**Robotics: Science and Systems 2025**
**Los Angeles, CA, USA, June 21–25, 2025**

# Bridging Model Predictive Control and Deep Learning for Scalable Reachability Analysis

Zeyuan Feng[1], Le Qiu[2], and Somil Bansal[1]

***Abstract*—Hamilton-Jacobi (HJ) reachability analysis is a widely used method for ensuring the safety of robotic systems. Traditional approaches compute reachable sets by numerically solving an HJ Partial Differential Equation (PDE) over a grid, which is computationally prohibitive due to the *curse of dimensionality*. Recent learning-based methods have sought to address this challenge by approximating reachability solutions using neural networks trained with PDE residual error. However, these approaches often suffer from unstable training dynamics and suboptimal solutions due to the weak learning signal provided by the residual loss. In this work, we propose a novel approach that leverages model predictive control (MPC) techniques to guide and accelerate the reachability learning process. Observing that HJ reachability is inherently rooted in optimal control, we utilize MPC to generate approximate reachability solutions at key collocation points, which are then used to tactically guide the neural network training by ensuring compliance with these approximations. Moreover, we iteratively refine the MPC-generated solutions using the learned reachability solution, mitigating convergence to local optima. Case studies on a 2D vertical drone, a 13D quadrotor, a 7D F1Tenth car, and a 40D publisher-subscriber system demonstrate that bridging MPC with deep learning yields significant improvements in the robustness and accuracy of reachable sets, as well as corresponding safety assurances, compared to existing methods.**

## I. Introduction

Hamilton-Jacobi (HJ) reachability analysis is one of the most widely used tools for providing formal safety assurances for autonomous systems. It characterizes the unsafe states of the system via Backward Reachable Tube (BRT) – the set of all initial states from which a system failure is inevitable. Thus, the complement of the BRT provides a safe set for the system. Along with the safe set, reachability analysis provides a safety controller for the system that can be used as it is or alongside a nominal (potentially data-driven) controller to maintain safety.

In HJ reachability, the BRT computation is framed as an optimal control problem that results in solving a certain Hamilton-Jacobi-Bellman PDE (HJB-PDE) [29, 26]. Solving this PDE yields a safety value function that implicitly represents both the BRT and the safety controller. Consequently, a number of methods have been developed to solve the HJB PDE. Traditional methods solve HJB PDE numerically over a grid, which suffers from the so-called "curse of dimensionality" [1]. Specifically, the computation scales exponentially with the system dimension, making systems of more than 6D intractable. Many techniques for speeding up reachability analysis put restrictions on the type of system allowed and/or assign predefined shapes to the safe set (e.g. ellipsoids, polytopes) [8, 7, 14, 19, 24, 25, 11, 15]. However, computing reachable sets for general nonlinear systems remains a challenge. This motivated the recent development of learning-based methods to approximate the HJB-PDE solution [30]. One line of research leverages Reinforcement Learning (RL) to approximate the safety value function, achieving impressive performance improvements [13, 17, 18, 20]. Another line of work [2, 34], which this paper aims to enhance, learns value function via self-supervised learning by minimizing the residuals of HJB-PDE. The latter set of approaches, termed DeepReach variants, are rooted in recent advances in Physics-Informed Machine Learning [31, 21] and have the theoretical advantage of recovering the exact HJB-PDE solution as the training loss converges to zero [16]. However, in practice, these methods often converge to suboptimal solutions due to the weak learning signal provided by the residual loss alone. Additionally, as we will demonstrate, this can result in non-physical solutions, where the learned value function significantly deviates from the ground truth, even when training loss appears low. Consequently, these methods exhibit instability and inaccuracies, especially for stiff dynamical systems or problems with complex boundary conditions (i.e., intricate safety specifications). To combat these challenges, [32] leverages the Hopf formula to generate HJB-VI solutions for linearized dynamics and learns the solution using semi-supervision. However, this method does not synthesize value function labels for the original nonlinear dynamics and relies heavily on the quality of the Hopf solutions. Another method imposes exact safety specification [34], but it offers limited improvements, and learning instability for general nonlinear systems remains a key challenge.

To overcome these challenges, we propose a framework that leverages approximate reachability solutions to guide the learning of the safety value function. Our approach is inspired by the success of incorporating data-driven loss terms in Physics-Informed Neural Networks (PINNs) [5], which effectively anchors trial solutions at key collocation points alongside PDE residuals, thereby accelerating convergence and guiding learning toward feasible solutions. Specifically, building on the optimal control foundations of HJ reachability, we propose a sampling-based MPC approach to gener-

[1]Authors are with the Department of Aeronautics and Astronautics at Stanford University, USA :{zeyuanf, somil}@stanford.edu.

[2]Author is with the Department of Electrical Engineering at Tsinghua University, China: {qiule1026@gmail.com}.

This research is supported in part by the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program and by the NSF CAREER program (2240163).

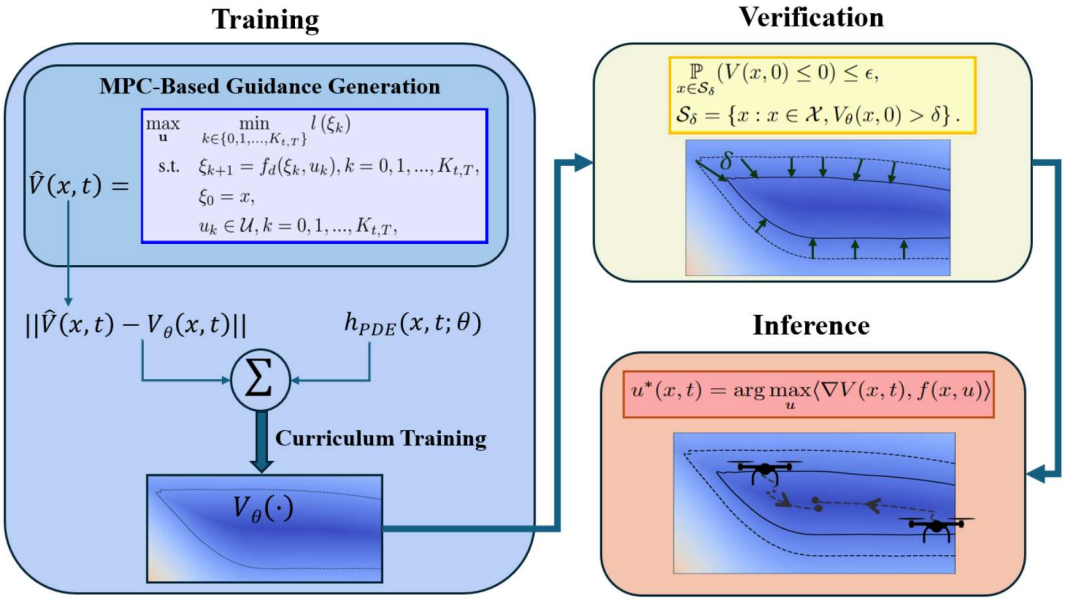


Fig. 1: We propose a framework that efficiently generates approximate safety value function datasets using a sampling-based MPC approach and integrates these data labels to guide the learning of reachability solutions for high-dimensional autonomous systems. The learned value function is then verified using conformal prediction, providing probabilistic safety assurances for the system under the induced safe policy.

ate approximate solutions for high-dimensional reachability problems. These solutions serve as semi-supervised labels to improve value function learning. Our method is designed to be highly parallelizable on GPUs, allowing for efficient synthesis of large datasets. Moreover, we provide an algorithm to iteratively update the MPC dataset during training using the learned DeepReach policy, progressively improving accuracy and stability of the learned value function. By combining sampling-based MPC with DeepReach-style self-supervised learning, this work integrates the strengths of both approaches – leveraging MPC for efficient and structured dataset generation and DeepReach for reliable safety assurances. To summarize, our overall contributions are:

- A novel data-generation approach that efficiently computes approximate HJB-VI solutions using an MPC-based sampling method.
- A hybrid training algorithm that balances residual loss with data-driven supervision to improve the convergence of value function learning, while iteratively refining the MPC dataset during training.
- Demonstration of the proposed approach on four challenging reachability problems: a 2D vertical drone avoiding ground and ceiling, a 13D quadrotor avoiding a cylindrical obstacle, a 7D F1Tenth car navigating a track, and a 40D publisher-subscriber system, highlighting a significant improvement in accuracy and stability of value function learning.

## II. Preliminaries

In this section, we formulate the reachability problem and provide a brief background on HJ reachability.

### A. Problem Setup

We consider a dynamical system with time-invariant, possibly nonlinear, dynamics: $\dot{x} = f(x,u)$, where $x \in \mathbb{R}^n$ is the state and $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$ is the control input. The safety requirement is defined by a Lipschitz-continuous function $l : \mathbb{R}^n \to \mathbb{R}$, with its sub-zero level set being the failure set, i.e., $\mathcal{L} = \{x : l(x) \le 0\}$. For example, in a robotic navigation setting, $\mathcal{L}$ may represent obstacles, while $l(x)$ could be the signed distance function to these obstacles.

