Assistax: A Multi-Agent Hardware-Accelerated Reinforcement Learning Benchmark for Assistive Robotics

Leonard Hinckeldey , Elliot Fosong , Elle Miller ,
Rimvydas Rubavicius , Trevor McInroe, Fan Zhang , Patricia Wollstadt ,
Stefano V. Albrecht , Subramanian Ramamoorthy

Keywords: Multi-Agent Reinforcement Learning, Ad-Hoc Teamwork, Zero-Shot Coordination, Assistive Robotics

arXiv:2507.21638v3 [cs.AI] 21 Jul 2026

Summary

We introduce Assistax, a benchmark for multi-agent reinforcement learning (MARL) and ad-hoc teamwork (AHT) in assistive robotics. Built entirely in JAX and MuJoCo MJX, it provides end-to-end hardware-accelerated RL training. To the best of our knowledge, Assistax is the first AHT benchmark for 3D continuous environments with two distinct embodied agents in an assistive context. We provide MARL baselines, a diverse population of pre-trained humanoid partners, and an AHT evaluation pipeline that reveals a coordination gap when existing RL algorithms face unseen partners with novel preferences.

Contribution(s)

1. A fully JAX and MuJoCo MJX-based implementation of five assistive robotics tasks within a scalable benchmark environment, together with a set of tuned MARL baselines that co-train a robot and humanoid agent to solve assistive tasks collaboratively. By leveraging hardware acceleration, Assistax achieves up to 412× faster open-loop simulation and 25× faster wall-clock training than comparable CPU-based environments, all on a single GPU.
   Context: JAX based RL environments have become increasingly popular, for MARL JaxMARL (Rutherford et al., 2024b) provide JAX reimplementations of many existing environment. Few benchmarks offer multi-agent 3D continuous-control (JaxMARL provide MaBrax) tasks at speeds competitive with other MARL environments. Assistax provides well-motivated tasks of real-world importance in simulation environments that run at the speed of grid-world environments, providing an extensible testbed for RL in assistive robotics.
2. An AHT baseline and training pipeline, including pre-training of diverse partner policies using our MARL implementations. These pre-trained partner policies are readily accessible on Hugging Face. We find that existing RL algorithms exhibit a coordination gap when confronted with unseen agents with novel preference combinations, though this gap closes significantly when training and test distributions are similar.
   Context: Some of the most popular AHT benchmarks are Overcooked (Carroll et al., 2019) and Hanabi (Bard et al., 2020). Neither of these provide a testbed which includes the challenges inherent in assistive robotics. Assistax provides a 3D continuous benchmarking suite that explicitly targets AHT within in an assistive context. We also provide thousands of pre-trained partner policies, for use by AHT and MARL researchers.

Assistax: A Multi-Agent Hardware-Accelerated Reinforcement Learning Benchmark for Assistive Robotics

Leonard Hinckeldey [1, †], **Elliot Fosong** [1, †], **Elle Miller** [1], **Rimvydas Rubavicius** [1], **Trevor McInroe**[1], **Fan Zhang** [2], **Patricia Wollstadt** [2], **Stefano V. Albrecht** [3], **Subramanian Ramamoorthy** [1]

{l.hinckeldey, e.fosong, elle.miller, rimvydas.rubavicius, t.mcinroe, s.ramamoorthy}@ed.ac.uk, patricia.wollstadt, fan.zhang@honda-ri.de

[1]**University of Edinburgh, Edinburgh, UK**
[2]**Honda Research Institute EU, Offenbach am Main, Germany**
[3]**DeepFlow, London, UK**

[†] indicating equal contribution

Abstract

As embodied autonomous systems capable of assisting humans in daily activities continue to be a major goal for the field of robotics, it becomes increasingly important to have efficient and appropriate reinforcement learning (RL) simulation testbeds. Many common RL environments are too simple to provide insight into complex robotics domains, and many robotics simulations have throughput that is too low for RL applications. In particular, very few robotic simulation environments target multi-agent interactions. While most simulators treat the robot as an isolated agent, real-world tasks such as home assistance and caretaking are inherently multi-agent. Assistax addresses these limitations by providing a high-throughput, scalable suite of GPU-accelerated assistive robotics tasks built on JAX and MuJoCo-MJX, while also including an active humanoid agent serving as a simulated human partner that can be trained alongside the robot using multi-agent RL (MARL). Beyond the application of Assistax as a MARL benchmark environment, we formulate the interaction between the human and the robot as an Ad-Hoc Teamwork (AHT) problem, where the robot's policy must generalise to unseen humans with varying disabilities and preferences. To this end, we provide an extensive AHT benchmarking pipeline: we use MARL to pre-train a diverse population of humanoid partners, and evaluate the ability of the robot policies to coordinate with a withheld set of humanoid policies. In contrast to other benchmarks, we also release reactive MARL-pre-trained humanoid policies via Hugging Face, enabling faster iteration in AHT research. By leveraging hardware acceleration, Assistax achieves up to 412$\times$ faster open-loop simulation than comparable CPU-based environments, all on a single GPU. Our AHT pipeline reveals a coordination gap when we evaluate existing RL algorithms on a set of unseen partners with novel preference combinations. This RL-native test suite for embodied multi-agent interaction provides a practical benchmark for advancing RL in assistive care. The code is readily available on GitHub.

1 Introduction

Assistive robotics (Chen et al., 2013; Savage, 2022) aims to develop autonomous systems that aid humans in performing daily activities, ranging from cleaning and cooking to caring for mobility-

Table 1: Assistax comparison with related benchmarks. Assistax provides hardware-accelerated, 3D, continuous, multi-agent environments alongside an ad-hoc teamwork (AHT) training pipeline.

Benchmarks	**Continuous actions**	**3D**	**Hardware acceleration**	**Multi-Agent**	**AHT**
Bi-DexHands (Chen et al., 2022)	✓	✓	✓	✓	✗
RLBench (James et al., 2020)	✓	✓	✗	✗	✗
Robotsuite (Zhu et al., 2020)	✓	✓	✗	✗	✗
Human-Robot Gym (Thumm et al., 2024)	✓	✓	✗	✓	✗
JaxRobotarium (Jain et al., 2025)	✓	✗	✓	✓	✗
Assistive Gym (Erickson et al., 2020)	✓	✓	✗	✓	✗
JaxMARL (Rutherford et al., 2024a)	✓	✓	✓	✓	✗
Multi-Agent Craftax (Omari et al., 2025)	✗	✗	✓	✓	✗
Assistax (ours)	✓	✓	✓	✓	✓

impaired individuals in tasks such as bathing and brushing their teeth. A key characteristic of these tasks is that the human is not merely a passive recipient but an active collaborator. In such scenarios, the robot must adapt to the human's behaviour as well as their preferences for how the robot should behave. This robot should also be capable of assisting any potential human; a robot capable of on-the-fly adaptation to any individual user would remove the need for explicit per-user training or fine-tuning.

Beyond dyadic interaction, many of the applications we target require the robot to co-inhabit an environment with multiple diverse agents. Consider a care home setting where the robot must attend not to a single user but to all inhabitants, while also adapting to a constantly changing roster of nurses, doctors, and visiting family members. Such scenarios are often formulated as Ad-Hoc Teamwork (AHT) (Stone et al., 2010), which requires the robot to coordinate with another agent without any prior experience of that agent. Unlike generalisation across static factors such as physical settings or user appearance, AHT specifically targets adaptation to the *policy* of the partner, a reactive, decision-making agent whose behaviour must be inferred on the fly.

There has been considerable interest in creating autonomous systems that can perform various activities of daily living (ADLs). Initial efforts to standardise research led to the development of frameworks like Assistive Gym (Erickson et al., 2020), which provides a physics-based environment for tasks such as dressing and bathing, and general-purpose manipulation platforms like robosuite (Zhu et al., 2020) that offer high-fidelity control for RL agents. Despite these advancements, few simulation environments actively model the multi-agent interaction between robot and user; for instance, many existing suites treat the human as a static or scripted entity (Thumm et al., 2024) rather than a reactive partner. Furthermore, even fewer such environments are optimised for modern RL research.

Recently, there has been increased interest in hardware-accelerated RL environments that allow for much higher throughput, reducing experiment turnaround to a matter of minutes. In particular, many well-known game and grid-world environments (Samvelyan et al., 2019; Lu et al., 2022) have been updated to take advantage of modern GPUs, in both RL and MARL (Rutherford et al., 2024b; Lange, 2022; Pignatelli et al., 2025). Similarly, robotics simulations are now leveraging hardware acceleration through tools such as IsaacLab (Mittal et al., 2025) and MuJoCo MJX (Todorov et al., 2012), giving rise to a number of accelerated robotics benchmarks, notably in manipulation and locomotion as seen in Bi-DexHands (Chen et al., 2022) and MuJoCo Playground (Zakka et al., 2025).

In light of these advancements, we introduce **Assistax** — a hardware-accelerated RL benchmark for assistive robotics. Assistax presents a combination of features that other benchmarks omit (see Table 1). Inspired by the recent success of JAX (Bradbury et al., 2018) in accelerating both RL environments and algorithms, we introduce a suite of MuJoCo MJX-based environments and a range of JAX-implemented algorithms, taking full advantage of hardware acceleration across the entire

RL training pipeline. This results in simulation speed-ups of up to 412× over the closely related, Assistive Gym (Erickson et al., 2020) environment. Assistax retains the co-trained humanoid agents of Assistive Gym but extends the evaluation paradigm: rather than testing only with co-trained partners, we evaluate whether robot policies generalise to previously unseen humans with varying preferences, framing the problem as AHT. Our contributions are:

1. A fully JAX and MuJoCo MJX-based implementation of five assistive robotics tasks.
2. A set of tuned MARL baselines that co-train a robot and humanoid agent to solve assistive tasks collaboratively.
3. AHT baselines and a training pipeline, including an accessible collection of pre-trained partner policies available on Hugging Face.

2 Background and Related Work

Hardware-Accelerated RL Benchmarks. Many environments leverage the JAX Python library (Bradbury et al., 2018) for easy parallelization via `vmap` (vectorization) and `pmap` (multi-device distribution) to take full advantage of hardware acceleration (Bettini et al., 2024; Richmond et al., 2023; Mittal et al., 2023). Additionally, MuJoCo's MJX enables collocating agents and environments on GPUs/TPUs, removing CPU-GPU memory transfers and enhancing performance through JAX's JIT compilation. Existing JAX-based RL environments include Gymnax (Lange, 2022), Pgx (Koyamada et al., 2023), Navix (Pignatelli et al., 2025), and MuJoCo-based benchmarks (Freeman et al., 2021; Zakka et al., 2025). JaxMARL (Rutherford et al., 2024b) specifically targets MARL, providing JAX reimplementations for a range of environments and achieving significant speed-ups. Beyond the frameworks mentioned, there are further hardware-accelerated simulation frameworks such as Madrona Engine (Shacklett et al., 2023; Kazemkhani et al., 2025) and PufferLib (Suarez, 2024).

Multi-Agent Reinforcement Learning. MARL considers multiple RL agents interacting and learning in an environment simultaneously (Albrecht et al., 2024). The Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016) is the canonical representation of the decision-making problem in cooperative MARL when agents share the same rewards. It is a tuple $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, T, p_0, \{\Omega^i\}_{i\in\mathcal{N}}, O, R, \gamma\rangle$ where: $\mathcal{N}$ is the set of agents; $\mathcal{S}$ is the state space; $\mathcal{A}^i$ is the action space for agent i; $T(s_{t+1} \mid s_t, a_t)$ is the state transition probability function; $p_0(s_0)$ is the initial state distribution; Ω^i is the observation space for agent i; $O(o_t \mid s_t, a_{t-1})$ is the observation probability function; $R(s_t, a_t)$ is the team reward function; and $\gamma \in [0, 1]$ is the discount factor. In a Dec-POMDP, at every time-step t all agents $i \in \mathcal{N}$ take an action a_t^i, each of these actions forms the joint action $a_t = (a_t^1, a_t^2, \ldots, a_t^{|\mathcal{N}|})$. After the action a_t is taken, the state transitions to the next state s_{t+1} given by the transition probability function $s_{t+1} \sim T(\cdot \mid s_t, a_t)$. Each agent receives an observation o_{t+1}^i given by the observation probability function $o_t \sim O(\cdot \mid s_t, a_{t-1})$ and receives the reward r_t given by the team reward function $R(s_t, a_t)$. Joint actions are sampled from the joint policy $\pi = (\pi^1, ..., \pi^{|\mathcal{N}|})$ i.e., $a_t \sim \pi(\cdot|h_t)$, where $h_t = (h_t^1, ..., h_t^{|\mathcal{N}|})$ is the joint history. The history of each agent $i \in \mathcal{N}$ consists of all observations and actions up until the current timestep $h^i = (o_0^i, a_0^i, o_1^i, a_1^i, \ldots o_t^i)$. The objective is to learn policies that maximize the discounted cumulative rewards over a finite/infinite horizon.

Ad-hoc Teamwork. AHT is the problem of controlling a single agent or a subset of agents within a team, to maximize the team's returns across a broad distribution of previously unknown teammate types (Mirsky et al., 2022). A teammate type refers to the distinct set of behavioural policies, capabilities, and preferences that define how a teammate operates and collaborates within a shared task (Albrecht & Stone, 2018; Stone et al., 2010). **Zero-Shot Coordination (ZSC)**, a special case of AHT, requires agents to coordinate with unseen partners without adaptation, typically generalizing from training with a limited set of partner policies (Hu et al., 2020). Existing popular benchmarks for AHT (e.g., Hanabi (Bard et al., 2020) and Overcooked (Carroll et al., 2019)) often lack the complexity

needed for assistive robotics, which involves continuous actions and multi-agent interactions in 3D environments. Assistax explicitly addresses these complexities.

3 Assistax

Assistax is an open-source python library providing 5 assistive robotics tasks, 3 baseline MARL implementations, an Ad-Hoc teamwork baselines, and thousands of pre-trained partner policies. All environments and algorithms are JAX-accelerated, providing a scalable end-to-end training pipeline.

