SWAP: SWITCHED ACTION CHUNKING POLICIES WITH MASSIVELY PARALLELIZED MULTI-TASK REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In massively parallel on-policy Reinforcement Learning (RL), performance saturates beyond a certain environment count because additional environments yield redundant trajectories due to executing similar actions sampled around the policy mean. Standard entropy regularization fails to address this, leading prior single-task methods to artificially partition environments across follower policies to force behavioral diversity with separate entropy bonuses. We propose that scaling *multi-task* reinforcement learning in this regime is unique in that policies inherently provide behavioral diversity by virtue of solving multiple tasks. The key insight is that agents can execute cross-task or *switched* actions during any task's rollout, using the otherwise excess rollout capacity of massively parallel simulation. However, utilizing this diversity to induce coherent exploration requires temporal commitment through action chunking. During a rollout, in a designated subset of parallel environments per task, policies replan by sampling action chunks across all task policies and executing the candidate that maximizes the Q value under the true task. By relying on pathwise policy gradients that differentiate directly through the critic, the policy absorbs this off-policy exploration without importance sampling correction. We call this framework Switched Action Chunking Policies (SWAP), which only incurs inference overhead during training that scales linearly with the number of tasks. In Meta-World, SWAP outperforms recent methods addressing performance saturation in the massively parallelized regime.

1 INTRODUCTION

Massively parallel simulation enables Reinforcement Learning (RL) agents to collect experience from tens of thousands of environments simultaneously (Makoviychuk et al., 2021). However, increasing the number of environments does not necessarily improve final performance. When a single policy generates similar trajectories across enough environments, additional experience yields diminishing returns (Singla et al., 2024), and standard entropy regularization does not resolve this. Prior single-task approaches address this limitation by artificially constructing behavioral diversity through policy populations (Petrenko et al., 2023; Singla et al., 2024; Wang et al., 2025; Shitanda et al., 2026) or environment-specific exploration noise (Li et al., 2023; Seo et al., 2025). This raises the question of whether generating such diversity needs to be explicitly designed in all cases.

Multi-task reinforcement learning (MTRL) provides a natural source of such diversity. A policy trained on different tasks acquires distinct behaviors such that the policy can generalize to a state in a different task by leveraging already learned behaviors (Pinto & Gupta, 2017). Rather than only reusing experience across tasks after it is collected (Li et al., 2020; Eysenbach et al., 2020; Kalashnikov et al., 2021; Yu et al., 2021), some methods share behavior during data collection itself by executing other tasks' policies in the current task's environment (He et al., 2024; Lee et al., 2025a; Zhang et al., 2025). In the latter setting, multi-task learning can leverage massively parallel simulation by using a portion of each task's otherwise redundant environments to execute exploratory, off-policy behaviors borrowed from other tasks, which we denote as policy switching, while still executing the current task's policy in the remaining environments to retain the stability of on-policy learning.

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.

However, when the policy to sample from is reselected at every timestep, the probability of following a single borrowed behavior for many consecutive steps decays exponentially with its length, resulting in excessive switching. We observe that the remedy found for this in single-task exploration, replacing single-step actions with temporally extended ones (Dabney et al., 2020), applies here as well. In particular, instead of simply repeating an action (Lee et al., 2025a) or restricting the cadence of selecting a new policy (He et al., 2024), we utilize action chunking (Zhao et al., 2023) as another form of temporally extended learning. Unlike the other forms of temporal commitment, a chunk is a sequence of distinct actions that a Q-function under the true task can score directly, and that the actor maximizes against, which uniquely synergizes with resolving saturation through exploration.

We propose **Sw**itched **A**ction Chunking **P**olicies (SWAP), which combines policy switching with action chunking. In a designated subset of parallel environments per task, SWAP samples candidate action chunks from all task policies and executes the chunk that maximizes the Q-value under the current task's critic. The remaining environments per task collect experience using the current task's policy. Policy switching generally introduces a mismatch between the data-collection policy and the current task's policy. Building on REPPO (Voelcker et al., 2026), SWAP uses pathwise actor gradients through the critic to learn from this experience without importance-sampling corrections in the actor update. Specifically, the actor is optimized using freshly sampled action chunks from the current task's policy, rather than the behavior actions executed during collection. Cross-task candidate generation and selection are used only during training, while evaluation uses the target task's policy.

SWAP consistently outperforms methods built on top of PPO, as well as REPPO itself, nearly solving the benchmark with up to 98.2% success. Notably, SWAP surpasses these baselines using only half as many environment steps, demonstrating substantially greater sample efficiency. Ablations show that combining policy switching with action chunking yields greater improvements than either component alone, supporting the need for coherent temporal exploration when borrowing behaviors from other tasks.

Our main contributions is, SWAP (*SWitched Action chunking Policies*), a massively parallel, multi-task RL method that combines policy switching with action chunking to exploit task-induced behavioral diversity. SWAP can be combined with existing MTRL advances, as it only requires modifications to the data collection loop and TD targets. SWAP demonstrates strong improvements in success rate and sample efficiency over existing baselines on Meta-World MT50 across multiple rollout horizons, and we show that both components of SWAP are important for overcoming performance saturation.

2 Related Work

2.1 Large-Scale Parallelization

Advances in GPU-accelerated simulators enable the collection of large-scale rollouts of tens of thousands of parallel environments (Makoviychuk et al., 2021; Mittal et al., 2025; Zakka et al., 2025; GenesisAuthors, 2024). While large-scale data collection can reduce gradient estimation variance, exploration with a single policy can limit behavioral diversity, and simply increasing the number of parallel environments leads to diminishing performance gains in PPO (Singla et al., 2024). To effectively utilize massively parallel environments and collect more diverse experiences, recent methods have allocated parallel environments among a population of policies (Petrenko et al., 2023; Singla et al., 2024; Wang et al., 2025; Shitanda et al., 2026) or kept a single policy that varied exploration noise per environment when using deterministic policies (Li et al., 2023; Seo et al., 2025).

In contrast, the multi-task learning problem inherently offers a source of behavioral diversity by virtue of solving different tasks. A closely related approach arises in the single-task setting where RLE (Mahankali et al., 2024) constructs a proxy for task diversity with goal-based exploration by conditioning the policy and value function on randomly sampled latents, each defining a random reward, so that different environments pursue different objectives. MTRL needs no construction, yet although Joshi et al. (2025) introduced MTBench to systematically evaluate RL algorithms and multi-task learning methods in the massively parallel regime, how to exploit task diversity to avoid performance saturation remains underexplored.

2.2 MULTI-TASK RL

One line of online MTRL work leverages shared structure between tasks through parameter sharing such as constraining separate policy networks to be close to a shared policy (Teh et al., 2017), softly routing over a mixture of experts (Yang et al., 2020), attending over task metadata (Sodhani et al., 2021), composing task-specific parameter sets (Sun et al., 2022), or mixing orthogonal experts (Hendawy et al., 2024). MTRL is also a challenging optimization problem, with many approaches such as tuning per-task gradient magnitudes (Chen et al., 2018), projecting conflicting task gradients onto one another (Yu et al., 2020a), solving for a conflict-averse update direction (Liu et al., 2021), or balancing loss decreases over training (Liu et al., 2023). More recently, Nauman et al. (2026) showed that scaling BroNet critics (Nauman et al., 2024) trained with a cross-entropy objective (Farebrother et al., 2024) as well as conditioning on learnable task embeddings reduces task interference. These methods are complementary to SWAP's performance and can be combined, but we find we already achieve strong performance solely by properly exploiting task diversity.

