Reward-Free Curricula for Training Robust World Models

Marc Rigter
University of Oxford

Minqi Jiang
University College London

Ingmar Posner
University of Oxford

Abstract

There has been a recent surge of interest in developing *generally-capable agents* that can adapt to new tasks without additional training in the environment. Learning *world models* from *reward-free* exploration is a promising approach, and enables policies to be trained using imagined experience for new tasks. Achieving a general agent requires *robustness* across different environments. However, different environments may require different amounts of data to learn a suitable world model. In this work, we address the problem of efficiently learning robust world models in the reward-free setting. As a measure of robustness, we consider the *minimax regret* objective. We show that the minimax regret objective can be connected to minimising the maximum error in the world model across environments. This informs our algorithm, *WAKER: Weighted Acquisition of Knowledge across Environments for Robustness*. WAKER selects environments for data collection based on the estimated error of the world model for each environment. Our experiments demonstrate that WAKER outperforms naïve domain randomisation, resulting in improved robustness, efficiency, and generalisation.

1 Introduction

Deep reinforcement learning (RL) has been successful on a number of challenging domains, such as Atari [41], Go [66], DOTA [7] and Starcraft [71]. While these domains are difficult, they are narrow in the sense that they require solving a single pre-specified task. More recently, there has been a surge of interest in developing *generally-capable agents* that can master many tasks and quickly adapt to new tasks without any additional training in the environment [9, 33, 40, 51, 63, 67, 29, 23].

With the goal of developing generalist agents that are not specialised for a single task, we consider the *reward-free* setting. In this setting, the agent first accumulates useful information about the environment in an initial, reward-free exploration phase. Afterwards, the agent is presented with specific tasks corresponding to arbitrary reward functions, and must quickly adapt to these tasks by utilising the information previously acquired during the reward-free exploration phase. For example, consider a kitchen robot that has obtained knowledge about possible behaviours in a kitchen. We could utilise this robot for a variety of downstream tasks specified by the user, such as cooking, cleaning, etc. By separating the learning of useful environment representations into an initial pre-training phase, reward-free learning provides a powerful strategy for data-efficient RL.

A promising line of work in the reward-free setting involves learning *world models* [18], a form of model-based RL [68], where the agent learns a predictive model of the environment. In the reward-free setting, the world model is trained without access to a reward function, and instead, is trained using data collected by a suitable exploration policy [63]. Once a world model has been trained for

Correspondence to mrigter@robots.ox.ac.uk.

an environment, it is possible to train policies entirely within the world model (i.e. "in imagination") for new tasks corresponding to specific reward functions within that environment [63, 74].

However, to realise the vision of a general agent, it is not only necessary for the agent to be able to learn multiple tasks within a single environment: the agent must also be robust to different environments. Consider again the kitchen robot. To be generally useful, the robot must be able to adapt to different kitchens, with different layouts, objects, and appearances. To enable this, one approach would be to apply *domain randomisation* (DR) [70] to sample different environments uniformly at random to gather a more diverse dataset. However, the amount of data required to learn a suitable world model might vary by environment: It may require less data to understand a kitchen with a simple layout and few objects than a cluttered kitchen with a complex layout. *Unsupervised Environment Design* (UED) [13] seeks to present the optimal environments to the agent at each point of training, with the aim of maximising the robustness of the final agent across a wide range of environments. However, existing UED approaches require a task-specific reward function during exploration [14, 38, 47, 13, 26, 39, 44, 72], and therefore cannot be directly applied in the reward-free setting that we consider. A core contribution of this work is proposing an objective for UED that is not tied to a task-specific reward function.

In this work, we first analyse the fundamental problem of learning a robust world model in the reward-free setting, such that the world model is robust across a wide range of environments and downstream tasks. We then operationalise these insights in the form of novel algorithms for robust, reward-free world model learning. Inspired by past works on UED with known reward functions [13, 26, 44], we consider robustness in terms of the minimax regret [58]. In summary, we make the following key contributions: a) First, we formally define the problem of learning a robust world model in the reward-free setting, in terms of minimax regret optimality, b) We then prove that this problem is equivalent to minimising the maximum expected error of the world model across all environments under a suitable exploration policy, and finally c) We introduce WAKER, an algorithm for actively sampling environments for exploration during reward-free training based on the estimated error of the world model in each environment. We introduce new pixel-based continuous control domains and tasks for benchmarking generalisation in the reward-free setting. We evaluate WAKER on these domains, by pairing it with both an instrinsically-motivated exploration policy and a random exploration policy. Our results show that WAKER outperforms naïve domain randomisation, producing significantly more performant and robust policies that generalise better to out-of-distribution (OOD) environments.

2 Preliminaries

A *reward-free* Markov Decision Process (MDP) is defined by $\mathcal{M} = \{S, A, T, \gamma\}$, where S is the set of states and A is the set of actions. $T : S \times A \to \Delta(S)$ is the transition function, where $\Delta(X)$ denotes the set of possible distributions over X, and γ is the discount factor. For some reward function, $R : S \times A \to [0, 1]$, we write $\mathcal{M}^R$ to denote the corresponding (standard) MDP [49] with reward function R.

A reward-free Partially Observable Markov Decision Process (POMDP) is an extension of a reward-free MDP and is defined by the tuple $\mathcal{P} = \{S, A, O, T, I, \gamma\}$, where O is the set of observations, and $I : S \to \Delta(O)$ is the observation function [30]. A *history* is a sequence of observations and actions, $h = o_0, a_0, \ldots, o_t, a_t$, where $o_i \in O$ and $a_i \in A$. We write $\mathcal{H}$ to denote the set of all possible histories. Analogous to the MDP case, we write $\mathcal{P}^R$ to denote a POMDP with reward function R.

In this work, we assume that there are many possible instantiations of the environment. To model an underspecified environment, we consider a reward-free Underspecified POMDP (UPOMDP): $\mathcal{U} = \{S, A, O, T_\Theta, I, \gamma, \Theta\}$ [13]. In contrast to a POMDP, the UPOMDP additionally includes a set of free parameters of the environment, Θ. Furthermore, the transition function depends on the setting of the environment parameters, i.e. $T_\Theta : S \times A \times \Theta \to \Delta(S)$. For each episode, the environment parameters are set to a specific value $\theta \in \Theta$. Therefore, for each episode the environment can be represented by a standard POMDP $\mathcal{P}_\theta = \{S, A, O, T_\theta, I, \gamma\}$, where $T_\theta(s, a) = T_\Theta(s, a, \theta)$.

World Models Model-based RL algorithms use the experience gathered by an agent to learn a model of the environment [68, 10, 25]. When the observations are high-dimensional, it is beneficial to learn a compact latent representation of the state, and predict the environment dynamics in this latent space. Furthermore, in partially observable environments where the optimal action depends on the history of observations and actions [30], recurrent neural networks can be used to encode the history into

a Markovian representation [59, 31]. In this work, we consider a *world model* to be a model that utilises a recurrent module to predict environment dynamics in a Markovian latent space [18, 22].

Let the environment be some reward-free POMDP, $\mathcal{P}$. A world model, W, can be thought of as consisting of two parts: $W = \{q, \widehat{T}\}$. The first part is the representation model $q : \mathcal{H} \to Z$, which encodes the history into a compact Markovian latent representation $z \in Z$. The second part is the latent transition dynamics model, $\widehat{T} : Z \times A \to \Delta(Z)$, which predicts the dynamics in latent space. Because the latent dynamics are Markovian, we can think of the world model is approximating the original reward-free POMDP $\mathcal{P}$ with a reward-free MDP in latent space: $\widehat{\mathcal{M}} = (Z, A, \widehat{T}, \gamma)$.

Minimax Regret In *robust* optimisation [6] there is a set of possible scenarios, each defined by parameters $\theta \in \Theta$. The goal is to find a solution that achieves strong performance across all scenarios. In the context of reinforcement learning, we can think of each scenario as a possible instantiation of the environment, $\mathcal{P}_\theta$. For some reward function, R, the expected value of a policy in environment $\mathcal{P}^R_\theta$ is $V(\pi, \mathcal{P}^R_\theta) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi, \mathcal{P}^R_\theta]$, where r_t are the rewards received by executing π in $\mathcal{P}^R_\theta$.

One objective for robust optimisation is *minimax regret* [58]. Minimax regret is commonly used for robust policy optimisation in reinforcement learning [13, 26, 44, 53, 52, 73]. To define the minimax regret objective, we begin by defining the optimal policy for a given environment and reward function, $\pi^*_{\theta,R} = \text{argmax}_\pi V(\pi, \mathcal{P}^R_\theta)$. The regret for some arbitrary policy π in environment $\mathcal{P}^R_\theta$ is

$$\text{REGRET}(\pi, \mathcal{P}^R_\theta) := V(\pi^*_{\theta,R}, \mathcal{P}^R_\theta) - V(\pi, \mathcal{P}^R_\theta). \tag{1}$$

The minimax regret objective finds the policy with the lowest regret across *all* environments:

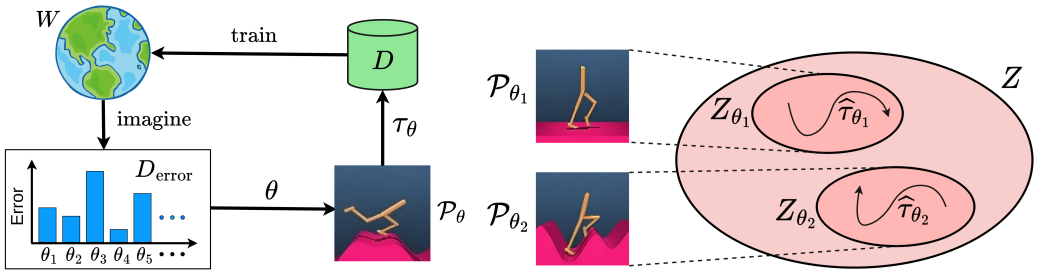
$$\pi^*_{\text{regret}} = \underset{\pi}{\text{argmin}} \max_{\theta \in \Theta} \text{REGRET}(\pi, \mathcal{P}^R_\theta). \tag{2}$$

Minimax regret aims to find a policy that is *near-optimal* in all environments, and is therefore robust.

3 Approach

The minimax regret objective defines how to optimise a *policy* to be robust to different environments *when the task is known*. However, our aim in this work is to train a world model such that policies *derived from the world model* for *future tasks* are robust to different environments. In this section, we present our approach for gathering data to train a robust world model to achieve this aim.

In Section 3.1, we outline how we learn a single world model for many possible environments. In Section 3.2 we define the Reward-Free Minimax Regret objective, which connects minimax regret to world model training by assuming that when a reward function is provided, the optimal policy in the world model for that reward function can be found. We then show that we can optimise an upper bound on this objective by minimising the maximum expected latent dynamics error in the world model across all environments, under a suitable exploration policy. This informs our algorithm for selecting environments to sample data from to train the world model, *WAKER: Weighted Acquisition*



(a) WAKER overview. (b) Illustration of world model latent space for a UPOMDP.

Figure 1: a) WAKER uses error estimates for each environment to choose the next environment to sample data from, $\mathcal{P}_\theta$. A trajectory τ_θ is collected by rolling out exploration policy π^{expl} in the selected environment. τ_θ is added to the data buffer D which is used to train the world model, W. Imagined trajectories in W are used to update the error estimates. b) In the world model, each environment is encoded to a subset, Z_θ, of the latent space Z by representation model q. To generate imagined trajectories, $\widehat{\tau}_\theta$, we use the latent dynamics model $\widehat{T}$, which predicts the dynamics for all of Z, and therefore all environments.

of Knowledge across Environments for Robustness, presented in Section 3.4. WAKER biases sampling towards environments where the world model is estimated to have the highest errors (Figure 1a).

