Unsupervised Model-based Pre-training for Data-efficient Reinforcement Learning from Pixels

Anonymous Authors[1]

Abstract

Reinforcement learning aims at autonomously performing complex tasks. To this end, a reward signal is used to steer the learning process. While successful in many circumstances, the approach is typically data hungry, requiring large amounts of task-specific interaction between agent and environment to learn efficient behaviors. To alleviate this, unsupervised reinforcement learning proposes to collect data through self-supervised interaction to accelerate task-specific adaptation. However, whether current unsupervised strategies lead to improved generalization capabilities is still unclear, more so when the input observations are high-dimensional. In this work, we advance the field by closing the performance gap in the Unsupervised Reinforcement Learning Benchmark, a collection of tasks to be solved in a data-efficient manner, after interacting with the environment in a self-supervised way. Our model-based approach combines exploration and planning to efficiently fine-tune unsupervised pre-trained models, achieving comparable results to task-specific baselines. We extensively evaluate our work, comparing several exploration methods and improving fine-tuning by studying the interaction between the model components. Furthermore, we investigate the limits of the learned model and of the unsupervised methods, to gain insights into how these influence the decision process, shedding light on new research directions.

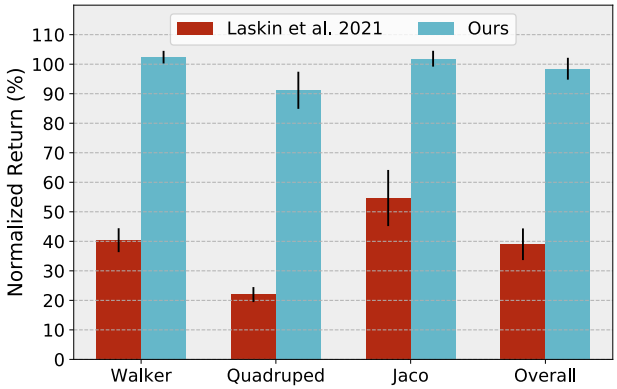


Figure 1. **Progress on the URL benchmark from pixels.** Comparison of the overall best performing approach from the URLB paper, i.e. Disagreement (Pathak et al., 2019) (**39.0±5.4%**), with our best performing approach (**98.5±3.7%**), fine-tuning for 100k steps pre-trained models that have been trained in an unsupervised way for 2M steps. Returns are normalized using the scores of supervised RL agents (details in Appendix).

1. Introduction

Modern successes of Reinforcement Learning (RL) have been realized through specialized agents that rely on task-specific rewards. Such autonomous agents have shown promising results scaling to raw high-dimensional inputs, e.g. playing Atari video games directly from pixels (Silver et al., 2016; Mnih et al., 2016), learning robotic manipulation policies from raw sensory input (Levine et al., 2016), etc. However, training an agent for each task individually requires a large amount of task-specific environment interactions, incurring huge redundancy. On the other hand, training agents that can generalize quickly on more than a single task is often desirable towards building intelligent autonomous systems. Developing algorithms that can efficiently adapt and generalize to new downstream tasks has hence become an active area of research in the RL community.

In computer vision and natural language processing, unsupervised learning strategies have enabled learning representations without supervision, that can quickly be adapted for a variety of tasks (Chen et al., 2020; Radford et al., 2019). In a similar fashion, unsupervised RL (URL) methods have focused on improving exploration performance (Pathak et al., 2017; Burda et al., 2019a; Bellemare et al., 2016) or learning generalizable skills (Eysenbach et al., 2019; Hansen et al.,

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2019). A key aspect here is to be able to generate useful behaviors in a self-supervised manner, without requiring any reward annotations, and subsequently adapt the learned behaviors to downstream tasks.

Recently, Laskin et al. (2021) introduced the Unsupervised Reinforcement Learning Benchmark (URLB), where agents are tested on different downstream tasks across three challenging continuous control domains of the DM Control Suite (Tassa et al., 2018). In this benchmark, an agent is first allowed a task-agnostic pre-training stage, where it can interact with the environment in an unsupervised way, followed by a fine-tuning stage where, given a limited budget of interactions with the environment, the agent should be able to quickly adapt for a specific task. The results obtained by Laskin et al. (2021) suggest that none of the tested unsupervised RL algorithms completely solve the benchmark and that models pre-trained by unsupervised RL are not universally better than random initialization. Furthermore, a large gap in performance was found between using state-based inputs versus high-dimensional pixel-based inputs, as the latter proved to be more difficult and sample inefficient. However, a string of recent literature on representation learning and model-based RL has shown that learning from pixel-based inputs can be effective, given an appropriate representation of input (Srinivas et al., 2020; Laskin et al.) and/or learning the environment dynamics (Hafner et al., 2019a;b). On control tasks (Tassa et al., 2018), these algorithms match state-based efficiency with learning from pixels, in a fully supervised setting. So the question remains as to how do we translate similar performance gains to unsupervised RL.

In this work, we improve upon the way unsupervised RL has been employed and show that, with the right precautions, it can be leveraged to solve the URL benchmark from pixels. By combining unsupervised RL approaches with a model-based algorithm, we show that a pre-trained task-agnostic model can significantly improve data-efficiency during the fine-tuning stage, enabling both planning and training task-specific policies on synthetic data generated by the model. Furthermore, we empirically study the interactions between the multiple modules learned during unsupervised pre-training and analyze the quality of the learned model, identifying appropriate design choices and shedding light on what could prevent the agents from adapting faster.

Our efforts are aimed at understanding and scaling unsupervised RL algorithms to operate efficiently on high-dimensional images. Our contributions can be summarized as: *(i)* the design of a class of model-based unsupervised RL approaches, which enable fast adaptation after an unsupervised pre-training stage, *(ii)* a study of the interplays between the pre-trained modules that allow to improve sample efficiency during the fine-tuning stage, *(iii)* an analysis of the model learned through unsupervised interaction with the environment, aimed at understanding what aspects could be improved to facilitate fast adaptation, *(iv)* novel quantitative metrics to evaluate misspecifications in the learned model.

We demonstrate through extensive experimentation that, following our approach, it is possible to bridge the performance gap between state-based and pixel-based inputs, and to achieve the asymptotic performance of supervised RL agents (Figure 1).

2. Preliminaries

The RL setting can be formalized as a Markov Decision Process (MDP), denoted with the tuple $\{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, T is the state transition dynamics, R is the reward function, and γ is a discount factor. The objective of an RL agent is to maximize the expected discounted sum of rewards over time for a given task, also called return, and indicated as $G_t = \sum_{k=t+1}^{T} \gamma^{(k-t-1)} r_k$. In continuous action settings, one popular approach to predict the most rewarding actions is to combine a model that learns to output the best action given a certain state, referred to as the actor model, and a model that learns to estimate the expected value of the actor's actions over time, given a certain state, referred to as the critic model. Actor-critic algorithms can be combined with the expressiveness of neural network models to solve complex continuous control tasks (Haarnoja et al., 2018; Lillicrap et al., 2016; Schulman et al., 2017).

In this work, we investigate the problem of fast adaptation for a downstream task, after a phase of unsupervised training and interaction with the environment. We adopt the URL benchmark, which consists of three control domains, *Walker*, *Quadruped* and *Jaco*, and twelve tasks, four per each domain. Consistently with URLB (Laskin et al., 2021), our experimental procedure is made of two phases: a pre-training (PT) phase, where the agent can interact with a task-agnostic version of the environment for up to $2M$ steps, and a fine-tuning phase (FT), where the agent interacts with the same environment, being provided a task to solve and a limited budget of $100k$ steps. During the PT phase, rewards are removed from the environment. Sensible information about the environment can be obtained by exploring the domain-dependent dynamics, which will remain unchanged in the downstream tasks. During FT, the agent receives task-specific rewards when interacting with the environment. As the agent has no prior knowledge of the task, it should both understand the task and solve it efficiently, in the limited budget.

