HE-FastSAC: Heterogeneous Training Objectives for Massively Parallel Off-Policy Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Off-policy reinforcement learning has recently emerged as an alternative to on-policy methods on massively parallel simulators, where reuse of replay-buffer data improves sample efficiency. Current recipes in this regime optimize actor and critic under a single training objective, with hyperparameters tuned per task or per benchmark suite. Building on FastSAC, we present HE-FastSAC, which instead trains a population of policies under heterogeneous training objectives in a single run with minimal overhead. We partition the parallel environments into groups, each with its own target entropy and discount factor. A single actor and critic are shared across groups, and every group learns from the experience collected by all groups. We find that a policy learns faster and reaches a higher final score when trained inside this population than when trained alone under the same training objective. The gain comes both from the experience the population collects and from training a single network under multiple training objectives. Using one fixed hyperparameter set across 56 continuous-control tasks from four benchmark suites, with the same group deployed on every task, HE-FastSAC outperforms strong off-policy baselines and matches a per-task hyperparameter sweep of the same agent trained alone. Selecting a trained policy per task from the population yields a further gain with no additional training cost.

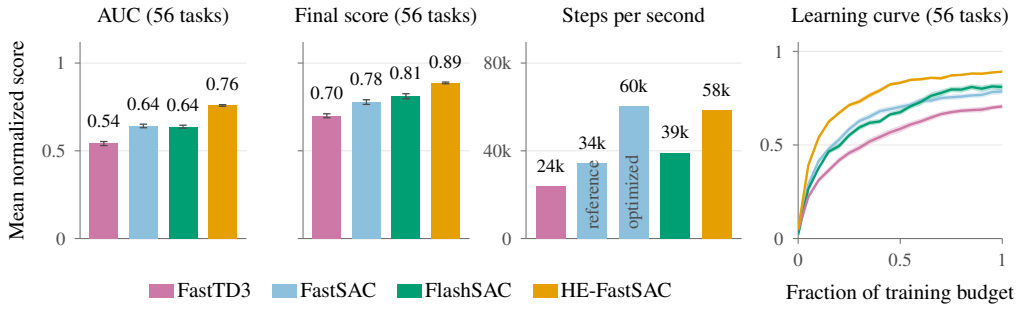

Figure 1: Benchmark summary. With one fixed hyperparameter set and the same group deployed on 56 tasks from four suites, HE-FastSAC outperforms FastSAC, FlashSAC, and FastTD3 in area under the learning curve (AUC) and in final score. Bars and curve show the mean of normalized scores over 5 seeds with 95% stratified bootstrap confidence intervals. The throughput panel shows steps per second on G1JoystickFlatTerrain at 4096 environments, with FastSAC in its reference implementation and in our optimized one.

1 Introduction

On massively parallel GPU simulators, which step thousands of environments together with the policy network, on-policy methods such as PPO (Schulman et al., 2017) have become the standard, training locomotion controllers in minutes that transfer to physical hardware (Rudin et al., 2021).

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.

More recently, off-policy methods including PQL, FastTD3, FastSAC, and FlashSAC have been adapted to this setting (Li et al., 2023; Seo et al., 2025b;a; Kim et al., 2026), where data reuse from a replay buffer further improves sample efficiency. These recipes optimize a single actor and critic under a single training objective and set its hyperparameters, such as the discount factor or the target entropy, per task or benchmark suite (Seo et al., 2025a; Kim et al., 2026). No principled rule exists for choosing them in advance (Hansen et al., 2024). The choice is made once per run, and every parallel environment is then stepped by the same policy. Policies under different training objectives, however, behave differently: a more exploratory policy reaches states that a less exploratory one rarely visits, and a policy with a longer horizon pursues rewards that a shorter one discounts away. We posit that stepping the environments with several such policies could fill the replay buffer with more diverse experience, and that training one network on all their training objectives at once could benefit each of them.

Families of policies with different training objectives and shared networks have been used in hard-exploration reinforcement learning. In NGU and Agent57, each member has its own discount factor, and all members but one add an intrinsic reward and serve as explorers and as auxiliary tasks for the shared networks (Badia et al., 2020b;a). Agent57 and MEME choose with a bandit which member acts, and MEME trains every member on the data of all members (Badia et al., 2020a; Kapturowski et al., 2023). In massively parallel simulation, SAPG trains a leader policy on its own data and on the data of follower policies, which in one variant have different entropy coefficients (Singla et al., 2024). We ask instead how much a policy gains from training in a simpler population, whose members share the task reward and are trained alike, without a leader or a bandit.

To this end, we present HE-FastSAC, an extension of FastSAC that trains a population of policies under heterogeneous (HE) training objectives in a single run. It partitions the parallel environments into equal groups and assigns each group its own target entropy and discount factor. One actor and one critic, conditioned on these two values, represent the entire population, so the networks do not grow with the number of groups. Because the learner is off-policy, each sampled transition can update a group drawn uniformly at random, regardless of the group that collected it, so that every group trains on the experience collected by all groups. Since the groups divide the environments of one run and share one network, the population adds little cost.

We find that a policy learns faster and reaches a higher final score inside the population than when it is trained alone with the same target entropy and discount factor. Pooled over 56 tasks, training inside the population raises the mean area under the learning curve from 0.71 to 0.76. The gain is not confined to one group. Every group trains better inside the population than alone, so one run yields a good policy for each training objective. Both the diverse experience and the training of one network on all training objectives contribute to the gain: each alone recovers about half of it, and when every group trains only on its own transitions, the gain disappears. Our interpretation is that the groups collect experience that a single training objective would not, and that the training objectives act as auxiliary tasks for one another in the shared network.

On 56 tasks from four benchmark suites, MuJoCo Playground (Zakka et al., 2025), IsaacLab (Mittal et al., 2025), HumanoidBench (Sferrazza et al., 2024), and ManiSkill (Tao et al., 2024), HE-FastSAC outperforms FastSAC, FlashSAC, and FastTD3 in sample efficiency and final score, with one set of hyperparameters and the same group deployed on every task (Figure 1). On twelve of these tasks, it matches a per-task sweep of target entropy and discount factor for the same agent trained alone, at about the cost of one of the sweep's runs. Which group is best depends on the task, and selecting the group to deploy per task, from the same run, adds a further gain. We will release our code.

2 PRELIMINARIES

Maximum-entropy reinforcement learning. We consider a Markov decision process with state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, transition kernel P, reward r, and discount factor $\gamma \in [0, 1)$. A stochastic policy $\pi_\theta(\cdot \mid s)$ acts in the environment, and transitions (s, a, r, s') are stored in a replay buffer $\mathcal{D}$. SAC (Haarnoja et al., 2018a) optimizes the maximum-entropy objective

$$J(\pi) \;=\; \mathbb{E}_\pi\left[\sum_t \gamma^t \big(r_t + \alpha\, \mathcal{H}\big(\pi(\cdot \mid s_t)\big)\big)\right],$$

where $\alpha \geq 0$ is the entropy temperature. Twin critics Q_{ϕ_1}, Q_{ϕ_2} are regressed to a soft Bellman target computed with slowly updated target networks, and the actor is updated by the reparameterization gradient. The temperature is adjusted automatically with the constrained formulation of Haarnoja et al. (2018b): α is learned to enforce $\mathbb{E}_{(s,a)\sim\pi}[-\log \pi(a \mid s)] \geq \bar{\mathcal{H}}$ for a fixed target entropy $\bar{\mathcal{H}}$, through the dual loss

$$\mathcal{L}_\alpha \;=\; -\alpha \left(\log \pi(a \mid s) + \bar{\mathcal{H}} \right).$$

The standard heuristic sets $\bar{\mathcal{H}} = -|\mathcal{A}|$, and we state target entropies in units of $|\mathcal{A}|$ throughout. A higher target entropy keeps the policy more random, so it explores more. The discount factor sets the effective horizon $1/(1-\gamma)$ of the objective. Together with the reward, $\bar{\mathcal{H}}$ and γ specify the training objective.

FastSAC. We build on FastSAC (Seo et al., 2025a), which adapts SAC to massively parallel simulation. N environments are stepped synchronously on one GPU, the actor is evaluated once per step over the batch of N states, and one replay buffer $\mathcal{D}$ collects the transitions of all environments. Each environment step is followed by a fixed number of gradient updates on mini-batches of size B. On top of SAC, FastSAC trains with large mini-batches, replaces the scalar critic with a categorical distributional critic (Bellemare et al., 2017), and applies observation and layer normalization. FastSAC was not evaluated on the benchmark suites we use, and its authors tune its hyperparameters, among them the target entropy and the discount factor, per task. We therefore use one set of our own on every task (Section 3).

3 HE-FastSAC

HE-FastSAC builds on FastSAC with two components adopted from FlashSAC (Kim et al., 2026), reward normalization into a fixed critic support (RN) and noise repeat (NR). We call this variant FastSAC+RN+NR and use it as the baseline wherever a comparison isolates the population. On top of it, HE-FastSAC adds groups of environments with their own target entropy and discount factor, one actor and one critic conditioned on the two values, and cross-mixture relabeling. With a single group it reduces to FastSAC+RN+NR.

Base recipe. A categorical critic represents only returns inside its support, which must match the task's return scale. We fix the support to $[-5, 5]$ instead and normalize rewards online so that returns fall inside it, with the running normalizer of SimbaV2 (Lee et al., 2025). Each environment tracks its discounted return G_t, and the normalized reward $\tilde{r}_t$ divides r_t by the larger of two terms. The running standard deviation of G_t keeps the scale of the returns consistent, and the running maximum of G_t divided by the support bound keeps the largest return seen inside the support. Noise repeat holds the exploration noise of the reparameterized action for k consecutive steps before resampling it, which makes exploration temporally correlated, as in gSDE (Raffin et al., 2021; Raffin, 2025b). The length k is drawn from a Zeta distribution, which favors short repeats and occasionally gives long ones (Dabney et al., 2021). The remaining hyperparameters form one set that we use unchanged on every task and suite (Appendix A).

Groups. The N parallel environments are split evenly into K groups. A group g is defined by its target entropy $\mathcal{H}_g$ and its discount factor γ_g, the two values that FastSAC tunes per task and that, with the reward all groups share, specify its training objective. We use $K = 9$ groups on the grid $\bar{\mathcal{H}}_g \in \{0.25, 0, -1\} \cdot |\mathcal{A}|$, $\gamma_g \in \{0.97, 0.99, 0.995\}$, which combines values common in prior recipes with a more exploratory target entropy of $0.25 \cdot |\mathcal{A}|$. Each group has its own temperature α_g, adjusted to $\bar{\mathcal{H}}_g$ by the dual loss of Section 2, and the reward normalizer is kept per discount factor, since the scale of G_t depends on γ.

One actor and one critic. One actor and one critic serve all groups, so that every group trains the same network and the network does not grow with K. To represent a different policy and value per group, we append the group's target entropy (in units of $|\mathcal{A}|$) and its log horizon $\log \frac{1}{1-\gamma_g}$ to the input, standardize them over the grid, and denote this vector by c_g. Unlike a one-hot group index, c_g has two dimensions for any K, gives groups with similar values similar inputs, and could in principle take values between the grid points, which we do not evaluate. The log horizon places the discount factors by the horizons they imply rather than by their values, which lie close together.

Cross-mixture relabeling. Trained on its own transitions alone, an exploitative group would rarely see the rewards found by the exploratory ones. Instead, at every gradient update each sampled transition is assigned a group g' drawn uniformly at random, independent of the group that collected it, and updates the actor, the critic, and the temperature of g'. For a one-step target, a transition (s, a, r, s') relabeled to g' has the critic target

$$y = \tilde{r} + \gamma_{g'}\Big(Q_{\bar{\phi}}(s', a', c_{g'}) - \alpha_{g'} \log \pi_\theta(a' \mid s', c_{g'})\Big), \qquad a' \sim \pi_\theta(\cdot \mid s', c_{g'}), \tag{1}$$

where $Q_{\bar{\phi}}$ is the target critic and $\tilde{r}$ is normalized with the normalizer of $\gamma_{g'}$. Because the next action a' is sampled from the policy of g', the target estimates the value of g''s own policy, no matter which group collected the transition. Every group-dependent quantity of the target is that of the relabeled group: the discount factor, the temperature, the reward normalizer, and the conditioning of the actor and the critic. The n-step categorical target and the actor loss use g' in the same way (Algorithm 1). Each group is thereby trained on a uniform mixture of all groups' experience, as in the cross-mixture training of MEME (Kapturowski et al., 2023), and one batch updates all groups at once. A run yields K policies, one per group. Unless stated otherwise, we deploy the default group, $\bar{\mathcal{H}} = 0$ and $\gamma = 0.99$, which has the baseline's target entropy and discount factor.

