BEYOND MARKOV ASSUMPTION: IMPROVING SAMPLE EFFICIENCY IN MDPS BY HISTORICAL AUGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Under the Markov assumption of Markov Decision Processes (MDPs), an optimal stationary policy does not need to consider history and is no worse than any non-stationary or history-dependent policy. Therefore, existing Deep Reinforcement Learning (DRL) algorithms usually model sequential decision-making as an MDP and then try to optimize a stationary policy by single-step state transitions. However, such optimization is often faced with sample inefficiency when the causal relationships of state transitions are complex. To address the above problem, this paper investigates if augmenting the states with their historical information can simplify the complex causal relationships in MDPs and thus improve the sample efficiency for DRL. First, we demonstrate that a complex causal relationship of single-step state transitions may be inferred by a simple causal function of the historically augmented states. Then, we propose a convolutional neural network architecture to learn the representation of the current state and its historical trajectory. The main idea of this representation learning is to compress the high-dimensional historical trajectories into a low-dimensional space. In this way, we can extract the simple causal relationships from historical information and avoid the overfitting caused by high-dimensional data. Finally, we formulate Historical Augmentation Aided Actor-Critic (HA3C) algorithm by adding the learned representations to the actor-critic method. The experiment on standard MDP tasks demonstrates that HA3C outperforms current state-of-the-art methods in terms of both sample efficiency and performance.

1 INTRODUCTION

Sequential decision-making widely exists in real-world control tasks, such as robot control and autonomous driving (Dorf & Bishop, 2011; Ibarz et al., 2021; Sallab et al., 2017). Generally speaking, it can be modelled as a Markov Decision Process (MDP), where an agent iteratively takes action in an environment for transiting from one state to another (Puterman, 1990). Each transition is evaluated by a reward signal passing from the environment to the agent so that Reinforcement Learning (RL) can learn the optimal policy by maximizing the cumulative reward (Sutton & Barto, 2018). The Markov Assumption of MDPs asserts that the probability distributions of the reward and next state depend only on the current state and action. Under the Markov assumption of MDPs, there exists an optimal stationary policy which does not need to consider history and is no worse than any non-stationary or history-dependent policy (Puterman, 2014). Therefore, existing RL algorithms usually try to optimize a stationary policy by single-step state transitions.

With advances in deep learning, many effective Deep RL (DRL) methods were proposed (Fujimoto et al., 2018; Haarnoja et al., 2018; Lillicrap et al., 2016; Mnih et al., 2016; 2015). Under the Markov assumption of MDPs, they are usually based on the actor-critic method where the critic estimates the Q-function, i.e., the expected cumulative reward after taking action at each state, while the actor updates the policy to choose the action which can maximize the estimated Q-function (Schulman et al., 2015; Silver et al., 2014). However, such optimization may miss the useful causal relationships of state transitions, leading to sample inefficiency (Allen et al., 2021; Buckman et al., 2018; Du et al., 2020; Guo et al., 2020). An existing partial solution to this issue is representation learning in which a neural network is trained to infer the causal relationships of state transitions by predicting the reward or future state of each state-action pair (Munk et al., 2016; Ni et al., 2023; Ravindran, 2004; Rezaei-Shoshtari et al., 2022). Then, the sample efficiency of DRL can be improved by adding the learned representations to the actor-critic method. Unfortunately, it is hard to train the neural networks which can infer complex causal relationships, e.g., polynomial causal relationships and the basic laws of physics (Andoni et al., 2014; Cranmer et al., 2020). Standard complexity-theoretic results strongly suggest that there is no algorithm efficient enough for learning arbitrary target functions, even for target functions representable by very low-depth networks (Applebaum et al., 2006). Therefore, the sample efficiency for DRL is still limited in complex MDP tasks.

This paper addresses the above problem by augmenting the states with their historical information. Based on the analysis in Section 3, we believe that historical augmentation can simplify the causal relationships of state transitions by its inherent contextual information and increasing the search space of the causal functions (Hallak et al., 2015; Sprunger & Jacobs, 2019). Therefore, we focus on optimzing a history-dependent stationary policy in an MDP. Our DRL approach comprises two key components: 1) Learning the state representations to capture the causal relationships in an MDP and 2) finding the optimal stationary policy by the learned representations. Given an action and the historically augmented current state, our representation learning utilizes a Convolutional Neural Network (CNN) architecture to compress the high-dimensional historical trajectory of the given state into a low-dimensional space while predicting the future state. The compressed historical trajectories can be seen as the abstracted features which can represent the simple causal relationships and avoid the overfitting caused by high-dimensional data (Andre & Russell, 2002). To keep the Markov assumption of MDPs, our representation learning does not compress the current state. We add the learned state representations to the actor-critic method. In this way, the causal relationships captured by our representation learning can be utilized to estimate the Q-function and update policy. Therefore, our new DRL approach can optimize the policy in a complex MDP with high sample efficiency. We combine historical augmentation, state representations, and TD3 in our approach to formulate a new DRL algorithm, Historical Augmentation Aided Actor-Critic (HA3C). The experiment on standard MDP tasks, i.e. Mujoco control tasks and Deep Mind Control (DMC) suite, empirically demonstrates that HA3C outperforms current state-of-the-art methods in terms of both sample efficiency and performance (Brockman et al., 2016; Todorov et al., 2012; Tassa et al., 2018).

Our contributions are as follows: 1) Existing RL methods usually utilize historical information to recover Markov assumption in dynamics. It is the first time in the literature that historical augmentation can be used to improve sample efficiency when Markov assumption is satisfied. 2) We propose a new DRL approach to address the problem of how to effectively utilize the historical information in MDPs. 3) Based on this approach, we formulate a new RL algorithm, HA3C, which outperforms existing state-of-the-art DRL algorithms, e.g. TD7 (Fujimoto et al., 2023). 4) Our examples, experiment, and discussion illustrate that in fact, DRL needs to consider historical information in complex MDP tasks.

2 BACKGROUND

An MDP can be written as a 5-tuple $\mathbb{M} = \langle \mathcal{S}, \mathcal{A}, R, \boldsymbol{P}, \gamma \rangle$ with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function R, transition model $\boldsymbol{P}$, and discount factor γ. In an MDP, RL can maximize the discounted cumulative reward by learning how to map the states to the actions (Baird, 1995; Duan et al., 2016; Williams, 1992). For a given state $\boldsymbol{s}_t \in \mathcal{S}$ at time step t, the agent executes an action $\boldsymbol{a}_t \in \mathcal{A}$ w.r.t. a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$, to obtain a reward $r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and transfer to a new state $\boldsymbol{s}_{t+1}$. The return of the agent is defined as the discounted cumulative reward $G_t = \sum_{i=t}^{+\infty} \gamma^{i-t} r_i$. Based on the Markov assumption of MDPs, RL can find the optimal policy to maximize the following value function which is the expected return when $\boldsymbol{s}_t = \boldsymbol{s}$ and following π thereafter.

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}^{\pi}\left[G_t | \boldsymbol{s}_t = \boldsymbol{s}\right] = \mathbb{E}^{\pi}\left[\sum_{i=0}^{+\infty} \gamma^i r_{t+i} | \boldsymbol{s}_t = \boldsymbol{s}\right],$$

where $\mathbb{E}^{\pi}[*]$ denotes the expected value of a random variable given that the agent follows policy π.

With advances in deep learning, combining neural networks into RL has drawn significant attention in the literature. Many DRL algorithms learn the optimal policy by the actor-critic method (Kaelbling et al., 1996), where the critic network estimates the Q-function which is the expected return when

$\boldsymbol{s}_t = \boldsymbol{s}$, $\boldsymbol{a}_t = \boldsymbol{a}$, and following policy π thereafter.

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}^\pi \left[G_t | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} \right] = \mathbb{E}^\pi \left[\sum_{i=0}^{+\infty} \gamma^i r_{t+i} | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a} \right],$$

while the actor network updates the policy to maximize the estimated Q-function.

To improve sample efficiency, some DRL methods learn the state representations of the collected state transitions and then add the learned representations to the actor-critic method (Anand et al., 2019; Dayan, 1993; Gelada et al., 2019; Li et al., 2006). This representation learning aims to capture the causal relationships in MDPs, and thus improves sample efficiency (Liu et al., 2020; Van Hoof et al., 2016; Ye et al., 2023; Zhang et al., 2021). For example, ML-DDPG learns the state representations by predicting the next state representation and the reward in MDPs (Munk et al., 2016). As an improvement of ML-DDPG, OFENet learns the high-dimensional state representations by predicting the next state in DenseNet architecture (Ota et al., 2020). TD7 improves the learning of state representations by AvgL1Norm and then combines the learned representations with TD3, checkpoints, and prioritized replay buffer (Fujimoto et al., 2023).

DRL algorithms need to consider historical information when the Markov assumption of MDPs is violated (Eysenbach et al., 2020; Majeed & Hutter, 2018; Hafner et al., 2019b). For Partially Observable MDPs (POMDPs), in which the states are partially observable, deep recurrent Q-network uses LSTMs to encode state transition trajectories in the Q-function estimation (Hausknecht & Stone, 2015). As an improvement of deep recurrent Q-network, deep transformer Q-network uses transformers to encode the state transition trajectories (Esslinger et al., 2022). As a famous DRL algorithm, Dreamer encodes the historical information into the state at every time step in POMDPs (Ha & Schmidhuber, 2018; Hafner et al., 2019a). In delayed MDPs, in which the current state and reward may arrive at the agent with a delay (Katsikopoulos & Engelbrecht, 2003), researchers usually recover the Markov assumption of MDPs by considering the historical actions (Bouteiller et al., 2020; Derman et al., 2021). When the Markov assumption of MDPs is violated by the state abstraction, it is possible to find a history-based policy which works in the abstracted space and is of the same quality as optimal policy (Patil et al., 2024). However, the history-based DRL for the dynamics which are under Markov assumption is largely absent from the literature.

3 Motivation

Let $\boldsymbol{h}_t = \{\boldsymbol{s}_0, \boldsymbol{a}_0, ..., \boldsymbol{s}_t\}$ as the history up to time step t in a sequential decision-making task. The optimal policy may change the decision rule in different time steps and select actions based on historical information. In this case, we should optimize a history-dependent policy $\pi = \{d_t | t = 0, 1, ...\}$ which selects action at time step t by decision-rule $d_t(\boldsymbol{a}_t | \boldsymbol{h}_t)$. Fortunately, based on the Markov assumption of MDPs, there is an optimal stationary policy $\pi(\boldsymbol{a}_t | \boldsymbol{s}_t)$ which is unrelated to time and selects action $\boldsymbol{a}_t$ by only the state $\boldsymbol{s}_t$. This Markov assumption asserts that the probability distributions of state $\boldsymbol{s}_{t+1}$ and reward r_t depend only on the $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ as

$$P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_0, \boldsymbol{a}_0, r_0, ..., \boldsymbol{s}_t, \boldsymbol{a}_t = P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_t, \boldsymbol{a}_t\},$$

where P is the probability distribution in $\boldsymbol{P}$. Let HR and SR denote the class of history-dependent and stationary policies, respectively. Lemma 3.1 is the key of most existing RL algorithms (Puterman, 2014)[Thm. 6.2.10]. The different types of policies are detailed in Appendix A.

Lemma 3.1. *Under the Markov assumption of MDPs, there exists a policy $\pi^* \in SR$ such that, for all t, $V_{\pi^*}(\boldsymbol{s}_t) = \sup_{\pi \in HR} V_\pi(\boldsymbol{h}_t)$.*

Based on Lemma 3.1, existing DRL algorithms for MDPs usually optimize a stationary policy by single-step transitions. If the causal relationships in the modelled MDP are simple, e.g., there are only linear causal relationships in this MDP, such optimization effectively finds the optimal policy. A classical result is that a neural network with a single hidden layer can successfully learn a linear function (Andoni et al., 2014). However, it is still hard to capture complex causal relationships by neural networks. Standard complexity-theoretic results strongly suggest that there is no algorithm efficient enough for learning arbitrary functions, even for target functions representable by very low-depth networks (Applebaum et al., 2006). In fact, a more complex causal function requires neural networks to approximate with more parameters, samples, and time consumption (Bianchini & Scarselli, 2014).

Historical augmentation has the potential to address the above problem by simplifying the causal relationships in MDPs as it can increase the search space of the causal functions and provide much contextual information on state transitions (Hallak et al., 2015; Sodhani et al., 2022).

