POLTER: Policy Trajectory Ensemble Regularization for Unsupervised Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

The goal of Unsupervised Reinforcement Learning (URL) is to find a reward-agnostic prior policy on a task domain, such that the sample-efficiency on supervised downstream tasks is improved. Although agents initialized with such a prior policy can achieve a significantly higher reward with fewer samples when finetuned on the downstream task, it is still an open question how an optimal pretrained prior policy can be achieved in practice. In this work, we present POLTER (Policy Trajectory Ensemble Regularization) – a general method to regularize the pretraining that can be applied to any URL algorithm and is especially useful on data- and knowledge-based URL algorithms. It utilizes an ensemble of policies that are discovered during pretraining and moves the policy of the URL algorithm closer to its optimal prior. Our method is based on a theoretical framework, and we analyze its practical effects on a white-box benchmark, allowing us to study POLTER with full control. In our main experiments, we evaluate POLTER on the Unsupervised Reinforcement Learning Benchmark (URLB), which consists of 12 tasks in 3 domains. We demonstrate the generality of our approach by improving the performance of a diverse set of data- and knowledge-based URL algorithms by 19% on average and up to 40% in the best case. Under a fair comparison with tuned baselines and tuned POLTER, we establish a new state-of-the-art for model-free methods on the URLB.

1 Introduction

Reinforcement Learning (RL) has shown many successes in recent years (Mnih et al., 2015; Silver et al., 2018; OpenAI et al., 2019) and is starting to have an impact on real-world applications (Bellemare et al., 2020; Mirhoseini et al., 2021; Degrave et al., 2022). However, all these applications require detailed knowledge of the task to train the agents in a sufficiently close simulation. Implementing each and every task and training a new agent is time-consuming and inefficient. In some cases, the task might be difficult to model in simulation, or it is unknown beforehand, which makes sample-efficiency even more important. Unsupervised Reinforcement Learning (URL), i.e., reward-free pretraining in a task domain (Srinivas and Abbeel, 2021), has shown to improve the sample-efficiency of finetuning RL algorithms on downstream tasks. This setup only requires a simulation of the agent's domain without having to model task-specific components in detail. Although URL results in a higher sample-efficiency on some applications, pretraining a policy still does not guarantee an improvement in performance on a wide range of tasks. For example, Laskin et al. (2021) observed that longer pretraining often degrades the performance of many URL algorithms. Also, many finetuning steps are still needed to achieve the optimal return, and some algorithms fail to learn a sufficient policy at all.

In this work, we introduce POLTER (Policy Trajectory Ensemble Regularization) – a general method to improve the performance of URL algorithms via a novel regularization term during pretraining that results in better prior policies for finetuning on specific tasks. Our theoretical motivation of POLTER builds on the geometric interpretation of URL by Eysenbach et al. (2022) who connect the optimal prior policy to a specific state distribution. Some URL algorithms try to approximate this optimal prior state distribution by learning separate skills during pretraining that are combined to initialize the finetuning policy. POLTER borrows this idea for URL algorithms that do not learn explicit skills, e.g., data- and knowledge-based URL

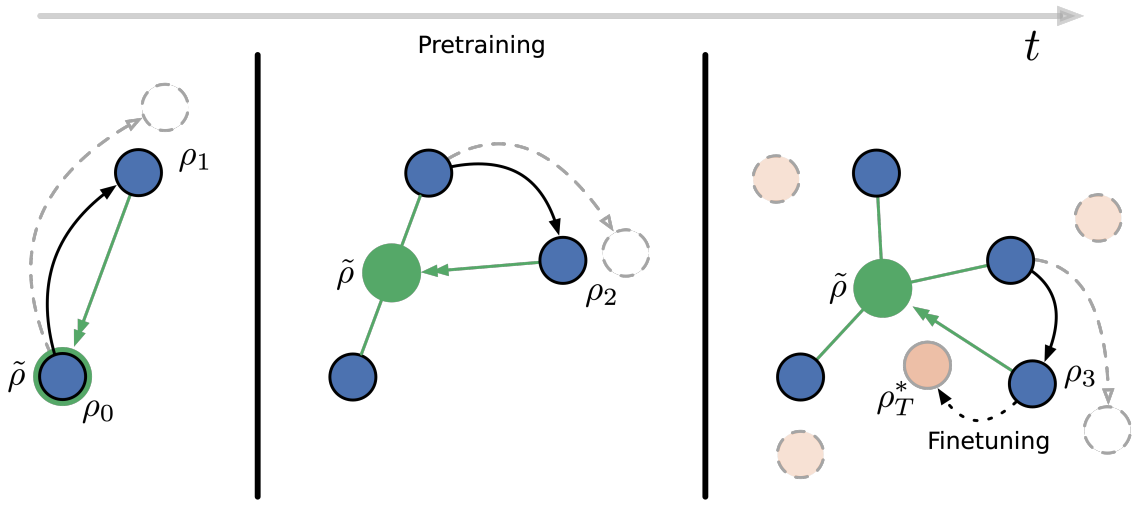


Figure 1: Optimization trajectory in the induced state-marginal space $\rho(s)$ of a policy $\pi(s)$ that optimizes an Unsupervised Reinforcement Learning objective with Policy Trajectory Ensemble Regularization (POLTER) regularization. The regularized policy's state distribution $\rho_t(s)$ is pulled towards the ensemble state distribution $\tilde{\rho}(s)$ during pretraining. The ensemble policy $\tilde{\pi}(s)$ approximates the optimal prior $\pi_T^*(s)$ with its induced state-marginal $\rho_T^*(s)$ for finetuning on a specific task T. Note that other tasks might lie further away from the average. However, as shown by Eysenbach et al. (2022), the average still minimizes the distance without knowledge about the specific downstream task. The state distribution trajectory of a policy without POLTER regularization is indicated by dashed lines.

algorithms. This is based on the insight that each point during pretraining encodes an implicit skill that depends on the pretraining objective. In a nutshell, our main idea for POLTER is to minimize the distance of the current policy to an ensemble policy during pretraining, which acts as a proxy for the optimal prior. As shown in Figure 1, the pretraining trajectory of the policy and hence the induced state distribution is affected by our regularization and attracted to the ensemble of earlier pretraining policies, each potentially encoding a different skill.

We demonstrate the effect of POLTER on the simplistic PointMass environment and two tasks in the Pendulum domain, and provide empirical evidence that our method reduces the distance to the optimal prior policy and improves the downstream performance. Extensive experiments on the Unsupervised Reinforcement Learning Benchmark (URLB) (Laskin et al., 2021) show that our method improves the Interquartile Mean (IQM) (Agarwal et al., 2021) performance of data- and knowledge-based URL algorithms on average by 19%. A sensitivity analysis reveals that POLTER is a robust method with a reasonable default setting but can further be tuned on some domains and algorithms, which enables us to set a new state-of-the-art of 84% IQM expert performance in the URLB. Additional experiments on the state-visitation entropy provide empirical evidence that POLTER improves the performance by finding better priors in contrast to an improved exploration. In summary, our **contributions** are:

1. We introduce POLTER, a novel regularization method for URL algorithms that uses an ensemble of pretraining policies to reduce the deviation from the optimal prior policy.
2. Our method is empirically validated on the white-box PointMass benchmark and demonstrated in the Pendulum domain.
3. An extensive evaluation on the URLB shows the effectiveness of POLTER for a broad range of URL algorithms.
4. With our sensitivity analysis, we demonstrate that POLTER is robust with a good default and can be further boosted via a task-specific strength of the regularization and achieves a new state-of-the-art for model-free algorithms on the URLB.

2 Preliminaries

2.1 Notation in Reinforcement Learning (RL)

RL relies on the notion of a Markov Decision Process (MDP) (Sutton and Barto, 1998). An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ which describes the states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $p(s_{t+1} \mid s_t, a_t) \sim \mathcal{P}$, initial state distribution $p_0(s_0) \sim \mathcal{P}$, reward function $r(s_t)$ and discount factor $\gamma \in (0, 1]$ of an environment. At each time step t, an agent observes the state s_t of the environment and chooses an action a_t using a policy $\pi(a_t \mid s_t)$ to transition into a new state s_{t+1}. The objective is to maximize the expected discounted return by finding a policy $\pi^*(s)$ that induces the optimal discounted state occupancy measure $\rho^*(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P_t^{\pi^*}(s)$, where $P_t^{\pi^*}(s)$ is the probability of being in state s at time step t when following policy π^*. As we focus on state-based reward functions $r(s)$, this optimal discounted state occupancy measure corresponds to the maximal discounted return of the optimal policy.

2.2 Related Work

Unsupervised Reinforcement Learning describes algorithms that train an RL agent without a learning signal, i.e., without a task-specific reward function. It encompasses two main research branches: The first is Unsupervised Representation Learning (van den Oord et al., 2018; Ebert et al., 2018; Ha and Schmidhuber, 2018; Laskin et al., 2020; Schwarzer et al., 2021a; Stooke et al., 2021) which focuses on learning a representation of the states or dynamics from noisy or incomplete observations. The second branch is called Unsupervised Behavioral Learning.