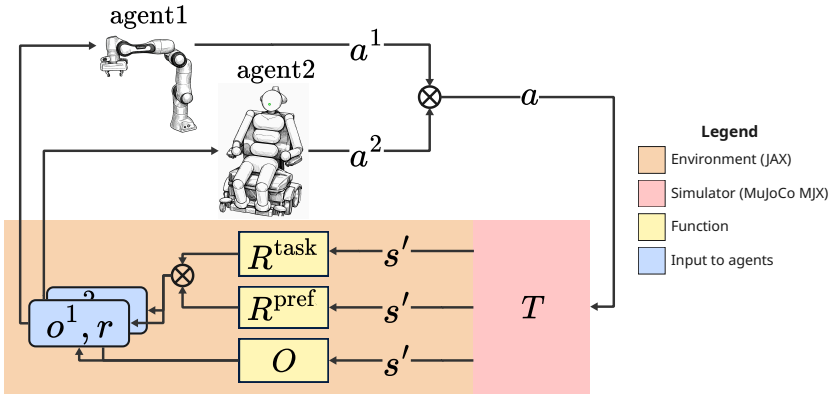
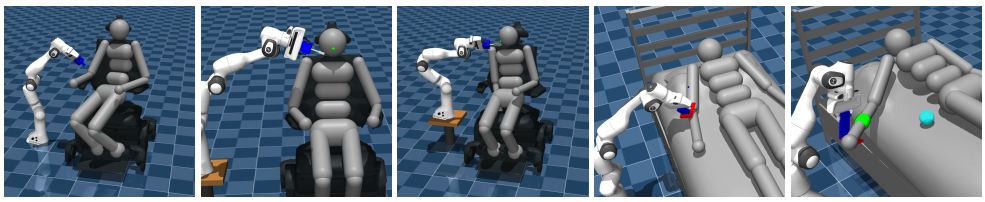


Figure 1: Overview of agent–environment interactions in Assistax. MuJoCo MJX handles state transitions $s' \sim T(\cdot|s, a)$; a JAX-based RL environment computes rewards $r = R(s, a)$ and per-agent observations $o^i \sim O(\cdot|s, a)$. When humanoid preferences are applied, preference rewards augment the task reward: $r = R^{\text{task}} + R^{\text{pref}}$. The entire loop can be arbitrarily parallelised with JAX across environments, seeds, hyperparameters, preference settings, and pre-trained humanoids.

3.1 Environments

The visualization of the Assistax environment suite is given in Figure 2. Inspired by Assistive Gym (Erickson et al., 2020), Assistax implements five environments based on real-world assistive robotic tasks involving collaboration between a human and a robot.



(a) *Scratch* (b) *Tooth Brushing* (c) *Feeding* (d) *Bed Bathing* (e) *Arm Assist*

Figure 2: Assistax environments

- *Scratch* (Figure 2(a)): A scratching target is randomly sampled on the surface of the humanoid's right arm. The robot must navigate its end-effector to this position and apply a contact force of 3 N near the target location, while moving its end-effector at 0.1 m/s to mimic a scratching motion. The scratching reward activates only when the end-effector is close to the target.
- *Tooth Brushing* (Figure 2(b)): The robot must navigate the toothbrush to the humanoid's mouth (shown in green in Figure 2(b)). Rewards are given for proximity to the mouth, alignment of the bristle side of the toothbrush with the mouth, and a brushing component: the Boltzmann product of applying 1 N of contact force with a tangential speed of 0.1 m/s. The brushing reward activates only when the toothbrush is within 0.05 m of the mouth.

- *Feeding* (Figure 2(c)): The robot must guide a spoon to the humanoid's mouth without spilling. Rewards are given for the distance of the spoon to the mouth, keeping the spoon upright while far from the mouth and tilting it towards the mouth when close, approaching at an appropriate speed, and applying 1 N of contact force to the mouth.
- *Bed Bath* (Figure 2(d)): The robot must wipe the humanoid's arm with a sponge. We distribute 52 target points evenly along the surface of the humanoid's arm. Rewards are given for the distance to the nearest unvisited target and a bonus for each newly visited target. A target counts as visited when the end-effector is within 0.05 M of it while making contact with the arm. The episode terminates early if all targets are visited.
- *Arm Assist* (Figure 2(e)): The robot uses a hook end-effector to lift and reposition the humanoid's weak right arm towards a waist-level target. Rewards are given for the distance between the hook and the wrist target (shown in green in Figure 2(e)), the distance between the humanoid's lower arm and the waist target (shown as the blue ball in Figure 2(e)), and the rotational alignment of the hook with the wrist.

All tasks use shared rewards between the robot and humanoid agents. Although the rewards target the robot's behaviour, the humanoid can collaborate by making targets more accessible, for example, lifting an arm to expose a scratching target or moving its mouth toward the spoon. Further details are provided in Appendix A.

Agents. Assistax trains agents that control either a robot or a simplified humanoid model, serving as a simulated human partner. For each task, we implement custom end-effectors, which do not have controllable joints. The robot is torque-controlled with 7 joints $\mathcal{A}^{\text{robot}} \in \mathbb{R}^7 := [-1, 1]^7$. The humanoid model is taken from the Brax humanoid tasks (Freeman et al., 2021) and is also torque-controlled. For the humanoid we restrict the action space to either the shoulder and elbow joints of the right arm $\mathcal{A}^{\text{human}} \in \mathbb{R}^3 := [-1, 1]^3$ in *Scratch*, *Bed Bath*, and *Arm Assist*; or the yaw and pitch of the humanoid head $\mathcal{A}^{\text{human}} \in \mathbb{R}^2 := [-1, 1]^2$ in *Tooth Brushing* and *Feeding*. Following Assistive Gym, in an attempt to simulate real-world challenges in assistive healthcare tasks, Assistax simulates tremors, joint weakness and limited range of motion as well as a range of preferences regarding how the task is carried out.

3.2 Preference Rewards

To encode a human's preferences for *how* the robot should perform the task, we augment each environment's task reward with a preference reward. Preferences are defined along three axes: 1. *force*—the preferred range of contact force the robot applies to the humanoid 2. *speed*—the preferred range of end-effector speed; and 3. *number of contacts*—a penalty for making new contacts after the initial touch, discouraging poking behaviour in favour of sustained contact (e.g. scratching). For force and speed, we specify a preferred range and a weighting reflecting how much the humanoid cares about that preference. For contact count, we specify a per-event penalty and an analogous weighting. The three components are normalised into a fixed preference reward budget, calibrated to be between 40% to 50% of the maximum attainable per-step reward; each component's share of this budget is proportional to its weighting. When using preference rewards we provide the preferences in the observations for both agents. For further details see Appendix A.3.

3.3 Multi-Agent Reinforcement Learning Baselines

For our MARL baselines we build upon JaxMARL (Rutherford et al., 2024b) for straightforward single-file implementations. We provide baselines for multi-agent variants of the widely-used proximal policy optimization (PPO) (Schulman et al., 2017) and soft-actor critic (SAC) (Haarnoja et al., 2018) algorithms, including independent (IPPO) and centralised critic (MAPPO, MASAC) variants. We also include GRU-based (Cho et al., 2014) recurrent variations for PPO-based algorithms. Using these baselines we co-train agents to complete the various tasks, i.e. maximizing the expected returns when interacting with the environment as shown in Figure 1.

3.4 Ad-Hoc Teamwork Baselines

Our AHT pipeline consists of utilising the basic Assistax training loop (as shown in Figure 1) to pre-train a diverse population of 630 humanoid agents per task. The diversity in the population is induced along three axes: 1. Over-fitting to **specialized behaviour conventions** which emerge through co-training of agents (Lupu et al., 2021); 2. varying the **preferences** of the humanoid when training the population; and 3. A variety of different **disability** setting for humanoids in the population.

When training the agent population we sample 610 preference combinations uniformly from a constrained set of possible preferences, for more details see the appendix A.3. We further use 7 distinct disability combinations targeting the mobility of the arm in the relevant tasks. We later discard the robot policies keeping only the pre-trained humanoid policies.

Assistax benchmarks AHT capabilities by considering a simple meta-learning baseline. Using our pre-trained population (Π) we create disjoint train Π^{train} and test Π^{test} sets sampled from Π. This means that only the robot is learning, while the humanoid follows it's pre-trained policy. Formally, we train the robot for the following objective (Rahman et al., 2024):

$$\pi^{i^*}(\Pi^{\text{train}}) = \arg\max_{\pi^i} \mathbb{E}_{\pi^{-i}\sim\mathbb{U}(\Pi^{\text{train}}),a_t^i\sim\pi_i,a_t^{-i}\sim\pi^{-i},T,O}\left[\sum_{t=0}^{\infty}\gamma^t R(a_t,s_t,a_{t+1})\right].$$

We then evaluate the measure $M_{\Pi^{\text{test}}}$ which measures the robustness of π^{i^*} when paired with unseen agents uniformly sampled from Π^{test}, defined as:

$$M_{\Pi^{\text{test}}} = \mathbb{E}_{\pi^{-i}\sim\mathbb{U}(\Pi^{\text{test}}),a_t^i\sim\pi^{i^*}(\Pi^{\text{train}}),a_t^{-i}\sim\pi^{-i},T,O}\left[\sum_{t=0}^{\infty}\gamma^t R(a_t,s_t,a_{t+1})\right].$$

Assistax provides the parameters of pre-trained partner policies via Hugging Face, allowing the benchmark users to recreate zero-shot coordination results but also to explore other AHT methods.

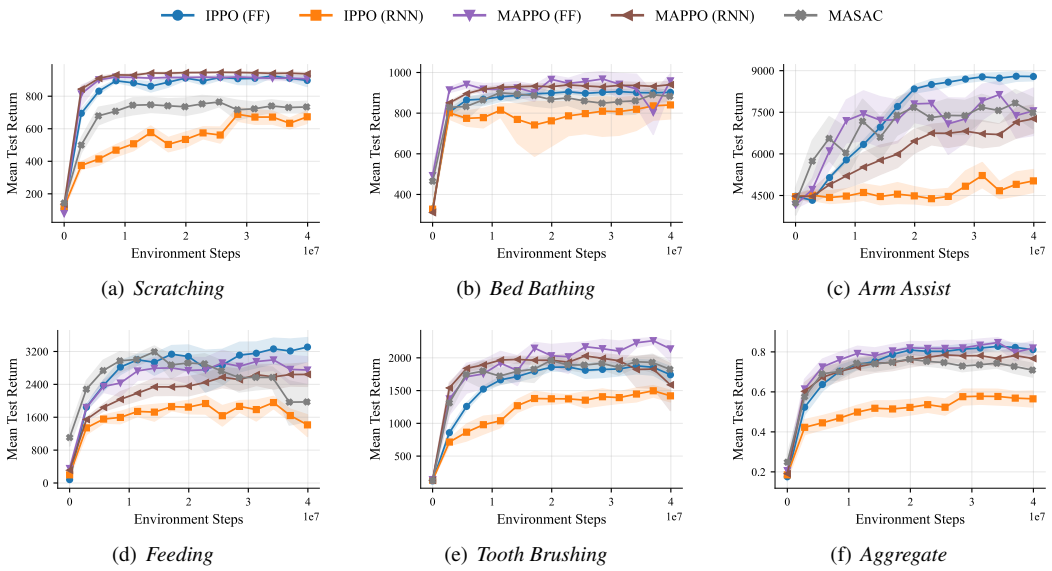
3.5 Optimized computation

Assistax prioritises simulation efficiency over high fidelity, a trade-off currently necessary as MuJoCo MJX scales poorly with the number of collisions, resulting in significant slowdowns when simulating deformable bodies such as cloth or detailed meshes. We thus implement solely rigid-body environments using primitive geometries (e.g. capsules for the Franka arm, boxes for wheelchairs and beds) and selectively disable collisions between geometries unlikely to interact during an episode.[1] While soft-body simulation would be preferable for assistive tasks, Assistax is primarily aimed at researchers in MARL and AHT, who presently often rely on even more simplified settings (Gorsane et al., 2022); the growing popularity of JAX-accelerated environments in these communities (Rutherford et al., 2024b) reflects a desire for high-throughput training that enables fast development and rigorous evaluation. See Figure 7 in Appendix A.2 for further details on collision geometries.

4 Results

In this section we show the two main results: 1. The MARL performance of robot-humanoid teams co-trained in accordance with Figure 1. 2. The AHT setting where the robot learns to complete the task together with a pre-trained humanoid, this policy is trained on a limited train set and evaluated against withheld humanoids, showcasing the zero-shot coordination capabilities of our baselines. For each MARL baseline we conduct extensive hyperparameter tuning, searching over 168 hyperparameter combinations for each baseline–environment pair (without using preference rewards). We use use the

[1]These defaults can be adjusted to increase physical fidelity by altering the library's MuJoCo XML files.



(a) *Scratching* (b) *Bed Bathing* (c) *Arm Assist*

(d) *Feeding* (e) *Tooth Brushing* (f) *Aggregate*

Figure 3: Learning curves for MARL baselines. Each curve shows mean test returns across 16 Seeds and 64 evaluation episodes, and ± 95% stratified-bootstrap confidence intervals over those 16 seeds. We also provide Min-Max normalised mean test returns across all environments (see 3(f)).

most performant hyperparameter combination found by our tuning, for all our baselines including AHT i.e. using IPPO for the PPO baseline and MASAC values for the SAC baselines. For further details on hyperparameters see Appendix C.

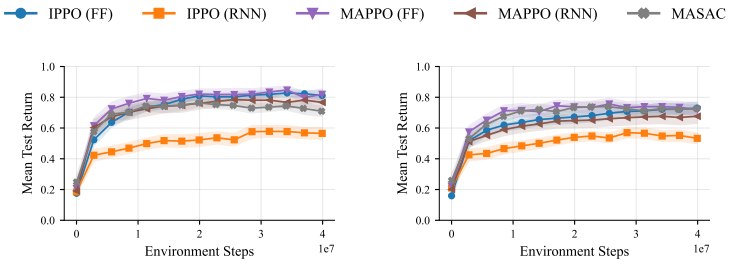
Table 2: Per-episode reward upper bounds for each environment. Task return reflects the task objective alone; the combined column adds the preference reward budget (40–50% of the overall reward). These bounds are not achievable in practice but serve as a reference for the results in this section.

Environment	**Task Return Max**	**Combined Max (+ Pref.)**
Scratching	1,000	2,000
Bed Bathing	1,156	2,312
Arm Assist	11,280	15,792
Feeding	4,700	6,580
Tooth Brushing	3,140	4,396

4.1 MARL

We provide evaluation for four baseline MARL algorithms across all 5 Assistax tasks. We evaluate MASAC, MAPPO and IPPO, each with no parameter sharing (Christianos et al., 2021). For MAPPO and IPPO we also provide recurrent baselines.

Figure 3 shows that both IPPO and MAPPO feed-forward implementations consistently rank among the top-performing baselines, with MASAC also performing well in many environments. Despite the partial observability of the environments, recurrent architectures for PPO-based baselines do not outperform their feed-forward counterparts, and actively degrade performance in the case of IPPO. This is in line with results from MABrax (Rutherford et al., 2024b), a JAX-accelerated MAMuJoCo benchmark, where feed-forward policies also tend to perform better (Tessera et al., 2026).



(a) *Without Preferences* (b) *With Preferences*

Figure 4: Min-Max normalised test returns and 95% stratified bootstrapped CI across 16 seeds and all environments. This compares performance with and without additional preference rewards.

We also compare min-max normalised mean test returns across all tasks comparing the performance with and without preference rewards (Figure 4). We do not note any meaningful difference between in performance of baselines in the two settings. For a more details see Figure 11 in Appendix B.

