SAC-MoE: Reinforcement Learning with Mixture-of-Experts for Control of Hybrid Dynamical Systems with Uncertainty

Leroy D'Souza, Akash Karthikeyan, Yash Vardhan Pant, Sebastian Fischmeister

***Abstract*— Hybrid dynamical systems result from the interaction of continuous-variable dynamics with discrete events and encompass various systems such as legged robots, vehicles and aircrafts. Challenges arise when the system's modes are characterized by *unobservable (latent) parameters* and the events that cause system dynamics to switch between different modes are *also unobservable*. Model-based control approaches typically do not account for such uncertainty in the hybrid dynamics, while standard model-free RL methods fail to account for abrupt mode switches, leading to poor generalization.**

To overcome this, we propose SAC-MoE which models the actor of the Soft Actor-Critic (SAC) framework as a Mixture of Experts (MoE) with a learned router that adaptively selects among learned experts. To further improve robustness, we develop a curriculum-based training algorithm to prioritize data collection in challenging settings, allowing better generalization to unseen modes and switching locations. Simulation studies in hybrid autonomous racing and legged locomotion tasks show that SAC-MoE outperforms baselines (up to 6x) in zero-shot generalization to unseen environments. Our curriculum strategy consistently improves performance across *all* evaluated policies. Qualitative analysis shows that the interpretable MoE router activates different experts for distinct latent modes. [1].

I. Introduction

Reinforcement Learning (RL) algorithms are typically developed under the assumption of continuous, stationary system dynamics that are invariant to the environment that a system is operating in. However, many real-world systems exhibit *hybrid* dynamics where the active dynamics model *changes* due to spatially varying environmental conditions (e.g., a race car on an asphalt track with a wet patch at some location) and internal system state (e.g., different gaits for legged robots produce different joint angles that affect the system's contact dynamics [21]). Such systems are called hybrid dynamical systems and designing RL policies to control them remains a challenge.

Example 1. *Consider an autonomous vehicle navigating an environment with different terrains such as dry asphalt, wet surfaces, and snow. Each surface induces distinct dynamics due to differing friction, corresponding to different operating* modes *of the hybrid system. As the vehicle transitions across terrains, its control strategy must be able to account for the changing dynamics during the transition. Figure 1 shows that switching between two policies learned* separately *on asphalt and wet roads, may fail to handle transitions and skid off the track (blue trajectory). We propose a method that learns to account for such transitions between modes and better deals with hybrid system control (red trajectory).*

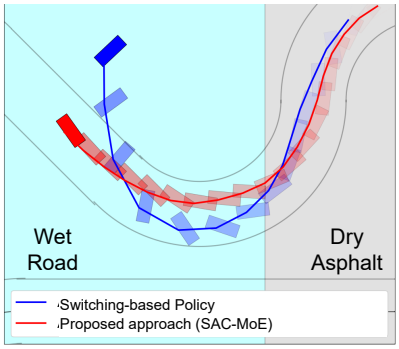

Fig. 1: Vehicle trajectories (right to left) in an environment with different road surfaces. The blue trajectory shows that switching between *separate* policies for each mode results in failure. The red trajectory is from our proposed method, SAC-MoE, which can successfully handle such transitions.

Example 1 indicates that we desire policies (controllers) for hybrid systems which account for *discrete switching* (transitions) between different modes that each have their own dynamics. In this work, we consider the problem where i) modes are characterized, in part, by *latent* (unobservable) parameters (e.g., friction coefficient) and, ii) mode switching locations are also *a priori unknown*. Both these components can *vary with the environment* and combine to form the (latent) hybrid system "context" (formalized in Section III).

We desire policies that account for such latent context-dependent dynamics. While the model-based control of hybrid systems [8] has long been studied using optimization techniques, they usually consider piecewise affine systems [19] and face tractability issues when working with general complex nonlinear dynamics [11]. Moreover, they do not consider the challenges indicated above that result from a lack of context knowledge. This motivates the consideration of model-free RL techniques to learn policies for hybrid systems that neither require such knowledge apriori nor suffer from tractability issues at inference time.

Contributions of this work. Motivated by the challenges described above, we now outline the key contributions of this work. We propose a) SAC-MoE which parameterizes the actor in Soft Actor-Critic (SAC) [14] as a Mixture of Experts (MoE) [17] to account for switching dynamics and robustly handle mode transitions using a *learned router* that activates different (learned) experts, b) an adaptive curriculum learning scheme for hybrid systems that *dynamically*

Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. {l8dsouza, a9karthi, yash.pant, sfischme}@uwaterloo.ca.

This work was supported in part by Magna International and the NSERC Discovery Grant.

[1] https://ljdsouza.github.io/sacmoe/

estimates context hardness and prioritizes data collection in more challenging contexts to improve policy generalization across the space of contexts.

We evaluate our approach through extensive simulations on hybrid systems viz. autonomous racing across varied surfaces and legged locomotion. We show that the proposed SAC-MoE approach outperforms the baselines by up to 6x in deployment settings with unseen contexts. We also show that the proposed curriculum approach consistently leads to better zero-shot generalization across *both* the baselines and SAC-MoE compared to other curricula demonstrating its general applicability. Finally, we qualitatively demonstrate that the interpretable MoE router learns to select different experts for modes with very different dynamics while using similar experts for similar modes.

II. Related Works

We briefly cover existing literature on the control of hybrid systems across both (model-free) RL and model-based control domains.

Reinforcement Learning for Hybrid System Control. Model-free RL has demonstrated impressive capabilities in tasks such as complex terrain traversal and legged locomotion. Hierarchical extensions, such as the option-critic architecture [3], enable end-to-end learning of policies over "options," thereby allowing the discovery of specialized sub-policies for different terrains or gaits. However, such approaches are prone to degenerate behaviors, where a single sub-policy dominates the entire episode or terminates immediately upon switching and often require auxiliary signals derived from domain knowledge [24]. Moreover, they usually assume access to ground-truth information (e.g., friction maps in Example 1) to guide switching. In our setting, we assume this information is *latent*, limiting the use of option-based methods in contrast to our proposed approach which employs a routing mechanism [17], [30] to learn specialized sub-policies without access to latent parameters.

The MoE architecture has been used in RL for efficient training [29] and multi-task learning [9], but has not been applied to hybrid system control. Our proposed approach adapts MoE to this setting, where mode parameters and switching locations are unobserved. Unlike prior work that can explicitly condition the router on task or context information, our formulation requires expert specialization to emerge implicitly, making it suitable for hybrid control problems of the type shown in Example 1.

