FastDSAC: Unlocking the Potential of Maximum Entropy RL in High-Dimensional Humanoid Control

Jun Xue[1,2] **Junze Wang**[2,3,‡] **Shanze Wang**[2]
Xinming Zhang[2] **Yanjun Chen**[2] **Wei Zhang**[2,✉]

[1]School of Computer Science, Shanghai Jiao Tong University, Shanghai, China

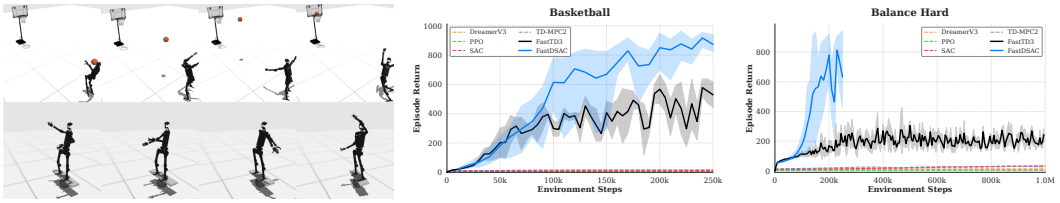
[2]Ningbo Key Laboratory of Spatial Intelligence and Digital Derivative,
Institute of Digital Twin, Eastern Institute of Technology, Ningbo, China

[3]College of Computer Science and Technology,
China University of Petroleum (East China), Qingdao, China

xuejun_dao@sjtu.edu.cn 2307010311@s.upc.edu.cn
{szwang,xmzhang,yychen,zhw}@eitech.edu.cn

Abstract

Scaling Maximum Entropy Reinforcement Learning (RL) to high-dimensional humanoid control remains a fundamental challenge, as the "curse of dimensionality" induces severe exploration inefficiency and training instability. Consequently, highly optimized deterministic policy gradients currently dominate high-throughput regimes. We address this limitation with FastDSAC, a framework that effectively unlocks the potential of maximum entropy stochastic policies for complex continuous control. We introduce Dimension-wise Entropy Modulation (DEM) to autonomously adapt exploration across action dimensions, alongside a continuous distributional critic that mitigates high-dimensional overestimation while avoiding the fixed supports and quantization artifacts of discrete critics. Extensive evaluations on HumanoidBench and a diverse set of continuous control tasks demonstrate that FastDSAC establishes state-of-the-art performance for high-dimensional stochastic policies on the evaluated benchmarks. Our method is competitive with and often outperforms strong deterministic baselines, with returns of 180% and 350% on the challenging *Basketball* and *Balance Hard* tasks, respectively.



(a) FastDSAC's performance on tasks (b) Basketball (c) Balance Hard

Figure 1: **Performance of FastDSAC on very high-dimensional humanoid control tasks.** (a) Visualizations of the challenging *Basketball* and *Balance Hard* environments. (b, c) Evaluation curves comparing FastDSAC with the FastTD3 baseline. FastDSAC is averaged over 5 seeds, with shaded regions spanning the min-max range across seeds. FastDSAC outperforms FastTD3, achieving final returns 1.8× and 3.5× higher on the respective tasks with stronger sample efficiency.

‡Work done during an internship at the Eastern Institute of Technology, Ningbo (EIT).
✉Corresponding to: Wei Zhang(zhw@eitech.edu.cn).

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

1 Introduction

Deep reinforcement learning (RL) has driven significant progress in high-dimensional robotics and has become a practical framework for learning sensorimotor skills in humanoid control [1–5]. As morphological complexity and task demands increase, the field has shifted towards leveraging massively parallel simulation to accelerate training [6–10]. In this high-throughput setting, deterministic policy-gradient methods, especially FastTD3 [5], have largely defined the state of the art. By leveraging the stability of deterministic policies in massively parallelized environments (e.g., IsaacLab [11], MuJoCo Playground [12], and HumanoidBench [3]), these approaches successfully solve challenging locomotion and manipulation tasks that were once intractable. While recent works, such as FlashSAC [13], have attempted to scale stochastic policy algorithms via system-level optimizations, deterministic methods remain the dominant and most reliable baselines for very high-dimensional humanoid whole-body control.

Despite the dominance of deterministic methods, maximum entropy RL algorithms (e.g., SAC [14]) theoretically offer compelling advantages, particularly in escaping local optima and fostering diverse behaviors, which are often absent in deterministic approaches. However, realizing these benefits in high-dimensional humanoid control is limited by the "curse of dimensionality" [15]. Effective exploration in high-dimensional humanoid control depends on accommodating the differing demands of precision-critical and redundant actuators [16]. Poorly matched perturbations can disrupt whole-body coordination. As dimensionality increases, Gaussian perturbations sampled independently at each joint can severely restrict effective movement, a phenomenon termed "vanishing exploration" [17, 18]. Although standard SAC learns state-dependent, dimension-specific variances, adapting exploration to these demands remains challenging across the expanded action volume ($|\mathcal{A}| > 20$). Standard diagonal-Gaussian policies lack a dedicated cross-dimensional structure for regulating exploration: an explicit representation of how its intensity should be coordinated across the action space. Without a principled mechanism to allocate exploration across these dimensions, stochastic agents remain vulnerable to training instability and suboptimal convergence, a limitation shared by several recent strong baselines [3, 5, 19, 18, 13]. This motivates a policy structure that enables the agent to autonomously modulate exploration according to joint-level control demands.

Compounding the exploration difficulty, critic networks in high-dimensional action spaces are particularly prone to severe value overestimation. As the action space expands, critics are frequently queried on state-action pairs lying far from the training distribution (OOD). In these sparse regions, neural approximations suffer from severe extrapolation error, generating spurious Q-value spikes [20, 21]. This issue becomes more acute in high-dimensional settings, where the policy tends to over-exploit such approximation artifacts and shift aggressively toward regions where the critic erroneously predicts high returns [22]. Crucially, standard mitigation strategies, such as clipped double Q-learning, are often insufficient to suppress this overestimation bias in complex dynamics [23, 24]. Even recent parallelized approaches, utilizing either standard architectures or discrete distributional critics [25, 19], struggle to resolve this instability, leading to suboptimal convergence. Beyond overestimation, discrete critics rely on fixed supports and discrete atoms: when reward scales vary widely across tasks, the support range itself becomes an additional tuning burden or design choice, while quantization errors induced by finite atoms can further distort value estimation. Such discretization artifacts further compound extrapolation errors, compromising the accurate value estimation required for high-precision control. This motivates a continuous distributional critic that avoids fixed-support discretization while helping to mitigate high-dimensional overestimation in high-throughput settings [26].

To bridge these gaps, we introduce FastDSAC, a framework that effectively scales maximum entropy stochastic policies to high-dimensional continuous control. FastDSAC integrates two core mechanisms to unlock the scalability and stability of learning in high-dimensional regimes. First, we propose **Dimension-wise Entropy Modulation (DEM)**, which enables the agent to autonomously adapt exploration across action dimensions through state-dependent modulation. By learning normalized weights that attenuate or amplify its base standard deviations, the actor can allocate exploration according to the differing precision requirements of action dimensions. This task-dependent modulation is learned under the original SAC objective, without prescribing joint importance or introducing an auxiliary actor loss. Second, we incorporate a **continuous distributional critic** parameterized by a Gaussian distribution [26]. Distinct from baselines limited to discrete approximations or deterministic estimates, this component models the return distribution continuously, avoiding fixed-support

specification and quantization artifacts while helping mitigate value overestimation arising from high-dimensional extrapolation errors. Our matched critic diagnostic provides direct evidence of reduced positive Q-error on *Balance Hard* (Appendix K). Extensive empirical evaluations on HumanoidBench, MuJoCo Playground, and IsaacLab confirm that FastDSAC consistently matches or outperforms strong baselines across the vast majority of evaluated tasks. It achieves approximately **180%** and **350%** of FastTD3's performance on the challenging *Basketball* and *Balance Hard* tasks, respectively, demonstrating that rigorously designed stochastic policies can achieve outstanding performance in very high-dimensional humanoid robotics. Furthermore, we demonstrate successful zero-shot transfer of the learned policies to a physical Unitree G1 humanoid within an established sim-to-real pipeline. Our implementation is available at `https://github.com/EIT-EAST-Lab/FastDSAC_official`.

2 Preliminaries

We formulate the continuous control problem as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, comprising the state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $p(s'|s,a)$, reward function $r(s,a)$, and discount factor $\gamma \in [0,1)$. Our approach integrates two RL paradigms:

Maximum Entropy Reinforcement Learning. To encourage exploration, we adopt the maximum entropy framework [14]. Instead of solely maximizing scalar returns, the agent learns a policy $\pi(\cdot|s)$ optimizing the expected return augmented by an entropy term:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \Big) \right], \tag{1}$$

where $\mathcal{H}$ is the policy entropy and α is a temperature parameter balancing reward and exploration.

Distributional Reinforcement Learning. Unlike standard methods estimating the expected value $Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)]$, we model the full distribution of the random return $Z^\pi(s,a)$, which follows the distributional Bellman equation:

$$Z^\pi(s,a) \overset{D}{=} r(s,a) + \gamma Z^\pi(s',a'), \tag{2}$$

where $s' \sim p(\cdot|s,a)$, $a' \sim \pi(\cdot|s')$, and $\overset{D}{=}$ denotes distributional equality. Capturing return distribution enables the agent to utilize higher-order moments (e.g., variance) for more stable learning signals.

2.1 Fast Actor-Critic (FastTD3)

Our work builds on the high-throughput recipe of FastTD3 [5], a TD3 variant designed for massive parallelism. Two ingredients are especially relevant for FastDSAC. (1) FastTD3 leverages massively parallel simulation together with large-batch replay updates, which accelerate training and increase data diversity within each critic update. (2) It equips TD3 with the categorical C51 critic [27], which represents returns on a fixed discrete support of N atoms $\{z_i\}_{i=1}^N$ spanning $[v_{\min}, v_{\max}]$. Target estimation uses the projected Bellman update $\boldsymbol{m} = \Phi \mathcal{T} Z_{\theta'}$, where the distributional Bellman operator $\mathcal{T}$ first shifts the return distribution and Φ then projects it back onto the fixed atoms; the full C51 parameterization and optimization objective are summarized in Appendix C.

However, this fixed discretization is limiting in our setting: the support range $[v_{\min}, v_{\max}]$ can become task-dependent when reward scales vary, and finite atoms introduce quantization error that restricts value estimation precision in continuous control tasks requiring fine-grained signal propagation.

2.2 Continuous Distributional Critics (DSAC)

To mitigate the limitations of discrete distributional approximations and value overestimation, we employ the Distributional Soft Actor-Critic (DSAC) framework [24, 26]. DSAC parameterizes the return distribution as a continuous Gaussian $Z_\theta(\cdot|s,a) \sim \mathcal{N}(Q_\theta(s,a), \sigma_\theta^2(s,a))$, thereby avoiding both quantization artifacts and manual support specification. Furthermore, to address the numerical instability of standard KL-divergence gradients, DSAC derives separate gradient estimators for the mean and standard deviation. This separation inversely scales the mean update by variance to reduce step sizes in high-uncertainty regions, thereby curbing overestimation, while stabilizing the variance

update via target clipping $C(\cdot; b)$ to prevent optimization collapse [26]:

$$\nabla_\theta J_Z \approx (\omega + \epsilon_\omega)\mathbb{E}\Bigg[\underbrace{-\frac{y_q^{\min} - Q_\theta}{\sigma_\theta^2 + \epsilon}\nabla_\theta Q_\theta}_{\text{(I) Mean Update with Expected Value}} \quad \underbrace{-\frac{(C(y_z^{\min}; b) - Q_\theta)^2 - \sigma_\theta^2}{\sigma_\theta^3 + \epsilon}\nabla_\theta \sigma_\theta}_{\text{(II) Variance Update with Clipping}} \Bigg]. \tag{3}$$

Here, $y_q^{\min}$ denotes the conservative target Q-value derived from the twin critics, while $y_z^{\min}$ represents a stochastic sample from the target return distribution. The parameter ϵ prevents gradient explosion when the variance σ_θ^2 approaches zero, whereas ϵ_ω prevents gradient disappearance as the scaling factor ω approaches zero. The scaling factor $\omega = \mathbb{E}_{(s,a)\sim\mathcal{B}}[\sigma_\theta(s,a)^2]$ and the clipping boundary b within $C(\cdot; b)$ are adaptively updated based on the running statistics of the learned variance. As detailed in DSAC-T [26], this estimator combines three mechanisms: Expected Value Substituting, Twin Value Distribution Learning, and Variance-Based Critic Gradient Adjustment. Appendix C summarizes their original formulation, while Section 3.2 explains how FastDSAC streamlines them for the high-throughput regime.

3 Methodology

In this section, we present FastDSAC, a high-throughput reinforcement learning framework tailored for high-dimensional continuous control. FastDSAC adapts the actor-critic paradigm to massive parallelization via structural innovations in both policy parameterization and value estimation. The overall architecture diagram of the algorithm and the pseudocode are shown in Appendix D.

Our method consists of three integrated components: (1) An actor employing Dimension-wise Entropy Modulation to autonomously learn exploration modulation across action dimensions (Section 3.1); (2) A streamlined Continuous Distributional Critic that avoids fixed-support specification and quantization artifacts while mitigating overestimation (Section 3.2); and (3) A unified Distributional Soft Policy Iteration (DSPI) loop optimized for large-batch stability (Section 3.3).

3.1 The FastDSAC Actor: Dimension-wise Entropy Modulation

Standard diagonal-Gaussian policies in SAC parameterize the action distribution as $\pi_\phi(a|s) = \mathcal{N}(\mu_\phi(s), \text{diag}(\sigma_\phi^2(s)))$, with a learned, state-dependent standard deviation $\sigma_{\phi,i}(s)$ for each action dimension. In high-dimensional systems, this parameterization lacks a dedicated cross-dimensional structure for regulating exploration according to the differing demands of precision-critical and redundant actuators [16, 3]. To address this, we propose Dimension-wise Entropy Modulation (DEM), a mechanism that enables the agent to autonomously modulate exploration variance across action dimensions. Rather than replacing the standard variance prediction, DEM augments it by learning a dimension-specific scaling weight w_i for each action dimension $i \in \{1, \ldots, |\mathcal{A}|\}$.

Softmax-Normalized Dimension-wise Modulation. Given the actor backbone features $h(s)$, we compute the modulation weights using a temperature-scaled Softmax operation:

$$l_i = f_\phi^{\text{logits}}(h(s))_i, \quad w_i = \frac{\exp(l_i/\tau)}{\sum_{j=1}^{N}\exp(l_j/\tau)} \cdot N, \tag{4}$$

where τ is a temperature hyperparameter that regulates the concentration of the weight distribution: a lower τ encourages more concentrated relative modulation, while a higher τ pushes the weights toward one. $N = |\mathcal{A}|$ denotes the action dimension. Multiplying the Softmax output by N gives $\sum_{i=1}^{N} w_i(s) = N$, so the mean modulation weight is one. This normalization couples the relative weights, modulating exploration variance across action dimension; it does not conserve total exploration variance or policy entropy. The final standard deviation for the i-th action dimension is modulated as:

$$\sigma_{\phi,i}(s) = w_i(s) \cdot \exp(\hat{\sigma}_{\phi,i}(s)), \tag{5}$$

where $\hat{\sigma}_{\phi,i}(s)$ represents the base log-standard deviation predicted by the network. The subsequent action sampling and policy evaluation follow the standard SAC reparameterization trick.

Interpretation: Autonomous Soft Exploration Pruning. By dynamically adjusting w_i, the agent can selectively attenuate or amplify exploration relative to its learned base standard deviations. We

call this mechanism *soft exploration pruning*: DEM autonomously attenuates exploration on precision-critical joints to prevent instability, effectively pruning unstable exploration search directions. This modulation emerges through the original maximum-entropy SAC objective, without manual joint-importance priors or kinematic specifications. This allows the policy to discover effective exploration and control patterns that may be non-intuitive from a human design perspective. Section 4.3 examines the learned modulation patterns and their corresponding effective exploration variance, which allows the agent to maintain high-precision control over stability-critical joints while exploring others.

