Skill or Luck?
Return Decomposition via Advantage Functions

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning from off-policy data is essential for sample-efficient reinforcement learning. In the present work, we build on the insight that the advantage function can be understood as the causal effect of an action on the return, and show that this allows us to decompose the return of a trajectory into parts caused by the agent's actions (skill) and parts outside of the agent's control (luck). Furthermore, this decomposition enables us to naturally extend Direct Advantage Estimation (DAE) to off-policy settings (Off-policy DAE). The resulting method can learn from off-policy trajectories without relying on importance sampling techniques or truncating off-policy actions. We compare the uncorrected multi-step method, which has shown strong empirical results despite ignoring off-policy corrections, to DAE and Off-policy DAE, and provide intuition on when the corrections can be omitted. Finally, we use the MinAtar environments to illustrate how ignoring off-policy corrections can lead to suboptimal policy optimization performance.

1 Introduction

Imagine the following scenario: One day, A and B both decide to purchase a lottery ticket, hoping to win the grand prize. Each of them chose their favorite set of numbers, but only A got *lucky* and won the million-dollar prize. In this story, we are likely to say that A got *lucky* because, while A's action (picking a set of numbers) led to the reward, the expected rewards are the same for both A and B (assuming the lottery is fair), and A was ultimately rewarded due to something outside of their control.

This shows that, in a decision-making problem, the return is not always determined solely by the actions of the agent, but also by the randomness of the environment. Therefore, for an agent to correctly distribute credit among its actions, it is crucial that the agent is able to reason about the effect of its actions on the rewards and disentangle it from factors outside its control. This is also known as the problem of credit assignment [Minsky, 1961]. While attributing *luck* to the drawing process in the lottery example may be easy, it becomes much more complex in sequential settings, where multiple actions are involved and rewards are delayed.

The key observation of the present work is that we can treat the randomness of the environment as actions from an imaginary agent, whose actions determine the future of the decision-making agent. Combining this with the idea that the advantage function can be understood as the causal effect of an action on the return [Pan et al., 2022], we show that the return can be decomposed into parts caused by the agent (skill) and parts that are outside the agent's control (luck). Furthermore, we show that this decomposition admits a natural way to extend Direct Advantage Estimation (DAE), an on-policy multi-step learning method, to off-policy settings (Off-policy DAE). The resulting method makes minimal assumptions about the behavior (data-collecting) policy and shows strong empirical performance.

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

Our contributions can be summarized as follows:

- We draw connections between Monte-Carlo (MC) methods and DAE, and show that DAE can be understood as a generalization of MC methods that utilizes the relationship between states and actions to achieve better sample efficiency.
- We demonstrate that applying DAE directly to off-policy data can be biased in stochastic environments, and propose a fix to extend DAE to off-policy settings (Off-policy DAE).
- We compare DAE, Off-policy DAE, and the uncorrected multi-step method [Hernandez-Garcia and Sutton, 2019] and evaluate their performance empirically in both deterministic and stochastic environments.

2 Background

In this work, we consider a discounted Markov Decision Process $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, transition probability $p(s'|s,a)$, expected reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. To reduce notational cluttering, we shall omit the discount factor in the following discussion unless otherwise stated. A policy is a function $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ which maps states to distributions over the action space. The goal of RL is to find a policy that maximizes the expected return, $\pi^* = \arg\max_\pi \mathbb{E}_\pi[G]$, where $G = r_0 + r_1 + ...$ and $r_t = r(s_t, a_t)$. The value function of a state is defined by $V^\pi(s) = \mathbb{E}_\pi[G|s_0{=}s]$, the Q-function of a state-action pair is similarly defined by $Q^\pi(s,a) = \mathbb{E}_\pi[G|s_0{=}s, a_0{=}a]$ [Sutton et al., 1998]. These functions quantify the return we can expect from a given state or state-action pair by following a given policy π, and turn out to be useful for policy improvements. They are typically unknown and are learned via interactions with the environment.

Direct Advantage Estimation The advantage function, defined by $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, is another quantity that is useful to policy optimization. Recently, Pan et al. [2022] showed that the advantage function can be understood as the causal effect of an action on the return, and is more stable under policy variations (under mild assumptions) compared to the Q-function. They argued that it might be an easier target to learn when used with function approximation, and proposed Direct Advantage Estimation (DAE), which estimates the advantage function directly by

$$A^\pi = \arg\min_{\hat{A} \in F_\pi} \mathop{\mathbb{E}}_{\pi} \left[\left(\sum_{t=0}^{\infty} (r_t - \hat{A}_t) \right)^2 \right], \quad F_\pi = \left\{ f \,\middle|\, \sum_{a \in \mathcal{A}} f(s,a)\pi(a|s) = 0 \right\} \tag{1}$$

where $\hat{A}_t = \hat{A}(s_t, a_t)$. The method can also be seamlessly combined with a bootstrapping target to perform multi-step learning by iteratively minimizing the constrained squared error

$$L(\hat{A}, \hat{V}) = \mathop{\mathbb{E}}_{\pi} \left[\left(\sum_{t=0}^{n-1} (r_t - \hat{A}_t) + V_{\text{target}}(s_n) - \hat{V}(s_0) \right)^2 \right] \quad \text{subject to } \hat{A} \in F_\pi, \tag{2}$$

where V_{target} is the bootstrapping target, and $(\hat{V}, \hat{A})$ are estimates of the value function and the advantage function. Policy optimization results were reported to improve upon generalized advantage estimation [Schulman et al., 2015b], a strong baseline for on-policy methods. One major drawback of DAE, however, is that it can only estimate the advantage function for on-policy data (note that the expectation and the constraints share the same policy). This limits the range of applications of DAE to on-policy scenarios, which tend to be less sample efficient.

Multi-step learning In RL, we often update estimates of the value functions using previous estimates (e.g., TD(0), SARSA [Sutton et al., 1998]). These methods, however, can suffer from excessive bias when the previous estimates differ significantly from the true value functions, and it was shown that such bias can greatly impact the performance when used with function approximators [Schulman et al., 2015b]. One remedy is to extend the backup length, that is, instead of using one-step targets such as $r(s_0, a_0) + \gamma Q_{\text{target}}(s_1, a_1)$ (Q_{target} being our previous estimate), we include more rewards along the trajectory, i.e., $r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + ... + \gamma^n Q_{\text{target}}(s_n, a_n)$. This way, we can diminish the impact of Q_{target} by the discount factor γ^n. However, using the rewards

along the trajectory relies on the assumption that the samples are on-policy (i.e., the behavior policy is the same as the target policy). To extend such methods to off-policy settings often requires techniques such as importance sampling [Munos et al., 2016, Rowland et al., 2020] or truncating (diminishing) off-policy actions [Precup et al., 2000, Watkins, 1989], which can suffer from high variance or low data utilization with long backup lengths. Surprisingly, empirical results have shown that ignoring off-policy corrections can still lead to substantial speed-ups and is widely adapted in modern deep RL algorithms [Hernandez-Garcia and Sutton, 2019, Hessel et al., 2018, Gruslys et al., 2017].

3 Monte-Carlo Methods and Linear Regression

To build intuition, let us first revisit Monte-Carlo (MC) methods through the lens of regression. In a typical linear regression problem, we are given a dataset $\{(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}\}$, and tasked to find coefficients $w \in \mathbb{R}^n$ minimizing the error $\sum_i (w \cdot x_i - y_i)^2$. In RL, the dataset often consists of transitions or sequences of transitions (as in multi-step methods) and their returns, that is, (τ_i, G_i) where τ_i has the form $(s_0, a_0, s_1, a_1, ...)$ and G_i is the return associated with τ_i. However, τ may be an abstract object which cannot be used directly for regression, and we must first map τ to a feature vector $\phi(\tau) \in \mathbb{R}^n$.[1] For example, in MC methods, we can estimate the value of a state by rolling out trajectories using the target policy starting from the given state and averaging the corresponding returns, i.e., $\mathbb{E}[\sum_{t\geq 0} r_t | s_0 = s] \approx \sum_{i=1}^n G_i/n$. This is equivalent to a linear regression problem, where we first map trajectories to a vector by $\phi_s(\tau) = \mathbb{I}(s_0 = s)$ (vector of length $|\mathcal{S}|$ with elements 1 if the starting state is s or 0 otherwise), and minimize the squared error

$$L(\mathbf{v}) = \sum_i \left[\left(\sum_s v_s \phi_s(\tau_i) - G_i \right)^2 \right], \tag{3}$$

where $\mathbf{v}$ is the vector of linear regression coefficients v_s. Similarly, we can construct feature maps such as $\phi_{s,a}(\tau) = \mathbb{I}(s_0{=}s, a_0{=}a)$ and solve the regression problem to arrive at $Q^\pi(s, a)$. This view shows that MC methods can be seen as linear regression problems with different feature maps. Furthermore, it shows that MC methods utilize rather little information from given trajectories (only the starting state(-action)). An interesting question is whether it is possible to construct features that include more information about the trajectory while retaining the usefulness of the coefficients. For example, let us consider a different feature map $\phi_{s,a}(\tau) = \sum_{t=0}^\infty \mathbb{I}(s_t{=}s, a_t{=}a)$, which results in a vector of size $|\mathcal{S}| \times |\mathcal{A}|$ that counts the multiplicity of each state-action pair in the trajectory. One trivial solution to this regression problem is the coefficients $c_{s,a} = r(s, a)$ since

$$\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} c_{s,a}\phi_{s,a}(\tau) = \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} r(s,a) \sum_{t=0}^\infty \mathbb{I}(s_t{=}s, a_t{=}a) = \sum_{t=0}^\infty r(s_t, a_t) = G. \tag{4}$$

While this set of features and coefficients can fully explain the return, the coefficients are less interesting as they do not reflect the long-term effect of the state-action pairs. However, if we add a constraint for the coefficients $c_{s,a}$ requiring $\sum_a c_{s,a}\pi(a|s) = 0$ for all $s \in \mathcal{S}$, then the solution becomes $c_{s,a} = A^\pi(s, a)$ as shown in Equation 1. This suggests DAE can be understood as a generalization of MC methods by using more informative features.