There exists work that applies RL for hybrid system control by using discrete hybrid automata [21], [27]. However, these do not consider transitions driven by environmental changes, a key challenge highlighted in Figure 1.

Control for Contextual Markov Decision Processes (cMDPs). The hybrid systems we deal with can be framed as a cMDP where the modes and mode switching describe the "context" (described in Section III). It has been demonstrated that policies conditioned on *observable* context can do better than ones without this information [4] but may also perform suboptimally in unseen contexts [33]. In our setting, the context is *unobservable* as indicated in Section I preventing explicit policy conditioning on context information. AMAGO [12] considers the problem of context-based meta-RL using sequence models but does not consider switching between different dynamics modes of a hybrid system *during* an episode. JCPL [26] learns to encode context for policy conditioning. In our setting, the current and previously active modes in an episode do not necessarily tell us about what modes could be expected *in the future*. This contrasts with JCPL's setting yielding a harder control task, particularly for zero-shot deployment.

Model-based Control of Hybrid Systems. Model-based control-theoretic work on hybrid systems [2] has used Lyapunov theory for stability analysis, while most recent efforts employ optimization-based Model Predictive Control (MPC) [8]. Mode switching typically yields mixed-integer problems [5], which are computationally demanding [11], [6]. Data-driven MPC methods extend this line by learning dynamics models for a fixed set of known modes, but require a priori mode-specific datasets [10]. Such assumptions limit generalization to previously unseen modes. In contrast, we address the setting where mode parameters are latent, mode-specific data is not available a priori, and we desire effective generalization to unseen modes.

III. Problem Statement

We consider a hybrid system with state $s_t \in \mathcal{S} \subset \mathbb{R}^n$, control input $a_t \in \mathcal{A} \subset \mathbb{R}^u$, space of latent mode parameters $\mathcal{L} \subset \mathbb{R}^l$ with discrete-time dynamics of the form,

$$s_{t+1} \sim \sum_{m=1}^{|\mathcal{M}|} \delta(m, s_t) f_m\left(s_t, a_t; \mu^m\right) \tag{1}$$

Here, $\mathcal{M} \subseteq \mathcal{L}$ denotes the set of latent mode vectors. $f_m : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the stochastic dynamics associated with mode m, characterized by the latent parameter vector $\mu^m \in \mathcal{M}$, where $\Delta(\mathcal{S})$ is a probability distribution over $\mathcal{S}$. The latent mode switching indicator, $\delta(m, s_t) : \{1, \ldots, |\mathcal{M}|\} \times \mathcal{S} \to \{0, 1\}$, determines which dynamics mode is active at a given state $s_t \in \mathcal{S}$. As indicated in Section I, we define the context assignment, $c = (\mathcal{M}, \delta(\cdot, \cdot))$, which characterizes where different dynamics modes occur across the state space $\mathcal{S}$, including spatial variations across an environment (as in Example 1). Thus, an environment is associated with a *unique* context assignment (visualized in Fig. 2), and we use the terms interchangeably henceforth. Only one mode is active in a given state i.e., $(\sum_{m=1}^{\mathcal{M}} \delta(m, s) = 1) \, \forall \, s \in \mathcal{S}$. We define $\mathcal{V}$ as the space of valid switching indicators that satisfy this and use $\mathcal{V}_\delta \subset \mathcal{V}$ to indicate the case where only a fraction of the possible mode switching locations are realized.

We now express these hybrid systems as a process over environments characterized by $\mathcal{V}_\delta, \mathcal{L}$ by adapting the definition of cMDPs [15].

Definition 1. *A **Hybrid Contextual Markov Decision Process (HcMDP)** is the tuple $(\mathcal{C}, \mathcal{L}, \mathcal{V}_\delta, \mathcal{S}, \mathcal{A}, R, \mathcal{T})$, where $\mathcal{S} \subset \mathbb{R}^n$, $\mathcal{A} \subset \mathbb{R}^u$, $\mathcal{L} \subset \mathbb{R}^l$ are the state, action and*

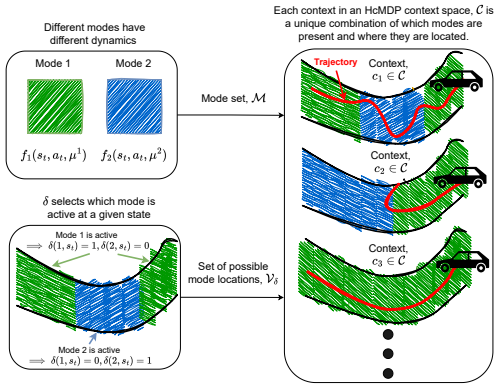

Fig. 2: A visualization of how mode parameters and mode location information are combined to generate various contexts that give rise to the HcMDP context space, $\mathcal{C}$ in Definition 1. Different contexts can lead to different trajectories for a given policy.

latent mode parameter spaces respectively. $\mathcal{C} = \{(\mathcal{M}, \delta) \mid \mathcal{M} \subseteq \mathcal{L}, \delta \in \mathcal{V}_\delta\}$ *represents the space of (latent) contexts where each context* $c \in \mathcal{C}$ *corresponds to an environment characterized by the modes present in it (determined by* $\mathcal{M}$*) and the locations where these modes occur across* $\mathcal{S}$ *(determined by* δ*).* $\mathcal{T}(c)$ *maps context* $c \in \mathcal{C}$ *to the environment-specific stochastic transition dynamics given by equation 1.* $R : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ *is the bounded reward function.*

The combination of unknown mode dynamics with abrupt switching between modes that varies across environments makes learning a policy for controlling the hybrid system across different contexts a hard task. We consider learning a policy, $\pi_\theta(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$ (where $\Delta(\mathcal{A})$ is a parametric family of distributions over $\mathcal{A}$, that only obtains samples from a finite set of contexts ($\mathcal{C}$) during training and must generalize to previously unseen contexts ($\mathcal{C}^{\text{test}}$) at test time. We formalize this in the problem statement below.