Diversity-driven Population. To diversify exploration across the parallel population, we employ a Heterogeneous Exploration scheme adapted from FastTD3 [5]. We assign a distinct scaling factor $\beta_e \sim U[\beta_{\min}, \beta_{\max}]$ to each environment e. Unlike prior works that scale noise magnitude [8, 5], we apply β_e to the modulation logits to vary the distribution shape:

$$w_{i,e} \propto \exp\left(\frac{l_i \cdot \beta_e}{\tau}\right). \tag{6}$$

This exposes parallel environments to different modulation patterns: some have weights closer to one, while others emphasize fewer action dimensions.

3.2 The FastDSAC Critic: Streamlined Distributional Learning

To address the limitations of discrete critics [27, 5], we adopt the Continuous Gaussian Critic from DSAC-T [26], parameterizing the return distribution as $Z_\theta(\cdot|s,a) \sim \mathcal{N}(Q_\theta(s,a), \sigma_\theta^2(s,a))$, to avoid fixed-support specification and discretization artifacts while mitigating overestimation.

While original DSAC-T employs complex variance-based gradient adjustments for low-data regimes, we find these constraints restrict accurate variance modeling in our settings. Leveraging the natural stability provided by large batch sizes, we streamline the objective by removing variance clipping boundaries, while retaining twin-critic target, Expected Value Substitution, and Gradient Scaling.

The parameters θ are updated by minimizing divergence between predicted and target distributions. Following DSAC-T [26], we decompose the gradient into mean- and variance-driven components:

$$\nabla_\theta J_Z \approx \omega \cdot \mathbb{E}\Bigg[\Bigg(\underbrace{-\frac{y_q^{\min} - Q_\theta}{\sigma_\theta^2 + \epsilon}\nabla_\theta Q_\theta}_{\text{(I) Stabilized Mean Update}} \underbrace{-\frac{(y_z^{\text{sample}} - Q_\theta)^2 - \sigma_\theta^2}{\sigma_\theta^3 + \epsilon}\nabla_\theta \sigma_\theta}_{\text{(II) Natural Variance Update}}\Bigg)\Bigg]. \tag{7}$$

Here, term (I) leverages the Expected Value Substitution: we anchor the mean update to the conservative target mean $y_q^{\min} = r + \gamma(\min_{j=1,2} Q_{\theta' j}(s', a') - \alpha \log \pi(a'|s'))$, rather than stochastic samples. In our high-dimensional setting, this deterministic target proved more stable than directly using stochastic return samples for the mean update. Term (II) updates the variance to capture the stochasticity induced by environmental dynamics and policy exploration. Unlike the original implementation which clips the target sample y_z^{sample} within a dynamic boundary, we remove this constraint to allow the critic to learn the true scale of environmental stochasticity without bias. The scalar $\omega \approx \mathbb{E}_{\mathcal{B}}[\sigma_\theta^2]$ is the gradient scaling factor handling reward scale sensitivity [26].

3.3 Distributional Soft Policy Iteration (DSPI)

We integrate the actor (Section 3.1) and critic (Section 3.2) into a unified Distributional Soft Policy Iteration (DSPI) loop. This process alternates between evaluating the policy's return distribution and improving the policy via entropy-regularized maximization. Theoretically, the DSPI framework is guaranteed to converge to the optimal policy, as established in DSAC [24].

Distributional Soft Policy Evaluation. In the evaluation step, we update the twin critic networks using the streamlined gradient estimator in Eq. 7. By maintaining a continuous Gaussian parameterization, the critic avoids fixed-support specification and discrete quantization artifacts, and models the full return distribution, capturing aleatoric uncertainty alongside the expected return. As the gradient magnitude is inversely scaled by the estimated variance, high uncertainty on OOD actions naturally dampens updates, reducing high-dimensional overestimation.

Distributional Soft Policy Improvement. In the improvement step, the actor π_ϕ is updated to maximize both the expected return and entropy. Following the reparameterization trick [14], the actor

utilizes the mean estimate of the distributional critic, $Q_\theta(s,a) = \mathbb{E}_{Z\sim Z_\theta}[Z]$, to guide policy updates. The actor is optimized by minimizing:

$$\mathcal{L}_{\text{actor}}(\phi) = \mathbb{E}_{s\sim\mathcal{B},a\sim\pi_\phi}\left[\alpha \log \pi_\phi(a|s) - Q_\theta(s,a)\right]. \tag{8}$$

DEM reparameterizes the Gaussian standard deviations entering $\log \pi_\phi(a|s)$ without altering the original SAC actor objective or introducing auxiliary losses. Because both the base scales and modulation weights are optimized end-to-end by the gradient of this standard reward maximization objective with the original entropy term, this elegantly guides the agent to autonomously modulate exploration while maximizing rewards. Simultaneously, we dynamically adjust the temperature parameter α via gradient descent to satisfy the target entropy constraint $\mathcal{H}$ [14]:

$$\alpha \leftarrow \alpha - \eta \cdot \mathbb{E}_{s\sim\mathcal{B},a\sim\pi_\phi}\left[-\log \pi_\phi(a|s) - \mathcal{H}\right] \tag{9}$$

Implementation Recipe Summary. Full implementation details are provided in Appendix A.4. Crucially, we set the target entropy to $\mathcal{H} = 0$ across all tasks. Compared with the standard heuristic ($\mathcal{H} = -|\mathcal{A}|$), this setting supports continued stochastic exploration, while DEM learns dimension-wise exploration modulation under the same SAC objective. Additionally, we apply LayerNorm only in the very high-dimensional HumanoidBench domain, where it serves as an important stabilizer for training, whereas smaller action spaces (MuJoCo Playground & IsaacLab) achieve SOTA performance without such regularization. Furthermore, unlike the original DSAC-T [26], which requires a large numerical stabilizer (bias $= 0.1$) to prevent division-by-zero, our large-batch regime allows drastically reducing the continuous critic's numerical stabilizer (bias $= 10^{-6}$), unlocking high-fidelity value estimation for low-variance states. Finally, while a fixed DEM temperature ($\tau = 1$) is our robust default, we introduce an end-to-end learnable variant, FastDSAC (auto-τ), evaluated only on *Balance Hard*, *Run*, and *Hurdle*. We leave a broader study of this variant to future work. Ablation experiments on target entropy and LayerNorm are in Appendix N and M.

4 Experiments

Our empirical evaluation aims to validate the scalability, algorithmic mechanisms, and real-world viability of FastDSAC in high-dimensional continuous control. Specifically, we design our experiments to address four key questions: **Q1 (Comparative Performance):** Can FastDSAC match or outperform strong deterministic, standard stochastic, and model-based baselines across complex domains? **Q2 (Actor Mechanism):** Does DEM improve performance and training stability beyond the continuous critic, and what dimension-wise modulation patterns does the actor learn? **Q3 (Critic Parameterization):** Does continuous Gaussian modeling improve training stability and mitigate value overestimation relative to discrete (C51) approximations in high-dimensional spaces? **Q4 (Sim-to-Real Deployment):** Can the stable, high-precision control policies discovered by FastDSAC in simulation transfer zero-shot to physical humanoid hardware?

To address these questions, we evaluate our framework across HumanoidBench, MuJoCo Playground, and IsaacLab, spanning diverse tasks from locomotion to complex whole-body manipulation, and ultimately deploy it on the Unitree G1.

4.1 Experimental Setup

Benchmarks. We aligned with the FastTD3 suite [5], evaluating on: (1) HumanoidBench [3]: 25 representative tasks spanning locomotion and whole-body manipulation, with very high-dimensional action spaces ($|\mathcal{A}| = 61$). (2) MuJoCo Playground [12] & IsaacLab [11]: 10 tasks across diverse physics engines to verify robustness. While their action spaces are smaller than those of HumanoidBench ($|\mathcal{A}| < 30$), they still represent challenging high-dimensional control settings in RL. We therefore omitted Layer Normalization, as FastDSAC retains inherent stability without it.

Baselines & Protocol. We benchmarked against FastTD3 (deterministic SOTA) and FastSAC (parallelized stochastic baseline) [5]. To ensure rigorous comparison in all tasks, we matched FastSAC to FastDSAC in its Layer Normalization setting and target entropy $\mathcal{H} = 0$, hereafter denoted as *FastSAC (Standard)*. For HumanoidBench, we incorporated official benchmark results for DreamerV3, TD-MPC2, SAC, and PPO; for MuJoCo Playground and IsaacLab, we additionally evaluated against PPO to contextualize performance against on-policy methods in these domains.

To guarantee fair evaluation, we strictly inherited the FastTD3 high-throughput training protocol (e.g., replay capacity, update schedules, parallel environment counts) for all corresponding tasks. We report

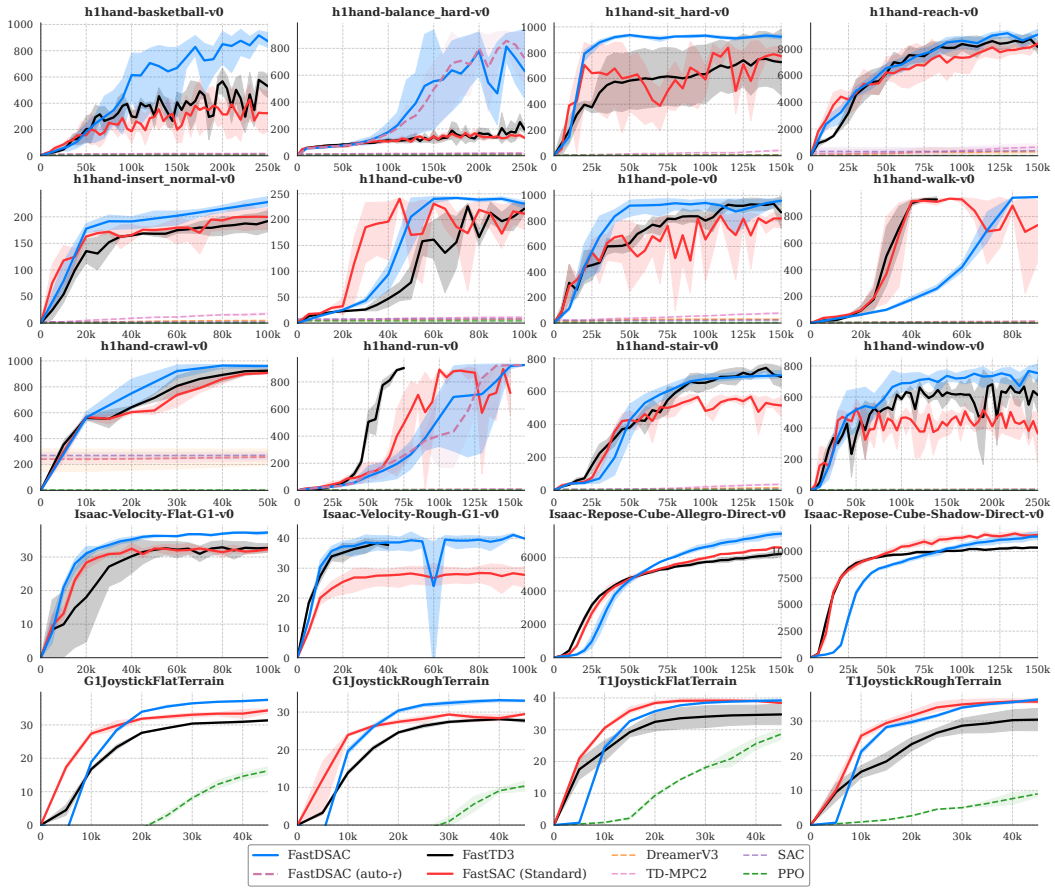


Figure 2: **Comparative evaluation on high-dimensional continuous control benchmarks.** Learning curves across selected tasks from **HumanoidBench** (top three rows), **IsaacLab** (fourth row), and **MuJoCo Playground** (bottom row). **FastDSAC** matches or often outperforms **FastTD3** and **FastSAC (Standard)**. It performs strongly on precision-demanding (e.g., *Basketball*, *Insert*) and stability-critical (e.g., *Balance Hard*) tasks, while retaining strong dexterous manipulation and rough-terrain locomotion performance. FastDSAC (auto-τ) is evaluated only on *Balance Hard*, *Run*, and *Hurdle*; all other FastDSAC curves use the fixed default $\tau = 1$.

mean returns aggregated over 3 seeds, with shaded regions indicating the min-max range. Notably, we use 5 seeds for *Basketball* and *Balance Hard* to demonstrate consistency, while maintaining 3 seeds for all other tasks to match FastTD3 and HumanoidBench protocols. Crucially, for FastDSAC-specific hyperparameters, we rigorously enforce **unified algorithmic configuration** within each evaluation domain (i.e., parameters remain strictly identical across all tasks within a given benchmark), in contrast to FastTD3, whose distributional critic may require task-specific tuning of the C51 support range. Full protocol details and hyperparameter configurations are summarized in Appendix A. In the sample-efficiency figures, the x-axes report simulator steps per environment: one plotted step advances all N_{env} parallel environments once, corresponding to N_{env} transitions. The benchmark-specific environment counts are given in Table 1.

4.2 Main Results (Q1)

We evaluated FastDSAC across 35 tasks spanning diverse dynamics. Figure 2 shows learning curves for 20 representative tasks from HumanoidBench, MuJoCo Playground, and IsaacLab (see Appendix P for full results). We observe three distinct performance regimes:

Strong Performance on Complex Coordination & Manipulation. FastDSAC achieves state-of-the-art performance on tasks demanding precise actuation and whole-body coordination, outperforming FastTD3 and FastSAC (Standard) on challenging tasks such as *Basketball*, *Balance Hard*, *Insert*

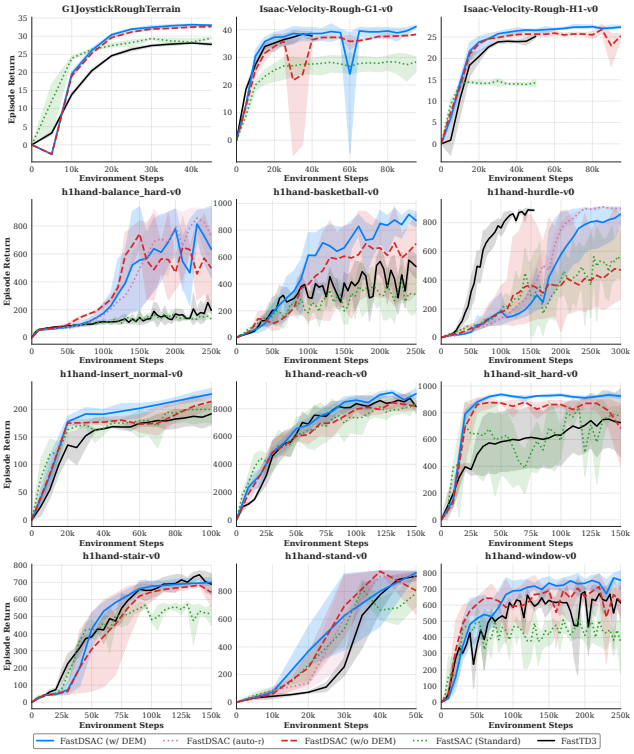


Figure 3: **DEM improves performance and stability in high-dimensional control.** Learning curves across 12 tasks from three benchmarks comparing FastDSAC, its ablations (w/o DEM and auto-τ), and FastTD3/FastSAC. Among these tasks, auto-τ is evaluated only on *Balance Hard* and *Hurdle*.

Normal, *Sit Hard*, and *Window*. This gap highlights the limitation of deterministic policies in exploring intricate control manifolds. By effectively adapting exploration, FastDSAC discovers superior whole-body coordination strategies that deterministic baselines fail to reach. The DEM ablation below examines the contribution of this learned dimension-wise modulation.

