturbs the trajectory beyond recovery. This example highlights the importance of incorporating reachable-set information during planning rather than relying only on nominal predictions.

Quantitative comparisons are reported in Fig. 8. Under no or small disturbances, all methods achieve similar performance. Under larger disturbances, however, reachability-aware planning with certified models consistently achieves higher success rates, while vanilla MPC with regular models degrades substantially, especially on the quadrotor task. These results suggest that reachability-aware MPC improves robustness

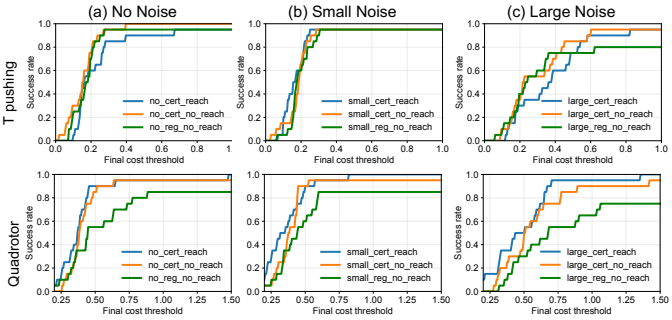

Fig. 8. **Reachability-aware planning and MPC.** MPC performance under **(a)** no, **(b)** small, and **(c)** large disturbances. We compare **1.** reachability-aware planning with certified models (blue), **2.** vanilla planning with certified models (orange), and **3.** vanilla planning with regular models (green). Performance is similar under small disturbances, while under large disturbances the regular-model baseline degrades substantially, especially for quadrotor control.

without excessive conservativeness in nominal settings.

We also demonstrate a hardware rollout of reachability-aware MPC on the T-pushing task using a Kuka iiwa platform (Fig. 1, middle row). The planner runs online at approximately 10 Hz (5 Hz with gradient-based refinement), enabling real-time planning and smooth execution. While the current hardware evaluation is qualitative, it demonstrates the practical feasibility of integrating differentiable reachability into online robotic control.

VII. Conclusion

We introduce a parallel, differentiable reachability framework for closed-loop robotic systems with NN dynamics and controllers. By combining Taylor-model flowpipes with CROWN-style linear bound propagation in JAX, we obtain a GPU-accelerated reachability primitive for robot learning and planning. We further develop certified training and reachability-aware MPC, enabling verification-aware optimization of models and actions. Experiments across manipulation and quadrotor tasks demonstrate practical online planning with certified reachable-set over-approximations under uncertainty.

Our current framework primarily targets smooth or learned dynamics and still faces challenges from long-horizon conservatism and hybrid or contact-rich dynamics. We hope this work motivates future research on scalable differentiable reachability and tighter integration between formal verification, robot learning, and control.

ACKNOWLEDGMENTS

This work was supported in part by a Mathworks Microgrant. We would like to thank the CROWN-Reach team [73], especially Xiangru Zhong and Prof. Huan Zhang, for insightful discussions. We also thank Jeffrey Fang and Wei-Chen Li for their assistance with hardware setup.

REFERENCES

[1] Matthias Althoff. An introduction to CORA 2015. In *Proc. of the 1st and 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151. EasyChair, December 2015. doi: 10.29007/zbkv. URL https://easychair.org/publications/paper/xMm.

[2] Martin Berz and Kyoko Makino. Verified integration of odes and flows using differential algebraic methods on high-order taylor models. *Reliable computing*, 4(4): 361–369, 1998.

[3] Victor Blomqvist. Pymunk. https://pymunk.org, November 2022.

[4] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. Juliareach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

[6] Christopher Brix, Stanley Bak, Taylor T Johnson, and Haoze Wu. The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results. *arXiv preprint arXiv:2412.19985*, 2024.

[7] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[8] Xin Chen. *Reachability analysis of non-linear hybrid systems using taylor models*. PhD thesis, Fachgruppe Informatik, RWTH Aachen University, 2015.

[9] Xin Chen and Sriram Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 13–24. IEEE, 2016.

[10] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, pages 183–192. IEEE, 2012.

[11] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[12] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[13] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3576–3583. IEEE, 2021.

[14] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Safe output feedback motion planning from images via learned perception modules and contraction theory. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 349–367. Springer, 2022.