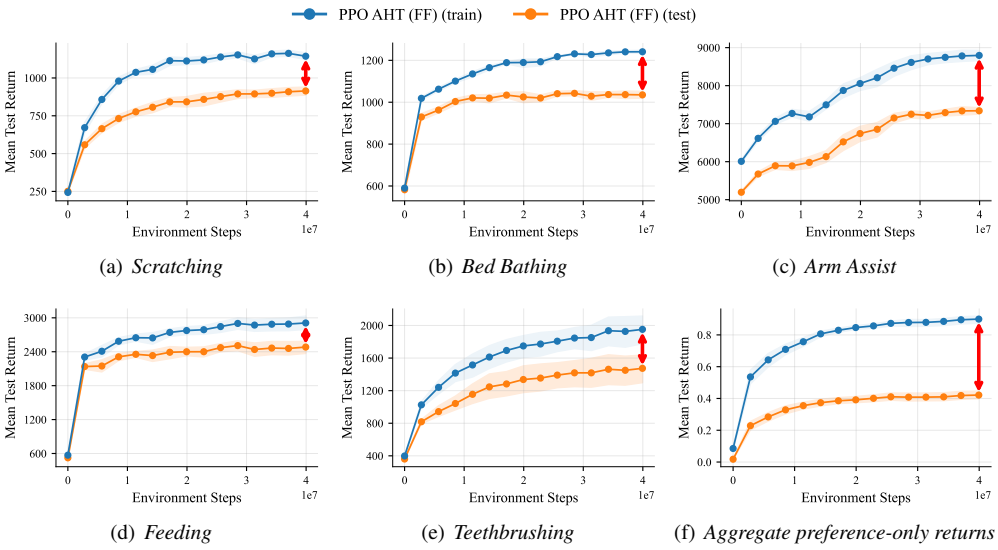
4.2 Ad Hoc Teamwork

Assistax can also be used as an AHT benchmark, to test the ability of the robot agent to adapt to previously-unseen humanoid partners. We provide simple baselines following the metalearning paradigm identified in Mirsky et al. (2022)—that is, we use a common RL algorithm (PPO or SAC) to train a robot policy with humanoid policies sampled from a training population. This training procedure follows the same principle used to evaluate zero-shot generalisation of RL algorithms in benchmarks like ProcGen (Cobbe et al., 2020), or the domain randomisation techniques employed for sim-to-real transfer in robotics (Tobin et al., 2017): train and test the algorithm against two distinct sets of partners—a training set and a held-out test set respectively. To ensure these ad-hoc teamwork benchmarks can be consistently used by other researchers, we make the training and test partner policies available via Hugging Face.

For the results shown in Figure 5, we train against a population of 5 agents and test against 625 unseen agents. We sort agents and select the 5 agents with the highest speed preference weighting. This sampling strategy is deliberately challenging: it minimises the overlap between train and test sets and makes it more likely that the test set includes out-of-distribution preferences. Note that the coordination gap drops significantly when including more agents in the training set or when sampling train and test agents uniformly from the population. In this hard setting, we observe a significant gap in performance that is almost entirely attributable to poor adaptation to new preferences. In Figure 5(f), we show the min-max normalised mean preference returns (i.e. excluding task rewards). A more detailed breakdown can be found in the Appendix B.3.

4.3 Runtime Experiments

A key benefit of implementing the environment in JAX is the significant speed-up in RL training pipelines. To contextualise the results presented here, a typical Assistax IPPO training run of 40 million environment timesteps takes roughly 20 minutes with 1024 parallel environments (see Appendix A.4); the equivalent run in Assistive Gym using RLlib multiprocessing (Liang et al., 2018) takes approximately 8.3 hours. This yields a roughly 25$\times$ wall-clock reduction, though we note this comparison is indicative rather than controlled—many factors beyond raw simulation speed, such as the RLlib parallelisation strategy used in Assistive Gym, affect end-to-end training time. Figure 6 shows how steps per second scale with the number of vectorised environments. Table 3 reports open-loop simulation speeds compared with Assistive Gym, providing an upper bound on potential speed-ups. Beyond environment throughput, JAX also enables vectorisation across training runs, seeds, and hyperparameter settings, making sweeps and rigorous evaluations far more efficient.



(a) *Scratching* (b) *Bed Bathing* (c) *Arm Assist*

(d) *Feeding* (e) *Teethbrushing* (f) *Aggregate preference-only returns*

Figure 5: Ad-hoc teamwork performance of PPO across all five tasks. Mean returns and stratified 95% bootstrapped CI across 16 seeds. Red arrows highlight the coordination gap.

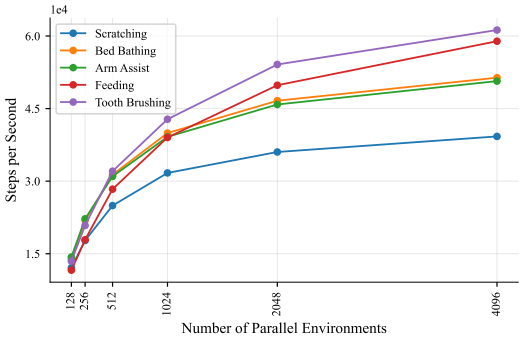


Figure 6: Steps per second and number of vectorized environments using a single A100 (40GB) across Assistax tasks.

Table 3: Open-loop simulation speed (steps per second, in hundreds) for Assistax with 1024 vectorised environments on an Nvidia A100 (40GB) GPU, averaged over 10 million timesteps. Relative speed-up is against Assistive Gym on a single AMD EPYC 7763 with no parallel environments.

	Assistax	**Assist. Gym**	**Speedup**
Scratching	316.98	2.28	139×
Bed Bathing	399.45	0.97	412×
Arm Assist	391.54	1.40	280×
Feeding	390.28	1.71	228×

5 Conclusion

We introduce Assistax, a hardware-accelerated MARL benchmark for assistive robotics. The presented task suite and experiments demonstrate that continuous, physics-based 3D environments can, from a computational-cost perspective, compete with simpler, game-like settings commonly used in RL, thereby enabling faster research iteration and more thorough evaluations. Beyond MARL, we provide an AHT benchmarking pipeline that offers a suitable testbed for advancing AHT in a range of assistive tasks. We release all environments, algorithms, and pre-trained partner policies to support reproducibility and future work. As assistive robotics grows in societal importance, we hope Assistax can provide a compelling benchmark to the RL community, bringing the computational efficiency of JAX to bear on the multi-agent challenges inherent in human-robot collaboration.

Limitations and Future Work. There is a trade-off between algorithm runtime and environment fidelity, which may come at the cost of accurately replicating real-world scenarios. Furthermore, MARL policies do not capture the full complexity of human behaviour; there thus remains a

considerable challenge in bridging the gap between high-throughput simulators and the real-world tasks we are targeting.

Acknowledgments

Work supported by a UKRI Turing AI World Leading Researcher Fellowship on AI for Person-Centred and Teachable Autonomy (grant EP/Z534833/1). This work was also supported by the Edinburgh International Data Facility (EIDF) and the Data-Driven Innovation Programme at the University of Edinburgh. Access to EIDF was facilitated through the University of Edinburgh's Generative AI Laboratory GAIL Fellow scheme.

We thank, Christiane Wiebel-Herboth for her contributions to the early stages of this project. We also thank Kale-ab Tessera and Ivalin Chobanov for their insightful discussions.

References

Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018. DOI: 10.1016/j.artint.2018.01.002.

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The Hanabi Challenge: A New Frontier for AI Research. *Artificial Intelligence*, 280:103216, 2020. DOI: 10.1016/j.artint.2019.103216.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. In *Distributed Autonomous Robotic Systems: 16th International Symposium (DARS 2022)*, volume 28 of *Springer Proceedings in Advanced Robotics*, pp. 42–56. Springer, 2024. DOI: 10.1007/978-3-031-51497-5_4.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the Utility of Learning about Humans for Human-AI Coordination. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. Curran Associates, Inc., 2019.

Tiffany L. Chen, Matei Ciocarlie, Steve Cousins, Phillip M. Grice, Kelsey Hawkins, Kaijen Hsiao, Charles C. Kemp, Chih-Hung King, Daniel A. Lazewatsky, Adam E. Leeper, Hai Nguyen, Andreas Paepcke, Caroline Pantofaru, William D. Smart, and Leila Takayama. Robots for humanity: using assistive robotics to empower people with disabilities. *IEEE Robotics & Automation Magazine*, 20 (1):30–39, 2013. DOI: 10.1109/MRA.2012.2229950.

Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen Marcus McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. 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.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–Decoder approaches. In Dekai Wu, Marine Carpuat,

Xavier Carreras, and Eva Maria Vecchi (eds.), *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. DOI: 10.3115/v1/W14-4012.

Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1989–1998. PMLR, 2021.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 2020.

Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Karen Liu, and Charles C. Kemp. Assistive Gym: A Physics Simulation Framework for Assistive Robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10169–10176. IEEE, 2020. DOI: 10.1109/ICRA40945.2020.9197411.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative MARL. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/249f73e01f0a2bb6c8d971b565f159a7-Abstract-Conference.html.

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)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-Play" for Zero-Shot Coordination. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4399–4410. PMLR, 2020.

Shalin Anand Jain, Jiazhen Liu, Siva Kailas, and Harish Ravichandar. Jaxrobotarium: Training and deploying multi-robot policies in 10 minutes. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of the 9th Conference on Robot Learning (CoRL 2025)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1975–1996. PMLR, 2025. URL https://proceedings.mlr.press/v305/jain25a.html.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. DOI: 10.1109/LRA.2020.2974707.

Saman Kazemkhani, Aarav Pandya, Daphne Cornelisse, Brennan Shacklett, and Eugene Vinitsky. GPUDrive: Data-driven, multi-agent driving simulation at 1 million FPS. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ERv8ptegFi.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45716–45743, 2023.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3053–3062. PMLR, 2018.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7204–7213. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/lupu21a.html.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V. Albrecht. A survey of ad hoc teamwork research. In Dorothea Baumeister and Jörg Rothe (eds.), *Multi-Agent Systems (EUMAS 2022)*, volume 13442 of *Lecture Notes in Computer Science*, pp. 275–293, Cham, 2022. Springer. DOI: 10.1007/978-3-031-20614-6_16.

Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. DOI: 10.1109/LRA.2023.3270034.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

Frans A. Oliehoek and Christopher Amato. The Decentralized POMDP Framework. In Frans A. Oliehoek and Christopher Amato (eds.), *A Concise Introduction to Decentralized POMDPs*, pp. 11–32. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28929-8. DOI: 10.1007/978-3-319-28929-8_2.

Bassel Al Omari, Michael Matthews, Alexander Rutherford, and Jakob Nicolaus Foerster. Multi-agent craftax: Benchmarking open-ended multi-agent reinforcement learning at the hyperscale, 2025. URL https://arxiv.org/abs/2511.04904.

Eduardo Pignatelli, Jarek Liesen, Robert Tjarko Lange, Chris Lu, Pablo Samuel Castro, and Laura Toni. Navix: Scaling minigrid environments with jax. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025), Datasets and Benchmarks Track*, 2025.

Arrasy Rahman, Elliot Fosong, Ignacio Carlucho, and Stefano V. Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=l5BzfQhROl.

Arrasy Rahman, Jiaxun Cui, and Peter Stone. Minimum coverage sets for training robust ad hoc teamwork agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17523–17530. AAAI Press, 2024. DOI: 10.1609/aaai.v38i16.29702.

Paul Richmond, Robert Chisholm, Peter Heywood, Mozhgan Kabiri Chimeh, and Matthew Leach. FLAME GPU 2: A framework for flexible and performant agent based simulation on gpus. *Softw. Pract. Exp.*, 53(8):1659–1680, 2023. DOI: 10.1002/SPE.3207. URL https://doi.org/10.1002/spe.3207.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schröder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert T. Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob N. Foerster. Jaxmarl: Multi-agent RL environments and algorithms in JAX. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/5aee125f052c90e326dcf6f380df94f6-Abstract-Datasets_and_Benchmarks_Track.html.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktaschel, Chris Lu, and Jakob Nicolaus Foerster. JaxMARL: Multi-Agent RL Environments and Algorithms in JAX. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2024), Datasets and Benchmarks Track*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/5aee125f052c90e326dcf6f380df94f6-Abstract-Datasets_and_Benchmarks_Track.html.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge, 2019. URL https://arxiv.org/abs/1902.04043.

Neil Savage. Robots rise to meet the challenge of caring for old people. *Nature*, 601:S8 – S10, 2022. URL https://api.semanticscholar.org/CorpusID:246064292.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

Brennan Shacklett, Luc Guy Rosenzweig, Zhiqiang Xie, Bidipta Sarkar, Andrew Szot, Erik Wijmans, Vladlen Koltun, Dhruv Batra, and Kayvon Fatahalian. An extensible, data-oriented architecture for high-performance, many-world simulation. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. DOI: 10.1145/3592427. URL https://doi.org/10.1145/3592427.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1504–1509, July 2010. ISSN 2374-3468, 2159-5399. DOI: 10.1609/aaai.v24i1.7529.

Joseph Suarez. Pufferlib: Making reinforcement learning libraries and environments play nice, 2024. URL https://arxiv.org/abs/2406.12905.

Kale-ab Tessera, Leonard Hinckeldey, Riccardo Zamboni, David Abel, and Amos Storkey. Probing dec-POMDP reasoning in cooperative MARL. In *The 25th International Conference on Autonomous Agents and Multi-Agent Systems*, 2026. URL https://openreview.net/forum?id=gSK8tR7du3.

Jakob Thumm, Felix Trost, and Matthias Althoff. Human-Robot Gym: Benchmarking Reinforcement Learning in Human-Robot Collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7405–7411. IEEE, 2024. DOI: 10.1109/ICRA57147.2024.10610705.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. DOI: 10.1109/IROS.2017.8202133.

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

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer., 2025. URL https://github.com/google-deepmind/mujoco_playground.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2020. URL https://arxiv.org/abs/2009.12293.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Benchmark Details

A.1 Observation Space.

A summary of the observations for each task is shown in Table 4. In our benchmark, we consider 3 types of observations: proprioception, tactile, and ground-truth information from the simulator. Proprioception is information relating to robot configuration. It is computed from the robot's internal sensors. Assistax considers tactile observation of the net contact forces between the end-effector and the humanoid arm, expressed as a force vector in the MuJoCo contact frame. Observations about the other body forces are not included. Ground truth refers to the privileged information available in simulation but requires estimation in the real world (e.g. end-effector to object distance, humanoid joint angles). Subscripts R and H denote robot and humanoid respectively. The end-effector is chosen to be an imaginary frame at the end of the robotic arm chosen. Note that *Arm Assist* has a larger observation space to take into account the increased task complexity. For this task, we provide the rotation matrix between the end-effector and the target on the humanoid arm (green in Figure 2(e)) to influence the robot to lift the arm in a particular way. We also provide an additional distance target for the second phase of bringing the arm back to the waist target position (blue in Figure 2(e)).

Table 4: Observations space overview. Assistax uses three types of observations: proprioception (prop.), tactile, and ground-truth (gt.) information from the simulator.

