arXiv:2110.15191v1 [cs.LG] 28 Oct 2021

URLB: Unsupervised Reinforcement Learning Benchmark

Michael Laskin[*] UC Berkeley mlaskin@berkeley.edu
Denis Yarats[*] NYU, FAIR denisyarats@cs.nyu.edu
Hao Liu UC Berkeley
Kimin Lee UC Berkeley
Albert Zhan UC Berkeley
Kevin Lu UC Berkeley
Catherine Cang UC Berkeley
Lerrel Pinto NYU
Pieter Abbeel UC Berkeley, Covariant

Abstract

Deep Reinforcement Learning (RL) has emerged as a powerful paradigm to solve a range of complex yet specific control tasks. Yet training generalist agents that can quickly adapt to new tasks remains an outstanding challenge. Recent advances in unsupervised RL have shown that pre-training RL agents with self-supervised intrinsic rewards can result in efficient adaptation. However, these algorithms have been hard to compare and develop due to the lack of a unified benchmark. To this end, we introduce the Unsupervised Reinforcement Learning Benchmark (URLB). URLB consists of two phases: reward-free pre-training and downstream task adaptation with extrinsic rewards. Building on the DeepMind Control Suite, we provide twelve continuous control tasks from three domains for evaluation and open-source code for eight leading unsupervised RL methods. We find that the implemented baselines make progress but are not able to solve URLB and propose directions for future research. Code for the benchmark and implemented baselines can be accessed at https://github.com/rll-research/url_benchmark.

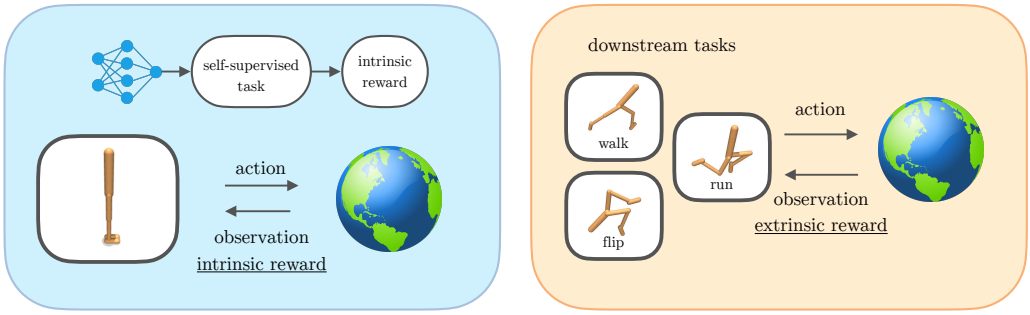
1 Introduction

Deep Reinforcement Learning (RL) has been at the source of a number of breakthroughs in autonomous control over the last five years. RL algorithms have been used to train agents to play Atari video games directly from pixels [44, 45], learn robotic locomotion [52–54] and manipulation [2] policies from raw sensory input, master the game of Go [58, 59], and play large-scale multiplayer video games [6, 65]. While these results were significant advances in autonomous decision making, a deeper look reveals a fundamental limitation. The above algorithms produced agents capable of only solving the single task they were trained to solve. As a result, current RL approaches produce brittle policies with poor generalization capabilities [16], which limits their applicability to many problems of interest [23]. It is therefore important to move beyond today's powerful but narrow RL systems toward generalist systems capable of quickly adapting to new downstream tasks.

In contrast, in the fields of Computer Vision (CV) and Natural Language Processing (NLP), large-scale unsupervised pre-training has enabled sample-efficient few-shot adaptation. In NLP, unsupervised sequential modeling has produced powerful few-shot learners [8, 17, 50]. In CV, unsupervised representation learning techniques such as contrastive learning have produced algorithms that are dramatically more label-efficient than their supervised counterparts [14, 31, 32, 25] and more capable of adapting to a host of downstream supervised tasks such as classification, segmentation, and object detection. While these advances in unsupervised learning have also benefited RL in terms of learning

[*]equal contribution, order determined by coin flip.

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.



Part 1: Unsupervised Pre-training with Intrinsic Rewards

Part 2: Supervised Finetuning to Downstream Tasks

Figure 1: Unlike supervised RL which requires reward interaction at every step, unsupervised RL has two phases: (i) reward-free pre-training and (ii) fine-tuning to an extrinsic reward. During phase (i) an agent explores the environment through reward-free interaction with the environment. The quality of exploration depends on the intrinsic reward that the agent sets for itself. During phase (ii) the quality of pre-training is evaluated by its adaptation efficiency to a downstream task.

efficiently from images [37, 38, 55, 62, 69] as well as introducing new architectures for RL [13, 35], the resulting agents have remained narrow since they still optimize a single extrinsic reward as before.

Fully unsupervised training of RL algorithms requires not only learning self-supervised representations but also learning policies without access to extrinsic rewards. Recently, unsupervised RL algorithms have begun to show progress toward more generalist systems by training policies without extrinsic rewards. Exploration with self-supervised prediction has enabled agents to explore video games from pixels [48, 49], mutual information-based approaches have demonstrated self-supervised skill discovery and generalization to downstream tasks in continuous control domains [21, 30, 43, 57], and maximal entropy RL has yielded policies capable of diverse exploration [42, 56, 68]. However, comparing and developing new algorithms has been challenging due to a lack of a unified evaluation benchmark. Reward-free RL algorithms often use different optimization schemes, different tasks for evaluation, and have different evaluation procedures. Additionally, unlike more mature supervised RL algorithms [27, 33, 54], there does not exist a unified codebase for unsupervised RL that can be used to develop new methods quickly.

To make benchmarking and developing new unsupervised RL approaches easier, we introduce the Unsupervised Reinforcement Learning Benchmark (URLB). Built on top of the widely adopted DeepMind Control Suite [64], URLB provides a suite of domains of varying difficulty for unsupervised pre-training with diverse downstream evaluation tasks. URLB standardizes evaluation of unsupervised RL algorithms by defining fixed pre-training and fine-tuning procedures across all baselines. Perhaps most importantly, we open-source code for URLB environments as well as 8 leading baselines that represent the main approaches taken towards unsupervised pre-training in RL to date. Unlike prior code releases for unsupervised RL, URLB uses the same exact optimization algorithm for each baseline which enables transparent benchmarking and lowers the barrier to entry for developing new algorithms. We summarize the main contributions of this paper below:

1. We introduce URLB, a new benchmark for evaluating unsupervised RL algorithms, which consists of three domains and twelve continuous control tasks of varying difficulty to evaluate the adaptation efficiency of unsupervised RL algorithms.
2. We open-source a unified codebase for eight leading unsupervised RL algorithms. Each algorithm is trained with the same optimization backbone for fairness of comparison.
3. We find that while the implemented baselines make progress on the proposed benchmark, no existing unsupervised RL algorithm can solve URLB, and consequently identify promising research directions to progress unsupervised RL.

The benchmark environments, algorithmic baselines, and pre-training and evaluation scripts are available at https://github.com/rll-research/url_benchmark. We believe that URLB will make the development of unsupervised RL agents easier and more transparent by providing a unified

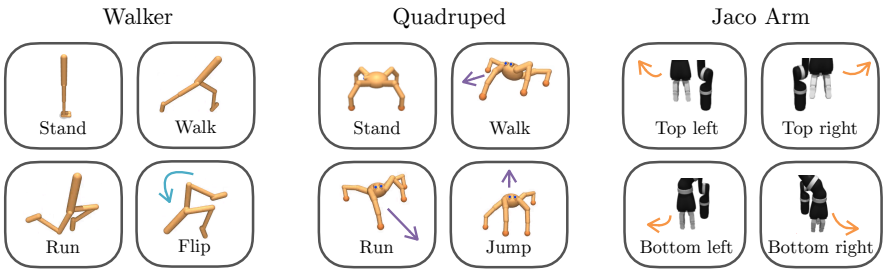


Figure 2: The three domains (walker, quadruped, jaco arm) and twelve downstream tasks considered in URLB. The environments include tasks of varying complexity and require an agent pre-trained on a given domain to adapt efficiently to the downstream tasks within that domain.

set of evaluation environments, systematic procedures for pre-training and evaluation, and algorithmic baselines that share the same optimization backbone.

2 Preliminaries and Notation

Markov Decision Process: We consider the typical Reinforcement Learning setting where an agent's interaction with the environment is modeled through a Markov Decision Process (MDP) [63]. In this work, we benchmark unsupervised RL algorithms in both fully observable MDPs where the agent learns from coordinate state as well as partially observable MDPs (POMDPs) where the agent learns from partially observable image observations. For simplicity we refer to both image and state-based observations as $\mathbf{o}$. At every timestep t, the agent sees an observation $\mathbf{o}_t$ and selects an action $\mathbf{a}_t$ based on its policy $\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{o}_t)$. The agent then sees the next observation $\mathbf{o}_{t+1}$ and an extrinsic reward r_t^{ext} provided by the environment (supervised RL) or an intrinsic reward r_t^{int} defined through a self-supervised objective (unsupervised RL). In this work, we pre-train agents with intrinsic rewards r_t^{ext} and fine-tune them to downstream tasks with extrinsic rewards r_t^{ext}. Some algorithms considered in this work condition the agent on a learned task vector which we denote as $\mathbf{w}$.

Learning from pixels vs states: We benchmark unsupervised RL where environment observations $\mathbf{o}_t$ can be either proprioceptive states or RGB images. When learning from pixels, rather than defining the self-supervised task directly as a function of image observations, it is usually more convenient to first embed the image and compute the intrinsic reward as a function of these lower dimensional features [10, 42, 43, 48]. We therefore define an embedding as $\mathbf{z}_t = f_\xi(\mathbf{o}_t)$ where $f_\xi(\mathbf{o}_t)$ is an encoder function. We employ different encoder f_ξ architectures depending on whether the algorithm receives pixel or state-based input. For pixel-based inputs we use the convolutional encoder architecture from SAC-AE [66], while for state-based inputs we use the identity function by default unless the unsupervised RL algorithm explicitly specifies a different encoding. The intrinsic reward r_t^{int} can be a function of any and all $(\mathbf{z}_t, \mathbf{a}_t, \mathbf{w}_t)$ depending on the algorithm. Finally, note that the encoder f_ξ may or may not be shared with components of the base RL algorithm such as the actor and critic.