The Exploration-Exploitation Trade-off in Locomotion. In dynamic locomotion (e.g., *Run*, *slide*), FastDSAC rises more slowly than FastTD3 but converges to higher final returns. This pattern is consistent with the maximum entropy objective: DEM may initially allocate variance to explore diverse gait patterns before convergence, whereas FastTD3 can settle earlier on lower-return gaits. Even on complex whole-body control tasks such as *Stair*, FastDSAC remains competitive with FastTD3, showing that our framework preserves the precision needed for rigorous terrain adaptation.

Robustness Across Simulators. FastDSAC scales consistently across physics engines. In MuJoCo Playground (e.g., *G1/T1 Joystick*), it maintains a clear lead. In IsaacLab (*Velocity-Rough*, *Repose-Cube-Allegro/Shadow*), FastDSAC outperforms FastTD3 after an initial adaptation phase, validating its robustness across dynamics. Although FastSAC (Standard) converges faster on the lower-dimensional *Repose-Cube-Shadow* task ($|\mathcal{A}| \approx 24$), FastDSAC reaches comparable performance, suggesting that while unmodulated exploration can suffice for specific hand-only tasks, our framework remains the stronger generalist across the broader benchmark suite.

4.3 Ablation Studies and Analysis (Q2 & Q3)

To assess component necessity, we conducted ablation studies across 3 benchmarks on representative tasks spanning rough-terrain locomotion, dynamic whole-body coordination, object interaction, and manipulation.

Q2: Necessity and Mechanism of Dimension-wise Entropy Modulation (DEM). To isolate the effect of DEM, we compare FastDSAC against an ablated variant (w/o DEM) that replaces the entropy-modulated actor with a standard diagonal Gaussian policy. As shown in Figure 3, across 12 diverse tasks from all 3 benchmarks, the full FastDSAC consistently yields higher asymptotic returns. Although the ablated agent often outperforms standard baselines, likely due to the continuous

distributional critic, removing DEM degrades performance, reduces training stability, and increases inter-seed variance in tasks with challenging dynamics (e.g., *Balance Hard*, *Basketball*, and *Hurdle*). Beyond stabilizing the baseline performance, the structure of DEM also supports end-to-end optimization of its temperature parameter, which can further improve returns on tasks with different exploration demands. These results show DEM is indispensable for managing high-dimensional exploration and reducing seed sensitivity. Appendix L further evaluates DEM with eight training seeds per method on six HumanoidBench tasks and five on IsaacLab *Velocity-Rough-G1*, reporting IQM returns and 95% bootstrap confidence intervals with the continuous critic held fixed.

Visualizing Autonomous Exploration Modulation. To understand how DEM shapes exploration, we analyzed the evolution of learned weights w_i during a *Basketball* episode (see the qualitative comparison in Figure 4 and the heatmap in Appendix I). The visualization reveals a task-aligned, dimension-wise exploration modulation pattern that dynamically shifts the exploration focus across its action space. **1) Selective Amplification:** We observed a prominent high-weight band (reaching $w_i \approx 8.0$) corresponding to task-irrelevant joints, such as the left thumb (`lh_A_THJ4`) and wrist, which act as entropy sinks. **2) Precision for Impact and Balance:** During the critical interaction phase (Steps 10-13), as the robot impacts the ball, weights for the legs, torso, and right arm drop to near-zero. This downward modulation persists immediately post-impact to counteract momentum, effectively attenuating the base standard deviations to minimize exploration variance. This demonstrates DEM's ability to autonomously reduce disruptive exploration and support whole-body stabilit.

Interpretation: Emergent Strategy with Learned Exploration Allocation. The agent autonomously discovers an unconventional "body-rebound" strategy, using the torso rather than the hands to redirect the ball. While counter-intuitive to human design, this behavior maximizes return by prioritizing post-throw stability over risky dexterity. DEM enables this strategy by adapting exploration across action dimensions: through *soft exploration pruning*, it autonomously attenuates noise on the stability-critical core body while preserving greater exploration for redundant hand joints. Across 25 checkpoints from four tasks, our effective-variance analysis confirms this pattern, showing lower per-dimension exploration variance in the legs and torso than in the hands (Appendix I). This reveals that the agent treats task-irrelevant actuators as a task-dependent *entropy sink*: it softly prunes unstable and risky search directions in the core to maintain balance, while retaining greater stochasticity in the extremities. This body-group allocation is learned entirely without prescribed joint importance and persists robustly across random seeds.

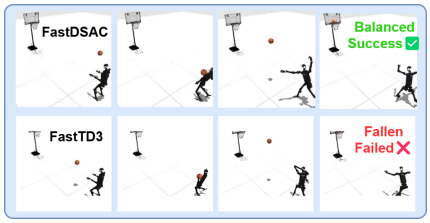


Figure 4: **Qualitative comparison on *Basketball*.** FastDSAC throws well and stays stable, while FastTD3 loses balance and fails.

Contrast with Deterministic Baseline. FastTD3 attempts an intuitive but unstable hand-catch strategy (Figure 4). However, it fails to balance manipulation with whole-body stability; the agent hits the ball but then falls, resulting in suboptimal returns (≈ 500). In contrast, FastDSAC's entropy modulation exploits the reward structure more effectively and finds a more stable solution. More broadly, this result suggests that in high-dimensional RL, the return-maximizing strategy need not align with human-intended behavior, and may instead favor whole-body survival over risky dexterity.

Q3: Continuous vs. Discrete Distributional Critics.

To isolate the effect of critic parameterization, we compared FastDSAC against a discrete baseline, FastSAC (C51+DEM), the C51 variant of FastSAC [28] augmented with our DEM module, using the same training pipeline and actor design. Figure 5 shows that FastDSAC outperforms both FastSAC (Standard) and its discrete variant, FastSAC (C51+DEM), on all nine evaluated tasks, including approximately $2\times$ higher returns on *Balance Hard*. Since the actor design and the overall training framework are otherwise matched, these results suggest that the continuous Gaussian critic supports more stable learning and provides a more effective value estimator than the discrete C51 critic in our high-throughput control regime. A likely reason is that the continuous critic avoids fixed-support specification and atom quantization, which may introduce representation errors that affect value learning in high-dimensional control. Consistent with this view, Appendix J shows on *Basketball* that the performance of C51+DEM is highly sensitive to the choice of support range. Moreover, prior DSAC analyses [26] show that distributional critics can reduce value overestimation, which becomes

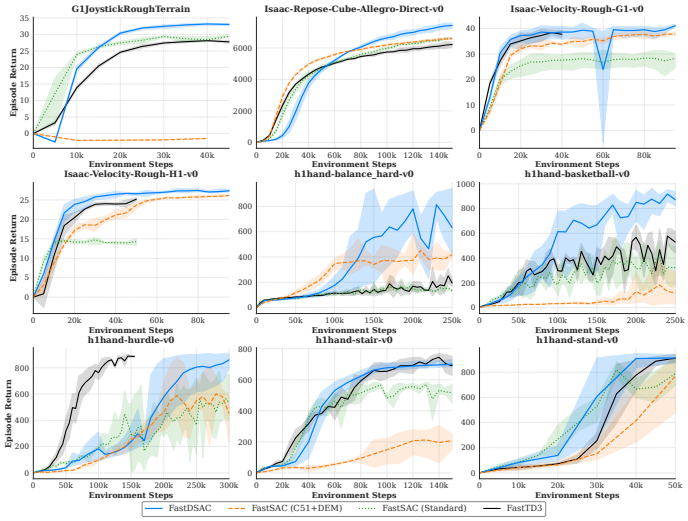


Figure 5: **Continuous critics outperform discrete C51 critics on 9 high-dimensional tasks.** Curves show the mean episode return of FastDSAC compared to FastSAC (C51+DEM) and FastSAC (Standard).

particularly important in high-dimensional action spaces. Our matched *Balance Hard* diagnostic likewise finds reduced positive Q-error and more conservative value estimates for FastDSAC relative to C51+DEM, supporting the role of its critic design in mitigating overestimation (Appendix K).

4.4 Sim-to-Real Validation on Unitree G1 (Q4)

To demonstrate zero-shot transfer to physical hardware, we deploy FastDSAC on a Unitree G1 humanoid, following the mature sim-to-real pipeline in [28]. Evaluated on joystick-conditioned locomotion and a complex "squat-carry-walk-squat" motion-tracking task, FastDSAC requires minimal adaptation: core hyperparameters remain unified, adjusting only task-specific settings (e.g., multi-step returns, γ). Running directly on the G1's onboard compute, the policies show zero-shot stability and robustness to moderate physical pushes (Figure 6). These deployments provide a qualitative zero-shot feasibility demonstration of FastDSAC within an established sim-to-real pipeline. See Appendix B for motion-tracking and backward locomotion results.

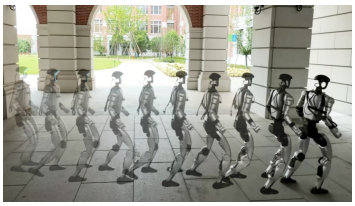


Figure 6: **Sim-to-real on Unitree G1.** Zero-shot joystick-conditioned forward walking on the physical G1 using a mature sim-to-real pipeline.

5 Conclusion

In this work, we challenge the dominance of deterministic policy gradients in high-dimensional humanoid control by introducing FastDSAC, a framework that unlocks the scalability of maximum entropy RL. FastDSAC addresses the curse of dimensionality by integrating Dimension-wise Entropy Modulation (DEM) to autonomously adapt exploration across action dimensions, together with a continuous distributional critic to mitigate high-dimensional overestimation and avoid discretization artifacts. Across 35 diverse tasks, our method establishes state-of-the-art performance for high-dimensional stochastic policies on the evaluated benchmarks, remaining competitive with and often outperforming strong baselines, achieving approximately 180% and 350% of their performance on challenging coordination tasks. Furthermore, FastDSAC policies successfully transfer zero-shot to a physical humanoid robot within an established sim-to-real pipeline. Our findings show that a rigorously designed stochastic policy is not only viable for the high-throughput regime, but can also achieve strong performance in high-precision humanoid control, bridging the gap between broad exploration and precise control. Detailed discussion of limitations, extended future directions, and broader societal impacts is provided in Appendix Q.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62503251) and was partially supported by the 2035 Key Research and Development Program of Ningbo City under Grant No. 2024Z127. The authors declare no competing interests.

References

[1] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

[2] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

[3] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. HumanoidBench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In *Robotics: Science and Systems*, 2024.

[4] Zifeng Zhuang, Diyuan Shi, Runze Suo, Xiao He, Hongyin Zhang, Ting Wang, Shangke Lyu, and Donglin Wang. TDMPBC: Self-imitative reinforcement learning for humanoid robot control. *arXiv preprint arXiv:2502.17322*, 2025.

[5] Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. FastTD3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

[6] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 2022. URL `https://proceedings.mlr.press/v164/rudin22a.html`.

[7] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[8] Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel q-learning: Scaling off-policy reinforcement learning under massively parallel simulation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19440–19459. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/li23f.html`.

[9] Arth Shukla. Speeding up SAC with massively parallel simulation, March 2025. URL `https://arthshukla.substack.com/p/speeding-up-sac-with-massively-parallel`.

[10] Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=7IzeL0kflu`.

[11] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan

Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[12] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo playground, 2025. URL https://arxiv.org/abs/2502.08844.

[13] Donghu Kim, Youngdo Lee, Minho Park, Kinam Kim, Takuma Seno, I. Made Aswin Nahrendra, Sehee Min, Daniel Palenicek, Florian Vogt, Danica Kragic, Jan Peters, Jaegul Choo, and Hojoon Lee. FlashSAC: Fast and Stable Off-Policy Reinforcement Learning for High-Dimensional Robot Control. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.099.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[15] Mario Köppen. The curse of dimensionality. In *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, volume 1, pages 4–8, 2000.

[16] Wenbo Zhang and Hengrui Cai. Where to intervene: Action selection in deep reinforcement learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=D3au9XkWuy.

[17] Pierre Schumacher, Thomas Geijtenbeek, Vittorio Caggiano, Vikash Kumar, Syn Schmitt, Georg Martius, and Daniel F.B. Haeufle. Emergence of natural and robust bipedal walking by learning from biologically plausible objectives. *iScience*, 28(4):112203, 2025. ISSN 2589-0042. doi: 10.1016/j.isci.2025.112203. URL https://www.sciencedirect.com/science/article/pii/S258900422500464X.

[18] Yunyue Wei, Chenhui Zuo, and Yanan Sui. Scalable exploration for high-dimensional continuous control via value-guided flow. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 54895–54913, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5a05d477fef5a43c805d3377700efa40-Paper-Conference.pdf.

[19] Zilin Kang, Chonghua Liao, Tingqiang Xu, and Huazhe Xu. Entropy regularizing activation: Boosting continuous control, large language models, and image classification with activation as entropy constraints. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 31978–32026, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/36636f1009939960aa2f9e28d28c8a7d-Paper-Conference.pdf.

[20] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PczQtTsTIX.

[21] Fabian Otto, Philipp Becker, Ngo Anh Vien, and Gerhard Neumann. Efficient off-policy learning for high-dimensional action spaces. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JDzTI9rKls.

[22] Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 109915–109940. Curran Associates, Inc., 2025. doi: 10.52202/085713-3673. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9eecce65f1f93d7a3e7ae677c31942ab-Paper-Conference.pdf.

[23] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5556–5566. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kuznetsov20a.html.

[24] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. Distributional Soft Actor-Critic: Off-Policy Reinforcement Learning for Addressing Value Estimation Errors. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6584–6598, 2022. doi: 10.1109/TNNLS.2021.3082568.

[25] Johan Obando-Ceron, Walter Mayor, Samuel Lavoie, Scott Fujimoto, Aaron Courville, and Pablo Samuel Castro. Simplicial embeddings improve sample efficiency in actor–critic agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mCpq1GCKxA.

[26] Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, Shengbo Eben Li, Chang Liu, Ya-Qin Zhang, Bo Cheng, and Keqiang Li. Distributional soft actor-critic with three refinements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3935–3946, 2025. doi: 10.1109/TPAMI.2025.3537087.

[27] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

[28] Younggyo Seo, Carmelo Sferrazza, Juyue Chen, Guanya Shi, Rocky Duan, and Pieter Abbeel. Learning sim-to-real humanoid locomotion in 15 minutes, 2025. URL https://arxiv.org/abs/2512.01996.

[29] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1577–1594. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/ball23a.html.

[30] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 113038–113071. Curran Associates, Inc., 2024. doi: 10.52202/079017-3592. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/cd3b5d2ed967e906af24b33d6a356cac-Paper-Conference.pdf.

[31] Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 50818–50858. Curran Associates, Inc., 2025. doi: 10.52202/085713-1696. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/48d6cbeeb8203421bc8ad1c05a9a26ee-Paper-Conference.pdf.

[32] Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 148719–148748, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/f0862823b6d749fe61bfae68dec2c2a9-Paper-Conference.pdf.

[33] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

APPENDIX CONTENTS

A Implementation Details and Hyperparameters 16

A.1 High-Throughput Training Protocol 16

A.2 Unified FastDSAC Hyperparameters 16

A.3 Baseline Configurations 16

A.4 Detailed Implementation Recipe 16

B Extended Details on Sim-to-Real Validation 18

C Supplementary Background on FastTD3 and DSAC-T 19

D Architecture and Pseudocode of FastDSAC 21

E Related Work 22

F Comparison with Contemporary Works 23

G Computational Resources 24

H Computational Efficiency and Wall-Clock Time 24

I Heatmaps of Basketball Task 27

J C51 Support Sensitivity on *Basketball* 29

K Post-Training Critic Diagnostics 29

L Multi-Seed Ablation of DEM 30

M Ablation Study on Layer Normalization 31

N Ablation Study on Target Entropy 32

O Sensitivity to DEM Temperature 33

P Full Results 34

Q Discussion of Limitations, Future Directions and Impact 35

A Implementation Details and Hyperparameters

A.1 High-Throughput Training Protocol

To ensure a rigorous and fair comparison, FastDSAC strictly inherits the high-throughput training protocol from FastTD3 [5]. All domain-specific configurations, including parallel environment counts, replay buffer sizes, and update schedules (Update-to-Data (UTD) ratios), are matched to the official baseline settings. This protocol-level alignment ensures that any performance gains are attributable to the algorithmic innovations of FastDSAC rather than to discrepancies in computational budget or training schedules.