To that end, one idea is to reuse experience across tasks *after* it has been collected. Hindsight relabeling stores individual transitions alongside sampled goals different than the task goal (Andrychowicz et al., 2017), while task impersonation (Kalashnikov et al., 2021) relabels entire episodes with some subset of other task rewards. Closest to SWAP are methods that share behaviors *during* data collection but differ in the duration of sharing. QMP (Zhang et al., 2025) executes the task policy maximizing the current task's Q-value at every timestep, whereas SWAP selects the Q-maximized action chunks that commit to multi-step behavior. CTPG (He et al., 2024) trains an additional guide policy that samples a task policy to execute closed-loop for a fixed K-step duration, whereas SWAP's chunked policy obtains such commitment without training an additional network. MT-Lévy (Lee et al., 2025a) samples a policy by semantic similarity between task descriptions and repeats a single, borrowed action for a Lévy-distributed duration, inspired by temporally-extended ϵ-greedy exploration (Dabney et al., 2020). In contrast, SWAP executes a sequence of distinct actions determined by value maximization.

2.3 RL WITH ACTION CHUNKING

Action chunking, where the policy predicts a sequence of N actions $a_t, \ldots, a_{t+N-1}$ from the original action space and executes them open-loop, originally found success in imitation learning (Zhao et al., 2023) for modeling non-Markovian behaviors in demonstration data. In RL, action chunking improves value learning, especially in off-policy RL, by using N-step returns to reduce reliance on biased bootstraps in long-horizon tasks (Li et al., 2026). Problematically, the discounted sum of rewards is not an unbiased estimate of the expected N-step reward of the current policy because every action after the first is from a stale behavior policy. Per-decision importance sampling corrects for this, at the cost of high variance (Precup et al., 2000). TOP-ERL (Li et al., 2025) first noted that a critic conditioned on the stored chunk, $Q(s_t, a_{t:t+N})$, avoids this correction because the N-step rewards are generated by exactly the actions the critic is conditioned on. At the same time, chunking expands the critic's action input to $N \cdot \dim \mathcal{A}$ dimensions while still supervising it with a single scalar target, making Q-function approximation substantially harder.

Multi-horizon, causal transformer critics address this by regressing all prefixes of the action sequence against their respective return, $G^{(1)}, \ldots, G^{(N)}$, and as a result, multiplying the targets per state in the buffer by N. TOP-ERL (Li et al., 2025) introduced such a critic for online, episodic RL, where the policy predicts movement-primitive parameters that generate an episodic-length action sequence. Then, the critic trains on segments of that trajectory, whose length can vary without changing the policy's action space dimension. T-SAC (Tian et al., 2026) applied this critic with single-step policies, and SEAR (Nagy et al., 2026) with chunked policies, where the enlarged action space makes it necessary to provide extra supervision to the critic.

In contrast, offline RL methods like Q-chunking (Li et al., 2026) and DEAS (Kim et al., 2026) keep a single chunk-level TD target, summing rewards over the chunk and at bootstrapping s_{t+N} with a freshly sampled chunk. This is valid for a window at any offset in the dataset, but stops reward propagation at the chunk boundary.

In the massively parallel on-policy regime, each T-step rollout comes from the current policy and is discarded after each update, so actions following any chunk are never stale. This means returns can

extend past the chunk boundary, so SWAP adopts a modified TD(λ) target instead of truncating the target after N steps and cutting off reward propagation that the rollout already supports. Massive parallelism also removes the need to increase supervision per window, since the critic can rely on the sheer number of windows. As a result, SWAP forgoes multi-horizon targets for a single-horizon target, avoiding the overhead of a causal transformer.

3 PRELIMINARIES

Notation We consider a family of K Markov Decision Processes (MDPs), where each task $\tau \in \{1, \dots, K\}$ is defined by a tuple $\mathcal{M}_\tau = (\mathcal{S}, \mathcal{A}, \mathcal{P}_\tau, r_\tau, \mu_\tau, \gamma)$. Here, $\mathcal{S}$ denotes the shared state space, $\mathcal{A}$ denotes the shared continuous action space, $\mathcal{P}_\tau : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the task-specific transition dynamics, $r_\tau : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the task reward function, $\mu_\tau \in \Delta(\mathcal{S})$ denotes the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. The only restriction we place upon $\mathcal{M}_\tau$ is that their union shares a universal state space $\mathcal{S}$. By appending a task embedding (one-hot or learnable) z_τ to the state s, we allow the models to distinguish which task each observation belongs to. We aim to learn a multi-task policy $\pi_\theta(a \mid s, z_\tau)$ that maximizes the multi-task maximum entropy objective:

$$\mathcal{J}(\theta) = \frac{1}{K} \sum_{\tau=1}^{K} \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_\tau(s_t, a_t) + \alpha \mathcal{H}\big(\pi_\theta(\cdot \mid s_t, z_\tau)\big) \Big) \right], \tag{1}$$

which implies the soft action-value function:

$$Q^\pi_\phi(s_t, a_t, z_\tau) = r_\tau(s_t, a_t) + \mathbb{E}_{\pi_\theta} \left[\sum_{j=1}^{\infty} \gamma^j \Big(r_\tau(s_{t+j}, a_{t+j}) + \alpha \mathcal{H}\big(\pi_\theta(\cdot \mid s_{t+j}, z_\tau)\big) \Big) \right], \tag{2}$$

where $\mathcal{H}(\pi_\theta(\cdot \mid s, z_\tau)) = \mathbb{E}_{a \sim \pi_\theta}[-\log \pi_\theta(a \mid s, z_\tau)]$ is the policy entropy, and the temperature parameter $\alpha > 0$ trades off reward maximization against entropy maximization.

Data collection proceeds synchronously across E parallel simulation environments per task, yielding a total of $K \times E$ concurrent environments stepping simultaneously on a GPU.

REPPO Unlike score-function estimators that compute policy gradients by differentiating the log-likelihood of actions executed by the behavior policy, REPPO (Voelcker et al., 2026) optimizes policies via a pathwise policy gradient differentiated through the Q function. Given a state $s \sim \mathcal{D}$, reparameterized action $a = g_\theta(s, \xi)$ with $\xi \sim \mathcal{N}(0, I)$, the constrained policy objective is defined as:

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{s \sim \mathcal{D},\, \xi \sim \mathcal{N}(0,I)} \left[-Q_\phi\big(s, g_\theta(s, \xi)\big) + e^\alpha \log \pi_\theta\big(g_\theta(s, \xi) \mid s\big) \right],$$

subject to a Lagrangian temperature parameter α that ensures the entropy of π_θ stays close to the target entropy.

Crucially, because the pathwise gradient $\nabla_\theta Q_\phi(s, g_\theta(s, \xi)) = \nabla_a Q_\phi(s, a) \nabla_\theta g_\theta(s, \xi)$ evaluates the critic along actions sampled from the current policy rather than the actions executed during rollouts, the gradient estimator is never evaluated on the actions sampled from the behavior policy. Importantly, for our use case, this decouples the actor update from the behavior policy, eliminating the need for likelihood-ratio importance sampling corrections when using rollouts with actions not from the task's behavior policy, i.e actions from a switched task policy.