3 URLB: Evaluation and Environments

3.1 Standardized of Pre-training and Fine-tuning Procedures

One reason why unsupervised RL has been hard to benchmark to date is that there is no agreed upon procedure for training and evaluating unsupervised RL agents. To this end, we standardize pre-training, fine-tuning, and evaluation in URLB. We split pre-training and fine-tuning into two phases consisting of N_{PT} and N_{FT} environment steps respectively. During pre-training, we checkpoint agents at 100k, 500k, 1M, 2M steps in order to evaluate downstream performance as a function of pre-training steps. For adapting the pre-trained policy to downstream tasks, we evaluate in the data-efficient regime where N_{FT} is 100k, since we are interested in agents that are quick to adapt.

3.2 Evaluation

We evaluate the performance of an unsupervised RL algorithm by measuring how quickly it adapts to a downstream task. For each fine-tuning task, we initialize the agent with the pre-trained network parameters, fine-tune the agent for 100k steps and measure its performance on the downstream task. This evaluation procedure is similar to how pre-trained networks in CV and NLP are fine-tuned to downstream tasks such as classification, object detection, and summarization. There exist other means of evaluating the quality of pre-trained RL agents such as measuring the diversity of data collected during exploration or zero-shot generalization of goal-conditioned agents. However, it is challenging to produce a general method to measure data diversity, and while zero-shot generalization with goal-conditioned agents can be powerful such a benchmark would be limited to goal-conditioned RL. For these reasons, data diversity and goal-conditioned zero-shot generalization are less common evaluation metrics. In an effort to provide a general benchmark, we focus on the fine-tuning efficiency of the agent after pre-training which allows us to evaluate a diverse set of baselines.

Unlike unsupervised methods in CV and NLP which focus solely on representation learning, unsupervised pre-training in RL requires both representation learning and behavior learning. For this reason, URLB benchmarks performance for both state-based and pixel-based agents. Benchmarking both state and pixel-based RL separately is important because it allows us to decouple unsupervised behavior learning from unsupervised representation learning. In state-based RL, the agent receives a near-optimal representation of the world through coordinate states. Evaluating state-based unsupervised RL agents allows us to isolate unsupervised behavior discovery without worrying about representation learning as confounding factor. Evaluating pixel-based unsupervised RL agents provides insight into how representations and behaviors can be learned jointly.

3.3 URLB Environments

We release a set of domains and downstream tasks for URLB that are based on the DeepMind Control Suite (DMC) [64]. The three reasons for building URLB on top of DMC are (i) DMC is already widely adopted and familiar to RL practitioners; (ii) DMC environments can be used with both state and pixel-based inputs; (iii) DMC features environments of varying difficulty which is useful for designing a benchmark that contains both challenging and feasible tasks. URLB evaluates performance on 12 continuous control tasks (3 domains with 4 downstream tasks per domain). From easiest to hardest, the URLB domains and tasks are:

Walker *(Stand, Walk, Flip, Run)*: A biped constrained to a 2D vertical plane. Walker is a challenging introduction domain for unsupervised RL because it requires the unsupervised agent to learn balancing and locomotion skills in order to fine-tune efficiently. **Quadruped** *(Stand, Walk, Jump, Run)*: A quadruped within a a 3D space. Like walker, quadruped requires the agent to learn to balance and move but is harder due to a high-dimensional state and action spaces and 3D environment. **Jaco Arm** *(Reach top left, Reach top right, Reach bottom left, Reach bottom right)*: Jaco Arm is a 6-DOF robotic arm with a three-finger gripper. This environment tests the unsupervised RL agent's ability to control the robot arm without locking and perform simple manipulation tasks. It was recently shown that this environment is particularly challenging for unsupervised RL [68].

4 URLB: Algorithmic Baselines for Unsupervised RL

In addition to introducing URLB, the other primary contribution of this work is open-sourcing a unified codebase for eight leading unsupervised RL algorithms. To date, unsupervised RL algorithms have been hard to compare due to confounding factors such as different evaluation procedures and optimization schemes. While URLB provides standardized pre-training, fine-tuning, and evaluation procedures, current algorithms are hard to compare since they rely on different optimization algorithms. For instance, Curiosity [48] utilizes PPO [54] while APT [42] uses SAC [27] for optimization. Moreover, even if two unsupervised RL methods use the same optimization algorithm, small differences in implementation can result in large performance differences that are independent of the pre-training algorithm. For this reason, it is important to provide a unified codebase with identical implementations of the optimization algorithm for each baseline. Providing such a unified codebase is one of the main contributions of this benchmark.

Algorithm 1 `Unsupervised RL`: Unsupervised Pre-training and Supervised Fine-tuning

Require: Randomly initialized actor π_θ, critic Q_ϕ, and encoder f_ξ networks, replay buffer $\mathcal{D}$.
Require: Intrinsic r^{int} and extrinsic r^{ext} reward functions, discount factor γ.
Require: Environment (env), M downstream tasks $T_k, k \in [1, \dots, M]$.
Require: pre-train N_{PT} and fine-tune N_{FT} steps.

for $t = 1..N_{\text{PT}}$ **do** ▷ Part 1: Unsupervised Pre-training
  $\mathbf{a}_t \leftarrow \pi_\theta(f_\xi(\mathbf{o}_t)) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$
  $\mathbf{o}_{t+1} \sim P(\cdot|\mathbf{o}_t, \mathbf{a}_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$
  Update π_θ, Q_ϕ, and f_ξ using minibatches from $\mathcal{D}$ and intrinsic reward r^{int} according to Eqs. 1 and 2.
end for
Outputs pre-trained parameters θ_{PT}, ϕ_{PT}, and ξ_{PT}
for $T_k \in [T_1, \dots, T_M]$ **do** ▷ Part 2: Supervised Fine-tuning
  initialize $\theta \leftarrow \theta_{\text{PT}}$,$\phi \leftarrow \phi_{\text{PT}}$, $\xi \leftarrow \xi_{\text{PT}}$, reset $\mathcal{D}$
  for $t = 1..N_{\text{FT}}$ **do**
    $\mathbf{a}_t \leftarrow \pi_\theta(f_\xi(\mathbf{o}_t)) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$
    $\mathbf{o}_{t+1}, r^{\text{ext}}_t \sim P(\cdot|\mathbf{o}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_t, r^{\text{ext}}_t, \mathbf{o}_{t+1})$
    Update π_θ, Q_ϕ, and f_ξ using minibatches from $\mathcal{D}$ according to Eqs. 1 and 2.
  end for
  Evaluate performance of RL agent on task T_k
end for

4.1 Backbone RL Algorithm

Since most of the above algorithms rely on off-policy optimization (and some cannot be optimized on-policy at all), we opt for a state-of-the-art off-policy optimization algorithm. While SAC [27] has been the de facto off-policy RL algorithm for many RL methods in the last few years, it is prone to suffering from policy entropy collapse. DrQ-v2 [67] recently showed that using DDPG [41] instead of SAC as a learning algorithm leads to a more robust performance on tasks from DMC. For this reason, we opt for DrQ-v2 [67] as our base optimization algorithm to learn from images, and DDPG, as implemented in DrQ-v2, to learn from states. DDPG is an actor-critic off-policy algorithm for continuous control tasks. The critic Q_ϕ minimizes the Bellman error

$$\mathcal{L}_Q(\phi, \mathcal{D}) = \mathbb{E}_{(\mathbf{o}_t, \mathbf{a}_t, r_t, \mathbf{o}_{t+1}) \sim \mathcal{D}} \left[\left(Q_\phi(\mathbf{o}_t, \mathbf{a}_t) - r_t - \gamma Q_{\bar{\phi}}(\mathbf{o}_{t+1}, \pi_\theta(\mathbf{o}_{t+1})) \right)^2 \right], \tag{1}$$

where $\bar{\phi}$ is an exponential moving average of the critic weights. The deterministic actor π_θ is learned by maximizing the expected returns

$$\mathcal{L}_\pi(\theta, \mathcal{D}) = \mathbb{E}_{\mathbf{o}_t \sim \mathcal{D}} \left[Q_\phi(\mathbf{o}_t, \pi_\theta(\mathbf{o}_t)) \right]. \tag{2}$$

4.2 Unsupervised RL Algorithms

As part of URLB, we open-source code for eight leading or well-known algorithms across all three of these categories all of which utilize the same optimization backbone. All algorithms provided with URLB differ only in their intrinsic reward while keeping all other parts of the RL architecture the same. We list all implemented baselines in Table 1 and provide a brief overview of the algorithms considered, which are binned into three categories – knowledge-based, data-based, and competence-based algorithms.[2] For detailed descriptions of each method we refer the reader to Appendix A.

Knowledge-based Baselines: Knowledge-based methods aim to increase knowledge about the world by maximizing prediction error. As part of the knowledge-based suite, we implement the Intrinsic Curiosity Module (ICM) [48], Disagreement [49], and Random Network Distillation (RND) [10]. All three methods utilize a function g to either predict the dynamics $g(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ (ICM, Disagreement) or predict the output of a random network $g(\mathbf{z}_t, \mathbf{a}_t)$ (RND), where $\mathbf{z}$ is the encoding of $\mathbf{o}$. ICM and RND maximize prediction error while Disagreement maximizes prediction uncertainty.

Data-based Baselines: Data-based methods aim to achieve data diversity by maximizing entropy. We implement APT [42] and ProtoRL [68] both of which maximize entropy $H(\mathbf{z})$ in different ways.

[2] We borrow this terminology from the following unsupervised RL tutorial [61].

Table 1: Unsupervised RL Algorithms implemented in URLB.

