Reinforcement Learning in Time-delay System

Simon Zhan
Northwestern University

March 14, 2026

Project Background

Background

The timing factor is critical in enabling reinforcement learning (RL) [1] for real-world applications, particularly regarding delays in agent-environment interactions.

Delays are inherent in real-world systems such as robotics[2, 3], high-frequency trading[4], and intelligent transportation[5], arising from physical distance, computational overhead, and signal transmission.

Delays are also prominent in current foundation model infrastructure especially from the inference side for both robotics [6].

Many existing RL methods neglect the impact of such delays, fundamentally affecting the system's safety [2], performance [5] and efficiency [3].

VLA Example

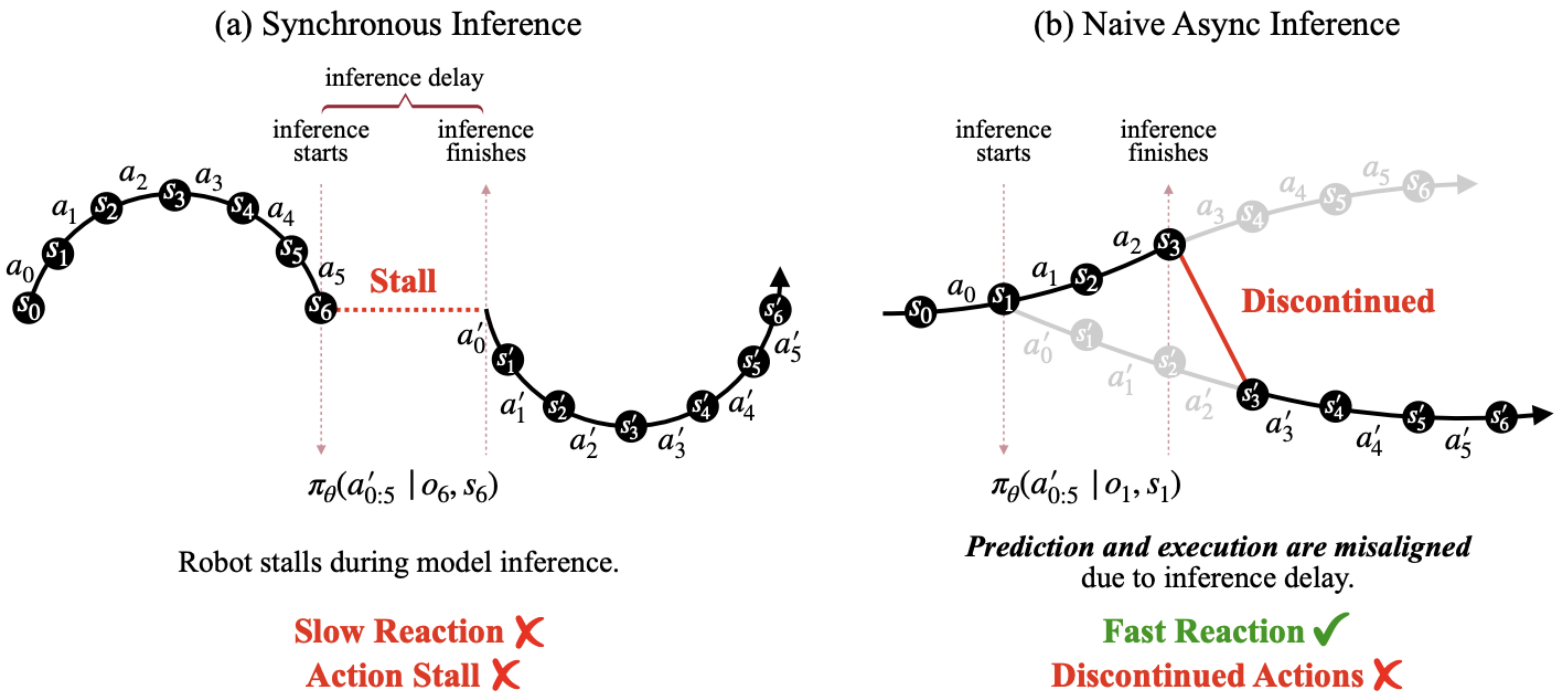


Figure: Delay interplays in the different format of VLA inference.

(Delay-free) Markov Decision Process (MDP)

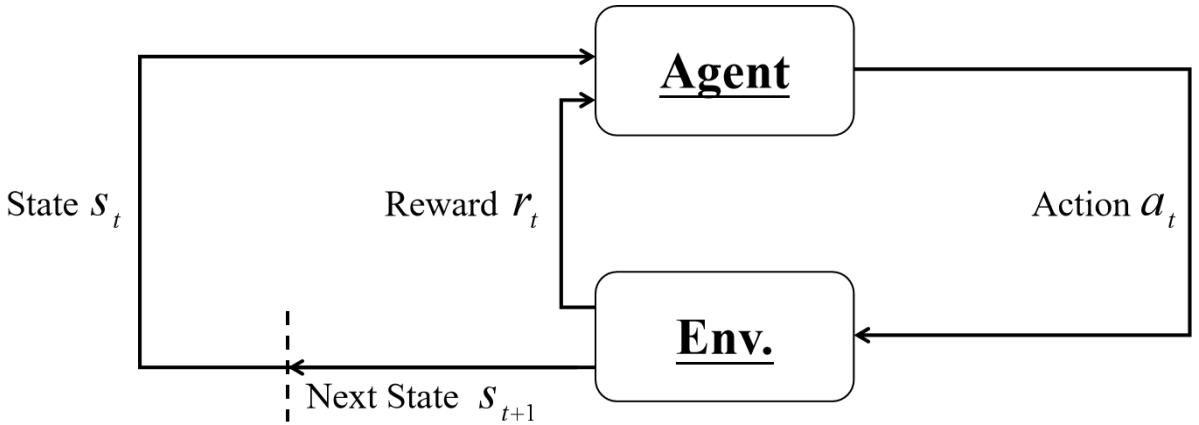


Figure: The interaction in RL is formalized as a Markov Decision Process (MDP).

A delay-free MDP is denoted as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho \rangle$, where

- state space $\mathcal{S}$
- action space $\mathcal{A}$
- transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Omega(\mathcal{S})$
- reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- discount factor $\gamma \in (0, 1)$
- initial state distribution ρ

Delayed MDP

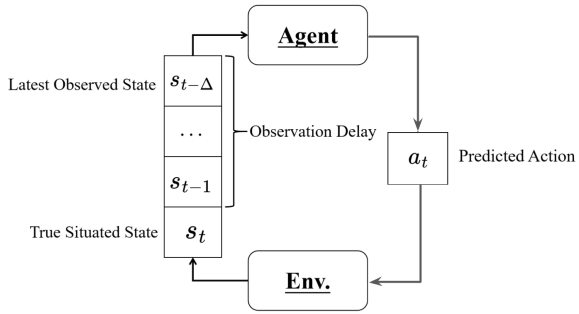


Figure: Constant Delayed MDP with observation-delay Δ.

A Constant Delayed MDP (CDMDP) can also be denoted as an MDP tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{P}_\Delta, \mathcal{R}_\Delta, \gamma, \rho_\Delta \rangle$, where

- augmented state space $\mathcal{X} := \mathcal{S} \times \mathcal{A}^\Delta$
- action space $\mathcal{A}$
- transition function $\mathcal{P}_\Delta(x_{t+1}|x_t, a_t) := \mathcal{P}(s_{t-\Delta+1}|s_{t-\Delta}, a_{t-\Delta})\delta_{a_t}(a'_t)\prod_{i=1}^{\Delta-1}\delta_{a_{t-i}}(a'_{t-i})$
- reward function $\mathcal{R}_\Delta(x_t, a_t) := \mathbb{E}_{s_t \sim b(\cdot|x_t)}[\mathcal{R}(s_t, a_t)]$
- discount factor $\gamma \in (0, 1)$
- initial state distribution $\rho_\Delta := \rho \prod_{i=1}^{\Delta} \delta_{a_{-i}}$

Belief Representation in Delayed RL

Delayed RL can be viewed as a special form of a partially observable RL problem. Therefore, Delayed RL has the belief representation b defined as follows:

$$b(s_t|x_t) = \int_{\mathcal{S}^\Delta} \prod_{i=0}^{\Delta-1} \mathcal{P}(s_{t-\Delta+i+1}|s_{t-\Delta+i}, a_{t-\Delta+i}) \mathrm{d}s_{t-\Delta+i+1} \tag{1}$$

Existing Methods

Augmentation-based Methods

Augmentation-based methods exploit the observation that the information state $x_t = \{s_{t-\Delta}, a_{t-\Delta:t-1}\}$, augmented from the last observable state and the sequential actions, carries the equivalent information with s_t.

Belief-based Methods

Belief-based methods aim to perform RL directly in the original state space $\mathcal{S}$ by estimating the unobservable state s_t from the information state x_t, of which the mapping is referred to as belief b.

Challenges and Contributions

Challenges

1. Augmentation-based methods suffer from the curse of dimensionality.
2. Belief-based methods are vulnerable to compounding errors.

