Data-Efficient RoA Verification of Black-box Controllers via Flow Matching and Conformal Inference

Anonymous submission

***Abstract*— Verification of robot controllers is important for ensuring safe and reliable robot operation. It can involve the characterization of a controller's Region of Attraction (RoA), i.e., the initial states that lead to a successful outcome. This is challenging, however, for black-box, learned controllers where analytical models are unavailable. In such scenarios, verification is conducted using only trajectory data, making data efficiency critical as each data point represents a costly physical experiment. To address this, this paper presents a framework that integrates three components: (1) a Conditional Flow Matching (CFM) model that learns the distribution of final states from initial conditions, capturing multi-modal outcome distributions that naturally express uncertainty near the boundary between success and failure regions; (2) a probabilistic classification mechanism that interprets the learned distribution via optimized decision thresholds to categorize states into the RoA, the failure region, or an uncertain region where the model cannot commit to an outcome; and (3) an adaptive sampling strategy that concentrates trajectory collection on the uncertain region to reduce it with fewer rollouts than uniform sampling. The resulting predictions are certified via conformal inference to provide finite-sample coverage guarantees. Experiments on benchmark systems ranging from a 2D pendulum to a 13D quadrotor demonstrate that the proposed framework achieves high verification accuracy with low uncertainty compared to existing methods, and that adaptive sampling further improves data efficiency over the non-adaptive variant.**
Website: https://tinyurl.com/adaptive-roa-via-fm

I. Introduction

Problem Domain: Verification of robot controllers is fundamental for ensuring safety and reliability in robotics [1]. A central challenge in this context is characterizing the *Region of Attraction* (RoA): the set of initial states from which a controller achieves success, such as converging to a desired goal within a finite time or reaching an attractor of the underlying dynamical system. An effective verifier should also characterize the RoA for failure and operate effectively close to the boundary between the two corresponding regions, where the switch from success to failure occurs.

Challenges: Traditionally, RoA characterization relies on accurate analytical models. Even for classical control, however, obtaining high-fidelity models is often difficult due to numerical approximations, parameter uncertainty, and unmodeled dynamics [3], [4]. This challenge is further exacerbated by the shift towards learned control, where policies are black-box in nature and defy analytical characterization [5]. Consequently, traditional tools, such as Lyapunov-based certificates or Sum-of-Squares optimization, are not directly applicable, as they require the dynamics expressed in an analytical form. So, learned controllers necessitate a shift toward *data-driven verification*, where the RoA must be characterized given only robot trajectory rollouts. Data-driven verification, however, introduces its own challenges:

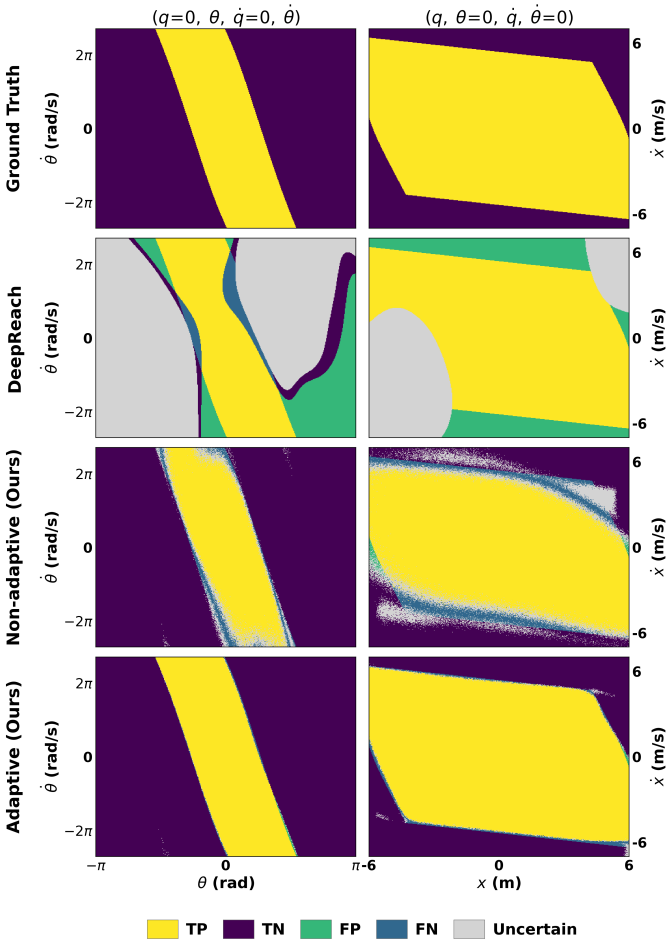

Fig. 1: RoA identification on two 2D slices of a 4D Cartpole system ($(\theta, \dot{\theta})$ on the left and $(x, \dot{x})$ on the right; non-visualized DoFs are set to zero). Top to bottom: Ground truth RoAs, DeepReach [2] (modified) baseline, proposed non-adaptive and adaptive variants. Yellow: True Positives (correctly identified success RoA states). Purple: True Negatives. Green: False Positives. Teal: False Negatives (missed success RoA states). Grey: uncertain region (model abstentions). The adaptive method closely recovers the ground-truth. The non-adaptive variant has increased uncertainty. The DeepReach baseline abstains on 16.7% of the state space and misclassifies a substantial portion of the true RoAs.

A. Near the boundary between success and failure, outcomes are multi-modal, i.e., nearby initial conditions lead to qualitatively different final states. A deterministic learned model cannot effectively express this ambiguity motivating the use

of models that effectively express multi-modal distributions.
B. Often data-driven control verification methods, e.g., DeepReach (Hamilton-Jacobi reachability) [2] and Neural Lyapunov functions [6], represent the RoA as a sublevel set of a learned scalar function. These representations, however, do not quantify the confidence of the prediction, motivating approaches that can express such confidence.
C. Generating trajectory data is the primary bottleneck in robotics, requiring costly physical experiments or high-fidelity simulations. Most of the state space, however, is homogeneous, either clearly within the success or failure RoA. The boundary occupies a thin manifold that is underrepresented under uniform sampling. This motivates informed strategies for identifying initial conditions to collect additional trajectory data.
D. For reliable and safe deployment, empirical accuracy alone is insufficient. Without calibration, a model's expressed confidence carries no formal guarantee of correctness, and its abstentions lack a principled statistical basis.

Requirements: The verifier must be *generative*: a model that learns *multi-modal distributions over final states* can naturally capture multiple possible outcomes near the boundary. It also decouples dynamics from fixed success criteria. In addition, predictions must be evaluated in terms of *confidence* and be *certifiable*: the framework must express its uncertainty, provide finite-sample coverage guarantees and enable principled abstention when uncertainty is high. Finally, data collection must be *adaptive* to uncertainty: sampling must be prioritized along states where the model cannot commit to an outcome, i.e., states in the *uncertain region*, improving accuracy relative to uninformed sampling.

