Direct Advantage Estimation for Scalable and Sample-efficient Deep Reinforcement Learning

Hsiao-Ru Pan, Bernhard Schölkopf

Keywords: advantage estimation, sample efficient deep RL, POMDP, multi-step learning

Summary

Direct Advantage Estimation (DAE) has been shown to improve the sample efficiency of deep reinforcement learning algorithms. However, its reliance on full environment observability limits its applicability in realistic settings, and its requirement to model transition probabilities incurs substantial computational overhead for high-dimensional observations. In the present work, we address both limitations. First, we extend the theoretical framework of DAE to partially observable domains with minimal modifications. Second, we reduce its computational complexity by introducing discrete latent dynamics models that efficiently approximate transition probabilities. We evaluate our approach on the Arcade Learning Environment and find that DAE scales effectively with function approximator capacity while retaining high sample efficiency.

Contribution(s)

1. We extend the theory of Direct Advantage Estimation to partially observable Markov decision processes, showing that it can be naturally applied to off-policy multi-step learning under partial observability.
   Context: Direct Advantage Estimation (DAE) (Pan et al., 2022; Pan & Schölkopf, 2024) was originally proposed for multi-step learning in fully observable Markov decision processes, which limits its applicability to simpler domains and excludes partially observable settings commonly encountered in practice.
2. We propose a discrete latent dynamics model that approximates transition probabilities in a compact latent space, significantly reducing the computational overhead of Off-policy DAE.
   Context: Off-policy corrections in DAE require approximations of transition probabilities, which was achieved by learning generative models that predict observations. This approach becomes expensive in environments with high-dimensional observations and has therefore only been evaluated in simpler domains (Pan & Schölkopf, 2024).

Direct Advantage Estimation for Scalable and Sample-efficient Deep Reinforcement Learning

Hsiao-Ru Pan[1], **Bernhard Schölkopf**[1,2,3]
{hpan,bs}@tuebingen.mpg.de

[1]**Max Planck Institute for Intelligent Systems, Tübingen**
[2]**ELLIS Institute Tübingen**
[3]**ETH Zürich**

Abstract

Direct Advantage Estimation (DAE) has been shown to improve the sample efficiency of deep reinforcement learning algorithms. However, its reliance on full environment observability limits its applicability in realistic settings, and its requirement to model transition probabilities incurs substantial computational overhead for high-dimensional observations. In the present work, we address both limitations. First, we extend the theoretical framework of DAE to partially observable domains with minimal modifications. Second, we reduce its computational complexity by introducing discrete latent dynamics models that efficiently approximate transition probabilities. We evaluate our approach on the Arcade Learning Environment and find that DAE scales effectively with function approximator capacity while retaining high sample efficiency.

1 Introduction

While reinforcement learning (RL) (Sutton & Barto, 2018) has achieved unprecedented results in various domains (Mnih et al., 2015; Berner et al., 2019; Schrittwieser et al., 2020; Wurman et al., 2022), training such agents remains challenging and often requires millions or even billions of samples (Henderson et al., 2018). A central component of deep RL is the approximation of state(-action) value functions, which are typically highly non-stationary and therefore difficult to learn. Pan et al. (2022) observed that the advantage function is comparatively more stable under policy variations and proposed Direct Advantage Estimation (DAE), which learns the advantage function directly rather than via value-function decomposition. DAE demonstrated strong empirical performance, but is restricted to on-policy settings. Subsequently, Pan & Schölkopf (2024) extended DAE to off-policy settings, achieving improved sample efficiency. However, the method suffers from significantly increased computational complexity due to the need to learn high dimensional generative models to approximate the transition probabilities. In addition, the method only applies to fully observable MDPs (Puterman, 2014), which can be limiting in more realistic settings.

The present work addresses the two limitations of Off-policy DAE: (1) applicability in POMDPs (Kaelbling et al., 1998), and (2) high computational overhead. More specifically, the contributions are:

- We extend the theory of DAE to POMDPs, providing a generalized return decomposition.
- We reduce the computational cost of Off-policy DAE by modeling stochastic transitions in a low dimensional embedding space.
- We evaluate our approach using the Arcade Learning Environment (Bellemare et al., 2013), and show that it (1) scales with the capacity of the function approximator, and (2) achieves perfor-

mance comparable to Rainbow DQN (Hessel et al., 2018) while only using 10% of the data. Additionally, we perform extensive ablation studies to quantify the contribution of each component.

2 Background

We consider a discounted POMDP defined by the tuple ($\mathcal{S}$, $\mathcal{A}$, T, Ω, $\mathcal{O}$, r, γ) (Kaelbling et al., 1998), where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s, a, s')$ denotes the transition probability from state s into state s' after taking action a, Ω is the observation space, $\mathcal{O}(s, o)$ denotes the probability of observing $o \in \Omega$ in state s, $r(s, a)$ denotes the reward received by the agent after taking action a in state s, and $\gamma \in [0, 1)$ denotes the discount factor. For simplicity, we denote $T(s, a, s')$ by $p(s'|s, a)$, $\mathcal{O}(s, o)$ by $p(o|s)$, and the probability of observing a trajectory under π by p_π. We consider the case where $\mathcal{S}$, $\mathcal{A}$, and Ω are finite. An agent in a POMDP cannot directly observe the states, but only the observations emitted from the state through $\mathcal{O}$. We focus on the infinite-horizon discounted setting, where the goal of an agent is to find a policy π that maximizes the expected return $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ (subscript indicates the actions follow π).

In MDPs, one can estimate the state(-action) value function $V^\pi(s)$ or $Q^\pi(s, a)$ as the states are observed directly. In POMDPs, however, agents do not observe states directly, and have to estimate the values based on the observed history (information vector) $h_t = (o_0, a_0, r_0, o_1, ..., o_t)$ (Bertsekas, 2012). As such, their counterparts in POMDPs are defined by:

$$V^\pi(h_t) = \mathbb{E}_\pi \left[\sum_{t'=0}^{\infty} \gamma^{t'} r_{t+t'} \middle| h_t \right], \quad Q^\pi(h_t, a_t) = \mathbb{E}_\pi \left[\sum_{t'=0}^{\infty} \gamma^{t'} r_{t+t'} \middle| h_t, a_t \right]. \tag{1}$$

2.1 Direct Advantage Estimation

Aside from Q and V, another function of interest is the advantage function defined by $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ (Baird, 1995). Pan et al. (2022) proposed Direct Advantage Estimation (DAE) to estimate the advantage function by minimizing the constrained objective:

$$\mathcal{L}(\hat{A}, \hat{V}) = \mathbb{E}_\pi \left[\left(\sum_{t=0}^{n-1} \gamma^t (r_t - \hat{A}_t) + \gamma^n \hat{V}_{\text{target}}(s_n) - \hat{V}(s_0) \right)^2 \right] \quad \text{s.t. } \mathbb{E}_\pi[\hat{A}(s, a)|s] = 0, \tag{2}$$

where $\hat{V}_{\text{target}}$ is a given bootstrapping target, $r_t = r(s_t, a_t)$, and $\hat{A}_t = \hat{A}(s_t, a_t)$. The constraint enforces the centering property of the advantage function (i.e., $\mathbb{E}_\pi[A^\pi(s, a)|s] = 0$). The minimizer of $\mathcal{L}(\hat{A}, \hat{V})$ can be viewed as a multi-step estimate of (A^π, V^π), as the objective includes multiple steps of unbiased rewards. One limitation of DAE is that it is on-policy, that is, the behavior policy ($\mathbb{E}_\pi$ in the objective) has to be the same as the target policy ($\mathbb{E}_\pi$ in the constraint). Pan & Schölkopf (2024) extended DAE to off-policy settings, by showing that if we view stochastic transitions as actions from an imaginary agent (nature), then the return of a trajectory can be decomposed into:

$$\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) = \sum_{t=0}^{\infty} \gamma^t \left(A^\pi(s_t, a_t) + B^\pi(s_t, a_t, s_{t+1}) \right) + V^\pi(s_0), \tag{3}$$

where $B^\pi(s_t, a_t, s_{t+1}) = \gamma V^\pi(s_{t+1}) - \gamma \mathbb{E}[V^\pi(s')|s_t, a_t]$ is the advantage function of nature, which quantifies how much of the return is caused by the randomness of the environment. This decomposition generalizes DAE into off-policy settings by incorporating $\hat{B}$ into the objective (Equation 2):

$$\begin{aligned} \mathcal{L}(\hat{A}, \hat{B}, \hat{V}) = \mathbb{E}_\mu \left[\left(\sum_{t=0}^{n-1} \gamma^t (r_t - \hat{A}_t - \hat{B}_t) + \gamma^n \hat{V}(s_n) - \hat{V}(s_0) \right)^2 \right] \\ \text{subject to } \begin{cases} \mathbb{E}_\pi[\hat{A}(s, a)|s] = 0 \\ \mathbb{E}_{s' \sim p(\cdot|s,a)}[\hat{B}(s, a, s')] = 0 \end{cases}. \end{aligned} \tag{4}$$

Contrary to Equation 2, the behavior policy ($\mathbb{E}_\mu$) and the target policy (π in the constraint) need not be equal. Intuitively, $\hat{A}$ and $\hat{B}$ can be viewed as corrections for stochasticity originating from the policy and the transitions, respectively. Under mild assumptions on the coverage of μ, one can show that (A^π, B^π, V^π) is the unique minimizer of this objective, suggesting that we can perform off-policy policy evaluation by minimizing this objective. One benefit of this approach is that it does not require importance sampling to correct for off-policy data, which can lead to unbounded variance. However, this approach has some limitations: (1) it only applies to MDPs, and (2) enforcing the $\hat{B}$ constraint requires estimating $p(s'|s,a)$, which can be expensive for large state spaces.

3 Return Decomposition in POMDPs

The key observation of Pan & Schölkopf (2024) is that the return can be decomposed using advantage functions (Equation 3). Here, we show that such a decomposition also exists in POMDPs.

Firstly, define the advantage function in POMDPs by $A^\pi(h_t, a_t) = Q^\pi(h_t, a_t) - V^\pi(h_t)$. Similar to its counterpart in MDPs, this function also satisfies the centering property, namely $\sum_{a\in\mathcal{A}} \pi(a|h_t) A^\pi(h_t, a) = 0$. The next question is how we can similarly define B^π such that the return can be decomposed, and whether this function also satisfies the centering condition. We proceed by examining the difference between the return and the sum of the advantages along a trajectory, namely:

$$\sum_{t=0}^{\infty} \gamma^t r_t - \left(\sum_{t=0}^{\infty} \gamma^t A^\pi(h_t, a_t) + V^\pi(h_0) \right) = \sum_{t=0}^{\infty} \gamma^t \left(r_t + \gamma V^\pi(h_{t+1}) - Q^\pi(h_t, a_t) \right). \quad (5)$$

This equation suggests the definition $B^\pi(h_t, a_t, h_{t+1}) \coloneqq r_t + \gamma V^\pi(h_{t+1}) - Q^\pi(h_t, a_t)$. Recall that h_{t+1} is the concatenation of h_t and (a_t, r_t, o_{t+1}), meaning that we can rewrite $B^\pi(h_t, a_t, h_{t+1})$ as $B^\pi(h_t, a_t, r_t, o_{t+1})$. This recovers the decomposition (Equation 3); furthermore, this B^π also satisfies a slightly different centering property, namely, $\mathbb{E}_{(r_t, o_{t+1})\sim p(\cdot|h_t, a_t)} [B^\pi(h_t, a_t, r_t, o_{t+1})] = 0$. This equation differs from its MDP counterpart by the variables that are being marginalized. In POMDPs, since the agent cannot observe the underlying states, we have to marginalize over the observed variables after taking an action (i.e., the immediate reward and the next observation). This brings us to the following generalization:

Proposition 1. *Given behavior policy μ, target policy π, and backup length $n > 0$. (A^π, B^π, V^π) is a minimizer of*

$$\mathcal{L}(\hat{A}, \hat{B}, \hat{V}) = \mathbb{E}_\mu \left[\left(\sum_{t'=0}^{n-1} \gamma^{t'} \left(r_{t+t'} - \hat{A}_{t+t'} - \hat{B}_{t+t'} \right) + \gamma^n \hat{V}(h_{n+t}) - \hat{V}(h_t) \right)^2 \right]$$
$$\text{subject to} \begin{cases} \mathbb{E}_{a\sim\pi(\cdot|h)}[\hat{A}(h,a)] = 0 \quad \forall h \in \mathcal{H} \\ \mathbb{E}_{(r,o')\sim p(\cdot|h,a)}[\hat{B}(h,a,r,o')] = 0 \quad \forall (h,a) \in \mathcal{H}\times\mathcal{A} \end{cases}, \quad (6)$$

where $\mathcal{H}$ is the set of all trajectories of the form $(o_0, a_0, r_0, ...o_t)$, $\hat{A}_t = \hat{A}(h_t, a_t)$, and $\hat{B}_t = \hat{B}(h_t, a_t, r_t, o_{t+1})$. Furthermore, if $p_\pi(h) > 0 \implies p_\mu(h) > 0$ holds for all h (i.e., the behavior policy has a larger coverage), then the minimizer is unique at trajectories covered by π.

