UNIFYING UNSUPERVISED AND OFFLINE RL FOR FAST ADAPTATION USING WORLD MODELS

Daniel Khapun
Department of Computer Science
University of Haifa
daniel.khapun@gmail.com

Dan Rosenbaum
Department of Computer Science
University of Haifa
danro@cs.haifa.ac.il

ABSTRACT

Deep reinforcement learning has proven an effective method to solve many intricate tasks, yet it still struggles with data efficiency and generalization to novel scenarios. Recent approaches to deal with this include (1) unsupervised pretraining of the agent in an environment without reward signals, and (2) training the agent using offline data coming from various possible sources. In this paper we propose to consider both of these approaches together, resulting in a setting where different types of data streams are available and fast online adaptation to new tasks is required. Towards this goal we consider the Unsupervised Reinforcement Learning Benchmark and show that access to unsupervised data is better used as a source of exploration trajectories rather than for pretraining a policy. Following this observation we develop a method based on training a world-model as a smart offline buffer of exploration data. We show that this approach outperforms previous methods and results in 10-times-faster adaptation. We then propose a setup that includes access to both unsupervised exploratory data and offline expert demonstrations when testing the agents' online performance on adaptation to novel tasks in the environment.

1 INTRODUCTION

Deep reinforcement learning (RL) has achieved remarkable success in addressing complex control tasks, yet a significant challenge persists in its ability to generalize and adapt to novel tasks. While transfer learning practices excel in simpler supervised setups, RL traditionally treats each task in isolation, hindering the utilization of knowledge from prior experiences.

This creates two related big challenges: first, this setup is extremely inefficient because of the vast compute that is necessary to train policies from scratch, when even small changes in the task requires the agent to expose itself to millions of experiences in the environment; and second, this results in brittle policies that fail when facing perturbations to the task or environment.

Two recent approaches try to deal with this problem, namely unsupervised RL and offline RL. While both approaches offer partial solutions, they often operate in distinct paradigms. In the unsupervised RL approach, an agent is first pretrained using interactions in a similar environment, but without access to a specific task and its reward function. The assumption is that this can lead to a better initialization of the network weights for the second phase, when the agent starts to receive a reward signal and uses it to finetune its weights towards a policy that solves the desired task. In the offline RL approach, the agent first acquires offline data that contains demonstrations of interactions with the environment and the task reward. The agent can then use this data in different ways to extract the optimal policy for the task at hand.

In this paper we propose to unify these two approaches. We argue that (1) it is better to treat the unsupervised phase as a method to acquire offline data rather than a method to pretrain the agent's weights, and (2) the usefulness of offline data should not be measured as a zero-shot method to solve a task, but rather by the acceleration it provides to subsequent online reinforcement learning.

Towards this goal, we propose a framework where unsupervised interaction primarily serves to gather diverse exploratory data, which is then refined and utilized alongside minimal online in-

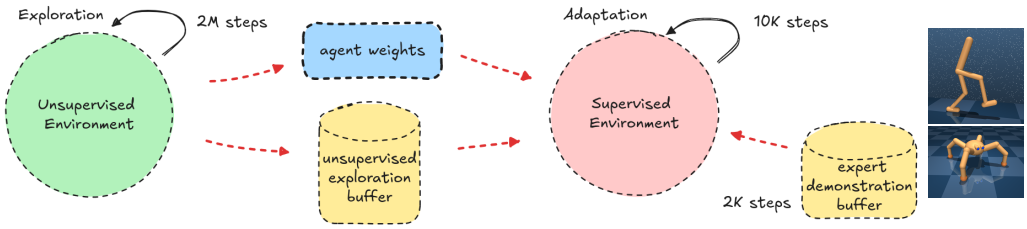

Figure 1: The proposed setup for testing fast adaptation to new tasks. The agent has access to many unsupervised interactions with the environment (e.g. 2M steps) and is then measured by its performance on a specific task in the same environment after a very small number of supervised interactions (e.g. 10K steps). We propose to use the unsupervised phase not only for pretraining the agent weights but also to populate a buffer of unsupervised exploration trajectories that can be used in the supervised adaptation phase. We also propose to potentially use an additional set of offline expert demonstrations. This setup tests the ability to combine different data sources in order to achieve 10 times faster adaptation than was tested before. Right: The two environments that we use, Walker and Quadruped (Tunyasuvunakool et al., 2020).

teraction for rapid task mastery. To achieve this we consider using world models as a method for effectively combining the different types of data streams. A recent paper (Rajeswar et al., 2023) showed that world models are effective in an unsupervised pretraining setup, achieving excellent results in the Unsupervised RL Benchmark (URLB) (Laskin et al., 2021). We claim that the reason for this is that unsupervised pretraining is best used as a source of exploratory data rather than a way to achieve better initialization of the policy weights, and that world models can serve as a smart buffer for exploratory data. By extending the method in Rajeswar et al. (2023) using an offline RL approach, we show significant improvements in adapting to novel tasks in 10 times less online steps.