The critic is trained with an HL-Gauss loss $\mathcal{L}_Q(\phi) = \mathbb{E}\left[\mathcal{L}_{\text{HL}}\big(Q_\phi(s_t, a_t, z_\tau),\, G^\lambda_t\big)\right]$ against the TD(λ) return target. The n-step return $G^{(n)}_t$ accumulates rewards over n steps and bootstraps using the soft state-value $V(s_{t+n}, z_\tau)$:

$$G^{(n)}_t = \sum_{k=0}^{n-1} \gamma^k r_\tau(s_{t+k}, a_{t+k}) + \gamma^n V(s_{t+n}, z_\tau), \tag{3}$$

where each $V(s_{t+n}, z_\tau)$ is a single sample estimate using $Q_\phi(s_{t+n}, a' \sim \pi, z_\tau) - \alpha \log \pi(a' \mid s_{t+n})$ and finally, the TD(λ) return target

$$G^\lambda_t = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G^{(n)}_t + \lambda^{T-t-1} G^{(T-t)}_t, \tag{4}$$

is the exponentially weighted mixture across all n-step returns given a rollout of length T.

4 METHOD

SWAP builds upon REPPO with two additions: (1) policy and value learning with action chunks, and (2) performing policy switching during data collection in a designated subset of environments by executing the candidate chunk sampled from all tasks that maximizes the multi-task critic Q_ϕ under the true task.

4.1 ACTION CHUNKING

Since multi-task learning inherently provides behavioral diversity, policy switching could be naively implemented as unconditionally executing an action sampled from any task j's policy in task i's environment at each timestep. However, switching policies at every timestep can result in meaningless jitter, instead of coherent exploration. SWAP proposes that action chunking enforces temporal commitment to an alternate behavior in policy-switched rollouts.

Under an action chunking horizon of N, we have the action-chunked policy $\pi_\theta(a_{t:t+N}|s_t, z_\tau) = \pi_\theta(a_t, \ldots, a_{t+N-1}|s_t, z_\tau)$ which executes the sampled chunk open-loop (during training and evaluation) until arriving at the next replanning state, s_{t+N}. The actor outputs $N \cdot \dim A$ means and $N \cdot \dim \mathcal{A}$ log-standard-deviations, defining a diagonal (tanh-squashed) Gaussian over the chunk, which implies a factorized joint $\pi_\theta(A_t \mid s_t, z_\tau) = \prod_{k=0}^{N-1} \pi_\theta(a_{t+k} \mid s_t, z_\tau)$. This means the chunk-level entropy bonus is a sum of discounted, per-step entropies:

$$\mathcal{H}\big(\pi_\theta(\cdot \mid s_t, z_\tau)\big) = \sum_{k=0}^{N-1} \gamma^k \mathbb{E}_{a_{t+k}\sim\pi_\theta} \left[-\log \pi_\theta(a_{t+k} \mid s_t, z_\tau)\right], \tag{5}$$

and that the chunk-level $Q(s_t, a_{t:t+N}, z_\tau)$ function is the discounted reward $R_\tau(s_t, a_{t:t+N}) = \sum_{k=0}^{N-1} \gamma^k r_\tau(s_{t+k}, a_{t+k})$ accumulated from open-loop execution plus the bootstrapped soft value function at the boundary timestep $t + N$ as:

$$Q^\pi(s_t, a_{t:t+N}, z_\tau) = \mathbb{E}\left[R_\tau(s_t, A_t) + \gamma^N V^\pi(s_{t+N}, z_\tau) \;\middle|\; s_t, A_t\right], \tag{6}$$

where the soft value function at a replanning timestep t is defined as

$$V^\pi(s, z_\tau) = \mathbb{E}_{a_{t:t+N}\sim\pi_\theta(\cdot|s,z_\tau)} \left[Q^\pi(s, a_{t:t+N}, z_\tau) + \alpha\mathcal{H}_\gamma\big(\pi_\theta(\cdot \mid s, z_\tau)\big)\right]. \tag{7}$$

4.2 $TD(\lambda)$ TARGETS

SWAP defines two types of TD targets, each meant for different rollout length regimes.

Option $TD(\lambda)$ Target We define the 'option' TD target formulation (Sutton et al., 1999) because under open-loop execution of each action chunk, the standard formulation of $TD(\lambda)$ becomes lifted to the chunk-level, but without a learned termination function. The standard n-step return becomes an m-chunk return $G_t^{(m)}$

$$G_t^{(m)} = \sum_{j=0}^{m-1} \gamma^{jN} R_\tau\big(s_{t+jN}, A_{t+jN}\big) + \gamma^{mN} V\big(s_{t+mN}, z_\tau\big), \tag{8}$$

and the option TD target exponentially weights these m-chunk returns analogous to TD(λ). Given a finite rollout of length T, option TD targets only keep the T/N windows that start at a replanning timestep $t \bmod N = 0$, and chain the lambda return window to window. Critically, the bootstrap term incurs no bias because the policy would actually replan there in the rollout. The drawback is that discarding the other $(N-1)/N$ of windows cuts the training examples per rollout by a factor of N. For example, at $T = 8$, $N = 4$ only two windows per rollout remain, and the target of the first can mix only two returns (Table 1). As a result, we only use the option formulation for $T \geq 32$.

Step $TD(\lambda)$ Target Alternatively, the step TD target keeps all $T - N + 1$ windows (number of sliding windows). Each window's target is the chunk's N-step reward plus the per-step TD(λ) return at s_{t+N}, $G_t = R_\tau(s_t, A_t) + \gamma^N G^\lambda_{t+N}$, so that the reward signal propagates across the rollout rather than stopping at each chunk boundary, by using a single-sample estimate with Q_ϕ, which would slow learning (Appendix A.1). At $N = 1$, this is exactly the TD(λ) target.

However, this bootstrap term is biased. Through the n-step returns in Equation 3, from which G^λ_{t+N} is constructed, G^λ_{t+N} mixes single-sample values at $s_{t+N+1}, \ldots, s_T$. Each is formed by sampling a fresh action chunk from π_θ at that state. However, only every Nth state is one where π_θ actually replans during the rollout. Every other state is also mid-chunk, and the bootstrap value there will, using a freshly sampled action, optimistically inflate the value of the state, incurring bias. Intuitively, if the policy can replan mid-chunk, the return estimate from that state can be higher than the prior, conditioned state s_t it was forced to predict from.

The size of this bias depends on the rollout length. G^λ_{t+N} places weight $(1-\lambda)\lambda^{n-1}$ on the bootstrap at each intermediate state s_{t+N+n}, and the remainder, $\lambda^{T-t-N-1}$, on the bootstrap at the final state s_T according to Equation 4. Since $T \bmod N = 0$, s_T is a replanning step and the final bootstrap is unbiased. As the remainder shrinks with T, the share of λ-weights inside G^λ_{t+N} placed on biased mid-chunk bootstraps grows (Table 1).

	windows per rollout		returns mixed at t=0		step target weight	
T	step	option	step	option	on $V(s_T)$	on mid-chunk
8	5	2	4	2	0.70	0.26
16	13	4	12	4	0.46	0.43
32	29	8	28	8	0.20	0.62
64	61	16	60	16	0.04	0.74

Table 1: $N = 4$, $\lambda = 0.95$. The step target mixes one-step returns and the option target mixes chunk-level returns. The last two columns are for the window at $t = 0$. The option target places no weight on mid-chunk bootstraps.

4.3 Switched Action Chunking

With the action chunking formalism above, we can detail how to take advantage of the behavioral diversity offered by MTRL.