See Appendix 7 for a proof. The assumption that the behavior policy has a larger coverage is common in off-policy settings (Precup et al., 2000; Thomas & Brunskill, 2016), and is required in our case to guarantee the solution is unique. Note that, slightly different from the MDP version, we do not require additional coverage regarding actions because h already encodes past actions. At its core, Proposition 1 differs from its MDP counterpart (Equation 4) by simply replacing states with histories, and transition probabilities with conditional densities of the observed variables (in the $\hat{B}$ constraint). This is a consequence of the fact that POMDPs can be reformulated as MDPs using information vectors (Bertsekas, 2012). Like DAE, this can be seen as an off-policy multi-step method for value approximation, as the objective function includes n steps of

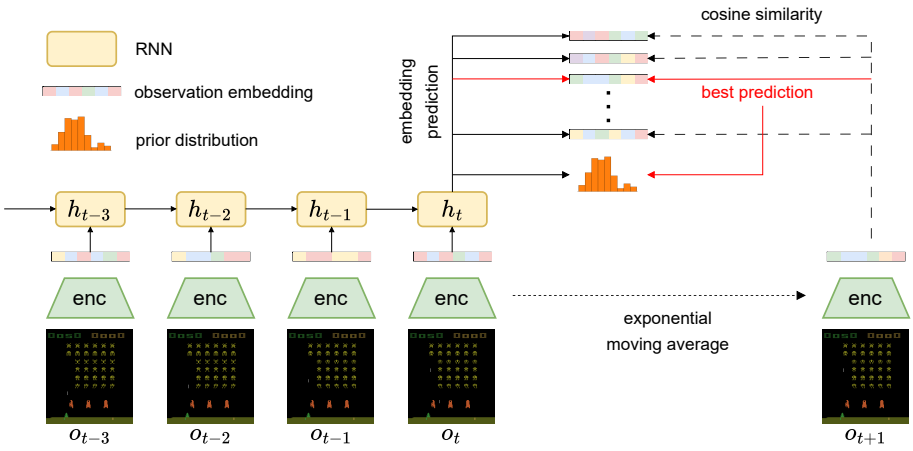

Figure 1: The latent dynamics model first embeds observations (o_t) into low dimensional embeddings (x_t), which are then processed by an RNN to capture the information vectors (h_t) (we omit conditioning of actions and rewards for illustrative purpose). At each time-step, the model makes $|\mathcal{Z}|$ predictions ($\hat{x}$) of the next embedding along with the prior distribution $p_\phi(\cdot|h,a)$ to capture the stochasticity. During training, the embedding predictions are compared to the embedding of the next embedding, and gradients only propagate through the best prediction with the corresponding index used to train the categorical prior distribution.

rewards. Deploying this method in practice, however, is non-trivial due to the constraints. The $\hat{A}$ constraint can be enforced upon a given approximator $f(h,a)$ for a given policy π by constructing $\hat{A}(h,a) = f(h,a) - \sum_{a\in\mathcal{A}} f(h,a)\pi(a|h)$ (Wang et al., 2016). Enforcing the $\hat{B}$ constraint is more challenging due to its dependency on $p(r,o'|h,a)$, which is typically unknown to the agent. We discuss how to efficiently approximate this constraint using latent dynamics models below.

3.1 Discrete Latent Dynamics Model for Constraint Approximation

In the original Off-policy DAE implementation (Pan & Schölkopf, 2024), the $\hat{B}$ constraint was approximated by first encoding transitions (h,a,r,o')[1] into a small discrete latent space $z \in \mathcal{Z}$ using a conditional variational autoencoder (CVAE) (Kingma & Welling, 2013; Sohn et al., 2015), and constructing $\hat{B}(h,a,r,o')$ from a given function approximator $g(h,a,z)$ by:

$$\hat{B}(h,a,r,o') = \mathbb{E}_{z\sim q_\phi(\cdot|h,a,r,o')}[g(h,a,z)] - \mathbb{E}_{z\sim p_\phi(\cdot|h,a)}[g(h,a,z)], \tag{7}$$

where $q_\phi(\cdot|h,a,r,o')$ is the approximated posterior (encoder), $p_\phi(\cdot|h,a)$ is the prior, and ϕ is the parameters of the CVAE. By using discrete latent variables, the expectations with respect to z can be computed efficiently. It then follows that $\mathbb{E}_{(r,o')\sim p(\cdot|h,a)}[\hat{B}(h,a,r,o')] \approx 0$. Learning the CVAE, however, can be computationally expensive if observations are high dimensional due to the need to reconstruct observations.

To reduce computational overhead, we propose to learn a discrete dynamics model purely in the embedding space[2] (see Figure 1). This is achieved by first embedding observations into a low dimensional vector $x = \texttt{enc}(o) \in \mathbb{R}^d$ (with $d \ll \dim(\Omega)$), where enc denotes the encoder (e.g., a convolutional network), and learning to predict $x_{t+1} = \texttt{enc}(o_{t+1})$ from the observed history (h_t, a_t). This approach is similar to the self-predictive representation (SPR) (Schwarzer et al., 2020); however, SPR only produces a single prediction, which cannot capture stochastic transitions. We address this

[1] We adopt the POMDP setting here for consistency, but note that this was originally developed for MDPs.

[2] We will refer to the space of encoded observations as the embedding space, and $\mathcal{Z}$ as the latent space of the CVAE to avoid confusion.

by combining SPR with the Winner-Takes-All (WTA) loss (Lee et al., 2015; Guzman-Rivera et al., 2012), which was shown to be useful for modeling stochastic predictions. More specifically, we combine them by: (1) making $|\mathcal{Z}|$ predictions of the next embedding (note that $|\mathcal{Z}|$ is finite), and (2) minimizing only the best prediction. This results in the following objective:

$$\mathcal{L}_{\text{rec}} = \sum_{z \in \mathcal{Z}} \mathbb{I}\big[z = \arg\min_i ||\hat{x}_{t+1,i}(h_t, a_t) - x_{t+1}||\big] ||\hat{x}_{t+1,z}(h_t, a_t) - \texttt{sg}(x_{t+1})||^2, \tag{8}$$

where $\mathbb{I}$ is the indicator function, and $\texttt{sg}$ denotes stop-gradient. Intuitively, this can be seen as performing k-means clustering (with $k = |\mathcal{Z}|$) in the embedding space with centroids $\hat{x}_{\cdot,z}$ (Rupprecht et al., 2017). The WTA loss is known to be difficult to train as the gradient only propagates through the best prediction, which can sometimes lead to collapse of predictions. As such, in practice, we use an annealing procedure similar to the evolving WTA (Makansi et al., 2019), where the indicator function is replaced by a soft weighting (see Appendix 8 for details). Next, note that the objective is equivalent to a conditional vector-quantized VAE (VQ-VAE) (Van Den Oord et al., 2017), with posterior $q_\phi(z|h_t, a_t, x_{t+1}) = \mathbb{I}[z = \arg\min_i ||\hat{x}_{t+1,i}(h_t, a_t) - x_{t+1}||]$, and codebook $\hat{x}_{t+1,z}(h_t, a_t)$ that are dependent on the information vector h_t. Consequently, we can learn the prior by minimizing the KL-divergence between the prior $p_\phi(z|h_t, a_t)$ and the posterior $q_\phi(z|h_t, a_t, x_{t+1})$. With this CVAE, we can then approximate the $\hat{B}$ constraint using Equation 7.[3]

In practice, we find that using shallow multilayer perceptrons (MLPs) to model the dynamics already achieves strong empirical performance with negligible computational overhead compared to other parts of the system. In addition, we find it possible to learn the RL objective (Equation 6) and the dynamics model jointly end-to-end to further reduce computational overhead compared to learning them separately as done by Pan & Schölkopf (2024).

It should be noted that the proposed discrete latent dynamics modeling approach is not specific to the DAE objective and is amenable to other model-based planning methods (e.g., tree search (Antonoglou et al., 2021)). However, in the present work, we use it solely for off-policy corrections, leaving the exploration of other potential applications to future work.

4 Experiments

We examine the performance of the POMDP version of DAE using 47 environments[4] from the Arcade Learning Environment (ALE) (Bellemare et al., 2013), which includes environments with diverse dynamics and various degrees of partial observability.

We use the same environment setting as the Dopamine baselines (Castro et al., 2018), which largely follows the modern evaluation protocols proposed by Machado et al. (2018), including the use of sticky actions (repeat previous action with a certain probability) and discarding end-of-life signals. Note that while sticky actions were originally proposed to inject stochasticity into the environments, they also introduce additional partial observability due to their dependencies on previous actions.

We evaluate our method using a DQN-like (Mnih et al., 2015) agent with some modifications, which we briefly summarize: (1) **Recurrent Architecture**: We use an LSTM (Hochreiter & Schmidhuber, 1997) after the convolutional encoder to process sequences of observations. Aside from observations, we also feed previous actions and rewards into the LSTM to model the full history. Similar to R2D2 (Kapturowski et al., 2018), we also store the recurrent states in the replay buffer and include a short burn-in sequence to initialize the LSTM states. (2) **DAE objective**: We replace the 1-step Q-learning objective with the multi-step DAE objective (Equation 6, we set n=16 by default), and use three separate MLPs on top of the LSTM to model $\hat{A}$, $\hat{B}$, and $\hat{V}$. (3) **Discrete Latent Dynamics Model**: We use three additional MLPs on top of the LSTM to estimate the next observation

[3]Note that the $\hat{B}$ constraint indicates that we should also consider stochasticity from the rewards. This can be achieved by adding another reward reconstruction term into Equation 8.

[4]We exclude hard exploration games such as Montezuma's Revenge, as they typically require specialized exploration strategies, and often have low predictive power on the overall performance (Aitchison et al., 2023).

