Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

FLASHSAC: Fast and Stable Off-Policy Reinforcement Learning for High-Dimensional Robot Control

Donghu Kim[1,2*], Youngdo Lee[2,3*], Minho Park[2], Kinam Kim[2], Takuma Seno[4], I Made Aswin Nahrendra[3], Sehee Min[1], Daniel Palenicek[5,6], Florian Vogt[7], Danica Kragic[7], Jan Peters[5,6,8,9], Jaegul Choo[2], Hojoon Lee[1,2]

[1]Holiday Robotics [2]KAIST [3]KRAFTON [4]Turing Inc [5]TU Darmstadt [6]hessian.AI
[7]KTH Royal Institute of Technology [8]German Research Center for AI (DFKI) [9]Robotics Institute Germany (RIG)
* equal contribution Correspondence to: Hojoon Lee <hojoon.lee@holiday-robotics.com>

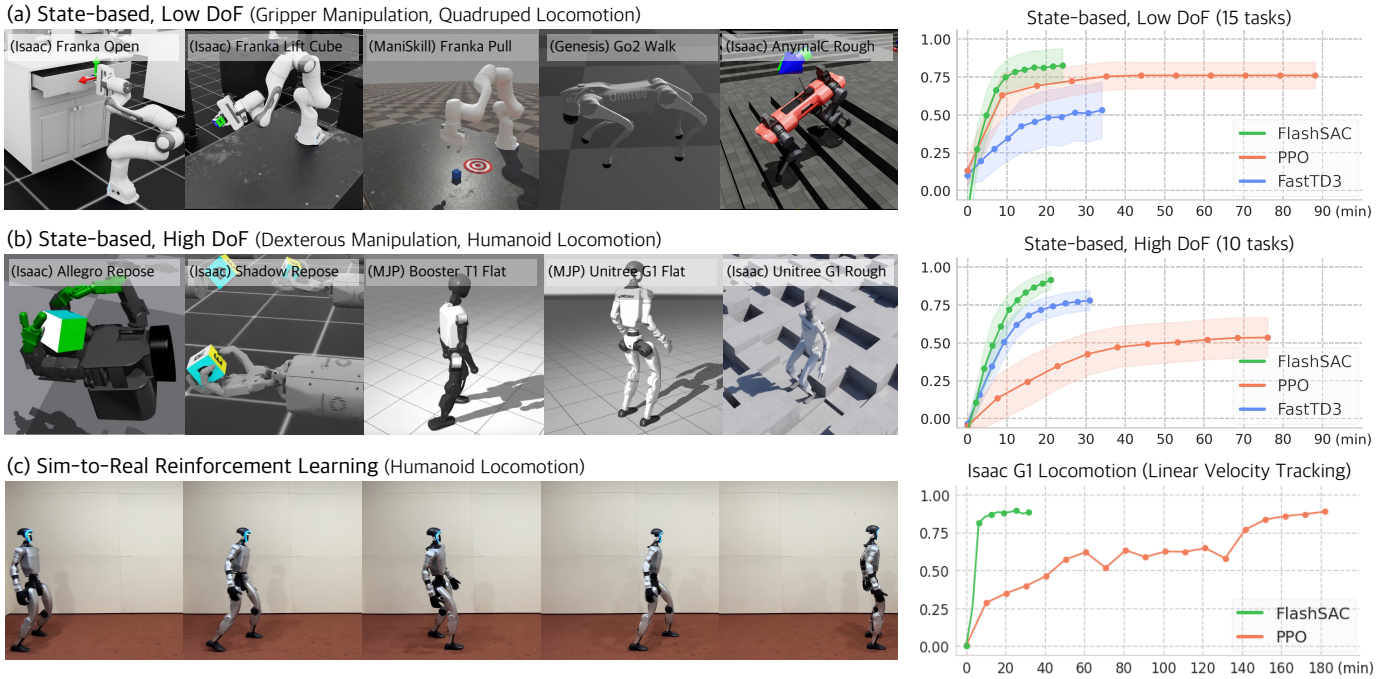

Fig. 1: **Results Overview.** Tasks grouped by state–action dimensionality, with representative examples shown for each category. **(a) State-based, Low DoF:** Gripper manipulation and quadruped locomotion tasks from IsaacLab, ManiSkill, and Genesis. In low-dimensional settings, FLASHSAC achieves performance comparable to PPO. **(b) State-based, High DoF:** Dexterous manipulation and humanoid locomotion tasks from IsaacLab and MuJoCo Playground. In high-dimensional settings, FLASHSAC substantially outperforms PPO in both asymptotic return and wall-clock efficiency. **(c) Sim-to-Real:** Humanoid locomotion on the Unitree G1 platform. FLASHSAC enables sim-to-real transfer within minutes, whereas PPO requires hours of training.

***Abstract*—Reinforcement learning (RL) is a core approach for robot control when expert demonstrations are unavailable. On-policy methods such as Proximal Policy Optimization (PPO) are widely used for their stability, but their reliance on narrowly distributed on-policy data limits accurate policy evaluation in high-dimensional state and action spaces. Off-policy methods can overcome this limitation by learning from a broader state-action distribution, yet suffer from slow convergence and instability, as fitting a value function over diverse data requires many gradient updates, causing critic errors to accumulate through bootstrapping. We present FLASHSAC, a fast and stable off-policy RL algorithm built on Soft Actor-Critic. Motivated by scaling laws observed in supervised learning, FLASHSAC sharply reduces gradient updates while compensating with larger models and higher data throughput. To maintain stability at increased scale, FLASHSAC explicitly bounds weight, feature and gradient norms, curbing critic error accumulation. Across over 60 tasks in 10 simulators, FLASHSAC consistently outperforms PPO and strong off-policy baselines in both final performance and training efficiency, with the largest gains on high-dimensional tasks such as dexterous manipulation. In sim-to-real humanoid locomotion, FLASHSAC reduces training time from hours to minutes, demonstrating the promise of off-policy RL for sim-to-real transfer.**

Project page: https://holiday-robot.github.io/FlashSAC
Public code: https://github.com/Holiday-Robot/FlashSAC

I. Introduction

The long-standing goal of robot learning is to develop agents that generalize across a wide range of tasks in the real world. While large-scale imitation learning from real-world data has recently yielded impressive results in robotic control [38, 97], reinforcement learning (RL) from simulation remains a core paradigm when expert demonstrations are unavailable, incomplete, or insufficient [39, 3, 114].

To date, sim-to-real RL has been most successful in relatively constrained domains such as quadruped locomotion [32, 81] and gripper-based manipulation [2], which are characterized by low-dimensional state–action spaces and extremely high-throughput simulators [95, 61, 4, 112]. In this regime, on-policy methods such as Proximal Policy Optimization (PPO) [85, 86] have proven effective: PPO is stable, easy to tune, and its data inefficiency is acceptable when fresh on-policy data can be collected cheaply.

However, this regime is becoming less representative of modern robot learning. Emerging applications—including humanoid locomotion [89], dexterous manipulation [11, 108], and vision-based control [66, 36]—involve much higher-dimensional state and action spaces, where policy evaluation and improvement from narrowly distributed on-policy data become substantially harder. Simultaneously, simulation grows increasingly expensive due to complex contact dynamics [52, 109], and larger policy architectures further raise rollout costs [38, 97]. In this setting, repeatedly discarding past experience in favor of freshly collected data becomes increasingly inefficient in both sample complexity and wall-clock time.