Unsupervised Behavioral Learning is a variant of the RL problem where the agent can interact with the environment in a reward-free setting before being assigned the downstream task T defined by an external reward function (Oudeyer et al., 2007; Jin et al., 2020). During pretraining, intrinsic rewards are used to train a policy to become an optimal prior and initialization for the subsequent finetuning task. We denote the policy after finetuning as π_T^* and the corresponding optimal oracle policy on task T as π_T^+. Algorithms for Unsupervised Behavioral Learning can be categorized into knowledge-, data- and competence-based approaches (Srinivas and Abbeel, 2021). **Knowledge-based** algorithms define a self-supervised task by making predictions on some aspect of the environment (Pathak et al., 2017). The approach **data-based** methods follow is to maximize the state visitation entropy to explore the environment (Hazan et al., 2019). **Competence-based** algorithms learn skills that maximize the mutual information between the trajectories and a space of skills (Mohamed and Rezende, 2015; Gregor et al., 2017). All categories have one thing in common: They traverse the policy space during pretraining and potentially encounter behavior close to the optimal prior policy for the finetuning tasks. There are also algorithms that show great performance by combining both branches of URL by simultaneously learning the dynamics and possible behaviors in a domain (Yuan et al., 2022). In this work, however, we focus on model-free approaches.

3 Method

In the following, we first discuss the motivation of our approach, called POLTER, by linking it to the theoretical framework of Eysenbach et al. (2022). Based on that, we derive the regularization strategy of POLTER using a mixture ensemble policy and discuss the concrete implementation of our method. To ground our theoretical motivation, we show exemplary behaviour of POLTER on simple control tasks.

3.1 Motivation

Our motivation for POLTER follows the theoretical framework of URL by Eysenbach et al. (2022). Their adaptation objective connects the policy prior after pretraining with the finetuned policy that has been adapted to an arbitrary downstream task:

$$\min_{\rho_T^*(s)\in\mathcal{C}} \underbrace{\max_{\rho_T^+(s)\in\mathcal{C}} \mathop{\mathbb{E}}_{s\sim\rho_T^+(s)}[r(s)] - \mathop{\mathbb{E}}_{s\sim\rho_T^*(s)}[r(s)]}_{\text{worst-case regret}} + \underbrace{D_{\mathrm{KL}}(\rho_T^*(s) \parallel \rho(s))}_{\text{information cost}} \tag{1}$$

The first term in Equation (1) represents the regret that a policy $\pi_T^*(s)$ achieves after finetuning with respect to an optimal oracle policy $\pi_T^+(s)$. The second term defines the information cost between the pretraining policy $\pi_T^*(s)$ and $\pi(s)$ via their induced state-marginals $\rho_T^*(s)$ and $\rho(s)$. The set of all feasible state-marginal distributions in an environment is denoted as $\mathcal{C}$.

Given Equation (1), Eysenbach et al. showed that the optimal prior policy after pretraining induces the average state marginal of the skills. Competence-based algorithms estimate this prior policy by averaging their skill space. Data- and knowledge-based algorithms, however, do not try to estimate the optimal prior and thus perform worse in the URL setting (Laskin et al., 2022). Additionally, these algorithms explore the policy space of the domain on a random trajectory that might be close to the optimal prior at one point, but can deviate from it arbitrarily everywhere else. These disadvantages motivate our key idea to improve the sample-efficiency of data- and knowledge-based URL algorithms.

3.2 POLTER: Policy Trajectory Ensemble Regularization

Our goal is to increase the sample-efficiency when finetuning from any snapshot during pretraining by pulling the policy of the URL algorithm towards the optimal prior policy $\pi_T^*(s)$. Eysenbach et al. (2022) proved that the optimal prior for competence-based algorithms under Equation (1) is given by the policy that induces the average state-marginal distribution of the skills. Data- and knowledge-based URL algorithms do not learn explicit skills. However, their objective at each point during pretraining encodes a different reward function, e.g., when they update their knowledge or observe new data. Given that the agent can maximize this reward, the intrinsic objective at different moments of pretraining results in a different implicit skill. Therefore, the policy at that point is conditioned on the skill.

This can be expressed as the latent variable model in Figure 2 where everything is observed except for the skill z which determines the intrinsic return r_{int}. At different points during the pretraining steps $0 \leq t < N_{\text{PT}}$, we observe different intrinsic returns for taking an action a in a state s. Thus, the underlying skill that the agent is supposed to learn must have changed. Therefore, we can approximate the optimal policy by an ensemble consisting of a set of policies from different points of pretraining.

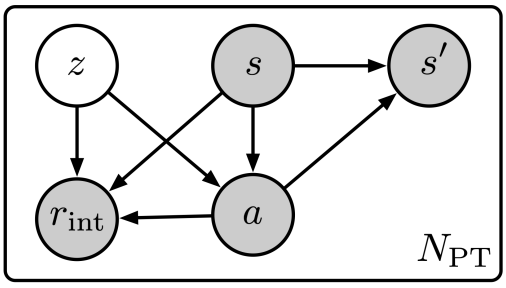


Figure 2: Probabilistic graphical model that underlies the assumption of POLTER for approximating the optimal policy using an ensemble of pretraining policies. The latent skill z determines the intrinsic return r_{int} via the pretraining objective, and the actions a of the agent's policy are implicitly conditioned on it.

The policy loss $\mathcal{L}^{\text{URL}}$ of the RL agent is augmented by our approach POLTER with the following regularization term:

$$\mathcal{L}^{\text{POLTER}}(\pi) = \mathcal{L}^{\text{URL}}(\pi) + \alpha D_{\text{KL}}(\tilde{\pi} \parallel \pi) \qquad (2)$$

where $\tilde{\pi}(s) = \frac{1}{k} \sum_k \pi_k(s)$ is the mixture ensemble policy consisting of k previous policies from the URL trajectory. In Figure 1, we illustrate the progression of the optimization trajectory in the induced state-marginal space $\rho(s)$ of a policy $\pi(s)$ under the URL objective with POLTER regularization. Along its trajectory, the policy $\pi(s)$ is attracted to the ensemble policy $\tilde{\pi}$ and, thus, closer to the optimal prior π_T^* for a specific finetuning task T. This results in a reduced information cost to adapt the pretraining policy to the task and improves the sample-efficiency of the regularized URL algorithm.

We outline the combination of URL with our regularization POLTER in Algorithm 1. We fix the ensemble policy to a finite set of states to stabilize training, i.e., each time we add a member to the ensemble (Line 5), we condition the ensemble policy on the next state. Finally, we compute the Kullback–Leibler divergence (KL-divergence) loss and add it to the policy loss of the URL algorithm (Line 11). The memory overhead is dominated by the requirement to store the ensemble policies. For the computational overhead, we only have to consider the k forward passes for conditioning the ensemble policy on a state when adding a new member.[1]

[1]In our experiments, each policy takes neglected amount of around 4MB of storage and the wall-clock time increases by 12%.

Algorithm 1 URL+POLTER

Require: Initialize URL algorithm, policy $\pi_{\theta,0}$, replay buffer $\mathcal{D}$, pretraining steps N_{PT} ▷ URL init
Require: Empty ensemble $E = \emptyset$, ensemble snapshot time steps $\mathcal{T}_E$, regularization weight α ▷ POLTER init
for $t = 0 \dots N_{\text{PT}} - 1$ **do** ▷ Unsupervised pretraining
  if Beginning of episode **then**
    Observe initial state $s_t \sim p_0(s_0)$
    if $t \in \mathcal{T}_E$ **then** ▷ Update ensemble policy
      Extend ensemble $E \leftarrow E \cup \pi_{\theta,t}$
      Update ensemble policy $\tilde{\pi} \leftarrow \frac{1}{|E|} \sum_{\pi \in E} \pi(s_t)$
  Choose action $a_t \leftarrow \pi_{\theta,t}(a_t \mid s_t)$
  Observe next state $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
  Add transition to replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, s_{t+1})$
  Sample a minibatch $B \sim \mathcal{D}$
  Compute loss $\mathcal{L}^{\text{POLTER}} = \mathcal{L}^{\text{URL}}(\pi_{\theta,t}) + \alpha D_{\text{KL}}(\tilde{\pi} \parallel \pi_{\theta,t})$
  Update policy $\pi_{\theta,t}$ with $\mathcal{L}^{\text{POLTER}}$
...
▷ Supervised finetuning on task T

3.3 Demonstration on PointMass and Pendulum

PointMass To ground our theoretical motivation in empirical evidence, we evaluate the effect of our regularization on Random Network Distillation (RND) (Burda et al., 2019), a well-known knowledge-based URL algorithm, in the simplistic PointMass environment (Tassa et al., 2018; 2020). In PointMass, we can define the optimal prior $\pi_T^*(s)$ that maximizes the average performance for the downstream finetuning tasks. This optimal prior policy moves the ball into the center of the area, which minimizes the distance to any possible target location that represent a finetuning task. With the optimal prior, we can compute the KL-divergence between the optimal prior policy and the current pretraining policy $D_{\text{KL}}(\pi_T^* \parallel \pi)$. With POLTER, the KL-divergence to the optimal prior policy π_T^* is generally lower throughout the whole pretraining phase. For more details on this demonstration, see Appendix A.