Our primary goal in this work is to compute the BRT of the system, denoted as $\mathcal{B}$, which consists of all initial states from which the system will inevitably enter the failure set $\mathcal{L}$ within the time horizon $[0,T]$, regardless of the control strategy. Formally:

$$\mathcal{B} = \left\{ x : \forall u(\cdot) \in \mathcal{U}, \exists \tau \in [0,T], \xi_{x,0}^{u(\cdot)}(\tau) \in \mathcal{L} \right\}, \quad (1)$$

where $\xi_{x,t}^{u(\cdot)}(\tau)$ is the state achieved at time $\tau$ by applying the control policy $u(\cdot)$, starting from initial state $x$ at time $t$. Thus, $\mathcal{B}$ captures all states from which the system is doomed to fail. The complement of the BRT, $\mathcal{S} := \mathcal{B}^C$, represents the safe set for the system, i.e., the initial states from which it is possible to avoid entering the failure region. Correspondingly, our second objective is to synthesize a safe control policy $u^*(\cdot)$ that ensures the system remains in the safe set $\mathcal{S}$ and does not enter $\mathcal{B}$, whenever possible.

### B. Computation of BRT Using HJ Reachability

In HJ reachability, the computation of $\mathcal{B}$ is formulated as a continuous-time optimal control problem, with the following cost function:

$$J(x,t,u(\cdot)) = \min_{\tau\in[t,T]} l\left(\xi_{x,t}^{u(\cdot)}(\tau)\right). \quad (2)$$

Here, $J(x,t,u(\cdot))$ represents the minimum "distance" to the failure set along the state trajectory starting at $x$ and governed by $u(\cdot)$ over $[t,T]$. The value function and the optimal controller can be obtained by maximizing this cost function:

$$\begin{aligned} V(x,t) &= \sup_{u(\cdot)\in\mathcal{U}_{[t,T]}} J(x,t,u(\cdot)), \\ u^*(\cdot) &= \arg\sup_{u(\cdot)\in\mathcal{U}_{[t,T]}} J(x,t,u(\cdot)). \end{aligned} \tag{3}$$

Once $V(x,t)$ is computed, the BRT is given by the sub-zero level set of the value function:

$$\mathcal{B} = \{x : V(x,0) \leq 0\}. \tag{4}$$

Mathematically, BRT consists of all states where the minimum distance to the failure set is negative under the optimal control. In other words, the system must have entered the failure set at some time in $[t,T]$ and hence these states are contained in the BRT.

The value function $V(x,t)$ can be computed using the principle of dynamic programming which results in the following Hamilton-Jacobi-Bellman Variational Inequality (HJB-VI):

$$\begin{aligned} \min\{D_t V(x,t) + H(x,t), l(x) - V(x,t)\} = 0, \\ V(x,T) = l(x), \\ H(x,t) = \max_{u\in\mathcal{U}} \langle \nabla V(x,t)\ ,\ f(x,u)\rangle, \end{aligned} \tag{5}$$

where the second line specifies the boundary condition, $H(x,t)$ is the Hamiltonian, $D_t$ represents the time derivatives and $\nabla$ denotes the spatial derivatives of the value function. For more details on HJ reachability analysis and the derivation of HJB-VI, we refer readers to [23, 28, 1].

With the value function in hand, the optimal safety controller to keep the system outside the BRT is given by:

$$u^*(x,t) = \arg\max_u \langle \nabla V(x,t), f(x,u)\rangle. \tag{6}$$

For low-dimensional systems, HJB-VI can be solved numerically on a state-space grid using various toolboxes [28, 29, 23, 26, 12]. However, for high-dimensional systems, these methods suffer from the curse of dimensionality; consequently, learning-based methods have been proposed to solve HJB-VI. This work builds upon one such line of learning methods called DeepReach [2, 34, 6], that learn the HJB-VI solution for high-dimensional problems in a self-supervised manner by minimizing the following HJB-VI residuals:

$$\begin{aligned} \mathop{\mathbb{E}}_{(x_i,t_i)\in\mathcal{X}\times[0,T]} [\| \min\{D_t V_\theta(x_i,t_i) + H(x_i,t_i), \\ l(x_i) - V_\theta(x_i,t_i)\} \|]. \end{aligned} \tag{7}$$

Here, the value function is approximated as $V_\theta(x,t) = l(x) + (T-t)\cdot O_\theta(x,t)$, where $O_\theta(x,t)$ is the output of a NN with trainable parameters $\theta$. This formulation inherently satisfies the boundary condition $V_\theta(x,T) = l(x)$, thereby leaving only the PDE residual errors in the loss function (7). Again, we emphasize that learning the value function for complex, high-dimensional reachability problems using pure self-supervision remains challenging and unstable. To address this, we aim to leverage approximated value function data to enhance the training process.

## III. Learning Reachability Solutions with MPC-Based Guidance

Our approach consists of three key steps: first, we generate an approximate safety value function using a sampling-based MPC method that optimizes the safety cost function in (2). These approximate value samples are then used to augment the training loss of DeepReach to incentivize the learned value function to be consistent with them. Finally, the MPC-guided value function is corrected for potential learning errors using a conformal prediction scheme, thereby providing a verified safe set of the system. We now explain each of these steps in detail. An overview of our approach is in Fig. 1.

### A. Generating Approximate Value Function Dataset Using Sampling-Based MPC

Our key idea is that HJ reachability computes the BRT via formulating it as an optimal control problem where the cost function is given by (2). Thus, an approximate safety value function can be obtained by solving this optimal control problem using alternative optimal control methods, such as MPC. Subsequently, this approximate value function can be leveraged to enhance the training of DeepReach.

Specifically, we compute an approximate value function, $\hat{V}$, by solving the following discrete-time version of the optimal control problem:

$$\begin{aligned} \hat{V}(x,t) = \max_{\mathbf{u}} \quad & \min_{h\in\{0,1,\ldots,H_{t,T}\}} l(\xi_h) \\ \text{s.t.} \quad & \xi_{h+1} = f_d(\xi_h, u_h), h = 0,1,...,H_{t,T}, \\ & \xi_0 = x, \\ & u_h \in \mathcal{U}, h = 0,1,...,H_{t,T}, \end{aligned} \tag{8}$$

where $h \in \{0,1,...,H\}$ denotes the time steps between $t$ and $T$, $\xi_h$ is the system state, and $\mathbf{u} := [u_0, \cdots, u_H]$ is the control sequence. $f_d$ are the discretized dynamics that can be obtained from the continuous dynamics using first-order Euler approximation. The optimization problem in (8) can be solved using a broad class of MPC methods. Specifically, we leverage sampling-based MPC methods to solve this discrete-time problem, as they are simple, fast, and highly parallelizable.

Our MPC algorithm is summarized in Alg. 1. The algorithm takes as inputs the number of initial states $|D_{MPC}|$, the number of hallucinated trajectories $N$, the number of iterative sampling steps $R$, the time horizon $H$, the nominal control $\mathbf{u}_{nom}$, the time step $\Delta$, and the state space of interest $\mathcal{X}$. The output of the algorithm is a dataset $\mathcal{D}_{MPC}$, where each data point consists of an initial time $t$, an initial state $x$, and a corresponding value function approximation $\hat{V}(t,x)$. To generate a single data point, the algorithm begins by sampling $N$ distinct control sequences, $\tilde{\mathbf{u}}_n$, around the nominal control sequence $\mathbf{u}_{nom}$. These $N$ trajectories are rolled out using discretized dynamics with a time step of $\Delta$, and the corresponding cost values $J_n$ are computed. Finally, the nominal control sequence is updated to the best-performing control sequence, and the process is repeated for $R$ iterations. This process is highly parallelizable in practice and can be significantly accelerated with modern GPU computation.

**Algorithm 1** MPC Dataset Generation

**Input:** $|D_{MPC}|$, $N$, $R$, $H$, $\mathbf{u}_{nom}$, $\Delta$, $\mathcal{X}$
**Output:** $\mathcal{D}_{MPC} = \left\{(t_i, x_i, \hat{V}(t_i, x_i))\right\}$
**Initialize:** $\mathcal{D}_{MPC} = \emptyset$
1: **for** $i = 1 : |D_{MPC}|$ **do**
2: $x_i \sim$ Uniform($\mathcal{X}$); $t_i \sim$ Uniform$(0, T)$
3: **for** $r = 1 : R$ **do**
4: **for** $n = 1 : N$ **do**
5: $\tilde{\mathbf{u}}^n \sim$ Gaussian$(\mathbf{u}_{nom}, \sigma^2)$
6: $\xi_0^n \leftarrow x_i$
7: **for** $h = 0 : H - 1$ **do**
8: $\xi_{h+1}^n = \xi_h^n + f(\xi_h^n, \tilde{\mathbf{u}}_h^n)\Delta$
9: $J^n = \min_{h=0,1,\ldots,H} l(\xi_h^n)$
10: $n^* = \arg\max_n J^n$
11: $\hat{V}(t_i, x_i) \leftarrow J^{n^*}$
12: $\mathbf{u}_{nom} \leftarrow \tilde{\mathbf{u}}^{n^*}$
13: $\mathcal{D}_{MPC} \leftarrow \mathcal{D}_{MPC} \cup (t_i, x_i, \hat{V}(t_i, x_i))$

We conclude this section with a few remarks about the proposed MPC method.

