AUGMENTED WORLD MODELS FACILITATE ZERO-SHOT DYNAMICS GENERALIZATION FROM A SINGLE OFFLINE ENVIRONMENT

Philip J. Ball[*], **Cong Lu**[*], **Jack Parker-Holder**, **Stephen Roberts**
University of Oxford

ABSTRACT

Reinforcement learning from large-scale offline datasets provides us with the ability to learn policies without potentially unsafe or impractical exploration. Significant progress has been made in the past few years in dealing with the challenge of correcting for differing behavior between the data collection and learned policies. However, little attention has been paid to potentially changing dynamics when transferring a policy to the online setting, where performance can be up to 90% reduced for existing methods. In this paper we address this problem with *Augmented World Models* (AugWM). We augment a learned dynamics model with simple transformations that seek to capture potential changes in physical properties of the robot, leading to more robust policies. We not only train our policy in this new setting, but also provide it with the sampled augmentation as a context, allowing it to adapt to changes in the environment. At test time we *learn the context* in a self-supervised fashion by approximating the augmentation which corresponds to the new environment. We rigorously evaluate our approach on over 100 different changed dynamics settings, and show that this simple approach can significantly improve the zero-shot generalization of a recent state-of-the-art baseline, often achieving successful policies where the baseline fails.

1 INTRODUCTION

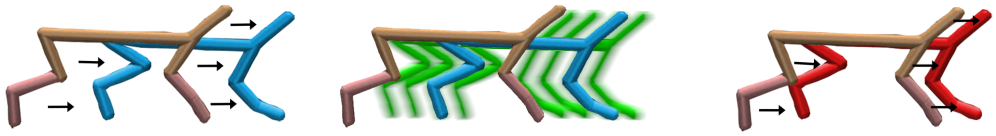
Offline reinforcement learning (RL) describes the problem setting where RL agents learn policies solely from previously collected experience without further interaction with the environment (12; 29). This could have tremendous implications for real world problems (10), with the potential to leverage rich datasets of past experience where exploration is either not feasible (e.g. a Mars Rover) or unsafe (e.g. in medical settings). As such, interest in offline RL has surged in recent times.

This work focuses on model-based offline RL, which has achieved state-of-the-art performance through the use of uncertainty penalized updates (45; 23). However, existing work only addresses the issue of transferring from different behavior policies in the *same environment*, ignoring any possibility of distribution shift. Consider the case where it is expensive to collect data, and we have access to a single dataset from a robot. Using existing methods we would be unable to make any changes that impact the dynamics, such as using a newer model of the robot or deploying it in a different room.

A related setting is the Sim2Real problem which considers transferring an agent from a simulated environment to the real world. A popular recent approach is domain randomization (42; 20), the process of randomizing non-essential regions of the observation space to make agents robust to 'observational overfitting' (41). Indeed, methods seeking to generalize to novel dynamics have also shown promise (34), by randomizing physical properties such as the mass of the agent. A significant limitation of these approaches is the requirement for a simulator, which may not be available.

In this work we take inspiration from Sim2Real to generalize solely from an offline dataset, in a learned simulator or *World Model* (WM). We therefore describe our problem setting as follows: an

[*]Equal contribution; the order of first authors was determined by a coin flip. Correspondence to ball@robots.ox.ac.uk, cong.lu@stats.ox.ac.uk



(a) World Model: $\widehat{P}$ (b) Augmented World Model: $\widehat{P}_z$ (c) Test: $P^\star$

Figure 1: An illustration of our approach, each figure shows a transition $P(s, a) \rightarrow s'$. In a) we show the World Model dynamics $\widehat{P}$, trained from $\mathcal{D}_{\text{env}}$, a single offline dataset. In b) we show the Augmented World Model, blue represents $\widehat{P}$, while each green agent illustrates an instantiation of augmented dynamics, which is sampled at each timestep: $\widehat{P}_z, z \sim \mathcal{Z}$. The goal is to approximate c) where we show an unseen test environment, with transition dynamics $P^\star$.

agent must learn to generalize to unseen test-time dynamics whilst having access to offline data from *only a single environment*; we call this "dynamics generalization from a single offline environment".

In this paper we concentrate on the zero-shot performance of our agents to unseen dynamics, as it may not be practical (nor safe) to perform multiple rollouts at test time. To tackle this problem, we propose a novel form of data augmentation: rather than augment observations, we focus on *augmenting the dynamics*. We first learn a world model of the environment, and then augment the transition function at policy training time, making the agent train under different imagined dynamics. In addition, our agent is given access to the augmentation itself as part of the observations, allowing it to consider the context of modified dynamics.

At test time we propose a simple, yet surprisingly effective, self-supervised approach to learning an agent's augmentation context. We learn a linear dynamics model which is then used to approximate the dynamics augmentation induced by the modified environment. This context is then given to the agent, allowing it to adapt on the fly to the new dynamics within a single episode (i.e. zero-shot). We show that our approach is capable of training agents that can vastly outperform existing Offline RL methods on the "dynamics generalization from a single offline environment" problem. We also note that this approach does *not* require access to environment rewards at test time. This facilitates application to Sim2Real problems whereby test time rewards may not be available.

Our contributions are twofold: 1) As far as we are aware, we are the first to propose **dynamics augmentation for model based RL**, allowing us to generalize to changing dynamics despite only training on a single setting. We do this without access to any environment parameters or prior knowledge. 2) We propose a simple **self-supervised context adaptation** reward-free algorithm, which allows our policy to use information from interactions in the environment to vary its behavior in a single episode, increasing zero-shot performance. We believe both of these approaches are not only novel, but offer significant improvement v.s. state-of-the-art methods, improving generalization and providing a promising approach for using offline RL in the real world.