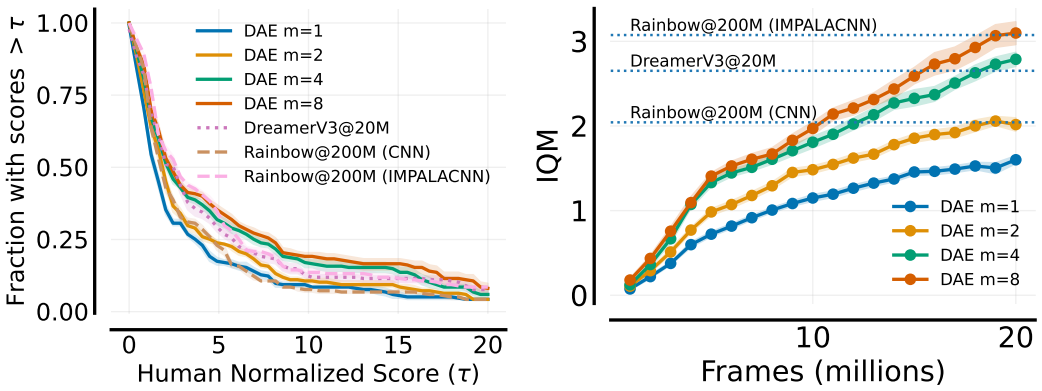

Figure 2: Performance profile (left) and sample efficiency (right) of DAE.

embedding $\hat{x}_{t+1,z}(h_t, a_t)$, the immediate reward $p(r_t|h_t, a_t)$, and the prior distribution $p(z|h_t, a_t)$ for approximating the $\hat{B}$ constraint. Similar to SPR, we use an exponential moving average of the online network as the target network to generate the next observation embeddings for the dynamics model training. This target network is also used to construct smoothly changing target policies and value bootstrapping targets for the DAE objective, which were found to be important for DAE (Pan & Schölkopf, 2024). (4) **Deeper Network**: We replace the shallow three-layer convolutional network used in the original DQN by the 15-layer deep residual network proposed by Espeholt et al. (2018) (denoted IMPALACNN below), which was found to enjoy better scalability and improved sample efficiency (Schwarzer et al., 2023). The CNN-LSTM backbone is shared for both the value heads and the dynamics heads to reduce computational overhead. For more details, we refer the reader to Appendix 8. In terms of RL, our agent can be viewed as a DQN variant with (a) **POMDP correction** and (b) **multi-step off-policy learning** (enabled by DAE). Below, we show how these changes affect the performance of the agent.

In the following experiments, we train our method for 20 million frames (5 million environment steps due to frame-skipping), and evaluate the agent every 1 million frames by averaging the cumulative scores of 50 episodes. We follow the protocol of Agarwal et al. (2021) and report the interquartile means (IQM) and performance profiles, along with 95% bootstrap confidence intervals aggregated over 5 random seeds and 47 environments.

Scalability and Sample Efficiency Obando-Ceron et al. (2024a) showed that naively scaling up the capacity of the function approximator does not always translate to an increase in performance. On the other hand, DAE was shown to be easily scalable in the on-policy setting (Pan et al., 2022). Here, we examine the scalability of DAE in the off-policy POMDP setting by increasing the width (multiplied by m) of the IMPALACNN backbone. We compare DAE to three baselines: (1) Dopamine Rainbow DQN (Castro et al., 2018; Hessel et al., 2018)[5] (2) A scaled-up version of Rainbow using IMPALACNN, and (3) DreamerV3 (Hafner et al., 2025). Both (1) and (2) represent classical frame-stacking model-free baselines, whereas (3) is a more recent recurrent model-based approach closer to our agent. For baselines (1) and (2), we use the scores reported by Castro et al. (2018), which were trained for 200 million frames. For (3), we train DreamerV3 using the preset 50M parameter network, which has a similar number of parameters to our m=8 variant, to establish a closer comparison (see Appendix 8.5 for more details). Figure 2 shows that the m=2 variant already performs similarly to Rainbow while using only 10% of the training frames, and by scaling up the network to m=8, we achieve performance comparable to Rainbow with IMPALACNN. Similarly, we find that DAE is competitive with DreamerV3 at m=4, and is even more sample efficient at m=8.

[5]Dopamine Rainbow DQN is a modern reimplementation of the Rainbow DQN, which includes 3 core improvements (n-step backup, prioritized replay, and distributional RL) from the original implementation.

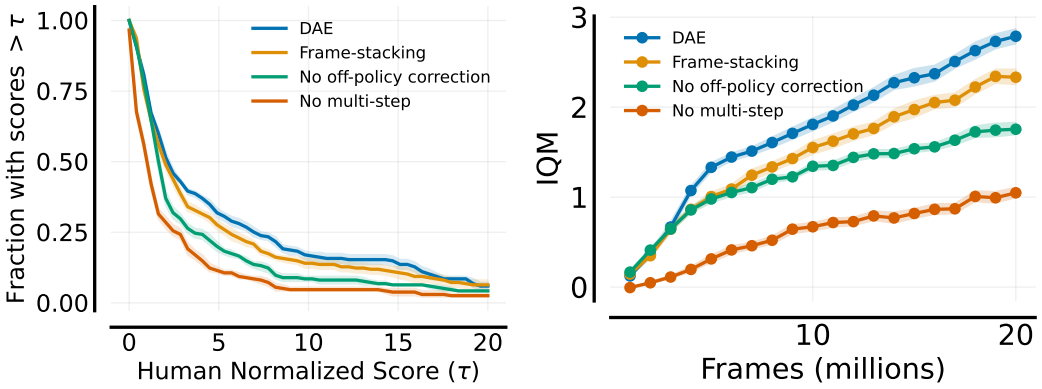

Figure 3: Performance profile (left) and sample efficiency (right) of the ablation study. For each ablation, we remove the corresponding component from the base DAE agent.

Next, we perform ablation studies to better understand the contribution of the modifications. To limit computational cost, we use only the $m = 4$ model. Figure 3 and Table 1 summarize the results.

Table 1: Effect of each component.

Ablation	IQM
DAE	2.79
Frame-Stacking	2.33 (-0.46)
No off-policy correction	1.75 (-1.04)
No multi-step	1.05 (-1.74)

Multi-step Learning Multi-step learning speeds up learning, reduces bias from bootstrapping, and was found to stabilize training (Hernandez-Garcia & Sutton, 2019; Van Hasselt et al., 2018). However, it also increases the variance of value updates, and choosing the backup length n can be seen as a bias-variance tradeoff (Kearns & Singh, 2000). Here, we compare multi-step learning ($n = 16$) to single-step learning ($n = 1$). From the learning efficiency curve (Figure 3, right), we see that multi-step learning significantly improves sample efficiency, and is, in fact, the most important component in this ablation study, accounting for a decrease of 1.74 in the IQM score. This suggests that the bias from bootstrapping significantly exceeds the variance from the multi-step learning in this case.

Off-policy Correction In previous studies, multi-step learning was often used without proper off-policy corrections (Van Hasselt et al., 2018; Hessel et al., 2018; Kapturowski et al., 2018; Hernandez-Garcia & Sutton, 2019; Schwarzer et al., 2023; D'Oro et al., 2023). Here, we demonstrate the importance of off-policy correction. Similar to Pan & Schölkopf (2024), we partially disable off-policy corrections by setting $\hat{B} \equiv 0$ during training, which can be seen as ignoring stochasticity from the environment (note that $B^\pi \equiv 0$ for deterministic environments).[6] Consistent with prior work, we find that, even without off-policy corrections, multi-step learning still significantly outperforms single-step learning. However, off-policy correction further improves the performance of our agent. This also indicates that the learned latent dynamics model can well approximate the dynamics, since the $\hat{B}$ constraint hinges on this approximation.

Frame-Stacking Frame-stacking has been the standard approach to approximate the ALE environments as MDPs since its introduction by Mnih et al. (2015). However, previous works have demonstrated that frame-stacking is not enough to fully capture the partial observability of the ALE (Kapturowski et al., 2018). Here, we demonstrate the effectiveness of the POMDP correction by replacing the recurrent layer by a single-layer MLP with a similar number of parameters. Consistent with prior work (Kapturowski et al., 2018; Hausknecht & Stone, 2015), we find that frame-stacking is suboptimal and accounts for $\sim$16% of the performance degradation (-0.46 in IQM score), indicating the importance of the POMDP correction. Aside from performance, we also see qualitative differences in the entropy of the prior distribution of the dynamics model. Since the ALE is deterministic by

[6]The typical multi-step method is more aggressive and equivalent to enforcing both $\hat{A} \equiv 0$ and $\hat{B} \equiv 0$.

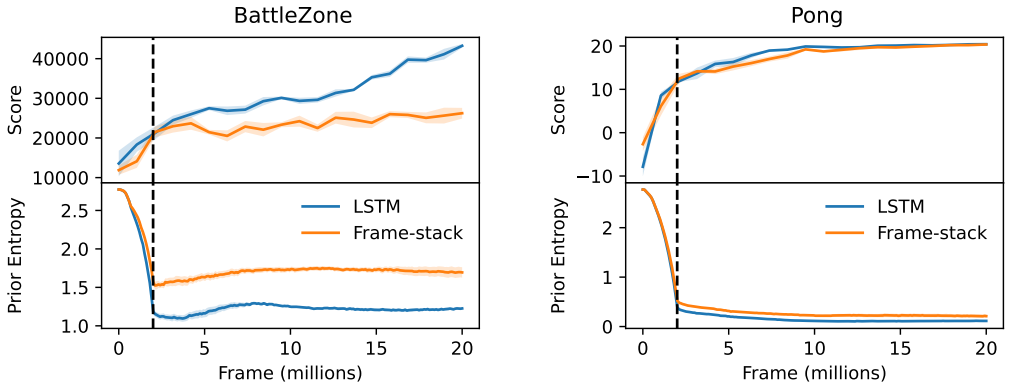

Figure 4: Entropy of the prior distribution $p(z|h, a)$ during training in BattleZone (left, more partially observable) and Pong (right, nearly fully observable). Lines and shadings represent average and 1 standard error, respectively. Dashed lines indicate the end of WTA annealing.

design, uncertainties of the next observation prediction comes primarily from partial observability of the environment (another source is sticky-actions, which also introduces stochasticity). Figure 4 compares the entropy of the prior distributions of the dynamics models and the learning curves between BattleZone and Pong. In BattleZone, an agent has to control a tank in a 3D environment with limited views of its surroundings in third-person. In contrast, Pong is almost fully observable except for the velocities of the objects (ball and paddles), which can be inferred from the past few frames. Here, we see that the LSTM agent converges to a much lower entropy compared to the frame-stacking agent in BattleZone, suggesting that the next observations have dependencies beyond the most recent 4 frames that cannot be utilized by the frame-stacking agent. On the other hand, both the LSTM and the frame-stacking agents performed similarly in Pong with very low entropy, indicating that the environment has a low degree of partial observability. In Appendix 8.6, we further investigate the link between changes in the entropy and changes in the performance, and provide additional evidence that partial observability degrades the performance of frame-stacking agents.

Discrete Latent Dynamics Model Our latent dynamics model leverages multiple predictions to model the stochasticity of the environments. To assess the robustness of this approach, we vary the number of predictions $|\mathcal{Z}|$ and report performance profile in Figure 5. We find that $|\mathcal{Z}| = 8$ and $|\mathcal{Z}| = 16$ yield nearly identical results, with noticeable degradation only when $|\mathcal{Z}|$ is reduced to 4. This indicates that, while the stochasticity cannot be ignored, in most environments it can be captured with relatively few modes, and our approach remains robust to the choice of $|\mathcal{Z}|$ provided it is not too small.

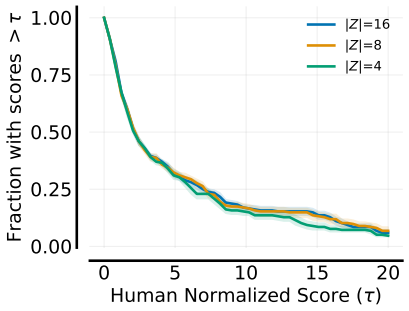

Figure 5: Effect of $|\mathcal{Z}|$.

To summarize the ablation studies, we observe from the performance profile (Figure 3, left) that the curves are almost dominated by the base agent, suggesting that the corrections are not environment-specific, but rather general algorithmic improvements. Additional per-environment results can be found in Appendix 8.6.