Table 1: Domain-specific training protocols inherited from FastTD3 [5]. These parameters are defined by the underlying high-throughput framework rather than being tuned for FastDSAC.

Parameter	**HumanoidBench**	**Isaac-Velocity**	**Isaac-General**	**MuJoCo PG**
Parallel Environments (N_{env})	128	4,096	4,096	1,024
Multi-step Return (N_{steps})	1	8	1	1
Updates per Step ($N_{updates}$)	2	4	2	2
Discount Factor (γ)	0.99	0.99	0.99	0.97
Replay Buffer Size	5×10^4	5×10^3	10^4	10^4
AMP / AMP Dtype	True / bf16	True / bf16	True / bf16	True / bf16
Batch Size	32,768	32,768	32,768	32,768

A.2 Unified FastDSAC Hyperparameters

In contrast to the per-task tuning often required by stochastic RL, FastDSAC utilizes a **unified algorithmic configuration** within each benchmark domain. As detailed in Table 2, core parameters such as DEM temperature, target entropy, and log-std bounds are kept strictly constant across all tasks within a given domain.

The differences in α_{init} and LayerNorm between HumanoidBench and lower-dimensional simulators (IsaacLab/MuJoCo PG) reflect basic adaptations to action dimensionality ($|\mathcal{A}| = 61$ vs. $|\mathcal{A}| < 30$) and are consistent with standard literature on high-dimensional SAC [29, 30].

Table 2: Unified FastDSAC algorithmic hyperparameters. These values define our core method and are fixed across all tasks within their respective domains.

Hyperparameter	**HumanoidBench**	**IsaacLab / MuJoCo PG**
DEM Temperature (τ)	1.0 (Default)	1.0 (Default)
Log-Std Max / Min	$1.0 \,/\, -10$	$1.0 \,/\, -10$
Target Entropy ($\mathcal{H}$)	0	0
Initial Alpha (α_{init})	0.001	0.01
Layer Normalization	Enabled	Disabled
Polyak Update Rate (τ_{polyak})	0.005	0.005
Weight Decay	10^{-4}	10^{-4}
AdamW Betas	$(0.9, 0.95)$	$(0.9, 0.95)$
Critic Stabilizer (ϵ)	10^{-6}	10^{-6}

A.3 Baseline Configurations.

Results for the FastTD3 baseline are taken directly from the original paper [5]. For the stochastic baseline, we use the FastSAC branch provided in the FastTD3 codebase and rerun it on the benchmark tasks considered in this paper. At the protocol level, FastSAC follows the same benchmark and task-specific setup as FastDSAC, including the FastTD3 replay/update schedule, number of training steps, replay buffer size, batch size, evaluation frequency, number of parallel environments, and task-specific discount factors. At the algorithm level, we largely retain the working configuration of this FastSAC implementation from the codebase, including the target-network soft update rate,

weight decay, and log-standard-deviation bounds of $[-5, 2]$, while aligning a small number of generic optimizer settings with FastDSAC by using AdamW with $\beta = (0.9, 0.95)$ and target entropy $\mathcal{H} = 0$. LayerNorm is enabled on HumanoidBench and disabled on MuJoCo Playground and IsaacLab, matching the benchmark-specific usage in FastDSAC.

For the critic-ablation baseline *FastSAC (C51+DEM)*, we retain the FastSAC training pipeline and replace the continuous critic with the same C51 critic family used in FastTD3, while adding DEM to the actor. All C51-specific hyperparameters follow FastTD3, including the support range and number of atoms. To keep the comparison controlled, we align shared hyperparameters such as weight decay, target entropy, and log-standard-deviation bounds ($[-10, 1]$) with FastDSAC, while leaving the remaining settings consistent with the inherited FastSAC/FastTD3 configurations. The DEM-specific settings follow the main FastDSAC setup, using a fixed temperature $\tau = 1$ together with the same heterogeneity range $\beta_e \sim U[\beta_{\min}, \beta_{\max}]$. LayerNorm usage again follows the same benchmark-specific rule as above.

Overall, these baselines rigorously adhere to the stable configurations from their official codebases. We introduce only minimal, strictly controlled alignments (e.g., matching the target entropy) to ensure a fair comparison, explicitly avoiding any extensive per-baseline retuning that might introduce bias.

Transferring the 15-minute FastSAC recipe. To compare against a stronger FastSAC recipe on our benchmarks, we transfer the portable algorithmic settings of Seo et al. [28] into each benchmark, denoted *Strong FastSAC*: a C51 critic with mean-Q aggregation, LayerNorm and observation normalization, automatic entropy tuning ($\mathcal{H} = 0$, $\alpha_{\text{init}} = 0.001$), log-standard-deviation bounds $[-5, 0]$, eight updates per step, policy frequency 4, Polyak rate 0.125, and its AdamW settings. Each benchmark retains its original environment, reward, action interface, and parallelism; only the task-dependent γ and the empirically strongest C51 support are pre-specified. Table 3 reports the results over three seeds.

Table 3: **Comparison with the transferred 15-minute FastSAC recipe.** Each cell reports final / best-checkpoint mean return over three seeds.

Task	FastSAC (Standard)	Strong FastSAC	FastTD3	FastDSAC
Balance Hard	159.5 / 171.4	130.5 / 151.3	244.0 / 304.6	**726.0 / 855.8**
Basketball	324.0 / 428.4	210.6 / 234.1	529.0 / 577.8	**864.7 / 898.0**
Hurdle	538.3 / 564.4	77.9 / 84.2	**886.8 / 889.6**	860.3 / 860.3
Run	718.4 / 894.9	874.1 / 874.1	902.0 / 902.0	**920.8 / 924.2**
Isaac Velocity-Rough-G1	27.8 / 28.4	18.0 / 21.0	37.8 / 38.4	**40.9 / 41.1**
Playground G1JoystickRough	29.5 / 29.5	-3.5 / -2.8	27.7 / 28.1	**33.0 / 33.3**

Under this transfer, FastSAC (Standard) attains higher best-checkpoint returns than Strong FastSAC on all six tasks and higher final returns on five, suggesting that the strength of the 15-minute recipe is closely tied to its native HoloSoma training pipeline. Neither FastSAC variant consistently reaches FastTD3, whereas FastDSAC exceeds it on five tasks and remains close on *Hurdle*. Comparisons within the native HoloSoma pipeline are reported in Appendix H.1.

A.4 Detailed Implementation Recipe

To ensure stability and peak performance in the high-throughput continuous control regime, we adopt specific algorithmic and engineering choices grounded in recent literature and our empirical experiments.

1. Layer Normalization (LayerNorm). We apply LayerNorm in the actor and critic networks exclusively for the ultra-high-dimensional HumanoidBench domain ($|\mathcal{A}| = 61$). Consistent with recent findings [29, 30, 28], this regularization is critical for maintaining stable gradient flow in expansive action spaces. Conversely, for the MuJoCo Playground and IsaacLab tasks featuring smaller action spaces, FastDSAC achieves peak performance without LayerNorm. This contrast demonstrates that our performance gains stem primarily from core algorithmic mechanisms (DEM and continuous critic) rather than architectural dependencies. A comprehensive empirical analysis justifying these domain-specific choices is provided in Appendix M.

2. Target Entropy. Following recent work [28], we set the target entropy $\mathcal{H} = 0$ across all tasks. In our *Basketball* ablation, the standard heuristic ($\mathcal{H} = -|\mathcal{A}|$) is associated with earlier variance decay

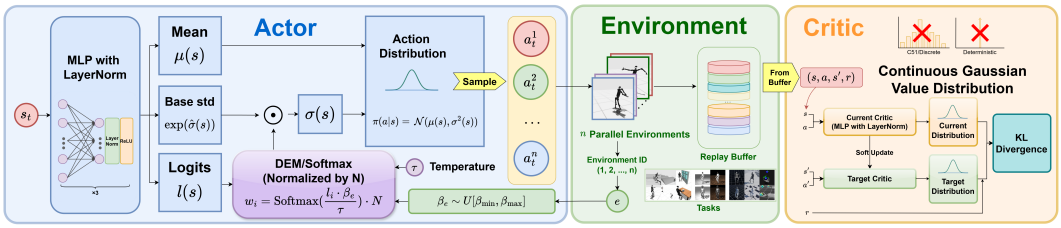


Figure 7: **Overview of the FastDSAC architecture. (Left) Actor with DEM:** Softmax-normalized weights w_i modulate learned base standard deviations $\exp(\hat{\sigma}_{\phi,i}(s))$ elementwise. The weights depend on logits $l(s)$ and an environment-conditioned factor β_e; the final standard deviation depends on both branches. **(Middle) Environment:** Massively parallel environments collect experiences into a shared Replay Buffer for high-throughput training. **(Right) Continuous Critic:** The critic models the return as a **continuous Gaussian distribution**. It minimizes the **KL Divergence** between the current and target distributions, avoiding the quantization artifacts of discrete baselines (marked by $\times$).

and less reliable convergence (Appendix N). The higher target entropy supports continued exploration, while DEM learns how to modulate its scale across action dimensions under the unchanged SAC objective.

3. Adaptive Temperature Optimization (auto-τ). The unified default DEM temperature ($\tau = 1.0$) yields strong and robust performance across the vast majority of evaluated tasks and remains our primary recommendation for general use. However, our temperature sweep suggests that some highly dynamic tasks (e.g., *Hurdle*) can benefit from less concentrated modulation weights. To examine this potential without manual intervention, we introduce an adaptive variant, FastDSAC (auto-τ), in which τ is treated as a network parameter and optimized end-to-end. We evaluate this variant only on *Balance Hard*, *Run*, and *Hurdle*. It is included in the main results (Figure 2) as a diagnostic "upper bound" of DEM's autonomous exploration modulation. While the fixed-τ baseline remains the primary recommendation for general use because of its simplicity, this adaptive extension highlights the potential for future work to discover task-specific coordination patterns without relying on manual kinematic priors.

4. High-Precision Critic Stability. Unlike the original DSAC-T [26], which requires a large numerical stabilizer (bias $= 0.1$) to prevent division-by-zero during continuous distribution modeling, the inherently stable statistics derived from our large-batch, high-throughput regime allow us to reduce this bias significantly to 10^{-6}. This crucial reduction enables the FastDSAC critic to model states with low-variance return distributions with high fidelity, accurately capturing fine-grained reward structures that would otherwise be smoothed out by conservative numerical stabilization.

5. Implementation Notes. In implementation, the temperature-scaled DEM logits are clipped to a bounded range before the Softmax for numerical stability, and the base log-standard deviation is mapped to fixed bounds via a tanh parameterization. For heterogeneous exploration, the environment-specific scaling factor is kept fixed within each episode and resampled only when the episode terminates, consistent with the mixed scaling exploration implementation inherited from FastTD3.

B Extended Details on Sim-to-Real Validation

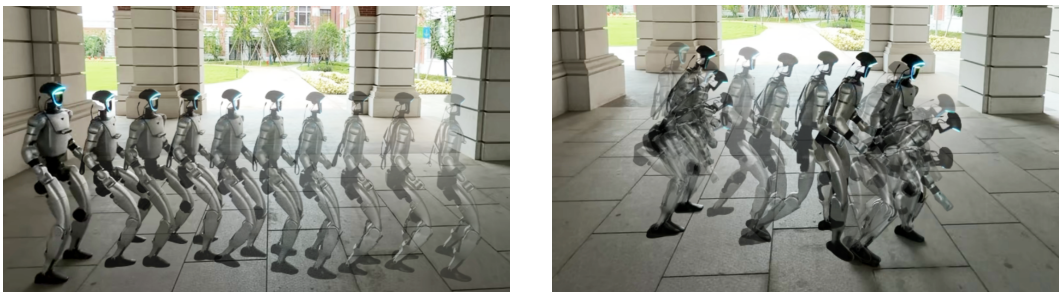
To further examine whether the simulated performance of FastDSAC carries over to physical hardware, we deploy the learned policies on a Unitree G1 humanoid using the mature sim-to-real pipeline [28]. This appendix summarizes the training pipeline, task setup, and deployment results behind the main-text results in Section 4.4.

Training Pipeline and Domain Randomization. Policies are trained entirely in simulation using massive parallelization. To reduce the reality gap, the training pipeline applies domain randomization to key physical factors, including mass, friction, motor damping, latency, and external push perturbations. The final real-world performance should therefore be interpreted as the combined outcome of the FastDSAC method and the maturity of this underlying domain randomization pipeline.

Task Formulations and Minimal Adaptation. We evaluate two policy classes: (1) a joystick-conditioned locomotion policy (shown in the main text, Figure 6), and (2) a complex motion-tracking

policy shown here. Crucially, the algorithmic adaptation is minimal. The core hyperparameters of FastDSAC (e.g., learning rates, DEM target entropy $\mathcal{H} = 0$) remain identical to those used in the simulated IsaacLab (and MuJoCo Playground) experiments. We modify only environment-specific rollout settings to match the established deployment recipes, using $\gamma = 0.99$ for motion tracking and $\gamma = 0.97$ for locomotion, together with the corresponding multi-step return horizons and update-to-data ratios.

Deployment Results. The learned policies run directly on the G1's onboard computer. For the locomotion task, the robot tracks real-time joystick commands; for motion tracking, it follows the reference "squat-carry-walk-squat" sequence. Figure 8 presents time-lapse sequences of backward walking and the motion-tracking policy. Both policies show zero-shot stability in sim-to-real transfer and remain stable under moderate physical-push perturbations. These observations are consistent with the main-text result that policies trained with FastDSAC in simulation can transfer to stable real-world control. Videos are available in our open-source repository (Section 1).



(a) Joystick-conditioned backward walking

(b) "Squat-carry-walk-squat" motion tracking

Figure 8: **Additional Sim-to-Real Results.** Time-lapse sequences of zero-shot FastDSAC policies deployed on the Unitree G1. The robot executes backward walking under real-time joystick commands and a "squat-carry-walk-squat" motion-tracking sequence.

C Supplementary Background on FastTD3 and DSAC-T

C.1 FastTD3 Recipe Inherited by FastDSAC

FastTD3 is presented as a simplified high-throughput TD3 recipe built from parallel simulation, large-batch updates, a distributional critic, and tuned hyperparameters [5]. The original paper emphasizes that massively parallel environments primarily improve wall-clock efficiency and also increase data diversity, which helps deterministic policy gradients compensate for weak single-trajectory exploration. It further reports that an unusually large batch size of $32{,}768$ stabilizes critic learning under massively parallel rollouts, and that the resulting diversity can reduce the effective off-policyness of updates enough to keep training stable even without residual connections or LayerNorm.

For value estimation, FastTD3 adopts a discrete distributional critic from the C51 family:

$$Z_\theta^{\mathrm{C51}}(s,a) = \sum_{i=1}^{N} p_{\theta,i}(s,a)\,\delta_{z_i}, \tag{10}$$

where $\{z_i\}_{i=1}^N$ is a fixed support spanning $[v_{\min}, v_{\max}]$, $p_{\theta,i}(s,a)$ is the predicted probability of atom z_i, and δ_{z_i} denotes a Dirac mass. Given the target action from the TD3 target actor, the distributional Bellman target is

$$\mathcal{T}Z_{\theta'}^{\mathrm{C51}}(s,a) = r(s,a) + \gamma Z_{\theta'}^{\mathrm{C51}}(s',a'), \tag{11}$$

which is projected back onto the fixed support to obtain $\boldsymbol{m} = \Phi\mathcal{T}Z_{\theta'}^{\mathrm{C51}}$. The critic is then trained by the Cross-Entropy objective

$$\mathcal{L}_{\mathrm{C51}}(\theta) = -\sum_{i=1}^{N} m_i \log p_{\theta,i}(s,a). \tag{12}$$

This discrete critic is helpful in the original FastTD3 study, but it introduces support-related hyperparameters $(v_{\min}, v_{\max})$ and keeps value learning tied to a fixed atomization of returns. FastDSAC retains the same high-throughput protocol and replay/update structure, but replaces this critic family with a continuous Gaussian critic to avoid support tuning and quantization artifacts.

