H-GenPO: Hierarchical Generative Policy Optimization via the Option-critic Framework

Wonhyeok Choi[1], **Minwoo Choi**[1], **Jaeyeul Kim**[2],
Kyumin Hwang[3], **Jongmin Gim**[3], **Sunghoon Im**[3,†]
DGIST[1], Pukyong National University[2], KAIST[3]
South Korea
{smu06117, subminu}@dgist.ac.kr[1], jykim@pknu.ac.kr[2],
{jongmin17, im}@kaist.ac.kr[3]

Abstract

Diffusion-based policies have demonstrated remarkable performance in online reinforcement learning (RL) by virtue of their expressive, multimodal action distributions. However, existing methods operate on a flat architecture that must generate a new action at every timestep, with no mechanism for temporal commitment to a behavioral mode—making them ill-suited for context-dependent tasks that require sustaining coherent behavioral modes, regardless of how expressive their action distributions are. We propose Hierarchical Generative Policy Optimization (H-GenPO), a framework that integrates on-policy diffusion-based RL with the temporal abstraction of the Option-Critic architecture. H-GenPO adopts a two-level hierarchy in which a high-level policy over options selects among a discrete set of options, while a unified flow matching model—conditioned on a learned option embedding—serves as the expressive low-level intra-option primitive. The shared intra-option policy design encapsulates diverse behavioral repertoires within a single set of parameters, enabling scalable hierarchical control without the parameter growth associated with maintaining separate policy networks per option. We evaluate H-GenPO on 8 standard continuous control tasks in Isaac Lab and 3 custom context-dependent tasks designed to require coordinated behavioral mode switching. Our empirical evaluation demonstrates that H-GenPO achieves the best mean rank among all baselines on both standard and context-dependent benchmarks, while also exhibiting interpretable option specialization that emerges without any option-level supervision.

1 Introduction

Diffusion policies have emerged as a promising approach for continuous robotic control, owing to their ability to represent complex, multimodal action distributions [7, 5, 19]. These generative models have recently garnered significant attention within the online RL community as a powerful alternative to traditional unimodal Gaussian policies [38]. Recent works bridging online RL and diffusion-based generative models [49, 9, 10] have demonstrated that diffusion-based policies [17, 28] can significantly outperform conventional RL baselines [27, 15, 38], establishing them as a compelling tool for sequential decision-making in diverse robotic tasks [44, 32].

Despite these advances, existing diffusion-based methods typically operate on a *flat* (i.e., *non-hierarchical*) policy architecture that selects primitive actions independently at every timestep, with no explicit mechanism to commit to or maintain a behavioral mode over extended horizons. Such a

[†]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

structural gap prevents the agent from leveraging temporal abstraction, which is critical in context-dependent tasks that demand diverse behavioral regimes [43]: a quadruped robot may need to switch between rapid sprinting in open areas and cautious, high-precision stepping through hazard-prone passages. Yet even an expressive diffusion-based policy must generate a new action without temporal context, and therefore cannot sustain a coherent behavioral mode—regardless of how rich its action distribution is. Action-level expressiveness and temporal behavioral commitment are fundamentally distinct: the former determines how rich the action distribution is at a single step, while the latter governs how consistently a behavioral strategy is maintained over time. This motivates integrating temporal abstraction into diffusion-based policies, enabling the agent to select and sustain appropriate behavioral modes as the situation demands.

In this work, we propose **H**ierarchical **Gen**erative **P**olicy **O**ptimization (**H-GenPO**), a framework that integrates flow-matching-based diffusion policies with end-to-end Hierarchical Reinforcement Learning (HRL) [43, 2, 16, 22], building on the on-policy diffusion RL framework of GenPO [10]. H-GenPO assigns complementary roles to each level of the hierarchy. The option structure provides temporal commitment: a high-level policy selects a behavioral mode and sustains it over extended horizons, decoupling strategic decision-making from low-level execution. Within each option, a flow-matching model—conditioned on a learned option embedding—provides the action-level expressiveness that standard Gaussian intra-option policies lack, enabling each option to represent rich, diverse behaviors rather than a single stereotyped primitive. This combination addresses what neither component achieves alone: a flat diffusion policy offers expressiveness but no temporal structure, while traditional HRL offers structure but limited per-option expressiveness. Unlike prior HRL methods that maintain separate policy networks per option [2, 16, 22], H-GenPO shares a single flow network across all options, distinguished solely by lightweight learned embeddings.

We evaluate the proposed H-GenPO on 8 standard tasks in Isaac Lab [32] and 3 additional tasks specifically designed to challenge hierarchical reasoning. Our results demonstrate that our approach achieves the best mean rank across all evaluated baselines, spanning traditional Gaussian RL [38, 15], non-hierarchical diffusion-based RL [49, 9, 47, 10], and traditional HRL [22]. Our main contributions are as follows:

- We identify action-level expressiveness and temporal behavioral commitment as distinct properties, and propose H-GenPO, which integrates on-policy flow-based RL with end-to-end option learning to provide both.
- We introduce an option-conditioned flow policy that injects learned option embeddings outside the invertible path of GenPO's normalizing flow, yielding option-specific multimodal action distributions within a single shared network while preserving the exact log-likelihood required for PPO-style optimization.
- We provide systematic experimental results on 8 standard and 3 custom context-dependent Isaac Lab tasks, the latter requiring different behavioral regimes, showing that H-GenPO achieves the best overall performance compared to a comprehensive set of online RL baselines.

2 Related Work

2.1 Diffusion Models as Policy Representations

Denoising diffusion probabilistic models (DDPMs) [17] and flow matching [28] have established themselves as powerful generative frameworks capable of modeling complex, multimodal distributions. Their application to robotic control has been pioneered by Diffusion Policy [7], which demonstrates that a diffusion-based behavioral cloning objective substantially outperforms a wide range of prior policy representations [1, 13, 39] on various robotic tasks. Subsequent works extended this idea to trajectory-level planning [19, 5], confirming the broad utility of generative models as expressive policy representations. Diffusion models have also been adopted in offline RL settings, where they serve as expressive policy classes for behavior regularization and Q-guided policy improvement [48, 11], demonstrating strong performance on standard offline benchmarks. Despite these advances, policies trained in imitation learning or offline RL settings are fundamentally bounded by the quality and coverage of the static dataset [8]; they cannot discover behaviors that surpass the demonstrations or explore beyond the support of the pre-collected data, necessitating an extension toward online RL.

2.2 Online Reinforcement Learning with Diffusion Policies

Extending diffusion policies to the online RL setting is non-trivial due to two fundamental challenges [8, 14]: (1) backpropagating policy gradients through the long reverse diffusion chain incurs prohibitive memory and computation costs, and (2) log-likelihood evaluation of sampled actions is intractable under standard diffusion formulations. Recent works address these issues via reduced denoising steps [47, 6] or Q-value weighting [9, 29] and action gradients [35, 49]. Proximity-based methods [10, 31] leverage PPO [38] by obtaining exact or approximate likelihoods through normalizing flows [46] or variational bounds [21]. Among these, GenPO [10] stands out by fully leveraging the on-policy advantage of cheap, parallelized sample collection in high-throughput robotic simulation [32], achieving superior performance across a range of continuous control benchmarks. Despite these advances, all of the above methods share a common structural limitation: they operate on a *flat* policy architecture that selects primitive actions at every timestep [43, 2], and therefore lack an explicit mechanism to commit to or sustain distinct behavioral modes across extended horizons in response to changing environmental contexts.

2.3 Hierarchical Reinforcement Learning

Hierarchical Reinforcement Learning (HRL) addresses long-horizon decision-making by decomposing a task into a high-level controller that selects sub-goals or behavioral modes and a low-level controller that executes primitive actions [43, 45]. Goal-conditioned approaches such as HIRO [33], HAC [26], and FuN [45] train a manager policy to output sub-goals consumed by a worker, but require hand-crafted sub-goal spaces or task-specific hyperparameter tuning, limiting their applicability when meaningful sub-goals are difficult to specify. The options framework [43] eliminates this requirement by allowing the hierarchical structure to emerge naturally through learning: each option is defined by an intra-option policy and a termination condition, forming a principled semi-MDP without sub-goal specification. The Option-Critic architecture [2] derives end-to-end policy gradient theorems for both components, enabling fully differentiable hierarchical learning. Subsequent works extended this to off-policy settings via importance correction [20, 24, 41], broadening its applicability to more sample-efficient regimes. PPOC [22] brought Option-Critic into the on-policy PPO regime, and its compatibility with parallelized simulation makes it particularly well-suited for high-throughput robotic training. Complementary lines of work have advanced how options are discovered and used, e.g., successor-representation and Laplacian-based temporal abstractions for exploration [30, 23], learned initiation sets [3], and unsupervised skill discovery [12, 34]. However, these methods focus on *which* options to learn and *when* to invoke them, and typically realize each option, whether as a separate network or a skill-conditioned shared one, with a traditional intra-option policy such as a Gaussian, which lacks the expressiveness to capture diverse behavioral repertoires in complex, context-dependent tasks. H-GenPO addresses the expressiveness limitation of existing HRL methods while preserving the end-to-end learnability of Option-Critic: it replaces the Gaussian intra-option policy with a generative flow model, endowing each option with the full expressive power of on-policy diffusion RL.

3 Preliminaries

3.1 On-Policy Reinforcement Learning and PPO

We formalize the RL problem as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $p(s' \mid s, a)$ is the transition dynamics, $r(s, a)$ is the reward function, ρ_0 is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. The goal is to maximize the expected discounted return $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ denotes a trajectory induced by policy π_θ.

Proximal Policy Optimization (PPO) [38] stabilizes on-policy updates via a surrogate objective:

$$\mathcal{L}_{\mathrm{CLIP}}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_\theta^{\mathrm{old}}}\Big[-\min\Big(\rho_t(\theta)\,\hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\Big)\Big], \qquad \rho_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_\theta^{\mathrm{old}}(a_t \mid s_t), \tag{1}$$

where $\hat{A}_t$ is the Generalized Advantage Estimate (GAE) [37] computed by $\hat{A}_t = \sum_{l=0}^{T-t}(\gamma\kappa)^l \delta_{t+l}$, truncated at rollout horizon T in practice. $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ denotes the TD residual, where V is a value function, and $\kappa \in [0, 1]$ controls the bias-variance trade-off.

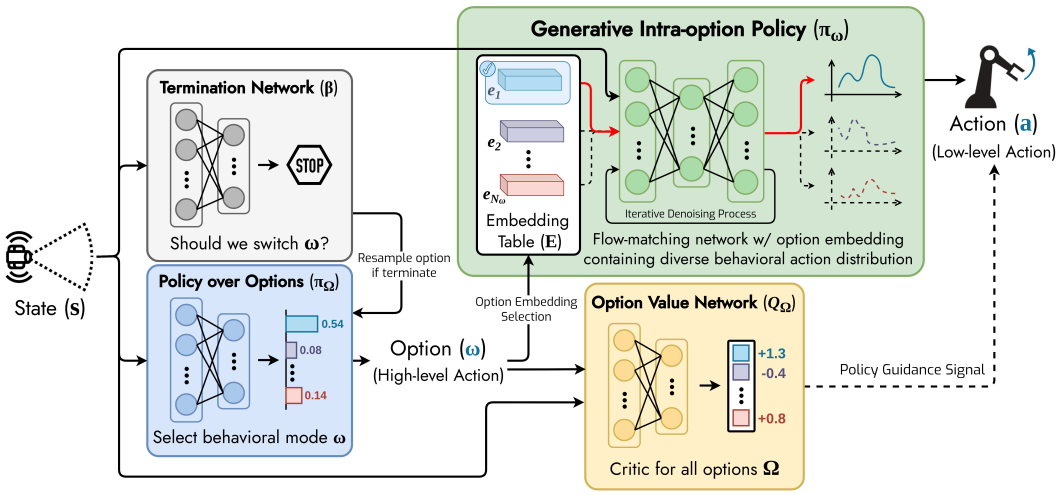


Figure 1: Overall pipeline of the proposed H-GenPO. Four modules operate in a closed loop: a high-level *policy over options* π_Ω selects an option ω, representing a specific behavioral mode; a low-level *intra-option policy* π_ω executes actions under the option ω; a *termination network* β decides when to switch to a new mode; and an *option-value network* Q_Ω evaluates the value of all options.