Name	Algo. Type	Intrinsic Reward
ICM [48]	Knowledge	$\|g(\mathbf{z}_{t+1}\|\mathbf{z}_t, \mathbf{a}_t) - \mathbf{z}_{t+1}\|^2$
Disagreement [49]	Knowledge	$\text{Var}\{g_i(\mathbf{z}_{t+1}\|\mathbf{z}_t, \mathbf{a}_t)\} \quad i = 1, \ldots, N$
RND [10]	Knowledge	$\|g(\mathbf{z}_t, \mathbf{a}_t) - \tilde{g}(\mathbf{z}_t, \mathbf{a}_t)\|_2^2$
APT [42]	Data	$\sum_{j \in \text{random}} \log \|\mathbf{z}_t - \mathbf{z}_j\| \quad j = 1, \ldots, K$
ProtoRL [68]	Data	$\sum_{j \in \text{prototypes}} \log \|\mathbf{z}_t - \mathbf{z}_j\| \quad j = 1, \ldots, K$
SMM [40]	Competence	$\log p^*(\mathbf{z}) - \log q_{\mathbf{w}}(\mathbf{z}) - \log p(\mathbf{w}) + \log d(\mathbf{w}\|\mathbf{z})$
DIAYN [21]	Competence	$\log q(\mathbf{w}\|\mathbf{z}) + \text{const.}$
APS [43]	Competence	$r_t^{\text{APT}}(\mathbf{z}) + \log q(\mathbf{z}\|\mathbf{w})$

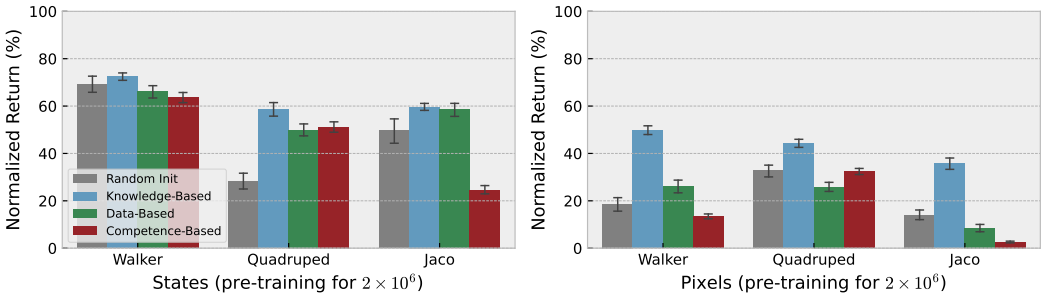


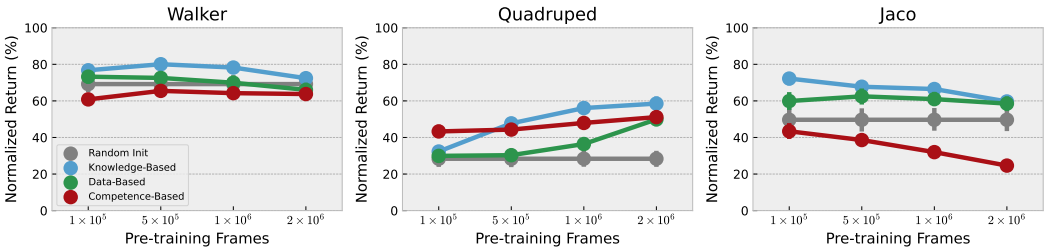
Figure 3: Aggregate results for each algorithm category after pre-training the agent with intrinsic rewards for 2M environment steps and finetuning with extrinisc rewards for 100k steps as described in Sec. 3.2. Scores are normalized by the asymptotic performance on each task (i.e., DrQ-v2 and DDPG performance after training from 2M steps on pixels and states correspondingly) and we show the mean and standard error of each category. Each algorithm is evaluated across ten random seeds. To provide an aggregate view of each algorithm category, the scores are averaged over individual tasks and methods (see Appendix C for detailed results for each algorithm and downstream task). The *Random Init* baseline represents DrQ-v2 and DDPG trained from a random initialization for 100k steps. Full results can be found in Section C.

Both methods utilize a particle estimator [60] to maximize the entropy by maximizing the distance between k-nearest neighbors (kNN) for each state or observation embedding $\mathbf{z}$. Since computing kNN over the entire replay buffer is expensive, APT estimates entropy across transitions in a randomly sampled minibatch. ProtoRL improves on APT by clustering the replay buffer with a contrastive deep clustering algorithm SWaV [12]. The centroids of the clusters are called *prototypes*, which are used by ProtoRL to estimate entropy.

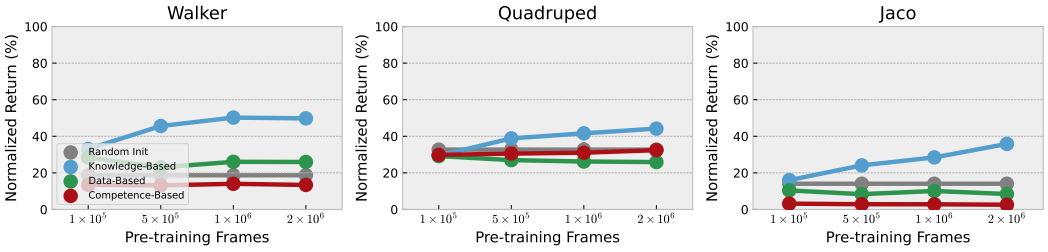
Competence-based Baselines: Competence-based algorithms, learn an explicit skill vector $\mathbf{w}$ by maximizing the mutual information between the encoded observation and skill $I(\mathbf{z}; \mathbf{w})$. This mutual information can be decomposed in two ways, $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z})$. We provide baselines for both decompositions. The former decomposition is utilized in skill discovery algorithms such as DIAYN [21], VIC [24], VALOR [1], which are conceptually similar. For URLB, we implement DIAYN. The latter decomposition, though less common, is implemented in the APS [43], which uses a particle estimator for the entropy term and successor features to represent the conditional entropy [30]. Lastly, we implement SMM [40] which combines both decompositions into one objective. Note that the SMM paper describes both skill-based and skill-free variants, so it can be categorized as both competence and data-based.

5 Experiments

We evaluate the algorithms listed in Table 1 by pre-training with the intrinsic reward objective and fine-tuning on the downstream task as described in Section 3.2. For DrQ-v2 optimization we fix the hyper-parameters from [67] and for algorithm-specific hyper-parameters we perform a grid sweep



(a) State-based learning.



(b) Pixel-based learning.

Figure 4: We display the fine-tuning efficiency as a function of pre-training steps. As in Fig. 3 scores are asymptotically normalized, averaged across tasks and algorithms on a per-category basis, and evaluated over ten seeds. Our expectation is that a longer pre-training phase should lead to more efficient fine-tuning. However, in several cases the empirical evidence goes against our intuition demonstrating that longer pre-training is not always beneficial. Understanding this shortcoming of current methods is an important direction for future research. Detailed results can be found in Figures 6 and 7.

and pick the best performing parameters. We benchmark both state and pixel-based experiments and keep all non-algorithm-specific architectural details the same with a full description available in Appendix B. Performance on each downstream task is evaluated over ten random seeds and we display the mean scores and standard errors. We summarize the main results of our evaluation in Figures 3 and 4, which show evaluation scores grouped by algorithm category, described in Section 4.2, and environment, described in Section 3.3. An extensive list of results across all algorithms considered in this work can be found in Appendix C.

By benchmarking a wide array of exploration algorithms on both state and pixel-based tasks we are able to get perspective on the current state of unsupervised RL. Overall, we find that while unsupervised RL shows promise, it is still far from solving the proposed benchmark and many open questions need to be addressed to make progress toward unsupervised pre-training for RL. We note that solving the benchmark means matching the asymptotic DrQ-v2 (for pixels) and DDPG (for states) performance within 100k steps of fine-tuning. The motivation for this definition is that unsupervised RL agents get access to unlimited reward-free environment interactions. After pre-training, we seek to develop agents that adapt quickly to the desired downstream task. We list our observations below:

O1: None of the implemented unsupervised RL algorithms solve the benchmark. Despite access to up to 2M pre-training steps, after 100k steps of fine-tuning no method matches asymptotic performance on most tasks. The best-performing benchmarked algorithms achieve $40 - 70\%$ normalized return whereas the benchmark is considered solved when the agent achieves near 100% normalized returns. This suggests that we are still far as a community from efficient generalization in deep RL.

O2: Unsupervised RL is not universally better than random initialization. We also observe that fine-tuning an unsupervised RL baseline is not always preferable to fine-tuning from a random initialization. In particular when learning from states, a random initialization is competitive with most baselines. However, when learning from pixels fine-tuning from random initialization degrades suggesting that representation learning is an important component of unsupervised pre-training.

O3: There exists a large gap in performance between exploring from states and exploring from pixels. Another observation that supports representation learning as an important aspect of exploration is that exploration algorithms degrade substantially when learning from pixels compared to learning from state. Shown in Figure 3, most algorithms lose $20 - 50\%$ when learning from pixels compared to state and especially so on the harder environments (Quadruped, Jaco Arm). These results suggest that better representation learning during pre-training is an important research direction.

O4: In aggregate, competence-based approaches underperform knowledge-based and data-based approaches. While knowledge-based and data-based approaches both perform competitively across URLB, we find that competence-based approaches are lagging behind. Specifically, there is no competence-based approach that achieves state-of-the-art mean performance on any of the URLB tasks, which points to competence-based unsupervised RL as an impactful research direction with significant room for improvement.

O5: There is not a single leading unsupervised RL algorithm for both states and pixels. We observe that there is no single state-of-the-art algorithm for unsupervised RL. At 2M pre-training steps, APT [42] and ProtoRL [68] are the leading algorithms for state-based URLB while ICM [48] achieves leading performance on pixel-based URLB despite the existence of more sophisticated knowledge-based methods [49, 10] (see Figure 5).

O6: For many unsupervised RL algorithms, rather than monotonically improving performance decays as a function of pre-training steps. We desire and would expect that the fine-tuning efficiency of unsupervised RL algorithms would improve as a function of pre-training steps. Surprisingly, we find that for 9 out of 18 experiments shown in Figure 4, performance either does not improve or even degrades as a function of pre-training steps. We see this as potentially the biggest drawback of current unsupervised RL approaches – they do not scale with the number of environment interactions. Developing algorithms that improve monotonically as a function of pre-training steps is an open and impactful line of research.