Beyond the critic parameterization, the FastTD3 recipe includes two implementation patterns that are inherited conceptually in FastDSAC. First, FastTD3 studies a mixed exploration-noise schedule across environments,

$$a_t^{(e)} = \mu_\phi(s_t^{(e)}) + \epsilon_t^{(e)}, \qquad \epsilon_t^{(e)} \sim \mathcal{N}(0, \sigma_e^2 I), \qquad \sigma_e \sim U[\sigma_{\min}, \sigma_{\max}], \tag{13}$$

and reports that although the mixed-noise scheme itself does not produce a major gain, it remains a cheap way to expose parallel environments to diverse exploration scales. FastDSAC inherits this spirit through heterogeneous DEM scaling factors β_e, while replacing global noise-scale variation with learned, dimension-wise modulation. Second, FastTD3 keeps the optimization recipe deliberately simple: standard MLP backbones, low update-to-data ratios, and benchmark-specific parallel rollout protocols summarized in Appendix A. FastDSAC preserves this training backbone so that the main algorithmic difference remains the actor-side exploration structure and the critic parameterization.

C.2 DSAC-T Components Used in FastDSAC

For completeness, we summarize only the DSAC-T ingredients directly relevant to FastDSAC's critic design. DSAC and DSAC-T model the return distribution as a continuous Gaussian, so the critic directly predicts both the mean Q-value and its uncertainty scale [24, 26]:

$$Z_\theta(\cdot|s,a) = \mathcal{N}(Q_\theta(s,a), \sigma_\theta^2(s,a)). \tag{14}$$

Given $a' \sim \pi_\phi(\cdot|s')$ and a target sample $z' \sim Z_{\theta'}(s', a')$, the standard distributional target is

$$y_z = r + \gamma\big(z' - \alpha \log \pi_\phi(a'|s')\big). \tag{15}$$

DSAC begins from a sample-based negative log-likelihood objective for this target return under the current Gaussian critic,

$$J_Z(\theta) = -\mathbb{E}\Big[\log \mathcal{P}\big(y_z \mid Z_\theta(\cdot|s,a)\big)\Big]. \tag{16}$$

Under the Gaussian parameterization above, this objective is equivalent, up to constants independent of θ, to

$$J_Z(\theta) = \mathbb{E}\left[\frac{\big(y_z - Q_\theta(s,a)\big)^2}{2\sigma_\theta^2(s,a)} + \log \sigma_\theta(s,a)\right]. \tag{17}$$

DSAC decomposes the resulting critic gradient into mean-related and variance-related parts, and stabilizes the latter by clipping the random target return:

$$C(y_z; b) = \operatorname{clip}\left(y_z, Q_\theta(s,a) - b, Q_\theta(s,a) + b\right). \tag{18}$$

This yields the clipped gradient form

$$\begin{aligned} \nabla_\theta J_Z \approx \mathbb{E}\Big[&- \frac{y_z - Q_\theta(s,a)}{\sigma_\theta(s,a)^2} \nabla_\theta Q_\theta(s,a) \\ &- \frac{\big(C(y_z;b) - Q_\theta(s,a)\big)^2 - \sigma_\theta(s,a)^2}{\sigma_\theta(s,a)^3} \nabla_\theta \sigma_\theta(s,a)\Big]. \end{aligned} \tag{19}$$

Expected Value Substituting. DSAC-T observes that the first term in Eq. (19) still inherits the randomness of y_z. It therefore introduces the target Q-value

$$y_q = r + \gamma\big(Q_{\theta'}(s', a') - \alpha \log \pi_\phi(a'|s')\big), \tag{20}$$

which satisfies $\mathbb{E}[y_z \mid a'] = y_q$. Replacing the first occurrence of y_z by y_q gives

$$\begin{aligned} \nabla_\theta J_Z \approx \mathbb{E}\Big[&- \frac{y_q - Q_\theta(s,a)}{\sigma_\theta(s,a)^2} \nabla_\theta Q_\theta(s,a) \\ &- \frac{\big(C(y_z;b) - Q_\theta(s,a)\big)^2 - \sigma_\theta(s,a)^2}{\sigma_\theta(s,a)^3} \nabla_\theta \sigma_\theta(s,a)\Big]. \end{aligned} \tag{21}$$

This refinement makes the mean update closer to the non-distributional SAC target and reduces gradient randomness in the critic update.

Twin Value Distribution Learning. The second refinement is a distributional analogue of clipped double Q-learning. DSAC-T trains two Gaussian critics Z_{θ_1} and Z_{θ_2} independently and selects the target critic with the smaller mean:

$$j^* := \arg\min_{j\in\{1,2\}} Q_{\theta'_j}(s', a'). \tag{22}$$

The target return and target Q-value are then defined as

$$\begin{aligned} y_z^{\min} &= r + \gamma\big(z'_{j^*} - \alpha \log \pi_\phi(a'|s')\big), \qquad z'_{j^*} \sim Z_{\theta'_{j^*}}(s', a'), \\ y_q^{\min} &= r + \gamma\big(Q_{\theta'_{j^*}}(s', a') - \alpha \log \pi_\phi(a'|s')\big). \end{aligned} \tag{23}$$

Substituting Eq. (23) into Eq. (21) yields the twin-distribution critic update, and the actor objective becomes

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{B}, a\sim\pi_\phi}\big[\min_{j\in\{1,2\}} Q_{\theta_j}(s,a) - \alpha \log \pi_\phi(a|s)\big]. \tag{24}$$

This refinement reduces overestimation bias by building both critic and actor targets from the more conservative value distribution.

Variance-Based Critic Gradient Adjustment. The third refinement removes the fixed clipping boundary and the reward-scale sensitivity of the mean update. DSAC-T sets the clipping boundary adaptively as

$$b = \xi \mathbb{E}_{(s,a)\sim\mathcal{B}}\big[\sigma_\theta(s,a)\big], \tag{25}$$

where ξ is a constant controlling the clipping range (typically set $\xi = 3$, following the three-sigma rule), and introduces the variance-based scaling weight

$$\omega = \mathbb{E}_{(s,a)\sim\mathcal{B}}\big[\sigma_\theta(s,a)^2\big]. \tag{26}$$

Using moving averages b_j and ω_j for the two critics, the DSAC-T gradient becomes

$$\begin{aligned} \nabla_{\theta_j} J_Z^{\text{scale}} \approx (\omega_j + \epsilon_\omega)\mathbb{E}\Big[&- \frac{y_q^{\min} - Q_{\theta_j}(s,a)}{\sigma_{\theta_j}(s,a)^2 + \epsilon} \nabla_{\theta_j} Q_{\theta_j}(s,a) \\ &- \frac{\big(C(y_z^{\min}; b_j) - Q_{\theta_j}(s,a)\big)^2 - \sigma_{\theta_j}(s,a)^2}{\sigma_{\theta_j}(s,a)^3 + \epsilon} \nabla_{\theta_j} \sigma_{\theta_j}(s,a)\Big]. \end{aligned} \tag{27}$$

The corresponding moving-average updates are

$$\begin{aligned} b_j &\leftarrow \tau\, \xi\, \mathbb{E}_{(s,a)\sim\mathcal{B}}\big[\sigma_{\theta_j}(s,a)\big] + (1-\tau) b_j, \\ \omega_j &\leftarrow \tau\, \mathbb{E}_{(s,a)\sim\mathcal{B}}\big[\sigma_{\theta_j}(s,a)^2\big] + (1-\tau)\omega_j. \end{aligned} \tag{28}$$

FastDSAC keeps the first two refinements because they are directly compatible with our high-throughput setting. For the third, we only retain the gradient scaling idea. In the final implementation, Section 3.2 removes the dynamic clipping boundary, uses an unclipped sampled-target variance update, and reduces the numerical stabilizer to better resolve low-variance states in the large-batch regime.

D Architecture and Pseudocode of FastDSAC

We provide the complete pseudocode for FastDSAC in Algorithm 1. For FastDSAC architecture, please refer to Figure 7.

Algorithm 1 FastDSAC

Require: n parallel environments, replay buffer $\mathcal{B}$, updates per step U, discount γ, Polyak coefficient τ_{polyak}, DEM temperature τ, heterogeneity range $[\beta_{\min}, \beta_{\max}]$, target entropy $\mathcal{H}$.
Ensure: Learned policy parameters ϕ.

Initialize actor π_ϕ with output heads $(\mu_\phi, \hat{\sigma}_\phi, l_\phi)$, twin critics $Z_{\theta_j}(s,a) \sim \mathcal{N}(Q_{\theta_j}(s,a), \sigma^2_{\theta_j}(s,a))$, target parameters $\theta'_j \leftarrow \theta_j$, entropy coefficient α, and $\mathcal{B} \leftarrow \emptyset$.
for each training iteration **do**
  Parallel Rollout:
  for environment $e \in [n]$ **in parallel do**
    Sample heterogeneity factor $\beta_e \sim \mathcal{U}[\beta_{\min}, \beta_{\max}]$, observe $s_t^{(e)}$.
    Forward pass: $(\mu, \hat{\sigma}, l) \leftarrow \pi_\phi(s_t^{(e)})$.
    Compute dimension-wise weights: $w_e \leftarrow N \cdot \mathrm{Softmax}(\beta_e l/\tau)$ per Eq. (4).
    Modulate exploration: $\sigma_e \leftarrow w_e \odot \exp(\hat{\sigma})$.
    Sample noise $\xi \sim \mathcal{N}(0, I)$, execute action $a_t^{(e)} \leftarrow \mu + \sigma_e \odot \xi$.
    Collect transition $(s_t^{(e)}, a_t^{(e)}, r_t^{(e)}, s_{t+1}^{(e)}, d_t^{(e)})$, store in $\mathcal{B}$.
  end for
  Distributional Soft Policy Evaluation & Improvement:
  for gradient step $u \in [U]$ **do**
    Sample batch $(s, a, r, s', d) \sim \mathcal{B}$, compute $\bar{\gamma} \leftarrow \gamma(1-d)$.
    Evaluation: Sample next action $a' \sim \pi_\phi(\cdot|s')$, compute target $y_q^{\min} \leftarrow r + \bar{\gamma}(\min_{j\in\{1,2\}} Q_{\theta'_j}(s', a') - \alpha \log \pi_\phi(a'|s'))$.
    for $j \in \{1, 2\}$ **do**
      Sample $z'_j \sim Z_{\theta'_j}(s', a')$, compute $y_{z,j} \leftarrow r + \bar{\gamma}\big(z'_j - \alpha \log \pi_\phi(a'|s')\big)$.
      Compute critic scaling: $\omega_j \leftarrow \mathbb{E}_{\mathcal{B}}[\sigma^2_{\theta_j}(s,a)]$.
      Update critic: $\theta_j \leftarrow \theta_j - \eta_Q \nabla_{\theta_j} J_Z$ via Eq. (7) with $(y_q^{\min}, y_{z,j}, \omega_j)$.
    end for
    Improvement: Sample action $a \sim \pi_\phi(\cdot|s)$ from batch.
    Update actor: $\phi \leftarrow \phi - \eta_\pi \nabla_\phi \mathcal{L}_{\text{actor}}$ where $\mathcal{L}_{\text{actor}} = \mathbb{E}[\alpha \log \pi_\phi(a|s) - Q_\theta(s,a)]$.
    Optimize entropy coefficient: $\alpha \leftarrow \alpha - \eta_\alpha \mathbb{E}[-\log \pi_\phi(a|s) - \mathcal{H}]$.
    Target update: $\theta'_j \leftarrow (1 - \tau_{polyak})\theta'_j + \tau_{polyak}\theta_j$ for $j \in \{1, 2\}$.
  end for
end for

E Related Work

Scalable Reinforcement Learning for High-Dimensional Control. Recent advances in physics simulation have shifted RL for high-dimensional humanoid control toward massively parallel training regimes [6, 11, 12]. In this high-throughput setting, deterministic off-policy methods such as Parallel Q-Learning (PQL) [8] and FastTD3 [5] have been especially successful, largely because their training dynamics remain stable under aggressive replay-based updates in large action spaces. More recently, stochastic off-policy RL has also begun to benefit from this regime. For example, FlashSAC [13] demonstrates that SAC can be scaled effectively through recipe-level stabilization and improved system throughput. However, its evidence on HumanoidBench is primarily concentrated on lower-dimensional H1 locomotion tasks, whereas our focus is the considerably more challenging 61-dimensional H1Hand whole-body control setting, evaluated across a comprehensive suite of 25 diverse tasks.

Model-based approaches such as DreamerV3 [2], TD-MPC2 [1], and TDMPBC [4] can achieve strong sample efficiency, but they typically introduce substantial computational overhead from world-model learning and inference, often leading to longer wall-clock training in large-scale continuous-control settings. By contrast, model-free off-policy methods remain attractive in massively parallel environments because they better exploit simulator throughput.

From a methodological perspective, Maximum Entropy RL methods such as SAC offer appealing exploration and robustness properties [14], but their effectiveness deteriorates sharply as action dimensionality grows. In practice, applying SAC to domains with $|\mathcal{A}| > 60$ has remained challenging

due to the curse of dimensionality, often requiring substantial architectural modification and still frequently underperforming strong deterministic baselines on the most difficult tasks [25, 5, 13]. FastDSAC is designed specifically for this regime. Rather than relying solely on scaling heuristics or abandoning stochasticity in favor of deterministic actors, FastDSAC directly targets the two core bottlenecks that hinder stochastic off-policy RL in very high-dimensional control: ineffective exploration and unstable value estimation. FastDSAC fills this gap by making stochastic maximum entropy RL practical in the high-throughput, 61-DoF control regime, where prior success has been dominated by deterministic methods.

Maximum Entropy RL and Value Estimation in High Dimensions. Scaling Soft Actor-Critic (SAC) to high-dimensional action spaces introduces two fundamental challenges: inefficient exploration and value overestimation.

First, standard diagonal-Gaussian policies can suffer from *vanishing exploration* as action dimensionality grows [18]. Although they learn state-dependent, dimension-specific variances, they lack a dedicated cross-dimensional structure for adapting exploration to differing joint-level control demands. Recent works address high-dimensional exploration from different perspectives. QFLEX [18] uses flow-based action modeling. ERA [19] imposes a minimum sampling-entropy threshold through output-layer activations and reports gains across several continuous-control algorithms, including FastSAC-ERA on HumanoidBench. FastDSAC is related to ERA in spirit, since both methods reshape how stochasticity is expressed by the actor, but the underlying mechanisms differ. ERA imposes an explicit minimum-entropy constraint through output-layer activations, whereas our Dimension-wise Entropy Modulation (DEM) instead autonomously learns softmax-normalized relative modulation of per-dimension standard deviations under the original SAC objective, without a prescribed joint allocation or a new entropy threshold.

Second, accurate Q-function estimation becomes increasingly difficult in high-dimensional spaces due to severe extrapolation errors on out-of-distribution (OOD) actions [20, 21]. This often results in *Q-bias explosion* [20] and pronounced overestimation, as the policy progressively over-exploits artifacts in the critic approximation [22]. To mitigate this issue, prior methods such as TQC [23] and FastTD3 [5] adopt distributional RL with truncated quantiles or categorical atoms (C51), respectively. While effective, these discrete approximations depend on manually specified value supports and a finite number of atoms, which can be brittle under varying reward scales and may introduce quantization error that limits value precision. Reward/return normalization, as in BRC [31] and XQC [32], offers an alternative to per-task support selection.