Type	**Symbol**	**Description**	**Dim.**	**Task**				
				Scratch	*Bed Bath*	*Arm Assist*	*Feeding*	*Teeth Brush*
prop.	θ_R	robot joint angles	7	✓	✓	✓	✓	✓
	$\dot{\theta}_R$	robot joint velocities	7	✓	✓	✓	✓	✓
	x_{ee}	end-effector position	3	✓	✓	✓	✓	✓
	q_{ee}	end-effector quaternion	4	✓	✓	✓	✓	✓
tactile	f_{ee}	end-effector force	3	✓	✓	✓	✓	✓
gt.	θ_H	humanoid joint angles	9	✓	✓	✓	✓	✓
	$\dot{\theta}_H$	humanoid joint velocities	9	✓	✓	✓	✓	✓
	$x_{H_{\text{lower}}}$	humanoid lower arm position	3	✓	✓	✓	✗	✗
	$x_{H_{\text{upper}}}$	humanoid upper arm position	3	✓	✓	✓	✗	✗
	$x_{H_{\text{mouth}}}$	humanoid mouth position	3	✗	✗	✗	✓	✓
	x_{ee_t}	end-effector to target distance	3	✓	✓	✓	✓	✓
	d_{ee_t}	end-effector to target euclidean distance	1	✓	✓	✓	✓	✓
	$\mathbf{R}_{ee_t}$	end-effector to target angular distance	9	✗	✗	✓	✓	✓
	$x_{H_t'}$	humanoid arm to waist target distance	3	✗	✗	✓	✗	✗
	$d_{H_t'}$	humanoid arm to waist target euclidean distance	1	✗	✗	✓	✗	✗

Table 5: Reward component overview. Each component is evaluated using the equation and scaled when computing the reward. In equations σ is a scaling factor we set to 0.1, v_{ee} is the end-effector velocity, $[\cdot]$ is the indicator function, f^* and v^* are the target forces and velocities respectively, v^*_{app} is a distance-dependent target approach speed, and $v_{\tan}$ is the tangential brushing speed.

Component	Symbol	Equation	Scale	Task				
				Scratch	*Bed Bath*	*Arm Assist*	*Feeding*	*Teeth Brush*
Reach target	r_t	$\exp\left(-\frac{d_{ee}^2}{\sigma}\right)$	1	✓	✓	✓	✗	✗
Reach target (exp)	r_{t_e}	$\exp(-3\, d_{ee})$	2	✗	✗	✗	✓	✓
Scratch	r_s	$[d_{ee_t} < 0.1] \cdot \left(\frac{v_{ee}}{v^*} \exp\left(-\frac{v_{ee}}{v^*}\right)\right) \cdot \left(\frac{f_{ee}}{f^*} \exp\left(-\frac{f_{ee}}{f^*}\right)\right)$	4	✓	✗	✗	✗	✗
Wipe	r_w	$[d_{ee_t} < 0.1] \cdot [f_{ee} > 0]$	3	✗	✓	✗	✗	✗
Reach waist	$r_{t'}$	$1 - \tanh\left(\frac{d_{H_{t'}}}{\sigma}\right)$	10	✗	✗	✓	✗	✗
Rotation	r_R	$\text{norm}(\mathbf{R}_{ee_t})$	0.1	✗	✗	✓	✗	✗
Orientation	r_o	$r_{\text{pour}} + (0.3 + 0.7 \cdot p)\, r_{\text{aim}}$	1	✗	✗	✗	✓	✗
Approach velocity	r_v	$\exp\left(-\frac{(v_{ee} - v^*_{\text{app}})^2}{0.01}\right)$	1	✗	✗	✗	✓	✗
Contact force	r_f	$\frac{f_{ee}}{f^*} \exp\left(-\frac{f_{ee}}{f^*}\right)$	1	✗	✗	✗	✓	✗
Alignment	r_a	$r_{\text{align}} \cdot \exp(-3\, d_{ee})$	1	✗	✗	✗	✗	✓
Brush	r_b	$[d_{ee} < 0.05] \cdot \left(\frac{v_{\tan}}{v^*} \exp\left(-\frac{v_{\tan}}{v^*}\right)\right) \cdot \left(\frac{f_{ee}}{f^*} \exp\left(-\frac{f_{ee}}{f^*}\right)\right)$	1	✗	✗	✗	✗	✓

A.2 Simulation Fidelity

See Figure 7 for an example of how we set collisions and ue primitive geometries in Assistax and how we use primitive geometries instead of mesh collisions.

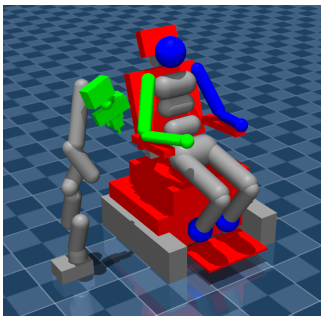


Figure 7: Shows the scratch-itch task with primitive geometries. Green geometries collide with both green and red geometries while blue geometries collide with blue and red geometries. Gray geometries have collision disabled.

A.3 Preference Reward System

The preference rewards augments the task reward with various terms.

Preference Components. The preference reward is a weighted sum of the components in Table 7.

Table 6: Preference reward inputs.

Variable	Units	Meaning
`ee_speed`	m/s	End-effector speed, $\|\mathbf{p}_t - \mathbf{p}_{t-1}\|/\Delta t$.
`ee_force`	N	Contact force magnitude on the humanoid, $\|\mathbf{f}\|$.
`new_touches`	–	Penalty for repeated unnecessary contacts, $N_{\text{touch}} - 1$.

Table 7: Preference reward components.

Component	Formula	Default	Interpretation / Sweep
Speed preference	$\text{Pref}(v; [v_{\min}, v_{\max}])$	$[0.06, 0.14]$ m/s	Rewards motion within the preferred speed range. Sweep: $v_{\min} \sim [0.03, 0.08]$, $v_{\max} \sim [v_{\min}, 0.20]$.
Force preference	$\text{Pref}(f; [f_{\min}, f_{\max}])$	$[1.5, 3.5]$ N	Rewards contact force within the preferred range. Sweep: $f_{\min} \sim [1.0, 2.0]$, $f_{\max} \sim [f_{\min}, 5.0]$.
Touch penalty	$\mathbb{1}[f_{t-1} < \tau \ \wedge \ f_t \geq \tau]$	weight $= -0.03$	Penalizes new touch events. Threshold $\tau = 0.1$ N.

Gaussian preference. For speed and force, the preference function is

$$\text{Pref}(x; [x_{\min}, x_{\max}]) = \begin{cases} 1, & x_{\min} \leq x \leq x_{\max}, \\ \exp\left(-\dfrac{(x-c)^2}{2w^2}\right), & \text{otherwise}, \end{cases}$$

where

$$c = \frac{x_{\min} + x_{\max}}{2}, \qquad w = \frac{x_{\max} - x_{\min}}{2}.$$

This yields full reward inside the preferred interval and Gaussian decay outside it.

Budget Normalisation. To ensure the total preference reward is calibrated relative to the task reward, we normalise the component weights into a fixed budget. Let w_s, w_f, and w_a denote the weights for speed, force, and action efficiency respectively, and let B denote the reward budget. We compute the normalisation factor as

$$\eta = \frac{B}{w_s + w_f + w_a}.$$

Each component is then scaled by η and its respective weight:

$$\begin{aligned} r_{\text{speed}} &= \eta\, w_s \cdot \text{Pref}(v;\, [v_{\min}, v_{\max}]) && \in \big[0,\ \eta\, w_s\big], \\ r_{\text{force}} &= \eta\, w_f \cdot \text{Pref}(f;\, [f_{\min}, f_{\max}]) && \in \big[0,\ \eta\, w_f\big], \\ r_{\text{action}} &= \eta\, w_a \cdot \text{ActionEff}(\mathbf{a}) && \in \big[0,\ \eta\, w_a\big], \\ r_{\text{touch}} &= \eta\, w_t \cdot \text{TouchPenalty}(f_t, f_{t-1}) && \in \big[\eta\, w_t,\ 0\big], \end{aligned}$$

where $w_t < 0$ is the touch penalty weight. By construction, the maximum positive contribution sums to exactly B regardless of the individual weight values:

$$\max\big(r_{\text{speed}} + r_{\text{force}} + r_{\text{action}}\big) = B.$$

The total preference reward is then

$$r_{\text{pref}} = w_{\text{overall}} \cdot \big(r_{\text{speed}} + r_{\text{force}} + r_{\text{action}} + r_{\text{touch}}\big),$$

where w_{overall} is a global scaling factor applied to the combined preference signal.

During Population Training

During population training we sample each component uniformly at random, in total we sample 630 preference combinations per task.

Preference Observations

When preference rewards are enabled we pass the chosen preferences to the agent observations by appending a 7D vector containing the preference ranges and weights.

A.4 Baseline Train Times

Table 8: Assistax wall-clock IPPO training times. Extrapolated from 5 IPPO updates to full 610 updates utilised when training 40 million timesteps.

	Mean 5 Updates (s)	Std 5 Updates (s)	Est. Full Runtime (s)	Est. Full Runtime (min)
Scratching	12.028	0.0044	1467.42	24.46
Bed Bathing	9.248	0.0025	1128.26	18.80
Arm Assist	9.989	0.0034	1218.64	20.31
Feeding	9.602	0.0050	1171.50	19.52
Teethbrushing	8.821	0.0089	1076.18	17.94

B Additional Results

B.1 Crossplay

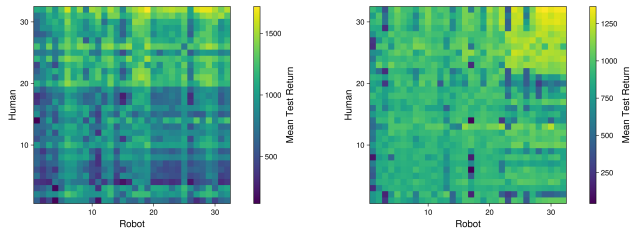
Figure 8 show the cross compatibility of random subsamples of our partner population. We note some mutually incompatible strategies emerge notably in Figure 8(a) at index 27, only co-trained agents perform well. We do not use explicit diversity metrics to encourage diversity in the population beyond preference rewards, future work could consider the use of methods for generating diverse populations of agents such as BRDiv (Rahman et al., 2023).

B.2 SAC AHT

See Figure 9, for SAC AHT trained across less timesteps.

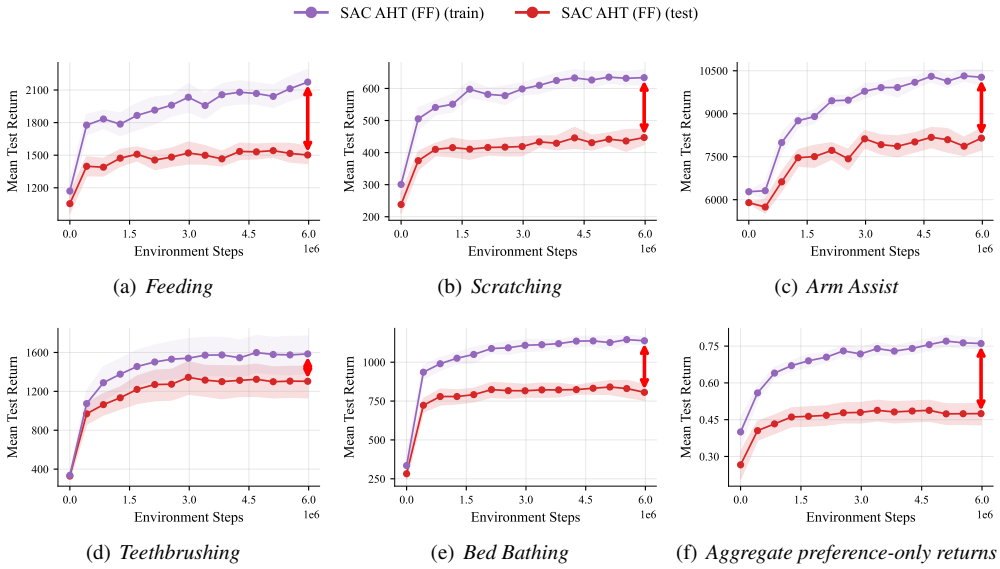
B.3 AHT Preference Returns Only

Figure 10 shows the coordination gap between only preference returns, excluding task returns. The coordination gap shown in Section 3.4 can attributed almost entirely to poor adaptation to new preferences.



(a) *Teethbrushing* (b) *Bed Bathing*

Figure 8: Cross-play matrices for a subsample of 32 agents per task. Each cell (i, j) shows the mean test return (averaged over 64 episodes and 16 seeds) when a robot i is paired with humanoid j, where the two policies were trained independently in different teams. Co-trained pairs lie on the diagonal.



(a) *Feeding* (b) *Scratching* (c) *Arm Assist*

(d) *Teethbrushing* (e) *Bed Bathing* (f) *Aggregate preference-only returns*

Figure 9: Ad-hoc teamwork performance of SAC across all five tasks. Mean returns and stratified 95% bootstrapped CI across 16 seeds are shown. Trained for 6 million timestep.

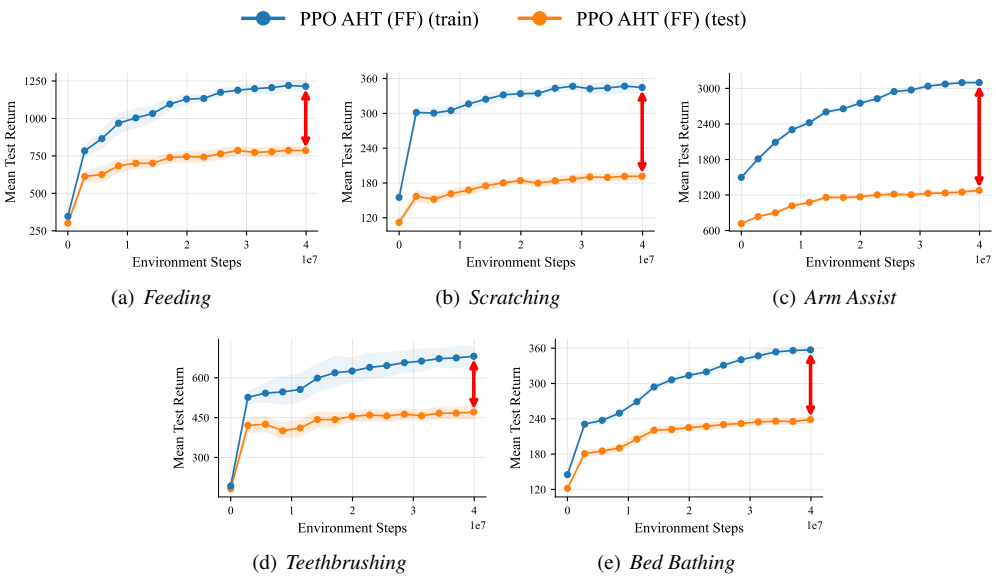
B.4 Per Environment Task only vs Task + Preference Rewards

See Figure 11

C Hyperparameters

Below we show the selected hyperparemeters for our baselines. We run one very extensive sweep across our MARL baselines, and reuse these for our AHT Baselines. We provide the same budget of sweeped hyperparemeter i.e. 168 different hyperparameter configurations for all baselines. We evaluete every hyperparameter setting across 12 seeds and 32 independent evaluation episodes.