Off-policy RL offers a natural alternative. By reusing diverse experience from a replay buffer, off-policy methods can achieve substantially higher data efficiency than on-policy approaches [62, 51, 22]. This advantage is particularly appealing in high-dimensional robotic tasks, where broader data coverage can support better policy evaluation and improvement. Yet despite this promise, off-policy RL has not become the default choice for sim-to-real transfer, as it often suffers from slow training and instability [17, 47].

A central challenge is learning an accurate value function from broad replay data. Off-policy methods train a critic Q_θ by minimizing a bootstrapped Bellman objective,

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\theta(s,a) - (r + \gamma Q_\theta(s', a'))\right)^2\right], \quad (1)$$

where transitions (s, a, r, s') are sampled from a replay buffer $\mathcal{D}$ and $a' \sim \pi(\cdot \mid s')$. In high-dimensional settings, fitting this critic accurately over diverse replay data often requires many gradient updates, which not only increases training time but also compounds estimation errors through repeated bootstrapping, as the critic is optimized toward targets that depend on its own predictions [94, 101].

In this paper, we present FlashSAC, a fast and stable off-policy RL algorithm built on Soft Actor-Critic. Motivated by scaling trends in supervised learning, FlashSAC sharply reduces the number of gradient updates while compensating with larger models and higher data throughput, improving training efficiency and better matching modern large-scale simulation pipelines. However, larger critics can further exacerbate instability under bootstrapping. To maintain stability, FlashSAC explicitly controls critic update dynamics by bounding weight, feature, and gradient norms, thereby preventing the accumulation of critic errors.

We evaluate FlashSAC on more than 60 locomotion and manipulation tasks across 10 simulators, spanning high-dimensional state-based control, vision-based control, and sim-to-real humanoid locomotion. Across this benchmark suite, FlashSAC consistently outperforms PPO and strong off-policy baselines in both final performance and training efficiency, with the largest gains on the most challenging tasks such as dexterous manipulation and humanoid locomotion. In sim-to-real humanoid walking, FlashSAC reduces training time from hours to minutes while maintaining stable real-world deployment, demonstrating that off-policy RL can be both fast and stable for scalable sim-to-real robot learning.

II. Related Work

A. On-Policy Reinforcement Learning

On-policy RL has been the dominant paradigm for simulation-based robot learning when environment interaction is cheap and massively parallelizable. Among on-policy methods, PPO [85] is particularly popular for its stability, ease of implementation, and robustness to hyperparameter choices. Combined with modern high-throughput simulators [58, 95, 61, 4], PPO has enabled successful sim-to-real transfer in relatively constrained domains such as quadruped locomotion [32, 81, 86] and rigid-body manipulation [2]. Its widespread adoption has motivated a line of work improving upon it, such as stronger trust-region [110], exploration via expanding action space [50], and lower gradient variance via pathwise estimates [107].

However, on-policy methods fundamentally rely on freshly collected data and discard experience generated by earlier policies [94]. As task dimensionality increases, achieving sufficient state–action coverage via on-policy rollouts becomes increasingly expensive. While importance sampling can, in principle, correct for policy mismatch and enable data reuse [90, 57], importance weights in high-dimensional continuous action spaces exhibit extremely high variance, rendering this strategy impractical in modern robotic learning.

B. Off-Policy Reinforcement Learning

Off-policy RL decouples data collection from policy optimization by storing transitions in a replay buffer and reusing them across updates [94, 62]. This is especially appealing in high-dimensional robotic tasks, where diverse experience supports better policy evaluation than narrowly distributed on-policy rollouts.

Off-Policy Model-Based RL: Model-based RL further improves sample efficiency by learning environment dynamics and using them for planning or imagined rollouts [93, 34]. Recent approaches such as DreamerV3 [23] and TD-MPC2 [24] have demonstrated strong performance in vision-based domains by planning in learned latent spaces. However, learning accurate

dynamics models and performing repeated planning procedures significantly increases per-step training cost [56], often limiting their scalability in wall-clock critical settings.

Off-Policy Model-Free RL: Model-free off-policy algorithms such as DDPG [51], TD3 [17], and SAC [22] learn policies and value functions directly from replayed experience without explicit dynamics models. Their simplicity and data reuse make them attractive for robotic control. However, as discussed in Section I, off-policy model-free RL suffers from three persistent challenges: *slow training*, *unstable training dynamics*, and *exploration in high-dimensional action spaces*. The first two challenges stem from the bootstrapped Bellman objective illustrated in Equation 1: fitting a critic over diverse replay data in high-dimensional state-action spaces requires many gradient updates, directly increasing training time. Because critic targets depend on the critic's own predictions, approximation and extrapolation errors at poorly supported state-action pairs compound across updates [94, 101]. The third arises because the maximum-entropy formulation of SAC alone is often insufficient to maintain coherent exploration in high-dimensional action spaces [14, 29], motivating dedicated noise mechanisms and exploration schemes.

Prior work has primarily addressed each challenge in isolation. To improve *speed*, one line of work scales data throughput via parallel simulation and large replay buffers [48, 77, 78, 88, 87, 70]. For example, FastTD3 [88] and FastSAC [87] achieve strong wall-clock efficiency in humanoid locomotion but relies on small networks ($\sim$0.2M parameters), which limits its asymptotic performance. Scaling to larger networks is difficult in this setting, as increased model capacity exacerbates instability under bootstrapped training.

To improve *stability*, a second line of work constrains value-function sensitivity by bounding feature, weight, and gradient norms [8, 42, 20, 67, 44, 45, 55, 71, 72, 68, 15, 31, 103], or by other measures such as reinitialization [69, 91], distillation [105], ensembling [10, 27], and alternative critic target networks [106, 26]. These constraints limit error amplification under distribution shift and repeated bootstrapping, enabling training with larger networks that achieve higher asymptotic performance. However, the increased model capacity requires more gradient updates to converge, resulting in slower training in data-rich simulation regimes.

To improve *exploration*, a third line of work exploits off-policy RL's ability to decouple data collection from policy optimization, injecting temporally-correlated action noise to obtain coherent trajectories such as Ornstein–Uhlenbeck processes [29], pink noise [14], parameter-space noise [75], state-dependent exploration [80, 79], and temporally-extended action repetition [12]. However, exploration mechanisms alone leave the bottlenecks of slow and unstable training untouched, limiting their impact as model size and data throughput scale.

FLASHSAC unifies these directions: it achieves fast training by sharply reducing gradient updates while scaling model capacity and data throughput, maintains stable training dynamics by jointly bounding weight, feature, and gradient norms, and adopts a lightweight noise-repetition scheme that produces temporally-correlated exploration without per-environment overhead.

III. PRELIMINARY

In this section, we introduce the RL framework and algorithmic foundation upon which FLASHSAC is built.

A. Markov Decision Process (MDP)

We model robotic control as a discounted Markov Decision Process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the continuous action space, $P(s'|s,a)$ denotes the transition dynamics, $r(s,a)$ denotes the reward function, and $\gamma \in [0,1)$ is the discount factor.

At each timestep t, the agent observes $s_t \in \mathcal{S}$, samples an action $a_t \in \mathcal{A}$, receives a reward $r_t = r(s_t, a_t)$, and transitions to the next state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The goal is to learn a policy $\pi(a|s)$ that maximizes the discounted sum of rewards.