2 Related Work

In this work we focus on Model Based RL (MBRL). A key challenge in MBRL is that an inaccurate model can be exploited by the policy, leading to behaviors that fail to transfer to the real environment. As such, a swathe of recent works have made use of model ensembles to improve robustness (26; 7; 9; 5; 21). With increased accuracy, MBRL has recently been shown to be competitive with model free methods in continuous control (14; 21; 7) and games (39; 22). We make use of the ensemble of probabilistic dynamics models, first introduced in (7), and subsequently used in (21).

In this paper we focus on Model-Based *offline* RL, where MOPO (45) and MOReL (23) have recently demonstrated the effectiveness of learned dynamics models, using model uncertainty to constrain policy optimization. We build upon this approach for zero-shot dynamics generalization from offline data. There have also been successes in off policy methods for offline RL (44; 25; 12) and context based approaches (1), although these works only consider tasks within the support of the offline dataset. Finally, MBOP (4) addresses the problem of goal-conditioned zero-shot transfer from offline datasets. However, their goal-conditioning relies on unchanged dynamics in the test environment.

In online RL, recent work has achieved strong dynamics generalization with a learned model (40). However, this required training under varied dynamics, assigning different experiences to models. In addition, this work used MPC whereas we train a policy inside the model, which is significantly faster at deployment time. Also related is (8; 30), where the model is trained to quickly to adapt to

new dynamics $P(s'|s,a)$, however both these works place more emphasis on model-adaption rather than zero-shot policy performance. Furthermore, access to an underlying task distribution is required, something we do not have in our offline setting. Also similar to our work is the recently proposed Policy Adaptation during Deployment (PAD, (18)) approach. Our approach differs in that we learn a context, whereas PAD uses a auxiliary objective to adapt its features. In addition, PAD considers the *online model free* setting, while our method is *offline and model based*.

Sim2Real is the setting where an agent trained in a simulator must transfer to the real world. A common approach to solve this problem is through domain randomization (42; 20), whereby parameters in the simulator are varied during training. This has shown to be effective for dynamics generalization (2; 3; 34; 46; 47; 32), but requires access to a simulator which we do not have. Another form of domain randomization, data augmentation, has proved to be effective for training RL policies (27; 28; 24; 36), resulting in improved efficiency and generalization. So far, these works have focused on online model free methods, and used data augmentation on the *state* space, reducing *observational* overfitting (41). In contrast, we focus on offline MBRL and instead augment the *dynamics*.

We are not the first to propose data augmentation in the MBRL setting, (35) proposed Counterfactual data augmentation for improving performance in the context of locally factored tasks. Approaches to ensuring adversarial robustness can include data augmentations that assist with out-of-domain generalization, as opposed to observational overfitting. In (43) this is done without a simulator and from a single source of data, however they only work on supervised learning problems and require an adversary to be learned, adding computational complexity.

3 PRELIMINARIES

We consider a Markov Decision Process (MDP), defined as a tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space respectively, $P(s'|s,a)$ the transition dynamics, $R(s,a)$ the reward function, ρ_0 the initial state distribution, and $\gamma \in (0,1)$ the discount factor. The goal in RL is to optimize a policy $\pi(a|s)$ that maximizes the expected discounted return $\mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right]$. The value function $V^\pi(s) := \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0=s\right]$ gives the expected discounted return under π when starting from state s. In *offline RL*, the policy is not deployed in the environment until test time. Instead, the algorithm only has access to a static dataset $\mathcal{D}_{env} = \{(s,a,r,s')\}$, collected by one or more behavioral policies π_b. We borrow notation from (45) and refer to the distribution from which $\mathcal{D}_{env}$ was sampled as the *behavioral distribution*.

When training a model, we follow MBPO (21) and MOPO (45) and train an ensemble of N probabilistic dynamics models (31). Each of the models learns to predict both the next state s' and reward r from a state-action pair, using $\mathcal{D}_{env}$ in a supervised fashion. Concretely, each of the N models output a Gaussian $\widehat{P}_i(s_{t+1}, r_t|s_t,a_t) = \mathcal{N}(\mu(s_t,a_t), \Sigma(s_t,a_t))$. The resulting dynamics model $\widehat{P}$ defines a *model MDP* $\widehat{M} = (\mathcal{S}, \mathcal{A}, \widehat{P}, \widehat{R}, \rho_0, \gamma)$, where $\widehat{R}$ refers to the learned reward model.

To train the policy, we use k step rollouts inside $\widehat{M}$, adding experience to a replay buffer $\mathcal{D}_{\widehat{\text{env}}}$ to learn an action-value function and a policy, using Soft Actor Critic (SAC (15)). SAC alternates between a soft policy evaluation step, which estimates $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t)+\alpha\mathcal{H}(\pi(\cdot|s_k)))|s_0 = s, a_0 = a]$ using Bellman backups (where α is a temperature parameter for policy entropy $\mathcal{H}$), and a policy improvement which learns a policy by minimizing the expected KL divergence $J_\pi(\phi,\mathcal{D}) = \mathbb{E}_{s_t\sim\mathcal{D}_{\widehat{\text{env}}}}[D_{KL}(\pi\|\exp\{\frac{1}{\alpha}\{Q^\pi - V^\pi\}\})]$. Note that the SAC algorithm is unchanged from the model-free setting, aside from the environment being a learned model, and the rollout horizon k being truncated. Perhaps surprisingly, this approach alone produces strong results in the offline setting. However MOPO (45) and MORel (23) show improvement in performance by penalizing rewards in regions of the state space where the ensemble of probabilistic models is less certain. This implicitly reduces reliance on samples which deviate beyond the support of $\mathcal{D}_{env}$.