Crucially, we focus on the pixel-based setup of URLB, where the environment is perceived by the agent only

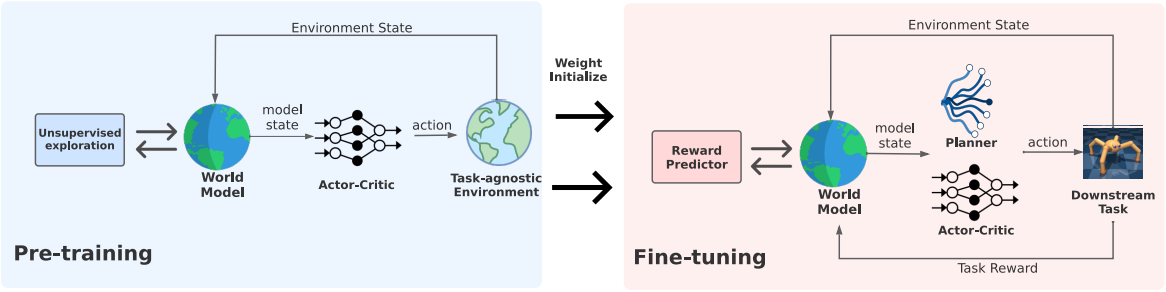


Figure 2. **Approach Overview.** The unsupervised benchmark consists of pre-training (PT) and fine-tuning (FT) stages. During pre-training, the agent interacts with the environment through an unsupervised RL strategy, maximizing an intrinsic reward function, and concurrently training a world model on the data collected. During fine-tuning, the agent exploits the world-model learned to efficiently plan and adapt for different downstream tasks, where it receives rewards from the environment to maximize.

through images. In this setting, the performance of several exploration strategies, combined with a state-of-the-art model-free approach (Yarats et al., 2021a), were shown to lack behind the asymptotic performance of an RL agent trained on the downstream task, as reported in (Laskin et al., 2021) and Figure 1. We believe one of the causes of this is that model-free RL algorithms cannot successfully leverage the information observed about the environment dynamics during PT, as they rely uniquely on actor and critic's predictions. To overcome this limitation, we ground our work on a model-based RL agent, namely the DreamerV2 algorithm (Hafner et al., 2021), whose learned model should allow preserving important information about the environment.

3. Approach

In order to perform well on the URL benchmark, it is important that an agent: (i) meaningfully interacts with the environment during the PT phase, to discover useful transitions; (ii) successfully reuses the modules learned during PT for fast adaptation; and (iii) efficiently employs the FT phase to understand and master the downstream task. In this section, we expand on how we addressed these challenges, giving rise to an approach that nearly solves the benchmark by achieving **98.5±3.7%** of a supervised RL agent's overall performance (Figure 1). We begin with a description of control with latent dynamics models.

3.1. Model-based Agent

The Dreamer agent attempts to learn a world model (Ha & Schmidhuber, 2018; Hafner et al., 2019b; 2021) that allows predicting the outcomes of future actions in the environment. The environment dynamics is captured into a latent space $\mathcal{Z}$, which allows a compact representation of the high-dimensional inputs of the agent. The world model consists of the following components:

$$
\begin{aligned}
&\text{Encoder:} && e_t = f_\phi(s_t), \\
&\text{Dynamics:} && p_\phi(z_t|z_{t-1}, a_{t-1}), \\
&\text{Posterior:} && q_\phi(z_t|z_{t-1}, a_{t-1}, e_t), \\
&\text{Image Decoder:} && p_\phi(s_t|z_t), \\
&\text{Reward Predictor:} && p_\phi(r_t|z_t).
\end{aligned}
$$

The model states z_t have both a deterministic component, modeled using the recurrent state of a GRU (Chung et al., 2014), and a (discrete) stochastic component. The encoder and decoder are convolutional neural networks (CNNs) and the remaining components are multi-layer perceptrons (MLPs). The world model is trained end-to-end by optimizing an evidence lower bound (ELBO) on the log-likelihood of the data collected in the environment (Hafner et al., 2019b;a).

In order to plan actions, Dreamer learns latent actor and critic networks:

$$\text{Actor:} \quad \pi_\theta(a_t|z_t), \qquad \text{Critic:} \quad v_\psi(z_t).$$

The actor is used to generate actions, given the model state, while the critic estimates the expected return for a certain model state, when following the actor's actions. Both components are trained online within the world model, by imagining the model state outcomes of the actions produced by the actor, using the model dynamics. Rewards for imagined trajectories are provided by the reward predictor and combined with the critic predictions to produce a GAE-λ estimate of the returns (Schulman et al., 2015). The actor maximizes such an estimate of the returns, backpropagating its gradients through the model dynamics.

For the encoder and the decoder networks, we used the same architecture as in DreamerV2 (Hafner et al., 2021). The hyperparameters for the agent, which we keep fixed across all domains and tasks, can be found in Appendix.

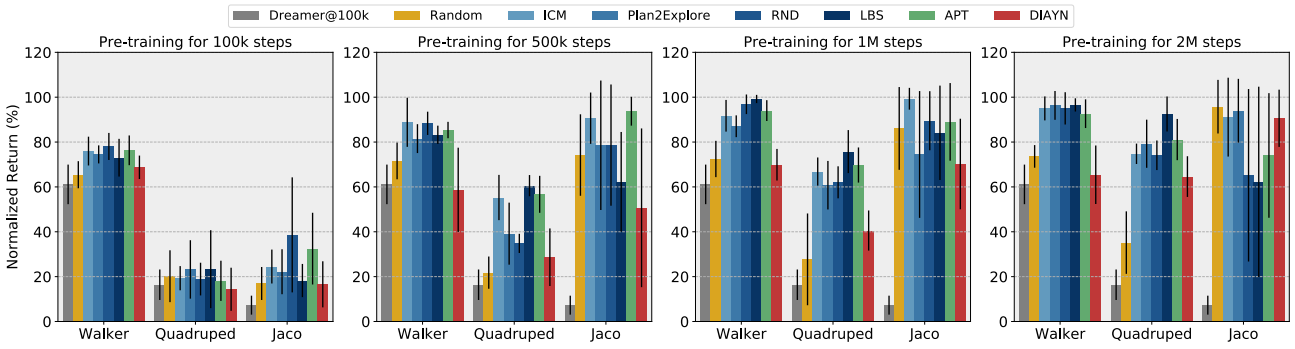


Figure 3. **Model-based URLB.** We combined multiple unsupervised RL approaches with the model-based Dreamer agent and present the performance across the different domains of URLB, after 100k steps of fine-tuning. Each plot presents result for snapshots taken after a certain number of pre-training, 100k, 500k, 1M and 2M steps, from left to right.

Score Normalization To serve as a reference point for the downstream tasks, we train Dreamer agents with 2M steps with full supervision on each task, which we refer to as *Dreamer@2M*. We use the mean performance of these oracle agents as our reference scores to normalize results, and we consider the URL benchmark solved when the performance of fine-tuned agents after unsupervised pre-training approaches the performance of the oracle agent. In Appendix A, we present the scores of *Dreamer@2M* and compare them to the DrQ-v2 baseline of Yarats et al. (2021a), showing that they are comparable.

3.2. Unsupervised Pre-training

Laskin et al. (2021) implemented several unsupervised strategies in the URLB, grouping them into three categories: knowledge-based, data-based and competence-based (Schmidhuber, 2010). These strategies are used to explore the environment during the pre-training stage, learning actor and critic networks of the agent. The resulting networks are then used to initialize respective components in the agent deployed for the downstream task, with the objective of reducing sample complexity on the downstream task.