Proposed Contributions: This paper proposes a generative model to predicts final states for a robot given a controller, evaluates the confidence of these estimates, adaptively acquires trajectory data given model uncertainty, and certifies predictions with statistical guarantees:

1. Generative Final-State Prediction: A Conditional Flow Matching (CFM) model [7] learns a multi-modal distribution of final states from raw trajectory rollouts of a robot under the influence of a controller to be verified.

2. Probabilistic Classification with Optimized Decision Thresholds. Monte Carlo samples of the model predictions are evaluated against user-specified success criteria to estimate outcome probabilities in terms of success, failure, or uncertain. A threshold-based decision rule classifies each state accordingly. Decision boundaries separating the different classifications are optimized to improve accuracy.

3. Uncertainty-Guided Adaptive Sampling: The model's uncertainty identifies states it cannot reliably classify, forming an "uncertain region". The framework preferentially collects data from this region, progressively refining the boundary characterization between success and failure RoA.

4. Inference with Conformal Guarantees: A split conformal prediction layer provides provable, finite-sample guarantees that the true outcome is contained within a calibrated prediction set at a user-specified confidence level, with non-singleton prediction sets serving as a certificate of abstention.

The framework is evaluated on a 2D pendulum, 4D cart-pole and 13D spatial quadrotor under LQR control, as well as a 6D planar quadrotor under a learned policy. Six baselines are considered spanning value-function, topological, and direct-prediction approaches. Results demonstrate higher F1 scores and a smaller uncertain region for the proposed solution, while maintaining target conformal coverage.

II. Related Work

Verification of Control Systems: Hamilton-Jacobi (HJ) reachability computes backward reachable sets [8], [9], but these grid-based solvers scale poorly with dimension. Neural approximations, such as DeepReach [2], extend to higher-dim. systems. Lyapunov stability theory certifies convergence via energy-like functions satisfying $\dot{V}(x) < 0$ [10]. Classical Lyapunov and Sum-of-Squares methods require the dynamics or the certificate to admit a functional form, but neural Lyapunov methods [6], [11], [5] remove this restriction. Topological methods decompose state spaces into Morse sets that capture qualitative dynamics without explicit certificates [12]. The MORALS framework [13] extends this to higher-dim. systems via a learned latent embedding. These topological methods assume the system stabilizes in the bounded domain, which is violated when failure trajectories diverge rather than converging to a failure attractor.

Generative Models for State Distribution Learning: An emerging paradigm uses generative models as predictive surrogates that learn the distribution of future states a system will visit. Training a generative model on discounted state occupancy measures [14] produces a reward-independent representation that can be queried under different success criteria without retraining. Subsequent work uses diffusion models to learn successor state measures with Bellman flow constraints [15] and policy-agnostic occupancy distributions from offline data [16]. Generative surrogates have also been applied directly to verification: conditional GANs that map system states to sensor observations enable formal reachability analysis of image-based neural network controllers [17]. Diffusion-based pipelines have been combined with control barrier functions for safe trajectory planning [18].

Conformal Prediction in Robotics: Conformal prediction (CP) [19], [20] provides a model-agnostic framework for uncertainty quantification with finite-sample statistical guarantees. Applications include calibrating forward reachable tubes [21], safe motion planning [22], and runtime monitoring of temporal logic specifications [23]. GenQPM [24] pairs diffusion models with conformalized quantile regression for runtime safety monitoring over a finite future horizon.

Adaptive Sampling and Active Learning: Adaptive sampling for RoA estimation has relied on Gaussian processes (GPs) [25], [26] or boundary-focused heuristics, such as Hybrid Active Learning [27]. These methods improve data efficiency but rely on kernel assumptions or heuristic confidence measures that lack distribution-free guarantees. Some recent works also explore combining conformal prediction with active learning: CoPAL [28] and ConformalDAgger [29] do so in regression settings unrelated to control verification.

III. Problem Formulation

Let $\mathcal{X}$ denote an n-dimensional smooth manifold embedded in $\mathbb{R}^d$ representing the state space of a robotic system controlled by a fixed policy. The closed-loop dynamics are deterministic, defining a flow map $\Phi : \mathcal{X} \to \mathcal{X}$ that maps each initial state x_0 to its terminal state $x_T = \Phi(x_0)$, where the trajectory runs until a maximum horizon T or until the system achieves success or failure, whichever comes first. Neither the system dynamics nor the controller structure are assumed known. The only available information comes from rollouts: executing the controller from a chosen initial state $x_0 \in \mathcal{X}$ and recording the resulting trajectory $\tau = (x_0, x_1, \ldots, x_{T'})$, where $T' \leq T$ is the termination time. For notational simplicity, the termination time is denoted T throughout.

Two task-specific criteria classify the outcome of a trajectory based on its terminal state: a success criterion $\gamma_{\mathrm{s}} : \mathcal{X} \to \{0, 1\}$, e.g., reaching a goal configuration, and a failure criterion $\gamma_{\mathrm{f}} : \mathcal{X} \to \{0, 1\}$, such as leaving the admissible state bounds. The time horizon T is assumed large enough that the trajectory reaches a terminal state almost everywhere (a.e.) satisfying either γ_{s} or γ_{f} before timeout. Therefore, each trajectory a.e. terminates in a single outcome. This induces a.e. a ground-truth label for each initial state:

$$y(x_0) = \begin{cases} \texttt{<Succ>} & \text{if } \gamma_{\mathrm{s}}(\Phi(x_0)) = 1, \\ \texttt{<Fail>} & \text{if } \gamma_{\mathrm{f}}(\Phi(x_0)) = 1, \end{cases} \tag{1}$$

partitioning $\mathcal{X}$ into the *region of attraction* $\texttt{RoA}_{\mathrm{s}} = \{x_0 \in \mathcal{X} : y(x_0) = \texttt{<Succ>}\}$, from which the controller succeeds, and the failure region $\texttt{RoA}_{\mathrm{f}} = \mathrm{int}(\mathcal{X} \backslash \texttt{RoA}_{\mathrm{s}})$, i.e., the $\mathcal{X} \backslash \texttt{RoA}_{\mathrm{s}}$ minus the boundary between $\texttt{RoA}_{\mathrm{s}}$ and $\texttt{RoA}_{\mathrm{f}}$, this boundary is known as the *true separatrix*.

Problem: RoA Estimation: Given a dataset of rollouts $\mathcal{D} = \{\tau_i\}_{i=1}^N$, determine whether the controller will succeed or fail from a given initial state x_0, i.e., estimate the regions $\texttt{RoA}_{\mathrm{s}}$ and $\texttt{RoA}_{\mathrm{f}}$. Since predictions inform deployment decisions, they must come with a known confidence: specifically, the probability of a correct prediction must be at least $(1-\alpha)$ for a user-specified $\alpha \in (0, 1)$.