Following our findings, we propose to test for faster adaptation than the standard URLB, moving from 100K to 10K steps, and we propose to extend the setup to also allow access to a few offline expert demonstrations. This results in a setup (Fig. 1) that contains unsupervised exploratory data, supervised expert demonstrations, and supervised online interactions, providing a unified test of the ability to use various data sources for fast adaptation. We believe this is a natural setup to start benchmarking, as it represents a realistic setting, e.g. in robotics, where different possible data sources are available, and extremely rapid adaptation is required.

Our results on this setup suggests that world models provide a particularly effective mechanism to fuse different data sources since they can act as sophisticated, generative buffers of experience. They not only compress vast amounts of interaction data gained during unsupervised exploration but also allow for separately dealing with the reward prediction in order to adapt to different tasks, effectively bridging the gap between diverse unsupervised data, offline expert knowledge, and efficient online adaptation. Our main contributions can be summarized as follows:

1. We show that access to an unsupervised environment is best used as a method to collect exploratory data for offline RL rather than for pretraining the weights of a policy.
2. We develop a method that outperforms all previous methods in the URLB benchmark by training a world model on unsupervised data using techniques from offline RL.
3. We propose to study extreme adaptation in a unified setup consisting of unsupervised pretraining, offline expert demonstrations, and online interactions in the environment. Towards this end we add extra expert demonstrations to the URLB tasks and show that our method can leverage this information to further improve adaptation results.

2 Preliminaries and Related Work

Unsupervised Reinforcement Learning Benchmark The Unsupervised Reinforcement Learning Benchmark (URLB) (Laskin et al., 2021) is a benchmark that compares the adaptation capabilities of different RL unsupervised exploration algorithms. The setup is separated into two stages: pretraining and finetuning. During the pretraining stage, there is no specific task required from the agent and the environment does not provide a reward signal. The agent is therefore trained using an intrinsic reward which is specified by the tested exploration method. At the second stage, the

resulted policy from the previous stage is used and finetuned to a specific task in the same environment. The underlining RL algorithm for pretraining and finetunung is DrQv2 (Yarats et al., 2021), which is a variant of DDPG (Lillicrap et al., 2019).

In the pretraining stage, the tested exploration methods are separated into knowledge-based, data-based and competence methods. Knowledge-based methods focus on maximizing the entropy of visited states. Competence methods focus on maximizing the entropy of the learned policy. Data-based methods learn an explicit skill vector by maximizing the mutual information between the encoded observation and skill. For each exploration method, the pretraining stage is performed on environments from the DeepMind Control Suit (Tunyasuvunakool et al., 2020): Walker, Quadruped and Jaco. Here we focus on Walker and Quadruped, as they present a more interesting challenge of exploration vs. fast adaptation.

In the finetuning stage, the exploration strategy used during pretraining is evaluated on set of tasks. First, the policy weights and part of the critic network are loaded from the pretraining stage. Then, the policy is trained on the task using the baseline RL algorithm DrQv2. The method is evaluated after 100K steps of finetuning, which is used as a measure of adaptation to the novel tasks.

Intrinsic Reward Models Intrinsic reward method are used to facilitate better exploration in sparse reward environments. The idea is to use an additional reward to promote the visiting of novel states. Intrinsic reward methods are specially important when dealing with unsupervised environments, such as the pretraining stage of URLB, as they form the only source of reward. In this work we focus on four leading methods for exploration as used in Laskin et al. (2020) and Rajeswar et al. (2023). The methods we consider are ICM Pathak et al. (2017), RND Burda et al. (2018), RE3 Seo et al. (2021), and Latent Bayesian Surprise (LBS) (Mazzaglia et al., 2022). The latter is used in Rajeswar et al. (2023), which introduce a world-model based method for unsupervised learning on which we build.

Offline RL In offline RL, rather than collecting data by interacting with the environment, data is collected by a different, usually unknown method, and used by the agent as a dataset to learn the policy. Data can come from expert demonstrations, historical data, or simulated environments. The aim is to reduce the need of costly interaction with the environment. For an overview of the approach see Levine et al. (2020), and for different datasets see Fu et al. (2021).

One of the main issues with offline RL relates to the inability to generalize from one setup to another. This is because the standard approach tests zero-shot generalization to the online environment, and therefore lacks a mechanism for adaptation. Several methods were developed to address this problem, such as CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021) and CRR (Wang et al., 2020). Cal-CQL (Nakamoto et al., 2024) focuses on using policies pretreined on supervised offline data to improve the training of downstream online RL. In our work, we propose to use *unsupervised* exploratory data as an offline dataset and measure its effectiveness by the acceleration it provides to subsequent learning of new tasks with online interactions, essentially measuring fast adaptation.

