Distributionally Robust Multi-Task Reinforcement Learning via Adaptive Task Sampling

Nicholas E. Corrado
Computer Sciences Department
University of Wisconsin – Madison
ncorrado@wisc.edu

Wenyuan Huang
Computer Sciences Department
University of Wisconsin – Madison
whuang369@wisc.edu

Josiah P. Hanna
Computer Sciences Department
University of Wisconsin – Madison
jphanna@cs.wisc.edu

Abstract

Multi-task reinforcement learning (MTRL) aims to train a single agent to efficiently optimize performance across multiple tasks simultaneously. However, jointly optimizing all tasks often yields imbalanced learning: agents quickly solve easy tasks but learn slowly on harder ones. While prior work primarily attributes this imbalance to conflicting task gradients and proposes gradient manipulation or specialized architectures to address it, we instead focus on a distinct and under-explored challenge: *imbalanced data allocation*. Standard MTRL allocates an equal number of environment interactions to each task, which over-allocates data to easy tasks that require relatively few interactions to solve and under-allocates data to hard tasks that require substantially more experience to solve. To address this challenge, we introduce **D**istributionally **R**obust **A**daptive **T**ask **S**ampling (DRATS), an algorithm that adaptively prioritizes sampling tasks furthest from being solved. We derive DRATS by formalizing MTRL as a feasibility problem from which we derive a minimax objective for minimizing the worst-case return gap, the difference between a desired target return and the agent's return on a task. In benchmarks like MetaWorld-MT10 and MT50, DRATS improves data efficiency and increases worst-task performance compared to existing task sampling algorithms.

1 Introduction

Multi-task reinforcement learning (RL) promises improved data efficiency by training a single agent to simultaneously solve many tasks. In principle, training on many structurally similar tasks enables knowledge transfer, reducing the number of interactions required to achieve strong performance [Brunskill and Li, 2013]. In practice, however, jointly optimizing multiple tasks leads to imbalanced learning: agents quickly solve easy tasks but learn slowly on harder ones. Prior work primarily attributes this imbalance to conflicting task gradients [Yu et al., 2020a, Liu et al., 2021a, Navon et al., 2022, Liu et al., 2023], motivating methods that manipulate gradients, reweight the learning objective, or use specialized architectures to facilitate transfer while mitigating conflict [Yang et al., 2020, Sodhani et al., 2021, Sun et al., 2022, Hendawy et al., 2023]. In this work, we instead focus on a distinct and under-explored cause of imbalanced multi-task learning: *imbalanced data allocation*.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Standard multi-task training allocates equal interaction budgets to all tasks. When tasks vary in difficulty, this allocation is inefficient: hard tasks require more experience to solve than easy ones [Cho et al., 2024]. Uniform task interaction therefore over-allocates data to easy tasks and under-allocates it to hard ones, ultimately causing imbalanced learning. Curriculum learning (CL) methods [Narvekar et al., 2020] adapt the task-sampling distribution over time, but typically prioritize easy tasks first assuming that positive transfer will facilitate learning on harder tasks. In settings where tasks do not exhibit significant positive transfer, this strategy can exacerbate data imbalance by over-allocating data to easy tasks that need it least while neglecting harder tasks that need it most.

These observations suggest that a data-efficient multi-task learner should prioritize interacting with tasks *furthest from being solved*. Distributionally robust optimization (DRO) methods take a similar perspective, optimizing worst-task performance by up-weighting the learning objective for tasks with the largest loss [Sagawa et al., 2019, Oren et al., 2019]. While DRO has been studied extensively in supervised learning [Corrado et al., 2025, Xie et al., 2024, He et al., 2024, Albalak et al., 2023, Xia et al., 2023], extensions to multi-task RL are relatively under-explored [Mandal et al., 2025, Panaganti et al., 2026]. Moreover, most existing DRO methods prioritize tasks by reweighting the learning objective while sampling tasks uniformly and thus still suffer from data imbalance.

In this work, we introduce a DRO-inspired multi-task learning algorithm that balances performance across all tasks in a data efficient manner. We first formalize multi-task RL as a feasibility problem in which the agent must reach a target return in each task and then derive a minimax optimization that minimizes the maximum *return gap*, the difference between a desired target return on a task and the agent's current return. Our algorithm, **D**istributionally **R**obust **A**daptive **T**ask **S**ampling (DRATS), adapts the task-sampling distribution to prioritize interacting with tasks in which the agent is furthest from its target returns. Because DRATS only modifies data collection, it can be combined with existing multi-task learning methods, such as those that use specialized architectures (*e.g.*, [Hendawy et al., 2023]). Empirically, DRATS yields more data efficient learning, higher aggregate return, and higher worst-task return on MetaWorld-MT10 and MT50 benchmarks [Yu et al., 2020b, McLean et al., 2025a] as well as a multi-task version of the standard 6-task MuJoCo benchmark [Todorov et al., 2012]. In summary, our contributions are:

1. We introduce DRATS, a multi-task RL algorithm that adaptively prioritizes sampling tasks with the largest return gap. Using standard online learning techniques, we prove that DRATS converges to within ϵ error of the optimal worst-task return gap at a standard $O(1/\sqrt{T})$ rate.
2. We empirically demonstrate that DRATS achieves greater data efficiency, higher aggregate return, and higher worst-task return than existing curriculum learning baselines.
3. We empirically demonstrate that DRATS is compatible with multi-task network architectures and improves data efficiency whether tasks have positive transfer or no transfer at all.

2 Related Work

Curriculum Learning. Curriculum learning (CL) [Bengio et al., 2009, Narvekar et al., 2020] improves data efficiency by controlling the order in which an agent samples training data or tasks. Most CL methods target *single-task* settings, decomposing a single complex task into a sequence of simpler sub-tasks [Andrychowicz et al., 2017, Florensa et al., 2018, Ren et al., 2019, Pong et al., 2019, Eysenbach et al., 2020, Zhang et al., 2021, Shah et al., 2021, Chane-Sane et al., 2021, Zhang et al., 2020, Jiang et al., 2021, Huang et al., 2022a, Cho et al., 2023, Liu et al., 2024]. We instead consider *multi-task* settings where tasks are not necessarily sub-problems of a single task and thus may not exhibit positive transfer or may even conflict with each other. Existing multi-task CL methods often adapt the task distribution heuristically using return or success rate thresholds [Wang et al., 2019, 2020, Cho et al., 2024] or by prioritizing tasks with the most learning progress [Matiisen et al., 2019, Mysore et al., 2019, Colas et al., 2019, Portelas et al., 2020, Kanitscheider et al., 2021]. Most prioritize easy tasks first, assuming positive transfer will accelerate learning on harder ones, which is not necessarily the most data-efficient approach when transfer is weak or tasks conflict. Moreover, balanced task performance is not a direct objective of these methods: improved worst-task performance is a hoped-for byproduct of transfer, whereas in DRATS it is the optimization target. The most closely related CL method is SMT [Cho et al., 2024], which prioritizes tasks with the smallest returns via manually-specified thresholds. DRATS has a similar goal but automatically adapts the task-sampling distribution via minimax optimization and has convergence guarantees that SMT lacks.

Distributionally Robust Optimization. DRATS is a distributionally robust optimization (DRO) method. In supervised learning, DRO methods minimize the worst-case loss across tasks and are widely used to determine the most effective way to combine data from different sources or tasks [Oren et al., 2019, Sagawa et al., 2019, Michel et al., 2021, Albalak et al., 2023, Xia et al., 2023, Xie et al., 2024, He et al., 2024]. In RL, DRO has been applied to RLHF with LLMs [Mandal et al., 2025, Panaganti et al., 2026], robust RL [Xu et al., 2023, Cui et al., 2026], and unsupervised environment design [Monette et al., 2025]. Most of these methods reweight the training objective to up-weight high-loss tasks while sampling tasks uniformly, so data imbalance is still a source of data inefficiency.[1] Our work is most closely related to methods that adapt the sampling distribution rather than the objective. Corrado et al. [2025] and Albalak et al. [2023] adapt the training data mixture for supervised LLM alignment and pretraining, whereas DRATS adapts data collection for continuous control. Monette et al. [2025] adapt the sampling distribution over environment configurations to improve generalization to unseen configurations, whereas DRATS balances performance across a fixed set of training tasks.

Gradient Manipulation. A core challenge in multi-task learning is *gradient conflict*: gradients from different tasks may point in opposing directions, so updating in the direction of the average gradient can impede learning on some tasks. Several methods mitigate conflict by finding a new gradient direction that simultaneously improves all tasks [Désidéri, 2012, Yu et al., 2020a, Liu et al., 2021a] or balances the rate of progress across tasks [Liu et al., 2021b, Navon et al., 2022, Liu et al., 2023]. DRATS is orthogonal to these methods: it only changes how much data is used to compute task gradients, leaving the expected gradient direction and magnitude unchanged. Gradient manipulation and DRATS therefore target distinct sources of imbalanced learning.

Network Architectures. Multi-task network architectures enable task transfer while also mitigating conflict. CARE [Sodhani et al., 2021] uses contextual attention to dynamically weight shared representations. PaCo [Sun et al., 2022] decomposes the model into shared and task-specific parameters. Soft Modularization [Yang et al., 2020] and MOORE [Hendawy et al., 2023] use mixture-of-experts to encode task representations. McLean et al. [2025b] find that simply scaling the number of parameters in multi-task architectures significantly improves performance. DRATS only adapts the amount of data collected for each task and can be applied on top of multi-task network architectures.

3 Preliminaries: Multi-Task Reinforcement Learning

We formalize an RL environment as a finite horizon Markov decision process (MDP) [Puterman, 2014] $(\mathcal{S}, \mathcal{A}, p, r, d^0, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution d^0, and discount factor $\gamma \in [0, 1)$. In the multi-task RL setting, we work with a collection of k environments (or *tasks*) indexed by $i \in \mathcal{T} = \{1, \ldots, k\}$. For notational simplicity, we assume tasks share the same state and action spaces but can have different initial state distributions d_i^0, transition dynamics p_i, and reward functions r_i. We consider a parameterized family of stochastic, task-conditioned policies $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}, i)$, denoting the probability of selecting action $\boldsymbol{a}$ in state $\boldsymbol{s}$ in task i. We condition policies on a task by appending a one-hot encoded task identifier to the state. We denote the expected discounted return of $\pi_{\boldsymbol{\theta}}$ in a specific task i as $J_i(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{H} \gamma^t r_i(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$, where τ is a trajectory sampled from $\pi_{\boldsymbol{\theta}}$ and H is a random variable denoting the episode horizon. The standard MTRL objective is to learn a single policy that maximizes the expected discounted return averaged across tasks, $J(\boldsymbol{\theta}) = \frac{1}{k} \sum_{i=1}^{k} J_i(\boldsymbol{\theta})$.

4 Distributionally Robust Adaptive Task Sampling (DRATS)

Maximizing the expected return averaged across tasks can yield imbalanced task performance, since an agent can achieve a high aggregate return by excelling on easy tasks while performing poorly on others. In this section, we instead formalize multi-task RL as a feasibility problem in which the agent must reach a target return in each task, and show that this perspective yields an adaptive task-sampling algorithm that prioritizes tasks based on the gap between the agent's current return and target return.

[1] We empirically validate this statement in Appendix F.1.

4.1 Multi-Task RL as a Feasibility Problem

Suppose we have a vector of *reference returns* $\boldsymbol{J}^{\text{ref}} = (J_1^{\text{ref}}, \ldots, J_k^{\text{ref}})$ representing a desirable return in each task. For now, we treat $\boldsymbol{J}^{\text{ref}}$ as known and focus on deriving a new optimization problem; we discuss how to determine $\boldsymbol{J}^{\text{ref}}$ later. We formalize the multi-task objective as a feasibility problem:

$$\text{Find } \boldsymbol{\theta} \quad \text{s.t.} \quad J_i(\boldsymbol{\theta}) \geq J_i^{\text{ref}}, \quad \forall i \in [k]. \tag{1}$$

We note that a solution to Eq. 1 does not maximize the standard MTRL objective $J(\boldsymbol{\theta})$ unless each J_i^{ref} equals the maximum achievable return in task i. Nevertheless, in Section 5, we show that the algorithm we derive from this feasibility problem achieves larger $J(\boldsymbol{\theta})$ than optimizing $J(\boldsymbol{\theta})$ directly.

We transform Eq. 1 into a minimization over constraint violations by defining the *return gap* $g_i(\boldsymbol{\theta}) \coloneqq \max\left\{J_i^{\text{ref}} - J_i(\boldsymbol{\theta}), 0\right\}$, which is positive when the constraint is violated and zero once it is satisfied.[2] A policy solves all tasks if and only if $g_i(\boldsymbol{\theta}) = 0$ for all i. When constraints cannot be simultaneously satisfied—due to limited model capacity or conflicting task objectives—we assume equal task importance and spread violations across tasks as evenly as possible. This goal is naturally captured by an exact penalty formulation which minimizes the maximum violation:

$$\min_{\boldsymbol{\theta}} \max_{i \in [k]} g_i(\boldsymbol{\theta}). \tag{2}$$

We can rewrite the inner maximization as a maximization over distributions of tasks:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{q} \in \Delta^k} \mathbb{E}_{i \sim \boldsymbol{q}} \left[g_i(\boldsymbol{\theta}) \right]. \tag{3}$$

where $\Delta^k = \left\{ \boldsymbol{q} \in \mathbb{R}^k \mid \sum_{i=1}^k q_i = 1, q_i \geq 0 \right\}$ denotes the k-dimensional probability simplex. Eq. 3 recasts multi-task RL as an active data collection problem: $\boldsymbol{q}$ defines the probability of interacting with each task, and the inner maximization adapts $\boldsymbol{q}$ to prioritize tasks with the largest return gaps. Solving the inner maximization over the full simplex Δ^k leads to a solution $\boldsymbol{q}^*$ that concentrates all probability on the task with the largest gap. This solution has three drawbacks: **(1) Non-smoothness.** Since the most-violated task shifts throughout optimization, Eq. 3 is generally non-smooth, a known source of instability in minimax optimization [Razaviyayn et al., 2020]. **(2) Data inefficiency.** Optimizing only the most-violated task at each update is greedy coordinate descent: it reduces that task's return gap but provides no guarantee of progress on others, which is inefficient when tasks share structure. **(3) Return gap estimation.** Return gaps $g_i(\boldsymbol{\theta})$ must be estimated online. If a task's probability q_i becomes too small, its gap estimate will be high variance. In the next section, we address these drawbacks by restricting the space of possible task distributions.