Environment Partitioning To balance between the stability of standard on-policy data collection and off-policy policy-switched exploration, SWAP partitions the environments assigned to each task $\tau \in \{1, \ldots, K\}$ via a shared, constant excess fraction $\rho \in [0, 1]$. The excess fraction of environments perform policy switching during data collection, and those that are not proceed to sample as normal from their true task policy.

Action Selection via Policy Switching At replanning timesteps $t \bmod N = 0$, the agent instance in the excess environment for task τ observes the normalized state s_t, strips the observation of its z_τ, and in parallel queries π_θ with that same observation by appending back all possible z_τ.

$$a^{(k)}_{t:t+N} \sim \pi_\theta(\cdot \mid s_t, z_k), \quad \forall k \in \{1, \ldots, K\}$$

The executed chunk $a^*_{t:t+N}$ maximizes the chunked critic Q_ϕ under the true task embedding z_τ

$$k^* = \arg\max_{k \in \{1,\ldots,K\}} Q_\phi\big(s_t, a^{(k)}_{t:t+N}, z_\tau\big), \quad a^*_{t:t+N} = a^{(k^*)}_{t:t+N} \tag{9}$$

The inference overhead incurred at chunk boundaries is minimal since it scales as O(K/N) forward passes.

Policy Learning Since REPPO's pathwise gradient $\nabla_\theta \mathcal{L}_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\, \xi\sim\mathcal{N}(0,I)} \left[-\nabla_A Q_\phi\big(s, g_\theta(s, z_\tau, \xi), z_\tau\big) \nabla_\theta g_\theta(s, z_\tau, \xi) + \ldots\right]$ differentiates through the

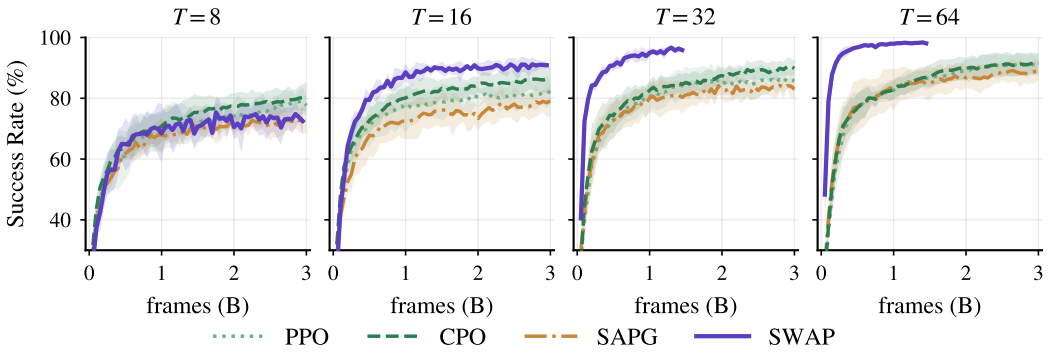


Figure 1: **Comparison of SWAP with baseline methods on MT50 across rollout horizons.** Lines show mean success rates, and shaded regions indicate 95% confidence intervals of the mean. SWAP uses action chunk length $N = 4$ and policy switching fraction $\rho = 0.75$. See Table 5 for exact numbers.

critic along actions freshly sampled from the current task policy rather than the executed behavior chunk, the actor update is decoupled from the actions taken in the rollout phase. As a result, the actor can absorb exploratory, switched actions that lead to high-Q regions without importance sampling.

5 Experiments and Results

Since learning performance in massively parallel RL is sensitive to batch size (Mayor et al., 2025), we evaluate at different rollout lengths $T \in \{8, 16, 32, 64\}$. We compare SWAP with existing baselines, examine the interaction between policy switching and action chunking, and investigate its key design choices through ablation studies. Finally, we visualize policy switching within episodes to illustrate how SWAP uses policies from different tasks during exploration.

5.1 Environment

We evaluate on MT50 from MTBench (Joshi et al., 2025), a massively parallel multi-task benchmark that reimplements all 50 Meta-World manipulation tasks (Yu et al., 2020b) in Isaac Gym (Makoviychuk et al., 2021) using a Franka Emika Panda robot. All tasks share a 39-dimensional observation, although the semantic meaning of some dimensions differs across tasks. The shared four-dimensional action space consists of three-dimensional end-effector displacement and gripper control. A one-hot task identifier is concatenated to the observation.

5.2 Baselines

We compare with prior methods that aim to resolve saturation in the single-task massively parallel regime. A brief overview of each is as follows:

SAPG (Singla et al., 2024) artificially partitions parallel environments among one leader policy and multiple follower policies to encourage diverse exploration. Each follower is trained with PPO, while the leader aggregates knowledge by using importance sampling to learn from off-policy data collected by the followers alongside its own on-policy data.

CPO (Shitanda et al., 2026) is a recently proposed method that addresses the reduced learning efficiency and stability caused by excessive inter-policy divergence in SAPG. It regulates policy diversity by introducing a KL constraint relative to the leader policy during follower updates.

Finally, we include PPO (Schulman et al., 2017) as a standard on-policy baseline. Hyperparameters for these baselines are provided in Table 3.

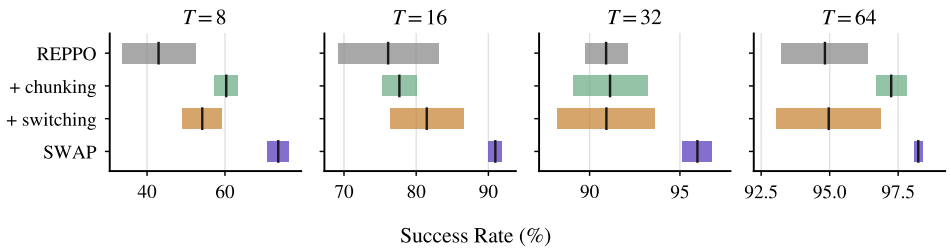


Figure 2: **Ablation of policy switching and action chunking across rollout horizons.** We compare REPPO, each component, and their combination (SWAP). Results show 95% confidence intervals of the mean success rate. See Table 6 for exact numbers.

5.3 Comparison with Baseline Methods

As shown in Figure 1, SWAP outperforms baselines for $T \in \{16, 32, 64\}$ in terms of final success rates and sample efficiency. The advantage is particularly pronounced at $T = 32$ and $T = 64$, where SWAP reaches approximately 96% and 98% success within 1.5 billion frames, surpassing the baselines even when they are trained for twice as many frames. At $T = 8$, however, SWAP performs comparably to SAPG and falls below CPO, most likely due to the relative difficulty of learning a Q function versus a value function.

5.4 Interaction Between Policy Switching and Action Chunking

In Figure 2, we observe that when testing REPPO with each component of SWAP, only when both components are combined, does SWAP achieve the highest mean success rate across all rollout horizons with much lower variance. This shows that temporal commitment through action chunking and policy switching synergize together into coherent exploration.

5.5 Ablation Studies

We conduct ablation studies to examine how SWAP's design choices affect performance: (A) the fraction of environments using policy switching, (B) the action chunk length, and (C) Q-value-based action selection.

A. Effect of excess fraction: Figure 3 investigates how the fraction of excess environments ρ affects learning performance. Increasing the ratio ρ improves performance monotonically up until all environments are switching. Similar to MT-Opt (Kalashnikov et al., 2021) and CDS (Yu et al., 2020a), which find that impersonating/sharing a task as all other tasks diminishes performance, we suspect retaining some non-negligible on-policy data is important for algorithmic stability.