IV. Proposed Method

The outcome of the proposed framework is a probabilistic classifier for RoA estimation shown in Fig. 2. The training for the corresponding classifier is illustrated in Fig. 3.

A. Generative Final-State Prediction

The proposed method learns a conditional generative model $p_\theta(\cdot \mid x)$ that maps any state x to a distribution over terminal states, instead of a direct classifier $x \mapsto \hat{y}$. This generative formulation naturally captures multi-modal outcome distributions near the boundary, where a single deterministic prediction may fail.

Training data: From the trajectory dataset $\mathcal{D}_{\mathrm{traj}}$, trajectories are split into a training set $\mathcal{D}_{\mathrm{train}}$ and a validation set $\mathcal{D}_{\mathrm{val}}$. Training pairs are constructed by pairing every non-terminal state x_t along a trajectory (i.e., $t = 0, \ldots, T-1$) with its terminal state x_T. Since the closed-loop dynamics are deterministic, any state along a trajectory uniquely determines the same terminal outcome, so each trajectory of length T contributes T valid supervision pairs. The resulting collections $\mathcal{D}_{\mathrm{pairs}}^{\mathrm{train}} = \{(x_t, x_T)\}$ and $\mathcal{D}_{\mathrm{pairs}}^{\mathrm{val}} = \{(x_t, x_T, y)\}$ are constructed from $\mathcal{D}_{\mathrm{train}}$ and $\mathcal{D}_{\mathrm{val}}$ respectively, where $\mathcal{D}_{\mathrm{pairs}}^{\mathrm{val}}$ additionally carries ground-truth labels y for threshold optimization. Crucially, $\mathcal{D}_{\mathrm{pairs}}^{\mathrm{train}}$ requires no success or failure labels at training time, decoupling the learned dynamics from any fixed success criterion.

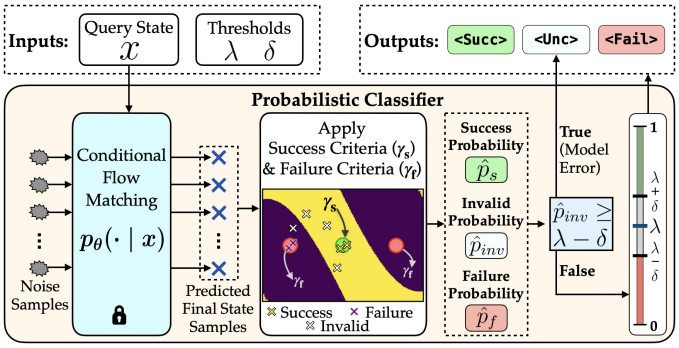

Fig. 2: Probabilistic Classifier via Conditional Flow Matching.

Conditional flow matching: The conditional distribution $p_\theta(\cdot \mid x)$ is realized via conditional flow matching (CFM) [7], using the rectified flow parameterization [30]. A neural vector field v_θ parameterizes a time-dependent velocity field that learns to transport samples from a standard Gaussian p_0 to the terminal-state distribution. The conditioning state x enters as an additional input , so v_θ produces a different flow for each x. Concretely, for a training pair $(x_t, x_T) \in \mathcal{D}_{\mathrm{pairs}}^{\mathrm{train}}$, the network receives x_t as the conditioning input and learns to transport noise samples toward the corresponding terminal state x_T. Pairs (x_t, x_T) are sampled from $\mathcal{D}_{\mathrm{pairs}}^{\mathrm{train}}$, noise samples z are drawn from p_0, and flow times s are drawn uniformly from $[0, 1]$. The network is trained to match the straight-line velocity between z and x_T at the corresponding flow time, with loss: $\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}\big\|v_\theta(z_s,\, s,\, x_t) - (x_T - z)\big\|^2$, where $z_s = (1-s)\, z + s\, x_T$ is the linear interpolation between z and x_T at flow time $s \in [0, 1]$.

B. Probabilistic Classification

The generative model is converted into a probabilistic classifier (Fig.2) by estimating outcome probabilities via Monte Carlo sampling and a threshold-based decision rule.

Monte Carlo probability estimation: For a state x, the model draws K final states $\hat{x}_T^{(1)}, \ldots, \hat{x}_T^{(K)} \sim p_\theta(\cdot \mid x)$. Each sample is evaluated against both criteria: $\hat{p}_{\mathrm{s}}(x) = \frac{1}{K}\sum_{k=1}^K \gamma_{\mathrm{s}}\big(\hat{x}_T^{(k)}\big)$, $\hat{p}_{\mathrm{f}}(x) = \frac{1}{K}\sum_{k=1}^K \gamma_{\mathrm{f}}\big(\hat{x}_T^{(k)}\big)$. For ground-truth trajectories with complementary criteria, γ_{s} and γ_{f} partition the terminal state space, so $\hat{p}_{\mathrm{s}} + \hat{p}_{\mathrm{f}}$ would equal one. In general, however, the generative model may produce invalid samples that satisfy neither criterion. The remainder $\hat{p}_{\mathrm{inv}}(x) = 1 - \hat{p}_{\mathrm{s}}(x) - \hat{p}_{\mathrm{f}}(x)$ quantifies the model error, reflecting regions where the model has not learned well.

Threshold-based decision rule: A dual-threshold rule converts probability estimates to predictions $\hat{y}(x) \in$

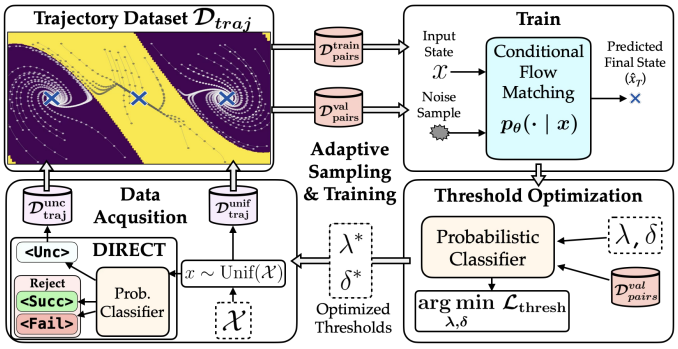

Fig. 3: Adaptive Sampling and Training of the Prob. Classifier.

$\{$<Succ>, <Fail>, <Unc>$\}$, with decision boundaries $\theta_{\text{s}} = \lambda + \delta$ and $\theta_{\text{f}} = 1 - (\lambda - \delta)$, where $\lambda \in (0, 1)$ is the decision center and $\delta \geq 0$ controls the width of the uncertain region. The rule applies in three stages: first, if $\hat{p}_{\text{inv}}(x) \geq \lambda - \delta$, the state is flagged as <Unc> due to *model error*, as too many invalid endpoints indicate the model has not learned this region. Otherwise, the rule predicts <Succ> if $\hat{p}_{\text{s}}(x) \geq \theta_{\text{s}}$, <Fail> if $\hat{p}_{\text{f}}(x) \geq \theta_{\text{f}}$, and <Unc> due to *outcome uncertainty* if neither threshold is satisfied, as the model generates valid but mixed outcomes and cannot confidently resolve the outcome at that state. Both sources of $\hat{y} =$ <Unc> form the uncertain region, but $\hat{p}_{\text{inv}}$ distinguishes them. The uncertain region need not coincide with the true boundary; it may also include states where data coverage is insufficient. As the model improves, the uncertain region is expected to contract toward the true boundary.