3.2 GenPO: Tractable Diffusion Policy for On-Policy RL

For self-containedness, we briefly recall the necessary background on GenPO. GenPO [10] resolves intractable computation of exact likelihood by constructing the policy as a normalizing flow [46]: a parameterized procedure f_θ that maps Gaussian latent noise $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to an action $a = f_\theta(z)$, such that its inverse f_θ^{-1} is computable. This inverse function can recover the original latent noise $z = f_\theta^{-1}(a)$ for any sampled action a, and the likelihood of a under the policy is computed directly via the change-of-variables rule [36] as follows:

$$\pi_\theta(a \mid s) = \mathcal{N}\big(f_\theta^{-1}(a) \mid \mathbf{0}, \mathbf{I}\big) \left| \det \frac{\partial f_\theta^{-1}}{\partial a} \right|. \tag{2}$$

With exact likelihoods available, GenPO optimizes the combined policy loss as follows:

$$\mathcal{L}_{\text{GenPO}}(\theta) = \mathcal{L}_{\text{CLIP}} + \lambda_{\text{ent}}\, \mathcal{L}_{\text{ent}} + \lambda_{\text{comp}}\, \mathcal{L}_{\text{comp}}, \tag{3}$$

where $\mathcal{L}_{\text{ent}} = \mathbb{E}[\log \pi_\theta(a \mid s)]$ is an entropy penalty that encourages exploration, and $\mathcal{L}_{\text{comp}}$ is a compression loss that prevents degenerate solutions in the latent space. We refer the reader to Appendix F-G for the full derivation of the invertible flow architecture and the Jacobian computation.

4 Method

We integrate on-policy diffusion-based RL with the Option-Critic framework for temporal abstraction, yielding H-GenPO. Built on the high-throughput strengths of parallelized simulation, H-GenPO jointly learns when to switch behavioral modes and how to execute them with the full expressiveness of a flow-matching policy. We describe the architectural design (Section 4.1), data collection (Section 4.2), and the training objectives (Section 4.3). Figure 1 provides an overview of the full architecture.

4.1 Architectural Design of H-GenPO

Policy over options. $\pi_\Omega(\cdot|s;\phi)$ is a categorical MLP that selects a discrete behavioral mode and commits to it over an extended horizon, allowing the agent to pursue context-specific strategies without re-selecting at every timestep. It outputs a probability distribution over N_ω discrete options via a softmax layer, and at the start of each episode and upon every option termination, it samples the active option $\omega \sim \pi_\Omega(\cdot \mid s;\phi)$ to be executed by the flow-based intra-option policy.

Intra-option policy. $\pi_\omega(\cdot|s, e_\omega; \theta)$ is an invertible flow-matching policy following GenPO [10, 46] that covers all specific behavioral modes with parameters shared across all options $\Omega = \{\omega\}$. Each option is distinguished solely by a learnable option embedding $e_\omega \in \mathbb{R}^{d_e}$ drawn from an embedding table $\mathbf{E}_\vartheta = \{e_1, \cdots, e_{N_\omega}\}$ parameterized by ϑ. The option embedding e_ω, whose dimension d_e is set to the action dimension (App. B), is concatenated with the state s and fed to the velocity network v_θ of the flow at every denoising step (App. F.3), conditioning the action distribution on the selected behavioral mode without duplicating network parameters. This shared design reduces parameter count from $O(N_\omega \cdot |\theta|)$ to $O(|\theta| + N_\omega \cdot d_e)$: since $|\theta| \gg d_e$ in practice, model size remains nearly constant as N_ω grows. It also promotes transfer of low-level motor skills across options, unlike separate per-option networks. For brevity, we detail the action sampling procedure which admits $a \sim \pi_\omega(\cdot \mid s, e_\omega; \theta)$ in the Appendix F.3.

Termination network. $\beta(s, \omega; \varphi) \in [0, 1]$ is an MLP that predicts the probability of terminating the current option ω upon arriving at state s, allowing the agent to adaptively switch behavioral modes in response to the current state. At every timestep, a termination event is sampled from $\mathrm{Bernoulli}(\beta(s_{t+1}, \omega_t))$ (Eq. (4)): upon termination, π_Ω re-selects an option, and otherwise the current option is retained. Together with π_Ω, β therefore determines how long each behavioral mode is sustained, and it is trained to lower the termination probability whenever the current option is more valuable than the average over options (Sec. 4.3).

Option-value network. $Q_\Omega(s, \omega; \psi)$ is an MLP that predicts option-values for all options given state s. Unlike prior works that maintain fully separate parameters per option [22, 24], we adopt a shared-bottom architecture in which a common trunk processes the state and a final layer produces values for all N_ω options simultaneously, reducing computational overhead while preserving expressive capacity.

4.2 Data Collection and Experience Rollout

Intra-option policy sampling. At each timestep t, the intra-option policy π_ω conditions on the state s_t and a learnable option-specific embedding e_{ω_t} in the option embedding table $\mathbf{E}_\vartheta$ to sample an action $a_t \sim \pi_{\omega_t}(\cdot \mid s_t, e_{\omega_t})$, and the exact log-likelihood $\log \pi_\omega(a_t|s_t, e_{\omega_t})$ is computed via Eq. (2). It is worth noting that the option embedding e_{ω_t} is a *deterministic* function of the selected option ω_t and does not participate in the invertible computation, so the exact log-likelihood derivation of Eq. (2) carries over unchanged to the conditioned policy.

Option switching. After executing a_t in the environment and observing reward r_t and next state s_{t+1}, the termination network decides whether to switch the current option ω_t at the next state s_{t+1}:

$$\omega_{t+1} = \begin{cases} \omega_t, & u_t \geq \beta(s_{t+1}, \omega_t), \\ \omega' \sim \pi_\Omega(\cdot \mid s_{t+1}), & u_t < \beta(s_{t+1}, \omega_t), \end{cases} \quad u_t \sim \mathrm{Uniform}(0, 1). \tag{4}$$

To prevent option chattering—a degenerate behavior in which the agent switches options at every timestep and the hierarchy reduces to a flat policy—a deliberation cost $\eta > 0$ is subtracted from the raw reward following [16]:

$$\tilde{r}_t = r_t - \eta \cdot \mathbb{1}[u_t < \beta(s_t, \omega_{t-1})], \quad u_t \sim \mathrm{Uniform}(0, 1), \tag{5}$$

where $\mathbb{1}$ is the function indicating whether the current option ω_t was freshly sampled upon arriving at s_t. By penalizing each option switch, the deliberation cost discourages frequent termination and promotes sustained commitment to a single behavioral mode, preserving the temporal structure that hierarchical control is designed to provide.

Advantage computation. To evaluate and train the intra-option policy π_ω, policy over options π_Ω, and termination network β, we compute three advantage estimates from critic Q_Ω: *intra-option advantage* $\hat{A}_t$, *option-level advantage* $\hat{A}_t^U$, and *termination advantage* A_t^β.

The intra-option advantage $\hat{A}_t$ is computed via GAE [37] using the penalized reward $\tilde{r}$ and the option-value $Q_\Omega(s_t, \omega)$ as the baseline:

$$\hat{A}_t = \mathrm{GAE}(\tilde{r}, Q_\Omega(\cdot, \omega);\, \gamma, \kappa)\,, \tag{6}$$

where $\kappa \in [0, 1]$ controls the bias-variance trade-off of the advantage estimate.

For the option-level advantage, we first compute the upon-arrival value [43, 2], which estimates the expected return of committing to option ω_{t-1} upon entering state s_t before any termination decision is made:

$$U(s_t, \omega_{t-1}) = \big(1 - \beta(s_t, \omega_{t-1})\big)\, Q_\Omega(s_t, \omega_{t-1}) + \beta(s_t, \omega_{t-1})\, V_\Omega(s_t), \tag{7}$$

where $V_\Omega(s_t) = \sum_\omega \pi_\Omega(\omega \mid s_t)\, Q_\Omega(s_t, \omega)$ is the marginal state value representing the average expected return across all options. The option-level advantage $\hat{A}_t^U$ is then computed via GAE using $U(s_t, \omega)$ as the baseline, which accounts for the expected cost of switching:

$$\hat{A}_t^U = \mathrm{GAE}(\tilde{r}, U(\cdot, \omega);\, \gamma, \kappa)\,. \tag{8}$$

Note that, unlike PPOC [22], which reuses the intra-option advantage to update π_Ω, $\hat{A}_t^U$ evaluates option selection as an SMDP-level decision against the value realized upon arrival, avoiding entangled credit between the two levels.

The *termination advantage* A_t^β, by contrast, does not use GAE: unlike the intra-option and option-level advantages, which benefit from multi-step return estimation to reduce variance in action and option selection, the termination decision requires only an instantaneous comparison of whether the current option outperforms the marginal state value $V_\Omega(s_t)$, as in Option-Critic [2]:

$$A_t^\beta = Q_\Omega(s_t, \omega_t) - V_\Omega(s_t), \tag{9}$$

4.3 Training Objective

Using the advantage estimates and buffered quantities $(s_t, a_t, \tilde{r}_t, \omega_t, \log \pi_\omega^{\mathrm{old}}(a_t|s_t, \omega_t))$ in rollout buffer $\mathcal{B}$ collected in Sec. 4.2, we optimize the following losses:

Policy over options loss. The option policy is updated with a REINFORCE-style log-likelihood objective [42, 22] augmented with an entropy regularizer as follows:

$$\mathcal{L}_\Omega(\phi) = -\mathbb{E}_{(s_t, \omega_t)\sim\mathcal{B}}\Big[\log \pi_\Omega(\omega_t \mid s_t; \phi)\, \hat{A}_t^U\Big] - \lambda_{\mathcal{H}}\, \mathcal{H}[\pi_\Omega(\cdot \mid s_t; \phi)]\,, \tag{10}$$

where $\hat{A}_t^U$ is the unnormalized GAE advantage from Eq. (8), $\mathcal{H}[\pi_\Omega(\cdot \mid s_t; \phi)] = -\sum_\omega \pi_\Omega(\omega \mid s_t; \phi) \log \pi_\Omega(\omega \mid s_t; \phi)$ is the entropy of the option distribution that encourages diverse option usage and prevents premature collapse to a single mode, and $\lambda_{\mathcal{H}}$ is the entropy regularization coefficient.

Intra-option policy loss. We compute $\pi_\omega(a_t \mid s_t, e_{\omega_t}; \theta)$ from the current policy and form the importance ratio against the buffered old intra-option policy likelihood $\pi_\omega^{\mathrm{old}}$. The intra-option loss extends the GenPO clipped surrogate (Eq. (3)):

$$\begin{aligned} \mathcal{L}_\omega(\theta) &= \mathbb{E}_{(s_t, a_t, \omega_t)\sim\mathcal{B}}\Big[-\min\Big(\rho_t(\theta)\, \hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1{-}\epsilon, 1{+}\epsilon)\, \hat{A}_t\Big)\Big]\,, \\ \rho_t(\theta) &= \frac{\pi_\omega(a_t \mid s_t, e_{\omega_t}; \theta)}{\pi_\omega^{\mathrm{old}}(a_t \mid s_t, e_{\omega_t})}, \end{aligned} \tag{11}$$

where $\hat{A}_t$ is the normalized GAE advantage from Eq. (6), and the option embedding e_{ω_t} is retrieved from the embedding table $\mathbf{E}_\vartheta$ corresponding to option ω_t. Since the embeddings constitute the learnable parameter set ϑ, they are optimized jointly with the flow parameters θ through this objective, so that each option's conditioning signal is shaped by the intra-option policy gradient.

Termination loss. The termination network is updated to reduce termination probability when the termination advantage A_t^β is positive, with the deliberation cost η from Eq. (5) acting as a termination margin [16], encouraging longer option commitments:

$$\mathcal{L}_\beta(\varphi) = \mathbb{E}_{(s_t, \omega_t)\sim\mathcal{B}}\Big[\beta(s_t, \omega_t; \varphi)\, \big(A_t^\beta + \eta\big)\Big]\,. \tag{12}$$

Note that, unlike the GAE-based advantages, A_t^β is recomputed from the current option-value network $Q_\Omega(s_t, \omega_t; \psi)$ at every mini-batch rather than being taken from the buffer, ensuring consistency with the latest value estimates as ψ is jointly updated within each optimization step.

Critic loss. $Q_\Omega(s, \omega; \psi)$ is trained by minimizing the MSE against GAE-based return estimates $\hat{G}_t$:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{(s_t, \omega_t)\sim\mathcal{B}}\Big[\big(Q_\Omega(s_t, \omega_t; \psi) - \hat{G}_t\big)^2\Big]\,. \tag{13}$$

Algorithm 1 H-GenPO: Hierarchical Generative Policy Optimization

Require: Policy over options $\pi_\Omega(\cdot|s;\phi)$, intra-option policy $\pi_\omega(\cdot|s,e_\omega;\theta)$, termination network $\beta(s,\omega;\varphi)$, option-value network $Q_\Omega(s,\omega;\psi)$, option embedding table $\mathbf{E}_\vartheta = \{e_1,\cdots,e_{N_\omega}\}$.