Example 3.1. For example, if we model the state transitions with Fibonacci sequence as $\boldsymbol{s}_0 = 1$, $\boldsymbol{s}_1 = 1$, $\boldsymbol{s}_2 = 2$, $\boldsymbol{s}_3 = 3$, $\boldsymbol{s}_4 = 5, \cdots$, when $t > 2$, the state transitions in this model will satisfy the Markov assumption of Markov Processes as (Dynkin, 1965)

$$P\{\boldsymbol{s}_{t+1} = \boldsymbol{s}' | \boldsymbol{s}_0, ..., \boldsymbol{s}_t\} = P\{\boldsymbol{s}_{t+1} = \boldsymbol{s}' | \boldsymbol{s}_t\}.$$

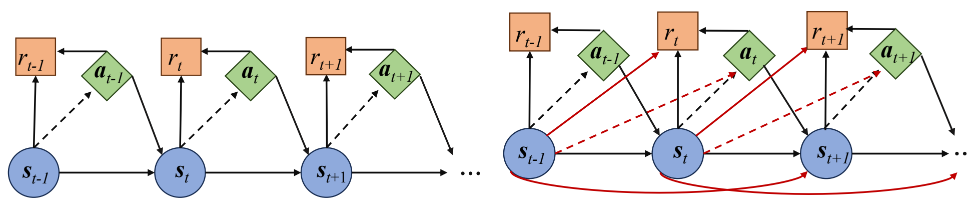
Without considering history, at $\boldsymbol{s}_t$, $\boldsymbol{s}_{t+1}$ will be predicted by a complex time-related formula

$$\boldsymbol{s}_{t+1} = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{t+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{t+1} \right].$$

Fortunately, when considering history, we can predict $\boldsymbol{s}_{t+1}$ by a simple linear function

$$\boldsymbol{s}_{t+1} = \boldsymbol{s}_{t-1} + \boldsymbol{s}_t.$$

In Appendix B, we give another example to illustrate that by historical augmentation, a non-linear causal relationship in single-step transitions may be simplified as a linear causal relationship. Fig. 1(a) presents the original MDP causal relationships and Fig. 1(b) demonstrates the MDP causal relationships with state augmentation. When inferring the causal relationships in a trajectory, the causal function in Fig 1(b) can be simpler than the causal function in Fig 1(a).



(a) MDP (b) MDP with historical augmentation

Figure 1: Causal diagrams of an MDP with or without historical augmentation. The black lines index the original MDP causal relationships and the red lines index the added causal relationships, e.g., the causal relationships from historical augmentation.The dashed lines indicate the information needed in the optimization.

From the analysis above, the motivation of our work is that historical information can simplify the complex causal relationships in MDPs and thus has the potential to improve the sample efficiency of DRL. However, the challenges are 1) how to ensure that the causal relationships learned from historical augmentation are simple and 2) avoiding overfitting caused by the high-dimensional historical data.

4 Method

In this section, we propose a new DRL approach by the representation learning of historically augmented states. Then, we formulate a new DRL algorithm, HA3C, and illustrate the advantage of this algorithm with a visual example.

4.1 Representation Learning on Historically Augmented States

To address the problem of how to effectively utilize the historical information in MDPs, we propose a new DRL approach by the representation learning of historically augmented states. The main idea of this representation learning is to compress the high-dimensional historical trajectories into a low-dimensional representation space (Andre & Russell, 2002; Li et al., 2006). On the one hand, the compressed historical trajectories can be seen as the abstracted features of the historical information to extract the simple causal relationships. On the other hand, this compression will avoid the overfitting caused by the high-dimensional historical data (Ying, 2019).

To keep the Markov assumption of MDPs, our representation learning does not compress the current state. Let $\boldsymbol{s}_{k,t} = \{\boldsymbol{s}_{t-k+1}, ..., \boldsymbol{s}_t\}$. If $t < k$, one can set each $\boldsymbol{s}_i \in \boldsymbol{s}_{k-t,-1}$ by the zero vector $\mathbf{0}$. The causal diagram of MDP with our state abstraction is in Fig. 2. As we can see, when predicting $\boldsymbol{s}_{t+1}$ by $\boldsymbol{s}_{k,t}$ and $\boldsymbol{a}_t$, the dimensionality reduction is only performed on $\boldsymbol{s}_{k-1,t-1}$.

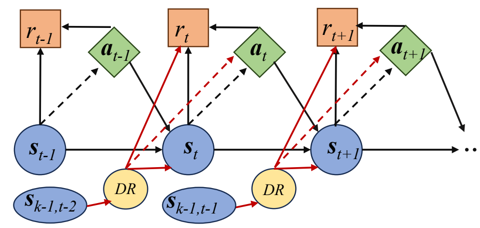


Figure 2: Causal diagram of a historically augmented MDP with state abstraction. DR represents the operation of dimensionality reduction.

Let $S_k D$ denote the class of the stationary deterministic policies based on k-order state trajectories. Theorem 4.1 forms the basis of our DRL approach. This theorem can be implied by Lemma 3.1. For completeness, we provide a proof in Appendix D.

Theorem 4.1. *Under the Markov assumption of MDPs, there exists a stationary deterministic policy $\mu^* \in S_k D$ such that, for all t, $V^{\mu^*}(\boldsymbol{s}_{k,t}) = \sup_{\pi \in HR} V^{\pi}(\boldsymbol{h}_t)$.*

To capture the simplified causal relationships in MDPs by historical augmentation, we define a pair of encoders $\boldsymbol{z}^{\boldsymbol{s}_{k,t}} = f(\boldsymbol{s}_{k,t})$ and $\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t} = g(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t)$. Based on the Markov assumption in MDPs, we can predict $\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}}$, i.e., the representation of $\boldsymbol{s}_{k,t+1}$, by $\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}$. Thus, the two encoders are trained by minimizing the following predicting loss:

$$L(f,g) = ||g(f(\boldsymbol{s}_{k,t}), \boldsymbol{a}_t) - |f(\boldsymbol{s}_{k,t+1})|_\times||_2^2 = ||\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t} - |\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}}|_\times||_2^2, \tag{1}$$

where $| * |_\times$ denotes the stop-gradient operation. As presented in Fig. 3, a simple yet effective CNN network architecture is utilized in our representation learning. In the network of $f(\boldsymbol{s}_{k,t})$, we first use a CNN layer to mine the historical information in $\boldsymbol{s}_{k-1,t-1}$. This layer produces the feature maps of $\boldsymbol{s}_{k-1,t-1}$ by the multiple filters, which are as wide as the state dimensionality. Second, we utilize a max pooling layer to capture the most important features and an average pooling layer to capture the tendency features. Third, we compress the captured features into a low-dimensional space and learn the features of $\boldsymbol{s}_t$. Finally, we concatenate the compressed features of $\boldsymbol{s}_{k-1,t-1}$ and the learned features of $\boldsymbol{s}_t$. The concatenated features are the input of the next fully connected layer to obtain the representation $\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$. In the network of $g(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t)$, we put the concatenation of $\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$ and $\boldsymbol{a}_t$ into the two fully connected layers to obtain the representation $\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}$.

We combine our learned representations with the actor-critic method and thus the Q-function can be defined as $\hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t)$ and the policy can be defined as $\mu(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}) \in S_k D$. Define the probability distribution of $\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}}$ under μ as

$$P^\mu\{\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} = \boldsymbol{z}^{\boldsymbol{s}'_{k,:}} | \boldsymbol{z}^{\boldsymbol{s}_{k,t}} = \boldsymbol{z}^{\boldsymbol{s}_{k,:}}\} = \int_{\mathcal{Z}} \mathbb{E}_{\boldsymbol{a} \sim \mu(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})} \left[p(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}} | \boldsymbol{z}^{\boldsymbol{s}_{k,:}}, \boldsymbol{a}) \right] d\boldsymbol{z}^{\boldsymbol{s}_{k,:}},$$

where $\boldsymbol{s}_{k,:}$ is a k-order state trajectory $\{\boldsymbol{s}_0, ..., \boldsymbol{s}_{k-1}\}$ ending with $\boldsymbol{s}$, i.e., $\boldsymbol{s}_{k-1} = \boldsymbol{s}$, $\mathcal{Z}$ is the set of all possible $\boldsymbol{z}^{\boldsymbol{s}_{k,:}}$, and $p(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}} | \boldsymbol{z}^{\boldsymbol{s}_{k,:}}, \boldsymbol{a})$ is the probability of transferring to $\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}$ with taking $\boldsymbol{a}$ at $\boldsymbol{z}^{\boldsymbol{s}_{k,:}}$. Our optimization is based on a Bellman optimality operator B for μ as

$$B_\mu \hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,:}}, \boldsymbol{a}) = \max_\mu \mathbb{E}_{\boldsymbol{a}_{t+1} \sim \mu, \boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} \sim P^\mu} [r_t + \gamma \hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}}, \boldsymbol{a}_{t+1})]. \tag{2}$$

The following theorem gives the conditions to find the optimal stationary policy in our approach. The proof of this theorem is given in Appendix D.

Theorem 4.2. *Given a finite MDP, if 1) $f(*)$ and $g(*)$ are fixed, 2) $\forall \boldsymbol{s}_{k,:}, \boldsymbol{s}'_{k,:} \in \mathcal{S}_{k,:}$, $\boldsymbol{s} \neq \boldsymbol{s}' \Leftrightarrow \boldsymbol{z}^{\boldsymbol{s}_{k,:}} \neq \boldsymbol{z}^{\boldsymbol{s}'_{k,:}}$, and 3) $L(f,g) \to 0$, then $\hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t)$ converges to the optimal $Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t)$ by the Bellman optimality operator in equation 2.*

This theorem illustrates that no matter whether different historical trajectories lead to different representations on the state $\boldsymbol{s}$, we can still find the optimal stationary policy in the representation space. To make condition 2) hold, we can increase the dimensionality of $\boldsymbol{s}$ in representation learning. This operation also can improve sample efficiency (Ota et al., 2020). To see condition 3) hold, there should exist a $\boldsymbol{s}'_{k,:}$ that satisfies

$$p\{\boldsymbol{s}_{k,t+1} = \boldsymbol{s}'_{k,:} | \boldsymbol{s}_{k,t}, \boldsymbol{a}_t\} \to 1.$$

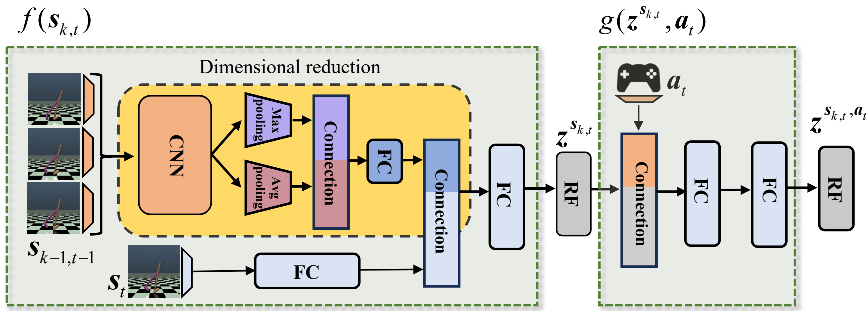


Figure 3: Network architecture of our representation learning. FC represents a fully connected layer and RF represents the state representation features.

There is an analysis of the function approximation error in Appendix D. We add $\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}$ to $\hat{Q}$ to consider the learned relationship between $\boldsymbol{a}_t$ and $\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$ in the representation space. We also add $\boldsymbol{s}_t$ to $\hat{Q}$ and μ to consolidate the relationships in single-step transitions. Thus Q and μ can be written as $\hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}, \boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{s}_t, \boldsymbol{a}_t)$ and $\mu(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{s}_t)$, respectively. The operations in $\hat{Q}$ and μ are shown in Fig. 4.

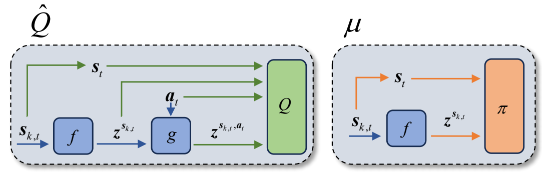


Figure 4: The operations in Q and μ.

Our approach can be connected with POMDPs, High-order MDPs (HMDPs), and state abstraction. A detailed analysis of the connections between our approach and the related work is shown in Appendix C.

4.2 HA3C Algorithm

In this subsection, we propose HA3C algorithm which is a combination of TD3, representation learning, historical augmentation. HA3C has several networks as follows. Two critic networks $(\hat{Q}_{\phi_1}, \hat{Q}_{\phi_2})$, two target critic networks $(\hat{Q}_{\phi_1^T}, \hat{Q}_{\phi_2^T})$, an actor network μ_θ, a target actor network μ_{θ^T}, two encoders (f_σ, g_σ), two fixed encoders $(f_{\sigma^F}, g_{\sigma^F})$, two target encoders $(f_{\sigma^T}, g_{\sigma^T})$, a checkpoint actor network π_{θ^C}, and a checkpoint encoder f_{σ^C}.