In contrast, FastDSAC employs a Continuous Gaussian Critic [26]. Rather than truncating the upper tail as in TQC, our critic models the full return distribution in a continuous form, thereby avoiding fixed support specification and atomization. Furthermore, by disentangling the gradient estimators, the continuous critic naturally suppresses mean updates on highly uncertain actions through inverse-variance scaling, providing a principled mechanism for reducing overestimation in high-dimensional settings.

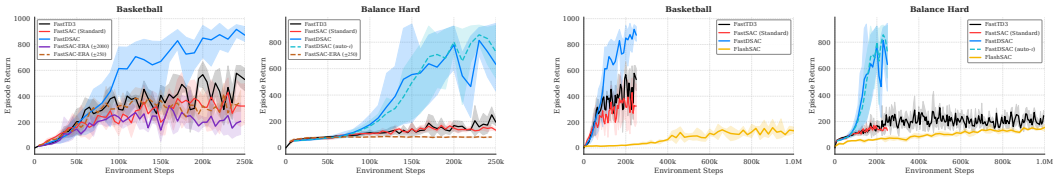
F Comparison with Contemporary Works

To strictly assess the scalability of FastDSAC, we compare it against two recent concurrent methods, FastSAC-ERA [19] and FlashSAC [13]. We evaluate these baselines on the challenging 61-DoF *Basketball* and *Balance Hard* tasks. The learning curves are presented in Figure 9.

Comparison with FastSAC-ERA. ERA maintains exploration through a global minimum-entropy threshold. In our evaluated 61-DoF tasks, FastSAC-ERA plateaus below FastDSAC (e.g., at ≈ 100 in *Balance Hard*; Figure 9(a)). A global entropy threshold alone regulates overall exploration but does not specify its allocation across action dimensions. For 61-DoF whole-body control, selective exploration modulation can help preserve precision on some joints while allowing broader exploration on others. DEM provides an explicit cross-dimensional structure for emerging autonomous exploration modulation.

Comparison with FlashSAC. FlashSAC improves wall-clock efficiency through system-level throughput optimizations and recipe-level heuristics, demonstrating strong results on lower-dimensional locomotion ($|\mathcal{A}| \approx 20$). However, when applied to the 61-DoF regime shown in Figure 9(b), it struggles to learn effective behaviors, scoring below 200 even after 10^6 environment

steps. While FlashSAC's system engineering is valuable for simulation performance and real robot deployment, its performance drop in higher dimensions suggests that raw throughput cannot substitute for explicit algorithmic mechanisms (such as DEM) needed to overcome the curse of dimensionality.



(a) Comparison with FastSAC-ERA

(b) Comparison with FlashSAC

Figure 9: Performance comparison against concurrent baselines on 61-DoF HumanoidBench tasks. **(a)** FastSAC-ERA plateaus below FastDSAC on the evaluated tasks. **(b)** FlashSAC struggles to scale to the 61-DoF regime despite dedicated training recipe and system engineering, highlighting the necessity of algorithmic design and mechanisms.

G Computational Resources

We conducted our experiments on a heterogeneous computing cluster consisting of high-performance workstations. The experiments were distributed across two primary hardware configurations:

(1) **Configuration A:** Equipped with an Intel Core i7-14700K CPU (20 physical cores, 28 logical threads) and 128 GB of system RAM. Accelerated computing was provided by a single NVIDIA GeForce RTX 4090.

(2) **Configuration B:** Equipped with an AMD Ryzen 9 9950X CPU (16 physical cores, 32 logical threads) and 192 GB of system RAM. This node features an NVIDIA RTX 6000 (Blackwell Workstation Edition) with 96 GB of VRAM, facilitating large-scale parallel training.

Resource Allocation. To ensure efficient training and accommodate varying memory requirements, the memory-intensive IsaacLab environments, specifically *Isaac-Velocity-Flat-G1-v0*, *Isaac-Velocity-Rough-G1-v0*, *Isaac-Repose-Cube-Allegro-Direct-v0*, and *Isaac-Repose-Cube-Shadow-Direct-v0*, were trained on **Configuration B** to leverage its extensive VRAM capacity. All other tasks, including the HumanoidBench and MuJoCo Playground suites, were trained on **Configuration A**. This hardware setup ensured efficient training on high-dimensional tasks, with each experiment converging in under 5 hours.

H Computational Efficiency and Wall-Clock Time

We compare the wall-clock training time of FastDSAC and FastTD3 across nine representative tasks from HumanoidBench, IsaacLab, and MuJoCo Playground (Figure 10 and Table 4). To ensure a strictly controlled evaluation, both algorithms were benchmarked locally on identical hardware (a single NVIDIA GeForce RTX 4090 GPU and an Intel Core i7-14700K CPU). For each task, we measure steady-state throughput over a 1,300-iteration probe and project the full-run training time from it. This hardware configuration naturally yields different absolute timings compared to the A100 GPUs used in the original FastTD3 publication.

Across all nine tasks, FastTD3 is projected to require 20.21 hours in total, whereas FastDSAC is projected to require 22.28 hours. This corresponds to an overall relative time ratio of $1.10\times$ (geometric mean: $1.09\times$). These results demonstrate that FastDSAC maintains comparable computational efficiency to the highly optimized FastTD3 baseline. Crucially, FastDSAC delivers significantly stronger performance and stability with only a marginal $\approx 10\%$ projected training-time overhead.

H.1 Fixed-Wall-Clock Comparison in HoloSoma

We further compare FastDSAC with HoloSoma's official FastSAC recipe from Seo et al. [28], PPO, and FastTD3 on native G1 locomotion and 29-DoF robot-only whole-body tracking. All methods use the same RTX 4090, 4,096 parallel training environments, and, within each task, the

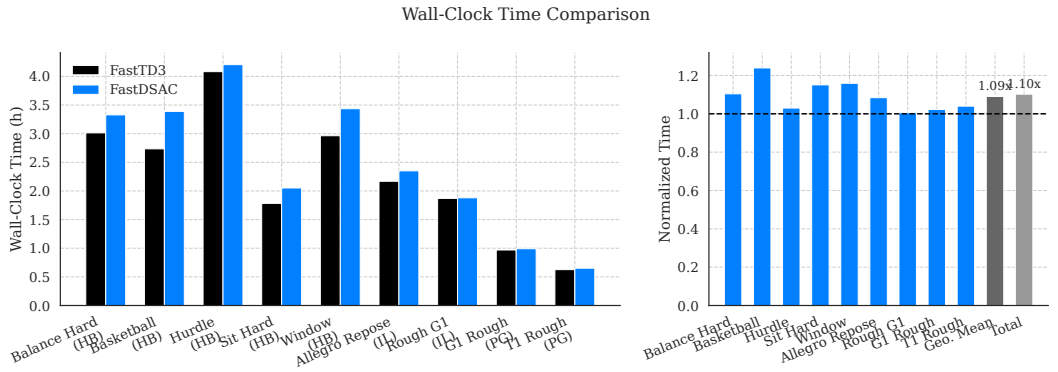


Figure 10: **Wall-clock time comparison between FastDSAC and FastTD3.** Evaluated on representative tasks from HumanoidBench, IsaacLab, and MuJoCo Playground. Left: projected per-task total training time (hours). Right: relative wall-clock time normalized against FastTD3, detailing per-task ratios, the geometric-mean ratio, and the overall total-time ratio across all nine tasks.

Benchmark	Task	Steps	FastTD3 (h)	FastDSAC (h)	Rel. Time
HB	Balance Hard	250K	3.01	3.33	1.10×
HB	Basketball	250K	2.73	3.39	1.24×
HB	Hurdle	300K	4.08	4.21	1.03×
HB	Sit Hard	150K	1.78	2.05	1.15×
HB	Window	250K	2.96	3.44	1.16×
IL	Allegro Repose	200K	2.17	2.35	1.08×
IL	Rough G1	100K	1.87	1.88	1.01×
PG	G1 Rough	45K	0.97	0.99	1.02×
PG	T1 Rough	45K	0.63	0.65	1.03×
HB	Geo. mean	–	–	–	1.13×
IL	Geo. mean	–	–	–	1.04×
PG	Geo. mean	–	–	–	1.03×
All	Geo. mean	–	–	–	1.09×
All	Total	–	20.21	22.28	1.10×

Table 4: **Wall-clock time comparison on representative HumanoidBench, IsaacLab, and MuJoCo Playground tasks.** All timings were measured locally on a single RTX 4090 GPU.

same environment, reward, curriculum, motion data, and action interface. We use the official task-specific FastSAC recipe and tune FastDSAC within the same task setup. Results report deterministic simulation evaluation as mean ± sample standard deviation over three training seeds.

Evaluation uses 256 environments. For locomotion, we fix the full penalty scale to 1.0 and use the same eight-command grid and 20-second episode horizon. For tracking, policies start from the same initial motion frame and follow a 324-step clip; completion is the fraction of environments that finish the entire clip without early termination.

Table 5: **Native HoloSoma locomotion under fixed time budgets.** Entries are evaluation returns (mean ± sample standard deviation over three training seeds). † denotes our best-effort HoloSoma implementation of FastTD3 with source-aligned core training settings.

Method	5 min	10 min	15 min	30 min
FastDSAC	8.50 ± 8.47	49.37 ± 5.53	75.19 ± 9.70	129.76 ± 1.22
Official FastSAC	-38.44 ± 17.00	-9.35 ± 11.76	50.91 ± 2.41	129.24 ± 3.78
PPO	6.11 ± 4.57	4.41 ± 15.87	3.37 ± 7.47	80.71 ± 2.14
FastTD3†	-53.09 ± 24.54	-59.92 ± 17.27	-71.09 ± 32.95	-121.08 ± 34.39

Table 6: **Native HoloSoma whole-body tracking under fixed time budgets.** $R@t$ is evaluation return after t minutes. Completion and termination events per 1,000 evaluated transitions are evaluated at 15 minutes. Entries are mean $\pm$ sample standard deviation over three training seeds.

Method	$R@5 \uparrow$	$R@10 \uparrow$	$R@15 \uparrow$	Completion (%) $\uparrow$	Term./1k $\downarrow$
FastDSAC	23.37 ± 0.95	25.60 ± 0.72	27.10 ± 0.05	93.23 ± 5.47	0.25 ± 0.22
Official FastSAC	15.09 ± 2.26	23.16 ± 1.26	24.96 ± 0.81	91.15 ± 6.56	0.28 ± 0.22
PPO	15.27 ± 4.44	19.44 ± 0.40	20.54 ± 0.88	0.10 ± 0.20	18.85 ± 0.79
FastTD3†	12.94 ± 0.49	12.43 ± 0.47	12.37 ± 0.09	0.00 ± 0.00	14.40 ± 0.58

FastDSAC achieves higher locomotion return than official FastSAC at 5, 10, and 15 minutes, with comparable returns at 30 minutes (Table 5). On whole-body tracking, FastDSAC achieves higher return at every evaluated budget (Table 6). At 15 minutes, it improves return by 8.60% and reduces joint-position error by 10.57% (0.961 ± 0.033 versus 1.075 ± 0.004), with higher return in all three training-seed pairs. It also processes 8.85% more transitions within the same budget (35.59 ± 3.75M versus 32.70 ± 3.36M). PPO and FastTD3 process 41.12 ± 0.37M and 41.45 ± 1.12M transitions, respectively, but achieve much lower return and completion. These results demonstrate FastDSAC's stronger fixed-time performance against the official FastSAC recipe within its native training framework.

H.2 Fixed-Wall-Clock Comparison in UniLab

We additionally compare PPO, APPO, FastTD3, FastSAC, and FastDSAC on the UniLab *G1WalkFlat* MuJoCo task with 2,048 parallel environments on the same RTX 4090 workstation, using five training seeds per method. FastSAC uses UniLab's official task-tuned G1 recipe aligned with HoloSoma, including the C51 critic, LayerNorm, eight updates per step, policy frequency 4, $\gamma = 0.97$, automatic entropy tuning, symmetry augmentation, AMP, and compilation. FastDSAC is tuned for UniLab but uses neither symmetry nor mirrored-transition augmentation.

Table 7: **UniLab *G1WalkFlat* under fixed time budgets.** Entries are returns (mean $\pm$ sample standard deviation over five training seeds).

Method	2.5 min	5 min	7.5 min	10 min
PPO	9.84 ± 0.68	30.00 ± 4.37	80.62 ± 8.19	109.77 ± 6.41
APPO	12.52 ± 0.74	217.12 ± 12.77	267.94 ± 7.68	266.50 ± 4.67
FastTD3	30.23 ± 6.52	32.31 ± 7.57	30.02 ± 3.79	33.30 ± 8.16
FastSAC	145.93 ± 50.46	291.93 ± 2.46	296.18 ± 1.03	298.38 ± 0.76
FastDSAC	$\mathbf{181.78 \pm 41.86}$	$\mathbf{294.88 \pm 1.03}$	$\mathbf{299.18 \pm 0.75}$	$\mathbf{300.20 \pm 0.45}$

FastDSAC achieves the highest mean return at every cutoff (Table 7). At 5, 7.5, and 10 minutes, it outperforms FastSAC on all five paired training seeds, and the paired bootstrap 95% confidence intervals exclude zero.

I Heatmaps of *Basketball* Task

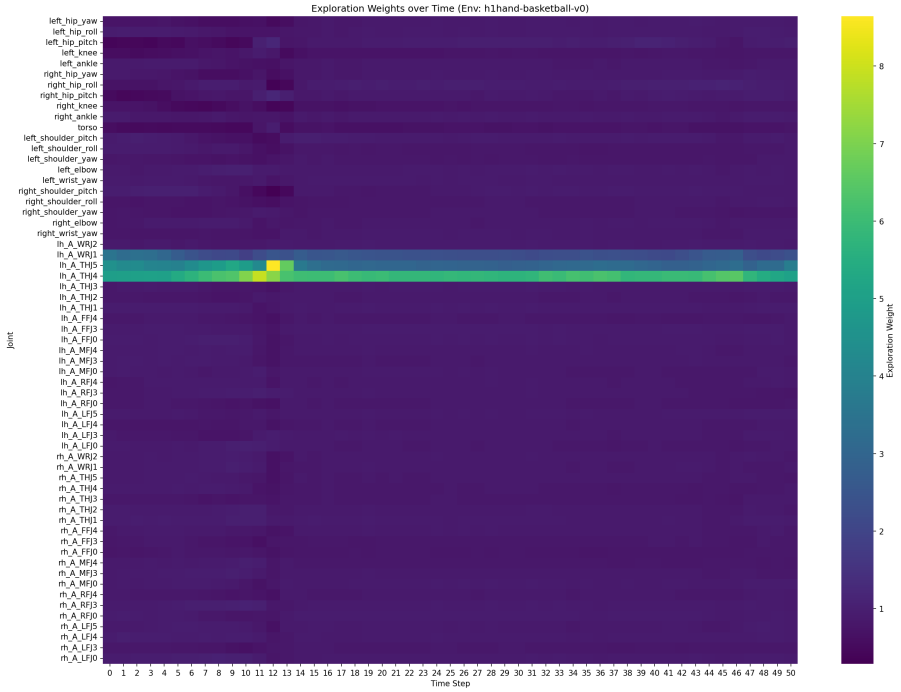


Figure 11: **DEM modulation weights (w_i) during one Basketball episode.** High weights (bright) appear on the left thumb and wrist, while lower weights (dark) appear on the legs, torso, and right arm. These weights attenuate or amplify the effective joint standard deviations: $\sigma_i = w_i \exp(\hat{\sigma}_i)$.