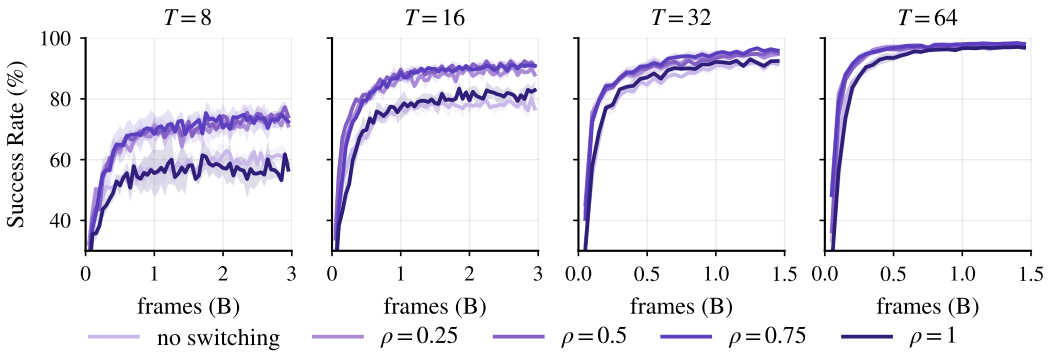


Figure 3: **Ablation of the policy switching fraction ρ across rollout horizons, with action chunk length fixed at $N = 4$.** Lines show mean success rates, and shaded regions indicate 95% confidence intervals. See Table 7 for exact numbers.

B. Effect of action chunk length: Figure 4 investigates how action chunk length N affects learning performance with a fixed excess environment ratio. Moderate chunk lengths improve performance over single-step actions ($N = 1$). Performance generally improves with larger chunk lengths (e.g at $T = 8$), but the degradation observed with longer chunks may partly reflect the limited expressivity and capacity of the actor with increased action dimensions. In addition, longer chunks reduce reactivity, which, without receding horizons, is consistent with (Nagy et al., 2026; Korniak et al., 2026).

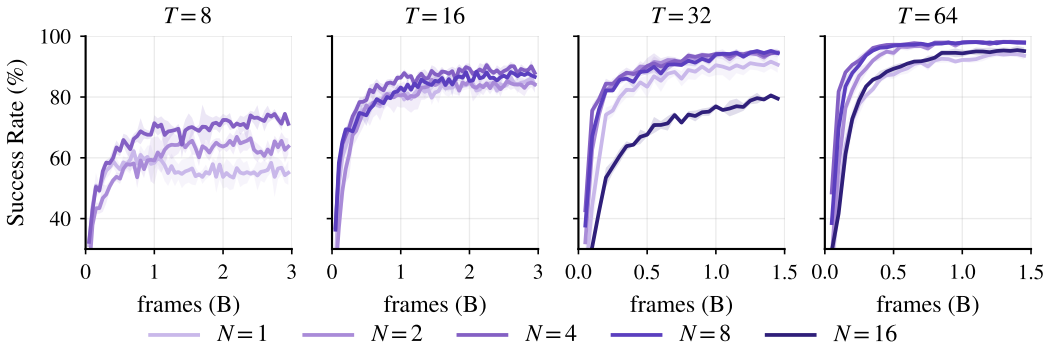


Figure 4: **Ablation of action chunk length N across rollout horizons, with policy switching fraction fixed at $\rho = 0.25$.** Lines show mean success rates, and shaded regions indicate 95% confidence intervals. See Table 8 for exact numbers.

C. Effect of Q-Guided Action Chunk Selection: Figure 5 examines the effect of each component of policy switching by comparing three conditions: (1) SWAP, which executes as in Equation 9, (2) 'random policy' where a uniformly randomly sampled task's action chunk is executed, and (3) 'own-task, Q' where 50 action chunks from the current task's policy are sampled and the maximizer of which is executed. Comparing (1) and (2), we confirm the effectiveness of Q-guided policy selection to enforce temporal commitment. Comparing (1) and (3), we confirm that the candidate set under which the action chunks were sampled is just as important as maximizing itself toward coherent, temporal commitment.

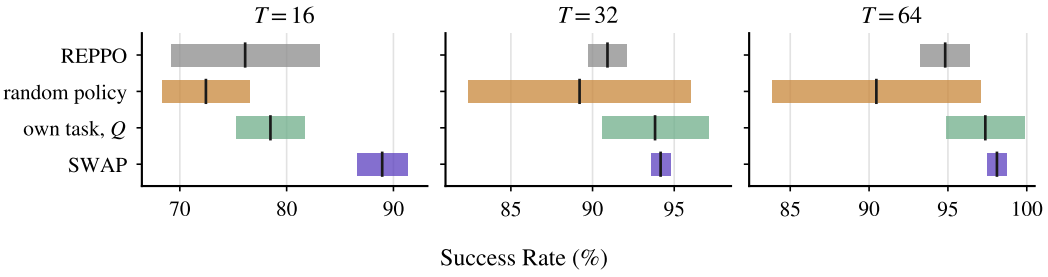


Figure 5: **Ablation on the two components of policy switching, the candidate set and the selection mechanism.** Results show 95% confidence intervals of the mean success rate. See Table 9 for exact numbers.

6 Conclusion

We introduced SWAP, an online MTRL method resolving performance saturation in the massively parallelized regime that exploits the behavioral diversity inherently present when learning multiple tasks. By utilizing the excess simulation capacity of massive parallelization to perform policy switching and action chunking to enforce temporal commitment of these exploratory behaviors, SWAP outperforms single-task baselines adapted for MTRL that artificially construct diversity. SWAP's minimal changes to data collection leave room for advances in MTRL architectures (Le et al., 2026) and optimization.

AI USE STATEMENT

In this work, we used generative AI tools for drafting, grammar checking, and editing the manuscript. Additionally, we used generative AI tools for reviewing experimental code, managing experimental runs, and visualizing experimental results. We have reviewed all AI-assisted work, checking the manuscript against the implementation and experimental results and verifying that the visualizations accurately represent the underlying data. 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 full hyperparameter details in Appendix C. In addition, we will release an open-source implementation built upon REPPO.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

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

Ben Eysenbach, Xinyang Geng, Sergey Levine, and Russ R Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in neural information processing systems*, 33:14783–14795, 2020.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

GenesisAuthors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL `https://github.com/Genesis-Embodied-AI/genesis-world`.

Jinmin He, Kai Li, Yifan Zang, Haobo Fu, Qiang Fu, Junliang Xing, and Jian Cheng. Efficient multi-task reinforcement learning with cross-task policy guidance. *Advances in Neural Information Processing Systems*, 37:117997–118024, 2024.

Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *International Conference on Learning Representations*, volume 2024, pp. 6344–6365, 2024.

Viraj Joshi, Zifan Xu, Bo Liu, Peter Stone, and Amy Zhang. Benchmarking massively parallelized multi-task reinforcement learning for robotics tasks. *arXiv preprint arXiv:2507.23172*, 2025.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Changyeon Kim, Haeone Lee, Younggyo Seo, Kimin Lee, and Yuke Zhu. Deas: Detached value learning with action sequence for scalable offline rl. In *International Conference on Learning Representations*, volume 2026, pp. 64035–64058, 2026.

Michal Korniak, Kamil Dybek, Benjamin Eysenbach, Marco Bagatella, and Michał Bortkiewicz. Three steps at a time: Learning representations from action sequences in contrastive rl, 2026. URL `https://arxiv.org/abs/2608.30640`.