[15] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[16] J. Zico Kolter Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, 2018.

[17] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *arXiv preprint arXiv:1605.07157*, 2016.

[18] Google DeepMind. jax_verify: Neural network verification in jax. https://github.com/google-deepmind/jax_verify, 2020. GitHub repository.

[19] 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. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[20] Sophie A Gruenbacher, Mathias Lechner, Ramin Hasani, Daniela Rus, Thomas A Henzinger, Scott A Smolka, and Radu Grosu. Gotube: Scalable statistical verification of continuous-depth models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6755–6764, 2022.

[21] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[22] Akash Harapanahalli, Saber Jafarpour, and Samuel Coogan. immrax: A parallelizable and differentiable toolbox for interval analysis and mixed monotone reachability in jax. *IFAC-PapersOnLine*, 58(11):75–80, 2024.

[23] Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522. IEEE, 2017.

[24] 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. Springer, 2022.

[25] Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in Neural Information Processing Systems*, 34:22745–22757, 2021.

[26] Konstantin Kaulen, Tobias Ladner, Stanley Bak, Christopher Brix, Hai Duong, Thomas Flinkow, Taylor T Johnson, Lukas Koller, Edoardo Manino, ThanhVu H Nguyen, et al. The 6th international verification of neural networks competition (vnn-comp 2025): Summary and results. *arXiv preprint arXiv:2512.19007*, 2025.

[27] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[28] 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.

[29] Craig Knuth, Glen Chou, Jamie Reese, and Joe Moore. Statistical safety and robustness guarantees for feedback motion planning of unknown underactuated stochastic systems. *arXiv preprint arXiv:2212.06874*, 2022.

[30] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[31] Haoyu Li, Xiangru Zhong, Bin Hu, and Huan Zhang. Neural contraction metrics with formal guarantees for discrete-time nonlinear dynamical systems. *arXiv preprint arXiv:2504.17102*, 2025.

[32] Haoyu Li, Xiangru Zhong, Bin Hu, and Huan Zhang. Two-stage learning of stabilizing neural controllers via zubov sampling and iterative domain expansion. *arXiv preprint arXiv:2506.01356*, 2025.

[33] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[34] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, and Mykel J. Kochenderfer. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4), 2021.

[35] 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, 2023.

[36] Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T Johnson. Nnv 2.0: The neural network verification tool. In *International Conference on Computer Aided Verification*, pages 397–412. Springer, 2023.

[37] Diego Manzanas Lopez, Matthias Althoff, Luis Benet, Clemens Blab, Marcelo Forets, Yuhao Jia, Taylor T Johnson, Manuel Kranzl, Tobias Ladner, Lukas Linauer, Philipp Neubauer, Sophie Neubauer, Christian Schilling, Huan Zhang, and Xiangru Zhong. Arch-comp24 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In Goran Frehse and Matthias Althoff, editors, *Proceedings of the 11th Int. Workshop on Applied Verification for Continuous and Hybrid Systems*, volume 103 of *EPiC Series in Computing*, pages 64–121. EasyChair, 2024. doi: 10.29007/mxld. URL /publications/paper/WsgX.

[38] Jingyue Lu and M. Pawan Kumar. Neural network branching for neural network verification. In *International Conference on Learning Representations (ICLR)*, 2020.

[39] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, 19 (3):20–32, 2012. doi: 10.1109/MRA.2012.2206474.

[40] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[41] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020.

[42] Diego Manzanas_Lopez, Matthias Althoff, Luis Benet, Clemens Blab, Marcelo Forets, Yuhao Jia, Taylor T Johnson, Manuel Kranzl, Tobias Ladner, Lukas Linauer, et al. Arch-comp24 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. 2024.

[43] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[44] 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 (NeurIPS)*, 2019.

[45] 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. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pages 5774–5781, 2025. doi: 10.1109/CDC57313.2025.11312100.

[46] Keyi Shen, Jiangwei Yu, Jose Barreiros, Huan Zhang, and Yunzhu Li. Bab-nd: Long-horizon motion planning with branch-and-bound and neural dynamics. *arXiv preprint arXiv:2412.09584*, 2024.

[47] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. *arXiv preprint arXiv:2205.02909*, 2022.