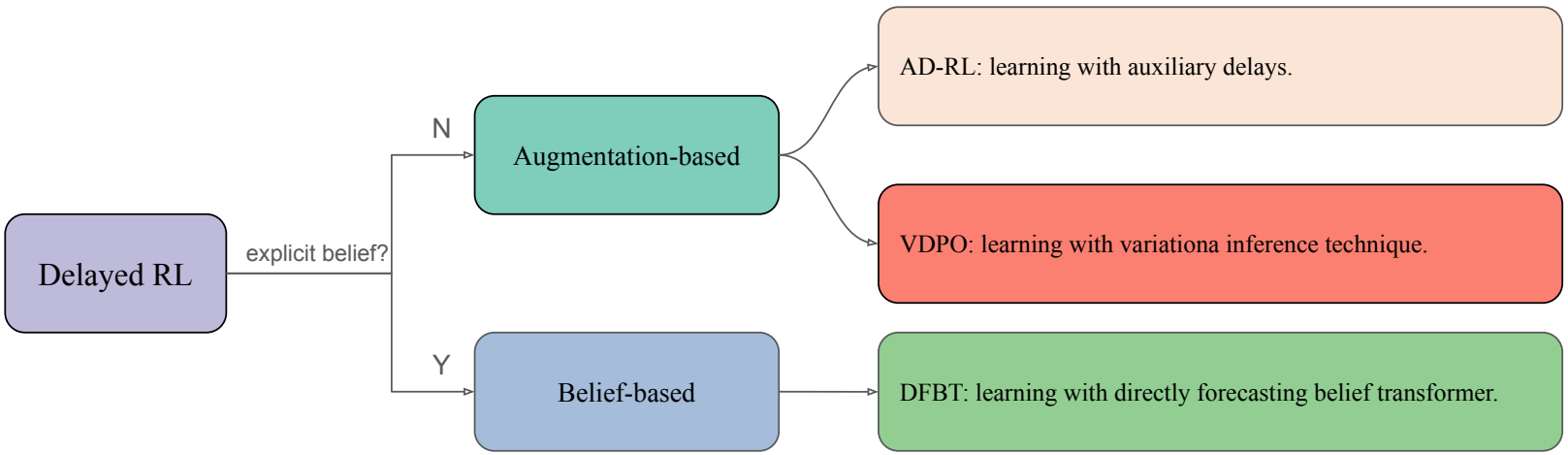


Figure: Our contributions.

Auxiliary Delayed Reinforcement Learning (AD-RL)

- AD-RL improves sample efficiency by bootstrapping from shorter auxiliary delays.
- Theoretically, AD-RL enhances sample efficiency while preserving robust performance.
- Empirically, AD-RL outperforms baselines in both sample efficiency and performance.

Delayed Belief

Instead of learning on the augmented state space $\mathcal{X}$ with delays Δ, AD-RL introduces an auxiliary task with shorter delays $\Delta^\tau (< \Delta)$ with a smaller augmented state space $\mathcal{X}^\tau$. Two augmented-state spaces can be connected by the delayed belief b_Δ.

$$b_\Delta(x_t^\tau | x_t) = \int_{\mathcal{S}^\Delta} \prod_{i=0}^{\Delta-\Delta^\tau-1} \mathcal{P}(s_{t-\Delta+i+1} | s_{t-\Delta+i}, a_{t-\Delta+i}) \mathrm{d}s_{t-\Delta+i+1} \quad (2)$$

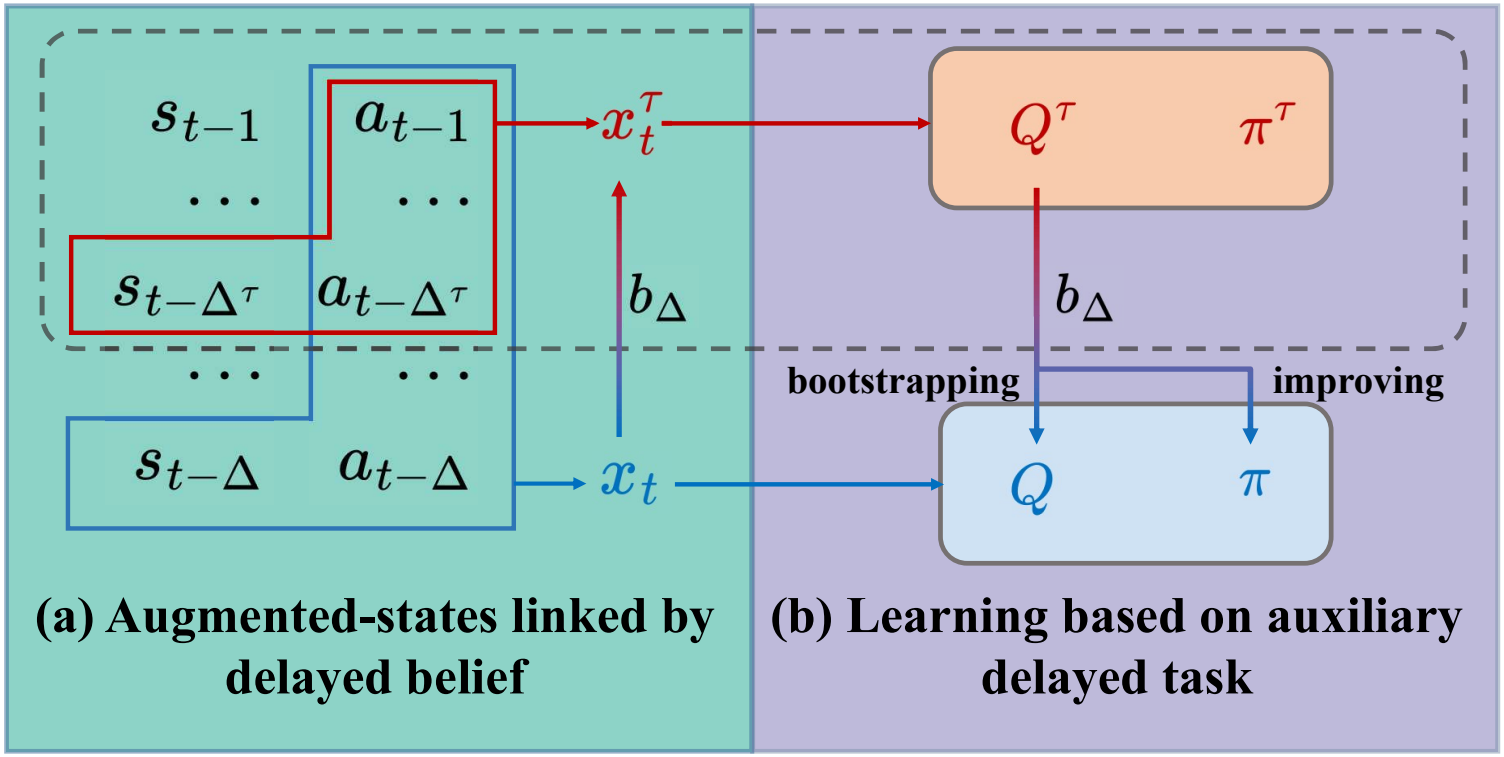


Figure: The overview of AD-RL. The auxiliary task with shorter delays is shown in the dashline box.

AD-VI

For discrete tasks, AD-RL can be applied to value iteration (VI).

VI

$$\mathcal{T}Q(x_t, a_t) \triangleq \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)} \left[Q(x_{t+1}, \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1})) \right]$$

AD-VI (ours)

$$\mathcal{T}Q(x_t, a_t) \triangleq \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x^\tau_{t+1}, \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1})) \right] \tag{3}$$

AD-SPI

For continuous tasks, AD-RL can be applied to soft policy iteration.

Soft Policy Evaluation

$$\mathcal{T}^{\pi} Q(x_t, a_t) \triangleq \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{a_{t+1}\sim\pi(\cdot|x_{t+1}) \\ x_{t+1}\sim\mathcal{P}_{\Delta}(\cdot|x_t,a_t)}} [Q(x_{t+1}, a_{t+1}) - \log \pi(a_{t+1}|x_{t+1})]$$

AD Soft Policy Evaluation (ours)

$$\mathcal{T}^{\pi} Q(x_t, a_t) \triangleq \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{a_{t+1}\sim\pi(\cdot|x_{t+1}) \\ x^{\tau}_{t+1}\sim b_{\Delta}(\cdot|x_{t+1}) \\ x_{t+1}\sim\mathcal{P}_{\Delta}(\cdot|x_t,a_t)}} \left[Q^{\tau}(x^{\tau}_{t+1}, a_{t+1}) - \log \pi(a_{t+1}|x_{t+1})\right] \quad (4)$$

AD-SPI

For continuous tasks, AD-RL can be applied to soft policy iteration.

Soft Policy Improvement

$$\pi_{new} = \arg\min_{\pi' \in \Pi} \mathrm{KL}\left(\pi'(\cdot|x_t) \middle\| \frac{\exp(\mathbb{E}\, Q(x_t, \cdot))}{Z(x_t, \cdot)}\right)$$

AD Soft Policy Improvement (ours)

$$\pi_{new} = \arg\min_{\pi' \in \Pi} \mathrm{KL}\left(\pi'(\cdot|x_t) \middle\| \frac{\exp(\mathbb{E}_{x_t^\tau \sim b_\Delta(\cdot|x_t)}[Q^\tau(x_t^\tau, \cdot)])}{Z(x_t^\tau, \cdot)}\right) \quad (5)$$

AD-RL Pseudocode

Algorithm 1 Auxiliary-Delayed Reinforcement Learning

Input: Q, π for Δ delays; Q^τ, π^τ for Δ^τ delays
for each update step **do**
  # Learning Δ^τ-delayed task
  Updating Q^τ, π^τ by a given delayed RL method
  Bootstrapping Q based on Q^τ via Eq. (3) # discrete
  Improving π based on Q^τ via Eq. (5) # continuous
end for
Output: Q, π

Theoretical Analysis: Sample Complexity

Sample Complexity

The sample complexity of the optimized Q-learning is $\mathcal{O}\left(\frac{\log(|\mathcal{S}||\mathcal{A}|)}{\epsilon^{2.5}(1-\gamma)^5}\right)$ [7].