Huy Le, Onur Celik, Denis Blessing, Tai Hoang, Claas A Voelcker, Axel Brunnbauer, Felix Richter, Michael Volpp, and Gerhard Neumann. Trust-region diffusion policies for massively parallel on-policy rl. *arXiv preprint arXiv:2606.15260*, 2026.

Gawon Lee, Daesol Cho, and H Jin Kim. Leveraging temporally extended behavior sharing for multi-task reinforcement learning. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9768–9775. IEEE, 2025a.

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

Alexander Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767, 2020.

Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 44534–44563, 2025.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026.

Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel q-learning: Scaling off-policy reinforcement learning under massively parallel simulation. In *International Conference on Machine Learning*, pp. 19440–19459. PMLR, 2023.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36:57226–57243, 2023.

Srinath Mahankali, Zhang-Wei Hong, Ayush Sekhari, Alexander Rakhlin, and Pulkit Agrawal. Random latent exploration for deep reinforcement learning. *arXiv preprint arXiv:2407.13755*, 2024.

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, 2021.

Walter Mayor, Johan Obando-Ceron, Aaron Courville, and Pablo Samuel Castro. The impact of on-policy parallelized data collection on deep reinforcement learning networks. *arXiv preprint arXiv:2506.03404*, 2025.

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.

CF Nagy, Onur Celik, Emiliyan Gospodinov, Florian Seligmann, Weiran Liao, Aryan Kaushik, and Gerhard Neumann. Sear: Sample efficient action chunking reinforcement learning. *arXiv preprint arXiv:2603.01891*, 2026.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *Advances in Neural Information Processing Systems*, 38:50818–50858, 2026.

Aleksei Petrenko, Arthur Allshire, Gavriel State, Ankur Handa, and Viktor Makoviychuk. Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training. *arXiv preprint arXiv:2305.12127*, 2023.

Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2161–2168. IEEE, 2017.

Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning*, 2000. URL `https://api.semanticscholar.org/CorpusID:1153355`.

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

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

Naoki Shitanda, Motoki Omura, Tatsuya Harada, and Takayuki Osa. Rethinking policy diversity in ensemble policy gradient in large-scale reinforcement learning. *arXiv preprint arXiv:2603.01741*, 2026.

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. Sapg: Split and aggregate policy gradients. *arXiv preprint arXiv:2407.20230*, 2024.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International conference on machine learning*, pp. 9767–9779. PMLR, 2021.

Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 21495–21507, 2022.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Dong Tian, Onur Celik, and Gerhard Neumann. Chunking the critic: A transformer-based soft actor-critic with n-step returns. In *International Conference on Learning Representations*, volume 2026, pp. 120056–120094, 2026.

Claas Voelcker, Axel Brunnbauer, Marcel Hussing, Michal Nauman, Pieter Abbeel, Radu Grosu, Eric Eaton, Amir-massoud Farahmand, and Igor Gilitschenski. Relative entropy pathwise policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 35761–35790, 2026.

Jianren Wang, Yifan Su, Abhinav Gupta, and Deepak Pathak. Evolutionary policy optimization. *arXiv preprint arXiv:2503.19037*, 2025.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020a.

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*, pp. 1094–1100. PMLR, 2020b.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:11501–11516, 2021.

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`.

Grace Zhang, Ayush Jain, Injune Hwang, Shao-Hua Sun, and Joseph Lim. Qmp: Q-switch mixture of policies for multi-task behavior sharing. In *International Conference on Learning Representations*, volume 2025, pp. 80488–80516, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A TD(λ) DETAILS

A.1 STEP-TARGET BOOTSTRAP

The step target runs REPPO's per-step TD(λ) sweep over the rollout, using the single-sample soft value $V(s_{t+1}, z_\tau) = Q_\phi(s_{t+1}, A', z_\tau) - \alpha \log \pi(A' \mid s_{t+1})$, $A' \sim \pi(\cdot \mid s_{t+1})$, as the bootstrap at every step. This yields G_t^λ at each t. Since G_t^λ is a soft Q-target at (s_t, a_t), it satisfies the soft Bellman equation $G_t^\lambda = r_\tau(s_t, a_t) + \gamma V(s_{t+1}, z_\tau)$, and the smoothed value at a chunk boundary is rearranged as

$$\gamma\, V(s_{t+N}, z_\tau) = G_{t+N-1}^\lambda - r_\tau(s_{t+N-1}, a_{t+N-1}). \tag{10}$$

The window target is then $G_t^{(N)} = R_\tau(s_t, A_t) + \gamma^N V(s_{t+N}, z_\tau)$. At $N = 1$ this reduces to G_t^λ exactly.

B EXPERIMENTAL DETAILS

B.1 EVALUATION PROTOCOL

We use five random seeds when making any comparison to a baseline and three for ablation studies, reporting the mean success rate and its 95% confidence interval. The success rate is computed according to Joshi et al. (2025) using 1 evaluation episode over all 24576 environments every 50M frames.

Evaluation uses stochastic policies. For methods with action chunking, each predicted chunk is executed open-loop for N steps before replanning from the latest observation, matching the chunk length used during training.

B.2 COMPUTATIONAL RESOURCES

We conducted our experiments primarily on a university cluster equipped with NVIDIA A100 GPUs. Although limited computational resources required us to use NVIDIA A40 GPUs as well, we verified that experiments run under the same configuration produced consistent results before aggregating results across clusters.

C HYPERPARAMETERS

Environment	
Tasks	MT50 (all 50)
Parallel environments	24576
Episode length	250 (no early termination)
Observation normalization	per task, running
Reward normalization	per task, running
Evaluation cadence	every 5×10^7 frames
Evaluation episodes	one per environment (≈491 per task), stochastic actions
Optimization	
Optimizer	AdamW
Learning rate	3×10^{-4} (constant)
Gradient clipping	0.5
Epochs per iteration	4
Discount γ	0.99
TD(λ) λ	0.95
Actor	
Architecture	SimbaV2 (Lee et al., 2025b)
Width / blocks	256 / 1
Block expansion, c_{shift}	4, 3.0
Standard deviation	one $\log \sigma$ per action dim, shared across the chunk
Minimum standard deviation	0
Critic	
Architecture	MLP
Width	512
Encoder / head / predictor layers	2 / 2 / 2
Distributional head	HL-Gauss, 201 bins
v_{min}	0
v_{max}	100
Layer normalization	yes
Self-predictive auxiliary loss weight	1.0
Trust region and entropy	
KL bound	0.075
KL multiplier, initial	0.01
KL clipping mode	clipped
Entropy multiplier, initial	0.01
Entropy target multiplier	0.2
Action chunking and policy switching	
Chunk length N	4
Execution length k (evaluation)	$= N$
Excess fraction ρ	0.75 (0.25 in the ablations)
Donor commitment	whole chunk
Random replanning from (Nagy et al., 2026)	off
Receding Horizon	off

Table 2: REPPO Hyperparameters, held identical across every experiment. Only the factor named in each ablation varies.