To learn the representations with historical augmentation, f_σ, and g_σ are trained by the transitions in buffer $\mathcal{B} = \{\boldsymbol{s}_{k,i}, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{k,i+1}\}$ to minimize the predicting loss in equation 1. For any parameter set α, we define

$$\boldsymbol{z}_\alpha^{\boldsymbol{s}_{k,t}} = f_\alpha(\boldsymbol{s}_{k,t}), \quad \boldsymbol{z}_\alpha^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t} = g_\alpha(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t).$$

Based on the assumption that f_{σ^F} and g_{σ^F} satisfy the conditions in Theorem 4.2 on the most transitions in $\mathcal{B}$, the Q-function is estimated by the following Huber loss function (Huber, 1992).

$$\begin{aligned} L(\phi_i, \mathcal{B}) &= Huber_{(\boldsymbol{s}_{k,t},\boldsymbol{a}_t,r_t,\boldsymbol{s}_{k,t+1})\sim\mathcal{B}}\left[x_t - (\hat{Q}_{\phi_i}(\boldsymbol{z}_{\sigma^F}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}, \boldsymbol{z}_{\sigma^F}^{\boldsymbol{s}_t}, \boldsymbol{s}_t, \boldsymbol{a}_t)\right], \\ x_t &= r_t + \gamma clip(\min(\hat{Q}_{\phi_i^T}(\boldsymbol{z}_{\sigma^T}^{\boldsymbol{s}_{k,t+1},\boldsymbol{a}'}, \boldsymbol{z}_{\sigma^T}^{\boldsymbol{s}_{t+1}}, \boldsymbol{s}_{t+1}, \boldsymbol{a}')), \hat{Q}^{\min}, \hat{Q}^{\max}), \\ \boldsymbol{a}' &= \mu_{\theta^T}(\boldsymbol{z}_{\sigma^T}^{\boldsymbol{s}_{k,t+1}}, \boldsymbol{s}_{t+1}) + \epsilon_T, \epsilon_T \sim \mathcal{N}, \end{aligned} \tag{3}$$

where ϵ_T is target policy noise (Fujimoto et al., 2018), $\mathcal{N}$ is a Gaussian distribution $\mathcal{N}(0, \sigma)$, and $\hat{Q}^{\min}$ and $\hat{Q}^{\max}$ are updated at each time step as

$$\hat{Q}^{\max} \leftarrow \max(x_t, \hat{Q}^{\max}), \quad \hat{Q}^{\min} \leftarrow \min(x_t, \hat{Q}^{\min}).$$

Based on the learned Q-function, the policy network π_θ is updated by

$$\begin{aligned} &\max_\theta \mathbb{E}_{\boldsymbol{s}_{k,t}\sim\mathcal{B}}\left[\textstyle\sum_{i=1,2}\hat{Q}_{\phi_i}(\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}}, \boldsymbol{z}^{\boldsymbol{s}_t}, \boldsymbol{s}_t, \boldsymbol{a})\right], \\ &\boldsymbol{a} = \mu_\theta(\boldsymbol{z}_{\sigma^F}^{\boldsymbol{s}_{k,t}}, \boldsymbol{s}_t). \end{aligned} \tag{4}$$

To explore the new actions and thus generate new transitions in $\mathcal{B}$, exploration noise ϵ is added as

$$\boldsymbol{a}_t \leftarrow \boldsymbol{a}_t + \epsilon_e, \epsilon_e \sim \mathcal{N}.$$

In our TD learning, σ^F, σ^T, ϕ^T, and θ^T are updated by

$$\sigma^F \leftarrow \sigma^T, \quad \sigma^T \leftarrow \sigma, \quad \phi^T \leftarrow \phi, \quad \theta^T \leftarrow \theta. \tag{5}$$

Because DRL algorithms are unstable (Henderson et al., 2018; Teh et al., 2017), we use the checkpoint policy to obtain the cumulative reward in our evaluation (Vaswani et al., 2017). In the training of HA3C, if the current policy outperforms the checkpoint policy, we will update the checkpoint policy with the current policy, then $\sigma^C \leftarrow \sigma$ and $\theta^C \leftarrow \theta$. The checkpoint policy can give a more accurate evaluation by maintaining the high-performance policy unchanged. Furthermore, the LAP replay buffer is utilized to store and replay the transitions (Fujimoto et al., 2023; 2020). The algorithm of online HA3C is presented in Algorithm 1.

Algorithm 1 Online HA3C

```
Initialize the hyper-parameters and networks
Initialize replay buffer 𝓑
for episode = 0 to episode_max do
    Collect k-order transitions by μ_θ and store them in LAP buffer 𝓑
    if Checkpoint condition then
        if μ_θ outperforms μ_{θ^c} then then
            Update checkpoint networks μ_{θ^c} ← μ_θ and f_{σ^c} ← f_σ
        end if
    end if
    Sample k-order transitions from LAP buffer 𝓑
    Train the encoder f_σ and g_σ by equation 1
    Train Q̂_{φ1} and Q̂_{φ2} by equation 3
    Train π_θ by equation 4
    if Target update frequency steps have passed then
        Update target networks by equation 5
    end if
end for
```

Fig. 5 is an example to illustrate the advantage of learning the policy in HA3C. We first collect the obtained states of Walker2d MuJoCo control task by learning the policy with and without historical augmentation, respectively. The max learning step is 4×10^5. Then we map the collected states in 2D space together by UMAP. Finally, we show the reached states without the learning of historical augmentation in the left subfigure of Fig. 5 and the reached states with the learning of historical augmentation in the right subfigure of Fig. 5. Each state is coloured by the reward of reaching it. As we can see, although the actions to obtain the states in high-reward regions (indexed by the red circles) can be explored, without historical augmentation, it is hard to learn the policy which can regenerate these explored actions. Therefore, in the left subfigure, there are only a few states in the high-reward regions. Fortunately, as shown in the right subfigure, there are a lot of states in the high-reward regions when learning the policy with historical augmentation. The visual results of other environments are shown in Appendix F.

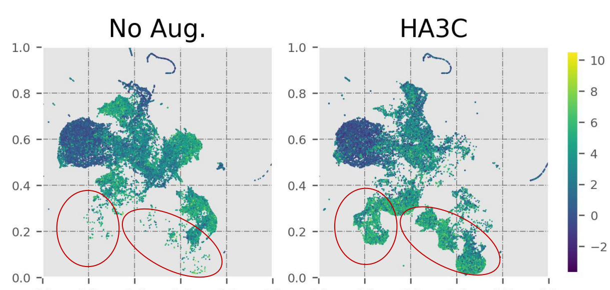


Figure 5: Visual results of the obtained states in Walker2d environment. Each state is coloured by the reward of reaching this state.

5 Experimental Result

In this section, first, we compare HA3C to five existing RL algorithms on five Mujoco control tasks (Todorov et al., 2012). Second, we give the ablation study of HA3C to illustrate that historical augmentation is the real source of the improvement in sample efficiency. Third, we analyze the parameter sensitivity on the length of the historical state trajectory and the number of dimensions of compressed historical trajectories. Finally, we give the running times of the different RL algorithms. The experimental setting is in Appendix E. Appendix F has some supplementary experiments including the state visualization and DMC experiment (Tassa et al., 2018).

5.1 Comparative Evaluation

In this subsection, we evaluate our HA3C on five MuJoCo control tasks including Walker2d, HalfCheetah, Ant, Humanoid, and Hopper. The compared algorithms are TD3 (Fujimoto et al., 2018), SAC (Haarnoja et al., 2018), TQC (Kuznetsov et al., 2020), TD3+OFE (Ota et al., 2020), and TD7 (Fujimoto et al., 2023). For all algorithms, each task runs 10 instances with different random seeds. In each instance, the evaluation is performed every 5000 time steps. The learning curves are shown in Fig. 6 and the numerical results at 400K time step and 1M time step are shown in Table 1.

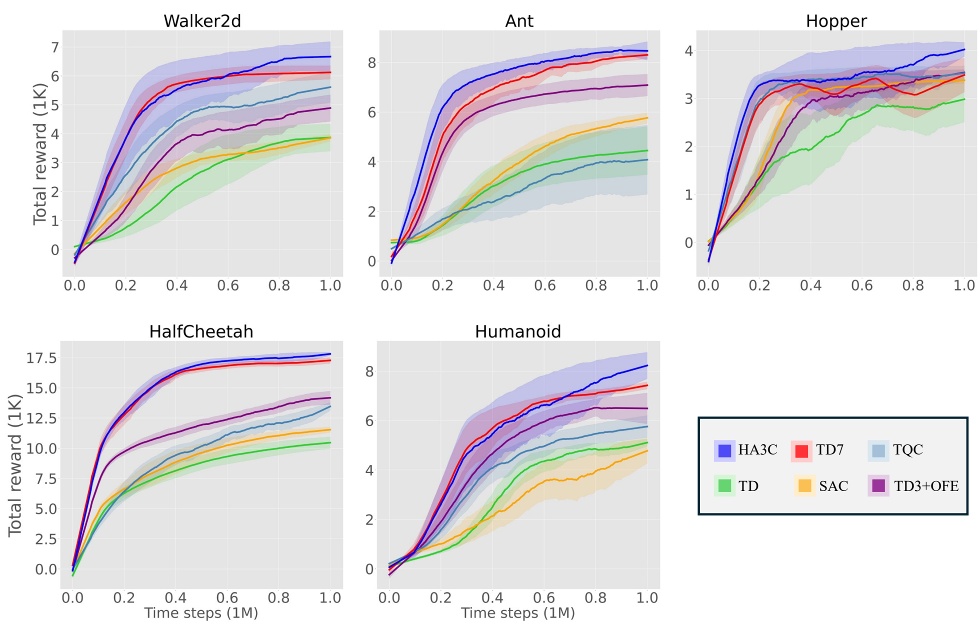


Figure 6: Learning curves of different RL algorithms on the MuJoCo control tasks. The shaded area captures a 90% confidence interval around the average performance.

From Fig. 6 and Table 1, we can see that 1) With the help of historical augmentation, HA3C significantly outperforms the compared algorithms in terms of the early average highest returns (400K time step) and final average highest returns (1M time step); 2) as shown in Fig. 6, because of the instability in rapidly learning complex causal relationships, the early average returns of HA3C on Walker2d and Humanoid are a little lower than the early average returns of TD7, however, HA3C can get the highest final average returns on all of the control tasks.

5.2 Ablation Study

Our ablation study aims to prove that our historical augmentation is the real source of the improvement in sample efficiency. Therefore, we compare HA3C to the following two ablations: 1) Copy Aug. copies the current state k times instead of augmenting with k steps of history in our CNN; 2) No Aug. is TD3 with single-step representation learning and LAP. Our ablation study is performed on Ant, Hopper, and Walker2d. All of the comparison methods have the same parameter setting.

Table 1: The average highest returns over 10 instances on the MuJoCo control tasks at 400K and 1M time steps. ± captures the standard deviation over trials. The best score is highlighted by cyan and the second best score is highlighted by orange.

Algorithm	Time step	Walker2d	HalfCheetah	Ant	Humanoid	Hopper
TD3	400K	2636±933	8229±757	3297±1084	1384±282	2876±859
	1M	4198±516	10560±675	4617±1287	5308±105	3387±137
SAC	400K	3122±156	8945±1368	3893±569	2268±905	3276±86
	1M	3921±163	11729±258	5956±2209	5498±131	3422±87,
TQC	400K	4994±397	9644±1006	3307±939	4061±703	3534±91
	1M	5895±552	13431±561	5258±1165	6140±426	3602±117
TD3+OFE	400K	4329±550	11508±635	6406±549	5193±797	3471±45
	1M	4574±551	14759±696	7246±497	7262±209	3616±28
TD7	400K	5787±444	15625±559	7305±197	5823±231	3440±92
	1M	6354±209	17343±359	8346±291	7405±236	3757±214
HA3C	400K	6441±366	16652±323	7838±138	6099±305	3783±153
	1M	7143±456	18108±294	8687±128	8584±273	4143±170

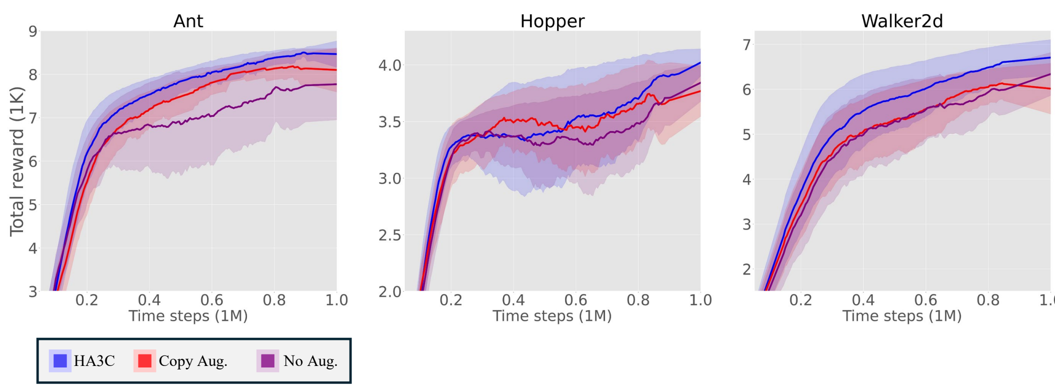


Figure 7: Learning curves of the ablation study on the MuJoCo benchmark. The shaded area captures a 90% confidence interval around the average performance.

As we can see from Fig. 7, HA3C significantly outperforms the compared algorithms in terms of both sample efficiency and performance on Ant and Walker2d. HA3C also significantly outperforms the compared algorithms in final performance on Hopper. This phenomenon illustrates that historical augmentation is the real source for improving sample efficiency.

5.3 Parameter Sensitivity Analysis

In Fig. 8, we analyze the sensitivities of two important parameters, k and N, on Ant. k is the length of the historical state trajectory and N is the number of dimensions of compressed historical trajectories. Both of the above parameters are not used in the previous representation-based RL algorithms. k is set from $\{6, 12, 18, 24\}$ and N is set from $\{8, 16, 64, 256\}$.