Threshold optimization: Since the optimal trade-off depends on the data distribution, thresholds (λ, δ) are optimized rather than fixed by minimizing $\mathcal{L}_{\text{thresh}}$ over $\mathcal{D}^{\text{val}}_{\text{pairs}}$: $(\lambda^*, \delta^*) = \arg\min_{\lambda', \delta'} \mathcal{L}_{\text{thresh}}$, where $\mathcal{L}_{\text{thresh}} = w \cdot R_{\text{err}}(\lambda', \delta') + (1 - w) \cdot R_{\text{unc}}(\lambda', \delta')$ R_{err} is the fraction of confident predictions ($\hat{y} \neq$ <Unc>) that disagree with the ground-truth label and R_{unc} is the fraction of all states that fall in the uncertain region ($\hat{y} =$ <Unc>). $w \in [0, 1]$ controls the trade-off between them. Both terms are computed only over states where $\hat{p}_{\text{inv}}(x) < \lambda - \delta$, excluding model error states. Those states are flagged as <Unc> upstream regardless of the threshold values. Larger w favors larger gaps between decision boundaries, classifying more states as uncertain, but with fewer misclassifications; smaller w tightens the gap, reducing the uncertain region rate at the cost of tolerating more errors. The optimized boundaries $\theta^*_{\text{s}} = \lambda^* + \delta^*$ and $\theta^*_{\text{f}} = 1 - (\lambda^* - \delta^*)$ are then used for probabilistic classification.

C. Adaptive Sampling and Training

Since trajectories are expensive to collect and evaluating $p_\theta(\cdot \mid x)$ is orders of magnitude cheaper than a rollout, candidate initial states can be screened before committing to data acquisition. The classifier (Sec. IV-B) provides an acquisition signal: states receiving $\hat{y}(x) =$ <Unc> are where the model lacks confidence, and where additional rollouts yield the greatest uncertainty reduction. This enables targeted acquisition that concentrates rollouts on the uncertain region rather than spreading them uniformly.

Acquisition strategy: DIRECT. Each acquisition round collects N new trajectories, governed by a *sampling ratio* $r_{\text{unc}} \in [0, 1]$ that controls the exploration-exploitation split. $\lfloor N \cdot (1 - r_{\text{unc}}) \rfloor$ initial states are drawn uniformly from $\mathcal{X}$, providing broad exploration of the state space. The remaining $\lceil N \cdot r_{\text{unc}} \rceil$ trajectories target the uncertain region via rejection sampling: candidates are drawn uniformly from $\mathcal{X}$, screened by probabilistic classification, and retained only if $\hat{y}(x) =$ <Unc>; drawing and screening continues until $\lceil N \cdot r_{\text{unc}} \rceil$ candidates are accepted.

Iterative procedure: Starting from $\mathcal{D}^{\text{init}}_{\text{traj}}$, a set of trajectories collected uniformly from $\mathcal{X}$, the pipeline applies DIRECT over E iterations (Algorithm 1). At each iteration the generative model is retrained on the accumulated $\mathcal{D}^{\text{train}}_{\text{pairs}}$, thresholds (λ^*, δ^*) are re-optimized on $\mathcal{D}^{\text{val}}_{\text{pairs}}$, and N new trajectories are acquired via DIRECT, split into train and validation subsets, and appended to $\mathcal{D}^{\text{train}}_{\text{pairs}}$ and $\mathcal{D}^{\text{val}}_{\text{pairs}}$ respectively. The total trajectory budget is $N_{\text{total}} = |\mathcal{D}_{\text{init}}| + E \cdot N$.

Algorithm 1: Adaptive RoA Estimation (Training)

Input: Initial trajectories $\mathcal{D}^{\text{init}}_{\text{traj}}$, state space $\mathcal{X}$, rollout budget N, sampling ratio r_{unc}, trade-off weight w, iterations E
Output: Trained model p_θ, optimized thresholds (λ^*, δ^*)

Initialize $\mathcal{D}^{\text{train}}_{\text{pairs}}$, $\mathcal{D}^{\text{val}}_{\text{pairs}}$ from $\mathcal{D}^{\text{init}}_{\text{traj}}$;
for $e = 1$ **to** E **do**
  Train p_θ via $\mathcal{L}_{\text{CFM}}$ on $\mathcal{D}^{\text{train}}_{\text{pairs}}$;
  Optimize (λ^*, δ^*) on $\mathcal{D}^{\text{val}}_{\text{pairs}}$;
  $\mathcal{D}^{\text{unif}}_{\text{traj}} \leftarrow$ roll out $\lfloor N(1 - r_{\text{unc}}) \rfloor$ states drawn uniformly from $\mathcal{X}$;
  $\mathcal{D}^{\text{unc}}_{\text{traj}} \leftarrow$ roll out $\lceil N \cdot r_{\text{unc}} \rceil$ states with $\hat{y}(x) =$ <Unc>;
  Split $(\mathcal{D}^{\text{unif}}_{\text{traj}} \cup \mathcal{D}^{\text{unc}}_{\text{traj}})$ into $\mathcal{D}^{\text{train}}_{\text{new}}$ and $\mathcal{D}^{\text{val}}_{\text{new}}$;
  $\mathcal{D}^{\text{train}}_{\text{pairs}} \leftarrow \mathcal{D}^{\text{train}}_{\text{pairs}} \cup \{(x_t, x_T)$ from $\mathcal{D}^{\text{train}}_{\text{new}}\}$;
  $\mathcal{D}^{\text{val}}_{\text{pairs}} \leftarrow \mathcal{D}^{\text{val}}_{\text{pairs}} \cup \{(x_t, x_T, y)$ from $\mathcal{D}^{\text{val}}_{\text{new}}\}$;
end
Train p_θ on $\mathcal{D}^{\text{train}}_{\text{pairs}}$;
Optimize (λ^*, δ^*) on $\mathcal{D}^{\text{val}}_{\text{pairs}}$;

D. Inference

After training completes, the pipeline produces a generative model p_θ and optimized thresholds (λ^*, δ^*). To classify a new initial state x, the model draws K terminal-state samples from $p_\theta(\cdot \mid x)$, evaluates each against γ_{s} and γ_{f} to obtain $\hat{p}_{\text{s}}(x)$ and $\hat{p}_{\text{f}}(x)$, and applies the threshold-based decision rule with the optimized boundaries θ^*_{s} and θ^*_{f}. The output is a point prediction $\hat{y}(x) \in \{$<Succ>, <Fail>, <Unc>$\}$: the controller is expected to succeed, fail, or the state lies in the uncertain region.