While MOPO and MORel have addressed the issue of training a policy in $\mathcal{D}_{env}$, and transferring to the true environment M, they only consider where the data in $\mathcal{D}_{env}$ is actually drawn from P. However, sometimes this may not be sufficient for deployment. For example, a robot could fail to walk when learning from data that was collected by a different version of the robot (with different mass), or if the same robot collected data but in a different room to deployment (with varied friction). It is this setting, where dynamics may vary at test time, that is the focus of our work. To learn successfully we propose a novel approach to training robust context-dependent policies.

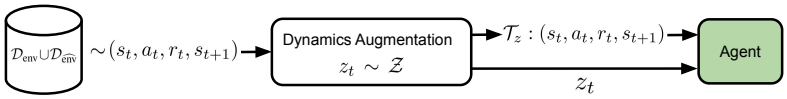


Figure 2: Training policies in Augmented World Models. For each state-action pair sampled from the buffer, a new augmentation $z_t \sim \mathcal{Z}$ is sampled to produce an augmentation operator $\mathcal{T}_z$, which is applied to the transition. The policy is then trained with the new tuple of data, with the context concatenated to the state.

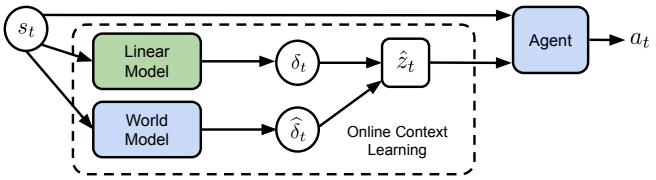


Figure 3: Self-supervised policy adaptation via learned context. At each timestep, the state s_t is fed into a linear model (trained online at each timestep) to predict the change in state δ_t, and also passed to the fixed World Model (trained on the offline data) to predict the change in state under $\widehat{P}$, $\widehat{\delta}_t$. The approximate context is then $z_t = {}^{\delta_t}/_{\hat{\delta}_t}$, which is concatenated with s_t and passed to the agent to produce an action.

4 Augmented World Models with Self Supervised Policy Adaptation

In this section we introduce our algorithm: Augmented World Models (AugWM). We first discuss our training procedure (Fig. 2) before moving onto our self-supervised approach to online context learning (Fig. 3).

4.1 Augmented World Models

Rather than seeking to transfer our policy from $\widehat{M}$ to M, we instead wish to transfer from $\widehat{M}$ to $M^\star$, where $M^\star = (\mathcal{S}, \mathcal{A}, P^\star, R^\star, \rho_0, \gamma)$ is an unseen environment with *different dynamics*. Thus, our emphasis shifts to producing experience inside the world model such that our agent is able to generalize to unseen, out of distribution dynamics. We approach this problem by training our policy in an *Augmented World Model*. Formally, we denote an augmentation as $z_t \sim \mathcal{Z}, z \in \mathbb{R}^{|\mathcal{S}|}$, which is sampled at each timestep to produce an augmentation operator $\mathcal{T}_z$. $\mathcal{T}_z$ is applied to (s, a, r, s') tuples from the dataset $\mathcal{D}$, and when used with tuples sampled from $\mathcal{D}_{\widehat{\text{env}}}$ indirectly induces an augmented distribution $\widehat{P}_z$. In principle, we wish to produce augmentations such that the true modified environment dynamics $P^\star$ lies in the support of the distribution of augmented dynamics, i.e.

$$\inf_z D(\widehat{P}_z(s,a) \| P^\star(s,a)) \leq \epsilon \tag{1}$$

for all s, a, some small $\epsilon > 0$, and suitable distance/divergence metric D. We consider several augmentations, beginning with existing works before moving to new approaches which specifically target the problem of dynamics generalization. We begin with **Random Amplitude Scaling** as in (27), which we refer to as RAD. RAD scales both s_t and s_{t+1} as follows:

$$\mathcal{T}_z : (s_t, a_t, r_t, s_{t+1}) \mapsto (z \odot s_t, a_t, r_t, z \odot s_{t+1}) \tag{2}$$

for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$. Given that our focus is on changing dynamics (vs. observational overfitting), we also propose to scale only the next state, i.e., **Random Amplitude Nextstate Scaling** (RANS):

$$\mathcal{T}_z : (s_t, a_t, r_t, s_{t+1}) \mapsto (s_t, a_t, r_t, z \odot s_{t+1}) \tag{3}$$

for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$. Note that while RANS is more focused on augmenting dynamics than RAS, it still suffers from a dependence on the magnitude of s_{t+1}. As such, we further propose a more targeted augmentation, which we call **Dynamics Amplitude Sampling** (DAS). Rather than directly scale the state, DAS scales the *change in the state* which we denote with $\delta_t = s_{t+1} - s_t$, as follows:

$$\mathcal{T}_z : (s_t, a_t, r_t, s_{t+1}) \mapsto (s_t, a_t, r_t, s_t + z \odot \delta_t) \tag{4}$$

for $z \sim \text{Unif}([a, b]^{|\mathcal{S}|})$. The full training procedure is shown in Algorithm 1.