5 Related Work

Advantage Estimation Baird (1994) first introduced the advantage function and the algorithm *advantage updating* to solve continuous time RL problems. Later, Kakade & Langford (2002) showed

that the performance difference between two policies can be described using the advantage function, which became the foundation of various modern policy optimization algorithms. More recently, Schulman et al. (2015) proposed Generalized Advantage Estimation (GAE), which utilizes TD(λ) (Sutton, 1988) to perform on-policy multi-step estimates of the advantage function, and demonstrated its effectiveness in continuous control settings. Wang et al. (2016) proposed dueling network, an extension of DQN, which parametrized $Q_\theta = V_\theta + A_\theta$ and showed that this parametrization improves the performance of the original DQN. Tang et al. (2023) proposed VA learning, which uses a similar decomposition, but updates V and A separately, and showed its convergence properties in the tabular setting and effectiveness when applied to deep RL settings. Pan et al. (2022) proposed DAE for on-policy estimation of the advantage function, and was later shown to be equivalent to learning control variates for policy evaluation (Pan & Schölkopf, 2025). Pan & Schölkopf (2024) generalized DAE to off-policy settings, and the present work extends its domain to POMDPs and improves its computational efficiency.

Partial Observability POMDPs provide a general framework for studying decision making with incomplete states (Åström, 1965). In RL, POMDPs are usually solved by first converting them into MDPs using belief states or information vectors (Bertsekas, 2012; Kaelbling et al., 1998). In deep RL, common approaches include frame-stacking (Mnih et al., 2015), or modeling the histories directly (Kapturowski et al., 2018; Hausknecht & Stone, 2015; Hafner et al., 2025).

Latent Dynamics Model Learning dynamics models in the latent space is a promising approach to model-based RL (Ha & Schmidhuber, 2018; Han et al., 2019; Schrittwieser et al., 2020; Hafner et al., 2025; Antonoglou et al., 2021). Similar ideas have also been explored using bisimulation metrics (Ferns et al., 2004; Zhang et al., 2020), and were shown to be effective in learning representations for downstream tasks. However, learning dynamics models in the latent space is prone to collapse, and it is common to rely on either reconstructing the observations or self-supervision to learn meaningful representations (Anand et al., 2021; Deng et al., 2022). In the present work, we combined self-supervised learning methods (Schwarzer et al., 2020; Grill et al., 2020) and the WTA loss (Makansi et al., 2019; Rupprecht et al., 2017) to overcome the collapsing problem and reduce computational overhead of learning high dimensional models.

Scaling Deep RL Scaling has been central to progress in deep learning, where larger models were shown to yield better performance (Hestness et al., 2017; Henighan et al., 2020; Zhai et al., 2022). However, scaling in deep reinforcement learning (RL) is more challenging due to unstable training dynamics, and often requires additional regularization or architectural changes (Obando-Ceron et al., 2024b; Schwarzer et al., 2023; Nauman et al., 2024; Castanyer et al., 2025).

6 Discussion

In the present work, we extended DAE for POMDPs and addressed its high computational complexity by using discrete latent dynamics models. This opens up possibilities of using DAE to build sample-efficient RL agents for challenging real-world domains, where partial observability and high-dimensional observations are common. Through experiments in the ALE with a modified DQN agent, we demonstrated that DAE is sample-efficient and scalable, and verified the effectiveness of the proposed corrections.

We emphasize that, although our method learns transition models for DAE, they are not used for rollouts as in typical model-based algorithms. Instead, they serve only to perform off-policy corrections (enforcing the $\hat{B}$ constraint in the DAE objective). From this perspective, our approach is more model-free than model-based. Its ability to scale easily without additional tuning suggests that one of the challenges in scaling up value-based model-free methods like DQN may lie in the objective function itself, which might not be well suited to current deep learning architectures. A more detailed investigation of this hypothesis is left for future work.

Finally, we note some limitations: (1) DAE requires learning the transition probabilities to approximate constraints for the objective. While we demonstrated the effectiveness of learning latent

dynamics models to achieve this, doing so inevitably introduces additional hyperparameters (e.g., network architectures of the dynamics models) and complicates the implementation. Interestingly, this type of problem falls under a more general setting known as conditional moment restriction, and is an active research area (Newey, 1990; Bennett et al., 2019; Muandet et al., 2020; Kremer & Schölkopf, 2024). One direction for future work is to explore more robust and efficient alternatives to approximate the constraints. (2) We explored scaling of the convolutional layers of the image encoders, but it remains an open question how scaling other components, such as other parts of the function approximator or number of updates, might impact performance. A more systematic study of scaling across the full architecture could yield further insights.

Acknowledgments

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Hsiao-Ru Pan.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pp. 421–438. PMLR, 2023.

Ankesh Anand, Jacob Walker, Yazhe Li, Eszter Vértes, Julian Schrittwieser, Sherjil Ozair, Théophane Weber, and Jessica B Hamrick. Procedural generalization by planning with self-supervised world models. *arXiv preprint arXiv:2111.01587*, 2021.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2021.

Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Leemon C Baird. Reinforcement learning in continuous time: Advantage updating. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pp. 2448–2453. IEEE, 1994.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Andrew Bennett, Nathan Kallus, and Tobias Schnabel. Deep generalized method of moments for instrumental variable analysis. *Advances in neural information processing systems*, 32, 2019.

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

Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

Roger Creus Castanyer, Johan Obando-Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron Courville, and Pablo Samuel Castro. Stable gradients for stable learning at scale in deep reinforcement learning. *arXiv preprint arXiv:2506.15544*, 2025.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL http://arxiv.org/abs/1812.06110.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 4956–4975. PMLR, 2022.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pp. 162–169, 2004.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc Bellemare, and Remi Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHVZWZAZ.

Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. *Advances in neural information processing systems*, 25, 2012.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.

Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

J Fernando Hernandez-Garcia and Richard S Sutton. Understanding multi-step deep reinforcement learning: A systematic study of the dqn target. *arXiv preprint arXiv:1901.07510*, 2019.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

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

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Michael J Kearns and Satinder Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pp. 142–147, 2000.

Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. Residual lstm: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Heiner Kremer and Bernhard Schölkopf. Geometry-aware instrumental variable regression. *arXiv preprint arXiv:2405.11633*, 2024.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7144–7153, 2019.

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

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Krikamol Muandet, Arash Mehrjou, Si Kai Lee, and Anant Raj. Dual instrumental variable regression. *Advances in Neural Information Processing Systems*, 33:2710–2721, 2020.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

Whitney K Newey. Efficient instrumental variables estimation of nonlinear models. *Econometrica: Journal of the Econometric Society*, pp. 809–837, 1990.

Johan Obando-Ceron, Aaron Courville, and Pablo Samuel Castro. In deep reinforcement learning, a pruned network is a good network. *arXiv e-prints*, pp. arXiv–2402, 2024a.

Johan Obando-Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob Foerster, Gintare Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep rl. *arXiv preprint arXiv:2402.08609*, 2024b.

Hsiao-Ru Pan and Bernhard Schölkopf. Skill or luck? return decomposition via advantage functions. *arXiv preprint arXiv:2402.12874*, 2024.

Hsiao-Ru Pan and Bernhard Schölkopf. On the variance of temporal difference learning and its reduction using control variates. In *Eighteenth European Workshop on Reinforcement Learning*, 2025.

Hsiao-Ru Pan, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf. Direct advantage estimation. *Advances in Neural Information Processing Systems*, 35:11869–11880, 2022.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Korbinian Pöppel, Maximilian Beck, and Sepp Hochreiter. Flashrnn: I/o-aware optimization of traditional rnns on modern hardware. *arXiv preprint arXiv:2412.07752*, 2024.

Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning*, 2000. URL https://api.semanticscholar.org/CorpusID:1153355.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE international conference on computer vision*, pp. 3591–3600, 2017.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

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

Yunhao Tang, Rémi Munos, Mark Rowland, and Michal Valko. Va-learning as a more efficient alternative to q-learning. In *International Conference on Machine Learning*, pp. 33739–33757. PMLR, 2023.

Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. *ArXiv*, abs/1604.00923, 2016. URL https://api.semanticscholar.org/CorpusID:9311215.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Matthijs Van Keirsbilck, Alexander Keller, and Xiaodong Yang. Rethinking full connectivity in recurrent neural networks. *arXiv preprint arXiv:1905.12340*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22409–22421. Curran Associates, Inc., 2022.

Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Supplementary Materials

The following content was not necessarily subject to peer review.

7 Proof of Proposition 1

Theorem 1 (Off-policy DAE for MDPs (Pan & Schölkopf, 2024))**.** *Given behavior policy μ, target policy π, and backup length $n \geq 0$. (A^π, B^π, V^π) is a minimizer of*

$$\mathcal{L}(\hat{A}, \hat{B}, \hat{V}) = \mathbb{E}_\mu \left[\left(\sum_{t=0}^{n-1} \gamma^t \left(r_t - \hat{A}_t - \gamma \hat{B}_t \right) + \gamma^n \hat{V}(s_n) - \hat{V}(s_0) \right)^2 \right] \\ \text{subject to} \begin{cases} \sum_{a \in \mathcal{A}} \hat{A}(s,a)\pi(a|s) = 0 \quad \forall s \in \mathcal{S} \\ \sum_{s' \in \mathcal{S}} \hat{B}(s,a,s')p(s'|s,a) = 0 \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \end{cases}, \tag{9}$$

where $\hat{A}_t = \hat{A}(s_t, a_t)$, $\hat{B}_t = \hat{B}(s_t, a_t, s_{t+1})$. Furthermore, if μ has a larger coverage than π, then minimizer is unique for states(-actions) covered by π.

Proposition 1. *Given behavior policy μ, target policy π, and backup length $n > 0$. (A^π, B^π, V^π) is a minimizer of*

$$\mathcal{L}(\hat{A}, \hat{B}, \hat{V}) = \mathbb{E}_\mu \left[\left(\sum_{t'=0}^{n-1} \gamma^{t'} \left(r_{t+t'} - \hat{A}_{t+t'} - \hat{B}_{t+t'} \right) + \gamma^n \hat{V}(h_{n+t}) - \hat{V}(h_t) \right)^2 \right] \\ \text{subject to} \begin{cases} \mathbb{E}_{a \sim \pi(\cdot|h)}[\hat{A}(h,a)] = 0 \quad \forall h \in \mathcal{H} \\ \mathbb{E}_{(r,o') \sim p(\cdot|h,a)}[\hat{B}(h,a,r,o')] = 0 \quad \forall (h,a) \in \mathcal{H} \times \mathcal{A} \end{cases}, \tag{6}$$

where $\mathcal{H}$ is the set of all trajectories of the form $(o_0, a_0, r_0, ...o_t)$, $\hat{A}_t = \hat{A}(h_t, a_t)$, and $\hat{B}_t = \hat{B}(h_t, a_t, r_t, o_{t+1})$. Furthermore, if $p_\pi(h) > 0 \implies p_\mu(h) > 0$ holds for all h (i.e., the behavior policy has a larger coverage), then the minimizer is unique at trajectories covered by π.