B. Soft Actor Critic (SAC)

FLASHSAC builds upon SAC [22], a widely used off-policy RL algorithm. SAC stores transitions (s, a, r, s') collected under past policies in a replay buffer $\mathcal{D}$, and trains the policy using samples drawn from this buffer.

Beyond maximizing expected return, SAC incorporates an entropy regularization term that encourages exploration. This entropy maximization is particularly important in high-dimensional state–action spaces, where insufficient exploration can lead to poor coverage of the replay buffer and exacerbate approximation and extrapolation errors.

To reduce approximation errors in bootstrapped value learning, SAC commonly employs clipped double Q-learning [17], maintaining two action-value functions $Q_{\phi_1}(s,a)$ and $Q_{\phi_2}(s,a)$. The minimum of the two estimates is used when forming targets, reducing the impact of optimistic value errors.

Concretely, the policy $\pi_\theta(a|s)$ is optimized by minimizing

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta}\big(\alpha \log \pi_\theta(a|s) - \min_{i=1,2} Q_{\phi_i}(s,a)\big), \quad (2)$$

where $\alpha > 0$ controls the relative importance of entropy.

Each critic is trained by minimizing a bootstrapped Bellman error using slowly updated target networks $\bar{\phi}_1$ and $\bar{\phi}_2$, which are updated via exponential moving average:

$$\bar{\phi}_j \leftarrow \tau\phi_j + (1-\tau)\bar{\phi}_j,\ j \in \{1,2\} \quad (3)$$

where $\tau \in (0,1)$ is the target update rate.

For $i \in \{1,2\}$, the critic weights ϕ_i are optimized by minimizing the Bellman loss

$$\mathcal{L}_Q(\phi_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[Q_{\phi_i}(s,a) - y\right]^2, \quad (4)$$

where the target value is

$$y = r + \gamma\big(\min_{j=1,2} Q_{\bar{\phi}_j}(s',a') - \alpha \log \pi_\theta(a'|s')\big), a' \sim \pi_\theta(\cdot|s'). \quad (5)$$

IV. FLASHSAC

FLASHSAC is a fast and stable off-policy RL algorithm for high-dimensional robotic control. It achieves strong asymptotic performance with fast wall-clock time through three complementary mechanisms: (i) fast training by scaling data and model while reducing gradient updates (§IV-A), (ii) stable training by constraining critic update dynamics (§IV-B), and (iii) broad exploration for diverse data coverage (§IV-C).

A. Fast Training

On-policy methods such as PPO discard all collected data after each iteration. In high-dimensional robotic tasks where simulation is expensive, this data inefficiency becomes a critical bottleneck. Off-policy RL reuses past experience from a replay buffer, but conventionally requires many gradient updates per transition to extract sufficient learning signal, which slows wall-clock time and compounds bootstrapping errors.

FLASHSAC takes a different approach inspired by the scaling trends observed in supervised learning: under a fixed compute budget, larger models trained with larger batches and fewer updates converge faster than smaller models with frequent updates [37]. This principle has been difficult to apply in off-policy RL, because increased model capacity tends to amplify critic instability under bootstrapping. FLASHSAC resolves this tension by stabilizing critic training through constrained update dynamics (§IV-B), enabling a regime of high data throughput, large models, and infrequent gradient updates.

1) Massively Parallel Simulation: We collect data using 1024 parallel simulation environments, enabling rapid accumulation of diverse trajectories. While many off-policy RL setups rely on a small number of environments [22, 17], high-throughput data collection is critical for maintaining adequate coverage of the state-action space in high-dimensional tasks.

2) Large-Capacity Replay Buffer: FLASHSAC uses a replay buffer of up to 10M transitions, an order of magnitude larger than the 1M commonly used in standard off-policy configurations [44, 72]. In high-dimensional tasks, rare but important state-action pairs can be easily overwritten in smaller buffers, leading to catastrophic forgetting and inducing extrapolation error. A larger buffer preserves such long-tail experiences and maintains the diversity of training data available to the critic throughout learning [16].

3) Large Model, Large Batch, Fewer Updates: Standard off-policy RL baselines use small MLPs (0.2-0.5M parameters, 2-3 layers) to avoid instability [22, 88]. In contrast, FLASHSAC employs a 2.5M-parameter, 6-layer network for both the actor and critic, paired with a batch size of 2048 that nearly saturates GPU utilization. The updates-to-data ratio is set to $2/1024$, meaning only 2 gradient updates are performed per 1024 new transitions. Although such infrequent updates are typically ineffective in off-policy RL, the combination of large batches, higher learning rates, and increased model capacity enables fast convergence with fewer updates.

4) Code Optimization: FLASHSAC is implemented in PyTorch [73], with both training and inference JIT-compiled to minimize Python overhead. We use mixed-precision throughout training [59], which reduces wall-clock time by 5-10%.

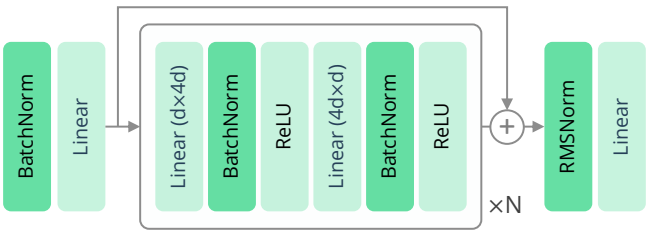

Fig. 2: **FLASHSAC Architecture.** The architecture consists of stacked inverted residual blocks with pre-activation batch normalization and post-RMS normalization.

B. Stable Training

Scaling data and model accelerates training but does not prevent instability arising from bootstrapped critic updates. In the Bellman backup, estimation errors at next-state action pairs propagate into the current Q-value targets and can be recursively amplified through repeated updates. This problem worsens with both state-action dimensionality and model capacity, making stability a prerequisite for scaling. FLASHSAC addresses this by constraining weight, feature, and gradient norms throughout training via the following mechanisms.

1) Inverted Residual Backbone: The backbone stacks inverted residual blocks inspired by the Transformer feedforward block [104] (Figure 2). Each block expands features to a higher dimension via an inverted bottleneck [30], projects back to the original dimension, and adds a residual connection [25] to stabilize gradient propagation. After the final block, we apply RMSNorm [113] to bound per-sample feature norms before value heads, preventing out-of-distribution inputs from producing unbounded activations that destabilize bootstrapping.

2) Pre-activation Batch Normalization: Replay data is collected by a mixture of evolving policies, inducing non-stationary input distributions. Without normalization, feature activations can saturate (e.g., dead ReLUs [1]), degrading gradient flow [13, 54, 43]. We apply batch normalization [33] before each nonlinearity to keep activations well-scaled. We choose batch normalization over layer normalization [5] because it exploits large-batch statistics from diverse replay data, yielding a smoother loss landscape with a lower effective condition number [83, 8, 72].

3) Cross-Batch Value Prediction: Batch normalization computes statistics per batch, so the predicted Q-values and target Q-values receive different normalization when computed in separate forward passes. Following [8], we concatenate current and next-state transitions into a single batch so that both share the same statistics, ensuring consistency in the Bellman update.