4.2 KL-Regularized Task Sampling

To address these challenges, we restrict the task distribution $\boldsymbol{q}$ to lie within a KL ball of radius ε centered at a base distribution $\boldsymbol{p}_0$ which we take to be the uniform distribution, yielding the following constrained optimization problem:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{q} \in \mathcal{Q}} \mathbb{E}_{i \sim \boldsymbol{q}}[g_i(\boldsymbol{\theta})], \qquad \text{where} \qquad \mathcal{Q} = \left\{ \boldsymbol{q} \in \Delta^k \mid \mathrm{KL}(\boldsymbol{q} \| \boldsymbol{p}_0) \leq \varepsilon \right\}. \tag{4}$$

The KL constraint prevents $\boldsymbol{q}$ from collapsing all probability onto a single task. We derive a closed-form solution to the inner maximization of Eq. 4 following a procedure similar to Peters et al. [2010], Abdolmaleki et al. [2018], Peng et al. [2019]. We provide a brief sketch here and a complete derivation in Appendix A. We first form the unconstrained Lagrangian relaxation of Eq. 4:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{q} \in \Delta^k} \left\{ \mathbb{E}_{i \sim \boldsymbol{q}}[g_i(\boldsymbol{\theta})] - \frac{1}{\eta} \mathrm{KL}(\boldsymbol{q} \| \boldsymbol{p}_0) \right\}, \tag{5}$$

where $\eta > 0$ is the inverse Lagrange multiplier. By strong duality, Eq. 4 and Eq. 5 have the same solution for an appropriate choice of η. Differentiating the objective in Eq. 5 with respect to $\boldsymbol{q}$, setting to zero, and solving for the optimal $\boldsymbol{q}^*$ yields:

$$q_i^* = \frac{\exp\left(\eta g_i(\boldsymbol{\theta})\right)}{\sum_{j=1}^k \exp\left(\eta g_j(\boldsymbol{\theta})\right)}, \qquad \text{or equivalently,} \qquad \boldsymbol{q}^* = \mathrm{softmax}(\eta \boldsymbol{g}(\boldsymbol{\theta})) \tag{6}$$

[2] For any task whose constraint is violated, minimizing the return gap is equivalent to maximizing the return.

Algorithm 1 Distributionally Robust Adaptive Task Sampling (DRATS)

1: **Inputs:** DRATS learning rate $\alpha \in [0, 1]$, minimum sampling probability $q_{\min}$
2: **Outputs:** Trained model parameters $\boldsymbol{\theta}$.
3: Initialize model parameters $\boldsymbol{\theta}$, task distribution $\boldsymbol{q}_1 = (q_1, \ldots, q_k) \leftarrow (\frac{1}{k}, \ldots, \frac{1}{k})$, buffer $\mathcal{D} \leftarrow \emptyset$
4: **for** iteration $t = 1, \ldots, T$ **do**
5: **while** $\mathcal{D}$ is not full **do**
6: Sample task $i \sim \boldsymbol{q}_t$, collect a trajectory from task i, and add it to $\mathcal{D}$.
7: **end while**
8: Compute a Monte Carlo estimate of the return $\widehat{J}_i$ for each task $i \in [k]$
9: Update reference returns $\boldsymbol{J}^{\text{ref}}$ (if applicable)
10: Compute the normalized return gap $\widehat{g}_i = (J_i^{\text{ref}} - \widehat{J}_i)/(J_i^{\text{ref}} - J_i^{\text{rand}})$ for each task $i \in [k]$.
11: Update $\boldsymbol{q}_t$ via exponentiated gradient ascent on $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{q}) = \mathbb{E}_{i\sim\boldsymbol{q}}[g_i(\boldsymbol{\theta})] - \frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0)$:

$$[\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)]_i = g_i(\boldsymbol{\theta}) - \frac{1}{\eta}\left(\log\frac{q_{t,i}}{p_{0,i}} + 1\right) \qquad q_{t+1,i} = \frac{q_{t,i}\exp\left(\alpha[\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)]_i\right)}{\sum_{j=1}^k q_{t,j}\exp\left(\alpha[\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)]_j\right)}$$

12: Enforce minimum sampling probability: $\boldsymbol{q}_{t+1} \leftarrow \arg\min_{\boldsymbol{q}\in\Delta^k,\, \boldsymbol{q}\geq q_{\min}} \mathrm{KL}(\boldsymbol{q}\|\boldsymbol{q}_{t+1})$
13: Update $\boldsymbol{\theta}$ with an on-policy algorithm that tries to minimize $\mathbb{E}_{i\sim\boldsymbol{q}}[g_i(\boldsymbol{\theta})] - \frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0)$.
14: **end for**

Here, η acts as an inverse temperature controlling the sharpness of $\boldsymbol{q}^*$: as $\eta \to \infty$, $\boldsymbol{q}^*$ concentrates all mass on the task with the largest return gap, recovering the unregularized solution of Eq. 3; as $\eta \to 0$, $\boldsymbol{q}^*$ approaches the uniform distribution. KL regularization smooths the optimization by shifting the task distribution gradually rather than abruptly and maintains nonzero probability on all tasks (for finite η), enabling an agent to learn many tasks simultaneously and reliably estimate return gaps. We optimize Eq. 5 using Algorithm 1, interleaving gradient updates to $\boldsymbol{q}$ and $\boldsymbol{\theta}$ similar to Sagawa et al. [2019], Xie et al. [2024], Corrado et al. [2025]. Specifically, we fill the agent's buffer by sampling tasks from $\boldsymbol{q}$, compute Monte Carlo return estimates for each task, and then update $\boldsymbol{q}$ via mirror ascent [Shalev-Shwartz et al., 2012] on $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{q}) = \mathbb{E}_{i\sim\boldsymbol{q}}[g_i(\boldsymbol{\theta})] - \frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0)$:

$$q_{t+1,i} = \frac{q_{t,i}\exp\left(\alpha[\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)]_i\right)}{\sum_{j=1}^k q_{t,j}\exp\left(\alpha[\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)]_j\right)} \qquad [\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)]_i = g_i(\boldsymbol{\theta}) - \frac{1}{\eta}\left(\log\frac{q_{t,i}}{p_{0,i}} + 1\right) \tag{7}$$

where α is the step size. For completeness, we formally describe the mirror ascent update in Appendix B. Since sampling probabilities can approach zero even with KL regularization if η is large, we project $\boldsymbol{q}$ onto the simplex constrained to $[q_{\min}, 1]^k$ via a KL projection after each update (Line 12). Since returns across tasks may differ in scale, we also normalize return gaps to be in $[0, 1]$ (Line 10).

Choosing Reference Returns. Assuming tasks do not fundamentally conflict, $\boldsymbol{J}^{\text{ref}}$ should ideally denote the maximum achievable return in each task.[3] Maximum returns are rarely known in advance, so we estimate them online as the maximum observed return. If the agent is stuck in a low-return plateau, we note that the observed maximum return may underestimate the true maximum. When tasks have a notion of success, we resolve this issue by initializing $J_i^{\text{ref}} = r_{\max} H_{\max}$, where $r_{\max}$ is the maximum per-step reward and $H_{\max}$ is the maximum episode length, and then updating the reference to the maximum observed return once the agent's success rate exceeds, *e.g.*, 0.5, at which point the agent is solving the task and the observed maximum is reliable. Requiring an estimate of the maximum return is no stronger an assumption than those made by closely related baselines, which require user-specified return or success rate thresholds [Cho et al., 2024, Kanitscheider et al., 2021, Wang et al., 2020]; a key distinction is that we can estimate ours *online*.

4.3 Convergence Analysis

We now show that DRATS converges to the minimax optimal worst-task return gap at a rate of $O(1/\sqrt{T})$ under standard assumptions. Our analysis uses standard online learning proof techniques [Duchi, 2023, Shalev-Shwartz et al., 2012] and is similar to the analysis of Corrado et al.

[3] When tasks conflict and their maxima cannot be simultaneously achieved, the user may specify lower reference returns for some tasks to encode preferences over performance tradeoffs.

[2025], Sagawa et al. [2019]. We model the optimization of Eq. 5 as a zero-sum game between two players, the $\boldsymbol{\theta}$-player and the $\boldsymbol{q}$-player. For each round t, the $\boldsymbol{q}$-player chooses $\boldsymbol{q}_t \in \Delta^k$, the $\boldsymbol{\theta}$-player chooses $\boldsymbol{\theta}_t \in \Theta$, and then the $\boldsymbol{q}$-player observes the vector of task gaps $g(\boldsymbol{\theta}_t) = (g_1(\boldsymbol{\theta}_t), \dots, g_k(\boldsymbol{\theta}_t))$ and runs mirror ascent on $\mathcal{L}$. For the $\boldsymbol{\theta}$-player, we require the following regret guarantee.

Definition 4.1. *The $\boldsymbol{\theta}$-player is C-low-regret if there exists $C > 0$ such that for any sequence $\boldsymbol{q}_1, \dots, \boldsymbol{q}_T \in \Delta^k$,*

$$\sum_{t=1}^{T} \langle \boldsymbol{q}_t, g(\boldsymbol{\theta}_t) \rangle \leq \min_{\boldsymbol{\theta} \in \Theta} \sum_{t=1}^{T} \langle \boldsymbol{q}_t, g(\boldsymbol{\theta}) \rangle + C\sqrt{T}. \tag{8}$$

This assumption is satisfied by any no-regret online learning algorithm such as online gradient descent [Shalev-Shwartz et al., 2012]—which in the context of RL is equivalent to REINFORCE [Williams, 1992].[4] Under this assumption, DRATS achieves the following convergence guarantee:

Theorem 4.2 (Convergence of DRATS)**.** *Suppose $g_i(\boldsymbol{\theta}) \in [0, M]$ for all $i \in [k]$ and $\boldsymbol{\theta} \in \Theta$, and that the $\boldsymbol{\theta}$-player is C-low-regret (Definition 4.1). For any target approximation error $\epsilon > 0$ w.r.t. $\min_{\boldsymbol{\theta} \in \Theta} \max_{i \in [k]} g_i(\boldsymbol{\theta})$ (Eq. 2), set $\eta = \frac{2 \log k}{\epsilon}$ and $\alpha = \frac{\sqrt{2 \log k}}{G\sqrt{T}}$ where $G := M + \frac{1 + \max(\eta M, \log k)}{\eta}$. Then for any $T \geq \frac{4(G\sqrt{2 \log k} + C)^2}{\epsilon^2}$, DRATS satisfies:*

$$\max_{i \in [k]} \frac{1}{T} \sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \leq \min_{\boldsymbol{\theta} \in \Theta} \max_{i \in [k]} g_i(\boldsymbol{\theta}) + \epsilon. \tag{9}$$

Proof. See Appendix C. □

5 Experiments

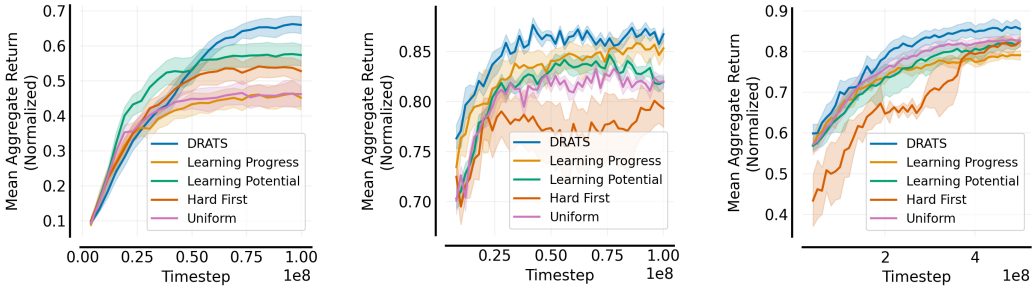
We design experiments to test two core hypotheses: **(H1)** DRATS will achieve a higher return aggregated across all tasks than baselines, or, if all methods converge to similar returns, DRATS will reduce the number of interactions required to match the returns of baselines; **(H2)** DRATS will achieve higher worst-task returns than baselines.

Environments. We evaluate on four benchmarks of increasing complexity: (1) a 4-task Gridworld (Fig. 5a); (2) MuJoCo6, comprising Swimmer-v4, Hopper-v4, HalfCheetah-v4, Walker2d-v4, Ant-v4, and Humanoid-v4 [Todorov et al., 2012]; (3) MetaWorld-MT10; and (4) MetaWorld-MT50, suites of 10 and 50 robotic manipulation tasks with a Sawyer arm [Yu et al., 2020b, McLean et al., 2025a].

Baselines. We use several task-sampling baselines listed below. Since their original names generally do not reflect their prioritization strategy, we rename each baseline based on how it prioritizes tasks.

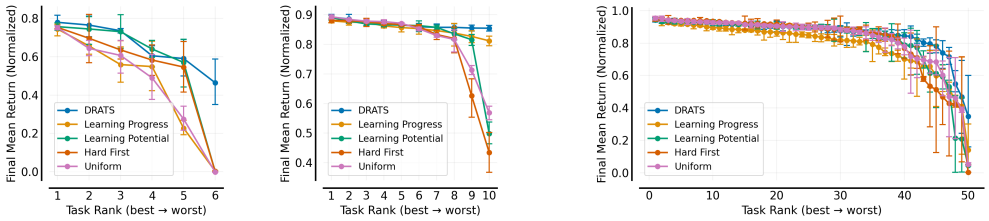
1. **UNIFORM.** Tasks are sampled from a fixed uniform distribution throughout training.
2. **LEARNING PROGRESS (*e.g.*, ALP-GMM, Portelas et al. [2020]; RIAC, Baranes and Oudeyer [2009]).** Prioritizes tasks with the largest absolute change in Monte Carlo return between consecutive updates (*i.e.*, the slope of the learning curve): $z_i = |\widehat{J}_{t,i} - \widehat{J}_{t-1,i}|$.
3. **LEARNING POTENTIAL (PLR, Jiang et al. [2021]).** Prioritizes tasks with the largest ℓ_1 value loss under GAE: $z_i = \frac{1}{T} \sum_{t=0}^{T} \left| \sum_{k=t}^{T} (\gamma\lambda)^{k-t} \delta_k \right|$, where $\delta_k = r_k + \gamma \widehat{V}(s_{k+1}) - \widehat{V}(s_k)$ is the TD-error at step k.
4. **HARD FIRST (SMT, Cho et al. [2024]).** Samples uniformly from the K unsolved tasks with the smallest current returns. A task is "solved" if its return exceeds threshold M_i and is then replaced by the unsolved task with the smallest return. A task is "unsolvable" if its return fails to exceed threshold m_i after a fixed number of steps and is then removed from the active set. After B_1 steps, the agent samples uniformly from all "unsolvable" tasks.
5. **EASY FIRST (*e.g.*, Joshi et al. [2025])**. Concentrates sampling on the easiest task and switches to the next hardest upon reaching a success rate of 0.9. We only use this baseline in Gridworld; it requires us to rank tasks by difficulty, and it is unclear how to rank all tasks in other benchmarks.