Proof. First, consider the MDP induced by the POMDP (with $\mathcal{S} = \mathcal{H}$) (Bertsekas, 2012). Now, note that the coverage statement $p_\pi(h_{t+1}) = p_\pi(h_t)\pi(a_t|h_t)p(o_{t+1}, r_t|h_t, a_t) > 0 \implies p_\mu(h_{t+1}) = p_\mu(h_t)\mu(a_t|h_t)p(o_{t+1}, r_t|h_t, a_t) > 0$ implies that $\pi(a_t|h_t) > 0 \implies \mu(a_t|h_t) > 0$ (i.e., μ has a larger coverage than π). With this condition, the proposition then follows from applying Off-policy DAE (Theorem 1) to the induced MDP. □

Remark: The original proof of Off-policy DAE assumes that the reward function is deterministic, which can be violated when converting POMDPs into MDPs. As such, our definition of $B^\pi(s,a,r,s') = r + \gamma V^\pi(s') - \mathbb{E}_{(r',s'') \sim p(\cdot|s,a)}[r' + \gamma V^\pi(s'')]$ (in a fully observable MDP) differs slightly from the original one $B^\pi(s,a,s') = \gamma V^\pi(s') - \mathbb{E}_{s'' \sim p(\cdot|s,a)}[\gamma V^\pi(s'')]$.

8 Experiment Details & Additional Results

8.1 Pseudocode and additional implementation details

We provide the pseudocode in Algorithm 1. For illustrative purpose, the pseudocode assumes a single actor during sampling and batch size 1 during training. Below, we discuss some implementation details.

WTA Training To avoid the WTA predictions from collapsing, we use a soft loss for the reconstruction by including $\epsilon_{\text{WTA}} \geq 0$ into the posterior construction. In practice, ϵ_{WTA} is linearly annealed from 1 to 0 in the early stage of training. More specifically, the posterior becomes

$$p(z|h_t, a_t, x_{t+1}) = \begin{cases} 1 - \epsilon_{\text{WTA}} + \frac{\epsilon_{\text{WTA}}}{|\mathcal{Z}|} & \text{if } z = \arg\min_z \|\hat{x}_{t+1,z} - x_{t+1}\| \\ \frac{\epsilon_{\text{WTA}}}{|\mathcal{Z}|} & \text{otherwise} \end{cases},$$

This is similar to the approach proposed by Makansi et al. (2019), which was found to make training less dependent on initialization, except that top-k nearest neighbors were used to construct the posterior. We found the posterior can change rapidly at the beginning of training and lead to instability of $\hat{B}$. As such, we do not include the $\hat{B}$ correction (simply force $\hat{B} \equiv 0$) for the first few steps, and only include it after $\epsilon_{\text{WTA}} \leq 0.75$ (approximately 125000 environment steps). In addition, we multiply the DAE objective by $(1 - \epsilon_{\text{WTA}})$, to prioritize model learning during the early phase of training.

Incorporating stochastic rewards in the transition model was achieved by adding a reward prediction head. In the case of the ALE, we exploit the discreteness of the rewards ($\mathcal{R} = \{-1, 0, 1\}$ due to clipping) and construct the latent space by $\mathcal{Z} = \mathcal{Z}_O \times \mathcal{R}$. This then allows us to decompose the prior and the posterior by $p(z|h, a) = p(z_o|h, a)p(r|h, a)$ and $q(z|h, a, r, o') = q(z_o|h, a, r, o')q(\hat{r}|h, a, r, o')$, respectively. In this case, the posterior $q(\hat{r}|h, a, r, o') = \mathbb{I}(\hat{r} = r)$ is simply the indicator function.

Target Policy As pointed out by Pan et al. (2022), having a smoothly changing target policy is crucial to optimizing the DAE objective function. Consequently, we construct the target policy using the softmax of $\hat{A}_{\theta_{\text{EMA}}}$. However, as reward densities can vary drastically between environments and lead to different scales of the advantage function. We additionally learn a temperature parameter T by minimizing $\log T + \beta_{\text{KL}}\text{KL}(\pi_{\text{EMA}}||\pi)$, where both policies π and π_{EMA} are softmax policies constructed using the advantage functions (i.e. $\pi = \text{softmax}(\frac{\hat{A}}{T})$). This KL divergence ensures that the online policy π does not deviate too much from the target policy π_{EMA}, and alleviates the need to tune the temperature manually for each environment. We note that this policy is only used for the DAE objective ($\hat{A}$ constraint), and not for data collection.

Finally, to balance the scales between various objective functions, we multiply the DAE loss by $\beta_V = \frac{1}{\text{Var}(G)}$, where $\text{Var}(G)$ is the variance of the returns, estimated from all trajectories in the replay buffer.

8.2 Environment Setting

The environment settings follow the ones used by the Dopamine baseline (Castro et al., 2018) (see Table 2). We use EnvPool (Weng et al., 2022) for efficient implementation of parallelized environments.

Table 2: ALE preprocessing parameters. Blue: Best practice suggested by Machado et al. (2018).

Parameter	Value
Grey-scaling	True
Observation Resolution	84×84
Frame Stack	4
Action Repetitions	4
Reward Clipping	[-1, 1]
Terminal on life-loss	False
Sticky Action Prob.	0.25
γ (discount factor)	0.99

Algorithm 1 DAE (POMDP)

Require: n (backup length), k (burn-in length), τ (EMA coefficient), `wta_scheduler`, `optimizer`, β_{prior}, β_{rec}, β_{KL}

Initialize network f_θ
$\theta_{\text{EMA}} \leftarrow \theta$
$T \leftarrow 1, T_{\text{EMA}} \leftarrow 1$
$D = \{\}$ (replay buffer)
$o_0 \leftarrow \texttt{env.reset}()$
$h_0 \leftarrow (o_0)$
for $t = 0, 1, 2, \ldots$ **do**
  $\hat{A}_t, H_t \leftarrow f_\theta(h_t)$ ($\hat{A}$: advantage, H_t: RNN state)
  $a \leftarrow \epsilon\text{-}\texttt{greedy}(\hat{A}_t)$
  $(r, o') \leftarrow \texttt{env.step}(a)$
  $h_{t+1} \leftarrow (h_t, a, r, o')$
  $D \leftarrow D \cup \{(o, a, r, o', H_t)\}$
  if $t + 1$ mod `steps_per_update` = 0 **then**
    $\epsilon_{\text{WTA}} \leftarrow \texttt{wta\_scheduler}(t)$
    `traj` $\leftarrow$ sample $(o_i, a_i, r_i, ..., o_{i+n+k})$ and H_i from D
    $\hat{V}, \hat{A}, \hat{B}, \hat{x}, \hat{p}(\cdot|h, a) \leftarrow f_\theta(\texttt{traj})$ (computed along the trajectory)
    $\hat{V}_{\text{EMA}}, \hat{A}_{\text{EMA}}, x_{\text{EMA}} \leftarrow f_{\theta_{\text{EMA}}}(\texttt{traj})$
    $q_i(z) \leftarrow \begin{cases} 1 - \epsilon_{\text{WTA}} + \frac{\epsilon_{\text{WTA}}}{|\mathcal{Z}|} & z = \arg\min_z \|\hat{x}_{i+1,z} - x_{\text{EMA},i+1}\| \\ \frac{\epsilon_{\text{WTA}}}{|\mathcal{Z}|} & \text{otherwise} \end{cases}$ (posterior)
    $\mathcal{L}_{\text{rec}} \leftarrow \sum_{i>k} \sum_z q_i(z) \|\hat{x}_{i+1,z} - x_{\text{EMA},i+1}\|^2$ (reconstruction loss)
    $\mathcal{L}_{\text{prior}} \leftarrow -\sum_{i>k} \sum_z q_i(z) \log \hat{p}(z|h_i, a_i)$ (prior loss)
    if $\epsilon_{\text{WTA}} < \epsilon_{\text{cutoff}}$ **then**
      $\hat{B}_i \leftarrow \sum_z \left(q_i(z) - \texttt{sg}(\hat{p}(z|h_i, a_i))\right) \hat{B}(h_i, a_i, z)$ ($\hat{B}$ constraint)
    else
      $\hat{B}_i \leftarrow 0$
    end if
    $\pi_{\text{target}} \leftarrow \texttt{softmax}(\frac{\hat{A}_{\text{EMA}}}{T_{\text{EMA}}})$, $\pi \leftarrow \texttt{softmax}(\frac{\texttt{sg}(\hat{A})}{T})$
    $\hat{A}_i \leftarrow \hat{A}(h_i, a_i) - \sum_a \hat{A}(h_i, a) \pi_{\text{target}}(h_i, a)$ ($\hat{A}$ constraint)
    $\mathcal{L}_{\text{DAE}} \leftarrow \left(\sum_{j=k}^{n} \gamma^{j-k}(r_{i+j} - \hat{A}_{i+j} - \hat{B}_{i+j}) + \gamma^{n-k+1} \hat{V}_{\text{EMA},i+n+k} - \hat{V}_i\right)^2$
    $\mathcal{L}_T \leftarrow \log T + \beta_{\text{KL}} \text{KL}(\pi_{\text{target}} || \pi)$
    $\beta_V \leftarrow \frac{1 - \epsilon_{\text{WTA}}}{\text{Var}_D(G)}$
    $\theta, T \leftarrow \texttt{optimizer}(\beta_V \mathcal{L}_{\text{DAE}} + \beta_{\text{prior}} \mathcal{L}_{\text{prior}} + \beta_{\text{rec}} \mathcal{L}_{\text{rec}} + \mathcal{L}_T)$
    $\theta_{\text{EMA}} \leftarrow \tau \theta_{\text{EMA}} + (1 - \tau)\theta$
    $T_{\text{EMA}} \leftarrow \tau T_{\text{EMA}} + (1 - \tau) T$
  end if
end for

8.3 Hyperparameters

Table 3 summarizes the default hyperparameters used in the experiments. For the learning rate, we found linear warmup to be important, which is likely due to the use of LSTMs that can be unstable in the early stage of training. The batch size indicates the number of trajectories instead of frames, and the number of frames per batch is (backup length + burn-in) $\times$ batch size.

Table 3: Default hyperparameters for the experiments.

Parameter	Value
Replay buffer size	1000000
Minimum Steps before training	20000
Number of parallel actors	16
ϵ (training)	Linearly annealed from 1 to 0.01 in the first 1M steps
ϵ (evaluation)	0.001
Optimizer	Adam (Kingma & Ba, 2014)
Learning rate	Linear warmup from 0 to 1.25×10^{-4} in the first 100000 steps then linearly annealed to 1.25×10^{-5} throughout training
Adam β	(0.9, 0.95)
Adam ϵ	10^{-6}
Replay ratio ($\frac{\text{Gradient updates}}{\text{Environment steps}}$)	0.0625
Backup length	16
Burn-in	16
Batch size	16
$\lvert\mathcal{Z}\rvert$	16
ϵ_{WTA}	Linearly annealed from 1 to 0 in the first 500000 steps
τ (target EMA)	0.995
β_{prior}	0.025
β_{rec}	1
β_{KL}	20

8.4 Network Architecture

Figure 6 shows the network architecture used in the experiments. In the scaling experiments, we only multiply the width of the convolutional layers in the ResNet by the multiplier, with the sizes of other layers fixed. Table 4 summarizes the number of parameters in each component.

We use Layer Normalization (Ba et al., 2016) before the nonlinear activations in the MLP heads and after the LSTM. In addition, we apply RMS normalization (Zhang & Sennrich, 2019) to the image embeddings (after the linear layer) such that the SPR objective (cosine similarity) becomes equivalent to L2 distance between the embeddings.

Similar to DreamerV3 (Hafner et al., 2025), we use block diagonal LSTM (Van Keirsbilck et al., 2019) (with 16 blocks) to reduce the computational complexity and number of parameters.

The neural network implementation is based on `PyTorch` (Paszke, 2019), except for the block diagonal LSTM, where we used an efficient implementation from `flashrnn` (Pöppel et al., 2024).

