Learning from Delay Distributions: A New Representation for Delay-Aware Reinforcement Learning

Zhuoru Yu*
Department of Computer Science and Engineering, Southeast University
Nanjing, China
zhuoruyu@seu.edu.cn

Chenchen Fu*
Department of Computer Science and Engineering, Southeast University
Nanjing, China
chenchen_fu@seu.edu.cn

Hengkai Zhong
Department of Computer Science and Engineering, Southeast University
Nanjing, China
hk_zhong@seu.edu.cn

Wanyuan Wang†
Department of Computer Science and Engineering, Southeast University
Nanjing, China
wywang@seu.edu.cn

Weiwei Wu
Department of Computer Science and Engineering, Southeast University
Nanjing, China
weiweiwu@seu.edu.cn

Chun Jason Xue
Department of Computer Science, Mohamed bin Zayed University of Artificial Intelligence
Abu Dhabi, United Arab Emirates
jason.xue@mbzuai.ac.ae

ABSTRACT

Delay remains a significant challenge for applying deep reinforcement learning (DRL) in real-world scenarios. Existing delay-aware DRL methods primarily rely on state augmentation to restore the Markov property in delayed environments, yet often assume the prior knowledge of the exact delay values and suffer from performance degradation in random delay environments. However, we observe that the random delays in real-world often follow specific statistical patterns. Based on this observation, we propose a novel method that leverages distributions to represent value functions, enabling a more accurate modeling of delay uncertainty beyond traditional expectation-based methods. Building upon delay distribution properties, we introduce a stochastic delay representation mechanism to reconstruct precise returns in delayed environments and prove its convergence to the optimal policy. Finally, we apply these techniques to design the delay-aware distributional actor-critic (D^2AC) DRL framework. Experimental results show that D^2AC significantly outperforms state-of-the-art delay-aware DRL methods across various random delay distributions in MuJoCo continuous control tasks. Open source code and appendix are available at: https://github.com/COOLAS-CS/D2AC.

KEYWORDS

Delay Aware, Deep Reinforcement Learning, Random Delays

ACM Reference Format:
Zhuoru Yu, Chenchen Fu, Hengkai Zhong, Wanyuan Wang†, Weiwei Wu, and Chun Jason Xue. 2026. Learning from Delay Distributions: A New Representation for Delay-Aware Reinforcement Learning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/XDQU6782

*Equal Contribution †Corresponding author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/XDQU6782

1 INTRODUCTION

Deep Reinforcement Learning (DRL) has advanced fields like gaming, robotics, and autonomous systems, achieving milestones such as surpassing human performance in Go. However, applying DRL to real-world scenarios faces significant challenges, with delay being a critical issue [10]. Conventional DRL assumes immediate interaction between the agent and environment, but real-world tasks often face delays due to physical or communication constraints. For example, teleoperation and self-driving cars are affected by network and sensing delays [27], hindering DRL efficiency in environments with unpredictable delays.

To handle with the above mentioned issues, a series of recent literature have proposed the delay-aware DRL methods. Most methods focus on *constant delay environments, assuming a known constant delay value* [5, 17, 20, 33, 34]. Even though some studies also address *random delay environments, they assumed that the methods have a complete knowledge of the exact random delay values* [4, 25, 32]. Obviously, such assumptions of the constant or predictable random delays are over-optimistic and unrealistic in real-world scenarios.

In most practical systems where the reinforcement learning tasks are applied to, it has been observed that real-world delays often follow specific statistical patterns, such as transmission and routing delays in communication networks [14, 19, 31, 36], or inference and computation delays in deep learning systems [1, 12]. Prior studies have shown that these delays are not entirely arbitrary, but rather tend to exhibit stationary statistical properties and probabilistic distributions, which can often be obtained through empirical profiling or statistical analysis [11, 22]. To better explore the delay features in the real-world scenarios, we conducted real measurements on the delays in some network control systems, the observation of which confirmed the presence of such distributional characteristics (Figure 1). Encouragingly, similar findings have been reported in delay-aware reinforcement learning studies. For example, [4] collected a dataset of communication delays between a decision-making computer and a flying robot over Wi-Fi, revealing that real-world Wi-Fi delays follow a distribution closely resembling a Gamma distribution, which was subsequently used to simulate

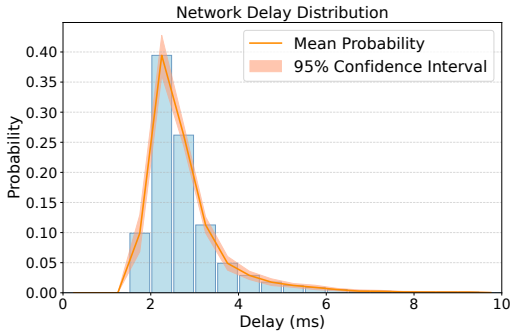
realistic delay conditions in their experiments. In addition, [25] investigated real-world delayed settings in the context of medical decision making, exhibiting statistical regularities in the delay patterns. Such an observation suggests that the distributional nature of delays can be exploited to design more effective delay-aware reinforcement learning algorithms.

Most recently, [30] proposed the State Augmentation-MLP, assuming knowledge of the maximum random delay, which has demonstrated outstanding performance compared to other SOTA methods in random delay environments. This method is based on state augmentation, a conventional method that, in random delay environments, concatenates the most recent delayed observation with a number of historical actions equal to the delayed timesteps, forming an augmented state that restores the Markov property, and is used in most existing delay-aware DRL methods [4, 5, 8]. However, the state space of such methods grows exponentially with increasing delay, significantly reducing learning efficiency and effectiveness, and showing limited adaptability in random delay environments.

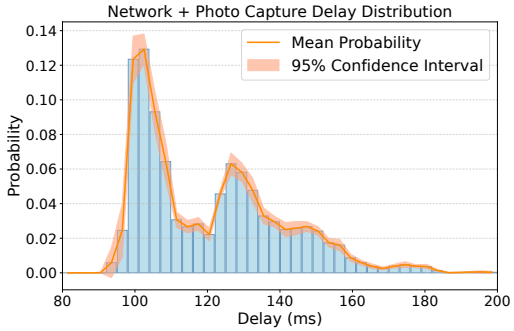
In this work, we propose a novel delay-aware distributional actor-critic method named D^{2}AC, which utilizes the delay distribution to address the challenges of deep reinforcement learning applied to real random delay environments, without assuming the knowledge of exact delay values as prior work do. Our method avoids modifying the state space and thus will not suffer from the exponential growth of the augmented state space as the delay increases, thereby improving learning efficiency. *Importantly, we recognize that random delays introduce probabilistic uncertainty in sequential feedback, where both observations and actions can influence a range of future timesteps, each with varying probabilities. The return thus must be computed to consider all affected timesteps, not just the next one. This delay distributional perspective is crucial for precise, robust learning in delay environments, enhancing adaptability while reducing computational overhead.* The main contributions include:

- We propose a new stochastic delay representation process for random delay environments, using delay distribution to account for the influence range of observations or actions. This accurately reconstructs the value representation of state-action pairs, enhancing performance in reinforcement learning with random delays.
- We represent the value function with distributions instead of expectation value, enabling more stable training in dynamic environments with random delays. The combination of stochastic delay representation and distributional value function enhances modeling in delayed environments and ensures convergence to the optimal policy.
- We empirically demonstrate that our proposed D^{2}AC achieves significant performance improvements over existing SOTA methods under random delay environments on the MuJoCo benchmark.