To see how using more informative features can enhance MC methods, let us consider an example (Figure 1) adapted from Szepesvári [2010]. Suppose the agent has visited state 2 for k times, then the MC estimate of the value has $\mathbf{Var}[\hat{V}(2)] = C/k$, where C is some constant. However, by the time state 2 has k visits, state 1 would have $\sim 9k$ visits, and state 3 would have $\sim 10k$ visits. Consequently, $\mathbf{Var}[\hat{V}(3)] \approx C/10k \ll \mathbf{Var}[\hat{V}(2)]$. This shows a major drawback of MC: it does not utilize the relationship between states 2 and 3, and therefore, an accurate estimate of $\hat{V}(3)$ does not improve the estimate of $\hat{V}(2)$. TD methods, on the other hand, can utilize this relationship to achieve better estimates. Since DAE also utilizes intermediate states and actions, we expect it to have a lower variance on the estimate of $V(2)$ compared to MC. In Figure 1 (right), we compare the estimates from Batch TD(0) (denoted TD(0) for simplicity) [Sutton et al., 1998], MC, and DAE (Equation 2 with $n \to \infty$). We make the following observations: (1) Since most of the trajectories start from

[1] This is not to be confused with the features of states, which are commonly used to approximate value functions.

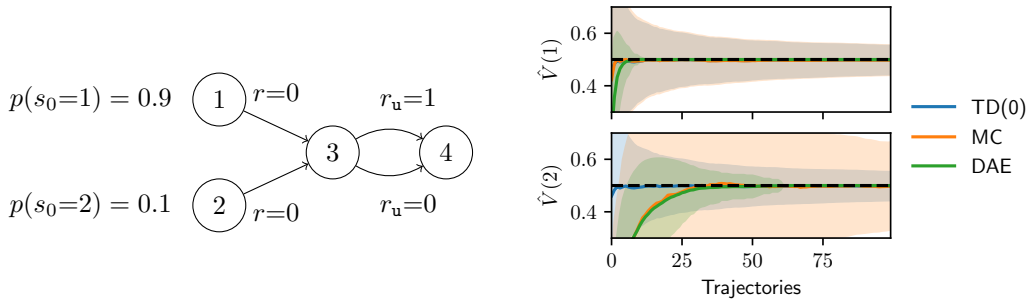

Figure 1: Left: An MDP with $\mathcal{S} = \{1, 2, 3, 4\}$. Both states 1 and 2 have only a single action with immediate rewards 0 that leads to state 3. State 3 has two actions, u and d, that lead to the terminal state 4 with immediate rewards 1 and 0, respectively. Right: We compare the values estimated by three different methods: Batch TD(0), MC, and DAE. Lines and shadings represent the average and one standard deviation of the estimated values over 1000 random seeds. The dashed line represents the true value $V(1) = V(2) = 0.5$. See Appendix A for details.

$s_0 = 1$, both TD(0) and MC share similar variance for $\hat{V}(1)$. (2) The variance of $\hat{V}(2)$ is much lower for TD(0) compared to MC, since TD(0) can utilize $\hat{V}(3)$. (3) DAE achieves even lower variance on both $\hat{V}(1)$ and $\hat{V}(2)$ compared to both TD(0) and MC. This is likely because DAE not only utilizes the relationship between states, but also the sampling policy (constraint of Equation 2).

Here, we demonstrated that DAE can be seen as a generalized MC method that utilizes relationships between states and actions to better estimate the return. However, it remains unclear whether the features used by DAE are enough to fully explain the return or additional features are needed.

4 Return Decomposition

To begin, we observe that, stochasticity of the returns can come from two sources, namely, (1) the stochastic policy employed by the agent, and (2) the stochastic transitions of the environment. To separate their effect, we begin by studying deterministic environments where the only source of stochasticity comes from the agent's policy. Afterward, we demonstrate why DAE fails when transitions are stochastic, and introduce a simple fix which generalizes DAE to off-policy settings.

4.1 The Deterministic Case

First, for deterministic environments, we have $s_{t+1} = h(s_t, a_t)$, where the transition probability is replaced by a deterministic transition function $h : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. As a consequence, the Q-function of a given policy π becomes $Q^\pi(s_t, a_t) = r(s_t, a_t) + V^\pi(s_{t+1})$, and the advantage function becomes $A^\pi(s_t, a_t) = r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t)$. Let's start by examining the sum of the advantage function along a trajectory $(s_0, a_0, s_1, a_1, ...)$ with return G,

$$\sum_{t=0}^{\infty} A^\pi(s_t, a_t) = \sum_{t=0}^{\infty} r(s_t, a_t) + \underbrace{\sum_{t=0}^{\infty} (V^\pi(s_{t+1}) - V^\pi(s_t))}_{\text{telescoping series}} = G - V^\pi(s_0), \tag{5}$$

or, with a simple rearrangement, $G = V^\pi(s_0) + \sum_{t=0}^{\infty} A^\pi(s_t, a_t)$. One intuitive interpretation of this equation is: The return of a trajectory is equal to the average return from the policy (V^π) plus the variations caused by the actions along the trajectory (A^π). Since Equation 5 holds for *any* trajectory, the following equation holds for *any* policy μ

$$\mathop{\mathbb{E}}_{\mu}\left[\left(G - \sum_{t=0}^{\infty} A_t^\pi - V^\pi(s_0)\right)^2\right] = 0. \tag{6}$$

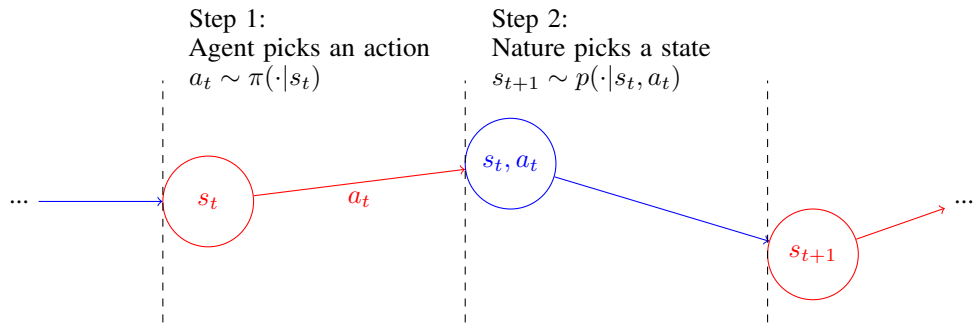

Figure 2: A two-step view of the state transition process. First, we introduce an imaginary agent *nature*, which controls the stochastic part of the transition process. In this view, nature lives in a world with state space $\bar{\mathcal{S}} = \mathcal{S} \times \mathcal{A}$ and action space $\bar{\mathcal{A}} = \mathcal{S}$. At each time step t, the agent chooses its action a_t based on s_t, and, instead of transitioning directly into the next state, it transitions into an intermediate state denoted $(s_t, a_t) \in \bar{\mathcal{S}}$, where nature chooses the next state $s_{t+1} \in \bar{\mathcal{A}}$ based on (s_t, a_t).We use nodes and arrows to represent states and actions by the agent (red) and nature (blue).

This means that (V^π, A^π) is a solution to the off-policy variant of DAE

$$L(\hat{A}, \hat{V}) = \mathbb{E}_{\mu}\left[\left(\sum_{t=0}^{\infty}(r_t - \hat{A}_t) - \hat{V}(s_0)\right)^2\right] \quad \text{s.t.} \sum_{a \in \mathcal{A}} \pi(a|s)\hat{A}(s,a) = 0 \; \forall s \in \mathcal{S}, \tag{7}$$

where the expectation is now taken with respect to an arbitrary behavior policy μ instead of the target policy π in the constraint (cf. Equation 2, with $n \to \infty$). We emphasize that this is a very general result, as we made no assumptions on the behavior policy μ, and only sample trajectories from μ are required to compute the squared error. However, two questions remain: (1) Is the solution unique? (2) Does this hold for stochastic environments? We shall answer these questions in the next section.