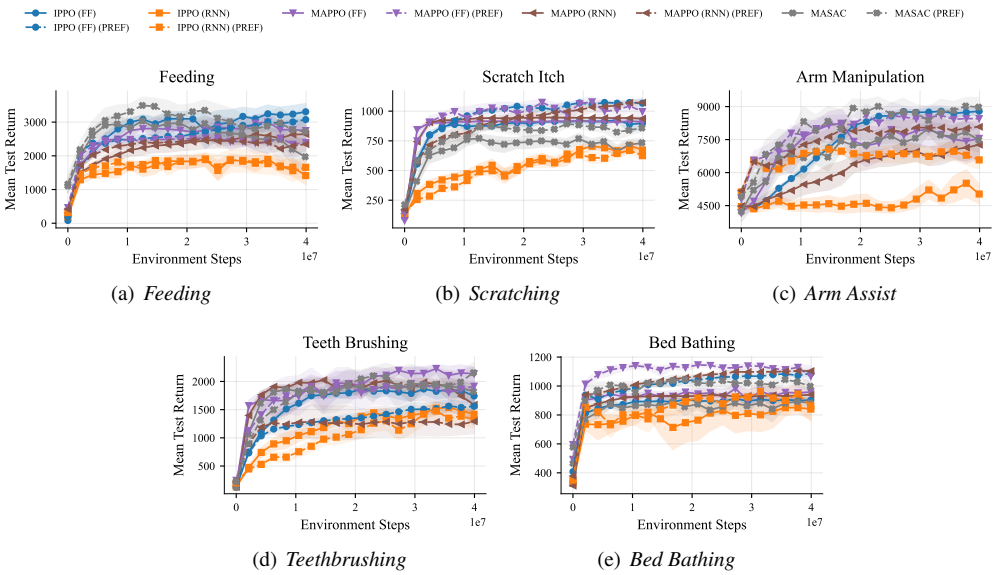
We note that since sac has an two additional hyper-parameter which are useful to sweep. This however results in covering less of the search space wihtin each individual hyperparemeter in order to test more different combination of all hyperparemeters. We however feel that by assigning the same compute budget for our sweeps across all algorithms we provide the most fair comparison between them.



(a) *Feeding* (b) *Scratching* (c) *Arm Assist*

(d) *Teethbrushing* (e) *Bed Bathing*

Figure 10: Mean preference returns and stratified 95% bootstrapped CI across 16 seeds are shown. These are returns coming exclusively from the preference reward component.



(a) *Feeding* (b) *Scratching* (c) *Arm Assist*

(d) *Teethbrushing* (e) *Bed Bathing*

Figure 11: Mean returns and stratified 95% bootstrapped CI across 16 seeds are shown, for environments with and without preference rewards.

Table 9: Hyperparameters for PPO-based algorithms. PPO_{AHT} reuses the IPPO (FF) configuration for each environment. Env. abbreviations: SI = Scratching, BB = Bed Bathing, AA = Arm Assist, Feed = Feeding, Teeth = TeethBrushing.

Algorithm	Arch.	Env.	LR	Epochs	Minibatches	Clip ϵ	Ent. Coef.	Num Steps
IPPO	FF	SI	3.34×10^{-4}	8	16	0.169	1.61×10^{-3}	64
		BB	8.93×10^{-4}	16	8	0.042	6.38×10^{-5}	64
		AA	2.45×10^{-4}	16	8	0.021	1.08×10^{-4}	64
		Feed	3.34×10^{-4}	16	16	0.169	1.61×10^{-3}	64
		Teeth	3.34×10^{-4}	4	16	0.169	1.61×10^{-3}	64
IPPO	RNN	SI	4.39×10^{-4}	8	4	0.046	5.45×10^{-4}	64
		BB	2.89×10^{-3}	16	4	0.066	5.73×10^{-4}	64
		AA	4.61×10^{-4}	8	4	0.253	1.69×10^{-3}	64
		Feed	4.61×10^{-4}	8	8	0.253	1.69×10^{-3}	64
		Teeth	1.25×10^{-4}	16	16	0.130	2.39×10^{-3}	64
MAPPO	FF	SI	2.56×10^{-3}	8	4	0.225	3.91×10^{-5}	128
		BB	1.12×10^{-3}	16	8	0.150	2.90×10^{-3}	128
		AA	1.54×10^{-3}	16	4	0.271	7.95×10^{-4}	128
		Feed	1.54×10^{-3}	4	16	0.271	7.95×10^{-4}	128
		Teeth	4.61×10^{-4}	8	16	0.253	1.69×10^{-3}	128
MAPPO	RNN	SI	4.61×10^{-4}	4	8	0.253	1.69×10^{-3}	128
		BB	1.04×10^{-4}	16	16	0.068	4.25×10^{-5}	128
		AA	4.61×10^{-4}	8	16	0.253	1.69×10^{-3}	128
		Feed	2.56×10^{-3}	8	8	0.225	3.91×10^{-5}	128
		Teeth	2.56×10^{-3}	4	8	0.225	3.91×10^{-5}	128
PPO_{AHT}	FF	SI	3.34×10^{-4}	8	16	0.169	1.61×10^{-3}	64
		BB	8.93×10^{-4}	16	8	0.042	6.38×10^{-5}	64
		AA	2.45×10^{-4}	16	8	0.021	1.08×10^{-4}	64
		Feed	3.34×10^{-4}	16	16	0.169	1.61×10^{-3}	64
		Teeth	3.34×10^{-4}	4	16	0.169	1.61×10^{-3}	64

Table 10: Hyperparameters for SAC-based algorithms. SAC_{AHT} reuses the MASAC configuration for each environment. All variants use feed-forward architectures. Env. abbreviations as in Table 9.

Algorithm	Env.	Policy LR	Q LR	α LR	τ	SAC Updates	Rollout / Batch
MASAC	SI	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	32	8 / 512
	BB	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	64	8 / 512
	AA	2.34×10^{-4}	1.37×10^{-5}	1.98×10^{-4}	1.08×10^{-2}	64	8 / 512
	Feed	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	128	8 / 128
	Teeth	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	32	8 / 128
SAC_{AHT}	SI	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	32	8 / 512
	BB	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	64	8 / 512
	AA	2.34×10^{-4}	1.37×10^{-5}	1.98×10^{-4}	1.08×10^{-2}	64	8 / 512
	Feed	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	128	8 / 128
	Teeth	5.62×10^{-5}	1.78×10^{-3}	2.52×10^{-4}	2.73×10^{-4}	32	8 / 128

C.1 Scratching Environment Sweep Plots

Here we visualize hyperparemeter sweeps for the scratching environment (note that we do sweep across all environments). We provide grids of sweeps where each subplot shows a combination of discrete hyepr parameters which with a random sweep over a continuous hyperparameters. We have 3 grids per algorithm environment pair 1 grid plot for each contionous hyperparameter.

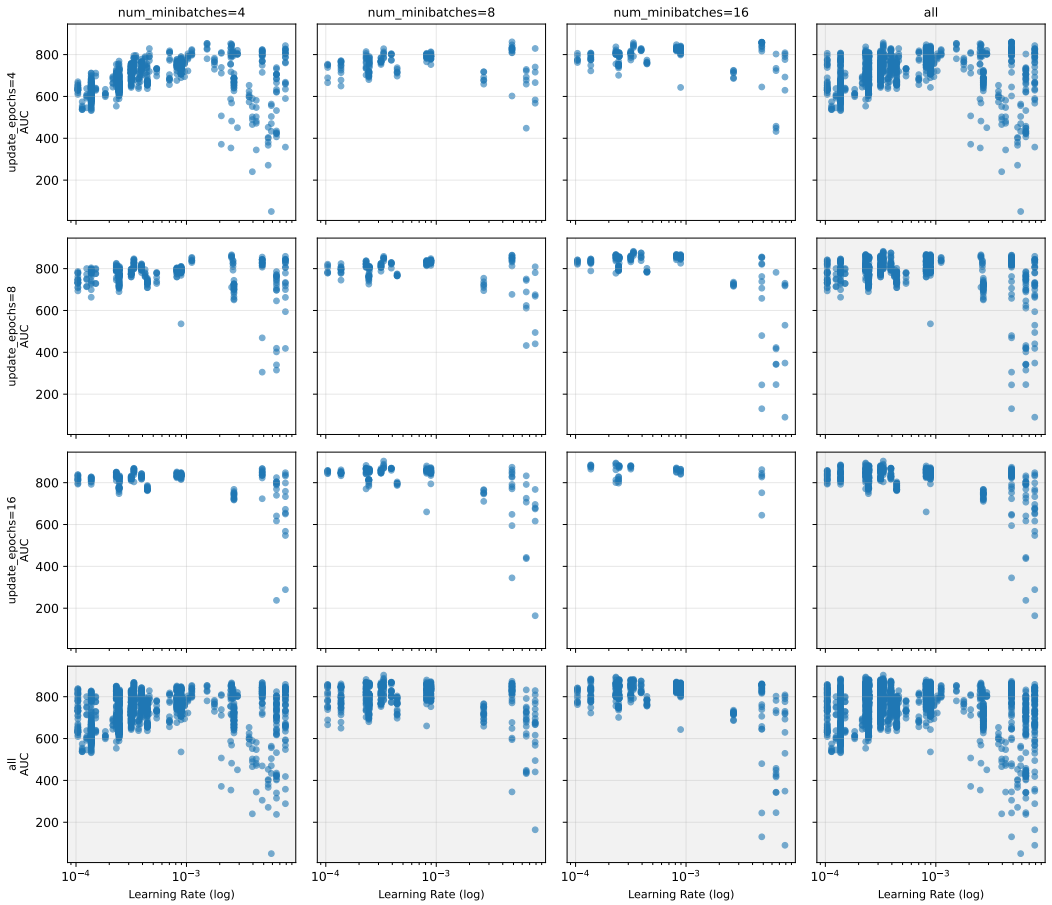


Figure 12: **IPPO (feed-forward), Scratch Itch — learning-rate sweep.** Each marker gives one configuration's AUC return as a function of the learning rate (log x-axis); higher is better. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

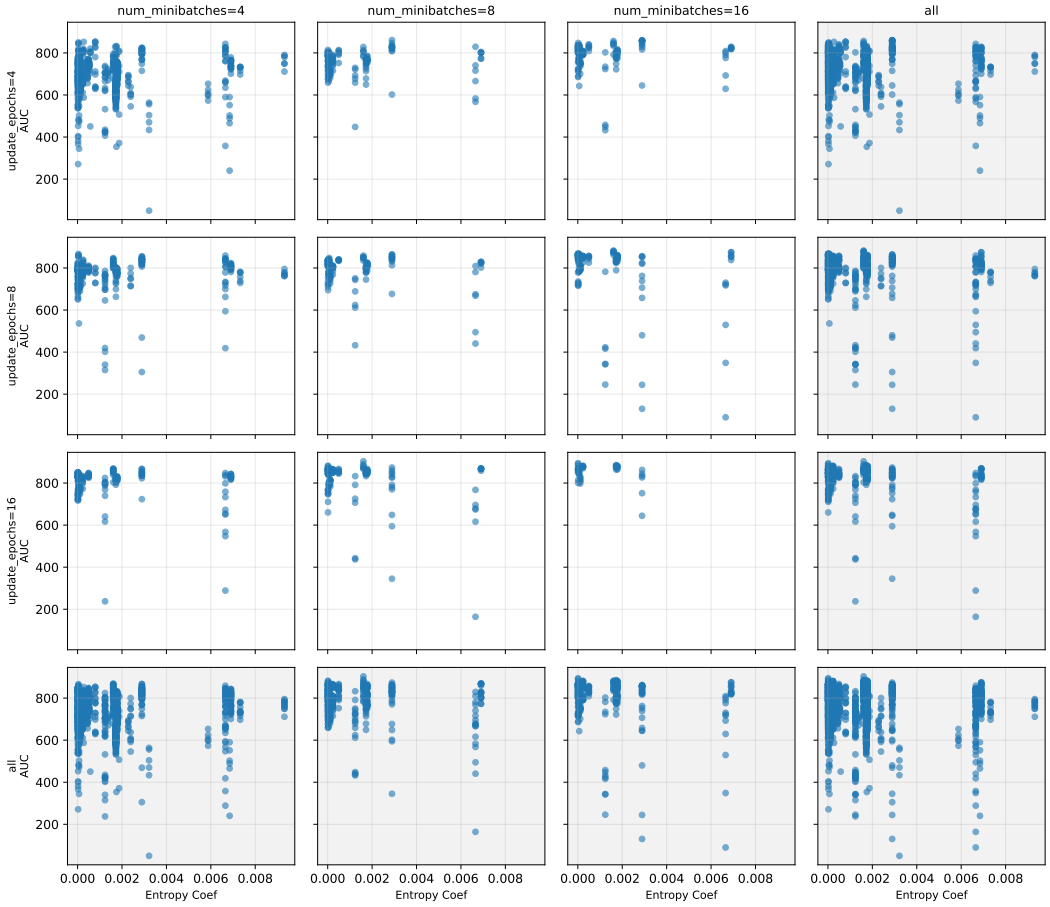


Figure 13: **IPPO (feed-forward), Scratch Itch — entropy-coefficient sweep.** AUC return against the entropy-regularisation coefficient, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

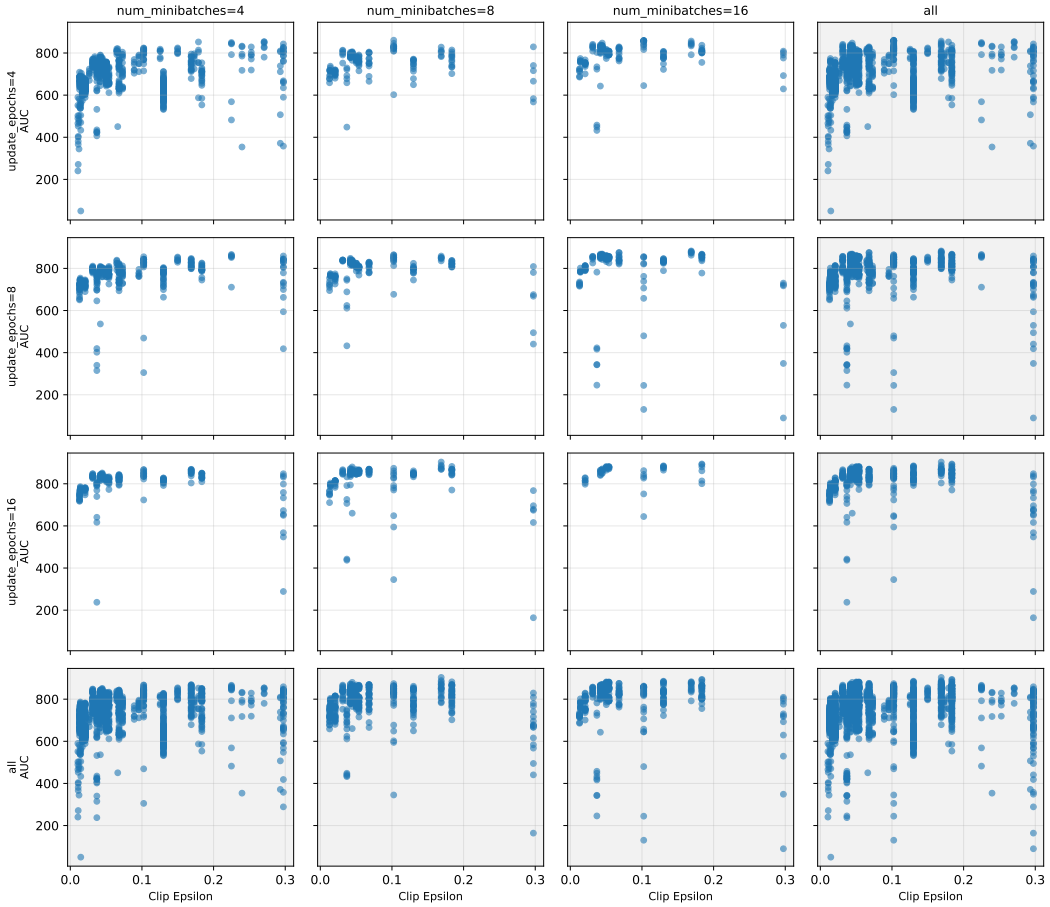


Figure 14: **IPPO (feed-forward), Scratch Itch — PPO clip-range sweep.** AUC return against the PPO clipping parameter ϵ, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

