Reinforcement Learning from Delayed Observations via World Models

Armin Karamzade
karamzaa@uci.edu
Department of Computer Science
University of California, Irvine

Kyungmin Kim
kyungk7@uci.edu
Department of Computer Science
University of California, Irvine

Montek Kalsi
kalsim@uci.edu
Department of Computer Science
University of California, Irvine

Roy Fox
royf@uci.edu
Department of Computer Science
University of California, Irvine

Abstract

In standard reinforcement learning settings, agents typically assume immediate feedback about the effects of their actions after taking them. However, in practice, this assumption may not hold true due to physical constraints and can significantly impact the performance of learning algorithms. In this paper, we address observation delays in partially observable environments. We propose leveraging world models, which have shown success in integrating past observations and learning dynamics, to handle observation delays. By reducing delayed POMDPs to delayed MDPs with world models, our methods can effectively handle partial observability, where existing approaches achieve sub-optimal performance or degrade quickly as observability decreases. Experiments suggest that one of our methods can outperform a naive model-based approach by up to 250%. Moreover, we evaluate our methods on visual delayed environments, for the first time showcasing delay-aware reinforcement learning continuous control with visual observations.

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful framework for training agents to make sequential decisions in their environment. In traditional RL settings, agents assume immediate observational feedback from the environment about the effect of their actions. However, in many real-world applications, observations are delayed due to physical or technological constraints on sensors and communication, challenging this fundamental assumption. Delay can arise from various sources, such as computational limitations (Dulac-Arnold et al., 2019), communication latency and interconnection (Ge et al., 2013; Rostami & Kia, 2023), or physical constraints in robotic systems (Imaida et al., 2004).

For example, drone navigation based on computation offloading might experience lag when the network is congested (Almutairi et al., 2022), or robots equipped with shielding may encounter delays in execution to ensure the safe behavior of the policy (Corsi et al., 2024). In scenarios where timely decision-making is critical and agents cannot afford to wait for updated state observations, RL algorithms must nonetheless find effective control policies subject to delay constraints. In this paper, we focus on observation delays that prevent the agent from immediately perceiving world state transitions, rather than execution delays that prevent the immediate application of the agent's control action, although these types of delay are interconnected and can in some settings be effectively addressed within a unified framework (Katsikopoulos & Engelbrecht, 2003). Specifically, at time t, the agent receives observation o_{t-d} and reward r_{t-d}, where d is the time delay.

The body of work on RL with delay has explored several approaches within the Markov Decision Process (MDP) framework. Memoryless approaches build a policy based on the last observed state (Schuitema et al., 2010). A second type of approach aims to reduce the problem into an undelayed MDP by extending the states with additional information, typically the actions taken since the last available observation (Walsh et al., 2007; Derman et al., 2020). Finally, recent approaches compute, from the extended state, perceptual features predictive of the hidden current state to inform action selection (Chen et al., 2021; Liotet et al., 2021; 2022; Wang et al., 2024). While there are many existing works on delays in MDPs, surprisingly few study delays in Partially Observable MDPs (POMDPs) where the delayed observations are non-Markov (Kim & Jeong, 1987; Varakantham & Marecki, 2012), and these works do not provide a learning paradigm.

World models have recently shown significant success in integrating past observations and learning the dynamics of the environment (Ha & Schmidhuber, 2018). These models, comprising a representation of the environment's state, a transition model depicting state evolution over time, and an observation model linking states to observations, have proven effective in capturing intricate temporal dependencies and enhancing decision-making. One such family is Dreamer (Hafner et al., 2023), a model-based RL framework that trains the agent through trajectories simulated by a learned world model. Dreamer benefits from the sample efficiency inherent in model-based RL techniques and is relatively insensitive to task-specific hyperparameter tuning.

In this paper, we propose leveraging world models to learn in the face of observation delays. We employ world models to form the extended state in the latent space, demonstrating that this latent extended state contains sufficient information for the current delayed state. This suggests two different strategies for adapting world models to POMDPs with observation delay: either by directly modifying the policy or by predicting the delayed latent state with imagination. While naively using world models for delays can lead to significant performance degradation as the delay increases, our methods exhibit greater resilience and one of them improves policy value upon the naive baseline by approximately 250%. Despite their simple implementation, these modifications achieve better or comparable performance to other approaches without the need for domain-specific hyperparameter tuning. Moreover, we evaluate our methods not only on vector inputs, but on continuous control tasks with visual inputs, a crucial aspect that was missing in the delayed RL community.

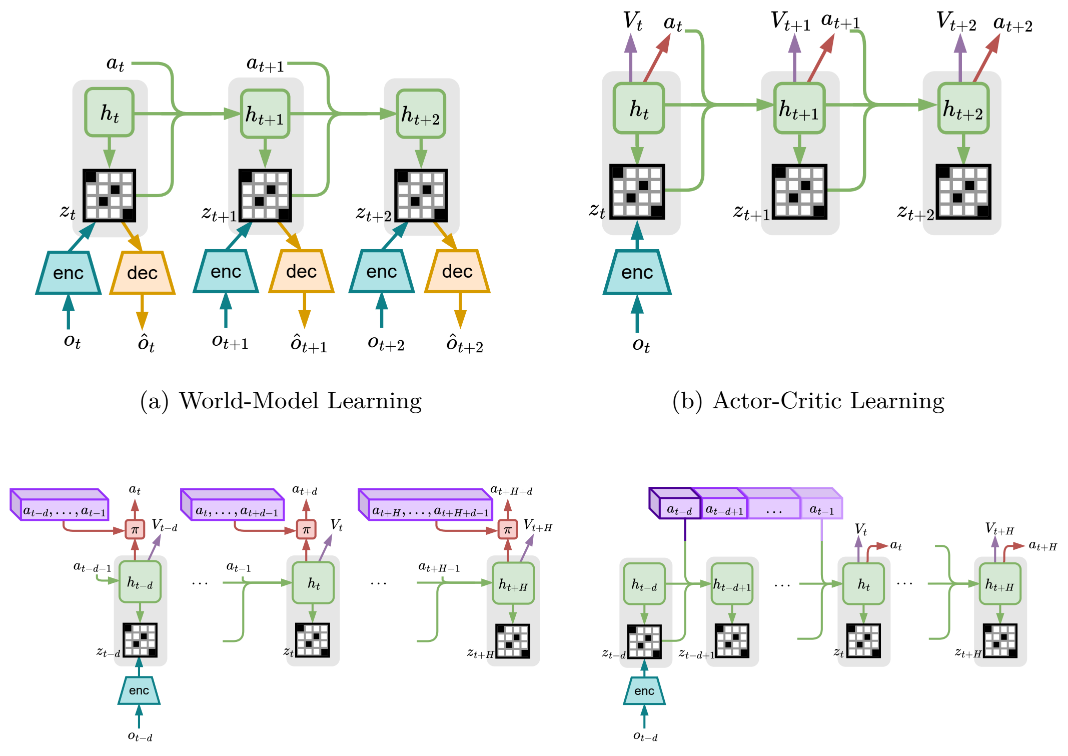
Contributions of this paper are summarized as follows:

- We propose three methods that use world models to address observation delays. As a case study, we have adapted Dreamer-V3 to evaluate the effectiveness of our proposed strategies.
- We formalize observation delays in POMDPs and establish a link between delays in MDPs and POMDPs.
- We conduct extensive experiments that, among other domains, benchmark for the first time delayed RL in visual domains that are inherently partially observable.

2 Related Work

Several prior works have addressed delays in the MDP framework. One line of research employs a memoryless approach, where an agent uses only the last available state as input. For instance, Schuitema et al. (2010) proposed dSARSA, a memoryless extension of SARSA (Sutton & Barto, 2018) for delays in MDPs. Although this approach perceives the environment with partial information, it was shown to work quite well in some domains.