[4] Most of our experiments use PPO [Schulman et al., 2017] since it is more data efficient than REINFORCE.



(a) MuJoCo6 (20 seeds) (b) MT10 (20 seeds) (c) MT50 (10 seeds)

Figure 1: Mean normalized return aggregated over tasks in each benchmark. Shaded regions denote 95% bootstrap confidence intervals.



(a) MuJoCo6 (20 seeds) (b) MT10 (20 seeds) (c) MT50 (10 seeds)

Figure 2: Final normalized return in each task, sorted from highest to lowest. Error bars denote 95% bootstrap confidence intervals.

To ensure learning differences reflect differences in prioritization rather than implementation differences, we use the same mirror ascent update rule as DRATS (Eq. 7) to update the task distribution in LEARNING PROGRESS and LEARNING POTENTIAL, using z_i in place of g_i. To prevent catastrophic forgetting, we enforce a minimum sampling probability in *all* baselines: $q_{\min} = 0.02$ in Gridworld and $q_{\min} = 1/(4k)$ in all other experiments. HARD FIRST (SMT) and LEARNING POTENTIAL (PLR) were originally proposed for off-policy RL, while we use on-policy RL. We describe minor adaptations of these methods to the on-policy setting in Appendix D. To estimate reference returns for DRATS in MT10 and MT50, we initialize $J_i^{\text{ref}} = r_{\max} H_{\max} = 5000$ and then update J_i^{ref} to the maximum observed return once the agent's success rate exceeds 0.5. In MuJoCo6 where tasks have no notion of success, we set J_i^{ref} to the maximum observed return. In Gridworld, we simply use $J_i^{\text{ref}} = 1$ for all tasks. We train agents using REINFORCE [Williams, 1992] in Gridworld and PPO [Schulman et al., 2017] in all other benchmarks. We provide hyperparameter tuning details for all baselines in Appendix G. We use the same PPO/REINFORCE hyperparameters for all baselines within each benchmark (also listed in Appendix G). We build on top of the `metaworld-algorithms` codebase [McLean et al., 2025a] for MT10/MT50 and CleanRL [Huang et al., 2022b] for MuJoCo6.

Per-Task Advantage Normalization. Standard advantage normalization computes mean and standard deviation over all data, aggregating across all tasks, which biases learning toward tasks with higher average return: if task 1 has a much larger mean return than task 2, global normalization makes task 2's advantages appear negative, which can make a policy gradient update decrease the probability of actions that are actually beneficial for task 2. We instead normalize advantages for each task separately. In Fig. 14 of Appendix E, we show that per-task normalization improves data efficiency compared to global normalization. We apply per-task normalization to *all* methods. We do not treat it as a contribution of DRATS.

Metrics. Since maximum returns differ across tasks, we normalize task return as $\tilde{J}_i = (J_i - J_i^{\min})/(J_i^{\max} - J_i^{\min}) \in [0, 1]$. [5] For **H1**, we report $\tilde{J}_i$ averaged across tasks; when methods reach

[5] We prefer normalized returns over success rates since return can still improve after achieving a perfect success rate by solving tasks in fewer steps. We plot success rates for MT10/MT50 in Fig. 13 in Appendix E.

similar final returns, we measure data efficiency as interactions required to converge. For **H2**, we report the distribution of final per-task $\tilde{J}_i$.

5.1 MuJoCo6, MT10, and MT50 Experiments

Fig. 1 shows that DRATS achieves the highest aggregate return in MuJoCo6, MT10, and MT50, supporting **H1**. Fig. 2 shows that this improvement is driven by higher worst-task returns rather than higher returns on easier tasks, supporting **H2**. We now discuss results in more detail. Due to space constraints, we show returns and sampling probabilities for individual tasks in Appendix E.

MuJoCo6. Fig. 3 shows results for the two hardest tasks, Ant and Humanoid; all methods perform similarly on the remaining four tasks (Fig. 7 in Appendix E), so performance differences are driven by these two. DRATS is the only method to solve both. DRATS solves Ant by allocating 40–70% of sampling probability to it for the first 50M timesteps. All baselines allocate approximately 30% for most of training and fail to make any progress on it. HARD FIRST and LEARNING POTENTIAL solve Humanoid by allocating 25–35% probability to it, while UNIFORM and LEARNING PROGRESS allocate at most 17% and thus learn more slowly. LEARNING PROGRESS learns more slowly than UNIFORM because the small slope of its learning curve causes it to *deprioritize* Humanoid even though the task is far from being solved.

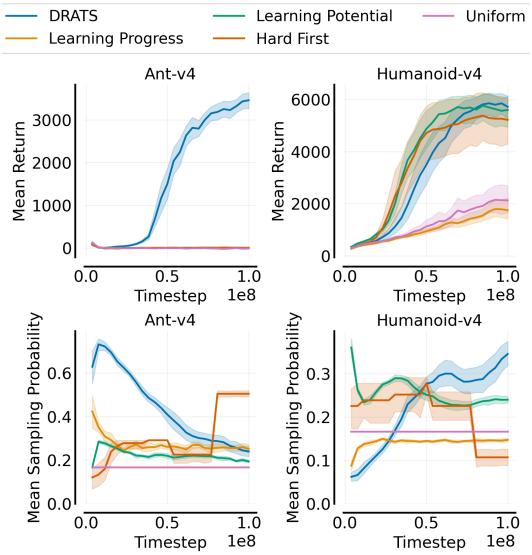
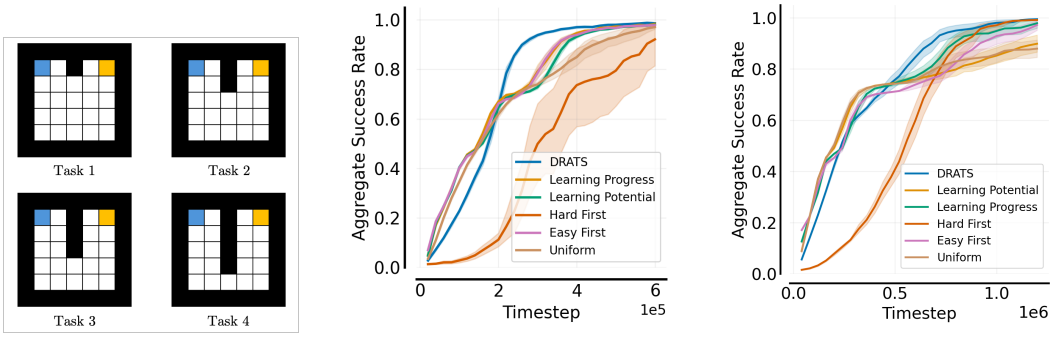


Figure 3: MuJoCo6 task returns and probabilities (20 seeds). Shaded regions denote 95% bootstrap confidence intervals.

MT10. As shown in Fig. 8 of Appendix E, performance differences are mostly driven by `pick-place`, `push`, and `peg-insert-side` tasks. DRATS shifts probability to them at the start of training and ultimately exceeds UNIFORM's final return on them after just 6M timesteps. LEARNING PROGRESS and LEARNING POTENTIAL heavily prioritize easy tasks initially (*e.g.*, window-open, window-close) and do not prioritize the harder tasks (`pick-place`, `push`) until much later in training. HARD FIRST performs poorly on `pick-place`, `push`, and `peg-insert-side`: once a task is declared "solved", it abruptly shifts probability away from it, causing performance to collapse and forcing the agent to relearn the task—visible as repeated peaks and dips in their learning curves (Fig. 8).

MT50. As shown in Fig. 9 and Fig. 10 of Appendix E, DRATS achieves significantly larger returns than all baselines on hard tasks like `disassemble` and `pick-place-wall` because it places 2-3 times more probability on them. LEARNING PROGRESS and LEARNING POTENTIAL perform no better than UNIFORM: both prioritize tasks that are already making rapid progress or have high value estimation error, which tends to coincide with the easier tasks that DRATS deprioritizes. LEARNING PROGRESS also achieves slightly lower final returns on many easy tasks because prioritization weakens near convergence as learning curves flatten. For instance, it reaches a return of approximately 3500 in `button-press-topdown` while all other baselines reach 3700. We observe similar behavior in tasks like `coffee-button`, `door-unlock`, and `hand-insert` as well (Fig. 9 of Appendix E). Similar to MT10, HARD FIRST exhibits performance collapse on many hard tasks (*e.g.*, `bin-picking`, `basketball`, `pick-place`) (Fig. 9 of Appendix E).

DRATS with Multi-Task Architectures. Since DRATS only modifies data collection, it is orthogonal to multi-task network architectures such as Soft Modularization [Yang et al., 2020] and MOORE [Hendawy et al., 2023]. We hypothesize that applying DRATS on top of these architectures improves over using them with uniform task sampling. Fig. 4 verifies this hypothesis, showing that DRATS achieves a larger aggregate return than UNIFORM when using MOORE or Soft Modulariza-



(a) Gridworld tasks (b) Gridworld, shared networks (c) Gridworld, separate networks

Figure 5: **(a)** A 4-task Gridworld. The agent must navigate from the blue cell to the gold cell, receiving a reward of $+1$ for reaching the goal and reward -0.001 otherwise. We truncate episodes after 15 steps. The length of the shortest path to the goal increases from Task 1 to Task 4, so Task 4 is most difficult. **(b)** Mean success rate over all tasks with shared actor/critic networks that enable positive transfer (50 seeds). **(c)** Mean success rate over all tasks with separate actor/critic networks for each task to eliminate transfer (50 seeds). Shaded regions denote 95% bootstrap confidence intervals.

tion networks. We include an analogous experiment with MOORE on MT50 in Fig. 12 of Appendix E with qualitatively similar results.

5.2 Gridworld Experiments

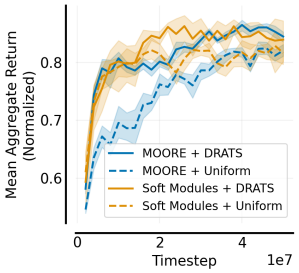


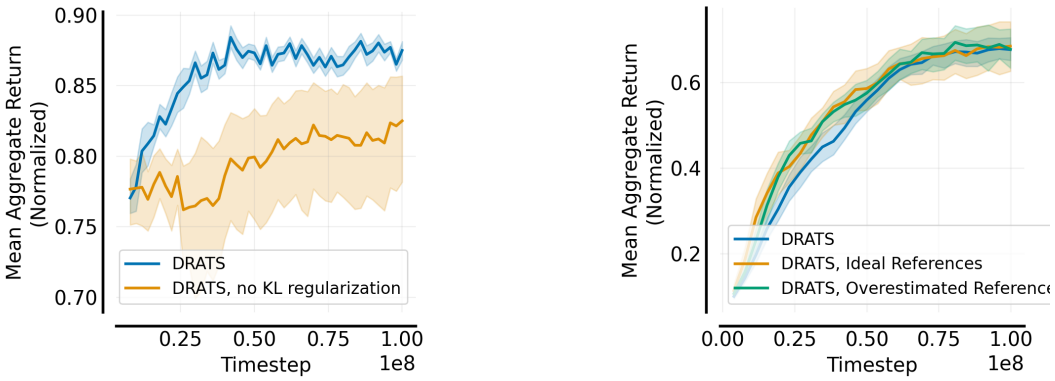
Figure 4: DRATS + MOORE and Soft Modularization in MT10 (10 seeds).

Curriculum learning (CL) methods prioritize easy tasks assuming that progress on easy tasks will accelerate learning on hard ones. We now use a 4-task Gridworld (Fig. 5a) to illustrate how DRATS is more data efficient than CL methods when tasks exhibit positive transfer—and also when they do not. Fig. 5c shows results when training separate networks per task, which eliminates transfer entirely. Fig. 5b shows results using a shared network for all tasks, which enables positive transfer.[6] DRATS is the most data-efficient in both settings. In the shared-network setting, DRATS prioritizes the hardest tasks while maintaining sufficient probability on easier tasks to benefit from transfer, solving all tasks simultaneously rather than sequentially (Fig. 11a and Fig. 11b of Appendix E). EASY FIRST, LEARNING PROGRESS, and LEARNING POTENTIAL do not prioritize Task 4 until Tasks 1–3 are mostly solved and thus converge more slowly. HARD FIRST converges slowly for the opposite reason: by neglecting easy tasks, it does not benefit as much from positive transfer that would help learn the hardest task. DRATS outperforms all baselines even without transfer, showing that its advantage is not contingent on shared task structure. These results also support **H1**.

5.3 Ablations

KL Regularization. We compare DRATS with and without KL regularization in MT10. Removing KL regularization recovers the unregularized objective of Eq. 3, whose solution $\boldsymbol{q}^*$ places all probability on the task with the largest gap. Fig. 6a shows aggregate normalized return; per-task returns and sampling probabilities are in Appendix F (Figs. 15 and 16). Without regularization, DRATS achieves lower return on most tasks because its sampling distribution concentrates on `pick-place` and allocates less data to `drawer-open`, `door-open`, `push`, and `peg-insert-side`. The undersampled tasks are solved more slowly and therefore provide less positive transfer to other tasks, which likely explains why DRATS without regularization performs worse even on `pick-place`. With KL regularization, $\boldsymbol{q}^*$ assigns each task probability that grows with its return gap, so all tasks continue to receive data, improve, and provide positive transfer to one another.

[6] All methods solve all tasks in fewer steps with shared networks, confirming that transfer is positive.



(a) KL regularization ablation (MT10).

(b) Reference return ablation (MuJoCo6).