3.1 World Models for Underspecified POMDPs

We utilise a single world model, $W = \{q, \widehat{T}\}$, to model the reward-free UPOMDP. Consider any parameter setting of the UPOMDP, $\theta \in \Theta$, and the corresponding reward-free POMDP, $\mathcal{P}_\theta$. For any history in $\mathcal{P}_\theta$, $h \in \mathcal{H}_\theta$, the representation model encodes this into a subset of the latent space, i.e. $q : \mathcal{H}_\theta \to Z_\theta$, where $Z_\theta \subset Z$. We can then use the latent dynamics model $\widehat{T}$ to predict the latent dynamics in Z_θ, corresponding to the dynamics of $\mathcal{P}_\theta$ for any $\theta \in \Theta$. Thus, we think of the world model as representing the set of reward-free POMDPs in the UPOMDP by a set of reward-free MDPs, each with their own latent state space: $\widehat{\mathcal{M}}_\theta = \{Z_\theta, A, \widehat{T}, \gamma\}$. This is illustrated in Figure 1b.

Using a single world model across all environment parameter settings is a natural approach as it a) utilises the recurrent module to infer the parameter setting (which may be partially observable), and b) enables generalisation between similar parameter settings. Futhermore, it is sufficient to train a *single generalist policy* over the entire latent space Z to obtain a policy for all environments.

3.2 Reward-Free Minimax Regret: Problem Definition

As discussed in Section 2, we can define robust policy optimisation via the minimax regret objective. However, this definition cannot be directly applied to our setting where we do not know the reward function during exploration, and our goal is to train a world model. In this section, we present the first contribution of this work, the Reward-Free Minimax Regret problem, which adapts the minimax regret objective to the setting of reward-free world model training that we address.

Consider some world model, W, which as discussed in Section 3.1 represents the possible environments by a set of reward-free MDPs in latent space, $\{\widehat{\mathcal{M}}_\theta\}_{\theta \in \Theta}$. Assume that after training W we are given some reward function, R. We define $\widehat{\pi}^*_{\theta,R}$ to be the *optimal world model policy* for that reward function R and parameter setting θ, i.e.

$$\widehat{\pi}^*_{\theta,R} = \underset{\pi}{\operatorname{argmax}}\, V(\pi, \widehat{\mathcal{M}}^R_\theta). \tag{3}$$

This is the optimal policy according the MDP defined in the latent space of the world model for parameter setting θ and reward function R, and does not necessarily correspond to the optimal policy in the real environment. From here onwards, we will make the following assumption.

Assumption 1 *Consider some world model W that defines a set of MDPs in latent space $\{\widehat{\mathcal{M}}_\theta\}_{\theta \in \Theta}$. Assume that given any reward function R, and parameter setting $\theta \in \Theta$, we can find $\widehat{\pi}^*_{\theta,R}$.*

Assumption 1 is reasonable because we can generate unlimited synthetic training data in the world model for any parameter setting, and we can use this data to find a policy that is near-optimal in the world model using RL. We now define the Reward-Free Minimax Regret problem.

Problem 1 (Reward-Free Minimax Regret) *Consider some UPOMDP, $\mathcal{U}$, with parameter set Θ. For world model W, and $\theta \in \Theta$, let $\widehat{\mathcal{M}}^R_\theta$ be the latent-space MDP defined by W, that represents real environment $\mathcal{P}_\theta$ with reward function R. Define the optimal world model policy as $\widehat{\pi}^*_{\theta,R} = \operatorname{argmax}_\pi V(\pi, \widehat{\mathcal{M}}^R_\theta)$. Find the world model, W^*, that minimises the maximum regret of the optimal world model policy across all parameter settings and reward functions:*

$$W^* = \underset{W}{\operatorname{argmin}} \max_{\theta,R} \textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_\theta) \tag{4}$$

Problem 1 differs from the standard minimax regret objective in Equation 2 in two ways. First, Problem 1 optimises a world model (not a policy) under the assumption that the optimal world model policy can later be recovered (i.e. Assumption 1). Second, the maximum is over all possible reward functions in addition to parameter settings. This makes Problem 1 suitable for the reward-free setting: we do not need access to a specific reward function when training the world model.

3.3 Theoretical Motivation

Before presenting our algorithm for Problem 1, we first provide the motivation for our approach. We make the assumption that the world model learns a suitable representation model, q, which encodes any sequence of observations and actions in the UPOMDP into a Markovian latent state.

Assumption 2 *Consider the representation model learnt by the world model, $q : \mathcal{H}_\theta \to Z_\theta$, for all $\theta \in \Theta$. Assume that given the representation model q, there exists a latent transition dynamics function, T, that models the real environment exactly, in the sense that $V(\pi, \mathcal{P}_\theta^R) = V(\pi, \mathcal{M}_\theta^R)$, where $\mathcal{M}_\theta^R = (Z_\theta, A, R, T, \gamma)$, for any policy π, reward function R and parameter setting θ.*

Assumption 2 states that the representation model successfully encodes any sequence of observations and actions into a Markovian latent state. Therefore, there exists a dynamics function T defined over the latent space that exactly models the real environment. Assumption 2 allows us to reason about the inaccuracy of the world model solely in terms of the difference between the learnt latent dynamics function, $\widehat{T}$, and the (unknown) exact latent dynamics, T. For this reason, from here onwards we will *solely refer to the latent dynamics function $\widehat{T}$* when discussing the world model, under the implicit assumption that a suitable representation model q is learnt according to Assumption 2.

Using Assumption 2 we can bound the sub-optimality of the optimal world model policy for any parameter setting θ according the difference between the learnt latent dynamics function, $\widehat{T}$, and the true latent dynamics, T, in latent MDP $\widehat{\mathcal{M}}_\theta$. This is stated formally in Proposition 1.

Proposition 1 *Let $\widehat{T}$ be the learnt latent dynamics in the world model. Assume the existence of a representation model q that adheres to Assumption 2, and let T be the true latent dynamics according to Assumption 2. Then, for any parameter setting θ and reward function R, the regret of the optimal world model policy is bounded according to:*

$$\textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_\theta) \leq \frac{2\gamma}{(1-\gamma)^2}\Big[\mathbb{E}_{z,a\sim d(\pi^*_{\theta,R},\widehat{\mathcal{M}}_\theta)}\big[\mathrm{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\big] + \mathbb{E}_{z,a\sim d(\widehat{\pi}^*_{\theta,R},\widehat{\mathcal{M}}_\theta)}\big[\mathrm{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\big]\Big]$$

where $d(\pi, \mathcal{M})$ denotes the state-action distribution of π in MDP $\mathcal{M}$, and $\mathrm{TV}(P, Q)$ is the total variation distance between distributions P and Q.

Proof Sketch: This can be proven by applying a version of the Simulation Lemma [32] twice. The full proof is provided in Appendix A.

Proposition 1 tells us that the optimal world model policy will have low regret if $\widehat{T}$ is accurate under the latent state-action distribution of both $\pi^*_{\theta,R}$ and $\widehat{\pi}^*_{\theta,R}$ in $\widehat{\mathcal{M}}_\theta$. However, during data collection we do not have access to the reward function. Therefore, we do not know these distributions as the state-action distribution induced by both $\pi^*_{\theta,R}$, and $\widehat{\pi}^*_{\theta,R}$ depends upon the reward function.

To alleviate this issue, we define an exploration policy, π^{expl}_θ, that maximises the expected error (in terms of total variation distance) of the latent dynamics model:

$$\pi^{\text{expl}}_\theta = \operatorname*{argmax}_\pi \mathbb{E}_{z,a\sim d(\pi,\widehat{\mathcal{M}}_\theta)}\Big[\mathrm{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\Big]. \tag{5}$$

This allows us to write an upper bound on the regret that has no dependence on the reward function:

$$\textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_\theta) \leq \frac{4\gamma}{(1-\gamma)^2}\mathbb{E}_{z,a\sim d(\pi^{\text{expl}}_\theta,\widehat{\mathcal{M}}_\theta)}\Big[\mathrm{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\Big] \text{ for all } R. \tag{6}$$

Therefore, we can upper bound the objective of Problem 1 by the maximum expected TV error in the latent dynamics function over all parameter settings:

$$\max_{\theta,R} \textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_\theta) \leq \max_\theta \frac{4\gamma}{(1-\gamma)^2}\mathbb{E}_{z,a\sim d(\pi^{\text{expl}}_\theta,\widehat{\mathcal{M}}_\theta)}\Big[\mathrm{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\Big]. \tag{7}$$

We now formally state the Minimax World Model Error problem, which proposes to optimise this upper bound as an approximation to the Reward-Free Minimax Regret objective in Problem 1.

Problem 2 (Minimax World Model Error) *Consider some UPOMDP, $\mathcal{U}$, with parameter set Θ, and world model latent dynamics function $\widehat{T}$. Let T be the true latent dynamics function according to Assumption 2. Define the world model error for some parameter setting θ as:*

$$\textsc{WorldModelError}(\widehat{T}, \theta) = \mathbb{E}_{z,a \sim d(\pi_\theta^{\text{expl}}, \widehat{\mathcal{M}}_\theta)} \Big[\text{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big) \Big] \tag{8}$$

Find the world model that minimises the maximum world model error across all parameter settings:

$$T^* = \operatorname*{argmin}_{\widehat{T}} \max_{\theta \in \Theta} \textsc{WorldModelError}(\widehat{T}, \theta) \tag{9}$$

Problem 2 optimises an upper bound on our original objective in Problem 1 (see Equation 7). Problem 2 finds a world model that has low prediction error across all environments under an exploration policy that seeks out the maximum error. This ensures that for any future reward function, the optimal world model policy will be near-optimal for all environments. In the next section, we present our algorithm for selecting environments to gather data for world model training, with the aim of minimising the maximum world model error across all environments, per Problem 2.

3.4 Weighted Acquisition of Knowledge across Environments for Robustness (WAKER)

Overview In this section, we address how to select environments to collect data for world model training, so that the world model approximately solves Problem 2. We cannot directly evaluate the world model error in Equation 8, as it requires knowing the true latent dynamics function, T. Therefore, following works on offline RL [37, 75, 34], we use the disagreement between an ensemble of neural networks as an estimate of the total variation distance between the learnt and true latent transition dynamics functions in Equation 8. This enables us to generate an estimate of the world model error for each environment. Then, when sampling environments, our algorithm (WAKER) biases sampling towards the environments that are estimated to have the highest error. By gathering more data for the environments with the highest estimated error, WAKER improves the world model on those environments, and therefore WAKER reduces the maximum world model error across environments as required by Problem 2.

WAKER is presented in Algorithm 1, and illustrated in Figure 1a. We train a single exploration policy over the entire latent space to approximately optimise the exploration objective in Equation 5 across all environments. Therefore, we refer to the exploration policy simply as π^{expl}, dropping the dependence on θ. We maintain a buffer D_{error} of the error estimate for each parameter setting θ that we have collected data for. To choose the environment for the next episode of exploration, with probability p_{DR} we sample θ using domain randomisation (Line 5). This ensures that we will eventually sample all environments. With probability $1 - p_{\text{DR}}$, we sample θ from a Boltzmann distribution, where the input to the Boltzmann distribution is the error estimate for each environment in D_{error} (Line 7). Once the environment has been selected we sample a trajectory, τ_θ, by rolling out π^{expl} in the environment (Line 8). We add τ_θ to the data buffer D, and perform supervised learning on D (Line 10) to update the world model.