*Remark 1:* In practice, we set one of the sampled control sequences in Line-5 to $\mathbf{u}_{nom}$. This allows for a monotonic improvement in the MPC value function over $R$ iterations.

*Remark 2:* We can efficiently bootstrap the MPC dataset by computing the running value function along the optimal state-control trajectory, terminating when the minimum $l(x)$ is reached. Specifically, if $\xi^{n^*}$ denotes the state trajectory under the optimal control sequence $\tilde{\mathbf{u}}^{n^*}$, we can estimate the value function at intermediate points along this trajectory as:

$$\hat{V}(\Delta \cdot h, \xi_h^{n^*}) = \min_{j=h,\ldots,H} l(\xi_j^{n^*}).$$

Accordingly, we add the data point $(\Delta \cdot h, \xi_h^{n^*}, \hat{V}(\Delta \cdot h, \xi_h^{n^*}))$ to $\mathcal{D}_{MPC}$. This approach enables rapid collection of a large, high-quality MPC dataset for guiding the learning process.

*Remark 3:* Several design choices remain for the MPC algorithm, including how to compute the next control input (e.g., weighted sum vs. best sample) and the choice of control perturbation distribution (e.g., uniform vs. Gaussian). Empirically, we found that selecting the best sequence and using Gaussian sampling yielded the least noisy datasets. However, these options remain configurable, allowing users to tailor the approach to their specific needs.

### *B. Learning Reachability Solution Using MPC Dataset and HJB-VI Residuals*

Our approach (summarized in Algorithm 2) leverages a three-phase learning scheme for learning the safety value function: pretraining, curriculum training, and fine-tuning. The goal is to train a value function that accurately captures the BRT, using data-driven supervision from the MPC dataset and HJ reachability-based PDE residual loss.

**Pretraining Phase: Supervised Learning from MPC Dataset.** During the pretraining phase, the model is trained using the collected MPC dataset, which consists of sampled states and their corresponding estimated value function. The training objective minimizes the prediction error between the learned value function $V_\theta(x, t)$ and the approximated value function:

$$\begin{aligned} \mathcal{L}_{data} &= \sum_{i=1}^{N_{MPC}} h_{data}(x_i^{MPC}, t_i^{MPC}, \hat{V}_i; \theta), \\ h_{data}(x, t, \hat{V}; \theta) &= \|\hat{V}(x, t) - V_\theta(x, t)\|. \end{aligned} \tag{9}$$

This supervised learning step aligns the learned value function with the approximated reachability solution inferred from the MPC dataset, providing an informative "warmstart" of safe and unsafe regions for the neural network.

**Curriculum Training: Joint Optimization with PDE Loss.** After pretraining, we refine the value function using a curriculum training strategy that gradually increases the horizon of training samples, meaning the time horizon is progressively increased from $[T, T]$ to $[0, T]$ over $N_c$ iterations. Since HJB-VI is a terminal time PDE, the accuracy of safety value function at earlier time steps (corresponding to smaller $t$) depends on the accuracy of predictions at later times. Intuitively, this temporal curriculum reduces the complexity of the learning problem. The loss function during curriculum training consists of two components: the data-driven loss ($\mathcal{L}_{data}$) from the MPC dataset and the HJB-VI residual loss ($\mathcal{L}_{PDE}$), which enforces the HJ reachability equation:

$$\begin{aligned} \mathcal{L}_{PDE} &= \sum_{i=1}^{N_{PDE}} h_{PDE}(x_i^{PDE}, t_i^{PDE}; \theta), \\ h_{PDE}(x, t; \theta) &= \| \min \{D_t V_\theta(x, t) + H(x, t), \\ &\qquad l(x) - V_\theta(x, t)\} \|. \end{aligned} \tag{10}$$

The overall loss function during the curriculum learning phase is given by:

$$\mathcal{L}_{combined} = \mathcal{L}_{PDE} + \lambda \mathcal{L}_{data}, \tag{11}$$

where a trade-off factor $\lambda$ is dynamically adjusted based on the loss gradients to balance the contributions of $\mathcal{L}_{PDE}$ and $\mathcal{L}_{data}$, making them equally important.

A crucial step in the curriculum training phase is MPC dataset refinement. The initial dataset generated by the sampling-based MPC method is inherently suboptimal and noisy for long horizons, which can hinder the convergence of the PDE loss and cause the model to overfit to outliers, leading to violations of the governing HJB-VI equations. To mitigate this, we periodically refine the MPC dataset using the learned value function.

Specifically, when the time curriculum reaches the effective horizon, $H_R$, of the current MPC dataset (i.e., when $t = t_R$), we leverage the learned value function and policy to generate a new MPC dataset with an extended time horizon of $[t_R - H_R, T]$. This updated dataset is computed by incorporating the learned value function as the terminal cost in the MPC formulation, providing a more accurate estimate of the

**Algorithm 2** DeepReach Training with MPC Dataset

**Input:** $N_{PDE}$, $N_{MPC}$, $\mathcal{D}_{MPC}$, $\alpha$, $N_c$, $H_R$
**Output:** $V_\theta(x,t)$
**Initialize:** $\lambda \leftarrow 1.0$, $t_R \leftarrow T - H_R$

**for** each pretraining step **do**
  $(t_i, x_i, \hat{V}_i) \sim Uniform(\mathcal{D}_{MPC})$, $i = 1, 2, ..., N_{MPC}$
  Compute $\mathcal{L}_{data}$ using (9)
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{data}$
**for** $n_c = 1 : N_c$ **do**
  $t = T\left(\frac{N_c - n_c}{N_c}\right)$
  $(x_i^{PDE}, t_i^{PDE}) \sim$ Uniform$(\mathcal{X} \times [t, T])$, $i = 1 : N_{PDE}$
  $(x_i^{MPC}, t_i^{MPC}, \hat{V}_i) \sim$ Uniform$(\mathcal{D}_{MPC})$
  Compute $\mathcal{L}_{data}$ and $\mathcal{L}_{PDE}$ using (9) and (10)
  $\lambda \leftarrow 0.9\lambda + 0.1 \frac{||\nabla_\theta \mathcal{L}_{PDE}||}{||\nabla_\theta \mathcal{L}_{data}||}$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{combined}$
  **if** $t < t_R$ **then**
    Refine $\mathcal{D}_{MPC}$ using $V_\theta$
    $t_R \leftarrow t_R - H_R$
**for** each fine-tuning step **do**
  Repeat the curriculum step with $t = 0$ and $\mathcal{L}_{data} := \mathcal{L}_{data,FT}$

optimal cost-to-go over the horizon $[t_R, T]$ while keeping the effective MPC horizon fixed at $H_R$. With this terminal cost, we repeat the MPC dataset generation procedure in Algorithm 1, effectively obtaining the MPC value function approximation over the time horizon $[t_R - H_R, T]$. This dataset is then used to continue the curriculum training for another $H_R$ seconds and the entire process is repeated again. By iteratively refining the dataset every $H_R$ seconds, we ensure that the MPC dataset remains informative and consistent, continuously guiding the learning process with higher-quality supervisory signals.

**Fine-Tuning: Reducing False Positives for Safety Guarantees.** The final fine-tuning phase improves the accuracy of the learned solution by employing a smaller learning rate and refining the safety decision boundary. A key challenge is minimizing false positives, where the learned value function incorrectly predicts a state as safe but is actually driven to failure under the safe policy induced by the value function. Such errors can lead to catastrophic consequences in real-world safety-critical systems.

To address this, the fine-tuning process introduces a modified data-driven loss function, where the loss for false positives is amplified by a scaler parameter $\lambda_{FP}$. This ensures that safety predictions remain conservative, reducing the likelihood of unsafe behaviors. The loss function is defined as follows:

$$\mathcal{L}_{data,FT} = \sum_{i=1}^{N_{MPC}} h_{FT}(x_i^{MPC}, t_i^{MPC}, \hat{V}; \theta),$$

$$h_{FT}(x, t, \hat{V}; \theta) = \begin{cases} \lambda_{FP} \|\hat{V} - V_\theta(x,t)\| V_\theta(x,t), & \text{if } V_\theta(x,t) \geq 0 \wedge \hat{V} < 0 \\ \|\hat{V} - V_\theta(x,t)\|. & \text{otherwise} \end{cases} \tag{12}$$

Note that since fine-tuning is done after the curriculum learning phase, MPC is able to use the learned safety policy (induced by the learned value function via Eqn (6)) to generate an informative nominal control sequence for computing $\hat{V}$. This results in a high quality approximation of the value function, which in turn, boosts the quality of the learned value function through the fine tuning loss in (12).