As we employ a model-based agent, we use the experience collected to train the agent's world model, along with the actor and the critic networks and re-implement some of the unsupervised approaches from URLB to work in tandem with the Dreamer architecture. We note that, during PT, the reward predictor of the world model is either used to predict intrinsic rewards, according to the unsupervised approach employed, or unused. During FT, the reward predictor is trained on the downstream task rewards.

We chose to focus on knowledge-based methods as these combine well with the model-based nature of Dreamer, and implemented ICM (Pathak et al., 2017), RND (Burda et al., 2019b), LBS (Mazzaglia et al., 2021), and Plan2Explore (Sekar et al., 2020), as a model-based variant of Disagreement (Pathak et al., 2019). As a data-based approach, we ground upon APT (Liu & Abbeel, 2021b), and as a competence-based approach, we use DIAYN (Eysenbach et al., 2019). Finally, we add a random action baseline, as a maximum entropy baseline (Haarnoja et al., 2018). Details on these approaches and how we reimplemented them for our model-based agent are discussed in Appendix B.

3.3. Fine-tuning for Downstream Tasks

During the unsupervised PT phase, the agent collects experience from the environment that is used to train several components: a task-agnostic world model (without the reward predictor), an actor and a critic network. Moving to the FT phase, the pre-trained weights of these components can be copied into a new instance of the model, aiming to leverage previous experience for faster adaptation.

Since the domain dynamics stays the same between the PT and FT phases, initializing the world model with the pre-trained one should facilitate adaptation. However, the reward is changing from *pseudo-reward* to task reward when changing from the PT phase to FT phase. Hence, it is not clear if pre-training of the actor and critic can help for the downstream task. To shed light on this question, we conduct experiments in Section 4 to determine if it useful to transfer the actor and the critic. Unless specified, for our **default** FT model, which we refer to as (*w/ model, w/ actor, w/o critic*), we copy the weights of the pre-trained world model and actor but initialize the critic from scratch.

3.4. Model-based Planning

Since we train a world model from high-dimensional inputs, we can exploit model-based planning to adapt with limited additional environment interaction. When an accurate model of the environment is available, traditional model-based control approaches, such as Model Predictive Control (MPC) (Williams et al., 2015; Chua et al., 2018;

Richards, 2005), can be used to plan the agent's action. Nonetheless, using an actor and a critic has several advantages, such as amortizing the cost of planning by caching previously computed (sub)optimal actions and amortizing the cost of computing long-term returns from a certain state, without the need to imagine outcomes that are far in the future. We found it useful to adopt a hybrid planning strategy, which exploits both the actor and critic's predictions and an evolutionary strategy based on the Cross-Entropy Method (CEM; Rubinstein & Kroese (2004)). An overview of the end-to-end approach is illustrated in Figure 2 and a detailed algorithm is presented in Appendix C for reference.

4. Experiments and Analysis

For all the experiments, results are presented with at least three random seeds. Plots show average normalized returns with error bars showing the standard deviation.

Model-based URLB. The results of the different exploration approaches are shown in Figure 3. Results are presented by taking snapshots of the agent at different times during training, i.e. 100k, 500k, 1M and 2M steps, and fine-tuning the pre-trained policies and models for 100k steps. As opposed to the model-free experiments in URLB, where they fine-tuned the actor and the critic, we found that leveraging a pre-trained world model during fine-tuning dramatically improves the performance. Simply using random actions for unsupervised exploration already increases performance compared to a Dreamer agent trained from scratch for 100k steps (*Dreamer@100k*). This is in contrast to the results in (Laskin et al., 2021), where the improvements when exploiting pre-training were less significant.

More sophisticated exploration strategies generally lead to higher performance in complex tasks, with performance that increases over time in the Walker and Quadruped tasks. We note that the performance in the Jaco domain is less consistent over time and tends to have higher variance. We hypothesize that this is due to a greater discrepancy between *pseudo-reward* and task-reward for this environment. DIAYN underperforms compared to the other methods, providing some support to the observation in (Laskin et al., 2021), claiming that competence-based approaches tend to perform worse than the rest.

Exploiting Pre-Trained Modules. The results of the ablation studies on fine-tuning different sets of pre-trained modules are presented in Figure 4, averaging across all unsupervised RL methods. Overall, the default configuration, which reuses the weights of the world model and the actor performs best. Initializing the agent with the pre-trained actor is particularly useful in the Walker and Quadruped domains, but it is harmful in the Jaco tasks. A possible explanation for this could be that the exploration actor that is transferred from pre-training might have a precise explorative goal when brought to the fine-tuning stage, which may be particularly far from the target state of the task. In the Walker and Quadruped tasks, where the rewards are denser, exploring multiple states, even far from the downstream task behavior, provides useful reward information to exploit for the task. On the other hand, for the Jaco sparser tasks, if the exploration is initialized to reach a point that is too far from the reaching target, it might becomes arduous to encounter useful rewarding states within the reduced budget of FT. This hypothesis is supported by the fact that the random action methods and the DIAYN approach, which are believed to explore less than the others (thus, staying closer to the initial agent's position), perform well, and in some cases even better, than the other exploration approaches (Figure 3).

Initializing the critic with the pre-trained one has little impact in Walker and Quadruped but has been problematic in the Jaco domain, likely because of the more sparse rewards of the Jaco tasks, which are very different from the dense pseudo-rewards used to pre-train the critic. For this reason, we default to not reusing the critic's weights. Finally, the world model is confirmed to be the most valuable component to initialize, as, independently from the other components, initializing the world model brings a significant improvement in performance. Detailed results per each method are available in Appendix.

Leveraging Planning. In order to better exploit the pre-trained model during FT, while also leveraging the advantages of using an RL-like learning mechanism, we employ the recently proposed TD-MPC approach (Hansen et al.,

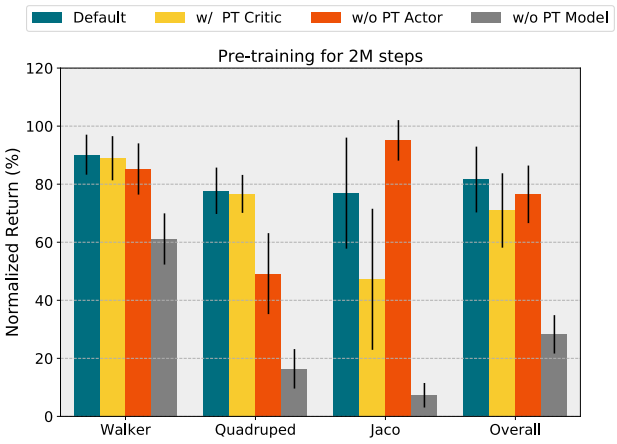


Figure 4. **Exploiting Pre-Trained Modules.** Comparison of the results when fine-tuning different pre-trained components of the agent. Results are averaged across all unsupervised RL methods (2M steps pre-training).

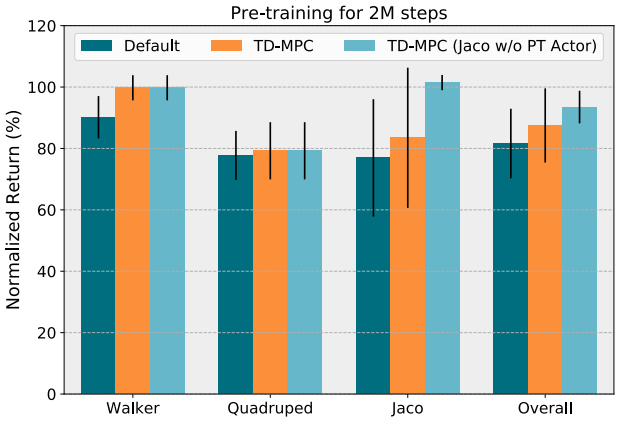


Figure 5. **Leveraging Planning.** Improved results on the benchmark, obtained by combining the actor-critic algorithm with MPC planning strategy (Hansen et al., 2022) and exploiting the previous insights on the Jaco domain. Results are averaged across all unsupervised RL methods (2M steps pre-training).