To the best of our knowledge, the proposed method is the first to explicitly leverage the stochastic nature of delays, avoiding unrealistic assumptions about the exact delay values to handle random delays. Additionally, it is the first to introduce the concept of distributional reinforcement learning to address reinforcement learning with random delays.



(a) Round-Trip Time (RTT) Delay Distribution. RTT of UDP packets is measured between the UAV and the base station.



(b) RTT+Photo Capture Delay. The total time, including the RTT of UDP packets between the UAV and the base station as well as the time taken for the UAV to capture an RGB image, is measured.

Figure 1: Statistical results of delay data under different settings in the remote-controlled UAV scenario, where UAV and base station are connected by Wifi. We conducted 10 measurements under different settings, collecting 1,000 delay data points in each measurement. The mean values across the 10 measurements (orange curve) and the 95% confidence intervals (shaded area) were calculated and analyzed.

2 RELATED WORK

Delay-aware Reinforcement Learning. Delays in the environment render the standard Markov decision process (MDP) inapplicable, prompting the development of variants such as delay-aware and constant delayed MDPs. To reformulate these as standard MDPs, a widely adopted state augmentation method concatenates the last observed state with intervening actions since that observation [5, 8]. In this line of research, the Delay-Correcting Actor-Critic (DCAC) algorithm [4] is capable of handling both random observation and action delays, which aligns closely with the problem we aim to address. However, this method suffers from extremely low computational efficiency caused by the recursive nature of the partial resampling operator. Furthermore, the augmented state space grows exponentially as the number of delayed timesteps increases.

To address this issue, several state prediction-based methods [15, 21, 29, 35] have been proposed to predict the undelayed state or belief representation by utilizing historical state and action information, by leveraging world models, recurrent neural networks

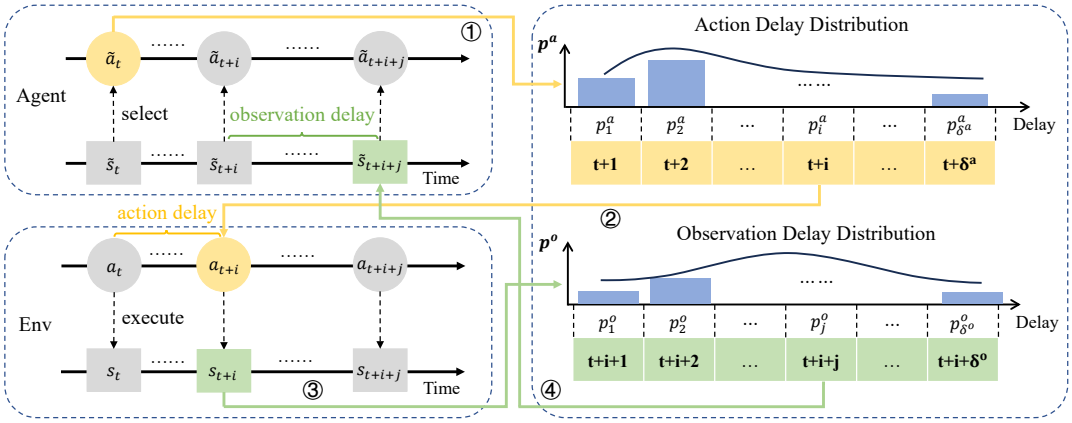


Figure 2: Illustration of the shifting process of actions and observations in random delay environments. ①the agent selects an action $\tilde{a}_t$ based on the observation $\tilde{s}_t$ and sends it to the environment; ②due to the action delay, the environment receives $\tilde{a}_t$ after i timesteps, denoted as a_{t+i}, i.e., $\tilde{a}_t$ and a_{t+i} are the same action; ③the environment executes a_{t+i} and returns the observation s_{t+i} to the agent; ④due to the observation delay, the agent receives s_{t+i} after j timesteps, denoted as $\tilde{s}_{t+i+j}$, i.e., s_{t+i} and $\tilde{s}_{t+i+j}$ are the same observation.

(RNNs), etc. However, in complex random delay environments, the accuracy and generality of these prediction methods significantly limit their broader application. Recently, DFBT [32] introduced a belief estimation framework that directly forecasts current states from observations without recursively estimating intermediate latent states. This design greatly mitigates the compounding errors of traditional recursive prediction approaches. However, DFBT assumes access to a delay-free offline dataset to pretrain the belief model before deployment in delayed online environments, an assumption that is highly unrealistic when delays are intrinsic to the real world applications. Additionally, [17] proposed a novel belief projection method that tackles the state-space explosion problem by projecting the augmented state space into a smaller one. However, this method is only applicable to constant delay environments, which do not reflect real-world practical features.

Alternatively, several auxiliary-policy-based methods have been proposed to mitigate limitations of state augmentation and prediction based methods. [33] framed the problem as variational inference and used behavior cloning to approximate undelayed policies, while [34] leveraged auxiliary tasks with short delays to improve policy learning for long-delay tasks. Similarly, [20] utilized imitation learning to teach delayed policies based on undelayed demonstrations. However, they rely on the unrealistic assumption that effective policies can be obtained in undelayed or short-delay environments, which is impractical due to the inherent and patterned nature of delays in real-world applications. Moreover, their focus on constant delays further limits practical applicability.

Distributional Reinforcement Learning. Conventional reinforcement learning optimizes the expected return, but randomness between the agent and the environment causes returns to follow a distribution under a policy π. [3] introduced the distributional DQN (C51), representing returns as discrete distributions, establishing the foundation for distributional reinforcement learning. Subsequently, several methods have refined distribution modeling, providing robust theoretical and practical advances [6, 7, 26, 37]. The distributional perspective has also been extended to actor-critic frameworks, such as Gaussian Mixture Actor-Critic [24], which models returns with Gaussian mixtures, and algorithms like D4PG [2] and DSAC [9], improving value estimation in complex scenarios. Building on these advancements, to the best of our knowledge, this work is the first to leverage distributional reinforcement learning to model uncertainty in random delay environments and address challenges in accurate return estimation under delays.

3 DELAY-AWARE DISTRIBUTIONAL VALUE FUNCTION AND STOCHASTIC DELAY REPRESENTATION

In this section, we first present the distributional value function that enables more stable learning in delayed environments. Building on this, we propose a novel stochastic delay representation mechanism, which accurately models state-action returns across all timesteps potentially affected by random delays. Finally, we further investigate theoretical guarantees of stochastic delay representation in settings involving both observation and action delays.