Another approach is to reduce a delayed MDP, which is known to be a structured POMDP, to an extended-state MDP by augmenting the recently observed state with the actions that have been



(a) World-Model Learning (b) Actor-Critic Learning

(c) Extended Actor (d) Latent State Imagination

Figure 1: Panels (a) and (b) depict the standard Dreamer learning process, while (c) and (d) illustrate two strategies for adapting Dreamer for observation delays. (see section 4.2 and 4.3)

taken since then (Walsh et al., 2007; Derman et al., 2020; Bouteiller et al., 2020). For instance, DCAC (Bouteiller et al., 2020) extends SAC (Haarnoja et al., 2018) to take the extended state and achieves good sample efficiency through resampling techniques, but it suffers from an exponentially growing input dimension as the delay increases.

Recent strategies focus on deriving useful features from the extended state for policy input. Walsh et al. (2009) developed a deterministic dynamics model to predict the unobserved state. Chen et al. (2021) employed a particle-based method to simulate potential current state outcomes. Similarly, D-TRPO (Liotet et al., 2021) obtains a belief representation of the current state using a normalizing flow, enhancing policy input with these features.

More recently, Liotet et al. (2022) applied imitation learning to train a delayed agent using an expert policy from an undelayed environment, though this approach is constrained by the need for access to the undelayed environment, limiting its practicality. Concurrently with our work, Wang et al. (2024) introduces a method for delay-reconciled training that integrates a critic and an extended-state actor and Valensi et al. (2024) uses EfficientZero (Ye et al., 2021) for inferring future states, similarly to one of our methods.

3 Preliminaries

3.1 Delayed POMDPs

A Partially Observable Markov Decision Process (POMDP) consists of a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, \mathcal{O}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, $\mathcal{T}(s', r|s, a)$ is the joint probability distribution of the next state and reward, Ω is the set of observations, $\mathcal{O}(o|s)$ is the conditional probability of observing $o \in \Omega$ in state s, and $\gamma \in [0, 1)$ is the discount factor. At each timestep t, in state s_t the agent receives an observation $o_t \sim \mathcal{O}(o_t|s_t)$ and selects an action a_t, causing a transition and reward with distribution $\mathcal{T}(s_{t+1}, r_t|s_t, a_t)$. The goal is to maximize the expected return $\mathbb{E}[\sum_{t \geq 0} \gamma^t r_t]$.

We define a Delayed Partially Observable Markov Decision Process (DPOMDP) as a tuple $\langle \mathcal{P}_o, \mathcal{D} \rangle$, where $\mathcal{P}_o$ represents a POMDP and $\mathcal{D}$ is a distribution dictating the delay in receiving observations from $\mathcal{P}_o$. Specifically, at time t, the agent receives observation o_{t-d} and reward r_{t-d}, where $d \sim \mathcal{D}$. The objective is to maximize the expected return while operating in $\mathcal{P}_o$ under these delays. For simplicity, we assume in this work that $\mathcal{D}$ generates a constant non-negative integer delay d, though our methods can be easily generalized to handle random delays as well.

In a similar fashion, delayed Markov Decision Processes (DMDPs) are defined in previous works (Wang et al., 2024; Liotet et al., 2022) as a tuple $\langle \mathcal{M}, \mathcal{D} \rangle$, where $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma \rangle$ is an MDP. Like their undelayed counterparts, DMDPs can be considered a special case of DPOMDPs in which the (delayed) observation gives full information about the state at that time. Any DMDP can be reduced to an extended MDP $\widetilde{\mathcal{M}}$ by defining the states as a concatenation of the last observed state and the subsequent actions, which the agent needs to explicitly remember (Altman & Nain, 1992). In particular, $\widetilde{\mathcal{M}} = \langle \mathcal{X}, \mathcal{A}, \widetilde{\mathcal{T}}, \gamma \rangle$, where $\mathcal{X} = \mathcal{S} \times \mathcal{A}^d$ and for $x = \left(s, a^1, \ldots, a^d\right), x' = \left(s', a'^1, \ldots, a'^d\right) \in \mathcal{X}$,

$$\widetilde{\mathcal{T}}(x', r|x, a) = \mathcal{T}(s', r|s, a^1) \mathbb{1}([a'^1, \ldots, a'^d] = [a^2, \ldots, a^d, a]). \tag{1}$$

By definition, any DPOMDP is another POMDP with a specific structure. It can be formally constructed by defining the set of states in the new PODMP as the concatenation of the last $d+1$ states, of which only the earliest one is observable, and defining the transition probabilities accordingly. Thus, we can employ the POMDP framework to tackle the observation delays by constructing the equivalent POMDP (Varakantham & Marecki, 2012). On the other hand, one can exploit the structure introduced by delay in the process. Specifically, delay in receiving observations is equivalent to delay in inferring the latent states. As latent space enjoys the Markov property, we can then work with the DMDP defined over latent states and induced by delay in observations. Proposition 3 in section 4.1 formalizes this intuition. Note, however, that there is a tradeoff between these two choices: on the one hand, POMDPs are harder to learn than MDPs; on the other hand, in the POMDP formulation, the world remembers the history for us, avoiding the curse of dimensionality in explicit agent context.

3.2 World models

World models (Schmidhuber, 2015; Ha & Schmidhuber, 2018) simulate aspects of the environment by learning an internal representation through an encoder and a dynamics model. The encoder compresses high-dimensional inputs, such as image observations, into a lower-dimensional embedding, while the dynamics model forecasts future states from historical information. This streamlined state representation then serves as input to an RL agent.

Dreamer (Hafner et al., 2020; 2021; 2023), a pioneering model-based RL (MBRL) approach utilizing world models, surpasses many model-free RL algorithms in data efficiency and performance. Dreamer's training involves three alternating phases: 1) training the world model on past experiences; 2) learning behaviors with actor–critic algorithm through imagined sequences; and 3) collecting data in the environment.

Figure 1a depicts the world model comprising an encoder-decoder and a Recurrent State Space Model (RSSM) (Hafner et al., 2019) to model dynamics. Alongside reconstructed observation $\hat{o}_t$, the model includes prediction heads for reward and episode continuation, omitted for clarity. The model state $m_t = [h_t, z_t]$ comprises deterministic and stochastic components, respectively, given by

$$h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \tag{2}$$

$$z_t \sim q_\phi(z_t|o_t, h_t). \tag{3}$$

Dreamer also learns a dynamic predictor for estimating imagined latent states as

$$z'_t \sim p_\phi(z'_t|h_t). \tag{4}$$

As (2) and (3) suggest, the latent states are constructed to have a Markov structure, regardless of whether the observations themselves are Markov. In other words, Dreamer attempts to learn a latent-state MDP that is as nearly as possible equivalent to the observed POMDP. If it succeeds, this equivalence means that, for any policy $\pi(a|m)$, the stochastic process $\{m_t, a_t, r_t\}_{t\geq 0}$ induced by the policy in the world model (Figure 1b) has the same joint distribution as the embedding of real policy rollouts (Figure 1a). Figure 1b also illustrates Dreamer's behavior learning using an actor–critic method, policy and value heads on latent trajectories predicted by the world model.

3.3 Hardness of delayed control

Before presenting our methods, we provide an example to show that optimal values in DMDPs are generally incomparable for different delays. Depending on the stochasticity of the environment and the length of the delay, the optimal value function can be made arbitrary worse compared to that of the undelayed environment.