4 Experiments

We compare HE-FastSAC at its default group with other off-policy baselines (Section 4.1), add its components to FastSAC one at a time to isolate what a policy gains from the population (Section 4.2), evaluate selecting the deploy group per task (Section 4.3), and report the training cost (Section 4.4). Section 5 analyzes the gain on twelve of the tasks: whether per-task tuning of the two values would give it as well, where it comes from, and whether it is specific to FastSAC.

Setup. We evaluate on 56 tasks from four benchmark suites: 25 DeepMind Control (DMC) tasks and four humanoid joystick-locomotion tasks from MuJoCo Playground (Zakka et al., 2025), 12 IsaacLab tasks (Mittal et al., 2025), 11 HumanoidBench tasks (Sferrazza et al., 2024), and four ManiSkill manipulation tasks (Tao et al., 2024). Per-suite results list DMC and joystick as separate suites. Every method runs 4096 parallel environments, or 128 on HumanoidBench, a CPU simulator, where compute and memory limit the count. On each suite every method performs the same number of gradient updates per environment step (Appendix A). We evaluate policies with the mean action at fixed intervals during training. For the grouped arms, the evaluation environments are partitioned across the groups, so that one evaluation pass yields a learning curve per group. We normalize returns per task by a published reference score and floor them at zero (Appendix B). We report the final score (the mean over the last three evaluations) and the area under the normalized learning curve (AUC). Over five seeds per task and method, both are aggregated as the mean and the interquartile mean (IQM) with 95% stratified bootstrap confidence intervals, together with the probability of improvement (Agarwal et al., 2021).

Baselines. We compare against FlashSAC (Kim et al., 2026), with its published discount factor and n-step returns per suite, and FastTD3 (Seo et al., 2025b), with its per-task configurations. Both run at the environment and update counts above and with their published configuration otherwise (Appendix A). FastSAC (Seo et al., 2025a) runs in our implementation with our hyperparameter set. FastSAC+RN+NR is the baseline that isolates the population: the same agent as HE-FastSAC with a single group at the default target entropy and discount factor.

4.1 Comparison with the baselines

Deployed at its default group, HE-FastSAC has the highest AUC and final score pooled over the 56 tasks, in mean and in IQM (Figure 2, top; Table 5). Against FlashSAC, the strongest baseline on final score, the mean final score is 0.89 against 0.81 ($+0.08$). The gap to each baseline is larger in AUC than in final score, so HE-FastSAC learns faster as well as ending higher. Per suite, HE-FastSAC has the highest AUC everywhere and the highest final score on every suite but joystick, where FlashSAC ends higher (Figure 2, bottom; Figure 6). On ManiSkill every method except FastTD3 is at or near the success ceiling on three of the four tasks, so in final score the suite separates the methods on the fourth alone.

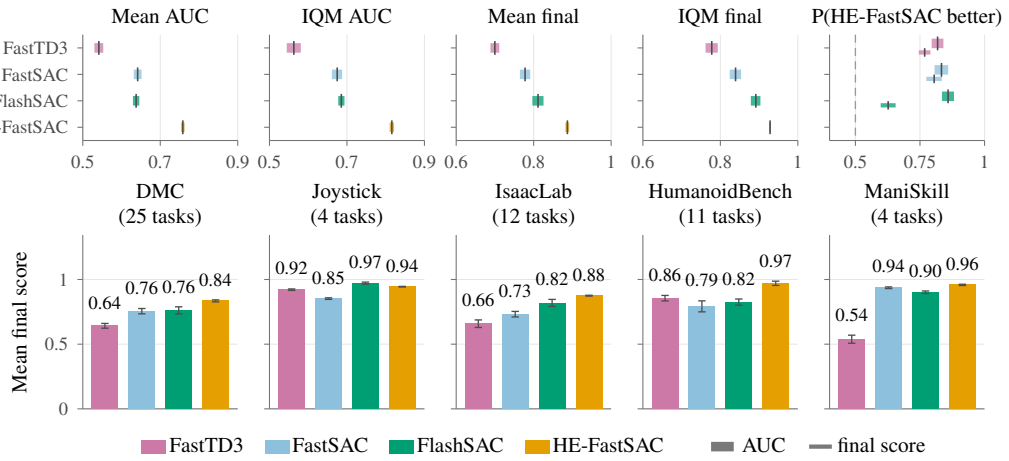

Figure 2: Comparison with the baselines. HE-FastSAC has the highest AUC and final score pooled over the 56 tasks and the highest final score on every suite except joystick. Top: mean and IQM of AUC and final score, 5 seeds, 95% stratified bootstrap confidence intervals, and the probability that HE-FastSAC improves on each baseline (thick bar AUC, thin bar final score). Bottom: mean final score per suite. Values in Table 5, per-suite AUC in Figure 6, per task in Appendix C.

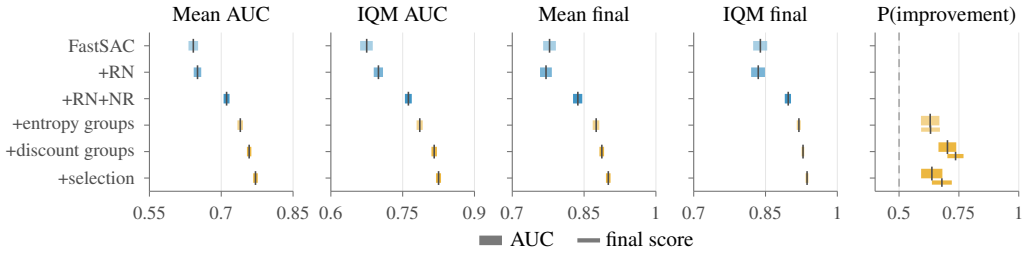

Figure 3: Ablation ladder. Each rung adds one component to FastSAC; a policy with the baseline's target entropy and discount factor trains better inside the population than alone, and selecting the deploy group per task adds a further gain. Pooled mean and IQM, 5 seeds, 95% stratified bootstrap confidence intervals; probability of improvement over FastSAC+RN+NR and, for selection, over the default group (thick bar AUC, thin bar final score). Values in Table 5, per-suite scores in Figure 6.

4.2 Ablation ladder

A policy with the baseline's target entropy and discount factor learns faster and ends higher inside the population than alone with the same values. The ladder of Figure 3 adds one component at a time to FastSAC. Of the two adopted components, reward normalization changes the scores little on its own, since its role is to replace a per-task critic support with one fixed support for every task. Noise repeat adds a clear gain on both metrics, in line with what FlashSAC reports. FastSAC+RN+NR is already above every baseline on both pooled metrics (Table 5), so the population's gain comes on top of that. The three target-entropy groups raise the default group's score on both metrics, and the three discount factors raise it further. Together they add $+0.05$ in mean final score over FastSAC+RN+NR and as much in AUC.

The two axes help on different suites: in final score, the target-entropy groups carry the gain on HumanoidBench and ManiSkill, and the discount groups on joystick and IsaacLab, at some cost on HumanoidBench (Figure 6). The gain is also not uniform across tasks. On about half of the DMC tasks FastSAC+RN+NR already comes within a few percent of the reference score and the population changes little, so the suite's gain comes from a handful of tasks with sparse rewards or slow learning. HE-FastSAC scores below the baseline on HumanoidBench stair and on DMC HopperHop (Appendix C).

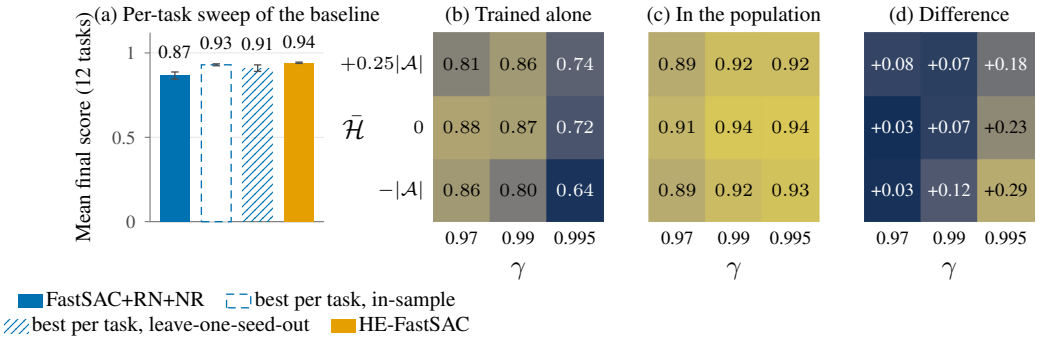

Figure 4: Sweep and population effect on the 12-task subset. The default group matches the per-task sweep, and the long-horizon configurations, the worst of the grid alone, reach the level of the rest inside the population. (a) Sweep of FastSAC+RN+NR, best configuration per task leave-one-seed-out (hatched) or in-sample (dashed, a bound). (b) Each configuration alone, (c) inside the population, one color scale, and (d) the difference. Mean final score, 5 seeds; intervals in (a) and in Tables 11 and 12.

4.3 Selecting the deploy group per task

Since every run evaluates all nine groups, the deploy group can be selected per task from the run's own evaluations, which adds a small and reliable gain. The rule deploys the default group at the first three evaluations and, at every later one, the group with the best mean return over the previous three, so that the selection never sees the evaluation it is scored on. Pooled over the 56 tasks it adds $+0.014$ in mean final score (Table 5), close to the best group chosen in-sample, which is a bound and not a comparable rule. It is largest on DMC and absent on HumanoidBench (Figure 6). Which group is best depends on the suite (Appendix D): most often the longest horizon on HumanoidBench, the least exploratory groups on DMC and joystick, and no single group on IsaacLab. We attribute this to the training objective being a proxy for the deployment objective, the undiscounted return of the mean action. The target entropy and the discount factor regularize the proxy, and the right amount of regularization depends on the task (Jiang et al., 2015; Amit et al., 2020). The default group already carries most of the gain, so selection is an option rather than a requirement. One run yields nine trained policies, among which a practitioner can choose by any criterion.

4.4 Training cost

HE-FastSAC trains nine policies for 2 to 11% more wall-clock time per iteration than FastSAC+RN+NR at the same environment and update counts (Table 10). Most of the overhead lies in the update, and it stays small because the nine policies share one network and every update uses one batch of the baseline's size. Our implementation of FastSAC reaches 1.7 to 2.4 times the throughput of the reference implementation (Amazon FAR et al., 2025) at identical hyperparameters, and HE-FastSAC reaches a higher throughput than FlashSAC and FastTD3 on every suite, although on ManiSkill the simulation cost depends on the policy (Appendix E). All timing measurements use one NVIDIA A100 GPU.

5 Analysis

The experiments of this section use 12 representative tasks spanning legged locomotion on flat and rough terrain, classic control with dense and sparse rewards, and in-hand manipulation (Appendix F), with five seeds throughout.

5.1 Comparison with a per-task sweep

HE-FastSAC trains under nine training objectives at once, whereas a single training objective, tuned per task, might do better. To test this, we train FastSAC+RN+NR at every configuration of the

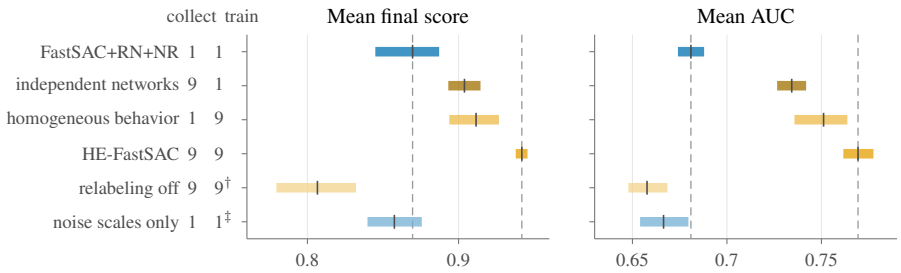

Figure 5: Heterogeneous collection versus joint training on the 12-task subset. Each factor gives part of the gain over FastSAC+RN+NR and only both give all of it; relabeling off falls below the baseline. The columns give the number of policies collecting experience and of training objectives per network (†: each group on its own data; ‡: three noise scales on one policy). Mean final score and mean AUC, 5 seeds, 95% stratified bootstrap confidence intervals, dashed rules at FastSAC+RN+NR and HE-FastSAC; values in Table 11.

same 3×3 grid, nine runs per task and seed, and pick the best configuration per task in two ways. The in-sample choice picks by the seed mean on the evaluations it is scored on. As the best of nine, it is biased upward and is a bound. The leave-one-seed-out choice picks on four seeds and is scored on the fifth. The default group, trained once with one hyperparameter set, matches the best configuration both in-sample and leave-one-seed-out in mean final score (Figure 4a), and in AUC it matches the in-sample bound (Table 11). Section 5.2 shows why a fixed choice loses little: inside the population, the nine groups score much closer to one another than the nine configurations alone. The sweep costs nine runs per task; the population costs one run at a small overhead (Section 4.4).