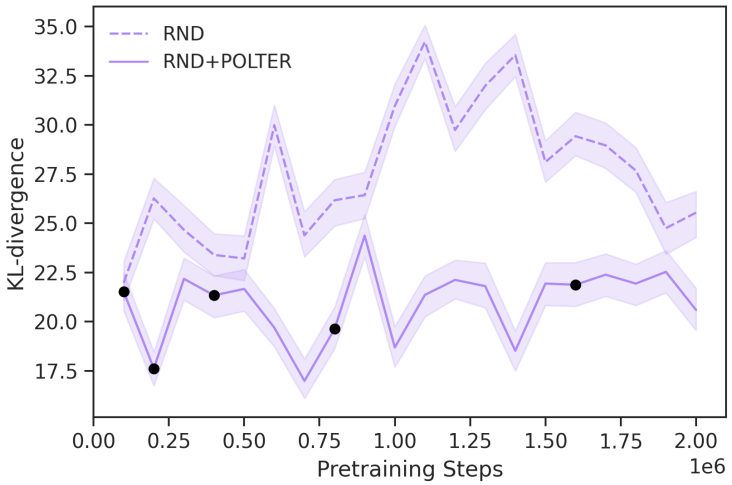
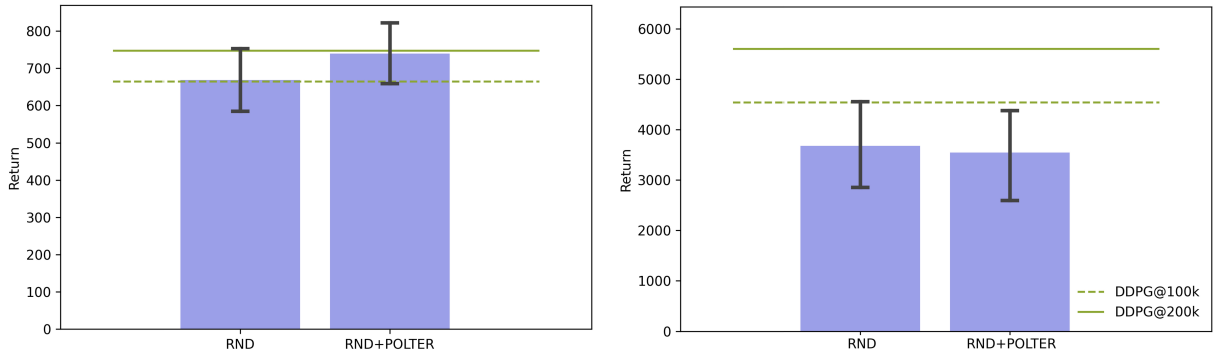


Figure 3: Average KL-divergence of RND (dashed) and RND+POLTER (solid) between the pretraining policy $\pi(s_0)$ and the optimal pretraining policy $\pi_T^*(s_0)$ in the PointMass domain during reward-free pretraining. The shaded area indicates the standard error over 10 seeds and the dots are the pretraining checkpoints that were used for the ensemble.

Pendulum As described in Figure 1, the effect of POLTER on the performance on a specific finetuning task in a given domain depends on the optimal policy for the given task. Thus, even though POLTER improves the average performance over all possible tasks, it might have no or even a detrimental effect on single downstream tasks. To test this hypothesis, we train POLTER on the Pendulum domain for two



(a) SwingUp (b) Propeller

Figure 4: Finetuning performance of RND and RND+POLTER in the Pendulum domain averaged over 10 seeds after 100 k of pretraining. The baseline of Deep Deterministic Policy Gradient (DDPG) is trained for 100 k or 200 k steps without pretraining. The error bars indicate the standard error over 10 seeds.

different downstream tasks. The first one is the known *SwingUp* task, where the agent has to balance the pole in an upright position. The second task is a *Propeller* task, which rewards the maximization of the angular velocity of the pole. This task's optimal policy induces a state-marginal that is further away from the average, which is reflected in Figure 4 in a slightly decreased performance when applying POLTER to RND. However, POLTER consistently increases the performance on the regular *SwingUp* task. Further results for this experiment can be found in Appendix B.

4 Experiments

With our theoretical motivation and our prospect on the toy environments PointMass and Pendulum, we now address the following empirical questions at a larger scale: **(Q1)** How does POLTER affect the performance after pretraining, and does the performance differ across URL algorithm categories? **(Q2)** Does POLTER improve the sample-efficiency for finetuning? **(Q3)** How does the performance vary with an increasing number of pretraining steps? **(Q4)** How does the strength of the regularization affect the performance of different algorithms in different domains?

To answer these questions, we evaluate several different URL algorithms with and without our POLTER regularization on the Unsupervised Reinforcement Learning Benchmark (Laskin et al., 2021). We use the provided code from Laskin et al. (2021) to aid reproducibility and provide our source code in the supplementary material.

Knowledge-based algorithms, such as RND (Burda et al., 2019), Disagreement (Pathak et al., 2019), and Intrinsic Curiosity Module (ICM) (Pathak et al., 2017), use the norm or variance of their prediction errors of some aspect of the environment as a learning signal. The second category consisting of Active Pre-training (APT) (Schwarzer et al., 2021b), and ProtoRL (Yarats et al., 2021), are data-based algorithms, that try to maximize the entropy of the state-visitation distribution to explore the environment. The last category is competence-based algorithms that try to learn a set of skills by maximizing the mutual information between a learned skill vector and some features of the observed states. Besides the three algorithms that were used by Laskin et al. (2021), Active Pre-training with Successor Features (APS) (Liu and Abbeel, 2021), State Marginal Matching (SMM) (Lee et al., 2020), and "Diversity is All You Need" (DIAYN) (Eysenbach et al., 2019), we evaluate our method with a newer method from this category, called Contrastive Intrinsic Control (CIC) (Laskin et al., 2022)[2].

[2]Since running these experiments, there has been a new publication by Zhao et al. (2022) that extended the CIC algorithm. However, their performance only improves slightly on the results of CIC.

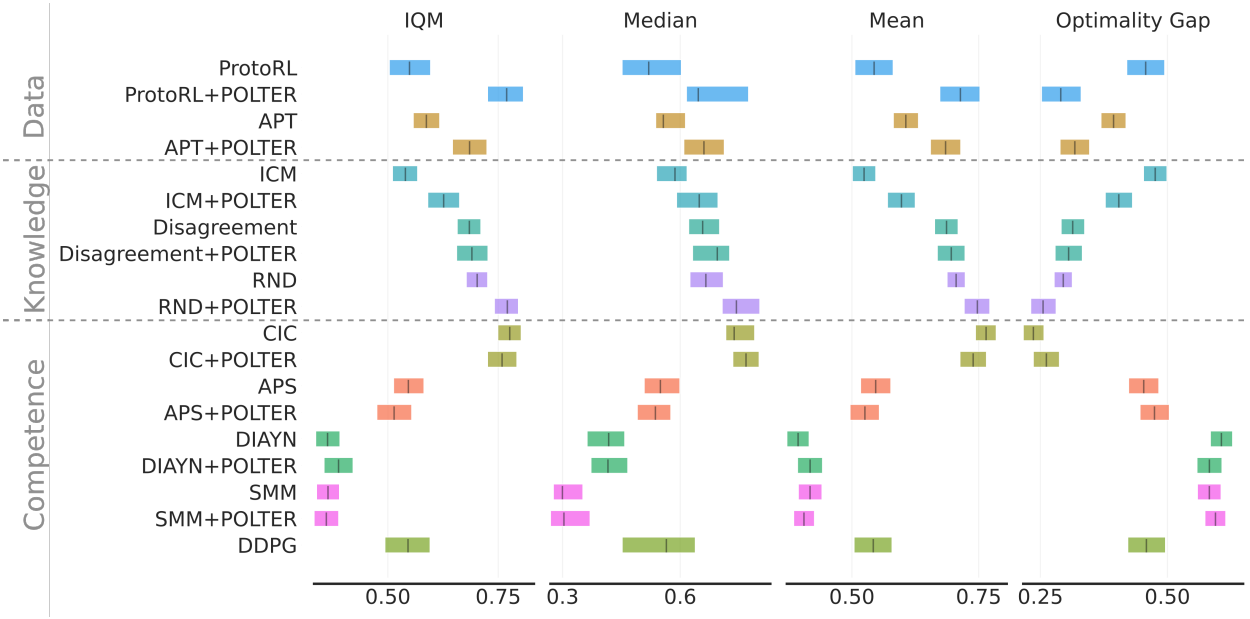


Figure 5: Aggregate statistics of applying POLTER to several URL algorithms after pretraining for 2 M steps. POLTER improves the performance of most algorithms substantially, for ProtoRL even by 40%. Each algorithm is tested on the URLB with its 12 tasks from 3 domains using 10 independent seeds, resulting in 120 runs per algorithm. The error bars indicate the 95% bootstrap confidence intervals. DDPG is the baseline without pretraining.

Evaluation For evaluation, the agent performs pretraining in a reward-free setting for a total of $N_{PT} = 2\,\text{M}$ steps in one of the three domains using one of the described algorithms with or without our POLTER regularization. The performance is then measured as the average return over 10 episodes after finetuning for 100 k steps given external rewards that describe the downstream task. We follow the exact evaluation protocol of the URLB and use DDPG (Lillicrap et al., 2016) on state-based observations. For its hyperparameters and the setup of POLTER, see Appendix E. We follow Agarwal et al. (2021) in their evaluation using the IQM as our main metric[3]. The expert performance is provided by Laskin et al. (2021). Each algorithm is tested on 12 tasks from 3 domains using 10 independent seeds, resulting in 120 runs per algorithm.

5 Results

Q1: How does POLTER affect the performance of different algorithm categories?