Figure 6: Ablating KL regularization and reference return estimation. Shaded regions denote 95% bootstrap confidence intervals.

Reference Return Estimation. We compare three estimation strategies on MuJoCo6: (1) **Underestimated:** initialize $J_i^{\text{ref}} = 0$ and update it to the maximum observed return, as in our main experiments; (2) **Ideal:** fix J_i^{ref} to the final maximum return from our main experiments; and (3) **Overestimated:** fix J_i^{ref} to the ideal reference for Hopper, Walker2d, and Humanoid, and to $1.2\times$ the ideal reference for the remaining tasks. All three yield similar aggregate return (Fig. 6b), suggesting that neither underestimating nor overestimating references is a major concern in practice. An underestimated reference under-prioritizes a task only until the agent observes a larger return (Section 6). An overestimated reference over-prioritizes a task because its gap never reaches zero, but KL regularization limits how much additional data DRATS allocates to it. For example, in MT50, DRATS plateaus at a return of ≈ 300 on `disassemble`, so its reference remains at $r_{\max} H_{\max} = 5000$ (Fig. 9), yet its sampling probability peaks at only ≈ 0.05, just $2.5\times$ more than uniform (Fig. 10).

Additional Ablations. In Appendix F, we include additional ablations showing that (1) DRATS is relatively robust to the temperature η and mirror ascent step size α, (2) per-task advantage normalization is more data efficient than global normalization, and (3) adaptive task sampling is more data efficient than sampling tasks uniformly and reweighting each task's return by q_i.

6 Limitations

DRATS has two main limitations. First, DRATS requires reference returns, which we estimate online rather than requiring the user to specify hardcoded thresholds—a limitation of common curriculum learning methods [Cho et al., 2024, Kanitscheider et al., 2021, Wang et al., 2020]. Online estimation may underestimate reference returns early in training, causing DRATS to under-allocate data to tasks that can still improve. Second, DRATS assumes a discrete task set, but tasks may also be continuous. In principle, one could model the task distribution as a neural network $\boldsymbol{q}_\phi$ with weights ϕ that outputs the parameters of a distribution over tasks (*e.g.*, a Gaussian) and then sample from $\boldsymbol{q}_\phi$ via the reparameterization trick [Kingma and Welling, 2013]. We leave this extension to future work.

7 Conclusion

We study *data imbalance* in multi-task reinforcement learning (MTRL). Standard MTRL allocates an equal interaction budget to all tasks, which over-allocates data to easy tasks and under-allocates data to hard ones, causing agents to learn easy tasks quickly and hard tasks slowly. To enable balanced RL across tasks, we introduce Distributionally Robust Adaptive Task Sampling (DRATS), an algorithm that adaptively prioritizes sampling tasks furthest from being solved. We derive DRATS by formalizing MTRL as a feasibility problem from which we derive a minimax objective for minimizing the worst-case return gap across tasks. In benchmarks like MetaWorld-MT10 and MT50, DRATS improves data efficiency and increases worst-task performance compared to existing task sampling algorithms.

Acknowledgments

The authors would like to thank Julian Katz-Samuels and Mohammad Ghavamzadeh for their helpful feedback during the early stages of this work. This work took place in the Prediction and Action Lab (PAL) at the University of Wisconsin–Madison. PAL research is supported by NSF (IIS-2410981, IIS-2551198) and the Wisconsin Alumni Research Foundation.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023.

Jing An, Lexing Ying, and Yuhua Zhu. Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients. *arXiv preprint arXiv:2009.13447*, 2020.

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.

Adrien Baranes and Pierre-Yves Oudeyer. R-iac: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169, 2009.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

Emma Brunskill and Lihong Li. Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*, 2013.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International conference on machine learning*, pages 1430–1440. PMLR, 2021.

Daesol Cho, Seungjae Lee, and H Jin Kim. Outcome-directed reinforcement learning by uncertainty & temporal distance-aware curriculum goal generation. *arXiv preprint arXiv:2301.11741*, 2023.

Myungsik Cho, Jongeui Park, Suyoung Lee, and Youngchul Sung. Hard tasks first: Multi-task reinforcement learning through task scheduling. In *Forty-first International Conference on Machine Learning*, 2024.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

Nicholas E Corrado, Julian Katz-Samuels, Adithya M Devraj, Hyokun Yun, Chao Zhang, Yi Xu, Yi Pan, Bing Yin, and Trishul Chilimbi. Automixalign: Adaptive data mixing for multi-task preference optimization in llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20234–20258, 2025.

Mingxuan Cui, Duo Zhou, Yuxuan Han, Grani A Hanasusanto, Qiong Wang, Huan Zhang, and Zhengyuan Zhou. Dr-sac: Distributionally robust soft actor-critic for reinforcement learning under uncertainty. In *International Conference on Learning Representations*, volume 2026, pages 92720–92750, 2026.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus. Mathématique*, 350(5-6):313–318, 2012.

John Duchi. Lecture notes on statistics and information theory. *URL: https://https://web. stanford. edu/class/stats311/lecture-notes. pdf*, 2023.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. *arXiv preprint arXiv:2011.08909*, 2020.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

Yifei He, Shiji Zhou, Guojun Zhang, Hyokun Yun, Yi Xu, Belinda Zeng, Trishul Chilimbi, and Han Zhao. Robust multi-task learning with excess risks. *arXiv preprint arXiv:2402.02009*, 2024.

Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. *arXiv preprint arXiv:2311.11385*, 2023.

Peide Huang, Mengdi Xu, Jiacheng Zhu, Laixi Shi, Fei Fang, and Ding Zhao. Curriculum reinforcement learning using optimal transport via gradual domain adaptation. *Advances in neural information processing systems*, 35:10656–10670, 2022a.

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

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

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.

Ingmar Kanitscheider, Joost Huizinga, David Farhi, William Hebgen Guss, Brandon Houghton, Raul Sampedro, Peter Zhokhov, Bowen Baker, Adrien Ecoffet, Jie Tang, et al. Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft. *arXiv preprint arXiv:2106.14876*, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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, 2021a.

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

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive rl without rewards, demonstrations, or subgoals. *arXiv preprint arXiv:2408.05804*, 2024.

Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. iclr, 2021b.

Debmalya Mandal, Paulius Sasnauskas, and Goran Radanovic. Distributionally robust reinforcement learning with human feedback. *arXiv preprint arXiv:2503.00539*, 2025.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, KR Zentner, Ryan Julian, Jordan K Terry, Isaac Woungang, et al. Meta-world+: An improved, standardized, rl benchmark. *arXiv preprint arXiv:2505.11289*, 2025a.

Reginald McLean, Evangelos Chatzaroulas, Jordan Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Multi-task reinforcement learning enables parameter scaling. *arXiv preprint arXiv:2503.05126*, 2025b.

Paul Michel, Sebastian Ruder, and Dani Yogatama. Balancing average and worst-case accuracy in multitask learning. *arXiv preprint arXiv:2110.05838*, 2021.

Nathan Monette, Alistair Letcher, Michael Beukman, Matthew Thomas Jackson, Alexander Rutherford, Alexander David Goldie, and Jakob Nicolaus Foerster. An optimisation framework for unsupervised environment design. In *Reinforcement Learning Conference*, 2025. URL https://openreview.net/forum?id=WnknYUybWX.

Siddharth Mysore, Robert Platt, and Kate Saenko. Reward-guided curriculum for robust reinforcement learning. In *Workshop on Multi-task and Lifelong Reinforcement Learning at ICML*, 2019.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2022.

Shay Oren et al. Distributionally robust language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Kishan Panaganti, Zhenwen Liang, Wenhao Yu, Haitao Mi, and Dong Yu. Group distributionally robust optimization-driven reinforcement learning for llm reasoning. *arXiv preprint arXiv:2601.19280*, 2026.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1607–1612, 2010.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Meisam Razaviyayn, Tianjian Huang, Songtao Lu, Maher Nouiehed, Maziar Sanjabi, and Mingyi Hong. Nonconvex min-max optimization: Applications, challenges, and recent theoretical advances. *IEEE Signal Processing Magazine*, 37:55–66, 2020. URL https://api.semanticscholar.org/CorpusID:219687311.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32, 2019.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

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

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. *arXiv preprint arXiv:2104.05859*, 2021.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International conference on machine learning*, pages 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.

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*, 2012.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International conference on machine learning*, pages 9940–9951. PMLR, 2020.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36, 2024.

Mengdi Xu, Peide Huang, Yaru Niu, Visak Kumar, Jielin Qiu, Chao Fang, Kuan-Hui Lee, Xuewei Qi, Henry Lam, Bo Li, et al. Group distributionally robust reinforcement learning with hierarchical latent variables. In *International Conference on Artificial Intelligence and Statistics*, pages 2677–2703. PMLR, 2023.

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*, pages 1094–1100. PMLR, 2020b.

Tianjun Zhang, Benjamin Eysenbach, Ruslan Salakhutdinov, Sergey Levine, and Joseph E Gonzalez. C-planning: An automatic curriculum for learning goal-reaching tasks. *arXiv preprint arXiv:2110.12080*, 2021.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

Table of Contents

A Derivation of the KL-Regularized Task Distribution **16**

B Mirror Ascent Update **17**

C Convergence Analysis **18**

D Baselines **20**

E Additional Experimental Results **20**

F Additional Ablations **25**

F.1 Adaptive Sampling vs. Objective Reweighting 25

F.2 DRATS Hyperparameter Ablations 27

G Hyperparameters **28**

H Computational Resources **32**

A Derivation of the KL-Regularized Task Distribution

In this section, we derive a closed-form solution to the inner maximization of Eq. 4 for a fixed $\boldsymbol{\theta}$:

$$\max_{\boldsymbol{q}\in\mathcal{Q}} \mathbb{E}_{i\sim\boldsymbol{q}}[g_i(\boldsymbol{\theta})] \qquad \text{where} \qquad \mathcal{Q} = \{\boldsymbol{q} \in \Delta^k \mid \textsc{kl}(\boldsymbol{q}||\boldsymbol{p}_0) \leq \epsilon\}. \tag{10}$$

Our derivation follows that of Peters et al. [2010], Abdolmaleki et al. [2018], Peng et al. [2019]. First, we show that strong duality holds so that we can justifiably optimize the unconstrained Lagrangian dual. Next, we show that the solution to the inner maximization is a weighted softmax distribution.

Showing strong duality. If $\epsilon = 0$, the feasible set reduces to $\boldsymbol{q} = \boldsymbol{p}_0$ and the solution is trivially $\boldsymbol{p}_0$. Henceforth, we assume $\varepsilon > 0$. We begin by rewriting Eq. 10 in primal form:

$$\max_{\boldsymbol{q}\in\mathbb{R}^k} \quad \sum_{i=1}^{k} q_i g_i(\boldsymbol{\theta}) \tag{11}$$

$$\text{s.t.} \quad \sum_{i=1}^{k} q_i \log \frac{q_i}{p_{0,i}} \leq \epsilon \tag{12}$$

$$\sum_{i=1}^{k} q_i = 1. \tag{13}$$

The objective is linear (and hence continuous) in $\boldsymbol{q}$, and the feasible set $\mathcal{Q}$ is convex and compact. Therefore, the primal problem admits at least one optimal solution. Moreover, since $\boldsymbol{q} = \boldsymbol{p}_0$ is strictly feasible (*i.e.*, $\textsc{kl}(\boldsymbol{p}_0||\boldsymbol{p}_0) = 0 < \epsilon$), Slater's condition holds. Hence, strong duality applies and the dual optimum is attained. As a result, it suffices to solve the Lagrange dual and recover the maximizer of the Lagrangian at the optimal dual variables.

Closed-form solution to the inner maximization. We construct the Lagrangian by introducing a multiplier $\mu \geq 0$ for the KL inequality constraint and a multiplier $\lambda \in \mathbb{R}$ for the equality constraint:

$$\mathcal{L}(\boldsymbol{q}, \mu, \lambda) = \sum_{i=1}^{k} q_i g_i(\boldsymbol{\theta}) - \mu \left(\sum_{i=1}^{k} q_i \log \frac{q_i}{p_{0,i}} - \epsilon \right) + \lambda \left(\sum_{i=1}^{k} q_i - 1 \right). \tag{14}$$

For fixed $\mu > 0$ and λ, the Lagrangian is strictly concave in $\boldsymbol{q}$ over the simplex due to the negative entropy term. Hence, the maximizer lies in the interior of the simplex and is characterized by first-order optimality conditions. Taking the derivative with respect to each q_i and setting it to zero yields

$$0 = \frac{\partial \mathcal{L}}{\partial q_i} = g_i(\boldsymbol{\theta}) - \mu \left(\log \frac{q_i}{p_{0,i}} + 1 \right) + \lambda. \tag{15}$$

Solving Eq. 15 for q_i directly gives

$$q_i = p_{0,i} \exp\left(\frac{g_i(\boldsymbol{\theta}) + \lambda - \mu}{\mu} \right). \tag{16}$$

Since the term $\exp\left(\frac{\lambda-\mu}{\mu}\right)$ is a constant, it can be absorbed into a normalization constant $C > 0$:

$$q_i = C p_{0,i} \exp\left(\frac{g_i(\boldsymbol{\theta})}{\mu} \right). \tag{17}$$

Define the inverse temperature parameter $\eta = 1/\mu \geq 0$. By enforcing the normalization constraint $\sum_i q_i = 1$, we eliminate the multiplier λ and determine C, giving the closed-form solution

$$q_i^* = \frac{p_{0,i} \exp\left(\eta g_i(\boldsymbol{\theta})\right)}{\sum_{j=1}^{k} p_{0,j} \exp\left(\eta g_j(\boldsymbol{\theta})\right)}. \tag{18}$$

For $\epsilon > 0$, the optimal η is the unique positive value such that $\textsc{kl}(\boldsymbol{q}^*||\boldsymbol{p}_0) = \epsilon$, unless $g_i(\boldsymbol{\theta})$ is constant across all i, in which case $\boldsymbol{q}^* = \boldsymbol{p}_0$. By strong duality (which is guaranteed by Slater's condition), letting (μ^*, λ^*) be dual-optimal implies that $\boldsymbol{q}^*$ in Eq. 18 is also a primal optimal solution to Eq. 10. For a uniform base distribution $\boldsymbol{p}_0$, the solution reduces to:

$$q_i^* = \frac{\exp\left(\eta g_i(\boldsymbol{\theta})\right)}{\sum_{j=1}^{k} \exp\left(\eta g_j(\boldsymbol{\theta})\right)}, \qquad \text{or equivalently,} \qquad \boldsymbol{q}^* = \text{softmax}(\eta \boldsymbol{g}(\boldsymbol{\theta})) \tag{19}$$