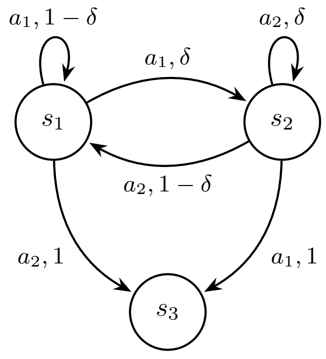


Figure 2

Let V^* be the optimal value function of the MDP sketched in Figure 2 and similarly $\widetilde{V}^*$ be the optimal value function with constant observation delay of 1. Starting at state s_1, the agent will receive a reward of $+1$ for taking a_1 in s_1 and 0 otherwise. $0 \leq \delta \leq \frac{1}{2}$ controls the stochasticity of the environment. To maximize the expected return, the agent should try to stay in s_1 by taking a_1 in s_1 and a_2 in s_2. When there is no delay, the agent can take the appropriate action and avoid the absorbing state s_3. However, with delay, the agent does not observe the current state and for $0 < \delta$ it eventually ends up in s_3 for any policy. The ratio between the optimal values of the delayed and undelayed case can be computed (see appendix A) as

$$\frac{\widetilde{V}^*(s_1)}{V^*(s_1)} = \frac{(1-\gamma)}{(1-\gamma\delta)\,(1-\gamma(1-\delta))}.$$

When $\delta = 0$ the ratio is 1, while the minimal ratio of $\frac{1-\gamma}{(1-\gamma/2)^2}$ is obtained for $\delta = \frac{1}{2}$ with the ratio approaching 0 as $\gamma \to 1$. These two extreme cases correspond to the scenarios with the least and the most stochasticity in the transitions, respectively.

In general, depending on the underlying MDP, even introducing small observation delays could downgrade the optimal policies much. By assuming smooth transition dynamics and rewards, Liotet et al. (2022) bounded this gap as a function of smoothness parameters of the underlying MDP.

4 Delayed Control via World Models

In this section, we begin with a seemingly simple yet crucial insight into the relationship between converting POMDPs into MDPs via world models, and translating DPOMDPs into DMDPs. This insight forms the basis for combining techniques initially developed for POMDPs and DMDPs to tackle delays in partially observable environments. Next, we elaborate on the adaptations required to incorporate delays and examine two distinct methodologies within this framework.

4.1 World models reduce DPOMDPs to DMDPs

A world model, denoted by $\widehat{M}$, has two modes of operation: *imagination*, where it can operate as a stateful simulator of the world with which the agent can interact (Figure 1b); and *interaction*, where it can ground its latent state m_t in real observations by sequentially incorporating their embedding into the state (Figure 1a). In some world models, imagination is implemented using the interaction mode, by replacing the real embedded observations by their reconstruction from the latent state (Ha & Schmidhuber, 2018). In Dreamer, the two modes are modeled separately through (3) and (4) and kept equivalent in training by a dedicated loss term. In the following, we keep our discussion general by not constraining the functional form of the world model, and only requiring the two modes to be equivalent in the sense of the following definition.

Definition 1. *A world model $\widehat{M}$ is congruent with a POMDP $\mathcal{P}_o$ if, for any action sequence $\vec{a} = \{a_t\}_{t\geq 0}$, the stochastic process $\{o_t, m_t, r_t\}_{t\geq 0}$ induced by rolling out $\vec{a}$ in $\mathcal{P}_o$ and feeding its observations into $\widehat{M}$'s interaction mode has the same joint distribution (marginalized over $\{o_t\}_{t\geq 0}$) as the process $\{m_t, r_t\}_{t\geq 0}$ induced by rolling out $\vec{a}$ in $\widehat{M}$'s imagination mode.*

Definition 2. *Given a world model $\widehat{M}$, the operation of the d-step delayed world model $\widehat{M}^d$ is defined as follows: in* imagination *step t, the delayed state m_{t-d} (or dummy for $t < d$) is read from $\widehat{M}$, action a_t is taken in $\widehat{M}$, and the delayed reward r_{t-d} (or 0 for $t < d$) is returned; in* interaction *step $t \geq d$ (following d dummy steps), the current observation $\tilde{o}_t = o_{t-d}$ is fed into $\widehat{M}$, the current state $\tilde{m}_t$ is read, the action a_t is taken in the delayed environment but $\tilde{a}_t = a_{t-d}$ in $\widehat{M}$, and the delayed environment reward $\tilde{r}_t = r_{t-d}$ is returned.*

Proposition 3. *If a world model $\widehat{M}$ is congruent with a POMDP $\mathcal{P}_o$, then the d-step delayed world model $\widehat{M}^d$ is congruent with the d-step delayed DPOMDP $\mathcal{P}_o^d$.*

Proof. For a given action sequence $\vec{a}$, let $\{\tilde{o}_t, \tilde{r}_t\}_{t\geq 0}$ be the stochastic process induced by rolling out $\vec{a}$ in $\mathcal{P}_o^d$. In $\widehat{M}^d$'s interaction mode, the first d dummy steps are skipped, and then the sequence $\{\tilde{o}_t, \tilde{a}_t\}_{t\geq d} = \{o_{t-d}, a_{t-d}\}_{t\geq d}$ is fed into $\widehat{M}$. Because this is the same process as $\widehat{M}$'s interaction with $\mathcal{P}_o$ using $\vec{a}$, it has the same distribution over $\{\tilde{m}_t, \tilde{r}_t\}_{t\geq d}$ as $\widehat{M}$'s imagination process. It remains to be verified that prepending d dummy states and rewards to $\widehat{M}$'s imagination process yields $\widehat{M}^d$'s imagination process with the same action sequence, completing the proof. □

Proposition 3 solves a critical issue with training world models in delayed environments: training $\widehat{M}^d$ with a blackbox $\mathcal{P}_o^d$ treats the latter as a POMDP and fails to leverage its specific structure as a DPOMDP (see Sec. 3.1). Instead, we can recover $\mathcal{P}_o$ from $\mathcal{P}_o^d$ by shifting back the delays in training time, train $\widehat{M}$ for $\mathcal{P}_o$, and then Definition 2 gives us the structure of $\widehat{M}^d$ in terms of $\widehat{M}$. Proposition 3 guarantees that this process indeed models $\mathcal{P}_o^d$ correctly.

4.2 Delay-aware training

The learning of the world model relies on data stored in an experience replay buffer, accumulated by the agent throughout training. With delays, the storage of these data remains unaffected, as the data collection mechanism can store a transition once the subsequent observation becomes available; thus, the world model can be trained using undelayed data $\{(o_t, a_t, r_t)\}_{t\geq 0}$. However, the distribution of collected samples is influenced by the fact that actions are selected without observing the past d states of the environment. This discrepancy leads to a divergence in the distribution of data trajectories between delayed and undelayed environments. Nevertheless, as we will see in the experiment section, the world model can still learn successfully.

In contrast to world model learning, the actor–critic component must take delays into account, as the agent is required to select a_t based on the available information at time t, generated at time $t-d$. To account for delayed observations, there are two primary design strategies: either to design the policy as $\pi(\cdot|x_t)$, conditioned on the extended state x_t, or condition the policy on the latent state $\hat{m}_t$ predicted from x_t. In both cases, the agent needs access to the action sequence $(a_{t-d}, ..., a_{t-1})$. Thus, we augment the experience replay buffer to store subsequent actions. In the following two sections, we will explore each of these design approaches.

4.3 Delayed actor-critic

In actor–critic learning, the critic provides an estimate of the value function V to aid the learning of the actor, while the actor aims to maximize the return guided by the critic's value. Since the world model is a DMDP and the extended state $x_t = (m_{t-d}, a_{t-d}, \ldots, a_{t-1})$ is a state of its equivalent extended MDP, it is sufficient to condition a_t on x_t. The critic, in contrast, only provides feedback in training time and can therefore wait to see the true state to provide a more accurate value estimate to the actor. This idea has also been explored concurrently with this work (Wang et al., 2024). Thus, to directly handle observation delays in the policy, we can use the extended state to design the policy $\pi(a_t|x_t)$ along with the same critic as in the undelayed case $V(m_t)$. We refer to this method as an **Extended** actor. In practice, the policy network can be implemented with any neural architecture, such as a Multi-Layer Perceptrons (MLP), Recurrent Neural Networks (RNN), or Transformers.