3.1 Distributional Value Function

Conventional reinforcement learning is generally modeled as a Markov decision process represented by a 5-tuple (S, A, R, P, γ), where S is the state space, A is the action space, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition probability, and $\gamma \in (0, 1)$ is the discount factor. The Bellman equation is utilized to describe the value function as shown below:

$$Q(s, a) = r(s, a) + \gamma \cdot Q(s', a'), \tag{1}$$

where $s' \sim P(\cdot|s, a)$ and $a' \sim \pi(\cdot|s')$.

In contrast to the common approach to RL which models the expectations of the return, the distributional value function effectively models the distributional characteristics of returns [3]. The

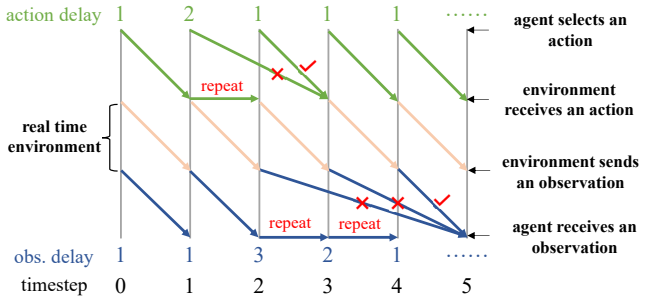


Figure 3: Interaction process with random delays. Black numbers and gray vertical lines denote timesteps. Green numbers/lines represent the magnitudes and effects of action delays. Pink lines indicate when the environment applies actions and transitions. Blue numbers/lines represent the magnitudes and effects of observation delays.

distributional perspective retains the expressive power of value distributions, improving learning stability and supporting specialized tasks. In environments with random delays, it effectively captures uncertainty, enhancing both robustness and representational flexibility in value estimation. Moreover, by substantially reducing estimation errors, it strengthens the effectiveness of the following stochastic delay representation mechanism.

Being aware of the representation capability of value distributions, this paper is the first to introduce the distributional value function in delayed environments. Formally, in DRL, let $Q(s,a)$ denote the expectation of the state-action value, the distributional value function can be expressed as $Z(s,a)$ whose expectation is the value $Q(s,a)$, i.e. $Q(s,a) = \mathbb{E}[Z(s,a)]$. This distributional value function can be also described by a recursive Bellman equation, but in a distributional manner:

$$Z(s,a) = r(s,a) + \gamma \cdot Z(s',a'), \tag{2}$$

where $s' \sim P(\cdot|s,a)$ and $a' \sim \pi(\cdot|s')$.

In this work, we adopt the Gaussian distribution, known for its high expressiveness, to approximate the return [23, 28]. Thus, the neural network designed for distribution estimation, outputs both the mean and standard deviation.

3.2 Delay-Aware Stochastic Delay Representation

Before formally introducing our proposed method, we first analyze the effect of delay distributions on the interaction dynamics between the agent and the environment. In random delay environments, the transitions recorded by the agent do not directly correspond to the real-time interactions and feedback from the environment. To clearly represent, in random delay environments, we use $\tilde{s}_t$, $\tilde{a}_t$, and $\tilde{r}_t$ to denote the observation, action, and reward at the agent's side at timestep t, distinguishing them from the oracle state s_t, action a_t, and reward r_t involved in the delay-free environment.

As an example, we consider the environment with action delay only, where the delayed observation $\tilde{s}_t$ is identical to the oracle state s_t. As shown in Figure 2, due to action delays, the current state-action pair $(\tilde{s}_t, \tilde{a}_t)$ may not immediately determine the next state $\tilde{s}_{t+1}$. Instead, the agent receives feedback for $(\tilde{s}_t, \tilde{a}_t)$ after a delay of i timesteps, where i is drawn from an action delay distribution p^a. Consequently, the reward $\tilde{r}_t$ observed at timestep t does not correspond to the state-action pair $(\tilde{s}_t, \tilde{a}_t)$; the true reward for this pair is delayed and becomes available at timestep $t+i$, denoted as $\tilde{r}_{t+i}$. Therefore, to accurately compute the return for $(\tilde{s}_t, \tilde{a}_t)$, it must be evaluated starting from timestep $t+i$. Similarly, in environments with observation delay only, when the environment provides observations and rewards, the agent may receive this feedback at different future timesteps, with varying probabilities for each timestep.

It is worth noting that we assume the environment's reward is bound to the observation, meaning that at the same timestep, the reward and observation share the same delay. If the agent receives multiple observations and rewards at a given timestep due to random delays, it selects the most recent ones; if none are received, it reuses those from the previous timestep. Similarly, for the environment side, the same rule applies, where it selects the most recent action if multiple are received; otherwise, it reuses the previous action, as illustrated in Figure 3.

Based on the analysis, we propose the stochastic delay representation method to address the DRL with random delays. We present insights into the characteristics of random delays, as outlined below. By moving beyond the conventional local horizon of policy updates, which rely on sampled single-step transitions, we observe that random delays introduce probabilistic uncertainty into the timing of feedback. Considering this probabilistic nature, it becomes clear that both observations and actions can influence a spectrum of future timesteps, each with varying probabilities, as shown in Figure 2. Therefore, the accurate computation of returns must account for all potential timesteps affected by these delayed actions and observations, rather than being limited to the next timestep, to recover accurate value functions in the presence of random delays.

As shown in Figure 2, from the perspective of delay distribution, an action can influence multiple future timesteps with certain probabilities. Therefore, the estimation of value function $Z(\tilde{s}_t, \tilde{a}_t)$ must account for the probabilistic effects of this delay distribution. Specifically, $Z(\tilde{s}_t, \tilde{a}_t)$ should reflect the returns between timesteps $t+1$ and $t+\delta_a$, where we assume that, in real world applications, when the probability of a delay falls below a threshold, its impact becomes negligible. We define the effective delay range as the set of delays in the distribution that exceed this threshold, with δ_a representing the maximum value of the action delay distribution. Based on this insight, we can derive the following stochastic delay representation for the environment with random action delays:

$$Z(\tilde{s}_t, \tilde{a}_t) = \sum_{i=1}^{\delta_a} p_i^a \cdot [\tilde{r}_{t+i}(\tilde{s}_{t+i}) + \gamma \cdot Z(\tilde{s}_{t+i+1}, \tilde{a}_{t+i+1})], \tag{3}$$

where p_i^a denotes the probability that the action delay is i timesteps, and we define $\tilde{r}(\tilde{s})$ as the reward associated with state $\tilde{s}$ in this paper.

Similarly, in environments with only observation delays, when the environment provides observations and rewards, the agent may receive this feedback at different future timesteps, with varying probabilities for each timestep. Based on the observation delay probability distribution, a similar form to Eq. (3) can be used to

reconstruct the value function by replacing the action delay distribution with the observation delay distribution. In the following, we provide the detailed theoretical analysis of the properties of our proposed D^{2}AC method.