The sample complexity of augmented Q-learning in the augmented state space with delays Δ is $\mathcal{O}\left(\frac{\log(|\mathcal{S}||\mathcal{A}|^{\Delta+1})}{\epsilon^{2.5}(1-\gamma)^5}\right)$.

AD-RL performs bootstrapping in the auxiliary Δ^{τ}-augmented state-space instead of the original Δ-augmented state-space, the sample efficiency is improved by $\mathcal{O}(|\mathcal{A}|^{\Delta-\Delta_{\tau}})$.

Theoretical Analysis: Performance Gap

Theorem 4.3 Delayed Q-value Difference Bound

Specially, for optimal policies $\pi_{(*)}$ and $\pi^{\tau}_{(*)}$, if $Q^{\tau}_{(*)}$ is L_Q-LC, the corresponding optimal Q-value difference can be bounded as follow

$$\left\| \mathop{\mathbb{E}}_{x_t^{\tau} \sim b_{\Delta}(\cdot|x_t)} \left[Q^{\tau}_{(*)}(x_t^{\tau}, a_t) \right] - Q_{(*)}(x_t, a_t) \right\|_{\infty}$$

$$\leq \frac{\gamma^2 L_Q}{(1-\gamma)^2} \mathop{\mathbb{E}}_{\substack{\hat{x}^{\tau} \sim b_{\Delta}(\cdot|\hat{x}) \\ \hat{x} \sim d^{\pi}_{x_{t+1}} \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t) \\ a_t \sim \pi_{(*)}(\cdot|x_t)}} \left[\mathcal{W}_1 \left(\pi^{\tau}_{(*)}(\cdot|\hat{x}^{\tau}) \middle\| \pi_{(*)}(\cdot|\hat{x}) \right) \right]$$

Theoretical Analysis: Performance Gap

Remark 4.4 Deterministic MDP Case

For a deterministic MDP, b_Δ is also deterministic and becomes an injection function, meaning that given the x, the x^τ is determined. And then due to $\pi_{(*)}(\cdot|x) = \mathbb{E}_{x^\tau \sim b_\Delta(\cdot|x)}\left[\pi^\tau_{(*)}(\cdot|x^\tau)\right]$, we have

$$\mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{(*)}(x_t^\tau, a_t)\right] = Q_{(*)}(x_t, a_t)$$

Remark 4.5 Stochastic MDP Case

In the case of stochastic MDP, the performance gap might become larger as the difference between Δ and Δ^τ increases. Using a moderate auxiliary delays Δ^τ could trade-off the sample efficiency (closer to 0) and performance consistency (closer to Δ).

Theoretical Analysis: Convergence Analysis

Theorem 4.6 AD-VI Convergence Guarantee

Consider the bellman operator $\mathcal{T}$ in Eq. (3) and the initial Q-function $Q_{(0)}$: $\mathcal{X} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define a sequence $\{Q_{(k)}\}_{k=0}^{\infty}$ where $Q_{(k+1)} = \mathcal{T} Q_{(k)}$. As $k \to \infty$, $Q_{(k)}$ will converge to the fixed point $Q_{(\approx)}$ with Q^{τ} converges to $Q^{\tau}_{(*)}$. And for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$, we have

$$Q_{(\approx)}(x_t, a_t) = \mathop{\mathbb{E}}_{x_t^{\tau} \sim b_{\Delta}(\cdot|x_t)} \left[Q^{\tau}_{(*)}(x_t^{\tau}, a_t) \right]$$

Theoretical Analysis: Convergence Analysis

Lemma 4.7 Policy Evaluation Convergence Guarantee

Consider the soft bellman operator $\mathcal{T}^{\pi}$ in Eq. (4) and the initial Q-value function $Q_{(0)}$: $\mathcal{X} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define a sequence $\{Q_{(k)}\}_{k=0}^{\infty}$ where $Q_{(k+1)} = \mathcal{T}^{\pi} Q_{(k)}$. Then for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$, as $k \to \infty$, $Q_{(k)}(x_t, a_t)$ will converge to the fixed point

$$\mathop{\mathbb{E}}_{x_t^{\tau} \sim b_{\Delta}(\cdot|x_t)} [Q_{soft}^{\tau}(x_t^{\tau}, a_t)] - \log \pi(a_t|x_t)$$

Lemma 4.8 Policy Improvement Guarantee

Consider the policy update rule in Eq. (5), and let π_{old}, π_{new} be the old policy and new policy improved from the old one, respectively. Then for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$, we have

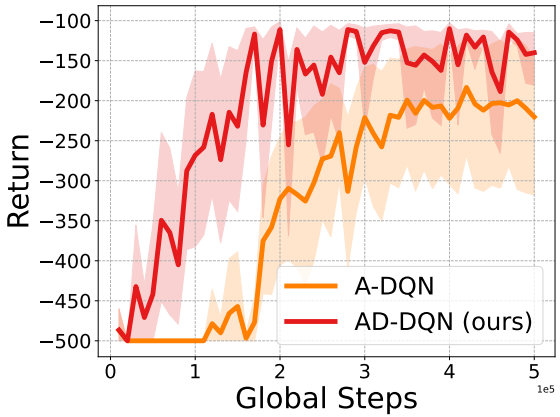
$$Q_{old}(x_t, a_t) \leq Q_{new}(x_t, a_t).$$

Theoretical Analysis: Convergence Analysis

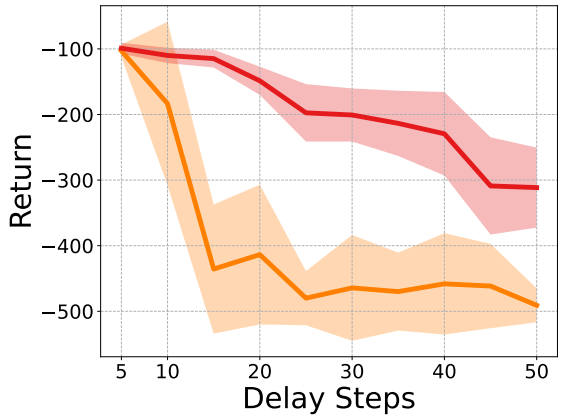
Theorem 4.9 AD-SPI Convergence Guarantee

Applying policy evaluation in Eq. (4) and policy improvement in Eq. (5) repeatedly to any given policy $\pi \in \Pi$, it converges to $\pi_{(\approx)}$ such that $Q^{\pi}(x_t, a_t) \leq Q^{\pi}_{(\approx)}(x_t, a_t)$ for any $\pi \in \Pi$, $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.

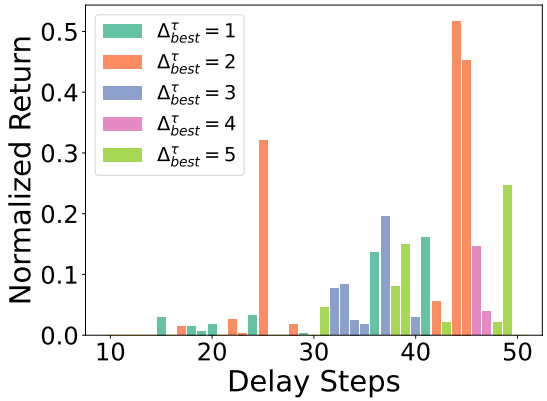
Experimental Results



(a) deterministic 10 delays



(b) deterministic varying delays



(c) stochastic performance of $\Delta^{\tau}_{\text{best}}$

Figure: Results on deterministic Acrobot include (a) learning curves with 10 different delay values, and (b) final performance across varying delays from 5 to 50. For stochastic Acrobot, (c) shows the normalized returns at the optimal auxiliary delay Δ^{τ}_{best} for delays ranging from 10 to 50. Different colors correspond to distinct best returns achieved by different optimal delays Δ^{τ}_{best}.

Experimental Results

Table: Normalized Performance on the MuJoCo tasks with 5, 25 and 50 delays.