[48] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools, 2023.

[49] Zhouxing Shi, Haoyu Li, Cho-Jui Hsieh, and Huan Zhang. Certified training with branch-and-bound for lyapunov-stable neural control, 2026. URL https://arxiv.org/abs/2411.18235.

[50] Gagandeep Singh, Timon Gehr, Markus Püschel, and

Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages (POPL)*, 2019.

[51] Sumeet Singh, Mo Chen, Sylvia L Herbert, Claire J Tomlin, and Marco Pavone. Robust tracking with model mismatch for fast and safe planning: an sos optimization approach. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 545–564. Springer, 2018.

[52] Sumeet Singh, Benoit Landry, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust feedback motion planning via contraction theory. *The International Journal of Robotics Research*, 42(9):655–688, 2023.

[53] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11532–11539. IEEE, 2024.

[54] Anutam Srinivasan, Antoine Leeman, and Glen Chou. Safety beyond the training data: Robust out-of-distribution mpc via conformalized system level synthesis. *arXiv preprint arXiv:2602.12047*, 2026.

[55] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming, 2019.

[56] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas 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. Springer, 2020.

[57] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

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

[59] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-Resolution Model Learning for Object Pile Manipulation. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.047.

[60] Yixuan Wang, Weichao Zhou, Jiameng Fan, Zhilu Wang, Jiajun Li, Xin Chen, Chao Huang, Wenchao 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*, 43(3):994–1007, 2023.

[61] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[62] Junlin Wu, Huan Zhang, and Yevgeniy Vorobeychik. Verified safe reinforcement learning for neural network dynamic models, 2024. URL https://arxiv.org/abs/2405.15994.

[63] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

[64] 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 (NeurIPS)*, 2020.

[65] 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. *International Conference on Learning Representations (ICLR)*, 2021.

[66] 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.

[67] He Yin, Monimoy Bujarbaruah, Murat Arcak, and Andrew Packard. Optimization based planner–tracker design for safety guarantees. In *2020 American Control Conference (ACC)*, pages 5194–5200. IEEE, 2020.

[68] Hongchao Zhang, Junlin Wu, Yevgeniy Vorobeychik, and Andrew Clark. Exact verification of relu neural control barrier functions. *Advances in neural information processing systems*, 36:5685–5705, 2023.

[69] 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 (NeurIPS)*, 2018.

[70] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019.

[71] 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.

[72] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of ReLU networks. In *International Conference on Machine Learning (ICML)*, pages 26591–26604. PMLR, 2022.

[73] Xiangru Zhong, Yuhao Jia, and Huan Zhang. Crown-reach: A reachability analysis tool for neural network controlled systems. https://github.com/Verified-Intelligence/CROWN-Reach, 2024. GitHub repository, accessed January 30, 2026.

Algorithm 3 Reachability of CT Dynamics with TM Flowpipe. Comments are in brown.

1: **Inputs:** dynamics $f_{\text{ct,dyn}}$ (analytical or NN), initial box $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$, step size h, number of steps N, TM order k, initial remainder estimate ϵ_{init}, remainder refinement rounds R, auxiliary-window size M
2: **Outputs:** certified reachable set $\{\mathcal{X}_i\}_{i=0}^{N} = \{\underline{x}_i, \overline{x}_i\}_{i=0}^{N}$

3: # Build initial TM for the initial set: $x = c + Sy$
4: $[\underline{x}_0, \overline{x}_0] \leftarrow \mathcal{X}_0$
5: $c_0 \leftarrow (\underline{x}_0 + \overline{x}_0)/2, \quad S_0 \leftarrow (\overline{x}_0 - \underline{x}_0)/2$
6: $X^{(0)} \leftarrow \texttt{BuildLinearTM}(c_0, S_0)$ ▷ affine TM over unit box
7: $\mathcal{T}^{(0)} \leftarrow \texttt{IdentityTM}()$ ▷ initial linear parameterization
8: $\mathcal{S}^{(0)} \leftarrow \texttt{InitSymbolicState}(M)$ ▷ wrapping-control state