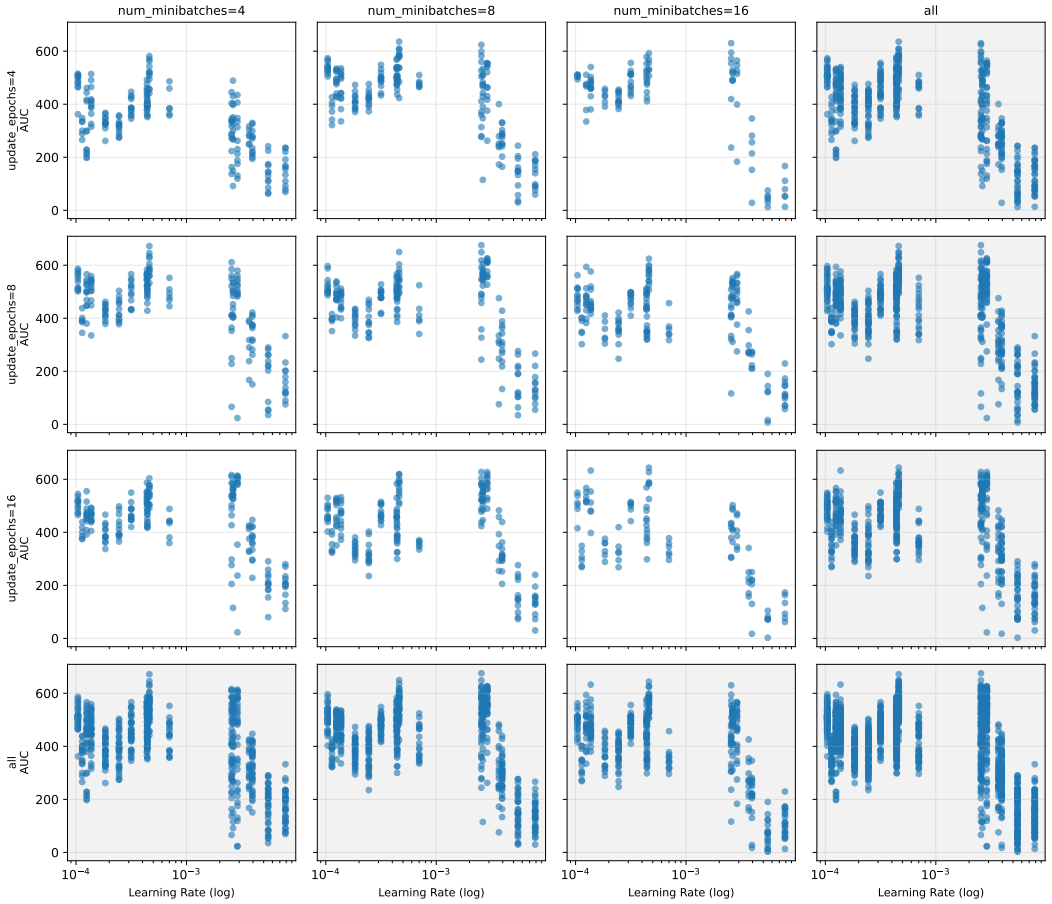


Figure 15: **IPPO (recurrent), Scratch Itch — learning-rate sweep.** Each marker gives one configuration's AUC return as a function of the learning rate (log x-axis); higher is better. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

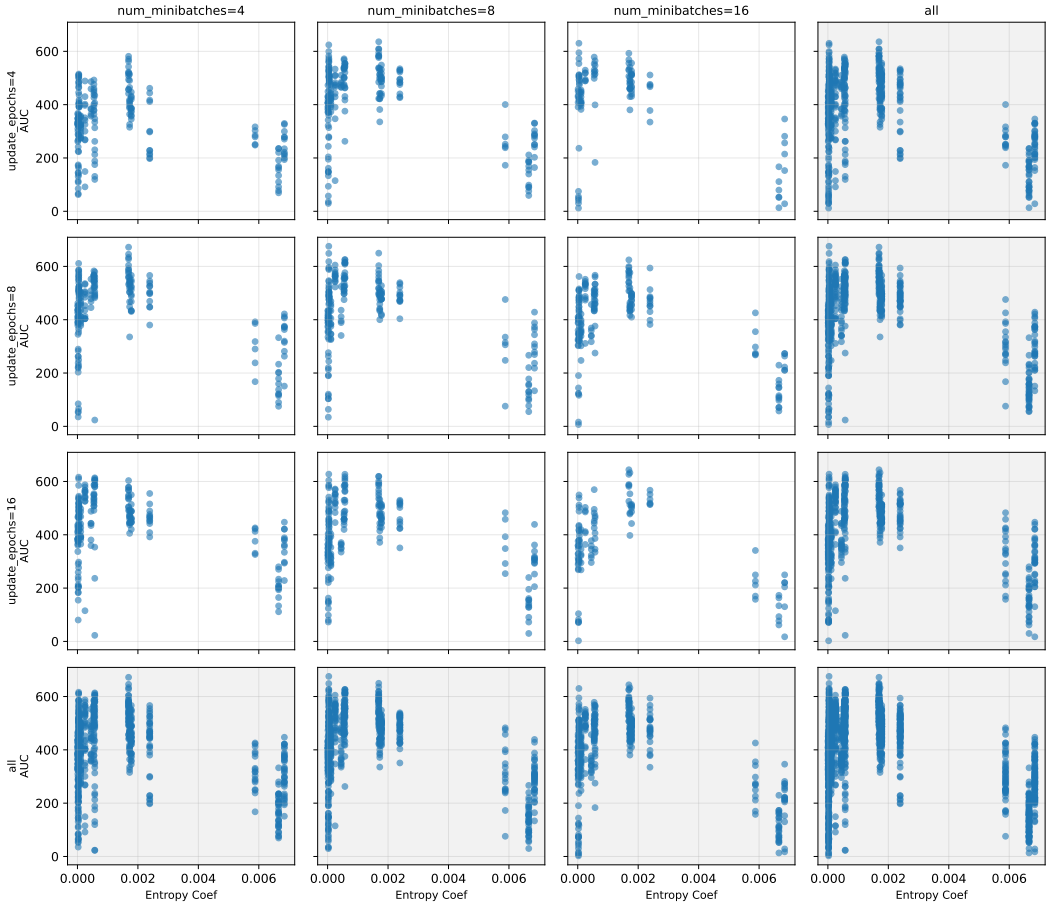


Figure 16: **IPPO (recurrent), Scratch Itch — entropy-coefficient sweep.** AUC return against the entropy-regularisation coefficient, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

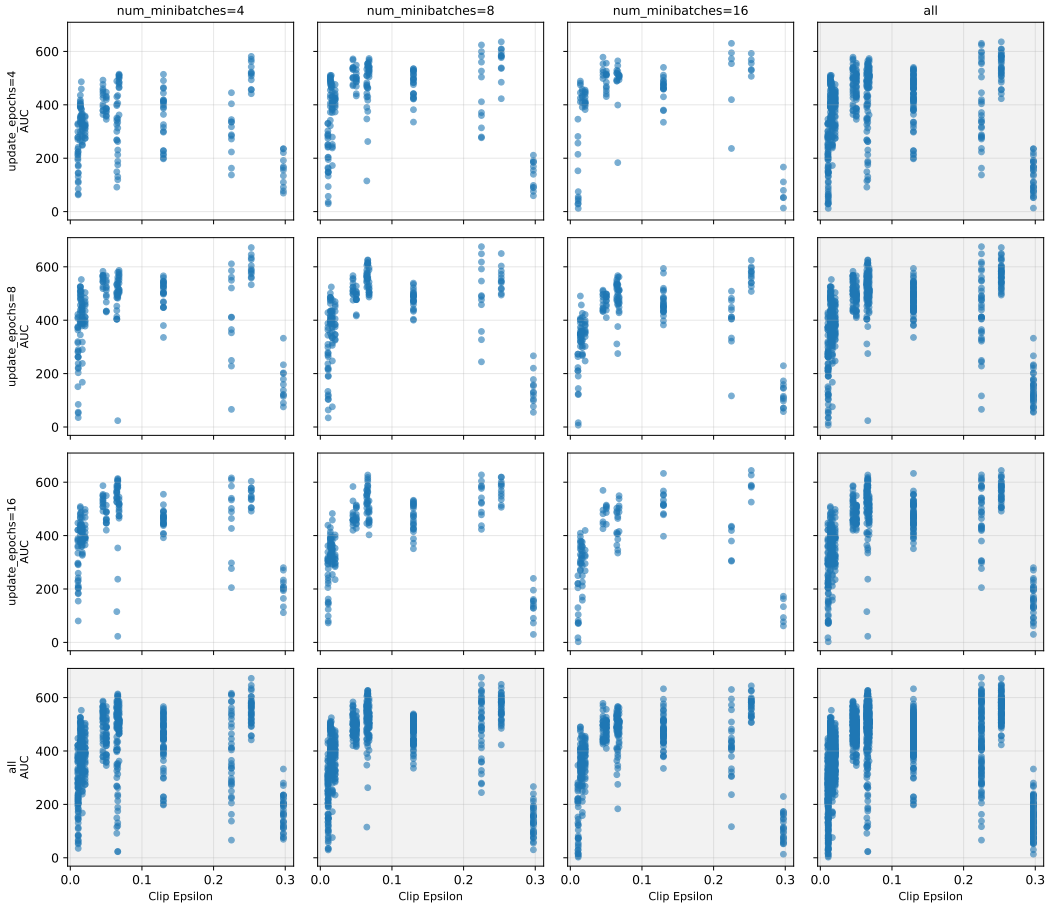


Figure 17: **IPPO (recurrent), Scratch Itch — PPO clip-range sweep.** AUC return against the PPO clipping parameter ϵ, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

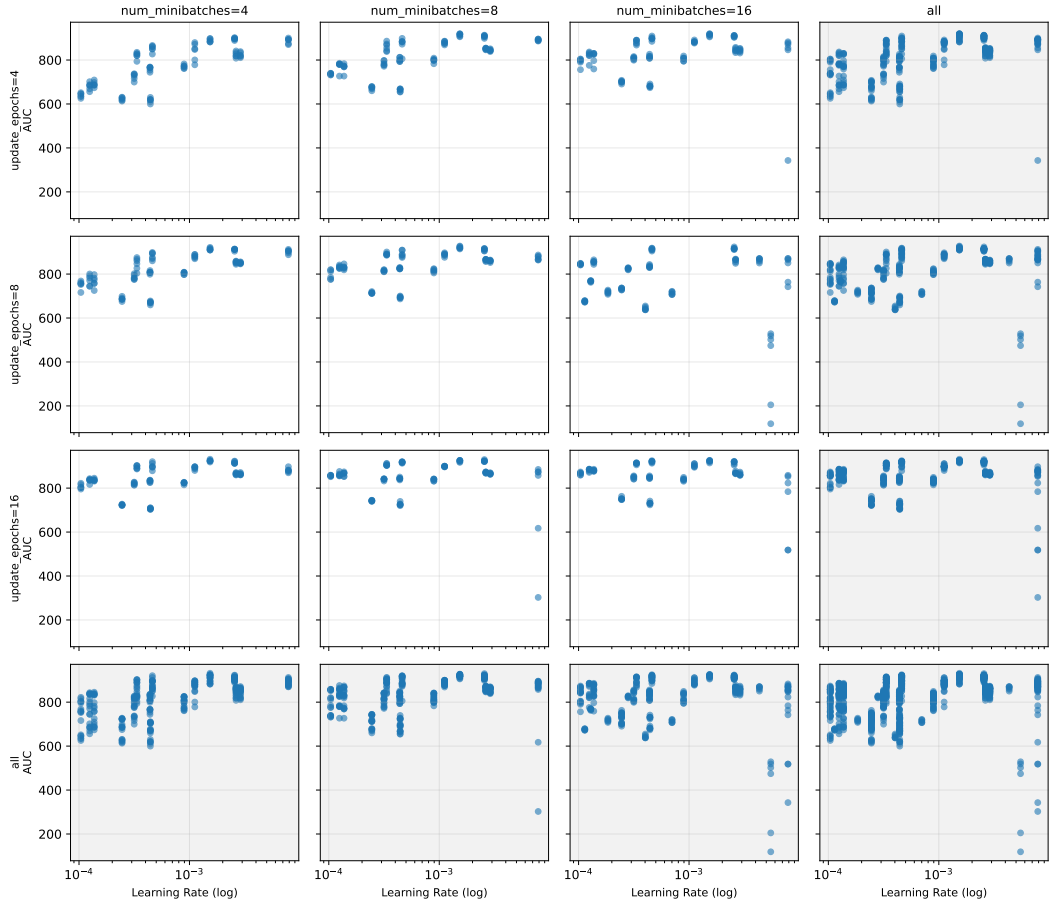


Figure 18: **MAPPO (feed-forward), Scratch Itch — learning-rate sweep.** Each marker gives one configuration's AUC return as a function of the learning rate (log x-axis); higher is better. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