We report the finetuning performance of each URL algorithm after 2 M pretraining steps with and without POLTER in Figure 5. With POLTER, we see an increase in performance for data- and knowledge-based algorithms by 19% IQM on average. For the data-based method ProtoRL pretraining with POLTER increases the performance even by 0.22 IQM (40%). The improvement due to POLTER also depends on the exploration capabilities of the URL algorithm as noted in Section 3.2. This can be seen for Disagreement+POLTER, which performs worst of all knowledge-based algorithms and is due to the limited exploration of the state space, as has recently been shown by Yarats et al. (2022). As noted by Laskin et al. (2021), the baseline performances notably differ among themselves, and most competence-based methods are worse than the supervised DDPG baseline without pretraining. The application of POLTER has no noticeable effect on their overall performance. This is due to the similar effect of their skill-averaging and the POLTER ensemble, which both pull the policy to the optimal prior. However, in Section 5, we show that POLTER can be beneficial even for these algorithms. For more detailed results see Appendix F.

[3]Although we used the provided code by Laskin et al. (2021; 2022) for all baselines, our reproduced results slightly differ from their reported performance.

Q2: Does POLTER improve the sample-efficiency for finetuning?

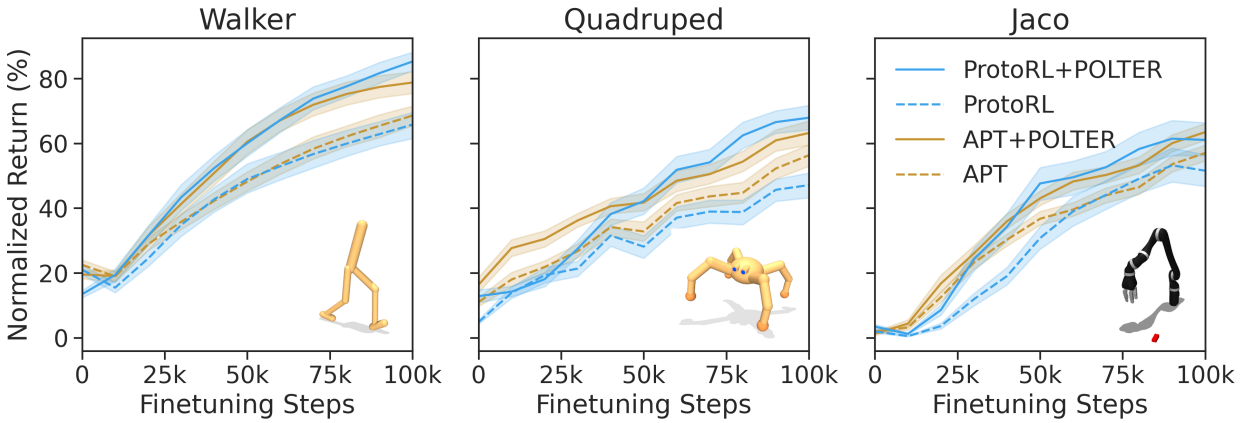


Figure 6: Data-based URL algorithms with (solid) and without POLTER (dashed) regularization after 2 M steps of pretraining. The shaded area indicates the standard error. POLTER speeds up finetuning by $\approx 40\%$ on Walker and Quadruped and $\approx 10\%$ on Jaco.

As described in Section 3, with POLTER, we try to reduce the information cost of finetuning a prior policy on a specific task to improve the sample-efficiency. Because the effect of our method is most pronounced in the data-based algorithm category, in Figure 6 we show the expert normalized return of finetuning APT and ProtoRL with and without POLTER after 2 M steps of pretraining. The improvement in sample-efficiency is clearly visible, especially in the locomotion domains. Here, POLTER achieves a speed-up of $\approx 40\%$ compared to the unregularized URL algorithms. It should be noted that the initial performance directly after pretraining is similar for all variants, and they all begin to improve after the same number of finetuning steps. This implies that exploration during finetuning is not the deciding factor for the performance differences.

Q3: How does the performance vary with an increasing number of pretraining steps?

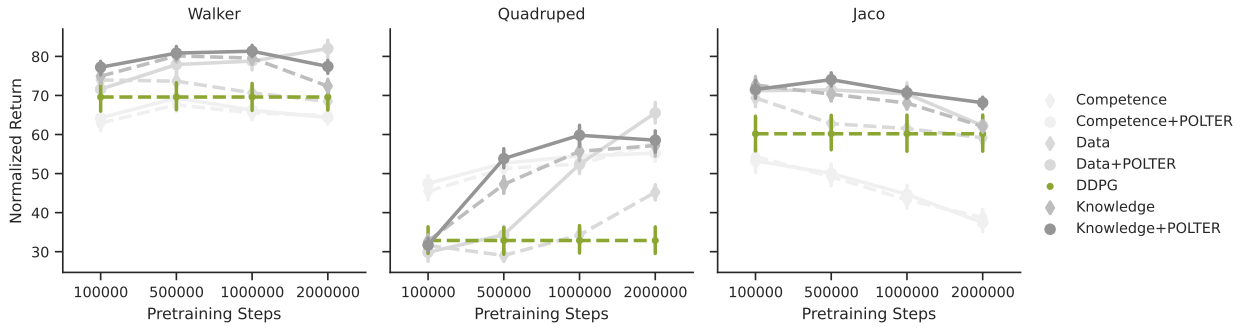


Figure 7: Normalized return per task and algorithm category with (solid) and without (dashed) POLTER regularization after finetuning from different pretraining snapshots. DDPG is the baseline without pretraining.

In their original publication, Laskin et al. (2021) noted that the performance of most URL algorithms decreased with an increasing number of pretraining steps. To evaluate the effect of applying POLTER during pretraining, we observe the performance at different numbers of pretraining frames for the three domains of URLB. In Figure 7, we present our results for knowledge- and data-based methods. Applying POLTER (solid) improves performance at all steps over the baseline (dashed) during pretraining, especially in the locomotion domains.

Jaco is a special case because the arm is in a fixed location, and the return is dominated by the ability to bring the arm into a specific position. Thus, the domain relies much less on deeper exploration compared to

the locomotion environments, and the initial amount of pretraining for 100 k steps is sufficient to improve on the DDPG baseline without pretraining.

Q4: How does the strength of the regularization affect the performance?

A hyperparameter of our method is the strength α of our regularization, see Equation (2). We set this to 1.0 in our preceding evaluations as a natural choice. To study the sensitivity of this hyperparameter, we evaluated POLTER with different settings on a set of representative URL algorithms.

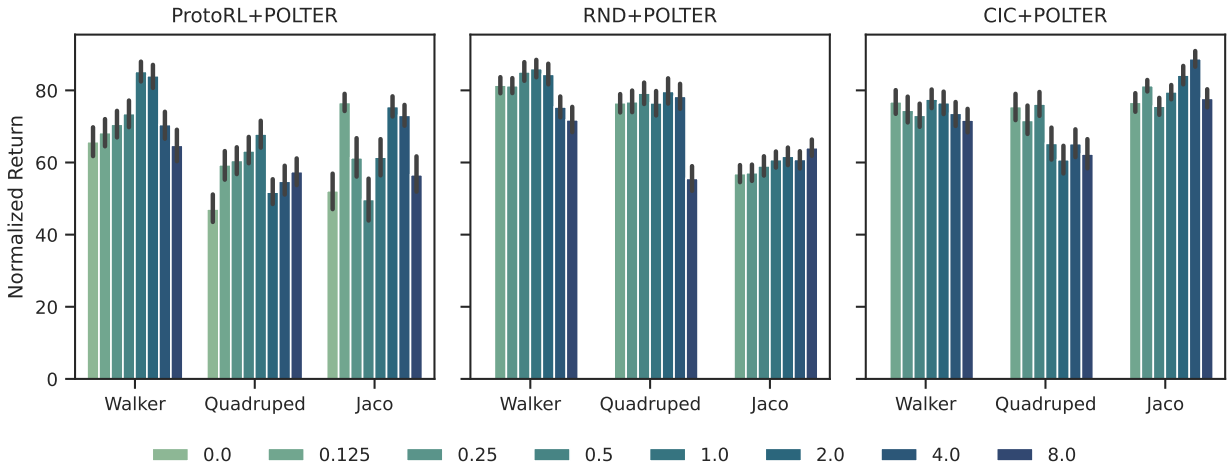


Figure 8: Average normalized return of ProtoRL+POLTER, RND+POLTER and CIC+POLTER after 2 M steps of pretraining and 100 k finetuning for different values of the regularization strength α.

In Figure 8, we present the performance of the POLTER regularized variants of ProtoRL, RND and CIC over different values of α. The results show a robust performance of POLTER with its default $\alpha = 1.0$, i.e., POLTER performs better or is on par compared to not using POLTER ($\alpha = 0.0$). On average, $\alpha = 1.0$ seems to be the best choice. Nevertheless, the performance of POLTER can be further improved if it is tuned depending on the task and algorithm. In the Jaco domain, a higher regularization helps RND but might be detrimental for ProtoRL. We hypothesize that this is due to increasingly extreme pretraining policies of RND that are dampened via POLTER, which is helpful for the tasks in this domain. For CIC, the regularization of POLTER has a minor effect or even degrades the performance in the locomotion domains. As Laskin et al. (2022) noted, locomotion requires a high-entropy pretraining policy that explores the environment well. POLTER is indifferent to the skill vector used by the CIC policy in each episode. So the ensemble contains different skill vectors that are likely to suggest different actions, which partly cancel each other and reduce the exploration.