2022), a hybrid planning strategy that combines MPC and temporal difference actor-critic learning. TD-MPC claims improved performance, thanks to the combination of short-term planning using MPC and long-term predictions, using the actor-critic architecture. We implement TD-MPC on top of our models and compare the performance with the standard Dreamer actor-critic learning in Figure 5. We observe that TD-MPC slightly improves the performance in all domains. Combining the improved planning strategy with the previous insight of not initializing the weights of the actor in the Jaco domain, we obtain our overall best performance (*TD-MPC (Jaco w/o PT actor)*), with the LBS-based model being the overall best performing method (**98.5±3.7%**; Figure 1). Detailed results for each method are available in Appendix.

4.1. Analysis

First, we showed how we managed to bridge the performance gap in the URL benchmark by leveraging model-based unsupervised RL, improving the exploitation of the different agent's components during FT, and leveraging planning to improve data efficiency. Now, we focus on providing an analysis of the world model learned during the unsupervised pre-training stage, aiming to gain insights on which aspects improve decision awareness in the URLB setting, i.e. how can we best exploit the two stages of training to improve the agent's decision making. To shed light on this, we analyze discrepancies in the model's dynamics and in the reward prediction process that is leveraged during the fine-tuning planning.

Zero-shot Performance. How useful is the model learned during the unsupervised pre-training stage? To gain insights into this matter, we perform some additional tests where we provide the FT agent with a pre-trained reward predictor, that we train on the data collected during PT, separately from the agent. Given such a reward predictor, it should be possible to achieve high performance on the downstream tasks by simply planning within the model, e.g. performing MPC in a zero-shot setting. This assumes that the model correctly learned the dynamics of the environment and explored rewarding transitions that are relevant to the downstream task. In Figure 6, we compare the results of performing MPC in a zero-shot setting with the performance of an MPC agent that is allowed the typical 100k steps for fine-tuning. As for the MPC method, we employ Model Predictive Path Integral control (MPPI) (Williams et al., 2015). Because MPC is particularly expensive to test, we just perform this experiment on top of the models trained with the Plan2Explore URL approach. We also plot the performance of a non-pre-trained model and of using an actor-critic planning strategy (also provided with the reward predictor since the beginning of fine-tuning), for comparison.

We observe that the performance of MPC (zero-shot) is generally weak. While it overall performs better than the non-pre-trained model, simply applying MPC leveraging only the pre-trained modules and the reward predictor trained on the pre-training stage data is not sufficient to guarantee satisfactory performance. The fact that exploiting the fine-tuning stage using the same MPC approach generally boosts performance demonstrates that the model has a major benefit from the fine-tuning stage. Still, the performance of MPC generally lacks behind the actor-critic performance, suggesting that, especially in a higher-dimensional action space such as the Quadruped one, amortizing the cost of

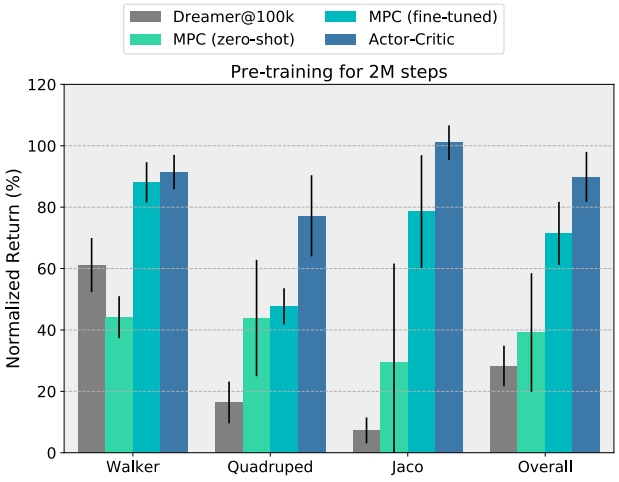


Figure 6. **Model Predictive Control.** Exploiting a pre-trained reward predictor to test whether is there a gap between zero-shot and fine-tuned MPC performance. Results refer to the Plan2Explore pre-trained agent (2M steps pre-training).

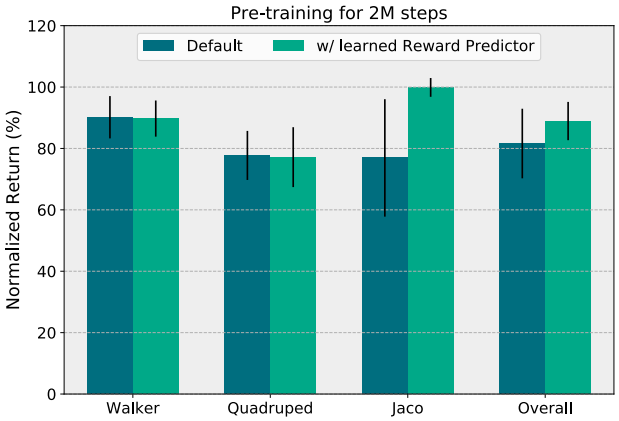


Figure 7. **Model-based agent reward predictor ablation.** Evaluating performance when providing a task-specific reward predictor trained on the transitions collected during the unsupervised pre-training stage. Results are averaged across all unsupervised RL methods (2M steps pre-training).

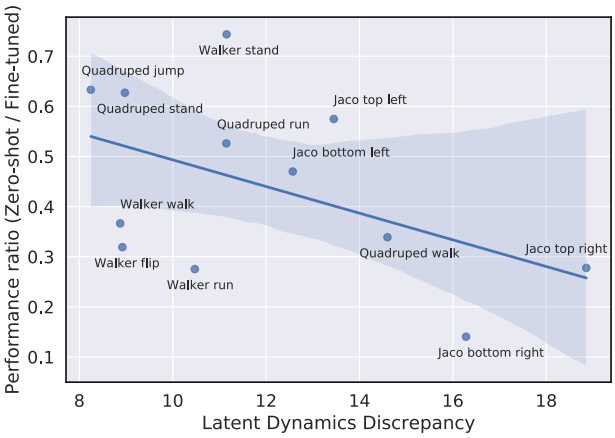


Figure 8. **Correlation between latent dynamics discrepancy and performance.** Results refer to the Plan2Explore pre-trained agent. Most difficult tasks are highlighted.

planning with actor-critic seems crucial to achieve higher performance.

Learning the Reward Predictor. Given that the agent strongly benefits from the 100k fine-tuning steps, we are interested in quantifying how much of this improvement is related to the necessity of learning a good reward function for the downstream task. In Figure 7, we measure the gap in performance between pre-trained agents that have no knowledge of the reward function at the beginning of fine-tuning and agents whose reward predictor is initialized from a reward predictor learned on top of the unsupervised pre-training data. Interestingly, the performance gap is overall small and irrelevant in the Quadruped and Walker domains. In the Jaco tasks, which have sparser reward functions, an a priori knowledge of the downstream task at the beginning of FT strongly improves performance.

According to our results, it is important that during the 100k steps of fine-tuning the actor is able to quickly obtain information about the downstream task. This might be easier for dense reward tasks, such as the Walker and Quadruped ones, but trickier in sparse settings like Jaco. As the initialization of the actor-critic modules also showed to be a compelling issue in our ablation study on the pre-trained modules exploitation (Figure 4), finding more efficient ways to fine-tune the actor-critic modules could be an impactful research direction that will facilitate the adoption of unsupervised pre-training for RL.