Algorithm 2 DRATS Mirror Ascent Update

Input: Step size $\alpha \in (0, \eta]$, inverse temperature $\eta > 0$.
for each round $t = 1, \ldots, T$ **do**
  Compute the gradient:

$$h_{t,i} = g_i(\boldsymbol{\theta}_t) - \frac{1}{\eta}\left(\log(kq_{t,i}) + 1\right). \tag{21}$$

  Perform the update:

$$q_{t+1,i} = \frac{q_{t,i}\exp\left(\alpha h_{t,i}\right)}{\sum_{j=1}^{k} q_{t,j}\exp\left(\alpha h_{t,j}\right)}. \tag{22}$$

end for

B Mirror Ascent Update

In this section, we formally describe the DRATS mirror ascent update to the task sampling distribution $\boldsymbol{q}$. We additionally show this update can be written as a weighted geometric mean of $\boldsymbol{q}_t$ and the best-response $\boldsymbol{q}^*$ (Eq. 19), a form we use in Appendix C to lower bound the task sampling probabilities $q_{t,i}$. Following Duchi [2023], the general mirror ascent update at round t is

$$\boldsymbol{q}_{t+1} = \arg\max_{\boldsymbol{q}\in\Delta^k}\left\{\langle h_t, \boldsymbol{q}\rangle - \frac{1}{\alpha}D_\psi(\boldsymbol{q}, \boldsymbol{q}_t)\right\}, \tag{20}$$

where $h_t = \nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}_t)$ is the gradient of the ascent objective $\mathcal{L}(\boldsymbol{q}) = \langle \boldsymbol{q}, g(\boldsymbol{\theta}_t)\rangle - \frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0)$, ψ is a mirror map, and D_ψ is Bregman divergence induced by ψ. We use the negative entropy mirror map $\psi(\boldsymbol{q}) = \sum_i q_i \log q_i$, which induces $D_\psi(\boldsymbol{q}, \boldsymbol{q}_t) = \mathrm{KL}(\boldsymbol{q}\|\boldsymbol{q}_t)$. Algorithm 2 describes the update we implement.

We now show that equation 22 can be written as a weighted geometric mean of $\boldsymbol{q}_t$ and $\boldsymbol{q}^*$. Substituting equation 21 into equation 22 and using $p_{0,i} = 1/k$,

$$\begin{aligned}
q_{t+1,i} &\propto q_{t,i}\exp\left(\alpha g_i(\boldsymbol{\theta}) - \frac{\alpha}{\eta}\log(q_{t,i}) - \frac{\alpha}{\eta}\log(k) - \frac{\alpha}{\eta}\right) && (23)\\
&\propto q_{t,i}\exp\left(\alpha g_i(\boldsymbol{\theta}) - \frac{\alpha}{\eta}\log(q_{t,i})\right) && (24)\\
&= q_{t,i}^{1-\alpha/\eta}\exp\left(\alpha g_i(\boldsymbol{\theta})\right) && (25)\\
&= q_{t,i}^{1-\alpha/\eta}\exp\left(\alpha\frac{\eta}{\eta}g_i(\boldsymbol{\theta})\right) && (26)\\
&= q_{t,i}^{1-\alpha/\eta}\exp\left(\eta g_i(\boldsymbol{\theta})\right)^{\frac{\alpha}{\eta}} && (27)\\
&\propto q_{t,i}^{1-\alpha/\eta}(q_i^*)^{\alpha/\eta}, && (28)
\end{aligned}$$

where in Eq. 24 we dropped $\exp(\alpha/\eta)$ and $\exp(-\frac{\alpha}{\eta}\log k)$ since both are independent of i and thus absorbed into normalization, and in Eq. 28 we used $q_i^* \propto \exp(\eta g_i(\boldsymbol{\theta}))$ (Eq. 19). The update is therefore a weighted geometric mean of $q_{t,i}$ and q_i^* when $\eta \in (0, \eta]$, which implies $q_{t+1,i} \geq \min(q_{t,i}, q_i^*)$.

C Convergence Analysis

In this appendix, we prove Theorem 4.2. Our proof bounds the regret of each player in the two-player game separately and then combines them. For the $\boldsymbol{q}$-player, we apply mirror *descent* to the convex loss $\ell_t(\boldsymbol{q}) = -\langle \boldsymbol{q}, g(\boldsymbol{\theta}_t)\rangle + \frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0)$, which is equivalent to the mirror ascent update derived in Appendix B since $\ell_t = -\mathcal{L}$. We begin by stating a standard mirror descent regret bound specialized to our setting. Then, we restate and prove our convergence theorem.

Lemma C.1 (Regret of Mirror Descent)**.** *(Special case of Theorem 18.2.5 in Duchi [2023].) Let $\ell_1, \dots, \ell_T$ be an arbitrary sequence of convex functions with subgradients $h_t \in \partial\ell_t(\boldsymbol{q}_t)$, and let $\boldsymbol{q}_1, \dots, \boldsymbol{q}_T$ be generated by mirror descent with mirror map $\psi(\boldsymbol{q}) = \sum_i q_i \log q_i$, which is 1-strongly convex with respect to $\|\cdot\|_1$ and induces Bregman divergence $D_\psi(\boldsymbol{u}, \boldsymbol{v}) = \mathrm{KL}(\boldsymbol{u}\|\boldsymbol{v}) = \sum_i u_i \log \frac{u_i}{v_i}$. Then for any $\boldsymbol{u} \in \Delta^k$,*

$$\sum_{t=1}^{T} \big(\ell_t(\boldsymbol{q}_t) - \ell_t(\boldsymbol{u})\big) \le \frac{D_\psi(\boldsymbol{u}, \boldsymbol{q}_1)}{\alpha} + \frac{\alpha}{2}\sum_{t=1}^{T} \|h_t\|_\infty^2, \tag{29}$$

where $\|\cdot\|_\infty$ is the dual norm of $\|\cdot\|_1$.

Theorem C.2 (Convergence of DRATS)**.** *Suppose $g_i(\boldsymbol{\theta}) \in [0, M]$ for all $i \in [k]$ and $\boldsymbol{\theta} \in \Theta$, and that the $\boldsymbol{\theta}$-player is C-low-regret (Definition 4.1). For any target accuracy $\epsilon > 0$, set $\eta = \frac{2\log k}{\epsilon}$ and $\alpha = \frac{\sqrt{2\log k}}{G\sqrt{T}}$ where $G \coloneqq M + \frac{1+\max(\eta M, \log k)}{\eta}$. Then for any $T \ge \frac{4(G\sqrt{2\log k}+C)^2}{\epsilon^2}$, DRATS satisfies:*

$$\max_{i\in[k]} \frac{1}{T}\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \le \min_{\boldsymbol{\theta}\in\Theta} \max_{i\in[k]} g_i(\boldsymbol{\theta}) + \epsilon. \tag{30}$$

Proof. **Step 1: the $\boldsymbol{q}$-player.** At round t, the $\boldsymbol{q}$-player minimizes the KL-regularized loss

$$\ell_t(\boldsymbol{q}) = -\langle \boldsymbol{q}, g(\boldsymbol{\theta}_t)\rangle + \frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0). \tag{31}$$

via mirror descent with the negative entropy mirror map $\psi(\boldsymbol{q}) = \sum_i q_i \log q_i$ with Bregman divergence $D_\psi(\boldsymbol{u}, \boldsymbol{v}) = \mathrm{KL}(\boldsymbol{u}||\boldsymbol{v})$, step size $\alpha \in (0, \eta]$, and uniform initialization $\boldsymbol{q}_1 = \boldsymbol{p}_0$. Since ℓ_t is the sum of the linear term $\langle \boldsymbol{q}, g(\boldsymbol{\theta}_t)\rangle$ and $\frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}\|\boldsymbol{p}_0)$, which is 1-strongly convex with respect to $\|\cdot\|_1$ [Duchi, 2023], ℓ_t is convex. Its gradient at interior points of the simplex is

$$h_{t,i} = \frac{\partial \ell_t}{\partial q_i}(\boldsymbol{q}_t) = -g_i(\boldsymbol{\theta}_t) + \frac{1}{\eta}\big(\log\frac{q_{t,i}}{p_{0,i}} + 1\big) = -g_i(\boldsymbol{\theta}_t) + \frac{1}{\eta}\big(\log(kq_{t,i}) + 1\big), \tag{32}$$

where we substituted $p_{0,i} = 1/k$. We now upper bound $\|h_t\|_\infty$ by deriving a lower bound on $\log q_{t,i}$.[7] Since $\alpha \in (0, \eta]$, the mirror descent update takes the form $q_{t+1,i} \propto q_{t,i}^{1-\alpha/\eta}(q_i^*)^{\alpha/\eta}$ (Eq. 28), a weighted geometric mean of $q_{t,i}$ and q_i^*. Therefore, $q_{t+1,i} \ge \min(q_{t,i}, q_i^*) \ge \frac{e^{-\eta M}}{k}$ for all t, i, where the last inequality uses $g_i(\boldsymbol{\theta}) \in [0, M]$. It follows that $\log(kq_{t,i}) \in [-\eta M, \log k]$, so

$$|h_{t,i}| \le M + \frac{1 + \max(\eta M, \log k)}{\eta} =: G \qquad \Longrightarrow \qquad \|h_t\|_\infty \le G \quad \forall\, t \tag{33}$$

i.e., ℓ_t is G-Lipschitz with respect to $\|\cdot\|_1$. Thus, we can apply Lemma C.1: For any $\boldsymbol{u} \in \Delta^k$:

$$\begin{aligned}\sum_{t=1}^{T} \big(\ell_t(\boldsymbol{q}_t) - \ell_t(\boldsymbol{u})\big) &\le \frac{\mathrm{KL}(\boldsymbol{u}\|\boldsymbol{q}_1)}{\alpha} + \frac{\alpha}{2}\sum_{t=1}^{T}\|h_t\|_\infty^2 \\ &\le \frac{\log k}{\alpha} + \frac{\alpha G^2 T}{2},\end{aligned} \tag{34}$$

where we substituted $\mathrm{KL}(\boldsymbol{u}\|\boldsymbol{q}_1) \le \log k$ (since $\boldsymbol{q}_1$ is uniform) and $\|h_t\|_\infty^2 \le G^2$. Setting $\boldsymbol{u} = e_i$ and expanding $\ell_t(e_i) = -g_i(\boldsymbol{\theta}_t) + \frac{\log k}{\eta}$, rearranging equation 34 gives

$$\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \le \sum_{t=1}^{T} -\ell_t(\boldsymbol{q}_t) + \frac{T\log k}{\eta} + \frac{\log k}{\alpha} + \frac{\alpha G^2 T}{2}. \tag{35}$$

[7] Algorithm 1 enforces a minimum sampling probability $q_{t,i} \ge q_{\min}$. However, this step is not required for convergence, so we instead derive the bound directly from the structure of the mirror descent update.

Substituting $-\ell_t(\boldsymbol{q}_t) \leq \langle \boldsymbol{q}_t, g(\boldsymbol{\theta}_t)\rangle$ and dividing by T:

$$\frac{1}{T}\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \leq \frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, g(\boldsymbol{\theta}_t)\rangle + \frac{\log k}{\eta} + \frac{\log k}{\alpha T} + \frac{\alpha G^2}{2}. \tag{36}$$

Since this bound holds for all $i \in [k]$ and the right-hand side is independent of i, the inequality is preserved when taking $\max_{i\in[k]}$ on the left:

$$\max_{i\in[k]} \frac{1}{T}\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \leq \frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, g(\boldsymbol{\theta}_t)\rangle + \frac{\log k}{\eta} + \frac{\log k}{\alpha T} + \frac{\alpha G^2}{2}. \tag{37}$$

Step 2: the $\boldsymbol{\theta}$-player. Since $\frac{1}{\eta}\mathrm{KL}(\boldsymbol{q}_t\|\boldsymbol{p}_0)$ does not depend on $\boldsymbol{\theta}$, the $\boldsymbol{\theta}$-player's effective loss at round t is $\langle \boldsymbol{q}_t, g(\boldsymbol{\theta})\rangle$. Thus, we can apply our C-low-regret assumption (Definition 4.1). After dividing by T, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, g(\boldsymbol{\theta}_t)\rangle \leq \min_{\boldsymbol{\theta}\in\Theta} \frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, g(\boldsymbol{\theta})\rangle + \frac{C}{\sqrt{T}}. \tag{38}$$

Since $\boldsymbol{q}_t \in \Delta^k$, $\langle \boldsymbol{q}_t, g(\boldsymbol{\theta})\rangle$ is a convex combination of $\{g_i(\boldsymbol{\theta})\}_{i=1}^k$, we have $\langle \boldsymbol{q}_t, g(\boldsymbol{\theta})\rangle \leq \max_{i\in[k]} g_i(\boldsymbol{\theta})$. Averaging over $t = 1, \ldots, T$ and taking $\min_{\boldsymbol{\theta}\in\Theta}$ on both sides yields:

$$\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, g(\boldsymbol{\theta})\rangle \leq \min_{\boldsymbol{\theta}\in\Theta}\max_{i\in[k]} g_i(\boldsymbol{\theta}). \tag{39}$$

Substituting equation 39 into equation 38:

$$\frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{q}_t, g(\boldsymbol{\theta}_t)\rangle \leq \min_{\boldsymbol{\theta}\in\Theta}\max_{i\in[k]} g_i(\boldsymbol{\theta}) + \frac{C}{\sqrt{T}}. \tag{40}$$

Step 3: Combining the bounds. Substituting equation 40 into equation 37:

$$\max_{i\in[k]} \frac{1}{T}\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \leq \min_{\boldsymbol{\theta}\in\Theta}\max_{i\in[k]} g_i(\boldsymbol{\theta}) + \frac{\log k}{\eta} + \frac{\log k}{\alpha T} + \frac{\alpha G^2}{2} + \frac{C}{\sqrt{T}}. \tag{41}$$

To balance the two α-dependent terms, we choose $\alpha^* = \frac{\sqrt{2\log k}}{G\sqrt{T}}$ so that $\frac{\log k}{\alpha T} = \frac{\alpha G^2}{2}$. Substituting into equation 41:

$$\max_{i\in[k]} \frac{1}{T}\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \leq \min_{\boldsymbol{\theta}\in\Theta}\max_{i\in[k]} g_i(\boldsymbol{\theta}) + \frac{\log k}{\eta} + \frac{G\sqrt{2\log k} + C}{\sqrt{T}}. \tag{42}$$

Setting $\eta = \frac{2\log k}{\epsilon}$ reduces the bias term to $\frac{\epsilon}{2}$. The $O(1/\sqrt{T})$ term reaches $\frac{\epsilon}{2}$ for $T \geq \frac{4(G\sqrt{2\log k}+C)^2}{\epsilon^2}$. Thus, for any T satisfying this condition:

$$\max_{i\in[k]} \frac{1}{T}\sum_{t=1}^{T} g_i(\boldsymbol{\theta}_t) \leq \min_{\boldsymbol{\theta}\in\Theta}\max_{i\in[k]} g_i(\boldsymbol{\theta}) + \epsilon. \quad \square \tag{43}$$

$\square$

D Baselines

In this section, we describe minor modifications we make to the original implementations of baselines.