V. PROBABILISTIC GUARANTEES

The method of Sec. IV produces a trained model p_θ and optimized thresholds (λ^*, δ^*). These predictions are augmented with coverage guarantees via conformal prediction [19], producing prediction sets that contain the true label with probability at least $1 - \alpha$.

A. Nonconformity Score

Using the optimized decision boundaries $\theta^*_{\text{s}} = \lambda^* + \delta^*$ and θ^*_{f}, a nonconformity score $s(x, y)$ is defined to measure

how inconsistent the model's probability estimates are with each candidate label $y \in \{\texttt{<Succ>}, \texttt{<Fail>}, \texttt{<Unc>}\}$:

$$s(x,y) = \begin{cases} \max\big(0,\ \theta_{\mathrm{s}}^* - \hat{p}_{\mathrm{s}}(x)\big) & y = \texttt{<Succ>}, \\ \max\big(0,\ \theta_{\mathrm{f}}^* - \hat{p}_{\mathrm{f}}(x)\big) & y = \texttt{<Fail>}, \\ \max\big(0,\ \hat{p}_{\mathrm{s}}(x) - \theta_{\mathrm{s}}^*,\ \hat{p}_{\mathrm{f}}(x) - \theta_{\mathrm{f}}^*\big) & y = \texttt{<Unc>}. \end{cases}$$

Intuitively, the score answers: how much would the model's estimates need to change for label y to become the natural prediction? A score of zero means y is already consistent with the model's output. For $y \in \{\texttt{<Succ>}, \texttt{<Fail>}\}$, this holds when the corresponding probability already exceeds its decision boundary. For $y = \texttt{<Unc>}$, this holds when neither probability exceeds its boundary, i.e., the model cannot commit to either outcome.

B. Calibration

A held-out calibration set $\mathcal{D}_{\mathrm{cal}} = \{(x_i, y_i)\}_{i=1}^{n_{\mathrm{cal}}}$ with ground-truth labels $y_i \in \{\texttt{<Succ>}, \texttt{<Fail>}\}$, disjoint from both the training data and the test set, is used to determine the conformal quantile. Because the true labels in $\mathcal{D}_{\mathrm{cal}}$ are always <Succ> or <Fail>, states where the model produces mostly invalid endpoints would contribute large nonconformity scores, since the model assigns low probability to the true label at those states. This would inflate $\hat{q}$ to accommodate modeling artifacts rather than genuine boundary ambiguity. Points with $\hat{p}_{\mathrm{inv}}(x_i) \geq \lambda^* - \delta^*$ (the same model-error filter of Section IV-B) are therefore removed before calibration. Let $\mathcal{D}'_{\mathrm{cal}} \subseteq \mathcal{D}_{\mathrm{cal}}$ denote the filtered set. The conformal quantile is: $\hat{q} = \mathrm{Quantile}\Big(\frac{\lceil (1-\alpha)(|\mathcal{D}'_{\mathrm{cal}}|+1) \rceil}{|\mathcal{D}'_{\mathrm{cal}}|},\ \{s(x_i,y_i)\}_{(x_i,y_i)\in\mathcal{D}'_{\mathrm{cal}}}\Big)$.

C. Prediction Sets

For a test state x, the prediction set collects all labels whose nonconformity score does not exceed the calibrated quantile: $\mathcal{C}(x) = \big\{y \in \{\texttt{<Succ>}, \texttt{<Fail>}, \texttt{<Unc>}\} : s(x,y) \leq \hat{q}\big\}$. A singleton $\{\texttt{<Succ>}\}$ or $\{\texttt{<Fail>}\}$ indicates a confident prediction. Sets containing <Unc> indicate states where neither $\hat{p}_{\mathrm{s}}$ nor $\hat{p}_{\mathrm{f}}$ is decisively large, so the model cannot confidently classify the state as <Succ> or <Fail>. Note that since $\hat{p}_{\mathrm{s}} + \hat{p}_{\mathrm{f}} \leq 1$ but $\theta_{\mathrm{s}}^* + \theta_{\mathrm{f}}^* > 1$, both probabilities cannot simultaneously exceed their respective boundaries, ensuring that any prediction set containing both <Succ> and <Fail> must also contain <Unc>.

D. Coverage Guarantee

Theorem 1 [19]: Suppose the calibration data $\mathcal{D}'_{\mathrm{cal}}$ and the test point $(x_{\mathrm{test}}, y_{\mathrm{test}})$ are exchangeable. Then, the prediction set $\mathcal{C}(x_{\mathrm{test}})$ satisfies: $\Pr\big(y_{\mathrm{test}} \in \mathcal{C}(x_{\mathrm{test}})\big) \geq 1 - \alpha$.

This guarantee applies only to states that are not filtered due to model error when applying the threshold-based decision rule. Such filtered states are classified directly as <Unc> without a coverage claim. Since the ground truth is always $y_{\mathrm{test}} \in \{\texttt{<Succ>}, \texttt{<Fail>}\}$, prediction sets containing <Unc> alongside the true label (e.g., $\{\texttt{<Succ>}, \texttt{<Unc>}\}$ with $y = \texttt{<Succ>}$) still count as coverage hits, while $\mathcal{C}(x) = \{\texttt{<Unc>}\}$ is always a miss. Since the nonconformity scores are computed from finite Monte Carlo estimates of $\hat{p}_{\mathrm{s}}$ and $\hat{p}_{\mathrm{f}}$, the guarantee holds for the implemented randomized predictor, with the Monte Carlo approximation absorbed into the score function.

E. Coverage Guaranteed Classification

A singleton prediction set $\{\texttt{<Succ>}\}$ or $\{\texttt{<Fail>}\}$ is a confident prediction that carries the $(1-\alpha)$ guarantee of Theorem 1. A non-singleton set or a single singleton containing only <Unc> indicates that the model cannot distinguish between its members at the calibrated confidence level, and the state is classified as <Unc>.

VI. Experiments

This section evaluates 7 methods on 4 dynamical systems of varying dimension (Figure 4). All methods operate over the same trajectory data without access to the dynamics or controller structure.

A. Benchmark Systems

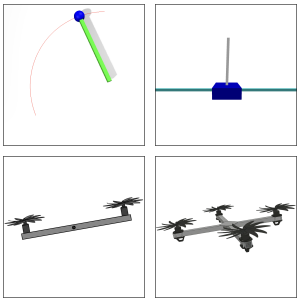

Fig. 4: Benchmark systems. Top: Pendulum (2D), Cartpole (4D). Bottom: Planar Quad (6D), Spatial Quad (13D).