8.5 DreamerV3 Baseline

We use the official reimplementation from `https://github.com/danijar/dreamerv3/`. By default, DreamerV3 uses a slightly different environment configuration compared to the Dopamine baseline, namely, higher observation resolution, full action sets, and a larger discount

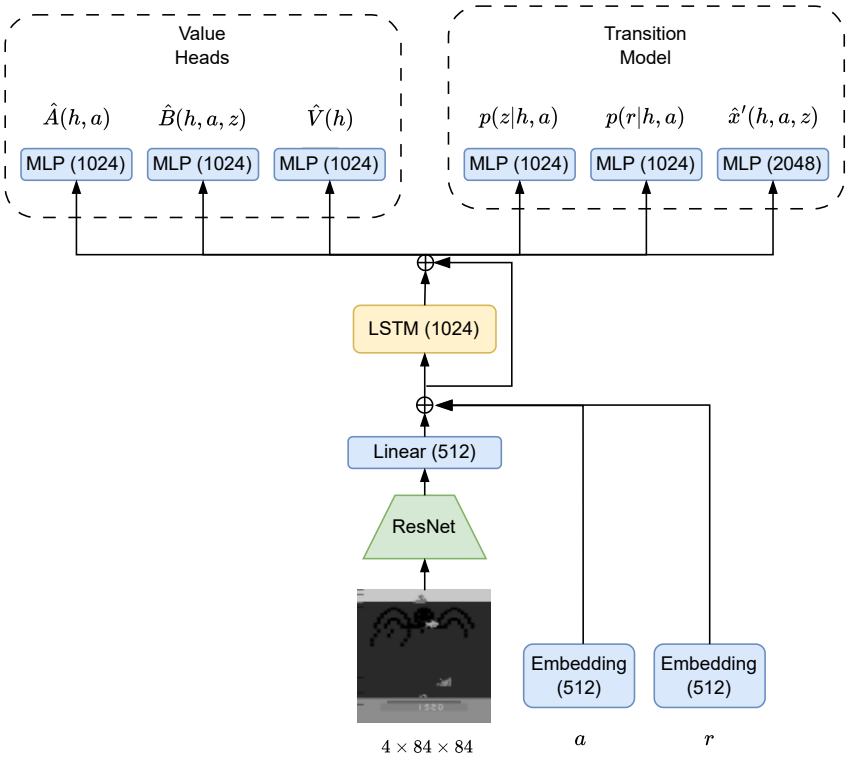

Figure 6: The network architecture. We use the same ResNet encoder proposed by Espeholt et al. (2018). All MLP heads have 1 hidden layer. Previous actions and rewards are first embedded into 512-dimensional vectors before summed together with the image embedding to form the final embedding vector. We use a residual connection around the LSTM similar to Kim et al. (2017).

Table 4: Number of parameters in each component.

Component	Parameters (millions)
IMPALACNN (m=1/2/4/8)	2 / 4 / 9 / 22
LSTM	3
Transition Model	21
Value heads ($\hat{A}$, $\hat{B}$, $\hat{V}$)	4

factor. For a closer comparison, we lower the resolution to 80×80[7], use the minimal action sets, and lower the discount factor to 0.99.

In addition, as DreamerV3 uses a lower replay ratio by default[8], we increase its replay ratio such that the number of gradients is the same as DAE. It should be noted that DreamerV3 trains the actor-critic network and the dynamics model separately, where the actor-critic learns from trajectories generated from the learned dynamics model. This makes a direct comparison difficult, as the number of frames seen by the dynamics model and the actor-critic model differs by a factor of the rollout (imagination)

[7]The preset network only accepts resolutions divisible by 16, so we lower it to 80×80 instead of the standard 84×84.

[8]The definition of replay ratio in Dreamer ($\frac{\text{frames per batch}}{\text{frames per gradient}}$) is slightly different from the convention ($\frac{\text{gradients}}{\text{env. steps}}$)

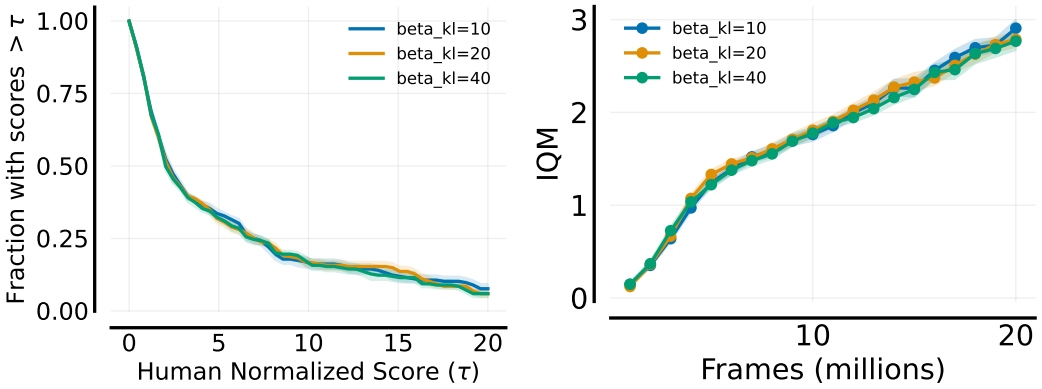

Figure 7: Performance profile (left) and sample efficiency (right) of DAE under variations of β_{KL}.

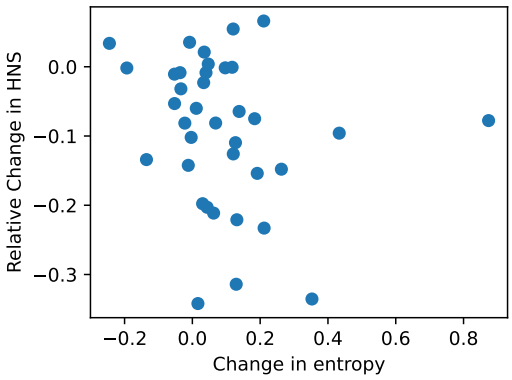

Figure 8: Scatter plot of changes in prior entropy (Δ_H) and changes in HNS (Δ_{HNS}) (aggregated over 5 seeds). Each point represents an environment. We remove the outliers (top/bottom 10% changes in HNS) for better visualization.

length. For simplicity, we adjust the batch size such that the ground truth frames seen by the agent throughout training remains fixed.

8.6 Additional Results

Per-environment learning curves and final evaluation scores of the scaling and the ablation experiments can be found in Figure 12, Figure 13, Table 5, and Table 6.

Sensitivity of β_{KL} This hyperparameter controls how much the online policy can deviate from the EMA policy. Figure 7 shows that the method is robust to variations in this hyperparameter.

Correlation between HNS and prior entropy Here, we examine how the changes in prior entropy ($\Delta_H = H_{\text{frame-stack}} - H_{\text{LSTM}}$) relate to the relative changes in human-normalized scores ($\Delta_{\text{HNS}} = \frac{\text{HNS}_{\text{frame-stack}} - \text{HNS}_{\text{LSTM}}}{\text{HNS}_{\text{LSTM}}}$, we use relative changes in HNS because the scales vary considerably across environments) by comparing the LSTM agent and the Frame-stacking agent. The prior entropy is estimated by averaging the entropy of $p(\cdot|h_t, a_t)$ over training samples throughout training. We find a weak but negative (Spearman) correlation ($\rho = -0.198$), suggesting that increase in partial observability (increase in entropy) is related to decrease in performance; however, the sample size is relatively small (47 environments), and we defer a more comprehensive analysis to future work.

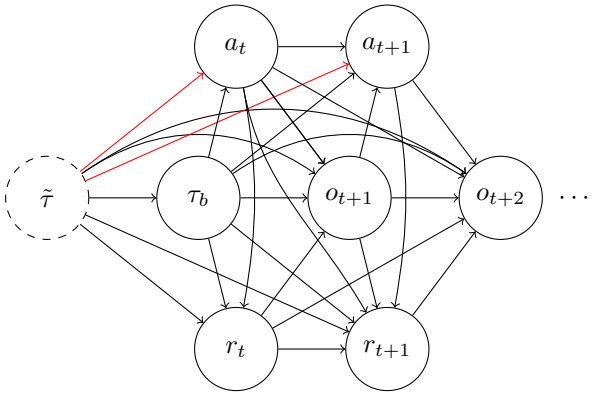

Figure 9: Causal relationship between variables of a truncated sequence for a general POMDP. $\tilde{\tau}$ denotes the truncated part of the sequence, and τ_b denotes the burn-in part of the sequence. The red arrows show the dependencies between actions and $\tilde{\tau}$ when using recurrent actors.

Confounding Confounding is a phenomenon in which unobserved variables influence both the actions and the outcomes, creating spurious correlations (Pearl, 2009). In the case of recurrent agents, this can happen when the behavior policy has access to variables that are not present during training.

For POMDPs, since states are replaced by histories, we have to process sequences of observations instead of singular states. In deep RL, this is typically achieved using recurrent neural networks (RNNs), such as LSTMs or GRUs (Hochreiter & Schmidhuber, 1997; Hausknecht & Stone, 2015; Mnih et al., 2016; Kapturowski et al., 2018; Gruslys et al., 2018; Cho et al., 2014; Hafner et al., 2025), but this can be computationally expensive during training when trajectories extend to thousands of steps. Instead, it is common to truncate histories by sampling random segments of contiguous trajectories from the replay buffer, and use the first few steps as the context (burn-in) before updating the values (Kapturowski et al., 2018):

$$\underbrace{o_0, a_0, r_0, \cdots, o_{t-k-1}}_{\tilde{\tau}\text{ (truncated)}}, \underbrace{a_{t-b-1}, r_{t-b-1}, o_{t-b}, \cdots}_{\tau_b\text{ (burn-in)}}, \underbrace{a_{t-1}, r_{t-1}, o_t, a_t, r_t, \cdots, o_{t+k}}_{\text{value updates}}$$

In this setting, the truncated part of a trajectory can act as confounders and create spurious correlation between the sampled segments and the future rewards (see Figure 9 for the causal graph). This can be mitigated by storing recurrent states in the replay buffer (Kapturowski et al., 2018); however, they may not always be available (e.g., offline settings, or policies with non-recurrent sequence models such as transformers (Vaswani et al., 2017; Parisotto et al., 2020)). If we learn the value functions (i.e., predict $\sum_{t'>t} r_{t'}$) by conditioning on $(\tau_b, a_t, r_t, o_{t+1}, \cdots)$, then $\tilde{\tau}$ can influence both the input variables and the output variables and lead to confounding.

In Figure 10, we construct a toy POMDP to illustrate this effect. In this environment, the optimal policy is $\pi^*(\text{up}|o_0) = p \in [0, 1]$ (arbitrary), and $\pi^*(a|o_0, a_0{=}a, o_1) = 1$ (repeat previous actions). Consider the case where the behavior policy is the optimal policy $\pi^*(\text{up}|o_0) = 0.5$, but the burn-in length is 0 (i.e., memoryless) for the target policy. In this case, we will incorrectly infer that $V^\pi(o_1) = Q^\pi(o_1, \cdot) = 1$ for any target policy π, since all the collected trajectories receive a reward 1 irrespective of the action a_1.

Here, we examine the effect of this misalignment between the behavior policy and the target policy in a larger scale using the ALE. Specifically, we consider two sampling strategies that differ in their dependencies on the truncated part of the trajectories (red arrows in Figure 9):

1. Aligned: the behavior policy only conditions on the past k frames, where k is equal to the burn-in length during training.
2. Recurrent: the behavior policy conditions on the full history.