O7: New fine-tuning strategies will likely be needed for fast adaptation. While not investigated in depth in this benchmark, new fine-tuning strategies could play a large role in the adoption of unsupervised RL. Perhaps part of the issue raised in O6 could be addressed with better fine-tuning. The algorithms in URLB are all fine-tuned by initializing the actor-critic with the pre-trained weights and fine-tuning with an extrinsic reward. There are likely other better strategies for fine-tuning, particularly for competence based approaches that are conditioned on the skill $\mathbf{w}$.

6 Related work

Deep Reinforcement Learning Benchmarks. Part of the accelerated progress in deep RL over the last few years has been due to the existence of stable benchmarks. Specifically, the Atari Arcade Learning Environment [5], the OpenAI gym [7], and more recently the DeepMind Control (DMC) Suite [64] have become standard benchmarks for evaluating supervised RL agents in both state and pixel-based observation spaces and discrete and continuous action spaces. Open-sourcing code for algorithms has been another aspect that accelerated progress in deep RL. For instance, Duan et al. [19] not only presented a benchmark for continuous control but also provided baselines for common supervised RL algorithms, which led to the development of the widely used OpenAI gym benchmark [7] and baselines [18]. The combination of challenging yet feasible benchmarks and open-sourced code were important components in the discovery of many widely adopted RL algorithms [27, 44, 52–54].

In addition to Atari, OpenAi gym, and DeepMind control, there have been many other benchmarks designed to study different aspects of supervised RL. DeepMind lab [4] benchmarks 3D navigation from pixels, ProcGen [15, 16] measures generalization of supervised agents in procedurally generated environments, D4RL [22] and RL unplugged [26] benchmark performance of offline RL methods, B-Pref [39] benchmarks performance of preference-based RL methods, Metaworld [70] measures the performance of multi-task and meta-RL algorithms, and SafetyGym [51] measures how RL agents can achieve tasks with safety constraints. However, while the existing benchmarks are suitable for supervised RL algorithms, there is no such benchmark and collections of easy-to-use baseline algorithms for unsupervised RL, which is our primary motivation for accelerating progress in unsupervised RL through URLB.

Unsupervised Reinforcement Learning. While investigations into unsupervised deep RL appeared shortly after the landmark DQN [44], the field has experienced accelerated progress over the last year, which has been in part due to advents in unsupervised representation learning in CV [14, 31, 32] and NLP [8, 17, 50] as well as the development for stable RL optimization algorithms [27, 33, 41, 54]. However, unlike CV and NLP which focus solely on unsupervised representation learning, unsupervised RL has required both unsupervised representation and behavioral learning.

Unsupervised Representation Learning for Deep RL: In order for an RL algorithm to learn a policy $\pi(a|s)$ it must first have a good representation for the state s. When working with coordinate state, the representation is supplied by a the human task designer but when operating from image observations o, we must first transform the observations into latent vectors z. This transformation comprises the study of representation learning for RL. One of the first seminal works on unsupervised representation learning for RL showed that unsupervised auxiliary tasks improve performance of supervised RL [34]. Over the last two years, a series of works in unsupervised representation learning for RL with world models [28, 29] contrastive learning [38, 62, 68], autoencoders [66], and data augmentation [37, 67, 69] have dramaticaly improved learning efficiency from pixels. On many tasks from the DMC suite, learning from pixels is now as data-efficient as learning from state [38].

Unsupervised Behavioral Learning for Deep RL: One caveat is that the above algorithms are not fully unsupervised since they still optimize for an extrinsic reward but with an auxiliary unsupervised loss. Fully unsupervised RL also requires unsupervised learning of behaviors, which is typically achieved by optimizing for an intrinsic reward [47]. Given that representation learning is already heavily benchmarked for RL [28, 38, 69], URLB focuses mostly on unsupervised behavior learning. Many recent algorithms have been proposed for intrinsic behavioral learning, which include prediction methods [9, 48, 49], maximal entropy-based methods [11, 42, 43, 46, 56, 68], and maximal mutual information-based methods [21, 30, 43, 57]. However, these methods use different pre-training and evaluation procedures, different optimization algorithms, and different environments. To make fully unsupervised RL algorithm comparisons transparent and easier to develop, we introduce URLB.

7 Conclusion

We presented URLB, a benchmark designed to measure the performance of unsupervised RL algorithms. URLB consists of a suite of twelve evaluation tasks of varying difficulty from three domains and standardized procedures for pre-training and evaluation. We've open-sourced implementations and evaluation scores for eight leading unsupervised RL algorithms from all major algorithm categories. To minimize confounding factors, we utilized the same optimization method across all baselines. While none of the implemented baselines solve URLB, many make substantial progress suggesting a number of fruitful directions for unsupervised RL research. We hope that this benchmark makes the development and comparison of unsupervised RL algorithms easier and clearer.

Limitations. There are a number of limitations for both URLB and unsupervised RL methods in general. While URLB tasks are designed to be challenging, they are far from the visual and combinatorial complexity of real-world robotics. However, existing algorithms are unable to solve the benchmark meaning there is substantial room for improvement on the URLB tasks before moving on to even more challenging ones. While we present standardized pre-training and evaluation procedures, there can be many other ways of measuring the quality of the exploration algorithm. For instance, the quality of pre-training can be evaluated not only through policy adaptation but also through dataset diversity which we do not consider in this paper. In this work, similar to the Atari [5] and DMC [64] benchmarks for supervised RL we do not consider goal-conditioned RL which can be quite powerful for exploration [20]. For generality, we chose the currently most commonly used evaluation procedure that allowed us to benchmark a diverse set of leading exploration algorithms but, of course, other choices are available and would be interesting to investigate in future work.

Potential negative impacts. Unsupervised RL has the benefits of requiring zero extrinsic reward interactions during pre-training, and due to this the resulting agents may develop policies that are not aligned with human intent. This could be problematic in the long-term if not addressed early and carefully because as unsupervised robotics get more capable they can inadvertently inflict harm on themselves or the environment. Methods for constraining exploration within a broad set of human preferences (e.g. explore without harming the environment) is an interesting and important direction for future research in order to produced safe agents.

Acknowledgements

This work was partially supported by Berkeley DeepDrive, BAIR, the Berkeley Center for Human-Compatible AI, the Office of Naval Research grant N00014-21-1-2769, and DARPA through the Machine Common Sense Program.

References

[1] Achiam, Joshua, Edwards, Harrison, Amodei, Dario, and Abbeel, Pieter. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[2] Akkaya, Ilge, Andrychowicz, Marcin, Chociej, Maciek, Litwin, Mateusz, McGrew, Bob, Petron, Arthur, Paino, Alex, Plappert, Matthias, Powell, Glenn, Ribas, Raphael, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Barreto, Andre, Borsa, Diana, Quan, John, Schaul, Tom, Silver, David, Hessel, Matteo, Mankowitz, Daniel, Zidek, Augustin, and Munos, Remi. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

[4] Beattie, Charles, Leibo, Joel Z, Teplyashin, Denis, Ward, Tom, Wainwright, Marcus, Küttler, Heinrich, Lefrancq, Andrew, Green, Simon, Valdés, Víctor, Sadik, Amir, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[5] Bellemare, Marc G, Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[6] Berner, Christopher, Brockman, Greg, Chan, Brooke, Cheung, Vicki, Dębiak, Przemysław, Dennison, Christy, Farhi, David, Fischer, Quirin, Hashme, Shariq, Hesse, Chris, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[7] Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[8] Brown, Tom B, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[9] Burda, Yuri, Edwards, Harri, Pathak, Deepak, Storkey, Amos, Darrell, Trevor, and Efros, Alexei A. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJNwDjAqYX.

[10] Burda, Yuri, Edwards, Harrison, Storkey, Amos, and Klimov, Oleg. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[11] Campos, Víctor, Sprechmann, Pablo, Hansen, Steven Stenberg, Barreto, Andre, Kapturowski, Steven, Vitvitskyi, Alex, Badia, Adria Puigdomenech, and Blundell, Charles. Beyond fine-tuning: Transferring behavior in reinforcement learning. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

[12] Caron, Mathilde, Misra, Ishan, Mairal, Julien, Goyal, Priya, Bojanowski, Piotr, and Joulin, Armand. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020.

[13] Chen, Lili, Lu, Kevin, Rajeswaran, Aravind, Lee, Kimin, Grover, Aditya, Laskin, Michael, Abbeel, Pieter, Srinivas, Aravind, and Mordatch, Igor. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[14] Chen, Ting, Kornblith, Simon, Norouzi, Mohammad, and Hinton, Geoffrey E. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020.

[15] Cobbe, Karl, Klimov, Oleg, Hesse, Chris, Kim, Taehoon, and Schulman, John. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, 2019.

[16] Cobbe, Karl, Hesse, Chris, Hilton, Jacob, and Schulman, John. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, 2020.

[17] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[18] Dhariwal, Prafulla, Hesse, Christopher, Klimov, Oleg, Nichol, Alex, Plappert, Matthias, Radford, Alec, Schulman, John, Sidor, Szymon, Wu, Yuhuai, and Zhokhov, Peter. Openai baselines. https://github.com/openai/baselines, 2017.

[19] Duan, Yan, Chen, Xi, Houthooft, Rein, Schulman, John, and Abbeel, Pieter. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, 2016.

[20] Ecoffet, Adrien, Huizinga, Joost, Lehman, Joel, Stanley, Kenneth O., and Clune, Jeff. First return, then explore. *arXiv preprint arXiv:2004.12919*, 2020. doi: 10.1038/s41586-020-03157-9.

[21] Eysenbach, Benjamin, Gupta, Abhishek, Ibarz, Julian, and Levine, Sergey. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[22] Fu, Justin, Kumar, Aviral, Nachum, Ofir, Tucker, George, and Levine, Sergey. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[23] Gleave, Adam, Dennis, Michael, Wild, Cody, Kant, Neel, Levine, Sergey, and Russell, Stuart. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[24] Gregor, Karol, Rezende, Danilo Jimenez, and Wierstra, Daan. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

[25] Grill, Jean-Bastien, Strub, Florian, Altché, Florent, Tallec, Corentin, Richemond, Pierre H, Buchatskaya, Elena, Doersch, Carl, Pires, Bernardo Avila, Guo, Zhaohan Daniel, Azar, Mohammad Gheshlaghi, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020.