4.2 The Stochastic Case

The major difficulty in applying the above argument to stochastic environments is that the telescoping sum (Equation 5) no longer holds because $A^\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}[V^\pi(s_{t+1})|s_t, a_t] - V^\pi(s_t) \neq r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t)$ and the sum of the advantage function becomes

$$\sum_{t=0}^{\infty} A^\pi(s_t, a_t) = \sum_{t=0}^{\infty}\left(r(s_t, a_t) + \mathbb{E}\left[V^\pi(s_{t+1})|s_t, a_t\right] - V^\pi(s_t)\right) = G - \sum_{t=0}^{\infty} B_t^\pi - V^\pi(s_0), \tag{8}$$

where $B_t^\pi = B^\pi(s_t, a_t, s_{t+1}) = V^\pi(s_{t+1}) - \mathbb{E}\left[V^\pi(s_{t+1})|s_t, a_t\right]$. This shows that $V^\pi(s_0)$ and the sum of A^π are not enough to fully characterize the return G (compared to Equation 5). Therefore, applying DAE may lead to biased results when the environment is stochastic and the data is off-policy (see Appendix B for a concrete example). In terms of the analogy we developed in Section 3, this is due to the features being not informative enough to fully capture the variance of the return, and one way to fix this is to include an additional feature $\phi'(\tau)_{s,a,s'} = \sum_{t=0}^{\infty} \mathbb{I}(s_t{=}s, a_t{=}a, s_{t+1}{=}s')$ with corresponding coefficients $B^\pi(s, a, s')$. But what exactly is B^π? To understand the meaning of B^π, we begin by dissecting state transitions into a two-step process, see Figure 2. In this view, we introduce an imaginary agent *nature*, also interacting with the environment, whose actions determine the next states of the decision-making agent. In this setting, nature follows a stationary policy $\bar{\pi}$ equal to the transition probability, i.e., $\bar{\pi}(s'|(s, a)) = p(s'|s, a)$. Since $\bar{\pi}$ is fixed, we omit it in the following discussion. The question we are interested in is, how much do nature's actions affect the return? We note that, while there are no immediate rewards associated with nature's actions, they can still influence future rewards by choosing whether we transition into high-rewarding states or otherwise. Since the advantage function was shown to characterize the causal effect of actions on the return, we now examine nature's advantage function.

By definition, the advantage function is equal to $Q(s, a) - V(s)$.We first compute both $\bar{Q}$ and $\bar{V}$ from nature's point of view (we use the bar notation to differentiate between nature's view and the

agent's view). Since $\bar{\mathcal{S}} = \mathcal{S} \times \mathcal{A}$ and $\bar{\mathcal{A}} = \mathcal{S}$, $\bar{V}$ is now a function of $\bar{\mathcal{S}} = \mathcal{S} \times \mathcal{A}$, and $\bar{Q}$ is a function of $\bar{\mathcal{S}} \times \bar{\mathcal{A}} = \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, taking the form

$$\bar{V}^\pi(s,a) = \mathop{\mathbb{E}}_{\pi}[\sum_{t>0} r_t | s_0{=}s, a_0{=}a] = \mathbb{E}[V^\pi(s_1)|s_0{=}s, a_0{=}a], \tag{9}$$

$$\bar{Q}^\pi(s,a,s') = \mathop{\mathbb{E}}_{\pi}[\sum_{t>0} r_t | s_0{=}s, a_0{=}a, s_1{=}s'] = V^\pi(s'). \tag{10}$$

We thus have $\bar{A}^\pi(s,a,s') = \bar{Q}^\pi(s,a,s') - \bar{V}^\pi(s,a) = V^\pi(s') - \mathbb{E}[V^\pi(s')|s,a]$, which is exactly $B^\pi(s,a,s')$ as introduced at the beginning of this section. Now, if we rearrange Equation 8 into

$$V^\pi(s_0) + \sum_{t=0}^{\infty} \left(A^\pi(s_t,a_t) + B^\pi(s_t,a_t,s_{t+1})\right) = G, \tag{11}$$

then an intuitive interpretation emerges, which reads: *The return of a trajectory can be decomposed into the average return following the policy $V^\pi(s_0)$, the causal effect of the agent's actions $A^\pi(s_t,a_t)$ (**skill**), and the causal effect of nature's actions $B^\pi(s_t,a_t,s_{t+1})$ (**luck**).*

Furthermore, this equation admits a generalization of DAE:

Theorem 1 (Off-policy DAE)**.** Given a behavior policy μ and a target policy π and backup length $n \geq 0$. Let $\hat{A}_t = \hat{A}(s_t,a_t)$, $\hat{B}_t = \hat{B}(s_t,a_t,s_{t+1})$, and the constrained squared error

$$\begin{gathered} L(\hat{A},\hat{B},\hat{V}) = \mathop{\mathbb{E}}_{\mu}\left[\left(\sum_{t=0}^{n} \gamma^t \left(r_t - \hat{A}_t - \gamma \hat{B}_t\right) + \gamma^{n+1}\hat{V}(s_{n+1}) - \hat{V}(s_0)\right)^2\right] \\ \text{subject to} \begin{cases} \sum_{a\in\mathcal{A}} \hat{A}(s,a)\pi(a|s) = 0 & \forall s \in \mathcal{S} \\ \sum_{s'\in\mathcal{S}} \hat{B}(s,a,s')p(s'|s,a) = 0 & \forall (s,a) \in \mathcal{S}\times\mathcal{A} \end{cases}, \end{gathered} \tag{12}$$

then (A^π, B^π, V^π) is a minimizer of the above problem. Furthermore, the minimizer is unique if μ is sufficiently explorative (i.e., non-zero probability of reaching all possible transitions (s,a,s')).

See Appendix C for a proof. In practice, we can minimize the empirical variant of Equation 20 from samples to estimate (V^π, A^π, B^π), which renders this an off-policy multi-step method. We highlight two major differences between this method and other off-policy multi-step methods. (1) Minimal assumptions on the behavior policy are made, and no knowledge of the behavior policy is required during training (in contrast to importance sampling methods). (2) It makes use of the full trajectory instead of truncating or diminishing future steps when off-policy actions are encountered [Watkins, 1989, Precup et al., 2000]. We note, however, that applying this method in practice can be non-trivial due to the constraint $\sum_{s'\in\mathcal{S}} \hat{B}(s,a,s')p(s'|s,a) = 0$. This constraint is equivalent to the $\hat{A}$ constraint in DAE, in the sense that they both ensure the functions satisfy the centering property of the advantage function (i.e., $\mathbb{E}_{a\sim\pi}[A^\pi(s,a)|s] = 0$). Below, we briefly discuss how to deal with this.

Approximating the constraint As a first step, we note that a similar constraint $\sum_{a\in\mathcal{A}} \hat{A}(s,a)\pi(a|s) = 0$ can be enforced through the following parametrization $\hat{A}_\theta(s,a) = f_\theta(s,a) - \sum_{a\in\mathcal{A}} f_\theta(s,a)\pi(a|s)$, where f_θ is the underlying function approximator [Wang et al., 2016b]. Unfortunately, this technique cannot be applied directly to the $\hat{B}$ constraint, because (1) it requires a sum over the state space, which is typically too large, and (2) the transition function $p(s'|s,a)$ is usually unknown. To overcome these difficulties, we use a Conditional Variational Auto-Encoder (CVAE) [Kingma and Welling, 2013, Sohn et al., 2015] to encode transitions into a discrete latent space $\mathcal{Z}$ such that the sum can be efficiently approximated, see Figure 3. The CVAE consists of three components: (1) an approximated conditional posterior $q_{\tilde{\phi}}(z|s,a,s')$ (encoder), (2) a conditional likelihood $p_{\tilde{\theta}}(s'|s,a,z)$ (decoder), and (3) a conditional prior $p_{\tilde{\theta}}(z|s,a)$. These components can then be learned jointly by maximizing the conditional evidence lower bound (ELBO),

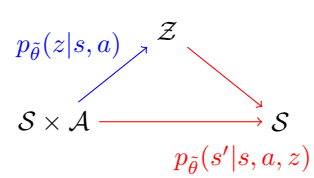

Figure 3: Graphical model of the CVAE; $\mathcal{Z}$ is a discrete latent space.

$$\text{ELBO} = -D_{\text{KL}}(q_{\tilde{\phi}}(z|s,a,s')||p_{\tilde{\theta}}(z|s,a)) + \mathop{\mathbb{E}}_{z\sim q_{\tilde{\phi}}(\cdot|s,a,s')}[\log p_{\tilde{\theta}}(s'|s,a,z)]. \tag{13}$$

Once a CVAE is learned, we can construct $B(s,a,s')$ from an arbitrary function $g_\theta(s,a,z)$ by $B(s,a,s') = \mathbb{E}_{z\sim q_{\tilde{\phi}}(\cdot|s,a,s')}[g_\theta(s,a,z)|s,a,s'] - \mathbb{E}_{z\sim p_{\tilde{\theta}}(\cdot|s,a)}[g_\theta(s,a,z)|s,a]$, which has the property that $\sum_{s'} p(s'|s,a)B(s,a,s') \approx 0$ because $q_{\tilde{\phi}}(z|s,a,s') \approx p_{\tilde{\theta}}(z|s,a,s')$.

5 Connection to the Uncorrected Method

The uncorrected method (simply "Uncorrected" in the following) updates its value estimates using the multi-step target $\sum_{t=0}^{n} \gamma^t r_t + \gamma^{n+1} V_{\text{target}}(s_{n+1})$ without any off-policy correction. Hernandez-Garcia and Sutton [2019] showed that Uncorrected can achieve performance competitive with true off-policy methods in deep RL. The authors also noted that the performance of Uncorrected may be problem-specific, although no criteria were given. Here, we examine how Off-policy DAE, DAE, and Uncorrected relate to each other, and give a possible explanation for when Uncorrected can be used.

We first rewrite the square error of Off-policy DAE (Equation 20) into the following form (for ease of comparison, we use a value target here):

$$\Big(\hat{V}(s_0) - \underbrace{\big(\underbrace{\sum_{t=0}^{n} \gamma^t r_t + \gamma^{n+1} V_{\text{target}}(s_{n+1})}_{\text{Uncorrected}} - \sum_{t=0}^{n} \gamma^t \hat{A}_t - \sum_{t=0}^{n} \gamma^{t+1} \hat{B}_t\big)}_{\text{DAE}}\Big)^2, \tag{14}$$

where the underbraces indicate the updating targets of the respective method. We can see now there is a clear hierarchy between these methods, where DAE is a special case of Off-policy DAE by assuming $\hat{B} \equiv 0$, and Uncorrected is a special case by assuming both $\hat{A} \equiv 0$ and $\hat{B} \equiv 0$.