4) Distributional Critic with Adaptive Reward Scaling: Following [45, 72], we represent the Q-value as a categorical distribution over n_{atom} atoms uniformly spaced on $[G_{\min}, G_{\max}]$. The network predicts atom probabilities and is trained via cross-entropy loss against the projected Bellman target [7]. This

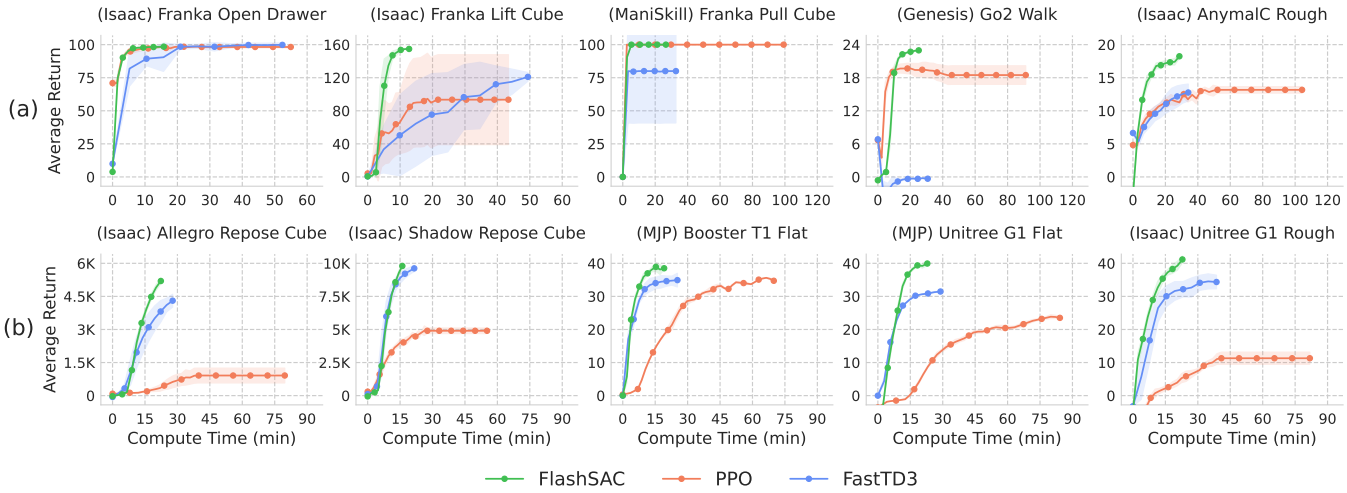

Fig. 3: **Results on State-Based RL, GPU-based Simulators.** Learning curves on select tasks from IsaacLab [61], ManiSkill [95], Genesis [4], and MuJoCo Playground [112]. We evaluate performance on **(a)** low-dimensional tasks with gripper manipulation and quadruped locomotion, and **(b)** high-dimensional tasks involving dexterous manipulation and humanoid locomotion. While comparable to PPO [86] in low-dimensional tasks, FLASHSAC significantly outperforms PPO in high-dimensional tasks.

distributional formulation smooths the optimization landscape and reduces sensitivity to noisy targets [72].

To keep returns within the distributional critic's fixed support, we normalize rewards directly rather than centering returns [65] or scaling losses [84]. We track the running discounted return variance $\sigma^2_{t,G}$ and maximum magnitude $G_{t,\max}$, and scale as:

$$\bar{r}_t = \frac{r_t}{\max\left(\sqrt{\sigma^2_{t,G} + \epsilon},\ G_{t,\max}/G_{\max}\right)}. \tag{6}$$

This bounds effective returns while maintaining a consistent scale throughout training.

5) Weight Normalization: Uncontrolled weight growth increases Q-value variance and amplifies estimation errors under bootstrapping [55]. After each gradient step, we project each weight vector onto the unit-norm sphere [102, 53, 45, 71, 72] and each normalization parameter vector (γ, β) to norm $\sqrt{d}$. This constrains the network to encode information through direction rather than scale.

C. Exploration

Off-policy RL can decouple data collection from policy optimization, allowing exploration strategies that pursue broad state-action coverage independently of the current policy. FLASHSAC employs two complementary mechanisms.

1) Unified Entropy Target: Maximum-entropy RL with automatic temperature tuning [22] encourages sustained exploration, but requires specifying a target entropy. Standard practice sets this target per task, which is impractical across embodiments with varying action dimensions. We instead parameterize the target entropy via a fixed action standard deviation σ_{tgt}. For a Gaussian policy with diagonal covariance, this gives:

$$\bar{\mathcal{H}} = \tfrac{1}{2}|\mathcal{A}| \log\left(2\pi e\, \sigma^2_{\text{tgt}}\right), \tag{7}$$

which scales linearly with action dimension, ensuring consistent exploration across embodiments without per-task tuning. We set $\sigma_{\text{tgt}} = 0.15$ in all experiments.

2) Noise Repetition: Temporally correlated action noise is commonly used to improve exploration in sparse-reward settings, with pink noise [14] and Ornstein–Uhlenbeck noise [29] being widely used. However, these methods are ill-suited to massively parallel simulations, as they require per-environment correlated-noise processes, which incur substantial computational and memory overhead.

We propose *Noise Repetition*, a lightweight alternative that induces temporal correlation using minimal local state. At each repetition interval, a noise vector $\epsilon \sim \mathcal{N}(0, I)$ is sampled for action selection and held constant for k consecutive steps. The repetition length k is drawn from a Zeta distribution with probability mass function $P(k) \propto k^{-s}$ [12], favoring short repeat intervals while occasionally producing long, correlated action sequences.

V. EXPERIMENTS

We evaluate FLASHSAC on a diverse suite of robotic control tasks, measuring both asymptotic performance and wall-clock time (measured on a single RTX 5090 GPU). Our experiments span low- and high-dimensional state-based control, vision-based control, and sim-to-real humanoid locomotion.

A. State-Based RL on GPU-based Simulators

Experimental Setup: We evaluate on 25 state-based control tasks drawn from four GPU-based simulators: IsaacLab [61], MuJoCo Playground [112], ManiSkill3 [95], and Genesis [4], all of which enable large-scale sample collection at minimal wall-clock cost.

The tasks span a wide range of state–action dimensionalities:

- Low-dim (15 tasks): Gripper-based manipulation (Franka) and quadruped locomotion (AnyMal-C/D, Unitree Go2).

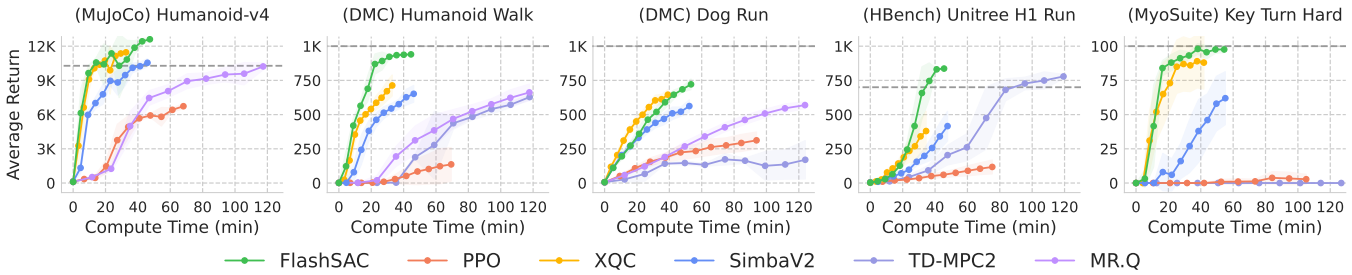

Fig. 4: **Results on State-Based RL, CPU-based Simulators.** Learning curves on select tasks from MuJoCo [99, 98], DMC [96], HumanoidBench [89] and MyoSuite [9]. We primarily evaluate high-dimensional tasks, involving dexterous manipulation and humanoid locomotion. FLASHSAC significantly outperforms PPO, as well as strong off-policy RL and model-based RL baselines in both compute efficiency and asymptotic performance.