[26] Gulcehre, Caglar, Wang, Ziyu, Novikov, Alexander, Paine, Tom Le, Colmenarejo, Sergio Gomez, Zolna, Konrad, Agarwal, Rishabh, Merel, Josh, Mankowitz, Daniel, Paduraru, Cosmin, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[27] Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[28] Hafner, Danijar, Lillicrap, Timothy, Fischer, Ian, Villegas, Ruben, Ha, David, Lee, Honglak, and Davidson, James. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

[29] Hafner, Danijar, Lillicrap, Timothy, Ba, Jimmy, and Norouzi, Mohammad. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[30] Hansen, Steven, Dabney, Will, Barreto, André, Warde-Farley, David, de Wiele, Tom Van, and Mnih, Volodymyr. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2020.

[31] He, Kaiming, Fan, Haoqi, Wu, Yuxin, Xie, Saining, and Girshick, Ross B. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[32] Hénaff, Olivier J., Srinivas, Aravind, Fauw, Jeffrey De, Razavi, Ali, Doersch, Carl, Eslami, S. M. Ali, and van den Oord, Aäron. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, 2020.

[33] Hessel, Matteo, Modayil, Joseph, van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, Horgan, Dan, Piot, Bilal, Azar, Mohammad Gheshlaghi, and Silver, David. Rainbow: Combining improvements in deep reinforcement learning. In *Conference on Artificial Intelligence*, 2018.

[34] Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z., Silver, David, and Kavukcuoglu, Koray. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.

[35] Janner, Michael, Li, Qiyang, and Levine, Sergey. Reinforcement learning as one big sequence modeling problem. *CoRR*, abs/2106.02039, 2021.

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

[37] Laskin, Michael, Lee, Kimin, Stooke, Adam, Pinto, Lerrel, Abbeel, Pieter, and Srinivas, Aravind. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, 2020.

[38] Laskin, Michael, Srinivas, Aravind, and Abbeel, Pieter. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

[39] Lee, Kimin, Smith, Laura, Dragan, Anca, and Abbeel, Pieter. B-pref: Benchmarking preference-based reinforcement learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[40] Lee, Lisa, Eysenbach, Benjamin, Parisotto, Emilio, Xing, Eric P., Levine, Sergey, and Salakhutdinov, Ruslan. Efficient exploration via state marginal matching. *CoRR*, abs/1906.05274, 2019.

[41] Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[42] Liu, Hao and Abbeel, Pieter. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021.

[43] Liu, Hao and Abbeel, Pieter. APS: active pretraining with successor features. In *International Conference on Machine Learning*, 2021.

[44] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[45] Mnih, Volodymyr, Badia, Adria Puigdomenech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy, Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[46] Mutti, Mirco, Pratissoli, Lorenzo, and Restelli, Marcello. A policy gradient method for task-agnostic exploration. In *Conference on Artificial Intelligence*, 2021.

[47] Oudeyer, Pierre-Yves, Kaplan, Frdric, and Hafner, Verena V. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2): 265–286, 2007.

[48] Pathak, Deepak, Agrawal, Pulkit, Efros, Alexei A, and Darrell, Trevor. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

[49] Pathak, Deepak, Gandhi, Dhiraj, and Gupta, Abhinav. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, 2019.

[50] Radford, Alec, Wu, Jeffrey, Child, Rewon, Luan, David, Amodei, Dario, and Sutskever, Ilya. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[51] Ray, Alex, Achiam, Joshua, and Amodei, Dario. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

[52] Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael, and Moritz, Philipp. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

[53] Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael, and Abbeel, Pieter. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

[55] Schwarzer, Max, Anand, Ankesh, Goel, Rishab, Hjelm, R Devon, Courville, Aaron, and Bachman, Philip. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.

[56] Seo, Younggyo, Chen, Lili, Shin, Jinwoo, Lee, Honglak, Abbeel, Pieter, and Lee, Kimin. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, 2021.

[57] Sharma, Archit, Gu, Shixiang, Levine, Sergey, Kumar, Vikash, and Hausman, Karol. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

[58] Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[59] Silver, David, Hubert, Thomas, Schrittwieser, Julian, Antonoglou, Ioannis, Lai, Matthew, Guez, Artfhur, Lanctot, Marc, Sifre, Laurent, Kumaran, Dharshan, Graepel, Thore, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[60] Singh, Harshinder, Misra, Neeraj, Hnizdo, Vladimir, Fedorowicz, Adam, and Demchuk, Eugene. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23(3-4):301–321, 2003.

[61] Srinivas, Aravind and Abbeel, Pieter. Unsupervised learning for reinforcement learning, 2021. URL https://icml.cc/media/icml-2021/Slides/10843_QHaHBNU.pdf.

[62] Stooke, Adam, Lee, Kimin, Abbeel, Pieter, and Laskin, Michael. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, 2021.

[63] Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*. MIT Press, 2018.

[64] Tassa, Yuval, Doron, Yotam, Muldal, Alistair, Erez, Tom, Li, Yazhe, Casas, Diego de Las, Budden, David, Abdolmaleki, Abbas, Merel, Josh, Lefrancq, Andrew, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[65] Vinyals, Oriol, Babuschkin, Igor, Czarnecki, Wojciech M, Mathieu, Michael, Dudzik, Andrew, Chung, Junyoung, Choi, David H, Powell, Richard, Ewalds, Timo, Georgiev, Petko, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

[66] Yarats, Denis, Zhang, Amy, Kostrikov, Ilya, Amos, Brandon, Pineau, Joelle, and Fergus, Rob. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[67] Yarats, Denis, Fergus, Rob, Lazaric, Alessandro, and Pinto, Lerrel. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021.

[68] Yarats, Denis, Fergus, Rob, Lazaric, Alessandro, and Pinto, Lerrel. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, 2021.

[69] Yarats, Denis, Kostrikov, Ilya, and Fergus, Rob. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[70] Yu, Tianhe, Quillen, Deirdre, He, Zhanpeng, Julian, Ryan, Hausman, Karol, Finn, Chelsea, and Levine, Sergey. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 5, the appendix for hyperparameters. You can access the code with full instructions in the supplementary materials or using this link `https://github.com/rll-research/url_benchmark`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and 3.2 and the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5 and the supplementary material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section F.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See `custom_dmc_tasks` folder in supplementary materials codebase.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix:

Unsupervised Reinforcement Learning Benchmark

A Unsupervised Reinforcement Learning Baselines

A.1 Knowledge-based Baselines

Prediction methods train a forward dynamics model $f(\mathbf{o}_{t+1}|\mathbf{o}_t, \mathbf{a}_t)$ and define a self-supervised task based on the outputs of the model prediction.

Curiosity [48]: The Intrinsic Curiosity Module (ICM) defines the self-supervised task as the error between the state prediction of a learned dynamics model $\hat{\mathbf{z}}' \sim g(\mathbf{z}'|\mathbf{z}, \mathbf{a})$ and the observation. The intuition is that parts of the state space that are hard to predict are good to explore because they were likely to be unseen before. An issue with Curiosity is that it is susceptible to the *noisy TV problem* wherein stochastic elements of the environment will always cause high prediction error while not being informative for exploration:

$$r_t^{\text{ICM}} \propto \|g(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) - \mathbf{z}_{t+1}\|^2.$$

Disagreement [49]: Disagreement is similar to ICM but instead trains an ensemble of forward models and defines the intrinsic reward as the variance (or disagreement) among the models. Disagreement has the favorable property of not being susceptible to the noisy TV problem, since high stochasticity in the environment will result high prediction error but low variance if it has been thoroughly explored:

$$r_t^{\text{Disagreement}} \propto \text{Var}\{g_i(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)\} \quad i = 1, \ldots, N.$$

RND [10]: Random Network Distillation (RND) defines the self-supervised task by predicting the output of a frozen randomly initialized neural network $\tilde{f}$. This differs from ICM only in that instead of predicting the next state, which is effectively an environment-defined function, it tries to predict the vector output of a randomly defined function. Similar to ICM, RND can suffer from the noisy TV problem:

$$r_t^{\text{RND}} \propto \|g(\mathbf{z}_t, \mathbf{a}_t) - \tilde{g}(\mathbf{z}_t, \mathbf{a}_t)\|_2^2.$$

A.2 Data-based Baselines

Recently, exploration through state entropy maximization has resulted in simple yet effective algorithms for unsupervised pre-training. We implement two leading variants of this approach for URLB.

APT [42]: Active Pre-training (APT) utilizes a particle-based estimator [60] that uses K nearest-neighbors to estimate entropy for a given state or image embedding. Since APT does not itself perform representation learning, it requires an auxiliary representation learning loss to provide latent vectors for entropy estimation, although it is also possible to use random network embeddings [56]. We provide implementations of APT with the forward $g(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ and inverse dynamics $h(\mathbf{a}_t|\mathbf{z}_{t+1}, \mathbf{z}_t)$ representation learning losses:

$$r_t^{\text{APT}} \propto \sum_{j \in \text{random}} \log \|\mathbf{z}_t - \mathbf{z}_j\| \quad j = 1, \ldots, K.$$

ProtoRL [68]: ProtoRL devises a self-supervised pre-training scheme that allows to decouple representation learning and exploration to enable efficient downstream generalization to previously unseen tasks. For this, ProtoRL uses the contrastive clustering assignment loss from SWaV [12] and learns latent representations and a set of prototypes to form the basis of the latent space. The prototypes are then used for more accurate estimation of entropy of the state-visitation distribution via KNN particle-based estimator:

$$r_t^{\text{Proto}} \propto \sum_{j \in \text{prototypes}} \log \|\mathbf{z}_t - \mathbf{z}_j\| \quad j = 1, \ldots, K.$$

A.3 Competence-based Baselines

Competence-based approaches learn skills $\mathbf{w}$ that maximize the mutual information between encoded observations (or states) and skills $I(\mathbf{z}; \mathbf{w})$. The mutual information has two decompositions $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{w})$. We provide baselines for both decompositions.