Latent Dynamics Discrepancy. A useful measure to assess the uncertainty or inaccuracy of a given model's dynamics is the model's misspecification, generally measured as the difference between the dynamics predictions and the real environment dynamics. When this metric is available, it is also possible to build robust RL strategies, that take the dynamics uncertainty into account while searching for the optimal behavior (Talvitie, 2018). Dealing with pixel-based inputs, we observe the dynamics of the environment through high-dimensional images, which hinders the possibility to evaluate such a metric, as distances in pixel space can be misleading.

In our approach, we use a model-based RL agent that learns a model of the environment in a compact latent space $\mathcal{Z}$. In order to quantify the "misspecification" of the learned latent dynamics, we propose a new metric, which we call *Latent Dynamics Discrepancy*, that suits the setup of URLB. We aim to quantify the distance between the predictions of the pre-trained model and the same model after fine-tuning on a downstream task. However, as the decoder of the world model gets updated during fine-tuning, the latent space mapping between model states z and environment states s might drift. For this reason, we ran fine-tuning experiments where the agent's decoder weights are frozen, so that the decoder cannot be updated and the model can only improve the posterior and the dynamics. This ensures that the mapping $\mathcal{Z} \to \mathcal{S}$ remains unchanged and allows to compare the dynamics model after fine-tuning with the one before fine-tuning. In order to measure the distance between the distribution output by the dynamics network, we chose the symmetrical Jensen-Shannon divergence:

$$\mathbb{E}_{(z_t,a_t)}\left[D_{\text{JS}}[p_{\text{FT}}(z_{t+1}|z_t,a_t)\|p_{\text{PT}}(z_{t+1}|z_t,a_t)]\right], \quad (1)$$

where the expectation is taken over the previous model states z_t sampled from the fine-tuned posterior $q_{\text{FT}}(z_t)$, actions a_{t-1} sampled from an oracle actor $\pi^*(a_t|z_t)$, so that we evaluate the metric on optimal trajectories, whose environment's state distribution corresponds to the stationary

Pre-training for 2M environment steps				
	ICM	LBS	P2E	RND
Pearson Correlation	-0.54	-0.60	-0.34	-0.03
p-value	0.07	**0.04**	0.28	0.91

Table 1. **Correlation between performance and intrinsic rewards.** Each column shows the Pearson correlation index and the p-value between fine-tuned performance across the URLB tasks and the intrinsic rewards computed on some oracle episodes.

distribution induced by the actor $s_t \sim d^{\pi^*}(s_t)$. We used 30 trajectories per task in our evaluation.

In Figure 8, we plot the correlation between our metric and the performance ratio between a zero-shot model and a fine-tuned model, where Plan2Explore was used for the 2M steps pre-training phase. We observed a strong negative Pearson correlation (-0.62), with a p-value of $0.03 < 0.05$, asserting that we must reject the null hypothesis, i.e. there exists a correlation between the two factors. This means that major updates in the model dynamics during fine-tuning played an important role in improving the agent's performance, compared to the pre-trained model and zero-shot performance. Future research may attempt to reduce such link, by either improving the model's learning process, so that the pre-trained dynamics could have greater accuracy, or the data collection process, proposing URL methods that directly aid to reduce such uncertainty.

Unsupervised Rewards and Performance. How well can the unsupervised RL approaches we employed help improving adaptation further? To answer this question, we analyze the correlation between the normalized performance of the different agents and the intrinsic rewards they provide for optimal trajectories obtained by an oracle agent. A strong negative correlation between the two factors would indicate that the agent will be more interested in seeing the optimal trajectories when its performance is low on the task. We summarize the results of our analysis in Table 1.

We observe that there is negative correlation between Plan2Explore (P2E), ICM, LBS's performance and their intrinsic rewards, while we found $\sim$0 correlation for RND. In particular, the correlation for LBS, which overall performed best in the benchmark, has a statistical significance, as its p-value is < 0.05. Given such correlation, we believe the intrinsic rewards of LBS might be one of the causes of its outstanding performance. As LBS searches for transitions of the environment that are difficult to predict for its dynamics, the model likely learns those transitions more accurately, facilitating planning during the fine-tuning stage and eventually leading to higher performance (particularly in the most difficult domain, Quadruped). It is important that future work would consider learning a less ambiguous dynamics during the unsupervised RL phase, which can be efficiently leveraged by the agent for fine-tuning.

5. Related Work

Our work lies at the intersection of unsupervised RL, model-based RL, and representation learning for RL. We discuss below the relevant literature in these three fields.

5.1. Model Based RL

In continuous control, model-based RL combined with powerful search methods has led to impressive results on a wide variety of tasks such as Atari (Schrittwieser et al., 2020) and continuous control (Hafner et al., 2019a). In this work, we used the MPC approach MPPI, which is based on the Cross-Entropy Method (CEM Rubinstein & Kroese (2004)). These methods perform trajectory optimization by fitting a multivariate Gaussian distribution to the imagined future actions allowing them to search the space efficiently. Alternative search methods such as Monte Carlo Tree search (Coulom, 2006) and Sequential Monte Carlo planning (Piché et al., 2018) could also have been used without much change to the algorithm. Given that we do not have the reward information during pre-training, we base our model on the Dreamer architecture which reconstructs the future frames (and rewards, when available) to learn a transition model. This has the advantage of being simple and not requiring the task specification a priori. Task and reward awareness is necessary to learn a model to be self-consistent in latent space (Schrittwieser et al., 2020; Grimm et al., 2020).

5.2. Unsupervised RL

The notion of a reward is sparse or could be missing completely in many real-world tasks. While it may be possible to handcraft dense reward signals for many real-world tasks, it is a worthwhile endeavor to investigate learning methods that do not require dense supervision. In such cases, it is important to generate meaningful behavior in a completely self-supervised way. This is also related to the notion of intrinsic motivation in Artificial Intelligence and Psychology. Research in this area spans many fields, from computational accounts of useful intrinsic motivations (Barto, 2004) to empirical evidence for certain intrinsic costs in humans (Kool et al., 2013). Such intrinsic behavior learning could aid an RL agent to adapt across tasks posed by the environment in a sample-efficient manner. Although it is unclear as to the exact nature and origin of good intrinsic reward functions, there have been extensive studies recently that streamlined the way one can conveniently categorize such approaches. Oudeyer & Kaplan (2008) classified intrinsic motivation algorithms into three different kinds - knowledge-based, competence-based and data-based models. Knowledge-based models rely on an prediction-error signal to build pseudo-rewards (Pathak et al., 2017;

2019; Burda et al., 2019a; Mazzaglia et al., 2021; Burda et al., 2019b; Rajeswar et al., 2021). Schmidhuber (2010) provided a coherent formulation of knowledge-based intrinsic motivation, which is measured by the performance of a predictive world model trained by the learning algorithm. Competence models aim at learning a set of diverse and repeatable policies through information-theoretic objectives (Mohamed & Rezende, 2015). This is achieved by maximizing the mutual information between the trajectory or states and latent skill variables (Eysenbach et al., 2019; Gregor et al., 2016; Liu & Abbeel, 2021a; Frank et al., 2014). Data-based methods try to increase the diversity of the dataset, often times through explicit maximum entropy objectives or count-based objectives. Bellemare et al. (2016); Ostrovski et al. (2017); Liu & Abbeel (2021b); Yarats et al. (2019). An unsupervised RL approach that is closely related to our method is Plan2Explore (Sekar et al., 2020), which combines Disagreement (Pathak et al., 2019), a knowledge-based exploration approach, with the first iteration of the Dreamer agent (Hafner et al., 2019a). In their work, they also presented few-shot adaptation experiments. We improve upon their method in several ways: combining various unsupervised RL strategies with the DreamerV2 agent (Hafner et al., 2021), exploiting the pre-trained components of the agent for fast adaptation, and employing a hybrid planning strategy to improve data-efficiency.

