REGULARISED RL IMPROVES SIM-TO-REAL TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Regularised reinforcement learning (RL) has provable robustness guarantees that should ease the transfer of policies from simulation to robots, known as sim-to-real. Yet, weakly regularised algorithms like Proximal Policy Optimisation (PPO) remain the de facto choice in sim-to-real pipelines, while the practical benefits of regularisation for transfer remain poorly characterised. In this work, we study the sim-to-real performance of commonly used RL algorithms, with the aim of characterising the effects of regularisation on robotic performance. To that end, we derive Soft MellowMax Actor-Critic (SMM-AC), a novel regularised RL algorithm based on the soft mellowmax value function, which carries strong robustness guarantees. On MuJoCo continuous control benchmarks, SMM-AC is competitive with baselines like Soft Actor-Critic (SAC). On real robots, SMM-AC achieves higher return, while exhibiting greater robustness to perturbations in observations, the environment, and dynamics than baselines. Together, these results demonstrate that regularised RL improves real-world policy deployment through better robustness to the sim-to-real gap, and changing real-world conditions.

1 INTRODUCTION

Reinforcement learning (RL) has become a standard tool for robotic control, with on-policy, weakly regularised algorithms such as Proximal Policy Optimisation (PPO) (Schulman et al., 2017) serving as the de facto choice in most sim-to-real pipelines. Recent work has shown that carefully stabilised off-policy RL algorithms can match or outperform PPO in massively parallel simulators while also transferring to real legged robots, including humanoids and quadrupeds (Gallici et al., 2025; Seo et al., 2025a;b; Kim et al., 2026; Sabatini et al., 2026). However, these evaluations primarily focus on task performance: whether a policy can complete a task under the conditions for which it was trained. In real-world deployment, policies must contend with changes in observations, environment properties, and system dynamics that are difficult to model exhaustively in simulation a priori. Robustness to such perturbations is therefore an important aspect of reliable sim-to-real transfer.

Regularisation provides a principled way to induce robustness. Regularised RL has long dominated benchmarks in continuous control and arcade games, demonstrating superior sample efficiency and stable optimisation (Dam et al., 2024; Haarnoja et al., 2018a; Ahmed et al., 2019). Beyond these practical benefits, the theoretical duality between regularised and robust RL suggests that regularisation can also provide implicit robustness to dynamics and task perturbations (Derman et al., 2021; Eysenbach & Levine, 2022; Husain et al., 2021; Brekelmans et al., 2022). This mechanism is complementary to Domain Randomisation (DR), which explicitly trains policies over a user-specified distribution of environment variations, such as changes to initial conditions or physical parameters (Eysenbach & Levine, 2022; Mittal et al., 2025). Whereas DR requires the practitioner to anticipate and specify the perturbations of interest, regularisation can induce robustness without explicitly enumerating them. This raises a natural empirical question: *does the robustness induced by regularisation translate into more reliable real-world policy deployment?*

Robotics is a natural testbed to evaluate the robustness of policies because the sim-to-real gap is precisely an unmodeled change in the dynamics under which the policy must achieve high return. **We hypothesize that policies robust to a larger set of perturbations achieve 1) higher real-world return under sim-to-real transfer, and 2) higher retention of their performance under deployment-time dynamics perturbations.** Jiralerspong et al. (2025) show that RL with the Soft MellowMax (SMM) value function is robust to a larger set of perturbations than entropy-regularised RL, making it a natural target for our investigation. However, SMM has only been defined for

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

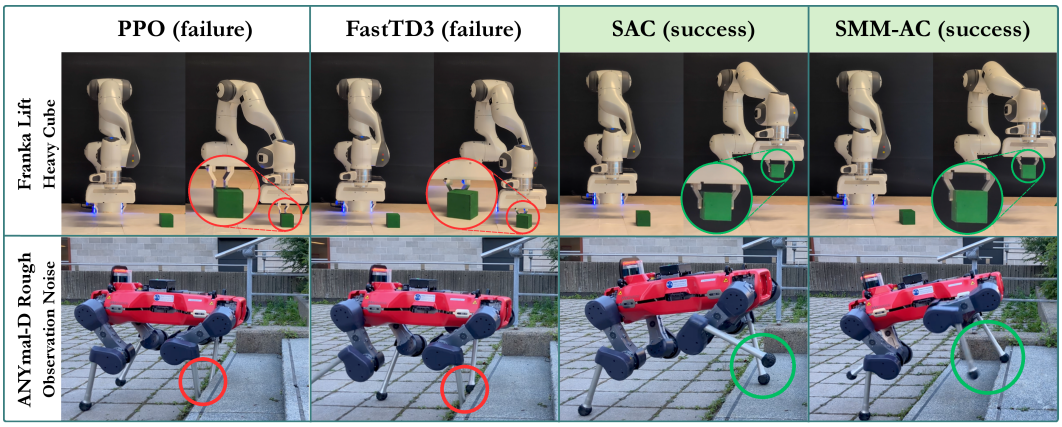

Figure 1: Overview of the Sim-to-Real Robustness Evaluation. Top: Agents trained on the Franka Lift task in simulation are deployed with a cube with a different center-of-mass than the one seen during training. The first frame shows the initial state, while the second shows the outcome, where the policy either successfully lifts the cube or fails to grasp it. Bottom: Agents trained on the ANYmal-D Rough task are deployed in the real world to climb up stairs. With noise added to the observation, non-regularised methods fail to climb the stairs.

discrete state-action spaces, usually in Q-learning algorithms (Asadi & Littman, 2017; Gan et al., 2021). Extending SMM to continuous continuous control is non-trivial, as the value function requires integrating over an intractable reference policy. To overcome this, we propose Soft MellowMax Actor-Critic (SMM-AC), which approximates the SMM reference policy using a dedicated, slowly updated reference network, ensuring stable actor-critic optimisation in continuous domains.

To test our hypotheses, we evaluate SMM-AC alongside PPO, SAC, and TD3 in simulation and on real-world robots, including a Franka arm and an ANYmal-D quadruped. In addition to real-world task performance, we evaluate policies under deployment-time perturbations to observations, environment properties, and system dynamics. Across these experiments, regularised methods maintain higher performance in sim-to-real transfer and under real-world perturbations, while SMM-AC achieves the strongest performance and robustness across tasks. We also show that regularised methods are more likely to be deployable than standard RL policies, even under large sim-to-real gaps. Figure 1 illustrates this sim-to-real robustness in the Franka cube-lifting task and in ANYmal-D rough-terrain locomotion, showing that under real-world perturbations, regularised methods deploy successfully. **Together, our results provide empirical evidence that the robustness induced by regularisation can translate into improved reliability during real-world deployment.** In this work we

- extend soft mellowmax to continuous state-action spaces and derive **Soft MellowMax Actor-Critic (SMM-AC)**, a practical off-policy actor-critic algorithm based on SAC (Sec. 3);
- validate SMM-AC's performance in simulation on MuJoCo continuous-control benchmarks, showing that it achieves similar average returns to SAC and other baselines (Sec. 4);
- characterise the sim-to-real performance and real-world perturbation robustness of RL policies on manipulation and locomotion tasks, showing that SMM-AC achieves higher real-world returns and maintains stronger performance under perturbations than the baselines (Sec. 5).

2 Background & Related Work

Regularized Markov decision processes and RL. Regularized Markov Decision Processes (MDPs) append a strongly convex regulariser $\Omega : \Delta_{\mathcal{A}} \to \mathbb{R}$ to the MDP tuple: $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \Omega, T)$ (Geist et al., 2019). Let $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $p : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, γ is the discount factor, and T and horizon. By $\Delta_{\mathcal{S}}$ and $\Delta_{\mathcal{A}}$, we denote the probability simplices over the state and action spaces, respectively.

Eq. 1 shows the regularised policy evaluation functions V^{π}_{Ω} and Q^{π}_{Ω}, defined over all states and state-action pairs, for policy $\pi(\cdot|s) \in \Delta_{\mathcal{A}}$ and inverse temperature ω. In regularised RL, $\Omega(\pi(\cdot|s))$ is subtracted from the value function, penalising deviation from some desirable property of the policy.

Table 1: Optimality equations for various regularised RL formulations, where $\pi^{\text{ref}}(\cdot|s) = \text{softmax}(\alpha Q^\star_\Omega(s))$ is an SMM reference policy, and $Q^\star_\Omega(s) = Q^\star_\Omega(s, \cdot)$, for inverse temperatures α and ω.

Name	**Regulariser** $\Omega(\pi)$	**Value** $V^\star_\Omega(s)$	**Policy** $\pi^\star_\Omega(s)$
Negative Entropy	$\sum_a \pi(a\|s) \log \pi(a\|s)$	$\frac{1}{\omega}\text{log-sum-exp}\big(\omega Q^\star_\Omega(s)\big)$	$\text{softmax}\big(\omega Q^\star_\Omega(s)\big)$
Soft Mellowmax	$D_{\text{KL}}\big(\pi(s) \,\|\, \pi^\alpha_Q(s)\big)$	$\frac{1}{\omega}\log \sum_a \pi^\alpha_Q(a\|s) e^{\omega Q^\star_\Omega(s,a)}$	$\text{softmax}\big((\alpha+\omega) Q^\star_\Omega(s)\big)$

For instance, using negative entropy results in more uniform policies, whereas KL divergence results in policies that are closer to a reference (Neu et al., 2017; Haarnoja et al., 2018a; Geist et al., 2019):

$$V^\pi_\Omega(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[Q^\pi_\Omega(s,a)] - \frac{1}{\omega}\Omega(\pi(\cdot|s)), \quad Q^\pi_\Omega(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim p(\cdot|s,a)}[V_\Omega(s')] \tag{1}$$

Maximising V^π_Ω with respect to the policy yields the optimal regularised value function. Table 1 lists the optimality equations for various regularised RL formulations in discrete action spaces. It should be noted that regularisation changes the optimal policy compared to standard RL, which can lead to performance failures in situations where deterministic policies are optimal (Zhang et al., 2025).