Task	Delays	A-SAC	DC/AC	DIDA	BPQL	AD-SAC (ours)
Ant-v4	5	$0.18_{\pm 0.01}$	$0.25_{\pm 0.05}$	$0.89_{\pm 0.03}$	$0.96_{\pm 0.03}$	$0.72_{\pm 0.25}$
	25	$0.07_{\pm 0.07}$	$0.19_{\pm 0.02}$	$0.29_{\pm 0.07}$	$0.57_{\pm 0.11}$	$0.66_{\pm 0.04}$
	50	$0.02_{\pm 0.04}$	$0.19_{\pm 0.02}$	$0.19_{\pm 0.05}$	$0.38_{\pm 0.07}$	$0.48_{\pm 0.06}$
HalfCheetah-v4	5	$0.35_{\pm 0.15}$	$0.40_{\pm 0.23}$	$0.90_{\pm 0.01}$	$1.00_{\pm 0.06}$	$1.07_{\pm 0.06}$
	25	$0.04_{\pm 0.01}$	$0.16_{\pm 0.07}$	$0.12_{\pm 0.03}$	$0.87_{\pm 0.04}$	$0.71_{\pm 0.12}$
	50	$0.12_{\pm 0.17}$	$0.12_{\pm 0.13}$	$0.15_{\pm 0.03}$	$0.73_{\pm 0.17}$	$0.74_{\pm 0.10}$
Hopper-v4	5	$1.02_{\pm 0.28}$	$1.16_{\pm 0.25}$	$0.40_{\pm 0.40}$	$1.25_{\pm 0.09}$	$1.07_{\pm 0.30}$
	25	$0.13_{\pm 0.04}$	$0.19_{\pm 0.04}$	$0.27_{\pm 0.08}$	$1.21_{\pm 0.18}$	$0.86_{\pm 0.25}$
	50	$0.04_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.09_{\pm 0.01}$	$0.71_{\pm 0.13}$	$0.72_{\pm 0.03}$
Humanoid-v4	5	$0.13_{\pm 0.02}$	$0.59_{\pm 0.17}$	$0.08_{\pm 0.04}$	$0.96_{\pm 0.05}$	$0.98_{\pm 0.07}$
	25	$0.05_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.07_{\pm 0.00}$	$0.12_{\pm 0.01}$	$0.25_{\pm 0.16}$
	50	$0.04_{\pm 0.01}$	$0.03_{\pm 0.01}$	$0.07_{\pm 0.00}$	$0.08_{\pm 0.01}$	$0.10_{\pm 0.01}$
HumanoidStandup-v4	5	$1.02_{\pm 0.08}$	$1.16_{\pm 0.12}$	$1.00_{\pm 0.00}$	$1.13_{\pm 0.07}$	$1.22_{\pm 0.03}$
	25	$0.97_{\pm 0.09}$	$1.03_{\pm 0.03}$	$0.97_{\pm 0.02}$	$1.09_{\pm 0.05}$	$1.15_{\pm 0.08}$
	50	$0.90_{\pm 0.02}$	$1.02_{\pm 0.07}$	$0.89_{\pm 0.06}$	$1.06_{\pm 0.04}$	$1.12_{\pm 0.02}$
Pusher-v4	5	$1.11_{\pm 0.02}$	$1.29_{\pm 0.05}$	$1.01_{\pm 0.01}$	$1.06_{\pm 0.08}$	$1.36_{\pm 0.01}$
	25	$0.49_{\pm 0.32}$	$1.12_{\pm 0.02}$	$1.04_{\pm 0.01}$	$1.07_{\pm 0.06}$	$1.29_{\pm 0.03}$
	50	$0.00_{\pm 0.05}$	$1.13_{\pm 0.01}$	$1.04_{\pm 0.02}$	$1.09_{\pm 0.05}$	$1.23_{\pm 0.02}$
Reacher-v4	5	$0.97_{\pm 0.01}$	$1.02_{\pm 0.00}$	$1.03_{\pm 0.00}$	$1.00_{\pm 0.01}$	$1.03_{\pm 0.01}$
	25	$0.96_{\pm 0.02}$	$1.00_{\pm 0.00}$	$0.98_{\pm 0.01}$	$0.87_{\pm 0.05}$	$0.98_{\pm 0.02}$
	50	$0.86_{\pm 0.02}$	$0.89_{\pm 0.01}$	$0.93_{\pm 0.02}$	$0.90_{\pm 0.02}$	$0.91_{\pm 0.03}$
Swimmer-v4	5	$0.88_{\pm 0.09}$	$1.11_{\pm 0.30}$	$1.05_{\pm 0.01}$	$0.97_{\pm 0.02}$	$1.82_{\pm 0.78}$
	25	$0.72_{\pm 0.02}$	$0.78_{\pm 0.12}$	$0.93_{\pm 0.09}$	$1.36_{\pm 0.56}$	$2.52_{\pm 0.40}$
	50	$0.69_{\pm 0.04}$	$0.68_{\pm 0.06}$	$0.87_{\pm 0.03}$	$2.23_{\pm 0.55}$	$2.71_{\pm 0.14}$
Walker2d-v4	5	$0.76_{\pm 0.21}$	$0.85_{\pm 0.12}$	$0.61_{\pm 0.07}$	$1.20_{\pm 0.11}$	$1.12_{\pm 0.09}$
	25	$0.12_{\pm 0.02}$	$0.26_{\pm 0.08}$	$0.10_{\pm 0.02}$	$0.59_{\pm 0.30}$	$0.72_{\pm 0.11}$
	50	$0.11_{\pm 0.02}$	$0.11_{\pm 0.02}$	$0.08_{\pm 0.01}$	$0.23_{\pm 0.10}$	$0.23_{\pm 0.11}$

AD-RL: Limitation

Limitation of AD-RL

While AD-RL improves efficiency in augmentation-based frameworks, it still relies on operating RL processes within the augmented state space, resulting in limited sample complexity. In addition, auxiliary delay step choosing could be tricky under some stochastic cases. Next, we introduce Variational Delayed Policy Optimization (VDPO), which formulates delayed RL as a variational inference problem.

Variational Delayed Policy Optimization (VDPO)

- VDPO improves sample efficiency by formulating delayed RL as a variational inference problem.
- Theoretically, VDPO offers better sample complexity and shares a consistent convergence point with existing methods.
- Empirically, VDPO achieves superior sample efficiency while comparable performance to existing SOTAs.

Background of Variational RL

Variational RL aims to find the policy π with the highest log evidence

$$\max_{\pi} \log p_{\pi}(O = 1),$$

where $O = 1$ is the optimality of the task (e.g., the trajectory τ obtains the maximum return).

$$\begin{aligned} \log p_{\pi}(O = 1) &\geq \mathbb{E}_{\tau \sim q(\tau)} \left[\log p(O = 1|\tau)\right] - \mathrm{KL}(q(\tau)||p_{\pi}(\tau)) \\ &= \mathrm{ELBO}(\pi, q), \end{aligned}$$

where KL is the Kullback-Leibler (KL) divergence, $q(\tau)$ is prior trajectory distribution and $\mathrm{ELBO}(\pi, q)$ is the evidence lower bound (ELBO).

Delayed RL as Variational Inference

The optimization objective of delayed RL is

$$\max_{\pi_\Delta} \log p_{\pi_\Delta}(O=1) = \max_{\pi_\Delta} \log \int p(O=1|\tau) p_{\pi_\Delta}(\tau) \mathrm{d}\tau, \tag{6}$$

where $p_{\pi_\Delta}(O=1)$ is the probability of the optimality of the delayed policy π_Δ, and $p_{\pi_\Delta}(\tau)$ is the trajectory distribution induced by π_Δ. The ELBO for optimization objective is

$$\begin{aligned} \log p_{\pi_\Delta}(O=1) &\geq \underbrace{\mathbb{E}_{\tau\sim p_\pi(\tau)}\left[\log p(O=1|\tau)\right]}_{A\uparrow} - \underbrace{\mathrm{KL}(p_\pi(\tau)||p_{\pi_\Delta}(\tau))}_{B\downarrow} \\ &= \mathrm{ELBO}(\pi, \pi_\Delta), \end{aligned} \tag{7}$$

where $p_\pi(\tau)$ is the trajectory distribution induced by an newly-introduced *reference policy* π.

VDPO

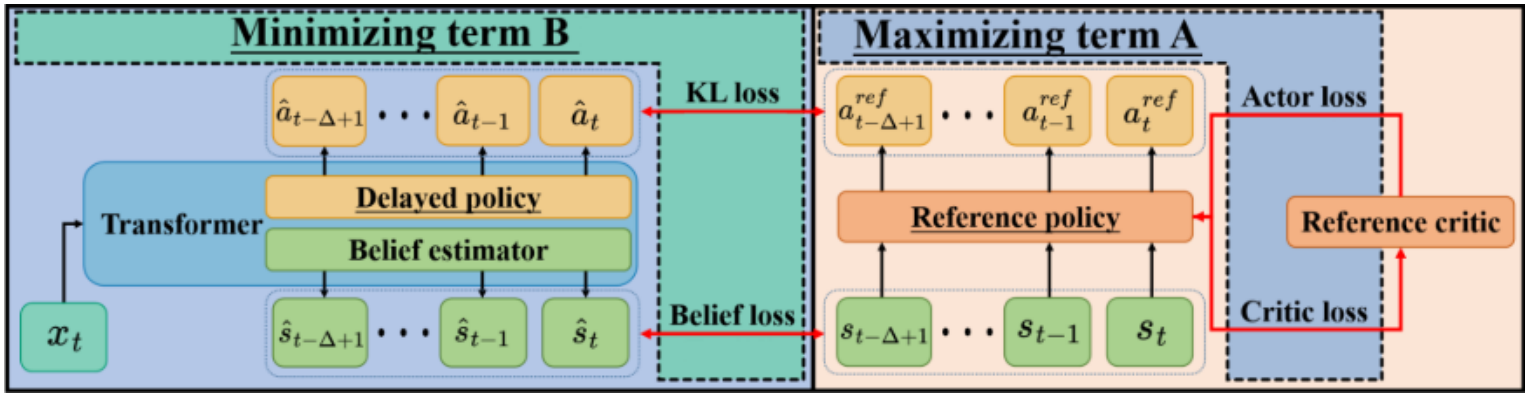


Figure: The pipeline of VDPO.

VDPO

Maximizing term A in ELBO

Maximizing the performance of the delay-free reference policy π by TD Learning.

$$\max_{\pi} \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\log p(O=1|\tau)\right] = \max_{\pi} \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\mathcal{J}(\tau)\right].$$

Lemma 5.1 Performance in Delayed MDP, Theorem 4.3.1 in [8]

Let $\mathcal{M}_1, \mathcal{M}_2$ be two constant delayed MDPs with respective delays $\Delta_1, \Delta_2 (\Delta_1 < \Delta_2)$. For the optimal policies in $\mathcal{M}_1, \mathcal{M}_2$, we have $\mathcal{J}_1^* \geq \mathcal{J}_2^*$.

Lemma 5.2 Sample complexity of model-based policy iteration, Theorem 2 in [9]

Let $\mathcal{M}$ be the constant delayed MDP with delays Δ. Model-based policy iteration finds an ϵ-optimal policy with probability $1-\sigma$ using sample size $\mathcal{O}\left(\frac{|\mathcal{X}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2} \ln \frac{1}{\sigma}\right)$, where $|\mathcal{X}| = |\mathcal{S}||\mathcal{A}|^{\Delta}$.