Initialize s_0; $\omega_0 \sim \pi_\Omega(\cdot|s_0)$, $\beta_0 \leftarrow 0$
for $t = 0, 1, \ldots, T$ **do**
  // — Rollout collection —
  for each actor **do**
    Sample $a_t \sim \pi_\omega(\cdot \mid s_t, e_{\omega_t})$ and compute $\log \pi_\omega(a_t|s_t, e_{\omega_t})$ by Eq. (2)
    Execute a_t; observe $r_t,\ s_{t+1}$
    Sample $\beta_{t+1} \sim \text{Bernoulli}(\beta(s_{t+1}, \omega_t))$
    Compute ω_{t+1} by Eq. (4)
    Compute penalized reward $\tilde{r}_t$ by Eq. (5)
    Store $(s_t,\ a_t,\ \omega_t,\ \tilde{r}_t,\ \log \pi_\omega(a_t|s_t, e_{\omega_t}))$ in the rollout buffer $\mathcal{B}$
  end for
  Compute $\hat{A}_t$ and $\hat{A}_t^U$ by Eq. (6)-(8)
  // — Policy update —
  for $k = 1, 2, \ldots, K$ **do**
    Compute A_t^β by Eq. (9)
    Update $\theta,\ \phi,\ \varphi,\ \psi,\ \vartheta$ by minimizing $\mathcal{L}_{\text{total}}$ from Eq. (14)
  end for
end for

Joint optimization. All losses are combined into a single weighted objective with $\mathcal{L}_{\text{ent}}$ and $\mathcal{L}_{\text{comp}}$ taken from Eq. (3) and optimized jointly per mini-batch:

$$\mathcal{L}_{\text{total}} = \lambda_\Omega\, \mathcal{L}_\Omega(\phi) + \lambda_\omega\, \mathcal{L}_\omega(\theta) + \lambda_\beta\, \mathcal{L}_\beta(\varphi) + \lambda_Q\, \mathcal{L}_Q(\psi) + \lambda_{\text{ent}}\, \mathcal{L}_{\text{ent}}(\theta) + \lambda_{\text{comp}}\, \mathcal{L}_{\text{comp}}(\theta), \quad (14)$$

where $\lambda_{\{\cdot\}}$ denote the loss scaling coefficients for each component. Following GenPO [10], we also adopt the KL-adaptive learning rate scheduler for the intra-option policy. The complete H-GenPO training procedure is summarized in Algorithm 1.

5 Experiments

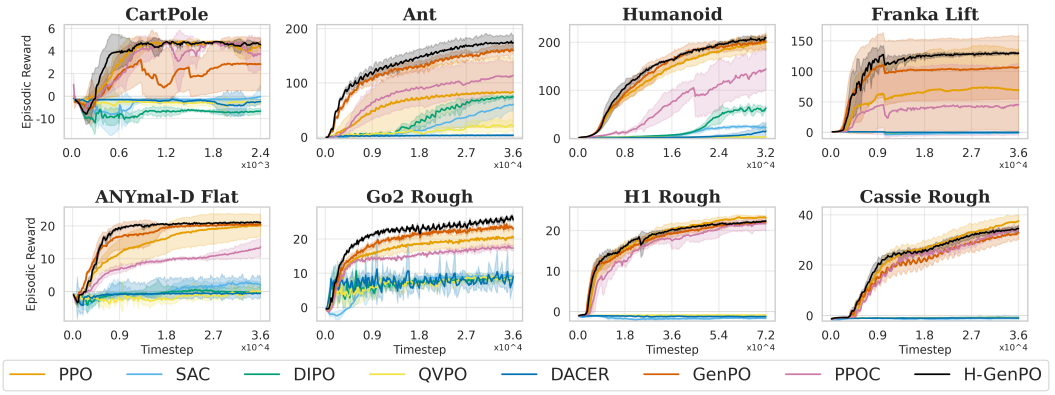
5.1 Experimental Setup

We evaluate H-GenPO on two sets of tasks implemented in Isaac Lab [32], a GPU-accelerated parallel simulation framework that enables high-throughput data collection essential for on-policy methods.

Standard tasks. We select 8 standard tasks spanning three categories: classic control (`CartPole`, `Ant`, `Humanoid`), manipulation (`Franka Lift`), and legged locomotion (`ANYmal-D Flat`, `Unitree Go2 Rough`, `H1 Rough`, `Cassie Rough`). These tasks provide a broad benchmark for evaluating general continuous control performance across diverse embodiments and task complexities.

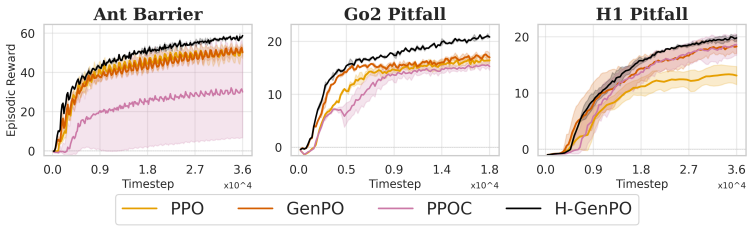
Context-dependent tasks. To specifically challenge hierarchical reasoning and multimodal behavior, we design 3 custom tasks that require the agent to coordinate distinct behavioral modes in response to environmental context. `Ant Barrier` extends the standard Ant locomotion task by equipping the agent with lidar exteroception and placing pillars throughout the environment, requiring the agent to switch between fast forward locomotion in open areas and evasive maneuvering to navigate around obstacles. `Unitree Go2 Pitfall` and `H1 Pitfall` augment the standard rough terrain locomotion tasks by adding negative hole terrain along the agent's path of travel, requiring the agent to detect and avoid falling into pits while still following velocity commands (refer to Appendix A for details).

Baselines. We compare H-GenPO against a comprehensive set of baselines spanning three categories. *Traditional RL*: PPO [38] and SAC [15], representing strong on-policy and off-policy Gaussian policy baselines respectively; *Diffusion-based RL*: DIPO [49], QVPO [9], DACER [47], and GenPO [10], covering the major paradigms of online RL with generative policies; *Hierarchical RL*: PPOC [22], the on-policy extension of Option-Critic that serves as the direct hierarchical baseline. All methods are trained with the same backbone architecture and simulation budget for fair comparison, and the H-GenPO-specific loss weights ($\lambda_\Omega, \lambda_\omega, \lambda_\beta, \lambda_\mathcal{H}$ in Eq. (14)) are fixed across all tasks, while



	Methods/Envs	CartPole	Ant	Humanoid	Franka	ANYmal-D	Go2	H1	Cassie	MR. ↓
RL	PPO [38]	4.38±0.45	83.08±71.96	200.29±19.72	69.33±75.57	20.10±3.68	20.38±2.83	**23.11±0.52**	**37.51±2.66**	2.50
	SAC [15]	-0.53±1.08	60.60±12.52	22.47±2.18	-5.96±1.50	2.03±0.58	9.17±1.05	-8.09±0.45	-4.76±4.12	6.25
Diff. RL	DIPO [49]	-6.80±1.41	73.84±3.58	62.60±6.16	0.13±0.22	-0.82±2.60	8.75±0.73	-4.71±0.50	-4.31±0.82	6.00
	QVPO [9]	-2.65±1.45	19.73±12.32	2.51±0.41	-0.96±1.99	-0.80±3.74	8.61±1.00	-4.22±0.11	-4.73±1.28	6.75
	DACER [47]	-2.61±1.46	3.51±1.64	14.88±21.03	-0.12±0.08	-2.36±7.03	8.96±2.39	-6.11±1.85	-5.10±0.83	7.00
	GenPO [10]	2.84±2.63	159.88±10.71	202.25±19.24	106.73±58.12	20.37±0.27	23.02±0.66	22.10±0.57	32.83±2.84	2.62
HRL	PPOC [22]	3.67±1.71	113.77±30.95	144.49±49.33	45.69±81.75	13.56±3.42	17.32±1.12	21.66±2.23	34.47±2.10	3.50
	H-GenPO (Ours)	**4.66 ± 0.11**	**173.94 ± 16.00**	**209.48±6.44**	**129.86±1.91**	**20.98±0.22**	**25.60±0.68**	22.32±0.23	34.46±1.53	**1.38**

Figure 2: Comparison of average final rewards and learning curves on 8 Isaac Lab standard tasks. The mean and standard deviation of 5 runs are summarized, with shaded regions in the curves indicating the standard deviation. **MR.** denotes the mean rank across all tasks. (**Best**, second-best)



Methods	Ant-B	Go2-P	H1-P
PPO	50.19 ±3.48	16.42 ±0.72	13.11 ±1.86
GenPO	50.35 ±2.29	20.97 ±0.16	18.28 ±1.72
PPOC	30.09 ±33.09	15.28 ±1.03	18.57 ±2.13
H-GenPO	**58.78** ±0.30	**22.84** ±0.67	**19.80** ±0.85

Figure 3: Learning curves and final rewards (mean±std over 5 seeds) on 3 custom context-dependent tasks (Ant Barrier, Go2 Pitfall, H1 Pitfall).

λ_Q and λ_{ent} follow the on-policy settings shared with the baselines (implementation details and hyperparameter setup are provided in Appendix B).

5.2 Evaluation Results

Standard benchmarks. Fig. 2 shows the learning curves across all standard 8 Isaac Lab tasks. A consistent pattern emerges: on-policy methods (PPO, GenPO, PPOC, H-GenPO) generally outperform off-policy methods (SAC, DIPO, QVPO, DACER). Aligned with the analysis in [8, 10], on-policy frameworks benefit from synchronized data collection, ensuring stability in large-scale parallel settings, while off-policy baselines—especially diffusion-based ones—struggle to track the rapidly shifting data distributions within the replay buffer.

Among on-policy methods, H-GenPO achieves the best mean rank of 1.38 across all 8 tasks. PPO is a strong baseline in this massively parallel regime [8, 10] (App. E); H-GenPO matches or exceeds it on six of eight tasks, while PPO remains slightly ahead on `H1` and `Cassie`. Notably, even on tasks that do not explicitly require hierarchical reasoning (e.g., `CartPole`, `Ant`), H-GenPO matches or exceeds the performance of non-hierarchical baselines, suggesting that the option structure does not impose a performance penalty when temporal abstraction is not strictly necessary. This comes at a modest cost: H-GenPO adds 8.1% training wall-time over GenPO with no measurable inference-latency overhead (App. C).

Context-dependent benchmarks. We further evaluate the on-policy baselines (PPO, GenPO, PPOC, H-GenPO) on the 3 custom context-dependent tasks, as these methods demonstrated superior

Table 1: Behavioral metrics on the context-dependent tasks, evaluated over 1,000 parallel agents per task (mean±std over 5 seeds). Early-term.: early-termination rate (%); Pit-fall: rate of episodes terminated by falling into a pit (%); Fwd. vel.: mean forward velocity (m/s); MAE v_{xy} / ω_z: mean absolute error of the commanded planar linear / yaw angular velocity. **Best** results are highlighted (see Appendix A for metric definitions).

	Ant Barrier		Go2 Pitfall				H1 Pitfall			
Methods	Early-term.↓	Fwd. vel.↑	Early-term.↓	Pit-fall↓	MAE v_{xy} ↓	MAE ω_z ↓	Early-term.↓	Pit-fall↓	MAE v_{xy} ↓	MAE ω_z ↓
PPO [38]	24.6±6.3	2.92±0.67	8.6±5.8	1.3±0.1	0.70±0.05	0.29±0.04	11.8±3.4	2.7±1.2	0.26±0.04	0.45±0.04
GenPO [10]	35.6±3.9	4.41±0.36	10.6±1.0	1.5±0.8	0.60±0.08	**0.20±0.02**	**5.7±0.5**	1.6±0.4	**0.18±0.00**	0.42±0.01
PPOC [22]	26.7±8.9	2.80±1.15	9.1±1.6	0.4±0.1	0.73±0.03	0.29±0.03	11.9±5.9	2.0±1.6	0.20±0.04	**0.30±0.11**
H-GenPO (Ours)	**22.3±2.1**	**5.25±0.21**	**7.7±1.1**	**0.1±0.0**	**0.46±0.03**	**0.20±0.01**	6.0±1.2	**0.7±0.3**	**0.18±0.01**	0.37±0.02

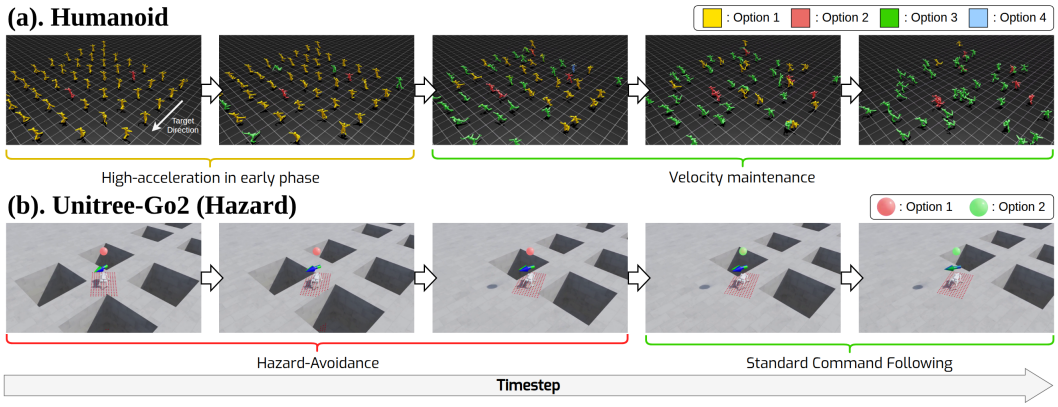


Figure 4: Qualitative visualization of emergent option behavior on Humanoid (top) and Unitree Go2 Pitfall (bottom), shown as post-hoc selected examples. Colors indicate the active option at each timestep. H-GenPO can discover semantically meaningful behavioral modes without any option-level supervision, although the degree of specialization varies across tasks and runs.

performance in the predefined task benchmark. As shown in Figure 3, H-GenPO achieves the highest final reward on all three tasks; one-sided exact Mann–Whitney U tests over 5 seeds show that the improvement is significant against all baselines on `Ant Barrier` and `Go2 Pitfall` ($p < 0.05$), whereas on `H1 Pitfall` it is significant against the flat baselines but comparable to PPOC ($p = 0.155$) while falling into pits less often. PPOC's large variance on `Ant Barrier` reflects a bimodal outcome across seeds, in which some runs fail to learn, rather than uniformly noisy performance. Table 1 further quantifies the behaviors underlying these rewards: H-GenPO attains the lowest pit-fall rate on both pitfall tasks with the lowest or tied-lowest planar velocity-tracking error, and on `Ant Barrier` it is simultaneously the fastest and the least prone to early termination, whereas GenPO trades a higher early-termination rate for speed. These results highlight the limitations of flat policy architectures in context-dependent settings: PPO and GenPO struggle to maintain consistent behavioral modes across the abrupt environmental transitions introduced by barriers and pitfalls, while PPOC sometimes benefits from temporal abstraction (41.6% higher final reward than PPO on `H1 Pitfall`) but is constrained by its less expressive Gaussian intra-option policy. H-GenPO, by contrast, leverages both the temporal structure of Option-Critic and the multimodal expressiveness of the flow policy, enabling reliable mode switching without sacrificing low-level execution quality.