Algorithm 1 Weighted Acquisition of Knowledge across Environments for Robustness (WAKER)

1: **Inputs:** UPOMDP with free parameters Θ; Boltzmann temperature η; Imagination horizon h;
2: **Initialise:** data buffer D; error buffer D_{error}; world model $W = \{q, \widehat{T}\}$; exploration policy π^{expl}
3: **while** training world model **do**
4: **if** $p \sim U_{[0,1]} < p_{\text{DR}}$ or D.`is_empty`() **then**
5: $\theta \sim$ `DomainRandomisation`(Θ)
6: **else**
7: $\theta \sim$ `Boltzmann`(`Normalize`$(D_{\text{error}}), \eta)$
8: $\tau_\theta \leftarrow$ rollout of π^{expl} in $\mathcal{P}_\theta$ *Collect real trajectory for θ*
9: Add τ_θ to D
10: Train W on D
11: $\pi^{\text{expl}}, D_{\text{error}} \leftarrow$ `Imagine`(D, D_{error}, W, π^{expl})
12: **function** `Imagine`(D, D_{error}, W, π^{expl})
13: **for** $i = 1, \ldots, K$ **do**
14: $\tau_\theta \leftarrow D$.`sample`() *Sample real trajectory*
15: $Z_{\tau_\theta} = \{z_t\}_{t=0}^{|\tau_\theta|} \leftarrow q(\tau_\theta)$ *Encode latent states*
16: $\widehat{\tau}_\theta \leftarrow$ rollout π^{expl} for h steps from $z \in Z_{\tau_\theta}$ in W
17: $\delta_\theta \leftarrow$ Error estimate for θ via Eq. 10 on $\widehat{\tau}_\theta$
18: Update D_{error} with δ_θ for θ
19: Train π^{expl} on $\widehat{\tau}_\theta$
20: **return** π^{expl}, D_{error}

`Imagine` updates π^{expl} and D_{error} using imagined rollouts (Line 11). For each imagined rollout, we first sample real trajectory $\tau_\theta \in D$ (Line 14). We encode τ_θ into latent states Z_{τ_θ} (Line 15). We then generate an imagined trajectory, $\widehat{\tau}_\theta$, by rolling out π^{expl} using $\widehat{T}$ starting from an initial latent state $z \in Z_{\tau_\theta}$ (Line 16). Thus, $\widehat{\tau}_\theta$ corresponds to an imagined trajectory in the environment with parameter

setting θ, as illustrated in Figure 1b. We wish to estimate the world model error for environment parameter setting θ from this imagined rollout. Following previous works [40, 55, 63], we learn an ensemble of N latent dynamics models: $\{\widehat{T}_i\}_{i=1}^N$. Like [75, 34, 37], we use the disagreement between the ensemble means as an approximation to the world model TV error for parameters θ:

$$\textsc{WorldModelError}(\widehat{T}, \theta) \approx \mathbb{E}_{z,a \sim d(\pi_\theta^{\text{expl}}, \widehat{\mathcal{M}}_\theta)} \left[\text{Var} \left\{ \mathbb{E}[\widehat{T}_i(\cdot|z,a)] \right\}_{i=1}^N \right] \tag{10}$$

We compute the error estimate using the latent states and actions in $\widehat{\tau}_\theta$ in Line 17. We use this to update our estimate of the world model error for environment θ in D_{error} (Line 18). Optionally, we may also use the imagined rollout to update the exploration policy in Line 19. For the world model architecture, we DreamerV2 [22]. More implementation details are in Appendix C.

Error Estimate Update Function We consider two possibilities for updating the error estimate for each θ in D_{error} on Line 18: 1) in WAKER-M, D_{error} maintains a smoothed average of the *magnitude* of the error estimate for each θ; 2) In WAKER-R, we update D_{error} to maintain a smoothed average of the *rate of reduction* in the error estimate for each θ. Thus, WAKER-M biases sampling towards environments that have highest error, while WAKER-R biases sampling towards environments that have the highest rate of reduction in error. More details are in Appendix C.

Exploration Policy We must learn an exploration policy π^{expl} that approximately maximises the world model error according to Equation 5. As a default, we use Plan2Explore [63], and train the exploration policy to maximise the approximation in Equation 10. To test whether our approach is agnostic to the exploration policy used, we also consider a uniform random exploration policy.

Zero-Shot Task Adaptation Once the world model has been trained, we use it to derive a task-specific policy without any further data collection. For reward function R, we find a single task policy $\widehat{\pi}_R^*$ that is defined over the entire latent space Z, and therefore all environments. We use the same approach as [63]: we label the data in D with the associated rewards and use this data to train a reward predictor. Then, we train the task policy in the world model to optimise the expected reward according to the reward predictor. In our experiments, all task policies are trained in this manner.

4 Experiments

We seek to answer the following questions: a) Does WAKER enable more robust policies to be trained in the world model? b) Does the performance of WAKER depend on the exploration policy used? c) Does WAKER lead to stronger generalisation to out-of-distribution environments? Code to reproduce our experiments is available at github.com/marc-rigter/waker.

Methods Compared To answer question a), we compare WAKER-M and WAKER-R with domain randomisation (DR), which samples uniformly from the default environment distribution. To our knowledge, there are no other baselines that are directly applicable to our reward-free setting. For both WAKER variants, we set $p_{\text{DR}} = 0.2$ and perform limited tuning of the Boltzmann temperature η. More details are in Appendix D.3. To investigate question b), we pair each algorithm with two different exploration policies: Plan2Explore [63] or a random exploration policy.

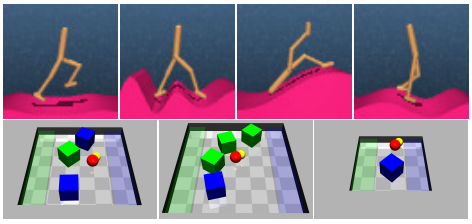


Figure 2: Examples of training environments. Top: Terrain Walker. Bottom: Clean Up.

Domains and Tasks For *Terrain Walker* we simulate Walker from the DeepMind Control Suite [69] on procedurally generated terrain. For each environment, there are two parameters (amplitude and length scale) that control the terrain generation. For this domain, we evaluate 5 downstream tasks: *walk*, *run*, *stand*, *flip*, and *walk-backward*. The *Clean Up* domain is based on SafetyGym [50] and consists of a robot and blocks that can be pushed. For *Clean Up*, there are three environment parameters that vary: the environment size, the number of blocks, and the block colours. The downstream tasks (*sort*, *push*, and *sort-reverse*) each correspond to different goal locations for each colour of block. For both domains, the agent receives image observations. Examples of training environments for each domain are shown in Figure 2. More details are in Appendix D.1.

Evaluation Our goal is to train the world model such that the policies obtained from the world model are robust, as measured by the minimax regret objective. However, we cannot directly evaluate the regret of a policy as it requires knowing the true optimal performance. Therefore, following [26, 55]

we evaluate conditional value at risk (CVaR) [56] to measure robustness. For confidence level α, CVaR_α is the average performance on the worst α-fraction of runs. We evaluate $\text{CVaR}_{0.1}$ by evaluating each policy on 100 environments sampled uniformly at random, and reporting the average of the worst 10 runs. We also report the average performance over all 100 runs in Appendix B.2.

To answer question c), we additionally evaluate the policies on out-of-distribution environments. For *Terrain Walker*, the out-of-distribution (OOD) environments are terrain with a length scale 25% shorter than seen in training (*Terrain Walker Steep*), and terrain containing stairs, in contrast to the smooth terrain seen in training (*Terrain Walker Stairs*). For *Clean Up*, the out-of-distribution environments contain one more block than was ever seen in training (*Clean Up Extra Block*).

Results Presentation Due to space limitations, in Tables 1-2 we present results for task policies obtained from the *final world model* at the end of six days of training. For each exploration policy, we highlight results within 2% of the best score, and $\pm$ indicates the S.D. over 5 seeds. *Learning curves* for the performance of policies obtained from snapshots of the world model are in Appendix B.3. Accompanying each table, we present 95% confidence intervals of the probability that algorithm X obtains improved performance over algorithm Y, computed using the aggregated results across all tasks with the *rliable* [1] framework.

Results Table 1 presents the results for the robustness evaluation. We find that for both exploration policies, WAKER-M outperforms DR across almost all tasks. For the Plan2Explore exploration policy, WAKER-R also outperforms DR. Figure 3 shows that these improvements over DR are statistically significant, as the lower bound on the probability of improvement is greater than 0.5. Both WAKER variants result in significant improvements over DR when using Plan2Explore, suggesting that WAKER is more effective when combined with a sophisticated exploration policy. Between WAKER-M and WAKER-R, WAKER-M (which prioritises sampling the most uncertain environments) obtains stronger performance. This is expected from our analysis in Section 3.3, which shows that minimising the maximum world model error across environments leads to improved robustness. Regardless of the environment selection method, Plan2Explore leads to much stronger performance than random exploration, verifying previous findings [63]. The results for average performance in Appendix B.2 show that WAKER achieves similar or improved performance on average compared to DR. These results demonstrate WAKER *improves robustness at no cost* to average performance.

Exploration Policy		Plan2Explore			Random Exploration		
Environment	Sampling	WAKER-M	WAKER-R	DR	WAKER-M	WAKER-R	DR
Clean Up	Sort	0.711 ± 0.09	0.643 ± 0.07	0.397 ± 0.12	0.007 ± 0.01	0.010 ± 0.01	0.000 ± 0.0
	Sort-Rev.	0.741 ± 0.06	0.586 ± 0.06	0.395 ± 0.10	0.000 ± 0.0	0.000 ± 0.0	0.000 ± 0.0
	push	0.716 ± 0.11	0. 702 ± 0.08	0.590 ± 0.093	0.124 ± 0.09	0.058 ± 0.06	0.023 ± 0.03
Terrain Walker	Walk	818.0 ± 15.3	805.3 ± 42.0	748.9 ± 39.5	243.9 ± 26.7	224.8 ± 41.9	224.3 ± 25.3
	Run	312.6 ± 19.9	303.0 ± 16.1	279.9 ± 18.1	120.4 ± 14.7	104.2 ± 9.7	114.1 ± 12.2
	Flip	955.0 ± 11.9	937.7 ± 10.5	936.1 ± 10.2	878.9 ± 18.4	850.7 ± 40.5	849.9 ± 27.4
	Stand	941.2 ± 12.3	945.4 ± 16.6	936.5 ± 17.5	585.1 ± 31.8	581.5 ± 68.8	591.3 ± 65.5
	Walk-Back.	752.5 ± 24.8	722.1 ± 33.5	729.6 ± 39.2	369.9 ± 13.1	311.5 ± 49.8	311.2 ± 48.4

Table 1: Robustness evaluation: $\text{CVaR}_{0.1}$ from evaluating policies on 100 randomly sampled environments and averaging the performance on the 10 worst runs.

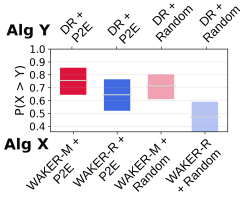


Figure 3: Robustness evaluation aggregated CIs.

Exploration Policy		Plan2Explore			Random Exploration		
Environment	Sampling	WAKER-M	WAKER-R	DR	WAKER-M	WAKER-R	DR
Clean Up Extra Block	Sort	0.499 ± 0.04	0.486 ± 0.02	0.289 ± 0.05	0.143 ± 0.03	0.107 ± 0.03	0.071 ± 0.03
	Sort-Rev.	0.524 ± 0.05	0.437 ± 0.05	0.295 ± 0.06	0.140 ± 0.04	0.110 ± 0.04	0.066 ± 0.05
	Push	0.567 ± 0.02	0.515 ± 0.01	0.402 ± 0.05	0.223 ± 0.05	0.180 ± 0.03	0.141 ± 0.04
Terrain Walker Steep	Walk	631.4 ± 20.8	583.8 ± 51.9	512.7 ± 31.2	212.3 ± 20.2	190.3 ± 36.1	185.1 ± 26.5
	Run	228.6 ± 12.8	213.2 ± 17.2	186.8 ± 9.1	119.3 ± 9.9	103.4 ± 10.0	105.2 ± 15.4
	Flip	946.3 ± 15.1	950.3 ± 8.0	929.1 ± 23.8	891.5 ± 14.8	817.0 ± 59.0	885.4 ± 29.6
	Stand	940.8 ± 15.2	934.9 ± 21.2	919.4 ± 23.1	690.6 ± 29.6	612.5 ± 57.5	625.5 ± 81.4
	Walk-Back.	554.0 ± 21.5	547.5 ± 39.4	465.9 ± 18.9	328.3 ± 18.0	213.7 ± 72.7	203.3 ± 33.5
Terrain Walker Stairs	Walk	686.6 ± 38.4	668.4 ± 45.7	611.7 ± 75.2	224.2 ± 27.3	217.3 ± 23.7	219.9 ± 17.0
	Run	241.2 ± 16.4	235.6 ± 14.1	216.1 ± 24.3	128.2 ± 6.0	114.5 ± 11.0	119.5 ± 9.7
	Flip	935.9 ± 15.9	939.3 ± 8.8	927.6 ± 8.4	892.0 ± 15.1	840.6 ± 39.7	822.5 ± 101.6
	Stand	972.8 ± 6.8	971.8 ± 7.6	970.2 ± 7.6	697.7 ± 41.8	665.8 ± 72.2	668.1 ± 72.1
	Walk-Back.	492.1 ± 55.1	548.1 ± 51.3	388.7 ± 34.6	313.6 ± 5.4	257.1 ± 21.3	242.6 ± 24.8

Table 2: Out-of-distribution evaluation: average performance on OOD environments.