### C. Safety Assurances for the Learned Value Function

In general, neural network can make errors and, as such, the safe set provided by the learned value function is only a candidate safe set – there might be states in the learned safe set that steer the system into the failure region. To overcome this challenge, we leverage the formal verification method proposed in [22] that uses conformal prediction to determine a probabilistic safe set given a candidate value function $V_\theta$ and the induced safety policy, $u_\theta$.

The overall idea is to provide a high-confidence bound $\delta$ on the value function error. The corrected value function is then used to compute the BRT and the safe set. Specifically, the method requires users to specify a confidence parameter $\beta \in (0, 1)$ and a safety violation parameter $\epsilon \in (0, 1)$. It then computes the error bound $\delta$ using conformal prediction to ensure that with at least $1 - \beta$ confidence, the probability of safety violation within the super-$\delta$ level set of $V_\theta$ is upper-bounded by $\epsilon$:

$$\mathbb{P}_{x \in \mathcal{S}_\delta}(V(x,0) \leq 0) \leq \epsilon, \tag{13}$$

where $\mathcal{S}_\delta = \{x : x \in \mathcal{X}, V_\theta(x,0) > \delta\}$ and $V(x,0)$ is the underlying ground-truth value function. In other words, the probability of a state within the super-$\delta$ level set of $V_\theta$ being actually unsafe is at most $\epsilon$. Thus, $\mathcal{S}_\delta$ represents a high-confidence estimate of the safe set.

## IV. Experiments

We now evaluate the benefits of including MPC-based guidance when learning reachability solutions. We consider four different case studies: a 2D vertical drone system, a 13D nonlinear quadrotor system, a 7D high-speed F1Tenth car system, and a 40D publisher-subscriber system. Each of these systems presents unique challenges for reachability analysis due to their dimensionality, complex dynamics, and/or complex geometry of the failure set.

### A. Baselines

We compare the safe set obtained via the proposed method with the following baselines:

- **Vanilla DeepReach** [34]: A NN-based safety value function obtained using DeepReach with exact boundary condition imposition (self-supervision only).
- **MPC Distillation**: A NN-based safety value function distilled from a dense MPC dataset (supervision only).
- **Neural CBF**: A NN-based control barrier function (CBF) learned using the approach proposed in [9, 10].

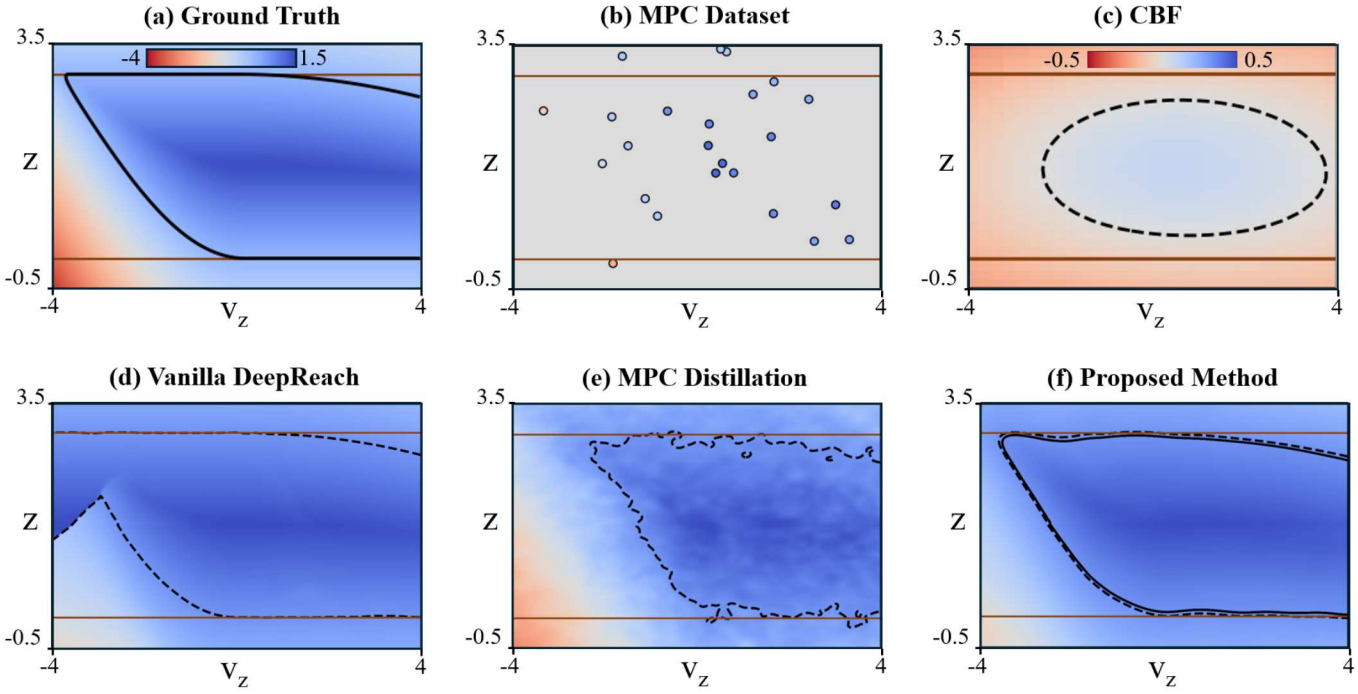


Fig. 2: Parameterized Vertical Drone: Value function slices at $K = 12$. The brown lines represent the ground and the ceiling – the failure set in this case. In (a), the solid black lines are the contours of ground-truth safe sets. In (b), all MPC data samples with $K \in (11, 12)$ used for the proposed approach are shown. (c) illustrates the learned safe set using the neural CBF approach. For (d), (e), and (f), the dashed contours illustrate the learned safe sets and the solid contours represent the verified safe sets. Note that the black solid contours are missing in (d) and (e) since Vanilla DeepReach and MPC Distillation have an empty verified safe set.

- **Ground Truth Value Function**: Wherever possible, we compute the ground-truth value function by solving the HJB-VI numerically using the OptimizedDP toolbox [4]. Note that this is computation is feasible only up to 5D-6D systems.

Each of the value functions corresponds to an implicit safe policy, as defined in Eqn (6). We additionally compare the efficacy of these learned safe policies in maintaining system safety.

| Parameter | Vertical Drone | Quad-rotor | F1Tenth | 40D Publisher-Subscriber |
|---|---|---|---|---|
| $T$ | 1.2 | 1.0 | 1.0 | 1.0 |
| $\mathcal{N}_{PDE}$ | 65K | 65K | 65K | 65K |
| $\mathcal{N}_{MPC}$ | 10K | 10K | 10K | 10K |
| Pretraining | 20K | 20K | 20K | 20K |
| Curriculum | 40K | 100K | 200K | 120K |
| Fine-Tuning | 10K | 10K | 10K | 10K |

TABLE I: Detailed parameters for training.

### B. Evaluation Metrics

Our key evaluation metric is the volume of the (verified) safe set, denoted as $\mu_\epsilon$, which we refer to as the **recovered volume**. For all value functions, the verified safe set is obtained by the verification procedure outlined in Sec. III-C, with $\epsilon = 10^{-3}$ and $\beta = 10^{-16}$. This corresponds to obtaining a safe set, $\mathcal{S}_\delta$ with a $99.9\%$ safety rate. To estimate the volume of the safe set over the state space of interest, we compute the likelihood of a uniformly sampled state lying in the set:

$$\mu_\epsilon \approx \frac{\sum_{i=1}^{M} \mathbb{1}(x_i \in \mathcal{S}_\delta)}{M}, x_i \sim Uniform(\mathcal{X}). \quad (14)$$

We choose a large value of $M = 1 \times 10^6$ to attain samples from a significant portion of the state space.

For low dimensional problems where the ground truth is available, we also compute the Mean Squared Error (MSE) and false positive rate compared to the ground-truth value function.

### C. Implementation Details

For all baselines, we employ a three-layer sinusoidal NN, with 512 neurons per layer. The NNs are trained using the Adam optimizer with a learning rate of $\alpha = 2 \times 10^{-5}$ on an NVIDIA GeForce RTX 4090 GPU. The sampling-based MPC method uses a time step of $\Delta = 0.02s$ and 10 iterative sampling steps ($R = 10$), where N=100 perturbed control sequences are generated per step. The fine-tuning loss weight $\lambda_{FT}$ is set to 100 and the dataset refinement horizon $H_R$ is set to $0.2s$ for all experiments.

In each case study, we use the same number of training iterations across all baselines to ensure a fair comparison. However, Vanilla DeepReach and MPC Distillation do not involve a fine-tuning phase, as we observe that fine-tuning encourages overfitting in these baselines and degrades their performance. Detailed training parameters are provided in Table I.