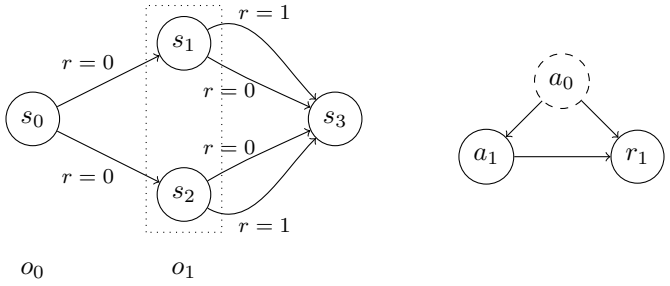

Figure 10: **Left**: A toy POMDP with 4 states and 2 actions. The nodes and the arrows represent the states and the actions (`up`, `down`), respectively. s_0 is the starting state and s_3 is the terminal (absorbing) state. The agent does not observe the underlying state but only the emitted observation at each time step, o_0 and o_1, where both s_1 and s_2 emit the same observation o_1. **Right**: The (simplified) causal relationship between a_0, a_1, and r_1. We ignore other variables as they do not influence r_1. The variable a_0 can act as a confounder during training when the target policy is memoryless.

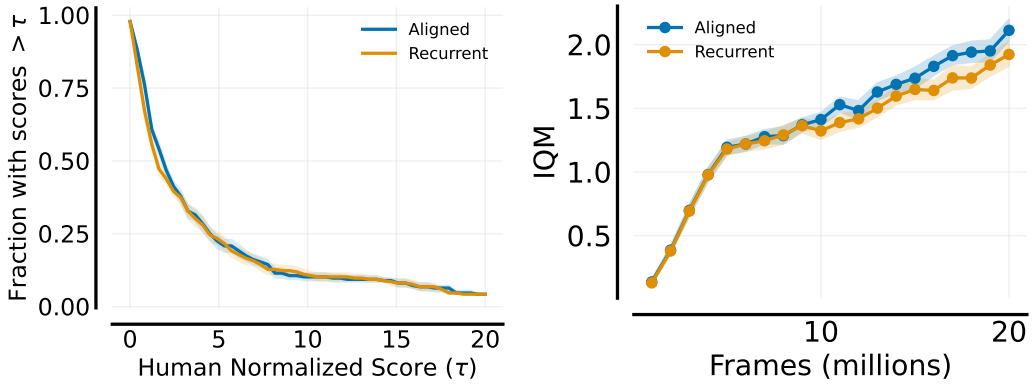

Figure 11: Effect of confounding.

In addition, we reduce the burn-in length (16→4), and do not store recurrent states in the replay buffer to enhance the effect of partial observability. Figure 11 shows that this subtle change in the behavior policy leads to an approximately 10% change in the IQM, suggesting that confounding should not be ignored when designing POMDP agents.

Finally, we note that, in general, deleting the red arrows is not enough to eliminate confounding since τ_b and r_t can still be influenced by $\tilde{\tau}$; however, our results suggest that this simple change can already have non-trivial effects on the agent's performance.

8.7 Compute Resources

All experiments were conducted using an internal cluster of Nvidia H100 GPUs. Runtime varies across environments and scales approximately proportionally with the model size (m), where a single run of the $m = 8$ model takes approximately 18 hours, while a run with $m = 4$ takes approximately 10 hours.

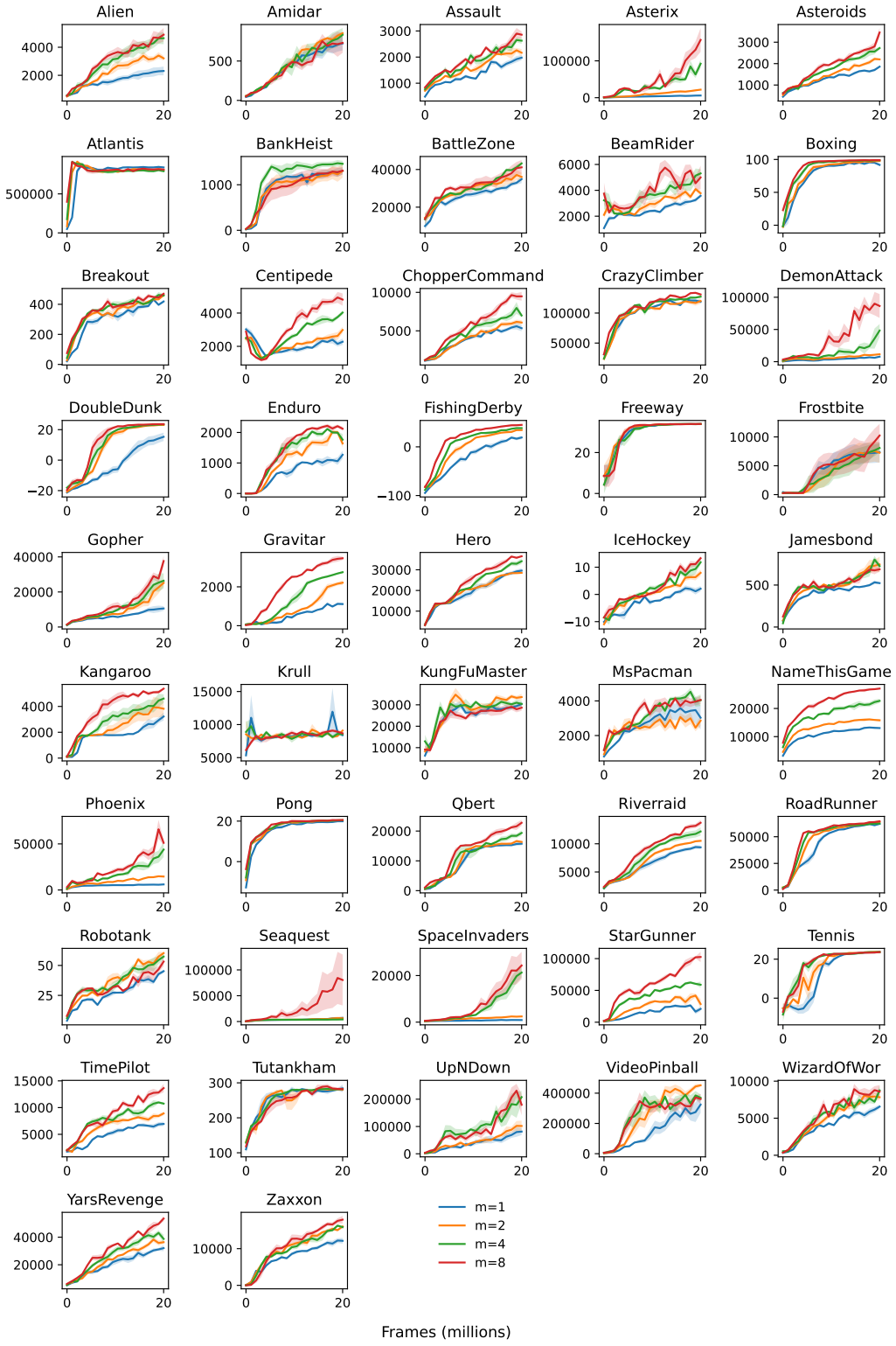

Figure 12: Per-environment learning curve. Lines and shadings represent the average and 1 standard error (5 random seeds), respectively.

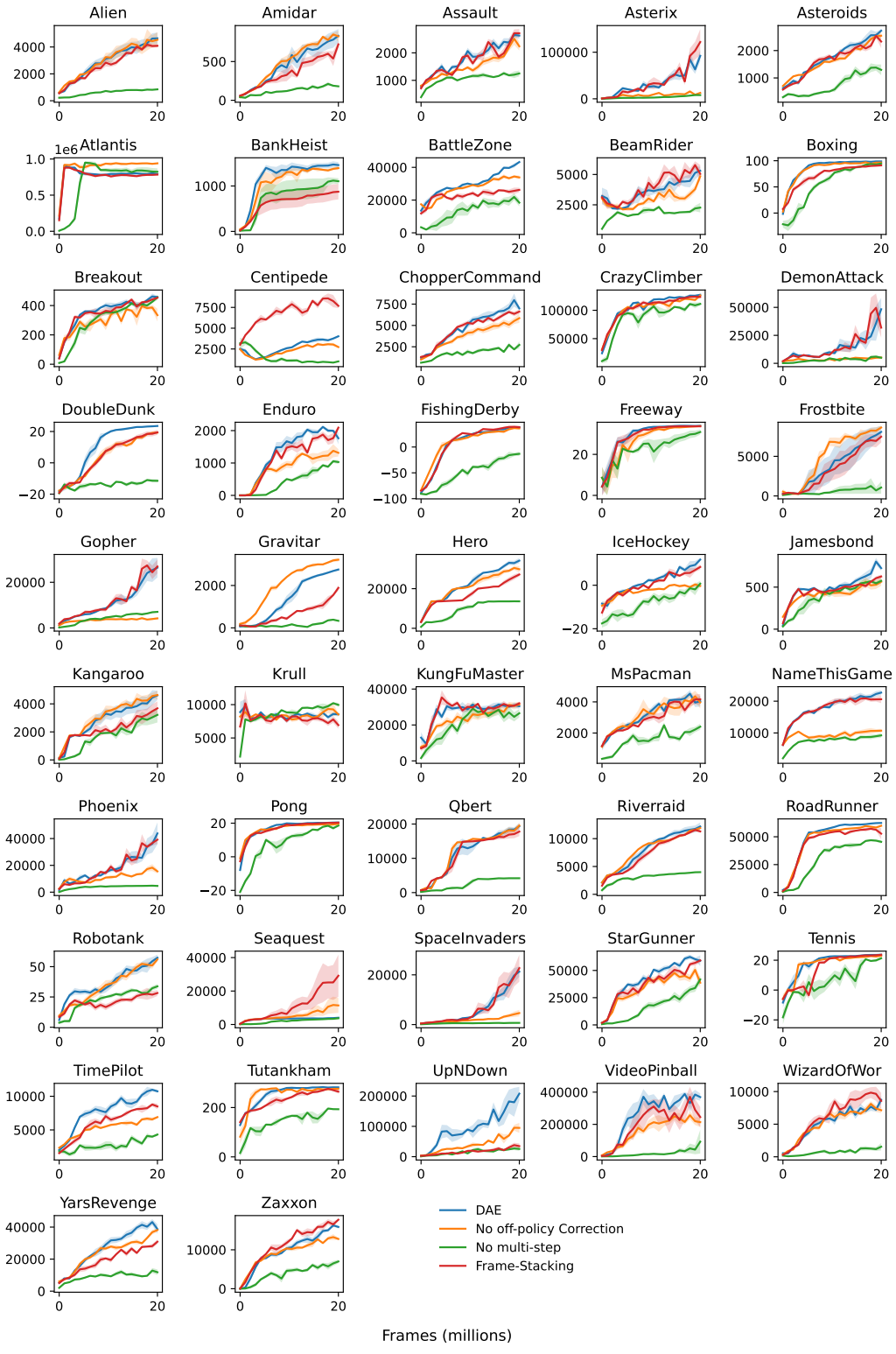

Figure 13: Per-environment learning curve for the ablation study. Lines and shadings represent the average and 1 standard error (5 random seeds), respectively.

Table 5: Per-environment score (average $\pm$ 1 standard error, 5 random seeds).