VDPO

Minimizing term B in ELBO

Minimizing the behaviour difference $\mathrm{KL}(\pi(a_t|s_t)||\pi_\Delta(a_t|x_t))$ by (state-level) imitation learning.

Proposition 5.3 State-level KL divergence

For a fixed reference policy π, the trajectory-level KL divergence can be reformulated to state-level KL divergence as follows.

$$\mathrm{KL}(p_\pi(\tau)||p_{\pi_\Delta}(\tau)) = \underbrace{\sum_{t=0}^{\infty}\int d^\pi(s_t)\mathrm{KL}(\pi(a_t|s_t)||\pi_\Delta(a_t|x_t))\mathrm{d}s_t}_{\text{State-level KL divergence}} + \mathit{Const.}, \tag{8}$$

$$\text{where } \mathit{Const.} = \mathrm{KL}(\rho(s_0)||\rho_\Delta(x_0))$$

$$+\sum_{t=0}^{\infty}\int d^\pi(s_t)\int \pi(a_t|s_t)\mathrm{KL}(\mathcal{P}(s_{t+1}|s_t,a_t)||b(s_t|x_t)\mathcal{P}_\Delta(x_{t+1}|x_t,a_t))\mathrm{d}a_t\mathrm{d}s_t.$$

Theoretical Analysis: Sample Complexity

Lemma 5.4 Sample complexity of behaviour cloning, Theorem 15.3 in [10]

Given the demonstration from the optimal policy, behaviour cloning finds an ϵ-optimal policy with probability $1-\sigma$ using sample size $\mathcal{O}\left(\frac{|\mathcal{X}|\ln|\mathcal{A}|}{(1-\gamma)^4\epsilon^2}\sigma\right)$.

Lemma 5.5 Sample complexity of VDPO

Assumed that maximizing A in Eq. (7) by model-based policy iteration while minimizing B in Eq. (7) by behaviour cloning, VDPO finds an ϵ-optimal policy with probability $1-\sigma$ using sample size

$$\mathcal{O}\left(\max\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}\ln\frac{1}{\sigma}, \frac{|\mathcal{X}|\ln|\mathcal{A}|}{(1-\gamma)^4\epsilon^2}\sigma\right)\right).$$

Theoretical Analysis: Sample Complexity

Proposition 5.6 Sample complexity comparison

In the delayed MDP, as $\sigma \to 0$, the sample complexity of VDPO is less than or equal to the sample complexity of model-based policy iteration:

$$\mathcal{O}\left(\max\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}\ln\frac{1}{\sigma}, \frac{|\mathcal{X}|\ln|\mathcal{A}|}{(1-\gamma)^4\epsilon^2}\sigma\right)\right) \leq \mathcal{O}\left(\frac{|\mathcal{X}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}\ln\frac{1}{\sigma}\right).$$

Theoretical Analysis: Performance Analysis

Lemma 5.7 Convergence of delayed policy in VDPO

Let π^* be the optimal reference policy which is trained by a delay-free RL algorithm. The delayed policy π_Δ converges to π^*_Δ satisfying that

$$\pi^*_\Delta(a_t|x_t) = \mathbb{E}_{s_t \sim b(\cdot|x_t)} \left[\pi^*(a_t|s_t)\right], \forall x_t \in \mathcal{X}. \tag{9}$$

Proposition 5.8 Consistent fixed point

VDPO shares the same fixed point with DIDA [11], BPQL [12] and AD-SAC [13] for the same delayed MDP.

Experimental Results

Table: The amount of steps required to reach the threshold Ret_{df} in MuJoCo tasks with 5 constant delays within 1M global steps.

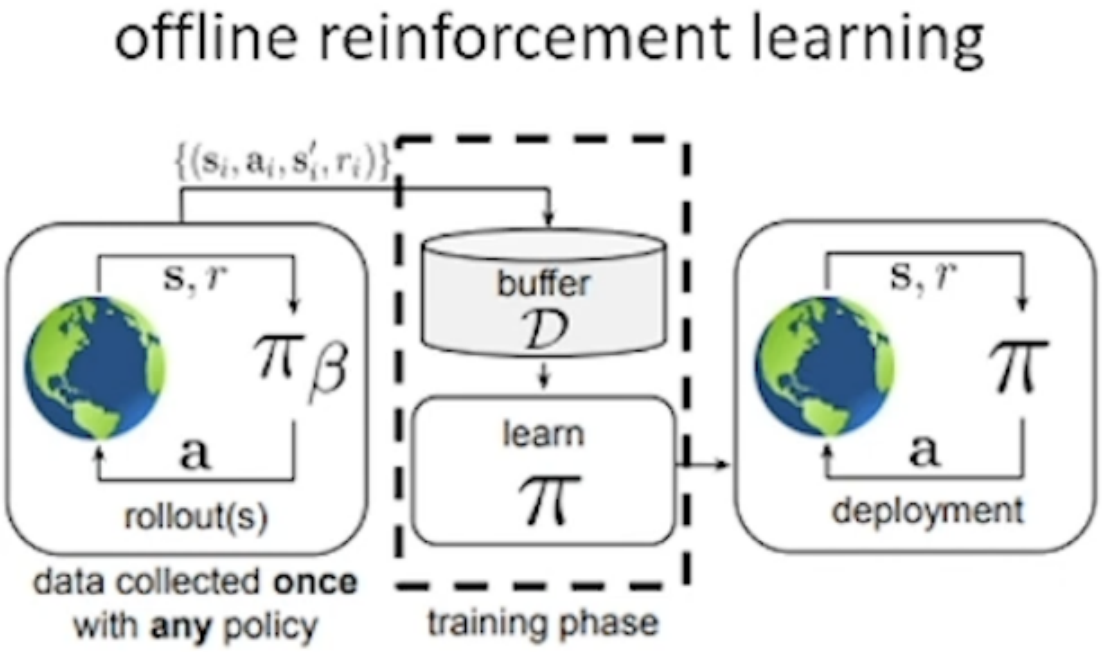
Task (Delays=5)	A-SAC	DC/AC	DIDA	BPQL	AD-SAC (ours)	VDPO (ours)
Ant-v4	×	×	×	×	×	0.42M
HalfCheetah-v4	×	×	×	0.99M	0.56M	0.44M
Hopper-v4	0.83M	0.35M	×	0.29M	0.12M	0.07M
Humanoid-v4	×	×	×	×	×	0.67M
HumanoidStandup-v4	0.64M	0.35M	0.10M	0.09M	0.14M	0.14M
Pusher-v4	0.17M	0.02M	0.10M	0.27M	0.04M	0.01M
Reacher-v4	×	0.61M	0.10M	0.90M	0.44M	0.77M
Swimmer-v4	×	0.94M	0.10M	×	0.13M	0.07M
Walker2d-v4	×	×	×	0.52M	0.67M	0.25M

Experimental Results

Table: Normalized Performance on MuJoCo tasks with 5, 25, and 50 constant delays for 1M global steps.