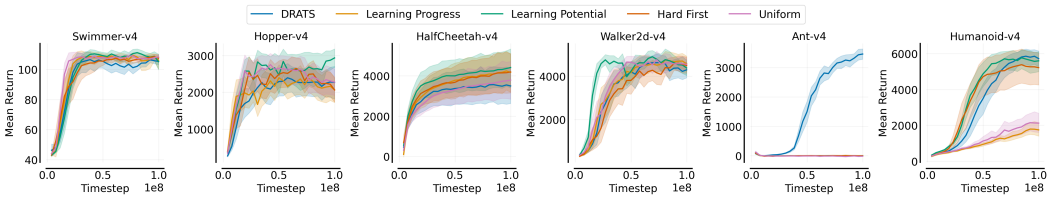
HARD FIRST (SMT). We make three modifications to SMT.

1. **No Resets.** SMT [Cho et al., 2024] periodically resets network parameters (without resetting the replay buffer) to mitigate primacy bias [Nikishin et al., 2022]. We do not implement network resets for two reasons. First, resets are designed for off-policy learning: after a reset, the agent can quickly relearn from its replay buffer using an off-policy algorithm. In on-policy learning, there is no replay buffer—on-policy algorithms discard data between updates because off-policy historic data biases gradient updates—so resetting network parameters is equivalent to restarting training entirely. Second, network resets are orthogonal to task sampling and could in principle be applied on top of any method (in an off-policy settings); including them for only SMT would confound the task-sampling comparison we aim to study.
2. **Task-Specific Thresholds.** We use task-specific thresholds M_i since tasks achieve different returns when solved, and using a single shared threshold might cause SMT to declare a task solved prematurely. We set each threshold by examining the training curves of DRATS to give SMT the best possible chance of success, and list all values in Table 3.
3. **Tasks can move between "solved" and "active" pools.** In the original SMT, once a task is declared solved or unsolvable, it is removed from the active pool and not resampled until the second stage. This is reasonable for off-policy learning, where the replay buffer retains data from inactive tasks and prevents forgetting. In on-policy learning, however, removing a task from the active pool causes the agent to forget it. We therefore allow tasks to move freely between the solved and active pools: if a solved task's return later drops below M_i, it is returned to the active pool.

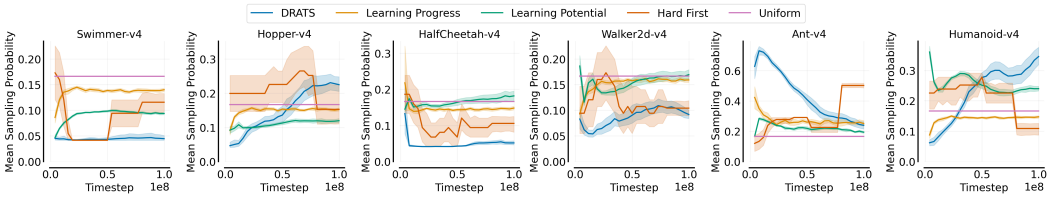
LEARNING PROGRESS (ALP-GMM). ALP-GMM [Portelas et al., 2020] was originally proposed for continuous task spaces, where a Gaussian Mixture Model (GMM) is used to estimate absolute learning progress (ALP) across the task parameter space and bias sampling toward high-ALP regions. In our setting, tasks are discrete, so the GMM is unnecessary: we directly track the absolute learning progress of each task as $z_i = |\widehat{J}_{t,i} - \widehat{J}_{t-1,i}|$ and update the task distribution $\boldsymbol{q}$ via mirror ascent with z_i in place of g_i.

LEARNING POTENTIAL (PLR). PLR [Jiang et al., 2021] originally constructs a task distribution by applying a rank-based prioritization function to the learning potential scores before normalizing into a distribution. In our setting, we update the task distribution $\boldsymbol{q}$ via mirror ascent with z_i in place of g_i to be consistent with how we update DRATS and LEARNING PROGRESS, ensuring that differences in performance reflect differences in prioritization strategy rather than differences in how scores are translated into sampling distributions.

E Additional Experimental Results



(a) Mean Return in each MuJoCo6 task.

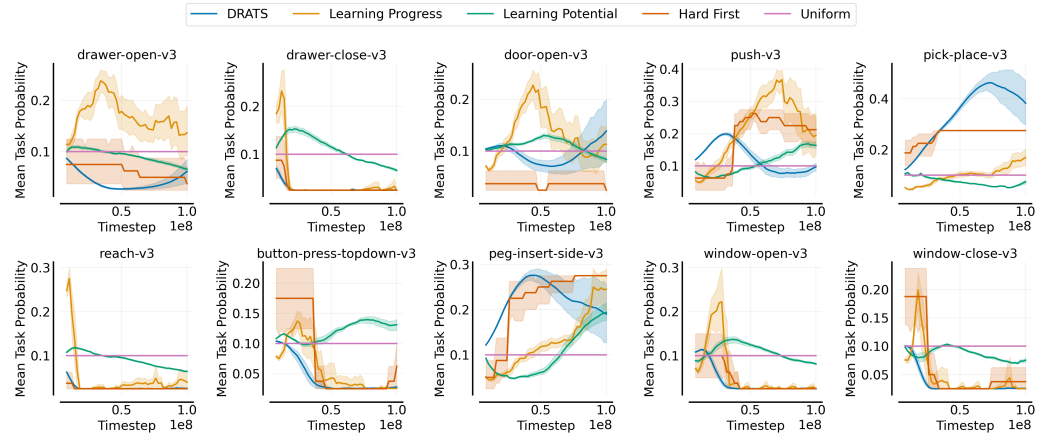


(b) Task sampling probabilities in each MuJoCo6 task.

Figure 7: Mean return and sampling probability in each MuJoCo6 task. Solid curves denote the mean over 20 seeds and shaded regions denote 95% bootstrap confidence intervals.



(a) Mean Return in each MT10 task.



(b) Task sampling probabilities in each MT10 task.

Figure 8: Mean return and sampling probability in each MT10 task. Solid curves denote the mean over 20 seeds and shaded regions denote 95% bootstrap confidence intervals.

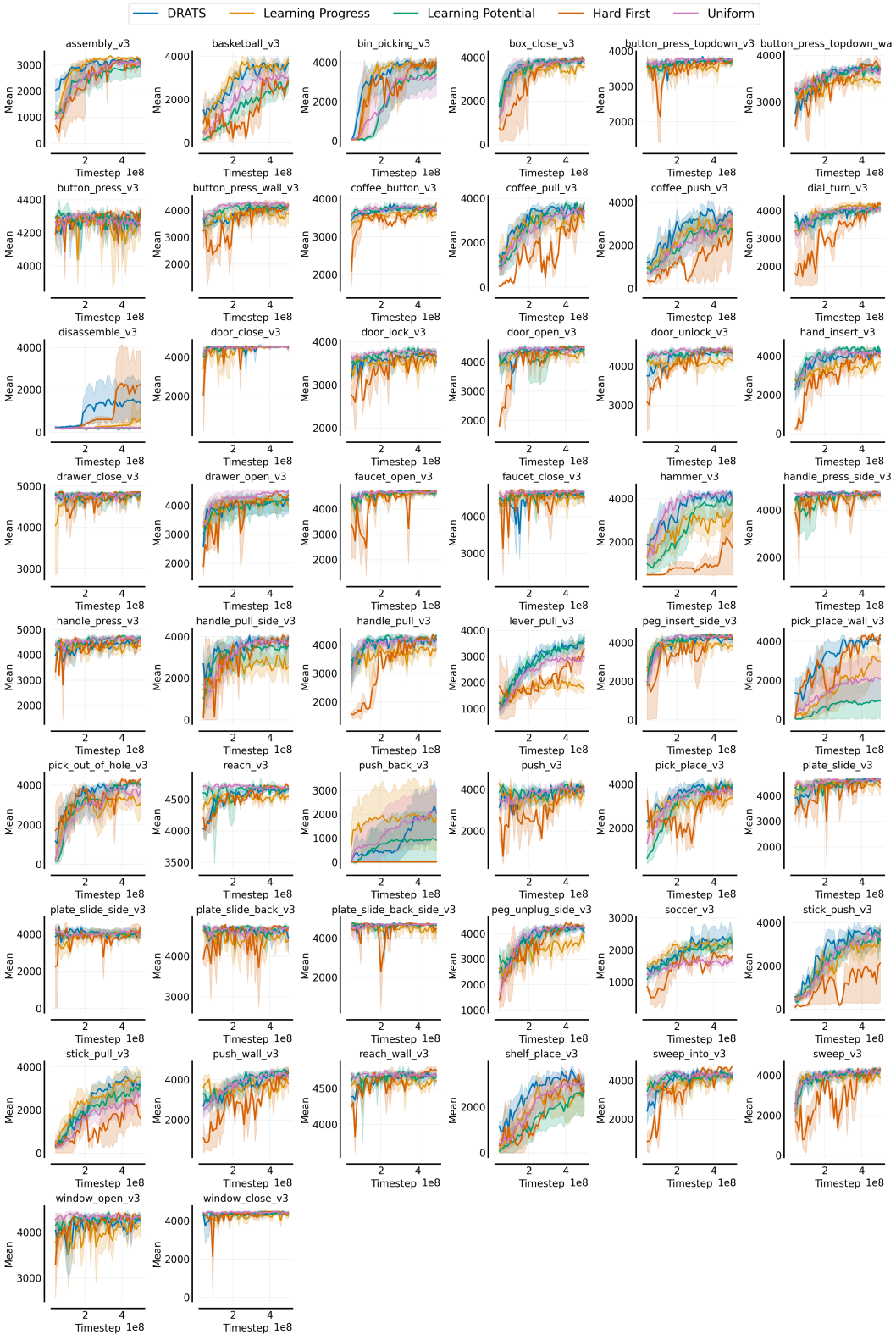


Figure 9: Task returns in MT50 tasks. Solid curves denote the mean over 10 seeds and shaded regions denote 95% bootstrap confidence intervals.

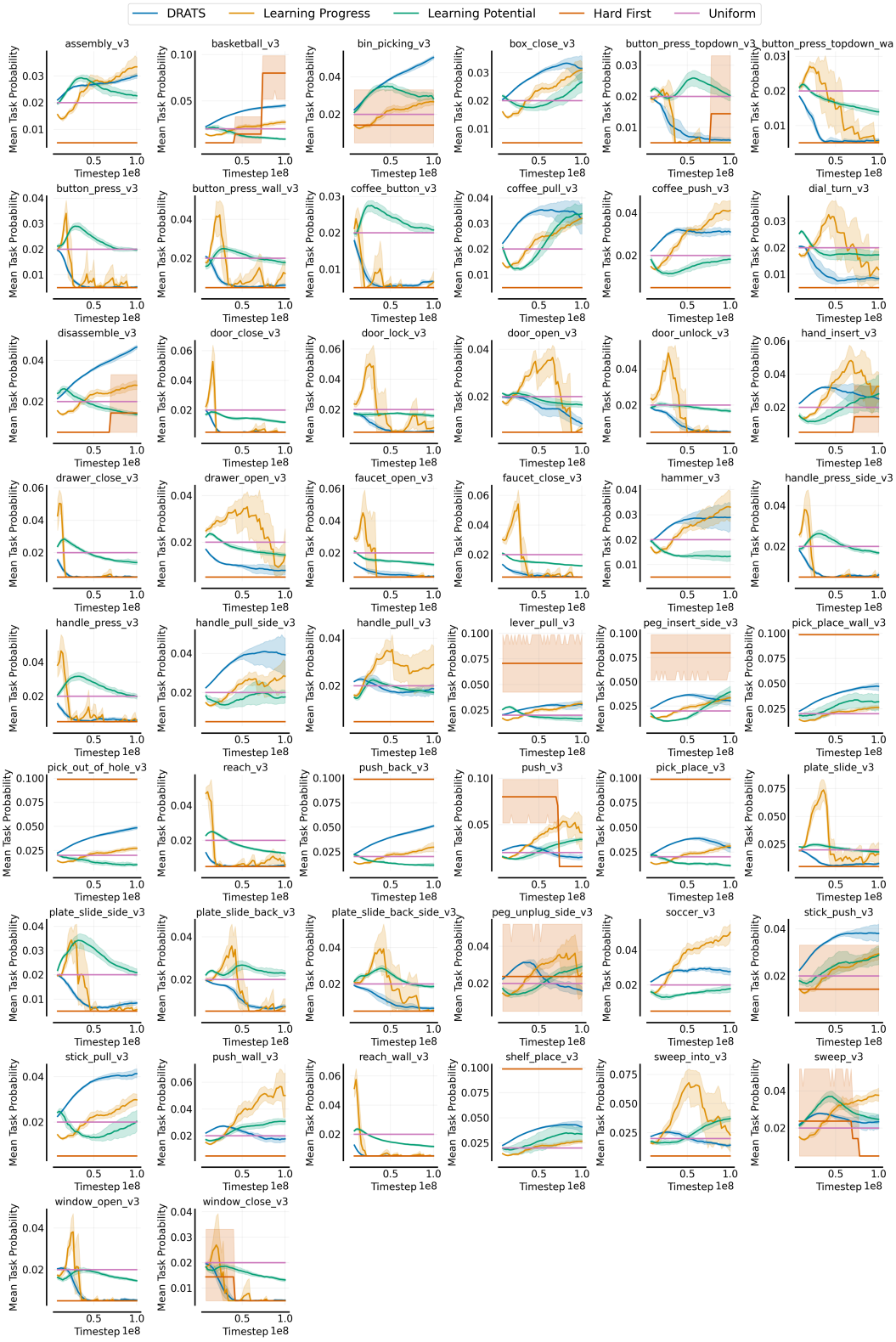
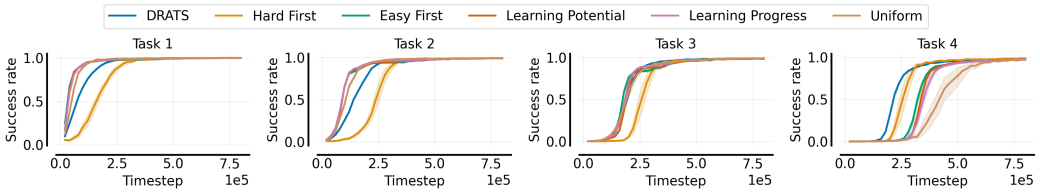
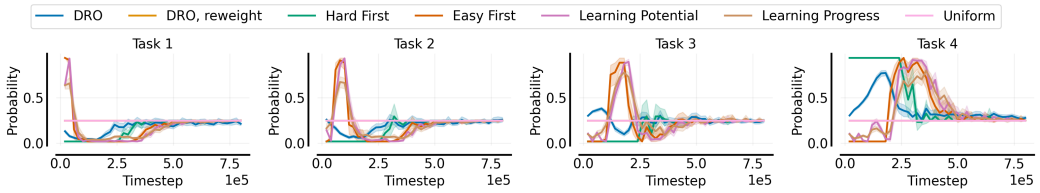


Figure 10: Task sampling probabilities in each MT50 task. Solid curves denote the mean over 10 seeds and shaded regions denote 95% bootstrap confidence intervals.