9: # Main loop
10: **for** $i = 0, \ldots, N-1$ **do**
11: # (A) Preconditioning / linear reparameterization
12: $\tilde{X} \leftarrow X^{(i)}\big|_{t:=h}$ ▷ evaluate previous step at end time
13: $(c, S, \mathcal{T}^{(i+1)}, \mathcal{S}^{(i+1)}) \leftarrow \texttt{SymbolicStep}(\mathcal{T}^{(i)}, \tilde{X}, \mathcal{S}^{(i)})$
14: $X \leftarrow \texttt{BuildLinearTM}(c, S)$ ▷ new affine seed $x = c + Sy$
15: # (B) Polynomial-only Picard iteration
16: $p_k \leftarrow \texttt{PolyPicard}(f_{\text{ct,dyn}},\ X,\ h,\ k)$
17: # (C) Seed and refine the remainder in Picard (soundness)
18: $X^{(i+1)} \leftarrow \texttt{RemainderPicard}\big(f_{\text{ct,dyn}},\ X,\ p_k,\ h,\ \epsilon_{\text{init}},\ R\big)$
19: # (D) Certified reachable set over the step segment
20: $X_{\text{seg}} \leftarrow X^{(i+1)} \circ \mathcal{T}^{(i+1)}$
21: $[\underline{x}_{i+1}, \overline{x}_{i+1}] \leftarrow \texttt{EvalNormInterval}(X_{\text{seg}},\ h)$

22: **return** $\{\underline{x}_i, \overline{x}_i\}_{i=0}^{N}$

Appendix A
Reachability of CT analytical dynamics

Algorithm 3 constructs a certified flowpipe for $\dot{x}(\tau) = f_{\text{ct,dyn}}(x(\tau))$ by propagating a TM enclosure over N fixed-duration segments. It first maps the initial box $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ to a normalized affine TM $X^{(0)} = c_0 + S_0 y$ where $y \in [-1, 1]^n$ and c_0 and S_0 are corresponding constant and linear coefficient (Lines 4–6). This normalization avoids unnecessary conservativeness when evaluating intervals over large and non-symmetric (zero-centered) input domains. It also defines a linear parameterization map $\mathcal{T}^{(i)}$ that tracks the cumulative normalization (preconditioning) applied on every step, which is initially an identity TM $\mathcal{T}^{(0)}$ (Line 7). The symbolic state $\mathcal{S}^{(i)}$ will be used to reduce the overestimation when propagating flowpipes step by step.

Each step then advances the enclosure with four operations.

- (A) *Preconditioning / reparameterization* (Lines 12-14) evaluates the previous step at $\tau = h$ to obtain an endpoint TM $\tilde{X}$, and computes a new affine seed $X_0 = c + Sy$ maintaining a short symbolic remainder state to keep computation tractable and subsequent enclosures tight.
- (B) *Polynomial Picard* (Line 16) applies truncated Picard iteration in TM arithmetic to obtain a k-order polynomial approximation p_k of the local flow map over $[-1, 1]^n$.
- (C) *Remainder Picard* (Line 18) then seeds and refines the interval remainder around p_k until the enclosure is self-consistent under the Picard update, yielding a sound TM flowpipe segment $X^{(i+1)}$.
- Finally, (D) the algorithm composes the segment with the accumulated linear parameterization $\mathcal{T}^{(i+1)}$ and evaluates it over $[-1, 1]^n$ to extract certified box bounds $[\underline{x}_{i+1}, \overline{x}_{i+1}]$ as reachable sets for use downstream in planning and learning (Lines 20,21).

Appendix B
Reachability of CT Neural dynamics

In this section, we aim to prove Theorem IV.3.

Proof: Let the linear TM input be

$$g(z) = c_g + A_g z \oplus I_g, \quad z \in \mathcal{Z},\ I_g = [\underline{I}_g, \overline{I}_g].$$