5.2 Population effect per group

Section 4.2 measures the gain from the population for the default group only. To test whether every group gains, we compare each group with the configuration trained alone at the same target entropy and discount factor, averaged over the tasks (Figure 4b to d). We take the groups from one HE-FastSAC run per task and seed, and the configurations from the sweep of Section 5.1. Every group gains in final score. The gain is largest for the longest-horizon groups, in every row of the grid, and depends less on the target entropy. Alone, the longest-horizon configurations are the worst of the grid, but inside the population they reach the level of the rest. A long-horizon value is the slowest to learn alone, whereas the shared critic learns it together with the shorter horizons. The population thus turns nine configurations of unequal quality into nine policies of similar quality.

5.3 Heterogeneous collection versus joint training

HE-FastSAC differs from FastSAC+RN+NR in two ways: nine policies collect the experience instead of one (heterogeneous collection), and one network is trained on nine training objectives instead of one (joint training). To separate the two, we train one arm with each factor alone (Figure 5). The independent networks keep the heterogeneous collection but give every group its own actor, critic, and temperature, trained on the shared replay buffer as the learners of CERL are (Khadka et al., 2019). Each of them is trained like FastSAC+RN+NR, with the same batch size and number of updates, so the arm has nine times the parameters and update compute of HE-FastSAC. The homogeneous-behavior arm keeps the joint training but lets the default group's policy step every environment.

The gain over FastSAC+RN+NR comes from both factors, and neither alone gives all of it. Each arm recovers about half of the gain in mean final score, and in AUC the homogeneous-behavior arm recovers most of it. The independent networks tie HE-FastSAC in AUC and trail it slightly in final score on the ten tasks with 4096 environments, but fall back to the baseline or below it on the HumanoidBench tasks. There the homogeneous-behavior arm keeps most of the gain in AUC, so the joint training carries it. We attribute the gain from heterogeneous collection to states and rewards that only some of the nine policies reach, and that from joint training to the training objectives acting as auxiliary tasks for one another, as value functions at several horizons do (Fedus et al., 2019).

Two further arms are controls. Relabeling off keeps both factors but sets g' to the group that collected each transition instead of drawing it at random (Section 3), the data scheme of Agent57 (Badia et al., 2020a). Each group then trains only on the transitions of its own environments, a ninth of the total, and the default group falls below FastSAC+RN+NR by 0.06 in mean final score, most on the HumanoidBench tasks. The second control tests whether noise gives the same diversity more simply. This arm keeps the one policy of FastSAC+RN+NR, but each third of the environments explores at the noise level that one of the three target entropies implies. It scores no better than FastSAC+RN+NR and below HE-FastSAC with only the three entropy groups (Table 11). Varying the noise of one policy thus does not replace a population of policies trained under different target entropies. A likely reason is that noise only perturbs the actions of a policy trained under one training objective. A group with a higher target entropy is instead trained to value entropy along its trajectories, so it changes which states it visits as well as how it acts.

5.4 Another base learner

The mechanism is not specific to FastSAC. To test this, we build HE-FlashSAC, which applies the same population of nine groups to FlashSAC (Kim et al., 2026): the groups span three target noise scales around FlashSAC's default and the same three discount factors, and FlashSAC's configuration is otherwise unchanged. Like HE-FastSAC, it trains the same nine training objectives on every suite and deploys the same group everywhere, whereas FlashSAC alone sets its discount factor per suite (Appendix F). On the twelve analysis tasks HE-FlashSAC is above FlashSAC by 0.02 in mean final score and by about as much in AUC (Table 11). The gain comes from DMC, IsaacLab, and HumanoidBench. On the two joystick tasks FlashSAC's per-suite discount factor is the shortest horizon of the grid, and there FlashSAC scores higher.

6 Related work

Reinforcement learning with massively parallel simulation. GPU-based simulators, and CPU-parallel pipelines paired with GPU learning, step thousands of environments alongside the learner (Makoviychuk et al., 2021; Mittal et al., 2025; Zakka et al., 2025; Tao et al., 2024; Jia et al., 2026). PPO has been the standard algorithm in this regime (Schulman et al., 2017; Rudin et al., 2021), but recent work shows that off-policy methods scale to it as well (Raffin, 2025a), as do methods that learn from the parallel stream without a replay buffer (Gallici et al., 2025; Voelcker et al., 2026). PQL trains DDPG with asynchronous actor and learner processes, and FastTD3 and FastSAC keep its large batches and distributional critic in synchronous TD3 and SAC (Li et al., 2023; Seo et al., 2025b;a). FlashSAC and other recent work address the stability and exploration of SAC in this setting (Kim et al., 2026; Sabatini et al., 2026; Xue et al., 2026; Wu et al., 2026). These recipes optimize a single actor and critic under a single training objective and set its target entropy or discount factor per task or per suite (Seo et al., 2025a; Kim et al., 2026). We build on FastSAC and change how its parallel environments are used.

Diverse behavior across parallel environments. Thousands of environments stepped by one policy collect similar trajectories, and the return of PPO saturates as environments are added (Singla et al., 2024; Shitanda et al., 2026). One remedy gives each environment its own exploration noise (Horgan et al., 2018; Li et al., 2023; Xue et al., 2026), with mixed results in this regime (Li et al., 2023; Seo et al., 2025b). Another splits the environments among a population of policies. Population-based training mutates the hyperparameters of independently trained agents (Jaderberg et al., 2017; Petrenko et al., 2023; Shahid et al., 2025; Flajolet et al., 2022). SAPG and Shitanda et al. (2026) pass the data of follower policies to a leader policy through importance sampling (Singla et al., 2024), and De Paola et al. (2025; 2026) train the members to maximize the entropy of their joint state visitation. Per-environment noise varies the behavior of a single policy, and in these populations at most a leader trains on the experience of the other members. We vary the training objective across groups of environments and train every group on the experience of all groups.

Policy families with shared parameters and experience. One network can represent several policies when it is conditioned on what distinguishes them, such as a goal, a task, or a loss function (Schaul et al., 2015; Yu et al., 2019; McLean et al., 2025; Dosovitskiy & Djolonga, 2020). NGU and Agent57 train a family of policies with different intrinsic-reward weights and discount

factors in networks conditioned on the member index, each member on its own episodes (Badia et al., 2020b;a). In Agent57 and MEME, a bandit selects the member that acts and the member that is evaluated (Badia et al., 2020a; Kapturowski et al., 2023), and MEME and SAC-X train each member on all collected data (Kapturowski et al., 2023; Riedmiller et al., 2018). Sharing data in this way tends to give efficiency gains on most games in MEME (Kapturowski et al., 2023) but slightly harms NGU on hard-exploration games (Badia et al., 2020b). SAPG tested letting every policy use the data of all others and found it worse than its leader-follower scheme (Singla et al., 2024). Separate learners can also share one replay buffer, for example the TD3 learners of CERL, which differ only in their discount factor (Khadka et al., 2019), or the agents of a hyperparameter sweep (Schmitt et al., 2020). In contrast to these methods, HE-FastSAC shares one actor and one critic among all groups of a massively parallel off-policy agent and conditions them on each group's target entropy and discount factor. Its groups share the task reward, and no bandit chooses which group acts. We ask whether a fixed group gains from training alongside groups trained with other values.

Target entropy and discount factor. A fixed target entropy does not suit every task, since it explores slowly under sparse rewards and converges lower under dense rewards (Hafner et al., 2023). SAC adjusts its temperature to meet a target entropy (Haarnoja et al., 2018b), and the target itself can be annealed over training (Xu et al., 2021). For the discount factor, other work adapts it online (Xu et al., 2018) or raises it on a schedule after each network reset (Schwarzer et al., 2023). Value functions for several discount factors have been learned as auxiliary tasks (Fedus et al., 2019) or with the discount factor as an input (Sherstan et al., 2020). These works schedule or adapt one target entropy or discount factor for a single policy, or estimate several horizons for it, whereas we train one deployable policy per pair of values.

7 Conclusion

We presented HE-FastSAC, which trains a population of policies under heterogeneous training objectives in a single run: groups of parallel environments with their own target entropy and discount factor, one actor and one critic conditioned on the two values, and every group trained on the experience of all groups. A policy trains better inside this population than alone with the same target entropy and discount factor. The gain comes both from the experience the population collects and from training one network on all its training objectives, and the longest-horizon groups, the worst of the grid when trained alone, gain most. With one fixed hyperparameter set and the same group deployed on every task, HE-FastSAC outperforms strong off-policy baselines across 56 tasks and matches a per-task sweep of the same agent trained alone on twelve of them. Selecting the deploy group per task adds a further gain, so the choice of target entropy and discount factor can be made from the evaluations of one run instead of before it. The gain is not uniform across suites and tasks, and our evidence covers one grid of nine training objectives per base learner, chosen once. Our experiments use massively parallel simulation throughout. Measuring how the gain depends on the number of environments per group and on their total number, and selecting the deploy group inside the agent, as Agent57's bandit does (Badia et al., 2020a), are directions we leave to future work.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code) for the following tasks: providing feedback on the design of the ablation and control experiments, refining the hypotheses about the mechanism that these experiments test, implementing methods, interpreting results, and launching and monitoring long-running experiments. We have not used generative AI tools to conceive the method, to develop the conceptual framework of this work, to generate synthetic data sets, or to formulate or prove mathematical claims; translation, data cleaning, and qualitative or thematic analysis are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: drafting and editing the paper text from the authors' earlier manuscript and notes, writing the figure and table scripts, identifying and summarizing related literature, and formatting references. We have reviewed all AI-assisted work: every number in the tables and figures is produced by scripts from the logged results and cross-checked against an independent analysis pipeline, LLM-written code was tested by the authors, and every statement about cited work was checked against the cited paper by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide our code, including the optimized implementation of FastSAC on which all FastSAC-based runs rely, as anonymized supplementary material. Appendix A lists hyperparameters and baseline configurations; Appendix B lists the tasks, their normalizers, and training budgets; Section 4 presents the evaluation protocol and statistics; and Appendix E describes the timing protocol.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2108.13264.

Amazon FAR, Pieter Abbeel, Juyue Chen, Rocky Duan, Alejandro Escontrela, Manan Gandhi, Samuel Gundry, Xiaoyu Huang, Angjoo Kanazawa, Tomasz Lewicki, Jiaman Li, Karen Liu, Clay Rosenthal, Younggyo Seo, Carlo Sferrazza, Guanya Shi, Linda Shih, Jonathan Tseng, Zhen Wu, Lujie Yang, Brent Yi, and Yuanhang Zhang. Holosoma, 2025. URL `https://github.com/amazon-far/holosoma`. Software.

Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. arXiv:2007.02040.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations (ICLR)*, 2020b.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 449–458, 2017.

Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended ϵ-greedy exploration. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2006.01782.

Vincenzo De Paola, Riccardo Zamboni, Mirco Mutti, and Marcello Restelli. Enhancing diversity in parallel agents: A maximum state entropy exploration story. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Vincenzo De Paola, Mirco Mutti, Riccardo Zamboni, and Marcello Restelli. K-Myriad: Jump-starting reinforcement learning with unsupervised parallel agents. *arXiv preprint arXiv:2601.18580*, 2026.

Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations (ICLR)*, 2020.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G. Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. arXiv:2007.06700.

Arthur Flajolet, Claire Bizon Monroc, Karim Beguir, and Thomas Pierrot. Fast population-based reinforcement learning on a single machine. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pp. 6533–6547, 2022.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.04811.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018a.

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

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

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

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2018. arXiv:1803.00933.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Yufei Jia, Zhanxiang Cao, Mingrui Yu, Heng Zhang, Shenyu Chen, Dixuan Jiang, Meng Li, Xiaofan Li, Yiyang Liu, Junzhe Wu, et al. UniLab: A heterogeneous architecture for robot RL beyond GPU-dominant paradigms. *arXiv preprint arXiv:2605.30313*, 2026.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1181–1189, 2015.

Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakićević, Hado van Hasselt, Charles Blundell, and Adrià Puigdomènech Badia. Human-level Atari 200x faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Shauharda Khadka, Somdeb Majumdar, Tarek Nassar, Zach Dwiel, Evren Tumer, Santiago Miret, Yinyin Liu, and Kagan Tumer. Collaborative evolutionary reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. arXiv:1905.00976.