The total training time for all baselines is reported in

| | Baseline | Recovered (%) | Time [h] |
|---|---|---|---|
| **Vertical Drone** | Vanilla | 0.00 | 0.05 |
| | Distillation | 0.00 | 0.1 |
| | Neural CBF | NA (31.28) | 0.3 |
| | Proposed | 24.62 (56.51) | 0.1 |
| **Quadrotor** | Vanilla | 71.26 | 1.6 |
| | Distillation | 42.99 | 2.5 |
| | Neural CBF | 51.70 | 6.0 |
| | Proposed | 93.69 | 3.0 |
| **F1Tenth** | Vanilla | 0.00 | 2.0 |
| | Distillation | 0.00 | 4.5 |
| | Neural CBF | 167.14 | 20.0 |
| | Proposed | 76.08 | 5.0 |
| **Publisher-Subscriber** | Vanilla | 38.21 | 1.3 |
| | Proposed | 97.14 | 2.8 |

TABLE II: Recovered safe set volumes and training time for different case studies. The proposed approach consistently achieves higher safe set volumes compared to the baselines. We note that the safe sets obtained by the NeuralCBF method cannot be verified directly; thus, the reported (learned) volumes are likely optimistic and would reduce further upon verification. For the vertical drone system, the volumes corresponding to $K = 12$ are shown in brackets.

Table II. The additional computation cost of the proposed method mainly arises from the weight-balancing step (Line 10 in Alg. 2) and computing the data-driven loss (gradients).

### *D. Parameterized 2D Vertical Drone*

We consider a drone navigating longitudinally along the z-axis, with its dynamics defined as:

$$\dot{z} = v_z, \;\; \dot{v}_z = Ku - g, \;\; \dot{K} = 0, \tag{15}$$

where $z \in [-0.5, 3.5]$ is the height, $v_z \in [-4.0, 4.0]$ denotes the vertical velocity, $g = 9.8$ is the gravitational acceleration, and $K \in [0.0, 12.0]$ is a constant control gain parameter. The control input $u \in [-1, 1]$ corresponds to the vertical acceleration effort. The system aims to avoid crashing into the ground or the ceiling, with its failure set $\mathcal{L} = \{x : |z - 1.5| \geq 1.5\}$. Our goal is to compute the safe set of the system for different values of $K$.

Since the dynamics are low-dimensional (2D state and 1D parameter), we compute the ground-truth value function through OptimizedDP [4] and use it to evaluate the mean squared error (MSE). The ground truth safe set volume is 26.22%. To train the proposed method, we generate 300 MPC data points ($|D_{MPC}| = 300$) in approximately 1s for the parameterized 2D vertical drone system. These data points were bootstrapped instantly using the procedure discussed in Remark 2, leading to an overall dataset of size around 10K.

We first compare the performance of all baselines. As shown in Fig. 2, Vanilla DeepReach converges to a completely non-physical solution, with an MSE of 7.81 and a dramatically overoptimistic safe set. This particular reachability problem represents a corner case where Vanilla DeepReach suffers from instability issue, despite the simplicity of both the system dynamics and boundary conditions, and results in a verified safe set volume of zero. On the other hand, the distilled value function, which minimizes data-driven loss on a dense dataset with roughly 10M MPC samples, attains a lower MSE of 0.41. Although the distilled value function is closer to the ground truth, it exhibits a noisy value profile due to the absence of PDE loss. Consequently, the learned value function and the induced safety policy suffer from inaccuracies, resulting in a recovered safe volume of zero, demonstrating that the MPC method alone is not sufficient to obtain a high-quality value function despite a dense dataset. In contrast, the proposed method shows a significant improvement with a small MPC dataset ($|D_{MPC}| = 300$), achieving a low MSE of 0.009 and a recovered volume of 24.62%.

Since the Neural CBF baseline learns a converged safe set, we only compare this baseline with the proposed approach for $K = 12$. For $K = 12$, the safe set converges within $T = 1.2$ $s$, enabling a fair comparison. At $K = 12$, the learned safe set volume from Neural CBF is 31.28%, whereas our method achieves a significantly larger recovered volume of 56.51%. This is also evident from the respective safe set boundaries (Fig. 2(c) and (f)).

*Ablation study: effect of MPC dataset size.* To assess the effect of dataset size, we train the value function with several sparse datasets. The number of initial states used to generate the MPC dataset ranges from 20 to 200, resulting in approximately 700 to 7000 data samples throughout the time horizon after bootstrapping. Each dataset takes less than a second to generate. For each dataset size, we train a 3-layer sinusoidal network with 128 neurons per layer using 5 different random seeds. As illustrated in Fig. 4, the recovered volume quickly converges to around 24% with $N_{MPC} \approx 160$. This result confirms that the proposed method exhibits a high data efficiency and learn accurate value function with minimal MPC guidance. This characteristic becomes increasingly important as system dimensionality increases and the data available becomes more sparse. The same trend is observed in the false positive rate (evaluated using ground truth value function).

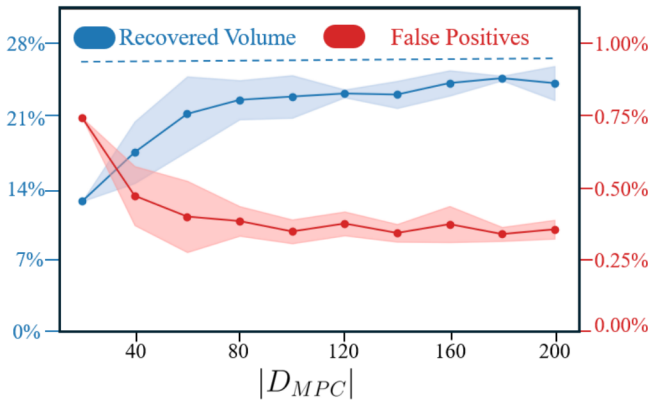


Fig. 4: The x-axis shows the size of non-bootstrapped dataset. The blue line represents the mean recovered volumes along with their standard deviation across five random seeds, whereas the red line shows the mean false positive rates.

*Ablation study: effect of MPC dataset refinement.* Given

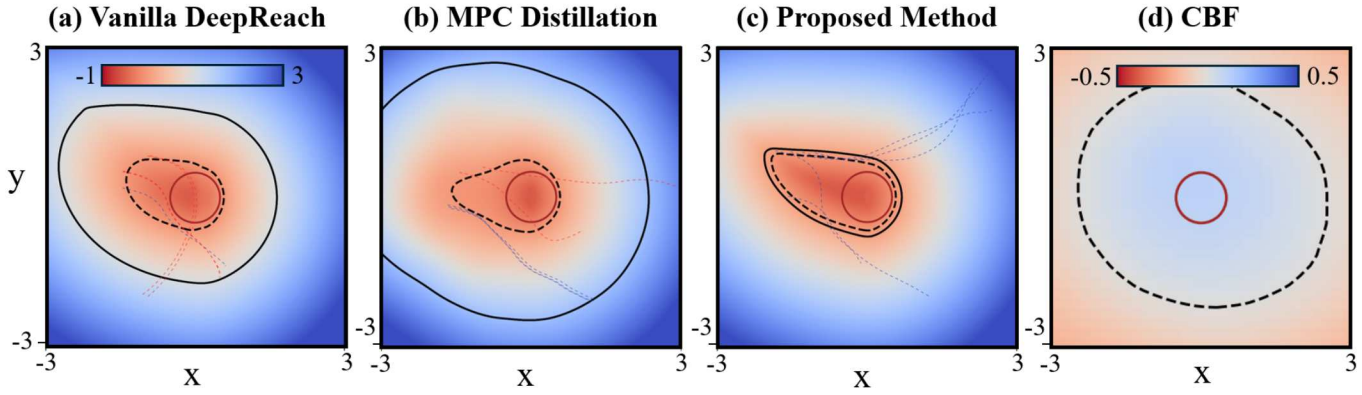


Fig. 3: 13D Quadrotor: slices of learned value functions on the X-Y plane at $(p_z, q_\omega, q_x, q_y, q_z, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z) = (0.54, 0.44, -0.45, 0.27, -0.73, 5.00, -1.07, -3.34, 3.19, -2.80, 3.43)$. The brown circles represent the cylinder obstacle (not the failure set contour due to the quadrotor's nonzero collision radius). Again, the solid contours are the verified safe sets while the dashed contours are the learned BRTs. The dashed lines are selected rollout trajectories, where red color indicates collisions.

the MPC dataset is inherently suboptimal and noisy, we investigate the impact of the iterative dataset refinement step on the algorithm's performance. To this end, we evaluate the proposed method without dataset refinement, which results in a higher MSE of $0.2285$ and a lower recovered volume of $20.04\%$. All metrics deteriorate compared to the case where the MPC dataset is refined periodically, highlighting the utility of the refinement step to reduce the suboptimality of the MPC guidance.