Qualitative analysis. Figure 4 visualizes post-hoc selected examples of option assignments discovered by H-GenPO during evaluation on two representative tasks. On the Humanoid task, H-GenPO spontaneously discovers two distinct behavioral phases: an *acceleration mode* (Option 1, yellow) in the early phase of each episode, during which the agent aggressively accelerates toward the target direction, and a *velocity maintenance mode* (Option 3, green) that activates once the agent approaches its target speed and focuses on stable, efficient locomotion. The transition between modes is temporally coherent, with the agent committing to each option for extended horizons rather than switching at every timestep.

On the Go2 Pitfall task, H-GenPO learns to associate options with environmental context rather than the locomotion phase. Option 1 (red) activates when the agent detects an impending pitfall in

its path, triggering a *hazard-avoidance mode* characterized by lateral evasion and reduced forward speed. Once the hazard is cleared, the agent transitions to Option 2 (green), resuming *standard command-following behavior*. In these examples, the semantically meaningful option assignments emerge entirely from the reward signal, without any option-level supervision or predefined sub-goal structure.

5.3 Sensitivity Analysis

We conduct a sensitivity analysis on the `Ant` and `Ant Barrier` tasks to evaluate the impact of key hyperparameters, with results summarized in Table 2.

Option cardinality (N_ω). Performance peaks at $N_\omega = 4$ for `Ant` and $N_\omega = 2$ for `Ant Barrier`. While a single option ($N_\omega = 1$) fails to exploit hierarchical structure, excessive cardinality (e.g., $N_\omega = 16$) leads to performance degradation due to the dispersion of samples across too many options, which hinders the convergence of individual intra-option policies.

Diffusion Timesteps (K). The optimal number of denoising steps is $K = 5$ for both `Ant` and `Ant Barrier`. We observe that insufficient steps ($K = 1$) limit the expressiveness of the action distribution, while increasing K beyond the optimal point (e.g., $K = 20$) does not yield further gains and instead leads to slow convergence.

Deliberation Cost (η). The impact of transition penalties varies significantly by task complexity. For the standard `Ant` locomotion, $\eta = 0$ is sufficient as the task does not demand frequent mode switching. However, in the `Ant Barrier` task, a moderate cost ($\eta = 1.0 \times 10^{-3}$) is crucial to prevent "option chattering"—a phenomenon where the high-level policy switches options too frequently, leading to suboptimal and jittery behaviors. Meanwhile, an excessively high η hinders learning by over-penalizing the necessary transitions between behavioral modes.

Table 2: Sensitivity analysis of key hyperparameters (option cardinality N_ω, deliberation cost η, diffusion timestep K) in H-GenPO.

Hp.	Value	Ant	Ant Barrier
GenPO (baseline)		159.88±10.71	50.35±2.29
N_ω	1	166.73±14.10	49.79±2.43
	2	170.64±8.45	**58.78±0.30**
	4	**173.94 ± 16.00**	54.91±2.68
	8	171.39±12.88	55.59±0.80
	16	169.12±14.85	53.61±3.96
η	0	**175.08±5.29**	54.19±2.00
	0.0001	172.31±5.92	56.57±4.81
	0.001	170.95±4.86	**57.12±3.55**
	0.01	159.52±13.08	53.39±10.51
K	1	163.17±6.76	52.47±3.82
	2	172.77±2.90	56.27±2.03
	5	**173.94±16.00**	**58.78±0.30**
	10	165.75±21.11	54.32±1.54
	20	150.07±10.14	52.93±2.55

6 Conclusion

We present H-GenPO, a framework that integrates on-policy diffusion-based RL with end-to-end hierarchical option learning via the Option-Critic architecture. Beyond raw performance gains, we observe that option specialization can emerge purely from the reward signal without any option-level supervision, although its degree is task- and run-dependent; this suggests that generative flow policies embedded in a hierarchical structure can partition behavioral space into semantically coherent modes, pointing toward their broader role as implicit discovery mechanisms for structured behavioral primitives.

Limitations. First, exact log-likelihood computation requires running the full reverse chain at every timestep, making it non-trivial to scale diffusion timesteps for more expressive policies in complex tasks. Second, the shared option embedding does not guarantee fully independent intra-option policies, which may become a limiting factor when options must exhibit highly disparate behavioral modes. Third, our evaluation is confined to single-level hierarchical control over short option horizons; whether H-GenPO extends gracefully to deeper hierarchical reasoning or longer temporal abstraction remains an open empirical question. Finally, our evidence is confined to simulated continuous control: the context-dependent suite is small and author-designed, gains on standard tasks are not fully disentangled from model capacity, behavioral claims rest partly on reward and proxy metrics, option specialization is task- and run-dependent and can occasionally collapse to a single option, and real-robot transfer remains to be validated. We hope these findings and open challenges inspire future work at the intersection of generative modeling and structured hierarchical control.

Acknowledgement

This work was supported by the Ministry of Science and ICT (MSIT) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) through the "Development of Core Technologies for Application Support of Autonomous Agents and Devices" program (RS-2026-25520169), specifically "(Sub-project 3) Development of On-Device VLM and Real-Time Mobility Intelligence Technologies for Mission Execution of Autonomous Agents" (25%), the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2026-25480865) (25%), the InnoCORE program of the Ministry of Science and ICT (N10260099) (25%), and the 2026 Cultural Technology Research and Development Project of the Ministry of Culture, Sports and Tourism and the Korea Creative Content Agency. (Project name: Development of high-precision hand-crafting and service technologies based on physical AI number: RS-2026-25510431) (25%).

References

[1] Alejandro Agostini and Enric Celaya. Reinforcement learning with a gaussian mixture model. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[3] Akhil Bagaria, Ben Abbatematteo, Omer Gottesman, Matt Corsaro, Sreehari Rammohan, and George Konidaris. Effectively learning initiation sets in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[4] Joseph K Blitzstein and Jessica Hwang. *Introduction to probability*. Chapman and Hall/CRC, 2019.

[5] Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[6] Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

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

[8] Wonhyeok Choi, Shutong Ding, Minwoo Choi, Jungwan Woo, Kyumin Hwang, Jaeyeul Kim, Ye Shi, and Sunghoon Im. A review of online diffusion policy rl algorithms for scalable robotic control. *arXiv preprint arXiv:2601.06133*, 2026.

[9] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

[10] Shutong Ding, Ke Hu, Shan Zhong, Haoyang Luo, Weinan Zhang, Jingya Wang, Jun Wang, and Ye Shi. GenPO: Generative diffusion models meet on-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

[11] Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *International Conference on Learning Representations*, 2024.

[12] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[13] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

[14] Chenxiao Gao, Edward Chen, Tianyi Chen, and Bo Dai. Flowrl: A taxonomy and modular framework for reinforcement learning with diffusion policies. *arXiv preprint arXiv:2603.27450*, 2026.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[16] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[19] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[20] Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, and Doina Precup. Options of interest: Temporal abstraction with interest functions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4444–4451, 2020.

[21] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[22] Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004*, 2017.

[23] Martin Klissarov and Marlos C. Machado. Deep Laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning*, pages 17198–17217, 2023.

[24] Martin Klissarov and Doina Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34:4632–4646, 2021.

[25] Peter E Kloeden and RA Pearson. The numerical solution of stochastic differential equations. *The ANZIAM Journal*, 20(1):8–12, 1977.

[26] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

[27] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[28] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[29] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

[30] Marlos C. Machado, André Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80):1–69, 2023.

[31] David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

[32] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[33] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[34] Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations*, 2024.

[35] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[36] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[37] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[39] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[40] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[41] Matthew Smith, Herke Hoof, and Joelle Pineau. An inference-based policy gradient method for learning options. In *International Conference on Machine Learning*, pages 4703–4712. PMLR, 2018.

[42] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[43] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[44] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[45] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pages 3540–3549. PMLR, 2017.

[46] Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22532–22541, 2023.

[47] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

[48] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

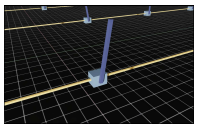
[49] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

A Simulation Environment and Task Details

A.1 Standard Benchmarks

We evaluate H-GenPO on 8 representative tasks from the NVIDIA Isaac Lab benchmark [32], spanning diverse robot morphologies and control challenges. (Images are sourced from the official Isaac Lab documentation `https://isaac-sim.github.io/IsaacLab/main/source/overview/environments.html`) Tasks are drawn from three categories: classic continuous control, robotic manipulation, and legged locomotion.

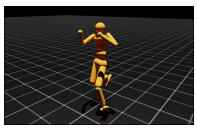
Classic tasks. Derived from the MuJoCo [44] simulation environment, these tasks have been widely adopted as standardized benchmarks for evaluating continuous control algorithms, providing well-understood baselines for comparing policy optimization methods across the RL community.



Cartpole
The goal is to move the cart along a rail to keep the pole balanced upright. This is a classic control task used as a baseline for reinforcement learning.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^4 \times \mathbb{R}^1$

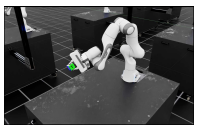


Ant
A four-legged MuJoCo ant robot must learn to coordinate its limbs to move forward in a specific direction as quickly as possible.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{60} \times \mathbb{R}^8$



Humanoid
A complex bipedal locomotion task where the MuJoCo humanoid robot aims to move forward without falling.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{87} \times \mathbb{R}^{21}$

Manipulation task. The following task evaluates dexterous manipulation capability, which demands fine-grained motor control and precise spatial reasoning under contact-rich dynamics.

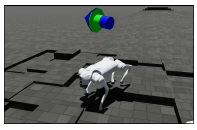


Franka Lift
Using the Franka Emika robotic arm, the agent must pick up a cube from a table and lift it to a randomly sampled target position.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{36} \times \mathbb{R}^8$

Locomotion tasks. We evaluate on four locomotion tasks that vary along two dimensions: terrain difficulty and robot embodiment. ANYmal-D operates on flat terrain using proprioception only, while the remaining three tasks—Unitree Go2 (quadruped), Unitree H1 (humanoid), and Cassie (bipedal)—require navigating rough terrain with additional LiDAR exteroceptive observations, resulting in substantially higher-dimensional observation spaces. This combination of varied morphologies and sensor configurations provides a comprehensive testbed for evaluating policy expressiveness across diverse continuous control settings.



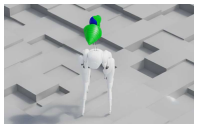
ANYmal-D
The ANYbotics ANYmal-D quadruped robot is tasked with tracking a target velocity command while walking on flat terrain.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{48} \times \mathbb{R}^{12}$



Unitree Go2
A challenging locomotion task where the Unitree Go2 quadruped tracks velocity commands on rough terrain using LiDAR exteroception.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{235} \times \mathbb{R}^{12}$



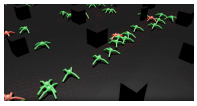
Unitree H1
A full-sized humanoid Unitree H1 robot must navigate rough terrain while maintaining its balance and tracking velocity.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{256} \times \mathbb{R}^{19}$



Cassie
The bipedal Cassie robot tracks velocity commands on rough terrain, relying on LiDAR exteroception for robust movement.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{235} \times \mathbb{R}^{12}$

A.2 Context-Dependent Benchmarks

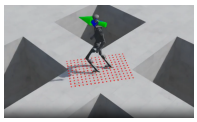
To evaluate H-GenPO's capacity for hierarchical behavioral mode switching, we introduce three custom context-dependent tasks that require the agent to identify the current terrain context and commit to qualitatively distinct locomotion strategies accordingly. Each task is constructed by replacing the standard terrain configuration with a mixed terrain layout that interleaves open traversable surfaces with geometrically impassable obstacles. Apart from the addition of a height-map sensor (LiDAR exteroception) to the Ant robot, no other modifications are made to the robot morphology or reward structure. The key design principle is that no single locomotion mode is optimal across all terrain types: the agent must learn to switch behavioral strategies in response to environmental context, a challenge that naturally motivates hierarchical control with explicit temporal commitment.