3.3 Unifying Observation and Action Delays

Real-world scenarios often involve various types of delays, which can significantly increase the problem complexity. When both observation delay and action delay are present, we decompose the process according to Eq. (3). As illustrated in Figure 2, due to the action delay, an action taken at timestep t will be executed at timestep $t+i$ with a probability of p_i^a. At timestep $t+i$, this action interacts with the environment, and the agent will receive environmental feedback (e.g., observation and reward) at timestep $t+i+j$ with a probability of p_j^o. Based on the influence of such delay distributions, we define stochastic delay representation with distributional returns in Eq. (4) to represent the value function, accounting for both observation and action delays.

$$\begin{aligned} Z(\tilde{s}_t, \tilde{a}_t) &= \sum_{i=1}^{\delta_a} p_i^a \cdot \sum_{j=1}^{\delta_o} p_j^o \cdot Z(\tilde{s}_{t+i+j}, \tilde{a}_{t+i+j}) \\ &= \sum_{i=1}^{\delta_a} p_i^a \cdot \sum_{j=1}^{\delta_o} p_j^o \cdot [\tilde{r}_{t+i+j}(\tilde{s}_{t+i+j}) \\ &\quad + \gamma \cdot Z(\tilde{s}_{t+i+j+1}, \tilde{a}_{t+i+j+1})], \end{aligned} \tag{4}$$

where δ_a and δ_o denote the maximum values of effective action delay range and effective observation delay range, respectively, while p_i^o and p_j^a represent the probabilities that the observation delay and action delay are equal to i and j, respectively. By applying stochastic delay representation with distributional returns in Eq. (4), we can efficiently and directly recover the return of state-action pairs $(\tilde{s}, \tilde{a})$ in environments with random delays, thereby enabling effective and stable policy improvement.

Proposition 1 [Equivalence of Observation and Action Delays]. *Assume the action delay $i \in [1, \delta_a]$ and the observation delay $j \in [1, \delta_o]$ are discrete variables and statistically independent of each other. The stochastic delay representation $Z(\tilde{s}_t, \tilde{a}_t)$ depends only on the total delay $k = i + j$ through the joint distribution $p_k^{joint} = \sum_{i+j=k} p_i^a \cdot p_j^o$. Thus action and observation delays contribute identically to the construction of $Z(\tilde{s}_t, \tilde{a}_t)$ and are equivalent in effect within this representation.*

Building on prior studies on the equivalence of various delay types [16, 25, 30], we provide a formal proposition that our proposed stochastic delay representation can equivalently handle these delays, as it addresses random delays through a novel value function update mechanism, which differs from the perspectives taken in prior studies. The formal proof is also provided (see Appendix B.1). Therefore, Eq. (4) can be simplified into a unified form:

$$\begin{aligned} Z(\tilde{s}_t, \tilde{a}_t) = \sum_{k=1}^{\delta_{joint}} p_k^{joint} \cdot [\tilde{r}_{t+k}(\tilde{s}_{t+k}) \\ +\gamma \cdot Z(\tilde{s}_{t+k+1}, \tilde{a}_{t+k+1})], \end{aligned} \tag{5}$$

where δ_{joint} denotes the maximum value of this joint delay. For simplicity, we let k start from 1, with $p_k^{joint} = 0$ if no valid (i, j) satisfies $i + j = k$; this does not affect the result.

Theorem 1 [Convergence of Stochastic Delay Representation with Distributional Returns]. *The stochastic delay representation with distributional returns in Eq.(4), which map the state-action pair $(\tilde{s}, \tilde{a})$ to a distributional return in delayed environments, can converge to a policy π^* such that $Q^{\pi^*}(\tilde{s}, \tilde{a}) \geq Q^{\pi}(\tilde{s}, \tilde{a})$ for $\forall \pi$ and $\forall (\tilde{s}, \tilde{a}) \in \mathcal{S} \times \mathcal{A}$, assuming that $|\mathcal{A}| < \infty$ and reward is bounded.*

The formal proof is provided in Appendix B.2. Theorem 1 relies on the fact that the distributional soft Bellman operator is a contraction mapping [9]. Under this property, we formally establish that the stochastic delay representation with distributional returns converges to an optimal policy that maximizes the expected value function (Q-value) rather than the distributional value function (Z-value). This result arises from the fact that the objective function in the policy improvement process remains grounded in the traditional expected value function, as detailed in Section 4.

Although the proposed method is primarily designed for random delay environments, it is also applicable to constant delay environments commonly addressed in the related works. This is because constant delay environments can be considered as a special case of random delay environments, where the probability of a specific delay is equal to one. This effectiveness and adaptability are also validated in our experiments.

4 DELAY-AWARE DISTRIBUTIONAL ACTOR-CRITIC (D²AC)

In this section, we propose the Delay-Aware Distributional Actor-Critic (D^{2}AC) and demonstrate that the stochastic delay representation with distributional returns is broadly applicable to any Actor-Critic algorithm. Although we use Soft Actor-Critic (SAC) as an example, our method is not tied to any specific properties of SAC. The framework of D^{2}AC is shown in Figure 4.

The core insight of D^{2}AC lies in fundamentally redefining the critic's role within the actor-critic paradigm. Instead of estimating a single expected return, the critic is designed to construct a target value distribution that encapsulates the uncertainty inherent in random delay environments. To this end, the critic reconstructs a more accurate return distribution by sampling the future trajectory from the replay buffer guided by the delay distribution and probabilistically weighting the returns within it. This information-rich distribution then provides a substantially more stable and expressive learning signal for the actor, which in turn learns to select actions that favorably shape future returns. This synergistic interplay is key to D^{2}AC's ability to learn robustly and effectively in such challenging environments.

The stochastic delay representation with distributional returns variant of the soft Bellman operator $\mathcal{T}_{\mathcal{D}}^{\pi} : \mathcal{Z} \to \mathcal{Z}$, where $\mathcal{Z}$ is defined as the distributional value function space, with the maximum entropy can be defined as

$$\begin{aligned} \mathcal{T}_{\mathcal{D}}^{\pi} Z(\tilde{s}, \tilde{a}) \overset{D}{=} \sum_{i=1}^{\delta_a} p_i^a \cdot \sum_{j=1}^{\delta_o} p_j^o \cdot [\tilde{r}_{i+j}(\tilde{s}_{i+j}) \\ + \gamma \left(Z\left(\tilde{s}_{i+j+1}, \tilde{a}_{i+j+1}\right) - \alpha \log \pi \left(\tilde{a}_{i+j+1} | \tilde{s}_{i+j+1}\right)\right)], \end{aligned} \tag{6}$$

where $\tilde{r}_{i+j}$ is defined as the reward at $i+j$ timesteps after the state-action pair $(\tilde{s}, \tilde{a})$, $(\tilde{s}_{i+j+1}, \tilde{a}_{i+j+1})$ as the state-action pair at $i+j+1$ timesteps after $(\tilde{s}, \tilde{a})$, and α is the temperature parameter.