Problem 1. *Given an HcMDP* $(\mathcal{C}, \mathcal{L}, \mathcal{V}_\delta, \mathcal{S}, \mathcal{A}, R, \mathcal{T})$ *defined by a finite set of latent modes,* $\mathcal{L}$*, and switching functions,* $\mathcal{V}_\delta \subset \mathcal{V}$*, we aim to learn a control policy* $\pi_\theta(a \mid s) : \mathcal{S} \to \Delta(\mathcal{A})$ *that maximizes the expected return,* $J(\pi) = \mathbb{E}_{c\sim\mathcal{C}, \tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ *where* τ *denotes trajectories sampled under* π*,* γ *is the discount factor and* r_t *is the reward at timestep* t*. At deployment, the HcMDP* $(\mathcal{C}^{test}, \mathcal{L}^{test}, \mathcal{V}_\delta^{test}, \mathcal{S}, \mathcal{A}, R, \mathcal{T})$ *with* $\mathcal{C}^{test} \supset \mathcal{C}$ *(since* $\mathcal{L}^{test} \supset \mathcal{L}, \mathcal{V}_\delta^{test} \supset \mathcal{V}_\delta$*) produces previously unseen contexts requiring the policy to generalize to maintain performance.*

A. Why switching policies are insufficient for hybrid systems

The hybrid structure in equation 1 naturally motivates a switching policy, π_{sw}, that selects among a set of component policies $\{\pi_m\}_{m=1}^{|\mathcal{M}|}$, each specific to a different mode i.e., π_i is a policy *specialized* for $f_i(s_t, a_t; \mu^i)$ alone. Formally,

$$\pi_{\text{sw}}(a \mid s) = \sum_{m=1}^{|\mathcal{M}|} \delta(m, s_t)\pi_m(a \mid s) \tag{2}$$

Thus, π_{sw}'s selection is based on the mode active at timestep t as specified by $\delta(m, s_t)$. In Example 2 below, we show that even if an oracle were to provide π_{sw} with access to $\delta(\cdot, \cdot)$ (which is otherwise unobservable), it still fails to generate desirable behavior across mode switches for a hybrid system.

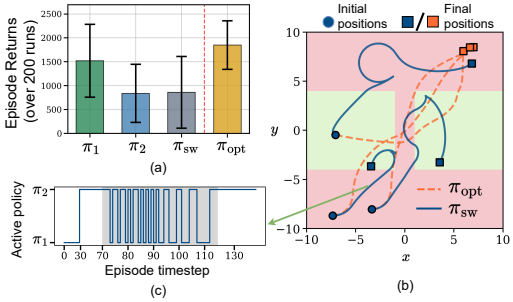

Fig. 3: (a) Episode returns (goal-seeking reward) over 200 runs for policies in a test environment. (b) Visualization of the test environment (mode 1 in red and mode 2 in green) and selected trajectories for π_{sw} and π_{opt}. (c) Visualization of π_{sw} switching between component policies over a trajectory.

Example 2. *Consider the nominal dynamics of a kinematic bicycle model [1] with additional perturbation terms parametrized by a latent mode parameter* $\mu \in \mathcal{L} = \{0, 6\}$*.* θ, v *are the heading angle and velocity and* a_{long}, ψ *are the input acceleration and steering angle.*

$$\begin{aligned} \dot{\theta} &= \dot{\theta}_{nominal} - \left(0.05\, e^{\psi} + 0.15\, a_{long} \tanh(\theta)\right) \mu \\ \dot{v} &= \dot{v}_{nominal} - \left(0.1\, a_{long} \cos(\theta) + 0.3\, v\, \sin(\psi)\right) \mu \end{aligned} \tag{3}$$

Policies π_1 *and* π_2 *are first trained separately in* single-mode *environments, obtained by fixing* $\mu = \mu^1 = 0$ *and* $\mu = \mu^2 = 6$ *in equation 3, respectively.*

Now, consider a test environment where both modes are present ($\mathcal{M} = \{\mu^1, \mu^2\}$*) and distributed spatially across the workspace. For reference, we also include* π_{opt}*, trained* directly *in this test environment. Figure 3(a) shows that the switching policy* π_{sw}*, which selects between* π_1 *and* π_2*, achieves* lower *return than even* π_1 *alone. As shown in Fig. 3(b),* π_{sw} *produces suboptimal trajectories, and in the worst case (Fig. 3(c)) becomes trapped at mode boundaries, oscillating between conflicting policies, despite being provided with information from an oracle.*

Example 2 highlights that different component policies aim to generate different (possibly conflicting) conflicting behaviors *over a trajectory* but are prevented from realizing their objectives due to the abrupt switching in π_{sw}. This leads to myopic and suboptimal behavior which Section V shows to also manifest in other systems. This motivates the need for methods that learn policies *not restricted to the form of the switching policy* in equation 2 to avoid these pitfalls.

IV. SAC-MoE: Method

In this section, we propose SAC-MoE, an actor–critic framework built on the SAC formulation [13] with the actor being parameterized as a Mixture of Experts (MoE), enabling it to handle mode switching in HcMDP contexts and generalize to *unseen* contexts at inference (Problem 1, Example 1). This design addresses two central challenges: i) decomposing the policy into expert sub-policies that specialize without access to mode labels, and ii) employing a

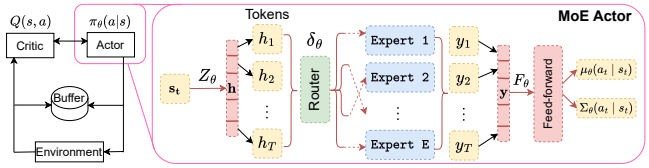

Fig. 4: **SAC-MoE Overview.** We adopt an actor-critic framework, where the actor is parameterized as a Mixture-of-Experts model. The encoder produces an embedding which is split into tokens. The router mechanism assigns tokens to different experts and merges their outputs to produce the action distribution.

sparse, learned router to adaptively compose experts based on the observed state, thereby improving generalization and control around mode transitions. We first detail the MoE actor in Section IV-A and compare it to the switching policy of Example 2. We then show how the MoE actor integrates into SAC (Fig. 4) and present the overall training objective in Section IV-B. Finally, Section IV-C introduces an adaptive curriculum algorithm that dynamically prioritizes learning on harder contexts, improving policy generalization while avoiding overfitting to easier contexts.

A. Policy Representation

Here, we define the components that make up our MoE actor as shown in Fig. 4.