Figure 1c illustrates the Extended actor diagram adapted for Dreamer. At time t, the agent retrieves x_t from the replay buffer and performs on-policy actor–critic learning in imagination by updating the extended state with the next action. In particular, $a_t \sim \pi_\theta(a_t|x_t)$ and $x_{t+1} = (m_{t-d+1}, \{a_i\}_{i=t-d+1}^{t})$ where m_{t-d+1} is the imagined latent state in (4). Note that the critic predicts the value for the current latent state $V_\psi(m_{t-d})$ while the actor outputs a_t based on x_t. Thus, the imagination horizon will be increased for d additional time steps since the critic provide feedback for actor's action d timesteps later. The estimates of the critic and the actions of the actor are then realigned to compute the policy gradient loss function.

Another variant of the extended actor involves drawing actions from the policy $\pi(a_t|m_{t-d})$ without maintaining a memory to track previously performed actions, a concept referred to as the **Memoryless** actor. While this design choice might appear to lack the ability to capture the necessary information, the rationale behind it is that the policy $\pi(a_t|m_{t-d})$ can theoretically represent the extended state's previously performed actions within its network. This is because no new information is introduced after time $t - d$, and therefore, no additional memory is needed to store those actions.

4.4 Latent state imagination

Another approach is to estimate the current latent state $\hat{m}_t$, without modifying the policy architecture, and draw actions from $\pi(a_t|\hat{m}_t)$. Specifically, we can use the world model's forward dynamics in (2) and (4) (Figure 1b) for d time steps, starting with m_{t-d}, to sample a prediction $\hat{m}_t$. Then, the agent uses $\hat{m}_t$ as the current latent state of the environment both in training and inference time. Figure 1d depicts this process, which we refer to as a **Latent** actor. After computing $\hat{m}_t$, the agent performs training or policy execution the same way as in the undelayed case.

Note that estimation of the current state happens in the latent space, otherwise this approach will lead to suboptimal decisions as the agent needs to form an approximate belief over the hidden state to act optimally in the presence of delays. In other words, the agent should account for uncertainty over the true state of the environment. This also implies that the latent state should have a deterministic component to allow the agent to avoid losing information through sampling. Furthermore, Extended and Latent actors do not assume that d is constant. The Latent actor can imagine for variable step lengths and Extended can employ RNNs or pad the action sequence with a special action to manage delay variability. In principle, Memoryless actors can also handle stochastic delays by conditioning the policy on d, though this has uncertain practical effectiveness.

5 Experiments

5.1 Experimental setup

Tasks. To evaluate our proposed methods, we conducted experiments across a diverse set of environments. We have considered four continuous control tasks from MuJoCo (Todorov et al., 2012) in Gymnasium (Gym) (Towers et al., 2023): HalfCheetah-v4, HumanoidStandup-v4, Reacher-v4, and Swimmer-v4 for comparison with previous studies. Also, we extended our evaluation to six more environments from the DeepMind Control Suite (DMC) (Tunyasuvunakool et al., 2020), to further examine our methods with both proprioceptive and visual observations. The distinction between these input types is critical; vector inputs provide a fully observable state of the environment, but image-based observations introduce partial observability, necessitating approaches capable of addressing delays within the POMDP framework.

Methods. We utilized Dreamer-V3 (Hafner et al., 2023) as our primary framework[1] and compare with prior studies, including D-TRPO (Liotet et al., 2021) and DC-AC (Bouteiller et al., 2020). We also evaluated a Dreamer-V3 agent, similar to the Latent method, except trained in an undelayed setting and tested under delayed conditions with latent imagination. This approach, referred to as **Agnostic**, can be e considered a naive use of world models to address delays. Although our methods can handle both constant and random delays, we chose to focus on fixed delays for simplicity and comparability with the baselines. While we experiment in both Gym and DMC, the baselines were originally designed and tuned for Gym environments with vector inputs, and we found the task of modifying them for image observations or tuning their hyperparameters for DMC nontrivial (Cetin et al., 2022) and thus outside our scope. Also, for Gym environments, we trained Dreamer variants with a budget of 500K interactions, while D-TRPO and DC-AC trained with 5M and 1M environment interactions, respectively. For DMC tasks with visual inputs, we have increased the number of interactions to 1M.

Architectures and hyperparameters. For D-TRPO, we adopted the hyperparameters and architecture detailed in Liotet et al. (2022). Similarly, for DC-AC, we replicated the hyperparameters and model from the original paper (Bouteiller et al., 2020). In our delayed variants of Dreamer-V3, we maintained consistency by using the same set of hyperparameters and architecture as the original implementation (Hafner et al., 2023). The only architectural adjustment was for the extended agent, where we incorporated a Multi-Layer Perceptron (MLP) for the policy network to extend the latent state with actions. Note that while we used the same set of hyperparameters provided in the original Dreamer, we conjecture that the optimal horizon length should be smaller in both the Extended and Latent methods, where the effective horizon is longer due to the action buffer (Figures 1c and 1d), because accumulation of one-step errors in imagination via forward dynamics could harm the actor-critic learning part. Each experiment has been repeated with 5 random seeds.

Note that, in all experiments, the agent will perform random actions until the first observation becomes available. While one could utilize better strategies for initial actions, using random actions is common in all existing delayed RL methods.

5.2 Results

Figures 3a–3d depict the results obtained from the experiments conducted on the selected Gym environments. The Dreamer variants demonstrate a significant performance improvement over D-TRPO across all tasks. However, DC-AC exhibits comparable performance to our methods on HalfCheetah-v4 and HumanoidStandup-v4, outperforms them on Reacher-v4, and underperforms on Swimmer-v4. One reason why our methods are not performing well on Reacher-v4 could be underperformance of the standard Dreamer, trained and tested on the undelayed environment itself. This is evident as DC-AC achieves comparable or superior performance to the standard Dreamer on these environments with small delays. One potential explanation for this phenomenon could be the significant portion of samples in actor–critic learning that fall outside the planning horizon. This is likely due to our use of the same imagination horizon length $H = 16$ and an episode length of 50 for this task.

Figures 3e and 3f display the performance of our methods averaged over selected suites in DMC with proprioceptive and image inputs, respectively. Remarkably, the Agnostic method performs very well across all tasks without knowing about the delay in training time. Also, the Agnostic method needs not know the delay distribution beforehand and can be deployed on any delayed environment. However, as delay increases the performance drops more rapidly than for other methods. This is

[1]The code is available at `https://github.com/indylab/DelayedDreamer`.