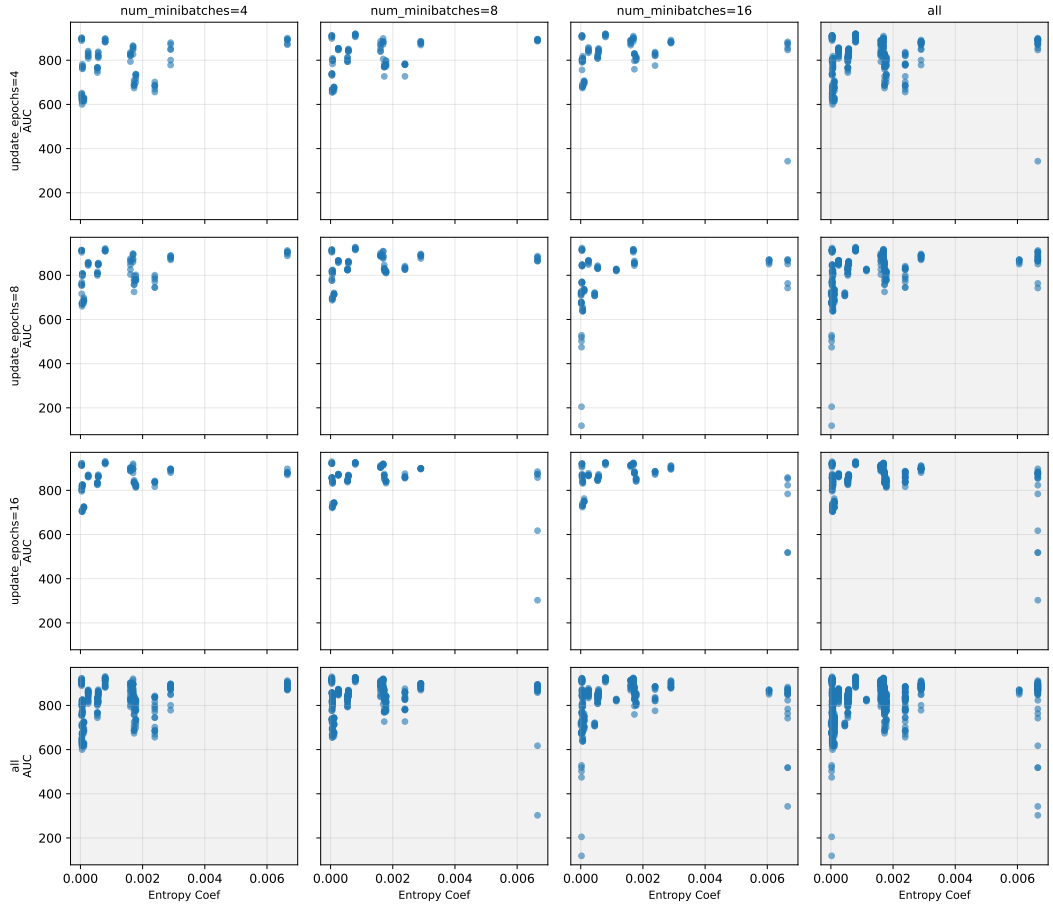


Figure 19: **MAPPO (feed-forward), Scratch Itch — entropy-coefficient sweep.** AUC return against the entropy-regularisation coefficient, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

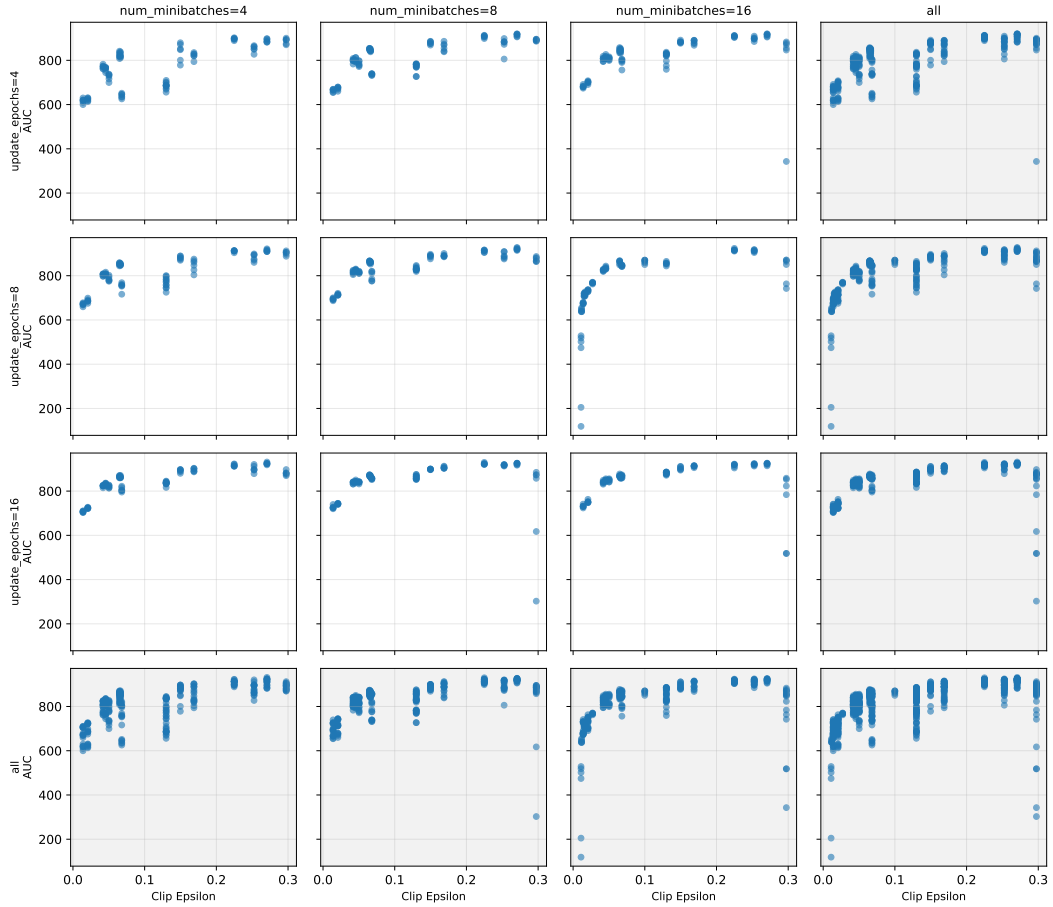


Figure 20: **MAPPO (feed-forward), Scratch Itch — PPO clip-range sweep.** AUC return against the PPO clipping parameter ϵ, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

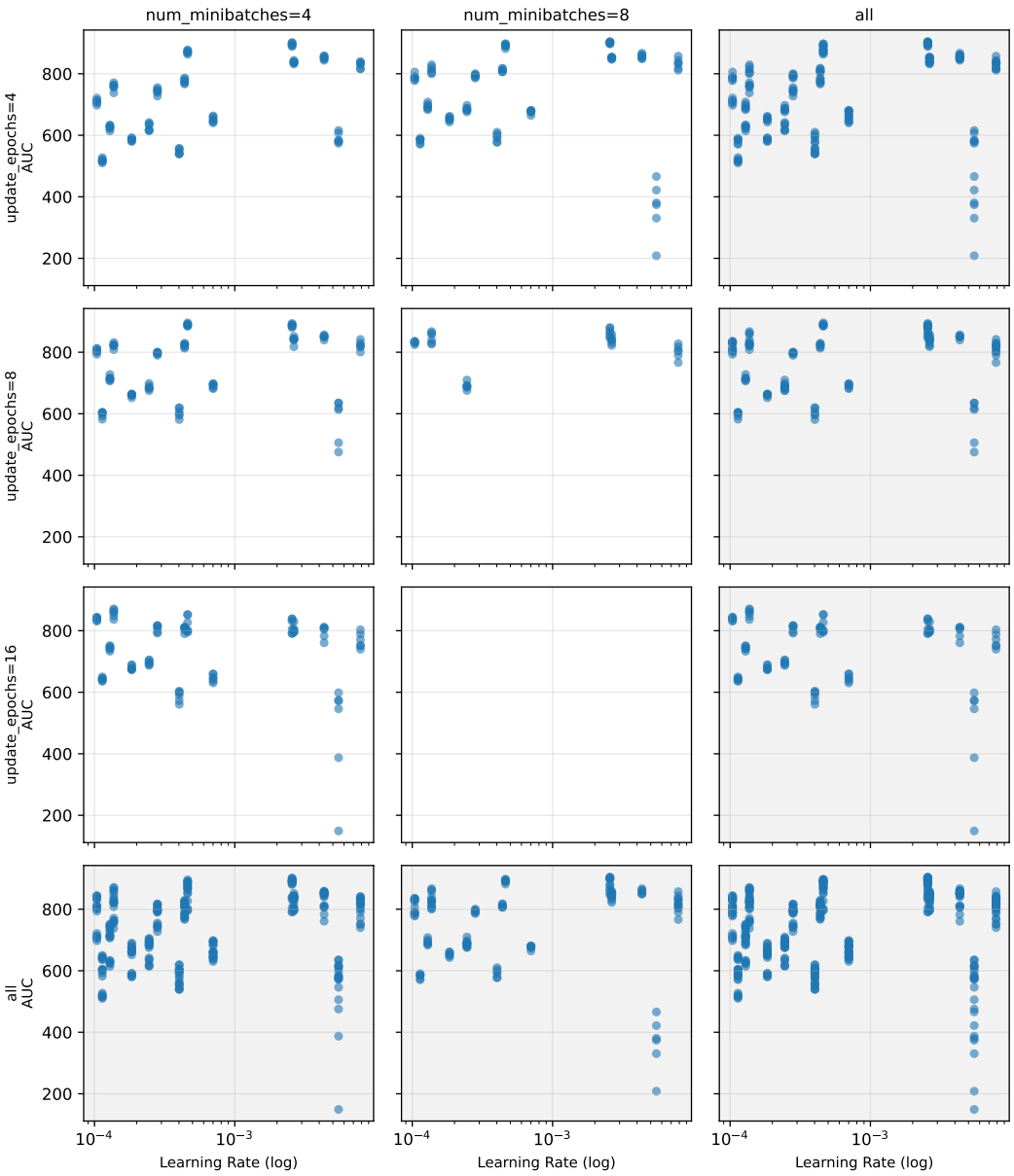


Figure 21: **MAPPO (recurrent), Scratch Itch — learning-rate sweep.** Each marker gives one configuration's AUC return as a function of the learning rate (log x-axis); higher is better. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

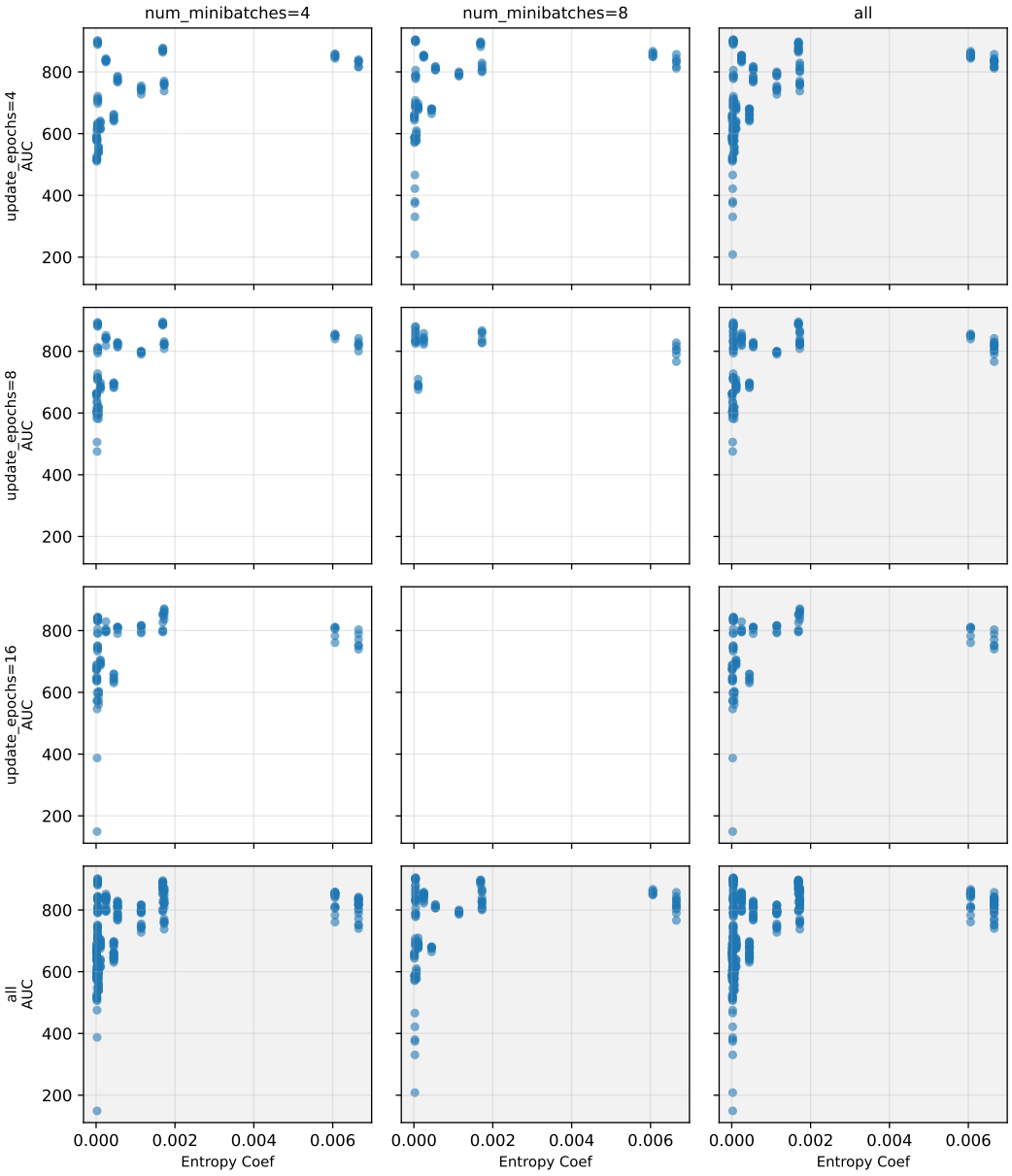


Figure 22: **MAPPO (recurrent), Scratch Itch — entropy-coefficient sweep.** AUC return against the entropy-regularisation coefficient, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

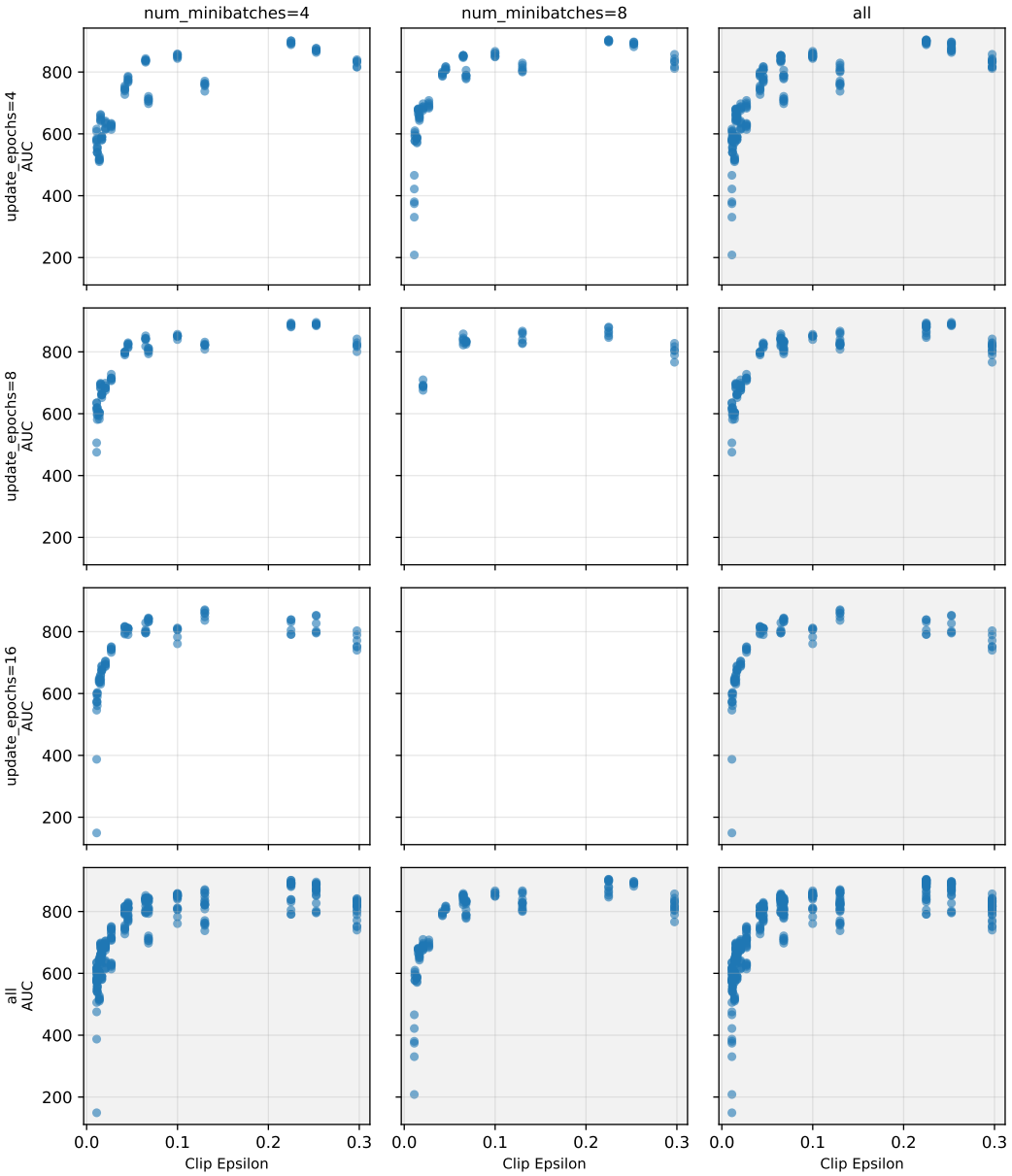


Figure 23: **MAPPO (recurrent), Scratch Itch — PPO clip-range sweep.** AUC return against the PPO clipping parameter ϵ, where each marker is a single configuration. Panels are faceted by minibatch count (columns) and update epochs (rows), with `all` pooling over that factor.