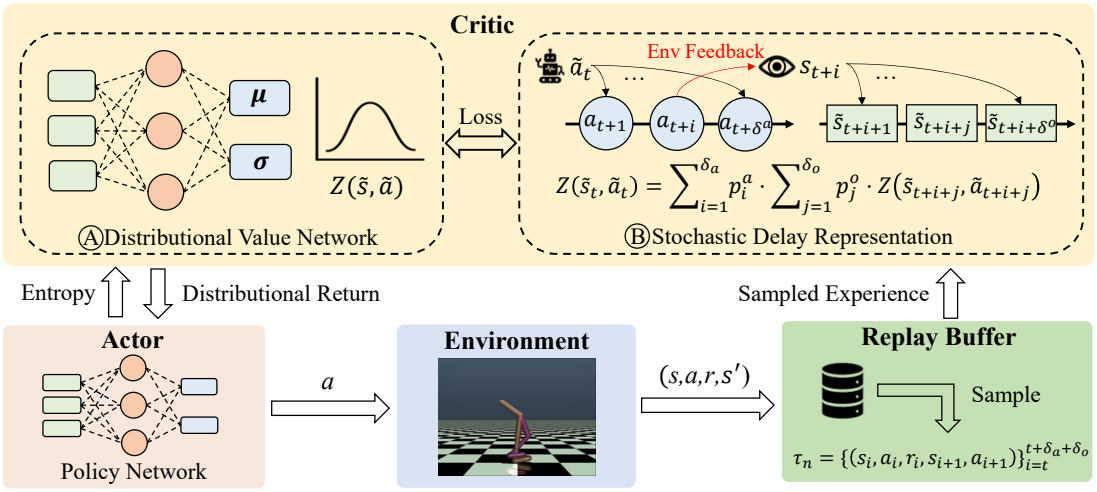


Figure 4: Overview of D^{2}AC: comprising two core components—Ⓐ the distributional value function, where the critic is trained to represent the value of state-action pairs as a distribution, and Ⓑ the stochastic delay representation, which reconstructs the return of state-action pairs in random delay environments by utilizing sampled sequential transition data.

D^{2}AC samples multiple sequential trajectory information from the replay buffer, represented as $\tau_n = \{(s_i, a_i, r_i, s_{i+1}, a_{i+1})\}_{i=t}^{t+\delta_a+\delta_o}$, where the length of the sampled trajectory is the maximum of total delays. Furthermore, for computational convenience, we use the KL divergence as the regulation metric during training. Although Theorem 1 employs the maximal form of the Wasserstein metric, KL divergence is a common practice in the field of distributional reinforcement learning [3, 9].

In the policy improvement step, the policy is learned by maximizing the following objective:

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\substack{\tilde{s}\sim\mathcal{B}\\ \tilde{a}\sim\pi_\phi}}\Big[\mathop{\mathbb{E}}_{Z\sim\mathcal{Z}_\theta}\big[Z(\tilde{s},\tilde{a})\big] - \alpha\log\big(\pi_\phi(\tilde{a}|\tilde{s})\big)\Big], \tag{7}$$

where $\mathcal{B}$ denotes the replay buffer containing collected experience in delayed environments, and ϕ denotes the parameters of policy π, while θ represents the parameters of $\mathcal{Z}$.

Specifically, the expectation of the stochastic delay representation with distributional returns is involved in the policy improvement, as shown in Eq. (8). This equation is equivalent to the update rule presented in Eq. (7).

$$\pi_{new} = \arg\min_{\pi_{old}\in\Pi} D_{KL} \left(\pi_{old}(\cdot|\tilde{s}) \,\middle\|\, \frac{\mathbb{E}_{Z\sim\mathcal{Z}_\theta,\,\tilde{a}\sim\pi_{old}}\big[Z^{\pi_{old}}_{value}(\tilde{s},\tilde{a})\big]}{Z^{\pi_{old}}_{action}(\tilde{s})}\right) \tag{8}$$

where Z_{value} denotes the stochastic delay representation with distributional returns, θ represents the parameters of Z_{value}, Z_{action} denotes the action distribution, and π_{old} is the previous policy.

5 EXPERIMENTAL RESULTS

Environments. We conducted experimental evaluations in the MuJoCo environments within Gymnasium and implemented the random delay settings using Wrappers based on [4]. In the setting of random delay environments, the analysis of Wi-Fi network delays in [4] shows that the statistical results closely resemble a gamma distribution, and in [18], the authors used Gaussian-related distributions to simulate delays. *These findings are consistent with the statistical results we presented in Figure 1, which inspired us to design gamma and double Gaussian distributions for delay environments.* While uniform distribution is a common choice in the experiments of related works [4, 25], it is also included. The details of delay distribution are provided in Appendix C.1.

Implementation. The implementation of the distributional value function in our method draws on the Distributional Soft Actor-Critic, which assumes that the random returns $Z(s,a)$ follow a Gaussian distribution [9]. *Given that most related work primarily considers observation delays, the following experimental results are presented in the context of observation delays for comparison.* Each task was executed with 8 runs, and the hyperparameter settings of D^{2}AC are presented in Appendix C.2.

5.1 Comparative Evaluation

Baselines. We compared the performance of our proposed D^{2}AC with six state-of-the-art methods, including DFBT [32], VDPO [33], AD-SAC [34], State Augmentation-MLP(abbreviated as State Aug-MLP) [30], BPQL [17] and DCAC [4]. In the experiments involving random delays, to ensure a fair comparison, we provided the mean value of the random delay distribution as prior knowledge for methods primarily designed to handle constant delays, such as VDPO, AD-SAC, and BPQL. For methods designed to handle random delays, we follow their original design schemes. Specifically, for State Augmentation-MLP, the maximum value of the random delay distribution is provided as prior knowledge. For DFBT and DCAC, the exact delay value at each timestep is provided accordingly.

It is important to note that DFBT requires the D4RL offline dataset to construct its training data. Since the MuJoCo tasks in D4RL are based on older environments (typically version v2), while newer online benchmarks have migrated to MuJoCo v4, a direct

Table 1: Results of comparative experiments within 1M global steps. Data are presented as mean ± standard error of the mean (S.E.M). The best performing methods, including those within the range of S.E.M of the best, are highlighted in bold. Each method was run 8 random seeds.