The question is, then, when is $\hat{A} \equiv 0$ or $\hat{B} \equiv 0$ a good assumption? Remember that, in deterministic environments, we have $B^\pi \equiv 0$ for any policy π; therefore, $\hat{B} \equiv 0$ is a correct estimate of B^π in this case. This means that DAE can be directly applied to off-policy data when the environment is deterministic. Next, to see when $\hat{A} \equiv 0$ is useful, remember that the advantage function can be interpreted as the causal effect of an action on the return. In other words, if actions in the environment tend to have minuscule impacts on the return, then Uncorrected can work with a carefully chosen backup length. This can partially explain why Uncorrected worked in environments like Atari games [Bellemare et al., 2013, Gruslys et al., 2017, Hessel et al., 2018] for small backup lengths, because the actions are fine-grained and have small impact ($A \approx 0$) in general.

6 Experiments

We now compare (1) Uncorrected, (2) DAE, (3) Off-policy DAE, and (4) Tree Backup [Precup et al., 2000] in terms of policy optimization performance using a simple off-policy actor-critic algorithm. By comparing (1), (2), and (3), we test the importance of $\hat{A}$ and $\hat{B}$ as discussed in Section 5. Method (4) serves as a baseline of true off-policy method, and Tree Backup was chosen because, like Off-policy DAE, it also assumes no knowledge of the behavior policy, in contrast to importance sampling methods. When comparing these methods, only the critic loss is adjusted based on the given method to minimize potential confounding.

Environment We perform our experiments using the MinAtar suite [Young and Tian, 2019]. The MinAtar suite consists of 5 environments that replicate the dynamics of a subset of environments from the Arcade Learning Environment (ALE) [Bellemare et al., 2013] with simplified state/action spaces. The MinAtar environments have several properties that are desirable for our study: (1) Actions tend to have significant consequences due to the coarse discretization of its state/action spaces. This suggests that ignoring other actions' effects ($\hat{A}$), as done in Uncorrected, may have a larger impact on its performance. (2) The MinAtar suite includes both deterministic and stochastic environments, which allows us to probe the importance of $\hat{B}$.

Agent Design Since (Off-policy) DAE's loss function depends heavily on the target policy, we found that having a smoothly changing target policy during training is critical, especially when the backup length is long. Preliminary experiments indicated that using the greedy policy, i.e.,

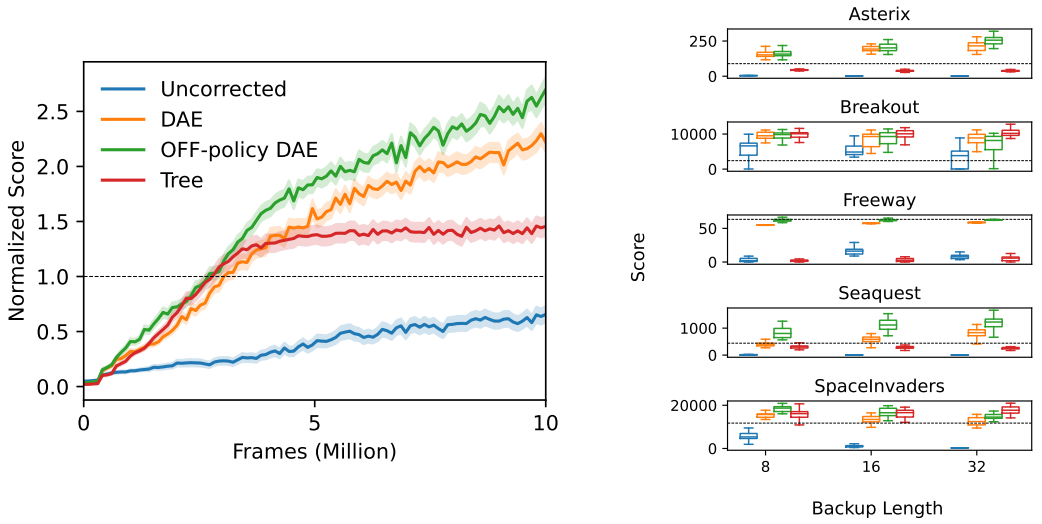

Figure 4: Left: Normalized training curves. Scores were first normalized using the PPO-DAE baseline and then aggregated over all random seeds, environments, and backup lengths. Lines and shadings represent the means and 1 standard error of the means, respectively. Right: Distributions of the evaluation scores obtained by each method with different backup lengths. The dotted horizontal lines represent the scores obtained by the PPO-DAE baseline.

$\arg\max_{a\in\mathcal{A}} \hat{A}(s,a)$, as the target policy to optimize the DAE loss can lead to divergence, which is likely due to the phenomenon of policy churning [Schaul et al., 2022]. To mitigate this problem, we distill a policy by maximizing $\mathbb{E}_{a\sim\pi_\theta}[\hat{A}(s,a)]$, where $\hat{A}$ is the estimated advantage function, and smooth it using exponential moving average (EMA). The smoothed policy π_{EMA} is then used as the target policy for the critic loss. Additionally, to avoid premature convergence, we include a KL-divergence penalty between π_θ and π_{EMA}, similar to trust-region methods [Schulman et al., 2015a]. For critic training, we also use an EMA of past value functions as the bootstrapping target. While in theory, Off-policy DAE does not require a target network for bootstrapping, we found that using one is much more stable in practice. In the following experiments, we use target networks for all methods.

For Off-policy DAE, we additionally learn a CVAE model of the environment. Since learning the dynamics of the environment may improve sample efficiency by learning a better representation [Gelada et al., 2019, Schwarzer et al., 2020], we isolate this effect by training a separate network for the CVAE and the agent can only query $p(z|s,a,s')$ (conditional posterior) and $p(z|s,a)$ (conditional prior) from the model, which are necessary to approximate the constraint. See Appendix D for detailed descriptions of the hyperparameters, network architectures and the pseudocode of the algorithm.

Evaluation Each agent is trained for 10 million frames, and evaluated by averaging the undiscounted scores of 100 rollout episodes using the trained policy. We also vary the backup length $n = \{8, 16, 32\}$, to test its effects on the performance of different backup methods.

Results For comparison, we use the scores reported by Pan et al. [2022] as on-policy baselines, which were trained using PPO and DAE (denoted PPO-DAE here). The results are summarized in Figure 4. See Appendix D for numerical results and learning curves for individual environments. We make the following observations: (1) From the training curves, we see a clear hierarchy between Uncorrected→DAE→Off-policy DAE, suggesting that correcting for $\hat{A}$ and $\hat{B}$ are both beneficial. (2) For deterministic environments (Breakout & Space Invaders), DAE and Off-policy DAE performed on par with Tree Backup, suggesting that they can compete with true off-policy methods. Interestingly, we see Off-policy DAE performing slightly better than DAE in Space Invaders, which is likely because the environment is partially observable, and modelling the transitions as stochastic in this case can be helpful. Uncorrected, on the other hand, tends to perform worse than other methods, and the performance deteriorates when the backup length increases. This suggests that including

$\hat{A}$ in the loss can have a major impact on performance. (3) For stochastic environments (Asterix, Freeway & Seaquest), we see Off-policy DAE outperforming DAE in Freeway and Seaquest, and on par in Asterix. This suggests that ignoring $\hat{B}$ can lead to suboptimal performance. Surprisingly, Tree Backup underperforms the PPO-DAE baselines in these environments, suggesting that stochasticity may have an adverse effect in this case. Uncorrected, on the other hand, significantly underperforms other methods, except for Freeway, where it did marginally better than Tree Backup, showing the importance of off-policy corrections.

7 Related Work

Advantage Function The advantage function was originally proposed by Baird [1994] to extend RL to continuous time or small time-step domains. Later, it was shown that the advantage function can be used to relate value functions of different policies [Kakade and Langford, 2002] or reduce the variance of policy gradient methods [Greensmith et al., 2004]. These properties led to wide adoption of the advantage function in modern policy optimization methods [Schulman et al., 2015a,b, 2017, Mnih et al., 2016]. More recently, the connection between causal effects and the advantage function was pointed out by Corcoll and Vicente [2020] and further strengthened by Pan et al. [2022], proposing DAE to estimate the advantage function in on-policy settings. Our work builds on DAE by first showing that it can be understood as a generalization of MC methods, and by demonstrating how it can be extended to off-policy settings.

Multi-step Learning Multi-step methods [Watkins, 1989, Sutton, 1988] have been widely adopted in recent deep RL research and shown to have a strong effect on performance [Schulman et al., 2015b, Hessel et al., 2018, Wang et al., 2016a, Gruslys et al., 2017, Espeholt et al., 2018, Hernandez-Garcia and Sutton, 2019]. Typical off-policy multi-step methods include importance sampling [Munos et al., 2016, Rowland et al., 2020, Precup et al., 2001], truncating (diminishing) off-policy actions [Watkins, 1989, Precup et al., 2000], a combination of the two [De Asis et al., 2018], or simply ignoring any correction. Our approach, on the other hand, does not require importance ratios and is capable of utilizing the full trajectory, at the cost of having to learn the transition probabilities when the environment is not deterministic.