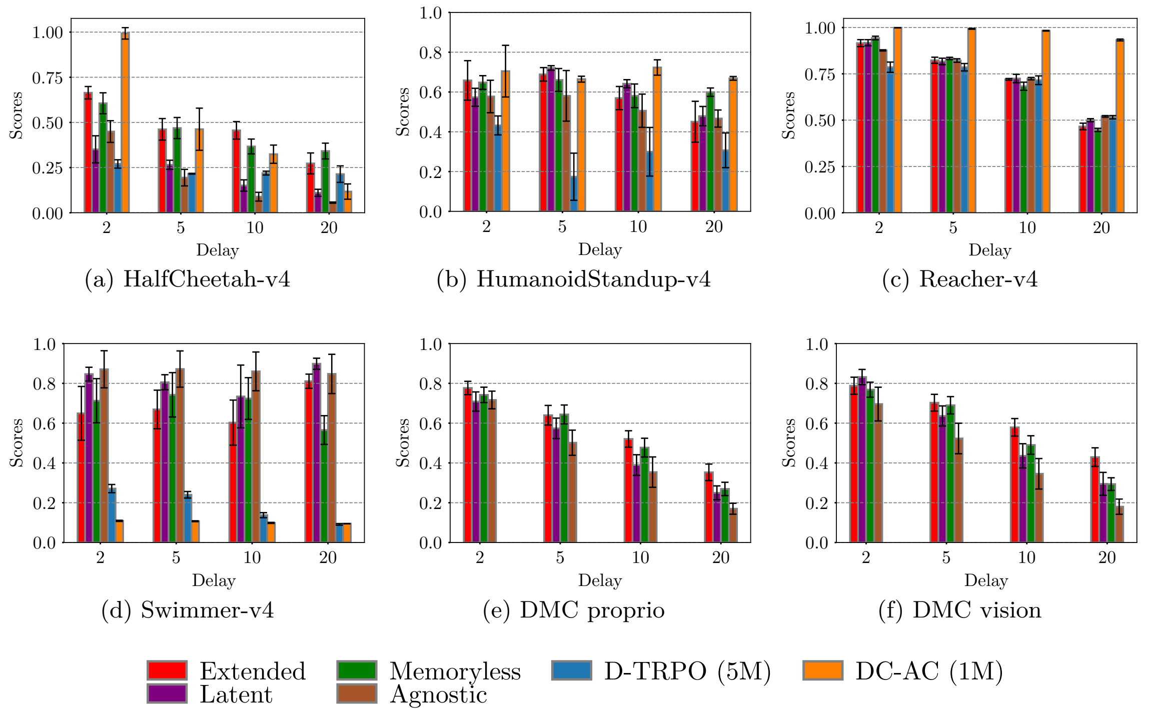


Figure 3: Normalized returns across different environments for varying delays. Bars and caps represent the mean and standard error of the mean over 5 trials, respectively. Panels (e) and (f) are averaged over the selected suites in DMC, after normalizing the agent in the undelayed environment to 1 and the random policy to 0.

because the distribution shift between the undelayed training and delayed evaluation increases for larger delays. Similarly, the Latent method does not exhibit robustness against long delays, as we are using a one-step prediction world model. The accumulation of one-step prediction errors over longer delays causes the predicted latent state to diverge significantly from the true latent state. As expected, Extended proves to be the most robust among our variants, as it utilizes next actions and avoids the accumulation of errors present in Latent and Agnostic. Notably, Extended improves by 250% on average in DMC vision tasks compared to Agnostic.

Additionally, we include training curves and tables summarizing the final test performance for all tasks in Appendix B.

5.3 Degree of observability

Figure 4 illustrates the resilience of the Extended method and the baselines faced with an increasing level of partial observability in the HalfCheetah-v4 environment. Originally, the environment's observations encompass both the positional and the velocity information for the agent's joints. To simulate partial observability, we modified the environment to omit a ρ percentage of the velocity components. The results demonstrate that both Extended and D-TRPO were capable of inferring the missing velocity components from historical observations. Although D-TRPO is specifically designed for delays in MDPs, in this particular scenario, it was able to compute the relative velocities by utilizing a transformer used to process the extended state. In contrast, DC-AC deteriorated significantly as environments became less observable.

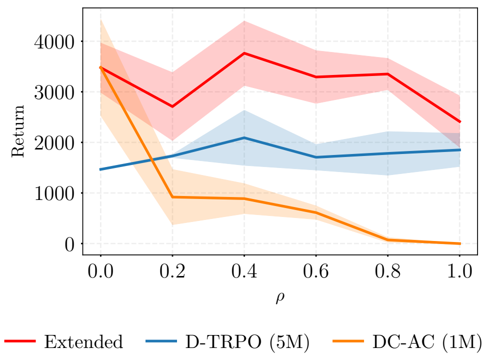


Figure 4: Return against the degree of observability in HalfCheeth-v4 for $d = 5$.

5.4 Takeaways

Although world models work as a good proxy of the true environment during training, we found that applying world models naively at test time (Agnostic) is not the most effective strategy. Our experiments revealed that for shorter delays, the Memoryless and Latent approaches work the best while keeping the original architecture unchanged. However, as delays get longer they degrade the performance due to the issues of a lack of action memory and the accumulation of one-step errors issues, respectively. The Extended method, on the other hand, can maintain its performance at the expense of adding architectural complexity to the undelayed model.

6 Conclusion

In this paper, we have proposed using world models for delayed observation within the POMDP framework. To showcase our methods, we adapted Dreamer-V3 for delay in observations and proposed two strategies, one using a delayed actor and the other latent state imagination. We discussed another version of the delayed actor which operates without action memory and additionally introduced a delay-agnostic strategy which needs not know the delay distribution beforehand. Evaluation revealed that the best of our methods, Extended, is robust to partial observability of the environment and can outperform the baselines overall, but can be sensitive to the tuning of hyperparameters.

7 Acknowledgement

This work was funded in part by the National Science Foundation (Award #2321786).

References

Jaber Almutairi, Mohammad Aldossary, Hatem A Alharbi, Barzan A Yosuf, and Jaafar MH Elmirghani. Delay-optimal task offloading for uav-enabled edge-cloud computing systems. *IEEE Access*, 10:51575–51586, 2022.

Eitan Altman and Philippe Nain. Closed-loop control with delayed information. *ACM sigmetrics performance evaluation review*, 20(1):193–204, 1992.

Yann Bouteiller, Simon Ramstedt, Giovanni Beltrame, Christopher Pal, and Jonathan Binas. Reinforcement learning with random delays. In *International conference on learning representations*, 2020.

Edoardo Cetin, Philip J Ball, Steve Roberts, and Oya Celiktutan. Stabilizing off-policy deep reinforcement learning from pixels. In *International Conference on Machine Learning*, 2022.

Baiming Chen, Mengdi Xu, Liang Li, and Ding Zhao. Delay-aware model-based reinforcement learning for continuous control. *Neurocomputing*, 450:119–128, 2021.

Davide Corsi, Guy Amir, Andoni Rodriguez, Cesar Sanchez, Guy Katz, and Roy Fox. Verification-guided shielding for deep reinforcement learning. *arXiv preprint arXiv:2406.06507*, 2024.

Esther Derman, Gal Dalal, and Shie Mannor. Acting in delayed environments with non-stationary markov policies. In *International Conference on Learning Representations*, 2020.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Yuan Ge, Qigong Chen, Ming Jiang, and Yiqing Huang. Modeling of random delays in networked control systems. *Journal of Control Science and Engineering*, 2013:8–8, 2013.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

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*, pp. 2555–2565. PMLR, 2019.

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

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Takashi Imaida, Yasuyoshi Yokokohji, Toshitsugu Doi, Mitsushige Oda, and Tsuneo Yoshikawa. Ground-space bilateral teleoperation of ets-vii robot arm by direct bilateral coupling under 7-s time delay condition. *IEEE Transactions on Robotics and Automation*, 20(3):499–511, 2004.

Konstantinos V Katsikopoulos and Sascha E Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *IEEE transactions on automatic control*, 48(4):568–574, 2003.

Soung Hie Kim and Byung Ho Jeong. A partially observable markov decision process with lagged information. *Journal of the Operational Research Society*, 38:439–446, 1987.

Pierre Liotet, Erick Venneri, and Marcello Restelli. Learning a belief representation for delayed reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Pierre Liotet, Davide Maran, Lorenzo Bisi, and Marcello Restelli. Delayed reinforcement learning by imitation. In *International Conference on Machine Learning*, pp. 13528–13556. PMLR, 2022.

Mohammadreza Rostami and Solmaz S Kia. Federated learning using variance reduced stochastic gradient for probabilistically activated agents. In *2023 American Control Conference (ACC)*, pp. 861–866. IEEE, 2023.

Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*, 2015.

Erik Schuitema, Lucian Buşoniu, Robert Babuška, and Pieter Jonker. Control delay in reinforcement learning for real-time dynamic systems: A memoryless approach. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3226–3231. IEEE, 2010.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

David Valensi, Esther Derman, Shie Mannor, and Gal Dalal. Tree search-based policy optimization under stochastic execution delay. *arXiv preprint arXiv:2404.05440*, 2024.

Pradeep Varakantham and Janusz Marecki. Delayed observation planning in partially observable domains. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1235–1236, 2012.

Thomas J Walsh, Ali Nouri, Lihong Li, and Michael L Littman. Planning and learning in environments with delayed feedback. In *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18*, pp. 442–453. Springer, 2007.

Thomas J Walsh, Ali Nouri, Lihong Li, and Michael L Littman. Learning and planning in environments with delayed feedback. *Autonomous Agents and Multi-Agent Systems*, 18:83–105, 2009.