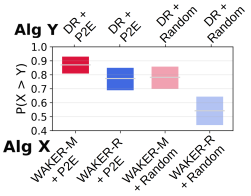
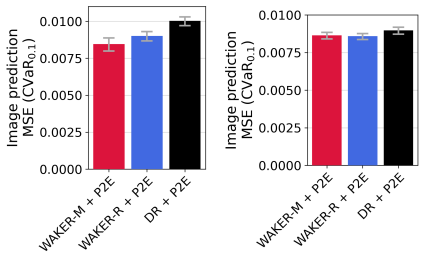


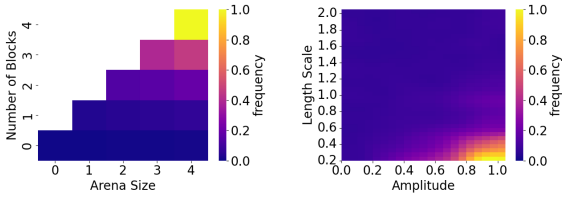
Figure 4: OOD evaluation aggregated CIs.

To assess the quality of the world models learnt, we evaluate the error between the transitions predicted by the world model and real transitions. To evaluate robustness, we compute the error on the worst 10% of trajectories on randomly sampled environments. These results are shown in Figure 5 for world models trained using the Plan2Explore exploration policy. In Figure 5, we observe that WAKER leads to lower prediction errors on the worst 10% of trajectories than DR. This verifies that by biasing sampling towards environments with higher error estimates, WAKER reduces the worst-case errors in the world model. This also suggests that these improvements to the world model



(a) Clean Up (b) Terrain Walker

Figure 5: Evaluation of $\text{CVaR}_{0.1}$ of the world model image prediction error.



(a) Clean Up (b) Terrain Walker

Figure 6: Heatmaps of environment parameters sampled by WAKER-M + P2E.

lead to the improved policy robustness that we observe in Table 1, as expected from the upper bound in Equation 7. Similar results are presented in Appendix B.1 for world models trained using the random exploration policy, showing that WAKER leads to improvement in world model errors for the Clean Up domain, and comparable errors in Terrain Walker. This echoes the results in Table 1, which indicate that WAKER is more effective when combined with the Plan2Explore exploration policy.

Figure 6 illustrates the frequency that environment parameters are sampled by WAKER-M + P2E. We observe that WAKER skews sampling towards environments that are intuitively more complex. In Terrain Walker, WAKER tends to sample environments where the terrain has a high amplitude and short length scale, corresponding to environments with the steepest terrain. In Clean Up, WAKER tends to sample larger arenas with more blocks in them. This is consistent with the intuition that to learn a robust world model, it is necessary to sample more data from more complex environments.

Table 2 and Figure 4 present the evaluation on OOD environments. WAKER-M results in a considerable performance improvement over DR for both exploration policies, and WAKER-R significantly improves performance when the exploration policy is Plan2Explore. Thus, WAKER-M again obtains better performance than WAKER-R, but both variants of our algorithm outperform DR. These results demonstrate that by actively sampling more uncertain environments for exploration, WAKER leads to world models that are able to generalise more broadly to environments *never seen during training*. This indicates that WAKER is a meaningful step towards developing more generally-capable agents.

5 Related Work

Our work focuses on training more robust world-models [18, 20, 21, 22, 25, 54, 62, 76] in the reward-free setting. Past works improve data collection by training an exploration policy within the world model (i.e. in imagination) [63, 4, 40, 74]. Unlike prior methods, WAKER augments action-based exploration by directly exploring over the space of environments via an autocurriculum. To our knowledge, this is the first work to address curricula for reward-free training of a world model.

WAKER can thus be viewed as an *automatic curriculum learning* (ACL) [17, 48, 42] method for training a world model. ACL methods optimise the order in which tasks or environments are presented during training, to improve the agent's final performance [3, 16, 15, 40, 14, 38, 47]. *Unsupervised environment design* (UED) [13] refers to a class of methods that generate curricula to produce more robust policies, and facilitate transfer to environment distributions differing from that used for training. UED typically frames curriculum learning as a zero-sum game between a teacher and a student, where the teacher actively selects environments to optimise an adversarial objective. When this objective is the student's regret in each environment, the student provably follows a minimax regret policy at the Nash equilibria of the resulting game, making minimax regret UED methods a principled approach for learning robust policies [13, 26]. Domain randomisation (DR) [24, 57, 70] samples environments uniformly, and can be viewed as the simplest approach to UED, where the teacher assigns equal utility to all environments. Extensions of DR such as Active DR [39, 2], which upweights environments that induce divergent behaviors in the task policy, can be seen as rudimentary forms of UED.

A particularly simple and effective form of UED is Prioritised Level Replay (PLR) [28, 26], which selectively *revisits* initially randomly-sampled environments that have higher estimated "learning potential", as measured by the average temporal difference (TD) error incurred in each environment. Extensions of PLR have been proposed, improving on its random search [44] and addressing general

issues with UED in stochastic settings [27]. However, these UED methods all *assume that the reward function is known during curriculum learning* and therefore cannot be directly applied to our reward-free setting. Nevertheless, we draw inspiration from PLR by considering replay-based curricula as a means to induce more robust world models that are learned *without task-specific rewards*.

WAKER uses ideas from *intrinsic motivation* [12, 61] as a bridge between UED and the reward-free setting. By providing agents with *intrinsic rewards*, such methods enable agents to efficiently explore complex environments in the absence of extrinsic rewards. Intrinsic motivation generalises active learning [11, 64] to sequential decision-making, by rewarding agents for visiting states that: a) have high uncertainty [19, 45, 60, 4, 63, 65], b) can be maximally influenced [8, 35], or c) have rarely been visited [5, 43]. Our work extends uncertainty-based intrinsic motivation beyond states within a single environment to the space of environment configurations, allowing novelty-related metrics to be used in place of typical UED objectives that require a task-specific reward function. WAKER thus highlights the fundamental connection between autocurricula and exploration.

6 Conclusion

We have introduced a formalisation for the problem of learning a robust world model in the reward-free setting. Building on prior works in robust RL, we considered robustness in terms of minimax regret with respect to the optimal world model policy across all possible environments and reward functions. We then derived a connection between this measure of regret for a given world model and the maximum error of its latent dynamics model across environments. We operationalised this insight in the form of WAKER, which trains a world model in the reward-free setting by selectively sampling the environment settings that induce the highest latent dynamics error. WAKER can thus be seen as an adversarial environment design method for training minimax-regret optimal world models over a given design space of environments (and their possible reward functions). In several pixel-based continuous control domains, we demonstrated that compared to domain randomisation, WAKER drastically improves the zero-shot task adaptation capabilities of the world model in terms of both robustness and generalisation. Policies trained for downstream tasks inside the learned world model exhibit significantly improved performance on out-of-distribution environments that were never encountered during world model training. WAKER therefore represents a meaningful step towards developing more generally-capable agents.

Given the strong empirical performance and theoretical underpinnings of WAKER, we are excited to scale our approach to even more complex domains, as well as investigate the use of more powerful generative methods to produce regret-maximising environments.

Acknowledgements

This work was supported by a Programme Grant from the Engineering and Physical Sciences Research Council (EP/V000748/1) and a gift from Amazon Web Services. Additionally, this project made use of the Tier 2 HPC facility JADE2, funded by the Engineering and Physical Sciences Research Council (EP/T022205/1).

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Process Systems*, 34:29304–29320, 2021.

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

[3] Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine learning*, 23:279–303, 1996.

[4] Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In *International Conference on Machine Learning*, pages 591–601. PMLR, 2020.

[5] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Process Systems*, 29, 2016.

[6] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

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

[8] Homanga Bharadhwaj, Mohammad Babaeizadeh, Dumitru Erhan, and Sergey Levine. Information prioritization through empowerment in visual model-based RL. *International Conference on Learning Representations*, 2022.

[9] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2022.

[10] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Process Systems*, 31, 2018.

[11] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[12] Edward L Deci and Richard M Ryan. *Intrinsic motivation and self-determination in human behavior*. Springer Science & Business Media, 1985.

[13] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in Neural Information Process Systems*, 33:13049–13061, 2020.

[14] Theresa Eimer, André Biedenkapp, Frank Hutter, and Marius Lindauer. Self-paced context evaluation for contextual reinforcement learning. In *International Conference on Machine Learning*, pages 2948–2958. PMLR, 2021.

[15] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1515–1528. PMLR, 2018.

[16] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495. PMLR, 2017.

[17] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[18] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in Neural Information Process Systems*, 31, 2018.

[19] Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. *Advances in Neural Information Process Systems*, 31, 2018.

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

[21] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[22] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. *International Conference on Learning Representations*, 2021.

[23] Marcus Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2004.

[24] Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2):325–368, 1997.

[25] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Process Systems*, 32, 2019.

[26] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

[27] Minqi Jiang, Michael D Dennis, Jack Parker-Holder, Andrei Lupu, Heinrich Kuttler, Edward Grefenstette, Tim Rocktäschel, and Jakob Nicolaus Foerster. Grounding aleatoric uncertainty for unsupervised environment design. In *Advances in Neural Information Processing Systems*.

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

[29] Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking exploration. *arXiv preprint arXiv:2211.07819*, 2022.

[30] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[31] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *International Conference on Learning Representations*, 2017.

[32] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49:209–232, 2002.

[33] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

[34] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOREL: Model-based offline reinforcement learning. *Advances in Neural Information Process Systems*, 33:21810–21823, 2020.

[35] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135. IEEE, 2005.

[36] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

[37] Cong Lu, Philip J Ball, Jack Parker-Holder, Michael A Osborne, and Stephen J Roberts. Revisiting design choices in offline model-based reinforcement learning. *International Conference on Learning Representations*, 2022.

[38] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, 2019.

[39] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[40] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[42] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

[43] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, pages 2721–2730. PMLR, 2017.

[44] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022.

[45] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

[46] Ken Perlin. Improving noise. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 681–682, 2002.

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

[48] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. In *IJCAI 2020-International Joint Conference on Artificial Intelligence*, 2021.

[49] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[50] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

[51] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

[52] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1127–1133, 2010.

[53] Marc Rigter, Bruno Lacerda, and Nick Hawes. Minimax regret optimisation for robust planning in uncertain Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11930–11938, 2021.

[54] Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: Robust adversarial model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

[55] Marc Rigter, Bruno Lacerda, and Nick Hawes. One risk to rule them all: Addressing distributional shift in offline reinforcement learning via risk-aversion. *arXiv preprint arXiv:2212.00124*, 2023.

[56] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[57] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. *Robotics: Science and Systems*, 2017.

[58] Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 46(253):55–67, 1951.

[59] Jürgen Schmidhuber. Reinforcement learning in Markovian and non-Markovian environments. *Advances in Neural Information Process Systems*, 3, 1990.

[60] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the International Conference on Simulation of Adaptive Behavior*, pages 222–227, 1991.