Afterstates The idea of dissecting transitions into a two-step process dates at least back to Sutton et al. [1998], where afterstates (equivalent to nature's states in Figure 2) were introduced. It was shown that learning the values of afterstates can be easier in some specialized problems. In the present work, in contrast, we focus on the *effect* of possible outcomes from given afterstates rather than on their values. Our use of CVAE is inspired by Stochastic MuZero [Antonoglou et al., 2021], which combined the idea of afterstates and discrete latent representations [van den Oord et al., 2017] to learn stochastic models with strong planning capabilities.

Luck Mesnard et al. [2021] proposed to use future-conditional value functions to capture the effect of luck, and demonstrated that these functions can be used as baselines in policy gradient methods to reduce variance. However, a clear definition of luck remained elusive. In this work, we approached this problem from a causal effect perspective and provide a quantitative definition of luck (see Equation 11).

8 Discussion

In the present work, we drew connections between MC methods and DAE, and extended DAE to off-policy settings. This was achieved by considering advantage functions not only from the decision-making agent, but also the stochastic transitions from the environment.Through experiments in both stochastic and deterministic environments, we verified that the off-policy correction is crucial for policy optimization when effects from actions or transitions are non-negligible.

Finally, we note a possible extension. In our work, we found that extending a single-agent problem into a two-agent problem allows us to fully characterize the return. One interesting question is whether a similar approach can be used in multi-agent RL problems [Littman, 1994].

References

I. Antonoglou, J. Schrittwieser, S. Ozair, T. K. Hubert, and D. Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2021.

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

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

O. Corcoll and R. Vicente. Disentangling causal effects for hierarchical reinforcement learning. *arXiv preprint arXiv:2010.01351*, 2020.

K. De Asis, J. Hernandez-Garcia, G. Holland, and R. Sutton. Multi-step reinforcement learning: A unifying algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2020.

C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

A. Gruslys, W. Dabney, M. G. Azar, B. Piot, M. Bellemare, and R. Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. *arXiv preprint arXiv:1704.04651*, 2017.

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

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

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

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

T. Mesnard, T. Weber, F. Viola, S. Thakoor, A. Saade, A. Harutyunyan, W. Dabney, T. S. Stepleton, N. Heess, A. Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning*, pages 7654–7664. PMLR, 2021.

M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

H.-R. Pan, N. Gürtler, A. Neitz, and B. Schölkopf. Direct advantage estimation. *Advances in Neural Information Processing Systems*, 35:11869–11880, 2022.

D. Precup, R. S. Sutton, and S. P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 759–766, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

D. Precup, R. S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

M. Rowland, W. Dabney, and R. Munos. Adaptive trade-offs in off-policy learning. In *International Conference on Artificial Intelligence and Statistics*, pages 34–44. PMLR, 2020.

T. Schaul, A. Barreto, J. Quan, and G. Ostrovski. The phenomenon of policy churn. *arXiv preprint arXiv:2206.00730*, 2022.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015a.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

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

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1): 9–44, 1988.

R. S. Sutton, A. G. Barto, et al. Introduction to reinforcement learning. 1998.

C. Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *NIPS*, 2017.

Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016a.

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016b.

C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, 1989.

K. Young and T. Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

A Details of Figure 1

In this example, we compare the sample efficiency of MC, Batch TD(0) and DAE. We note that there are only 4 possible trajectories in this environment, depending on the starting state (1 or 2) and the action chosen at state 3 ($\mathtt{u}$ or $\mathtt{d}$). We denote the number of trajectories starting from i and choosing action $a \in \{\mathtt{u}, \mathtt{d}\}$ by $n_{i,a}$, and $n_i = n_{i,\mathtt{u}} + n_{i,\mathtt{d}}$. The trajectories were sampled using the uniform policy, i.e., $\pi(\mathtt{u}|3) = \pi(\mathtt{d}|3) = 0.5$. In the following list, we summarize the estimates from each method.

- MC: $\hat{V}(1) = \frac{n_{1,\mathtt{u}}}{n_1}, \quad \hat{V}(2) = \frac{n_{2,\mathtt{u}}}{n_2}$
- Batch TD(0): $\hat{V}(1) = \hat{V}(2) = \hat{V}(3) = \frac{n_{1,\mathtt{u}}+n_{2,\mathtt{u}}}{n_1+n_2}$
- DAE: The minimizer of

$$L(\hat{V}(1), \hat{V}(2), \hat{A}(3,\mathtt{u}), \hat{A}(3,\mathtt{d})) = n_{1,\mathtt{u}}(1 - \hat{A}(3,\mathtt{u}) - \hat{V}(1))^2 + n_{1,\mathtt{d}}(0 - \hat{A}(3,\mathtt{d}) - \hat{V}(1))^2 + n_{2,\mathtt{u}}(1 - \hat{A}(3,\mathtt{u}) - \hat{V}(2))^2 + n_{2,\mathtt{d}}(0 - \hat{A}(3,\mathtt{d}) - \hat{V}(2))^2$$

  subject to $\hat{A}(3,\mathtt{u}) + \hat{A}(3,\mathtt{d}) = 0$ (since the sampling policy is uniform).

One can use the method of Lagrange multiplier to solve the DAE problem and arrive at the following linear equations:

$$\begin{pmatrix} n_{1,u} - n_{1,d} & n_1 & 0 \\ n_{2,u} - n_{2,d} & 0 & n_2 \\ (n_{1,u} + n_{2,u}) & n_{1,u} & n_{2,u} \end{pmatrix} \begin{pmatrix} \hat{A}(3,\mathtt{u}) \\ \hat{V}(1) \\ \hat{V}(2) \end{pmatrix} = \begin{pmatrix} n_{1,u} \\ n_{2,u} \\ n_{1,u} + n_{2,u} \end{pmatrix}, \tag{15}$$

Note that there are only 3 equations, since $\hat{A}(3,\mathtt{u}) + \hat{A}(3,\mathtt{d}) = 0$. Additionally, the solution is unique only when both $n_1 > 0$ and $n_2 > 0$, otherwise the first row or the second row of the matrix would be 0. For simplicity, we use the pseudoinverse to compute the solution:

$$\begin{pmatrix} \hat{A}(3,\mathtt{u}) \\ \hat{V}(1) \\ \hat{V}(2) \end{pmatrix} = \begin{pmatrix} n_{1,u} - n_{1,d} & n_1 & 0 \\ n_{2,u} - n_{2,d} & 0 & n_2 \\ (n_{1,u} + n_{2,u}) & n_{1,u} & n_{2,u} \end{pmatrix}^{+} \begin{pmatrix} n_{1,u} \\ n_{2,u} \\ n_{1,u} + n_{2,u} \end{pmatrix}, \tag{16}$$

where $+$ denotes the pseudoinverse. This explains why the DAE estimates in Figure 1 are slightly skewed towards 0 at the beginning.

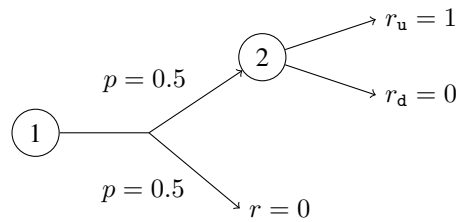

Figure 5: A simple MDP with $\mathcal{S} = \{1, 2\}$. State 1 has only a single action, which can lead to state 2 or the terminal state with equal probabilities. State 2 has two actions u and d with rewards 1 and 0, respectively, and both actions lead to the terminal state.

B Counterexample

In this section, we construct an example to demonstrate that naively applying DAE to off-policy data can lead to biased results. Consider the environment in Figure 5. Suppose the data is collected with a behavior policy μ and we wish to estimate values for a target policy π. If we apply DAE directly to this problem without any off-policy correction, then the loss is equal to

$$L(\hat{A}, \hat{V}) = \frac{1}{2}\mu(\mathtt{u}|2)\left(1 - \hat{A}(2, \mathtt{u}) - \hat{V}(1)\right)^2 + \frac{1}{2}\mu(\mathtt{d}|2)\left(0 - \hat{A}(2, \mathtt{d}) - \hat{V}(1)\right)^2 + \frac{1}{2}(0 - \hat{V}(1))^2, \tag{17}$$

since there are only three possible trajectories in this environment. Now, if we include the constraint that $\sum_a \hat{A}(2, a)\pi(a|2) = 0$, then the problem can be solved by the method of Lagrange multiplier using the following Lagrangian:

$$L(\hat{A}, \hat{V}) + \lambda \sum_a \hat{A}(2, a)\pi(a|2). \tag{18}$$

The minimizer $(A^*, V^*) = \arg\min L(\hat{A}, \hat{V})$ is given by:

$$\begin{cases} V^*(1) = \frac{\pi(\mathtt{u}|2)}{1 + \frac{\pi(\mathtt{u}|2)^2}{\mu(\mathtt{u}|2)} + \frac{\pi(\mathtt{d}|2)^2}{\mu(\mathtt{d}|2)}} \\ \lambda = V^*(1) \\ A^*(2, \mathtt{u}) = 1 - V^*(1) - \frac{V^*(1)\pi(\mathtt{u}|2)}{\mu(\mathtt{u}|2)} \\ A^*(2, \mathtt{d}) = 0 - V^*(1) - \frac{V^*(1)\pi(\mathtt{d}|2)}{\mu(\mathtt{d}|2)} \end{cases} \tag{19}$$

meaning that $V^*(1) \neq V^\pi(1) = \frac{\pi(\mathtt{u}|2)}{2}$, and $A^* \neq A^\pi$ if $\pi \neq \mu$. One can also verify that, if $\pi = \mu$ (on-policy), then $(V^*, A^*) = (V^\pi, A^\pi)$.