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. In *Robotics: Science and Systems (RSS)*, 2026. arXiv:2604.04539.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

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 40th International Conference on Machine Learning (ICML)*, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU-based physics simulation for robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Reginald McLean, Evangelos Chatzaroulas, Jordan Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Multi-task reinforcement learning enables parameter scaling. In *Reinforcement Learning Conference (RLC)*, 2025.

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.

Aleksei Petrenko, Arthur Allshire, Gavriel State, Ankur Handa, and Viktor Makoviychuk. DexPBT: Scaling up dexterous manipulation for hand-arm systems with population based training. In *Robotics: Science and Systems (RSS)*, 2023.

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

Antonin Raffin. Getting SAC to work on a massive parallel simulator: Tuning for speed (Part II). *araffin.github.io*, July 2025b. URL `https://araffin.github.io/post/tune-sac-isaac-sim/`.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2021. arXiv:2005.05719.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing – solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. arXiv:1802.10567.

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 (CoRL)*, pp. 91–100, 2021.

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.

Tom Schaul, Dan Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Simon Schmitt, Matteo Hessel, and Karen Simonyan. Off-policy actor-critic with shared experience replay. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 8545–8554, 2020. arXiv:1909.11583.

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

Max Schwarzer, Johan Obando-Ceron, Aaron Courville, Marc G. Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. arXiv:2305.19452.

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.

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

Asad Ali Shahid, Yashraj Narang, Vincenzo Petrone, Enrico Ferrentino, Ankur Handa, Dieter Fox, Marco Pavone, and Loris Roveda. Benchmarking population-based reinforcement learning across robotic tasks with GPU-accelerated simulation. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1231–1238, 2025. arXiv:2404.03336.

Craig Sherstan, Shibhansh Dohare, James MacGlashan, Johannes Günther, and Patrick M. Pilarski. Gamma-Nets: Generalizing value estimation over timescale. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 5717–5725, 2020. arXiv:1911.07794.

Naoki Shitanda, Motoki Omura, Tatsuya Harada, and Takayuki Osa. Rethinking policy diversity in ensemble policy gradient in large-scale reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2603.01741.

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. SAPG: Split and aggregate policy gradients. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. arXiv:2407.20230.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI. *arXiv preprint arXiv:2410.00425*, 2024.

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. In *International Conference on Learning Representations (ICLR)*, 2026.

Zihao Wu, Hongyao Tang, Yi Ma, Huizhong Song, Pengyi Li, Yifu Yuan, Fei Ni, Jinyi Liu, Wei Wei, Jianrong Wang, Yan Zheng, and Jianye Hao. WarpSAC: Towards the pinnacle of scalable off-policy RL by rethinking exploration and exploitation. *arXiv preprint arXiv:2608.24479*, 2026.

Yaosheng Xu, Dailin Hu, Litian Liang, Stephen McAleer, Pieter Abbeel, and Roy Fox. Target entropy annealing for discrete soft actor-critic. In *NeurIPS 2021 Deep Reinforcement Learning Workshop*, 2021.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arXiv:1805.09801.

Jun Xue, Junze Wang, Shanze Wang, Xinming Zhang, Yanjun Chen, and Wei Zhang. FastDSAC: Unlocking the potential of maximum entropy RL in high-dimensional humanoid control. *arXiv preprint arXiv:2603.12612*, 2026.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo Playground. *arXiv preprint arXiv:2502.08844*, 2025.

A HYPERPARAMETERS AND IMPLEMENTATION DETAILS

FastSAC+RN+NR and HE-FastSAC use one hyperparameter set across all benchmarks (Table 1), chosen for FastSAC and used unchanged for HE-FastSAC. The set was fixed in preliminary experiments before the benchmark runs and was not tuned per task or per suite; most values are the defaults of FastSAC's reference implementation or values commonly used in this regime (Seo et al., 2025b; Kim et al., 2026).

Table 1: Hyperparameters of the FastSAC rung, FastSAC+RN+NR, and HE-FastSAC. Values spanning the three columns are shared. Per-task critic supports of the FastSAC rung are in Table 4.

Hyperparameter	FastSAC	FastSAC+RN+NR	HE-FastSAC
Training			
parallel environments N		4096 (HumanoidBench 128)	
gradient updates per environment step		8 (HumanoidBench 2)	
discount factor γ	0.99	0.99	$\{0.97, 0.99, 0.995\}$ per group
n-step return		3	
target smoothing coefficient τ		0.01	
policy delay		2	
warm-up before the first update		10 iterations	
mixed precision		bf16	
Replay			
replay capacity, all environments		4 194 304	
batch size B		8192	
Actor			
hidden layers		$[512, 256, 128]$, SiLU, layer normalization	
action squashing		$\tanh$	
log-standard-deviation bounds		$[-5, 0]$	
observation normalization		yes	
Critic			
hidden layers		$[1024, 512, 256]$, SiLU, layer normalization	
critic heads		2, own target head each; actor takes the minimum	
value distribution		categorical, 101 atoms	
critic support $[V_{\min}, V_{\max}]$	per task	$[-5, 5]$	$[-5, 5]$
reward normalization	none	SimbaV2 running	one tracker per γ
Exploration			
noise repeat	no	yes	yes
Zeta exponent	–	2	2
maximum repeat length	–	16	16
Groups			
number of groups K	1	1	9
target entropy $\bar{\mathcal{H}}$	0	0	$\{0.25, 0, -1\} \cdot \lvert\mathcal{A}\rvert$
initial temperature α_{init}		0.01	
conditioning input c_g	none	none	standardized $(\bar{\mathcal{H}}_g/\lvert\mathcal{A}\rvert,\ \log \frac{1}{1-\gamma_g})$
cross-mixture relabeling	–	–	every sampled transition
Optimizer			
optimizer		AdamW	
learning rate (actor, critic, α)		3×10^{-4}	
momentum (β_1, β_2)		$(0.9, 0.95)$	
weight decay (actor, critic)		10^{-3}	
gradient clipping (actor, critic)		norm 1.0	

Baselines. FlashSAC and FastTD3 are trained from the authors' released code with their published configurations, changed only to the environment and update counts of each suite (Tables 2 and 3). Each keeps its own batch size. FlashSAC's discount factor and n-step follow its released per-suite run scripts except on DMC (Table 2). FlashSAC's paper trains HumanoidBench in a single environment; we run it in the parallel regime of the suite. The FastSAC rung is FastSAC in our implementation with our hyperparameter set and critic supports set per task (Table 4). Appendix E checks our implementation against the reference implementation.

Implementation. HE-FastSAC reuses FastSAC's actor, critic, replay buffer, and update loop; Algorithm 1 lists one training iteration. At every update, the n-step return of a transition relabeled to g' is recomputed from the stored per-step rewards with $\gamma_{g'}$ and the reward normalizer of $\gamma_{g'}$. No off-policy correction is applied to the n-step return, as in the implementation of FastTD3 and in PQL (Seo et al., 2025b; Li et al., 2023). Uncorrected n-step returns from replay are not theoretically justified for off-policy data but are used for their convenience and empirical benefits (Fedus et al.,

Table 2: FlashSAC configuration per suite: the values of its released per-suite run scripts, at the environment and update counts of the suite (the paper's hyperparameter table lists $n = 1$ for every GPU simulator; the IsaacLab script sets $n = 3$). On DMC we use $\gamma = 0.99$ and $n = 3$, which scored higher in our runs than the Playground script's 0.97 and 1.

Suite	γ	n-step	Batch size
Playground DMC	0.99	3	2048
Playground joystick	0.97	1	2048
IsaacLab	0.99	3	2048
HumanoidBench	0.99	1	2048
ManiSkill	0.9	1	2048

Table 3: FastTD3 configuration per suite. Author presets as run, at the environment and update counts of the suite, with batch size 32 768 throughout; per-task critic supports in Table 4. Replay slots per environment are rescaled so that the total equals the preset's total at its own environment count.

Suite	γ	n-step	Atoms	Deviations from the presets
DMC	0.97	3	101	critic support ± 150 (the FastSAC rung's) in place of ± 500
Joystick	0.97	1	101	none
IsaacLab	0.99	8 / 1	251	$n = 8$ on the velocity tasks, 1 otherwise; rough Anymal-C/D take $n = 8$ of the G1/H1 presets and 5120 replay slots
HumanoidBench	0.99	1	101	none
ManiSkill	0.97	1	251	no preset exists; IsaacLab defaults with the FastSAC rung's support ± 100 and $\gamma = 0.97$, our choice

2020). Each critic head is regressed by cross-entropy to the projection of its own target head's distribution onto the support (Bellemare et al., 2017). Each reward normalizer tracks the discounted return $G_t = \gamma(1 - d_t)G_{t-1} + r_t$ of every environment, whichever group it belongs to, with $d_t = 1$ at a termination, and divides the reward by $\max(\sqrt{\sigma_G^2 + \epsilon},\ \max_t G_t/5)$. Noise repeat draws one repeat length for all environments at a time, and each environment keeps its own noise vector for that many steps.

Algorithm 1 One training iteration of HE-FastSAC in the form of FastSAC's Algorithm 1, with the categorical critic omitted as there. The marked lines and the group subscripts are the changes against FastSAC+RN+NR; with $K = 1$ they vanish. $R_{g'}$ is the n-step return computed with $\gamma_{g'}$ and its reward normalizer, m its length, $b = 0$ if it ends in a termination and 1 otherwise, and $\bar{\ell}_g$ the mean of $\log \pi_\theta(a' \mid s_{t+m}, c_g)$ over the transitions with $g' = g$.

Input: groups $(\bar{\mathcal{H}}_g, \gamma_g)$ with conditioning vectors c_g; environment i in group $g(i) = i \bmod K$; actor π_θ; critics Q_{ϕ_1}, Q_{ϕ_2} with targets $Q_{\bar{\phi}_1}, Q_{\bar{\phi}_2}$; temperatures $\alpha_1, \ldots, \alpha_K$; replay buffer $\mathcal{D}$

for each environment step **do**
  $a_i \sim \pi_\theta(\cdot \mid s_i, c_{g(i)})$ for every environment i, with noise repeat
  step the environments; store the transitions in $\mathcal{D}$; update the reward normalizer of each γ_g ■
  for $u = 1, \ldots, U$ **do**
    sample B n-step transitions from $\mathcal{D}$
    draw $g' \sim \mathrm{Uniform}\{1, \ldots, K\}$ for each transition ■
    $a' \sim \pi_\theta(\cdot \mid s_{t+m}, c_{g'})$ for each transition
    $y_j = R_{g'} + b\,\gamma_{g'}^m \big(Q_{\bar{\phi}_j}(s_{t+m}, a', c_{g'}) - \alpha_{g'} \log \pi_\theta(a' \mid s_{t+m}, c_{g'})\big)$ for $j \in \{1, 2\}$
    $\phi_j \leftarrow \phi_j - \lambda \hat{\nabla}_{\phi_j} \mathcal{L}_Q(\phi_j; y_j)$ for $j \in \{1, 2\}$
    $\log \alpha_g \leftarrow \log \alpha_g - \lambda \hat{\nabla}_{\log \alpha_g} \big[-\alpha_g \big(\bar{\ell}_g + \bar{\mathcal{H}}_g\big)\big]$ for each group g ■
    if u is even **then**
      $\theta \leftarrow \theta - \lambda \hat{\nabla}_\theta\, \mathbb{E}\big[\alpha_{g'} \log \pi_\theta(a \mid s, c_{g'}) - \min_j Q_{\phi_j}(s, a, c_{g'})\big], \quad a \sim \pi_\theta(\cdot \mid s, c_{g'})$
    end if
    $\bar{\phi}_j \leftarrow \tau \phi_j + (1 - \tau)\, \bar{\phi}_j$ for $j \in \{1, 2\}$
  end for
end for

B ENVIRONMENT DETAILS

Table 4 lists every task with its normalizer, training budget, and critic support. DMC returns are divided by their bound of 1000, joystick and IsaacLab returns by the near-asymptotic reference returns reported by FlashSAC (Kim et al., 2026), and HumanoidBench returns by the benchmark's success thresholds (Sferrazza et al., 2024). ManiSkill reports the success rate, which is already in $[0, 1]$. IsaacLab environments define no action bounds; every method uses FastTD3's per-task values, an action bound of 3.0 on the two Franka tasks and 1.0 on the other ten, which FlashSAC also uses. The HumanoidBench tasks use the `h1hand` embodiment; their budgets are per-task caps in iterations of 128 environments, based on FastTD3's per-task budgets (Seo et al., 2025b).