Weights versus effective variance. The heatmap reports modulation weights, whereas realized pre-$\tanh$ exploration depends on $\sigma_i^2 = w_i^2 \exp(2\hat{\sigma}_i)$. We analyzed 25 fixed-$\tau = 1$ checkpoints across four tasks, with one rollout per checkpoint. For a nonnegative vector x, the normalized entropy-based effective dimension is $D(x)/N = \exp(-\sum_i q_i \log q_i)/N$, where $q_i = x_i / \sum_j x_j$; one indicates a uniform distribution across dimensions. We compute it from trajectory-mean weights $\bar{w}_i = \langle w_i \rangle_t$ and variance shares $\bar{p}_i = \langle \sigma_i^2 / \sum_j \sigma_j^2 \rangle_t$, then average across checkpoints. Each body-group value below is its share of total effective variance divided by its share of action dimensions, so one denotes a proportional share.

Task (checkpoints)	$D(\bar{w})/N \to D(\bar{p})/N$	Hands / legs / torso
Balance Hard (8)	$0.128 \to 0.929$	$1.183/0.537/0.186$
Basketball (7)	$0.176 \to 0.961$	$1.121/0.670/0.403$
Run (3)	$0.152 \to 0.963$	$1.106/0.639/0.373$
Window (7)	$0.284 \to 0.939$	$1.156/0.690/0.210$

Table 8: DEM modulation weights versus effective pre-$\tanh$ variance (across-checkpoint means). Group values normalize variance share by group size; they are not inferred from the weight heatmap.

DEM's modulation remains visible in effective exploration despite partial compensation by the learned base scales: the task-mean Spearman correlation between $\log w_i$ and $\log \sigma_i$ across action dimensions is 0.58–0.79 in these four tasks. As Table 8 shows, effective variance is broadly distributed but consistently modulated across body groups. Relative to group size, hands receive more effective variance (1.106–1.183), whereas legs (0.537–0.690) and torso (0.186–0.403) receive less. The same ordering persists in offline post-$\tanh$ stochastic action deviations, indicating that this modulation carries through to sampled actions. Individual high-variance joints vary across seeds (mean top-10

Jaccard overlap 7.4%–16.9%), while the consistent group-level pattern shows that DEM adapts exploration without prescribing a fixed set of joints.

Exploration in FastSAC during training. We tracked pre-$\tanh$ exploration variance in one FastSAC (Standard) run on the 61-dimensional *Basketball* task, sampling 1,024 replay-buffer states every 5k per-environment steps. The variance-participation ratio, computed per state and averaged over sampled states, ranges from 0.900 to 0.931 between 50k and 250k steps and ends at 0.917. Exploration remains broadly distributed across dimensions, with persistent dimension-specific differences. This observation partially motivates examining an explicit cross-dimensional allocation structure; its benefit is evaluated separately through the matched multi-seed DEM ablations in Appendix L.

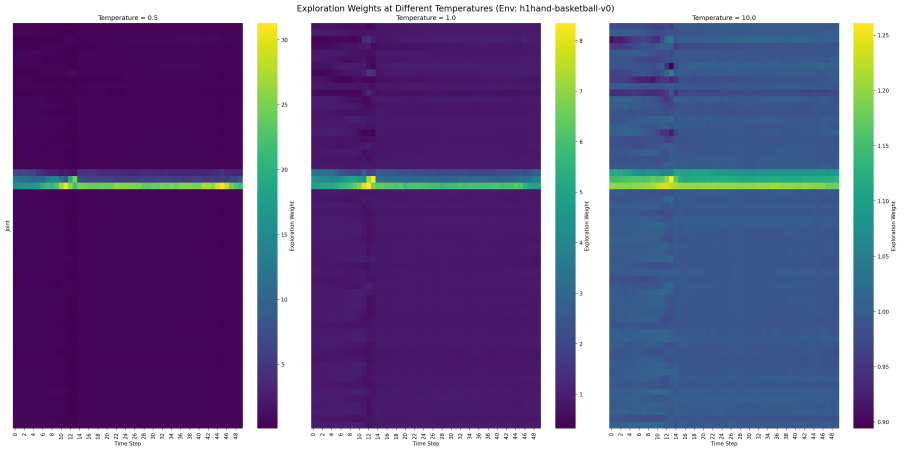


Figure 12: **DEM modulation weights at different temperatures.** Lower temperatures ($\tau = 0.5$) yield more concentrated relative weights, whereas higher temperatures ($\tau = 10.0$) move the weights closer to a uniform distribution; neither heatmap alone specifies the effective joint variance.

J C51 Support Sensitivity on *Basketball*

To directly examine the fixed-support issue for discrete critics, we varied the support range of C51+DEM on *h1hand-basketball-v0* while keeping the actor architecture and the remaining training pipeline unchanged. This sweep uses reward scale 1 without reward or return normalization; its support-sensitivity result is specific to this protocol. As shown in Figure 13, the performance of C51+DEM changes substantially with the support range: a narrower support achieves the best performance, while wider supports lead to noticeably worse results for *Basketball* task. This sensitivity indicates that the support range constitutes an additional tuning axis for discrete critics in our setting. In contrast, FastDSAC avoids this support-specification burden by modeling the return distribution continuously.

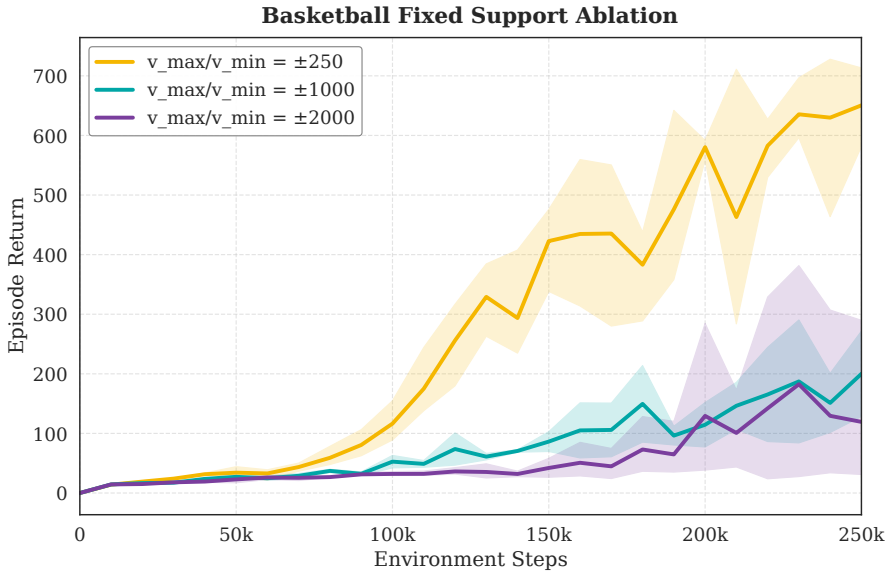


Figure 13: **Support sensitivity of C51+DEM on *Basketball*.** We vary the C51 support range while keeping the remaining training pipeline unchanged. The resulting performance differences show that discrete critics can be sensitive to support specification on this task, whereas FastDSAC avoids this additional tuning axis.

K Post-Training Critic Diagnostics

Evaluation protocol. We compare the final FastDSAC and FastSAC (C51+DEM) checkpoints on *Balance Hard* across two matched training seeds. Both methods retain DEM. For each checkpoint, we collect stochastic trajectories under its own policy with $\gamma = 0.99$ and its final learned entropy coefficient α. Trajectories contain at most 1,000 steps, and critic predictions are recorded every 10 steps until the first episode terminates. We extend the diagnostic time limit because training targets bootstrap time-limit truncations. The two training-seed pairs contain 16 and 64 episodes per method, respectively, paired by common initial-state seeds.

For each sampled state–action pair, we compare the critic prediction with its realized entropy-regularized return-to-go:

$$G_t = r_t + \sum_{k=t+1}^{T-1} \gamma^{k-t} \big[r_k - \alpha \log \pi(a_k \mid s_k) \big], \tag{29}$$

where $T \leq 1{,}000$ denotes episode termination or the rollout cutoff; returns use the remaining observed rewards and entropy terms without a bootstrapped tail. Positive Q-error is $\max(Q(s_t, a_t) - G_t, 0)$, while signed bias and absolute error are $Q - G$ and $|Q - G|$. Critic predictions follow each method's training rule: clipped double-Q for FastDSAC and ensemble-mean Q for C51+DEM. We average

probes within each episode and give the two training seeds equal weight. Confidence intervals use 10,000 hierarchical bootstrap resamples of training-seed pairs and then paired episodes.

Table 9: **Critic diagnostics on matched *Balance Hard* final checkpoints.** Differences are FastDSAC minus C51+DEM; brackets give hierarchical 95% bootstrap confidence intervals. Each method is evaluated under its own stochastic policy.

Metric	FastDSAC	C51+DEM	Difference [95% CI]
Positive Q-error ↓	0.360	17.770	-17.410 $[-24.407, -10.208]$
Signed bias	-53.726	10.971	-64.697 $[-87.120, -39.882]$
Mean absolute error ↓	54.446	24.570	$+29.876$ $[17.149, 40.341]$

Reduced positive overestimation. FastDSAC has lower positive Q-error in both matched training-seed pairs. The mean reduction is 17.41 return units (95% confidence interval: $[10.21, 24.41]$; Table 9). This reduction accompanies more conservative estimates, as reflected in the negative signed bias and higher absolute error. These results support reduced positive overestimation for the evaluated FastDSAC critic design, complementing the learning-curve ablation and support-sensitivity study. Since each checkpoint is evaluated under its own policy and Q-aggregation rule, this comparison assesses the complete critic design rather than Gaussian parameterization alone.

L Multi-Seed Ablation of DEM

Controlled comparison and statistical protocol. To quantify DEM's contribution beyond the continuous critic, we extend the ablation to eight independent training seeds per method on six HumanoidBench tasks and five seeds per method on IsaacLab *Velocity-Rough-G1*. FastDSAC (w/o DEM) retains the continuous critic and training recipe but uses an ordinary state-dependent diagonal-Gaussian actor. Following the robust evaluation methodology of Agarwal et al. [33], we report per-task interquartile mean (IQM) returns and 95% percentile bootstrap confidence intervals from 20,000 resamples of training seeds. Table 10 also reports sample standard deviations to quantify across-seed variability separately from uncertainty in the IQM estimate.

Figure 14 includes FastTD3 and FastSAC (Standard) as reference baselines.

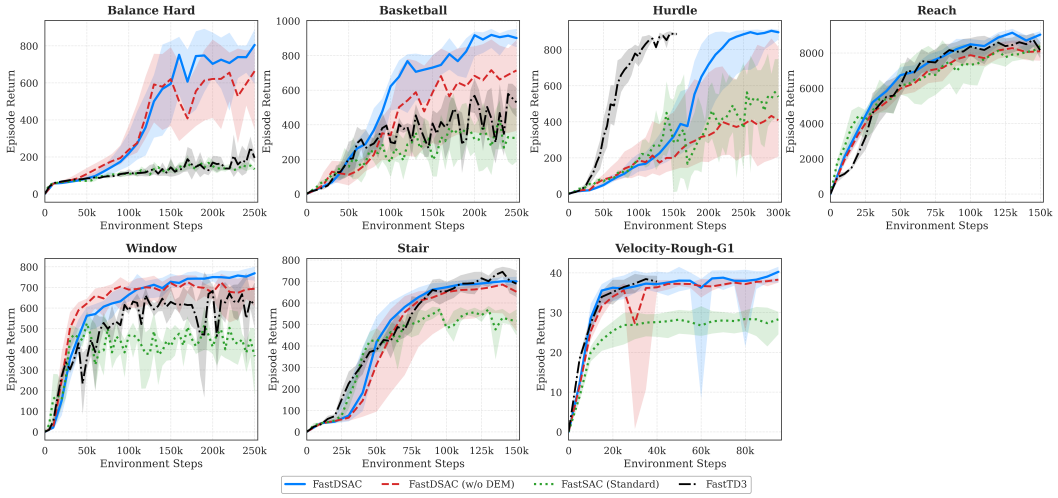


Figure 14: **Multi-seed DEM ablation with baseline comparisons.** Curves show IQM return with 95% bootstrap intervals.

Performance gains beyond the critic. With the continuous critic held fixed, DEM increases IQM return on all seven tasks (Table 10). The improvement is particularly pronounced on *Hurdle*, where IQM rises from 408.6 to 896.4. On *Reach*, it rises from 8095.6 to 9038.9, showing that DEM's

Table 10: **Returns at the end of each common evaluation budget.** IQM brackets give 95% bootstrap confidence intervals. SD is the sample standard deviation across all training seeds. Steps are per environment; the two variants use the same budget within each task.

		FastDSAC (w/o DEM)		FastDSAC	
Task	Steps	IQM [95% CI]	SD	IQM [95% CI]	SD
Balance Hard	250k	663.2 [350.0, 787.0]	268.6	804.8 [594.8, 889.3]	182.7
Basketball	250k	713.8 [364.6, 890.1]	269.3	900.7 [848.5, 954.2]	60.9
Hurdle	300k	408.6 [207.8, 815.5]	327.3	896.4 [856.2, 910.6]	41.2
Reach	150k	8095.6 [7565.2, 8457.6]	584.7	9038.9 [8562.4, 9236.9]	579.5
Window	250k	693.4 [606.6, 752.6]	87.0	768.6 [733.4, 800.2]	41.5
Stair	150k	651.8 [468.7, 684.0]	178.7	700.4 [682.8, 721.9]	22.5
Velocity-Rough-G1	95k	38.3 [37.5, 38.6]	0.5	40.3 [37.6, 41.5]	2.0

benefit extends beyond *Balance Hard* and *Basketball*. On both *Hurdle* and *Reach*, the endpoint 95% confidence intervals do not overlap. The remaining tasks also show higher IQM returns, with the uncertainty reported in the confidence intervals.

Reduced sensitivity to the training seed. Across all six HumanoidBench tasks, DEM yields higher IQM returns and lower across-seed standard deviations. The reduction is especially pronounced on *Hurdle* (327.3 to 41.2) and *Stair* (178.7 to 22.5). The critic-only variant can also be seed-sensitive: on *Basketball*, three of eight FastDSAC (w/o DEM) runs finish at 329–372, while the other five reach 774–923; all eight full FastDSAC runs reach 825–991. On *Hurdle*, five of eight runs without DEM remain below 300, whereas all full-method runs reach 793–916. Thus, the continuous critic alone can produce large gains but does not consistently avoid low-return runs. On *Velocity-Rough-G1*, DEM improves IQM from 38.3 to 40.3, although its final across-seed variability is higher. These results show that DEM improves performance beyond the continuous critic alone and substantially reduces seed sensitivity on challenging HumanoidBench tasks.

M Ablation Study on Layer Normalization

To isolate the effect of Layer Normalization (LayerNorm) from our core algorithmic contributions, we conduct an ablation study on representative tasks from all three benchmarks, evaluating FastDSAC with and without LayerNorm. The results are shown in Figure 15.

Importance in High-Dimensional Settings (HumanoidBench). In the very high-dimensional domain ($|\mathcal{A}| = 61$), LayerNorm serves as a key architectural stabilizer. As shown in the top row of Figure 15, removing LayerNorm substantially degrades training dynamics. Specifically, the agent exhibits high inter-seed variance in *Basketball*, severe optimization oscillations and lower asymptotic returns in *Balance Hard*, and drastically reduced sample efficiency in *Walk*. These results suggest that, LayerNorm helps stabilize intermediate feature scales, leading to more reliable convergence and more sample-efficient learning in expansive continuous action spaces.

Limited Impact in Lower-Dimensional Settings (IsaacLab & MuJoCo Playground). Conversely, in domains with moderately high action dimensionality ($|\mathcal{A}| < 30$), the FastDSAC framework remains stable even without LayerNorm. As shown in the bottom row of Figure 15, while adding LayerNorm yields marginal asymptotic improvements in certain manipulation tasks (e.g., *Allegro Repose*), it has little effect on locomotion tasks overall, slightly degrading performance on *Rough H1* while providing only a marginal improvement on *G1 Rough Terrain* (MuJoCo Playground). More importantly, even without LayerNorm, FastDSAC consistently outperforms the FastTD3 baselines in these domains (as established in Section 4.2). We therefore omit LayerNorm in these environments to highlight that FastDSAC's strong performance primarily stems from Dimension-wise Entropy Modulation (DEM) and the continuous distributional critic, rather than from architectural heuristics.