Task	Delays	A-SAC	DC/AC	DIDA	BPQL	AD-SAC (ours)	VDPO (ours)
Ant-v4	5	$0.18_{\pm 0.01}$	$0.25_{\pm 0.05}$	$0.89_{\pm 0.03}$	$0.96_{\pm 0.03}$	$0.72_{\pm 0.25}$	$1.11_{\pm 0.04}$
	25	$0.07_{\pm 0.07}$	$0.19_{\pm 0.02}$	$0.29_{\pm 0.07}$	$0.57_{\pm 0.11}$	$0.66_{\pm 0.04}$	$0.56_{\pm 0.06}$
	50	$0.02_{\pm 0.04}$	$0.19_{\pm 0.02}$	$0.19_{\pm 0.05}$	$0.38_{\pm 0.07}$	$0.48_{\pm 0.06}$	$0.46_{\pm 0.07}$
HalfCheetah-v4	5	$0.35_{\pm 0.15}$	$0.40_{\pm 0.23}$	$0.90_{\pm 0.01}$	$1.00_{\pm 0.06}$	$1.07_{\pm 0.06}$	$1.03_{\pm 0.08}$
	25	$0.04_{\pm 0.01}$	$0.16_{\pm 0.07}$	$0.12_{\pm 0.03}$	$0.87_{\pm 0.04}$	$0.71_{\pm 0.12}$	$0.70_{\pm 0.17}$
	50	$0.12_{\pm 0.17}$	$0.12_{\pm 0.13}$	$0.15_{\pm 0.03}$	$0.73_{\pm 0.17}$	$0.74_{\pm 0.10}$	$0.72_{\pm 0.21}$
Hopper-v4	5	$1.02_{\pm 0.28}$	$1.16_{\pm 0.25}$	$0.40_{\pm 0.40}$	$1.25_{\pm 0.09}$	$1.07_{\pm 0.30}$	$1.22_{\pm 0.08}$
	25	$0.13_{\pm 0.04}$	$0.19_{\pm 0.04}$	$0.27_{\pm 0.08}$	$1.21_{\pm 0.18}$	$0.86_{\pm 0.25}$	$0.82_{\pm 0.40}$
	50	$0.04_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.09_{\pm 0.01}$	$0.71_{\pm 0.13}$	$0.72_{\pm 0.03}$	$0.22_{\pm 0.04}$
Humanoid-v4	5	$0.13_{\pm 0.02}$	$0.59_{\pm 0.17}$	$0.08_{\pm 0.04}$	$0.96_{\pm 0.05}$	$0.98_{\pm 0.07}$	$1.15_{\pm 0.07}$
	25	$0.05_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.07_{\pm 0.00}$	$0.12_{\pm 0.01}$	$0.25_{\pm 0.16}$	$0.12_{\pm 0.02}$
	50	$0.04_{\pm 0.01}$	$0.03_{\pm 0.01}$	$0.07_{\pm 0.00}$	$0.08_{\pm 0.01}$	$0.10_{\pm 0.01}$	$0.12_{\pm 0.00}$
HumanoidStandup-v4	5	$1.02_{\pm 0.08}$	$1.16_{\pm 0.12}$	$1.00_{\pm 0.00}$	$1.13_{\pm 0.07}$	$1.22_{\pm 0.03}$	$1.29_{\pm 0.02}$
	25	$0.97_{\pm 0.09}$	$1.03_{\pm 0.03}$	$0.97_{\pm 0.02}$	$1.09_{\pm 0.05}$	$1.15_{\pm 0.08}$	$1.13_{\pm 0.12}$
	50	$0.90_{\pm 0.02}$	$1.02_{\pm 0.07}$	$0.89_{\pm 0.06}$	$1.06_{\pm 0.04}$	$1.12_{\pm 0.02}$	$1.04_{\pm 0.16}$
Pusher-v4	5	$1.11_{\pm 0.02}$	$1.29_{\pm 0.05}$	$1.01_{\pm 0.01}$	$1.06_{\pm 0.08}$	$1.36_{\pm 0.01}$	$1.17_{\pm 0.06}$
	25	$0.49_{\pm 0.32}$	$1.12_{\pm 0.02}$	$1.04_{\pm 0.01}$	$1.07_{\pm 0.06}$	$1.29_{\pm 0.03}$	$1.31_{\pm 0.07}$
	50	$0.00_{\pm 0.05}$	$1.13_{\pm 0.01}$	$1.04_{\pm 0.02}$	$1.09_{\pm 0.05}$	$1.23_{\pm 0.02}$	$1.33_{\pm 0.05}$
Reacher-v4	5	$0.97_{\pm 0.01}$	$1.02_{\pm 0.00}$	$1.03_{\pm 0.00}$	$1.00_{\pm 0.01}$	$1.03_{\pm 0.01}$	$1.02_{\pm 0.03}$
	25	$0.96_{\pm 0.02}$	$1.00_{\pm 0.00}$	$0.98_{\pm 0.01}$	$0.87_{\pm 0.05}$	$0.98_{\pm 0.02}$	$1.02_{\pm 0.03}$
	50	$0.86_{\pm 0.02}$	$0.89_{\pm 0.01}$	$0.93_{\pm 0.02}$	$0.90_{\pm 0.02}$	$0.91_{\pm 0.03}$	$1.02_{\pm 0.03}$
Swimmer-v4	5	$0.88_{\pm 0.09}$	$1.11_{\pm 0.30}$	$1.05_{\pm 0.01}$	$0.97_{\pm 0.02}$	$1.82_{\pm 0.78}$	$2.30_{\pm 0.36}$
	25	$0.72_{\pm 0.02}$	$0.78_{\pm 0.12}$	$0.93_{\pm 0.09}$	$1.36_{\pm 0.56}$	$2.52_{\pm 0.40}$	$2.35_{\pm 0.27}$
	50	$0.69_{\pm 0.04}$	$0.68_{\pm 0.06}$	$0.87_{\pm 0.03}$	$2.23_{\pm 0.55}$	$2.71_{\pm 0.14}$	$2.42_{\pm 0.22}$
Walker2d-v4	5	$0.76_{\pm 0.21}$	$0.85_{\pm 0.12}$	$0.61_{\pm 0.07}$	$1.20_{\pm 0.11}$	$1.12_{\pm 0.09}$	$1.27_{\pm 0.04}$
	25	$0.12_{\pm 0.02}$	$0.26_{\pm 0.08}$	$0.10_{\pm 0.02}$	$0.59_{\pm 0.30}$	$0.72_{\pm 0.11}$	$0.27_{\pm 0.11}$
	50	$0.11_{\pm 0.02}$	$0.11_{\pm 0.02}$	$0.08_{\pm 0.01}$	$0.23_{\pm 0.10}$	$0.23_{\pm 0.11}$	$0.11_{\pm 0.03}$

What about more realistic Offline Setting?



- Simulation or offline dataset are normally perfectly delay-free.
- How can we learn delay-adapted policy from delay-free offline dataset with prior delay knowledge?

DT-CORL (**D**elay-**T**ransformer belief policy **C**onstrained **O**ffline **RL**)

- **Problem setting:** Learn delay-adapted control policies from **delay-free** offline datasets, given prior delay information at deployment.
- **Core idea:** Use a transformer-based belief model to recover delayed latent states from historical trajectory tokens.
- **Joint learning:** Couple belief estimation with policy/value optimization to keep representation and control objectives aligned.
- **Offline safety:** Add a behavior-regularized policy constraint to reduce out-of-distribution action extrapolation.
- **Empirical outcome:** Strong robustness under deterministic and stochastic delays, including high-dimensional Adroit tasks.

Formulation (1/2): Augmented Delayed Constrained RL

Start from delayed information state:

$$x_t = \{s_{t-\Delta}, a_{t-\Delta:t-1}, r_{t-\Delta:t-1}\}.$$

Augmented-state formulation

The delayed state lies in an augmented space:

$$\mathcal{X} = \mathcal{S} \times \mathcal{A}^{\Delta} \times \mathcal{R}^{\Delta}.$$

Constrained offline optimization on augmented states:

$$\max_{\pi} \ \mathbb{E}_{x_t \sim \mathcal{D}}[Q(x_t, \pi(\cdot \mid x_t))] - \alpha \, \mathcal{D}(\pi \| \pi_\beta).$$

Bottleneck

- As delay Δ grows, optimization in $\mathcal{X}$ becomes high-dimensional.
- Constraint learning becomes less stable and less sample-efficient.

Formulation (2/2): Belief-based DT-CORL

Step 1: Belief inference

Infer compact latent state from delayed history:

$$\hat{s}_t \sim b_\theta(\cdot \mid x_t), \qquad \hat{s}_t \in \mathcal{S}.$$

- Transformer captures long-range delayed dependencies.
- Belief training minimizes delayed sequence prediction errors.

Step 2: Constrained policy learning

Optimize policy/value directly on belief states:

$$\max_{\pi} \ \mathbb{E}_{\hat{s}_t \sim b_\theta}[Q(\hat{s}_t, \pi(\cdot \mid \hat{s}_t))] - \alpha \, \mathcal{D}(\pi \| \pi_\beta)$$

- Keeps optimization in compact space $\mathcal{S}$.
- Better bridges delay-free offline data and delayed deployment.

Framework

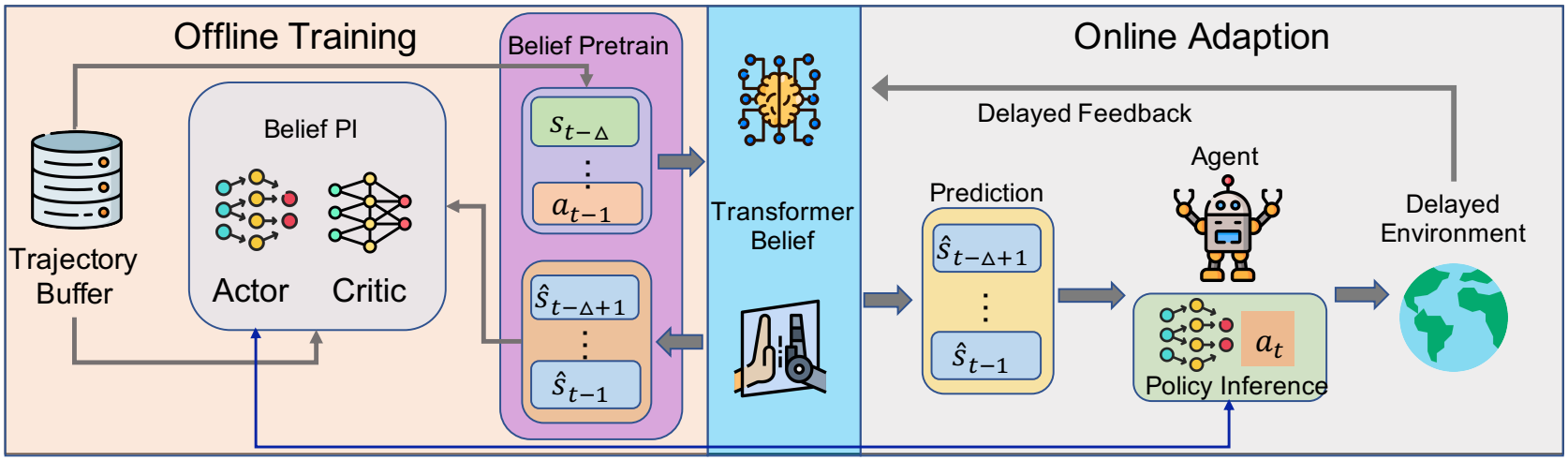


Figure: Transformer belief-based offline policy constrained RL training paradigm

Training loop: alternating (or joint) belief-model updates and constrained actor-critic updates.

Experimental Results(Offline)

Table 2: DT-CORL vs. belief-based baselines (Belief-CQL, Belief-IQL) on D4RL MuJoCo tasks. Normalized returns (%). Delays: deterministic $\Delta \in \{4, 8, 16\}$, stochastic $\Delta \sim \mathcal{U}(1, k)$, $k \in \{4, 8, 16\}$. Best results per column are shown in **bold** with light blue background.