Tokenization. Given a state s, an encoder Z_θ produces a latent representation $\mathbf{h} = Z_\theta(s) \in \mathbb{R}^{T\times d}$, which we split into T tokens $\{h_i\}_{i=1}^T$, each of dimension d [32] allowing each token to capture distinct (complementary) aspects of the state for more holistic control.

Experts. We use a set of E learnable experts, $\{\pi_\theta^e\}_{e=1}^E$, where each expert is a parameterized sub-policy that maps input tokens to outputs $\{\, y_i^e = \pi_\theta^e(h_i) \mid e \in \{1,\dots,E\}\}\, \forall\, i \in \{1,\dots,T\}$. In practice, different experts may learn to specialize on particular subsets of tokens which necessitates learning a *weighted* combination of their outputs, as described below.

Router. The router δ_θ assigns each token h_i a probability distribution over the E experts by producing logits that are transformed into routing probabilities $\alpha_i = \mathrm{softmax}(\delta_\theta(h_i)) \in \mathbb{R}^E$. We then select the k experts with the highest routing probabilities for token h_i, denoted as $S_i = \mathtt{top}_k(\alpha_i, k)$, and dispatch h_i to those experts. The outputs of the selected experts $e \in S_i$ are weighted by their normalized routing score $\tilde{\alpha}_i^e = \frac{\alpha_i^e}{\sum_{j\in S_i}\alpha_i^j}$. The aggregated output for h_i is then,

$$y_i = \sum_{e\in S_i} \tilde{\alpha}_i^e\, y_i^e$$

Action Distribution. The token outputs $\{y_i\}_{i=1}^T$ are concatenated and passed through a feed-forward layer, F_θ, that parameterizes the mean $\mu_\theta(a \mid s)$ and covariance $\Sigma_\theta(a \mid s)$ of the action distribution.

Comparing MoE and switching policies. The resulting actor can be viewed as,

$$\pi(a \mid s) = \sum_{e=1}^{E} \delta_\theta^e(Z_\theta(s))\, \pi_\theta^e(a \mid s), \tag{4}$$

where $\delta_\theta^e(Z_\theta(s))$ denotes the learned routing weight for expert e. This form resembles the switching policy in equation 2, but with two key differences. i) In π_{sw}, $\delta(m, s_t)$ are *binary* indicators that activate a single policy at a time. In contrast, δ_θ^e are real-valued weights, enabling the router to *compose* multiple expert sub-policies. ii) In π_{sw}, the policy selector and mode-specific policies are *decoupled* as two separate components. In contrast, our MoE formulation has both the router and the experts are trained *jointly* in an end-to-end framework.

These distinctions highlight that MoE policy is not just a relaxation of the switching policy, but has additional expressivity to allow adaptive specialization and behavior composition across latent modes inspite of not needing to condition the router on mode information.

B. Policy Learning

We train our policy using the off-policy Soft Actor-Critic (SAC) framework [13] with the actor parameterized using MoE described in Section IV-A. The actor's objective maximizes expected return and policy entropy,

$$L_{\mathcal{A}} = \mathbb{E}_{s_t\sim\mathcal{D},\, a_t\sim\pi_\theta(\cdot|s_t)}\Big[-\big(\min_{i=1,2} Q_{\psi_i}(s_t,a_t)\big) - \alpha\, \mathcal{H}(\pi_\theta(\cdot|s_t))\Big] \tag{5}$$

where θ denotes parameters from the encoder, router, and experts, $\mathcal{H}(\pi_\theta(\cdot|s_t)) = -\mathbb{E}_{a_t\sim\pi_\theta}[\log \pi_\theta(a_t|s_t)]$ is the Shannon entropy of the policy, α is the entropy temperature, and $\mathcal{D}$ is the replay buffer. The critic $Q_\psi(s,a)$ is trained with temporal-difference targets using double Q-learning [16].

To better capture a policy that can handle hybrid systems across the expert sub-policies, we include an auxiliary regularization load loss term to promote balanced expert usage and encourage expert specialization within the MoE router.

Load Loss. To regularize expert assignments, we adopt the *load loss* [30], which penalizes variance in the number of tokens routed to different experts. For a mini-batch $\mathcal{B}$, we estimate the probability of each expert e being selected for a token $b \in \mathcal{B}$ under noisy routing, defined as $p_e(b) = \mathbb{P}((Wb)_e + \epsilon_{\text{new}} \geq \text{threshold}_k(b))$, where $\epsilon_{\text{new}} \sim \mathcal{N}(0, 1/E^2)$ and $\text{threshold}_k(b)$ is the k-th largest router score for token b. The expected token assignment count for each expert is then $\text{load}_e(\mathcal{B}) = \sum_{b\in\mathcal{B}} p_e(b)$, and the load-balancing loss is given by the squared coefficient of variation, $L_{\text{load}}(\mathcal{B}) = \left(\frac{\text{std}(\{\text{load}_e(\mathcal{B})\}_{e=1}^E)}{\text{mean}(\{\text{load}_e(\mathcal{B})\}_{e=1}^E)}\right)^2$. Minimizing this term encourages more uniform expert usage, preventing collapse to a few experts and promoting stable specialization.

Training Objective. The overall training procedure alternates between updating the actor and critic. The actor is trained with the combined loss $L(\theta) = L_{\mathcal{A}} + \lambda_{\text{load}}\, L_{\text{load}}$, where $L_{\mathcal{A}}$ is the standard SAC actor loss (Eq. 5), L_{load} is the load-balancing regularizer, and λ_{load} is a weighting hyperparameter. The critic is updated using the Bellman residual with double Q-learning [16].

C. Curriculum Learning for HcMDPs

In cMDPs, contexts are typically sampled from a *static uniform* distribution, i.e., $c^i \sim \text{Uniform}(\mathcal{C})$ at the start of

Algorithm 1 Curriculum learning for training in a HcMDP

1: **Input:** Training context set $\mathcal{C}$, curriculum metric g, distribution function d_g
2: Initialize $\mathcal{G} \leftarrow \mathbf{0}_{|\mathcal{C}|}$
3: Initialize $\Delta(\mathcal{C}) \leftarrow \text{Uniform}(\mathcal{C})$
4: **for** $i = 1$ **to** N_E **do**
5: $\quad c^i \sim \Delta(\mathcal{C})$
6: $\quad U \leftarrow \text{RunEpisode}(c_i)$ ▷ Record episode
7: $\quad \mathcal{G}[c^i] \leftarrow g(\mathcal{G}[c^i], U)$ ▷ Hardness estimate of c^i
8: $\quad \Delta(\mathcal{C}) \leftarrow d_g(\mathcal{G})$ ▷ Context sampling dist.
9: **end for**

each episode i [18]. While this can be applied to HcMDPs, it ignores the fact that some contexts can be more challenging than others. With reference to Example 1, i) controlling the vehicle during an asphalt-to-ice transition is harder during a corner bend than a straight road due to lateral dynamics [1] and ii) controlling the vehicle during an asphalt-to-ice switch on a turn is harder than an asphalt-to-damp road switch since the dynamics shift is more drastic in the former case. Thus, both the mode switch locations (affected by δ) and the modes present (affected by $\mathcal{M}$) govern the hardness of a context $c \in \mathcal{C}$. We propose a curriculum learning strategy [25] based on this insight.