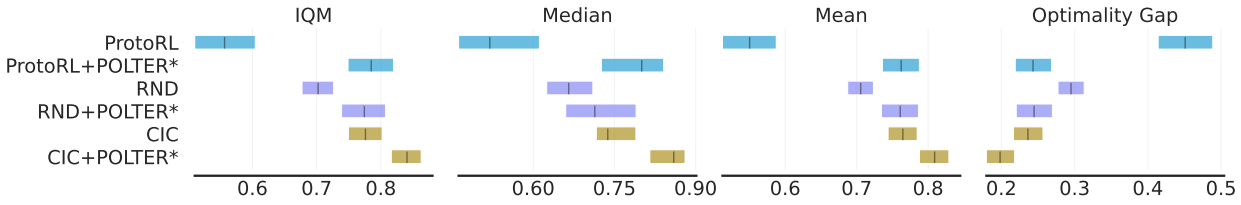


Figure 9: Aggregate statistics of three representative algorithms from each URL category after 2 M steps of pretraining and 100 k finetuning with tuned regularization strength α in Equation (2).

However, a stronger regularization up to a certain value significantly improves the performance on Jaco, as it does not require much exploration to perform well. This demonstrates the trade-off between a good prior for

finetuning and an attenuated exploration if the regularization is too strong. It also shows that POLTER should be tuned to the domain.

If we apply the insights from Figure 8, we can improve the performance of the three evaluated algorithms even further by tuning α to the locomotion and manipulation domains. For the results in Figure 9, we set α for ProtoRL to 1.0/2.0 (locomotion/manipulation), for RND to 2.0/8.0 and for CIC to 0.0/4.0. We call the tuned variants ProtoRL+POLTER*, RND+POLTER* and CIC+POLTER*. Note that our regularization complements the discriminator loss of CIC, which is set to 0.9/0.0 on the two domain categories. These optimized settings lead to a new state-of-the-art in the URLB of 0.84 IQM. We note that previous state-of-the-art methods such as CIC are also tuned based on the domain of the task and that this experiment allows for a fair comparison.

6 Limitations and Future Work

Although we evaluated our method on state-based MDPs, it can be readily applied to pixel-based environments. In this work, we focused our experiments on DDPG to compare our performance on the URLB. This, however, limits our scope to continuous control problems. Therefore, extending URL to environments with discrete action spaces is a possible next step. Moreover, a deeper theoretical understanding of the pretraining processes and the connection to the optimal prior policy would enable more sophisticated mixture policies and a better approximation of the optimal prior in the future. POLTER is not adapted to the category of the URL algorithm it is applied to. So its performance with competence-based URL algorithms leaves room for improvement by taking the observed explicit skills into account.

7 Conclusion

In this work, we introduced POLTER (Policy Trajectory Ensemble Regularization) – a general method to improve the performance of URL algorithms. URL is a method to increase the sample-efficiency of RL algorithms on a set of tasks by pretraining the policy with an intrinsic reward in the task's domain. Our regularization pulls the pretraining policy closer to an ensemble of pretraining policies seen during pretraining that correspond to a set of explicit or implicit skills. We demonstrated the effect of POLTER in the PointMass and Pendulum domain and extensively evaluated baseline algorithms regularized with POLTER on URLB and established its effectiveness in our main experiments. For data- and knowledge-based URL methods, we improved performance on average by 19% and up to 40% (IQM). We argue that our method is a suitable regularization for those algorithms, in contrast to competence-based methods where the effect of POLTER is highly algorithm-dependent. Finally, we showed that the regularization strength of POLTER can be tuned to the domain and algorithm at hand for further performance improvements, achieving a new state-of-the-art performance with CIC+POLTER*. In future work, this manual tuning could be automated utilizing AutoRL (Parker-Holder et al., 2022). With POLTER's easy implementation and negligible computational requirements, we hope it finds its way into more URL algorithms and spurs further research on how we can learn general priors for arbitrary tasks.

References

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236. URL `http://www.nature.com/articles/nature14236`.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL `https://www.science.org/doi/abs/10.1126/science.aar6404`.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv:1910.07113 [cs, stat]*, October 2019. URL http://arxiv.org/abs/1910.07113. arXiv: 1910.07113.

Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, December 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-2939-8. URL https://www.nature.com/articles/s41586-020-2939-8.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, June 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03544-w. URL https://www.nature.com/articles/s41586-021-03544-w. Number: 7862 Publisher: Nature Publishing Group.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414–419, February 2022. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-021-04301-9. URL https://www.nature.com/articles/s41586-021-04301-9.

Aravind Srinivas and Pieter Abbeel. Unsupervised Learning for Reinforcement Learning, 2021. URL https://icml.cc/media/icml-2021/Slides/10843_QHaHBNU.pdf.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised Reinforcement Learning Benchmark. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/091d584fced301b442654dd8c23b3fc9-Abstract-round2.html.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The Information Geometry of Unsupervised Reinforcement Learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=3wU2UX0voE.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6. URL https://www.worldcat.org/oclc/37293240.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex X. Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *CoRR*, abs/1812.00568, 2018. URL http://arxiv.org/abs/1812.00568.

David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 2020. URL http://proceedings.mlr.press/v119/laskin20a.html.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=uCQfPZwRaUu.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9870–9879. PMLR, 2021. URL http://proceedings.mlr.press/v139/stooke21a.html.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V. Hafner. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Trans. Evol. Computat.*, 11(2):265–286, April 2007. ISSN 1089-778X. doi: 10.1109/TEVC.2006.890271. URL http://ieeexplore.ieee.org/document/4141061/.

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-Free Exploration for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4870–4879. PMLR, 2020. URL http://proceedings.mlr.press/v119/jin20d.html.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, 2017. URL http://proceedings.mlr.press/v70/pathak17a.html.

Elad Hazan, Sham M. Kakade, Karan Singh, and Abby Van Soest. Provably Efficient Maximum Entropy Exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2681–2691. PMLR, 2019. URL http://proceedings.mlr.press/v97/hazan19a.html.

Shakir Mohamed and Danilo Jimenez Rezende. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2125–2133, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/e00406144c1e7e35240afed70f34166a-Abstract.html.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational Intrinsic Control. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Skc-Fo4Yg.

Yifu Yuan, Jianye Hao, Fei Ni, Yao Mu, Yan Zheng, Yujing Hu, Jinyi Liu, Yingfeng Chen, and Changjie Fan. EUCLID: towards efficient unsupervised reinforcement learning with multi-choice dynamics model. *CoRR*, abs/2210.00498, 2022. doi: 10.48550/arXiv.2210.00498. URL https://doi.org/10.48550/arXiv.2210.00498.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. CIC: Contrastive Intrinsic Control for Unsupervised Skill Discovery. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, December 2022.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv:1801.00690 [cs]*, January 2018. URL http://arxiv.org/abs/1801.00690. arXiv: 1801.00690.

Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Piotr Trochim, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm_control: Software and Tasks for Continuous Control. *Software Impacts*, 6:100022, November 2020. ISSN 26659638. doi: 10.1016/j.simpa.2020.100022. URL http://arxiv.org/abs/2006.12983. arXiv: 2006.12983.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-Supervised Exploration via Disagreement. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5062–5071. PMLR, 2019. URL http://proceedings.mlr.press/v97/pathak19a.html.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R. Devon Hjelm, Philip Bachman, and Aaron C. Courville. Pretraining Representations for Data-Efficient Reinforcement Learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12686–12699, 2021b. URL https://proceedings.neurips.cc/paper/2021/hash/69eba34671b3ef1ef38ee85caae6b2a1-Abstract.html.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement Learning with Prototypical Representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11920–11931. PMLR, 2021. URL http://proceedings.mlr.press/v139/yarats21a.html.

Hao Liu and Pieter Abbeel. APS: Active Pretraining with Successor Features. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6736–6747, 2021. URL http://proceedings.mlr.press/v139/liu21b.html.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient Exploration via State Marginal Matching. *arXiv:1906.05274 [cs, stat]*, February 2020. URL http://arxiv.org/abs/1906.05274. Task-Agnostic Reinforcement Learning Workshop at ICLR 2019.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Andrew Zhao, Matthieu Gaetan Lin, Yangguang Li, Yong-Jin Liu, and Gao Huang. A mixture of surprises for unsupervised reinforcement learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, December 2022.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't Change the Algorithm, Change the Data: Exploratory Data for Offline Reinforcement Learning. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. URL https://openreview.net/forum?id=Su-zh4a41Z5.

Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, Frank Hutter, and Marius Lindauer. Automated Reinforcement Learning (AutoRL): A Survey and Open Problems. *Journal of Artificial Intelligence Research*, 2022. To appear. Preprint at https://arxiv.org/abs/2201.03916.

A Extended Demonstration on PointMass

We evaluate the effect of our regularization on RND (Burda et al., 2019), a well-known knowledge-based URL algorithm, in the simplistic PointMass environment (Tassa et al., 2018; 2020). In this environment, the agent has to apply a force to a mass (green in Figure 10b) in a 2D plane to reach a target (red) and observes its position and speed. We pretrain a DDPG agent using RND with and without POLTER for 2 M steps and evaluate the policies every 100 k steps, in accordance with the URL benchmark evaluation protocol. Because PointMass is a much simpler environment than the ones contained in URLB and to gain a better resolution for the start of the pretraining, we repeat the experiment with 200 k pretraining and 100 k finetuning steps. We use 10 seeds.