5.3. Representation Learning

When inputs are high-dimensional images, it is beneficial to learn compact state representations of the inputs. Much progress in unsupervised representation learning for RL has been influenced by developments in vision-based unsupervised learning (Chen et al., 2020; Kingma & Welling, 2013). More recently, a number of works have investigated representation learning for RL (Srinivas et al., 2020; Laskin et al.; Yarats et al., 2021b). In our work, we focus on representation learning with autoencoders (Hafner et al., 2019b; Yarats et al., 2019). Specifically, we leverage DreamerV2 (Hafner et al., 2021) for efficient exploration and fast adaptation. Predicting ahead in learned latent space facilitates both long-term predictions and allows to efficiently predict thousands of compact state sequences in parallel.

6. Conclusion

Unsupervised RL has been a promising direction aimed at training generalist fast adapting agents. However, the existing approaches have only yielded sub-optimal results when evaluated on a standardized setup, especially on high-dimensional pixel inputs. We build on existing unsupervised RL algorithms with a model-based formulation to improve the sample-efficiency and performance on pixel-based inputs. We empirically show that algorithms that learn a world model through self-supervised exploration can significantly improve their performance, compared to model-free approaches. Specifically, with our approach we were able to attain near-optimal performance on URLB, a challenging benchmark to evaluate unsupervised RL algorithms. Furthermore, we propose useful evaluation strategies to assess the quality of the learned model, quantify the uncertainty of the latent dynamics, and assess the intrinsic rewards provided by different unsupervised RL strategies, granting additional understanding of our approach.

Given that our model-based framework relies on the predictions of a learned reward function and, in turn, from the value predictions of the critic, we aim to find faster ways to adapt those functions, given the pre-trained experience. One idea would be to leverage the flexibility of successor representations and features (Barreto, 2018), which allow learning a task-agnostic estimate of the features expected under the actions of a certain actor. These predictions could then be exploited for transfer and adaptation. If we could learn useful successor features during pre-training, we might use them to solve downstream tasks faster during fine-tuning.

We aimed at improving the action selection process during fine-tuning, by employing a hybrid planner that adopts both a classical MPC-based planner and an actor-critic architecture. However, our strategy overlooks the uncertainty of the model. In order to account for this, Bayes-adaptive RL strategies could be attempted (Mehta et al., 2021).

Different approaches to learning could also be studied for this particular training setting, where the agent experiences training through two different stages: pre-training and fine-tuning. Meta-learning, or learning to learn, strategies for RL could help designing a new approach to tackle the problem end-to-end (Finn et al., 2017; Gupta et al., 2018).

References

Barreto, A. Transfer in reinforcement learning with successor features and generalised policy improvement. In *ICML*, 2018.

Barto, A. G. Intrinsically motivated learning of hierarchical collections of skills. pp. 112–119, 2004.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29, 2016.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Largescale study of curiosity-driven learning. *ICLR*, 2019a.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. Exploration by random network distillation. *ICLR*, 2019b.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations, 2020. URL https://arxiv.org/abs/2002.05709.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018. URL https://arxiv.org/abs/1805.12114.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop on Deep Learning, 2014*, 2014.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

Frank, M., Leitner, J., Stollenga, M., Förster, A., and Schmidhuber, J. Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in Neurorobotics*, 7, 2014.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *CoRR*, 2016.

Grimm, C., Barreto, A., Singh, S., and Silver, D. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:5541–5552, 2020.

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. Unsupervised meta-learning for reinforcement learning. *CoRR*, 2018.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019a.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565, 2019b.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control, 2022. URL https://arxiv.org/abs/2203.04955.

Hansen, S., Dabney, W., Barreto, A., Van de Wiele, T., Warde-Farley, D., and Mnih, V. Fast task inference with variational intrinsic successor features, 2019. URL https://arxiv.org/abs/1906.05030.

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

Kool, W., McGuire, J., Wang, G., and Botvinick, M. Neural and behavioral evidence for an intrinsic cost of self-control. *PloS one*, 8, 2013.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. arXiv:2004.14990.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. URLB: Unsupervised reinforcement learning benchmark. In *NeurIPS Datasets and Benchmarks Track (Round 2)*, 2021.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR*, 2016.

Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6736–6747, 2021a.

Liu, H. and Abbeel, P. Unsupervised active pre-training for reinforcement learning. *ICLR*, 2021b.

Mazzaglia, P., Çatal, O., Verbelen, T., and Dhoedt, B. Curiosity-driven exploration via latent bayesian surprise. *ArXiv*, abs/2104.07495, 2021.

Mehta, V., Paria, B., Schneider, J., Ermon, S., and Neiswanger, W. An experimental design perspective on model-based reinforcement learning, 2021. URL https://arxiv.org/abs/2112.05244.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *ICML*, 2016.

Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pp. 2125–2133. MIT Press, 2015.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017.

Oudeyer, P.-Y. and Kaplan, F. How can we define intrinsic motivation ? In *the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Lund University Cognitive Studies, Lund:LUCS, Brighton, 2008.

Pathak, D., Agrawal, P., Efros, A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. *ICML*, 2017.

Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. In *ICML*, 2019.

Piché, A., Thomas, V., Ibrahim, C., Bengio, Y., and Pal, C. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Rajeswar, S., Ibrahim, C., Surya, N., Golemo, F., Vazquez, D., Courville, A., and Pinheiro, P. O. Haptics-based curiosity for sparse-reward tasks. In *5th Annual Conference on Robot Learning*, 2021.

Richards, A. G. *Robust constrained model predictive control*. PhD thesis, Massachusetts Institute of Technology, 2005.

Rubinstein, R. Y. and Kroese, D. P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, pp. 230–247, 2010.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation, 2015. URL https://arxiv.org/abs/1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *ICML*, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

Singh, H., Hnizdo, V., Demchuk, A., and Misra, N. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23, 02 2003. doi: 10.1080/01966324.2003.10737616.

Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020. arXiv:2004.04136.

Talvitie, E. Learning the reward function for a misspecified model. In Dy, J. and Krause, A. (eds.), *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4838–4847. PMLR, 10–15 Jul 2018.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., and Riedmiller, M. A. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling, 2015. URL https://arxiv.org/abs/1509.01149.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. 2019.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021a. URL https://arxiv.org/abs/2107.09645.

Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=GY6-6sTvGaf.

A. Reference scores

We report the performance of the *Dreamer@2M* oracle agent, trained with environment rewards on the twelve tasks of the URL benchmark. In Figure 9, We show normalized performance to compare to the DrQ-v2's scores from Laskin et al. (2021), which can be found at: `https://github.com/rll-research/url_benchmark/issues/1`.

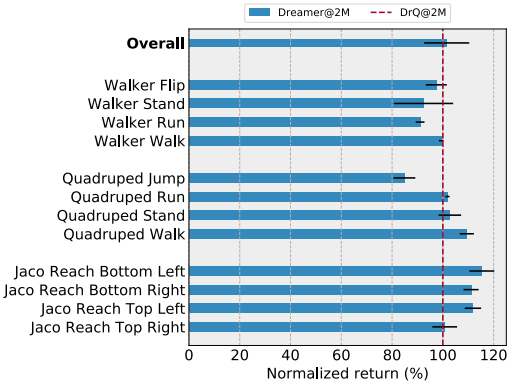


Figure 9. **Dreamer oracle normalized scores.** Dreamer scores on the URLB tasks normalized by the scores of DrQ-v2 (Laskin et al., 2021).