Delay	Method	Walker2d-v4	Hopper-v4	Ant-v4	Humanoid-v4	Reacher-v4
Gamma	VDPO[33]	3596.3$_{\pm 1342.8}$	1279.4$_{\pm 280.4}$	2541.5$_{\pm 833.1}$	4539.9$_{\pm 1749.2}$	-5.5$_{\pm 1.8}$
	AD-SAC[34]	3710.0$_{\pm 1087.5}$	1193.3$_{\pm 293.0}$	1187.1$_{\pm 657.1}$	2405.4$_{\pm 1218.3}$	-5.3$_{\pm 0.6}$
	State Aug-MLP[30]	3703.3$_{\pm 503.4}$	2753.1$_{\pm 571.7}$	2459.0$_{\pm 396.1}$	1030.4$_{\pm 181.5}$	**-3.9$_{\pm 0.5}$**
	BPQL[17]	2365.1$_{\pm 880.1}$	2414.9$_{\pm 447.3}$	1070.5$_{\pm 220.6}$	77.2$_{\pm 1.5}$	-7.2$_{\pm 1.6}$
	DCAC[4]	3229.4$_{\pm 923.2}$	3159.7$_{\pm 334.5}$	1057.9$_{\pm 617.7}$	4836.3$_{\pm 74.6}$	-9.9$_{\pm 2.9}$
	D^{2}AC(ours)	**4584.5$_{\pm 595.6}$**	**3331.0$_{\pm 323.4}$**	**2856.8$_{\pm 421.1}$**	**4847.2$_{\pm 675.8}$**	-4.3$_{\pm 0.5}$
Double Gaussian	VDPO[33]	2181.7$_{\pm 1324.5}$	611.6$_{\pm 217.4}$	1206.8$_{\pm 499.6}$	2581.7$_{\pm 1239.4}$	-6.3$_{\pm 2.0}$
	AD-SAC[34]	1613.7$_{\pm 880.3}$	528.8$_{\pm 290.9}$	244.6$_{\pm 153.5}$	1425.8$_{\pm 1084.0}$	-7.0$_{\pm 0.9}$
	State Aug-MLP[30]	2624.9$_{\pm 911.5}$	2731.1$_{\pm 446.0}$	**1788.8$_{\pm 348.0}$**	714.2$_{\pm 123.2}$	-4.8$_{\pm 1.1}$
	BPQL[17]	517.3$_{\pm 88.3}$	752.5$_{\pm 465.9}$	791.2$_{\pm 325.5}$	272.2$_{\pm 162.6}$	-11.7$_{\pm 1.4}$
	DCAC[4]	3608.6$_{\pm 212.9}$	2915.4$_{\pm 399.0}$	330.8$_{\pm 127.8}$	2950.5$_{\pm 1381.8}$	-8.6$_{\pm 1.9}$
	D^{2}AC(ours)	**4680.8$_{\pm 741.7}$**	**3091.1$_{\pm 284.0}$**	1778.8$_{\pm 105.3}$	**4023.9$_{\pm 722.6}$**	**-4.8$_{\pm 0.3}$**
Uniform	VDPO[33]	1039.4$_{\pm 524.8}$	326.1$_{\pm 138.7}$	1635.0$_{\pm 418.9}$	2143.9$_{\pm 1011.1}$	-8.5$_{\pm 2.7}$
	AD-SAC[34]	1030.2$_{\pm 359.2}$	327.7$_{\pm 89.7}$	455.8$_{\pm 318.1}$	2302.4$_{\pm 1039.4}$	-8.8$_{\pm 1.2}$
	State Aug-MLP[30]	2568.2$_{\pm 1154.8}$	1904.7$_{\pm 506.6}$	1621.3$_{\pm 253.8}$	638.5$_{\pm 54.0}$	-4.9$_{\pm 0.8}$
	BPQL[17]	360.6$_{\pm 70.6}$	546.9$_{\pm 253.4}$	465.2$_{\pm 359.2}$	116.9$_{\pm 51.1}$	-12.4$_{\pm 2.1}$
	DCAC[4]	736.2$_{\pm 474.9}$	856.8$_{\pm 633.3}$	449.8$_{\pm 185.9}$	888.2$_{\pm 233.4}$	-9.9$_{\pm 1.7}$
	D^{2}AC(ours)	**4266.8$_{\pm 523.6}$**	**2827.3$_{\pm 235.9}$**	**1700.4$_{\pm 188.8}$**	**3862.1$_{\pm 613.5}$**	**-4.8$_{\pm 0.4}$**
Constant 5	VDPO[33]	3589.2$_{\pm 1458.7}$	1912.5$_{\pm 439.5}$	4057.2$_{\pm 2041.7}$	4950.0$_{\pm 1201.4}$	-5.1$_{\pm 1.5}$
	AD-SAC[34]	3713.2$_{\pm 943.1}$	2129.7$_{\pm 635.5}$	3391.2$_{\pm 1014.4}$	3650.0$_{\pm 1546.9}$	**-5.1$_{\pm 0.5}$**
	State Aug-MLP[30]	3355.7$_{\pm 793.7}$	2828.4$_{\pm 802.7}$	2171.1$_{\pm 185.4}$	708.1$_{\pm 60.6}$	**-5.1$_{\pm 0.5}$**
	BPQL[17]	4250.3$_{\pm 409.9}$	3224.4$_{\pm 607.9}$	**5458.4$_{\pm 167.5}$**	5181.4$_{\pm 519.0}$	-5.2$_{\pm 2.1}$
	DCAC[4]	4320.3$_{\pm 432.9}$	**3364.6$_{\pm 163.8}$**	2629.8$_{\pm 417.9}$	5213.0$_{\pm 199.8}$	-7.7$_{\pm 2.0}$
	D^{2}AC(ours)	**4564.2$_{\pm 377.4}$**	2947.8$_{\pm 235.8}$	3002.2$_{\pm 245.6}$	**5280.1$_{\pm 517.2}$**	-5.2$_{\pm 0.6}$

Table 2: Results of comparative experiments with DFBT within 1M global steps. Data are presented as mean ± standard error of the mean (S.E.M). The best performing methods, including those within the range of S.E.M of the best, are highlighted in bold. Each method was run 8 random seeds.

Delay	Method	Walker2d-v2	Hopper-v2	Ant-v2
Gamma	DFBT[32]	3350.9$_{\pm 469.2}$	2509.5$_{\pm 259.3}$	2429.3$_{\pm 308.9}$
	D^{2}AC(ours)	**4819.8$_{\pm 288.4}$**	**3398.7$_{\pm 368.6}$**	**2791.2$_{\pm 254.4}$**
Double Gaussian	DFBT[32]	3584.3$_{\pm 602.3}$	2036.9$_{\pm 742.5}$	**2622.6$_{\pm 499.6}$**
	D^{2}AC(ours)	**4522.8$_{\pm 187.7}$**	**3338.1$_{\pm 110.3}$**	1954.5$_{\pm 138.3}$
Uniform	DFBT[32]	2233.1$_{\pm 329.1}$	2653.1$_{\pm 697.8}$	**2006.0$_{\pm 495.0}$**
	D^{2}AC(ours)	**4443.3$_{\pm 318.3}$**	**2932.3$_{\pm 134.1}$**	1677.1$_{\pm 46.5}$
Constant 5	DFBT[32]	3388.4$_{\pm 635.9}$	2796.1$_{\pm 237.8}$	2231.6$_{\pm 316.2}$
	D^{2}AC(ours)	**5025.4$_{\pm 106.6}$**	**3088.2$_{\pm 107.0}$**	**2934.1$_{\pm 109.4}$**