The benefits of regularisation. Regularisation confers certain benefits to both the learning process and the robustness of the policy. Ahmed et al. (2019) show that entropy regularisation smooths the loss landscape, leading to more stable and faster training. Haarnoja et al. (2018a) demonstrate that SAC, an entropy-regularised off-policy RL algorithm, is significantly more sample efficient than standard RL. Alongside the benefits to learning, many works have examined the theoretical duality between regularisation and robustness. For instance, Eysenbach & Levine (2022) and Brekelmans et al. (2022) show that entropy- and KL-regularised RL are robust to reward and dynamics perturbations, indicating that policy regularisation improves robustness overall.

The duality between regularisation and robustness is useful in many application domains, *especially when the environment is not perfectly modelled* (Derman et al., 2021; Jiralerspong et al., 2025). Such is the case in robotics, making the sim-to-real transfer of robotic policies a natural evaluation of how theoretical robustness guarantees transfer to the real world. Theoretical robustness guarantees for regularised RL are stated as the sets of worst-case adversarial perturbations on the reward or dynamics for which the policy still achieves high return. Thus, we hypothesize that the larger the robust set, the more robust a policy to perturbations, and the better the sim-to-real transfer. The size of these robust sets is controlled by the regulariser and its hyperparameters, like temperature (Eysenbach & Levine, 2022; Derman et al., 2021; Husain et al., 2021; Brekelmans et al., 2022).

Mellowmax and soft mellowmax value functions. Regularised value functions often take the form of a soft maximum, such as the log-sum-exp function in entropy-regularised RL. Mellowmax and SMM are alternative soft value functions (Table 1) proposed to improve the convergence and stability properties of Q-learning algorithms in discrete action spaces (Asadi & Littman, 2017; Gan et al., 2021). Jiralerspong et al. (2025) demonstrate that RL with SMM induces larger robust sets than entropy-regularised RL in tree-structured state-action spaces. These benefits make SMM-based RL enticing for robotics, improving stability and expanding robustness guarantees, which we expect to improve 1) sim-to-real transfer and 2) performance under perturbations. Note that we do not expect SMM to dominate SAC in simulation. *We expect SMM to achieve better sim-to-real properties*.

Soft actor-critic. Haarnoja et al. (2018a) proposed SAC, the go-to RL algorithm for learning stochastic entropy-regularised policies in continuous action spaces. In continuous action spaces, we are often required to take integrals over the action space, for instance, the log-sum-exp value function becomes log-integral-exp in continuous spaces, which is intractable and whose estimator would be biased. SAC designs around these integrals by instead constructing one-sample Monte Carlo estimators of the value function V^π_Ω (Eq. 1), for use in the Q-value target:

$$\hat{Q}_{\text{SAC}}(s,a) = r(s,a) + \gamma \hat{V}_{\text{SAC}}(s'), \quad (s,a,s',r) \sim \mathcal{D}, \tag{2}$$

$$\hat{V}_{\text{SAC}}(s') = Q_\theta(s',a') - \frac{1}{\omega}\log(\pi(a'|s')), \quad a' \sim \pi(\cdot|s'), \tag{3}$$

where $\mathcal{D}$ is the replay buffer, storing previously sampled state, action, next state, and reward tuples, and ω is the inverse temperature. SAC parameterises its policy as a Gaussian whose mean and

standard deviation are learned, so the log-probability in Eq. 3 is available in closed form. Writing Q_θ for the critic network with parameters θ, the critic is trained by constructing a loss based on this bootstrapped Q-value target:

$$J_Q(\theta) = \mathop{\mathbb{E}}_{(s,a,s',r)\sim\mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(s,a) - \mathrm{sg}\left(\hat{Q}_{\mathrm{SAC}}(s,a) \right) \right)^2 \right], \tag{4}$$

where $\mathrm{sg}(\cdot)$ is the stop gradient function. The stochastic policy is trained in the same fashion, with the policy target being the softmax of the Q-values, and the KL divergence as the loss function. Writing π_z^{act} for the actor network with parameters z, and drawing actions through the reparameterisation trick for noise ϵ, the policy loss without terms that would cancel under a gradient is:

$$J_{\mathrm{act}}(z) = \mathop{\mathbb{E}}_{s\sim\mathcal{D},\epsilon\sim\mathcal{N}(0,I)} \left[\log \pi_z^{\mathrm{act}}(a|s) - \omega Q_\theta(s,a) \right], \quad a = f_z(s,\epsilon). \tag{5}$$

Many subsequent works have improved SAC through techniques such as double Q-networks, delayed policy updates, larger training datasets, distributional critics, and architectural improvements (Haarnoja et al., 2018b; Seo et al., 2025a; Kim et al., 2026).

Accelerated off-policy algorithms. Off-policy learning with function approximation and bootstrapping is notoriously unstable, often referred to as the "deadly triad" (Van Hasselt et al., 2018). To address these instabilities while leveraging modern hardware, recent works have shifted toward massively parallel simulators (Mittal et al., 2025; Genesis, 2026) and high-throughput training algorithms. Algorithms such as Parallel Q-learning (Li et al., 2023) and Parallel Q-networks (Gallici et al., 2025) stabilise training by scaling batch sizes and network capacity. Building on these principles, FastTD3 (Seo et al., 2025b), FastSAC (Seo et al., 2025a), and FlashSAC (Kim et al., 2026) further improve stability and sample efficiency with distributional critics (Bellemare et al., 2017). We implement SMM-AC in both the CleanRL (Huang et al., 2022) and FlashSAC frameworks, showing that its stability and scalability are preserved across implementations and training regimes.

RL in robotics. The deployment of deep RL in robotics heavily relies on careful reward design and policy stability. For instance, Mock & Muknahallipatna (2023) compared PPO, TD3, and SAC on simulated legged robots, demonstrating that each algorithm's performance is highly sensitive to the chosen reward function. To improve stability independent of reward tuning, Mysore et al. (2021) introduced a state-to-state KL regularisation scheme that minimizes the temporal and spatial variation of the policy to improve deployment in quadcopters. The resulting regularised policies were more energy efficient and more often flight-worthy than standard PPO. Entropy-regularised policies like SAC have also shown advantages over counterparts in robotics. For example, SAC consistently outperforms baselines in mobile robot navigation, and has been successfully deployed on real-world humanoid robots and in quadrupedal locomotion and dexterous manipulation tasks (de Jesus et al., 2021; Haarnoja et al., 2018b; Seo et al., 2025a; Kim et al., 2026). Building upon this foundation, our work demonstrates that regularisation, with its dynamics robustness, not only matches the performance of baselines in simulation, but also leads to better sim-to-real transfer.

3 Soft MellowMax Actor-Critic

Given its stability and theoretical robustness guarantees, SMM is an attractive candidate for continuous control. Adapting SMM into continuous state-action spaces for an actor-critic algorithm requires many of the same design choices as for SAC. However, SMM is KL-regularised against an intractable reference policy π^{ref} (*c.f.*, Table 1), which critically depends on the Q-values:

$$\pi^{\mathrm{ref}}(a|s) = \frac{\exp(\alpha Q(s,a))}{\int_{b\in\mathcal{A}} \exp(\alpha Q(s,b)) db} \tag{6}$$

The policy evaluation function for SMM is stated in Eq. 7 for the SMM policy π. At each iteration, an SMM policy is rewarded for staying close to the drifting reference policy. Note the difference between SAC and SMM: SAC's policy is rewarded for being uniform over the action space, while SMM's policy follows a greedier softmax. Compared to PPO, SMM's reference policy keeps the actor iterates anchored to a high-performing region, whereas PPO just keeps actor iterates close together (Schulman et al., 2017). Intuitively, SMM-AC is thus a sort of interpolation between PPO and SAC: SMM-AC's policy can be sharper than SAC's, while still finding multiple high-return

trajectories, because the reference provides a known, high-performing region. Thus, even under dynamics perturbations, an SMM policy can likely find high-return trajectories to complete a task.

$$V^{\pi}_{\text{SMM}}(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[Q(s,a)\right] - \frac{1}{\omega} D_{KL}(\pi(\cdot|s)||\pi^{\text{ref}}(\cdot|s)) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[Q(s,a) - \frac{1}{\omega}\log\left(\frac{\pi(a|s)}{\pi^{\text{ref}}(a|s)}\right)\right] \quad (7)$$

Neither Eq. 6 nor these expectations can be evaluated exactly in continuous action spaces, because the integral of the reference policy is intractable. The rest of this section replaces each intractable quantity with a tractable estimator. In order to do so, we propose learning three networks: the actor π^{act}_z, the critic Q_θ and the reference policy π^{ref}_ϕ.