In Table A, we report the mean scores for DrQ-v2, used as reference scores in the URLB paper, and for Dreamer@2M, which we use to normalize returns in our work. We additionally report mean and standard deviations for the best performing baseline from URLB. which is Disagreement (Pathak et al., 2019), and our best performing method, which is LBS+TD-MPC (with no actor initialization for Jaco). We notice that the LBS+TD-MPC scores approach the Dreamer@2M's scores in several tasks, eventually outperforming them in a few tasks (e.g. Walker Flip, Quadruped Jump). We believe this merit of LBS+TD-MPC is due both to the exploration pre-training, which may have found more rewarding trajectories than greedy supervised RL optimization, and of the improved planning strategy (Hansen et al., 2022).

Pre-trainining for 2M environment steps					
Domain	Task	URLB Expert	URLB Disagreement	Dreamer@2M	LBS+TD-MPC
Walker	Flip	799	346 ± 13	778	938 ± 12
	Run	796	208 ± 15	724	596 ± 38
	Stand	984	746 ± 34	909	973 ± 14
	Walk	971	549 ± 37	965	959 ± 1
Quadruped	Jump	888	389 ± 62	753	822 ± 33
	Run	888	337 ± 30	904	642 ± 99
	Stand	920	512 ± 89	945	927 ± 28
	Walk	866	293 ± 37	947	816 ± 61
Jaco	Reach bottom left	193	124 ± 7	222	225 ± 6
	Reach bottom right	203	115 ± 10	225	221 ± 10
	Reach top left	191	106 ± 12	213	226 ± 5
	Reach top right	223	139 ± 7	224	227 ± 2

Table 2. Performance of expert baseline and the best method on pixel-based URLB from (Laskin et al., 2021) and performance of our oracle baseline (Dreamer@2M) and best approach (LBS+TD-MPC), after pre-training for 2M steps and fine-tuning for 100k steps.

B. Unsupervised Reinforcement Learning Strategies

We summarize the unsupervised RL approaches adopted in our work. For all approaches, rewards have been normalized during training using an exponential moving average with momentum 0.95.

ICM. The Intrinsic Curiosity Module (ICM; Pathak et al. (2017)) defines intrinsic rewards as the error between states projected in a feature space and a feature dynamics model's predictions. We use the Dreamer agent encoder $e_t = f_\phi(s_t)$ to obtain features and train a forward dynamics model $g(e_t|f_{t-1}, a_{t-1})$ to compute rewards as:

$$r_t^{\text{ICM}} \propto \|g(e_t|f_{t-1}, a_{t-1}) - e_t\|^2.$$

As ICM requires environment states to compute rewards, we train a reward predictor to allow estimating rewards in imagination.

Plan2Explore. The Plan2Explore algorithm (Sekar et al., 2020) is an adaptation of the Disagreement algorithm (Pathak et al., 2019) for latent dynamics models. An ensemble of forward dynamics models is trained to predict the features embedding $e_t = f_\phi(s_t)$, given the previous latent state and actions, i.e. $g(e_t|z_{t-1}, a_{t-1}, w_k)$, where w_k are the parameters of the k-th predictor. Intrinsic rewards are defined as the variance of the ensemble predictions:

$$r_t^{\text{P2E}} \propto \text{Var}(\{g(e_t|z_{t-1}, a_{t-1}, w_k)|k \in [1, ..., K]\}).$$

Plan2Explore requires only latent states and actions, thus it can be computed directly in imagination. We used an ensemble of 5 models.

RND. Random Network Distillation (RND; Burda et al. (2019b)) learns to predict the output of a randomly initialized network $n(s_t)$ that projects the states into a more compact random feature space. As the random network is not updated during training, the prediction error should diminish for already visited states. Intrinsic reward here is defined as:

$$r_t^{\text{RND}} \propto \|g(s_t) - n(s_t)\|^2$$

As RND requires environment states to compute rewards, we train a reward predictor to allow estimating rewards in imagination.

LBS. In Latent Bayesian Surprise (LBS; Mazzaglia et al. (2021)), they use the KL divergence between the posterior and the prior of a latent dynamics model as a proxy for the information gained with respect to the latent state variable, by observing new states. Rewards are computed as:

$$r_t^{\text{LBS}} \propto D_{\text{KL}}[q(z_t|z_{t-1}, a_{t-1}, e_t)\|p(z_t|z_{t-1}, a_{t-1})]$$

As LBS requires environment states to compute rewards, we train a reward predictor to allow estimating rewards in imagination.

APT. Active Pre-training (APT; Liu & Abbeel (2021b)) uses a particle-based estimator based on the K nearest-neighbors algorithm (Singh et al., 2003) to estimate entropy for a given state. We implement APT on top of the deterministic component of the latent states $\bar{z}_t$, providing rewards as:

$$r_t^{\text{APT}} \propto \sum_i^k \log \|\bar{z}_t - \bar{z}_t^i\|^2,$$

where k are the nearest-neighbors states in latent space. As APT requires only latent states, it can be computed directly in imagination. We used $k = 12$ nearest neighbors.

DIAYN. Diversity is All you need (DIAYN; Eysenbach et al. (2019)) maximizes the mutual information between the states and latent skills w. We implement DIAYN on top of the latent space of Dreamer, writing the mutual information as $I(w_t, z_t) = H(w_t) - H(w_t|z_t)$. The entropy $H(w_t)$ is kept maximal by sampling $w_t \sim p(w_t)$ from a discrete uniform

prior distribution, while $H(w_t|z_t)$ is estimated learning a discriminator $q(w_t|z_t)$. Additionally, DIAYN maximizes the entropy of the actor, so we compute intrinsic rewards as:

$$r_t^{\text{DIAYN}} \propto \log q(w_t|z_t) - \log \pi(a_t|z_t)$$

As DIAYN requires environment states and sampled skills to compute rewards, we train a reward predictor to allow estimating rewards in imagination.

C. Algorithm

Algorithm 1 Model-based Unsupervised RL

Require: Actor θ, Critic ψ, World Model ϕ
Require: Intrinsic reward r^{int}, extrinsic reward r^{ext}
Require: Environment, M, downstream tasks T_k, $k \in [1, \ldots, M]$
Require: Pre-train steps N_{PT}, fine-tune steps N_{FT}, environment steps/update τ
Require: Initial model state z_0, planner TD-MPC (Hansen et al., 2022), replay buffers $\mathcal{D}_{\text{PT}}$, $\mathcal{D}_{\text{FT}}$

for $t = 0, \ldots, N_{\text{PT}}$ **do**
  Draw action from the actor, $\mathbf{a}_t \sim \pi_\theta(a_t|z_t)$
  Apply action to the environment, $\mathbf{s}_{t+1} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)$
  Add transition to replay buffer, $\mathcal{D}_{\text{PT}} \leftarrow \mathcal{D}_{\text{PT}} \cup (\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$
  Infer model state, $z_{t+1} \sim q(z_{t+1}|z_t, a_t, f(s_{t+1}))$
  if $t \mod \tau = 0$ **then**
    Update world model parameters ϕ on the data from the replay buffer $\mathcal{D}_{\text{PT}}$
    Update actor-critic parameters $\{\theta, \psi\}$ in imagination, maximizing r^{int}
  end if
end for
Output pre-trained parameters $\{\psi_{\text{PT}}, \theta_{\text{PT}}, \phi_{\text{PT}}\}$
for $T_k \in [T_1, \ldots, T_M]$ **do**
  Initialize fine-tuning world-model with ϕ_{PT}
  (Optional) Initialize fine-tuning actor with θ_{PT}
  for $t = 0, \ldots, N_{\text{FT}}$ **do**
    Use the planner for selecting action, $\mathbf{a}_t \sim$ TD-MPC(z_t)
    Apply action to the environment, $\mathbf{s}_{t+1}, r_t^{\text{ext}} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)$
    Add transition to replay buffer, $\mathcal{D}_{\text{FT}} \leftarrow \mathcal{D}_{\text{FT}} \cup (\mathbf{s}_t, \mathbf{a}_t, r_t^{\text{ext}}, \mathbf{s}_{t+1})$
    Infer model state, $z_{t+1} \sim q(z_{t+1}|z_t, a_t, f(s_{t+1}))$
    if $t \mod \tau = 0$ **then**
      Update world model parameters ϕ on the data from the replay buffer $\mathcal{D}_{\text{FT}}$
      Update actor-critic parameters $\{\theta, \psi\}$ in imagination, maximizing r^{ext}
    end if
  end for
  Evaluate performance on T_k
end for