ExORL ExORL Yarats et al. (2022) is a different benchmark that tests ways to utilize unsupervised exploration data from the perspective of offline RL. In URLB, the setup builds on pretraining a policy and later finetuning on a specific task. In contrast, ExORL uses unsupervised exploration to collect offline data, and then, given a task in the environment, it assumes it has access to the ground truth reward function and uses it to fix the unsupervised exploration buffer. This is done without collecting any additional supervised data. Similar to ExoRL, we also show that using the unsupervised interactions as offline data is more beneficial then using the policy that was used during the unsupervised data collection stage, however we show this in a more realistic setup where there is no acess to the ground-truth reward function which needs to be estimated from supervised interactions with the environment.

World Model Pretraining A major development in model based RL is the success of learned world models, which allow simulating environment transitions, and can facilitate sample efficient training. A learned world model is a data driven model that can estimate the distribution over the next state given a history of previous transitions:

$$p_\theta(s_{t+1}, r_{t+1} | s_t, a_t, s_{t-1}, \dots). \tag{1}$$

In essence, a well trained world model can replace a complex simulation of the environment. This, in turn, allows training the agent with almost no interaction with the actual environment, when the underlining environment dynamics stay the same. Additionally, planning techniques can be used by generating possible futures from the current states.

An example of such an architecture for world models is Dreamer (Hafner et al., 2022) which uses an RNN to learn the environment dynamics in a sequential way. In addition, it learns the reward model $R(s, a)$. This allows generating partial trajectories, using some policy, and training an off-policy agent on the generated data. Recently, excellent results were achieved on the tasks tested in URLB (Rajeswar et al., 2023). The method, which we call here **Dreamer-MPC (DMPC)**, uses the Dreamer architecture as a world model, trained as a supervised next-state predictor, using data from the unsupervised RL stage collected with LBS as the exploration intrinsic reward. Additionally, during the finetuning stage, a planning strategy called Model Predictive Control (MPC) (Hansen et al., 2022) is used to generate trajectories.

In MPC, the planning policy consists of a network that predicts the value function $V_\xi(s)$ and a Gaussian action sequence distribution estimator $p(a_1, a_2, \ldots, a_t|s_0)$. For each state, several planning iterations are done, to gradually find the optimal trajectory and then the first planned action is used to proceed in the actual environment. For each planning iteration, given the sequence distribution, an expected return is estimated using the world model, and the value function. The return estimation is done in the following way: (1) the state s_0 is received from the environment; (2) an initialized Gaussian sequence distribution $p(a_1, a_2, \ldots, a_t|s_0)$ is used to generate a sequence of actions; (3) the world model $p_\theta(s_{t+1}, r_{t+1}|s_t, a_t)$ is used to predict the reward at each time step until s_t; (4) the rest of the expected return is calculated by $V_\xi(s_{t+1})$. Top sampled sequences are used to update the distribution, and another set of sequences are sampled. This process continues for a set amount of iterations. We base our method on this model but make important modifications treating the model as a smart offline buffer of exploration trajectories rather than a model pretrained on an unsupervised environment.

3 Unsupervised Pretraining as Offline Exploration

We start by considering unsupervised RL, where agents have access to an unsupervised environment without a reward signal. We base our setup on the URLB with the DeepMind Control Suite environments (Tunyasuvunakool et al., 2020), using 2M unsupervised steps in the first phase, and then testing the capacity of adaptation after 100K supervised steps. We compare the standard approach of policy pretraining to an approach that uses the unsupervised environment for data collection (Fig. 2 and Fig. 3). The standard method, as proposed in the original benchmark of URLB, is to pretrain an agent in the unsupervised environment, using various possible exploration methods, and then use the weights as initial values when finetuning the policy using a reward function for some given task. This is denoted by **Finetune** in the figures, where the performance is measured for each task after 100K supervised environment steps. While finetuning a pretrained agent leads to some acceleration in training time for some of the tasks, it is not always significantly better than initializing the weights randomly and completely discarding the output of the pretraining phase denoted as **baseline** (horizontal line).

We argue that this happens because the benefit of the pretrained agent does not necessarily come from the value of its weights, but rather from its behavior in the environment in the initial steps. In other words, fast adaptation occurs not because the policy weights are closer to their optimal values, but rather because the pretrained policy can start generating useful exploration trajectories from the first episodes.

To test this, we use a different method to exploit unsupervised exploration, where the exploration trajectories in the pretraining phase are stored in a buffer, and then used together with the online data while training the agent on a specific task. This means that the pretraining exploration is treated as offline data. However, since it comes from unsupervised exploration, the reward signal corresponds to some intrinsic exploration reward rather than the true task reward. To correct this, we implement a reward-model component, that uses the online supervised data to learn a reward predictor. In turn, the reward model can be used to predict the reward in each step of the exploratory trajectories in the pretrained buffer.