Common: environment and evaluation	
Tasks	MT50 (all 50)
Parallel environments	24,576
Episode length	250 (no termination on success)
Task identifier	50-dimensional one-hot vector
Observation normalization	Per task, running; task one-hot excluded
Value-target normalization	Per task, running
Advantage normalization	Per task
Environment reward-scale setting	100
Training budget	3×10^9 environment steps
Evaluation cadence	Every 5×10^7 environment steps
Evaluation episodes	One per environment (491–492 per task)
Evaluation actions	Deterministic
Random seeds	5 (42–46)
Common: optimization	
Optimizer	Adam
Initial learning rate	5×10^{-4}
Learning-rate schedule	KL-adaptive, target KL 0.016
Learning-rate range	$[10^{-5}, 10^{-2}]$
Gradient clipping	1.0
Epochs per iteration	4
Rollout horizon T	$\{8, 16, 32, 64\}$
Base minibatch size	16,384
Minibatches per epoch	12, 24, 48, 96 for $T = 8, 16, 32, 64$
Discount γ / GAE λ	0.99 / 0.95
PPO clipping parameter	0.2
Value-loss coefficient	4.0 (clipped value loss)
Entropy coefficient	0.005
Action-bound loss coefficient	10^{-4} (soft bound 1.0)
Common: actor and critic	
Actor architecture	MLP, hidden widths $(256, 128, 64)$
Critic architecture	MLP, hidden widths $(256, 256, 256)$
Activation	ELU
Initial action standard deviation	1.0
PPO only	
Standard deviation	Learned per action dimension, shared across tasks
SAPG and CPO	
Policy allocation	One multi-task leader and 50 task-specific followers
Leader environments	5,376 (107–108 per task)
Environments per follower	384
Policy identifier	16-dimensional sinusoidal encoding
Standard deviation	Learned per policy and action dimension
Log-standard-deviation bounds	$[-5, 2]$
Follower policies sampled for leader data	10 per minibatch
Evaluation policy	Leader only
CPO only	
AWAC loss weight β	5×10^{-4}
AWAC temperature	0.2
Maximum AWAC weight	100
Leader / follower PPO loss weights	1.0 / 1.0
Explicit leader-relative KL coefficient	0

Table 3: Hyperparameters for baseline methods. Common settings apply to PPO, SAPG, and CPO.

	$T = 8$	$T = 16$	$T = 32$	$T = 64$
Minibatches per epoch	48	48	48	96
Target construction	`step`	`step`	`option`	`option`
Training frames	3×10^9	3×10^9	1.5×10^9	1.5×10^9

Table 4: Hyperparameters that vary with the rollout length T. The minibatch count is held fixed within a horizon so that the number of gradient steps and the per-iteration KL budget do not change with N or ρ; $N = 1$ uses `step` at every horizon, where the two constructions are equivalent.

D Detailed Results

method	$T = 8$	$T = 16$	$T = 32$	$T = 64$
PPO	78.5 [75.0, 82.1]	82.0 [77.6, 86.5]	86.2 [82.9, 89.4]	91.1 [87.7, 94.5]
CPO	**80.2** [75.8, 84.7]	86.2 [81.7, 90.6]	90.2 [87.5, 93.0]	91.5 [88.0, 94.9]
SAPG	72.9 [68.8, 77.1]	78.9 [74.4, 83.4]	83.7 [81.3, 86.1]	88.7 [85.1, 92.4]
SWAP	73.7 [70.9, 76.5]	**91.0** [90.0, 91.9]	**96.0** [95.2, 96.8]	**98.2** [98.1, 98.4]

Table 5: MT50 success rate (%) across rollout lengths T, reported as mean with 95% CI in brackets over 5 seeds and averaged over the last 3 evaluations. SWAP uses $N = 4$, $\rho = 0.75$.

	$T = 8$	$T = 16$	$T = 32$	$T = 64$
REPPO (N=1)	43.0 [33.5, 52.4]	76.1 [69.2, 83.1]$^{(3)}$	90.9 [89.7, 92.1]	94.8 [93.2, 96.4]
+ chunking (N=4)	60.3 [57.3, 63.3]	77.7 [75.3, 80.1]	91.1 [89.1, 93.2]	97.2 [96.7, 97.8]
+ switching (ρ=0.75)	54.2 [49.1, 59.2]	81.5 [76.4, 86.6]	90.9 [88.2, 93.6]	95.0 [93.1, 96.9]
SWAP	**73.7** [70.9, 76.5]	**91.0** [90.0, 91.9]	**96.0** [95.2, 96.8]	**98.2** [98.1, 98.4]

Table 6: Ablation of action chunking and policy switching at each rollout length. MT50 success rate (%), reported as mean with 95% CI in brackets over 5 seeds, unless marked $^{(n)}$ and averaged over the last 3 evaluations.

ρ	$T = 8$	$T = 16$	$T = 32$	$T = 64$
0 (off)	60.3 [57.3, 63.3]$^{(5)}$	77.7 [75.3, 80.1]$^{(5)}$	91.1 [89.1, 93.2]$^{(5)}$	97.2 [96.7, 97.8]$^{(5)}$
0.25	72.3 [70.4, 74.1]	88.9 [86.6, 91.3]	94.2 [93.6, 94.8]	98.1 [97.5, 98.7]
0.5	**75.5** [69.2, 81.8]	**91.4** [87.3, 95.5]	95.1 [93.4, 96.8]$^{(5)}$	97.3 [94.7, 100.0]
0.75	73.7 [70.9, 76.5]$^{(5)}$	91.0 [90.0, 91.9]$^{(5)}$	**96.0** [95.2, 96.8]$^{(5)}$	**98.2** [98.1, 98.4]$^{(5)}$
1	57.2 [55.9, 58.6]	82.7 [76.8, 88.6]	92.1 [91.9, 92.4]	97.0 [95.2, 98.8]

Table 7: MT50 success rate (%) over ρ, reported as mean with 95% CI in brackets over 3 seeds, unless marked $^{(n)}$, and averaged over the last 3 evaluations.

N	$T = 8$	$T = 16$	$T = 32$	$T = 64$
1	55.4 [46.8, 64.0]	85.2 [79.8, 90.7]	90.9 [83.5, 98.3]	93.9 [91.0, 96.8]
2	64.0 [54.5, 73.4]	84.5 [78.7, 90.3]	94.1 [89.9, 98.2]	98.0 [97.9, 98.1]
4	**72.3** [70.4, 74.1]	**88.9** [86.6, 91.3]	94.2 [93.6, 94.8]	**98.1** [97.5, 98.7]
8	—	87.2 [85.2, 89.2]	**94.7** [92.5, 96.9]	97.9 [97.5, 98.3]
16	—	—	79.7 [78.7, 80.8]	95.2 [91.7, 98.8]

Table 8: MT50 success rate (%) over N, reported as mean with 95% CI in brackets over 3 seeds, unless unless marked $^{(n)}$, and averaged over the last 3 evaluations, at $\rho = 0.25$.

	$T = 16$	$T = 32$	$T = 64$
REPPO (N=1, no switching)	76.1 [69.2, 83.1]	90.9 [89.7, 92.1]$^{(5)}$	94.8 [93.2, 96.4]$^{(5)}$
uniform random selection	72.4 [68.3, 76.6]	89.2 [82.4, 96.0]	90.5 [83.9, 97.1]
own task only, Q-argmax over 50 samples	78.5 [75.3, 81.7]$^{(4)}$	93.8 [90.6, 97.1]	97.4 [94.9, 99.9]
SWAP (Q-argmax over 50 task policies)	**88.9** [86.6, 91.3]	**94.2** [93.6, 94.8]	**98.1** [97.5, 98.7]

Table 9: MT50 success rate (%) over Q-guided selection and cross-task behavioral diversity at $N = 4$, $\rho = 0.25$. Uniform random selection replaces the Q-argmax with a uniformly drawn task policy. Own-task sampling draws 50 chunks from the current task's policy and keeps the Q-maximizer. Reported as mean with 95% CI in brackets over 3 seeds, unless marked $^{(n)}$, and averaged over the last 3 evaluations.