Table 4: Task details. Normalizer: the reference score by which returns are divided. Budget: environment steps of training in millions. Critic support: symmetric categorical support $[-V, V]$ of the FastSAC rung and of FastTD3; the other arms use a fixed support with normalized rewards.

Suite	Task	Normalizer	Budget (M steps)	Critic support $\pm$	
				FastSAC	FastTD3
DMC	all 25 tasks	1000	81.92	150	150
Joystick	G1JoystickFlatTerrain	40	81.92	20	10
	G1JoystickRoughTerrain	40	81.92	20	10
	T1JoystickFlatTerrain	40	81.92	20	10
	T1JoystickRoughTerrain	40	81.92	20	10
IsaacLab	Lift Cube Franka	160	81.92	50	50
	Open Drawer Franka	100	81.92	50	50
	Repose Cube Allegro	6000	81.92	500	500
	Repose Cube Shadow	10000	81.92	500	500
	Flat Anymal C	30	81.92	10	10
	Flat Anymal D	30	81.92	10	10
	Flat G1	40	81.92	10	10
	Flat H1	40	81.92	10	10
	Rough Anymal C	30	81.92	10	10
	Rough Anymal D	30	81.92	10	10
	Rough G1	40	81.92	10	10
	Rough H1	40	81.92	10	10
HumanoidBench	balance hard	800	32.00	250	250
	basketball	1200	32.00	2000	2000
	bookshelf simple	2000	12.80	250	250
	hurdle	700	19.84	250	250
	push	700	64.00	500	1000
	room	400	12.80	250	250
	run	700	12.80	250	250
	sit hard	750	19.20	250	250
	stair	700	19.20	250	250
	walk	700	12.80	250	250
	window	650	32.00	250	250
ManiSkill	LiftPegUpright	1	40.96	100	100
	PegInsertionSide	1	40.96	100	100
	PickSingleYCB	1	40.96	100	100
	PullCube	1	40.96	100	100

Policies are evaluated with the mean action every 1000 iterations, every 5000 on HumanoidBench, on evaluation environments separate from the training environments. On the eight IsaacLab tasks without terrain curriculum, every method is evaluated on its training environments instead. The four rough-terrain IsaacLab tasks (Rough Anymal C, Rough Anymal D, Rough G1, and Rough H1) train with IsaacLab's terrain curriculum and without online evaluation. Every method saves a checkpoint every 1000 iterations, and each checkpoint is evaluated afterwards on IsaacLab's play variant of the task, which spawns the robots uniformly over all terrain levels and turns off the curriculum, the observation noise, and the external pushes. Every checkpoint of every method is thus scored on the same terrain distribution.

The ManiSkill tasks (PullCube, LiftPegUpright, PickSingleYCB, and PegInsertionSide) run on ManiSkill's GPU backend with state observations and its normalized dense reward. Episodes last the task's registered length, 50 control steps and 100 on PegInsertionSide, and success does not end an episode. The score of an evaluation is the fraction of its episodes in which the task's success condition holds at some step.

C FULL RESULTS

Table 5 gives the pooled values behind Figures 2 and 3, and Table 6 their sensitivity to the aggregation. Figure 6 breaks them down per suite, Tables 7 and 8 per task, and Figures 7 to 11 show the learning curves of all 56 tasks.

Statistics. Every interval is a 95% stratified bootstrap confidence interval in the manner of Agarwal et al. (2021). The tasks are fixed strata and the five runs of each task are resampled with replacement, so the intervals describe the uncertainty over seeds on these tasks and not over unseen tasks. The interval on a difference between two arms resamples the two arms independently, which is conservative for arms computed from the same runs, such as the selected group against the default group. The IQM is the mean of the middle half of all runs of all tasks. On the 12-task subset of Section 5 this window holds 30 of 60 runs and covers different tasks for different arms, so the tables of Appendix F report the mean only. Normalized scores are floored at zero, which affects 2.2% of the evaluation points, two thirds of them on HumanoidBench push, where every method has negative returns in part of its evaluations. The floor raises FlashSAC's pooled mean AUC by 0.104 and moves no other row by more than 0.002, so it favors FlashSAC rather than HE-FastSAC.

Table 5: Pooled results on the 56 tasks. Mean and IQM of the normalized AUC and final score, 5 seeds, 95% stratified bootstrap confidence intervals in brackets; bold marks the largest point estimate in a column. Each rung of the lower block adds one component to FastSAC; the $K=9$ rung is HE-FastSAC at the default group.

	AUC		Final score	
Method	Mean	IQM	Mean	IQM
FastTD3	0.541 [0.530, 0.553]	0.563 [0.545, 0.581]	0.700 [0.688, 0.711]	0.778 [0.762, 0.793]
FlashSAC	0.638 [0.629, 0.646]	0.685 [0.678, 0.694]	0.811 [0.797, 0.825]	0.892 [0.879, 0.903]
FastSAC	0.642 [0.632, 0.652]	0.675 [0.661, 0.688]	0.778 [0.764, 0.791]	0.839 [0.824, 0.854]
+RN	0.650 [0.642, 0.658]	0.699 [0.690, 0.709]	0.770 [0.758, 0.783]	0.835 [0.820, 0.849]
+RN+NR	0.711 [0.705, 0.717]	0.762 [0.755, 0.769]	0.837 [0.827, 0.846]	0.897 [0.890, 0.903]
+entropy groups ($K=3$)	0.740 [0.734, 0.745]	0.786 [0.779, 0.792]	0.875 [0.868, 0.882]	0.920 [0.915, 0.924]
+discount groups ($K=9$): HE-FastSAC	0.758 [0.754, 0.763]	0.816 [0.810, 0.822]	0.887 [0.882, 0.892]	0.928 [0.926, 0.931]
+deploy-group selection	**0.772** [0.767, 0.776]	**0.825** [0.820, 0.830]	**0.901** [0.896, 0.905]	**0.937** [0.934, 0.939]

Table 6: Sensitivity of the pooled means to the aggregation. Mean AUC and mean final score of the rows of Table 5, pooled over the 56 tasks, suite-balanced (the mean of the five per-suite means), and pooled without the floor at zero; 5 seeds, 95% stratified bootstrap confidence intervals in brackets.

Method	Pooled (Table 5)	Suite-balanced	No floor at zero
Mean AUC			
FastTD3	0.541 [0.530, 0.553]	0.553 [0.542, 0.564]	0.539 [0.528, 0.551]
FlashSAC	0.638 [0.629, 0.646]	0.666 [0.659, 0.673]	0.534 [0.336, 0.639]
FastSAC	0.642 [0.632, 0.652]	0.654 [0.646, 0.663]	0.640 [0.630, 0.651]
+RN	0.650 [0.642, 0.658]	0.655 [0.648, 0.663]	0.649 [0.641, 0.657]
+RN+NR	0.711 [0.705, 0.717]	0.713 [0.705, 0.721]	0.709 [0.703, 0.716]
+entropy groups ($K=3$)	0.740 [0.734, 0.745]	0.745 [0.739, 0.750]	0.738 [0.732, 0.744]
+discount groups ($K=9$): HE-FastSAC	0.758 [0.754, 0.763]	0.775 [0.771, 0.780]	0.757 [0.752, 0.762]
+deploy-group selection	0.772 [0.767, 0.776]	0.783 [0.779, 0.788]	0.770 [0.765, 0.775]
Mean final score			
FastTD3	0.700 [0.688, 0.711]	0.723 [0.713, 0.733]	0.700 [0.688, 0.711]
FlashSAC	0.811 [0.797, 0.825]	0.856 [0.846, 0.865]	0.810 [0.796, 0.824]
FastSAC	0.778 [0.764, 0.791]	0.814 [0.804, 0.825]	0.778 [0.764, 0.791]
+RN	0.770 [0.758, 0.783]	0.788 [0.778, 0.797]	0.770 [0.758, 0.782]
+RN+NR	0.837 [0.827, 0.846]	0.857 [0.844, 0.870]	0.837 [0.827, 0.846]
+entropy groups ($K=3$)	0.875 [0.868, 0.882]	0.905 [0.900, 0.909]	0.875 [0.868, 0.882]
+discount groups ($K=9$): HE-FastSAC	0.887 [0.882, 0.892]	0.917 [0.913, 0.921]	0.887 [0.882, 0.892]
+deploy-group selection	0.901 [0.896, 0.905]	0.927 [0.923, 0.930]	0.901 [0.896, 0.905]

Table 7: Per-task final score on the 56 tasks. Mean over 5 seeds of the eight arms of Table 5 and of the group with the highest mean final score per task, which is selected on the evaluations it is scored on and is a bound, not a comparable arm. Bold marks the largest value among the eight arms in a row.

Task	FastTD3	FlashSAC	Ladder of Table 5						Best group
			FastSAC	+RN	+RN+NR	+K=3	HE-FastSAC	+selection	(bound)
DMC									
AcrobotSwingup	0.14	0.35	0.34	0.34	0.40	0.41	0.40	**0.55**	0.55
AcrobotSwingupSparse	0.06	0.14	0.17	0.17	0.21	0.22	0.30	**0.38**	0.38
BallInCup	0.00	0.84	0.44	0.40	0.72	0.77	0.89	**0.90**	0.91
CartpoleBalance	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00
CartpoleBalanceSparse	0.99	**1.00**	0.91	0.87	0.96	**1.00**	**1.00**	0.97	1.00
CartpoleSwingup	0.86	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	0.87
CartpoleSwingupSparse	0.14	0.71	0.43	0.57	0.74	0.76	**0.84**	0.77	0.84
CheetahRun	0.89	0.90	0.90	0.88	**0.93**	**0.93**	**0.93**	**0.93**	0.93
FingerSpin	0.80	0.91	0.74	0.73	0.78	0.87	0.89	**0.98**	0.98
FingerTurnEasy	**0.98**	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
FingerTurnHard	**0.98**	0.95	0.97	0.96	0.96	0.97	0.96	0.96	0.96
FishSwim	0.26	0.47	0.56	0.56	0.56	0.57	0.58	**0.71**	0.71
HopperHop	0.01	0.44	0.36	0.35	**0.47**	**0.47**	0.39	**0.47**	0.47
HopperStand	0.65	0.95	0.93	0.94	0.95	0.95	**0.96**	**0.96**	0.96
HumanoidRun	0.16	0.37	0.56	0.63	0.70	0.75	0.80	**0.83**	0.83
HumanoidStand	0.83	0.56	0.94	0.67	0.98	**1.00**	**1.00**	**1.00**	1.00
HumanoidWalk	0.46	0.59	0.77	0.97	**0.99**	0.89	**0.99**	**0.99**	0.99
PendulumSwingup	0.53	0.81	0.83	0.81	0.82	0.83	0.83	**0.84**	0.84
PointMass	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	0.90
ReacherEasy	0.98	0.98	0.98	0.98	**0.99**	0.98	**0.99**	**0.99**	0.99
ReacherHard	**0.98**	0.95	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.98
SwimmerSwimmer6	0.58	0.51	0.53	0.48	0.50	0.52	0.62	**0.67**	0.66
WalkerRun	0.90	**0.91**	0.83	0.83	0.82	0.86	0.83	0.89	0.89
WalkerStand	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	0.99
WalkerWalk	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.98
Joystick									
G1JoystickFlatTerrain	0.94	**1.03**	0.86	0.87	0.93	0.92	0.98	0.98	0.98
G1JoystickRoughTerrain	0.84	**0.87**	0.73	0.74	0.78	0.80	0.85	0.85	0.85
T1JoystickFlatTerrain	1.00	**1.07**	0.95	0.98	1.00	0.97	1.03	1.04	1.04
T1JoystickRoughTerrain	0.90	0.91	0.87	0.90	0.91	0.90	0.93	**0.94**	0.94
IsaacLab									
Lift Cube Franka	0.76	0.93	0.74	0.90	0.93	0.94	**0.97**	**0.97**	0.98
Open Drawer Franka	0.82	0.39	0.89	0.88	0.97	0.97	**0.99**	**0.99**	0.99
Repose Cube Allegro	0.68	**0.92**	0.73	0.67	0.77	0.75	0.86	0.86	0.86
Repose Cube Shadow	0.96	**1.06**	0.87	0.79	0.80	0.83	0.92	0.95	0.95
Flat Anymal C	0.73	0.87	0.85	0.86	**0.88**	0.87	**0.88**	**0.88**	0.88
Flat Anymal D	0.59	**0.88**	0.86	0.87	**0.88**	**0.88**	**0.88**	**0.88**	0.88
Flat G1	0.71	**0.93**	0.87	0.86	0.89	0.88	0.91	0.91	0.91
Flat H1	0.77	**0.97**	0.95	0.95	0.95	0.94	0.95	0.96	0.96
Rough Anymal C	0.34	0.55	0.42	0.44	0.56	0.60	**0.64**	**0.64**	0.64
Rough Anymal D	0.35	0.58	0.51	0.48	0.60	0.63	0.67	**0.68**	0.68
Rough G1	0.76	1.05	0.59	0.66	0.88	1.01	1.08	**1.09**	1.09
Rough H1	0.44	0.69	0.52	0.51	0.64	0.71	**0.76**	**0.76**	0.76
HumanoidBench									
balance hard	0.30	0.99	0.52	0.57	0.91	**1.05**	1.04	1.01	1.04
basketball	0.43	0.46	0.57	0.38	0.54	**0.75**	0.64	0.65	0.65
bookshelf simple	0.38	0.20	0.40	**0.43**	0.41	**0.43**	**0.43**	**0.43**	0.43
hurdle	1.09	**1.30**	0.73	0.89	1.08	1.14	1.23	1.26	1.26
push	1.08	0.32	1.00	1.16	1.18	1.34	**1.36**	1.32	1.37
room	0.42	**0.49**	0.04	0.06	0.09	0.27	0.31	0.31	0.31
run	1.30	1.31	1.27	1.22	1.30	1.31	1.32	**1.33**	1.33
sit hard	1.19	1.25	1.08	1.07	1.21	**1.27**	**1.27**	1.26	1.28
stair	0.92	**1.34**	0.81	0.75	0.86	1.05	0.54	0.55	0.55
walk	1.34	**1.36**	1.24	1.28	1.21	1.31	1.33	**1.36**	1.36
window	0.94	0.05	1.04	1.08	1.08	1.18	**1.23**	1.21	1.23
ManiSkill									
LiftPegUpright	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00
PegInsertionSide	0.00	0.93	0.99	0.99	**1.00**	0.99	0.99	0.99	1.00
PickSingleYCB	0.15	0.68	0.76	0.07	0.46	0.87	0.85	**0.89**	0.89
PullCube	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00