Ant Barrier
The MuJoCo Ant robot navigates a terrain populated with sparse, fixed-height pillar obstacles (height: 2 m; width: 1.5–2.0 m) generated via a discrete obstacle heightfield, with a 3 m flat platform at the center of each tile serving as a safe spawning zone. Because the pillars are far taller than the robot, the agent cannot climb over them and must actively detour around each obstacle, alternating between a forward-locomotion mode for open traversal and a lateral maneuvering mode for obstacle circumvention.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{252} \times \mathbb{R}^{8}$



Go2 Pitfall
The Unitree Go2 quadruped navigates a mixed terrain in which 50% of tiles are standard rough terrain (stairs, slopes, boxes, random roughness) and 50% are star-shaped trench tiles featuring 1.2 m-wide, 2 m-deep channels. Critically, the channel width exceeds the Go2's effective leg span, making it physically impossible for the robot to stride across; the agent must instead detour around the trench. This demands two qualitatively distinct behavioral modes—robust rough-terrain locomotion and active pitfall avoidance—that must be *sustained* over appropriate temporal horizons rather than re-selected at every step.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{235} \times \mathbb{R}^{12}$



H1 Pitfall
An analogous context-dependent task for the Unitree H1 humanoid robot, using the same mixed terrain configuration as Go2 Pitfall but scaled to the H1's larger body dimensions. The trench width exceeds the H1's stride capability under normal gait, requiring the robot to switch between standard bipedal locomotion on rough terrain tiles and active pitfall-avoidance behavior near trench tiles. The higher-dimensional state and action spaces of the H1 amplify the difficulty of mode discovery, making this task a more demanding test of whether the policy's hierarchical reasoning can induce interpretable and sustained behavioral specialization in a complex morphology.
$(\mathcal{S}, \mathcal{A}) \in \mathbb{R}^{256} \times \mathbb{R}^{19}$

Behavioral metrics. To complement episodic reward, Table 1 reports direct behavioral metrics on the context-dependent tasks. Each trained policy is evaluated over 1,000 parallel agents per task for all 5 seeds, and we report: the *early-termination rate*, i.e., the fraction of episodes that end in a termination condition before the time limit, which subsumes falls; the *pit-fall rate* on `Go2 Pitfall` and `H1 Pitfall`, i.e., the fraction of episodes terminated by falling into a trench (its complement is the pit-avoidance rate); the mean absolute error between the commanded and achieved planar linear velocity (v_{xy}) and yaw angular velocity (ω_z) on the two pitfall tasks; and the mean forward velocity on `Ant Barrier`, whose reward targets forward progress. We do not report a discrete

detour-success rate for `Ant Barrier`, since circumventing a pillar has no well-defined trigger event; forward velocity and early-termination rate serve as its quantitative proxies.

B Hyperparameter Setup & Implementation Details of H-GenPO

B.1 Implementation Details of H-GenPO

Option embedding dimension. The option embedding dimension d_e is set equal to the action space dimension of each task. This allows the expressiveness of the option conditioning signal to scale adaptively with task complexity, without requiring manual tuning per environment.

Advantage normalization and clipping. For the intra-option policy, we apply standard advantage normalization to stabilize PPO-style updates. For the option-level policy, however, we deliberately omit advantage normalization to preserve the absolute scale of option-switching signals, which carry meaningful information about the relative value of committing to or terminating an option. To ensure numerical stability, we apply min-max clipping to the termination advantage in the range $[-5.0, 5.0]$.

Deliberation cost. The deliberation cost η directly enters the reward signal as a penalty for option switches, and therefore requires adjustment according to the scale of environment rewards to produce appropriately calibrated incentives for temporal commitment. In practice, we set η per task heuristically to a value that is small relative to the per-step reward scale and does not impede early training progress, rather than tuning it on final evaluation performance; its sensitivity is analyzed in Table 2. The task-specific values of η used in our experiments are reported in Table 3. All remaining H-GenPO-specific loss scales $\lambda_\Omega, \lambda_\omega, \lambda_\beta, \lambda_{\mathcal{H}}$ are held fixed across all tasks, and the number of options is set to $N_\omega = 4$ for the standard tasks and $N_\omega = 2$ for the context-dependent tasks (see Table 4), demonstrating the robustness of H-GenPO under a unified configuration.

B.2 Hyperparameter Setup

For each algorithm, algorithm-specific hyperparameters are reported in Table 4. To ensure a fair and controlled comparison, all remaining hyperparameters are standardized across algorithms. The full configurations are organized as follows: Table 5 lists the general hyperparameters shared by all algorithms; Table 6 and Table 7 detail the parameters specific to the on-policy group (PPO, GenPO, PPOC, H-GenPO) and the off-policy group (SAC, DIPO, QVPO, DACER), respectively.

Table 3: Task-specific deliberation cost η used for H-GenPO across all evaluated environments.

Environment	$\boldsymbol{\eta}$
Cartpole	0.001
Ant	0.002
Humanoid	0.01
Franka Lift	0.0002
ANYmal-D	0.005
Unitree Go2	0.005
Unitree H1	0.005
Cassie	0.005
Ant Barrier	0.002
Go2 Pitfall	0.005
H1 Pitfall	0.005

Table 4: Algorithm-specific hyperparameters of online DPRL methods (DIPO [49], QVPO [9], DACER [47], and GenPO [10]) and HRL methods (PPOC [22] and H-GenPO). These hyperparameters are shared across all environments unless otherwise noted.

Hyperparameters	**DIPO**
Diffusion sampling steps	100
Action gradient steps	20
Target policy update freq.	1
Action learning rate	3.0×10^{-2}

Hyperparameters	**QVPO**
Diffusion sampling steps	20
Target policy update freq.	1
Running q mean	1
Running q std.	0
Alpha mean	0.001
Alpha std.	0.001
Entropy weight	0.02
Diffusion sampling number	64
Batch sampling number	10

Hyperparameters	**DACER**
Diffusion sampling steps	20
Target policy update freq.	1
Entropy learning rate	0.03
Initial entropy value	0.02
Target entropy	-dim(action)
Alpha update frequency	total timesteps / 10

Hyperparameters	**GenPO**
Diffusion sampling steps	5
Target policy update freq.	1
Compress coefficient	0.01
Mixing coefficient	0.9

Hyperparameters	**PPOC & H-GenPO**
N_ω	4 (2 for custom tasks)
λ_Ω	0.01
λ_ω	1.0
$\lambda_\mathcal{H}$	0.1
λ_β	0.1

Table 5: General hyperparameters shared across all algorithms. In the case of DACER, the actor learning rate is fixed at 5.0×10^{-4} across all environments following its original implementation [47], while all other algorithms use the task-specific values listed below.

Environments	**Cartpole**	**Ant**	**Humanoid**	**Franka Lift**
# of environment	1024	1024	1024	1024
Learning rate (Actor)	3.0×10^{-4}	3.0×10^{-4}	5.0×10^{-4}	1.0×10^{-4}
Learning rate (Critic)	3.0×10^{-4}	3.0×10^{-4}	5.0×10^{-4}	1.0×10^{-4}
Hidden layers (Actor)	[32, 32]	[256, 128, 64]	[400, 200, 100]	[256, 128, 64]
Hidden layers (Critic)	[32, 32]	[256, 128, 64]	[400, 200, 100]	[256, 128, 64]
Discount factor	0.99	0.99	0.99	0.99
GradNorm clip	1.0	1.0	1.0	1.0
Environments	**Anymal-D**	**Unitree Go2**	**Unitree H1**	**Cassie**
# of environment	1024	1024	1024	1024
Learning rate (Actor)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Learning rate (Critic)	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Hidden layers (Actor)	[128, 128, 128]	[512, 256, 128]	[512, 256, 128]	[512, 256, 128]
Hidden layers (Critic)	[128, 128, 128]	[512, 256, 128]	[512, 256, 128]	[512, 256, 128]
Discount factor	0.99	0.99	0.995	0.99
GradNorm clip	1.0	1.0	1.0	1.0

Table 6: Common hyperparameters shared across on-policy Algorithms (PPO, GenPO, PPOC, H-GenPO).

Environments	**Cartpole**	**Ant**	**Humanoid**	**Franka Lift**
Rollouts	16	16	32	24
Learning epochs	8	4	5	8
# of mini-batches	8	2	4	4
GAE Smoothing Param.	0.95	0.95	0.95	0.95
Desired KL	0.008	0.008	0.01	0.01
Surrogate clip	0.2	0.2	0.2	0.2
Value clip	0.2	0.2	0.2	0.2
Entropy coefficient	0.0	0.0	0.0	0.001
Value loss coefficient	2.0	1.0	2.0	2.0
Environments	**Anymal-D**	**Unitree Go2**	**Unitree H1**	**Cassie**
Rollouts	24	24	24	24
Learning epochs	5	5	5	5
# of mini-batches	4	4	4	4
GAE Smoothing Param.	0.95	0.95	0.95	0.95
Desired KL	0.01	0.01	0.01	0.01
Surrogate clip	0.2	0.2	0.2	0.2
Value clip	0.2	0.2	0.2	0.2
Entropy coefficient	0.005	0.01	0.01	0.01
Value loss coefficient	1.0	1.0	1.0	1.0

Table 7: Common hyperparameters shared across off-policy Algorithms (SAC, DIPO, QVPO, DACER).

Environments	**Cartpole**	**Ant**	**Humanoid**	**Franka Lift**
Batch size	4096	4096	4096	4096
Replay buffer size	4096	4096	4096	4096
Soft update parameter	0.005	0.005	0.005	0.005
Random action steps	4	4	4	4
Environments	**Anymal-D**	**Unitree Go2**	**Unitree H1**	**Cassie**
Batch size	4096	4096	4096	4096
Replay buffer size	4096	4096	4096	4096
Soft update parameter	0.005	0.005	0.005	0.005
Random action steps	4	4	4	4

C Computational Cost and Model Capacity

C.1 Cost–Performance Comparison

Table 8 compares all eight algorithms on the Ant task using a single NVIDIA A6000 GPU, reporting the number of sampling iterations per action, total training wall-time, per-step inference latency, peak GPU memory, parameter count, and final reward.

Table 8: Cost–performance comparison on the Ant task (single NVIDIA A6000 GPU). Steps: sampling iterations per action. Inference: median action-generation latency per environment step at the training batch size (1,024 parallel environments). Final reward: mean±std over 5 seeds.

Algorithm	Steps	Train wall-time (min)	Inference (ms/step)	Peak GPU mem. (GB)	#Params	Final reward
PPO	–	8.0	2.23	3.36	114,129	83.08±71.96
SAC	–	14.0	2.26	11.44	175,058	60.60±12.52
DIPO	100	102.7	82.83	15.62	201,962	73.84±3.58
QVPO	20	172.7	55.68	11.66	201,962	19.73±12.32
DACER	20	44.0	39.10	11.37	202,092	3.51±1.64
GenPO	5	24.7	15.40	3.38	143,089	159.88±10.71
PPOC	–	13.3	3.94	3.36	343,500	113.77±30.95
H-GenPO	5	26.7	14.39	3.32	200,596	173.94±16.00

Parameter efficiency. PPOC, which instantiates separate per-option policy networks, is the largest model among all eight methods (343.5K parameters, 286.5K of which belong to the policy), whereas H-GenPO conditions a single shared flow on option embeddings and requires 200.6K parameters—41.6% fewer than PPOC—empirically confirming the $O(|\theta|+N_\omega \cdot d_e)$ scaling discussed in Section 4.1.

Inference latency. Per-step latency tracks the number of sampling iterations: 100-step DIPO and 20-step QVPO and DACER require 82.8, 55.7, and 39.1 ms per step, respectively, whereas the fixed 5-step EDICT flow keeps GenPO and H-GenPO at about 14–15 ms, 2.5–6× below the diffusion baselines (one-step Gaussian policies require 2.2–3.9 ms).

Overhead of the hierarchy. The hierarchical modules add no measurable inference or memory cost: the median latencies of H-GenPO and GenPO are 14.39 and 15.40 ms with overlapping interquartile ranges ($[14.07, 14.70]$ vs. $[14.00, 15.60]$), and their peak memory is 3.32 and 3.38 GB, since π_Ω and β are lightweight MLPs that output only an option index and a scalar. The measurable overheads over GenPO are therefore an 8.1% increase in training wall-time, primarily attributable to the additional loss terms in the optimization phase rather than to action generation, and a 40.2% increase in parameters (0.20M in total), which we control for in Appendix C.2.