Pendulum (2D): An inverted pendulum with state $(\theta, \dot{\theta})$ under LQR control [12], tasked with stabilizing at the upright equilibrium $(\theta, \dot{\theta}) = (0, 0)$. Success and failure criteria are complementary ($\gamma_{\mathrm{f}} = 1 - \gamma_{\mathrm{s}}$). Evaluation: 50K states on a grid over the bounded state space, with 39% of the states terminating in success.

Cartpole (4D): A cart-pole system with state $(q, \theta, \dot{q}, \dot{\theta})$ under LQR control, tasked with stabilizing at the origin, requiring both pole balance and cart position regulation. Evaluation: 115K states on a grid, with 18% of the states terminating in success.

Planar Quadrotor (6D): A 2D quadrotor with state $(q_x, q_z, \phi, \dot{q}_x, \dot{q}_z, \dot{\phi})$ controlled by a learned RL policy trained via PPO, tasked with reaching a hovering state at a unit height. This system tests the framework on a black-box learned controller whose decision boundary may be irregular compared to analytically designed policies. Evaluation: 500K states on a grid, with 8% of the states terminating in success.

Spatial Quadrotor (13D): A full 3D quadrotor with state $(q, \dot{q}, q_{\mathrm{quat}}, \omega) \in \mathbb{R}^{13}$ under LQR control, tasked with reaching hover at unit height with identity orientation (quaternion $[1, 0, 0, 0]$). The 13-dimensional state space tests scalability and data efficiency. Because grid-based evaluation is infeasible at this dimensionality, the evaluation set of 1M states was constructed by discretizing the state space into a grid, identifying occupied cells from a held-out set of trajectories, and sampling equally from each occupied cell. 22% of the states terminate in success.

The cartpole and both quadrotor environments, including their controllers, are implemented using

Method	Pendulum (2D)		Cartpole (4D)		Planar Quad (6D)		Spatial Quad (13D)	
	F1↑	Unc.↓	F1↑	Unc.↓	F1↑	Unc.↓	F1↑	Unc.↓
DeepReach	.982	3.6%	.901	16.7%	.353	0%	.701	42.8%
Lyapunov NN	.851	0%	.978	71.9%	.571	97.5%	.921	32.1%
MORALS	.905	24.4%	.699	17.9%	.292	32.4%	.194	81.8%
Determ. Final State	.768	0.6%	.346	1.0%	.067	1.0%	.029	1.0%
Classification	.974	0.2%	.946	0.2%	.777	0.5%	.763	0.6%
Non-adaptive (Ours)	**.986**	4.2%	.984	7.6%	.855	13.2%	.947	25.3%
Adaptive (Ours)	.974	6.0%	**.990**	1.0%	**.933**	6.0%	**.953**	25.2%

TABLE I. Comparison across the four dynamical systems. The Adaptive variant uses $r_{\text{unc}} = 1.0$.

`safe-control-gym` [31]. For these three systems, failure is defined as exceeding the state space bounds.

B. Methods Compared

All methods receive trajectory data $\mathcal{D}_{\text{traj}} = \{\tau_i\}_{i=1}^{N_{\text{total}}}$ without access to an analytical model. Each method predicts a label or classifies the state as `<Unc>` for a query state x. All methods are evaluated on point predictions $\hat{y} \in \{\texttt{<Succ>}, \texttt{<Fail>}, \texttt{<Unc>}\}$ without conformal calibration; coverage guarantees are validated separately in Sec. VI-F.

Structured baselines: These methods learn an intermediate representation with control-theoretic or topological meaning. *DeepReach* [2]: approximates the Hamilton-Jacobi value function via a neural PDE solver, adapted to the model-free setting by estimating the Hamiltonian from trajectory data using finite differences. *Neural Lyapunov Function* [6]: learns a Lyapunov function $V(x)$ from trajectory data with a discrete decrease condition enforced at sampled transitions, implemented using `NeuroMANCER` [32]. For both DeepReach and Neural Lyapunov, a threshold band $[v_{\text{low}}, v_{\text{high}}]$ on the learned value function is calibrated post-training on validation data to maximize classification quality. *MORALS* [13]: learns a dynamics-aware latent representation via an autoencoder and constructs a Morse graph over the latent space to classify states into attracting or uncertain regions.

Direct prediction baselines: These methods map states directly to outcomes without learning an intermediate dynamical structure. *Deterministic Final-State:* predicts a single final state $\hat{x}_T$ via a deterministic regression network, with no distributional output. The predicted final state is evaluated against the success and failure criteria to produce a label; predictions satisfying neither criterion are classified as `<Unc>`. *Classification:* a feedforward network that estimates the probability of `<Succ>` for a query state. A threshold band is applied around the decision boundary for final classification; states falling within the band are classified as `<Unc>`.

Proposed variants: *Non-adaptive*: the generative probabilistic classifier of Sec. IV with uniformly sampled trajectories. *Adaptive*: identical, but new trajectories are concentrated on the separatrix via adaptive sampling and training of Sec. IV-C. Comparing the two isolates the contribution of adaptive sampling. Both variants use $K = 20$ Monte Carlo samples and $w = 0.9$ for threshold optimization, and are evaluated using point predictions $\hat{y}(x_0)$ from the threshold decision rule only (no conformal inference).

C. Metrics

All metrics are computed on a held-out evaluation set $\mathcal{D}_{\text{test}}$ with known ground-truth labels. The reported metrics are: *uncertain region rate*, the fraction of states classified as $\hat{y} =$ `<Unc>` (abstentions); and *F1 score*, the harmonic mean of precision and recall for the success class, computed only over confident predictions ($\hat{y} \neq$ `<Unc>`). The two must be interpreted jointly: high F1 is meaningful only at moderate uncertain region rates.

D. Results

Table I compares 7 methods across 4 systems using $N_{\text{total}} \in \{500, 1{,}000, 12{,}000, 25{,}000\}$ training trajectories respectively. The proposed method achieves the highest F1 on all four benchmarks, with the non-adaptive variant leading on the pendulum and the adaptive variant on the remaining three.

Baselines: DeepReach exhibits two failure modes: on the planar quadrotor it commits on every state but achieves only .35 F1 (overconfident misclassification), while on the spatial quadrotor it abstains on 42.8% of states yet still reaches only .70 F1. Neural Lyapunov commits on every state on the pendulum yet achieves only .85 F1, indicating direct misclassification errors rather than over-conservatism. On higher-dimensional systems it compensates through aggressive abstention, reaching 71.9% uncertain region rate on the cartpole and 97.5% on the planar quadrotor, where the reported .57 F1 reflects only 2.5% of states. MORALS degrades on both axes simultaneously: F1 falls from .91 to .19 while uncertain region rates rise from 24.4% to 81.8% across the four systems, consistent with its assumption that the failure trajectories need to remain in the bounded state space, which is violated in all systems except the pendulum. The classification baseline is the most competitive direct method (.76 to .97 F1 at near-zero uncertain region rates), but the gap to the proposed approach widens on higher-dimensional systems. On the planar quadrotor (.78 vs. .93), the gap is consistent with the irregular `RoA` boundary produced by the learned RL controller. On the spatial quadrotor (.76 vs. .95), the gap suggests a scaling advantage of generative modeling

in high-dimensional state spaces.