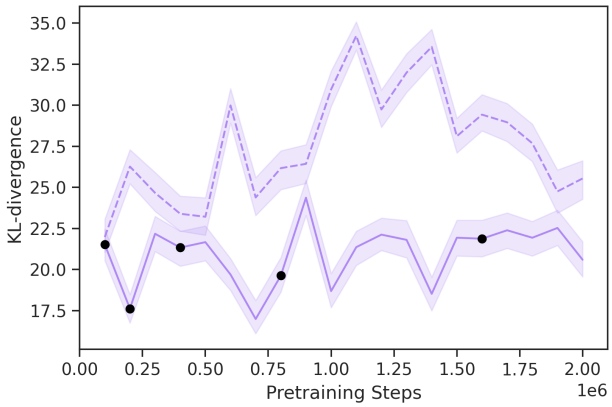
In our whitebox benchmark PointMass, we can train the optimal prior policy and compute the KL-divergence between the optimal prior policy and the current pretraining policy $D_{\mathrm{KL}}(\pi_T^* \parallel \pi)$. Comparing the KL-divergence during pretraining with and without POLTER in Figure 10a shows that with our method, the KL-divergence to the optimal prior policy π_T^* is generally lower, which is also apparent for the short pretraining (Figure 10c). POLTER keeps the KL-divergence at bay, whereas RND without POLTER diverges much more, which is consistent with prior work showing that URL tends to diverge with too many pretraining steps.

A.1 Effect of POLTER on Finetuning

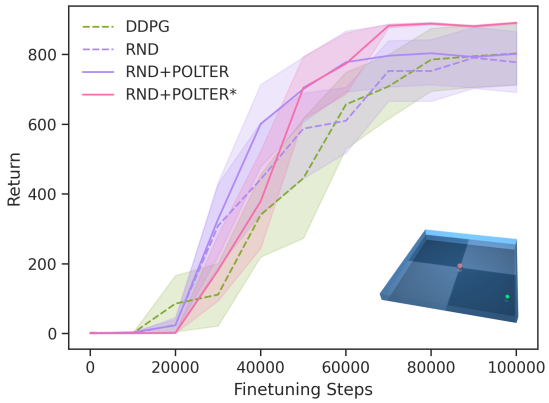
The effect of this improved prior is also apparent during finetuning in Figure 10b. Here, we see that using an ensemble is a good proxy for the optimal prior due to the similar sample-efficiency. In addition, we observe a speed-up of approximately 25 % (2 M pretraining steps) and 12.5 % (200 k pretraining steps) when using RND+POLTER in reaching the final performance of DDPG.

A.2 Visualization of State Distributions

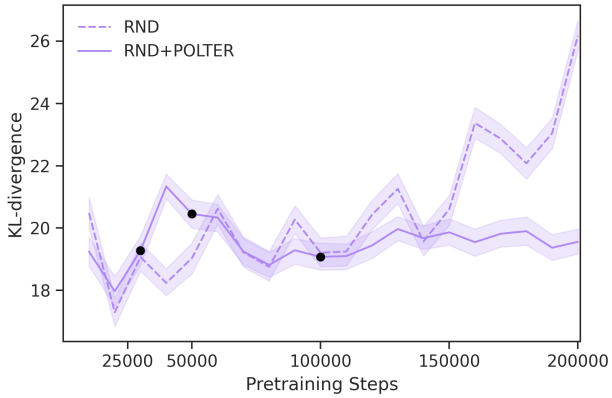
In Figure 11a, we can see the state distribution changes during pretraining with and without POLTER. With POLTER, the state space coverage is less, and the trajectories seem more ordered. RND without POLTER also seems to visit the edges often at the end. When using the POLTER regularization, we can see that each pretraining checkpoint is visiting different states, as indicated by the visibility of the previous checkpoint's state visitations. When not using POLTER, we can see that the visitations overlap. Figures 11b and 11c show the discretized position and speed the agent explores throughout pretraining. Especially the discretized speed (Figure 11c) demonstrates the tradeoff between dampening the exploration with POLTER and finding better prior policies because with POLTER, fewer states are frequented, and the states are less extreme.



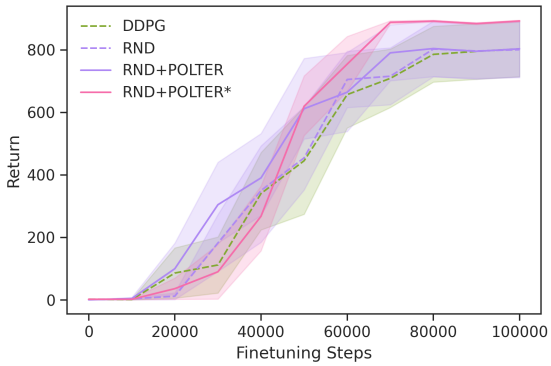
(a) Pretraining (2 M steps)



(b) Finetuning (100 k steps)



(c) Pretraining (200 k steps)



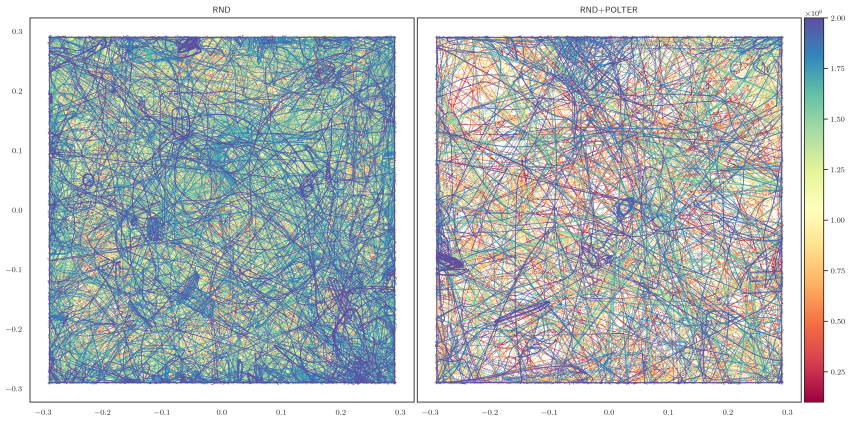
(d) Finetuning (100 k steps)

Figure 10: **(a)**, **(c)** Average KL-divergence of RND (dashed) and RND+POLTER (solid) between the pretraining policy $\pi(s_0)$ and the optimal pretraining policy $\pi^*_T(s_0)$ in the PointMass domain during reward-free pretraining. **(a)** is trained for 2 M steps, and **(c)** is trained for 200 k. Each policy is evaluated every 100 k steps on 20 initial states over 10 seeds. The black dots indicate the steps a snapshot is added to the ensemble. **(b)**, **(d)** Return during finetuning after 2 M and 200 k pretraining steps, where the target is placed at a fixed random position for each of the 10 seeds. We also provide a DDPG baseline without pretraining and RND+POLTER* using the optimal policy instead of the ensemble. The shaded area indicates the standard error.

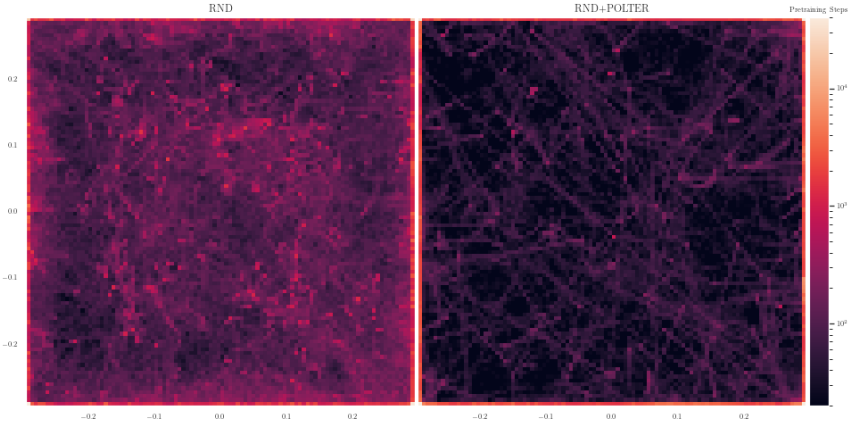
B Extended Experiments on Pendulum

In this experiment, we shed further light on the properties of POLTER and its effect under different finetuning tasks. For example, in the PendulumPropeller task, the goal is to maximize the angular velocity of the pole. However, this requires a rather extreme policy, whereas, in the PendulumSwingUp task, the agent has to balance the pole and keep it upright.