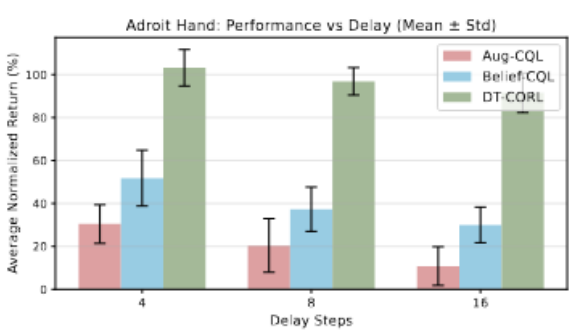
Task	Method	medium			med-expert			med-replay			expert		
		4	8	16	4	8	16	4	8	16	4	8	16
Deterministic Delays													
Hopper	IQL	5.3	5.8	4.8	5.5	6.4	5.3	5.8	6.4	2.4	4.5	9.0	5.6
	CQL	8.2	4.3	4.3	7.3	3.4	5.9	8.6	4.4	6.1	7.9	3.9	4.1
	Belief-IQL	27.7	29.3	25.4	26.7	26.6	24.7	24.7	25.5	23.4	18.7	17.4	15.9
	Belief-CQL	75.4	56.8	42.9	92.9	39.5	35.2	**110.1**	99.7	96.6	81.1	43.3	45.9
	DT-CORL	**79.4**	**85.0**	**71.8**	**113.0**	**112.2**	**109.9**	99.4	**100.8**	**100.2**	**112.9**	**113.1**	**112.2**
HalfCheetah	Belief-IQL	30.8	10.6	5.3	24.8	6.1	3.3	23.3	13.8	9.7	6.8	4.8	3.6
	Belief-CQL	**49.2**	8.9	3.0	22.7	6.5	1.5	36.1	14.4	6.4	1.5	1.5	1.5
	DT-CORL	47.4	**27.8**	**6.4**	**44.7**	**21.3**	**8.7**	**43.6**	**27.1**	**7.9**	**20.6**	**5.1**	**5.2**
Walker2d	Belief-IQL	33.4	25.7	24.6	49.6	17.3	16.4	28.2	20.6	18.3	25.5	19.9	16.7
	Belief-CQL	87.0	64.1	39.2	105.8	99.5	51.0	93.3	**93.5**	61.0	**111.1**	110.8	97.7
	DT-CORL	**87.4**	**87.6**	**86.8**	**112.1**	**112.0**	**118.1**	**93.6**	90.5	**88.1**	110.9	**111.2**	**110.5**
Stochastic Delays													
Hopper	IQL	8.3	6.0	12.9	9.4	6.4	10.4	8.6	4.0	2.9	11.4	3.1	11.6
	CQL	8.9	4.3	11.1	7.4	6.0	11.5	6.1	5.3	9.1	6.9	3.4	4.2
	Belief-IQL	27.4	27.4	28.4	28.3	27.9	23.8	25.0	24.5	26.0	18.6	16.7	16.9
	Belief-CQL	**80.8**	**67.8**	74.3	91.8	48.9	57.5	**100.8**	99.8	**99.2**	72.3	35.4	20.1
	DT-CORL	78.5	72.1	**79.3**	**113.6**	**112.7**	**85.4**	99.4	**100.1**	98.8	**113.2**	**112.8**	**113.1**
HalfCheetah	Belief-IQL	33.4	31.2	21.6	31.1	16.3	8.2	24.9	20.0	15.3	13.1	9.2	4.3
	Belief-CQL	**52.6**	46.6	16.2	48.0	22.1	3.6	**47.1**	41.4	19.7	6.5	2.8	1.7
	DT-CORL	48.2	**47.5**	**38.4**	**70.0**	**44.3**	**31.7**	47.7	**43.3**	**30.4**	**85.1**	**12.7**	**5.8**
Walker2d	Belief-IQL	34.6	37.5	36.7	49.4	25.0	20.4	31.4	25.4	24.2	43.1	27.7	20.7
	Belief-CQL	84.8	81.7	78.9	112.1	106.5	85.1	**94.9**	**97.7**	**95.3**	**111.1**	**111.1**	109.6
	DT-CORL	**86.8**	**87.4**	**87.0**	**114.1**	**113.6**	**111.5**	93.0	90.9	91.8	110.9	110.9	**110.5**

Experimental Results

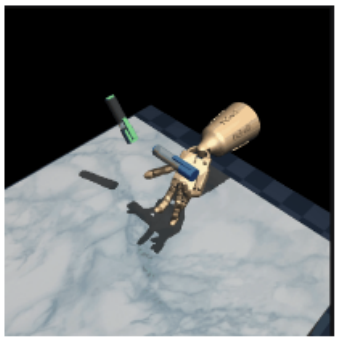
Table 1: Normalized returns (%) on D4RL AntMaze tasks under deterministic and stochastic observation delays $\Delta \in \{4, 8, 16\}$. Results are averaged over 3 seeds. Best per column (including ties) is shown in **bold** with a light blue background.

Setting	Method	umaze			umaze-diverse			medium-play			large-play		
		4	**8**	**16**	**4**	**8**	**16**	**4**	**8**	**16**	**4**	**8**	**16**
Deterministic	DBPT-SAC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0
	Augmented-BC	60.7	62.3	31.0	**69.0**	58.7	30.0	0.0	1.67	1.0	**0.0**	0.3	0.0
	Augmented-CQL	72.3	44.0	12.7	27.7	23.7	22.0	0.33	1.67	**4.67**	**0.0**	**1.33**	0.67
	Augmented-COMBO	76.0	37.0	13.3	26.0	19.0	23.0	**6.33**	**5.67**	3.0	**0.0**	**1.33**	**1.0**
	DT-CORL	**83.3**	**76.7**	**40.0**	65.3	**62.0**	**32.0**	1.33	2.33	2.33	**0.0**	0.33	0.67
Stochastic	DBPT-SAC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Augmented-BC	60.3	55.7	24.7	59.7	51.7	40.7	1.33	2.33	2.0	0.0	0.67	0.0
	Augmented-CQL	61.3	37.0	12.7	29.3	17.3	13.3	8.0	**9.67**	**5.67**	**1.33**	1.0	0.33
	Augmented-COMBO	64.7	36.7	13.3	32.7	15.0	12.7	**8.67**	7.67	3.67	0.67	1.33	1.0
	DT-CORL	**88.3**	**88.0**	**67.3**	**74.0**	**58.3**	**56.0**	2.00	2.67	3.33	0.0	**1.67**	**1.67**

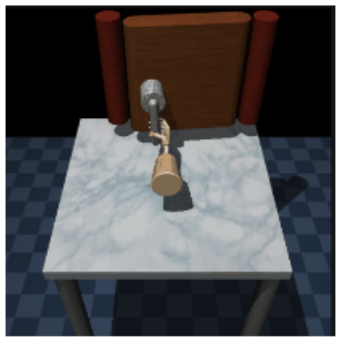
Experimental Results



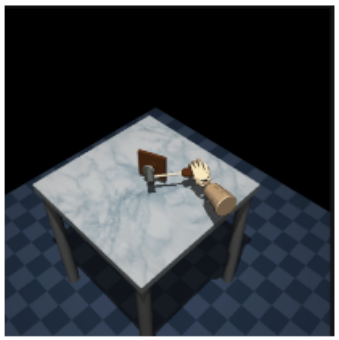
(a) Average performance



(b) Pen rotation



(c) Open door

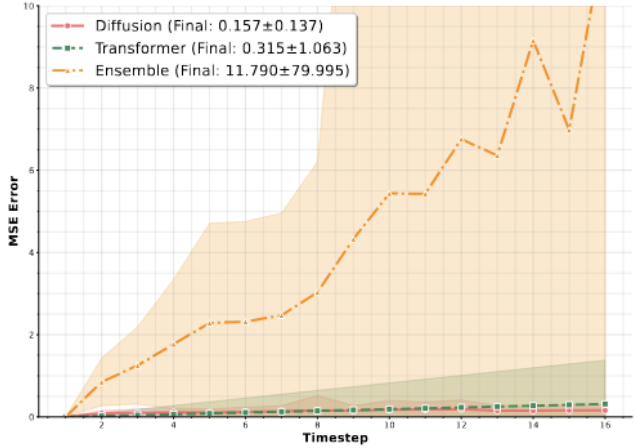


(d) Hammer smash

Figure 3: (a) Describes the average performance of *Aug-CQL*, *Belief-CQL*, and *DT-CORL* across three dexterous hand manipulation tasks (a)-(c) under various delay setting ranging from 4 to 16.

Experimental Results(Online)

Figure 2: Step-by-step detailed comparison of prediction accuracy for different models.

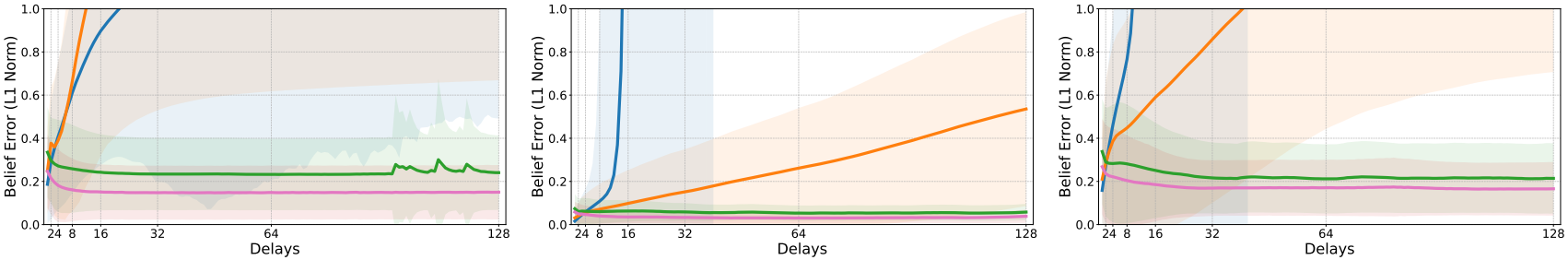


Model	Total Parameters (M)	Inference (ms)
Ensemble MLP	1.80	69.50±2.86
Transformer	7.87	22.27±9.02
Diffusion	3.96	665.21±55.5

Table 3: Comparison of total parameters (M) and inference speed (ms) for each model up to 16 steps.