As we can see, HA3C is a little sensitive to k and N. When $k \leq 12$ and $N \leq 16$, our historical augmentation will significantly improve the sample efficiency. When $N = 256$, the historical information cannot improve neither sample efficiency nor final performance. This phenomenon illustrates that compressing the historical trajectories into a low-dimensional space is the key to effectively utilize the historical information in MDP tasks.

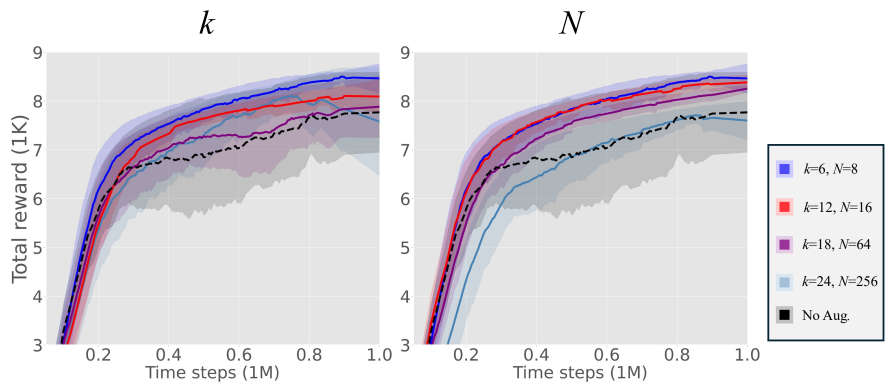


Figure 8: Learning curves of the parameter sensitivity analysis on the MuJoCo benchmark. The shaded area captures a 90% confidence interval around the average performance.

5.4 Running Time

To understand the computational cost of HA3C, we compare the running times of different algorithms with identical computational resources in HalfCheetah control task. The result is shown in Fig. 9. As we can see, the computational cost of HA3C is less than the computational costs of TD3+OFE and TQC but is more than the computational costs of TD3, SAC, and TD7.

6 Conclusion

Under the Markov assumption of MDPs, the probability distributions of the next state and reward depend only on the current state and action. Therefore, given a finite Q-table, we can find the optimal policy in an MDP by a heuristic algorithm which only considers single-step transitions. Different from the heuristic algorithm, DRL algorithms need to approximate the causal functions by learning the causal relationships in MDPs. In this case, DRL is often faced with sample inefficiency from complex causal relationships, as a more complex causal function requires neural networks to approximate with more parameters, samples, and time consumption.

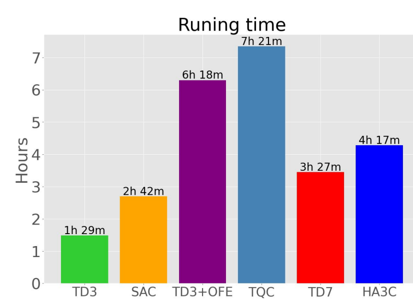


Figure 9: Running times of different algorithms for 1M time steps.

This paper addresses the above problem by augmenting the current state with historical information. We believe that historical augmentation can simplify the causal relationships of state transitions by its inherent contextual information and increasing the search space of the causal functions. Therefore, we focus on optimzing a history-dependent stationary policy in MDPs and propose a new RL algorithm, HA3C. The main idea of HA3C is to learn the state representations by compressing the high-dimensional historical trajectories into a low-dimensional space. In this way, we can extract the simple causal relationships from historical trajectories and avoid the overfitting caused by high-dimensional historical data. Our experiment demonstrates the superior performance of HA3C over five state-of-the-art RL algorithms on MuJoCo control tasks.

References

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8229–8241, 2021.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *Advances in Neural Information Processing Systems*, 32, 2019.

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning polynomials with neural networks. In *International Conference on Machine Learning*, pp. 1908–1916. PMLR, 2014.

David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *AAAI*, pp. 119–125, 2002.

Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

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

Maurice Stevenson Bartlett. *An introduction to stochastic processes*. University Press Cambridge, 1966.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, 2014.

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

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in Neural Information Processing Systems*, 31, 2018.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

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

Esther Derman, Gal Dalal, and Shie Mannor. Acting in delayed environments with non-stationary markov policies. *arXiv preprint arXiv:2101.11992*, 2021.

Richard C Dorf and Robert H Bishop. *Modern control systems*. Pearson, 2011.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, volume 28, pp. 1329–1338, New York, USA, 2016.

EB Dynkin. *Markov processes*. Springer, 1965.

Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer q-networks for partially observable reinforcement learning. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Ben Eysenbach, Xinyang Geng, Sergey Levine, and Russ R Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in Neural Information Processing Systems*, 33:14783–14795, 2020.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, volume 80, pp. 1587–1596, Stockholm, Sweden, 2018.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in Neural Information Processing Systems*, 33: 14219–14230, 2020.

Scott Fujimoto, Wei-Di Chang, Edward J Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For SALE: State-action representation learning for deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

Ruocheng Guo, Lu Cheng, Jundong Li, P Richard Hahn, and Huan Liu. A survey of learning causality with data: Problems and methods. *ACM Computing Surveys (CSUR)*, 53(4):1–37, 2020.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, volume 80, pp. 1861–1870, Stockholm, Sweden, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019a.

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

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *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.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in Statistics: Methodology and Distribution*, pp. 492–518. Springer, 1992.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Tommi Jaakkola, Michael Jordan, and Satinder Singh. Convergence of stochastic iterative dynamic programming algorithms. *Advances in Neural Information Processing Systems*, 6, 1993.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

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

DP Kingma. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pp. 5556–5566. PMLR, 2020.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *AI&M*, 2006.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

Guoqing Liu, Chuheng Zhang, Li Zhao, Tao Qin, Jinhua Zhu, Li Jian, Nenghai Yu, and Tie-Yan Liu. Return-based contrastive representation learning for reinforcement learning. In *International Conference on Learning Representations*, 2020.

Sultan Javed Majeed and Marcus Hutter. On q-learning convergence for non-markov decision processes. In *IJCAI*, volume 18, pp. 2546–2552, 2018.

Francisco S Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pp. 1–4, 2001.

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.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in applied probability*, 29(2):429–443, 1997.

Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4667–4673. IEEE, 2016.

Chengzhuo Ni, Yaqi Duan, Munther Dahleh, Mengdi Wang, and Anru R Zhang. Learning good state and action representations for markov decision process via tensor decomposition. *Journal of Machine Learning Research*, 24(115):1–53, 2023.

Kei Ota, Tomoaki Oiki, Devesh Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In *International Conference on Machine Learning*, pp. 7424–7433. PMLR, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Gandharv Patil, Aditya Mahajan, and Doina Precup. On learning history-based policies for controlling markov decision processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 3511–3519. PMLR, 2024.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

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

Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

Sahand Rezaei-Shoshtari, Rosie Zhao, Prakash Panangaden, David Meger, and Doina Precup. Continuous mdp homomorphisms and homomorphic policy gradient. *Advances in Neural Information Processing Systems*, 35:20189–20204, 2022.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, volume 37, pp. 1889–1897, Lille, France, 2015.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, volume 32, pp. 1387–1395, Beijing, China, 2014.

Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. Block contextual mdps for continual learning. In *Learning for Dynamics and Control Conference*, pp. 608–623. PMLR, 2022.

David Sprunger and Bart Jacobs. The differential calculus of causal functions. *arXiv preprint arXiv:1904.10611*, 2019.

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

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

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

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

Herke Van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3928–3934. IEEE, 2016.

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.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Mingxuan Ye, Yufei Kuang, Jie Wang, Rui Yang, Wengang Zhou, Houqiang Li, and Feng Wu. State sequences prediction via fourier transform for representation learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168, pp. 022022. IOP Publishing, 2019.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

A Different Policies

Time-related policies can be History-dependent (H) or k-order Markov (M_k) (Derman et al., 2020; Puterman, 2014). Denote $\mathcal{H}_t$ as the set of possible histories up to time step t. A history-dependent policy $\pi = \{d_t|t = 0, 1, ...\}$ at t maps histories to actions as $d_t : \mathcal{H}_t \mapsto \mathcal{A}$. A k-order Markov policy $\pi = \{d_t|t = 0, 1, ...\}$ at t maps k-order state transition trajectories to actions as $d_t : \mathcal{S}_{k,t} \mapsto \mathcal{A}$. A k-order stationary (S_k) policy is unrelated to time as $\pi : \mathcal{S}_{k,:} \mapsto \mathcal{A}$. In general, a randomized (R) policy selects the actions by a probability distribution as $\pi(\boldsymbol{a}|*)$. π is a deterministic (D) policy if and only if $\pi(\boldsymbol{a}|*) \in \{0, 1\}$. Based on the above analysis, we can obtain History-dependent Random (HR) policies, History-dependent Deterministic (HD) policies, k-order Markov Random (M_kR) policies, k-order Markov Deterministic (M_kD) policies, k-order Stationary Random (S_kR) policies, and k-order Stationary Deterministic (S_kD) policies.

The above policies are summarized in Table 2. The relationships among them are demonstrated in Fig. 10. It is noteworthy that sometimes historical actions will be considered in decision-making. In this case, without loss of generality, a historical state $\boldsymbol{s}_{i|i\leq t-1}$ can be updated by $\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i \cup \boldsymbol{a}_i$.

Table 2: Different types of policies.

Policy	Abbreviation	Action
History-dependent Random	HR	$\boldsymbol{a}_t \sim d_t(\boldsymbol{s}_{0,t}),\, d_t \in \pi$
History-dependent Deterministic	HD	$\boldsymbol{a}_t = d_t(\boldsymbol{s}_{0,t}),\, d_t \in \pi$
k-order Markov Random	M_kR	$\boldsymbol{a}_t \sim d_t(\boldsymbol{s}_{k-t+1,t}),\, d_t \in \pi$
k-order Markov Deterministic	M_kD	$\boldsymbol{a}_t = d_t(\boldsymbol{s}_{k-t+1,t}),\, d_t \in \pi$
k-order Stationary Random	S_kR	$\boldsymbol{a}_t \sim \pi(\boldsymbol{s}_{k-t+1,t})$
k-order Stationary Deterministic	S_kD	$\boldsymbol{a}_t = \pi(\boldsymbol{s}_{k-t+1,t})$

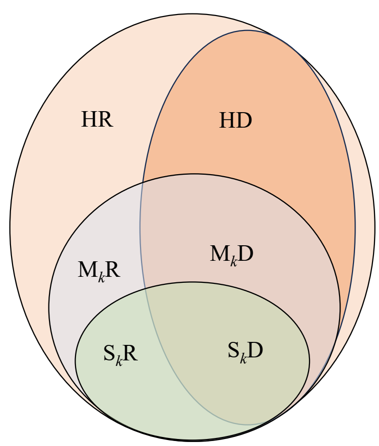


Figure 10: The relations among different policies.

B AN EXAMPLE OF IMPROVING SAMPLE EFFICIENCY IN MDPS BY HISTORICAL AUGMENTATION

Define a sequence as follows: 1) $|\beta_0| \neq 1$; 2) If $i > 1$, then $\beta_{i+1} = \beta_i^2$.

Based on the sequence above, we can define an MDP $\mathbb{M} = \langle \mathcal{S}, \mathcal{A}, R, \boldsymbol{P}, \gamma \rangle$. At time step t, state $\boldsymbol{s}_t = [\beta_t, \beta_{t+2}]^\top$ and action $\boldsymbol{a}_t$ is computed by a linear function $f(*)$ on state $\boldsymbol{s}_t$ or augmented state $\boldsymbol{s}_{k,t}$. Without considering historical information, reward r_t is defined as

$$r_t = -|f(\boldsymbol{s}_t) - (\beta_t + \sqrt{\beta_{t+2}} + \beta_{t+2})| = -|\boldsymbol{w}\boldsymbol{s}_t + b - (\beta_t + \sqrt{\beta_{t+2}} + \beta_{t+2})|, \tag{6}$$

where $\boldsymbol{w}$ is a two-dimensional vector and b is a constant. In transition model $\boldsymbol{P}$, $\boldsymbol{s}_0$ can be defined as $[\beta_0, \beta_2]^\top$ and $\boldsymbol{s}_{t+1}$ can be computed by $\boldsymbol{s}_t$ as

$$\boldsymbol{s}_{t+1} = [\beta_t^2, \beta_{t+2}^2]^\top = \boldsymbol{s}_t \odot \boldsymbol{s}_t, \tag{7}$$

where $\odot$ is Hadamard product. $\gamma = 0.99$.

From equation 6 and equation 7, it is easy to see that $\mathbb{M}$ satisfies the Markov assumption of MDPs. To maximize the discounted cumulative reward in $\mathbb{M}$, we should minimize

$$\arg\min_{\boldsymbol{w},b} ||f(\boldsymbol{s}_t) - (\beta_t + \sqrt{\beta_{t+2}} + \beta_{t+2})||_2 = \arg\min_{\boldsymbol{w},b} ||\boldsymbol{w}\boldsymbol{s}_t + b - (\beta_t^2 + \sqrt{\beta_{t+2}} + \beta_{t+2})||_2 \tag{8}$$

at each time step t. However, it is hard to minimize equation 8 by $f(\boldsymbol{s}_t)$, which is a linear model on $\boldsymbol{s}_t$.