Critic. The critic network, Q_θ, produces Q-value estimates of state-action pairs. Like in SAC, Q-value estimates are generated to train the critic using an estimator based on the regularised policy evaluation function: V^{π}_{SMM}. Since we learn π^{act}_z and π^{ref}_ϕ, their log-probabilities can be obtained for any sampled action. The one-sample Monte Carlo Q-value target is generated as follows:

$$\hat{Q}_{\text{SMM}}(s,a) = r(s,a) + \gamma \hat{V}_{\text{SMM}}(s'), \quad (s,a,s',r)\sim\mathcal{D} \quad (8)$$

$$\hat{V}_{\text{SMM}}(s') = Q_\theta(s',a') - \frac{1}{\omega}\log\left(\frac{\pi^{\text{act}}_z(a'|s')}{\pi^{\text{ref}}_\phi(a'|s')}\right), \quad a'\sim\pi^{\text{act}}_z(\cdot|s') \quad (9)$$

This value estimate serves only to bootstrap the temporal-difference target of the critic, which is trained by minimising the Q-value prediction error:

$$J_Q(\theta) = \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s,a) - \text{sg}\left(\hat{Q}_{\text{SMM}}(s,a)\right)\right)^2\right] \quad (10)$$

where the gradient is stopped on $\hat{Q}_{\text{SMM}}(s,a)$ to avoid poor training dynamics, as is standard in off-policy actor-critic methods (Fujimoto et al., 2018; Haarnoja et al., 2018b).

Actor. Training the actor means fitting π^{act}_z to the regularised policy: $\text{softmax}((\alpha+\omega)Q_\theta(s))$ (Table 1). However, naively learning the actor through a loss function like SAC's in Eq. 5 leads to tremendous instability, often leading to divergence in the training process. Intuitively, the reason for this instability is a lack of grounding between the actor and the reference policy. In KL-regularised algorithms like PPO and TRPO, the actor is always trained with respect to a previous policy iterate, which provides improved stability and performance (Schulman et al., 2015; 2017). Similarly, we found that using a slowly updated reference network as a low-variance surrogate of the softmax of the raw Q-values, insulated the actor from critic noise. The actor's loss function is constructed as follows, where we use the reparameterisation trick (Kingma & Welling, 2014) to estimate the expectation with low variance, and remove terms whose gradient is zero (derivation in App. B):

$$J_{\text{act}}(z) = \mathop{\mathbb{E}}_{s\sim\mathcal{D},\epsilon\sim\mathcal{N}(0,I)}\left[\log\pi^{\text{act}}_z(a|s) - \log\pi^{\text{ref}}_\phi(a|s) - \omega Q_\theta(s,a)\right], \quad a = f_z(s,\epsilon) \quad (11)$$

Reference network. The reference network must track the softmax distribution over the Q-values as shown in Eq. 6. We train π^{ref}_ϕ using the actor loss of SAC (Eq. 5), with a substantially smaller learning rate than the actor (Table 3). This ensures that the reference network is a stable, low-variance surrogate for the actor updates and for generating Q-values estimates. The reference policy's loss is constructed as follows, again using the reparameterisation trick:

$$J_{\text{ref}}(\phi) = \mathop{\mathbb{E}}_{s\sim\mathcal{D},\epsilon\sim\mathcal{N}(0,I)}\left[\log\pi^{\text{ref}}_\phi(a|s) - \alpha Q_\theta(s,a)\right], \quad a = f_\phi(s,\epsilon) \quad (12)$$

Putting it together. On each batch drawn from $\mathcal{D}$, SMM-AC descends Eq. 10 for the critic, Eq. 11 for the actor, and Eq. 12 for the reference network at its slower rate. By keeping the replay buffer, target networks and critic architecture identical to SAC, the effect of the regulariser is isolated, allowing us to test our primary hypothesis: that SMM's expanded robust sets (*i.e.*, from the greedier π^{ref}) translate into better sim-to-real performance and perturbation resistance. This also means that SMM-AC can be easily adapted to fit the FastSAC (Seo et al., 2025a), and FlashSAC (Kim et al., 2026) frameworks, just with the addition of the reference network and its loss.

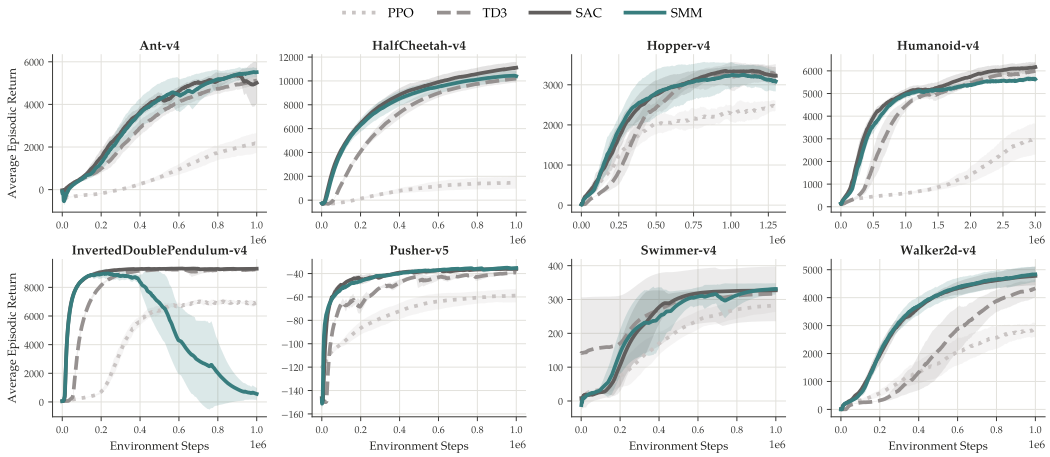

Figure 2: Smoothed episodic returns of SMM-AC, SAC, PPO, and TD3 over the number of steps taken in the environment. The best results were selected according to the performance of the algorithm at the final step.

4 Experimental results on MuJoCo

We evaluate SMM-AC in simulation on MuJoCo continuous-control tasks (Todorov et al., 2012) to demonstrate that it achieves similar performance to widely used RL baselines: SAC, TD3, and PPO (Haarnoja et al., 2018b; Fujimoto et al., 2018; Schulman et al., 2017). We implement SMM-AC in the CleanRL (Huang et al., 2022) framework, to demonstrate its performance in basic RL setups (in contrast to FlashSAC, which employs many acceleration techniques). Table 3 reports the hyperparameter settings we swept over for each algorithm, with five random seeds each. We tried to keep the number of training runs the same between each baseline, but some algorithms had fewer hyperparameters to sweep across. We report the best-performing hyperparameter settings in Table 4, see App. C for more experiment details. For SMM-AC specifically, we set $\alpha = 1$ (inverse temperature of the reference policy, π^{ref}) and sweep ω (inverse temperature of the SMM policy) values, for all experiments. SAC uses temperature autotuning, which is not yet possible for SMM.[1]

Figure 2 compares the performance of SMM-AC and the baselines across the MuJoCo tasks. SMM-AC is competitive with SAC and TD3, the strongest baselines, although its performance collapses on `InvertedDoublePendulum-v4`.[2] Outside this failure case, SMM-AC reaches returns comparable to SAC at a similar rate, suggesting comparable sample efficiency on the evaluated tasks. Additional experiments using the highly optimised FlashSAC library (Kim et al., 2026) show that FlashSMM-AC (our method implemented on top of FlashSAC) achieves returns comparable to FlashSAC, with both outperforming FastTD3 (Seo et al., 2025b) and PPO (App. D). Clearly, SMM-AC and FlashSMM-AC are highly capable for continuous control. As the next section shows, however, higher average returns in simulation do not necessarily translate into better sim-to-real transfer.

5 Robotic experiments

Our goal is to test whether regularisation makes robotic policies more robust to the sim-to-real gap and to perturbations after deployment. While Sec. 4 demonstrates that SMM-AC learns competitive policies in simulation, the question of real-world transfer robustness remains. In this section, we deploy policies trained in simulation on two robots. We begin with evaluation on a simulation-to-real matched setup, then perturb individual deployment parameters (sensor noise, object mass, surface friction) to assess robustness under domain shift. Subsec. 5.2 evaluates sim-to-real transfer, and Subsec. 5.3 evaluates robustness to these deployment-time perturbations. SMM-AC achieves the highest return under every perturbation we tested, supporting our hypothesis that regularised policies would be the most robust to the sim-to-real gap and perturbations.

[1] We found autotuning ω or α to be far too unstable, so we elected to fix α and sweep ω.

[2] We couldn't identify the reason for this failure, but the consistency with which it occurred suggests some learning instability, likely with the reference policy.

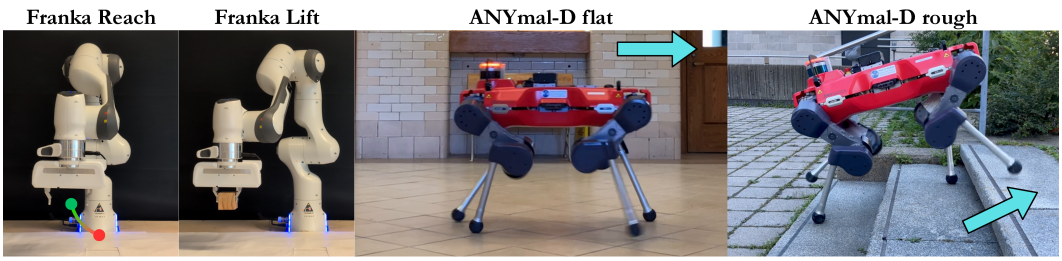

Figure 3: The four real-world evaluation settings: Franka Reach and Lift, and ANYmal-D velocity tracking on flat and rough terrain. For the ANYmal-D tasks, blue arrows indicate the robot's direction of commanded velocity.