### *E. 13D Quadrotor System*

We next evaluate the proposed method and the baselines on a high-dimensional reachability problem involving an extremely agile quadrotor system. The state of the drone is represented as $x = [p_x, p_y, p_z, q_\omega, q_x, q_y, q_z, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]$, where $(p_x, p_y, p_z) \in [-3, 3]^3$ denote the position and $(v_x, v_y, v_z) \in [-5, 5]^3$ denote the linear velocities. $(q_\omega, q_x, q_y, q_z) \in [-1, 1]^4$ represent the quaternion and $(\omega_x, \omega_y, \omega_z) \in [-5, 5]^3$ are angular velocities. The control inputs consist of a collective thrust and angular acceleration efforts $u = [F, \alpha_x, \alpha_y, \alpha_z]$, where $F \in [-20, 20]$, $\alpha_x, \alpha_y \in [-8, 8]$, and $\alpha_z \in [-4, 4]$. The quadrotor is modeled as a disk with a radius of $0.17$ m. The safety function $l(x)$ is defined as the signed distance from the disk to an infinitely long cylinder with a radius of $0.5$ m centered at the origin. Consequently, the failure set $\mathcal{L}$ comprises all states where the quadrotor collides with the obstacle. As a reference, the failure set volume is $3.09\%$. A detailed description of $l(x)$ and the quadrotor dynamics is provided in the Appendix VII-A1.

Once again, Vanilla DeepReach learns an overly optimistic value function. The inaccuracies in the value function leads to a low recovered safe volume of $71.26\%$. In contrast, the proposed method, trained with 1M data samples, significantly outperforms Vanilla DeepReach by achieving a higher recovered volume of $93.69\%$, highlighting the utility of MPC-based guidance. The MPC Distillation baseline, despite being trained on 15M data samples, leads to the lowest recovered volume of $42.99\%$. This is because the dataset remains too sparse to provide adequate interpolation in this high-dimensional problem, highlighting the utility of incorporating PDE loss during the value function learning. The Neural CBF baseline attains a learned volume of $51.7\%$ and does not have a recovered volume since the verification method is not applicable. Overall, the proposed method yields the best safe controller along with the largest verified safe set. Note that although all baselines appear to achieve a substantial verified safe set, the performance gaps remain significant given the small failure set and the agility of the quadrotor, as better illustrated in Fig. 3.

*Ablation studies.* We conducted ablation studies on the MPC dataset size and found that the proposed method can synthesize high-fidelity BRTs using relatively sparse data (Table III). Additionally, we observed that removing the time curriculum reduced the recovered volume to zero, whereas omitting either of the other two stages had minimal impact (Table IV). These results highlight the critical role of the time curriculum in solving complex reachability problems despite its additional computational cost, while the other two stages are optional and primarily enhance stability further.

| $\lvert D_{MPC}\rvert$ Quadrotor/F1 | | Bootstrapped Dataset Size | | Dataset Generation Time | | Recovered Volume | |
|---|---|---|---|---|---|---|---|
| 5k | 150k | 35k | 1.8M | 20 s | 6 m | 89.8% | 0.0% |
| 15k | 210k | 110k | 2.6M | 1.5 m | 8 m | 91.7% | 53.4% |
| 30k | 300k | 230k | 3.3M | 2.5 m | 10 m | 93.7% | 76.3% |
| 50k | 360k | 390k | 4.5M | 4 m | 13 m | 93.8% | 76.0% |
| 100k | 450k | 750k | 5.9M | 6 m | 15 m | 93.5% | 74.2% |

TABLE III: Dataset statistics for Quadrotor (Left) and F1Tenth (right).

### *F. 7D F1Tenth*

To further assess the robustness and accuracy of our method, we introduce an extremely challenging reachability problem with both stiff dynamics and a complex boundary condition: the F1Tenth racing car problem. The system is tasked with navigating a high-speed RC car on a track while avoiding the

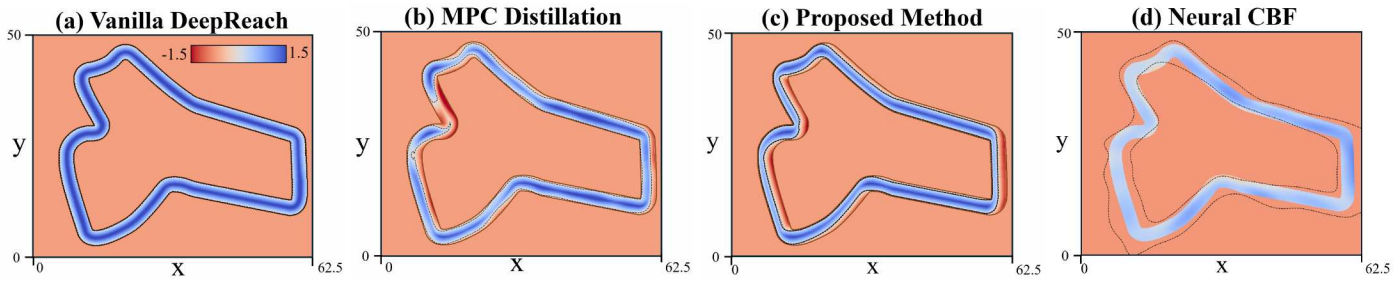


Fig. 5: F1Tenth: X-Y Plane Slices of the Learned Value Functions. The remaining state variables are $(\phi, v, \theta_{yaw}, \omega_{yaw}, \beta_{slip}) = (0, 8, 0, 0, 0)$. The track boundaries are shown as the brown contours. The black solid contours are the verified safe sets while the dashed contours are the learned BRTs. Note that the black solid contours are missing in (a), (b), and (d) since these baselines have an empty verified safe set. On the other hand, the black solid contour and dashed contour overlap significantly for the proposed method, with $\delta = 0.12$, highlighting the high accuracy of the learned safety value function.

| Training Configuration | Vertical Drone | Quadrotor | F1Tenth |
|---|---|---|---|
| Full Training (All Stages) | 24.62% | 93.69% | 76.08% |
| Without Fine-tuning | 24.37% | 93.18% | 66.53% |
| Without Pretraining | 24.00% | 93.87% | 65.61% |
| Without Time Curriculum | 23.50% | 0.00% | 0.00% |

TABLE IV: Effect of training stages on recovered volume.

curbs. The failure set $\mathcal{L}$ consists of all states where the car is outside the track, and $l(x)$ is defined as the signed distance function to the edge of the track.

The state variables of F1Tenth is given by $x = (p_x, p_y, \phi, v, \theta_{yaw}, \omega_{yaw}, \beta_{slip}) \in [0, 62.5] \times [0, 50] \times [-0.4189, 0.4189] \times [0, 8] \times [-\pi, \pi] \times [-5, 5] \times [-0.8, 0.8]$, where $(p_x, p_y)$ are the global coordinates of the vehicle, $\phi$ is the front-wheel steering angle, $v$ is the velocity, $\theta_{yaw}$ is the yaw angle, $\omega_{yaw}$ is the yaw rate, and $\beta_{slip}$ denotes the slip angle at the vehicle's center. The control inputs are the rate of the front-wheel steering angle and the longitudinal acceleration, denoted as $u = [\dot{\phi}, a] \in [-3.2, 3.2] \times [-9.51, 9.51]$. The longitudinal acceleration is further capped when the current velocity $v$ is high, and the system switches to kinematic mode when the velocity is too low. A detailed description of the system dynamics is provided in the Appendix VII-B.

Given the hybrid dynamics and high agility of the vehicle, learning an accurate safety value function is challenging. This challenge is further exasperated by the complex geometry of the failure set and the large state space of interest. Both Vanilla DeepReach and the MPC Distillation approach fail completely to recover any verified safe set volume in this case. Vanilla DeepReach suffers again from the instability issue, causing the learned value function to not evolve through the time curriculum and instead converges to $l(x)$, as shown in Fig. 5a. In this case study, the Neural CBF baseline performs the worst, producing a completely nonphysical safe set that intersects heavily with the failure set. In contrast, our method significantly outperforms all baselines by achieving an impressive recovered volume of 76.08% with 3M data samples.

We further roll out the obtained safety policy, as well as the sampling-based MPC policy, from different initial states and quantify the empirical safe volume under these policies. We compare the two policies for $v > 6$ as the race car primarily operates at a high speed. The policy learned by our method demonstrates a more aggressive (fallback) driving strategy, achieving a safe volume of 76.87%, compared to 69.38% for the MPC policy, highlighting the utility of HJB-VI residuals that encourage global optimality in the learned value function and the safety policy. In terms of online inference speed, our method computes the optimal safe control in approximately 2 ms, whereas the sampling-based MPC approach requires 39 ms. Both methods could be further accelerated by migrating to C++.

*Ablation studies.* The results of the MPC dataset size and training stage ablation studies are presented in Table III and Table IV, respectively, and yield consistent conclusions. The curriculum training remains the key training phase for the proposed approach. Notably, both computation time and dataset size scale primarily with problem complexity rather than system dimensionality, as the 7D F1Tenth Race Car problem requires more computation time and dataset samples compared to the higher-dimensional 13D quadrotor problem. This conclusion is further corroborated with a 40D publisher-subscriber system in the next case study.