Algorithm 1: Augmented World Models: Training

Input: Offline data $\mathcal{D}_{\text{env}}$, Penalty λ, horizon h, batchsize B, augmentation $\mathcal{Z}$.
Initialize: Ensemble of N dynamics models $\widehat{P}$, policy π. Replay buffer $\mathcal{D}_{\widehat{\text{env}}}$
1. Train $\widehat{P}$ in a supervised fashion using $\mathcal{D}_{\text{env}}$.
for *epoch* $= 1, 2, \ldots$ **do**
  (i) Sample initial states: $\{s_1^1, \ldots, s_1^B\} \sim \mathcal{D}_{\text{env}}$
  (ii) Rollout policy (in parallel), using λ-penalized reward, storing all data in $\mathcal{D}_{\widehat{\text{env}}}$
  (iii) Train policy using $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\widehat{\text{env}}}$. For each (s, a, r, s'), sample $z \sim \mathcal{Z}$ and apply $\mathcal{T}_z$.
end
Return: Policy π

One crucial addition to our method is the use of context. Concretely, when we are optimizing the policy using a batch of data, we concatenate the next state with the augmentation vector z. This allows our policy to be informed of the specific augmentation that was applied to the environment and thus behave accordingly. However, at test time we do not know z, so what can we use? Next we propose a solution to this problem, learning the context on the fly.

4.2 Self-Supervised Policy Selection

In the meta-learning literature there have been many recent successes making use of a *learned context* to adapt a policy at test time to a new environment (38; 48), typically using a blackbox model with a latent state. Crucially, these approaches require several episodes to adapt at test time, making them unfeasible in our zero-shot setting. What makes our setting unique is we explicitly know what the context represents: a linear transformation of s_t, or δ_t. Using this insight, we are able to learn an effective context on the fly at test time. Concretely, we observe that from a state s_t drawn from $M^\star$, we can sample an action $a_t \sim \pi$ and then compute an approximate $\widehat{s}_{t+1}$ using our model $\widehat{P}$. With $\widehat{s}_{t+1}$, we have a sample estimate of the *state change under* $\widehat{P}$, i.e. $\widehat{\delta}_t = \widehat{s}_{t+1} - s_t$. We can make this approximation of the next state without interacting with the environment, but once we do take the action a_t in the environment, we then receive the *true next state* s_{t+1} and can store the true difference $\delta_t = s_{t+1} - s_t$. Using the DAS augmentation, we can approximate z as ${}^{\delta_t}/_{\hat{\delta}_t}$.

Algorithm 2: Augmented World Models: Testing

Input: Initial state s_1, horizon H, policy π, world model $\widehat{P}$, initial context $\hat{z} = \mathbf{1}^{|\mathcal{S}|}$
Initialize: Linear model f_ψ, dataset $\mathcal{D} = \emptyset$, return $R_0 = 0$.
for *step* $= 1, 2, \ldots H$ **do**
  Select action: $a_t \sim \pi(s_t, \widehat{z}_t)$
  Take action: $s_{t+1} \sim P^\star(s_t, a_t)$
  Update return: $R_{t+1} = R_t + R^\star(s_t, a_t, s_{t+1})$
  Update Dataset: $\mathcal{D} \cup (s_t, \delta_t)$.
  Update Linear model by minimizing $\mathcal{L}_{\text{MSE}}(\psi, \mathcal{D})$.
  Predict new context using $\widehat{z}_t = {}^{\delta_t}/_{\hat{\delta}_t}$.
end
Return: R_T

This however is retrospective; we can only approximate z having already seen the next state, by which time our agent has already acted. Furthermore, we believe under changed dynamics the true z likely depends on s, thus we cannot use a previous z for future timesteps. Therefore, we learn a forward dynamics model using the data collected *during* the test rollout. After h timesteps in the environment, we have the following dataset: $\mathcal{D} = \{(s_1, s_2), \ldots, (s_{h-1}, s_h)\}$. This allows us to learn a simple dynamics model $f_\psi : (s_t) \mapsto \delta_t = s_{t+1} - s_t$, by minimizing the mean squared error $\mathcal{L}_{\text{MSE}}(\psi, \mathcal{D})$. Notably, since we never actually plan with this model, it does not need to be as accurate as a typical dynamics model in MBRL. Instead, it is crucial that the model learns quickly enough such that we can use it in a zero-shot evaluation. Thus, we choose to use a linear model for f. To show the effectiveness of this, in Fig. 4 we show the mean R-squared of linear models learned on the fly during evaluation rollouts. We observe that in less than 100 timesteps the linear model achieves high accuracy on the test data. Subsequently, equipped with f_ψ, we can approximate δ_t, and predict the augmentation as $\widehat{z}_t = {}^{\delta_t}/_{\hat{\delta}_t}$. We then provide the agent with $\widehat{z}_t$ to compute action a_t. The full procedure is shown in Algorithm 2.

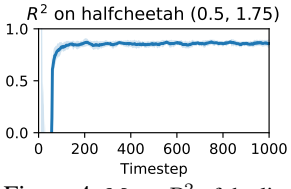


Figure 4: Mean R^2 of the linear model across 20 rollouts.

5 EXPERIMENTS

In our experiments we aim to investigate the effectiveness of our approach for zero-shot dynamics generalization from a single offline dataset. To assess this, we will answer a series of questions, beginning with a question on the necessity of our method:

Do we really need to develop methods specifically for dynamics generalization?

To answer this, we train MOPO using offline data from a single environment, and test it under changed dynamics. We consider the HalfCheetah environment from the OpenAI Gym (6), using offline data from D4RL (11). We train a MOPO agent using the mixed dataset, using our own implementation of the algorithm (but using the same hyperparameters as the original authors). To test the trained policy, we vary both the mass of the agent and damping coefficient by a multiplicative factor.[1] In this work we consider a grid of the following values for HalfCheetah: $\{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$ and $\{0.5, 0.75, 1.0, 1.25, 1.5\}$ for Walker2d, representing a significant out-of-distribution shift.