(a) Mean success rate on each task (50 seeds)



(b) Task sampling probabilities (50 seeds)

Figure 11: **(a)** . Mean success rates and task sampling probability in each Gridworld task with shared actor/critic networks (50 seeds). Shaded regions denote 95% bootstrap confidence intervals.

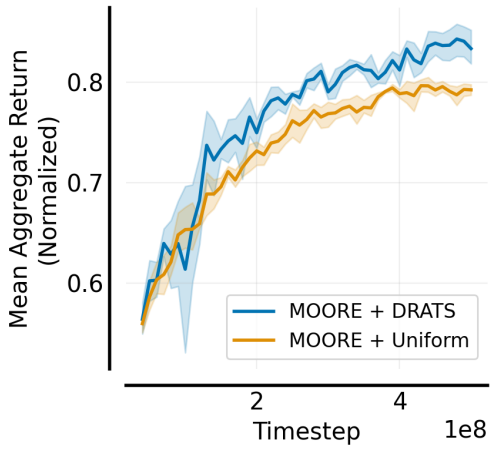
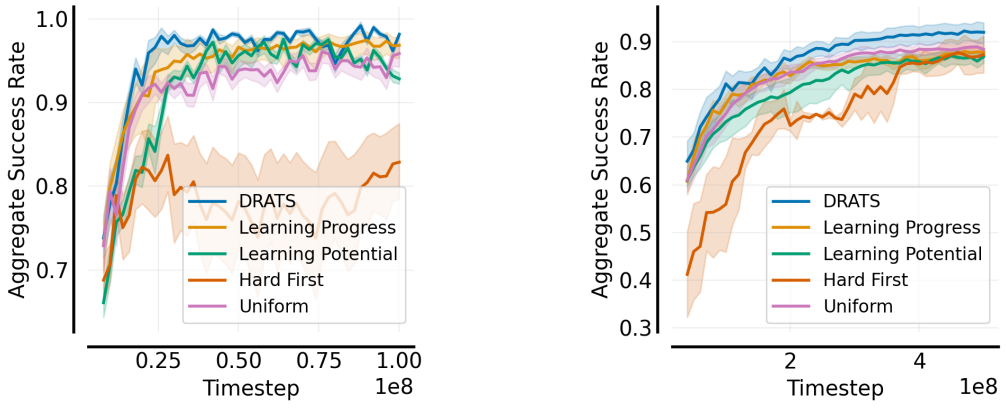


Figure 12: Combing DRATS with MOORE on MT50 (5 seeds). Shaded regions denote 95% bootstrap confidence intervals.



(a) MT10 (20 seeds)

(b) MT50 (10 seeds)

Figure 13: Mean aggregate success rates in MT10 and MT50. Shaded regions denote 95% bootstrap confidence intervals.

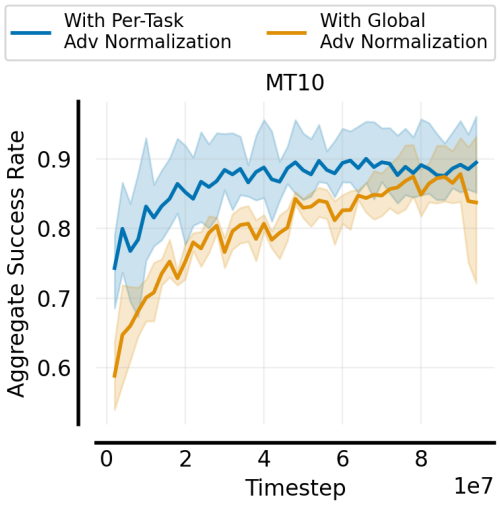


Figure 14: Aggregate success rate of DRATS in MT10 with per-task and global advantage normalization. These training curves use a smaller learning rate of 3e-4 (the default value in the `metaworld-algorithms` codebase [McLean et al., 2025a]) whereas all other MT10 experiments use 5e-4.

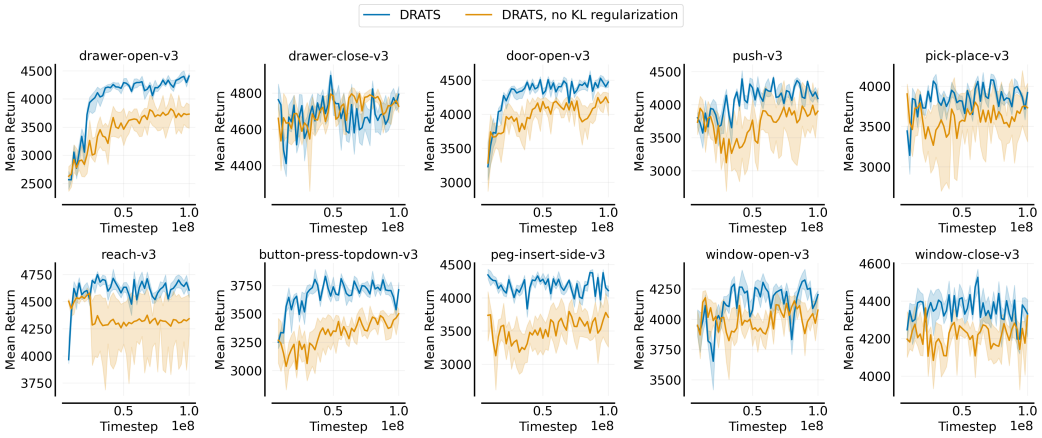


Figure 15: Per-task returns in MT10 for DRATS with and without KL regularization. Solid curves denote the mean and shaded regions denote 95% bootstrap confidence intervals.

F Additional Ablations

F.1 Adaptive Sampling vs. Objective Reweighting

In this section, we compare two ways of optimizing the DRATS objective : adaptive resampling of tasks from $\boldsymbol{q}$ (DRATS as described in the main paper), and reweighting the learning objective by q_i while sampling tasks uniformly.

The minimax objective in Eq. 3 can be optimized in two ways: by resampling tasks from $\boldsymbol{q}$ or by sampling tasks uniformly and reweighting each task's loss by q_i . Both are equivalent in expectation, since the expectation in Eq. 3 can be written as task reweighting: $\min_{\boldsymbol{\theta}} \max_{\boldsymbol{q} \in \Delta^k} \sum_{i=1}^{k} q_i g_i(\boldsymbol{\theta})$. Thus, DRATS can be implemented either by drawing tasks from $\boldsymbol{q}$ or by reweighting the returns of a uniformly sampled batch by q_i (UNIFORM + DRATS Reweighting). However, reweighting is known to produce higher-variance gradient estimates than resampling [An et al., 2020]. As shown in Fig. 17a, UNIFORM + DRATS Reweighting is much less data-efficient than DRATS: despite up-weighting hard tasks in the objective, it under-allocates interactions to them and continues sampling easy tasks

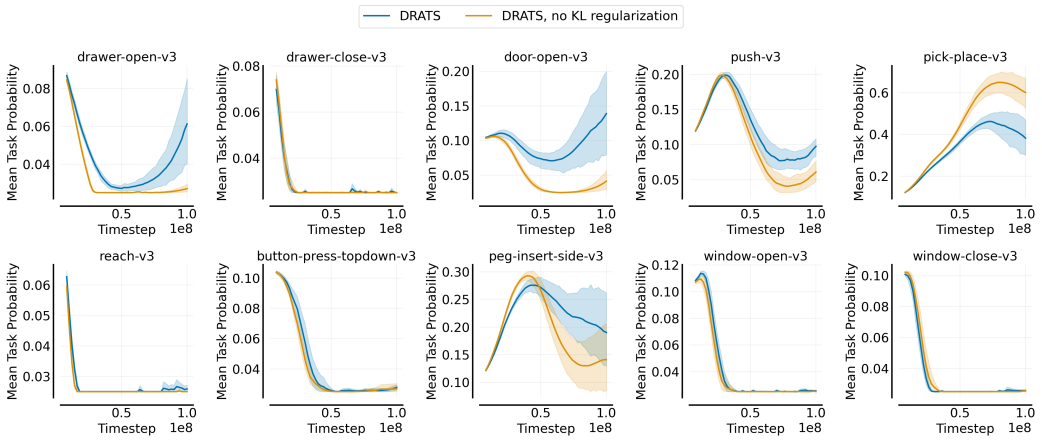
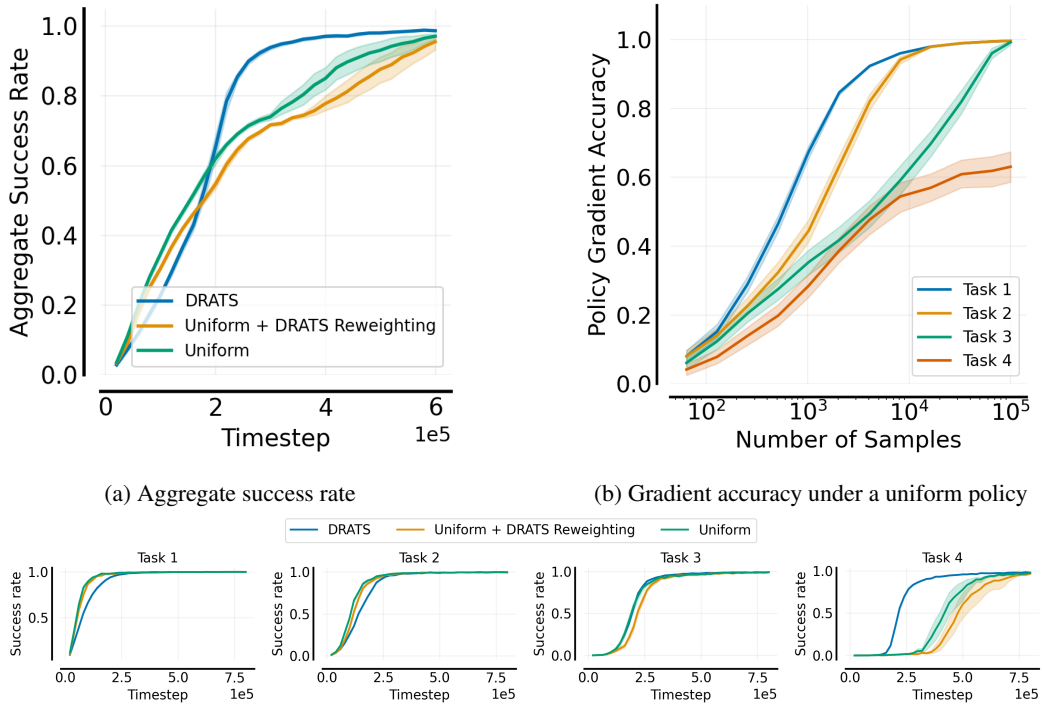


Figure 16: Task sampling probabilities in MT10 for DRATS with and without KL regularization. Solid curves denote the mean and shaded regions denote 95% bootstrap confidence intervals.



(a) Aggregate success rate

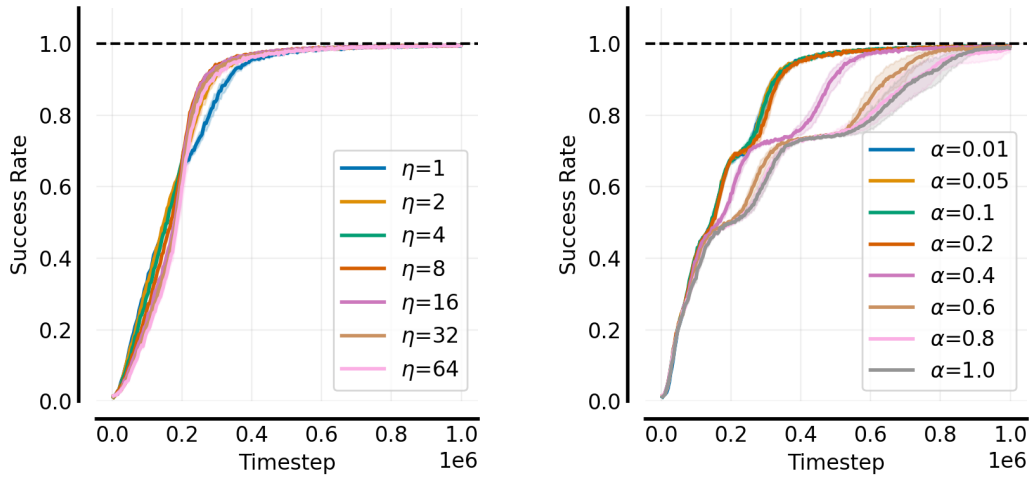
(b) Gradient accuracy under a uniform policy

(c) Success rate in each task.