Table 8: Per-task AUC on the 56 tasks. Mean over 5 seeds of the eight arms of Table 5 and of the group with the highest mean final score per task, which is picked by final score and is not a bound on AUC; bold marks the largest value among the eight arms in a row.

Task	FastTD3	FlashSAC	Ladder of Table 5						Best group
			FastSAC	+RN	+RN+NR	$+K{=}3$	HE-FastSAC	+selection	(by final score)
DMC									
AcrobotSwingup	0.10	0.28	0.29	0.29	0.32	0.33	0.32	**0.43**	0.43
AcrobotSwingupSparse	0.03	0.07	0.11	0.08	0.14	0.14	0.17	**0.22**	0.22
BallInCup	0.00	0.80	0.23	0.26	0.71	0.73	0.81	**0.85**	0.69
CartpoleBalance	0.96	0.98	0.98	0.98	**0.99**	**0.99**	**0.99**	**0.99**	0.99
CartpoleBalanceSparse	**0.95**	0.94	0.91	0.88	0.93	0.91	0.94	0.91	0.94
CartpoleSwingup	0.81	0.82	0.81	0.82	**0.84**	**0.84**	**0.84**	**0.84**	0.84
CartpoleSwingupSparse	0.09	0.61	0.43	0.42	0.59	0.63	**0.71**	0.68	0.71
CheetahRun	0.74	0.84	0.84	0.83	0.85	0.87	**0.88**	**0.88**	0.88
FingerSpin	0.60	0.75	0.66	0.66	0.73	0.76	0.78	**0.88**	0.90
FingerTurnEasy	**0.93**	0.89	0.92	0.91	**0.93**	**0.93**	0.91	0.91	0.91
FingerTurnHard	0.88	0.87	**0.90**	0.89	0.89	**0.90**	0.88	0.88	0.89
FishSwim	0.24	0.39	0.49	0.50	0.50	0.51	0.50	**0.62**	0.62
HopperHop	0.00	0.33	0.19	0.24	**0.44**	0.40	0.35	0.41	0.41
HopperStand	0.47	0.81	0.78	0.77	0.91	**0.92**	**0.92**	**0.92**	0.92
HumanoidRun	0.15	0.13	0.37	0.43	0.56	0.58	0.57	**0.61**	0.60
HumanoidStand	0.66	0.24	0.74	0.70	0.86	**0.90**	0.89	0.89	0.90
HumanoidWalk	0.33	0.39	0.55	0.78	**0.90**	0.81	0.89	0.88	0.89
PendulumSwingup	0.48	0.42	0.60	0.53	0.73	0.77	0.80	**0.81**	0.82
PointMass	**0.88**	0.85	0.87	0.87	**0.88**	**0.88**	**0.88**	**0.88**	0.88
ReacherEasy	0.93	0.95	0.94	0.95	0.95	0.95	**0.96**	**0.96**	0.96
ReacherHard	0.91	0.94	0.93	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	0.95
SwimmerSwimmer6	0.52	0.43	0.45	0.46	0.48	0.48	0.52	**0.57**	0.57
WalkerRun	**0.82**	**0.82**	0.77	0.78	0.77	0.79	0.75	0.81	0.81
WalkerStand	**0.97**	0.96	0.96	0.96	**0.97**	**0.97**	**0.97**	**0.97**	0.97
WalkerWalk	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	0.95
Joystick									
G1JoystickFlatTerrain	0.81	0.81	0.69	0.75	0.81	0.83	0.86	**0.87**	0.85
G1JoystickRoughTerrain	0.72	0.69	0.60	0.64	0.69	0.71	**0.75**	**0.75**	0.74
T1JoystickFlatTerrain	0.91	0.91	0.85	0.91	0.92	0.90	0.96	**0.97**	0.97
T1JoystickRoughTerrain	0.76	0.76	0.77	0.82	0.84	0.82	0.85	**0.86**	0.86
IsaacLab									
Lift Cube Franka	0.42	0.74	0.67	0.86	0.86	0.87	0.88	**0.89**	0.89
Open Drawer Franka	0.73	0.40	0.85	0.78	0.92	0.92	**0.94**	**0.94**	0.94
Repose Cube Allegro	0.41	0.54	0.48	0.46	0.54	0.51	**0.57**	**0.57**	0.57
Repose Cube Shadow	**0.68**	0.63	0.67	0.60	0.62	0.63	0.66	**0.68**	0.68
Flat Anymal C	0.42	0.79	0.79	0.82	**0.84**	0.83	**0.84**	**0.84**	0.84
Flat Anymal D	0.34	0.77	0.79	0.83	0.84	0.84	**0.85**	**0.85**	0.84
Flat G1	0.50	0.78	0.76	0.77	0.80	0.79	**0.82**	**0.82**	0.83
Flat H1	0.61	0.84	0.87	0.87	**0.89**	0.86	0.88	0.88	0.88
Rough Anymal C	0.26	0.47	0.36	0.37	0.47	0.54	0.56	**0.57**	0.57
Rough Anymal D	0.26	0.49	0.43	0.41	0.51	0.57	0.59	**0.61**	0.61
Rough G1	0.46	0.77	0.40	0.45	0.69	0.84	**0.90**	**0.90**	0.88
Rough H1	0.23	0.51	0.38	0.36	0.48	0.60	**0.64**	**0.64**	0.64
HumanoidBench									
balance hard	0.19	0.79	0.49	0.56	0.83	**0.87**	0.82	0.83	0.82
basketball	0.28	0.27	0.41	0.24	0.32	**0.50**	0.37	0.37	0.38
bookshelf simple	0.33	0.21	**0.37**	0.35	0.34	0.34	**0.37**	0.36	0.36
hurdle	0.55	0.61	0.30	0.43	0.53	0.53	0.69	**0.70**	0.70
push	0.57	0.16	0.77	0.87	0.78	**1.01**	0.99	**1.01**	1.04
room	0.21	**0.31**	0.04	0.04	0.06	0.11	0.14	0.13	0.13
run	0.56	0.49	0.55	0.71	0.55	0.64	**0.81**	**0.81**	0.82
sit hard	0.97	0.99	0.98	1.01	0.93	1.01	**1.11**	**1.11**	1.12
stair	0.50	**0.68**	0.50	0.44	0.49	0.66	0.30	0.34	0.33
walk	0.80	0.63	0.86	0.87	0.77	0.79	**0.89**	**0.89**	0.89
window	0.73	0.17	0.83	0.88	0.96	1.06	**1.07**	**1.07**	1.07
ManiSkill									
LiftPegUpright	0.71	0.87	0.79	0.79	0.88	0.89	**0.95**	**0.95**	0.95
PegInsertionSide	0.00	0.66	0.67	0.65	**0.71**	0.56	0.68	0.67	0.68
PickSingleYCB	0.06	0.53	0.41	0.04	0.24	0.54	**0.66**	**0.66**	0.67
PullCube	0.93	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	0.95

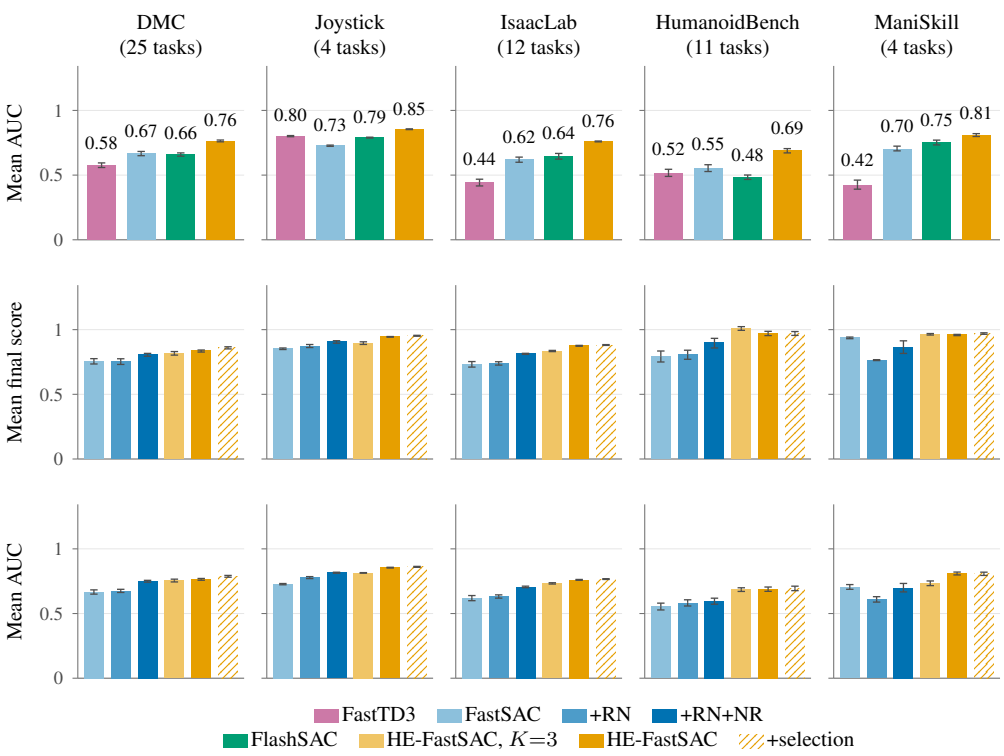

Figure 6: Per-suite scores. Top: AUC of the four methods of Figure 2, whose bottom row shows their final score; middle and bottom: final score and AUC of the six rungs of Figure 3. Mean, 5 seeds, 95% stratified bootstrap confidence intervals. HE-FastSAC has the highest AUC on every suite; in final score the entropy step carries HumanoidBench and ManiSkill, the discount step joystick and IsaacLab.