- High-dim (10 tasks): Dexterous manipulation (Allegro, Shadow Hand) and humanoid locomotion (Unitree G1, H1, Booster T1).

A complete task list is provided in § X. We compare FLASHSAC against strong, widely adopted baselines:

- PPO [86]: A highly optimized on-policy implementation from RSL-RL, representative of current best practices in sim-to-real robotic RL.
- FastTD3 [88]: A wall-clock–optimized off-policy method designed for high throughput simulations.

Whenever available, we report published results; otherwise, we reproduce results using official implementations.

Off-policy methods (FLASHSAC and FastTD3) are trained for 50M environment steps. To probe asymptotic performance, PPO is trained for 200M steps, requiring approximately $3\times$ the compute of FLASHSAC. While baseline methods use task-specific hyperparameter tuning, FLASHSAC is evaluated using a single unified configuration across all tasks, varying only the discount factor γ to match simulator defaults (e.g., 0.99 for IsaacLab, 0.97 for Playground).

Experimental Results: Figure 3 summarizes performance on representative tasks, with full results in § XII.

On low-dimensional tasks, FLASHSAC slightly outperforms PPO (Figure 3.a). As consistent with prior findings, on-policy methods remain effective when state–action spaces are small, and simulation throughput is high enough to collect a large volume of samples.

On high-dimensional tasks, FLASHSAC demonstrates a clear and consistent advantage (Figure 3.b). Across dexterous manipulation and humanoid locomotion benchmarks, FLASHSAC converges reliably to higher asymptotic performance while requiring substantially less wall-clock time than PPO.

Compared to FastTD3, FLASHSAC is markedly more stable, converging across all tasks where FastTD3 frequently fails or underperforms (e.g., Go2Walk, Franka Pull Cube). When both methods converge, FLASHSAC achieves higher asymptotic performance, with the largest gains observed in humanoid locomotion, where larger model capacity is particularly beneficial.

B. State-Based RL on CPU-based Simulators

Experimental Setup: We further evaluate FLASHSAC on 40 single-environment, CPU-based continuous-control tasks drawn from four established benchmarks: MuJoCo [98], DeepMind Control Suite [96], MyoSuite [9], and HumanoidBench [89]. Unlike GPU-based simulators, these benchmarks use a single environment instance, placing greater emphasis on sample efficiency rather than wall-clock throughput.

We compare FLASHSAC against strong sample-efficient baselines:

- PPO [86]: A highly optimized on-policy implementation from RSL-RL, included to assess whether on-policy methods remain viable in the low-sample regime.
- XQC [72]: A recent off-policy method coupled with batch-normalization designed for high sample efficiency.
- SimbaV2 [45]: An improved variant of Simba [44] for stable off-policy learning.
- TD-MPC2 [24]: A model-based method that combines off-policy RL with model-predictive planning.
- MR.Q [19]: A recent model-free method using a model-based objective for better representation learning.

As in the GPU-based setting, FLASHSAC uses a single unified configuration across all tasks, with only minimal adjustments to match each benchmark's conventions. Since sample collection is considerably slower with a single environment, the CPU-based configuration differs from the GPU setting by reducing the batch size from 2048 to 512 and setting the update-to-data ratio to 1, reflecting the lower data throughput.

Experimental Results: As illustrated in Figure V-B, FLASHSAC consistently outperforms all baselines across representative tasks in this sample-efficient regime. Full per-task results appear in § XII. PPO performs particularly poorly here, as on-policy methods cannot reuse experience and thus suffer under limited sample budgets.

These results confirm that the design choices in FLASHSAC generalize beyond massively parallel GPU-based simulation: even in the classical single-environment setting, where sample efficiency is the primary bottleneck, FLASHSAC matches or exceeds dedicated sample-efficient methods without task-specific tuning.

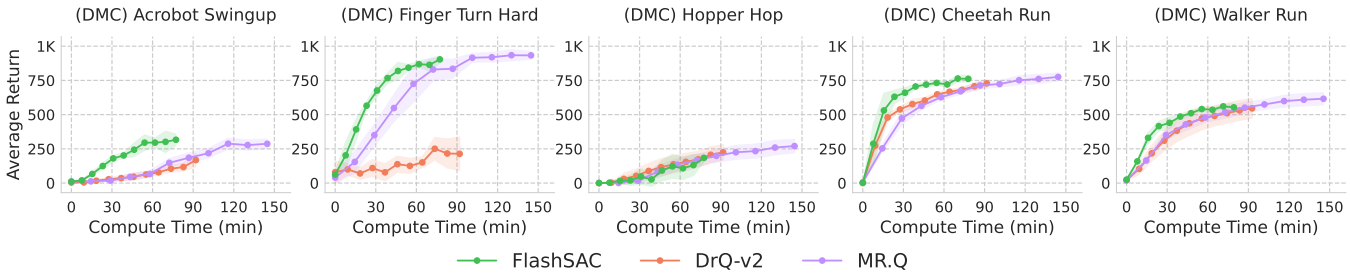

Fig. 5: **Results on Vision-Based RL.** Learning curves on selected tasks from vision-based DMControl Suite [96]. We assess learning performance in low-dimensional environments, including pendulum manipulation and bipedal locomotion. FLASHSAC achieves better compute efficiency and higher asymptotic performance.

Fig. 6: **Sim-to-real Stair Climbing on Unitree G1.** FLASHSAC achieves stable real-world stair climbing after only 4 hours of training on simulation, whereas PPO requires nearly 20 hours to reach the same capability.

C. Vision-based RL

Experimental Setup: We extend our evaluation to vision-based control, where high rendering cost and low environment throughput severely limit the number of transitions collected per unit time, making data efficiency critical. We evaluate on 8 tasks from the DMControl Suite [100], spanning manipulation and mono/bi-pedal locomotion. See § X for complete task list.

Given the low throughput of visual environments, we focus on comparing FLASHSAC against the off-policy baselines:

- DrQ-v2 [111]: A DDPG-based [51] method that improves data efficiency through image augmentation [41].
- MR.Q [19]: An off-policy method that incorporates a dynamics modeling objective to improve representation learning.

As in the CPU-based state experiments (§ V-B), sample collection is slow; we reuse the same hyperparameters from that setup, adapting only the following to match the standard DrQ-v2 configuration [111]: (i) a lightweight convolutional encoder (3 convolutional layers followed by a linear bottleneck), (ii) frame stacking of the three most recent frames ($84 \times 84 \times 9$) for temporal reasoning without recurrent architectures, and (iii) 3-step returns for better credit assignment. All methods are trained for 1M environment steps with an action repeat of 2. Full details are in § X.