Environment	$m = 1$	$m = 2$	$m = 4$	$m = 8$
Alien	2310.7 ± 314.3	3211.6 ± 169.2	4644.2 ± 426.6	4893.1 ± 477.5
Amidar	723.5 ± 87.1	853.9 ± 9.1	832.8 ± 86.4	730.4 ± 126.1
Assault	1978.4 ± 98.1	2166.2 ± 139.7	2628.5 ± 121.5	2857.1 ± 125.3
Asterix	5873.4 ± 622.8	21777.0 ± 1557.2	92247.0 ± 12015.9	156815.4 ± 32296.4
Asteroids	1857.8 ± 48.8	2198.1 ± 19.9	2726.2 ± 59.2	3454.6 ± 208.3
Atlantis	839273.2 ± 11004.1	808648.4 ± 9425.6	791305.6 ± 13727.6	811127.6 ± 13961.9
BankHeist	1305.4 ± 76.5	1304.1 ± 149.7	1460.8 ± 64.0	1309.1 ± 80.6
BattleZone	34892.0 ± 2089.0	36060.0 ± 2278.4	43248.0 ± 457.3	41152.0 ± 3899.3
BeamRider	3577.3 ± 124.5	3772.2 ± 224.1	5309.6 ± 329.0	4995.9 ± 912.4
Boxing	91.7 ± 2.3	97.7 ± 0.7	98.7 ± 0.2	99.0 ± 0.2
Breakout	418.9 ± 8.5	461.6 ± 12.7	458.2 ± 7.2	470.1 ± 10.3
Centipede	2277.0 ± 192.8	2982.9 ± 185.7	4028.5 ± 126.8	4810.4 ± 368.8
ChopperCommand	5342.0 ± 404.1	6079.2 ± 301.8	6994.4 ± 206.5	9482.0 ± 568.2
CrazyClimber	118894.8 ± 2867.3	120037.6 ± 4495.2	127332.4 ± 2315.6	131008.4 ± 2073.0
DemonAttack	7789.3 ± 394.1	11520.5 ± 581.2	48341.6 ± 10113.5	86428.9 ± 18579.7
DoubleDunk	15.3 ± 2.7	23.1 ± 0.2	23.5 ± 0.2	23.6 ± 0.1
Enduro	1269.1 ± 132.5	1639.5 ± 142.5	1763.9 ± 185.4	2122.5 ± 63.3
FishingDerby	19.2 ± 3.5	34.5 ± 1.4	38.5 ± 0.8	45.1 ± 1.9
Freeway	34.0 ± 0.0	33.9 ± 0.0	34.0 ± 0.0	34.0 ± 0.0
Frostbite	7360.8 ± 1772.6	7254.0 ± 1741.3	8085.2 ± 781.1	10241.8 ± 2006.0
Gopher	10577.7 ± 1347.5	25171.1 ± 1970.2	26382.4 ± 4267.6	37633.7 ± 2919.8
Gravitar	1112.8 ± 66.4	2211.2 ± 119.7	2752.2 ± 57.4	3469.4 ± 89.7
Hero	29598.3 ± 1142.3	28531.4 ± 91.3	34141.4 ± 1363.4	36544.7 ± 158.9
IceHockey	2.1 ± 1.0	7.9 ± 1.2	11.8 ± 0.7	13.3 ± 0.8
Jamesbond	523.0 ± 9.0	745.0 ± 92.3	725.8 ± 30.2	687.8 ± 27.8
Kangaroo	3221.2 ± 312.8	3821.6 ± 604.8	4606.8 ± 330.1	5374.8 ± 58.7
Krull	8406.2 ± 272.8	9092.9 ± 113.6	8547.8 ± 310.5	8676.5 ± 207.5
KungFuMaster	30375.2 ± 1098.0	33618.4 ± 1513.4	30416.8 ± 1440.7	28644.8 ± 1421.5
MsPacman	3028.5 ± 673.5	2861.5 ± 308.6	4041.3 ± 334.5	4056.8 ± 230.5
NameThisGame	13043.0 ± 195.5	15804.1 ± 472.5	22694.5 ± 770.2	27069.2 ± 236.4
Phoenix	6029.2 ± 340.8	14570.2 ± 834.6	43868.0 ± 8193.5	51149.6 ± 2753.9
Pong	19.9 ± 0.3	20.4 ± 0.1	20.4 ± 0.0	20.5 ± 0.1
Qbert	15747.4 ± 580.5	16345.5 ± 430.6	19361.2 ± 877.3	22770.8 ± 971.5
Riverraid	9367.6 ± 305.8	10520.2 ± 137.2	12180.8 ± 739.3	13728.1 ± 505.7
RoadRunner	62068.4 ± 738.6	63964.8 ± 1631.6	62404.0 ± 502.7	64405.2 ± 1384.1
Robotank	45.2 ± 1.5	60.3 ± 1.1	57.4 ± 2.5	53.2 ± 3.8
Seaquest	7098.9 ± 1574.6	6303.8 ± 2701.4	4040.7 ± 232.9	80438.1 ± 49314.9
SpaceInvaders	888.5 ± 29.2	2440.2 ± 89.0	21300.5 ± 2075.8	24328.6 ± 5850.2
StarGunner	20946.4 ± 3482.2	28576.4 ± 7833.9	58930.0 ± 4928.2	102573.6 ± 8047.6
Tennis	23.6 ± 0.0	23.7 ± 0.0	23.7 ± 0.0	23.6 ± 0.0
TimePilot	6943.6 ± 398.8	8904.0 ± 355.0	10742.8 ± 222.8	13643.6 ± 796.8
Tutankham	284.2 ± 10.9	280.7 ± 1.5	282.0 ± 1.4	282.2 ± 6.9
UpNDown	81297.4 ± 12656.5	101890.8 ± 19512.4	207808.6 ± 22451.9	179719.8 ± 39120.9
VideoPinball	324508.5 ± 55021.9	452230.7 ± 9440.9	369343.4 ± 22801.6	361586.3 ± 33070.4
WizardOfWor	6562.8 ± 285.9	7830.8 ± 409.1	8750.4 ± 658.9	8589.6 ± 880.4
YarsRevenge	32066.6 ± 2213.8	36369.9 ± 1346.9	38757.1 ± 2986.1	53512.9 ± 1991.0
Zaxxon	12138.0 ± 773.9	16053.2 ± 198.0	15867.6 ± 399.5	17897.6 ± 1049.9

Table 6: Per-environment score (average ± 1 standard error, 5 random seeds).

Environment	DAE	No off-policy corr.	No multi-step	Frame-Stacking
Alien	4644.2 ± 426.6	4534.9 ± 569.2	847.8 ± 43.4	4088.2 ± 103.6
Amidar	832.8 ± 86.4	825.2 ± 71.6	180.3 ± 7.2	721.9 ± 42.6
Assault	2628.5 ± 121.5	2238.6 ± 154.8	1251.0 ± 105.6	2713.5 ± 152.0
Asterix	92247.0 ± 12015.9	12720.2 ± 2084.5	7962.0 ± 1074.1	121897.6 ± 27785.2
Asteroids	2726.2 ± 59.2	2542.4 ± 146.8	1295.8 ± 196.2	2329.3 ± 241.7
Atlantis	791305.6 ± 13727.6	940546.0 ± 9328.4	823348.0 ± 47007.6	784672.8 ± 11931.7
BankHeist	1460.8 ± 64.0	1394.2 ± 40.3	1105.2 ± 65.5	874.0 ± 168.6
BattleZone	43248.0 ± 457.3	33860.0 ± 1104.6	18440.0 ± 2342.9	26240.0 ± 1349.7
BeamRider	5309.6 ± 329.0	4848.9 ± 285.8	2267.9 ± 397.3	5046.7 ± 736.5
Boxing	98.7 ± 0.2	96.5 ± 0.8	94.7 ± 2.2	91.1 ± 1.3
Breakout	458.2 ± 7.2	334.5 ± 43.5	455.9 ± 18.3	453.3 ± 8.0
Centipede	4028.5 ± 126.8	2732.9 ± 132.1	1007.8 ± 85.8	7687.2 ± 274.5
ChopperCommand	6994.4 ± 206.5	5850.8 ± 549.5	2720.0 ± 413.6	6623.2 ± 244.3
CrazyClimber	127332.4 ± 2315.6	126313.6 ± 2115.7	111269.2 ± 2672.8	123614.4 ± 3092.7
DemonAttack	48341.6 ± 10113.5	5218.7 ± 686.2	5020.6 ± 989.9	31863.0 ± 8886.9
DoubleDunk	23.5 ± 0.2	19.3 ± 0.6	-11.5 ± 1.0	19.5 ± 1.6
Enduro	1763.9 ± 185.4	1314.1 ± 112.3	1029.2 ± 59.7	2091.5 ± 46.7
FishingDerby	38.5 ± 0.8	36.1 ± 3.0	-13.2 ± 4.1	38.3 ± 3.0
Freeway	34.0 ± 0.0	33.8 ± 0.1	30.9 ± 1.3	33.7 ± 0.2
Frostbite	8085.2 ± 781.1	8635.1 ± 207.1	1064.0 ± 782.3	7483.8 ± 736.3
Gopher	26382.4 ± 4267.6	4191.4 ± 450.7	7041.7 ± 211.0	26935.4 ± 3043.6
Gravitar	2752.2 ± 57.4	3216.6 ± 82.3	331.8 ± 14.9	1887.2 ± 133.9
Hero	34141.4 ± 1363.4	29812.6 ± 1610.7	13605.3 ± 37.5	27143.2 ± 967.4
IceHockey	11.8 ± 0.7	-0.3 ± 0.6	0.7 ± 1.3	8.4 ± 1.2
Jamesbond	725.8 ± 30.2	564.0 ± 48.3	574.6 ± 31.6	626.6 ± 22.9
Kangaroo	4606.8 ± 330.1	4632.0 ± 307.4	3214.4 ± 541.7	3682.4 ± 494.5
Krull	8547.8 ± 310.5	8526.4 ± 96.9	9962.0 ± 317.9	6928.6 ± 548.1
KungFuMaster	30416.8 ± 1440.7	31408.0 ± 1416.6	26532.4 ± 2345.3	32060.8 ± 997.7
MsPacman	4041.3 ± 334.5	3967.0 ± 387.1	2407.8 ± 197.2	4167.5 ± 336.8
NameThisGame	22694.5 ± 770.2	10720.4 ± 569.4	9238.1 ± 741.8	20613.1 ± 1231.7
Phoenix	43868.0 ± 8193.5	15533.0 ± 1577.2	4650.3 ± 159.4	39143.1 ± 3136.0
Pong	20.4 ± 0.0	19.5 ± 0.2	18.6 ± 0.2	20.4 ± 0.1
Qbert	19361.2 ± 877.3	19561.2 ± 915.4	4189.0 ± 87.4	17798.3 ± 1326.0
Riverraid	12180.8 ± 739.3	11916.2 ± 237.8	3980.9 ± 79.9	11299.7 ± 214.2
RoadRunner	62404.0 ± 502.7	59853.2 ± 767.2	45465.6 ± 1543.8	52798.4 ± 5170.7
Robotank	57.4 ± 2.5	56.0 ± 1.7	33.8 ± 2.0	28.3 ± 2.4
Seaquest	4040.7 ± 232.9	11388.6 ± 4435.2	3739.4 ± 618.7	29157.9 ± 12614.7
SpaceInvaders	21300.5 ± 2075.8	4626.3 ± 1308.1	679.9 ± 38.2	22698.2 ± 5437.3
StarGunner	58930.0 ± 4928.2	38745.6 ± 4996.7	41747.6 ± 3573.0	59159.2 ± 1743.8
Tennis	23.7 ± 0.0	23.0 ± 0.1	21.3 ± 0.8	23.7 ± 0.0
TimePilot	10742.8 ± 222.8	6904.8 ± 124.2	4312.8 ± 212.6	8489.2 ± 400.4
Tutankham	282.0 ± 1.4	276.0 ± 6.7	193.1 ± 1.4	264.5 ± 8.3
UpNDown	207808.6 ± 22451.9	94944.8 ± 16508.2	25498.3 ± 6082.6	34799.4 ± 9229.8
VideoPinball	369343.4 ± 22801.6	213833.7 ± 30383.0	92563.9 ± 73976.6	245359.0 ± 62400.9
WizardOfWor	8750.4 ± 658.9	7143.6 ± 449.0	1544.0 ± 562.6	8562.8 ± 1234.9
YarsRevenge	38757.1 ± 2986.1	38156.7 ± 1201.2	11820.9 ± 1718.7	30876.9 ± 1361.5
Zaxxon	15867.6 ± 399.5	12788.4 ± 300.6	7003.2 ± 466.9	17726.8 ± 171.2