5.1 Experimental setup

Tasks and platforms. We evaluate four tasks with two robots: a manipulator and a quadruped. On the Franka Panda arm, the Reach task requires moving the end effector to a target pose, while the Lift task requires grasping a cube and lifting it to a target position. On an ANYmal-D quadruped, we evaluate velocity tracking on flat and rough terrains. The flat-terrain policy is blind to exteroceptive information, whereas the rough-terrain policy receives perceptual observations. Policies are trained in the corresponding `Isaac-Reach-Franka-v0`, `Isaac-Lift-Cube-Franka-v0`, `Isaac-Velocity-Flat-Anymal-D-v0`, and `Isaac-Velocity-Rough-Anymal-D-v0` environments using the default DR ranges specified in App. E. Figure 3 shows the four deployment settings.

Algorithms and configurations. We compare PPO, FastTD3, SAC, and SMM-AC. PPO is implemented with RSL-RL (Schwarke et al., 2025), with sweeps over its entropy coefficient, learning rate, target KL, and rollout length. FastTD3 is a strong unregularised off-policy baseline (Seo et al., 2025b). We report both at their best configuration per task. SAC and SMM-AC use the FlashSAC implementation (Kim et al., 2026). We refer to these simply as SAC and SMM-AC below. For both regularised methods, we fix $\alpha = 1$ and sweep $\omega \in \{400, 800, 1200\}$. We do not use SAC temperature autotuning, since doing so would remove the controlled variation in regularisation strength that is central to this experiment. Each configuration is trained with five random seeds.

Reward scales. The reward functions have substantially different scales across tasks, with a good episode returning roughly 0.7 on Reach, 150 on Lift, and 20 on either ANYmal task. Because the regularisation term in the value function is scaled by $1/\omega$ (Eq. 1), a fixed inverse temperature induces different effective regularisation strengths across tasks. We therefore sweep ω separately for each task and report all three values.

Perturbations. To test robustness, we perturb task parameters while keeping each trained policy fixed. The policy, command, and low-level controller are unchanged throughout. On Franka, we add Gaussian observation noise ($\sigma = 0.005$ on Reach, and $\sigma = 0.001$ and $\sigma = 0.005$ on Lift). On Lift, we additionally use a cube of twice the training mass, or a cube with a 15mm center-of-mass (CoM) offset. On ANYmal, we lower surface friction by deploying on wet concrete, push the base (flat terrain only), add a 7kg payload, and add Gaussian noise ($\sigma = 0.001$) to the proprioceptive inputs at every step. For safety reasons, a configuration is deployed under a perturbation only if it fails fewer than 80% of simulated episodes under the same perturbation. This rule is fixed in advance and applied to every configuration. As a result, PPO and FastTD3 are excluded from all ANYmal perturbation experiments. These perturbations are similar to Danesh et al. (2025), who evaluate ANYmal-D on surfaces with different friction and with a 3kg payload.

Metrics. We report the mean and one standard deviation of the episode return across seeds. Table 2 lists the return of every configuration without perturbation and under each perturbation. Figure 4 shows the return under each perturbation as a percentage of the same configuration's return without it. Therefore, a value of 100% indicates that the perturbation incurs no performance cost. Note that values slightly above 100% indicate no loss rather than an improvement. In App. F, we additionally report the task error without perturbation (Table 13), measured as the distance between the end effector or cube and the target at the end of the episode for Franka, and as the root mean squared error

Table 2: Return on hardware with and without perturbations. Entries are the mean and one standard deviation of the return across seeds. Each column is one trained configuration, and reading down a column within a task shows how its return changes under each perturbation. The *none* row is the deployment without perturbation. PPO and FastTD3 are shown at their best configuration per task. Highlighted entries indicate the highest return in each row. Entries marked with "—" were not deployed, because the configuration failed the simulation safety check (Subsec. 5.1). Task errors for the *none* rows are reported in Table 13.

			SAC			SMM-AC (ours)		
Perturbation	PPO	FastTD3	ω=400	800	1200	ω=400	800	1200
Franka Reach								
none	0.69 ±0.04	0.40 ±0.09	0.48 ±0.12	0.58 ±0.09	0.53 ±0.09	0.51 ±0.08	0.61 ±0.03	0.45 ±0.05
obs. noise (σ=0.005)	0.19 ±0.05	0.24 ±0.03	0.35 ±0.03	0.40 ±0.03	0.35 ±0.02	0.45 ±0.02	0.56 ±0.02	0.33 ±0.01
Franka Lift								
none	133.1 ±54.8	123.1 ±4.1	92.9 ±83.8	125.8 ±69.9	94.9 ±86.0	148.5 ±8.6	152.9 ±2.5	123.9 ±69.4
cube mass 2×	97.5 ±11.0	102.9 ±9.5	81.9 ±5.2	99.1 ±4.4	87.1 ±5.4	152.5 ±9.1	151.6 ±3.1	128.3 ±8.2
CoM offset (15 mm)	118.2 ±5.6	106.8 ±3.9	93.4 ±4.5	115.0 ±7.8	96.2 ±5.9	154.4 ±9.2	153.0 ±2.8	127.6 ±10.3
obs. noise (σ=0.001)	38.0 ±8.4	96.5 ±3.0	81.1 ±4.8	103.4 ±11.6	82.1 ±5.5	147.0 ±3.1	150.2 ±2.3	122.4 ±2.5
obs. noise (σ=0.005)	19.4 ±9.2	72.4 ±5.7	60.7 ±7.2	68.5 ±7.8	62.5 ±6.6	120.9 ±12.2	120.9 ±4.0	107.7 ±4.6
ANYmal flat								
none	18.9 ±5.6	23.6 ±0.7	25.3 ±0.3	25.2 ±0.5	24.8 ±0.1	25.0 ±0.4	25.9 ±0.4	25.5 ±0.3
low friction	—	—	22.3 ±1.3	22.5 ±1.1	21.5 ±1.5	24.5 ±0.2	24.7 ±0.4	24.6 ±0.3
push	—	—	23.0 ±1.2	22.8 ±1.0	21.8 ±1.4	25.3 ±0.2	25.3 ±0.3	24.9 ±0.3
payload	—	—	24.1 ±0.7	22.6 ±1.1	22.8 ±0.9	25.1 ±0.2	25.2 ±0.3	25.0 ±0.2
proprioceptive noise	—	—	20.7 ±2.2	20.5 ±2.0	20.5 ±1.9	24.7 ±0.2	24.7 ±0.4	24.7 ±0.3
ANYmal rough								
none	8.6 ±3.7	13.7 ±4.1	19.1 ±1.6	19.3 ±4.9	18.8 ±3.4	21.0 ±1.2	20.8 ±1.3	21.9 ±1.6
low friction	—	—	16.4 ±1.5	17.0 ±0.8	17.4 ±1.0	18.5 ±0.6	21.2 ±0.6	21.9 ±1.1
payload	—	—	17.2 ±0.6	16.8 ±0.9	16.8 ±1.9	21.2 ±0.8	20.5 ±0.5	21.5 ±1.1
proprioceptive noise	—	—	15.3 ±1.6	15.3 ±1.9	13.5 ±1.0	18.8 ±1.4	19.3 ±0.7	18.9 ±1.9

between commanded and measured base velocity for ANYmal. An ANYmal episode is considered successful if the robot remains upright for the full 20-second episode.

5.2 Sim-to-real transfer

SMM-AC transfers best on Lift and both locomotion tasks. The *none* rows of Table 2 report the return of each policy deployed without perturbation. In locomotion, every regularised configuration achieves a higher return that PPO and FastTD3 on both terrains, with the largest gap on rough terrain. On Lift, the two best configurations are both SMM-AC. The best of these, at $\omega = 800$, is also far more consistent across seeds than PPO and SAC, whose large standard deviations indicate that their transfer varies considerably from seed to seed.

Regularised policies remain deployable under a larger sim-to-real gap. We further widen the sim-to-real gap by limiting domain randomisation in training (App. E) and deploying the resulting policies on hardware (App. F.1). Limiting domain randomisation lowers the return of every method, indicating that DR remains important for regularised policies as well. On ANYmal, however, PPO and FastTD3 no longer pass the simulation safety check once the randomisation is limited, and are thus not deployable. On Lift, SMM-AC achieves the highest return at every DR level. Regularisation and DR thus appear complementary, since limiting the randomisation hurts every method, while regularised policies remain deployable where unregularised ones are not.

5.3 Robustness to deployment-time perturbations

We now perturb task parameters and report the resulting return as a percentage of its unperturbed return in Figure 4. Table 2 reports the corresponding absolute returns.