Memory. The on-policy methods require 3–5× less peak GPU memory (3.3–3.4 GB) than the off-policy diffusion baselines (11.4–15.6 GB), which rely on large replay buffers.

C.2 Capacity-Matched Flat Baseline

To test whether the gains of H-GenPO stem from its additional parameters rather than from its option structure, we construct GenPO-L, a flat GenPO whose network is enlarged so that its total parameter count matches that of the full H-GenPO, with all other hyperparameters unchanged. We train GenPO-L on the three standard tasks with the largest absolute reward margins over GenPO (Ant, Humanoid, Franka Lift) and on all three context-dependent tasks under the identical protocol (5 seeds).

Table 9: Capacity-matched control. GenPO-L is a flat GenPO enlarged to match the total parameter count of H-GenPO; all other hyperparameters are unchanged. Final reward, mean±std over 5 seeds.

	Standard tasks			Context-dependent tasks		
Method	**Ant**	**Humanoid**	**Franka**	**Ant Barrier**	**Go2 Pitfall**	**H1 Pitfall**
GenPO	159.88±10.71	202.25±19.24	106.73±58.12	50.35±2.29	20.97±0.16	18.28±1.72
GenPO-L	137.15±8.52	192.06±9.22	106.86±58.86	51.11±3.43	20.94±0.18	16.98±2.42
H-GenPO	**173.94±16.00**	**209.48±6.44**	**129.86±1.91**	**58.78±0.30**	**22.84±0.67**	**19.80±0.85**

As shown in Table 9, GenPO-L does not recover the margin of H-GenPO on any task: it is comparable to GenPO on Franka Lift, Ant Barrier, and Go2 Pitfall, slightly below on H1 Pitfall (with overlapping standard deviations), and degrades on Ant and Humanoid. At a fixed on-policy sample budget, enlarging the network beyond its tuned size increases the optimization burden without adding a useful inductive bias, so performance is not monotone in parameter count. Since only the network size was changed, re-tuning GenPO-L might recover part of this loss; nevertheless, GenPO-L does not meaningfully exceed GenPO on any task (all differences lie within one standard deviation), whereas H-GenPO spends a comparable parameter budget and improves on the same tasks.

Two further observations support attributing the gains to the option structure. First, the $N_\omega = 1$ setting in Table 2 serves as an architecture-only control: all architectural components remain intact—the flow is still conditioned on an option embedding and Q_Ω reduces to a standard critic—while the option structure has no effect. It recovers part of the gain on Ant (166.73 vs. 159.88 for GenPO) but none on Ant Barrier (49.79 vs. 50.35), where $N_\omega = 2$ reaches 58.78. Second, π_Ω and β cannot add capacity to the action pathway: they output only a categorical index, which selects a row of the embedding table, and a scalar termination probability, and none of their internal features enter the intra-option policy, so their only causal route to behavior is the option structure itself.

On standard tasks that do not explicitly require hierarchy, a plausible mechanism is conditional computation: as in mixture-of-experts models [18, 40], input-dependent selection among specialized modes increases effective capacity at constant per-step compute. Consistent with this view, all four options remain in use on Ant throughout training (Fig. 8, shown for a single seed), suggesting a state-conditional mixture over behavioral phases rather than a single dominant mode.

D Reward Curves of Sensitivity Analysis

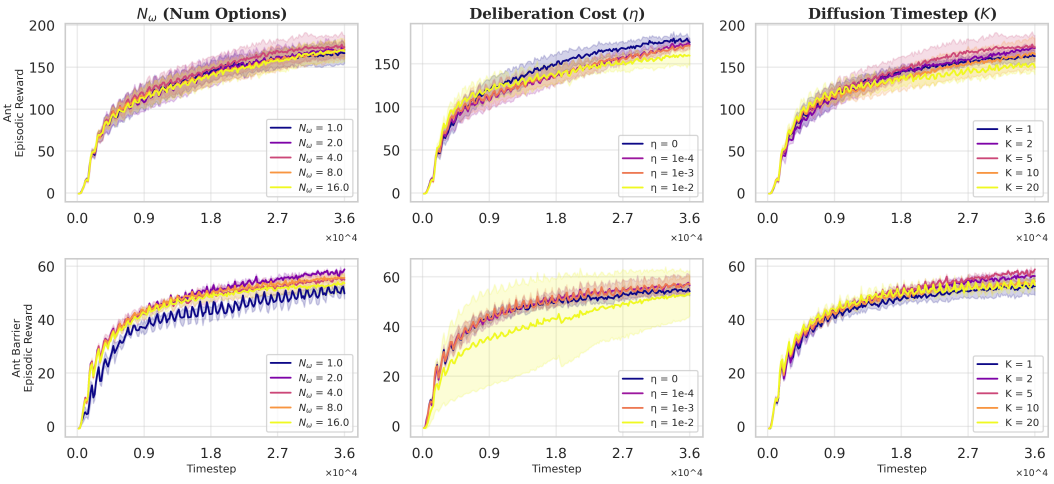


Figure 5: Training reward curves for the sensitivity analysis of H-GenPO on the `Ant` (top) and `Ant Barrier` (bottom) tasks. Each row varies one hyperparameter while keeping others fixed at their default configurations. *Left*: option cardinality $N_\omega \in \{1, 2, 4, 8, 16\}$. *Center*: deliberation cost $\eta \in \{0, 10^{-4}, 10^{-3}, 10^{-2}\}$. *Right*: number of diffusion timesteps $K \in \{1, 2, 5, 10, 20\}$. Shaded regions indicate standard deviation over 5 seeds.

The sensitivity analysis in the main paper (Section 5.3) reports final episode rewards for varying option cardinality (N_ω), diffusion timesteps (K), and deliberation cost (η) in tabular form. Here we supplement those results with the full training reward curves to provide a more complete picture of how each hyperparameter affects not only asymptotic performance but also convergence speed and training stability. Figure 5 shows reward curves on the `Ant` and `Ant Barrier` tasks for all evaluated configurations of N_ω, K, and η.

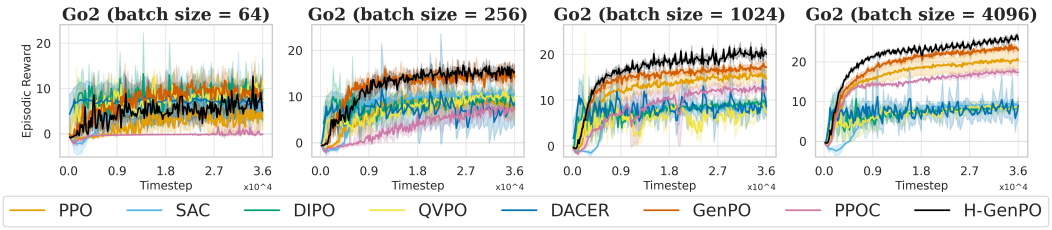


Figure 6: Effect of the number of parallel environments (batch size) on final episode reward on the Unitree Go2 task. Each entry reports the mean final reward over 5 independent seeds; values in parentheses denote standard deviation. On-policy methods benefit substantially from larger batch sizes due to synchronized rollout collection, while off-policy methods remain comparatively insensitive to this factor.

Table 10: Final episode reward on the Unitree Go2 task across varying numbers of parallel environments (batch sizes). Mean ± std over 5 seeds. **Bold**: best. Underline: second best.

Batch size	PPO	SAC	DIPO	QVPO	DACER	GenPO	PPOC	H-GenPO
64	3.12 ± 2.52	8.12 ± 6.25	9.73 ± 7.76	$\mathbf{11.61 \pm 2.02}$	6.91 ± 4.26	7.67 ± 2.11	-0.44 ± 0.68	5.08 ± 2.25
256	7.14 ± 2.25	9.37 ± 0.86	8.17 ± 1.13	10.00 ± 2.25	7.38 ± 4.11	14.22 ± 2.33	7.33 ± 1.08	$\mathbf{15.33 \pm 1.82}$
1024	14.66 ± 0.86	8.03 ± 2.06	9.53 ± 0.69	7.22 ± 1.62	8.58 ± 2.33	17.23 ± 1.24	12.32 ± 1.52	$\mathbf{20.09 \pm 1.14}$
4096	20.38 ± 2.83	9.17 ± 1.05	8.75 ± 0.73	8.61 ± 1.00	8.96 ± 2.39	23.02 ± 0.66	17.32 ± 1.12	$\mathbf{25.60 \pm 0.68}$

E Effect of Batch Size

We investigate the effect of the number of parallel environments on algorithm performance, as batch size is a key factor governing the relative advantage of on-policy versus off-policy methods in GPU-parallelized simulation. Figure 6 and Table 10 show performance across a range of environment counts (64–4096 parallel environments) on the Unitree Go2 task. With very small batch sizes (fewer than 256 environments), off-policy methods (SAC, QVPO) achieve competitive or superior performance, as their replay buffer provides effective sample reuse under data-scarce conditions. As batch size increases, on-policy methods rapidly overtake off-policy baselines, and at or above 256 parallel environments, GenPO, PPOC, and H-GenPO consistently outperform all other algorithms. Notably, H-GenPO benefits most from increased batch size, achieving the highest final reward at all batch sizes above 64 among on-policy methods, and its performance advantage over PPOC widens as more environment instances are available. This confirms that the high-throughput simulation setting is a key enabler of the on-policy advantage observed throughout our experiments, and that hierarchical generative policies are particularly well-suited to this data-rich regime.

F Action sampling and Reverse Process of the Invertible Flow Policy in H-GenPO

In this appendix, we provide a self-contained and detailed derivation of the invertible flow architecture used in GenPO [10], which H-GenPO inherits as its intra-option policy. We cover (1) why standard flow matching cannot compute exact log-likelihoods, (2) the doubled dummy action construction that resolves this, (3) the forward process for action sampling, (4) the reverse process for exact log-likelihood computation, and (5) how the real action is recovered.

F.1 Why Standard Flow Matching Cannot Compute Exact Log-Likelihoods

A flow matching model [28] learns a time-dependent velocity field $v_\theta(\mathbf{x}, t)$ that transports samples from a Gaussian prior $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the target distribution $\mathbf{x}_1 \sim p_{\text{data}}$ by solving the ODE:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + v_\theta(\mathbf{x}_t, t)\, \Delta t. \tag{15}$$

To compute the log-likelihood of a given action a via the change-of-variables formula (Lemma 1), the mapping $f_\theta : \mathbf{x}_0 \mapsto \mathbf{x}_1$ must be *exactly invertible*. However, in practice, the forward integration of Eq. (15) and the reverse integration (inversion) are computed independently, and the Euler-Maruyama discretization [25] introduces approximation errors that cause them to be inconsistent. Concretely,

the forward step approximates $\mathbf{x}_t$ with $\mathbf{x}_{t-1}$ in the velocity evaluation, so the forward and reverse processes do not cancel exactly, preventing exact inversion and making log-likelihood computation intractable.

Lemma 1 (Change of Variables [4])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible and smooth mapping. If $X \sim q(x)$ and $Y = f(X)$, then the density of Y is:*

$$p(y) = q(x) \left| \det \frac{\partial f}{\partial x} \right|^{-1}. \tag{16}$$

F.2 Doubled Dummy Action Space

To realize an exactly invertible flow, GenPO adopts a coupled alternating update scheme motivated by EDICT [46]. Instead of a single noise vector, two noise vectors $\mathbf{x}_0, \mathbf{y}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are independently sampled, forming a *dummy action* $\tilde{a} = (\mathbf{x}_1, \mathbf{y}_1)$ in a doubled action space $\widetilde{\mathcal{A}} = \mathcal{A} \times \mathcal{A}$.

The two components are updated alternately at each denoising step, with information exchange via a mixing scheme controlled by a coefficient $p \in (0, 1)$ (see Sec. F.3-F.4). This alternating structure is the key that makes the entire forward-reverse process an exactly invertible diffeomorphism, resolving the inconsistency described above.

The action executed in the environment is recovered by averaging the two components as follows:

$$a = \frac{\mathbf{x}_1 + \mathbf{y}_1}{2} \in \mathcal{A}. \tag{17}$$

When the dummy policy is optimal in the reformulated MDP with action space $\widetilde{\mathcal{A}}$, the averaged action is also optimal in the original MDP, since the averaging is a deterministic linear map [10].

F.3 Forward Process: Action Sampling

For brevity, we omit the environment timestep index and denote the variables solely by their denoising step t. Given current state s, option embedding e_ω, and the noise vectors $(\mathbf{x}_t, \mathbf{y}_t)$ at t, the forward process generates the dummy action $\tilde{a} = (\mathbf{x}_1, \mathbf{y}_1)$ by iterating the following two-step update from $t = 0$ to $t = 1$:

Step 1: Alternating velocity update. The two components are updated by applying the velocity field v_θ alternately, where the second update uses the already-updated first component:

$$\tilde{\mathbf{x}}_{t+\Delta t} = \mathbf{x}_t + v_\theta(\mathbf{y}_t, t, s, e_\omega)\, \Delta t, \tag{18}$$

$$\tilde{\mathbf{y}}_{t+\Delta t} = \mathbf{y}_t + v_\theta(\tilde{\mathbf{x}}_{t+\Delta t}, t, s, e_\omega)\, \Delta t. \tag{19}$$

Note that $\tilde{\mathbf{x}}_{t+\Delta t}$ is used in computing $\tilde{\mathbf{y}}_{t+\Delta t}$, creating a causal dependency that enables exact inversion.