[61] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[62] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[63] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

[64] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[65] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pages 5779–5788. PMLR, 2019.

[66] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[67] Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

[68] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

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

[70] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30. IEEE, 2017.

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

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

[73] Huan Xu and Shie Mannor. Parametric regret in uncertain markov decision processes. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3606–3613. IEEE, 2009.

[74] Yingchen Xu, Jack Parker-Holder, Aldo Pacchiano, Philip Ball, Oleh Rybkin, S Roberts, Tim Rocktäschel, and Edward Grefenstette. Learning general world models in a handful of reward-free deployments. *Advances in Neural Information Processing Systems*, 35:26820–26838, 2022.

[75] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[76] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. SOLAR: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 7444–7453. PMLR, 2019.

A Proof of Proposition 1

We begin by stating and proving a version of the classic Simulation Lemma from [32].

Lemma 1 (Simulation Lemma [32]) *Consider two infinite horizon MDPs, $\mathcal{M}^R = \{S, A, R, T, \gamma\}$ and $\widehat{\mathcal{M}}^R = \{S, A, R, \widehat{T}, \gamma\}$, with reward function $R : S \times A \to [0, 1]$. Consider any stochastic policy $\pi : S \to \Delta(A)$. Then:*

$$\left|V(\pi, \widehat{\mathcal{M}}^R) - V(\pi, \mathcal{M}^R)\right| \leq \frac{2\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d(\pi,\widehat{\mathcal{M}}^R)}\left[\mathrm{TV}\big(\widehat{T}(\cdot|s,a), T(\cdot|s,a)\big)\right] \tag{11}$$

where $d(\pi, \widehat{\mathcal{M}}^R)$ is the state-action distribution of π in $\widehat{\mathcal{M}}^R$, and $\mathrm{TV}(P, Q)$ is the total variation distance between two distributions P and Q.

Proof: To simplify the notation of the proof, we will use the notation $V^\pi = V(\pi, \mathcal{M}^R)$ and $\widehat{V}^\pi = V(\pi, \widehat{\mathcal{M}}^R)$.

Using the Bellman equation for $\widehat{V}^\pi$ and V^π, we have:

$$\begin{aligned}
\widehat{V}^\pi(s_0) - V^\pi(s_0) &= \mathbb{E}_{a_0\sim\pi(s_0)}\Big[R(s_0,a_0) + \gamma\mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}\widehat{V}^\pi(s')\Big] \\
&\qquad - \mathbb{E}_{a_0\sim\pi(s_0)}\Big[R(s_0,a_0) + \gamma\mathbb{E}_{s'\sim T(s_0,a_0)}V^\pi(s')\Big] \\
&= \gamma\mathbb{E}_{a_0\sim\pi(s_0)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}\widehat{V}^\pi(s') - \mathbb{E}_{s'\sim T(s_0,a_0)}V^\pi(s')\Big] \\
&= \gamma\mathbb{E}_{a_0\sim\pi(s_0)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}\widehat{V}^\pi(s') - \mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}V^\pi(s') \\
&\qquad + \mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}V^\pi(s') - \mathbb{E}_{s'\sim T(s_0,a_0)}V^\pi(s')\Big] \\
&= \gamma\mathbb{E}_{a_0\sim\pi(s_0)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}\widehat{V}^\pi(s') - \mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}V^\pi(s')\Big] \\
&\qquad + \gamma\mathbb{E}_{a_0\sim\pi(s_0)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}V^\pi(s') - \mathbb{E}_{s'\sim T(s_0,a_0)}V^\pi(s')\Big]
\end{aligned} \tag{12}$$

We define $\widehat{P}_1^\pi(s_1|s_0) = \int_{a\in A}\pi(a|s_0)\widehat{T}(s_1|s_0,a)\mathrm{d}a$. This allows us to rewrite the first term from the last line as:

$$\gamma\mathbb{E}_{a_0\sim\pi(s_0)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}\widehat{V}^\pi(s') - \mathbb{E}_{s'\sim\widehat{T}(s_0,a_0)}V^\pi(s')\Big] = \gamma\mathbb{E}_{s_1\sim\widehat{P}_1^\pi(\cdot|s_0)}\Big[\widehat{V}^\pi(s_1) - V^\pi(s_1)\Big]$$

We define $\widehat{P}_1^\pi(s_1, a_1|s_0) = \pi(a_1|s_1)\widehat{P}_1^\pi(s_1|s_0)$, and $\widehat{P}_2^\pi(s_2|s_0) = \int_{s\in S}\int_{a\in A}\widehat{P}_1^\pi(s,a|s_0)\widehat{T}(s_2|s,a)\mathrm{d}a\mathrm{d}s$. We again apply the Bellman equation:

$$\begin{aligned}
&\gamma\mathbb{E}_{s_1\sim\widehat{P}_1^\pi(\cdot|s_0)}\Big[\widehat{V}^\pi(s_1) - V^\pi(s_1)\Big] = \\
&\qquad \gamma\mathbb{E}_{s_1\sim\widehat{P}_1^\pi(\cdot|s_0)}\Big[\gamma\mathbb{E}_{a_1\sim\pi(\cdot|s_1)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_1,a_1)}\widehat{V}^\pi(s') - \mathbb{E}_{s'\sim T(s_1,a_1)}V^\pi(s')\Big]\Big] \\
&= \gamma\mathbb{E}_{s_1\sim\widehat{P}_1^\pi(\cdot|s_0)}\Big[\gamma\mathbb{E}_{a_1\sim\pi(\cdot|s_1)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_1,a_1)}\widehat{V}^\pi(s') - \mathbb{E}_{s'\sim\widehat{T}(s_1,a_1)}V^\pi(s')\Big]\Big] + \\
&\qquad \gamma\mathbb{E}_{s_1\sim\widehat{P}_1^\pi(\cdot|s_0)}\Big[\gamma\mathbb{E}_{a_1\sim\pi(\cdot|s_1)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_1,a_1)}V^\pi(s') - \mathbb{E}_{s'\sim T(s_1,a_1)}V^\pi(s')\Big]\Big] \\
&= \gamma^2\mathbb{E}_{s_2\sim\widehat{P}_2^\pi(\cdot|s_0)}\Big[\widehat{V}^\pi(s_2) - V^\pi(s_2)\Big] + \gamma\mathbb{E}_{s_1,a_1\sim\widehat{P}_1^\pi(\cdot,\cdot|s_0)}\Big[\mathbb{E}_{s'\sim\widehat{T}(s_1,a_1)}V^\pi(s') - \mathbb{E}_{s'\sim T(s_1,a_1)}V^\pi(s')\Big].
\end{aligned} \tag{13}$$

Combining Equations 12 and 13 we have

$$\begin{aligned}\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \gamma \mathbb{E}_{a_0 \sim \pi(s_0)} \Big[\mathbb{E}_{s' \sim \widehat{T}(s_0, a_0)} V^{\pi}(s') - \mathbb{E}_{s' \sim T(s_0, a_0)} V^{\pi}(s') \Big] + \\ \gamma \mathbb{E}_{s_1, a_1 \sim \widehat{P}_1^{\pi}(\cdot, \cdot | s_0)} \Big[\mathbb{E}_{s' \sim \widehat{T}(s_1, a_1)} V^{\pi}(s') - \mathbb{E}_{s' \sim T(s_1, a_1)} V^{\pi}(s') \Big] + \\ \gamma^2 \mathbb{E}_{s_2 \sim \widehat{P}_2^{\pi}(\cdot | s_0)} \Big[\widehat{V}^{\pi}(s_2) - V^{\pi}(s_2) \Big] \end{aligned} \tag{14}$$

Repeatedly applying the reasoning in Equations 12-14 we have

$$\begin{aligned}\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) &= \sum_{t=0}^{\infty} \gamma^{t+1} \mathbb{E}_{s,a \sim \widehat{P}_t^{\pi}(\cdot, \cdot | s_0)} \Big[\mathbb{E}_{s' \sim \widehat{T}(s,a)} V^{\pi}(s_t) - \mathbb{E}_{s' \sim T(s,a)} V^{\pi}(s_t) \Big] \\ &= \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a \sim d(\pi, \widehat{\mathcal{M}}^R)} \Big[\mathbb{E}_{s' \sim \widehat{T}(s,a)} V^{\pi}(s_t) - \mathbb{E}_{s' \sim T(s,a)} V^{\pi}(s_t) \Big] \end{aligned} \tag{15}$$

where $d(\pi, \widehat{\mathcal{M}}^R)$ is the state-action distribution of π in $\widehat{\mathcal{M}}^R$. By the definition of the reward function ($R(s,a) \in [0,1]$), we have that $V^{\pi}(s) \in [0, 1/(1-\gamma)]$. We utilise the inequality that: $|\mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{x \sim Q} f(x)| \leq 2 \max_x |f(x)| \text{TV}(P, Q)$. Then, taking the absolute value of Equation 15 and applying the inequality we have:

$$\begin{aligned}\Big| \widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) \Big| &\leq \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a \sim d(\pi, \widehat{\mathcal{M}}^R)} \Big| \mathbb{E}_{s' \sim \widehat{T}(s,a)} V^{\pi}(s_t) - \mathbb{E}_{s' \sim T(s,a)} V^{\pi}(s_t) \Big| \\ &\leq \frac{2\gamma}{(1-\gamma)^2} \mathbb{E}_{s,a \sim d(\pi, \widehat{\mathcal{M}}^R)} \Big[\text{TV}\big(\widehat{T}(\cdot | s,a), T(\cdot | s,a) \big) \Big] \quad \square \end{aligned} \tag{16}$$

Now that we have proven Lemma 1, we return to our original purpose of proving Proposition 1. We begin by restating Proposition 1:

Proposition 1 *Let $\widehat{T}$ be the learnt latent dynamics in the world model. Assume the existence of a representation model q that adheres to Assumption 2, and let T be the true latent dynamics according to Assumption 2. Then, for any parameter setting θ and reward function R, the regret of the optimal world model policy is bounded according to:*

$$\begin{aligned}\textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_{\theta}) \leq \frac{2\gamma}{(1-\gamma)^2} \Big[&\mathbb{E}_{z,a \sim d(\pi^*_{\theta,R}, \widehat{\mathcal{M}}_{\theta})} \big[\text{TV}\big(\widehat{T}(\cdot | z,a), T(\cdot | z,a) \big) \big] \\ &+ \mathbb{E}_{z,a \sim d(\widehat{\pi}^*_{\theta,R}, \widehat{\mathcal{M}}_{\theta})} \big[\text{TV}\big(\widehat{T}(\cdot | z,a), T(\cdot | z,a) \big) \big] \Big] \end{aligned}$$

where $d(\pi, \mathcal{M})$ denotes the state-action distribution of π in MDP $\mathcal{M}$, and $\text{TV}(P, Q)$ is the total variation distance between distributions P and Q.