SMM-AC keeps more of its return than the baselines. At the same ω, SMM-AC keeps more of its return than SAC under every perturbation, in most cases by more than two standard deviations combined. It retains more than PPO and FastTD3 wherever those were deployed, in every case by more than the combined standard deviations. Its results are also more consistent across seeds, with about half the standard deviation of SAC on average. Under most perturbations, SMM-AC loses

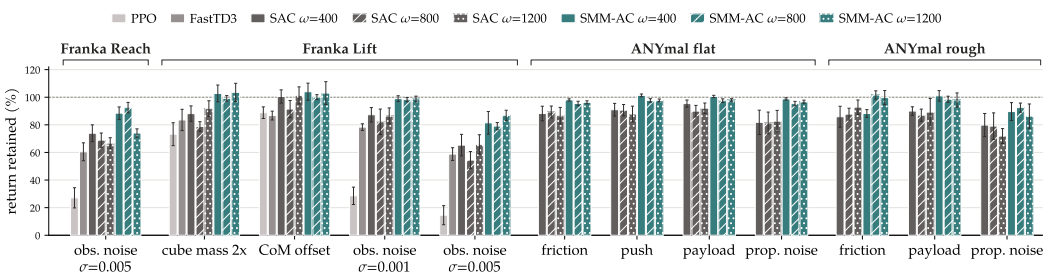

Figure 4: Return kept under real-world deployment-time perturbations. Return under each perturbation as a percentage of the same configuration's return without it, so 100% means no loss. Bars are seed means, and error bars show one standard deviation. Franka is perturbed with Gaussian observation noise and, on Lift, with a cube of $2\times$ the training mass and with a 15mm CoM offset. ANYmal is perturbed with lower surface friction, pushes, a payload, and proprioceptive noise. PPO and FastTD3 do not appear on ANYmal because they fail the simulation safety check, which is applied identically to every configuration.

little or none of its return, while the baselines retain considerably less. However, the empirical drop in performance depends on the type of perturbation.

Observation noise causes the largest losses. Every method has its largest loss under observation noise, and this is also where the methods differ the most. On Franka, PPO loses most of its return under noise, and FastTD3 and SAC lose a substantial part of theirs. Nevertheless, SMM-AC keeps most of its return. On ANYmal, proprioceptive noise separates SAC and SMM-AC more than any change to the physical system. Since the dynamics are untouched, these losses come entirely from how each policy responds to corrupted inputs.

Changes to the physical system cost less. A heavier or unbalanced cube costs SMM-AC nothing, while baselines lose part of their returns. On ANYmal, SMM-AC loses little under lower friction, pushes, and payload on either terrain, while SAC loses noticeably more. The one exception is lower friction on rough terrain at $\omega = 400$ where SMM-AC loses about as much as SAC. At other values of ω, it loses nothing.

SMM-AC maintains the highest absolute return. Note that a policy with a low return can still keep a high percentage of it. Therefore, Table 2 reports absolute returns as well. Under every perturbation, the highest return in achieved by SMM-AC. On Lift and locomotion tasks, every configuration of SMM-AC outperforms every deployed baseline under every perturbation. On Reach, under observation noise, SMM-AC at $\omega = 800$ achieves the highest return and PPO the lowest.

Discussion. These experiments separate two effects that are easy to conflate, namely transferring well to hardware, and keeping that performance as conditions move away from training. Regularisation helps most clearly with the second. Across all perturbations, SMM-AC keeps the most return and achieves the highest absolute return. Its advantage is largest under observation noise, which every real robot faces to some degree in real-world deployment. These results support our hypothesis that regularisation improves sim-to-real performance and robustness to perturbations. They also empirically validate theoretical robustness claims, providing evidence that SMM, with its larger robust sets, is indeed more robust.

6 Conclusion

We introduced SMM-AC, which extends soft mellowmax to continuous control by approximating its intractable reference distribution with a slowly updated reference network. SMM-AC is competitive with SAC and TD3 on MuJoCo tasks. On a Franka arm and an ANYmal-D quadruped, regularised methods outperformed PPO and FastTD3 on lift cube and both locomotion tasks. Among regularised methods, SMM-AC's best configuration achieved the strongest sim-to-real transfer performance on Lift and both locomotion tasks, and it retained more of its return than SAC in every perturbation condition. Reducing training-time domain randomisation degraded all methods, suggesting that regularisation complements DR rather than replacing it. Because the optimal temperature varied across tasks, future work should develop adaptive temperature scheduling that preserves robustness.

7 AI DISCLOSURE

In this work, we used generative AI tools for a number of tasks including

- to transcribe hand-written notes to Latex
- to check code for correctness
- to help implement methods (like SMM in FlashSAC) and IsaacLab environments in CleanRL.
- to write plotting and latex table generation code
- to help write some experiment code, particularly boilerplate evaluation scripts and bash scripts
- reviewing the writing of the paper, especially to focus narratives and reduce fluff

We took care to ensure that all the written work in this manuscript was human-generated. We have also not blindly followed the written code. We again, took great care to ensure that our work was sensible, and that it was honestly reported and conducted. The other AI disclosures are not relevant to this work.

We have reviewed all AI-assisted work. In particular, the core codebase has been written entirely by humans, and has been thoroughly checked, tested, and debugged. We used AI tools to modify pre-written code so that it could work in different codebases. The plotting code was already written, but AI was used to modify it for different plots and objectives. These were always thoroughly vetted for correctness. All latex transcriptions were treated as a starting point, and modified to fit the math we wrote down.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pp. 151–160. PMLR, 2019.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pp. 243–252. PMLR, 2017.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. Pmlr, 2017.

Rob Brekelmans, Tim Genewein, Jordi Grau-Moya, Grégoire Delétang, Markus Kunesch, Shane Legg, and Pedro Ortega. Your policy regularizer is secretly an adversary. *arXiv preprint arXiv:2203.12592*, 2022.

Tuan Dam, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. A unified perspective on value backup and exploration in monte-carlo tree search. *Journal of Artificial Intelligence Research*, 81:511–577, 2024.

Mohamad H Danesh, Maxime Wabartha, Stanley Wu, Joelle Pineau, and Hsiu-Chin Lin. Safe domain randomization via uncertainty-aware out-of-distribution detection and policy adaptation. *arXiv preprint arXiv:2507.06111*, 2025.

Junior Costa de Jesus, Victor Augusto Kich, Alisson Henrique Kolling, Ricardo Bedin Grando, Marco Antonio de Souza Leite Cuadros, and Daniel Fernando Tello Gamarra. Soft actor-critic for navigation of mobile robots. *Journal of Intelligent and Robotic Systems*, 102, 2021. doi: 10.1007/s10846-021-01367-5.

Esther Derman, Matthieu Geist, and Shie Mannor. Twice regularized mdps and the equivalence between robustness and regularization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22274–22287. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bb1443cc31d7396bf73e7858cea114e1-Paper.pdf.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=PtSAD3caaA2.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *International Conference on Learning Representations*, volume 2025, pp. 78148–78190, 2025.

Yaozhong Gan, Zhe Zhang, and Xiaoyang Tan. Stabilizing q learning via soft mellowmax operator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7501–7509, 2021.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *International conference on machine learning*, pp. 2160–2169. PMLR, 2019.

AI Team Genesis. The role of simulation in scalable robotics, genesis world 1.0, and the path forward. *Genesis AI Blog*, May 2026.

Mingrong Gong, Zhengkun Yi, Yidong Chen, Huiyun Li, and Yunduan Cui. Reducing the value function over-estimation by kullback-leibler divergence regularized distributional actor-critic. *Applied Intelligence*, 55, 2025.

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

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Hisham Husain, Kamil Ciosek, and Ryota Tomioka. Regularized policies are reward robust. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 64–72. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/husain21a.html.

Marco Jiralerspong, Esther Derman, Danilo Vucetic, Nikolay Malkin, Bilun Sun, Tianyu Zhang, Pierre-Luc Bacon, and Gauthier Gidel. Robust reinforcement learning for discrete compositional generation via general soft operators. *arXiv preprint arXiv:2506.17007*, 2025.

Donghu Kim, Youngdo Lee, Minho Park, Kinam Kim, I Made Aswin Nahendra, Takuma Seno, 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. *arXiv preprint arXiv:2604.04539*, 2026.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: removing the need for a target network in deep q-learning. In *Proceedings of the twenty eighth international joint conference on artificial intelligence*, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations, ICLR*, 2014. URL https://arxiv.org/abs/1312.6114.

Kyungjae Lee, Sungyub Kim, Sungbin Lim, Sungjoon Choi, Mineui Hong, Jae In Kim, Yong-Lae Park, and Songhwai Oh. Generalized tsallis entropy reinforcement learning and its application to soft mobile robots. In *Robotics: science and systems*, volume 16, pp. 1–10, 2020.

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 *International Conference on Machine Learning*. PMLR, 2023.

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, 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, 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. doi: 10.48550/arXiv.2511.04831. URL https://arxiv.org/abs/2511.04831.

James Mock and Suresh Muknahallipatna. A comparison of ppo, td3 and sac reinforcement algorithms for quadruped walking gait generation. *Journal of Intelligent Learning Systems and Applications*, 15:36–56, 01 2023. doi: 10.4236/jilsa.2023.151003.

Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1810–1816. IEEE, 2021.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

Gianluca Sabatini, Chenhao Li, and Marco Hutter. Bridging the gap: Enabling soft actor critic for high performance legged locomotion. *arXiv preprint arXiv:2605.24975*, 2026.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.

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

Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

Younggyo Seo, Carmelo Sferrazza, Juyue Chen, Guanya Shi, Rocky Duan, and Pieter Abbeel. Learning sim-to-real humanoid locomotion in 15 minutes. *arXiv preprint arXiv:2512.01996*, 2025a.

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*, 2025b.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Ruipeng Zhang, Ya-Chien Chang, and Sicun Gao. When maximum entropy misleads policy optimization. In *Proceedings of the 42nd International Conference on Machine Learning, ICML 2025, Vancouver, Canada*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025.

A Related work continued

Alternatively regularised actor-critic algorithms Gong et al. (2025) proposes a KL-regularised actor-critic algorithm that penalises deviation from the previous policy iterate. Similarly to Kim et al. (2019), they find that KL-regularised actor-critic is able to mitigate overestimation bias without the need for two Q-networks. Lee et al. (2020) proposes a Tsallis entropy-regularised actor-critic algorithm, demonstrating improved performance over SAC in a number of MuJoCo environments and on a real mobile robot (Todorov et al., 2012).