Step 2: Mixing. To keep $\mathbf{x}$ and $\mathbf{y}$ closely aligned (preventing degenerate exploration in the doubled action space), the two components are mixed:

$$\mathbf{x}_{t+\Delta t} = p\, \tilde{\mathbf{x}}_{t+\Delta t} + (1 - p)\, \tilde{\mathbf{y}}_{t+\Delta t}, \tag{20}$$

$$\mathbf{y}_{t+\Delta t} = p\, \tilde{\mathbf{y}}_{t+\Delta t} + (1 - p)\, \mathbf{x}_{t+\Delta t}. \tag{21}$$

The coefficient $p \in (0, 1)$ controls the intensity of information exchange; setting p too small causes numerical instability, while p too large reduces alignment between $\mathbf{x}$ and $\mathbf{y}$.

Starting from $(\mathbf{x}_0, \mathbf{y}_0) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \times \mathcal{N}(\mathbf{0}, \mathbf{I})$, iterating Steps 1–2 for K denoising steps yields the dummy action $\tilde{a} = (\mathbf{x}_1, \mathbf{y}_1)$, from which the primitive action $a = (\mathbf{x}_1 + \mathbf{y}_1)/2$ is recovered by Eq. (17).

F.4 Reverse Process: Exact Log-Likelihood Computation

Given a dummy action $\tilde{a} = (\mathbf{x}_1, \mathbf{y}_1)$ (or equivalently, a primitive action a from which $\tilde{a}$ is obtained via the relation $\mathbf{x}_1 = \mathbf{y}_1 = a$ as a special case), the reverse process recovers the original noise vectors $(\mathbf{x}_0, \mathbf{y}_0)$ by exactly inverting Steps 1–2 of the forward process.

Step 3: Unmixing. The mixing of Eqs. (20)–(21) is inverted analytically.

$$\tilde{\mathbf{y}}_{t+\Delta t} = \frac{\mathbf{y}_{t+\Delta t} - (1-p)\,\mathbf{x}_{t+\Delta t}}{p}, \tag{22}$$

$$\tilde{\mathbf{x}}_{t+\Delta t} = \frac{\mathbf{x}_{t+\Delta t} - (1-p)\,\tilde{\mathbf{y}}_{t+\Delta t}}{p}. \tag{23}$$

Step 4: Inverse velocity update. The alternating update of Eqs. (18)–(19) is inverted in reverse order.

$$\mathbf{y}_t = \tilde{\mathbf{y}}_{t+\Delta t} - v_\theta(\tilde{\mathbf{x}}_{t+\Delta t}, t, s, e_\omega)\,\Delta t, \tag{24}$$

$$\mathbf{x}_t = \tilde{\mathbf{x}}_{t+\Delta t} - v_\theta(\mathbf{y}_t, t, s, e_\omega)\,\Delta t. \tag{25}$$

Note that $\mathbf{y}_t$ from Eq. (24) is used in Eq. (25), exactly mirroring the causal dependency in the forward process.

Iterating Steps 3–4 from $t = 1$ back to $t = 0$ recovers $(\mathbf{x}_0, \mathbf{y}_0)$ exactly, with no approximation error.

G Tractable Jacobian Computation via EDICT

A key practical concern in applying the change-of-variables formula (Eq. (2)) is the computation of the Jacobian determinant $|\det \partial(\mathbf{x}_1, \mathbf{y}_1)/\partial(\mathbf{x}_0, \mathbf{y}_0)|$. Naively, this would require differentiating through the velocity network v_θ at every denoising step, incurring prohibitive cost. We show here that the alternating coupled structure of the EDICT update yields a Jacobian determinant that is an analytic constant, independent of the network parameters, requiring no Jacobian computation at runtime.

G.1 Jacobian of the Alternating Velocity Update

Consider a single forward step. The alternating update maps $(\mathbf{x}_t, \mathbf{y}_t) \mapsto (\tilde{\mathbf{x}}_{t+\Delta t}, \tilde{\mathbf{y}}_{t+\Delta t})$ via:

$$\tilde{\mathbf{x}} = \mathbf{x}_t + v_\theta(\mathbf{y}_t, t, s, e_\omega)\,\Delta t, \tag{26}$$

$$\tilde{\mathbf{y}} = \mathbf{y}_t + v_\theta(\tilde{\mathbf{x}}, t, s, e_\omega)\,\Delta t. \tag{27}$$

Denote $J_x = \partial v_\theta(\tilde{\mathbf{x}}, t, s, e_\omega)/\partial \tilde{\mathbf{x}}$ and $J_y = \partial v_\theta(\mathbf{y}_t, t, s, e_\omega)/\partial \mathbf{y}_t$. The full Jacobian matrix of this mapping is:

$$J_{\text{alt}} = \frac{\partial(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})}{\partial(\mathbf{x}_t, \mathbf{y}_t)} = \begin{pmatrix} I & J_y\,\Delta t \\ J_x\,\Delta t & I + J_x J_y\,(\Delta t)^2 \end{pmatrix}. \tag{28}$$

Note that although the velocity network v_θ is conditioned on the state s and option embedding e_ω, these are fixed inputs with respect to the differentiation variables $(\mathbf{x}_t, \mathbf{y}_t)$; hence J_x and J_y may implicitly depend on s and e_ω, but this does not affect the cancellation in Eq. (29), which holds for any values of J_x and J_y.

Applying the Schur complement formula $\det\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det(A)\det(D - CA^{-1}B)$ with $A = I$:

$$\begin{aligned} \det(J_{\text{alt}}) &= \det(I) \cdot \det\Big((I + J_x J_y(\Delta t)^2) - J_x \Delta t \cdot I^{-1} \cdot J_y \Delta t\Big) \\ &= \det\Big(I + J_x J_y(\Delta t)^2 - J_x J_y(\Delta t)^2\Big) \\ &= \det(I) = 1. \end{aligned} \tag{29}$$

The Jacobian determinant of the alternating velocity update is *exactly 1*, regardless of the network weights θ or the step size Δt.

Jacobian of the mixing step. The mixing step maps $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \mapsto (\mathbf{x}_{t+\Delta t}, \mathbf{y}_{t+\Delta t})$ via:

$$\mathbf{x}_{t+\Delta t} = p\,\tilde{\mathbf{x}} + (1-p)\,\tilde{\mathbf{y}}, \tag{30}$$

$$\mathbf{y}_{t+\Delta t} = p\,\tilde{\mathbf{y}} + (1-p)\,\mathbf{x}_{t+\Delta t} = p(1-p)\,\tilde{\mathbf{x}} + \big(p + (1-p)^2\big)\,\tilde{\mathbf{y}}. \tag{31}$$

The Jacobian matrix of the mixing step is therefore:

$$J_{\text{mix}} = \frac{\partial(\mathbf{x}_{t+\Delta t},\, \mathbf{y}_{t+\Delta t})}{\partial(\tilde{\mathbf{x}},\, \tilde{\mathbf{y}})} = \begin{pmatrix} p\, I & (1-p)\, I \\ p(1-p)\, I & \big(p + (1-p)^2\big) I \end{pmatrix}. \tag{32}$$

Applying the Schur complement with $A = pI$, $B = (1-p)I$, $C = p(1-p)I$, $D = (p + (1-p)^2)I$:

$$\begin{aligned} D - CA^{-1}B &= \big(p + (1-p)^2\big) I - p(1-p) \cdot \frac{1}{p} \cdot (1-p) I \\ &= \big(p + (1-p)^2\big) I - (1-p)^2 I = p\, I. \end{aligned} \tag{33}$$

Therefore:

$$\det(J_{\text{mix}}) = \det(p\, I) \cdot \det(p\, I) = p^n \cdot p^n = p^{2n}, \tag{34}$$

where $n = |\mathcal{A}|$ is the dimension of the primitive action space. This is also a constant, depending only on the mixing coefficient p and the action dimensionality.

Total Jacobian determinant over K steps. Since each of the K denoising steps consists of an alternating update followed by a mixing step, and since the Jacobians compose multiplicatively, the total Jacobian determinant of the full forward mapping $(\mathbf{x}_0, \mathbf{y}_0) \mapsto (\mathbf{x}_1, \mathbf{y}_1)$ is:

$$\left|\det \frac{\partial(\mathbf{x}_1, \mathbf{y}_1)}{\partial(\mathbf{x}_0, \mathbf{y}_0)}\right| = \prod_{k=1}^{K} \left|\det J_{\text{alt}}^{(k)}\right| \cdot \left|\det J_{\text{mix}}^{(k)}\right| = \prod_{k=1}^{K} 1 \cdot p^{2n} = p^{2nK}. \tag{35}$$

Closed-form log-likelihood. Substituting Eq. (35) into the change-of-variables formula (Eq. (2)) yields a fully closed-form log-likelihood:

$$\log \pi_\theta(\tilde{a} \mid s) = \log \mathcal{N}(\mathbf{x}_0 \mid \mathbf{0}, \mathbf{I}) + \log \mathcal{N}(\mathbf{y}_0 \mid \mathbf{0}, \mathbf{I}) - 2nK \log p, \tag{36}$$

where $(\mathbf{x}_0, \mathbf{y}_0)$ are recovered from $(\mathbf{x}_1, \mathbf{y}_1)$ via the reverse process (Section F.4), and $2nK \log p$ is a *constant correction term* that depends only on the action dimension n, the number of denoising steps K, and the mixing coefficient p.

This result is remarkably clean: the log-likelihood requires *no Jacobian computation of the neural network* v_θ at any step. The only quantities needed are the recovered noise vectors $(\mathbf{x}_0, \mathbf{y}_0)$ via Eq. (22)-(25) and the constant offset $2nK \log p$, making exact likelihood evaluation as efficient as a single reverse-process pass through the network.

H Training Dynamics of H-GenPO

Finally, to provide transparency into how H-GenPO's hierarchical structure emerges over the course of training, we report six diagnostic metrics for each of the 8 standard benchmark tasks. These metrics jointly characterize both the stability of the optimization process and the degree to which meaningful option specialization develops from the reward signal alone, without any option-level supervision.

Metrics. *Option Selection Rate* tracks the proportion of timesteps each option ω is active, revealing whether and how the high-level policy distributes behavioral responsibility across options over time. *Termination Rate* measures the fraction of timesteps at which the termination network β_ω outputs a termination signal, serving as an indicator of temporal commitment: high termination rate early in training reflects frequent, exploratory switching, while convergence toward zero indicates that the agent has learned to sustain options over extended horizons. The remaining four panels report the training losses: *Policy Loss* ($\mathcal{L}_\omega$, intra-option PPO surrogate), *Option Policy Loss* ($\mathcal{L}_\Omega$, high-level REINFORCE loss), *Termination Loss* ($\mathcal{L}_\beta$), and *Value Loss* ($\mathcal{L}_Q$), each of which converge stably, as shown in the Fig. 7-14.

Cross-task observations. Several consistent patterns emerge across all tasks. First, termination rate universally decreases from a high initial value toward near zero, confirming that H-GenPO reliably learns to commit to options for longer durations as training progresses, regardless of task type. Second, option specialization is task-dependent: the degree to which options differentiate and the resulting selection distribution vary across tasks, suggesting that H-GenPO adapts its hierarchical structure to

the nature of each environment without any explicit supervision. Third, all loss curves converge stably across tasks without signs of divergence or oscillation, validating that the joint optimization of the intra-option, termination, and option-value objectives remains well-conditioned throughout training. We observe no persistent conflict among the jointly optimized objectives: the intra-option surrogate carries the dominant weight among the policy objectives ($\lambda_\omega = 1.0$ vs. $\lambda_\Omega = 0.01$ and $\lambda_\beta = 0.1$), the hierarchical terms act as lightly weighted shaping terms, and the KL-adaptive learning-rate scheduler bounds the size of each intra-option policy update.

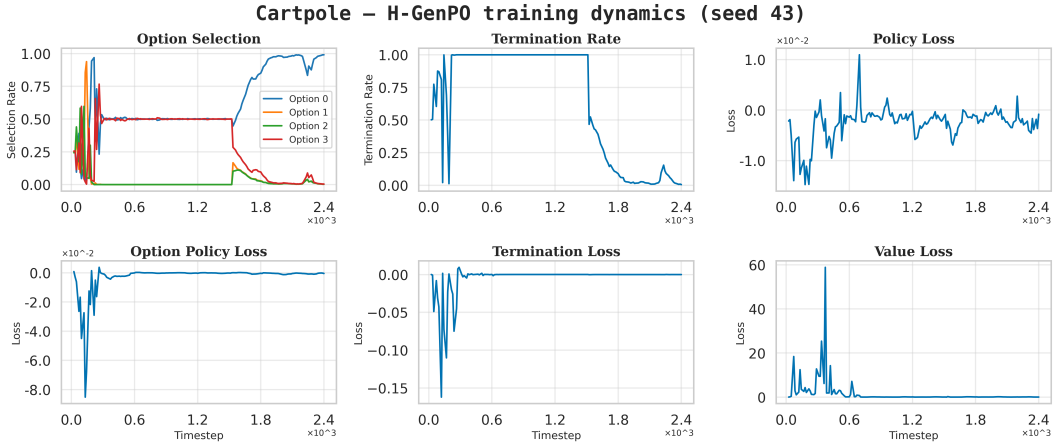


Figure 7: Training dynamics of H-GenPO on `Cartpole`.