Remark 1. *To simplify the discussion, we talk about sampling $c^i \sim \Delta(\mathcal{C})$. However, we do not assume access to the context itself. c^i should be interpreted as an* index *into the finite training context set $\mathcal{C}$.*

However, HcMDPs present unique challenges for curriculum learning since, i) the relative hardness of contexts is not known *a priori* and depends on the *evolving* policy, ii) contexts may *share* some modes and switching locations, but this structure is *unobservable* to the curriculum, so such similarities cannot inform hardness estimates over $\mathcal{C}$. These challenges contrast with typical curriculum learning settings, where i) difficulty can be ordered using domain knowledge [22], or ii) the curriculum has *access to context parameters* and can manipulate them directly to probe similarities between contexts [23], in contrast to our setting in Remark 1 where only elements of $\mathcal{C}$ can be sampled, *not the underlying parameters*.

To address this, we develop the adaptive curriculum strategy in Algorithm 1 that maintains a *dynamic* sampling distribution $\Delta^i(\mathcal{C})$ over $\mathcal{C}$. Each episode produces an outcome U (e.g., return), recorded for the sampled context c^i (Line 6). A curriculum metric g updates $\mathcal{G}[c^i]$, the empirical difficulty estimate of context c^i, using U (Line 7). Finally, a distribution function d_g sets $\Delta^i(\mathcal{C})$ based on $\mathcal{G}$ to prioritize sampling of harder contexts (Line 8). In Section V-B, we instantiate this framework with specific choices of U, g, and d_g to show how an adaptive curriculum can improve performance without requiring prior knowledge of hardness or latent mode structure.

V. Simulation Studies

In this section, we evaluate and compare SAC-MoE against baseline algorithms (Section V-A) on two challenging hybrid control domains, i) autonomous racing [20] and ii) legged locomotion using the Walker2d-v5 in MuJoCo [31]. Both tasks involve latent mode dynamics and spatially varying mode transitions. Our experiments aim to answer the following questions,

P1. Does adaptive curriculum learning in HcMDPs improve performance? We test whether dynamically prioritizing harder contexts (Algorithm 1) leads to better policy performance compared to uniform sampling.

P2. Can SAC-MoE generalize to novel contexts? At inference, policies encounter unseen contexts $\mathcal{C}^{\text{test}} \supset \mathcal{C}$ (Problem 1). We examine whether the MoE actor improves generalization relative to baselines.

P3. Does the MoE router capture latent mode structure? We analyze router selections over trajectories to assess if expert activations correlate with different latent modes and interpret behavior captured by the learned sub-policies.

A. Baselines

SAC [14]: Standard off-policy actor-critic with maximum entropy regularization.

SAC-UPTrue [33]: SAC augmented with *oracle access* to the active mode at every timestep. Unlike prior work [33], we allow training in the full context space $\mathcal{C}$ rather than restricting learning to contexts with a *single* latent mode.

SAC-Switching (SAC-Sw): A switching policy π_{sw} (Example 2) composed of policies trained *separately* on single modes. During deployment, it is given *oracle access* to the current active mode and selects the policy trained on the closest mode for control at each timestep.

SAC and the proposed SAC-MoE do not observe mode labels during training or inference. For fair comparison, all methods are implemented in the stable-baselines3 framework [28]. Experiments use Python 3.11 on a 12-core CPU with an RTX A6000 GPU. For SAC-MoE we use 8 heads, 4 experts, and noisy top-k routing with $k = 2$ and a load loss weight (λ_{load}) of 0.01.

B. Case Study 1: Autonomous Racing

We first evaluate on an autonomous racing task where policies are incentivized to complete laps as quickly as possible without crashing. Hybrid dynamics arise from latent friction parameters, with mode switching induced by spatial variations in friction across the track (Fig. 5). Policies must therefore learn to handle abrupt friction changes, especially at high speeds and during cornering. We now make the training and testing HcMDPs from Problem 1 explicit.

Training setup. Training is performed on Track 1 with context space given by $\mathcal{L} = \{1.0, 0.5, 0.3\}$ (friction assignments) and $\delta \in \mathcal{V}_\delta$ as seen in Fig. 5 **A**. The SAC-Sw baseline consists of three component policies, one per friction value. For this section, we also define **SAC-SM**, trained like SAC but restricted to single-mode contexts i.e.,

Model	Curriculum	Single Surface		Multi-Surface		
		{High}	{Low}	{Low, Medium}	{Low, High}	{Low, Medium, High}
SAC-Sw		0.62 / 13.2 / 0.89	2.76 / 14.6 / 0.19	0.03 / 14.1 / 1.00	- / - / 1.00	- / - / 1.00
SAC-SM	*C*	3.23 / 14.4 / 0.11	1.26 / 15.2 / 0.64	0.03 / 16.0 / 1.00	- / - / 1.00	0.02 / 15.0 / 1.00
SAC	*A*	2.26 / **13.3** / 0.50	- / - / **1.00**	- / - / 1.00	- / - / 1.00	- / - / 1.00
	B	2.81 / 13.9 / 0.29	- / - / **1.00**	0.02 / 16.5 / 1.00	- / - / 1.00	0.08 / **14.9** / 1.00
	C	**3.05** / 15.9 / **0.02**	- / - / **1.00**	**0.95** / **15.9** / **0.82**	**0.22** / 15.5 / **0.97**	**1.06** / 15.9 / **0.73**
SAC-UPTrue	*A*	2.53 / **13.4** / 0.49	- / - / 1.00	0.01 / 15.8 / 1.00	- / - / 1.00	0.03 / 15.2 / 1.00
	B	1.53 / 13.6 / 0.75	0.02 / 16.4 / 1.00	0.04 / **15.4** / 1.00	- / - / 1.00	0.02 / 14.3 / 1.00
	C	**3.04** / 15.2 / **0.08**	**0.18** / **15.9** / **0.97**	**1.43** / 16.2 / **0.61**	**0.22** / **15.7** / **0.95**	**1.35** / 15.7 / **0.66**
SAC-MoE (ours)	*A*	**3.15** / **13.6** / 0.17	- / - / 1.00	0.04 / 15.4 / 1.00	- / - / 1.00	- / - / 1.00
	B	2.93 / 14.2 / 0.28	0.05 / 16.4 / 1.00	0.03 / **15.3** / 1.00	- / - / 1.00	0.09 / **15.0** / 0.99
	C	2.98 / 16.5 / **0.00**	**0.69** / **15.7** / **0.81**	2.32 / 16.4 / **0.24**	1.37 / **16.1** / **0.67**	2.03 / 16.3 / **0.40**