Now, let us consider the latent MDP learnt by our world model $\widehat{\mathcal{M}}_{\theta} = (Z_{\theta}, A, \widehat{T}, \gamma)$ for some parameters $\theta \in \Theta$, as well as the latent space MDP that exactly models the true environment dynamics according to Assumption 2, $\mathcal{M}_{\theta} = (Z_{\theta}, A, T, \gamma)$. Recall that for some reward function, R, the optimal world model policy is defined to be

$$\widehat{\pi}^*_{\theta,R} = \operatorname*{argmax}_{\pi} V(\pi, \widehat{\mathcal{M}}^R_{\theta})$$

The regret of the optimal world model policy for parameter setting θ and reward function R is

$$\textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_{\theta}) = V(\pi^*_{\theta,R}, \mathcal{P}^R_{\theta}) - V(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_{\theta}) \tag{17}$$

By Assumption 2 we have that $V(\pi, \mathcal{P}^R_{\theta}) = V(\pi, \mathcal{M}^R_{\theta})$ for all π. This allows us to write the regret as the performance difference in $\mathcal{M}^R_{\theta}$ rather than $\mathcal{P}^R_{\theta}$:

$$\textsc{Regret}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_{\theta}) = V(\pi^*_{\theta,R}, \mathcal{M}^R_{\theta}) - V(\widehat{\pi}^*_{\theta,R}, \mathcal{M}^R_{\theta}) \tag{18}$$

By the definition of the optimal world model policy, we have that

$$V(\widehat{\pi}^*_{\theta,R}, \widehat{\mathcal{M}}^R_{\theta}) - V(\pi^*_{\theta,R}, \widehat{\mathcal{M}}^R_{\theta}) \geq 0 \tag{19}$$

Adding together Equations 18 and 19 we have

$$\begin{aligned}\text{REGRET}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_\theta) &\leq V(\pi^*_{\theta,R}, \mathcal{M}^R_\theta) - V(\pi^*_{\theta,R}, \widehat{\mathcal{M}}^R_\theta) + V(\widehat{\pi}^*_{\theta,R}, \widehat{\mathcal{M}}^R_\theta) - V(\widehat{\pi}^*_{\theta,R}, \mathcal{M}^R_\theta) \\ &\leq \left|V(\pi^*_{\theta,R}, \mathcal{M}^R_\theta) - V(\pi^*_{\theta,R}, \widehat{\mathcal{M}}^R_\theta)\right| + \left|V(\widehat{\pi}^*_{\theta,R}, \widehat{\mathcal{M}}^R_\theta) - V(\widehat{\pi}^*_{\theta,R}, \mathcal{M}^R_\theta)\right|\end{aligned}$$

Then, applying the Lemma 1 to both terms on the right-hand side we have

$$\begin{aligned}\text{REGRET}(\widehat{\pi}^*_{\theta,R}, \mathcal{P}^R_\theta) \leq \frac{2\gamma}{(1-\gamma)^2}\Big[&\mathbb{E}_{z,a\sim d(\pi^*_{\theta,R}, \widehat{\mathcal{M}}_\theta)}\left[\text{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\right] \\ &+ \mathbb{E}_{z,a\sim d(\widehat{\pi}^*_{\theta,R}, \widehat{\mathcal{M}}_\theta)}\left[\text{TV}\big(\widehat{T}(\cdot|z,a), T(\cdot|z,a)\big)\right]\Big] \square\end{aligned}$$

B Additional Results

B.1 World Model Image Prediction Errors

To compare the quality of the world model predictions, we compute the errors between the images predicted by the world model, and the actual next image. For each world model, we collect 200 trajectories under a uniform random policy in each of 200 randomly sampled environments. Along each trajectory, we compute the next image predicted by the world model by decoding the mean of the next latent state prediction, using the decoder learned by DreamerV2. We compute the mean squared error between the image prediction and the actual next image. Then, we average the errors along the trajectory to compute the error for the trajectory.

We repeat this process along all trajectories to compute the error for each trajectory. Then, we compute $\text{CVaR}_{0.1}$ of these error evaluations by taking the average over the worst 10% of trajectories. These values are plotted for each domain in Figures 7 and 8.

We observe that the CVaR of the image prediction errors are generally lower for world models trained with WAKER than DR. The difference in performance is especially large for the Clean Up domain. This mirrors the large difference in policy performance between WAKER and DR on the Clean Up domain. This verifies that WAKER is successfully able to learn world models that are more robust to different environments than DR. For Terrain Walker, the difference in the error evaluations between methods is smaller, but we still observe that WAKER produces smaller errors than DR when using the Plan2Explore exploration policy.

Note that the error evaluation is performed using trajectories generated by a uniform random policy. Therefore, we observe that the world models trained using data collected by a random policy tend to have lower errors relative to world models trained using Plan2Explore as the exploration policy.

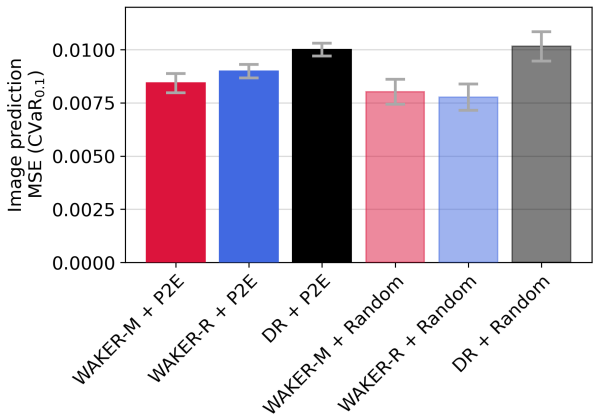


Figure 7: Clean Up image prediction errors. Error bars indicate standard deviations across 5 seeds.

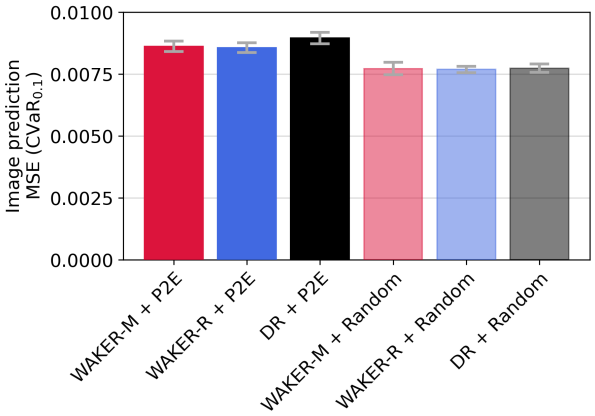


Figure 8: Terrain Walker image prediction errors. Error bars indicate standard deviations across 5 seeds.

B.2 Average Performance Results

In Table 1 in the main results in Section 4, we presented the results for the robustness evaluation, where we averaged performance over the 10 worst runs on 100 randomly sampled environments. In Table 3 and Figure 9 we present the results from averaging over all 100 episodes. Note that we *do not* expect WAKER to improve over DR for this evaluation, as the DR baseline directly optimises for this exact training distribution.

We observe that WAKER achieves similar or improved average performance relative to DR for both exploration policies. Figure 9 shows that WAKER-M achieves a statistically significant improvement in average performance over DR when using a random exploration policy.

Exploration Policy		Plan2Explore			Random Exploration		
Environment	Sampling	WAKER-M	WAKER-R	DR	WAKER-M	WAKER-R	DR
Clean Up	Sort	0.973 ± 0.01	0.966 ± 0.01	0.929 ± 0.02	0.779 ± 0.04	0.766 ± 0.03	0.743 ± 0.03
	Sort-Rev.	0.976 ± 0.01	0.957 ± 0.01	0.934 ± 0.01	0.771 ± 0.03	0.758 ± 0.05	0.739 ± 0.04
	Push	0.972 ± 0.01	0.969 ± 0.01	0.957 ± 0.01	0.839 ± 0.03	0.819 ± 0.03	0.786 ± 0.04
Terrain Walker	Walk	909.1 ± 12.1	915.4 ± 10.3	900.4 ± 22.7	385.8 ± 34.3	360.5 ± 42.0	386.7 ± 24.6
	Run	393.3 ± 8.5	388.7 ± 14.7	397.5 ± 13.1	149.4 ± 10.7	138.4 ± 9.6	149.6 ± 15.2
	Flip	973.2 ± 9.8	965.3 ± 8.5	967.1 ± 5.1	935.6 ± 8.0	924.3 ± 13.7	924.8 ± 9.4
	Stand	970.4 ± 4.9	973.3 ± 6.8	970.3 ± 8.2	673.1 ± 22.5	669.6 ± 66.7	689.7 ± 84.7
	Walk-Back.	861.2 ± 10.0	845.0 ± 25.1	871.6 ± 13.6	474.1 ± 9.6	450.9 ± 19.2	446.7 ± 32.1

Table 3: Average performance evaluation: Average performance on 100 randomly sampled environments. For each exploration policy, we highlight results within 2% of the best score, and ± indicates the S.D. over 5 seeds.

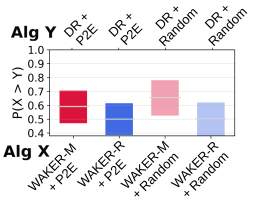


Figure 9: Average performance aggregated CIs.

B.3 Performance on Snapshots of World Model Trained with Variable Amounts of Data

In the main results presented in Section 4, we report the performance of policies obtained from the *final* world model at the end of six days of training. In this section, we present the performance of policies obtained from snapshots of the world model trained with variable amounts of data. These results are presented in Figures 10, 11 and 12. In each of the plots, the x-axis indicates the number of steps of data collected within the environment that was used to train the world model. The values on the y-axis indicate the performance of policies obtained from imagined training within each snapshot of the world model.

As expected, we observe that as the amount of data used to train the world model increases, the performance of policies trained within the world model increases. We observe that for most levels of world model training data, and for both exploration policies, the WAKER variants improve upon DR in the robustness and OOD evaluations in Figures 10 and 12. The magnitude of the improvement tends to become larger as the amount of training data from the environment increases. This suggests that WAKER might be especially effective when scaled to larger amounts of training data.

The average performance of policies trained in snapshots of the world model is presented in Figure 11. For Terrain Walker, the average performance between WAKER and DR for the same exploration policy is similar across all levels of environment data. For Clean Up, we observe that both WAKER variants improve the average performance relative to DR when using Plan2Explore as the exploration policy, for almost all levels of data. For the random exploration policy, we observe that the average

performance initially improves more slowly when using WAKER, but eventually surpasses the performance of DR. This initially slower improvement in average performance with WAKER is likely due to WAKER putting increased emphasis on collecting data from more complex environments.

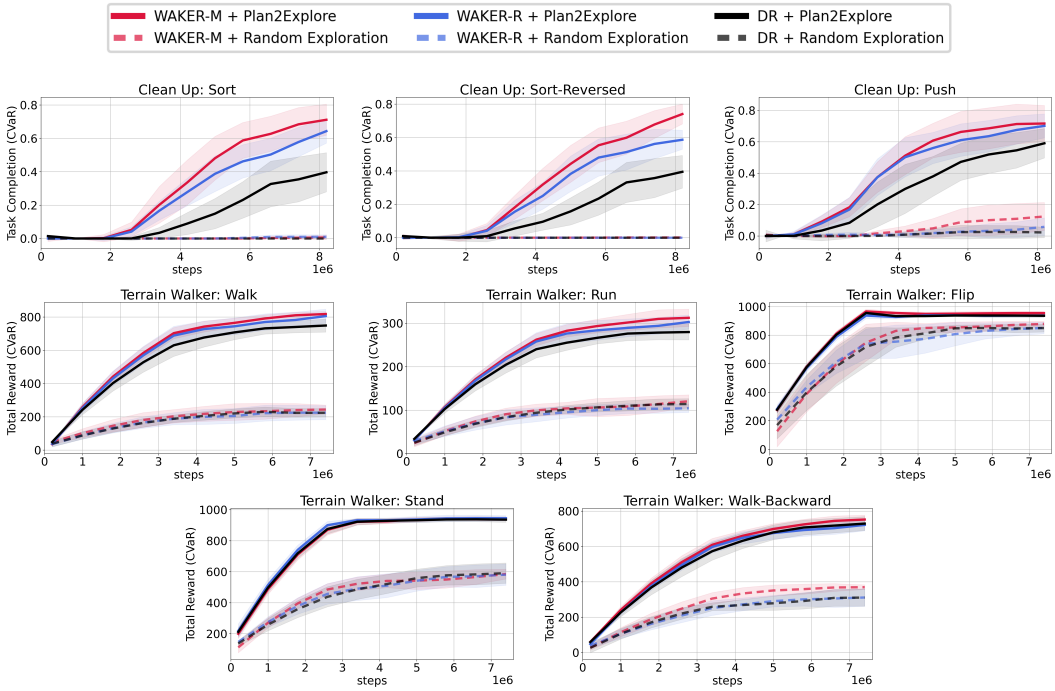


Figure 10: Robustness evaluation. Plots show the $\text{CVaR}_{0.1}$ metric: the mean of worst 10 runs on 100 randomly sampled environments. The policies are trained zero-shot on snapshots of world model trained on amount of data labelled on x-axis. Results are averaged across 5 seeds, and shaded regions are standard deviations across seeds.