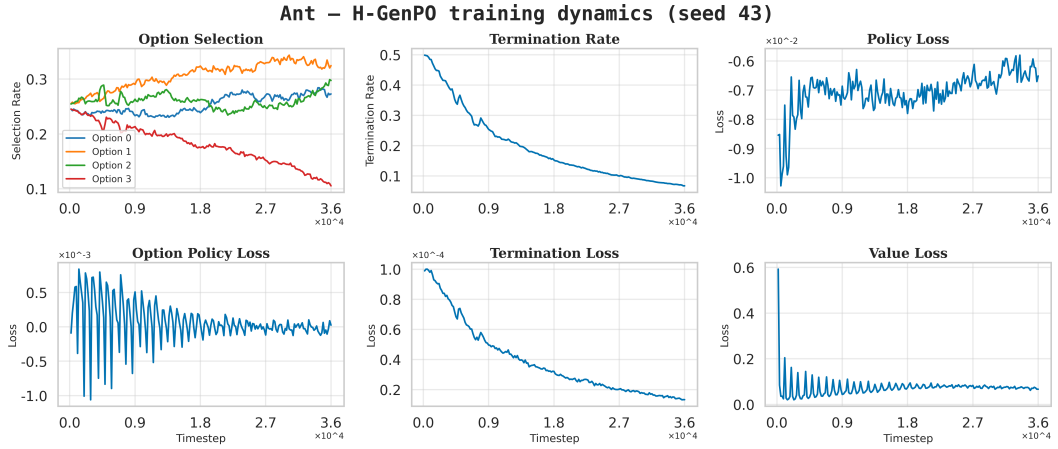


Figure 8: Training dynamics of H-GenPO on `Ant`.

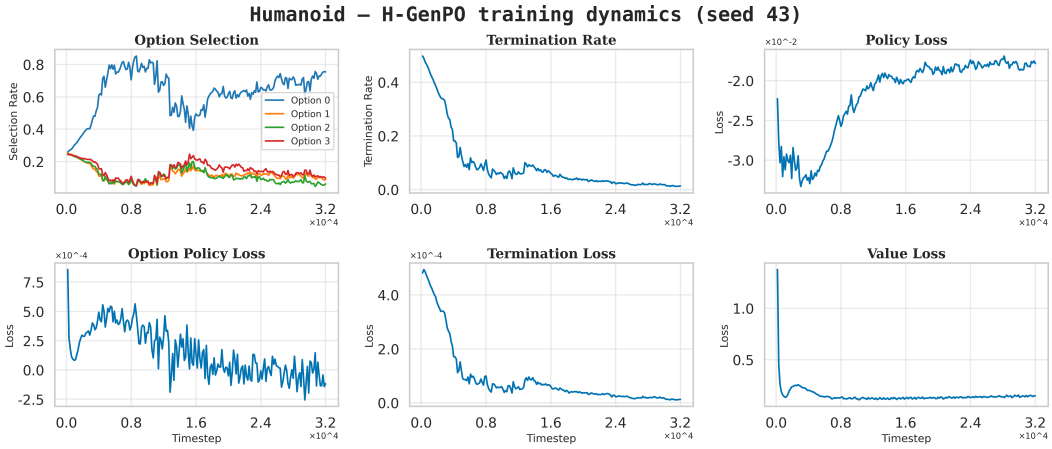


Figure 9: Training dynamics of H-GenPO on `Humanoid`.

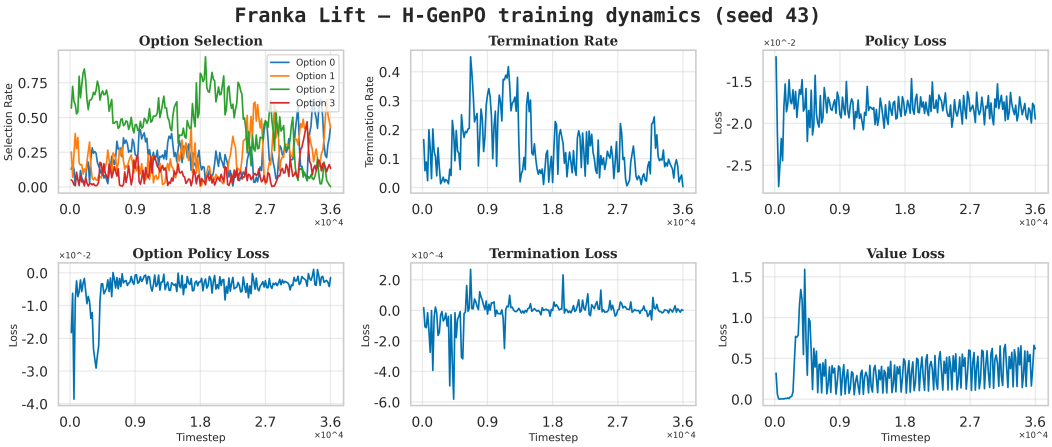


Figure 10: Training dynamics of H-GenPO on `Franka Lift`.

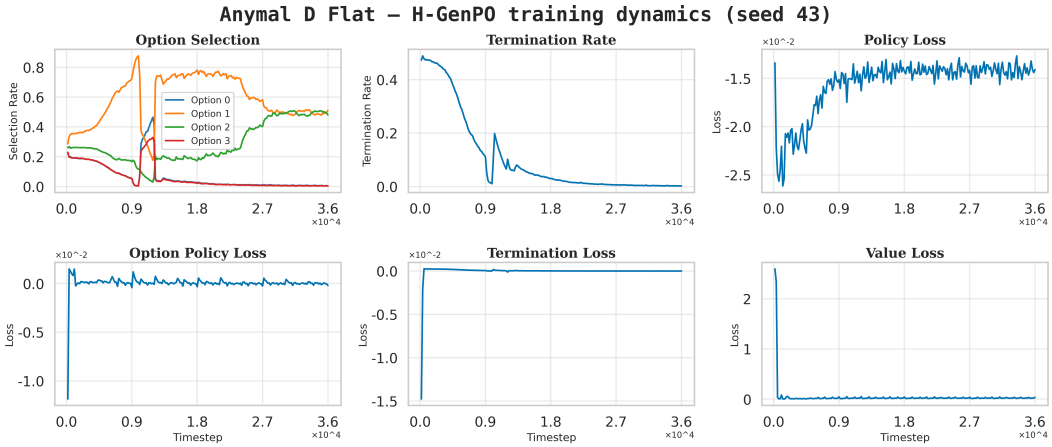


Figure 11: Training dynamics of H-GenPO on `ANYmal-D`.

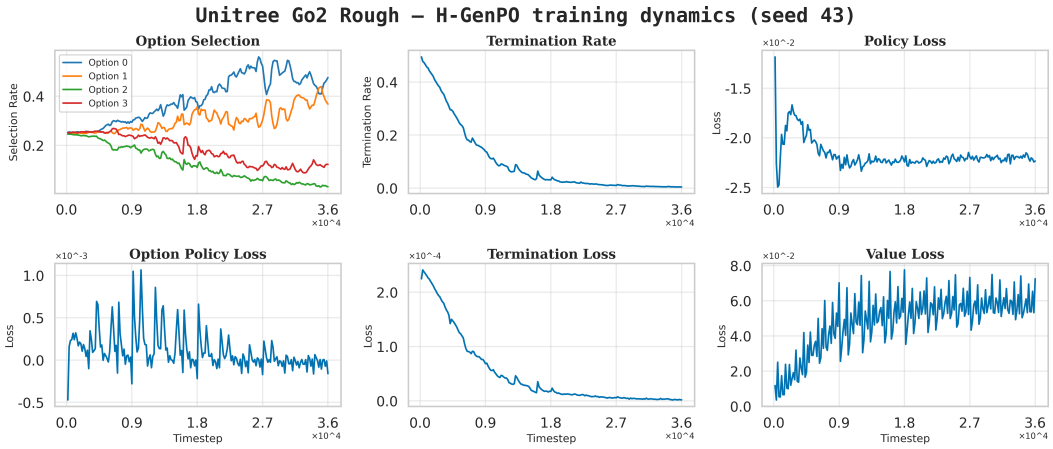


Figure 12: Training dynamics of H-GenPO on `Unitree Go2`.

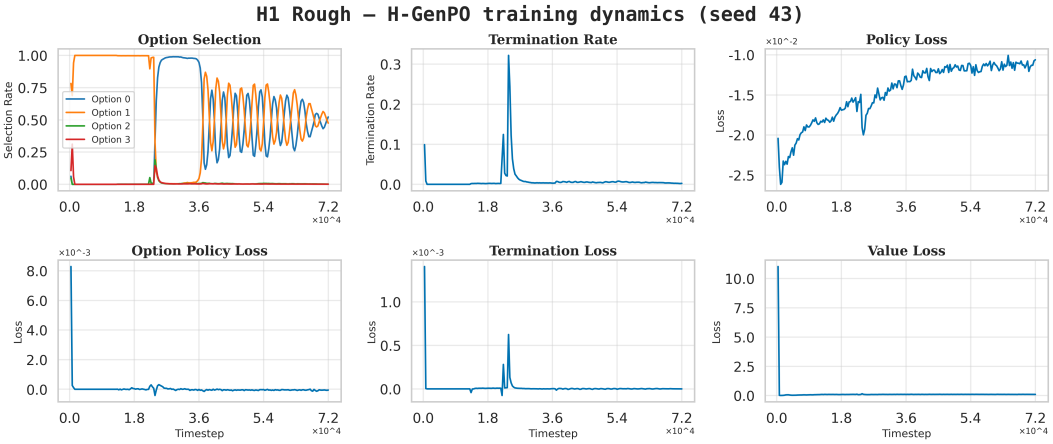


Figure 13: Training dynamics of H-GenPO on `Unitree H1`.

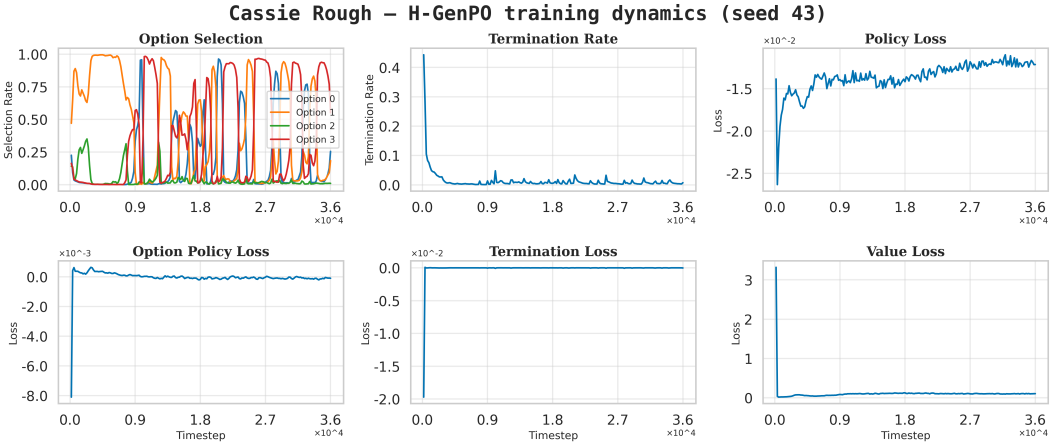


Figure 14: Training dynamics of H-GenPO on `Cassie`.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the contributions and scope (simulated continuous control); claims on the context-dependent tasks are qualified by statistical tests and behavioral metrics (Section 5), and limitations are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This paper include limitation paragraph in last part of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We believe that our implementation details in the appendix are described with sufficient clarity to ensure reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Our implementation will be open-sourced following the publication of this work.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We believe that our implementation details in the appendix are described with sufficient clarity to ensure reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For all experiments, we run 5 random seeds and report the mean and standard deviation (not the standard error) across seeds. For the context-dependent tasks, we additionally report one-sided exact Mann–Whitney U tests on per-seed final rewards (Section 5).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C reports the GPU type, training wall-clock time, inference latency, peak GPU memory, and parameter count of each algorithm.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The proposed method is a general-purpose optimization framework. Due to its purely technical nature and focus on simulation-based robotics, it does not present immediate societal risks or direct ethical concerns.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not pose the aforementioned risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All of the assets in the paper are properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: For the new evaluation benchmark (context-dependent Isaac Lab tasks), we provide a detailed description in the appendix section.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We did not use LLM except for writing, editing, or formatting purpose.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.