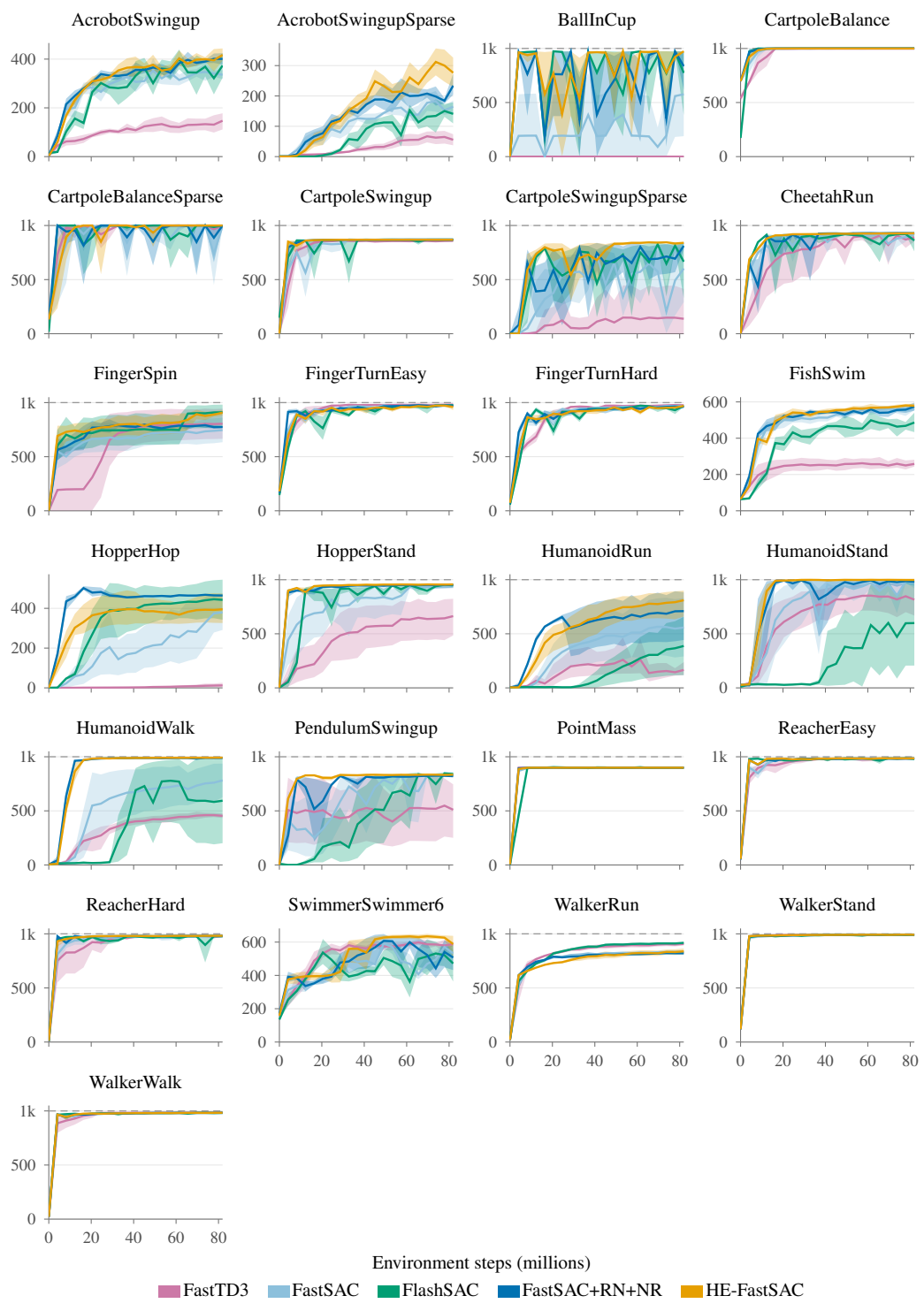

Figure 7: Per-task learning curves, MuJoCo Playground DMC (25 tasks). Mean return over 5 seeds with 95% bootstrap confidence intervals over seeds; HE-FastSAC at its default group. The dashed gray line is the reference score of the normalization. Returns below zero are not shown.

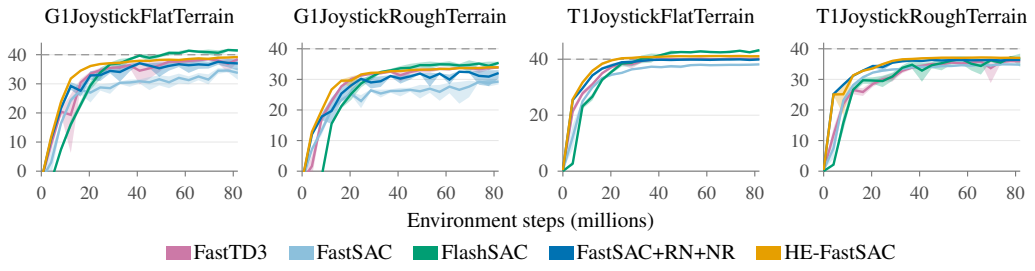

Figure 8: Per-task learning curves, MuJoCo Playground joystick (4 tasks), as in Figure 7.

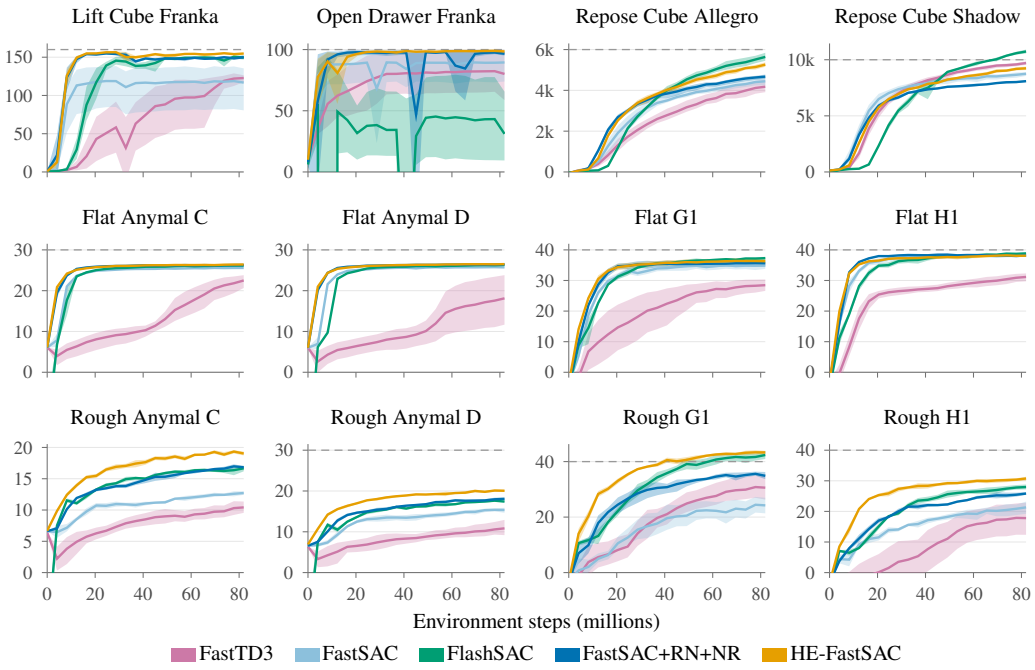

Figure 9: Per-task learning curves, IsaacLab (12 tasks), as in Figure 7.

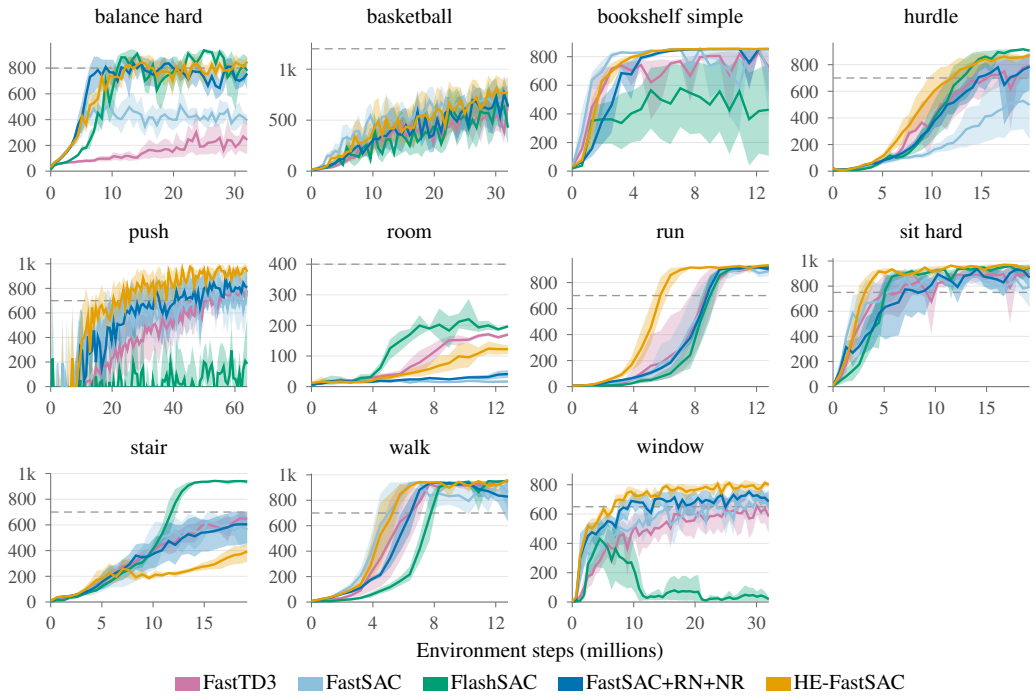

Figure 10: Per-task learning curves, HumanoidBench (11 tasks), as in Figure 7.

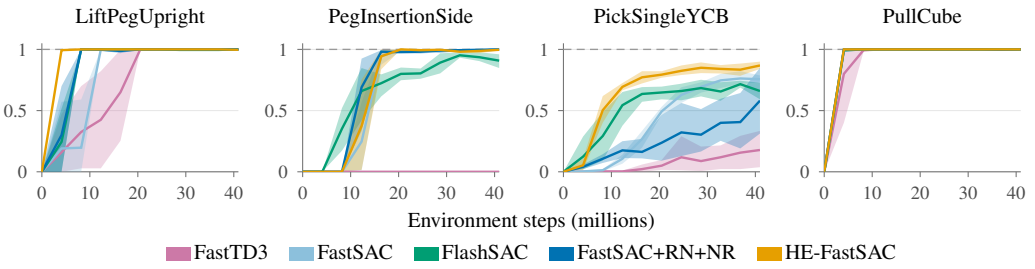

Figure 11: Per-task learning curves, ManiSkill (4 tasks), as in Figure 7.

D DEPLOY-GROUP SELECTION

The in-sample bound picks, per task, the group with the highest mean final score over the five seeds and reports it on those same evaluations. It bounds any rule that deploys one group per task and is not a comparable arm. The rule of Section 4.3 may switch groups between evaluations, so the bound does not bound it in principle; pooled over the 56 tasks the rule stays 0.004 below it in mean final score. The rule's most frequent pick is the group with the highest mean final score on 37 of the 56 tasks (Table 9).

Table 9: Deploy-group selection per suite. For each group of the grid, the number of tasks on which it has the highest mean final score (best) and the share of run-evaluations at which the rule of Section 4.3 deploys it over the last three evaluations (rule). Groups are labeled by target entropy in units of $|\mathcal{A}|$ and discount factor; task counts in parentheses.

Group $(\bar{\mathcal{H}}/\lvert\mathcal{A}\rvert, \gamma)$	DMC (25)		Joystick (4)		IsaacLab (12)		HumanoidBench (11)		ManiSkill (4)	
	best	rule	best	rule	best	rule	best	rule	best	rule
$(+0.25, 0.97)$	1	9%	–	–	–	2%	–	3%	2	57%
$(+0.25, 0.99)$	3	9%	–	–	2	7%	2	20%	–	7%
$(+0.25, 0.995)$	2	5%	–	–	–	2%	2	19%	–	3%
$(0, 0.97)$	1	4%	–	22%	2	16%	–	4%	–	–
$(0, 0.99)$	3	7%	–	15%	1	23%	2	25%	–	5%
$(0, 0.995)$	–	5%	–	–	1	9%	5	22%	1	3%
$(-1, 0.97)$	3	11%	–	2%	2	16%	–	2%	–	8%
$(-1, 0.99)$	4	26%	4	50%	1	5%	–	1%	1	7%
$(-1, 0.995)$	8	24%	–	12%	3	20%	–	4%	–	10%

E TIMING AND IMPLEMENTATION

We measure throughput as environment steps per second over all parallel environments, on one task per suite at the environment and update counts of the results runs: CartpoleSwingupSparse, G1JoystickFlatTerrain, Flat Anymal C, HumanoidBench run, and PegInsertionSide. Every run uses one NVIDIA A100-SXM4-40GB GPU with nothing else on the machine, and evaluation, video recording, and checkpointing are off. We take the median time per iteration over iterations 2500 to 3500 of a 3500-iteration run and report the mean over two or three passes. Batch sizes are each method's own. Collection and learning time are reported only for the arms of our implementation, whose timer synchronizes at the boundary between the two phases. On ManiSkill the cost of a simulation step depends on the policy, so totals are not comparable across methods there, and the ranges of Section 4.4 leave the suite out.