Equivalently, for every admissible input $x \in g(\mathcal{Z})$, there exist $z \in \mathcal{Z}$ and $r \in I_g$ such that

$$x = c_g + A_g z + r.$$

Define the reparameterized network

$$\tilde{f}(z, r) := f_{\text{NN}}(c_g + A_g z + r),$$

with inputs $(z, r) \in \mathcal{Z} \times I_g$. By construction,

$$f_{\text{NN}}(g(z)) = \tilde{f}(z, r) \quad \text{for some } r \in I_g.$$

Applying CROWN to $\tilde{f}$ over the box domain $\mathcal{Z} \times I_g$, with the enforced shared-slope constraint $\underline{\mathbf{W}} = \overline{\mathbf{W}}$, yields affine bounds

$$\widetilde{\mathbf{W}}_z z + \widetilde{\mathbf{W}}_r r + \tilde{b} \ \le\ \tilde{f}(z, r) \ \le\ \widetilde{\mathbf{W}}_z z + \widetilde{\mathbf{W}}_r r + \tilde{b},$$

where $\widetilde{\mathbf{W}}_z \in \mathbb{R}^{n_o \times n_z}$, $\widetilde{\mathbf{W}}_r \in \mathbb{R}^{n_o \times n_i}$, and $\tilde{b} \in \mathbb{R}^{n_o}$. Since the slopes are shared, the only remaining uncertainty is due to $r \in I_g$. Over-approximating the contribution of r by interval arithmetic gives

$$\widetilde{\mathbf{W}}_r r \in \widetilde{\mathbf{W}}_r I_g = [\underline{I}, \overline{I}],$$

where $\underline{I}, \overline{I}$ are obtained by elementwise interval evaluation. Now we define

$$P := \widetilde{\mathbf{W}}_z, \qquad c := \tilde{b}, \qquad I := [\underline{I}, \overline{I}].$$

Then for all $z \in \mathcal{Z}$,

$$f_{\text{NN}}(g(z)) \in c + Pz \oplus I,$$

which is a linear Taylor model that soundly over-approximates $f_{\text{NN}}(g(z))$. ∎

Appendix C
Certified training algorithm

We present our certified training algorithm for DT neural dynamics discussed in Section V-A in Algorithm 4. It includes multi-step loss, reachability regularization and curriculum schedules and can be adapted to train CT neural dynamics and controllers.

Algorithm 4 Training of DT Neural Dynamics.

Inputs: dataset $\mathcal{D} = \{(x_t^m, u_t^m)\}$, model $f_{\text{dt,dyn}}^{\theta}$, target horizon $T_h^{\max}$, target radius ϵ^{final}, weights $\{w_t\}$, penalty weight λ, optimizer `Opt`, DT reachability engine `DTReach`$(\cdot)$
Outputs: trained parameters θ

Reformat $\mathcal{D}$ into M' episodes $\{(x_{0:T_h^{\max}}^m, u_{0:T_h^{\max}-1}^m)\}_{m=0}^{M'-1}$
for training iteration $s = 0, \dots, S-1$ **do**
  # Curriculum schedules (horizon & uncertainty)
  $T_h \leftarrow$ `HorizonSchedule`$(s; T_h^{\max})$ ▷ logarithmic increase
  $\epsilon \leftarrow$ `EpsSchedule`$(s; \epsilon^{\text{final}}, T_h)$ ▷ smooth decrease
  Sample a mini-batch $\mathcal{B} \subset \{0, \dots, M'-1\}$
  $\mathcal{L}_{\text{pred}} \leftarrow 0, \quad \mathcal{L}_{\text{reach}} \leftarrow 0$
  for each episode $m \in \mathcal{B}$ **do**
    # (A) Autoregressive rollout for prediction loss
    $\hat{x}_0^m \leftarrow x_0^m$
    for $t = 0, \dots, T_h - 1$ **do**
      $\hat{x}_{t+1}^m \leftarrow f_{\text{dt,dyn}}^{\theta}(\hat{x}_t^m, u_t^m)$
      $\mathcal{L}_{\text{pred}} \mathrel{+}= w_t \|\hat{x}_{t+1}^m - x_{t+1}^m\|_2^2$
    # (B) DT reachability for reachability loss
    $\mathcal{X}_0^m \leftarrow \mathcal{B}_\epsilon(x_0^m)$
    $\{\mathcal{X}_t^m\}_{t=1}^{T_h} \leftarrow$ `DTReach`$\big(f_{\text{dt,dyn}}^{\theta}, \mathcal{X}_0^m, u_{0:T_h-1}^m\big)$
    $V_{\mathcal{R},\epsilon}^m \leftarrow \sum_{t=1}^{T_h} \sum_{j=0}^{n-1}$ `width`$(\mathcal{X}_{t,j}^m)$
    $\mathcal{L}_{\text{reach}} \mathrel{+}= \log(1 + V_{\mathcal{R},\epsilon}^m)$
  # (C) Combine losses and update parameters
  Normalize: $\mathcal{L}_{\text{pred}} \leftarrow \mathcal{L}_{\text{pred}}/(|\mathcal{B}|T_h)$, $\mathcal{L}_{\text{reach}} \leftarrow \mathcal{L}_{\text{reach}}/|\mathcal{B}|$
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{pred}} + \lambda\, \mathcal{L}_{\text{reach}}$
  $\theta \leftarrow$ `OptUpdate`$(\texttt{Opt}, \theta, \nabla_\theta \mathcal{L}_{\text{total}})$
return θ