Experimental Results: Figure 5 shows representative learning curves, with full results in § XII. Across tasks, FLASHSAC matches or exceeds all baselines in asymptotic performance while converging faster in wall-clock time. DrQ-v2 is sample-efficient but unstable, failing to converge in several environments (e.g., Finger Turn Hard). MR.Q achieves high final performance but incurs additional computational cost from its auxiliary dynamics model. In contrast, FLASHSAC achieves competitive or superior results with a single set of hyperparameters and without auxiliary objectives.

We note that the stabilization techniques in FLASHSAC are orthogonal to such extensions; for example, MR.Q's representation learning objective could be layered on top for further gains in visual feature learning.

D. Sim-to-Real Transfer

Off-policy RL is often regarded as unreliable for sim-to-real transfer, particularly in high-dimensional systems, where training instability can lead to unsafe behaviors [63]. We evaluate whether FLASHSAC enables reliable sim-to-real transfer on a challenging 29-DoF Unitree G1 humanoid performing blind locomotion.

Experimental Setup: We train blind locomotion policies in simulation using a terrain curriculum comprising pyramid stairs, discrete grids, waves, and pits. The curriculum consists of 10 terrain levels with stair heights ranging from 0 to 23cm (step width 32cm, platform width 3m). Terrain difficulty is increased automatically using a game-inspired curriculum [82]. To facilitate sim-to-real transfer, we apply large-scale domain randomization alongside the terrain curriculum; full details are provided in § XI.

As a baseline, we use PPO with the sim-to-real pipeline of [64]. FLASHSAC adopts the same sim-to-real adaptation techniques for a fair comparison. Both methods use implicit system identification via a context estimator [64] and an asymmetric actor–critic formulation [74], where the critic receives privileged information (e.g., contact states and height maps) during training. Both methods share identical reward design and coefficients, combining velocity tracking with

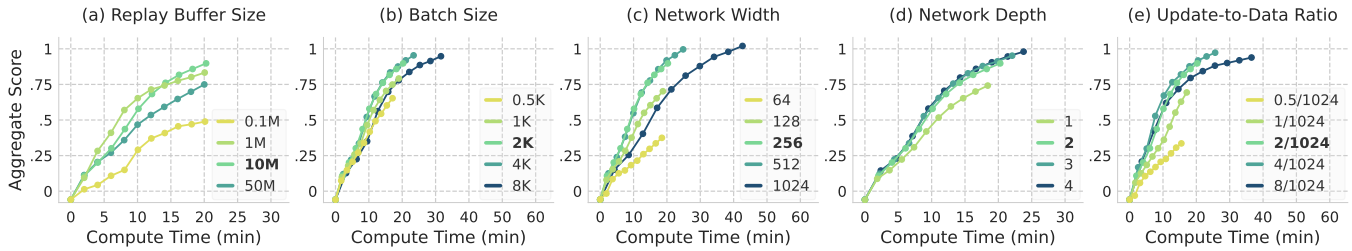

Fig. 8: **Scaling Results. (a)** Replay buffer size trades off training stability and efficiency. **(b–e)** Scaling batch size, and model capacity while reducing the UTD ratio accelerates convergence.

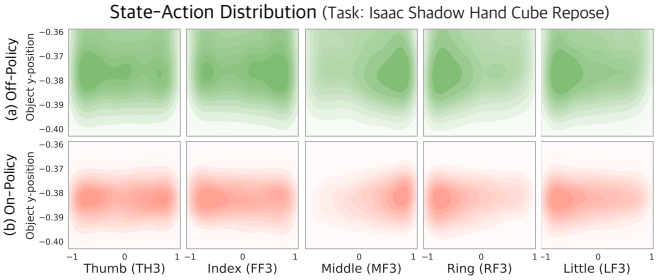

Fig. 7: **Off-Policy vs. On-Policy Data Coverage.** We train FLASHSAC for 1M steps on the Shadow Hand cube reorientation task. 2D density plots show the joint distribution of object y-position and fingertip joint actions for **(a) off-policy** samples from the replay buffer and **(b) on-policy** samples collected by rolling out the final policy for an additional 1M steps. Off-policy data covers a substantially broader region of the state–action space.

regularization terms penalizing foot slip, excessive torque, action discontinuities, and orientation instability. FLASHSAC uses the same architecture and hyperparameters as in the state-based experiments V-B.

Experimental Results: On flat terrain (Figure 1.(c)), FLASHSAC achieves stable real-world locomotion after approximately 20 minutes of training, whereas PPO requires about 3 hours to reach comparable performance. The learned policy supports omnidirectional locomotion (forward, backward, and lateral) without re-training.

The advantage of FLASHSAC is more pronounced on rough terrain (Figure 6). In the real-world setup, the robot faces stairs of 15cm height, 60cm width, and 1.5m platform width—conditions unseen during training, which uses different stair dimensions. FLASHSAC successfully climbs these stairs after approximately 4 hours of training, while PPO requires nearly 20 hours to achieve a similar capability.

Overall, FLASHSAC reduces the training time for sim-to-real humanoid locomotion by nearly an order of magnitude compared to PPO while maintaining stable and safe behaviors.

VI. ANALYSIS

In this section, we analyze the factors underlying FLASHSAC's performance across four aspects. We first examine how off-policy learning yields broader state–action coverage than on-policy methods (§VI-A). We then investigate three design choices central to FLASHSAC: scaling data collection and model capacity for faster training (§VI-B), architectural ablations that improve training stability (§VI-C), and the effect of entropy and temporal correlation on exploration (§VI-D). All experiments are conducted in four IsaacLab environments [61]: cube reorientation with the Allegro and Shadow Hands, and flat and rough terrain locomotion with the G1.

A. Off-Policy vs On-Policy

We train FLASHSAC for 1M steps on the IsaacLab Shadow Hand task with a replay buffer of size 1M, then collect 1M additional on-policy transitions by rolling out the final policy.

Figure 7 compares the state–action coverage of the two datasets via 2D density plots of finger actions versus object y-position. Off-policy data (Figure 7.(a)) covers a substantially broader region of the state–action space, reflecting experience accumulated across diverse behavior policies stored in the replay buffer. On-policy data (Figure 7.(b)), by contrast, is tightly concentrated around the final policy's distribution. This disparity suggests that limited state–action coverage is a key factor in the reduced effectiveness of on-policy methods on high-dimensional tasks, where achieving comparable coverage would require substantially more data collection.

B. Scaling Ablation for Faster Training

We study how scaling data, model capacity, and reducing the number of gradient updates (§IV-A) affects the compute efficiency of FLASHSAC. We perform univariate ablations over five hyperparameters: batch size, replay buffer size, network width, network depth, and update-to-data (UTD) ratio.

Figure 8 shows learning curves plotted against wall-clock time. Increasing replay buffer size improves performance up to 10M transitions by stabilizing training (Figure 8.(a)). However, overly large buffers (e.g., 50M) slow learning because recent high-quality samples are drawn less frequently, though they can achieve slightly higher asymptotic performance given sufficient training time.