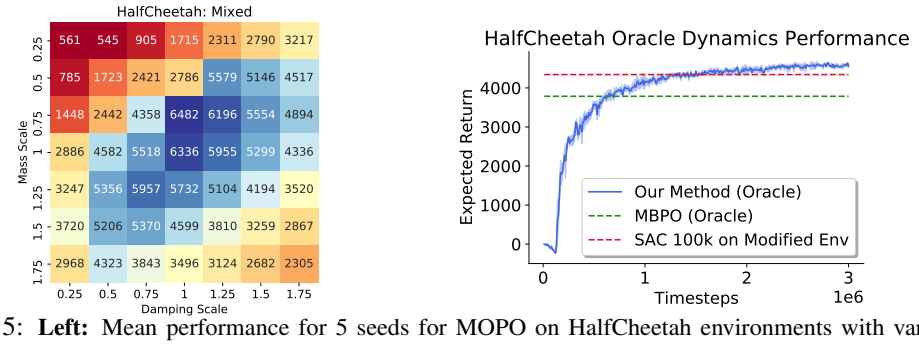


Figure 5: **Left:** Mean performance for 5 seeds for MOPO on HalfCheetah environments with varying dynamics. Note the central cell (1,1) corresponds to the in-sample data. **Right:** Mean performance for 3 seeds for MBPO with a dynamics oracle on HalfCheetah with $1.5\times$ mass and damping.

The results (Fig. 5, left), show that MOPO performance is clearly impacted by changing dynamics. We see in the central cell, that performance for our version of MOPO matches the author results (45), and in some cases we even see small gains (e.g. mass $= 0.75$, damping $= 1.0$). However, on the top left we see dramatically weaker performance, often below 1k, indicating the robot is failing to achieve locomotion. Before evaluating AugWM, we first test whether training with the "correct" augmentation improves generalization performance. In short, we ask:

Is augmenting dynamics even worthwhile?

To answer this, we train SAC for 1×10^5 steps and save the states visited in the 'true' environment. We then use these starting states to train a policy using an offline MBPO[2] approach with AugWM. However, instead of sampling $z_t \sim \mathcal{Z}$, we provide the actual $z = {}^{\delta^\star}/_{\hat{\delta}}$ as we have access to the 'true' and 'modified' environments; we refer to this as an *oracle* version of our method, and is designed to assess the viability of our approach. Note that we *do not* augment the 'true' environment rewards. We consider two baselines: a) offline MBPO in the 'true' environment; b) online SAC in the 'modified' environment. We train MBPO until convergence, and SAC for 1×10^5 steps. As shown Fig. 5 (**Right**), when provided with the true z, AugWM outperforms both baselines. The SAC result is surprising: the baseline agent was trained directly on the 'modified' environment for the same number of steps as the policy that generated our oracle starting states. One explanation is the greater exploration induced by the 'easier' dynamics of the 'true' environment. This validates our approach; if we augment the dynamics $\widehat{P}$ from a model correctly, we can generalize to unseen dynamics. Neither the starting states nor rewards need to be from the 'modified' environment. Our next question is a simple one:

Which augmentation strategy is most effective?

To test this, we train as in Algorithm 1, *without* context, to isolate the effectiveness of the training process. We use the HalfCheetah mixed dataset and train a MOPO agent, augmenting either both s and s' (RAD), just s' (RANS) or just δ (DAS); the results are shown in Fig. 6. As we see, the RAD augmentation fails to improve dynamics generalization, actually leading to worse performance

[1] The standard environment (both in Gym and D4RL) corresponds to both these values being set to 1.0.

[2] Since we have access to the true environment, there is no need for the MOPO penalty.

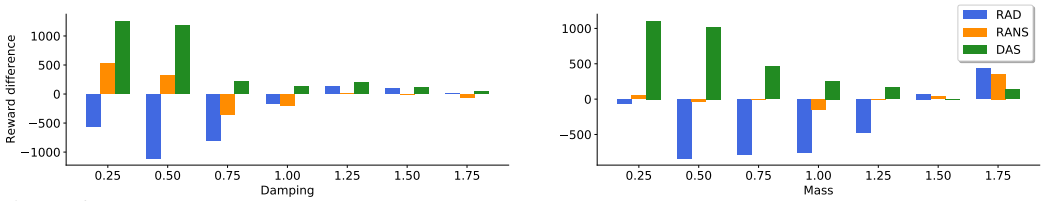


Figure 6: Average performance gains of different World Model augmentations over base MOPO for different levels of damping and mass in the HalfCheetah test environment (5 seeds).

overall. RANS does improve performance on unseen dynamics, as we are influencing the *dynamics*, not just the observation. However, DAS clearly provides the strongest performance. As a result, we use DAS for AugWM. Our final algorithm design question is as follows:

Does training with context improve performance?

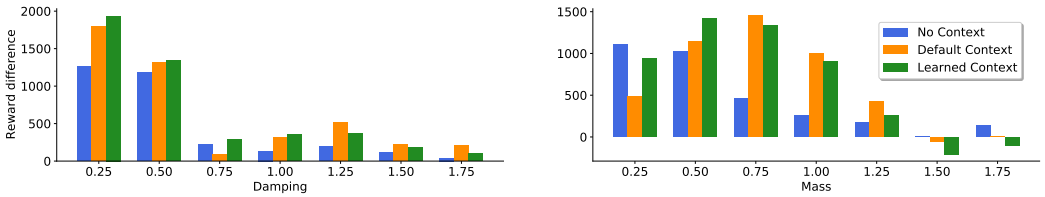


Figure 7: Average performance gains of adding contexts over base MOPO for different levels of damping and mass in the HalfCheetah test environment (5 seeds).