The above problem can be solved by the historical augmentation of $\boldsymbol{s}_t$. When considering the historical augmentation of $\boldsymbol{s}_t$, $f(*)$ on $\boldsymbol{s}_{2,t}$ can be defined as

$$f(\boldsymbol{s}_{2,t}) = \boldsymbol{w}_0\boldsymbol{s}_t + \boldsymbol{w}_1\boldsymbol{s}_{t-1} + b.$$

Instead of minimizing equation 8, we can minimize

$$\begin{aligned} & \arg\min_{\boldsymbol{w}_0,\boldsymbol{w}_1,b} ||f(\boldsymbol{s}_{2,t}) - (\beta_t + \sqrt{\beta_{t+2}} + \beta_{t+2})||_2 \\ = \quad & \arg\min_{\boldsymbol{w}_0,\boldsymbol{w}_1,b} ||\boldsymbol{w}_0\boldsymbol{s}_t + \boldsymbol{w}_1\boldsymbol{s}_{t-1} + b - (\beta_t^2 + \sqrt{\beta_{t+2}} + \beta_{t+2})||_2. \end{aligned}$$

Let $\boldsymbol{w}_0 = [1, 1]$, $\boldsymbol{w}_1 = [0, 1]$, and $b = 0$. From $\beta_{t+1} = \sqrt{\beta_{t+2}}$, we have

$$\begin{aligned} & ||\boldsymbol{w}_0\boldsymbol{s}_t + \boldsymbol{w}_1\boldsymbol{s}_{t-1} + b - (\beta_t + \sqrt{\beta_{t+2}} + \beta_{t+2})||_2 \\ = \quad & ||\boldsymbol{w}_0\boldsymbol{s}_t + \boldsymbol{w}_1\boldsymbol{s}_{t-1} + b - (\beta_t + \beta_{t+1} + \beta_{t+2})||_2 \\ = \quad & ||([1, 1][\beta_t, \beta_{t+2}]^\top + [0, 1][\beta_{t-1}, \beta_{t+1}]^\top - (\beta_t + \beta_{t+1} + \beta_{t+2})||_2 \\ = \quad & 0. \end{aligned}$$

In this case, the cumulative reward in $\mathbb{M}$ can be maximized.

C CONNECTED TO RELATED WORK

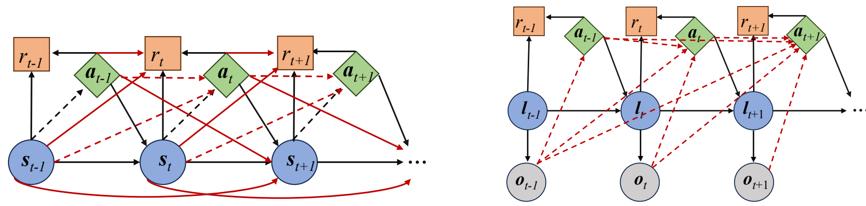
C.1 CONNECTED TO HMDPS

In HMDPs, the probability distributions of the reward and next state depend not only on the current state and action but also on the historical states and actions. For a k-order HMDPs, we have

$$P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_0, \boldsymbol{a}_0, r_0, ..., \boldsymbol{s}_t, \boldsymbol{a}_t\} = P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_{t-k+1}, \boldsymbol{a}_{t-k+1}, .., \boldsymbol{s}_t, \boldsymbol{a}_t\}.$$

The causal diagram of HMDP is presented in Fig. 11(a). Our approach optimizes the policy by a simplified HMDP model in which the probability distributions of the reward and next state depend on the current state-action pair and compressed historical trajectory as

$$P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_0, \boldsymbol{a}_0, r_0, ..., \boldsymbol{s}_t, \boldsymbol{a}_t\} = P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | DR(\boldsymbol{s}_{t-1,k-1}), .., \boldsymbol{s}_t, \boldsymbol{a}_t\}.$$



(a) HMDP (b) POMDP

Figure 11: Causal diagram of HMDPs and POMDPs.

C.2 Connected to POMDPs

In POMDPs, the states are partially observable. Define the partially observable state at time step t as $\boldsymbol{l}_t$ and the observable part of $\boldsymbol{l}_t$ as $\boldsymbol{o}_t$. The causal diagram of POMDPs is shown in Fig. 11(b). Under the faithfulness assumption, $\boldsymbol{o}_t$ and $\boldsymbol{o}_{t+k}$ are mutually dependent conditional on $\forall k > 1$, $\{\boldsymbol{o}_i, \boldsymbol{a}_i\}_{t<i<t+k}$ (Kalisch & Bühlman, 2007). Therefore, in a POMDP, the optimal policy π at time step t should consider not only $\boldsymbol{o}_t$ but also the historical information $\{\boldsymbol{o}_i, \boldsymbol{a}_i\}_{0\leq i<t}$. When k is large, long-length rollout estimation is needed in POMDPs.

RL algorithms of world models, such as Dream, model the sequential decision-making as a POMDP (Ha & Schmidhuber, 2018; Hafner et al., 2019a). They usually encode the historical information at t by an encoder f^t to construct $\boldsymbol{s}_{t+1}$ as

$$\boldsymbol{s}_{t+1} = f^t(\boldsymbol{o}_t, \boldsymbol{a}_t, ..., f^1(\boldsymbol{o}_1, \boldsymbol{a}_1, f^0(\boldsymbol{o}_0, \boldsymbol{a}_0)).$$

When t is large, some partially observable states will be encoded many times, leading to the loss of some important discriminative information.

Compared with POMDP-based RL algorithms, our HA3C can better adjust the considered steps in history according to the actual task and thus effectively find the optimal policy in history-based sequential decision-making.

C.3 Connected to State Abstraction

State abstraction aims to reduce ground MDPs with large state spaces to abstract MDPs with smaller state spaces by aggregating states according to some notion of equality or similarity (Bartlett, 1966). Through abstraction, intelligent agents may need to consider only the salient distinguishing information of their environments. Given an abstraction function as $F : \mathcal{S} \to \overline{\mathcal{S}}$, we can define the abstract version of MDP $\mathbb{M}$ as $\overline{\mathbb{M}} = \langle \overline{\mathcal{S}}, \mathcal{A}, \overline{R}, \overline{\boldsymbol{P}}, \gamma \rangle$. A Q-irrelevance abstraction function F^Q is that for any action $\boldsymbol{a}$, $F^Q(\boldsymbol{s}) = F^Q(\boldsymbol{s'})$ implies $Q(\boldsymbol{s}, \boldsymbol{a}) = Q(\boldsymbol{s'}, \boldsymbol{a})$. Then we have the following theorem.

Theorem C.1. *Define an MDP as $\mathbb{M}_k = \langle \mathcal{S}_{k,:}, \mathcal{A}, R, \boldsymbol{P}_k, \gamma \rangle$. Under the conditions 1), 2), and 3) in Theorem 4.2, encoder f is a Q-irrelevance abstraction on $\boldsymbol{s}_{k,:}$.*

Theorem C.1 illustrates that our representation learning can be seen as the Q-irrelevance abstraction of the historically augmented states. The proof of this theorem is given in Appendix D.

D Theoretical Analysis

D.1 Proof of Theorem 4.1

Now we give the proof to Theorem 4.1. The different types of policies in this proof are summarized in Table 2. The relationships between these policies are shown in Fig. 10.

Based on the Markov assumption of MDPs, we have

$$\begin{aligned} &P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_0, \boldsymbol{a}_0, r_0, ..., \boldsymbol{s}_t, \boldsymbol{a}_t\} \\ &= P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_{t-k+1,t}, \boldsymbol{a}_t\} \\ &= P\{\boldsymbol{s}_{t+1} = \boldsymbol{s'}, r_t = r | \boldsymbol{s}_t, \boldsymbol{a}_t\}. \end{aligned} \tag{9}$$

For any $\pi \in HR$, we can define $V_\pi(\boldsymbol{h}_t)$ by

$$V^\pi(\boldsymbol{h}_t) = \mathbb{E}^\pi \left[\sum_{i=t}^{+\infty} \gamma^i R(\boldsymbol{h}_{t+i}, \boldsymbol{a}_{t+i}) \right].$$

From Fig. 10, we have $S_k D \in M_k D \in M_k R \in HR$. In view of equation 9, we see for all t that

$$\sup_{\pi \in HR} V^\pi(\boldsymbol{h}_t) = \sup_{\pi \in S_k D} V^\pi(\boldsymbol{s}_{k,t}).$$

First, for all t, we demonstrate that

$$\sup_{\pi \in HR} V^\pi(\boldsymbol{h}_t) = \sup_{\pi \in M_k R} V^\pi(\boldsymbol{s}_{k,t}). \tag{10}$$

This is a direct result of Theorem D.1. The proof of this theorem is presented in D.1.1.

Theorem D.1. *Let $\pi = \{d_t | t = 0, 1, ...\} \in HR$. Then $\forall \boldsymbol{s}_{k,:} \in \mathcal{S}_{k,:}$, based on equation 9, there exists a policy $\pi' = \{d'_t | t = 0, 1, ...\} \in M_k R$ satisfying*

$$p^\pi(\boldsymbol{a}_{t+i} = \boldsymbol{a'}, \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) = p^{\pi'}(\boldsymbol{a}_{t+i} = \boldsymbol{a'}, \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}),$$

where $p^\pi(*)$ denotes the probability of $*$ provided that the agent follows policy π.

Then Theorem D.2 illustrates that the value functions of $\pi \in M_k D$ and $\pi \in M_k R$ have the same upper bound. The proof of this theorem is demonstrated in D.1.2.

Theorem D.2. *If a bounded function V on $\mathcal{S}_{k,:}$ satisfies the optimal Bellman equation that*

$$V(\boldsymbol{s}_{k,t}) = \sup_{\boldsymbol{a} \in \mathcal{A}} \left\{ R(\boldsymbol{s}_{k,t}, \boldsymbol{a}) + \gamma \int_{\mathcal{S}_{k,:}} V(\boldsymbol{s}_{k,t+1} | \boldsymbol{s}_{t+1} = \boldsymbol{s'}) p(\boldsymbol{s'} | \boldsymbol{s}_{k,t}, \boldsymbol{a}) d\boldsymbol{s'}_{k,:} \right\},$$

then

$$\sup_{\pi \in M_k D} V^\pi(\boldsymbol{s}_{k,t}) = \sup_{\pi \in M_k R} V^\pi(\boldsymbol{s}_{k,t}).$$

Finally, based on equation 9, for all $\boldsymbol{s}_{k,:} \in \mathcal{S}_{k,:}$, if $\boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}$, then

$$\sup_{\boldsymbol{a} \in \mathcal{A}} V(\boldsymbol{s}_{k,t}) = \sup_{\boldsymbol{a} \in \mathcal{A}} V(\boldsymbol{s}_{k,:}). \tag{11}$$

Let $\boldsymbol{a} = \pi(\boldsymbol{s}_{k,:})$, where $\pi \in S_k D$. It follows that

$$\sup_{\pi \in S_k D} V^\pi(\boldsymbol{s}_{k,:}) = \sup_{\pi \in M_k D} V^\pi(\boldsymbol{s}_{k,t}). \tag{12}$$

Under equation 10, equation 11 and equation 12, $\forall t$, if $\boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}$, then

$$\sup_{\pi \in HR} V^\pi(\boldsymbol{h}_t) = \sup_{\pi \in M_k R} V^\pi(\boldsymbol{s}_{k,t}) = \sup_{\pi \in M_k D} V^\pi(\boldsymbol{s}_{k,t}) = \sup_{\pi \in S_k D} V^\pi(\boldsymbol{s}_{k,:}).$$

D.1.1 PROOF OF THEOREM D.1

We assume that Theorem D.1 holds for $i = 1, 2, 3, ..., n-1$. Given a policy $\pi \in HR$, based on equation 9, we see that there exists a policy $\pi' \in M_k R$ satisfying

$$\begin{aligned}
& p^\pi(\boldsymbol{s}_{k,t+i} = \boldsymbol{s''}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) \\
= \; & \int_{\mathcal{S}_{k,:}} \int_{\mathcal{A}} p^\pi(\boldsymbol{s}_{k,t+i-1} = \boldsymbol{s'}_{k,:}, \boldsymbol{a}_{t+i-1} = \boldsymbol{a'} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) p(\boldsymbol{s''} | \boldsymbol{s'}_{k,:}, \boldsymbol{a'}) d\boldsymbol{a'} d\boldsymbol{s'}_{k,:} \\
= \; & \int_{\mathcal{S}_{k,:}} \int_{\mathcal{A}} p^{\pi'}(\boldsymbol{s}_{k,t+i-1} = \boldsymbol{s'}_{k,:}, \boldsymbol{a}_{t+i-1} = \boldsymbol{a'} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) p(\boldsymbol{s''} | \boldsymbol{s'}_{k,:}, \boldsymbol{a'}) d\boldsymbol{a'} d\boldsymbol{s'}_{k,:} \\
= \; & p^{\pi'}(\boldsymbol{s}_{k,t+i} = \boldsymbol{s''}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}).
\end{aligned}$$