E ADDITIONAL RESULTS

E.1 HOW DOES POLICY SWITCHING SWITCH OVER AN EPISODE?

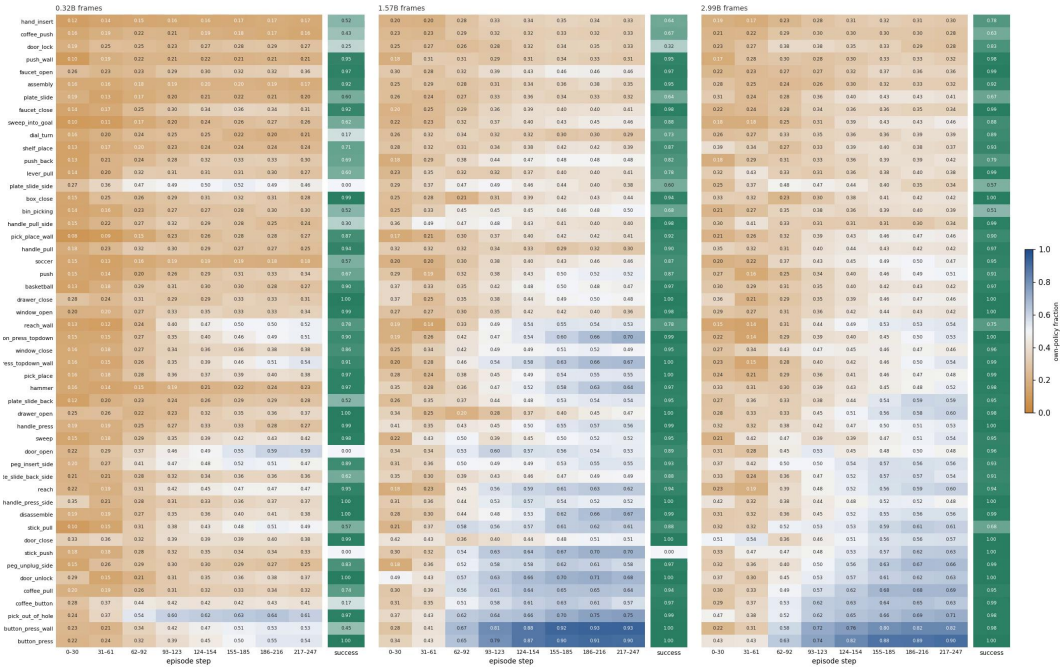


Figure 6: **Comparison of the fraction of the time a policy switches per-task as training proceeds for SWAP using** $T = 16, N = 4$

In Figure 6, we derive the fraction of time an agent executed an action sampled from its own task policy by tracking which policy actually produced each executed action. On the excess fraction ρ of environments, the critic scores candidate chunks from all tasks, so a task's own policy is executed only when it maximizes its own Q. We divide each 250-step episode into eight equal intervals and, for every task and interval, measure the fraction of executed decisions that used the task's own policy. For each task and interval, we track how often the agent chose to act as itself rather than borrow another task's policy. The reported evaluation in the last column of each block is extracted at the timestep indicated in the top left of each block.

Throughout training, we note that the start of an episode across all tasks is heavily shared, as all policies share a common behavior of moving to begin manipulating the object placed in front of them. However, as training progresses, some tasks begin to specialize towards the end of their episode, as seen in the dark blue regions. Interestingly, a policy can retain very similar evaluation performance despite the percentage of switching over an episode changing very drastically (e.g. see the last row of each block), due to the stabilizing presence of on-policy data.

E.2 WHICH TASKS ARE DONORS OR RECEIVERS DURING POLICY SWITCHING?

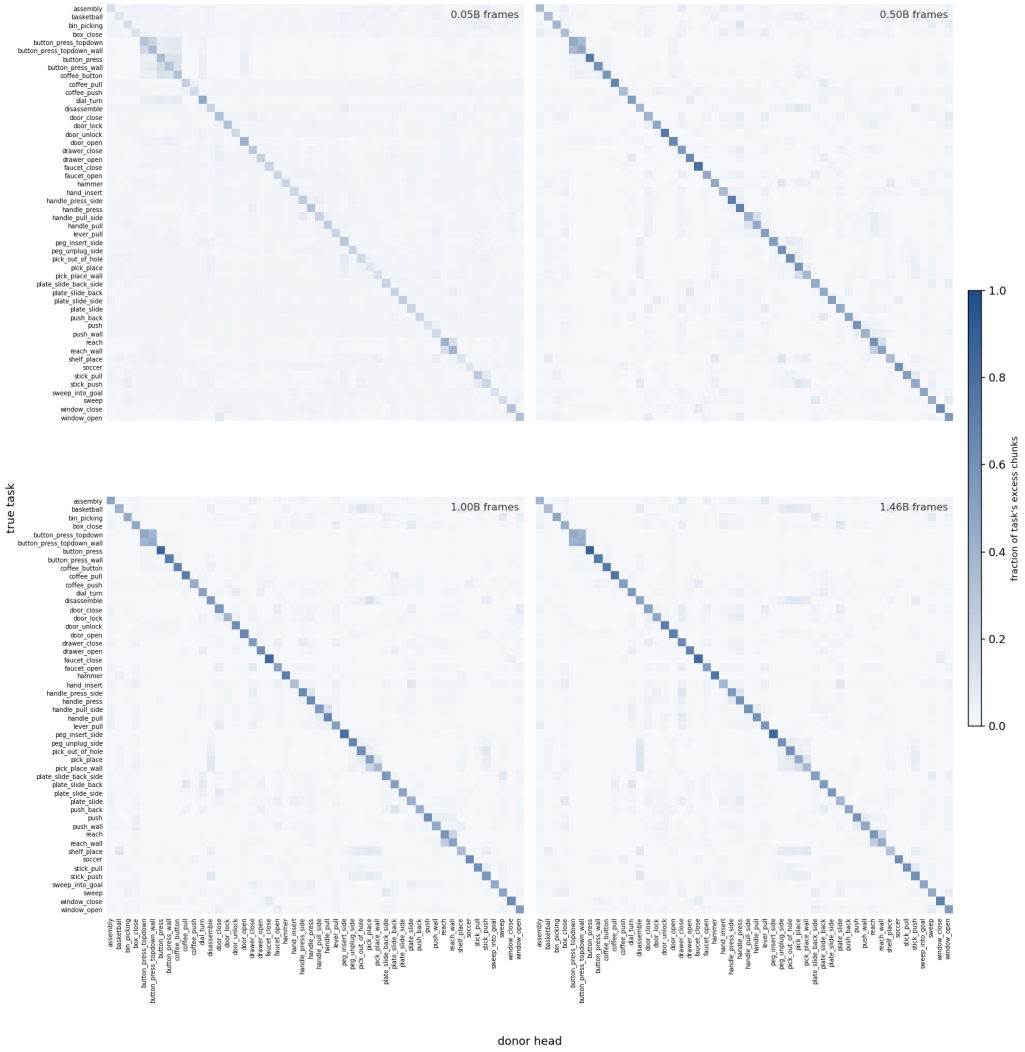


Figure 7: **Donor heatmap for SWAP using** $T = 16, N = 4$

In Figure 7, each heatmap is generated from the last logged iteration of training. Row i, column j is the fraction of task i's switched chunks produced by task j's policy. Rows sum to 1, and the diagonal is the own-policy rate.