To answer this question we return to the HalfCheetah mixed setting from Fig. 6, taking the policy trained with DAS. We now train two additional agents: 1) Default Context: at *train* time the agent is provided with the DAS augmentation z as context, at *test* time it is provided with a vector of ones, $\mathbf{1}^{|\mathcal{S}|}$; 2) Learned Context: trained as in 1), but context is learned online using Algorithm 2. The results are shown in in Fig. 7. We observe that training with context (orange) improves performance on average, while adapting the context on the fly (green) leads to further gains (+80 on average). These methods combine to produce our AugWM algorithm. We are now ready for the final question:

Can Augmented World Models improve zero-shot generalization?

Table 1: Each entry for HalfCheetah is the mean of 49 different dynamics, while for Walker2d it is over 25 dynamics. Results are mean $\pm$1std. $\star$ indicates $p < 0.05$ for Welch's t-test for gain over MOPO (5 seeds).

Dataset Type	**Environment**	**MOPO**	**AugWM (Ours)**
Random	HalfCheetah	2303 ± 112	2818 ± 197 $\star$
Random	Walker2d	569 ± 103	706 ± 139
Mixed	HalfCheetah	3447 ± 218	3948 ± 122 $\star$
Mixed	Walker2d	946 ± 95	1317 ± 206 $\star$
Medium	HalfCheetah	2954 ± 89	2967 ± 106
Medium	Walker2d	1477 ± 337	1614 ± 440
Med-Expert	HalfCheetah	1590 ± 766	2885 ± 432 $\star$
Med-Expert	Walker2d	1062 ± 334	2521 ± 316 $\star$

To answer this question we perform a rigorous analysis, using multiple benchmarks from the D4RL dataset (11). Namely, we consider the random, medium, mixed and med-expert datasets for both Walker2d and HalfCheetah. In each setting, we compare AugWM against base MOPO on zero-shot performance, training entirely on the data provided, but not seeing the test environment until evaluation. The results are shown as a change v.s. MOPO, averaged over one dimension in Fig. 10, and as a total return number averaged over both dimensions in Table 1. For additional implementation details (e.g., hyperparameters) see Appendix B. AugWM provides *statistically significant* improvements in zero-shot performance v.s. MOPO in many cases, achieving successful policies where MOPO fails.

By now we have provided significant evidence that AugWM can significantly improve performance for HalfCheetah and Walker2d with varied mass and damping. However, this is only a small subset of possible dynamics changes. We next consider several significantly harder settings. We test increased dimensionality, using the Ant environment from MOPO (45), and also consider varied dynamics changes (Ant with crippled legs, HalfCheetah with varied physical properties from (19)). We show the mean results over each of these factors of variation in Table 2, where once again AugWM provides a non-trivial improvement over a strong baseline. For more details see Appendix B.

Table 2: Mean performance for MOPO, AugWM with the default context, and AugWM with learned context (LM). Entries are mean zero-shot reward for all dynamics. Bold = highest (5 seeds).

Setting	MOPO	AugWM (Default)	AugWM (LM)
Ant: Mass/Damp	1634	1715	**1804**
Ant: One Crippled Leg	1370	1572	**1680**
Ant: Two Crippled Legs	700	697	**795**
HalfCheetah: Big	4891	**5194**	4968
HalfCheetah: Small	5151	**5488**	5263

Finally, we note that dynamics may change *during* an episode; consider a robot that suffers a motor fault, reducing the power delivered to its joints. Evidently the underlying dynamics have been altered, and being robust to such changes when only training from a single dataset of offline experience is challenging. To illustrate this, we perform a 1500 step rollout in the HalfCheetah environment, starting with offline dynamics (mass/damping = 1), before changing to mass = 0.75, damping = 0.5 after 500 steps; performance is shown in Fig.8. Observe that after 500 steps, MOPO performance is dramatically reduced. This is because the agent continues to apply the same force and thus falls forward with lighter mass. For our AugWM agent, performance initially drops, then when the new context is learned we achieve *higher* performance than before, making use of the lighter torso.

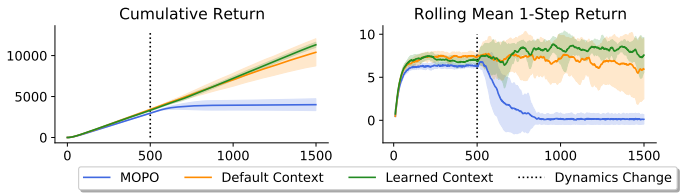


Figure 8: Performance under changing dynamics. Left = cumulative returns, Right = rolling average single step reward. Both averaged over twenty seeds, shaded area shows $\pm$1std.

Discussion We believe that our experiments provide significant support to the claim that training with AugWM improves zero-shot dynamics generalization. In a broad set of commonly used datasets, and with a wide range of out-of-distribution dynamics, our algorithm learns good policies where a state-of-the-art baseline fails.[3] This is due to a number of novel contributions: 1) using *dynamics augmentation* rather than *observation augmentation*; 2) training and testing with a context-based policy. Regarding limitations, we note that training inside the WM with context generally takes longer to converge (Appendix B). Furthermore, in more nonlinear settings such as the HalfCheetah modified body part setting we saw a reduced performance for the learned context. This could be because the dynamics changes are out of the distribution of DAS augmentations (violating Eqn. 1), or due to the difficulty of modeling the task with a linear model. We note that linear models have achieved success in planning (13) and meta learning (33), and are effective in our case due to their data efficiency, but can be replaced by more flexible models to deal with different augmentations.

6 Conclusion and Future Work

In this paper we propose Augmented World Models (AugWM), which we show sufficiently simulates changes in dynamics such that agents can generalize in a zero-shot manner. We believe that we are the first to propose this problem setting, and our results show a significant improvement over existing state-of-the-art methods which ignore this problem.