Wei Wang, Dongqi Han, Xufang Luo, and Dongsheng Li. Addressing signal delay in deep reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Z8UfDs4J46.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25476–25488. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d5eca8dc3820cad9fe56a3bafda65ca1-Paper.pdf.

A MDP example

The optimal policy will select a_1 in s_1 and a_2 in s_2. Then, by Bellman optimal equation for the state-value function we have

$$V^*(s_1) = 1 + \gamma(1-\delta)V^*(s_1) + \gamma\delta V^*(s_2) \tag{5}$$
$$V^*(s_2) = \gamma(1-\delta)V^*(s_1) + \gamma\delta V^*(s_2), \tag{6}$$

which yields $V^*(s_1) = \frac{1-\gamma\delta}{1-\gamma}$. In the case of observation delay with $d = 1$, the optimal policy will select a_1 in every state since $\frac{1}{2} \leq \delta$ and thus,

$$\widetilde{V}^*(x_1) = (1-\delta)\left(1 + \gamma\widetilde{V}^*(x_1)\right), \tag{7}$$

where x_1 is the extended state (s_1, a_1). For the current state of the environment in s_1, we have $\widetilde{V}^*(s_1) = 1 + \gamma\widetilde{V}^*(x_1)$. Therefore, $\widetilde{V}^*(s_1) = \frac{1}{1-\gamma(1-\delta)}$.

B Experiment Details

In this section, we have included the training curves and the final results of our experiments across the selected environments in Gym and DMC. In order to have a fair comparison between the methods, we have used the same random seed for generating random actions at the beginning of the episode. We refer to DMC tasks with proprioceptive and image observations as DMC proprio and vision, respectively.

B.1 Gym results

Task	Delay	Extended	Memoryless	Latent	Agnostic	D-TRPO	DC-AC
HalfCheetah-v4 ($\times 10^3$)	2	5.13 ± 0.28	4.65 ± 0.47	2.57 ± 0.61	3.37 ± 0.49	1.91 ± 0.18	$\mathbf{7.82 \pm 0.25}$
	5	3.47 ± 0.48	$\mathbf{3.53 \pm 0.47}$	1.87 ± 0.20	1.28 ± 0.37	1.46 ± 0.01	3.48 ± 0.95
	10	$\mathbf{3.43 \pm 0.39}$	2.70 ± 0.33	0.93 ± 0.25	0.42 ± 0.19	1.49 ± 0.08	2.35 ± 0.41
	20	1.93 ± 0.47	$\mathbf{2.49 \pm 0.35}$	0.60 ± 0.16	0.15 ± 0.02	1.44 ± 0.37	0.65 ± 0.34
HumanoidStandup-v4 ($\times 10^5$)	2	1.51 ± 0.18	1.49 ± 0.06	1.36 ± 0.08	1.37 ± 0.15	1.11 ± 0.09	$\mathbf{1.60 \pm 0.23}$
	5	1.57 ± 0.06	1.52 ± 0.10	$\mathbf{1.63 \pm 0.02}$	1.37 ± 0.23	0.64 ± 0.21	1.52 ± 0.03
	10	1.35 ± 0.11	1.37 ± 0.11	1.48 ± 0.04	1.24 ± 0.15	0.87 ± 0.22	$\mathbf{1.63 \pm 0.07}$
	20	1.14 ± 0.19	1.41 ± 0.04	1.19 ± 0.09	1.17 ± 0.08	0.88 ± 0.16	$\mathbf{1.53 \pm 0.02}$
Reacher-v4	2	-7.8 ± 0.7	-6.8 ± 0.4	-7.8 ± 0.6	-9.4 ± 0.1	-12.9 ± 1.1	$\mathbf{-4.6 \pm 0.0}$
	5	-11.4 ± 0.6	-11.0 ± 0.2	-11.7 ± 0.7	-11.5 ± 0.4	-12.9 ± 0.8	$\mathbf{-4.8 \pm 0.1}$
	10	-15.4 ± 0.2	-16.8 ± 0.8	-15.2 ± 0.9	-15.3 ± 0.3	-15.6 ± 0.9	$\mathbf{-5.3 \pm 0.1}$
	20	-25.3 ± 0.7	-26.0 ± 0.3	-23.9 ± 0.3	-23.1 ± 0.2	-23.3 ± 0.4	$\mathbf{-7.2 \pm 0.2}$
Swimmer-v4	2	229.2 ± 46.8	251.2 ± 38.5	297.5 ± 12.0	$\mathbf{305.9 \pm 32.3}$	98.3 ± 7.1	42.1 ± 1.0
	5	236.0 ± 33.6	261.5 ± 38.6	283.3 ± 13.1	$\mathbf{306.3 \pm 31.5}$	87.5 ± 5.7	41.5 ± 0.7
	10	213.1 ± 39.2	255.0 ± 36.3	258.6 ± 54.6	$\mathbf{302.3 \pm 33.7}$	52.1 ± 4.4	38.6 ± 0.9
	20	285.1 ± 12.4	200.2 ± 25.0	$\mathbf{315.5 \pm 9.7}$	297.8 ± 34.2	36.0 ± 1.4	37.2 ± 0.2

Table 1: Final test returns on tasks in Gym. Results are presented as the mean ± standard error of the mean.

B.2 DMC proprio results

Task	Delay	Extended	Memoryless	Latent	Agnostic
Acrobot Swingup	2	223.0 ± 9.5	234.8 ± 26.0	224.4 ± 19.7	**256.6 ± 22.9**
	5	208.5 ± 42.1	228.6 ± 32.6	**257.9 ± 36.2**	240.9 ± 13.1
	10	120.7 ± 12.8	159.7 ± 15.2	193.2 ± 30.3	**197.0 ± 17.0**
	20	78.7 ± 15.0	93.7 ± 15.6	**144.0 ± 14.5**	129.7 ± 11.3
Cartpole Balance	2	986.7 ± 4.2	**993.2 ± 0.4**	990.4 ± 1.0	992.4 ± 2.4
	5	**990.6 ± 1.1**	986.5 ± 2.0	985.0 ± 3.9	990.0 ± 3.4
	10	966.4 ± 4.9	948.0 ± 12.0	916.7 ± 48.7	**980.4 ± 2.1**
	20	**799.8 ± 32.5**	664.8 ± 15.0	699.9 ± 40.0	511.9 ± 32.8
Cheetah Run	2	632.9 ± 25.3	666.3 ± 22.7	**668.9 ± 30.0**	596.1 ± 72.8
	5	600.2 ± 24.8	**648.3 ± 17.4**	534.9 ± 54.7	373.2 ± 39.5
	10	446.7 ± 20.1	**496.1 ± 6.6**	304.0 ± 60.3	173.6 ± 20.6
	20	**418.0 ± 16.9**	337.3 ± 14.2	204.4 ± 11.7	119.7 ± 3.4
Finger Spin	2	504.0 ± 21.1	426.7 ± 18.4	253.4 ± 83.0	**561.1 ± 93.5**
	5	303.6 ± 14.8	279.8 ± 17.8	**310.0 ± 57.0**	278.3 ± 31.1
	10	**307.0 ± 23.3**	171.5 ± 12.4	153.7 ± 22.1	111.1 ± 7.5
	20	**183.9 ± 13.6**	116.3 ± 4.6	118.4 ± 11.0	59.5 ± 8.1
Hopper Hop	2	**206.6 ± 31.7**	143.5 ± 15.9	164.4 ± 22.3	121.0 ± 11.9
	5	99.3 ± 23.4	**130.8 ± 37.4**	54.1 ± 11.4	53.6 ± 6.2
	10	**112.9 ± 16.4**	64.6 ± 15.4	33.9 ± 8.5	20.0 ± 4.7
	20	**31.8 ± 19.8**	0.0 ± 0.0	7.7 ± 1.8	11.1 ± 3.6
Walker Walk	2	877.9 ± 17.2	**916.1 ± 6.7**	873.5 ± 29.8	732.5 ± 39.1
	5	**789.5 ± 50.2**	622.9 ± 47.2	530.9 ± 22.2	389.5 ± 18.0
	10	**518.9 ± 16.9**	461.8 ± 44.9	278.0 ± 13.8	230.6 ± 14.3
	20	**381.6 ± 19.3**	339.5 ± 43.1	173.2 ± 7.7	170.3 ± 0.2

Table 2: Final test returns on tasks in DMC with proprioceptive inputs. Results are presented as the mean ± standard error of the mean.