Adaptive sampling and data efficiency: Both variants

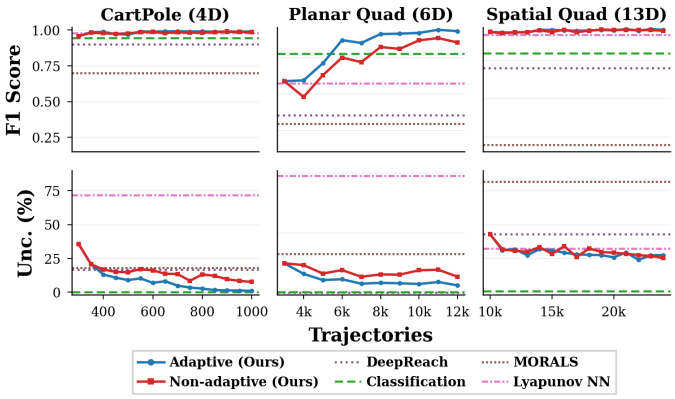

Fig. 5: F1 score (top) and uncertain region rate (bottom) as a function of trajectory budget for the variants across 3 systems. Horizontal lines show baseline performance at full data budget.

reduce the uncertain region rate and improve F1 with increasing trajectory budget, surpassing baselines trained at full data. On the cartpole, F1 is competitive from the start; the primary gain from adaptive sampling is in the uncertain region rate, which drops to near 1% compared to ∼10% for the non-adaptive variant, yielding ∼1.4× data savings. On the planar quadrotor, both variants initially fall below the classification baseline on F1 before surpassing it with sufficient data, with adaptive sampling accelerating the crossover (∼1.5× data savings) and reaching higher final F1. On the spatial quadrotor, F1 surpasses the classification baseline throughout, but the uncertain region rate plateaus at ∼25% for both variants, consistent with the model error analysis described next. Across all systems, both variants maintain false positive rates below 1%; the F1 gains from adaptive sampling are driven by reduced false negatives, e.g., 10.4% on the planar quadrotor versus 22.6% for the non-adaptive variant.

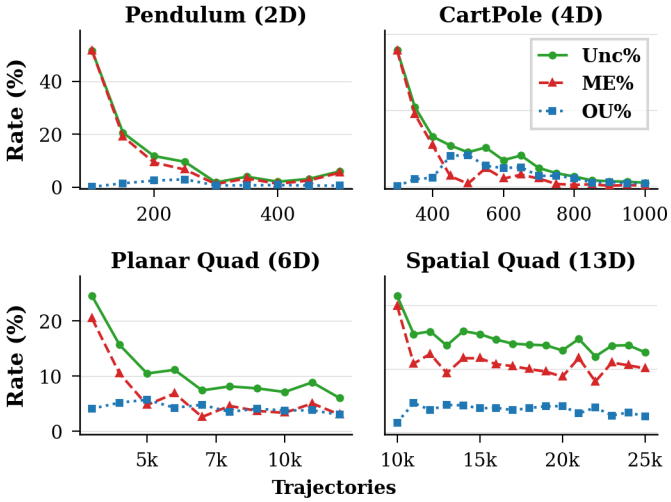

Fig. 6: Uncertain region decomposition across adaptive iterations. Green: total uncertain region rate. Red (dashed): invalid final state predictions classified (model-error) as <Unc>. Blue (dotted): true classification uncertainty (outcome uncertainty).

Uncertain region decomposition and adaptive divergence: Figure 6 reveals a three-phase pattern: a *model-error phase*, where invalid predictions dominate the uncertain region; a *transfer phase*, where the model begins producing valid final states in previously poorly-modeled regions, revealing multimodal outcome distributions that convert model error into outcome uncertainty; and a *resolution phase*, where both components decline together. The transition from the first to the second phase defines the adaptive divergence point: ∼450 trajectories on the cartpole (model error falling from 35.6% to 0.2%) and ∼5,000 on the planar quadrotor (∼3%). The 13D quadrotor remains in the transfer phase throughout the trajectory budget (model error ∼20%, outcome uncertainty ∼5–8%), explaining the r_{unc} plateau in Figure 7. This is consistent with the expectation that adaptive sampling requires a minimum dataset size before outperforming uniform sampling [27], with the divergence point scaling with dimensionality [33] as the true boundary occupies a vanishingly thin manifold in the 13D state space [34].

E. Ablation Studies

Variant	Planar Quad(6D) F1↑	Planar Quad(6D) Unc.↓	Spatial Quad(13D) F1↑	Spatial Quad(13D) Unc.↓
Non-Adaptive	.855	13.2%	.947	25.3%
Adaptive (fixed threshold)	.945	9.3%	.960	31.9%
Adaptive (optimized)	.933	6.0%	.953	25.2%

TABLE II. Adaptive (fixed) uses guided sampling with classification thresholds ($\lambda = 0.5, \delta = 0.1$); Adaptive (optimized) jointly optimizes thresholds (λ^*, δ^*) with the adaptive sampling loop.

Fixed vs. Optimized Thresholds: Table II shows that with fixed thresholds (λ=0.5, δ=0.1), adaptive sampling improves F1 over the non-adaptive baseline on both systems but increases the uncertain region rate on the spatial quadrotor (25.3% to 31.9%). This occurs because adaptive sampling shifts the training distribution toward the boundary, and fixed thresholds do not account for this shift. Jointly optimizing thresholds at each iteration corrects this: uncertain region rates drop to 6.0% and 25.2% respectively, recovering a rate below the non-adaptive baseline on the spatial quadrotor.

Sensitivity to r_{unc}: Figure 7 varies r_{unc} from 0 to 1.0 across all four systems. On the cartpole and planar quadrotor, increasing r_{unc} monotonically reduces the uncertain region rate while F1 remains stable or improves. On the pendulum, moderate values ($r_{\text{unc}} = 0.75$) achieve the lowest uncertain region rate (1.3%), but $r_{\text{unc}} = 1.0$ reverses the gain (6.0%), suggesting that in this low-dimensional setting, fully concentrating sampling near the boundary deprives the model of coverage in the unambiguous success and failure regions.