TABLE I: **Quantitative racing results (Track 1).** Performance comparison across curricula and policies for varied contexts. Entries show Laps Completed / Avg. Lap Time / Crash Rate. For contexts from a given set of evaluated surfaces, best results across all curricula for a given model are **bolded** and best results across all models are highlighted (red: laps (↑), blue: lap time (↓), green: crash rate (↓)).

$c = (\mathcal{M}, \delta)$ s.t. $|\mathcal{M}| = 1$. All policies are trained for 750K steps.

Evaluation setup. At test time, we evaluate generalization to unseen contexts $\mathcal{C}^{\text{test}} \supset \mathcal{C}$ across both Track 1 and Track 2 (out-of-distribution track layout) each partitioned into regions that yield $\delta \in \mathcal{V}_\delta^{\text{test}}$ (exemplified in Fig. 5 **B**, **C** respectively). Each region is randomly assigned a surface type that can have friction over a range of values viz., low $[0.25-0.35]$, medium $[0.45-0.6]$, or high $[0.8-1.0]$ friction. The union of these ranges yields the space $\mathcal{L}^{\text{test}}$ of test latent mode parameters. This yields a much larger set of contexts $\mathcal{C}^{\text{test}}$ than in training. Episodes are over 250 steps on Track 1 and 400 steps on Track 2.

Metrics. We report average lap time, average number of laps completed without a crash, and crash rate, over 200 evaluation episodes per policy.

Curricula to Compare. To address **P1**, we evaluate three curricula, with two derived from intuitive choices for the inputs to Algorithm 1 to yield $\Delta^i(\mathcal{C})$,

Curriculum A (Uniform). Static uniform sampling over $\mathcal{C}$, as in prior cMDP work [18].

Curriculum B (Step balancing). U = episode length. Contexts with fewer accumulated steps ($\mathcal{G}[c^i]$) are prioritized (by d_g), encouraging balanced context sampling.

Curriculum C (Performance-aware). U = episode return. Context with lowest moving average return ($\mathcal{G}[c^i]$) is selected with probability $p = 0.95$, and others are sampled randomly to avoid forgetting.

Result Discussion: Our results on Track 1 and Track 2 are summarized in Table I and Table II, respectively. We structure the discussion in three parts, i) comparing different curricula (**P1**) ii) contrasting training in HcMDPs with mode switches against *single-mode* contexts and, iii) evaluating zero-shot generalization to unseen contexts (**P2**).

Effect of Curricula (P1). Curriculum C consistently outperforms A and B across contexts (Table I). A and B overfit to high-friction cases, failing to generalize across the remaining contexts. In contrast, C balances learning on harder (low return) contexts with random sampling to avoid forgetting. We therefore adopt Curriculum C for all subsequent experiments.

Training in hybrid vs. single-mode contexts. SAC-SM, restricted to training in single-mode contexts, specializes in single-surface evaluations but fails to adapt contexts with multiple surfaces that can have switches between drastically different modes. SAC-Switching, which selects among single-mode policies, exhibits similar limitations and even has component policies that overfit (e.g., the $\mu = 1.0$ policy fails to generalize to the $[0.8, 1.0]$ friction range). This confirms that training in single-mode contexts *alone* is insufficient for control over general HcMDP contexts.

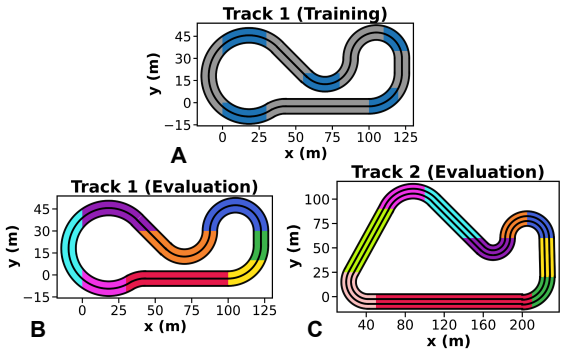

Fig. 5: **Autonomous racing setup. (A)** Training racetrack with $\mathcal{C}$ where each uniquely colored set of regions corresponds to a particular mode's (with value from $\mathcal{L} = \{1.0, 0.5, 0.3\}$) locations. **(B,C)** Evaluation tracks where colored regions represent surfaces with friction values sampled from predefined ranges, yielding diverse test contexts $\mathcal{C}^{\text{test}}$. Track 2 is out-of-distribution.