C Proof of Theorem 1

Theorem (Off-policy DAE)**.** Given a behavior policy μ and a target policy π and backup length $n \geq 0$. Let $\hat{A}_t = \hat{A}(s_t, a_t)$, $\hat{B}_t = \hat{B}(s_t, a_t, s_{t+1})$, and the constrained squared error

$$L(\hat{A}, \hat{B}, \hat{V}) = \mathop{\mathbb{E}}_{\mu}\left[\left(\sum_{t=0}^{n}\gamma^t\left(r_t - \hat{A}_t - \gamma\hat{B}_t\right) + \gamma^{n+1}\hat{V}(s_{n+1}) - \hat{V}(s_0)\right)^2\right] \tag{20}$$
$$\text{subject to}\begin{cases}\sum_{a\in\mathcal{A}}\hat{A}(s,a)\pi(a|s) = 0 \quad \forall s \in \mathcal{S} \\ \sum_{s'\in\mathcal{S}}\hat{B}(s,a,s')p(s'|s,a) = 0 \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}\end{cases},$$

then (A^π, B^π, V^π) is a minimizer of the above problem. Furthermore, the minimizer is unique if μ is sufficiently explorative (i.e., non-zero probability of reaching all possible transitions (s, a, s')).

Proof. Since

$$0 \leq L(A^\pi, B^\pi, V^\pi) = \mathop{\mathbb{E}}_{\mu}\left[\left(\sum_{t=0}^{n}\gamma^t(r_t - A_t^\pi - \gamma B_t^\pi) + \gamma^{n+1}V^\pi(s_{n+1}) - V^\pi(s_0)\right)^2\right] = 0, \tag{21}$$

and both $\sum_{a\in\mathcal{A}}\pi(a|s)A^\pi(s,a) = 0$ and $\sum_{s'\in\mathcal{S}}p(s'|s,a)B^\pi(s,a,s') = 0$ constraints are satisfied, (A^π, B^π, V^π) is a minimizer of the problem. For uniqueness, we assume the behavior policy is sufficiently explorative such that any sequence $(s_0, a_0, r_0, ...s_{n+1})$ has non-zero probability of being visited. Now, suppose there exists (A', B', V') that also minimizes L, i.e., $L(A', B', V') = 0$, then for any sequence $(s_0, a_0, ...s_{t+1})$, we must have

$$\sum_{t=0}^{n}\gamma^t(r_t - A_t' - \gamma B_t') + \gamma^{n+1}V'(s_{n+1}) - V'(s_0) = 0, \tag{22}$$

otherwise $L(A', B', V') \neq 0$. If we take the conditional expectation over $(a_0, s_1, a_1, ...s_{n+1})$ conditioned on s_0 using the target policy π, then

$$V'(s_0) = \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{n}\gamma^t(r_t - A_t' - \gamma B_t') + \gamma^{n+1}V'(s_{n+1})|s_0] \tag{23}$$
$$= \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{n}\gamma^t r_t + \gamma^{n+1}V'(s_{n+1})|s_0], \tag{24}$$

which means that V' satisfies the Bellman Equation. Therefore $V' = V^\pi$ uniquely. Similarly, if we take the expectation over $(s_1, a_1, ...s_{n+1})$ conditioned on s_0, a_0, then

$$A'(s_0, a_0) = r(s_0, a_0) + \mathop{\mathbb{E}}_{\pi}[\sum_{t=1}^{n}\gamma^t r_t + \gamma^{n+1}V^\pi(s_{n+1})|s_0, a_0] - V^\pi(s_0) = A^\pi(s_0, a_0) \tag{25}$$

Finally, if we take the expectation over $(a_1, s_2, ..., s_{t+1})$ conditioned on s_0, a_0, s_1, then

$$\gamma B'(s_0, a_0, s_1) = r(s_0, a_0) - A^\pi(s_0, a_0) + \mathop{\mathbb{E}}_{\pi}[\sum_{t=1}^{n}\gamma^{t'}r_{t'} + \gamma^{t+1}V^\pi(s_{t+1})|s_0, a_0, s_1] - V^\pi(s_0) \tag{26}$$
$$= \gamma(V^\pi(s_1) - \mathbb{E}[V^\pi(s_1)|s_0, a_0]) = \gamma B^\pi(s_0, a_0, s_1). \tag{27}$$

Similarly, we get $(A', B', V') = (A^\pi, B^\pi, V^\pi)$ for all $(s_{t'}, a_{t'}, s_{t'+1})$ with $0 \leq t' \leq n$ by taking the expectation conditioned on $s_0, a_0, s_1, ..., s_{t'-1}$ in the sequence. By the assumption that μ has non-zero probability of visiting any sequence, we have $(A', B', V') = (A^\pi, B^\pi, V^\pi)$ for all $(s, a, s') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}$. □

Remarks: (1) While we used the squared error as the objective function in the theorem, it can be replaced with an arbitrary metric in $\mathbb{R}$, as the proof does not rely on properties of the squared error. (2) For uniqueness, the condition on the behavior policy μ can be relaxed if we only care about states/actions covered by the target policy π. In that case, we only need the coverage of μ to include the coverage of π.

D Details of the MinAtar Experiments

D.1 Pseudocode

The pseudocode of the proposed actor-critic method is provided in Algorithm 1. Here we note some details about the training process. Unlike typical methods that store 1-step transitions in the replay buffer, our buffer consists of n-step trajectories. When computing the critic loss, we also compute the loss for each sub-trajectory as in Pan et al. [2022]. For example, if $(s_0, a_0, ..., s_n)$ is a sample trajectory, we accumulate the critic loss for all sub-trajectory $(s_i, a_i, ..., s_n)$ for all $i \in \{0, 1, 2, ..., n-1\}$. Also, to speed up training, we use parallel actors to sample transitions from the environments.

Algorithm 1 A simple Off-policy Actor-Critic Algorithm

Require: `backup` $\in$ {Uncorrected, DAE, Off-policy DAE}
Require: n (backup length)
Initialize actor-critic components $A_\theta(s,a)$, $V_\theta(s)$, $B_\theta(s,a,z)$, $\pi_\theta(a|s)$
Initialize CVAE $q_{\tilde{\phi}}(z|s,a,s')$, $p_{\tilde{\theta}}(z|s,a)$, $p_{\tilde{\theta}}(s'|s,a,z)$
$\theta_{\text{EMA}} \leftarrow \theta$
$D = \{\}$
$D_n = \{\}$
for $t = 0, 1, 2, \ldots$ **do**
  Sample transition (s,a,r,s') with policy π_θ
  $D_n \leftarrow D_n \cup \{(s,a,r,s')\}$
  if s' is terminal **or** $|D_{\text{n-step}}| = n$ **then**
    $D \leftarrow D \cup \{\texttt{concatenate}(D_n)\}$
    $D_n \leftarrow \{\}$
  end if
  if $t+1$ mod `steps_per_update` = 0 **then**
    Sample batch of trajectories `batch` from D
    if backup = Off-policy DAE **then**
      Train $q_{\tilde{\phi}}(z|s,a,s')$, $p_{\tilde{\theta}}(z|s,a)$, $p_{\tilde{\theta}}(s'|s,a,z)$ using `batch` by Equation 13
      $B_\theta(s,a,s') \leftarrow \mathbb{E}_{z\sim q_{\tilde{\phi}}(\cdot|s,a,s')}[B_\theta(s,a,z)|s,a,s'] - \mathbb{E}_{z\sim p_{\tilde{\theta}}(\cdot|s,a)}[B_\theta(s,a,z)|s,a]$
    end if
    $A_\theta(s,a) \leftarrow A_\theta(s,a) - \mathbb{E}_{a\sim\pi_{\theta_{\text{EMA}}}}[A_\theta(s,a)]$
    Compute critic loss L_{critic} according to `backup` with A_θ, V_θ
    $A_{\text{normalized}} \leftarrow \texttt{stop_gradient}(A_\theta(s,a)/\sqrt{\mathbf{Var}[A_\theta]})$
    Compute actor loss $L_{\text{actor}} = -\mathbb{E}_{a\sim\pi_\theta}[A_{\text{normalized}}] + \beta_{\text{KL}} D_{\text{KL}}(\pi_\theta || \pi_{\theta_{\text{EMA}}})$
    Train $L_{\text{critic}} + L_{\text{actor}}$ with Adam [Kingma and Ba, 2014]
    $\theta_{\text{EMA}} \leftarrow \tau\theta_{\text{EMA}} + (1-\tau)\theta$
  end if
end for

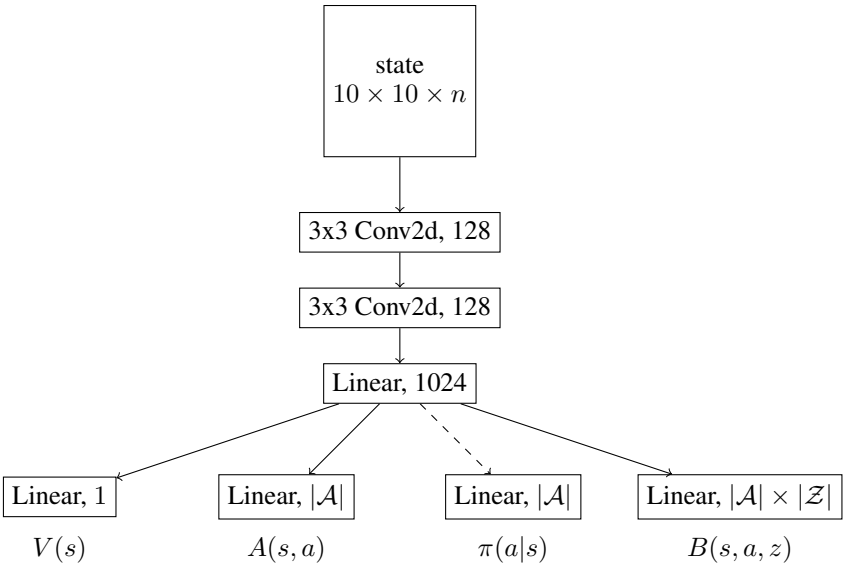

Figure 6: Network architecture for the actor-critic. Each layer is followed by a ReLU activation function, except for the output layer. The dashed line indicates that gradients are stopped.