The above equality follows from the induction hypothesis. The π' also can satisfy

$$p^{\pi'}(\boldsymbol{a}_{t+i} = \boldsymbol{a'} | \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:}) = p^\pi(\boldsymbol{a}_{t+i} = \boldsymbol{a'} | \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:}).$$

Therefore,

$$\begin{aligned}
& p^{\pi'}(\boldsymbol{a}_{t+i} = \boldsymbol{a'}, \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) \\
= \; & p^{\pi'}(\boldsymbol{a}_{t+i} = \boldsymbol{a'} | \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:}) p^{\pi'}(\boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) \\
= \; & p^{\pi}(\boldsymbol{a}_{t+i} = \boldsymbol{a'} | \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:}) p^{\pi}(\boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}) \\
= \; & p^{\pi}(\boldsymbol{a}_{t+i} = \boldsymbol{a'}, \boldsymbol{s}_{k,t+i} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_{k,t} = \boldsymbol{s}_{k,:}).
\end{aligned}$$

D.1.2 Proof of Theorem D.2

In view of $M_k D \in M_k R$, we have

$$\sup_{\pi \in M_k D} V^{\pi}(\boldsymbol{s}_{k,t}) \leq \sup_{\pi \in M_k R} V^{\pi}(\boldsymbol{s}_{k,t}). \tag{13}$$

It follows that

$$\begin{aligned}
&\sup_{\boldsymbol{a} \in \mathcal{A}} \left\{ R(\boldsymbol{s}_{k,t}, \boldsymbol{a}) + \gamma \int_{\mathcal{S}_{k,:}} V(\boldsymbol{s}_{k,t+1} | \boldsymbol{s}_{k,t+1} = \boldsymbol{s'}) p(\boldsymbol{s'} | \boldsymbol{s}_{k,t}, \boldsymbol{a})) d\boldsymbol{s'} \right\} \\
\geq \quad & \int_{\mathcal{A}} p(d_t(\boldsymbol{s}_{k,t}) = \boldsymbol{a}) \left[R(\boldsymbol{s}_{k,t}, \boldsymbol{a}) + \gamma \int_{\mathcal{S}_{k,:}} V(\boldsymbol{s}_{k,t+1} | \boldsymbol{s}_{t+1} = \boldsymbol{s'}) p(\boldsymbol{s'} | \boldsymbol{s}_{k,t}, \boldsymbol{a})) d\boldsymbol{s'}_{k,:} \right] d\boldsymbol{a},
\end{aligned}$$

where $d_t \in M_k R$. Thus

$$\sup_{\pi \in M_k D} V^{\pi}(\boldsymbol{s}_{k,t}) \geq \sup_{\pi \in M_k R} V^{\pi}(\boldsymbol{s}_{k,t}). \tag{14}$$

Combining equation 13 and equation 14, we have

$$\sup_{\pi \in M_k D} V^{\pi}(\boldsymbol{s}_{k,t}) = \sup_{\pi \in M_k R} V^{\pi}(\boldsymbol{s}_{k,t}).$$

D.2 Proof of Theorem 4.2

To prove Theorem 4.2, we give the proof of Theorem C.1 first. Under the condition 1) of Theorem 4.2, one sees that there are only two independent variables $\boldsymbol{s}_{k,:}$ and $\boldsymbol{a}$. Under the Markov assumption and the condition 2) of Theorem 4.2, we have

$$P\{\boldsymbol{s}_{k,t+1} = \boldsymbol{s'}_{k,:} | \boldsymbol{s}_0, \boldsymbol{a}_0, r_0, ..., \boldsymbol{s}_t, \boldsymbol{a}_t\} = P\{\boldsymbol{s}_{k,t+1} = \boldsymbol{s'}_{k,:} | \boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t\}. \tag{15}$$

Then, under the condition 3) of Theorem 4.2, we have

$$\begin{aligned}
P\{\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} = \boldsymbol{z}^{\boldsymbol{s'}_{k,:}} | \boldsymbol{s}_{k,t}, \boldsymbol{a}_t\} &\doteq P\{\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} = \boldsymbol{z}^{\boldsymbol{s'}_{k,:}} | \boldsymbol{z}^{\boldsymbol{s}_{k,t}, \boldsymbol{a}_t}\} \\
&= P\{\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} = \boldsymbol{z}^{\boldsymbol{s'}_{k,:}} | g(f(\boldsymbol{s}_{k,t}), \boldsymbol{a}_t)\} \\
&= P\{\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} = \boldsymbol{z}^{\boldsymbol{s'}_{k,:}} | g(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t)\} \\
&= P\{\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}} = \boldsymbol{z}^{\boldsymbol{s'}_{k,:}} | \boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t\}.
\end{aligned} \tag{16}$$

Define an MDP as $\mathbb{M}_k = \langle \mathcal{S}_{k,:}, \mathcal{A}, R, \boldsymbol{P}_k, \gamma \rangle$. From equation 15 and equation 16, we obtain

$$\boldsymbol{z}^{\boldsymbol{s}_{k,:}} = \boldsymbol{z}^{\boldsymbol{s'}_{k,:}} \rightarrow Q(\boldsymbol{s}_{k,:}, \boldsymbol{a}) = Q(\boldsymbol{s'}_{k,:}, \boldsymbol{a})$$

Because $\boldsymbol{z}^{\boldsymbol{s}_{k,:}} = f(\boldsymbol{s}_{k,:})$, we see that encoder f is a Q-irrelevance abstraction on $\boldsymbol{s}_{k,:}$.

Define an abstracted MDP of $\mathbb{M}_k$ as $\overline{\mathbb{M}}_k = \langle \mathcal{Z}, \mathcal{A}, R, \boldsymbol{P}_k, \gamma \rangle$, where $\mathcal{Z}$ is the encoded space of $\mathcal{S}_{k,:}$. Operator B_μ can be written as

$$B_\mu \hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,:}}, \boldsymbol{a}) = R(\boldsymbol{z}^{\boldsymbol{s}_{k,:}}, \boldsymbol{a}) + \max_{\mu} \gamma \int_{\mathcal{Z}} \hat{Q}(\boldsymbol{z}^{\boldsymbol{s}_{k,:}}, \mu(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})) p(\boldsymbol{z}^{\boldsymbol{s'}_{k,:}} | \boldsymbol{z}^{\boldsymbol{s}_{k,:}, \boldsymbol{a}}) d\boldsymbol{z}^{\boldsymbol{s'}_{k,:}}.$$

Now we provide a proof (sketch) to Theorem 4.2. Since the optimality of μ follows from the optimal actions as well as their Q-values are preserved after abstraction, we see that B is a contraction in the sup-norm and the optimal Q-function $\hat{Q}^*$ is the unique fixed point of B. Thus we can finally find the optimal policy μ^* by B_μ (Melo, 2001). When the agent estimates the optimal Q-function based on experience, we have the following update rule in each time step T by Lemma D.3 (Jaakkola et al., 1993; Melo, 2001).

$$\hat{Q}_{t+1}(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t) = \hat{Q}_t(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t) + \alpha_t (r_t + \gamma \max_{\mu} \hat{Q}_t(\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}}, \mu(\boldsymbol{z}^{\boldsymbol{s}_{k,t+1}})) - \hat{Q}_t(\boldsymbol{z}^{\boldsymbol{s}_{k,t}}, \boldsymbol{a}_t)).$$

$\hat{Q}_t$ converges to Q^* as long as

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty.$$

Lemma D.3. *The random process $\{\Delta_t\}$ taking values in $\mathbb{R}^n$ and defined as*

$$\Delta_{t+1}(\boldsymbol{y}) = (1-\alpha_t)\Delta_t(\boldsymbol{y}) + \alpha_t F_t(\boldsymbol{y})$$

converges to zero under the following assumptions:
1) $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$,
2) $\mathbb{E}\left[||F_t(\boldsymbol{y})|\mathcal{F}_t||_w\right] \leq \gamma||\Delta_t||_w$ with $\gamma < 1$, and
3) $\mathrm{Var}[F_t(\boldsymbol{y})|\mathcal{F}_t] \leq C(1+||\Delta_t||_w^2)$ for $C > 0$,
where $\mathcal{F} = \{\Delta_t, \Delta_{t-1}, ..., F_{t-1}, ..., \alpha_{t-1}, ...,\}$ strands for the past at step t and $||*||_w$ refers to some weighted maximum norm.

D.3 Approximation Error Analysis

Define the value function in $\mathcal{Z}$ as $\hat{V}$. The bound of the approximation error between the transition probabilities in space $\mathcal{S}_{k,:}$, and $\mathcal{Z}$ based on the optimal value function $\hat{V}^*$ can be defined as (Müller, 1997)

$$\max_{\boldsymbol{s}_{k,:},\boldsymbol{a}} \left| \int_{\mathcal{S}_{k,:}} \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}) p(\boldsymbol{s}'_{k,:}|\boldsymbol{s}_{k,:},\boldsymbol{a}) d\boldsymbol{s}'_{k,:} - \int_{\mathcal{Z}} \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}) p(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}|\boldsymbol{z}^{\boldsymbol{s}_{k,:}},\boldsymbol{a}) d\boldsymbol{z}^{\boldsymbol{s}'_{k,:}} \right| = \delta.$$

Based on δ, we analyze the approximation error in Theorem D.4.

Theorem D.4. *The worst-case difference between $V^{\mu}(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})$ and optimal value function $V^*(\boldsymbol{s})$ is bounded as:*

$$||V^*(\boldsymbol{s}) - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})||_\infty \leq \frac{\gamma\delta}{1-\gamma}.$$

We provide the proof to the above theorem as follows. Based on the Markov assumption of MDPs, we have

$$||V^*(\boldsymbol{s}) - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})||_\infty = ||V^*(\boldsymbol{s}_{k,:}) - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})||_\infty.$$

Now we prove that

$$||V^*(\boldsymbol{s}_{k,:}) - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})||_\infty \leq \frac{\gamma\delta}{1-\gamma}. \tag{17}$$

In view of $R(\boldsymbol{s},\boldsymbol{a}) = R(\boldsymbol{s}_{k,:},\boldsymbol{a}) = R(\boldsymbol{z}^{\boldsymbol{s}_{k,:}},\boldsymbol{a})$ in the value function approximation, we have

$$\begin{aligned}
& ||V^*(\boldsymbol{s}_{k,:}) - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})||_\infty \\
\leq\; & \max_{\boldsymbol{s}_{k,:},\boldsymbol{a}} ||Q^*(\boldsymbol{s}_{k,:},\boldsymbol{a}) - \hat{Q}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}},\boldsymbol{a})|| \\
=\; & \max_{\boldsymbol{s}_{k,:},\boldsymbol{a}} \Bigg| R(\boldsymbol{s}_{k,:},\boldsymbol{a}) + \gamma \int_{\mathcal{S}_{k,:}} V^*(\boldsymbol{s}'_{k,:}) p(\boldsymbol{s}'_{k,:}|\boldsymbol{s}_{k,:},\boldsymbol{a}) d\boldsymbol{s}'_{k,:} \\
-\; & R(\boldsymbol{z}^{\boldsymbol{s}_{k,:}},\boldsymbol{a}) - \gamma \int_{\mathcal{Z}} \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}) p(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}|\boldsymbol{z}^{\boldsymbol{s}_{k,:}},\boldsymbol{a}) d\boldsymbol{z}^{\boldsymbol{s}'_{k,:}} \Bigg| \\
\leq\; & \gamma \max_{\boldsymbol{s}_{k,:},\boldsymbol{a}} \left| \int_{\mathcal{S}_{k,:}} V^*(\boldsymbol{s}'_{k,:}) p(\boldsymbol{s}'_{k,:}|\boldsymbol{s}_{k,:},\boldsymbol{a}) d\boldsymbol{s}'_{k,:} - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}) p(\boldsymbol{s}'_{k,:}|\boldsymbol{s}_{k,:},\boldsymbol{a}) d\boldsymbol{s}'_{k,:} \right| \\
+\; & \gamma \max_{\boldsymbol{s}_{k,:},\boldsymbol{a}} \left| \int_{\mathcal{S}_{k,:}} \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}) p(\boldsymbol{s}'_{k,:}|\boldsymbol{s}_{k,:},\boldsymbol{a}) d\boldsymbol{s}'_{k,:} - \int_{\mathcal{Z}} \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}) p(\boldsymbol{z}^{\boldsymbol{s}'_{k,:}}|\boldsymbol{z}^{\boldsymbol{s}_{k,:}},\boldsymbol{a}) d\boldsymbol{z}^{\boldsymbol{s}'_{k,:}} \right| \\
\leq\; & \gamma \left(||V^*(\boldsymbol{s}_{k,:}) - \hat{V}^*(\boldsymbol{z}^{\boldsymbol{s}_{k,:}})||_\infty + \delta \right).
\end{aligned}$$

This proves equation 17. Thus Theorem D.4 holds.