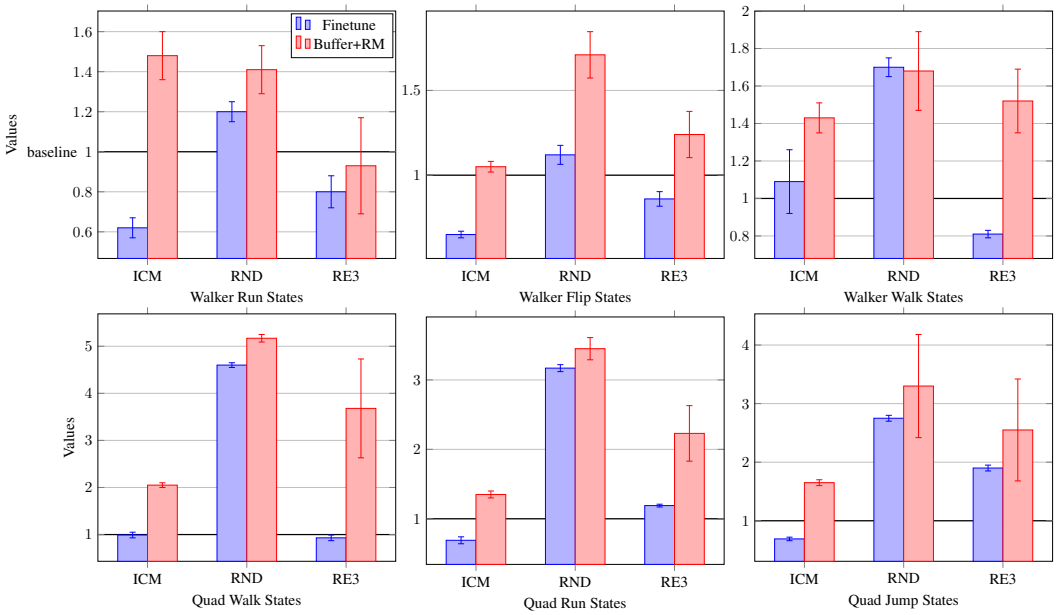

*Figure 2: Average episodic returns (over 10 episodes) on the URLB benchmark with **state** observations after 100K training steps. We show two environments and three different tasks for each and values are normalized to a baseline that does not use any pretraining information. Results show that (1) finetuning a pretrained agent (Finetune) is not always better than training from scratch (baseline); and (2) Using the unsupervised pretraining to collect offline data (Buffer+RM) leads to better performance. This is implemented by training a reward model to predict the reward of the unsupervised data and label the unsupervised trajectories.*

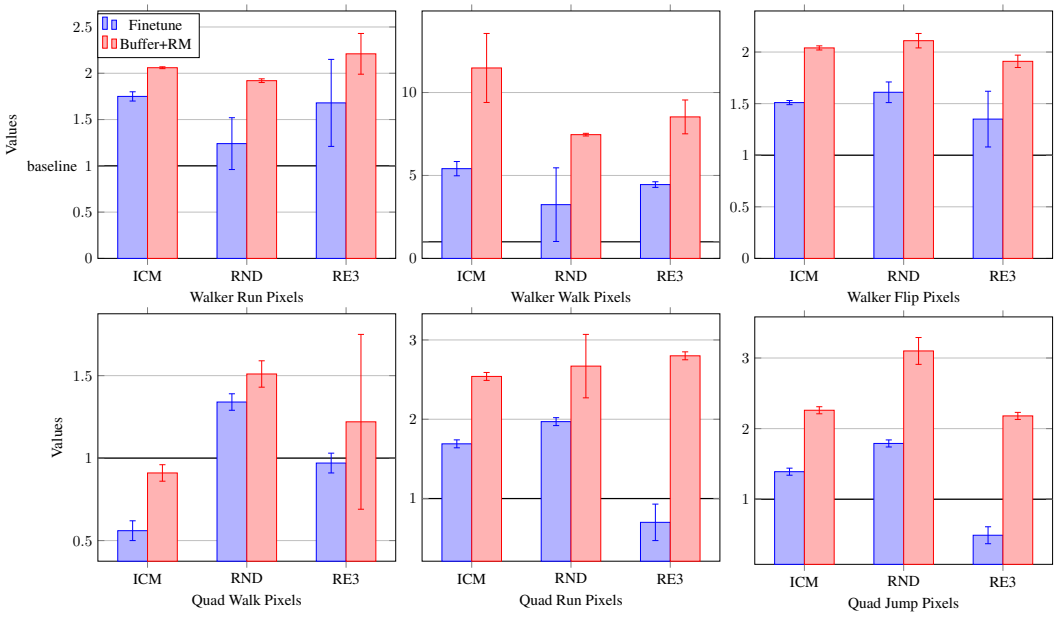

*Figure 3: Average episodic returns (over 10 episodes) on the URLB benchmark for **pixel** observations after 100K training steps (normalized to a baseline with no pretraining). The behavior is similar to state observation in Fig. 2, namely (1) finetuning a pretrained agent is not always beneficial (Finetune vs. baseline), and (2) using the unsupervised pretraining as an offline buffer where the reward signal is predicted by a learned reward model leads to better performance (Buffer+RM).*