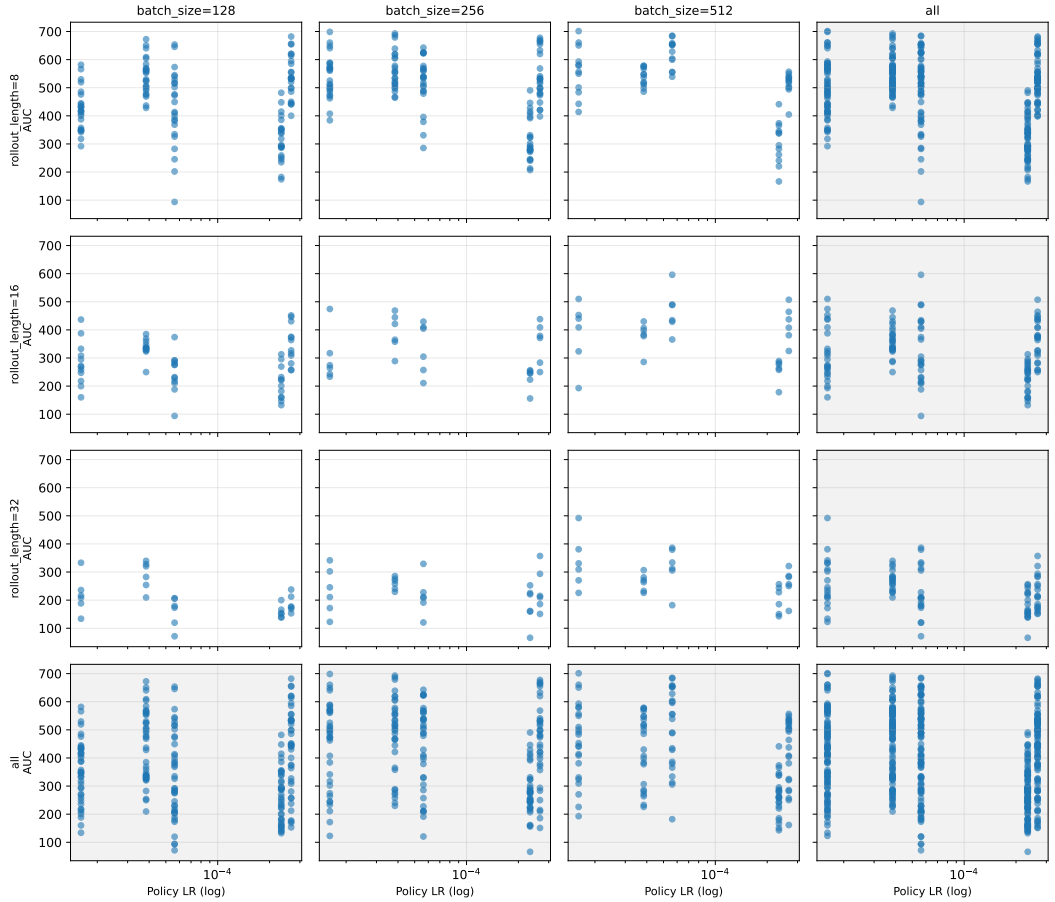


Figure 24: **MASAC (feed-forward), Scratch Itch — policy (actor) learning-rate sweep.** Each marker gives one configuration's AUC return as a function of the actor learning rate (log x-axis); higher is better. Panels are faceted by batch size (columns) and rollout length (rows), with `all` pooling over that factor.

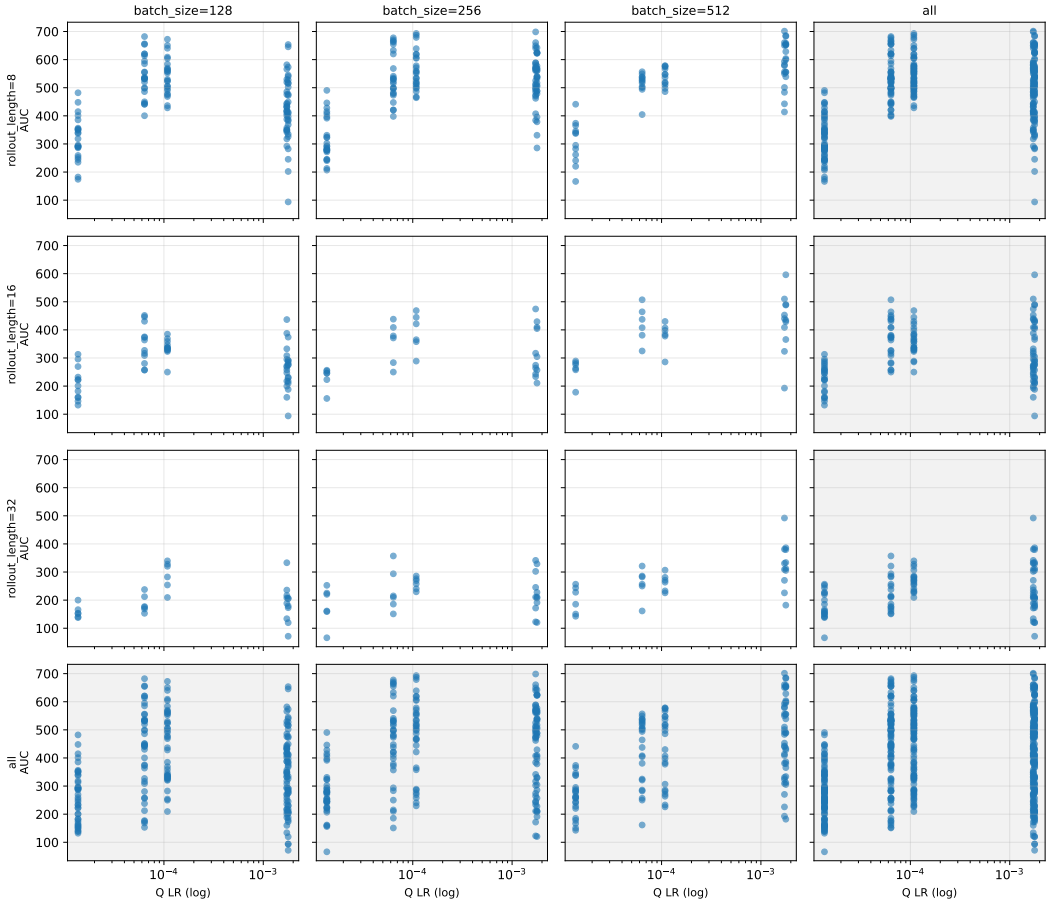


Figure 25: **MASAC (feed-forward), Scratch Itch — critic (Q) learning-rate sweep.** AUC return against the critic learning rate (log x-axis), where each marker is a single configuration. Panels are faceted by batch size (columns) and rollout length (rows), with `all` pooling over that factor.

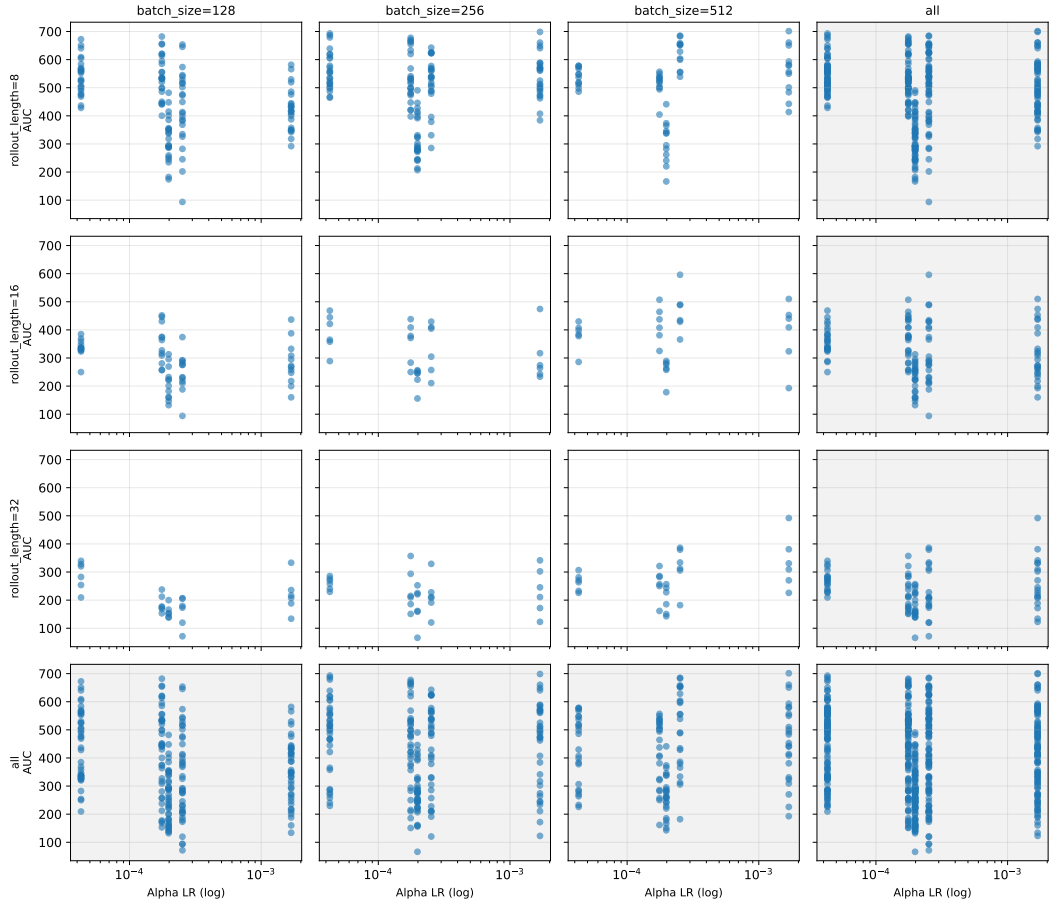


Figure 26: **MASAC (feed-forward), Scratch Itch — temperature (α) learning-rate sweep.** AUC return against the learning rate of the entropy-temperature α (log x-axis), where each marker is a single configuration. Panels are faceted by batch size (columns) and rollout length (rows), with `all` pooling over that factor.

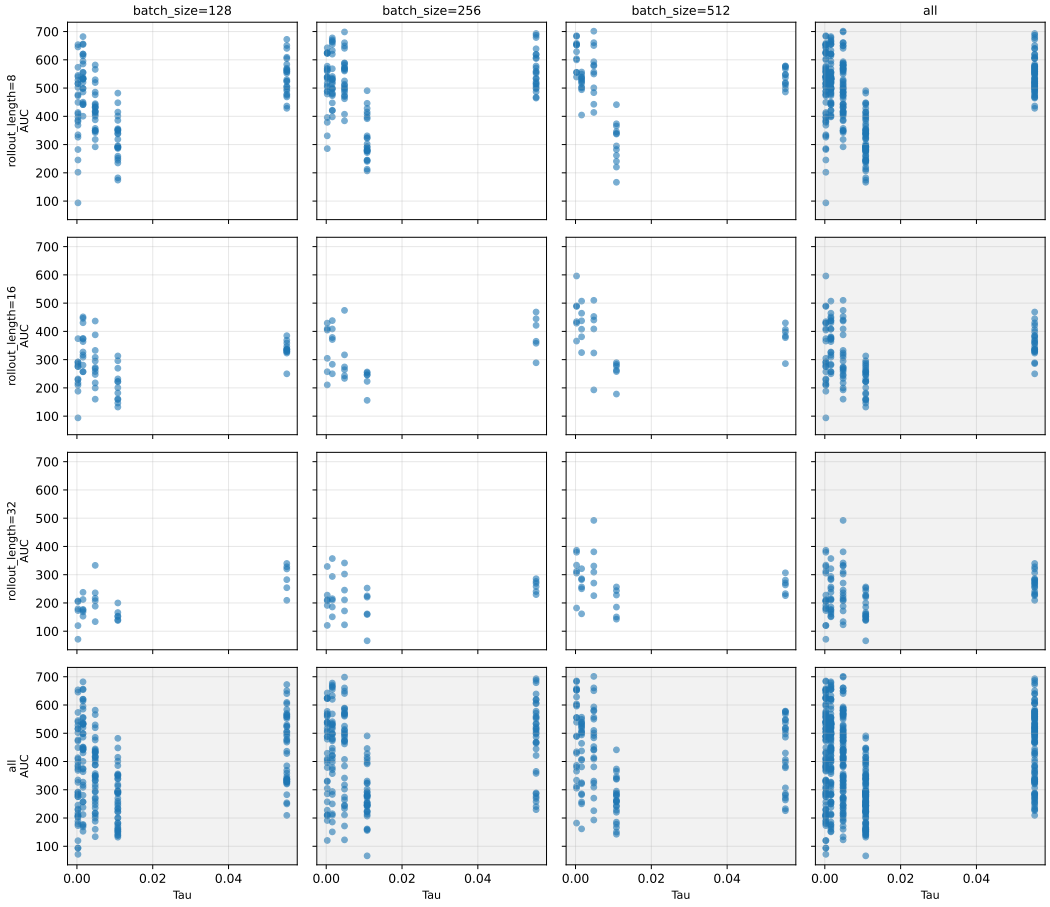


Figure 27: **MASAC (feed-forward), Scratch Itch — target-smoothing (τ) sweep.** AUC return against the target-network soft-update coefficient τ, where each marker is a single configuration. Panels are faceted by batch size (columns) and rollout length (rows), with `all` pooling over that factor.