D.2 Actor-Critic Network

In our experiments, we use a convolutional neural network followed by multiple heads to approximate $A_\theta, B_\theta, V_\theta$, and π_θ (see Figure 6)[Mnih et al., 2016, Wang et al., 2016b]. Since we train both the actor and the critic using a single network simultaneously, to avoid interference between the two losses (L_{critic} and L_{actor}), we simply use the representation learned from by the critic to train the actor by stopping the gradients from L_{actor} to the shared network. This eliminates the need to balance for the different loss functions.

D.3 Conditional Variational AutoEncoder (CVAE) Network

We illustrate the training process and the network architecture in Figure 7. First, we use a discrete latent space $\mathcal{Z}$ which allows us to compute expectations over $\mathcal{Z}$ efficiently. This also alleviates the need to use heuristics such as VQ-VAE [van den Oord et al., 2017] or Gumbel-Softmax [Jang et al., 2016, Maddison et al., 2016], because we can now compute $\mathbb{E}_{z\sim q_{\tilde{\phi}}(\cdot|s,a,s')}[\log p_{\tilde{\theta}}(s'|s,a,z)] = \sum_{z\in\mathcal{Z}} q_{\tilde{\phi}}(z|s,a,s') \log p_{\tilde{\theta}}(s'|s,a,z)$ exactly. Second, to eliminate the need to balance between KL-divergence loss and the reconstruction loss, the conditional prior is trained using the representation from the encoder with gradients stopped. This is similar to the approach in VQ-VAE where a prior is trained separately. Finally, we observed that the posterior can sometimes collapse early in training. To mitigate this, we add a small entropy penalty for the posterior. Combining everything together, we have the loss function for CVAE:

$$\mathcal{L}_{\text{CVAE}}(\tilde{\theta}, \tilde{\phi}; s, a, s') = D_{\text{KL}}(q_{\tilde{\phi}}(z|s,a,s')||p_{\tilde{\theta}}(z|s,a)) - \mathop{\mathbb{E}}_{z\sim q_{\tilde{\phi}}(\cdot|s,a,s')}[\log p_{\tilde{\theta}}(s'|s,a,z)] - \beta_{\text{ent}} H(q_{\tilde{\phi}}(\cdot|s,a,s'))$$

where $H(\cdot)$ is the entropy, and β_{ent} controls the strength of the entropy penalty.

D.4 Hyperparameters

In Table 1, we summarize the hyperparameters used in the MinAtar experiments. In general, the agent was designed to have very few hyperparameters to reduce potential confounding when comparing different backup methods. Our preliminary experiments found that the effects of the hyperparameters to be agnostic to backup methods, except for τ which tend to have a larger impact on Off-policy DAE and DAE, which is likely due to the heavy dependence on the policy when training with DAE-like loss functions.

For CVAE training, we found the Adam β to have a huge impact on training stability, and using the default $\beta = (0.9, 0.999)$ often leads to divergence. Instead, we use the parameters provided by Esser et al. [2020] and found them to be very effective.

Table 1: List of hyperparameters. Note that for Off-policy DAE, there are two separate optimizers used for actor-critic and CVAE training. †: not used in Pan et al. [2022].

Group	Parameter	Value
Environment setting	Sticky Action	False
	Difficulty Ramping	False
	Maximum Episode Length†	108000 frames
Shared (actor-critic training)	Discount γ	0.99
	Parallel actors	128
	Initial steps before training	25000 frames
	Replay Buffer Size	1000000 frames
	Backup Length	8/16/32
	Optimizer	Adam[Kingma and Ba, 2014]
	Learning rate	0.00025 (linearly annealed to 0)
	Adam β	(0.9, 0.999)
	Adam ϵ	10^{-4}
	Env. steps per update	32
	Batch Size	1024 frames
	β_{KL}	3.0
	τ (EMA weight)	0.999
Off-policy DAE only (CVAE training)	Latent size $\lvert\mathcal{Z}\rvert$	16
	Optimizer	Adam
	Learning rate	0.00025
	Adam β	(0.5, 0.9)
	Adam ϵ	10^{-8}
	β_{ent}	0.0001

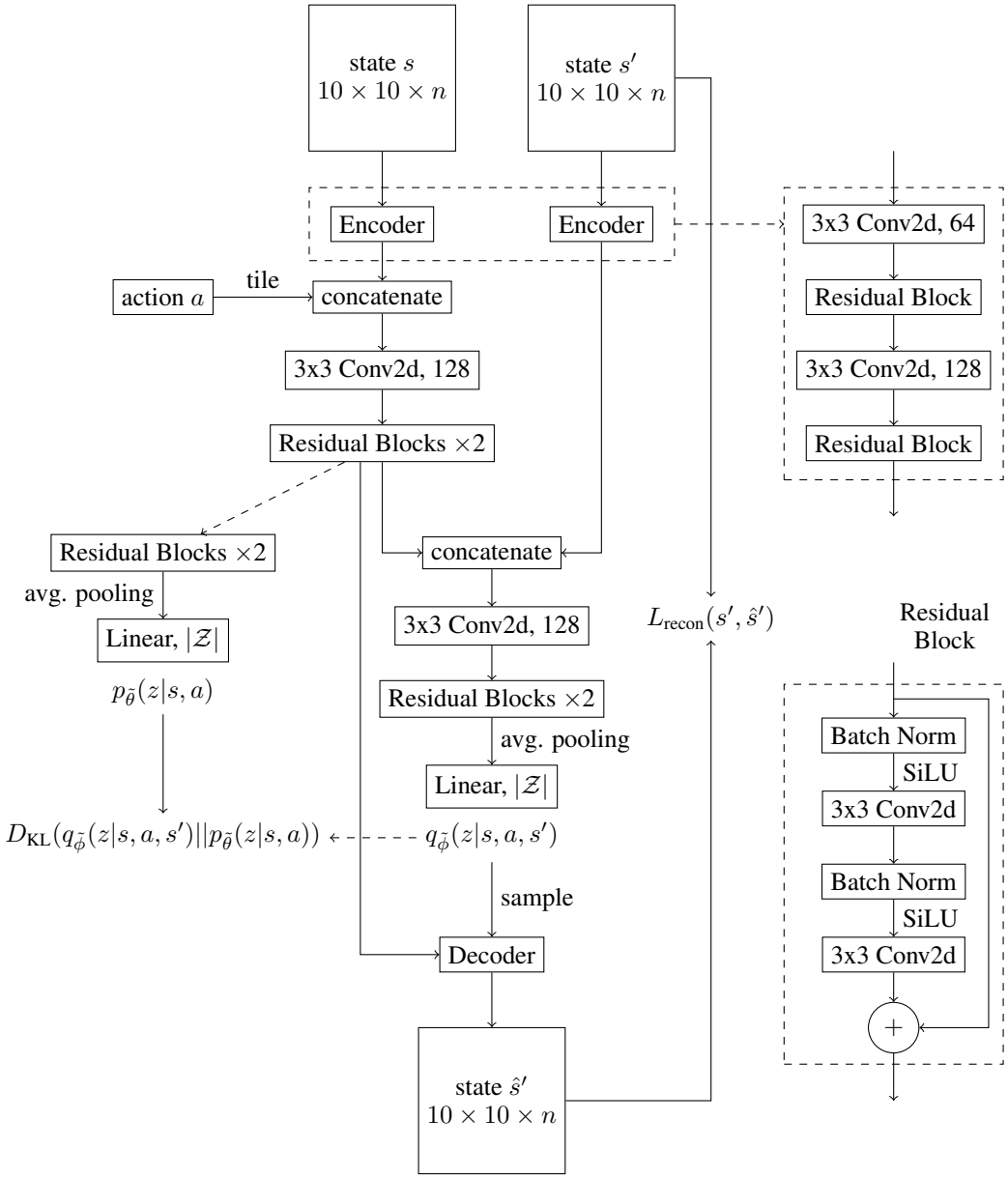

Figure 7: Network architecture and the training graph for the CVAE. Dashed lines indicate that gradients are stopped. The Decoder uses the same architecture as the encoder with orders reversed and convolutions replaced with transposed convolutions. Softmax is applied to both $q_{\tilde{\phi}}(z|s,a,s')$ and $p_{\tilde{\theta}}(z|s,a))$ to ensure they are probability distributions over $\mathcal{Z}$ (note the Z is a discrete space in this case). We use binary cross entropy for the reconstruction loss, since states in the MinAtar environments are binary images.