SMM [40]: SMM minimizes $D_{KL}(p_\pi(\mathbf{z}) \parallel p^*(\mathbf{z}))$, which maximizes the state entropy, while minimizing the cross entropy from the state to the target state distribution. When using skills, $H(\mathbf{z})$ can be rewritten as $H(\mathbf{z}|\mathbf{w}) + I(\mathbf{z}; \mathbf{w})$. $H(\mathbf{z}|\mathbf{w})$ can maximized by optimizing the reward $r = \log q_{\mathbf{w}}(\mathbf{z})$, which is estimated using a VAE [36] that models the density of $\mathbf{z}$ while executing skill $\mathbf{w}$. Similar to other mutual information methods that decompose $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z})$, SMM learns a discriminator $d(\mathbf{w}|\mathbf{z})$ over a set of discrete skills with a uniform prior that maximizes $H(\mathbf{w})$:

$$r_t^{\text{SMM}} \triangleq \log p^*(\mathbf{z}) - \log q_{\mathbf{w}}(\mathbf{z}) - \log p(\mathbf{w}) + \log d(\mathbf{w}|\mathbf{z}).$$

DIAYN [21]: DIAYN and similar algorithms such as VIC [24] and VALOR [1] are perhaps the best competence-based exploration algorithms. These methods estimate the mutual information through the first decomposition $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}|\mathbf{z})$. $H(\mathbf{w})$ is kept maximal by drawing $\mathbf{w} \sim p(\mathbf{w})$ from a discrete uniform prior distribution and the density $-H(\mathbf{w}|\mathbf{z})$ is estimated with a discriminator $\log q(\mathbf{w}|\mathbf{z})$.

$$r_t^{\text{DIAYN}} \propto \log q(\mathbf{w}|\mathbf{z}) + \text{const.}$$

APS [43]: APS is a recent leading mutual information exploration method that uses the second decomposition $I(\mathbf{z}; \mathbf{w}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{w})$. $H(\mathbf{z})$ is estimated with a particle estimator as in APT [42] while $H(\mathbf{z}|\mathbf{w})$ is estimated with successor features as in VISR [30].[3]

$$r_t^{\text{APS}} \propto r_t^{\text{APT}}(\mathbf{z}) + \log q(\mathbf{z}|\mathbf{w})$$

B Hyper-parameters

In Table 2 we present a common set of hyper-parameters used in our experiments, while in table 3 we list individual hyper-parameters for each method.

Table 2: A common set of hyper-parameters used in our experiments.

Common hyper-parameter	Value
Replay buffer capacity	10^6
Action repeat	1 states-based and 2 for pixels-based
Seed frames	4000
n-step returns	3
Mini-batch size	1024 states-based and 256 for pixels-based
Seed frames	4000
Discount (γ)	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Critic target EMA rate (τ_Q)	0.01
Features dim.	1024 states-based and 50 for pixels-based
Hidden dim.	1024
Exploration stddev clip	0.3
Exploration stddev value	0.2
Number pre-training frames	up to 2×10^6
Number fine-turning frames	1×10^5

[3] In this benchmark, the generalized policy improvement(GPI) [3] that is used in Atari games for APS and VISR is not implemented for continuous control experiments.

Table 3: Per algorithm sets of hyper-parameters used in our experiments.

ICM hyper-parameter	Value
Representation dim.	512
Reward transformation	$\log(r + 1.0)$
Forward net arch.	$(\lvert\mathcal{O}\rvert + \lvert\mathcal{A}\rvert) \to 1024 \to 1024 \to \lvert\mathcal{O}\rvert$ ReLU MLP
Inverse net arch.	$(2 \times \lvert\mathcal{O}\rvert) \to 1024 \to 1024 \to \lvert\mathcal{A}\rvert$ ReLU MLP
Disagreement hyper-parameter	**Value**
Ensemble size	5
Forward net arch:	$(\lvert\mathcal{O}\rvert + \lvert\mathcal{A}\rvert) \to 1024 \to 1024 \to \lvert\mathcal{O}\rvert$ ReLU MLP
RND hyper-parameter	**Value**
Representation dim.	512
Predictor & target net arch.	$\lvert\mathcal{O}\rvert \to 1024 \to 1024 \to 512$ ReLU MLP
Normalized observation clipping	5
APT hyper-parameter	**Value**
Representation dim.	512
Reward transformation	$\log(r + 1.0)$
Forward net arch.	$(512 + \lvert\mathcal{A}\rvert) \to 1024 \to 512$ ReLU MLP
Inverse net arch.	$(2 \times 512) \to 1024 \to \lvert\mathcal{A}\rvert$ ReLU MLP
k in NN	12
Avg top k in NN	True
ProtoRL hyper-parameter	**Value**
Predictor dim.	128
Projector dim.	512
Number of prototypes	512
Softmax temperature	0.1
k in NN	3
Number of candidates per prototype	4
Encoder target EMA rate (τ_{enc})	0.05
SMM hyper-parameter	**Value**
Skill dim.	4
Skill discrim lr	10^{-3}
VAE lr	10^{-2}
DIAYN hyper-parameter	**Value**
Skill dim	16
Skill sampling frequency (steps)	50
Discriminator net arch.	$512 \to 1024 \to 1024 \to 16$ ReLU MLP
APS hyper-parameter	**Value**
Representation dim.	512
Reward transformation	$\log(r + 1.0)$
Successor feature dim.	10
Successor feature net arch.	$\lvert\mathcal{O}\rvert \to 1024 \to 1024 \to 10$ ReLU MLP
k in NN	12
Avg top k in NN	True
Least square batch size	4096

C Per-domain Individual Results

Individual fine-tuning results for each methods are shown in Figure 5. Furthermore, Figures 6 and 7 demonstrate individual results of states and pixels based fine-tuning performance as a function of pre-training steps for each considered method and task.

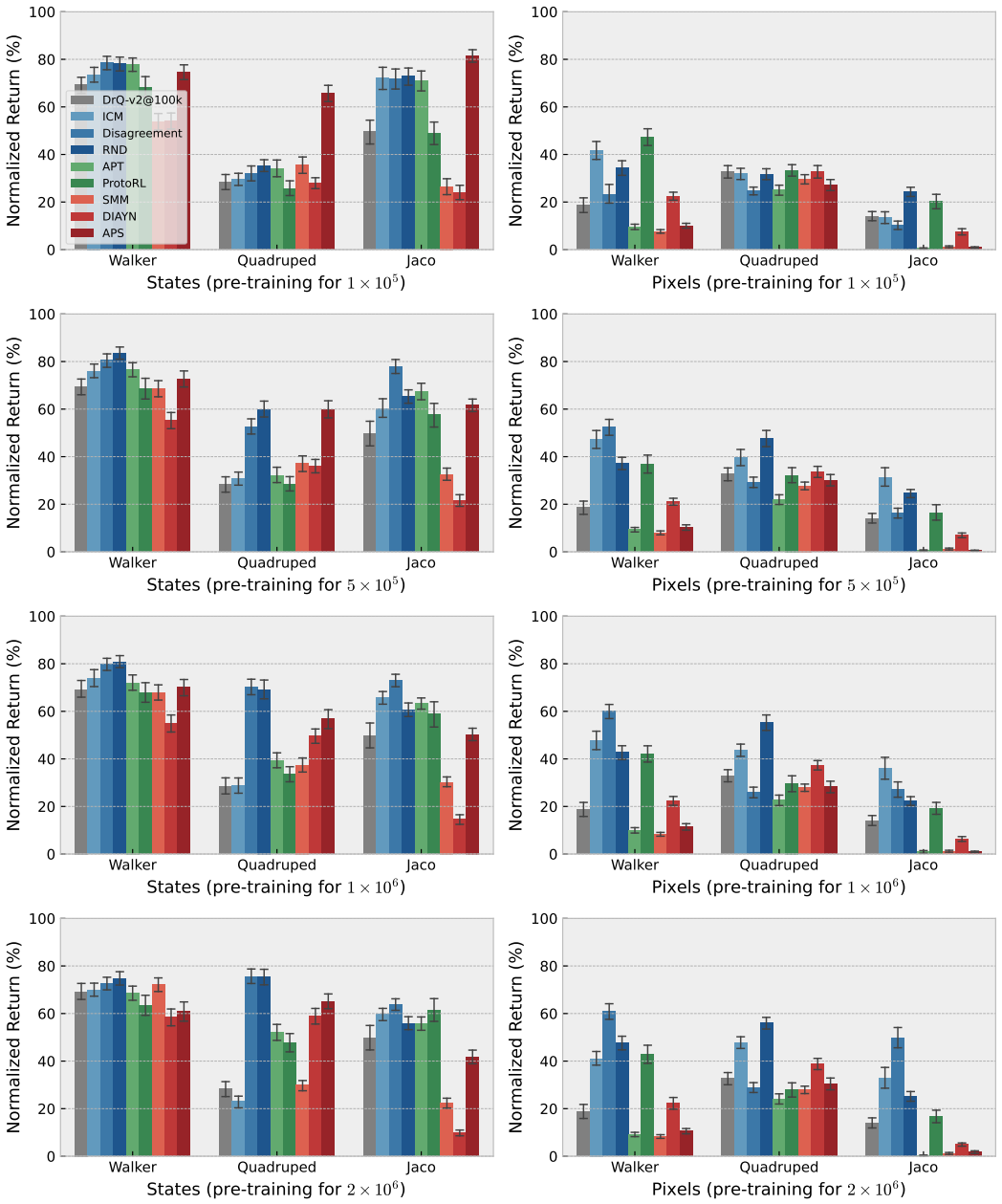


Figure 5: Individual results of fine-tuning for 100k steps after different degrees of pre-training for each considered method. The performance is aggregated across all the tasks within a domain and normalized with respect to the optimal performance.