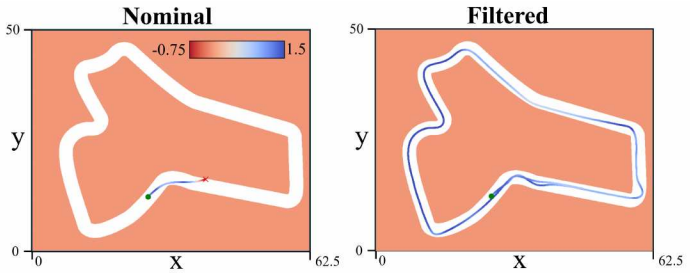


Fig. 6: Trajectories under the nominal controller and the safety filter with $\gamma = 1$. The green dot marks the starting points of the trajectories, while the red "X" denotes collisions. The filtered policy successfully completes an entire lap without collision.

***Safety Filtering:*** Reachability-based safety value function can be used for safety filtering where a potentially unsafe nom-

inal controller is minimally modified to render it certifiably safe. The nominal controller here is an imitation policy trained using an MPC-based expert. We implement a smooth blending safety filter, inspired by CBF-QP, following the approach in [3] with $\gamma = 1.0$.

To assess both performance and safety, we evaluate the nominal and filtered policies based on two metrics: collision rate and distance travelled before collision. We evaluate the system for a horizon of 25 s and each policy is evaluated over 30 different trajectories. At the start of each trajectory, the F1Tenth car is initialized inside the verified safe set (as per our method). Fig. 6 illustrates trajectories under the nominal controller and the safety filter along with the value function predictions. The nominal controller fails to keep the car on the track, whereas the filtering technique—leveraging the learned safety value function—enables the car to complete an entire lap without collision. Quantitatively, the nominal policy achieves an average distance travelled of 36.17 m with a 100% collision rate. The filtered policy successfully operates without any collision, achieving average distances of 187.01 m. These results highlight a substantial improvement in both performance and safety when leveraging the learned value function for downstream safety filtering and control.

### G. Case Study: 40 Dimensional Publisher-Subscriber System

To demonstrate the scalability of the proposed approach to high-dimensional systems, we apply it to a 40-dimensional "publisher-subscriber" system, defined as:

$$\begin{bmatrix} \dot{\mathrm{x}}_0 \\ \dot{\mathrm{x}}_i \end{bmatrix} \triangleq \begin{bmatrix} a & 0 \\ -1 & a \end{bmatrix} \begin{bmatrix} \mathrm{x}_0 \\ \mathrm{x}_i \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} u_i + \begin{bmatrix} \alpha \sin(\mathrm{x}_0)\, \mathrm{x}_0^2 \\ -\beta \mathrm{x}_0 \mathrm{x}_i^2 \end{bmatrix}, \quad i = 1, ..., 39, \tag{16}$$

where $x_0$ represents the publisher state, which unidirectionally influences each subscriber state $x_i$. For this study, we set $\alpha = 20$, $\beta = 0$, and constrain the control input by $|u_i| \leq 1$. The target set which the agents aim to reach is given by $\mathcal{L} = \left\{x : \frac{1}{2}(x_0^2 + x_i^2 - 0.5) \leq 0, \forall i = 1, ..., 39\right\}$. The ground truth value function is obtainable via system decomposition; we refer interested readers to [33] for further system details.

We adapt the same network architecture as the previous case studies and the training parameters are provided in Table I. Training required approximately 3 hours with a $|D_{MPC}|$ of 80K—roughly the same computational burden as the 13D quadrotor system despite the tripled dimensionality. This further supports our claim that the computational burden of the proposed method scales primarily with problem complexity rather than system dimensionality.

Vanilla DeepReach produced a value function with an MSE of 11.27 and recovered only 38.21% of the ground truth BRT volume. In contrast, our method achieved a significantly more accurate value function with an MSE of 0.0062 and recovered 97.14% of the ground truth BRT volume (see Fig. 7), demonstrating its effectiveness in high-dimensional settings.

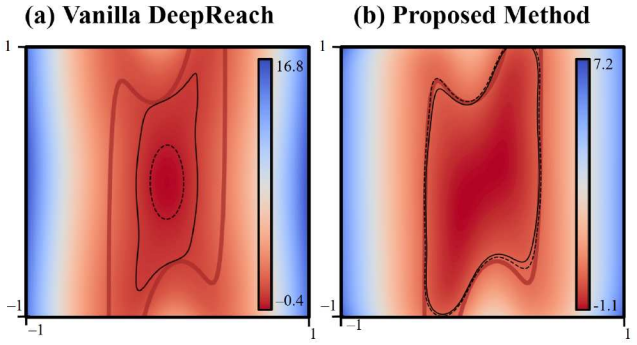


Fig. 7: Value function slices for the 40D system. The brown contours represent the ground-truth BRT slices. The dashed lines indicate the learned BRTs while the solid lines are the verified safe set contours.

## V. Conclusion

In this work, we propose a framework that efficiently generates approximate value function datasets using a sampling-based MPC approach and integrates these data labels to improve state-of-the-art learning-based reachability methods. By leveraging data-driven supervision alongside PDE-based residuals, our method enhances the accuracy and scalability of reachability analysis for high-dimensional systems. Through several case studies on challenging reachability problems, we demonstrate that our method consistently outperforms existing approaches, achieving better accuracy and faster convergence.

## VI. Limitations

While the results are promising, several open questions remain. One key challenge is understanding how the distribution of data across the temporal-spatial domain affects the training performance – specifically, what sampling strategies could maximize learning efficiency. Developing importance sampling techniques, which further improve data efficiency by prioritizing informative samples, would be an interesting future direction.

Additionally, this work focuses on safety problems, where value functions typically converge over short horizons, and has not explored long-horizon problems (e.g., reach-avoid problems). It would be interesting to combine the proposed approach with alternative neural PDE solution methods, such as sequence-to-sequence learning [27], to effectively synthesize long-horizon PDE solutions. It would also be interesting to consider alternative optimal control methods to provide guidance for learning safety value function.

Beyond these algorithmic considerations, our approach does not currently account for disturbances and dynamics uncertainty, which are critical in ensuring safety in uncertain and adversarial settings. Extending the framework to incorporate robust safety under disturbances presents a non-trivial challenge and is an important direction for future work.

## References

[1] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and re-

cent advances. In *IEEE Conference on Decision and Control*, 2017.

[2] Somil Bansal and Claire J. Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. doi: 10.1109/ICRA48506.2021.9561949.

[3] Javier Borquez, Kaustav Chakraborty, Hao Wang, and Somil Bansal. On safety and liveness filtering using hamilton-jacobi reachability analysis. *IEEE Transactions on Robotics*, 2024.

[4] Minh Bui, George Giovanis, Mo Chen, and Arrvindh Shriraman. Optimizeddp: An efficient, user-friendly library for optimal control and dynamic programming. *arXiv preprint arXiv:2204.05520*, 2022.

[5] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[6] Vamsi Krishna Chilakamarri, Zeyuan Feng, and Somil Bansal. Reachability analysis for black-box dynamical systems. *arXiv preprint arXiv:2410.07796*, 2024.

[7] Yat Tin Chow, Jérôme Darbon, Stanley Osher, and Wotao Yin. Algorithm for overcoming the curse of dimensionality for time-dependent non-convex Hamilton–Jacobi equations arising from optimal control and differential games problems. *Journal of Scientific Computing*, 73 (2-3):617–643, 2017. ISSN 0885-7474.

[8] Jérôme Darbon and Stanley Osher. Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere. *Research in the Mathematical Sciences*, 3(1):19, 2016. ISSN 2197-9847.

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

[10] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1724–1735. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/dawson22a.html.

[11] Tommaso Dreossi, Thao Dang, and Carla Piazza. Parallelotope bundles for polynomial reachability. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 297–306, 2016.

[12] Jaime F Fisac, Mo Chen, Claire J Tomlin, and S Shankar Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th international conference on hybrid systems: computation and control*, pages 11–20, 2015.

[13] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[14] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

[15] Didier Henrion and Milan Korda. Convex computation of the region of attraction of polynomial control systems. *Transactions on Automatic Control (TAC)*, 59(2):297–312, 2013. ISSN 0018-9286.

[16] William Hofgard. Convergence guarantees for neural network-based hamilton-jacobi reachability. *arXiv preprint arXiv:2410.02904*, 2024.

[17] Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. *arXiv preprint arXiv:2112.12288*, 2021.

[18] Kai-Chieh Hsu, Duy Phuong Nguyen, and Jaime Fernandez Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. In *Learning for Dynamics and Control Conference*, pages 90–103. PMLR, 2023.

[19] Alexander B Kurzhanski and Pravin Varaiya. On ellipsoidal techniques for reachability analysis. part ii: Internal approximations box-valued constraints. *Optimization methods and software*, 17(2):207–237, 2002. ISSN 1055-6788.