D.4 Analyzing Sample Efficiency in Exploration and Exploitation

In this subsection, we illustrate the benefit of sample efficiency from history augmentation based on two facts:
1) Historical augmentation can improve exploration in DRL. The policy can generate different actions for different transition trajectories that end with the same state;
2) Historical augmentation can also improve exploitation in DRL. History augmentation may simplify the causal relationships between the state and the explored high-reward action, thus the policy network can effectively learn and then regenerate this action.

The detailed analysis of these two facts is as follows. In the previous DRL methods for MDPs, when the policy μ and $\boldsymbol{s}_t = \boldsymbol{s}$ are fixed, we can get only one action by

$$\boldsymbol{a}_t = \mu(\boldsymbol{s}_t), \quad \mu \in S_1D.$$

However, based on our history-based policy

$$\boldsymbol{a}_t = \mu(\boldsymbol{s}_{k,t}), \quad \mu \in S_kD|_{k\geq 2}.$$

$\boldsymbol{a}_t$ can be changed by the change of the $\boldsymbol{s}_{k-1,t-1}$. We define the set of possible actions from policy $\mu \in S_kD$ at state $\boldsymbol{s}$ as $\boldsymbol{A}^{\boldsymbol{s}}_{\mu}$ and the set of possible k-order trajectories end with state $\boldsymbol{s}$ as $\boldsymbol{S}^{\boldsymbol{s}}_{k}$. As we can see, $|\boldsymbol{A}^{\boldsymbol{s}}_{\mu}| \leq |\boldsymbol{S}^{\boldsymbol{s}}_{k}|$.

Fig. 12 is the causal diagram of regenerating a high-reward action with or without historical augmentation. For a policy network $\mu_\theta \in S_1D$ and $\boldsymbol{a} = \mu_\theta(\boldsymbol{s})$, we may get $\boldsymbol{a}^* = \boldsymbol{a} + \epsilon$ with $R(\boldsymbol{s}, \boldsymbol{a}^*) > R(\boldsymbol{s}, \boldsymbol{a})$. However, it may be hard to regenerate $\boldsymbol{a}^*$ by the policy network $\mu_\theta(\boldsymbol{s})$ because the noise ϵ is independent of parameter θ. Fortunately, the causal relationship between $\boldsymbol{s}_{k,t}|_{k\geq 2}$ and $\boldsymbol{a}^*$ may be simpler than the causal relationship between $\boldsymbol{s}_t$ and $\boldsymbol{a}^*$ (See the example in Appendix B). In this case, we can effectively learn the policy $\mu_\theta \in S_kD$ to regenerate the $\boldsymbol{a}^*$ at state $\boldsymbol{s}$ by $\boldsymbol{a}^* = \mu_\theta(\boldsymbol{s}_{k,t})$ (See the example in Fig. 5).

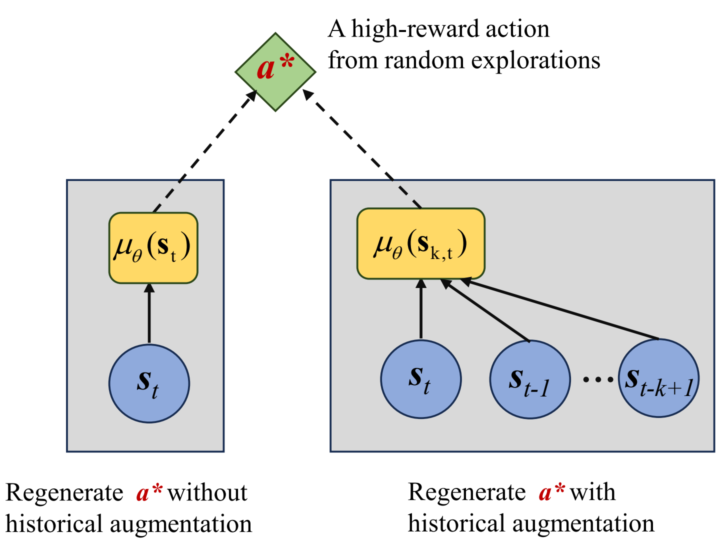


Figure 12: The causal diagram of regenerating a high-reward action with or without historical augmentation. The dashed lines indicate the information needed in the optimization.

E Experimental Setting

All experiments are run on a single Nvidia 3090 GPU and AMD 5900X CPU. We use the following software versions:
• Python 3.9.12
• Pytorch 2.0.0 (Paszke et al., 2019)
• CUDA 12.2
• Gymnasium 0.29.1 (Brockman et al., 2016)
• MuJoCo 3.2.3 (Todorov et al., 2012)

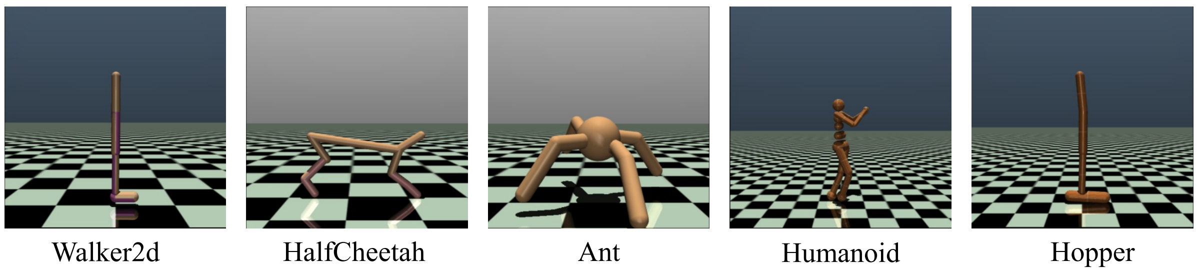


Figure 13: The environments in our experiments.

The environments in our experiment are shown in Fig. 13 and detailed as follows:
1) Walker2d aims to walk in the forward direction as fast as possible.
2) HalfCheetah aims to run forward as fast as possible.
3) Ant aims to coordinate the four legs to move in the forward direction as fast as possible.
4) Humanoid aims to walk forward as fast as possible without falling over.
5) Hopper aims to make hops that move in the forward direction as fast as possible.

The compared RL algorithms in our experiment are detailed as follows.
• Online:
1) TD3 takes the minimum value between a pair of critic networks to address the overestimation of Q-value and reduces per-update error by delaying policy updates (Fujimoto et al., 2018).
2) SAC is an actor-critic algorithm based on the maximum entropy approach. The objective encourages policy stochasticity by augmenting the reward with the entropy at each step (Haarnoja et al., 2018).
3) OFE-TD3 increases the input dimensionality of the networks by representation learning to improve the sample efficiency of TD3 (Ota et al., 2020).
4) TQC addresses the overestimation of Q-value by the combination of the distributional representation of a critic, truncation of critic prediction, and ensembling of multiple critics (Kuznetsov et al., 2020).
5) TD7 is an effective DRL algorithm which combines TD3, state representation learning, checkpoints, prioritized experience replay, and a behaviour cloning term (only used for offline RL) (Fujimoto et al., 2023).

The hyper-parameters of HA3C are shown in Table 3. For Hopper, γ is set as 0.992. Network architecture details are described in Pseudocode 1-3. The optimizer of our networks is Adam Kingma (2015).

Table 3: Hyper-parameters.

Parameter	Value	Brief explanation
Start-timesteps	25000	Time steps of the initial random policy is used
Batch-size	512	Batch size for both actor and critic
t_{pol}	2	Policy update frequency
t_{tar}	250	Target update rate
t_{ear}	1	Early assessment episodes for checkpoint operation
t_{lat}	3	Late assessment episodes for checkpoint operation
T_{ear}	750K	Early time steps for checkpoint operation
σ_e	0.1	Std of exploration noise
σ_T	0.005	Std of target policy noise
c	(-0.11,0.11)	Target policy noise clipping
k	6	The length of the considering state sequences
γ	0.99	Discount factor
l_e	0.0006	The learning rate of the encoder network
l_p	0.0003	The learning rate of the actor-network
l_Q	0.0003	The learning rate of the network of the Q-functions
α	0.25	Controlling the amount of prioritization in LAP
P_m	1.1	Minimum priority in LAP

Pseudocode 1: Critic network Details

Critic network:
L1 = Linear(state-dim + action-dim, 256)
L2 = Linear($\boldsymbol{z}^{\boldsymbol{s}}$-dim * 2 + 256, 256)
L3 = Linear(256, 256)
L4 = Linear(256, 1)
Critic forward pass:
$\boldsymbol{x}$ = Concatenate([$\boldsymbol{s}_t$, $\boldsymbol{a}_t$])
$\boldsymbol{x}$ = AvgL1Norm(L1($\boldsymbol{x}$))
$\boldsymbol{x}$ = Concatenate([$\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}$, $\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$, $\boldsymbol{x}$])
$\boldsymbol{x}$ = Elu(L2($\boldsymbol{x}$))
$\boldsymbol{x}$ = Elu(L3($\boldsymbol{x}$))
$\tau(\boldsymbol{s}_{k,t}, \boldsymbol{a}_t)$ = L4($\boldsymbol{x}$)

Pseudocode 2: Actor network Details

Actor network:
L1 = Linear(state-dim, 256)
L2 = Linear($\boldsymbol{z}^{\boldsymbol{s}}$-dim + 256, 256)
L3 = Linear(256, 256)
L4 = Linear(256, action-dim)
Actor forward pass:
$\boldsymbol{x}$ = AvgL1Norm(L1($\boldsymbol{s}_t$))
$\boldsymbol{x}$ = Concatenate([$\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$, $\boldsymbol{x}$])
$\boldsymbol{x}$ = ReLU(l1($\boldsymbol{x}$))
$\boldsymbol{x}$ = ReLU(l2($\boldsymbol{x}$))
$\boldsymbol{a}_t$ = Tanh(l3($\boldsymbol{x}$))

Pseudocode 3: Encoder Details

State Encoder f Network:
Conv = Conv2d(kernel-num=64, kernel-size=(3, state-dim), stride=1)
Pool = MaxPool2d((1, 1))
L1 = Linear(64, 16)
L2 = Linear(state-dim, 256)
L3 = Linear(256+16, 256)
L4 = Linear(256, zs-dim)

State Encoder f Forward Pass:
$\boldsymbol{x}$ = Conv($\boldsymbol{s}_{k-1,t-1}$)
$\boldsymbol{x}$ = Pool($\boldsymbol{x}$)
$\boldsymbol{x}$ = Elu(L1($\boldsymbol{x}$))
$\boldsymbol{x}$ = AvgL1Norm($\boldsymbol{x}$)
$\boldsymbol{y}$ = Elu(L2($\boldsymbol{s}_t$))
$\boldsymbol{x}$ = Concatenate([$\boldsymbol{x}$, $\boldsymbol{y}$])
$\boldsymbol{x}$ = Elu(L3($\boldsymbol{x}$))
$\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$ = AvgL1Norm(L4($\boldsymbol{x}$))

State-Action Encoder g Network:
L1 = Linear(action-dim + $\boldsymbol{z}^{\boldsymbol{s}}$-dim, 256)
L2 = Linear(256, 256)
L3 = Linear(256, $\boldsymbol{z}^{\boldsymbol{s}}$-dim)
State-Action Encoder g Forward Pass:
$\boldsymbol{x}$ = Concatenate([$\boldsymbol{a}_t$, $\boldsymbol{z}^{\boldsymbol{s}_{k,t}}$])
$\boldsymbol{x}$ = Elu(L1($\boldsymbol{x}$))
$\boldsymbol{x}$ = Elu(L2($\boldsymbol{x}$))
$\boldsymbol{z}^{\boldsymbol{s}_{k,t},\boldsymbol{a}_t}$ = L3($\boldsymbol{x}$)

F Supplementary Experiment

F.1 BipedaWalker Experiment

To illustrate the benefit of history augmentation for complex MDP tasks, we test HA3C and No Aug. (HA3C without historical augmentation) on BipedalWalker and BipedalWalker-hardcore tasks. In BipedalWalker a robot is trained to move forward with slightly uneven terrain. Compared with BipedalWalker, BipedalWalker-hardcore is a more complex task, where the above robot is trained to move forward with ladders, stumps, and pitfalls. Therefore, the causal relationships in the transitions of BipedalWalker-hardcore are more complex than those in the transitions of BipedalWalker. The environments and learning curves are shown in Fig. 14 and the numerical results are shown in Table 4.