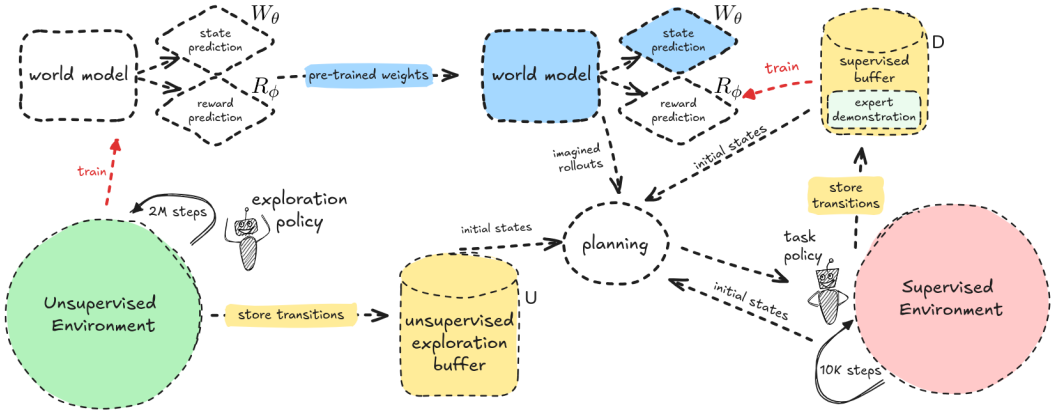

Figure 4: A diagram of the Off-DMPC method. In the first stage the world model is pretrained and an unsupervised buffer is collected. In the second stage the world model's reward head is trained on supervised data while supervised replay buffer is collected, and the model together with the unsupervised and supervised buffers are used for planning. The supervised data can potentially also contain expert demonstrations.

Implementation Details The reward model shares the architecture of the Q network used by the policy (based on DrQv2). During the supervised phase, we intermittently sample trajectories from both the unsupervised buffer and the online supervised data. When the batch comes from the supervised buffer, which contains the true task reward, it is used to train the reward model and the policy. When the data comes from the unsupervised buffer, the reward model is used to predict the task reward and the trajectory is then used to train the policy. In both cases, the trajectories, which contain either the true or a predicted reward, are used in the same way to update the policy.

Results The results in Fig. 2 and Fig. 3, corresponding to experiments in state space and pixel space respectively, show that the method based on using an unsupervised buffer and a reward model, denoted by **Buffer+RM**, outperforms simple finetuning of the agent weights. Buffer+RM achieves better results than Finetune for 17 out of the 18 experiments (3 tasks for each of the 2 environments using 3 exploration methods) performed in state space (Fig. 2) and for all the 18 experiments in pixel space (Fig. 3).

4 Off-DMPC: World Model as an Offline Buffer

Dreamer-MPC (DMPC) (Rajeswar et al., 2023) is an agent based on planning with a world model and is shown to significantly outperform all other methods in URLB. While this can be a result of the general efficiency of planning in RL, we argue that the reasons for the favorable performance of DMPC are related to the results discussed in Sec. 3, because training a world model on unsupervised exploration can be viewed as a way to store the exploration trajectories. The world model is essentially a smart buffer of the exploration trajectories that also allows generalization and planning.

Following the results of training a reward predictor and using it for the unsupervised trajectory buffer, a natural next step is to consider a full world model. While DMPC showed excellent results on the original URLB, our goal is to achieve even faster adaptation. We make modifications to the training method, based on the approach that the model should treat unsupervised exploration as offline data rather than pretraining data.