Generalization performance (P2). SAC-MoE consistently outperforms the baselines in zero-shot generalization to novel contexts. On Track 1, it completes more laps with fewer crashes across most contexts, while maintaining lap times comparable to SAC and SAC-UPTrue's more aggressive policies (Table I). Unlike SAC-UPTrue, which benefits from oracle access to the active mode but lacks a mechanism for sub-policy specialization, SAC-MoE leverages its router to adaptively compose experts, enabling robust generalization. This carries over to the unseen Track 2 layout (Fig. 5**C**), where SAC-MoE achieves the highest number of laps and lowest crash rate across *all* surface assignments (Table II). In contrast, SAC-UPTrue, which was closest to SAC-MoE on Track 1, falls significantly behind, in line with prior work [33] that mode-aware policies struggle in out-of-distribution settings. SAC matches SAC-MoE in lap times but suffers up to 3x higher crash rates, indicating that ag-

Eval. Surfaces	SAC	SAC-UPTrue	SAC-MoE (ours)
	Single Surface		
{L}	0.3 / **30.2** / 0.98	0.1 / 34.4 / 1.00	**0.5** / 33.3 / **0.94**
{M}	2.2 / **30.5** / 0.21	0.3 / 32.1 / 0.99	**2.3** / 32.7 / **0.07**
{H}	2.3 / 31.0 / 0.15	0.3 / **30.4** / 0.97	**2.5** / 31.9 / **0.00**
	Multi-Surface		
{L, M}	1.5 / **30.6** / 0.52	0.4 / 33.5 / 0.90	**1.7** / 32.9 / **0.37**
{L, H}	1.2 / **31.1** / 0.60	0.3 / 32.9 / 0.96	**1.5** / 32.8 / **0.43**
{M, H}	**2.3** / **30.9** / 0.14	0.3 / 31.5 / 0.98	**2.3** / 32.3 / **0.06**
{L, M, H}	1.5 / **30.7** / 0.52	0.2 / 32.1 / 0.99	**1.6** / 32.6 / **0.40**

TABLE II: **Quantitative racing results (Track 2).** Entries show Laps Completed (↑) / Avg. Lap Time (s) (↓) / Crash Rate (↓). **L, M, H** are low, med., high friction surfaces resp. For contexts across each set of surfaces, best results across models are **bolded**.

gressive performance alone is insufficient without robustness to mode switches.

C. Case Study 2 - MuJoCo Simulations

We evaluate policies in the Walker2d-v5 environment [31], where agents are incentivized to walk quickly while maintaining the torso angle within safe bounds. As in Section V-B, we introduce spatially varying friction. In addition, legged robots exhibit latent modes arising from internal joint configurations and gait patterns [7], [21]. As a result, these modes depend on both spatial variation and internal state, and are only partially observable, even in simulation. Hence, the SAC-UPTrue, SAC-Switching baselines receive partial mode information (friction), but not full knowledge of the latent mode parameters, which are difficult to define without expert domain knowledge. This makes the task particularly challenging and motivates our study of whether MoE can apply to such systems with complex contact dynamics.

Training and Evaluation setup. Training is performed on with context space given by $\mathcal{L} = \{1.0, 0.1\}$ (friction assignments). All policies are trained for 1M steps using Curriculum C from Section V-B. At test time, we evaluate generalization to unseen contexts $\mathcal{C}^{\text{test}} \supset \mathcal{C}$ characterized by novel friction coefficients and mode switching locations (as visualized in Fig. 6**C**). Episodes are over 1500 steps. We report average distance travelled and torso velocity over 100 evaluation episodes per policy.

Result Discussion: Our results on the test HcMDP are summarized in Table III. We structure the discussion in two parts, i) evaluating zero-shot generalization and qualitatively visualizing SAC-MoE's robustness to novel contexts (**P2**) ii) visualizing how the learned router activates different expert sub-policies corresponding to different latent modes to demonstrate how the MoE actor captures information about hybrid system dynamics implicitly without requiring context knowledge (**P3**).

Generalization performance (P2). Table III highlights several trends. SAC, despite showing competitive zero-shot generalization in the racetrack task, now fails to maintain performance across contexts. SAC-Switching likewise struggles to compose its single-mode policies under switching. SAC-UPTrue performs best in high-friction settings but degrades sharply in harder single- and multi-friction contexts. In contrast, SAC-MoE retains performance on increasingly

$\mathcal{M}$	SAC	SAC-UPTrue	SAC-Sw	SAC-MoE (ours)
		Single Friction		
$\{1.0\}$	37.3 / 3.12	38.8 / 4.16	**39.5 / 3.41**	32.5 / 2.71
$\{0.3\}$	4.5 / 1.72	**28.4 / 2.37**	5.0 / 0.93	28.2 / 2.36
$\{0.1\}$	6.5 / 1.33	18.0 / 1.59	13.1 / 1.26	**22.7 / 1.90**
$\{0.05\}$	6.9 / 0.96	13.8 / 1.20	7.5 / 0.84	**17.8 / 1.48**
		Multi-Friction		
$\{1.0, 0.5\}$	8.8 / 2.26	**38.1 / 3.72**	6.2 / 2.52	31.8 / 2.70
$\{0.1, 1.0\}$	5.6 / 1.35	18.1 / 2.08	8.8 / 1.42	**28.5 / 2.38**
$\{0.1, 0.3\}$	4.8 / 1.24	20.3 / 1.78	14.6 / 1.34	**25.5 / 2.14**
$\{0.05, 0.5\}$	4.5 / 0.92	14.4 / 1.59	6.3 / 0.88	**24.2 / 2.04**

TABLE III: **Quantitative results on Walker2D.** We report distance traveled (m) / torso velocity (m/s) for various models on single- and mixed-friction settings. Higher is better for both metrics; bold indicates the best *distance* per row.

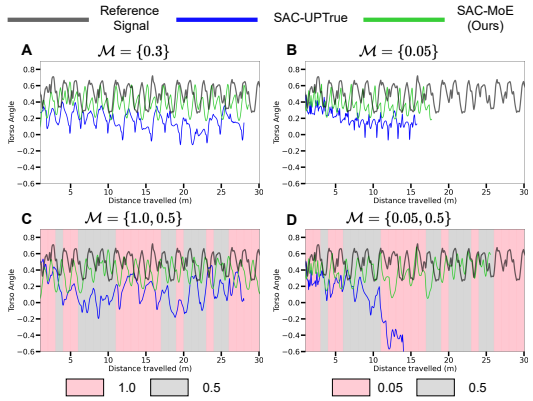

Fig. 6: **Torso-angle trajectories vs. friction context.** Comparison of SAC-UPTrue and SAC-MoE across single- and multi- friction settings. Reference signal (black) is generated by an optimal SAC policy trained and tested *only* in $\mathcal{M} = \{1.0\}$. Mode switching locations for **C**, **D** are visualized by colored boxes.

difficult novel contexts, demonstrating robustness consistent with Section V-B.

Fig. 7 further illustrates this robustness by visualizing torso angle trajectories across contexts. Episodes terminate if the torso angle leaves $[-0.8, 0.8]$ radians, making this variable critical. As a reference, we train an SAC policy in a single high-friction environment ($\mu = 1.0$) and evaluate it in the *same* environment to get a near-optimal trajectory. Across all visualized contexts, SAC-MoE tracks this reference more closely than SAC-UPTrue, thus resulting in up to 1.7x performance improvement in harder contexts.