A promising line of future work would be to meta-train a policy over AugWM such that it can quickly adapt to new dynamics in the few-shot setting. There is evidence that data augmentation can improve robustness in meta-learning (37), and could extend to strong performance in out-of-distribution tasks. We also wish to consider varying goals at test time, and other potential sources of non-stationarity which may impact policy performance. It may also be possible to extend AugWM to pixel-based tasks, which have received a great deal of recent attention (17; 16). We believe that our transition based augmentations will be applicable to a latent representation, as commonly used in state-of-the-art vision MBRL approaches. Thus we think that extending our work to this setting, while a considerable feat of engineering, should not require significant methodological changes.

[3]For videos see: `https://sites.google.com/view/augmentedworldmodels/`

REFERENCES

[1] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

[2] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] R. Antonova, S. Cruciani, C. Smith, and D. Kragic. Reinforcement learning for pivoting task, 2017.

[4] A. Argenson and G. Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

[5] P. Ball, J. Parker-Holder, A. Pacchiano, K. Choromanski, and S. Roberts. Ready policy one: World building through active learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, 2020.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31*, pages 4754–4765, 2018.

[8] I. Clavera, A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019.

[9] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 617–629. PMLR, 2018.

[10] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019.

[11] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4{rl}: Datasets for deep data-driven reinforcement learning, 2021.

[12] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[13] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2829–2838, New York, New York, USA, 20–22 Jun 2016. PMLR.

[14] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NeurIPS'18, pages 2455–2467, 2018.

[15] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

[16] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[17] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, pages 2555–2565, 2019.

[18] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*, 2021.

[19] P. Henderson, W.-D. Chang, F. Shkurti, J. Hansen, D. Meger, and G. Dudek. Benchmark environments for multitask learning in continuous domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*, 2017.

[20] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *1st Conference on Robot Learning*, 2017.

[21] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

[22] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osiński, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski. Model based reinforcement learning for Atari. In *International Conference on Learning Representations*, 2020.

[23] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel : Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[24] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[25] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[26] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[27] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems 33*, 2020.

[28] M. Laskin, A. Srinivas, and P. Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[29] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[30] A. Nagabandi, C. Finn, and S. Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*, 2019.

[31] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60 vol.1, 1994.

[32] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[33] M. Peng, B. Zhu, and J. Jiao. Linear representation meta-reinforcement learning for instant adaptation. *CoRR*, 2021.

[34] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation, ICRA*, 2018.

[35] S. Pitis, E. Creager, and A. Garg. Counterfactual data augmentation using locally factored dynamics. In *Advances in Neural Information Processing Systems*, 2020.

[36] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *CoRR*, abs/2006.12862, 2020.

[37] J. Rajendran, A. Irpan, and E. Jang. Meta-learning requires meta-augmentation. In *Advances in Neural Information Processing Systems*, 2020.

[38] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 5331–5340. PMLR, 2019.

[39] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *CoRR*, abs/1911.08265, 2019.

[40] Y. Seo, K. Lee, I. Clavera, T. Kurutach, J. Shin, and P. Abbeel. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[41] X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

[42] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[43] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 5334–5344. Curran Associates, Inc., 2018.

[44] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[45] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

[46] W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. In N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, editors, *Robotics: Science and Systems XIII*, 2017.

[47] W. Zhou, L. Pinto, and A. Gupta. Environment probing interaction policies. In *International Conference on Learning Representations*, 2019.

[48] L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

APPENDIX

A ADDITIONAL EXPERIMENTS

Here we show the more granular experimental results, presented in condensed form in Table 1. In Fig. 10 we show the improvement vs. MOPO for each dimension of mass (green) or damping (blue), for HalfCheetah (top) and Walker2d (bottom).

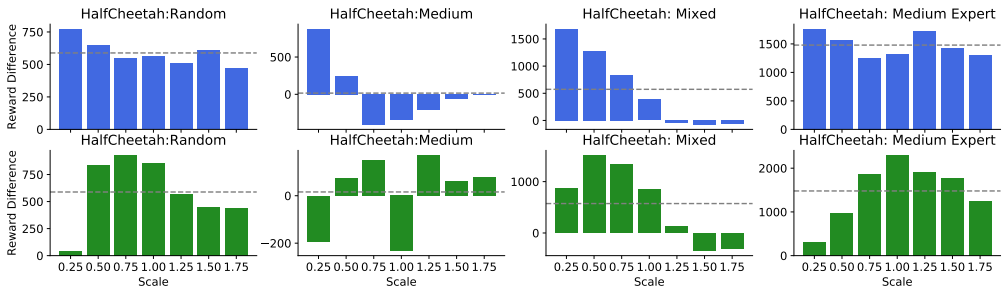


Figure 9: Mean improvement for Augmented World Models over MOPO for the HalfCheetah environment, averaged over five seeds. Top row (blue) = damping scale, bottom row (green) = mass scale. Dotted line is the mean, the same value for both damping and mass.

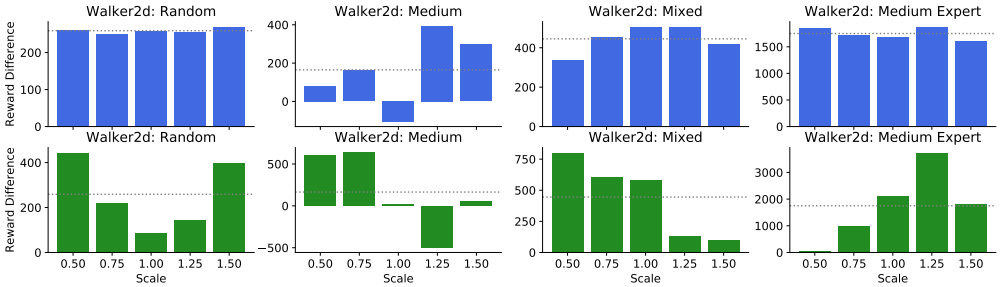


Figure 10: Mean improvement for Augmented World Models over MOPO for the Walker2d environment, averaged over five seeds. Top row (blue) = damping scale, bottom row (green) = mass scale. Dotted line is the mean, the same value for both damping and mass.