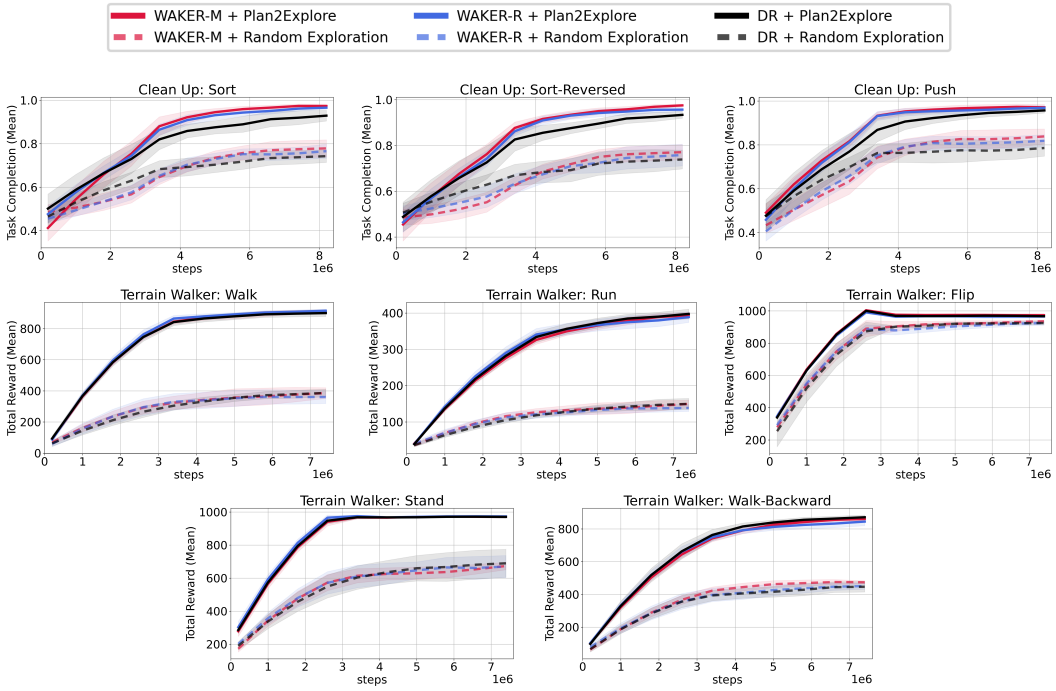


Figure 11: Average performance. Plots show the average performance on 100 evaluations on environments sampled uniformly at random. The policies are trained zero-shot on snapshots of the world model trained on amount of data labelled on x-axis. Results are averaged across 5 seeds, and shaded regions are standard deviations across seeds.

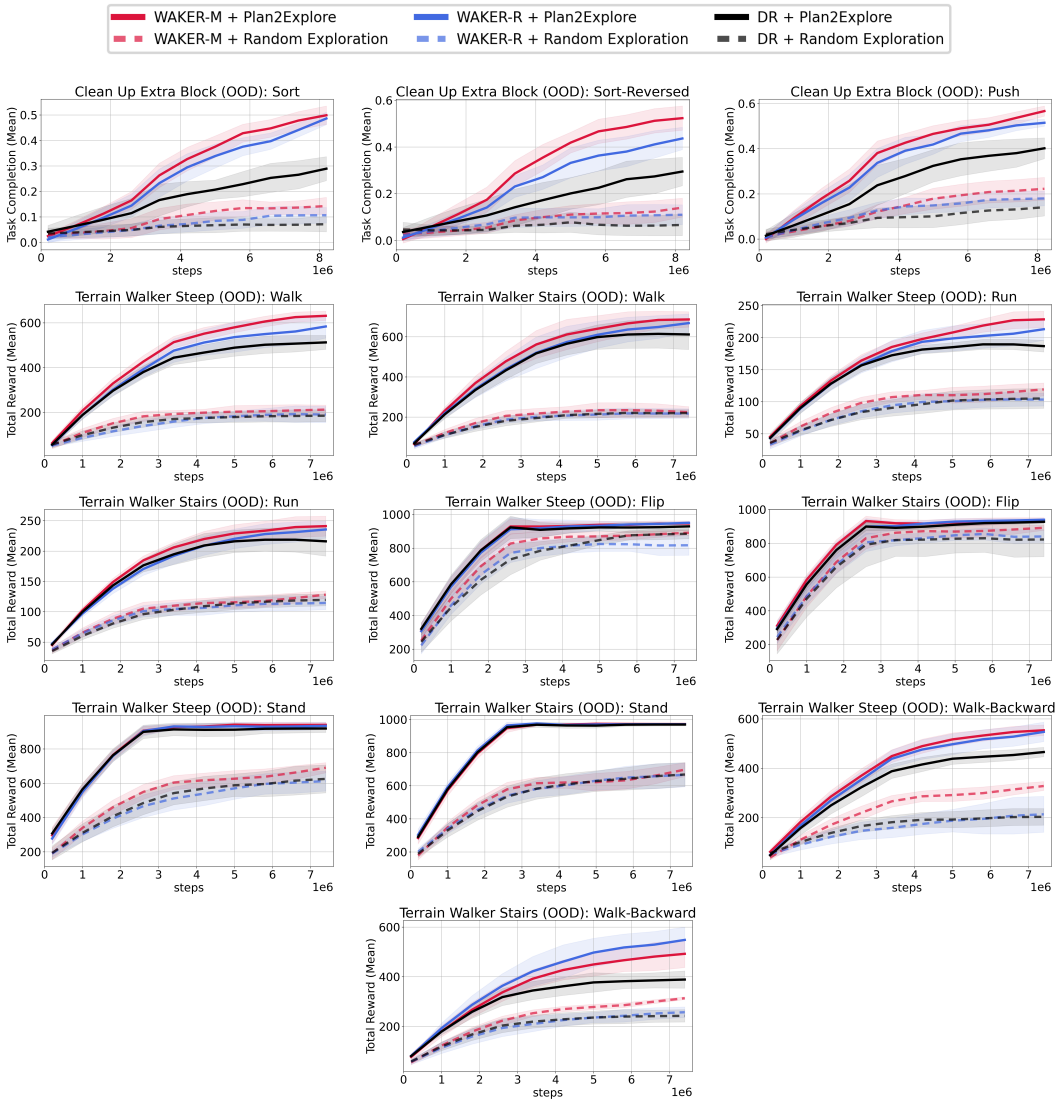


Figure 12: Out-of-distribution evaluation. Plots show the average performance on out-of-distribution environments. The policies are trained zero-shot on snapshots of world model trained on amount of data labelled on x-axis. Results are averaged across 5 seeds, and shaded regions are standard deviations across seeds.

B.4 Results for Reward on Clean Up Domain

In Tables1, 2 and 3, we report the *task completion* for the Clean Up domain (as described in Appendix D.1) as this is easier to interpret than the raw total reward. In Tables 4-6, we present the results for the Clean Up domain in terms of the total rewards. We see that the reward results correspond directly to the task completion results.

Table 4: Robustness evaluation: $\text{CVaR}_{0.1}$ computed by evaluating each policy on 100 randomly sampled environments and averaging the performance on the 10 worst runs. We report the total rewards obtained on the Clean Up domain. For each exploration policy, we highlight results within 2% of the best score, and $\pm$ indicates the S.D. over 5 seeds.

Exploration Policy		**Plan2Explore**			**RandomExplore**		
Environment Sampling		**WAKER-M**	**WAKER-R**	**DR**	**WAKER-M**	**WAKER-R**	**DR**
Clean Up	Sort	424.2 $\pm$ 23.4	395.2 $\pm$ 28.6	265.4 $\pm$ 55.7	15.8 $\pm$ 22.3	8.9 $\pm$ 16.6	-11.7 $\pm$ 14.9
	Sort-Reversed	430.2 $\pm$ 18.9	391.5 $\pm$ 23.7	261.6 $\pm$ 50.7	4.8 $\pm$ 18.1	1.8 $\pm$ 21.3	-11.0 $\pm$ 17.4
	Push	457.8 $\pm$ 10.8	437.2 $\pm$ 24.6	373.3 $\pm$ 42.4	99.2 $\pm$ 40.6	76.3 $\pm$ 28.1	41.8 $\pm$ 25.9

Table 5: Average performance on 100 randomly sampled environments. We report the total rewards obtained on the Clean Up domain. For each exploration policy, we highlight results within 2% of the best score, and $\pm$ indicates the S.D. over 5 seeds.

Exploration Policy		**Plan2Explore**			**RandomExplore**		
Environment Sampling		**WAKER-M**	**WAKER-R**	**DR**	**WAKER-M**	**WAKER-R**	**DR**
Clean Up	Sort	502.0 $\pm$ 2.7	498.8 $\pm$ 5.5	475.1 $\pm$ 12.6	376.1 $\pm$ 18.3	370.2 $\pm$ 13.5	360.9 $\pm$ 16.9
	Sort-Reversed	502.1 $\pm$ 3.0	497.3 $\pm$ 4.1	477.1 $\pm$ 8.3	374.6 $\pm$ 14.1	368.8 $\pm$ 22.1	356.8 $\pm$ 18.4
	Push	506.8 $\pm$ 2.1	502.7 $\pm$ 5.1	492.2 $\pm$ 5.9	408.2 $\pm$ 16.5	400.2 $\pm$ 11.8	387.1 $\pm$ 17.0

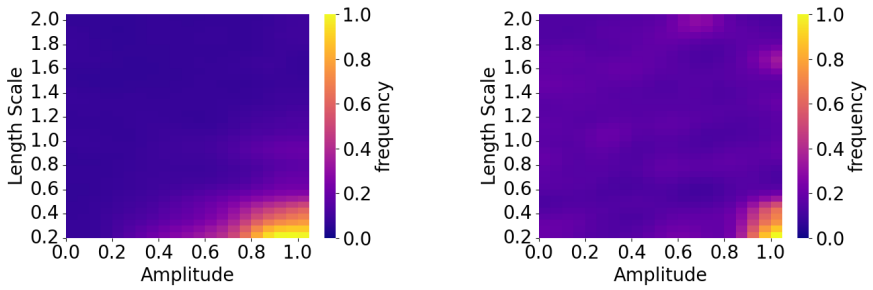
Table 6: Out-of-distribution evaluation: average performance on out-of-distribution environments. We report the total rewards obtained on the Clean Up domain. For each exploration policy, we highlight results within 2% of the best score, and $\pm$ indicates the S.D. over 5 seeds.

Exploration Policy		**Plan2Explore**			**RandomExplore**		
Environment Sampling		**WAKER-M**	**WAKER-R**	**DR**	**WAKER-M**	**WAKER-R**	**DR**
Clean Up	Sort	440.1 $\pm$ 24.6	413.9 $\pm$ 35.3	282.0 $\pm$ 45.2	118.5 $\pm$ 24.7	92.1 $\pm$ 26.6	59.0 $\pm$ 26.2
	Sort-Reversed	462.2 $\pm$ 29.7	385.3 $\pm$ 39.4	266.3 $\pm$ 34.6	120.9 $\pm$ 29.1	102.5 $\pm$ 29.9	56.1 $\pm$ 36.9
	Push	522.6 $\pm$ 19.9	481.6 $\pm$ 21.8	369.9 $\pm$ 48.8	182.7 $\pm$ 34.9	157.8 $\pm$ 17.7	128.6 $\pm$ 28.0

B.5 Heatmaps of Environment Parameters Selected

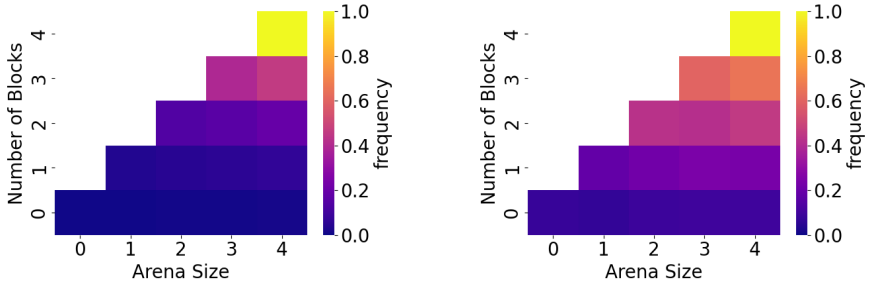
In Figures 13 and Figure 14 we present heatmaps of the environment parameters selected by WAKER during world model training. We observe that WAKER more frequently selects parameters that generate environments which are intuitively more complex. In Terrain Walker, WAKER tends to sample environments where the terrain has a high amplitude and short length scale, corresponding to environments with the steepest terrain. In Clean Up, WAKER tends to sample larger arenas with more blocks in them. This is consistent with the intuition that to learn a robust world model, we need to sample more data from more complex environments.