Our method, which we denote by **Offline-Dreamer-MPC (Off-DMPC)**, is depicted in Fig. 4. Similarly to DMPC, it operates in two phases - an unsupervised exploration phase and an online finetuning phase. During the unsupervised phase, Off-DMPC preforms exploration using the LBS intrinsic reward and pretrains a Dreamer world model. While pretraining the world model, we also save the collected exploration buffer to be used as a source of starting points when training in the supervised stage. In addition, we leave the reward head of the world model un-trained in the first stage, and only train it once we have access to some supervision of the reward in the second stage. During the supervised stage we finetune the reward head using the online data, and treat the exploration data in

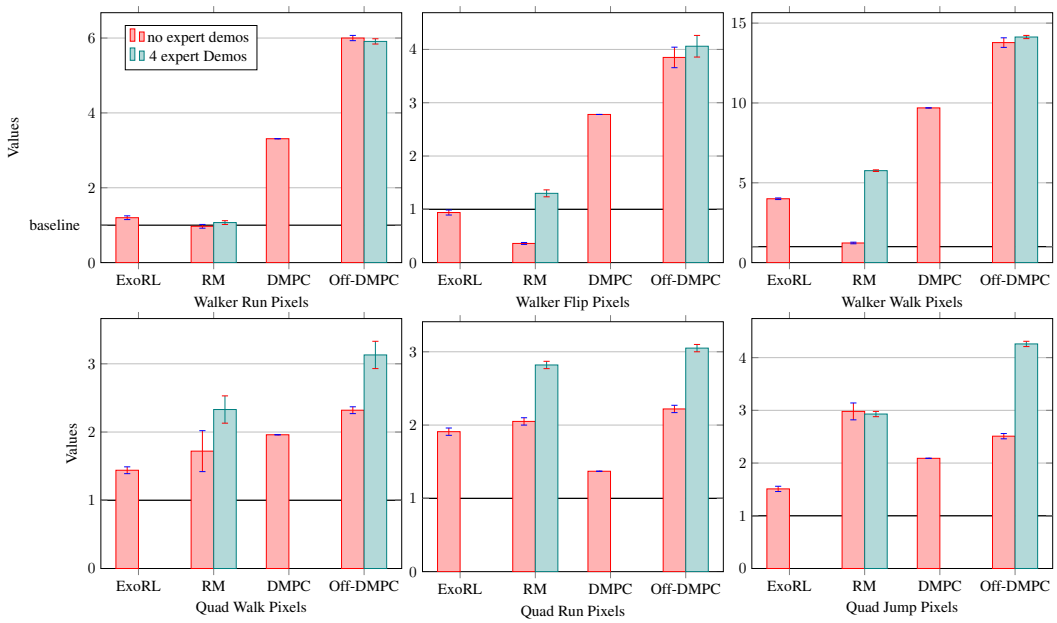

Figure 5: Average episodic return (over 10 episodes) on URLB tasks. ExORL uses a GT reward function and no online data, RM, DMPC and Off-DMPC are trained on 10K online steps with/without 2K expert demonstrations, and all results are normalized relative to a baseline that was trained on 100K online steps. The results show: (1) 10x faster adaptation is possible (2) The method based on a world model significantly outperforms other methods. (3) Using expert data is not always beneficial, but can lead to significant improvements.

conjunction with the learned world model as auxiliary sources of offline exploration data. Using the unsupervised exploration data we can provide much more diverse starting points to initiate the planning trajectories using the world model. As a result, we can make more parameter updates relative to steps in the environment and continue optimization after we finished collecting supervised data, utilizing the large unsupervised exploration dataset.

The online finetuning stage is described in Algorithm 1. After pretraining the DMPC in the unsupervised stage using an exploration reward we use the following components: an unsupervised buffer U consisting of trajectories from the first stage, a pretrained state transition model W_θ, an untrained reward model R_ϕ, a pretrained exploration actor policy π_ψ and an untrained value model (critic) V_ξ. Then, N_{seed} steps of exploration in the online environment are performed to collect an initial set of trajectories stored in the supervised replay buffer D. These transitions are collected using the pretrained exploration policy π_ψ. Afterwords a mixed training process is performed using online and offline data. For each update, we first perform a number of steps in the environment to collect more supervised data into the replay buffer D. We then sample a batch of trajectories either from the unsupervised buffer U or the supervised buffer D. When the batch comes from D we use it to update the weights of W_θ and R_ϕ. When the batch comes from U we only update W_θ. In either case, the batch is used as a source of states s which are fed as initial states to sample trajectories from W_θ and R_ϕ in order to train the critique V_ξ. The probability of using a batch of unsupervised trajectories from U decreases based on a counter c_u and as more online data is collected in D. After acting a total of N_{steps} steps, the training process continues without collecting more online trajectories from the environment.

In comparison to the original DMPC that uses the unsupervised phase to pretrain the world model, the modifications we make are: (1) Collect an exploration buffer during pretraining (similar to the method described in Sec. 3) and use it in the finetuning stage to extract diverse starting points for the MPC planner; (2) Train the reward head of the world model only in the finetuning stage; (3) keep making offline updates after the online steps collection is finished; (4) make more weight updates relative to steps in the environment by decreasing the step-per-updates ratio from 10 to 5; and (5) add finetuning updates to the world model on the online episodes.