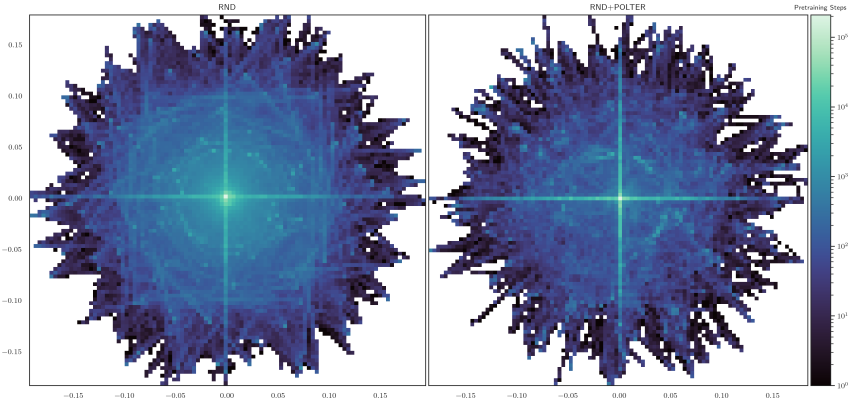
The SwingUp task demonstrates that POLTER can consistently improve the performance of the baseline URL agent. However, for the Propeller task, the results are mixed. For ProtoRL, POLTER again shows a great improvement over the baseline. This supports the performance that we observed in the main experiments. But for CIC and RND, POLTER decreases performance. The extreme policy of achieving a high angular velocity is further away from the average, so applying POLTER has a detrimental effect. Nevertheless, note that regardless of this failure case, on average, the tasks will lie closer to the average state distribution, and thus, POLTER will increase the performance.



(a) State distribution $\rho(s)$ of several checkpoints.



(b) Discretized state histogram of the positions. Looking closely, we can see that the edges of the 2D plane are very often visited. This is due to policies constantly applying the same force and thus reaching the edge of the 2D plane.



(c) Discretized state histogram of the speeds. Note that the prominent horizontal and vertical lines result from moving at the edges and being stuck at the corners of the 2D plane.

Figure 11: State distribution and histogram of RND (always left column) and RND+POLTER (right column) during pretraining on the PointMass environment.

C State-Visitation Entropy

To gain further insights into POLTER, we conduct an experiment following Hazan et al. (2019). We discretize the state space of the Walker environment and compute the state-visitation entropy during reward-free

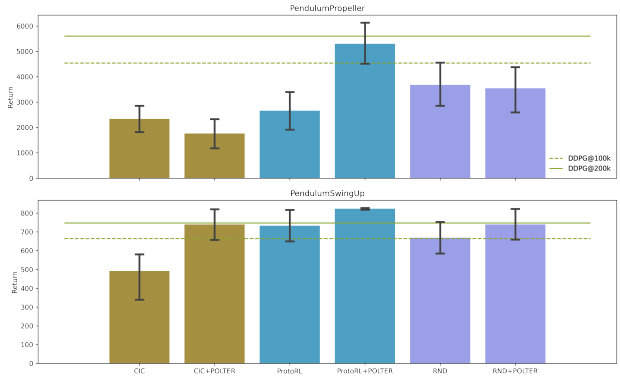


Figure 12: Average return over 10 seeds on two tasks in the Pendulum domain with different characteristics. Algorithms from all three URL categories are compared to a DDPG baseline that is trained for either 100 k or 200 k steps without pretraining.

pretraining. In Table 1, we see that the entropy of the distribution of POLTER regularized algorithms is lower than that of their counterpart. This effect is most pronounced in data-based algorithms, such as ProtoRL and APT, where the performance is also improved the most (see Table 3).

Table 1: State-visitation entropy of the evaluated URL algorithm categories in the Walker domain during pretraining. Averaged over 10 seeds with 50 k states each at pretraining steps 100 k, 500 k, 1 M and 2 M.

POLTER	Data	Knowledge	Competence
✘	0.2772 ± 0.0188	0.2863 ± 0.0036	0.2540 ± 0.0429
✔	0.2545 ± 0.0493	0.2848 ± 0.0054	0.2511 ± 0.0422

Knowledge-based algorithms also benefit from the regularization but have a slightly reduced entropy. Because competence-based algorithms already average over a set of skills found during pretraining, the effect of POLTER is the smallest.

These results imply that POLTER does not lead to a better state-space exploration. Instead, its performance gains are the result of an improved prior as indicated by the reduced KL-divergence between the policy and the optimal pretraining policy on PointMass (Section 3.3). This experiment showed that a good exploration of the state-space is required but not sufficient to achieve good performance with URL algorithms.

D Environments in the Unsupervised Reinforcement Learning Benchmark

The Unsupervised Reinforcement Learning Benchmark Laskin et al. (2021) contains three domains with the topics of locomotion and manipulation. **Walker**, **Quadruped** and **Jaco**, are used to explore the effects of different URL algorithms. It has a specific training and evaluation protocol which we also follow in this work. The **Walker** domain contains a planar walker constrained to a 2D vertical plane, with an 18-dimensional observation space and a 6-dimensional action space with three actuators for each leg. The associated tasks are *stand*, *walk*, *run* and *flip*. The walker domain provides a challenging start for the agent since it needs to learn balancing and locomotion skills to be able to adapt to the given tasks. The next domain is **Quadruped**, which expands to the 3D space. It has a much larger state space of 56 dimensions and a 12-dimensional action space with three actuators for each leg. The tasks in this environment are *stand*, *walk*, *jump* and *run*. The last environment used is the **Jaco Arm**, which is a robotic arm with 6-DOF and a three-finger gripper. This domain is very different from the other two, as its setting is manipulation and not locomotion. The tasks are *Reach top left*, *Reach top right*, *Reach bottom left* and *Reach bottom right*.

E Hyperparameters and Resources

POLTER Hyperparameters During pretraining, we construct the mixture ensemble policy $\tilde{\pi}$ with $k = 7$ members at specific time steps $\mathcal{T}_E$. For adding each member we choose the ensemble snapshot time steps $\mathcal{T}_E = \{25\,\text{k}, 50\,\text{k}, 100\,\text{k}, 200\,\text{k}, 400\,\text{k}, 800\,\text{k}, 1.6\,\text{M}\}$. The steps were chosen according to initial experiments of applying RND in the Quadruped domain, where there are large changes of the intrinsic reward at the beginning, which become progressively smaller over time. We set the regularization strength $\alpha = 1$ and use the same hyperparameters for each of the three domains unless specified otherwise.

Baseline Hyperparameters The hyperparameters for our baseline algorithms follow Laskin et al. (2021) and Laskin et al. (2022). The hyperparameters for the DDPG baseline agent are described in Table 2.

Table 2: Hyperparameters for the DDPG algorithm.

Hyperparameter	Value
Replay buffer capacity	1×10^6
Action repeat	1
Seed frames	4000
n-step returns	3
Batch size	1024
Discount factor γ	0.99
Optimizer	Adam
Learning rate	1×10^{-4}
Agent update frequency	2
Critic target EMA rate	0.01
Feature size	1024
Hidden size	1024
Exploration noise std clip	0.3
Exploration noise std value	0.2
Pretraining frames	2×10^6
Finetuning frames	1×10^5

Compute Resources All experiments were run on our internal compute cluster on NVIDIA RTX 1080 Ti and NVIDIA RTX 2080 Ti GPUs and had 64GB of RAM and 10 CPU cores. In total, we trained over 12 000 models and performed $\approx 3\,500\,200\,000$ environment steps.

F Detailed Results on Unsupervised Reinforcement Learning Benchmark

This section provides additional results for our experiments on Unsupervised Reinforcement Learning Benchmark. In the supplementary, we provide the raw scores. The statistics comparing URL algorithms with and without POLTER aggregated for finetuning on 12 tasks across 10 seeds can be found in Table 3. In addition we show aggregate statistics of the absolute improvement in expert performance in Figure 13 and the performance profiles (Agarwal et al., 2021) per URL algorithm category in Figure 14. As before, we see a large improvement for data- and knowledge-based algorithms and a small or negative for competence-based algorithms. The improvement sometimes varies strongly across seeds and tasks. Also, we show the normalized return after finetuning from different pretraining snapshots for each domain and URL category in Figure 15. In the Jaco domain, URL algorithms with and without POLTER mostly deteriorate with an increasing number of pretraining steps. Each category shows a different trend in each domain. Interestingly, the competence-based algorithms SMM and DIAYN fail during pretraining in the Jaco domain. In Figure 16 we see the normalized return over finetuning steps. POLTER is mostly on par or speeds up compared to the URL algorithm without POLTER. In total, most algorithms do not converge yet after 100 k steps.

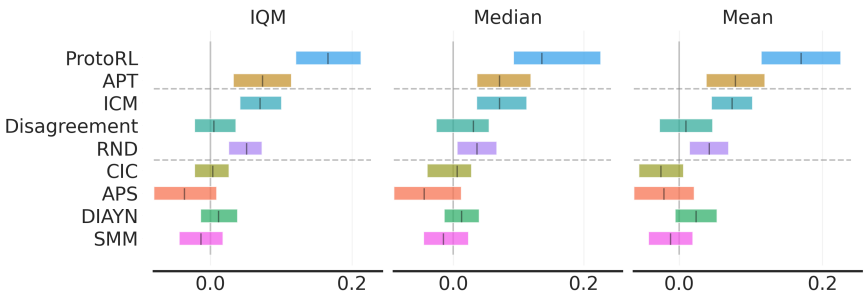


Figure 13: Aggregate statistics of the absolute improvement with POLTER per URL category.

Table 3: Raw aggregate statistics following Agarwal et al. (2021) of evaluated URL algorithms with and without POLTER regularization. The results marked with POLTER* were obtained by tuning the regularization strength to the task domain of locomotion (Walker and Quadruped) and manipulation (Jaco).