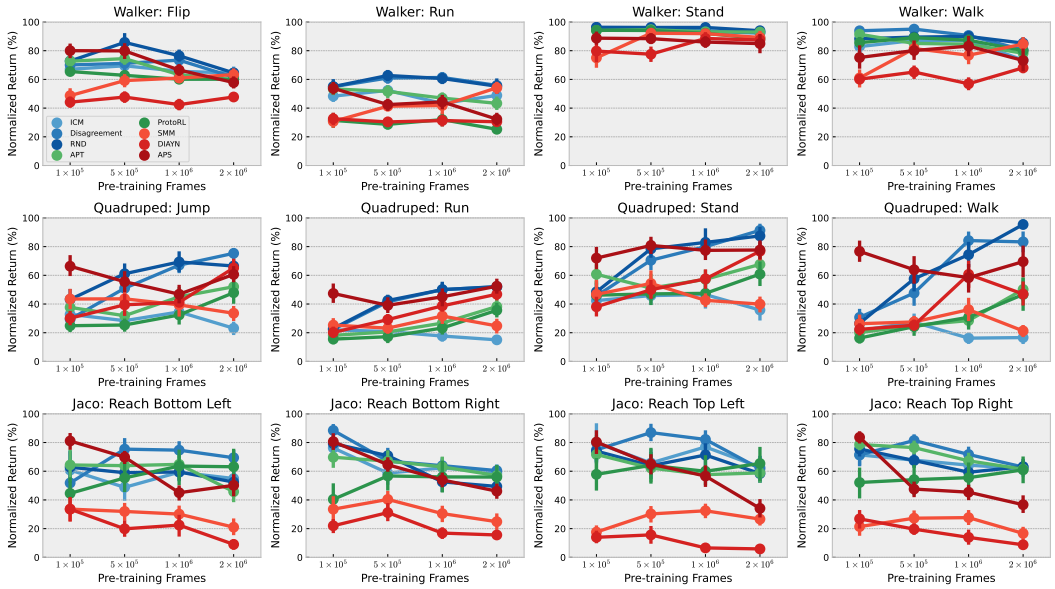


Figure 6: Individual results of fine-tuning efficiency as a function of pre-training steps for states-based learning.

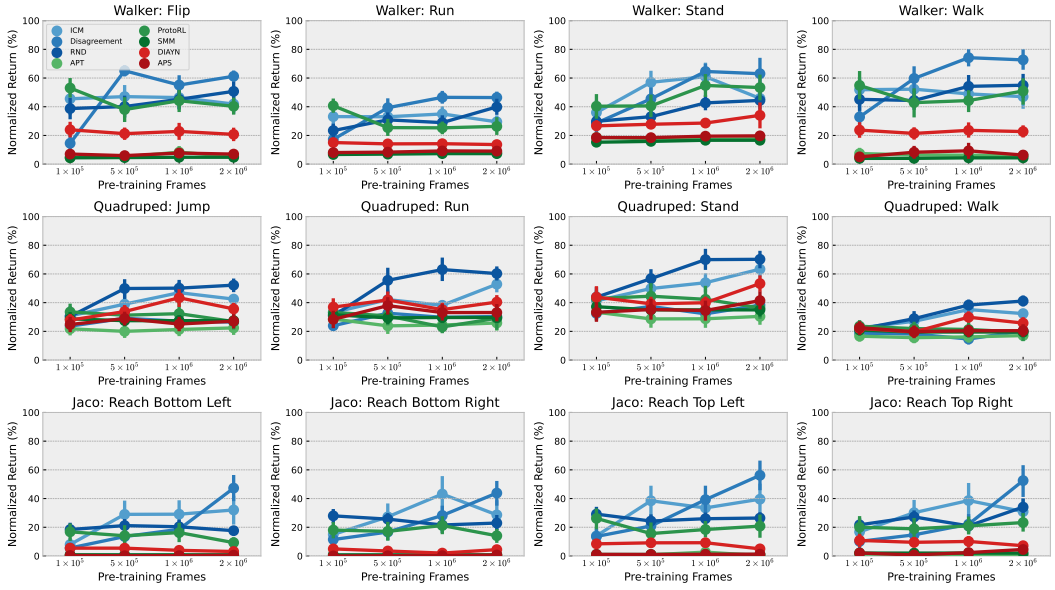


Figure 7: Individual results of fine-tuning efficiency as a function of pre-training steps for pixels-based learning.

D Finetuning Learning Curves

We provide finetuning learning curves for agents pre-trained for 2M steps with intrinsic rewards.

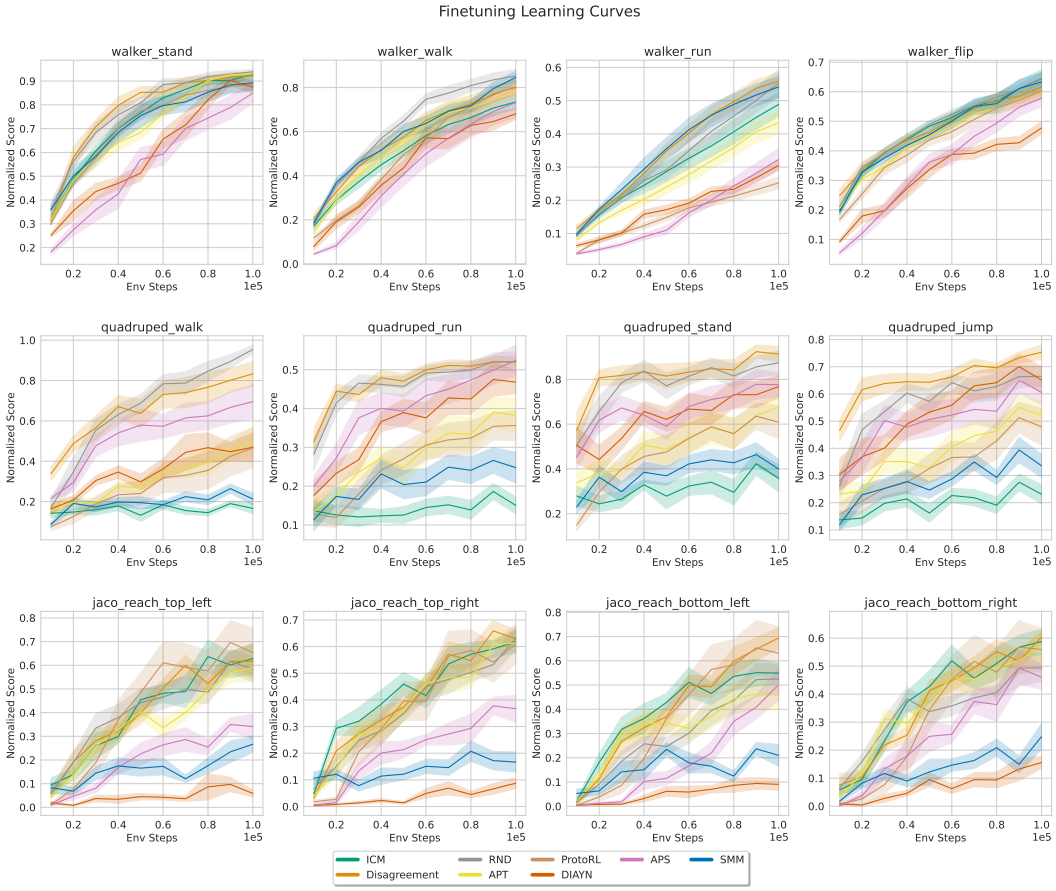


Figure 8: Finetuning curves for each evaluated unsupervised algorithm for each task considered in this benchmark after the agent has been pre-trained with intrinsic rewards.

E Individual Numerical Results

The individual numerical results of fine-tuning for each task and each method are presented in Table 4 for states-based learning, and in Table 5 for pixels-based learning.

Pre-trainining for 1×10^5 frames										
Domain	Task	DDPG (DrQ-v2)	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	535±19	559±20	581±21	580±28	523±16	388±36	352±22	638±33
	Run	325±25	384±24	437±24	437±33	424±28	250±34	244±28	259±26	428±26
	Stand	899±23	944±5	937±6	947±6	925±9	926±24	738±61	784±68	872±34
	Walk	748±47	805±46	911±14	857±32	888±19	831±31	592±53	584±25	731±70
Quadruped	Jump	236±48	291±35	261±45	383±57	334±48	220±33	386±56	267±34	589±57
	Run	157±31	195±31	198±40	203±20	161±27	138±21	224±33	179±26	420±49
	Stand	392±73	390±59	420±76	446±17	559±57	425±82	430±86	350±55	662±64
	Walk	229±57	185±20	265±45	229±26	173±29	141±23	227±47	193±29	664±56
Jaco	Reach bottom left	72±22	117±24	100±20	121±19	124±19	86±20	64±13	64±15	156±9
	Reach bottom right	117±18	155±10	179±6	161±8	141±14	82±21	68±17	44±8	164±10
	Reach top left	116±22	152±24	143±14	141±15	136±23	110±19	33±6	26±6	153±13
	Reach top right	94±18	159±15	159±15	168±9	175±6	116±22	47±12	59±12	186±7
Pre-trainining for 5×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	554±27	568±42	685±46	594±29	501±19	472±30	380±24	637±36
	Run	325±25	416±18	485±25	499±21	410±27	228±18	328±19	241±19	337±25
	Stand	899±23	930±7	940±8	946±5	930±5	925±17	906±18	762±44	869±27
	Walk	748±47	846±30	923±5	869±23	826±40	865±16	791±43	632±34	778±58
Quadruped	Jump	236±48	252±41	452±45	542±51	282±48	225±32	387±56	350±59	493±55
	Run	157±31	184±42	368±28	377±28	182±22	153±29	205±26	258±24	347±39
	Stand	392±73	422±49	649±53	722±49	470±80	433±63	499±78	459±54	743±46
	Walk	229±57	237±43	412±67	498±68	217±29	209±47	238±27	218±24	553±74
Jaco	Reach bottom left	72±22	94±16	145±13	113±11	123±15	106±18	61±10	38±9	134±6
	Reach bottom right	117±18	119±15	136±15	144±6	136±13	115±19	82±10	63±11	131±8
	Reach top left	116±22	125±18	165±9	121±16	118±18	122±23	57±9	29±10	124±11
	Reach top right	94±18	151±8	181±7	150±8	170±8	120±22	60±10	43±6	106±10
Pre-trainining for 1×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	524±20	586±36	610±27	505±22	480±17	486±17	338±23	531±24
	Run	325±25	344±30	488±23	482±23	373±22	254±29	332±40	249±24	352±31
	Stand	899±23	922±15	919±11	946±5	916±20	905±25	903±18	870±34	846±28
	Walk	748±47	845±27	880±19	873±24	821±35	848±29	746±51	553±33	808±61
Quadruped	Jump	236±48	306±42	595±39	615±53	400±53	287±52	349±63	365±15	415±46
	Run	157±31	157±24	444±39	444±42	237±35	206±32	280±40	343±25	400±48
	Stand	392±73	428±79	736±51	763±79	526±58	436±62	391±36	529±52	712±57
	Walk	229±57	140±24	729±46	644±70	246±33	266±64	312±63	525±76	505±84
Jaco	Reach bottom left	72±22	114±9	144±9	114±10	125±10	122±22	58±8	43±13	87±8
	Reach bottom right	117±18	126±10	129±10	106±13	128±12	113±20	62±9	34±6	109±9
	Reach top left	116±22	146±11	156±10	136±13	110±5	114±19	61±7	12±2	108±13
	Reach top right	94±18	143±10	159±10	132±10	149±11	123±21	61±9	31±9	101±9
Pre-trainining for 2×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	538±27	514±25	491±21	515±17	477±16	480±23	505±26	381±17	461±24
	Run	325±25	388±30	444±21	439±34	344±28	200±15	430±26	242±11	257±27
	Stand	899±23	913±12	907±15	923±9	914±8	870±23	877±34	860±26	835±54
	Walk	748±47	713±31	782±33	828±29	759±35	777±33	821±36	661±26	711±68
Quadruped	Jump	236±48	205±33	668±24	590±33	462±48	425±63	298±39	578±46	538±42
	Run	157±31	133±20	461±12	462±23	339±40	316±36	220±37	415±28	465±37
	Stand	392±73	329±58	840±33	804±50	622±57	560±71	367±42	706±48	714±50
	Walk	229±57	143±31	721±56	826±19	434±64	403±91	184±26	406±64	602±86
Jaco	Reach bottom left	72±22	106±8	134±8	101±12	88±12	121±22	40±9	17±5	96±13
	Reach bottom right	117±18	119±9	122±4	100±10	115±12	113±16	50±9	31±4	93±9
	Reach top left	116±22	119±12	117±14	111±10	112±11	124±20	50±7	11±3	65±10
	Reach top right	94±18	137±9	140±7	140±10	136±5	135±19	37±8	19±4	81±11