Table 1: Average episodic returns after 10K supervised training steps on URLB tasks. Off-DMPC significantly outperforms other methods and effectively combines unsupervised data and expert demonstration.

	no expert demo.				expert demo.	
Task/Method	ExORL (GT reward)	RM	DMPC	Off-DMPC	RM	Off-DMPC
Walker Walk	260	238	630	**896**	375	**918**
Walker Run	121	96	328	**594**	106	**585**
Walker Flip	207	236	607	**841**	285	**889**
Quadruped Walk	250	298	339	**401**	402	**540**
Quadruped Run	279	299	200	**324**	412	**446**
Quadruped Jump	324	**640**	449	539	630	**915**

Figure 6: A visual demonstration of fast adaptation compared to previous methods in Walker.

Implementation Details We implement the method described in Algorithm 1 using a total of 10K steps in the environment, where 4K of them are collected into the buffer as an initial seed before making any updates. After collecting the seed steps we perform 5 steps per update, and after 10K environment steps we continue to update the model based on the collected data for a total of 100K steps. In summary: $N_{seed} = 4K, N_{steps} = 10K, N_{updates} = 100K$ and $N_s/u = 5$.

Results Table 1 (left block of methods) and Fig. 5 (in red) show results of Off-DMPC compared to other models in a fast adaptation setup. We report episode rewards after 20 episodes consisting of 10K supervised steps for two different environments, and three different tasks in each. Using 10K online steps is a test of faster adaptation than the original URLB which reports results after 100K steps. This is because in this setup, adaptation after 100K steps is already solved with the vanilla DMPC. To test the effectiveness of a world model as a method to consolidate offline exploration data for fast adaptation we also compare to the method we described in Sec. 3 that collects an unsupervised buffer and corrects the reward with a reward model. We denote it here by **RM** and use the same settings as used for Off-DMPC (i.e. the same number of steps, seed steps, updates and step-per-updates). In addition we compare to ExoRL Yarats et al. (2022) which is similar to RM but assumes ground truth access to the task reward function and does not finetune using the online interactions. In order to run ExoRL which was originally implemented for state observation we adapat it to image observations. Both RM and ExoRL are methods that use an offline buffer rather than a world model. For a fair comparison all methods are based on the LBS exploration in the unsupervised stage. The results in Fig. 5 are normalized relative to the same baseline used in Sec. 3, namely a vanilla DrQv2 agent trained on 100K supervised environment steps. Note that results of RM can seem lower than in Fig. 3 because now we test the RM performance after 10K rather than 100K supervised steps.

Our method Off-DMPC significantly outperforms the vanilla world model DMPC for all tasks and achieves a new state-of-the-art in fast adaptation. Off-DMPC also outperforms the offline buffer methods - RM and ExoRL, where the only exception is on the Quadruped Jump task compared to RM. These results highlight the potential for faster adaptation than previously tested and suggest that training and using a world model as a smart buffer for offline data is an effective way to consolidate offline exploration data with online interactions.

5 ADDING EXPERT DEMONSTRATIONS

Motivated by the results and the points discussed above, we propose a new setup to test fast adaptation to novel tasks. In this setup we are interested in leveraging different sources of information in order to achieve even better results for fast adaptation after 10K steps. Towards this end we propose to use an additional small number of expert demonstrations that can be used together with the unsupervised data and online supervised data. Our setup can be summarized as follows:

1. The agent has access to a very large number of unsupervised interaction with the environment. This is similar to the URLB setup, however this can be used either for pretraining an agent, populating a buffer, or training a world model.
2. The agent has access to a few expert trajectories, containing the task's reward. This data can serve two purposes. First, it can be used as initial information about the task, e.g to train a reward predictor. Second, it can disentangle the problem of exploration in the environment, as using this data is somewhat equivalent to a successful exploration that ensures the important states in the environment were covered.
3. Given the above data, the agent is assessed on its performance after a small amount of supervised online interactions in the environment, measuring its capacity for fast adaptation.

Specifically, we use 2M unsupervised exploration steps and 2K supervised expert demonstration steps (4 episodes), and then measure the performance of the agent after 10K online supervised steps. Compared to URLB, the only addition are 4 episodes of supervised expert demonstration that we provide for each task. These episodes are achieved by training the DreamerMPC model on a supervised environment for 2M steps. We use 4 random episodes from the last 100 training episodes.

To test the effect of expert data on fast adaptation we compare to the results from previous sections. The results can be seen in Table 1 and Fig. 5 (in green). To make a fair comparison with the same number of supervised steps, we compare methods that use 10K online steps and no expert data with methods that use 8K online steps and 2K offline expert data. This comparison evaluates the benefit of replacing 2K steps from the online interactions with expert demonstrations. We implement this in Algorithm 1 by pre-populating the supervised replay buffer D with the expert demonstrations and reducing 2K from N_{seed} and N_{steps}. As an additional comparison we perform the same change for the RM model, replacing initial 2K steps with expert demonstrations.