APPENDIX D
ADDITIONAL EXPERIMENTAL DETAILS

a) T-pushing

We define the state of the T-pushing task as the concatenation of 3D T object pose (2D position and 1D orientation of the center of mass of the T object) and 2D position of the cylinder pusher. We train an MLP with hidden layers $[96, 96, 96]$ as the DT neural dynamics. We then train an MLP with hidden layers $[64, 64]$ as the CT neural dynamics. We finally train the neural controller (an MLP with hidden layers $[64, 64]$) with fixed CT neural dynamics for multi-step rollout. The action of this task is defined as the 2D velocity of the cylinder pusher. We collect our training data from the simulator Pymunk [3].

b) Quadrotor

We use a standard rigid-body quadrotor model [39] with world-frame translation, ZYX Euler angles, and body-frame angular rates.

The state is

$$x = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & p & q & r \end{bmatrix}^\top \in \mathbb{R}^{12},$$

where (x, y, z) are world positions, $(\dot{x}, \dot{y}, \dot{z})$ are world velocities, (ϕ, θ, ψ) are roll, pitch, and yaw (ZYX convention), and (p, q, r) are body angular rates. The control input is

$$u = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\top,$$

where u_1 is the total thrust magnitude along the body z-axis and (u_2, u_3, u_4) are body torques. In our experiments, the yaw torque input is fixed to zero ($u_4 = 0$), removing active yaw control, which is sufficient for the planar and position-focused maneuvers considered in this work.

Its position kinematics are as follows.

$$\dot{x} = \dot{x},$$
$$\dot{y} = \dot{y},$$
$$\dot{z} = \dot{z}.$$

Let R_{WB} denote the body-to-world rotation matrix under ZYX Euler angles and

$$b_3 = R_{WB} e_3 = \begin{bmatrix} \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \\ \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi \\ \cos\phi \cos\theta \end{bmatrix}.$$

The translational dynamics in the world frame are

$$\ddot{x} = \frac{u_1}{m} b_{3x},$$
$$\ddot{y} = \frac{u_1}{m} b_{3y},$$
$$\ddot{z} = \frac{u_1}{m} b_{3z} - g,$$

where m is the mass and g is gravitational acceleration.

The Euler angle rates driven by body angular rates (p, q, r) are

$$\dot{\phi} = p + \sin\phi \tan\theta\, q + \cos\phi \tan\theta\, r,$$
$$\dot{\theta} = \cos\phi\, q - \sin\phi\, r,$$
$$\dot{\psi} = \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r.$$

Assuming a diagonal inertia matrix $J = \mathrm{diag}(J_x, J_y, J_z)$, the rigid-body rotational dynamics are

$$\dot{p} = \frac{J_y - J_z}{J_x} qr + \frac{u_2}{J_x},$$
$$\dot{q} = \frac{J_z - J_x}{J_y} pr + \frac{u_3}{J_y},$$
$$\dot{r} = \frac{J_x - J_y}{J_z} pq + \frac{u_4}{J_z}.$$

We train an MLP with hidden layers $[64, 64]$ as the DT dynamics for planning. Its state is defined as the 3D position of the quadrotor without including all remaining states in GT analytical ODE dynamics. We apply velocity control with a neural controller (an MLP with hidden layers $[64, 64]$) accepting the desired 3D velocity command as the input and output CT actions u. We collect training data from the analytical dynamics with a nominal PD controller.