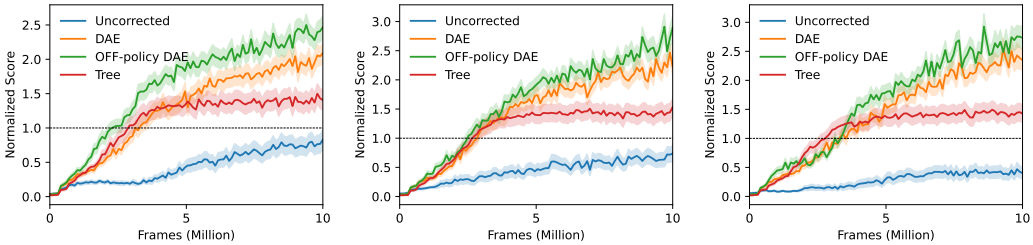

Figure 8: Normalized training curves for each backup method and backup length (from left to right: 8, 16, 32). The dashed horizontal line represents the PPO-DAE baseline.

D.5 More Results

In the MinAtar experiments, normalized score were calculated by $\text{score}_{\text{norm}} = \frac{\text{score}-\text{baseline}}{\text{baseline}}$, where we use the scores reported by Pan et al. [2022] as baselines.

Table 2: Score comparison between different methods and backup lengths. Results were aggregated over 20 random seeds. Numbers represent (mean)±(1 standard error).

Environment	N	Backup Method			
		Uncorrected	DAE	Off-policy DAE	Tree
Asterix	8	4.5 ± 0.2	155.6 ± 5.4	161.5 ± 5.9	44.3 ± 1.1
	16	2.5 ± 0.1	194.2 ± 4.8	207.0 ± 9.2	39.0 ± 1.4
	32	2.0 ± 0.1	215.4 ± 8.1	268.9 ± 14.0	39.0 ± 1.3
Breakout	8	5799.9 ± 611.3	9573.0 ± 250.1	9423.3 ± 351.9	9411.4 ± 505.5
	16	5220.1 ± 448.7	8506.1 ± 476.6	8887.9 ± 429.0	10069.0 ± 288.8
	32	3179.3 ± 661.8	8119.8 ± 617.5	7372.1 ± 582.0	10139.8 ± 338.1
Freeway	8	5.5 ± 1.8	55.1 ± 0.1	61.9 ± 0.6	2.2 ± 0.4
	16	16.8 ± 1.3	57.6 ± 0.1	62.3 ± 0.3	4.1 ± 1.0
	32	8.2 ± 0.8	58.8 ± 0.1	62.9 ± 0.3	5.1 ± 0.8
Seaquest	8	13.5 ± 1.5	413.5 ± 25.1	839.4 ± 48.3	312.3 ± 16.4
	16	4.1 ± 0.1	594.3 ± 36.3	1171.6 ± 66.6	286.4 ± 11.6
	32	4.0 ± 0.1	821.6 ± 52.5	1225.3 ± 63.7	266.2 ± 19.1
SpaceInvaders	8	5561.5 ± 452.3	15116.1 ± 377.4	18086.9 ± 419.1	15615.3 ± 563.0
	16	1180.3 ± 145.0	13478.7 ± 391.2	16756.8 ± 460.4	16009.6 ± 508.1
	32	307.7 ± 32.0	12560.4 ± 441.4	14970.3 ± 348.9	17374.2 ± 584.9

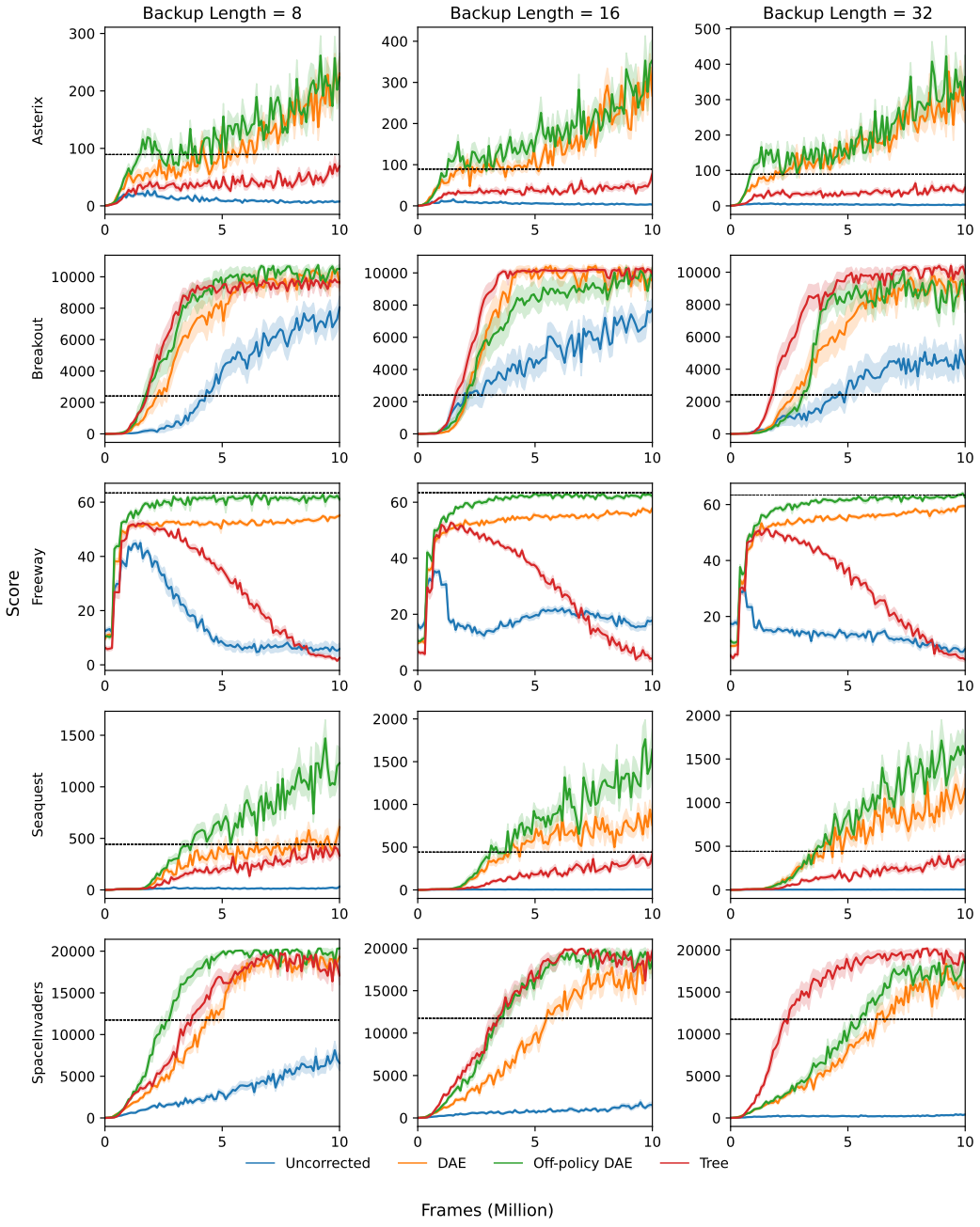

Figure 9: Training curves of each environment.

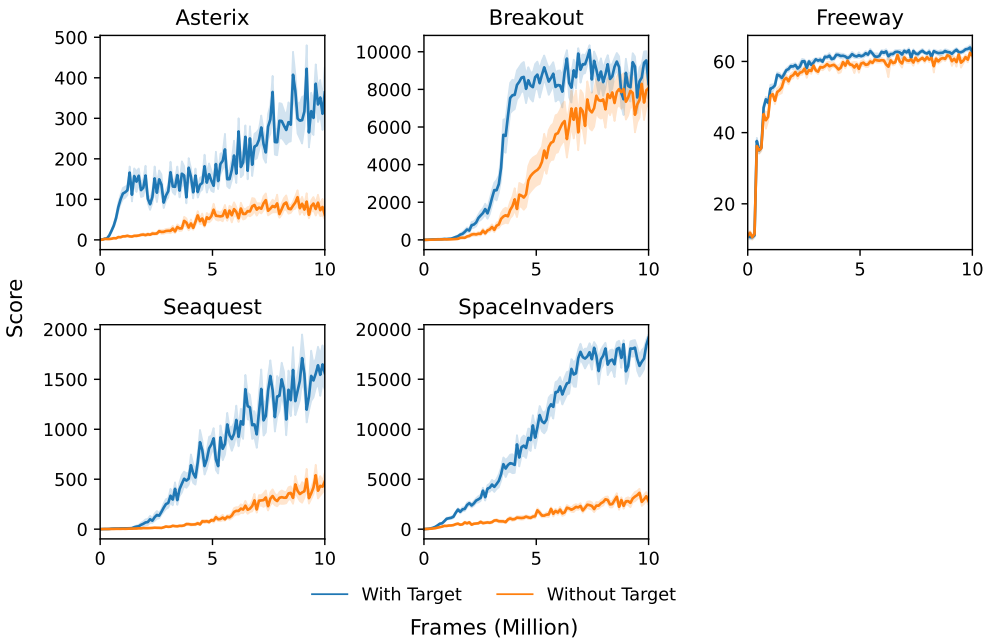

Figure 10: Comparison between Off-policy DAE with and without target network.

D.6 With and Without Target Network

While Equation 20 suggests that a separate target network for critic training is not necessary, in practice, we have found that using a target network leads to better performance. See Figure 10. In general, we found the critic loss to be lower when using target networks. One possible explanation is that using target networks results in biased, but lower variance estimates, which in turn makes the loss easier to optimize. Further investigation is required to understand the tradeoffs.

D.7 Computational Resources

All experiments were performed on an internal cluster of NVIDIA A100 GPUs. Training an agent takes approximately 2 hours, depending on the backup method and the environment. For Off-policy DAE, the training time is significantly longer (approx. 15 hours) due to CVAE training.