In this section we show the performance for Augmented World Models with different training ranges for the DAS augmentation (z train in Table 4). We train with adaptive context on the HalfCheetah mixed dataset, and present the results in Fig. 11. As we see, $[0.75, 1.25]$ and $[0.5, 1.5]$ perform the best. Based on this, we use $[0.5, 1.5]$ for our experiments as we believe this helps us sample a wider set of dynamics, helping us generalize better across all environments and data sets.

B IMPLEMENTATION DETAILS

B.1 HYPERPARAMETERS

Our algorithm is based on MOPO (45) with values for the rollout length h and penalty coefficient λ shown in Table 3.

AugWM specific hyperparameters are listed in Table 4. For each evaluation rollout, we clear the buffer of stored true modified environment transitions to measure zero-shot performance. We adapt using the context after a set number of steps, k, in the environment to train the linear model. The two ranges used for the context z during training and test time are different. At test time, the estimated context is clipped to remain within the given bounds.

[4]We follow the original MOPO hyperparameters for all datasets except for walker2d-medium where we found (1, 1) worked better for both MOPO and our method than (5, 5).

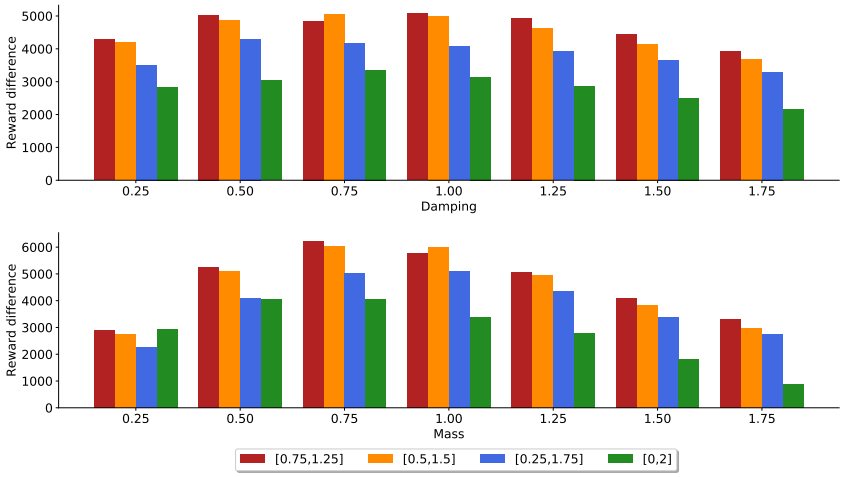


Figure 11: Performance for Augmented World Models with the DAS augmentation. Each plot shows different values for a and b, the ranges for the sampled noise.

Table 3: Hyperparameters used in the D4RL datasets.

Dataset Type	**Environment**	**MOPO (h, λ)**
random	halfcheetah	5, 0.5
random	walker2d	1, 1
medium	halfcheetah	1, 1
medium	walker2d	1, 1 [4]
mixed	halfcheetah	5, 1
mixed	walker2d	1, 1
med-expert	halfcheetah	5, 1
med-expert	walker2d	1, 2

Table 4: AugWM Hyperparameters

Parameter	**Value**
evaluation rollouts	5
MOPO offline epochs	400
AugWM offline epochs	900
k, steps for adaptation	300
z train range	[0.5, 1.5]
z test range	[0.93, 1.07]

B.2 D4RL DATASET

We evaluate our method on D4RL (11) datasets based on the MuJoCo continuous control tasks (halfcheetah and walker2d). The four dataset types we evaluate on are:

- **random:** roll out a randomly initialized policy for 1M steps.
- **medium:** partially train a policy using SAC, then roll it out for 1M steps.
- **mixed:** train a policy using SAC until a certain (environment-specific) performance threshold is reached, and take the replay buffer as the batch.
- **medium-expert:** combine 1M samples of rollouts from a fully-trained policy with another 1M samples of rollouts from a partially trained policy or a random policy.

This gives us a total of 8 experiments.

B.3 ANT ENVIRONMENT

For the Ant experiments, we follow the Ant Changed Direction approach in MOPO (45). Since this offline dataset is not provided in the authors' code, nor is it in the standard D4RL library (11), we were required to generate our own offline Ant dataset. Since the authors' did not outline certain details in their experiment, we found the following was required to match their performance with our codebase: 1) Training our SAC policy for 1×10^6 timesteps in the Ant environment provided by the authors' code in (45); 2) relabelling each reward in the buffer using the new direction, without the living reward; 3) training a world model over this offline dataset; 4) training a policy in the world model, adding in living reward post-hoc; 5) evaluating the policy with the living reward.

B.4 HALFCHEETAH MODIFIED AGENT

We use the modified HalfCheetah environments from (19). In each setting one body part of the agent is changed, from following set: $\{$Foot, Leg, Thigh, Torso, Head$\}$. The body part can either be "Big" or "Small", where Big bodyparts involve scaling the mass and width of the limb by 1.25 and Small bodyparts are scaled by 0.75. In Table 2 we show the mean over each of these five body parts, for agents trained on each of the D4RL datasets, repeated for five seeds.