Router expert activations and latent modes (P3). To address **P3**, we examine how router activations correlate with latent modes by tracking token assignments to experts under different friction coefficients. Fig. 7**A** shows that activations vary across friction contexts. A t-SNE projection of state observations shows that large friction differences (Fig. 7**B**) yield distinct state distributions and correspondingly different expert activations, while similar friction (Fig. 7**C**) produces overlapping distributions and similar activation patterns. In harder (lower-friction) contexts, activations spread across more experts, highlighting the benefit of the MoE router's adaptive composition advantage over π_{sw} (Section IV-A).

VI. Conclusion

We formalize the problem of learning policies for hybrid dynamical systems under uncertainty by introducing the

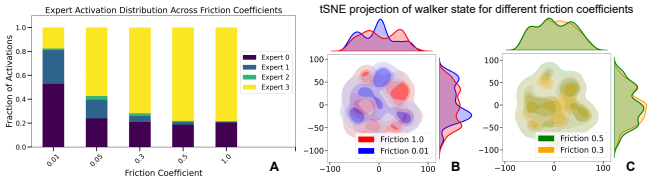

Fig. 7: **Correlation between expert activation and friction coefficient. (A)** We show the distribution of expert activations across friction parameters. **(B,C)** We compute a t-SNE projection of the agent's state vectors over an episode and plot their densities for different friction coefficients.

Hybrid Contextual Markov Decision Process (HcMDP). We highlight the challenges of controlling such systems with switching-based policies and address them by proposing SAC-MoE capable of adaptively composing sub-policies. We develop a curriculum learning approach for systematic training across contexts, which consistently improves performance across all evaluated models. Through extensive simulations on autonomous racing and locomotion tasks, we show that MoE approach consistently outperforms baselines in zero-shot generalization to novel contexts.

Limitations. A key limitation of our zero-shot generalization setting is the requirement on the policy to adapt to unseen contexts during a single episode, despite observations early in the episode provide little information about potential future mode transitions. This makes our policy more conservative than necessary due to operation under uncertainty. Moreover while our adaptive curriculum improves training in finite context sets, scaling such approaches to larger context spaces remains a significant problem under the difficulties generated by HcMDPs highlighted in Section IV-C.

Future work. Future work could involve considering a few-shot generalization setting to allow use of a latent context encoder by progressively refining context estimates across multiple episodes. This would serve to reduce policy conservatism by reducing context uncertainty while better capturing relations across the context space. The above also allows better curriculum strategies that use latent encodings to capture similarities between contexts. This allows for treating contexts as more than just elements in $\mathcal{C}$ and addresses the problem highlighted in Section IV-C.

References

[1] M. Althoff, M. Koschi, and S. Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017.

[2] P. J. Antsaklis and X. D. Koutsoukos. Hybrid systems: Review and recent progress. *Software-Enabled Control: Information Technology for Dynamical Systems*, 2003.

[3] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, 2017.

[4] C. Benjamins, T. Eimer, F. Schubert, A. Mohan, S. Döhler, A. Biedenkapp, B. Rosenhahn, F. Hutter, and M. Lindauer. Contextualize me – the case for context in reinforcement learning. *Transactions on Machine Learning Research*, 2023.

[5] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[6] M. Buss, M. Glocker, M. Hardt, O. von Stryk, R. Bulirsch, and G. Schmidt. Nonlinear hybrid dynamical systems: Modeling, optimal control, and applications. In *Modelling, Analysis, and Design of Hybrid Systems*, 2002.

[7] R. Calandra, A. Seyfarth, J. Peters, and M. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 2015.

[8] E. F. Camacho, D. R. Ramírez, D. Limón, D. M. De La Peña, and T. Alamo. Model predictive control techniques for hybrid systems. *Annual reviews in control*, 2010.

[9] G. Cheng, L. Dong, W. Cai, and C. Sun. Multi-task reinforcement learning with attention-based mixture of experts. *IEEE Robotics and Automation Letters*, 2023.

[10] L. D'Souza, Y. V. Pant, and S. Fischmeister. Tractable stochastic hybrid model predictive control using gaussian processes for repetitive tasks in unseen environments. *arXiv preprint arXiv:2508.18203*, 2025.

[11] L. D'Souza and Y. V. Pant. Stochastic hybrid model predictive control using gaussian processes for systems with piecewise residual dynamics. In *American Control Conference (ACC)*, 2023.

[12] J. Grigsby, J. Fan, and Y. Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. In , *International Conference on Representation Learning*, 2024.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 2018.

[14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[15] A. Hallak, D. Di Castro, and S. Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[16] H. Hasselt. Double q-learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[17] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 1991.

[18] J. Kwon, Y. Efroni, C. Caramanis, and S. Mannor. Rl for latent mdps: Regret guarantees and a lower bound. *Advances in Neural Information Processing Systems*, 2021.

[19] M. Lazar, W. Heemels, S. Weiland, and A. Bemporad. Stabilizing model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 2006.

[20] E. Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

[21] H. Liu, S. Teng, B. Liu, W. Zhang, and M. Ghaffari. Discrete-time hybrid automata learning: Legged locomotion meets skateboarding. In *Robotics: Science and Systems*, 2025.

[22] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 2024.

[23] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In *Conference on Robot Learning*. PMLR, 2020.

[24] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 2018.

[25] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 2020.

[26] T. C. NDIR, A. Biedenkapp, and N. Awad. Inferring behavior-specific context improves zero-shot generalization in reinforcement learning. In *Seventeenth European Workshop on Reinforcement Learning*, 2024.

[27] M. Poli, S. Massaroli, L. Scimeca, S. Chun, S. J. Oh, A. Yamashita, H. Asama, J. Park, and A. Garg. Neural hybrid automata: Learning dynamics with multiple modes and stochastic transitions. In , *Advances in Neural Information Processing Systems*, 2021.

[28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[29] J. Ren, Y. Li, Z. Ding, W. Pan, and H. Dong. Probabilistic mixture-of-experts for efficient deep reinforcement learning. *arXiv preprint arXiv:2104.09122*, 2021.

[30] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[31] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[32] X. Wu, S. Huang, W. Wang, S. Ma, L. Dong, and F. Wei. Multi-head mixture-of-experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[33] W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017.