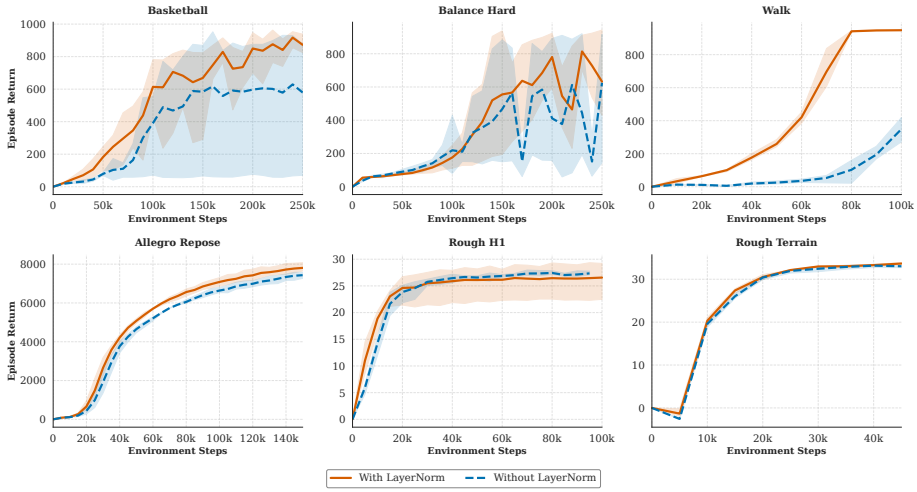


Figure 15: Ablation study of Layer Normalization. **Top row (HumanoidBench, $|\mathcal{A}| = 61$):** Removing LayerNorm induces seed sensitivity, severe oscillations, and poor sample efficiency. **Bottom row (IsaacLab & MuJoCo Playground, $|\mathcal{A}| < 30$):** FastDSAC remains highly stable and sample-efficient even without LayerNorm, indicating that the core algorithm is robust in lower-dimensional settings.

N Ablation Study on Target Entropy

In standard maximum entropy RL, the target entropy is typically set to $\mathcal{H} = -|\mathcal{A}|$ to regulate exploration. To evaluate our choice of $\mathcal{H} = 0$, we conduct an ablation study on the 61-DoF *Basketball* task, comparing it against the standard heuristic ($\mathcal{H} = -61$).

Premature Convergence under the Standard Setting. As shown in Figure 16, enforcing the standard target ($\mathcal{H} = -61$) results in high inter-seed variance. While some seeds converge to near-optimal policies, others become prematurely trapped in local minima (converging to returns of approximately 300). This substantially degrades the average asymptotic performance, with final returns reaching only around 650.

Analysis: In these 61-DoF *Basketball* runs, the lower entropy target coincides with faster variance decay before robust whole-body coordination emerges.

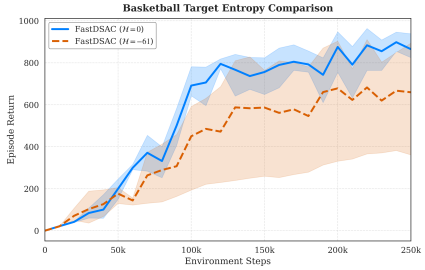


Figure 16: Ablation on target entropy in *Basketball*. The standard heuristic ($\mathcal{H} = -61$) is associated with rapid variance decay, high inter-seed variance, and premature convergence (orange curve). FastDSAC maintains $\mathcal{H} = 0$ to provide sufficient exploration capacity, with DEM modulating exploration across action dimensions. Both curves report results over 3 seeds.

Target Entropy and DEM. In this ablation, FastDSAC with $\mathcal{H} = 0$ converges more reliably than with $\mathcal{H} = -61$. The entropy target controls overall stochasticity, while DEM lets the actor autonomously modulate exploration across action dimensions. This combination supports continued exploration while accounting for learned differences in exploration requirements at the joint-level exploration scale.

O Sensitivity to DEM Temperature

To examine how the fixed DEM temperature affects learning, we compare different τ values on three representative tasks: *Hurdle*, *Room*, and *Stair* (Figure 17). For *Room* and *Stair*, we evaluate $\tau \in \{0.5, 1, 5\}$; for *Hurdle*, we additionally include $\tau = 10$.

The effect is clearly task-dependent. On *Stair*, the three temperatures $\tau \in \{0.5, 1, 5\}$ all lead to similar final performance, indicating that this task is relatively insensitive to the concentration of DEM's modulation weights. On *Room*, the default setting $\tau = 1$ performs best, which aligns with our use of $\tau = 1$ as the unified default configuration. On *Hurdle*, however, a larger temperature is more effective: $\tau = 10$ achieves the best final return, while the default $\tau = 1$ is weaker than $\tau = 0.5$, $\tau = 5$, and $\tau = 10$. This suggests that some highly dynamic coordination tasks can benefit from less concentrated relative modulation.

These observations motivate the FastDSAC (auto-τ) variant, which we evaluate only on *Balance Hard*, *Run*, and *Hurdle*. We observe that end-to-end temperature optimization can further improve performance in these experiments. At the same time, since the fixed-τ configuration already performs strongly across the three benchmarks, we do not pursue this direction further here and instead leave a broader study of automatic temperature adaptation to future work.

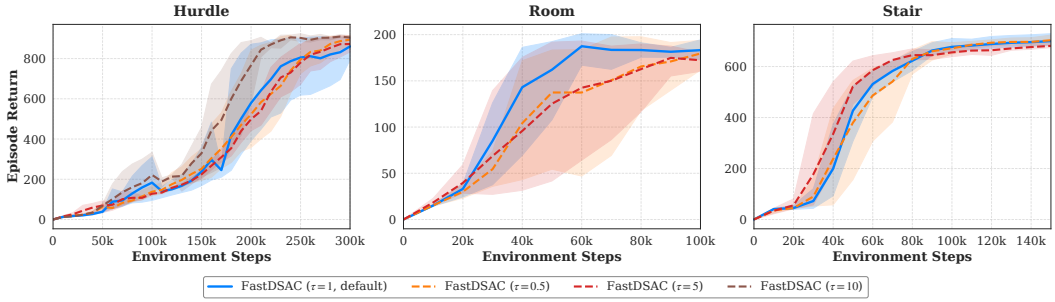
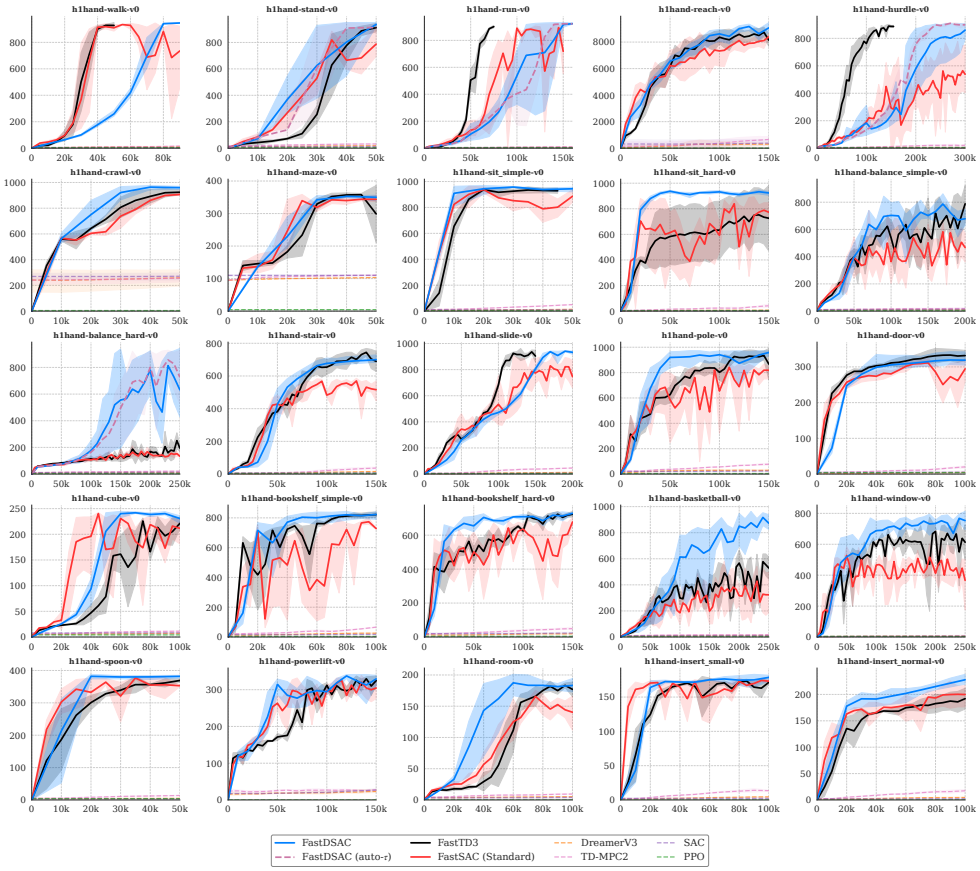
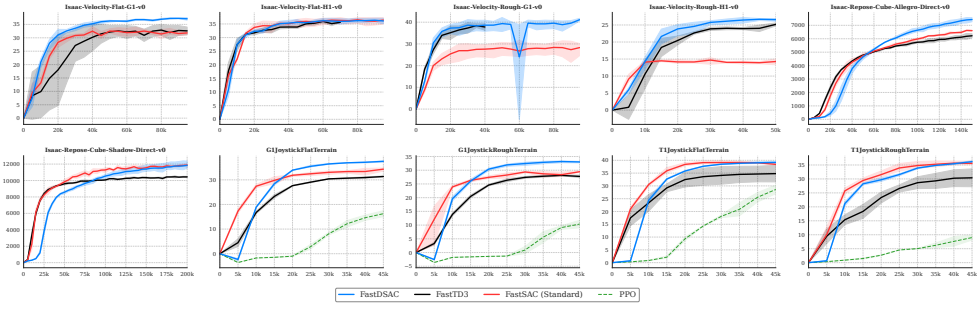


Figure 17: **Sensitivity to the DEM temperature τ.** Learning curves on *Hurdle*, *Room*, and *Stair* under different fixed temperatures. The default setting $\tau = 1$ remains a robust choice overall. *Room* favors $\tau = 1$, *Stair* shows similar performance across $\tau \in \{0.5, 1, 5\}$, and *Hurdle* benefits from a larger temperature, with $\tau = 10$ achieving the best final return.

P Full Results



(a) Full results on HumanoidBench (25 tasks).



(b) Full results on IsaacLab and MuJoCo Playground (10 tasks).

Figure 18: **Full results across all benchmarks.** Curves show mean return over training steps with shaded regions indicating the min-max range across 3 seeds. **(a) HumanoidBench (25 tasks):** FastDSAC (blue) is compared against FastTD3 (black), FastSAC Standard (red), and leading baselines including DreamerV3 (orange), TD-MPC2 (pink), SAC (purple), and PPO (green). FastDSAC often outperforms or matches these baselines across locomotion (e.g., *Run*, *Hurdle*), fine manipulation (e.g., *Insert*), and complex whole-body coordination (e.g., *Basketball*, *Balance*), combining strong exploration with stable training. **(b) IsaacLab and MuJoCo Playground (10 tasks):** FastDSAC (blue) is compared against FastTD3 (black), FastSAC Standard (red), and PPO (green). On IsaacLab, FastDSAC performs strongly on locomotion (*Velocity Flat/Rough H1/G1*) and *Repose Cube Allegro*, while remaining competitive on *Repose Cube Shadow*. On MuJoCo Playground, FastDSAC consistently outperforms baselines on *G1/T1 Joystick* with lower variance and higher asymptotic returns.

Q Discussion of Limitations, Future Directions and Impact

Limitations. The primary limitation of FastDSAC is that Dimension-wise Entropy Modulation (DEM) assumes independent exploration noise across action dimensions (i.e., utilizing a diagonal covariance matrix). While this formulation effectively scales to 61-DoF systems, it precludes the modeling of correlated exploration strategies among interdependent joints. Furthermore, maintaining a continuous distributional critic comes at the cost of a marginal increase in computational overhead during training compared to purely deterministic baselines. Finally, the Unitree G1 deployment is a qualitative feasibility demonstration; we do not report quantitative real-world metrics and comparison.

Future Directions. An important direction for future work is to study how structured covariance models might be used to capture joint dependencies, and how end-to-end optimization of DEM's temperature parameter can enable fully autonomous variance scheduling.

Impact Statements. This paper presents work whose goal is to advance the field of Deep Reinforcement Learning. The deployment of such agile autonomous systems raises valid safety concerns regarding unintended physical interaction and potential long-term economic impacts on the workforce. Consequently, real-world application of these policies necessitates robust verification protocols and rigorous safety constraints to prevent accidents and ensure physical safety.

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 same core contributions that are later supported by the experiments and conclusion: DEM, the continuous distributional critic, strong results on 35 tasks, and zero-shot sim-to-real transfer (Abstract, Introduction, Sections 4.2–4.4, Conclusion).

   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: The paper includes a dedicated limitations section in Section Q. It explicitly notes the diagonal-covariance assumption in DEM and the marginal training overhead of the continuous critic, and it also points to structured covariance models and auto-τ as future directions.

   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: The paper does not present proofs; the convergence statement is attributed to prior DSAC/DSAC-T work, while the rest of the paper focuses on algorithm design and experiments.

   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: The paper provides: (1) the official public implementation link in the Introduction; (2) complete hyperparameter configurations in Appendix A; (3) benchmark/task lists, training protocols, seed counts, and evaluation settings in Section 4 and Appendix A; and (4) sim-to-real deployment details in Appendix B. All baselines use public codebases with configuration details specified.

   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: [Yes]

   Justification: The paper provides a public implementation link and describes how the benchmarks and evaluation pipelines are configured. The experiments rely on public benchmark environments rather than proprietary datasets, and the appendix gives the settings needed to reproduce them (Introduction, Appendix A).

   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: Section 4 and Appendix A specify the benchmarks, baselines, seed counts, parallel environment counts, replay/update schedules, discount factors, target entropy, and benchmark-specific architectural choices such as LayerNorm.

   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: The main figures report mean returns over 3 seeds with shaded min-max ranges, and the hardest tasks use 5 seeds with shaded regions spanning the min-max range across seeds, so the paper does provide a clear measure of run-to-run variability for the key results (Sections 4, 4.2, Appendix P).

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: The Computational Resources section and Appendix H specify the CPU/GPU/RAM configurations, task allocation across hardware, and per-task as well as total wall-clock time needed for the representative experiments.

   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: The work uses public simulation benchmarks and a robot deployment, and the paper does not describe any practice that appears to conflict with the NeurIPS Code of Ethics. No human-subject data, privacy-sensitive data, or prohibited collection practices are involved.

   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: [Yes]

    Justification: See Appendix Q.

    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: The paper does not release a high-risk dual-use model or a scraped dataset. Therefore, no additional safeguard procedures are required beyond the safety considerations discussed in Appendix Q.

    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: The paper uses open-source benchmarks (HumanoidBench, MuJoCo Playground, IsaacLab) and the FastTD3 codebase. All assets are properly cited in the text with their original papers. HumanoidBench is under MIT License, MuJoCo Playground under Apache 2.0, IsaacLab under BSD 3-Clause, and FastTD3 under MIT License. The Unitree G1 robot is a commercially available platform.

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: The new asset is the public implementation of FastDSAC. It is documented through the methodology, Algorithm 1, the hyperparameter tables, and the sim-to-real appendix.

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 or any human-subject study; the real-world evaluation is on a humanoid robot, not on participants.

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: Because the paper does not collect data from human participants or run a human-subject study, IRB approval is not applicable.

    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: The core method, experiments, and results do not rely on LLMs, and LLMs were only used for polishing and formatting purposes; the paper's scientific contributions are entirely in reinforcement learning and robotics.

    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.