D. Hyperparameters

Most of the hyperparameters we used for world-model training are the same as in the original DreamerV2 work (Hafner et al., 2021). Specific details are as outline here:

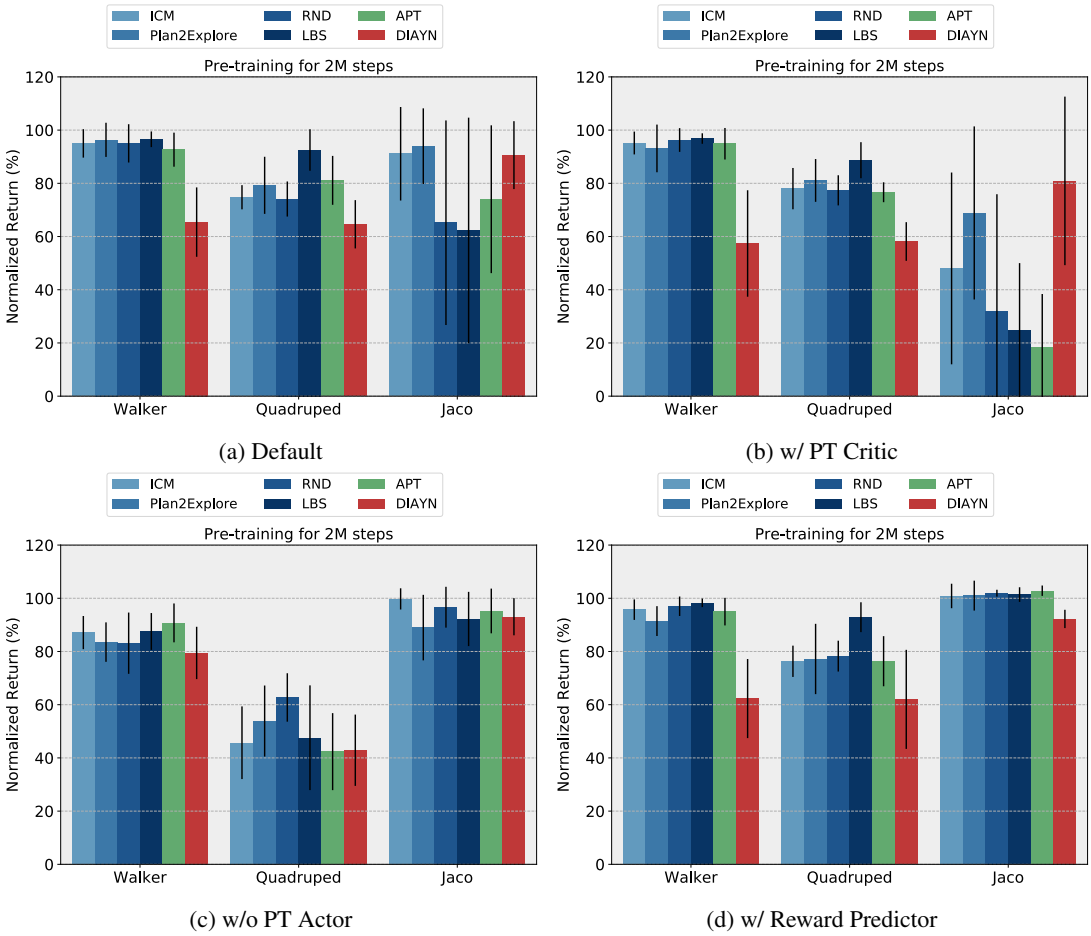
Name	Value
World Model	
Batch size	50
Sequence length	50
Discrete latent state dimension	32
Discrete latent classes	32
GRU cell dimension	200
KL free nats	1
KL balancing	0.8
Adam learning rate	$3 \cdot 10^{-4}$
Slow critic update interval	100
Actor-Critic	
Imagination horizon	15
γ parameter	0.99
λ parameter	0.95
Adam learning rate	$8 \cdot 10^{-5}$
Actor entropy loss scale	$1 \cdot 10^{-4}$
TD-MPC	
Iterations	12
Number of samples	512
Number of elite actions	64
Mixture coefficient (Actor/CEM)	0.05
Min std (fixed)	0.1
Temperature	0.5
Momentum	0.1
Horizon	5
Common	
Environment steps/update	5
MLP number of layers	4
MLP number of units	400
Hidden layers dimension	400
Adam epsilon	$1 \cdot 10^{-5}$
Weight decay	$1 \cdot 10^{-6}$
Gradient clipping	100

Table 3. World model, actor-critic, planner (TD-MPC) and common hyperparameters.

For the pure MPC-based experiments, we increased the number of MPPI samples from 512 to 1000, the number of elite actions from 64 to 100, and the horizon from 5 to 15.

E. Additional Results

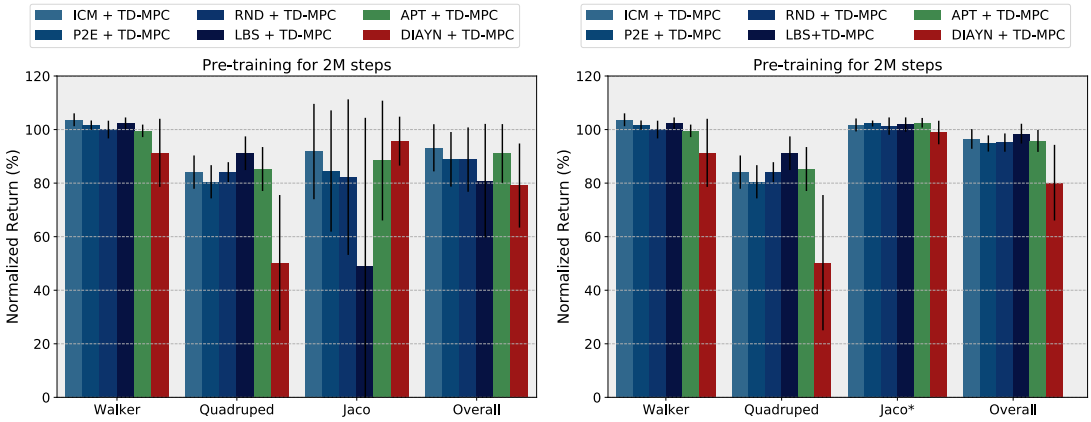
We present complete results, for each unsupervised RL method, for the experiments in Section 4, when using only the actor-critic algorithm, in Figure 10, and when also employing the TD-MPC planner, in Figure 11.



(a) Default (b) w/ PT Critic

(c) w/o PT Actor (d) w/ Reward Predictor

Figure 10. Results for all unsupervised approaches, when using actor-critic for planning.



(a) TD-MPC (b) TD-MPC (Jaco w/o PT Actor)

Figure 11. Results for all unsupervised approaches, when using TD-MPC (Hansen et al., 2022) for planning.