Algorithm	IQM ↑	Mean ↑	Median ↑	Optimality Gap ↓	POLTER IQM Improvement
ProtoRL	0.56	0.55	0.52	0.45	
ProtoRL+POLTER	0.77	0.71	0.65	0.29	+40%
ProtoRL+POLTER*	0.79	0.76	0.80	0.24	+41%
APT	0.59	0.61	0.56	0.39	
APT+POLTER	0.69	0.68	0.66	0.32	+17%
RND	0.70	0.71	0.67	0.30	
RND+POLTER	0.77	0.75	0.74	0.26	+10%
RND+POLTER*	0.77	0.76	0.71	0.24	+10%
ICM	0.54	0.52	0.59	0.48	
ICM+POLTER	0.63	0.60	0.65	0.40	+17%
Disagreement	0.68	0.69	0.66	0.31	
Disagreement+POLTER	0.69	0.70	0.69	0.31	+1%
CIC	0.78	0.76	0.74	0.24	
CIC+POLTER	0.76	0.74	0.77	0.26	-2%
CIC+POLTER*	0.84	0.81	0.86	0.20	+7%
DIAYN	0.36	0.39	0.42	0.61	
DIAYN+POLTER	0.39	0.42	0.42	0.58	+8%
SMM	0.36	0.42	0.30	0.58	
SMM+POLTER	0.36	0.41	0.30	0.59	± 0%
APS	0.56	0.58	0.55	0.42	
APS+POLTER	0.52	0.53	0.54	0.47	-7%
DDPG	0.55	0.54	0.56	0.46	

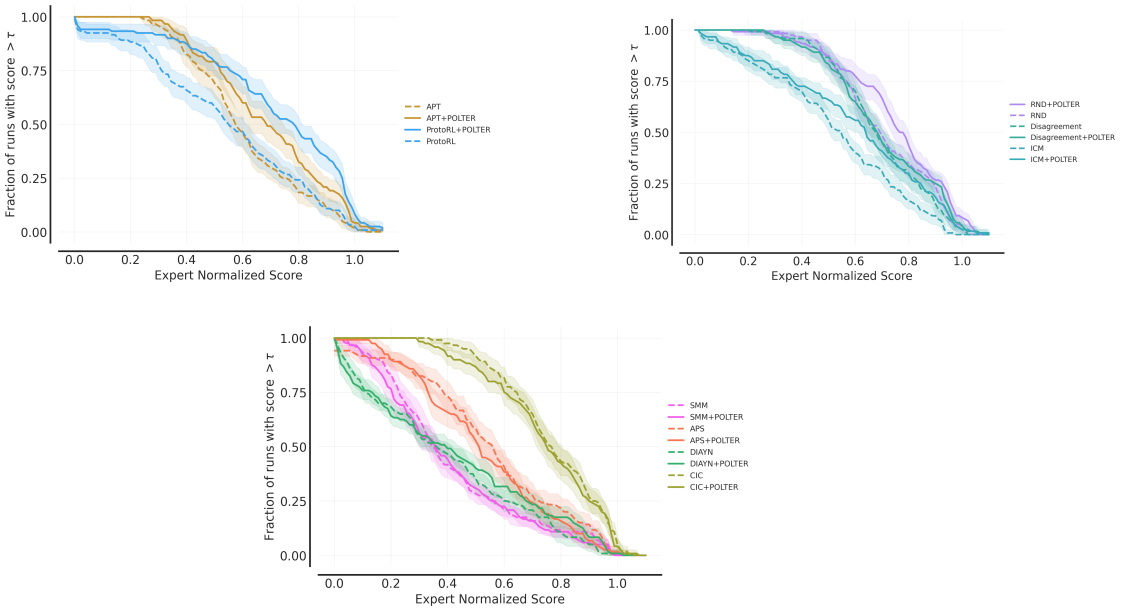


Figure 14: Performance profiles after finetuning of the different algorithms averaged over 10 seeds where the shaded region indicates the standard error. Variants without POLTER are dashed.

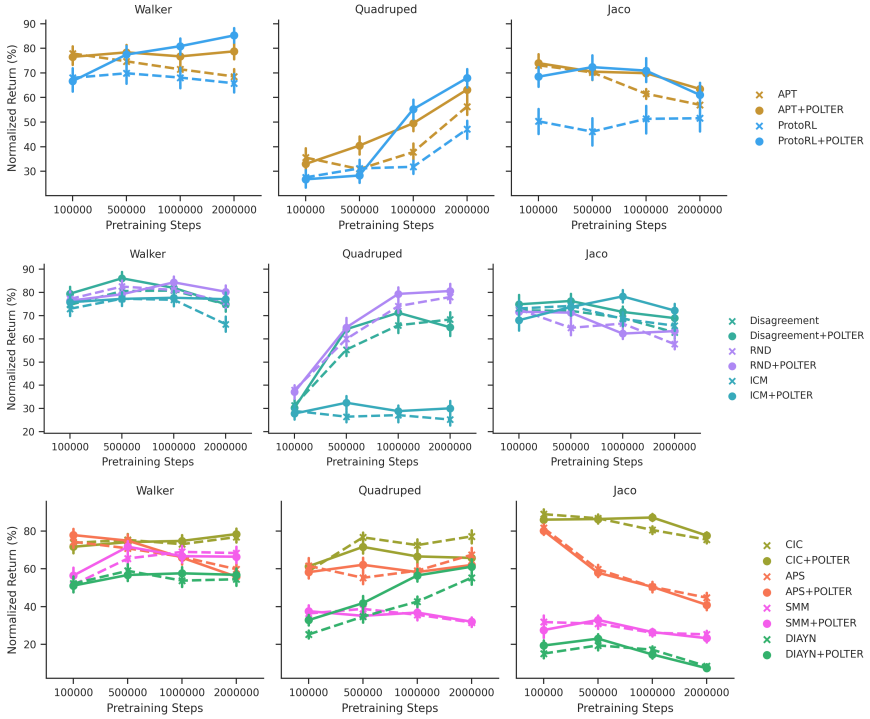


Figure 15: Finetuning from different pretraining snapshots of data-, knowledge- and competence-based algorithms. The error bars indicate the standard error of the mean.

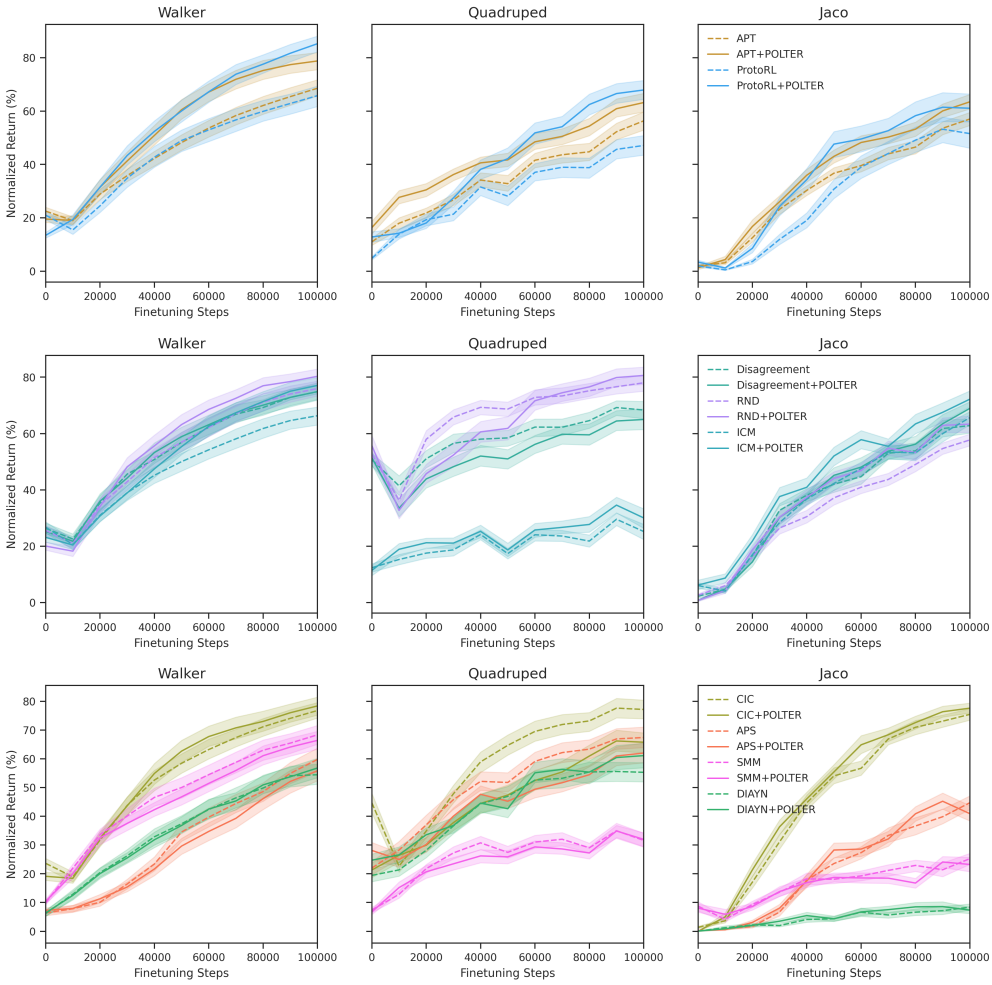


Figure 16: Finetuning curves of data-, knowledge- and competence-based algorithms after pretraining for 2 M steps. The shaded area indicates the standard error.