Data	DT-CORL	Belief-CQL	Aug-CQL
25%	9.8	3.08	2.47
50%	12.0	3.80	2.80
75%	15.3	4.05	3.89
100%	27.8	8.90	3.80

Table 4: Performance of DT-CORL under different data availability levels for the `HalfCheetah-medium-v2`.



(a) HalfCheetah-v2 (b) Hopper-v2 (c) Walker2d-v2

Figure: Belief errors comparison.

Experimental Results

Table: Performance on MuJoCo with Deterministic Delays.

Task	Delays	Augmentation-based			Belief-based			
		A-SAC	BPQL	ADRL	DATS	D-Dreamer	D-SAC	DFBT-SAC (ours)
HalfCheetah-v2	8	$0.10_{\pm 0.01}$	$0.40_{\pm 0.04}$	$0.44_{\pm 0.03}$	$0.08_{\pm 0.01}$	$0.08_{\pm 0.01}$	$0.12_{\pm 0.06}$	$0.35_{\pm 0.12}$
	32	$0.02_{\pm 0.02}$	$0.40_{\pm 0.03}$	$0.26_{\pm 0.04}$	$0.11_{\pm 0.04}$	$0.08_{\pm 0.00}$	$0.08_{\pm 0.02}$	$0.42_{\pm 0.03}$
	128	$0.04_{\pm 0.06}$	$0.08_{\pm 0.13}$	$0.14_{\pm 0.02}$	$0.10_{\pm 0.08}$	$0.15_{\pm 0.05}$	$0.09_{\pm 0.04}$	$0.41_{\pm 0.03}$
Hopper-v2	8	$0.61_{\pm 0.31}$	$0.87_{\pm 0.09}$	$0.95_{\pm 0.16}$	$0.41_{\pm 0.31}$	$0.11_{\pm 0.01}$	$0.16_{\pm 0.05}$	$0.77_{\pm 0.18}$
	32	$0.11_{\pm 0.02}$	$0.89_{\pm 0.14}$	$0.73_{\pm 0.20}$	$0.07_{\pm 0.04}$	$0.11_{\pm 0.05}$	$0.11_{\pm 0.01}$	$0.68_{\pm 0.20}$
	128	$0.04_{\pm 0.01}$	$0.08_{\pm 0.02}$	$0.07_{\pm 0.01}$	$0.08_{\pm 0.01}$	$0.09_{\pm 0.03}$	$0.06_{\pm 0.01}$	$0.20_{\pm 0.03}$
Walker2d-v2	8	$0.44_{\pm 0.26}$	$1.07_{\pm 0.02}$	$0.97_{\pm 0.10}$	$0.13_{\pm 0.05}$	$0.11_{\pm 0.06}$	$0.09_{\pm 0.05}$	$0.99_{\pm 0.03}$
	32	$0.10_{\pm 0.02}$	$0.37_{\pm 0.25}$	$0.16_{\pm 0.08}$	$0.02_{\pm 0.03}$	$0.08_{\pm 0.05}$	$0.08_{\pm 0.02}$	$0.64_{\pm 0.10}$
	128	$0.06_{\pm 0.00}$	$0.07_{\pm 0.03}$	$0.08_{\pm 0.01}$	$0.02_{\pm 0.02}$	$0.08_{\pm 0.05}$	$0.11_{\pm 0.06}$	$0.40_{\pm 0.08}$

Experimental Results

Table: Performance on MuJoCo with Stochastic Delays.

Task	Delays	Augmentation-based			Belief-based			
		A-SAC	BPQL	ADRL	DATS	D-Dreamer	D-SAC	DFBT-SAC (ours)
HalfCheetah-v2	$U(1,8)$	$0.09_{\pm 0.01}$	$0.21_{\pm 0.07}$	$0.17_{\pm 0.07}$	$0.09_{\pm 0.03}$	$0.02_{\pm 0.01}$	$0.03_{\pm 0.01}$	$0.37_{\pm 0.12}$
	$U(1,32)$	$0.01_{\pm 0.00}$	$0.33_{\pm 0.07}$	$0.23_{\pm 0.02}$	$0.11_{\pm 0.04}$	$0.02_{\pm 0.00}$	$0.01_{\pm 0.01}$	$0.31_{\pm 0.16}$
	$U(1,128)$	$0.01_{\pm 0.01}$	$0.03_{\pm 0.03}$	$0.15_{\pm 0.02}$	$0.16_{\pm 0.03}$	$0.16_{\pm 0.00}$	$0.02_{\pm 0.00}$	$0.39_{\pm 0.04}$
Hopper-v2	$U(1,8)$	$0.17_{\pm 0.05}$	$0.20_{\pm 0.04}$	$0.18_{\pm 0.04}$	$0.04_{\pm 0.01}$	$0.07_{\pm 0.05}$	$0.14_{\pm 0.04}$	$0.86_{\pm 0.18}$
	$U(1,32)$	$0.05_{\pm 0.01}$	$0.07_{\pm 0.09}$	$0.05_{\pm 0.01}$	$0.05_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.03_{\pm 0.01}$	$0.43_{\pm 0.21}$
	$U(1,128)$	$0.03_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.04_{\pm 0.02}$	$0.05_{\pm 0.00}$	$0.03_{\pm 0.01}$	$0.03_{\pm 0.00}$	$0.14_{\pm 0.01}$
Walker2d-v2	$U(1,8)$	$0.36_{\pm 0.24}$	$0.40_{\pm 0.32}$	$0.41_{\pm 0.15}$	$0.07_{\pm 0.01}$	$0.07_{\pm 0.05}$	$0.12_{\pm 0.04}$	$1.11_{\pm 0.10}$
	$U(1,32)$	$0.12_{\pm 0.03}$	$0.16_{\pm 0.04}$	$0.11_{\pm 0.05}$	$0.09_{\pm 0.04}$	$0.12_{\pm 0.04}$	$0.05_{\pm 0.02}$	$0.67_{\pm 0.15}$
	$U(1,128)$	$0.06_{\pm 0.01}$	$0.06_{\pm 0.06}$	$0.04_{\pm 0.02}$	$0.10_{\pm 0.04}$	$0.15_{\pm 0.07}$	$0.03_{\pm 0.04}$	$0.30_{\pm 0.13}$

Conclusion

Augmentation-based methods grow high-dimensional states, causing inefficient learning, while belief-based methods reduce dimensionality but suffer compounding errors with long delays. To address these limitations, we propose

1. AD-RL: introducing auxiliary tasks with shorter delays to bootstrap learning in large augmented state spaces, thereby reducing sample complexity without sacrificing convergence guarantees.
2. VDPO: formulating delayed RL as a variational inference problem to connect delay-free and delayed policies via imitation learning, enabling the use of advanced optimization techniques to improve sample efficiency.
3. DT-CORL: introducing a transformer-based belief model to infer latent states from delayed observations and jointly trains this belief with a constrained policy objective, ensuring that value estimation and belief representation remain aligned throughout learning.

Next Steps

As the next step, we plan to investigate the following topics:

- Delays in VLA infrastructure and real robotic applications;
- ...

Thank You!
Question?

Reference I

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

[2] A Rupam Mahmood et al. “Setting up a reinforcement learning task with a real-world robot”. In: **2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE. 2018, pp. 4635–4640.

[3] Jemin Hwangbo et al. “Control of a quadrotor with reinforcement learning”. In: **IEEE Robotics and Automation Letters** 2.4 (2017), pp. 2096–2103.

[4] Joel Hasbrouck and Gideon Saar. “Low-latency trading”. In: **Journal of Financial Markets** 16.4 (2013), pp. 646–679.

[5] Zhiguang Cao et al. “Using reinforcement learning to minimize the probability of delay occurrence in transportation”. In: **IEEE transactions on vehicular technology** 69.3 (2020), pp. 2424–2436.

Reference II

[6] Jiaming Tang et al. “Vlash: Real-time vlas via future-state-aware asynchronous inference”. In: **arXiv preprint arXiv:2512.01031** (2025).

[7] Mohammad Gheshlaghi Azar et al. “Speedy Q-learning”. In: **Advances in neural information processing systems**. 2011.

[8] Pierre Liotet. “Delays in Reinforcement Learning”. In: **arXiv preprint arXiv:2309.11096** (2023).

[9] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. “Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model”. In: **Machine learning** 91 (2013), pp. 325–349.

[10] Alekh Agarwal, Nan Jiang, and Sham M Kakade. “Reinforcement learning: Theory and algorithms”. In: (2019).

[11] Pierre Liotet et al. “Delayed reinforcement learning by imitation”. In: **International Conference on Machine Learning**. PMLR. 2022, pp. 13528–13556.

Reference III

[12] Jangwon Kim et al. “Belief Projection-Based Reinforcement Learning for Environments with Delayed Feedback”. In: **Thirty-seventh Conference on Neural Information Processing Systems**. 2023.

[13] Qingyuan Wu et al. “Boosting Reinforcement Learning with Strongly Delayed Feedback Through Auxiliary Short Delays”. In: **Forty-first International Conference on Machine Learning (ICML 2024)**. 2024.