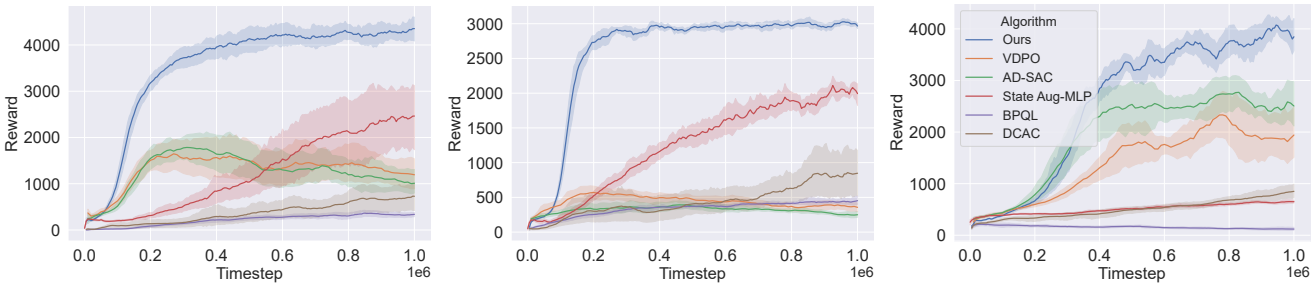
comparison across versions would introduce discrepancies in task dynamics and reward scaling. Therefore, for a fair and consistent evaluation, we compare our method with DFBT exclusively on MuJoCo v2 tasks. For other methods that do not rely on offline datasets, we perform evaluations on the MuJoCo v4 tasks.

The comparative evaluation results in random delay environments are shown in Table 1 and Table 2, and the smoothed learning curves are provided in Figure 5 and Appendix D.1. *In random delay environments, D^{2}AC demonstrates significant performance advantages across most tasks. By integrating the distributional value function with the stochastic delay representation mechanism, D^{2}AC achieves not only faster convergence but also lower performance variance across multiple runs under the same conditions, showcasing its exceptional adaptability and stability in random delay environments.*

5.2 D^{2}AC in Handling Different Types of Delays

Applicable to environments with constant delays: We conducted experiments with a constant delay of 5 to evaluate the generalization of D^{2}AC (results given in Table 1 and Table 2 and learning curves given in Appendix D.2). While a constant delay environment can be viewed as a degenerate case of a delay distribution, it lacks the inherent statistical properties that our method is designed to leverage. This leads to a degeneracy in our core mechanism, which prevents D^{2}AC from consistently outperforming all state-of-the-art methods designed for constant delays across all tasks in this specific delay setting. Nevertheless, D^{2}AC holds a significant advantage over methods specifically designed for random delays, showcasing its superior adaptability and effectiveness under different delay conditions.

Applicable to environments with action delays: Due to the theoretical equivalence of different delay types, most related works primarily focus on methods designed for observation delays. We experimentally demonstrate that the proposed D^{2}AC can effectively handle scenarios with different types of delays. In the experiments, both observation delays and action delays follow a gamma distribution. The experimental results, presented in Appendix D.3, show that, despite some randomness leading to slight variations in the



(a) Walker2d-v4 (b) Hopper-v4 (c) Humanoid-v4

Figure 5: Comparison results of the smoothed learning curves for partial MuJoCo tasks in the uniform delay environment.

Table 3: Results of ablation experiments within 1M global steps. Data are presented as mean ± standard error of the mean (S.E.M). The best performing methods, including those within the range of S.E.M of the best, are highlighted in bold. Each method was run 8 random seeds. SDR denotes the proposed stochastic delay representation method.

Delay	Method	Walker2d-v4	Hopper-v4	Ant-v4	Humanoid-v4	Reacher-v4
Gamma	SAC	$2020.7_{\pm 1208.8}$	$1574.4_{\pm 767.5}$	$60.9_{\pm 148.4}$	$109.9_{\pm 34.6}$	$-9.0_{\pm 3.2}$
	SAC+SDR	$3376.9_{\pm 576.7}$	$3184.5_{\pm 140.6}$	$771.6_{\pm 246.2}$	$538.6_{\pm 98.1}$	$-5.7_{\pm 0.7}$
	D^{2}AC (ours)	$\mathbf{4584.5_{\pm 595.6}}$	$\mathbf{3331.0_{\pm 323.4}}$	$\mathbf{2856.8_{\pm 421.1}}$	$\mathbf{4847.2_{\pm 675.8}}$	$\mathbf{-4.3_{\pm 0.5}}$
Double Gaussian	SAC	$190.8_{\pm 262.6}$	$277.7_{\pm 149.0}$	$-173.6_{\pm 278.6}$	$88.7_{\pm 11.3}$	$-10.9_{\pm 1.2}$
	SAC+SDR	$3052.1_{\pm 733.9}$	$1628.2_{\pm 1094.5}$	$-18.9_{\pm 4.1}$	$673.4_{\pm 156.2}$	$-8.3_{\pm 3.8}$
	D^{2}AC (ours)	$\mathbf{4680.8_{\pm 741.7}}$	$\mathbf{3091.1_{\pm 284.0}}$	$\mathbf{1778.8_{\pm 105.3}}$	$\mathbf{4023.9_{\pm 722.6}}$	$\mathbf{-4.8_{\pm 0.3}}$

outcomes for the two types of delays, D^{2}AC achieves comparable performance in environments with observation and action delays, highlighting its effectiveness in handling various types of random delay environments. This result is consistent with Proposition 1.

5.3 Ablation Study

To better evaluate the contribution of each component in our proposed D^{2}AC, we conducted ablation experiments comparing it with normal SAC and SAC+Stochastic Delay Representation (SAC+SDR). Both normal SAC and SAC+Stochastic Delay Representation are based on CleanRL [13], where SAC+Stochastic Delay Representation refers to applying the stochastic delay representation to normal SAC while maintaining the return as an expectation rather than as a distribution. We conducted ablation experiments in both gamma and double Gaussian delay distributions, with the results shown in Table 3 and the learning curves provided in Appendix D.4.

The ablation experiment results demonstrate that distributional returns and the stochastic delay representation mechanism are complementary and interdependent. Random delays may introduce missing, duplicate, or out-of-order observations, actions, and rewards, leading to inconsistent sequential feedback between the agent-environment interaction and the transition stored in the replay buffer. This inconsistency causes SAC+Stochastic Delay Representation to suffer from inaccurate Q-value estimation based on data sampled from the replay buffer, thus reducing the effectiveness of the stochastic delay representation mechanism. In contrast, distributional returns effectively model the uncertainty in delayed environments, enhancing the robustness and representational flexibility of the value function estimation. Moreover, by leveraging the advantage of distributional returns in significantly reducing value function estimation errors, the stochastic delay representation mechanism becomes more effective. This synergy enables our proposed D^{2}AC method to demonstrate superior performance in random delay environments.