Table 10: Throughput in environment steps per second, and time per iteration in milliseconds spent in collection and learning for the arms of our implementation, on one task per suite and one NVIDIA A100-SXM4-40GB GPU. †On ManiSkill the simulation cost depends on the policy.

Method	DMC 4096 envs, 8 updates	Joystick 4096 envs, 8 updates	IsaacLab 4096 envs, 8 updates	HumanoidBench 128 envs, 2 updates	ManiSkill† 4096 envs, 8 updates
FastSAC, reference impl.	50.7k	34.4k	31.3k	2.4k	16.0k
FastSAC, our impl.	123.8k	60.2k	53.6k	4.6k	18.5k
collection (ms)	4.0	36.2	38.0	12.0	189.2
learning (ms)	29.1	31.8	38.5	16.1	32.0
FastSAC+RN+NR	117.5k	60.5k	54.2k	4.6k	19.6k
collection (ms)	4.6	36.6	37.5	12.5	176.0
learning (ms)	30.3	31.1	38.0	15.4	32.8
HE-FastSAC	106.2k	58.4k	50.4k	4.5k	23.2k
collection (ms)	4.8	35.8	39.8	11.8	141.3
learning (ms)	33.8	34.3	41.5	16.6	35.1
FlashSAC	66.6k	39.2k	37.7k	3.1k	17.1k
FastTD3	66.9k	23.8k	42.8k	2.8k	18.7k

Our implementation of FastSAC follows the reference implementation, Holosoma (Amazon FAR et al., 2025), and changes how the computation is executed. The critic heads are evaluated as one batched matrix multiplication over stacked weights instead of one network per head. The update

steps and the action step, which normalizes the observation and samples the action, are compiled with CUDA-graph capture. Gradient clipping uses foreach operations over parameter lists collected once, so that it is captured together with the update. The reference implementation's gradient scaler is removed: loss scaling is not needed under bf16, which has the exponent range of fp32, and the scaler's overflow check synchronizes with the host at every optimizer step. We checked that our implementation reproduces the learning curves of the reference implementation at identical hyperparameters.

F ANALYSIS DETAILS

The analysis tasks are CartpoleSwingupSparse, SwimmerSwimmer6, CheetahRun, and WalkerWalk from DMC, G1JoystickRoughTerrain and T1JoystickFlatTerrain from the joystick suite, Rough G1, Repose Cube Shadow, Rough Anymal C, and Flat Anymal C from IsaacLab, and run and hurdle from HumanoidBench. The sweep trains FastSAC+RN+NR at every configuration of the same 3×3 grid of target entropy and discount factor as the population, five seeds per configuration and task; the $(0, 0.99)$ configuration is the baseline of Section 4. A configuration is written as (target entropy in units of $|\mathcal{A}|$, discount factor). The independent networks of Section 5.3 have no conditioning input, since each network serves one group. The per-environment noise control multiplies the policy's pre-tanh standard deviation in the training rollouts by $e^{0.25}$, 1, or e^{-1}, cycling over the environments. These are the ratios that the target entropies $+0.25|\mathcal{A}|$, 0, and $-|\mathcal{A}|$ imply for a Gaussian policy; the update and the evaluation are unchanged. In the homogeneous-behavior arm every sampled transition is still relabeled to a random group, the evaluation partitions its environments across the groups, and the default group is deployed. The groups of HE-FlashSAC (Section 5.4) span FlashSAC's target noise scales of 0.2, 0.15, and 0.1, which correspond to target entropies of about $-0.19|\mathcal{A}|$, $-0.48|\mathcal{A}|$, and $-0.88|\mathcal{A}|$, and the discount factors 0.97, 0.99, and 0.995. Its deploy group is FlashSAC's default noise scale of 0.15 at discount 0.99. FlashSAC alone runs at its per-suite settings (Table 2); on DMC both arms use n-step 3.

Table 11: Pooled results on the 12-task analysis subset (Figures 4 and 5 and Section 5.4). Mean of the normalized AUC and final score, 5 seeds, 95% stratified bootstrap confidence intervals in brackets. The in-sample best configuration is selected on the evaluations it is scored on and is a bound, not a comparable arm.

Arm	Mean AUC	Mean final score
FastSAC+RN+NR $(0, 0.99)$	0.681 [0.674, 0.688]	0.870 [0.845, 0.887]
best per task, in-sample (bound)	0.767 [0.758, 0.777]	0.930 [0.924, 0.935]
best per task, leave-one-seed-out	0.757 [0.751, 0.764]	0.911 [0.890, 0.929]
per-environment noise scales	0.666 [0.654, 0.680]	0.858 [0.840, 0.876]
HE-FastSAC, entropy groups only (K=3)	0.714 [0.706, 0.722]	0.895 [0.884, 0.904]
HE-FastSAC, default group	0.769 [0.762, 0.778]	0.942 [0.938, 0.946]
+deploy-group selection	0.776 [0.768, 0.784]	0.948 [0.940, 0.954]
relabeling off (shared network, own data)	0.658 [0.648, 0.669]	0.807 [0.779, 0.832]
independent networks (shared data)	0.734 [0.727, 0.742]	0.904 [0.893, 0.915]
homogeneous behavior (training objectives only)	0.751 [0.736, 0.764]	0.912 [0.894, 0.927]
FlashSAC (per-suite discount factor)	0.683 [0.675, 0.691]	0.932 [0.920, 0.943]
HE-FlashSAC, deploy group	0.702 [0.696, 0.709]	0.949 [0.944, 0.953]

Table 12 lists every configuration alone and every group inside the population, the values behind Figure 4b to d. Table 13 gives the differences that Section 5 rests on, with their intervals and probabilities of improvement. Figure 12 shows the learning curves of the arms of Section 5.3 per task.

Table 12: Every configuration alone and every group inside the population on the analysis tasks. Mean final score and mean AUC of FastSAC+RN+NR trained alone at the group's target entropy and discount factor, of the group inside HE-FastSAC, and their difference, 5 seeds, 95% stratified bootstrap confidence intervals in brackets; the values behind Figure 4b to d. Groups are labeled by target entropy in units of $|\mathcal{A}|$ and discount factor.

Group $(\bar{\mathcal{H}}/\lvert\mathcal{A}\rvert, \gamma)$	Mean final score			Mean AUC		
	alone	inside	difference	alone	inside	difference
$(+0.25, 0.97)$	0.813	0.893	+0.080 [+0.059, +0.102]	0.676	0.717	+0.041 [+0.027, +0.055]
$(+0.25, 0.99)$	0.858	0.925	+0.067 [+0.051, +0.083]	0.671	0.751	+0.080 [+0.068, +0.092]
$(+0.25, 0.995)$	0.745	0.924	+0.179 [+0.158, +0.196]	0.605	0.746	+0.142 [+0.132, +0.152]
$(0, 0.97)$	0.879	0.910	+0.031 [+0.006, +0.061]	0.731	0.739	+0.008 [-0.005, +0.021]
$(0, 0.99)$	0.870	0.942	+0.072 [+0.054, +0.097]	0.681	0.769	+0.088 [+0.078, +0.099]
$(0, 0.995)$	0.716	0.943	+0.228 [+0.212, +0.245]	0.589	0.766	+0.177 [+0.164, +0.191]
$(-1, 0.97)$	0.857	0.892	+0.035 [+0.022, +0.048]	0.699	0.724	+0.026 [+0.013, +0.039]
$(-1, 0.99)$	0.798	0.922	+0.125 [+0.098, +0.157]	0.620	0.756	+0.136 [+0.122, +0.150]
$(-1, 0.995)$	0.641	0.934	+0.293 [+0.265, +0.316]	0.538	0.758	+0.220 [+0.208, +0.233]

Table 13: Differences behind Section 5 on the analysis tasks. Mean difference of the first arm minus the second with a 95% stratified bootstrap confidence interval, and the probability P that the first arm improves on the second, 5 seeds.

	Mean AUC		Mean final score	
Difference	value	P	value	P
HE-FastSAC − FastSAC+RN+NR	+0.088 [+0.078, +0.099]	0.93	+0.072 [+0.054, +0.097]	0.83
HE-FastSAC − best per task, in-sample (bound)	+0.002 [-0.011, +0.014]	0.41	+0.012 [+0.005, +0.019]	0.45
HE-FastSAC − best per task, leave-one-seed-out	+0.012 [+0.001, +0.023]	0.46	+0.031 [+0.012, +0.052]	0.46
+deploy-group selection − HE-FastSAC	+0.006 [-0.005, +0.018]	0.70	+0.006 [-0.003, +0.014]	0.73
entropy groups only (K=3) − FastSAC+RN+NR	+0.033 [+0.022, +0.044]	0.67	+0.025 [+0.004, +0.052]	0.57
per-environment noise scales − FastSAC+RN+NR	-0.014 [-0.029, +0.000]	0.43	-0.012 [-0.038, +0.018]	0.48
per-environment noise scales − entropy groups only	-0.048 [-0.062, -0.032]	0.39	-0.037 [-0.058, -0.016]	0.47
relabeling off − FastSAC+RN+NR	-0.023 [-0.035, -0.010]	0.61	-0.063 [-0.097, -0.028]	0.43
independent networks − FastSAC+RN+NR	+0.053 [+0.043, +0.064]	0.86	+0.034 [+0.013, +0.060]	0.76
independent networks − HE-FastSAC	-0.035 [-0.046, -0.024]	0.45	-0.038 [-0.049, -0.027]	0.43
homogeneous behavior − FastSAC+RN+NR	+0.070 [+0.053, +0.085]	0.84	+0.042 [+0.016, +0.071]	0.71
homogeneous behavior − HE-FastSAC	-0.018 [-0.036, -0.003]	0.43	-0.030 [-0.048, -0.015]	0.40
homogeneous behavior − independent networks	+0.017 [-0.001, +0.032]	0.53	+0.008 [-0.013, +0.027]	0.51
HE-FlashSAC − FlashSAC	+0.019 [+0.009, +0.030]	0.54	+0.017 [+0.005, +0.029]	0.63

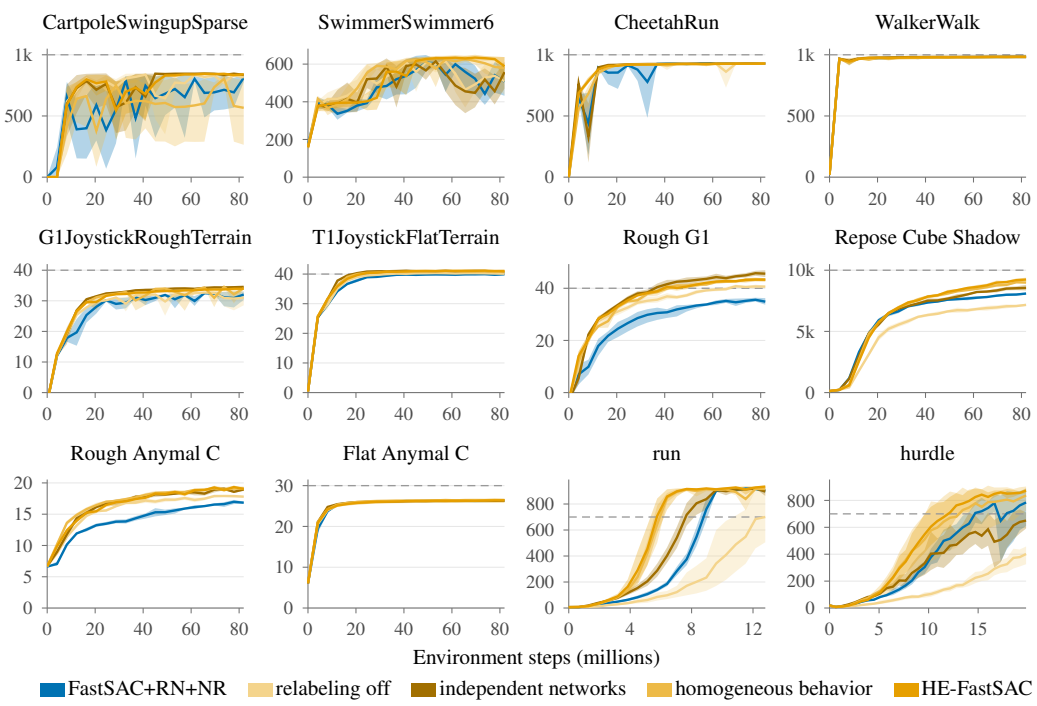

Figure 12: Per-task learning curves of the arms of Figure 5 on the 12 analysis tasks, as in Figure 7.