We observe that WAKER-M more heavily skews sampling towards the more complex environments than WAKER-R. This is to be expected, as WAKER-M directly samples environments with higher uncertainty.



(a) WAKER-M (b) WAKER-R

Figure 13: Normalised frequency of environment parameters selected by WAKER in the Terrain Walker domain. These results use the Plan2Explore exploration policy.



(a) WAKER-M (b) WAKER-R

Figure 14: Normalised frequency of environment parameters selected by WAKER in the Clean Up domain. These results use the Plan2Explore exploration policy. Note that the colour of each block is also selected by WAKER, but these plots only show the total number of blocks to enable 2D visualisation.

C WAKER Implementation Details

Here, we provide additional details on our implementation of the WAKER algorithm that were omitted from the main paper due to space constraints. Code to reproduce our experiments is available at github.com/marc-rigter/waker.

Error Estimate Update and Smoothing In Line 17 of Algorithm 1 we compute the error estimate δ_θ according to the average ensemble disagreement over an imagined trajectory. Each δ_θ value is noisy, so we maintain a smoothed average $\overline{\delta}_\theta$ of the δ_θ values. Specifically, we use an exponential moving average (EMA). For each parameter setting θ, when we receive a new δ_θ value, we update the EMA according to: $\overline{\delta}_\theta \leftarrow \alpha\overline{\delta}_\theta + (1-\alpha)\delta_\theta$, where we set $\alpha = 0.9999$.

For WAKER-M, the error buffer D_{error} contains the smoothed average of the δ_θ values for each parameter setting: $D_{\text{error}} = \{\overline{\delta}_{\theta_1}, \overline{\delta}_{\theta_2}, \overline{\delta}_{\theta_3}, \ldots\}$. These values are used as input to the Boltzmann distribution used to sample the environment parameters in Line 7 of Algorithm 1.

For WAKER-R, we compute the reduction of the $\overline{\delta}_\theta$ values for each parameter setting between each interval of 10,000 environment steps: $\Delta_\theta = \overline{\delta}_\theta^{\text{old}} - \overline{\delta}_\theta^{\text{new}}$. Because the error estimates change very slowly, we perform a further smoothing of the Δ_θ values using an exponential moving average: $\overline{\Delta}_\theta \leftarrow \alpha_\Delta\overline{\Delta}_\theta + (1-\alpha_\Delta)\Delta_\theta$, where we set $\alpha_\Delta = 0.95$. For WAKER-R, the error estimate buffer contains these $\overline{\Delta}_\theta$ values: $D_{\text{error}} = \{\overline{\Delta}_{\theta_1}, \overline{\Delta}_{\theta_2}, \overline{\Delta}_{\theta_3}, \ldots\}$, and these values determine the environment sampling distribution in Line 7 of Algorithm 1.

Error Estimate Normalisation We normalize the error values in Line 7 of Algorithm 1 to reduce the sensitivity of our approach to the scale of the error estimates, and reduce the need for hyperparameter tuning between domains. For WAKER-R, we divide each $\overline{\Delta}_\theta$ value by the mean absolute value

of $\overline{\Delta}_\theta$ across all parameter settings. The rationale for this form of normalisation is that if the rate of reduction of error is similar across all environments, then WAKER-R will sample the environments with approximately equal probability.

For WAKER-M, for each $\overline{\delta}_\theta$ value, we subtract the mean of $\overline{\delta}_\theta$ across all parameter settings, and divide by the standard deviation. This means that regardless of the scale of the error estimates, WAKER-M will always strongly favour the environments with the highest error estimates, as motivated by Problem 2.

World Model Training For the world model, we use the official open-source implementation of DreamerV2 [22] at `https://github.com/danijar/dreamerv2`. For the world model training we use the default hyperparameters from DreamerV2, with the default batch size of 16 trajectories with 50 steps each. For the ensemble of latent dynamics functions, we use 10 fully-connected neural networks, following the implementation of Plan2Explore [63] in the official DreamerV2 repository. We perform one update for every eight environment steps added to the data buffer.

Policy Training For the exploration and task policies, we use the actor-critic implementation from the official DreamerV2 repository. During each update, we sample a batch of 16 trajectories from the data buffer of 50 steps each. For each latent state corresponding to a data sample in the batch, we generate an imagined trajectory starting from that latent state, with a horizon of 15 steps. We then update both the task and exploration policies using dynamics gradients (i.e. backpropagation through the world model) computed on these imagined trajectories. We perform one update to each of the task and exploration policies for every eight environment steps.

D Experiment Details

D.1 Domains and Tasks

Terrain Walker We simulate the Walker robot from the DeepMind Control Suite [69]. The robot has a 6-dimensional action space, and the observations are images of shape $64 \times 64 \times 3$. For each episode, we generate terrain using Perlin noise [46], a standard technique for procedural terrain generation. The terrain generation is controlled by two parameters, the amplitude, with values in $[0, 1]$, and the length scale, with values in $[0.2, 2]$. Domain randomisation samples both of these parameters uniformly from their respective ranges. For examples of the terrain generated, see Figure 15.

Figure 15: Examples of training environments for the Terrain Walker domain.

We evaluate 5 different downstream tasks for Terrain Walker. The tasks *walk*, *run*, *stand*, *flip* are from the URLB benchmark [36], with the exception that we adapt the computation of the standing height to account for the variable height of the terrain. We also include *walk-backward*, which uses the same reward function as *walk*, except that the robot is rewarded for moving backwards rather than forwards.

Terrain Walker Out-of-Distribution Environments We use two different types of out-of-distribution environments for Terrain Walker. The first is *Terrain Walker Steep*, where the length scale of the terrain is 0.15. This is 25% shorter than ever seen during training. This results in terrain with sharper peaks than seen in training. The second is *Terrain Walker Stairs*, where the terrain contains stairs, in contrast to the smooth terrain seen in training. The out-of-distribution environments are shown in Figure 16.

Clean Up The *Clean Up* domain is based on SafetyGym [50] and consists of a robot and blocks that can be pushed. The robot has a two-dimensional action space, one for turning and one for moving forwards and backwards. The observations are images from a top-down view with dimension $64 \times 64 \times 3$.

(a) Terrain Walker Steep (b) Terrain Walker Stairs

Figure 16: Examples of out-of-distribution environments for the Terrain Walker domain.

For each environment, there are three factors that vary: the size of the environment, the number of blocks, and the colour of the blocks. For the default domain randomisation sampling distribution, the size of the environment is first sampled uniformly from $\texttt{size} \in \{0, 1, 2, 3, 4\}$. The number of blocks is then sampled uniformly from $\{0, 1, \ldots, \texttt{size}\}$. The number of green vs blue blocks is then also sampled uniformly. Examples of the training environments generated for the Clean Up domain are in Figure 17.

There are three different tasks for Clean Up: *sort*, *push*, and *sort-reversed*. The tasks vary by the goal location for each colour of block. For *sort*, each block must be moved to the goal location of the corresponding colour. For *sort-reverse*, each block must be moved to the goal location of the opposite colour. For *push*, all blocks must be pushed to the blue goal location, irrespective of the colour of the block.

We define the *task completion* to be the number of blocks in the environment that are in the correct goal region divided by the number of blocks in the environment. If there are no blocks in the environment, then the task completion is 1. The reward function for each task is defined as follows: The agent receives a dense reward for moving any block closer to the desired goal region, and the agent also receives a reward at each time step that is proportional to the task completion.

In the main results we report the task completion at the end of each episode, as it is easier to interpret. We also present results for the total reward in B.4. We observe that the results for the total reward directly correlate to those for the task completion.

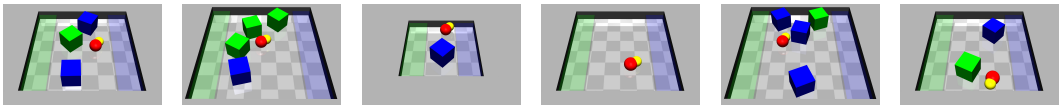


Figure 17: Examples of training environments for the Clean Up domain. The environments differ by their size, the number of blocks, and the colour of each block. The shaded regions indicate the goal locations.

Clean Up Out-of-Distribution Environments For the Clean Up Extra Block out-of-distribution environments, we place one more block in the environment than was ever seen during training. We set the size of the environment to $\texttt{size} = 4$, and there are 5 blocks in the environment. The task completion and reward function is defined the same as for the training environments. Examples of the out-of-distribution environments are in Figure 18.

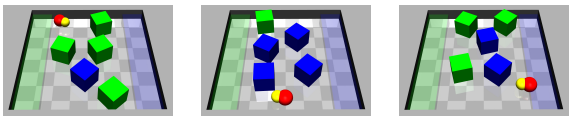


Figure 18: Examples of out-of-distribution environments for Clean Up Extra Block domain.

D.2 Length of Training Runs

Due to resource limitations, each training run is limited to six days of run time. This corresponds to a total of 7.4×10^6 environment steps for the Terrain Walker domain, and a total of 8.2×10^6 environment steps for the Clean Up domain. The results reported in Section 4 are for task policies trained in imagination in the final world models at the end of these training runs.

D.3 Hyperparameter Tuning

We use the default parameters for DreamerV2 [22] for training the world model.

For WAKER, there are two hyperparameters: the probability of sampling uniformly from the default environment distribution, p_{DR}, and the temperature parameter for the Boltzmann environment distribution, η. In our experiments, we set $p_{\text{DR}} = 0.2$ for all experiments and did not tune this value. We performed limited hyperparameter tuning of the Boltzmann temperature parameter, η. We ran WAKER-M + Plan2Explore and WAKER-R + Plan2Explore for each of three different values of $\eta \in \{0.5, 1.0, 1.5\}$. For each algorithm and η value, we ran two seeds for 5e6 environment steps on the Clean Up domain. At the end of 5e6 environment steps, we chose the value of η that obtained the best performance for each algorithm for $\text{CVaR}_{0.1}$ on the $sort$ task. We then use this value of η for WAKER-M and WAKER-R across all experiments, when using both the Plan2Explore and the random exploration policies.

The hyperparameters used in our experiments for WAKER are summarised in Table 7.

Table 7: Summary of hyperparameters used by WAKER.

	WAKER-M	WAKER-R
p_{DR}	0.2	0.2
η	1	0.5

D.4 Confidence Interval Details

Figures 3, 4 and 9 present 95% confidence intervals of the probability of improvement, computed using the *rliable* framework [1]. To compute these values, we first normalise the results for each algorithm and task to between $[0, 1]$ by dividing by the highest value obtained by any algorithm. We then input the normalised scores in the *rliable* package to compute the confidence intervals.

A confidence interval where the lower bound on the probability of improvement is greater than 0.5 indicates that the algorithm is a statistically significant improvement over the baseline.

D.5 Computational Resources

Each world model training run takes 6 days on an NVIDIA V100 GPU. In our experiments, we train 60 world models in total, resulting in our experiments using approximately 360 GPU days.