Figure 17: Adaptive sampling vs. Objective reweighting on Gridworld (50 seeds). Shaded regions denote 95% bootstrap confidence belts.

even after they are solved. Notably, UNIFORM + DRATS Reweighting is even less data-efficient than UNIFORM: by down-weighting easy tasks in the objective, it solves them more slowly than UNIFORM and therefore benefits less from positive transfer.

Fig. 17c provides intuition for why resampling yields more data efficient learning than reweighting by plotting the accuracy of policy gradient estimates as a function of sample size, measured as the cosine similarity between a Monte Carlo estimate and the true policy gradient. Harder tasks require more samples to obtain accurate gradient estimates: achieving a cosine similarity of 0.2 across all four tasks requires data from each task in a $1 : 1.5 : 2 : 4$ ratio. Objective reweighting samples tasks uniformly, so gradient estimates are more accurate for easy tasks and less accurate for hard



(a) Average Success Rate

(b) Average Success Rate

Figure 18: DRATS ablation on η and α (50 seeds). Shaded regions denote 95% bootstrap confidence intervals.

ones. DRATS with adaptive sampling allocates more data to hard tasks, improving gradient accuracy precisely where it is most needed.

F.2 DRATS Hyperparameter Ablations

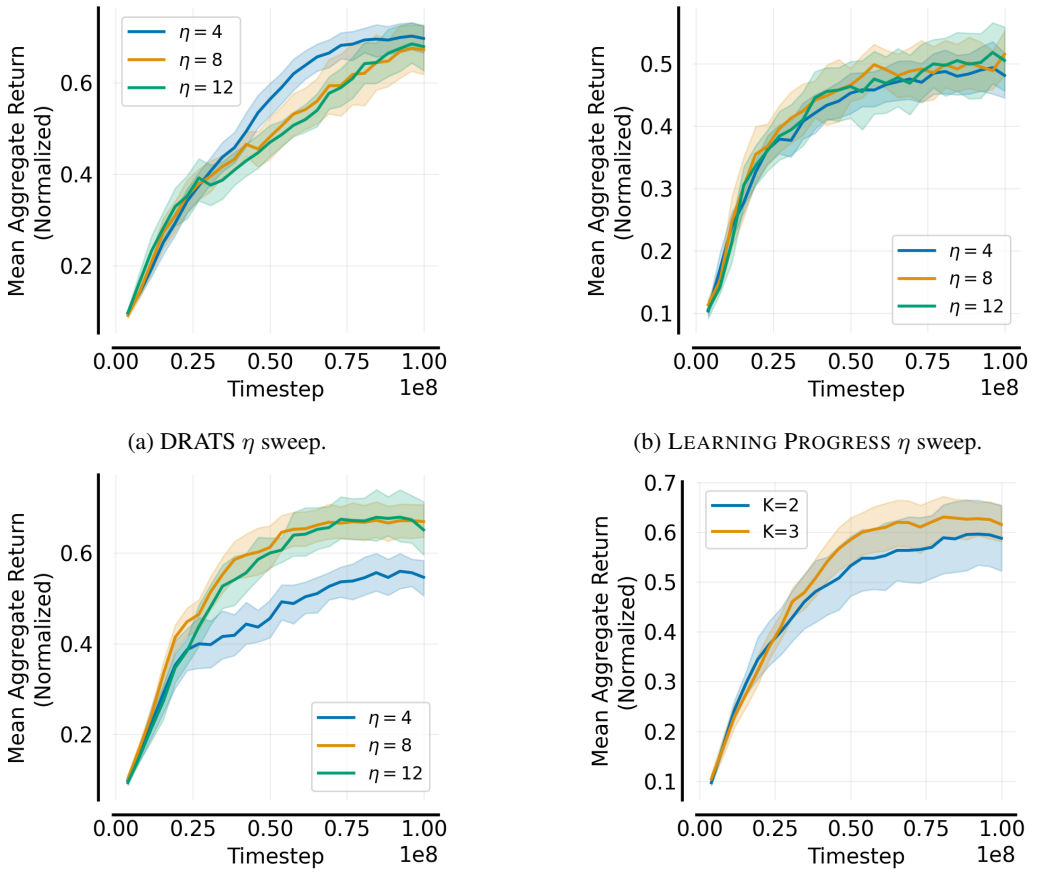
In this section, we ablate the two core hyperparameters of DRATS: the inverse temperature η, which controls the sharpness of the task sampling distribution, and the mirror ascent step size α, which controls how much the task distribution changes between updates. We vary one parameter at a time while keeping the other fixed at its base value, evaluating the following values:

- **Temperature:** $\eta \in \{1, 2, 4, 8, 16, 32, 64\}$
- **Step size:** $\alpha = c \cdot \eta$ for $c \in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$

A step size of $\alpha = \eta$ (*i.e.*, $c = 1$) corresponds to setting the task distribution to the "best response" $\boldsymbol{q}^\star$ (Eq. 6), so $c = 1$ denotes the maximum valid step size. A step size $\alpha < \eta$ corresponds to take a step toward $\boldsymbol{q}^*$. We note that our hyperparameter sweep on MuJoCo6 in Fig. 19 also ablates η for DRATS, LEARNING PROGRESS, and LEARNING POTENTIAL.

Inverse Temperature (η). Fig. 18a shows that DRATS is most data efficient with $\eta = 8$. Smaller values keep the task distribution closer to uniform, reducing the degree to which DRATS prioritizes hard tasks. Larger values concentrate too much probability on the hardest task, which harms data efficiency on easier tasks. Nevertheless, learning curves are nearly identical for $\eta \in [2, 16]$, indicating robustness to this hyperparameter.

Mirror Ascent Step Size (α). Fig. 18b shows that data efficiency improves monotonically as α decreases, with optimal performance at $\alpha = 0.01$. Large step sizes cause the task distribution to shift dramatically between updates, especially when a small number of trajectories are collected between updates. In particular, solved tasks are assigned low sampling probability under DRATS, so if the batch size is small, it may not contain any trajectories from these tasks. In such case, their return estimate defaults to zero and their return gaps thus appears large, causing a large step size to shift substantial probability back onto it despite the task being solved. Large step sizes cause the task distribution to shift dramatically between updates, which is especially problematic when only a few trajectories are collected per update. In particular, solved tasks are assigned low sampling probability under DRATS, so with a small batch size, the batch may contain no trajectories from these tasks; their return estimates then default to zero, making their return gaps appear large, ultimately causing a large step size to shift substantial probability back onto them despite the tasks



(a) DRATS η sweep.

(b) LEARNING PROGRESS η sweep.

(c) LEARNING POTENTIAL η sweep.

(d) HARD FIRST K sweep.

Figure 19: Hyperparameter sweeps in MuJoCo6. Shaded regions denote 95% bootstrap confidence intervals.

being solved. Smaller step sizes mitigate this issue by taking gradual steps toward $\boldsymbol{q}^*$, preventing noisy gap estimates from causing large swings in the sampling distribution. We note that in our MuJoCo6, MT10, and MT50 experiments, we collect several hundred trajectories between updates, so the DRATS sampling distribution evolves smoothly throughout training even with a larger step size of $\alpha = 0.5 \cdot \eta$ (Figs. 7b, 8b, and 10).

G Hyperparameters

In this section, we describe how we set hyperparameters for each baseline. We tune MuJoCo6 hyperparameters via parameter sweep; since MT10 and MT50 experiments are significantly more expensive, we use a single hyperparameter setting for each, following the original papers where possible. We fix the mirror ascent step size to $\alpha = 0.5 \cdot \eta$ for all methods. All hyperparameters are summarized in Tables 1 and 2.

DRATS. For MuJoCo6, we sweep $\eta \in \{4, 8, 12\}$ and report results for $\eta = 4$. For MT10 and MT50, we use $\eta = 8$ and $\eta = 3$, respectively.

HARD FIRST (SMT, Cho et al. [2024]). For MuJoCo6, we sweep $K \in \{2, 3\}$ with $B_1 = 80\%$ and find $K = 3$ is most data efficient. For MT10 and MT50, we follow Cho et al. [2024] and use $K = 3$ and $K = 8$ with $B_1 = 85\%$. Thresholds are listed in Table 3.

Description	Value
Episode length	500
Batch size	100,000 (200 episodes) / 500,000 (1000 episodes)
Number of epochs	16
Number of mini-batches per epoch	32
Discount factor	0.99
Clip ratio	0.2
Policy entropy coefficient	0.01 / 0.03
Optimizer learning rate	5e-4
Advantage estimate tau	0.95
Value Normalization by task	False
Input Normalization by task	False
Per-task Advantage Normalization	True
Separate critic and policy networks	True
Network Architecture	Multi-head MLP with layers (400, 400, 400)
DRATS-Specific Hyperparameters	
Inverse temperature η	8 / 3
Mirror ascent step size α	$0.5 \cdot \eta$
Learning Progress-Specific Hyperparameters	
Inverse temperature η	8 / 3
Mirror ascent step size α	$0.5 \cdot \eta$
Learning Potential-Specific Hyperparameters	
Inverse temperature η	8 / 3
Mirror ascent step size α	$0.5 \cdot \eta$
Hard First-Specific Hyperparameters	
Active task set size K	3 / 8
Stage 1 budget B_1	85% of total training budget
Thresholds m_i, M_i	See Table 3
MOORE-Specific Hyperparameters	
MoE experts	2
MoE layers	2
MoE hidden dim	256
Activation before/after task encoding weighting	[Linear, Linear]
Task encoder hidden sizes	[128]
Task encoder bias	False
Soft-Modularization-Specific Hyperparameters	
MoE experts	2
MoE layers	2
State encoder hidden sizes	256
Task encoder hidden sizes	256

Table 1: Hyperparameters used in MT10 / MT50.

LEARNING PROGRESS (ALP-GMM, Portelas et al. [2020]). For MuJoCo6, we sweep $\eta \in \{4, 8, 12\}$ and report results for $\eta = 12$. For MT10 and MT50, we use $\eta = 12$ and $\eta = 3$, respectively.

LEARNING POTENTIAL (PLR, Jiang et al. [2021]). For MuJoCo6, we sweep $\eta \in \{4, 8, 12\}$ and report results for $\eta = 8$. For MT10 and MT50, we use $\eta = 5$. We choose η larger than DRATS's for MT10/MT50 because our MuJoCo6 sweep showed LEARNING POTENTIAL performed best with a larger value than DRATS.

Description	Value
Episode length	1000
Batch size	48000 $\times$ 4 parallel environments
Number of epochs	16
Number of mini-batches per epoch	32
Discount factor	0.99
Clip ratio	0.2
Policy entropy coefficient	0.01
Optimizer learning rate	1e-4
Advantage estimate tau	0.95
Value Normalization by task	True
Input Normalization by task	True
Per-task Advantage Normalization	True
Separate critic and policy networks	True
Network Architecture	Multi-head MLP with layers (256, 256, 256)
DRATS-Specific Hyperparameters	
Inverse temperature η	4
Mirror ascent step size α	$0.5 \cdot \eta$
Learning Progress-Specific Hyperparameters	
Inverse temperature η	8
Mirror ascent step size α	$0.5 \cdot \eta$
Learning Potential-Specific Hyperparameters	
Inverse temperature η	8
Mirror ascent step size α	$0.5 \cdot \eta$
Hard First-Specific Hyperparameters	
Active task set size K	3
Stage 1 budget B_1	80% of total training budget
Thresholds m_i, M_i	See Table 3

Table 2: RL hyperparameters used in MuJoCo6.

Task	m_i	M_i
Swimmer-v4	60	90
Hopper-v4	1500	2500
HalfCheetah-v4	1500	3000
Walker2d-v4	1500	4000
Ant-v4	1000	3000
Humanoid-v4	2000	5000
assembly-v3	2000	3000
basketball-v3	2000	2000
bin-picking-v3	2000	3000
box-close-v3	2000	3000
button-press-topdown-v3	2000	3000
button-press-topdown-wall-v3	2000	3000
button-press-v3	2000	3500
button-press-wall-v3	2000	3500
coffee-button-v3	2000	3000
coffee-pull-v3	2000	2000
coffee-push-v3	2000	2000
dial-turn-v3	2000	3500
disassemble-v3	2000	3000
door-close-v3	2000	4000
door-lock-v3	2000	3000
door-open-v3	2000	4000
door-unlock-v3	2000	3000
hand-insert-v3	2000	3500
drawer-close-v3	2000	4000
drawer-open-v3	2000	4000
faucet-open-v3	2000	4000
faucet-close-v3	2000	4000
hammer-v3	2000	3000
handle-press-side-v3	2000	4000
handle-press-v3	2000	4000
handle-pull-side-v3	2000	3000
handle-pull-v3	2000	3500
lever-pull-v3	2000	2500
pick-place-wall-v3	2000	3000
pick-out-of-hole-v3	2000	3000
pick-place-v3	2000	3000
plate-slide-v3	2000	4000
plate-slide-side-v3	2000	3500
plate-slide-back-v3	2000	4000
plate-slide-back-side-v3	2000	4000
peg-insert-side-v3	2000	3500
peg-unplug-side-v3	2000	3500
soccer-v3	2000	2000
stick-push-v3	2000	2500
stick-pull-v3	2000	2000
push-v3	2000	3000
push-wall-v3	2000	3000
push-back-v3	2000	3000
reach-v3	2000	4000
reach-wall-v3	2000	4000
shelf-place-v3	2000	3000
sweep-into-v3	2000	3500
sweep-v3	2000	3000
window-open-v3	2000	4000
window-close-v3	2000	4000

Table 3: HARD FIRST (SMT) unsolvable (m_i) and solved (M_i) thresholds for all tasks.

H Computational Resources

A full MT10 or MT50 training run takes 20-30 hours on a single A100 GPUs with 80GB of memory. We have access to a computing cluster with many GPUs and train each agent on separate GPU and use parallel environments across 10 CPUs. MT10 runs require 12GB memory and 15GB disk; MT50 runs require 85GB memory and 15GB disk. We run MuJoCo6 experiments on 4 CPUs only, and each training run takes approximately 40 hours.