B Derivation of SMM-AC

$$\begin{aligned}\operatorname{softmax}((\alpha+\omega)Q(s))(a) &= \frac{\exp(\alpha Q(s,a))\exp(\omega Q(s,a))}{\int_{\mathcal{A}}\exp(\alpha Q(s,b))\exp(\omega Q(s,b))db} \\ &= \frac{\exp(\alpha Q(s,a))\exp(\omega Q(s,a))}{\int_{\mathcal{A}}\exp(\alpha Q(s,b))\exp(\omega Q(s,b))db}\frac{\int_{\mathcal{A}}\exp(\alpha Q(s,c))dc}{\int_{\mathcal{A}}\exp(\alpha Q(s,c))dc} \\ &= \frac{\pi^{\text{ref}}(a|s)\exp(\omega Q(s,a))}{\int_{\mathcal{A}}\pi^{\text{ref}}(b|s)\exp(\omega Q(s,b))db}\end{aligned}$$

Since the denominator does not depend on the action $a \sim \pi_z^{\text{act}}$, it can be discarded from the loss function since its gradient will be zero.

$$\begin{aligned}J_{\text{act}}(z) &= \mathop{\mathbb{E}}_{s\sim\mathcal{D}}\left[D_{KL}\left(\pi_z^{\text{act}}(\cdot|s)\parallel \operatorname{softmax}((\alpha+\omega)Q_\theta(s,\cdot))\right)\right] \\ &= \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D}\\ a\sim\pi_z^{\text{act}}(\cdot|s)}}\left[\log(\pi_z^{\text{act}}(a|s)) - \log(\pi^{\text{ref}}(a|s)) - \omega Q_\theta(s,a) + \log\int_{\mathcal{A}}\pi^{\text{ref}}(b|s)\exp(\omega Q_\theta(s,b))\,db\right]\end{aligned}$$

Then, we can simply drop the last term in the expectation, and convert the π^{ref} to the parameterised reference policy π_ϕ^{ref}.

C Additional details on the MuJoCo experiments

Table 3 lists the hyperparameters that we swept over for each algorithm. We tried to keep the number of experiments per algorithm roughly the same, but struggled to find useful hyperparameters to sweep over for PPO. Thus, we run 18 experiments per environment for PPO, and 54 experiments per environment for the rest. The "Freq" hyperparameter refers to the update frequency of the policy. It has been well-understood that delaying policy updates can improve the stability of learning in actor-critic methods (Fujimoto et al., 2018). We sweep the policy learning rates for all methods, but for SMM-AC we only sweep the reference policy's learning rate. For SMM-AC we use a cosine annealing learning rate scheduler, decreasing the learning rate by 10x by the end of training, for improved reference policy stability. This presents a limitation in the hyperparameter sweep, since SMM-AC has too many hyperparameters to fully sweep for optimal performance. As such, SMM-AC's policy and Q-function learning rates were fixed at 3e-4 and 1e-3 respectively. For SMM-AC we swept the ω hyperparameter, while fixing $\alpha = 1$. For the Pusher-v5 and Swimmer-v4 tasks, we found that setting $\omega = 400$ achieved the best performance, we consequently report that result. We used SAC's temperature autotuning instead of sweeping the hyperparameter.

The best-performing hyperparameters from the above sweep are listed in Table 4.

Table 3: Hyperparameter search spaces for MuJoCo experiments. *For Pusher-v5 and Swimmer-v4, we additionally evaluated $\omega = 400$ to increase greediness, which was crucial for matching SAC performance. Since such high inverse temperatures were not part of the general sweep, the total number of experiments per method remains comparable.

Algorithm	**Hyperparameter search space**
PPO	$\mathrm{LR} \in \{1\mathrm{e}{-5}, 3\mathrm{e}{-5}, 8\mathrm{e}{-5}, 1\mathrm{e}{-4}, 3\mathrm{e}{-4}, 8\mathrm{e}{-4}\}$ $\mathrm{EntCoef} \in \{0, 0.01, 0.1\}$
SAC	$\mathrm{PolicyLR} \in \{1\mathrm{e}{-5}, 3\mathrm{e}{-5}, 8\mathrm{e}{-5}, 1\mathrm{e}{-4}, 3\mathrm{e}{-4}, 8\mathrm{e}{-4}\}$ $\mathrm{QLR} \in \{5\mathrm{e}{-4}, 1\mathrm{e}{-3}, 5\mathrm{e}{-3}\}$ $\mathrm{Freq} \in \{2, 4, 6\}$
TD3	$\mathrm{LR} \in \{1\mathrm{e}{-5}, 3\mathrm{e}{-5}, 8\mathrm{e}{-5}, 1\mathrm{e}{-4}, 3\mathrm{e}{-4}, 8\mathrm{e}{-4}\}$ $\mathrm{Freq} \in \{2, 4, 6\}$ $\mathrm{Noise} \in \{0.1, 0.2, 0.3\}$
SMM-AC	$\mathrm{RefLR} \in \{1\mathrm{e}{-6}, 3\mathrm{e}{-6}, 8\mathrm{e}{-6}, 1\mathrm{e}{-5}, 3\mathrm{e}{-5}, 8\mathrm{e}{-5}\}$ $\mathrm{Freq} \in \{2, 4, 6\}$ $\omega \in \{5, 10, 15, 400^*\}$

Table 4: Best hyperparameters for each environment, selected based on final performance. Each entry reports the learning rate and the corresponding algorithm-specific hyperparameter.

Environment	**PPO** LR / Ent.	**SAC** Pol. LR / Freq.	**SMM-AC** Ref. LR / ω	**TD3** LR / Noise
Ant-v4	3e−4 / 0.0	3e−4 / 2.0	1e−6 / 15.0	1e−4 / 0.3
HalfCheetah-v4	1e−4 / 0.01	3e−4 / 2.0	8e−5 / 15.0	8e−4 / 0.3
Hopper-v4	3e−4 / 0.0	3e−5 / 2.0	3e−6 / 15.0	8e−4 / 0.2
Humanoid-v4	3e−5 / 0.0	1e−4 / 2.0	8e−5 / 10.0	3e−4 / 0.1
Inverted Double Pendulum	1e−4 / 0.0	3e−5 / 6.0	8e−5 / 15.0	1e−4 / 0.1
Pusher-v5	8e−4 / 0.0	1e−4 / 6.0	3e−5 / 400.0	8e−5 / 0.2
Swimmer-v4	3e−4 / 0.01	8e−4 / 6.0	1e−5 / 400.0	8e−5 / 0.3
Walker2d-v4	3e−4 / 0.01	8e−4 / 6.0	3e−5 / 15.0	8e−4 / 0.1

D ADDITIONAL MUJOCO RESULTS WITH FLASH AND FAST ALGORITHMS

Initial implementation and testing of the SAC, SMM-AC, TD3, and PPO algorithms were done in CleanRL (Huang et al., 2022). Since then, we have transitioned over to the FlashSAC codebase for both SAC and SMM-AC (Kim et al., 2026). We have similarly replaced TD3 with FastTD3 (Seo et al., 2025b). Given these new codebases, we felt it necessary to re-run the MuJoCo experiments to again demonstrate that FlashSMM-AC was competitive with FlashSAC and FastTD3. Figure 5 illustrates the performance of FlashSAC, FlashSMM-AC, and FastTD3 on five MuJoCo tasks. For FlashSAC and FlashSMM-AC, we sweep the temperature hyperparameter, selecting the best-performing (at the end of training) setting for the figure. We also show FlashSAC-tune, which uses temperature autotuning to learn the temperature during training. It is clear that FlashSMM-AC and FlashSAC perform similarly, with autotuning often outperforming static temperatures. **These results again support the idea that autotuning is important for achieving maximal average returns.** In most environments, FastTD3 performs competitively with FlashSAC and FlashSMM-AC, but suffers from far greater instability in most environments (*i.e.*, the average return curve is more jagged).

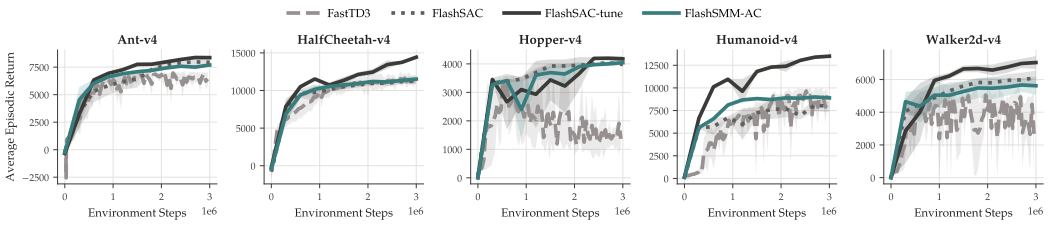

Figure 5: Average returns in MuJoCo tasks against environment steps. Each algorithm is run five times, for three million steps, with the shaded areas representing the standard deviation, and the solid line being the mean of the average return curves.

Table 5 lists the best-performing hyperparameters for FlashSAC and FlashSMM-AC in each task. Note how the inverse temperatures here are significantly higher than those that were used in the CleanRL implementation. This indicates that the FlashSAC algorithm is capable of converging even when the optimal distribution is quite peaky, due to the high inverse temperature. Moreover, this indicates that FlashSAC and FlashSMM-AC are stable, and capable of performing well in these high inverse temperature regimes. To find these optimal inverse temperatures, we swept $\omega \in \{15, 66, 100, 200, 400\}$, with 15 commonly being the best inverse temperature in our prior MuJoCo tests, see Table 4. Note also that we didn't need learning rate annealing techniques to stabilise learning as we did in the CleanRL MuJoCo experiments. This indicates that FlashSMM-AC is significantly more stable.