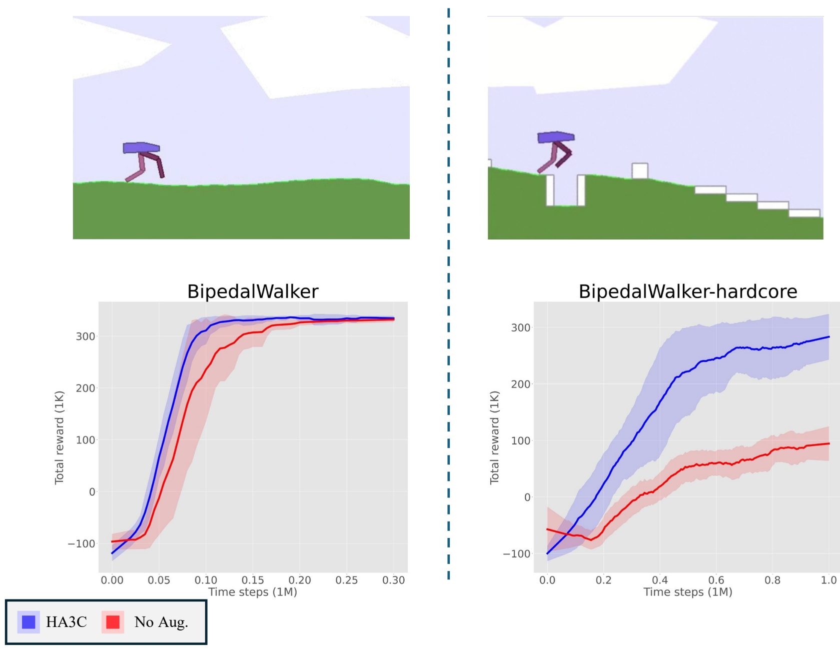


Figure 14: The environments and learning curves on BipedalWalker and BipedalWalker-hardcore tasks.

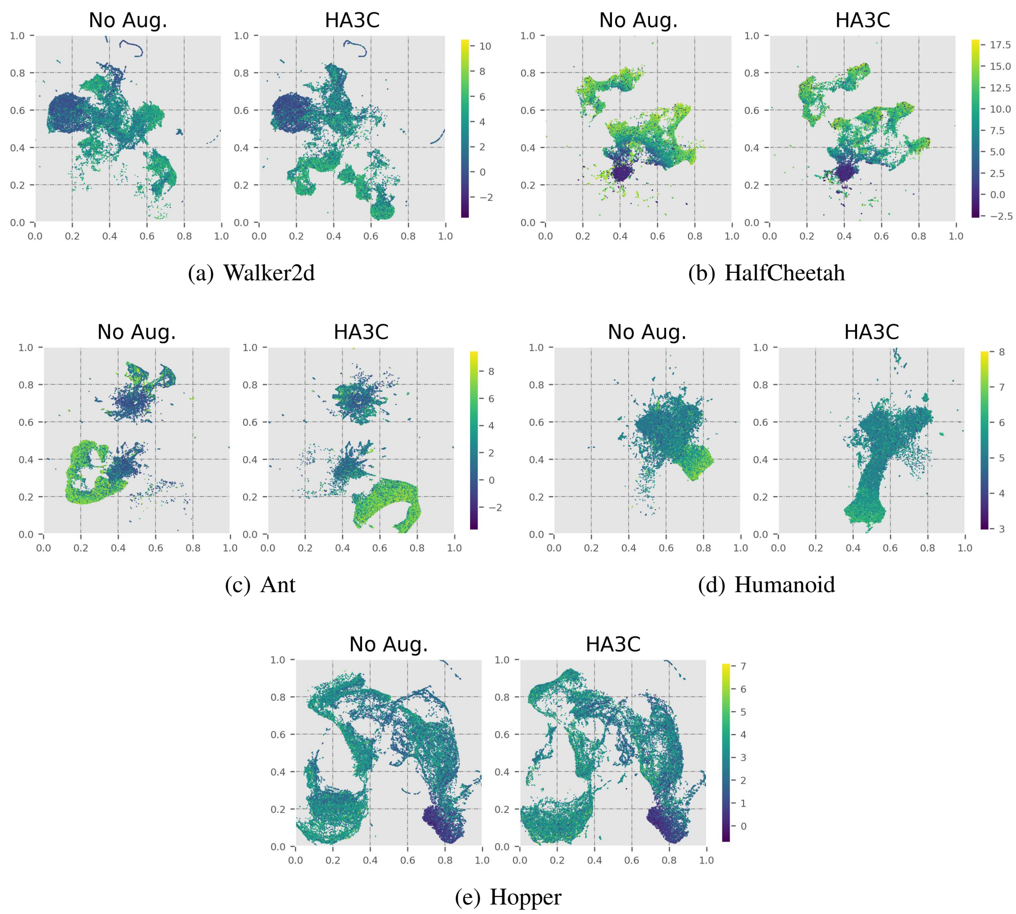
Table 4: The average highest returns of HA3C and No Aug. on BipedaWalker and BipedaWalker-hardcore tasks.

Algorithm	BipedalWalke	BipedalWalker-hardcore
HA3C	332 ± 27	316 ± 19
No Aug.	325 ± 31	171 ± 21

As we can see, although, both HA3C and No Aug. can get the high cumulative rewards in BipedalWalker, only HA3C can get the high cumulative rewards in BipedalWalker-hardcore. This is because by historical augmentation our HA3C can simplify the causal relationships in the transitions of BipedalWalker-hardcore.

F.2 Visualized Results of HA3C

Fig. 15 presents the visual results of the transitions in HA3C and No Aug. The collected states of each control task are mapped together by UMAP. The max learning step is 4×10^5 and each state is coloured by the reward of reaching it.



(a) Walker2d (b) HalfCheetah (c) Ant (d) Humanoid (e) Hopper

Figure 15: Visualized results of the explored states in No Aug. and HA3C.

As we can see, in Walker2d, Ant, and Humanoid, the high-reward states from HA3C are more than those from No Aug. This result illustrates that the sample efficiency of DRL can be effectively improved by learning the state representations with historical augmentation.

F.3 Deep Mind Control Suite Experiment

In this subsection, we evaluate our HA3C on five DMC tasks including ball_in_cup-catch, walker-run, quadruped-run, cheetah-run, and reacher-hard (Tassa et al., 2018). The compared algorithms are TD3 (Fujimoto et al., 2018) and TD7 (Fujimoto et al., 2023). For all algorithms, each task runs 10 instances with 10^6 time steps with different random seeds. In each instance, the evaluation is performed every 5000 time steps. Some parameters are changed as follows. For quadruped-run, l_e is set as 0.0006, σ_T is set as 0.06, and c is set as $(-0.12, 0.12)$. For other tasks, l_e is set as 0.0005 and c is set as $(-0.1, 0.1)$. The learning curves are shown in Fig. 18 and the numerical results at 300K time step and 1M time step are shown in Table 5.

As we can see, in most cases, HA3C has higher cumulative rewards than the compared algorithms. For walker-run, quadruped-run, and reacher-hard, HA3C outperforms the compared algorithms in terms of both the early performance and the final performance. For ball_in_cup-catch and cheetah-run, HA3C outperforms all of the compared algorithms in the final performance but the average return of HA3C is lower than the average return of TD7 in the early performance.

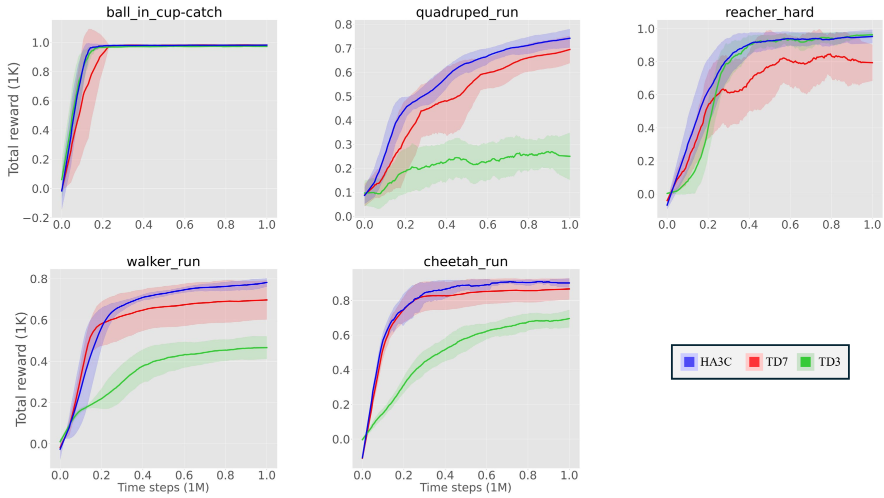


Figure 16: Learning curves of different RL algorithms on the deep mind control suite tasks.

Table 5: The average highest returns over 10 instances on the deep mind control suite tasks at 400K and 1M time steps.

Algorithm	Time step	ball_in_cup-catch	walker-run	quadruped-run	cheetah-run	reacher-hard
TD3	400K	981±2	387±71	331±65	550±76	971±3
	1M	985±1	481±54	444±22	729±39	979±1
TD7	400K	990±2	654±96	531±69	836±75	879±91
	1M	991±1	706±95	703±54	868±56	979±5
HA3C	400K	989±2	713±41	598±36	834±108	976±5
	1M	992±1	789±19	758±24	916±5	985±5

F.4 Longer Training Runs

In this section, we compare our HA3C with TD7 on five Mujoco control tasks with 3M training steps. The learning curves are shown in Fig. 17 and the numerical results are shown in Table 6.

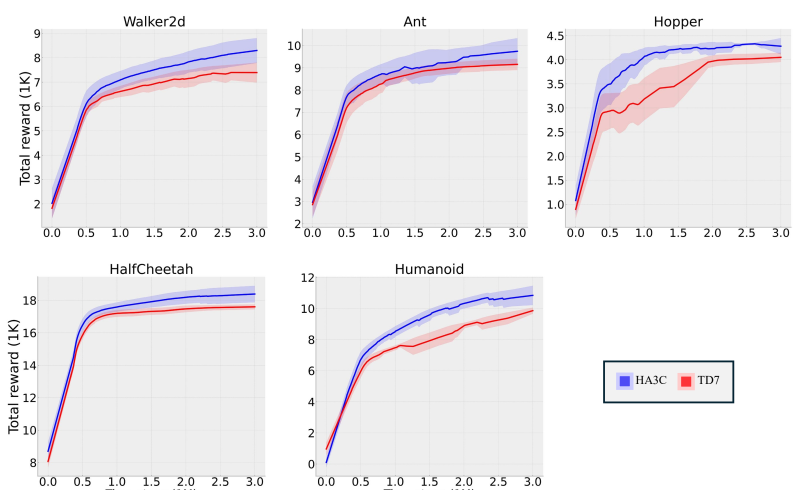


Figure 17: Learning curves of HA3C and TD7 on the Mujoco control tasks.

Table 6: The average highest returns over 10 instances of HA3C and TD7 at 3M time steps. ± captures the standard deviation over trials.

Algorithm	Walker2d	HalfCheetah	Ant	Humanoid	Hopper
TD7	7570±321	17787±286	9225±450	9850±226	4049±156
HA3C	8463±829	18687±683	9794±891	11381±344	4413±59

As we can see, HA3C outperforms TD7 on the five Mujoco control tasks. It is noteworthy that the cumulative rewards of HA3C are significantly higher than the cumulative rewards of TD7 on Walker2d, Humanoid, and Hopper.

F.5 Combining Historical Representation Learning with SAC

In this section, we combine our historical representation learning with SAC to construct HA3C-SAC method (Haarnoja et al., 2018). Then we evaluate HA3C-SAC on three MuJoCo control tasks including Walker2d, Humanoid, and Hopper. The compared methods includes the original SAC and SALE-SAC, which combines the representation learning with SAC without historical augmentation (Fujimoto et al., 2023). The learning curves are shown in Fig. 17 and the numerical results are shown in Table 7.

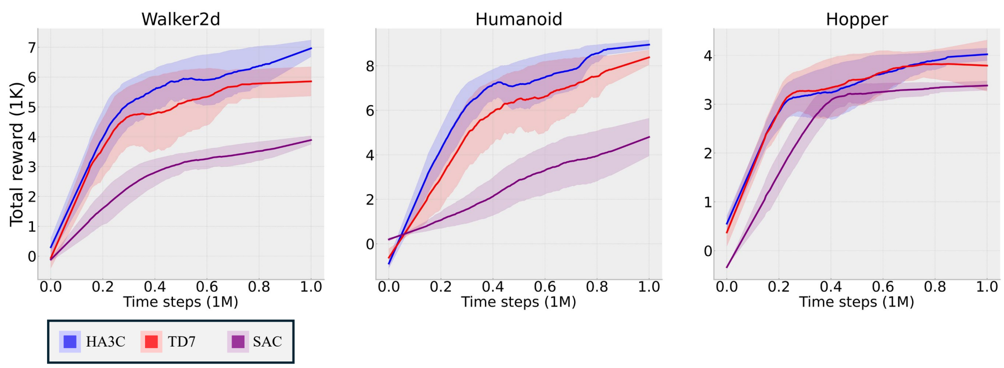


Figure 18: Learning curves of different RL algorithms on the deep mind control suite tasks.

Table 7: The average highest returns on Mujoco control tasks at 400K and 1M time steps.

Algorithm	Time step	Walker2d	Humanoid	Hopper
SAC	400K	2843±148	2268±905	3195±33
	1M	3921±163	5498±131	3422±86
SALE-SAC	400K	5414±377	6430±191	3515±125
	1M	6021±492	8368±330	4038±126
HA3C-SAC	400K	5796±395	7112±339	3566±39
	1M	6950±623	9047±238	4131±48

As we can see, HA3C-SAC outperforms SAC and SALE-SAC on the three Mujoco control tasks. The above results and the results Section 5.1 illustrate that our historical representation learning is robust to different algorithms and tasks.