[20] Jingqi Li, Donggun Lee, Jaewon Lee, Kris Shengjun Dong, Somayeh Sojoudi, and Claire Tomlin. Certifiable deep learning for reachability using a new lipschitz continuous value function, 2025. URL https://arxiv.org/abs/2408.07866.

[21] Zongyi Li, Hongkai Zheng, Nikola Borislavov Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Andrew Stuart, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations, 2022. URL https://openreview.net/forum?id=dtYnHcmQKeM.

[22] Albert Lin and Somil Bansal. Verification of neural reachable tubes via scenario optimization and conformal prediction. In *6th Annual Learning for Dynamics & Control Conference*, pages 719–731. PMLR, 2024.

[23] John Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.

[24] John N Maidens, Shahab Kaynama, Ian M Mitchell, Meeko M K Oishi, and Guy A Dumont. Lagrangian methods for approximating the viability kernel in high-dimensional systems. *Automatica*, 49(7):2017–2029, 2013. ISSN 0005-1098.

[25] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *International Conference on Robotics and Automation (ICRA)*, pages 4054–4061. IEEE, 2013. ISBN 1467356433.

[26] K. Margellos and J. Lygeros. Hamilton-Jacobi Formulation for Reach–Avoid Differential Games. *IEEE Transactions on Automatic Control*, 56(8):1849–1861, 2011.

[27] Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

[28] I. Mitchell. A toolbox of level set methods. *http://www.cs. ubc. ca/mitchell/ToolboxLS/toolboxLS.pdf*, 2004.

[29] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[30] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.

[31] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[32] William Sharpless, Zeyuan Feng, Somil Bansal, and Sylvia Herbert. Linear supervision for nonlinear, high-dimensional neural control and differential games. *arXiv preprint arXiv:2412.02033*, 2024.

[33] William Sharpless, Zeyuan Feng, Somil Bansal, and Sylvia Herbert. Linear supervision for nonlinear, high-dimensional neural control and differential games, 2024. URL https://arxiv.org/abs/2412.02033.

[34] Aditya Singh, Zeyuan Feng, and Somil Bansal. Imposing exact safety specifications in neural reachable tubes. *arXiv preprint arXiv:2404.00814*, 2024.

## VII. Appendix

### A. Quadrotor

*1) Dynamics:* The quadrotor dynamics is given as follows:

$$\begin{aligned}
\dot{p_x} &= \nu_x, \\
\dot{p_y} &= v_y, \\
\dot{p_z} &= v_z, \\
\dot{q}_\omega &= -(\omega_x \cdot q_x)/2 - (\omega_y \cdot q_y)/2 - (\omega_z \cdot q_z)/2, \\
\dot{q}_x &= (\omega_x \cdot q_w)/2 + (\omega_z \cdot q_y)/2 - (\omega_y \cdot q_z)/2, \\
\dot{q}_y &= (\omega_y \cdot q_w)/2 - (\omega_z \cdot q_x)/2 + (\omega_x \cdot q_z)/2, \\
\dot{q}_z &= (\omega_z \cdot q_w)/2 + (\omega_y \cdot q_x)/2 - (\omega_x \cdot q_y)/2, \\
\dot{v}_x &= CT \cdot (2 \cdot q_w \cdot q_y + 2 \cdot q_x \cdot q_z)F/m, \\
\dot{v}_y &= CT \cdot (-2 \cdot q_w \cdot q_x + 2 \cdot q_y \cdot q_z)F/m, \\
\dot{v}_z &= Gz - CT \cdot (2 \cdot q_x^2 + 2 \cdot q_y^2 - 1)F/m, \\
\dot{\omega}_x &= \alpha_x - \frac{5}{9}\omega_y \cdot \omega_z, \\
\dot{\omega}_y &= \alpha_y + \frac{5}{9}\omega_x \cdot \omega_z, \\
\dot{\omega}_z &= \alpha_z,
\end{aligned} \tag{17}$$

where $CT = 1$ is the lifting coefficient and $m = 1\ kg$ is the mass. $(p_x, p_y, p_z)$ denotes the position and $(v_x, v_y, v_z)$ denotes the linear velocities. $(q_\omega, q_x, q_y, q_z) \in [-1, 1]^4$ represents the quaternion and $(\omega_x, \omega_y, \omega_z)$ are the angular velocities.

*2) Boundary condition:* The safety function $l(x)$ for the quadrotor case study is computed as:

$$\begin{aligned}
\mathbf{v}^n &= \mathbf{q} \cdot \mathbf{e}_3 \cdot \bar{\mathbf{q}}, \\
d_x &= \frac{r_a^2 p_x^2 \nu_z^2}{p_x^2\nu_x^2 + p_x^2\nu_z^2 + 2p_x p_y \nu_x \nu_y + p_y^2\nu_y^2 + p_y^2\nu_z^2}, \\
d_y &= \frac{r_a^2 p_y^2 \nu_z^2}{p_x^2\nu_x^2 + p_x^2\nu_z^2 + 2p_x p_y \nu_x \nu_y + p_y^2\nu_y^2 + p_y^2\nu_z^2}, \\
l(x) &= max(\sqrt{x^2 + y^2} - \sqrt{d_x + d_y}, 0) - r_o,
\end{aligned}$$

where $\mathbf{q} = q_\omega + q_x \cdot i + q_y \cdot j + q_z \cdot k$, $\bar{\mathbf{q}} = q_\omega - q_x \cdot i - q_y \cdot j - q_z \cdot k$, $\mathbf{e}_3 = [0, 0, 1]$, $r_a = 0.17\ m$ is the radius of the quadrotor, and $r_o = 0.5\ m$ denotes the radius of the obstacle. Essentially, $l(x)$ represents the signed distance function from a disk (i.e., the quadrotor) to the cylindrical obstacle centered at the origin with an infinite length along the z-axis.

### B. F1Tenth

The F1Tenth has a hybrid dynamics with a kinematic mode and a dynamic mode. It's in kinematic mode if the magnitude of speed is less than 0.5. Otherwise, the dynamic mode will be employed. We now present the dynamics equation for each mode.

*1) Kinematic Model Dynamics ($|v| < 0.5$):* For low velocities ($|v| < 0.5$), the kinematic model is used.

The kinematic-mode dynamics are:

$$\mathbf{f} = \begin{bmatrix} v\cos(\theta_{yaw}) \\ v\sin(\theta_{yaw}) \\ \dot{\phi} \\ a \\ \frac{v}{l_r+l_f}\tan(\phi) \\ \frac{a}{l_r+l_f}\tan(\phi) + \frac{v}{(l_r+l_f)\cos^2(\phi)}\dot{\phi} \\ 0 \end{bmatrix}.$$

*2) Dynamic Model Dynamics ($|v| \geq 0.5$):* For higher velocities ($|v| \geq 0.5$), the dynamic model is used.

The dynamic-mode dynamics are:

$$\mathbf{f} = \begin{bmatrix} v\cos(\theta_{yaw} + \beta_{slip}) \\ v\sin(\theta_{yaw} + \beta_{slip}) \\ \dot{\phi} \\ a \\ \omega_{yaw} \\ -\frac{\mu m}{vI(l_r+l_f)}\left(l_f^2 C_{Sf}(gl_r - ah) + l_r^2 C_{Sr}(gl_f + ah)\right)\omega_{yaw} \\ +\frac{\mu m}{I(l_r+l_f)}\left(l_r C_{Sr}(gl_f + ah) - l_f C_{Sf}(gl_r - ah)\right)\beta_{slip} \\ +\frac{\mu m}{I(l_r+l_f)} l_f C_{Sf}(gl_r - ah)\phi, \\ \left(\frac{\mu}{v^2(l_r+l_f)}\left(C_{Sr}(gl_f + ah)l_r - C_{Sf}(gl_r - ah)l_f\right) - 1\right)\omega_{yaw} \\ -\frac{\mu}{v(l_r+l_f)}\left(C_{Sr}(gl_f + ah) + C_{Sf}(gl_r - ah)\right)\beta_{slip} \\ +\frac{\mu}{v(l_r+l_f)} C_{Sf}(gl_r - ah)\phi \end{bmatrix}.$$

The vehicle parameters are listed below:

- $\mu$: Surface friction coefficient: $1.0489$
- $C_{Sf}$: Cornering stiffness coefficient, front: $4.718/\,\text{rad}$
- $C_{Sr}$: Cornering stiffness coefficient, rear: $5.4562/\,\text{rad}$
- $l_f$: Distance from center of gravity to front axle: $0.15875\,\text{m}$
- $l_r$: Distance from center of gravity to rear axle: $0.17145\,\text{m}$
- $h$: Height of center of gravity: $0.074\,\text{m}$
- $m$: Total mass of the vehicle: $3.74\,\text{kg}$
- $I$: Moment of inertia of the entire vehicle about the $z$-axis: $0.04712\,\text{kg}\cdot\text{m}^2$