6 CONCLUSIONS

In this paper, we tackle deep reinforcement learning with random delays. To model the delay uncertainty, we represent the Q-value as a distribution and develop a stochastic delay representation method to recover true returns from delay distributions. Integrating these components into the actor-critic framework yields the delay-aware distributional actor-critic (D^{2}AC), offering a new perspective on delay-aware DRL. Experimental results show that D^{2}AC robustly and consistently outperforms state-of-the-art delay-aware DRL methods in random delay environments.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 62232004, 62272099, 61972086 and 62476121, the National Key Research and Development Program of China under Grant No. 2024YFB4303805, the Natural Science Foundation of Jiangsu Province under Grant Nos. BK20231543 and BK20230024, the Key Research and Development Projects in Jiangsu Province under Grant No. BE2021001-2, and the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education.

REFERENCES

[1] Haibo Bao and Jinde Cao. 2011. Delay-distribution-dependent state estimation for discrete-time stochastic neural networks with random delay. *Neural Networks* 24, 1, 19–28.

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. 2018. Distributional Policy Gradients. In *International Conference on Learning Representations*.

[3] Marc G Bellemare, Will Dabney, and Rémi Munos. 2017. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*. PMLR, 449–458.

[4] Yann Bouteiller, Simon Ramstedt, Giovanni Beltrame, Christopher Pal, and Jonathan Binas. 2021. Reinforcement Learning with Random Delays. In *International Conference on Learning Representations*.

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

[6] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. 2018. Implicit quantile networks for distributional reinforcement learning. In *International Conference on Machine Learning*. PMLR, 1096–1105.

[7] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, Vol. 32.

[8] Esther Derman, Gal Dalal, and Shie Mannor. 2021. Acting in Delayed Environments with Non-Stationary Markov Policies. In *International Conference on Learning Representations*.

[9] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. 2022. Distributional Soft Actor-Critic: Off-Policy Reinforcement Learning for Addressing Value Estimation Errors. *IEEE Transactions on Neural Networks and Learning Systems* 33, 11, 6584–6598.

[10] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9, 2419–2468.

[11] Mark Hauschild and Martin Pelikan. 2011. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation* 1, 3 (2011), 111–128.

[12] Wenchen He, Shaoyong Guo, Song Guo, Xuesong Qiu, and Feng Qi. 2020. Joint DNN partition deployment and resource allocation for delay-sensitive deep learning inference in IoT. *IEEE Internet of Things Journal* 7, 10, 9241–9254.

[13] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. 2022. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research* 23, 274, 1–18.

[14] Wanguo Jiao, Min Sheng, King-Shan Lui, and Yan Shi. 2014. End-to-end delay distribution analysis for stochastic admission control in multi-hop wireless networks. *IEEE Transactions on Wireless Communications* 13, 3, 1308–1320.

[15] Armin Karamzade, Kyungmin Kim, Montek Kalsi, and Roy Fox. 2024. Reinforcement learning from delayed observations via world models. *arXiv preprint arXiv:2403.12309*.

[16] Konstantinos V Katsikopoulos and Sascha E Engelbrecht. 2003. Markov decision processes with delays and asynchronous cost collection. *IEEE Trans. Automat. Control* 48, 4, 568–574.

[17] Jangwon Kim, Hangyeol Kim, Jiwook Kang, Jongchan Baek, and Soohee Han. 2023. Belief projection-based reinforcement learning for environments with delayed feedback. *Advances in Neural Information Processing Systems* 36, 678–696.

[18] Filip Krasniqi, Jocelyne Elias, Jérémie Leguay, and Alessandro EC Redondi. 2020. End-to-end delay prediction based on traffic matrix sampling. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 774–779.

[19] Xiaolong Li, Jun Cai, Junfeng Yang, Liyong Guo, Shaonian Huang, and Yunfei Yi. 2021. Performance analysis of delay distribution and packet loss ratio for body-to-body networks. *IEEE Internet of Things Journal* 8, 22, 16598–16612.

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

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

[22] Yulong Lu and Jianfeng Lu. 2020. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems* 33 (2020), 3094–3105.

[23] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. 2012. Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*.

[24] Daniel W Nam, Younghoon Kim, and Chan Y Park. 2021. GMAC: A distributional perspective on actor-critic framework. In *International Conference on Machine Learning*. PMLR, 7927–7936.

[25] Somjit Nath, Mayank Baranwal, and Harshad Khadilkar. 2021. Revisiting state augmentation methods for reinforcement learning with stochastic delays. In *ACM International Conference on Information & Knowledge Management*. 1346–1355.

[26] Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G Bellemare, and Will Dabney. 2019. Statistics and samples in distributional reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5528–5536.

[27] Kieran Strobel, Sibo Zhu, Raphael Chang, and Skanda Koppula. 2020. Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1969–1975.

[28] Aviv Tamar, Dotan Di Castro, and Shie Mannor. 2016. Learning the variance of the reward-to-go. *Journal of Machine Learning Research* 17, 13, 1–36.

[29] David Valensi, Esther Derman, Shie Mannor, and Gal Dalal. 2024. Tree Search-Based Policy Optimization under Stochastic Execution Delay. In *International Conference on Learning Representations*.

[30] Wei Wang, Dongqi Han, Xufang Luo, and Dongsheng Li. 2024. Addressing Signal Delay in Deep Reinforcement Learning. In *International Conference on Learning Representations*.

[31] Yunbo Wang, Mehmet C Vuran, and Steve Goddard. 2011. Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks. *IEEE/ACM Transactions on Networking* 20, 1, 305–318.

[32] Qingyuan Wu, Yuhui Wang, Simon Sinong Zhan, Yixuan Wang, Chung-Wei Lin, Chen Lv, Qi Zhu, Jürgen Schmidhuber, and Chao Huang. 2025. Directly Forecasting Belief for Reinforcement Learning with Delays. In *Forty-second International Conference on Machine Learning*.

[33] Qingyuan Wu, Simon Sinong Zhan, Yixuan Wang, Yuhui Wang, Chung-Wei Lin, Chen Lv, Qi Zhu, and Chao Huang. 2024. Variational Delayed Policy Optimization. In *Advances in Neural Information Processing Systems*, Vol. 37. 54330–54356.

[34] Qingyuan Wu, Simon Sinong Zhan, Yixuan Wang, Yuhui Wang, Chung-Wei Lin, Chen Lv, Qi Zhu, Jürgen Schmidhuber, and Chao Huang. 2024. Boosting Reinforcement Learning with Strongly Delayed Feedback Through Auxiliary Short Delays. In *International Conference on Machine Learning*.

[35] YaLou Yu, Bo Xia, Minzhi Xie, Xueqian Wang, Zhiheng Li, and Yongzhe Chang. 2023. Overcoming Delayed Feedback via Overlook Decision Making. In *IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 31–37.

[36] Tianchu Zhao, Sheng Zhou, Xueying Guo, Yun Zhao, and Zhisheng Niu. 2015. A Cooperative Scheduling Scheme of Local Cloud and Internet Cloud for Delay-Aware Mobile Cloud Computing. In *2015 IEEE Globecom Workshops (GC Wkshps)*. 1–6.

[37] Fan Zhou, Jianing Wang, and Xingdong Feng. 2020. Non-crossing quantile regression for distributional reinforcement learning. *Advances in Neural Information Processing Systems* 33, 15909–15919.