Results Comparing between methods with and without access to expert data demonstrates its potential benefit for fast adaptation. The results show that in some cases the improvement is only minor and for other cases it is substantial. A potential reason for this is that in cases where the exploration covered sufficient states to solve the task, further expert demonstration is not needed. The results also show that Off-DMPC outperforms RM, suggesting that our method of using world models as a smart offline exploration buffer provides an effective mechanism to consolidate various types of data sources for fast adaptation, namely unsupervised exploration of the environment, offline expert demonstrations and supervised online interactions. Fig. 6 visualizes the resulting fast adaptation compared to previous methods.

DISCUSSION

In this paper we proposed to use a method based on world models to combine two approaches to fast adaptation in RL, namely unsupervised RL and offline RL. We argued that unsupervised pretraining in RL is best used as a source of exploration data and demonstrated how world models provide an effective way to distill the exploration trajectories. We presented a method that achieved adaptation to new tasks using 10 times less supervised steps than previously tested in URLB. Finally, we proposed to consider a fast adaptation setup that consists of both unlimited unsupervised access to the environment and a small amount of supervised expert demonstrations. We believe this setup is an interesting extension to standard benchmarks providing a realistic setting such as in robotics, where various sources of data can be used and extremely fast adaptation is required.

ACKNOWLEDGMENTS

The authors would like to thank Vlad Mnih for valuable discussions throughout the research.

REFERENCES

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation, October 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models, February 2022.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal Difference Learning for Model Predictive Control, July 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement Learning with Augmented Data, November 2020.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised Reinforcement Learning Benchmark, October 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, July 2019.

Pietro Mazzaglia, Ozan Catal, Tim Verbelen, and Bart Dhoedt. Curiosity-Driven Exploration via Latent Bayesian Surprise, February 2022.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction, May 2017.

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the Unsupervised Reinforcement Learning Benchmark from Pixels, May 2023.

Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State Entropy Maximization with Random Encoders for Efficient Exploration, June 2021.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning, July 2021.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't Change the Algorithm, Change the Data: Exploratory Data for Offline Reinforcement Learning, April 2022.

A IMPLEMENTATION

Algorithm 1 Finetuning With Offline Dreamer MPC

Require: Unsupervised data buffer U, supervised replay buffer D (possibly containing expert demonstrations, see Sec. 5)
Require: Dreamer world model containing:
1. pretrained transition model W_θ
2. untrained reward head R_ϕ
3. pretrained exploration policy π_ψ
4. untrained critic V_ξ

Require: number of seed steps N_{seed}, online supervised steps N_{steps}, updates $N_{updates}$ and steps-per-updates $N_{s/u}$
Require: MPC planning horizon H, number of trajectory samples K

Step counter $c_{step} \leftarrow 0$
Unsupervised batch counter $c_u \leftarrow 0$
for $t \leftarrow 1$ to N_{seed} **do** ▷ collect seed data using exploration policy
  $a \leftarrow \pi_\psi(s)$ ▷ choose action according to pretrained exploration policy
  $s', r \leftarrow P(\cdot|s, a)$ ▷ execute action in online environment
  $D \leftarrow D \cup (s, a, s', r)$ ▷ add to replay buffer
  $c_{step} \leftarrow c_{step} + 1$
end for

for $t \leftarrow 1$ to $N_{updates}$ **do**

  if $c_{step} < N_{steps}$ **then** ▷ collect new data
    for $i \leftarrow 1$ to $N_{s/u}$ **do**
      $a \leftarrow MPC(s)$ ▷ choose action based on MPC starting from current online state
      $s', r \leftarrow P(\cdot|s, a)$ ▷ execute action in online environment
      $D \leftarrow D \cup (s, a, s', r)$ ▷ add to replay buffer
      $c_{step} \leftarrow c_{step} + 1$
    end for
  end if

  Sample batch b from U or D with prob.=$\left(\frac{|U|}{|U|+|D|+c_u}, \frac{|D|+c_u}{|U|+|D|+c_u}\right)$
  $(s, a, s', r) = b$
  if $b \subset D$ **then** ▷ we sampled from the supervised replay buffer
    Update W_θ and R_ϕ using (s, a, s', r)
  else if $b \subset U$ **then** ▷ we sampled from the unsupervised data
    Update W_θ using (s, a, s')
    $c_u \leftarrow c_u + 1$
  end if
  Simulate trajectories using MPC starting from s
  Compute rewards for trajectories using R_ϕ
  Compute returns: $J^k = \sum_{h=1}^{H} \gamma^h r_h^k + \gamma^H V_\xi(\hat{s}_H^k)$
  Update V_ξ using TD learning on simulated trajectories
end for