Table 4: Individual results of fine-tuning for 1×10^5 frames after different levels of pre-training in the states-based settings.

Pre-trainining for 1×10^5 frames										
Domain	Task	DrQ-v2	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	252±54	80±33	214±38	25±1	293±33	24±1	132±24	38±7
	Run	41±11	110±21	57±14	78±13	25±1	135±13	22±1	50±6	26±1
	Stand	212±28	315±67	250±62	261±34	162±9	353±67	133±8	233±22	162±9
	Walk	141±53	302±45	192±68	263±43	43±16	320±52	23±1	138±25	29±2
Quadruped	Jump	278±35	226±40	173±15	223±30	160±24	246±33	211±25	204±24	182±32
	Run	156±21	156±13	112±12	145±17	134±21	156±27	148±18	173±23	133±24
	Stand	309±47	329±49	259±31	350±43	266±37	342±35	297±36	350±48	265±48
	Walk	151±31	160±10	134±24	154±16	119±17	168±24	149±18	157±23	161±27
Jaco	Reach bottom left	23±10	18±7	12±4	41±7	0±0	38±11	1±1	12±4	0±0
	Reach bottom right	23±8	30±12	23±8	57±8	0±0	37±9	1±0	10±3	0±0
	Reach top left	40±9	31±11	30±9	66±9	0±0	59±14	2±1	19±4	2±1
	Reach top right	37±9	37±13	22±8	48±7	3±2	45±16	4±3	24±8	4±2
Pre-trainining for 5×10^5 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	260±39	360±16	222±37	28±2	210±44	25±1	117±18	32±2
	Run	41±11	110±15	131±19	103±13	26±1	85±16	23±1	47±4	27±1
	Stand	212±28	499±62	398±65	289±26	155±11	355±61	139±9	243±16	161±9
	Walk	141±53	305±51	348±46	258±33	37±10	250±54	23±1	125±19	48±19
Quadruped	Jump	278±35	286±50	214±24	366±44	147±26	229±42	201±20	248±28	212±27
	Run	156±21	198±29	153±19	261±38	112±20	144±27	138±16	197±24	178±23
	Stand	309±47	398±69	298±35	453±47	229±42	355±67	279±26	313±31	281±51
	Walk	151±31	193±31	129±20	206±30	111±20	157±25	139±13	140±19	141±24
Jaco	Reach bottom left	23±10	65±20	30±8	47±8	0±0	31±14	1±1	12±3	0±0
	Reach bottom right	23±8	56±16	34±10	52±6	0±0	35±11	1±0	7±2	0±0
	Reach top left	40±9	87±22	47±11	55±8	1±1	35±15	2±1	20±4	2±1
	Reach top right	37±9	68±17	33±5	61±8	2±1	42±14	4±3	21±5	1±1
Pre-trainining for 1×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	256±49	305±34	250±29	46±17	244±37	26±1	126±26	42±12
	Run	41±11	116±16	155±12	96±13	26±1	84±11	24±1	47±4	30±3
	Stand	212±28	534±73	565±37	374±37	150±13	480±63	145±6	251±19	170±10
	Walk	141±53	285±45	433±29	316±41	36±9	258±39	25±1	137±25	54±25
Quadruped	Jump	278±35	345±47	199±31	368±38	156±27	237±50	201±20	319±38	184±29
	Run	156±21	179±11	140±24	297±36	115±21	108±16	139±13	165±17	155±22
	Stand	309±47	430±44	257±35	559±51	229±42	338±71	279±26	319±27	275±48
	Walk	151±31	251±25	104±19	274±19	115±21	152±28	139±13	213±20	146±23
Jaco	Reach bottom left	23±10	65±19	42±10	46±9	0±0	36±13	1±1	8±3	0±0
	Reach bottom right	23±8	88±23	58±11	44±9	0±0	43±10	1±0	4±1	0±0
	Reach top left	40±9	76±19	89±19	59±7	6±4	41±10	2±1	20±4	2±0
	Reach top right	37±9	87±24	49±12	47±7	2±1	47±12	4±3	22±6	5±1
Pre-trainining for 2×10^6 frames										
Domain	Task	DDPG	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	81±23	231±34	339±16	280±31	28±2	223±27	26±1	114±21	38±9
	Run	41±11	98±11	154±9	133±15	25±2	87±18	24±1	45±3	30±3
	Stand	212±28	401±40	552±92	389±49	155±11	467±69	145±6	298±71	172±10
	Walk	141±53	274±44	424±36	321±43	35±8	297±48	25±1	132±19	37±5
Quadruped	Jump	278±35	312±18	194±21	383±28	164±27	197±35	201±20	262±22	199±29
	Run	156±21	249±21	143±25	284±18	121±20	137±35	139±13	190±18	156±24
	Stand	309±47	506±40	305±35	561±43	243±41	290±56	279±26	426±40	331±43
	Walk	151±31	231±15	145±10	294±20	122±21	138±35	139±13	184±23	146±24
Jaco	Reach bottom left	23±10	72±20	106±18	39±3	0±0	21±5	1±1	7±2	1±0
	Reach bottom right	23±8	58±19	90±15	47±9	0±0	28±7	1±0	9±3	1±1
	Reach top left	40±9	89±22	127±21	60±6	0±0	47±16	2±1	11±2	2±1
	Reach top right	37±9	69±18	118±23	76±11	1±1	52±12	4±3	16±3	10±3

Table 5: Individual results of fine-tuning for 1×10^5 frames after different levels of pre-training in the pixels-based settings.

F Compute Resources

URLB is designed to be accessible to the RL research community. Both state and pixel-based algorithms are implemented such that each algorithm requires a single GPU. For local debugging experiments we used NVIDIA RTX GPUs. For large-scale runs used to generate all results in this manuscripts, we used NVIDIA Tesla V100 GPU instances. All experiments were run on internal clusters. Each algorithm trains in roughly 30 mins - 12 hours depending on the snapshot (100k, 500k, 1M, 2M) and input (states, pixels). Since this benchmark required roughly 8k experiments (2 states / pixels, 12 tasks, 8 algorithms, 10 seeds, 4 snapshots) a total of 100 V100 GPUs were used to produce the results in this benchmark. Researchers who wish to build on URLB will, of course, not need to run this many experiments since they can utilize the results presented in this benchmark.

G Intuition on Competence-based Approaches Underperform on URLB

Across the three methods - data-based, knowledge-based, and competence-based - the best data-based and knowledge-based methods are competitive with one another. For instance, RND (a leading knowledge-based methods) and ProtoRL (a leading data-based method) achieve similar finetuning scores. Both are maximizing data diversity in two different ways - one through maximizing prediction error and the other through entropy maximization.

On the other hand, competence-based methods as a whole do much worse than data-based and knowledge-based ones. We hypothesize that this is due to current competence-based methods only supporting small skill spaces. Competence-based methods maximize a variational lower bound to the mutual information of the form:

$$I(\tau; z) = H(z) - H(z|\tau) = H(z) + \mathbb{E}[\log p(z|\tau)] \geq H(z) + \mathbb{E}[\log q(z|\tau)]$$

where $q(z|s)$ is called the discriminator. The discriminator can be interpreted as a classifier from $s \rightarrow z$ (or vice versa depending on how you decompose $I(s; z)$). In order to have an accurate discriminator, z is chosen to be small in practice (DIAYN - z is a 16 dim one-hot, SMM - z is 4 dim continuous, APS - z is 10 dim continuous).

OpenAI gym environments for continuous control mask this limitation because they terminate if the agent falls over and hence leak extrinsic signal about the downstream task into the environment. This means that the agent learns only useful behaviors that keep it balanced and therefore a small skill vector is sufficient for classifying these behaviors. However, in DeepMind control (and hence URLB) the episodes have fixed length and therefore the set of possible behaviors is much larger. If the skill space is too small, the most likely skills to be classified are different configurations of the agent lying on the ground. We hypothesize that building more powerful discriminators would improve competence-based exploration.