Figures 8.(b)–(e) exhibit trends consistent with established scaling laws [37]: increasing batch size and model capacity, along with reducing the UTD ratio, accelerate convergence. Most existing off-policy RL methods rely on small architectures for training stability (e.g., width 128 with inverted bottlenecks and block depth 1), which limits convergence speed. The scaling

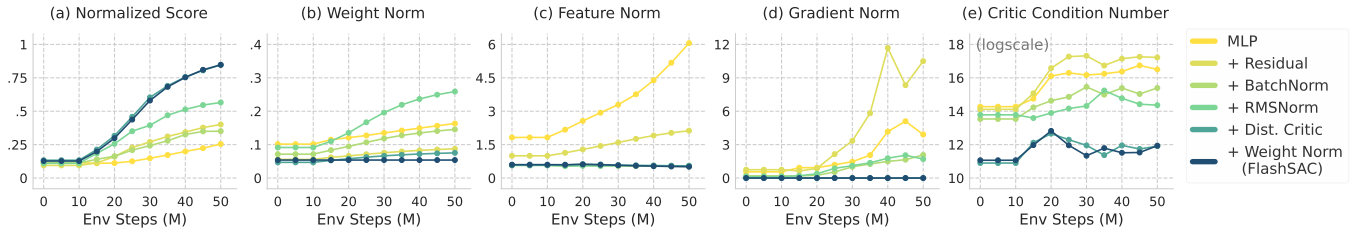

Fig. 9: **Architectural Ablations.** Starting from a standard MLP, each component is incrementally added to build up to FlashSAC. Each addition stabilizes training by constraining weight, feature, and gradient norms while reducing the condition number.

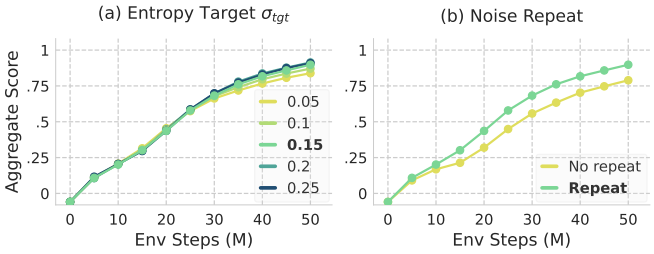

Fig. 10: **Exploration Ablation Results. (a) Unified entropy target:** The optimal entropy target σ_{tgt}lies in the range 0.15 to 0.2 across tasks, enabling a unified setting without task-specific tuning. **(b) Noise repetition:** Repeating action noise accelerates convergence and improves asymptotic performance.

mechanisms of FlashSAC enable higher-capacity models, resulting in substantially faster convergence.

C. Architectural Ablation for Stable Training

We analyze the contribution of each architectural component in FlashSAC (§IV-B) to determine whether the proposed design stabilizes training. Beyond final task performance, we measure parameter, feature, and gradient norms throughout training as indicators of optimization stability. Following [72], we also measure the condition number of the critic loss landscape, where larger values correspond to poorly conditioned updates that can exacerbate critic error amplification.

Starting from a standard MLP critic, we incrementally add: Residual Blocks, Batch Normalization, Post RMSNorm, Distributional Critics with Reward Scaling, and Weight Normalization. Figure 9 summarizes the results. As components are added, parameter, feature, and gradient norms remain bounded throughout training with no uncontrolled growth, indicating well-behaved critic updates and reduced error amplification. The condition number also decreases monotonically, reaching its lowest value with the full FlashSAC architecture.

These gains in optimization stability directly translate to improved task performance (Figure 9.(a)), underscoring the importance of controlling update dynamics in off-policy RL. While weight normalization alone yields modest gains, it improves robustness in sample-limited regimes and is therefore retained in the final design.

D. Exploration Ablation

We analyze FlashSAC's exploration strategy (§IV-C): unifying the entropy target σ_{tgt} and noise repetition.

Figure 10.(a) shows the effect of varying the entropy target σ_{tgt} across values $\{0.05, 0.1, 0.15, 0.2, 0.25\}$. Performance is largely insensitive to this hyperparameter, with all settings converging to similar asymptotic scores. This robustness simplifies tuning in practice, as the default value ($\sigma_{tgt} = 0.15$) performs well without task-specific adjustment.

Figure 10.(b) compares training with and without noise repetition. Disabling noise repeat leads to slower convergence and lower aggregate scores, confirming that temporally correlated exploration is crucial for FlashSAC. Repeating sampled action noise across consecutive steps produces coherent exploratory trajectories rather than uncorrelated perturbations that are quickly averaged out by the dynamics in high-dimensional control tasks.

VII. Lessons and Opportunities

We presented FlashSAC, a fast and stable off-policy RL framework for high-dimensional robotics. As the robotics community moves toward high-dimensional [21], perception-rich [66, 36], and contact-intensive tasks [109], the scalability of on-policy RL becomes increasingly constrained. Off-policy RL is an appealing alternative, but its adoption has been limited by slow training speed and instability in critic learning arising from function approximation error and bootstrapped updates. FlashSAC addresses these challenges through two complementary mechanisms: scaling data and model capacity while reducing the number of gradient updates for faster training, and integrating explicit architectural constraints on critic updates for stable optimization. Together, these yield strong asymptotic performance and up to an order-of-magnitude reduction in wall-clock time compared to on-policy methods.

Stabilized off-policy learning opens new opportunities for robot learning. Improved data efficiency makes it feasible to train larger policies, incorporate vision and other rich sensory inputs [92], and leverage slower but more realistic simulators [76]. Off-policy methods also naturally support learning from a mixture of demonstrations and self-collected experience [6]. While this work focuses on state-based control, extending these critic-stabilization principles to tactile-based learning is a promising direction for future work.

Acknowledgements

We would like to express our gratitude to Younggyo Seo and Yekyung Nah for their valuable feedback on this paper. This work was supported by the Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019- II190075, Artificial Intelligence Graduate School Program (KAIST)). This research was also funded by the research cluster "Third Wave of AI", funded by the excellence program of the Hessian Ministry of Higher Education, Science, Research and the Arts, hessian.AI and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy (EXC-3057/1 "Reasonable Artificial Intelligence", Project No. 533677015). It was further partially supported by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG).

References

[1] Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pages 620–636. PMLR, 2023.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1): 3–20, 2020.

[3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[4] Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[7] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[8] 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. *International Conference on Learning Representations (ICLR)*, 2024.

[9] Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. Myosuite–a contact-rich simulation suite for musculoskeletal motor control. *arXiv preprint arXiv:2205.13600*, 2022.

[10] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

[11] Yuanpei Chen, Yaodong Yang, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen Marcus McAleer, Hao Dong, and Song-Chun Zhu. Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=D29JbExncTP.

[12] Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

[13] Shibhansh Dohare, J Fernando Hernandez-Garcia, Parash Rahman, Richard S Sutton, and A Rupam Mahmood. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023.

[14] Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Mohamed Elsayed, Qingfeng Lan, Clare Lyle, and A. Rupam Mahmood. Weight clipping for deep continual and reinforcement learning. *Reinforcement Learning Journal*, 5:2198–2217, 2024.

[16] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International conference on machine learning*, pages 3061–3071. PMLR, 2020.

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

[18] Scott Fujimoto, Wei-Di Chang, Edward J Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. *arXiv preprint arXiv:2306.02451*, 2023.

[19] Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. *arXiv preprint arXiv:2501.16142*, 2025.

[20] Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

[21] Zhaoyuan Gu, Junheng Li, Wenlan Shen, Wenhao Yu, Zhaoming Xie, Stephen McCrory, Xianyi Cheng, Abdulaziz Shamsah, Robert Griffin, C Karen Liu, et al. Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning. *arXiv preprint arXiv:2501.02116*, 2025.