Table 5: Best-performing inverse temperature hyperparameters for FlashSAC and FlashSMM-AC.

Environment	FlashSAC w	FlashSMM-AC w
Ant-v4	400	400
HalfCheetah-v4	400	400
Hopper-v4	400	400
Humanoid-v4	200	400
Walker2d-v4	400	400

E LIMITING DOMAIN RANDOMISATION IN ISAACLAB TASKS

This experiment is designed to test how each algorithm's simulated and real-world performance changes with gradual limitations to DR in training. We hypothesize that regularised algorithms, given their robustness guarantees, would perform better than unregularised algorithms like PPO and TD3 under decreasing DR. Note that IsaacLab cannot support multiple environments running at

the same time, so we could not evaluate the DR-limited models throughout the course of training on non-limited environments. This means that training-time evaluation curves showing the average return are doing so in the DR-limited environments.

E.1 Limiting domain randomization in Isaac-Reach-Franka-v0

We test three settings of DR, with index one in Table 6 representing the normal DR ranges for this task from the IsaacLab configuration files.[3] We gradually limit the boxes for both the arm's randomly selected target position and its randomly selected starting position. By index 3, the target positions are limited to a small box, the yaw target barely changes the wrist position of the robot, and the reset position barely changes.

Table 6: Command (target position of the arm, first 4 columns) and reset-randomization (where the arm can start, last column) ranges swept for Isaac-Franka-Reach-v0. Each index corresponds to one of the three experiments.

Index	x (m)	y (m)	z (m)	yaw (rad)	Joint pos. scale
DR 1 (Default)	$[0.35, 0.65]$	$[-0.20, 0.20]$	$[0.15, 0.50]$	$[-3.14, 3.14]$	$[0.50, 1.50]$
DR 2 (Limited)	$[0.45, 0.65]$	$[-0.10, 0.20]$	$[0.25, 0.50]$	$[-1.50, 1.50]$	$[0.50, 1.00]$
DR 3 (Severe)	$[0.55, 0.65]$	$[0.00, 0.20]$	$[0.35, 0.50]$	$[-0.50, 0.50]$	$[0.50, 0.75]$

Table 7 lists the evaluation performance of DR limited agents on the non-limited task. As the table shows, performance drastically drops as the DR index increases, indicating that domain randomisation improves task generalisation.

Table 7: Reach: mean $\pm$ standard deviation return over full episodes (pooled over 5 to 10 seeds and 32 envs per cell), where all models are evaluated on the fully randomised environments, DR 1.

Agent	ω	DR 1 (Default)	DR 2 (Limited)	DR 3 (Severe)
SAC	400	0.52 ± 0.26	-0.05 ± 0.53	-1.18 ± 0.63
	800	0.52 ± 0.25	-0.18 ± 0.53	-1.31 ± 0.61
	1200	0.56 ± 0.23	0.04 ± 0.42	-1.46 ± 0.74
SMM-AC	400	0.52 ± 0.23	0.17 ± 0.23	-1.64 ± 0.71
	800	0.52 ± 0.25	0.03 ± 0.36	-1.49 ± 0.73
	1200	0.53 ± 0.23	0.08 ± 0.24	-1.57 ± 0.75
FastTD3	N/A	0.47 ± 0.26	-0.26 ± 0.48	-1.69 ± 0.73
PPO	N/A	0.62 ± 0.22	0.13 ± 0.31	-1.40 ± 0.61

E.2 Limiting domain randomisation in Isaac-Lift-Cube-Franka-v0

We again evaluate three DR limitations in Franka Lift, with similar restrictions to Reach, where the first DR index in Table 8 is the default randomisation from the IsaacLab configuration files.[4] We gradually limit the boxes for both the arm's randomly selected target position and the randomly selected starting position of the cube. By index 3, the target positions are limited to a small box and the object reset position barely changes.

E.3 Limiting domain randomisation in Isaac ANYmal-D tasks

We evaluate three distinct DR levels, where Level 1 corresponds to the full DR distribution specified in the baseline IsaacLab environment configurations, with added randomisation over friction.[5] We restrict physical dynamics and external disturbance ranges by linearly interpolating parameters from

[3] Franka Reach configurations available here.
[4] Lift Cube configurations available here
[5] ANYmal-D configurations available here.

Table 8: Command (target position of the arm, first 3 columns) and reset-randomization (where the object can start, last two columns) ranges swept for Isaac-Lift-Cube-Franka-v0. Each index corresponds to one of the three experiments.

Index	target x (m)	target y (m)	target z (m)	object x (m)	object y (m)
DR 1 (Default)	$[0.4, 0.6]$	$[-0.25, 0.25]$	$[0.25, 0.50]$	$[-0.1, 0.1]$	$[-0.25, 0.25]$
DR 2 (Limited)	$[0.475, 0.6]$	$[-0.1, 0.25]$	$[0.35, 0.50]$	$[-0.025, 0.1]$	$[-0.05, 0.25]$
DR 3 (Severe)	$[0.55, 0.6]$	$[0.05, 0.25]$	$[0.45, 0.50]$	$[0.05, 0.1]$	$[0.15, 0.25]$

Table 9: Lift-Cube: mean $\pm$ standard deviation return over full episodes (pooled over 5 to 10 seeds and 32 envs per cell), where all models are evaluated on the fully randomised environment, DR 1

Agent	ω	DR 1	DR 2	DR 3
SAC	400	139.17 ± 37.56	59.23 ± 73.42	62.20 ± 65.76
	800	94.20 ± 76.37	90.80 ± 74.79	64.21 ± 67.76
	1200	157.67 ± 2.57	127.06 ± 50.64	47.21 ± 63.69
SMM-AC	400	115.00 ± 65.88	111.52 ± 64.29	52.96 ± 63.27
	800	152.56 ± 3.44	108.16 ± 63.19	73.16 ± 62.76
	1200	153.05 ± 4.88	107.34 ± 66.51	39.61 ± 58.51
FastTD3	N/A	140.47 ± 6.94	125.09 ± 23.66	69.18 ± 55.33
PPO	N/A	136.75 ± 45.60	92.26 ± 64.24	22.76 ± 31.43

full randomization at Level 1 down to nominal, zero-perturbation conditions at Level 3. Specifically, we contract the bounds for surface friction (static and dynamic coefficients), added base mass, and external linear push velocities applied to the robot's base. By Level 3, the friction coefficients are fixed at their nominal values, the added-mass perturbation is fixed to zero, and the periodic push disturbance is completely disabled.

Independently, we scale the magnitude of additive uniform observation noise applied to the policy's proprioceptive inputs — base linear velocity, base angular velocity, projected gravity, joint position, and joint velocity — halving each noise bound at every successive DR level, so that Level 3 noise is one quarter of its full, Level-1 magnitude.

We further restrict the distribution of initial states sampled at each episode reset using the same linear-interpolation scheme as the dynamics terms, contracting the robot's initial base position, heading, and linear/angular velocity, together with the initial joint-position scale, toward their nominal, non-randomized values. By Level 3, every episode begins from an identical, fully deterministic nominal state, eliminating reset-state diversity entirely so as to induce a pronounced train/eval distribution shift when the resulting policy is evaluated under full randomization.

The exact parameter ranges swept across each level are summarized in Table 10.

F Additional hardware results

Table 13 extends the *none* rows of Table 2 with the task error of each configuration. The errors follow the returns, and the configuration with the highest return on a task also has the lowest error on Reach, Lift, and both ANYmal tasks.

F.1 Hardware deployment with limited domain randomisation

In App. E, we train policies with progressively narrower DR (DR 2 and DR 3) and evaluate them in simulation under the default ranges (DR 1). Here we deploy the same checkpoints on hardware without any perturbation, as in Subsec. 5.2. This can be viewed as another way of increasing the sim-to-real gap, complementing the experiments in Subsec. 5.3. Before deployment, each checkpoint goes through the simulation safety check described in Subsec. 5.1. This check rules out PPO at DR

Table 10: Anymal-D domain-randomization (DR) limits by level. Observation noise ranges are additive uniform noise on the policy input; reset ranges control the initial state distribution at episode start.

Quantity	DR 1 (Default)	DR 2 (Limited)	DR 3 (Severe)
Observation noise (additive uniform)			
Base linear velocity	$\pm$0.10 m/s	$\pm$0.05 m/s	$\pm$0.025 m/s
Base angular velocity	$\pm$0.20 rad/s	$\pm$0.10 rad/s	$\pm$0.05 rad/s
Projected gravity	$\pm$0.05	$\pm$0.025	$\pm$0.0125
Joint position	$\pm$0.01 rad	$\pm$0.005 rad	$\pm$0.0025 rad
Joint velocity	$\pm$1.5 rad/s	$\pm$0.75 rad/s	$\pm$0.375 rad/s
Initial state (reset) randomization			
Base position (x, y)	$\pm$0.50 m	$\pm$0.25 m	0 m (fixed)
Base heading (yaw)	$\pm\pi$ rad	$\pm\pi/2$ rad	0 rad (fixed)
Base velocity (all 6 DOF)	$\pm$0.50 (m/s, rad/s)	$\pm$0.25 (m/s, rad/s)	0 (fixed)
Joint position scale ($\times$ default)	$\times[0.5, 1.5]$	$\times[0.75, 1.25]$	$\times$1.0 (fixed)
Physics-event randomization			
Static friction coefficient	$[0.20, 1.25]$	$[0.50, 1.025]$	0.8 (fixed)
Dynamic friction coefficient	$[0.10, 1.00]$	$[0.35, 0.80]$	0.6 (fixed)
Added base mass multiplier	$[0.80, 1.25]$	$[0.90, 1.125]$	1.0 (fixed)
Push-robot velocity kick	$\pm$0.50 m/s	$\pm$0.25 m/s	0 (fixed)

Table 11: AnymalD-Flat: mean $\pm$ standard deviation return over full episodes (pooled over 5 to 10 seeds and 32 envs per cell), where all models are evaluated on the fully randomised environments, DR 1.