B.3 DMC vision results

Task	Delay	Extended	Memoryless	Latent	Agnostic
Acrobot Swingup	2	300.8 ± 32.6	355.4 ± 38.8	374.9 ± 35.0	**396.3 ± 67.3**
	5	301.1 ± 16.8	284.6 ± 26.5	336.3 ± 26.8	**382.6 ± 28.5**
	10	278.7 ± 11.7	191.0 ± 18.2	**328.0 ± 28.0**	326.0 ± 49.3
	20	139.4 ± 7.9	108.1 ± 21.3	212.9 ± 34.7	**230.4 ± 16.5**
Cartpole Balance	2	994.7 ± 1.3	996.0 ± 0.2	**996.3 ± 0.1**	**996.3 ± 0.1**
	5	994.6 ± 0.1	992.6 ± 0.3	994.8 ± 0.5	**995.2 ± 0.1**
	10	975.7 ± 5.6	953.8 ± 12.5	**979.9 ± 5.3**	926.0 ± 60.7
	20	928.0 ± 11.6	681.3 ± 4.2	**935.3 ± 8.4**	570.8 ± 27.0
Cheetah Run	2	871.7 ± 11.3	839.8 ± 28.9	**873.0 ± 4.1**	799.1 ± 32.0
	5	**816.2 ± 20.9**	812.1 ± 13.0	783.9 ± 44.3	533.1 ± 37.3
	10	**640.0 ± 32.9**	610.9 ± 16.4	542.2 ± 24.8	251.2 ± 49.5
	20	**493.3 ± 21.8**	395.5 ± 21.2	273.1 ± 24.6	118.8 ± 11.4
Finger Spin	2	521.6 ± 123.3	391.4 ± 25.2	**576.7 ± 117.7**	418.0 ± 211.1
	5	**374.9 ± 11.3**	321.7 ± 19.3	316.7 ± 34.6	254.4 ± 128.8
	10	**291.7 ± 30.9**	186.9 ± 7.7	53.0 ± 35.3	99.7 ± 50.1
	20	**172.2 ± 11.9**	108.3 ± 3.4	33.8 ± 22.8	49.3 ± 24.2
Hopper Hop	2	304.6 ± 27.2	313.2 ± 27.4	**325.2 ± 41.5**	64.0 ± 64.0
	5	232.5 ± 33.6	**246.7 ± 32.3**	114.8 ± 25.9	18.0 ± 17.8
	10	**136.7 ± 6.3**	131.3 ± 13.4	33.3 ± 4.9	3.5 ± 3.5
	20	**103.0 ± 28.3**	45.3 ± 18.1	4.4 ± 1.3	2.3 ± 2.3
Walker Walk	2	932.0 ± 9.6	895.6 ± 27.7	**944.7 ± 8.1**	916.0 ± 12.8
	5	**821.8 ± 60.8**	819.2 ± 33.8	718.7 ± 30.1	524.4 ± 35.6
	10	**657.1 ± 56.2**	499.7 ± 23.4	344.4 ± 18.0	219.4 ± 5.5
	20	**474.4 ± 39.2**	399.0 ± 38.3	166.3 ± 3.1	147.3 ± 11.2

Table 3: Final test returns on tasks in DMC with visual inputs. Results are presented as the mean ± standard error of the mean.

B.4 Gym training curves

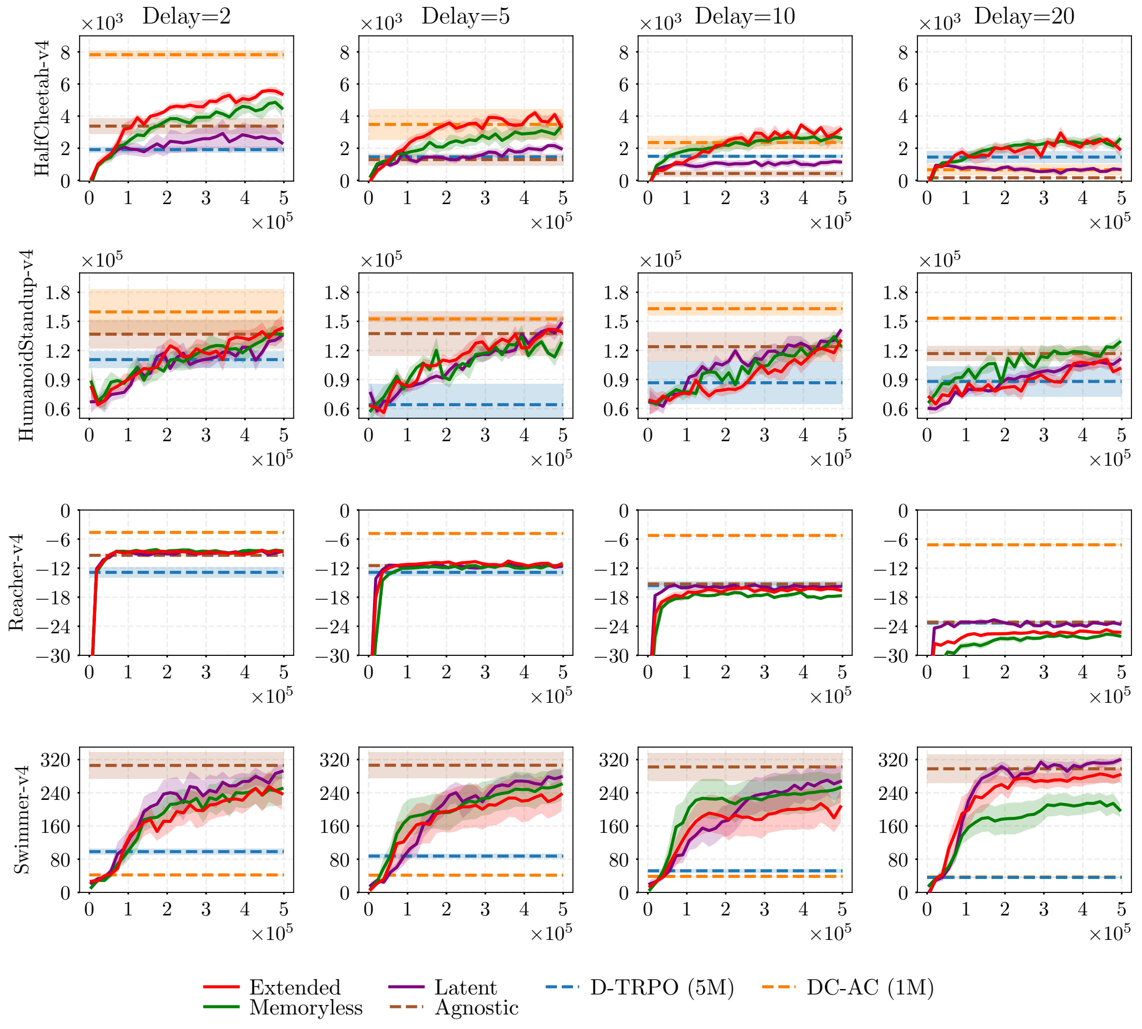


Figure 5: Training curves for the set of tasks in Gym. Dreamer variants trained with 500K interactions of the environment, while D-TRPO and DC-AC used 5M and 1M interactions, respectively. For D-TRPO and DC-AC, we have plotted the final training performance.