[22] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and

Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[23] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[24] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Ahmed Hendawy, Henrik Metternich, Théo Vincent, Mahdi Kallel, Jan Peters, and Carlo D'Eramo. Use the online network if you can: Towards fast and stable reinforcement learning. *arXiv preprint arXiv:2510.02590*, 2025.

[27] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.

[28] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[29] Jakob Hollenstein, Sayantan Auddy, Matteo Saveriano, Erwan Renaudo, and Justus Piater. Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *arXiv preprint arXiv:2206.03787*, 2022.

[30] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[31] Marcel Hussing, Claas Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton. Dissecting deep RL with high update ratios: Combatting value divergence. In *Reinforcement Learning Conference (RLC)*, volume 5, 2024.

[32] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[34] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[35] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7 (2):4630–4637, April 2022. ISSN 2377-3774. doi: 10.1109/lra.2022.3151396. URL http://dx.doi.org/10.1109/LRA.2022.3151396.

[36] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16923–16930. IEEE, 2025.

[37] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[38] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *CoRL*, 2025.

[39] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[40] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[41] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[42] Hojoon Lee, Hanseul Cho, Hyunseung Kim, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, and Chulhee Yun. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Hojoon Lee, Hyeonseo Cho, Hyunseung Kim, Donghu Kim, Dugki Min, Jaegul Choo, and Clare Lyle. Slow and steady wins the race: Maintaining plasticity with hare and tortoise networks. *arXiv preprint arXiv:2406.02596*, 2024.

[44] Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. *arXiv preprint arXiv:2410.09754*, 2024.

[45] Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

[46] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), October 2020. ISSN 2470-9476. doi: 10.1126/scirobotics.abc5986. URL http://dx.doi.org/10.1126/scirobotics.abc5986.

[47] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. *arXiv preprint arXiv:2304.10466*, 2023.

[48] 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 *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

[49] Qiayuan Liao, Takara E Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241*, 2025.

[50] Shuhao Liao, Peizhuo Li, Xinrong Yang, Linnan Chang, Zhaoxin Fan, Qing Wang, Lei Shi, Yuhong Cao, Wenjun Wu, and Guillaume Sartoretti. Gpo: Growing policy optimization for legged robot locomotion and whole-body control. *arXiv preprint arXiv:2601.20668*, 2026.

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

[52] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[53] Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*, 2024.

[54] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. *Proc. the International Conference on Machine Learning (ICML)*, 2023.

[55] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado P van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:106440–106473, 2024.

[56] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

[57] A Rupam Mahmood, Hado P Van Hasselt, and Richard S Sutton. Weighted importance sampling for off-policy learning with linear function approximation. *Advances in neural information processing systems*, 27, 2014.

[58] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[59] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[60] Mayank Mittal, Nikita Rudin, Victor Klemm, Arthur Allshire, and Marco Hutter. Symmetry considerations for learning task symmetric robot policies. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7433–7439. IEEE, 2024.

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

[62] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[63] J.W. Mock and University of Wyoming. Department of Electrical Engineering. *A Comparison of PPO, TD3, and SAC Reinforcement Algorithms for Quadruped Walking Gait Generation and Transfer Learning to a Physical Robot*. University of Wyoming, 2023. ISBN 9798379561789. URL https://books.google.co.kr/books?id=waUG0AEACAAJ.

[64] I Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. *arXiv preprint arXiv:2301.10602*, 2023.

[65] Abhishek Naik, Yi Wan, Manan Tomar, and Richard S Sutton. Reward centering. *arXiv preprint arXiv:2405.09999*, 2024.

[66] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

[67] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. *arXiv preprint arXiv:2405.16158*, 2024.

[68] Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *arXiv preprint arXiv:2505.23150*, 2025.

[69] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[70] Johan Obando-Ceron, Walter Mayor, Samuel Lavoie, Scott Fujimoto, Aaron Courville, and Pablo Samuel Castro. Simplicial embeddings improve sample efficiency in actor-critic agents. *arXiv preprint arXiv:2510.13704*, 2025.

[71] Daniel Palenicek, Florian Vogt, Joe Watson, and Jan Peters. Scaling off-policy reinforcement learning with batch and weight normalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[72] Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. XQC: Well-conditioned optimization accelerates deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2026.

[73] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[74] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[75] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[76] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.

[77] Antonin Raffin. Getting SAC to work on a massive parallel simulator: An RL journey with off-policy algorithms (part I). https://araffin.github.io/post/sac-massive-sim/, February 2025.

[78] Antonin Raffin. Getting SAC to work on a massive parallel simulator: Tuning for speed (part II). https://araffin.github.io/post/tune-sac-isaac-sim/, July 2025.

[79] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pages 1634–1644, 2022.

[80] Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. State-dependent exploration for policy gradient methods. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2008.

[81] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pages 91–100. PMLR, 2022.

[82] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pages 91–100. PMLR, 2022.

[83] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[84] Tom Schaul, Georg Ostrovski, Iurii Kemaev, and Diana Borsa. Return-based scaling: Yet another normalisation trick for deep rl. *arXiv preprint arXiv:2105.05347*, 2021.

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

[86] 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.

[87] 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.

[88] 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.

[89] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

[90] Christian Robert Shelton. Importance sampling for reinforcement learning with multiple objectives. *PhD thesis, Massachusetts Institute of Technology*, 2001.

[91] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

[92] Entong Su, Chengzhe Jia, Yuzhe Qin, Wenxuan Zhou, Annabella Macaluso, Binghao Huang, and Xiaolong Wang. Sim2real manipulation on unknown objects with tactile-based reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9234–9241. IEEE, 2024.

[93] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[94] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[95] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong

Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

[96] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[97] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[98] 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*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

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

[100] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[101] 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.

[102] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

[103] Gautham Vasan, Mohamed Elsayed, Alireza Azimi, Jiamin He, Fahim Shariar, Colin Bellinger, Martha White, and A. Rupam Mahmood. Deep policy gradient methods without batch updates, target networks, or replay buffers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[104] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[105] Théo Vincent, Tim Faust, Yogesh Tripathi, Jan Peters, and Carlo D'Eramo. Eau de Q-network: Adaptive distillation of neural networks in deep reinforcement learning. In *Reinforcement Learning Journal / Reinforcement Learning Conference (RLC)*, 2025.

[106] Théo Vincent, Yogesh Tripathi, Tim Faust, Abdullah Akgül, Yaniv Oren, Melih Kandemir, Jan Peters, and Carlo D'Eramo. Bridging the performance gap between target-free and target-based reinforcement learning. *arXiv preprint arXiv:2506.04398*, 2025.

[107] Claas Voelcker, Axel Brunnbauer, Marcel Hussing, Michal Nauman, Pieter Abbeel, Eric Eaton, Radu Grosu, Amir-massoud Farahmand, and Igor Gilitschenski. Relative entropy pathwise policy optimization. *arXiv preprint arXiv:2507.11019*, 2025.

[108] Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from learning to spin" pens". *arXiv preprint arXiv:2407.18902*, 2024.

[109] Yuran Wang, Ruihai Wu, Yue Chen, Jiarui Wang, Jiaqi Liang, Ziyu Zhu, Haoran Geng, Jitendra Malik, Pieter Abbeel, and Hao Dong. Dexgarmentlab: Dexterous garment manipulation environment with generalizable policy. *arXiv preprint arXiv:2505.11032*, 2025.

[110] Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. *arXiv preprint arXiv:2401.16025*, 2024.

[111] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[112] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

[113] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[114] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.