Agent	ω	DR 1	DR 2	DR 3
SAC	400	24.75 $\pm$ 1.67	19.87 $\pm$ 8.58	9.48 $\pm$ 13.51
	800	25.35 $\pm$ 2.09	19.54 $\pm$ 10.17	4.93 $\pm$ 16.92
	1200	25.29 $\pm$ 2.82	17.48 $\pm$ 12.99	-0.17 $\pm$ 22.83
SMM-AC	400	24.60 $\pm$ 2.31	20.90 $\pm$ 7.93	9.53 $\pm$ 12.07
	800	25.64 $\pm$ 1.25	19.02 $\pm$ 9.96	8.98 $\pm$ 12.15
	1200	25.83 $\pm$ 0.94	18.30 $\pm$ 11.06	3.07 $\pm$ 19.64
FastTD3	N/A	25.36 $\pm$ 2.56	18.37 $\pm$ 13.47	1.78 $\pm$ 21.37
PPO	N/A	21.73 $\pm$ 6.42	10.35 $\pm$ 16.57	-1.51 $\pm$ 22.86

2 and PPO and FastTD3 at DR 3 on both ANYmal tasks, as well as every configuration at DR 3 on flat terrain. Table 14 reports the returns, and its DR 1 column repeats the *none* rows of Table 2.

Limiting domain randomisation lowers the return on hardware. Every configuration deployed at DR 3 achieves a lower return than at DR 1. On Reach, every DR 3 configuration obtains a negative return, and on flat terrain, no DR 3 configuration passes the safety check at all. The effect of DR 2 depends on the task. On Reach and flat terrain, every deployed configuration already loses return at DR 2, and on Lift all but one do. On rough terrain, the return at DR 2 stays close to DR 1, and several configurations score slightly higher, in line with the simulation results in Table 12.

SMM-AC has the highest return on Lift and in locomotion. On Lift and both ANYmal tasks, the highest return at every deployed DR level comes from a SMM-AC configuration. Note, however, that the seed-to-seed variation under limited DR is large, and most differences between SAC and SMM-AC at the same ω are smaller than their combined standard deviations.

Discussion. Limiting DR costs every method return on hardware, so regularisation does not replace it. Under limited randomisation, however, the regularised policies still pass the safety check on ANYmal, which the unregularised baselines struggle more.

Table 12: AnymalD-Rough: mean ± standard deviation return over full episodes (pooled over 5 to 10 seeds and 32 envs per cell), where all models are evaluated on the fully randomised environments, DR 1.

Agent	ω	DR 1	DR 2	DR 3
SAC	400	18.04 ± 6.75	20.85 ± 5.80	18.59 ± 7.39
	800	21.13 ± 4.56	19.96 ± 6.26	18.48 ± 7.42
	1200	19.61 ± 4.83	20.54 ± 5.55	17.76 ± 8.88
SMM-AC	400	17.91 ± 7.21	20.18 ± 6.51	18.19 ± 7.08
	800	20.57 ± 4.87	20.21 ± 5.75	17.71 ± 8.32
	1200	19.96 ± 3.88	20.71 ± 4.91	17.45 ± 8.39
FastTD3	N/A	10.91 ± 5.45	7.26 ± 7.17	5.70 ± 8.40
PPO	N/A	12.90 ± 7.14	14.52 ± 7.15	13.17 ± 11.22

Table 13: Hardware deployment without perturbation: task error and return, mean and one standard deviation over seeds. Each column is one trained configuration. On Franka, the error is the distance between the end effector (Reach) or the cube (Lift) and the target at the end of the episode. On ANYmal, it is the RMSE between commanded and measured base velocity. PPO and FastTD3 are shown at their best configuration per task. Highlighted entries indicate the best mean within each task metric.

			SAC			SMM-AC (ours)		
Task	PPO	FastTD3	ω=400	800	1200	ω=400	800	1200
Franka Reach								
Error (cm)	0.64 ±0.11	0.93 ±0.14	0.93 ±0.16	0.79 ±0.12	0.91 ±0.15	1.00 ±0.15	0.76 ±0.10	1.07 ±0.17
Return	0.69 ±0.04	0.40 ±0.09	0.48 ±0.12	0.58 ±0.09	0.53 ±0.09	0.51 ±0.08	0.61 ±0.03	0.45 ±0.05
Franka Lift								
Error (cm)	5.79 ±3.6	4.26 ±0.6	16.53 ±9.4	8.54 ±7.1	17.32 ±9.7	1.53 ±0.6	0.82 ±0.2	8.57 ±6.6
Return	133.10 ±54.75	123.14 ±4.05	92.90 ±83.80	125.78 ±69.93	94.93 ±86.04	148.48 ±8.61	152.87 ±2.52	123.94 ±69.39
ANYmal flat								
RMSE (m/s)	0.2289 ±0.047	0.1955 ±0.018	0.1439 ±0.012	0.1369 ±0.010	0.1355 ±0.009	0.1509 ±0.011	0.1268 ±0.008	0.1523 ±0.012
Return	18.94 ±5.55	23.60 ±0.70	25.28 ±0.28	25.18 ±0.45	24.77 ±0.09	24.98 ±0.42	25.89 ±0.40	25.51 ±0.30
ANYmal rough								
RMSE (m/s)	0.5775 ±0.081	0.5295 ±0.073	0.3689 ±0.044	0.3471 ±0.057	0.3386 ±0.049	0.4112 ±0.045	0.3252 ±0.038	0.2938 ±0.034
Return	8.60 ±3.66	13.72 ±4.13	19.11 ±1.62	19.33 ±4.90	18.78 ±3.40	20.96 ±1.18	20.82 ±1.32	21.93 ±1.60

Table 14: Hardware return of policies trained with limited DR, deployed without perturbation. Each column is one training configuration, and reading down a column within a task shows how its hardware return changes with the training-time DR level of App. E. The *DR 1* row corresponds to the *none* rows of Table 2. Entries are the mean and one standard deviation of the return across seeds. Highlighted entries indicate the highest return in each column and task. Entries marked with "—" were not deployed, because the configuration failed the simulation safety check (Subsec. 5.1).

			SAC			SMM-AC (ours)		
DR level	PPO	FastTD3	ω=400	800	1200	ω=400	800	1200
Franka Reach								
DR 1 (Default)	0.69 ±0.04	0.40 ±0.09	0.48 ±0.12	0.58 ±0.09	0.53 ±0.09	0.51 ±0.08	0.61 ±0.03	0.45 ±0.05
DR 2 (Limited)	0.24 ±0.08	-0.16 ±0.11	-0.01 ±0.14	-0.11 ±0.13	0.08 ±0.11	0.21 ±0.07	0.12 ±0.09	0.16 ±0.06
DR 3 (Severe)	-1.18 ±0.13	-1.42 ±0.18	-1.03 ±0.16	-1.15 ±0.15	-1.27 ±0.19	-1.38 ±0.14	-1.29 ±0.16	-1.34 ±0.17
Franka Lift								
DR 1 (Default)	133.1 ±54.8	123.1 ±4.1	92.9 ±83.8	125.8 ±69.9	94.9 ±86.0	148.5 ±8.6	152.9 ±2.5	123.9 ±69.4
DR 2 (Limited)	98.4 ±61.4	117.8 ±19.3	47.6 ±67.6	112.4 ±62.2	106.8 ±53.3	136.3 ±24.6	118.5 ±49.8	93.8 ±71.2
DR 3 (Severe)	16.8 ±31.5	58.4 ±48.6	64.2 ±63.8	71.6 ±66.0	37.4 ±56.9	61.9 ±59.4	81.2 ±58.1	31.7 ±49.4
ANYmal flat								
DR 1 (Default)	18.9 ±5.6	23.6 ±0.7	25.3 ±0.3	25.2 ±0.5	24.8 ±0.1	25.0 ±0.4	25.9 ±0.4	25.5 ±0.3
DR 2 (Limited)	—	16.4 ±1.1	20.8 ±2.0	11.4 ±4.9	17.6 ±3.4	22.2 ±1.2	20.5 ±1.8	19.2 ±2.1
DR 3 (Severe)	—	—	—	—	—	—	—	—
ANYmal rough								
DR 1 (Default)	8.6 ±3.7	13.7 ±4.1	19.1 ±1.6	19.3 ±4.9	18.8 ±3.4	21.0 ±1.2	20.8 ±1.3	21.9 ±1.6
DR 2 (Limited)	—	—	21.4 ±1.9	18.6 ±3.8	19.8 ±2.5	19.7 ±2.1	21.2 ±1.5	22.0 ±1.5
DR 3 (Severe)	—	—	17.7 ±2.9	16.9 ±4.1	15.8 ±4.6	18.3 ±2.6	17.4 ±3.3	18.1 ±3.1