F. Coverage Guarantees

This section empirically validates the conformal layer of Sec. V at $\alpha = 0.1$. Across all 4 systems at the final adaptive iteration, the conformal quantile is $\hat{q} = 0$ and empirical coverage exceeds 0.98, meaning the classification pipeline's point predictions already satisfy the $(1-\alpha) = 0.90$ coverage

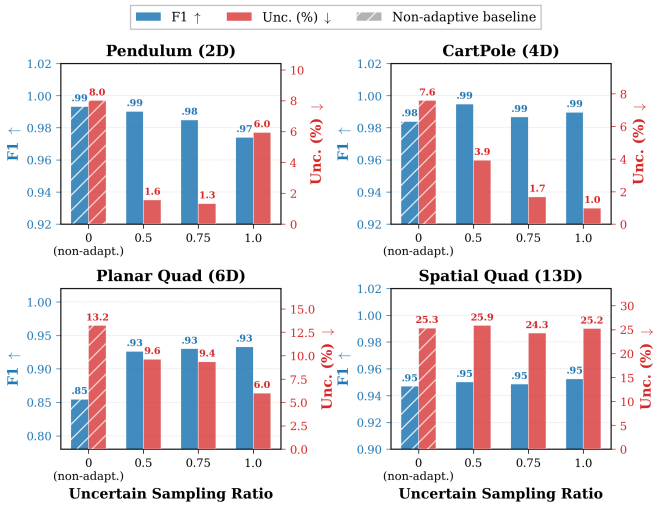

Fig. 7: Effect of r_{unc} on classification quality across systems. Blue bars: F1 score (left axis). Red bars: uncertain region rate (right axis). Hatched bars at $r_{\text{unc}} = 0$ denote the non-adaptive baseline.

target without conformal widening. The average prediction set size (Sec. V-E) is exactly 1.0 on the pendulum, cartpole, and spatial quadrotor, and 1.01 on the planar quadrotor, confirming that nearly all predictions are singletons.

VII. Conclusion

This paper presents a data-driven framework for RoA estimation that combines generative final-state prediction via conditional flow matching, uncertainty-guided adaptive sampling, and conformal coverage guarantees, without requiring access to system dynamics or controller structure. Evaluated on four systems spanning 2D to 13D, the proposed method achieves the highest F1 on all benchmarks with a formal $(1-\alpha)$ coverage guarantee and up to $1.5\times$ data savings over uniform sampling. The advantage over discriminative baselines widens with state-space dimensionality, and the conformal layer provides guarantees at no cost to accuracy, with $\hat{q} = 0$ across all systems. A limitation of the proposed framework is the assumption of deterministic dynamics and noise-free observations. Extending the framework to stochastic systems and image-based state representations would broaden its applicability. Additionally, since the generative model is trained without outcome labels, it could directly support controller composition by evaluating new success criteria at query time without retraining.

References

[1] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *CSUR*, 2019.

[2] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," in *ICRA*, 2021.

[3] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *SSCI*, 2020.

[4] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, 2021.

[5] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE TRO*, 2023.

[6] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *CoRL*, 2018.

[7] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *ICLR*, 2023.

[8] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of backward reachable sets for continuous dynamic games," *IEEE TAC*, 2005.

[9] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *CDC*, 2017.

[10] S. Sastry, "Nonlinear systems," *IAM*, 1999.

[11] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," *NeurIPS*, 2019.

[12] E. R. Vieira, E. Granados, A. Sivaramakrishnan, M. Gameiro, K. Mischaikow, and K. E. Bekris, "Morse graphs: Topological tools for analyzing the global dynamics of robot controllers," in *WAFR*, 2023.

[13] E. R. Vieira, A. Sivaramakrishnan, S. Tangirala, E. Granados, K. Mischaikow, and K. Bekris, "Morals: Analysis of high-dimensional robot controllers via topological tools in a latent space," in *ICRA*, 2024.

[14] M. Janner, I. Mordatch, and S. Levine, "Gamma-models: Generative temporal difference learning for infinite-horizon prediction," *NeurIPS*, 2020.

[15] L. Schramm and A. Boularias, "Bellman diffusion models for offline reinforcement learning," in *18th EWRL*.

[16] C. Zhu, X. Wang, T. Han, S. S. Du, and A. Gupta, "Transferable reinforcement learning via generalized occupancy models," in *ICML 2024 Workshop*.

[17] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *JAIS*, 2022.

[18] N. Botteghi, F. Califano, M. Poel, and C. Brune, "Trajectory generation, control, and safety with denoising diffusion probabilistic models," *arXiv preprint arXiv:2306.15512*, 2023.

[19] V. Vovk, A. Gammerman, and G. Shafer, *ALRW*. Springer, 2005.

[20] A. N. Angelopoulos, S. Bates, *et al.*, "Conformal prediction: A gentle introduction," *Foundations and trends® in machine learning*, 2023.

[21] A. Lin and S. Bansal, "Verification of neural reachable tubes via scenario optimization and conformal prediction," in *L4DC*, 2024.

[22] A. Dixit, L. Lindemann, S. X. Wei, M. Cleaveland, G. J. Pappas, and J. W. Burdick, "Adaptive conformal prediction for motion planning among dynamic agents," in *L4DC*, 2023.

[23] L. Lindemann, Y. Zhao, X. Yu, G. J. Pappas, and J. V. Deshmukh, "Formal verification and control with conformal prediction," *arXiv preprint arXiv:2409.00536*, 2024.

[24] F. Cairoli, L. Bortolussi, J. V. Deshmukh, L. Lindemann, and N. Paoletti, "Conformal predictive monitoring for multi-modal scenarios," in *ICRV*, 2025.

[25] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *NeurIPS*, 2017.

[26] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *CDC*, 2016.

[27] X.-S. Wang, S. A. Moore, J. D. Turner, and B. P. Mann, "A model-free sampling method for basins of attraction using hybrid active learning (hal)," *Commun. Nonlinear Sci. Numer. Simul.*, 2022.

[28] Z. Kharazian, T. Lindgren, S. Magnússon, and H. Boström, "Copal: Conformal prediction for active learning with application to remaining useful life estimation in predictive maintenance," *PMLR*, 2024.

[29] M. D. Zhao, H. Admoni, R. Simmons, A. Ramdas, and A. Bajcsy, "Conformalized interactive imitation learning: Handling expert shift and intermittent feedback," in *ICLR*, 2024.

[30] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *ICLR*, 2023.

[31] Z. Yuan, A. W. Hall, S. Zhou, L. Brunke, M. Greeff, J. Panerati, and A. P. Schoellig, "Safe-control-gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics," *IEEE RAL*, 2022.

[32] J. Drgona, A. Tuor, J. Koch, M. Shapiro, B. Jacob, and D. Vrabie, "NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations," 2023. https://github.com/pnnl/neuromancer.

[33] S. Dasgupta, "Two faces of active learning," *TCS*, 2011.

[34] H. Ha, S. Gupta, S. Rana, and S. Venkatesh, "High dimensional level set estimation with bayesian neural network," in *AAAI*, 2021.