B.5 DMC proprio training curves

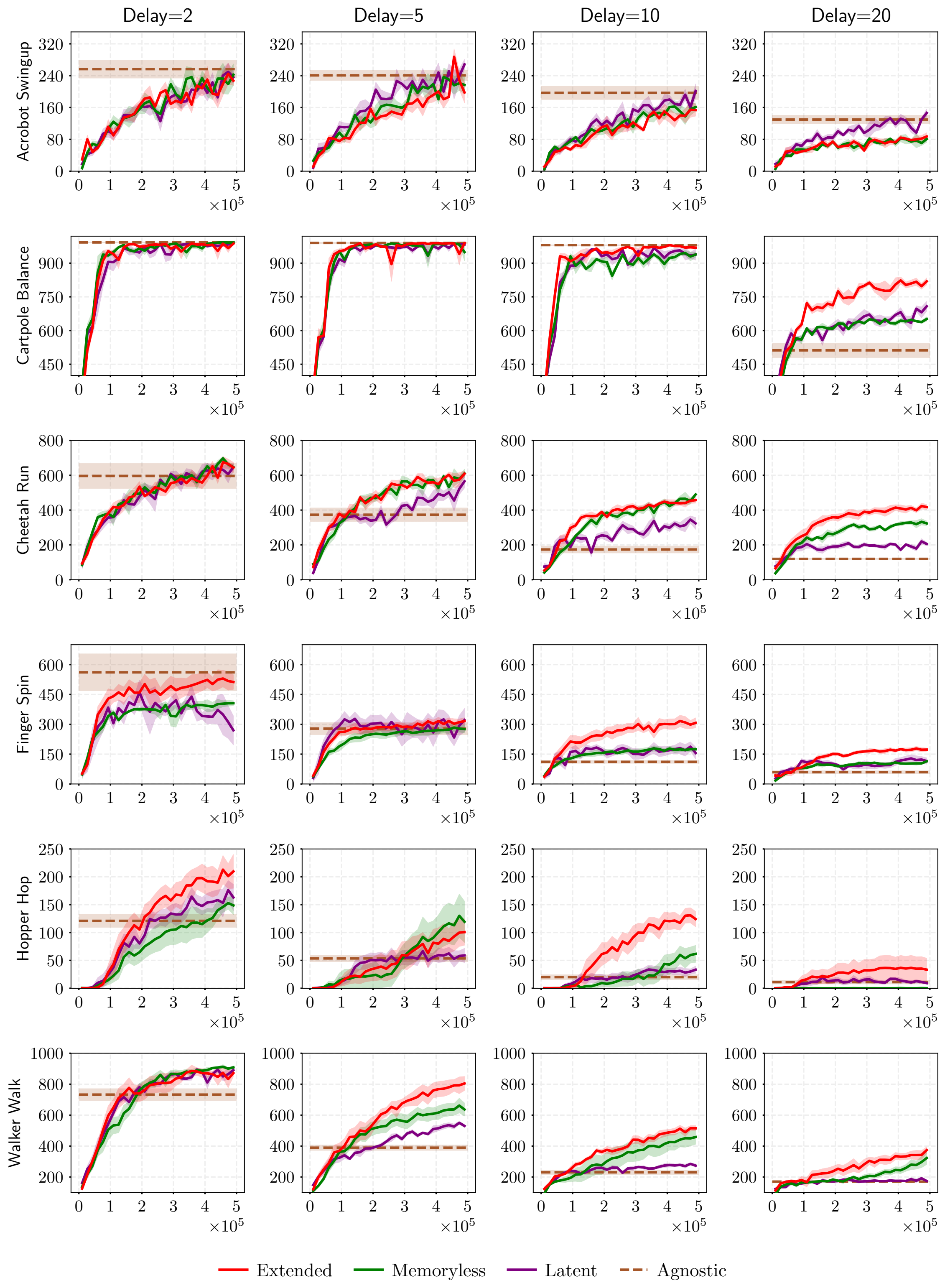


Figure 6: Training curves for the set of tasks in DMC with proprioceptive inputs with 500K interactions.

B.6 DMC vision training curves

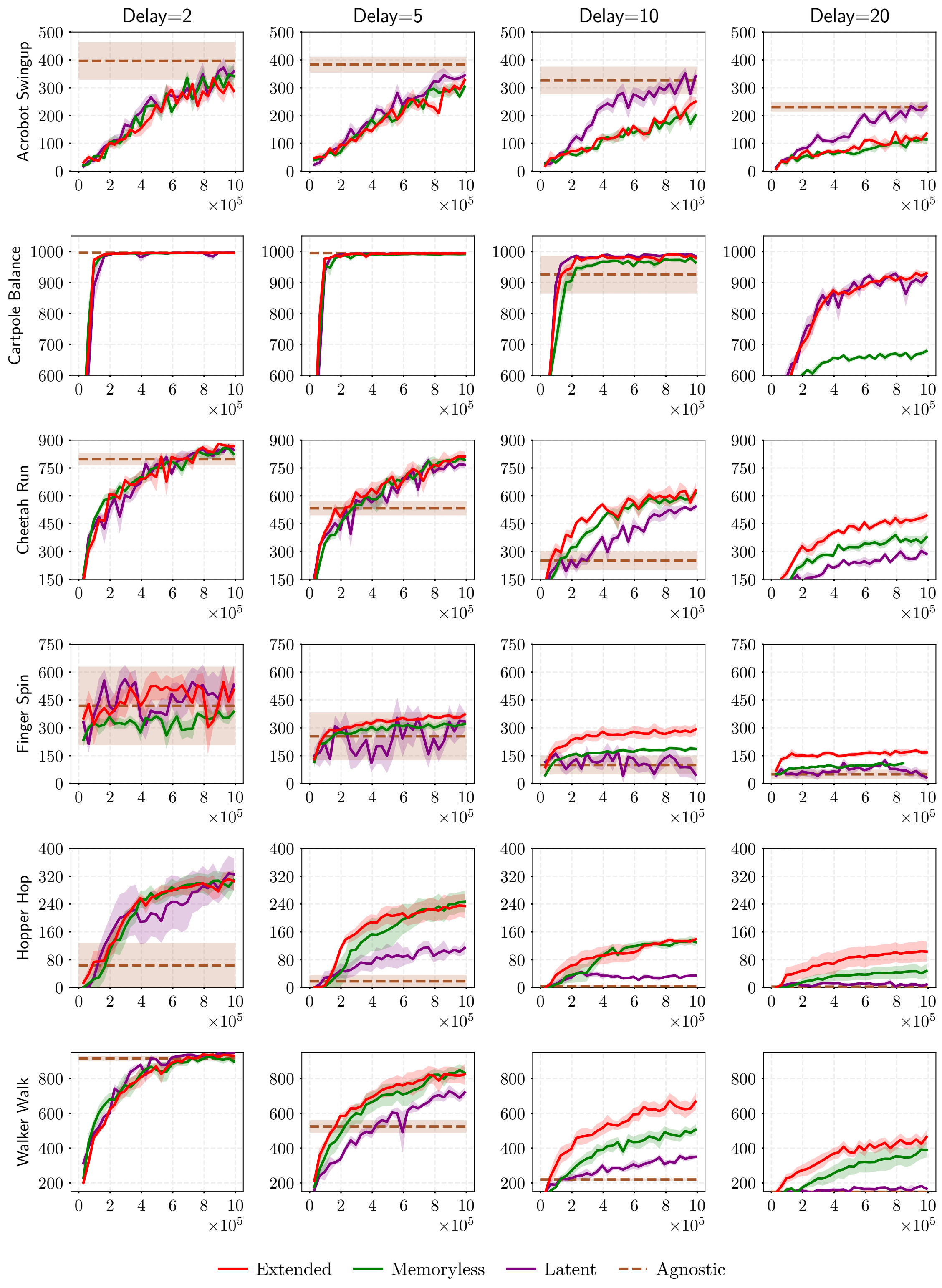


Figure 7: Training curves for the set of tasks in DMC with visual inputs with 1M interactions.