DSAC: Distributional Soft Actor-Critic for Risk-Sensitive Reinforcement Learning

XIAOTENG MA*, Department of Automation, Tsinghua University, China
JUNYAO CHEN*, School of Engineering and Applied Science, Columbia University, United States
LI XIA†, School of Business, Sun Yat-sen University, China
JUN YANG, Department of Automation, Tsinghua University, China
QIANCHUAN ZHAO, Department of Automation, Tsinghua University, China
ZHENGYUAN ZHOU, Stern School of Business, New York University, United States

We present Distributional Soft Actor-Critic (DSAC), a distributional reinforcement learning (RL) algorithm that combines the strengths of distributional information of accumulated rewards and entropy-driven exploration from Soft Actor-Critic (SAC) algorithm. DSAC models the randomness in both action and rewards, surpassing baseline performances on various continuous control tasks. Unlike standard approaches that solely maximize expected rewards, we propose a unified framework for risk-sensitive learning, one that optimizes the risk-related objective while balancing entropy to encourage exploration. Extensive experiments demonstrate DSAC's effectiveness in enhancing agent performances for both risk-neutral and risk-sensitive control tasks.

JAIR Associate Editor: Scott Sanner

JAIR Reference Format:
Xiaoteng Ma, Junyao Chen, Li Xia, Jun Yang, Qianchuan Zhao, and Zhengyuan Zhou. 2025. DSAC: Distributional Soft Actor-Critic for Risk-Sensitive Reinforcement Learning. *Journal of Artificial Intelligence Research* 83, Article 4 (June 2025), 28 pages. DOI: 10.1613/jair.1.17526

1 Introduction

In the past few years, model-free deep reinforcement learning [59] has been a powerful and applicable paradigm [21, 34, 53]. Such well-founded optimism stems from the substantial performance gains that deep neural networks can potentially unlock in reinforcement learning — whether by parameterizing value functions or policies — by leveraging their powerful representational capacity. However, simply throwing in "deep function approximator" is far from sufficient. Building effective RL algorithms often requires well-integrating accurate and efficient function approximation.

*Equal contribution.
†Corresponding author.

Authors' Contact Information: Xiaoteng Ma, ORCID: 0000-0002-7250-6268, pony.xtma@gmail.com, Department of Automation, Tsinghua University, Beijing, China; Junyao Chen, ORCID: 0009-0002-5849-2020, junyao.chen@columbia.edu, School of Engineering and Applied Science, Columbia University, New York, NY, United States; Li Xia, ORCID: 0000-0001-9141-2569, xiali5@sysu.edu.cn, School of Business, Sun Yat-sen University, Guangzhou, Guangdong, China; Jun Yang, ORCID: 0000-0002-9386-5825, yangjun603@tsinghua.edu.cn, Department of Automation, Tsinghua University, Beijing, China; Qianchuan Zhao, ORCID: 0000-0002-7952-5621, zhaoqc@tsinghua.edu.cn, Department of Automation, Tsinghua University, Beijing, China; Zhengyuan Zhou, ORCID: 0000-0002-0005-9411, zzhou@stern.nyu.edu, Stern School of Business, New York University, New York, NY, United States.

DOI: 10.1613/jair.1.17526

One important such aspect is exploration: since current actions influence the future state and hence the future rewards of the underlying Markov Decision Process [7, 42], effective exploration is a key aspect of RL algorithms. In model-free RL literature, randomness in action space is widely employed as a primitive to balance exploration and exploitation. On-policy algorithms, such as A3C [33], TRPO [50] and PPO [51], use stochastic action space and optimize the parameters by policy gradient. However, on-policy algorithms are not data-efficient as they require new experience for policy evaluation. On the other hand, off-policy methods can reuse past experience and hence improve data efficiency. However, most off-policy methods, such as DQN [35], DDPG [64] and TD3 [19], inherit the simple ϵ-greedy strategy from Q-learning by adding a small noise to a deterministic policy for exploration, where the scale of action perturbation is hard to choose in practice. To improve robustness of off-policy methods, SVG [24] introduces re-parameterized stochastic policy, while its objective is still standard maximum expected discounted returns.

The *maximum entropy* (MaxEnt) RL [22, 37] has shown promise for encouraging the randomness (and hence diversity) of actions by formally formulating the entropy of a policy into the objective. This maximum entropy approach is based on theoretical principles and has been applied to inverse reinforcement learning [71, 72] and optimal control [44, 61]. Haarnoja et al. [22] propose soft Actor-Critic (SAC), a continuous action space algorithm that has achieved superior performance in many continuous action control tasks. SAC computes an optimal policy by minimizing the KL-divergence between the action distribution and the exponential form of the soft action-value function. Many works have been done for understanding the effectiveness of the maximum entropy objective [1, 11, 27, 49], which reveal that objective with entropy regularization enjoys the smoother optimization landscape and the faster convergence rate, and builds connections between RL with probabilistic graphical models and convex optimization.

Concurrently, *distributional RL* [4, 36, 56] considers the whole distribution of value functions, rather than just the expectation, to make more informed decisions that lead to superior rewards. Incidentally, experiments show that similar encouraging mechanisms also exist in human brains [14]. To address the challenge of approximating the distribution of value functions, two categories of approaches dominate: learning discrete categorical distributions (CDRL) [3–5, 43, 46] and learning quantiles of distributions (QDRL) [9, 15, 16, 48, 68]. Comparing with CDRL, QDRL learns quantiles of distribution directly without any assumption about the range of return. Moreover, quantiles are naturally related to risk measures. Thus, QDRL shows more potential than CDRL in recent works.

SAC and distributional RL each have limitations: SAC only considers the first moment of values, while distributional RL lacks action diversity for exploration. This leads to the question: *Can we exploit randomness to enhance action diversity and leverage distributional information?* As demonstrated in this paper, the answer is yes. Previous work on integrating of SAC and distributional RL assumes a Gaussian distribution on the return distribution Z for its parametrization [17], which may not capture skewed or heavy-tailed returns. Our method uses quantile regression, providing a non-parametric model that better captures asymmetries and tail risks, leading to more robust decision-making under uncertainty.

1.1 Our Contributions

Our contributions are threefold. First, we present Distributional Soft Actor-Critic (DSAC), that combines MaxEnt RL with distributional RL, by leveraging the distributional information of value functions in SAC and encouraging action randomness through entropy in distributional RL. We define a distributional soft Bellman operator and use quantile regression to estimate the soft discounted returns for continuous action tasks.

Second, we adapt DSAC for risk-sensitive learning, providing a unified framework that handles multiple risk measures, such as variance [40, 56, 66], CVaR [13] and CPT [63]. Aforementioned risk measures all involve higher moments of the underlying value distribution while they do not satisfy the Bellman equation. Prior work [15, 55] has demonstrated the ability to handle multiple risk measures. Inspired by these foundations, our approach

supports multiple metrics and allows for optimizing risk while balancing the entropy of policy, ensuring the exploratory capability and robustness of the policy in risky scenarios.

Third, we perform extensive experiments comparing DSAC with existing model-free RL algorithms in continuous state and actions spaces. DSAC achieves superior performance on standard benchmarks (MuJoCo and Box2d in OpenAI Gym), surpassing SAC, TD3, SDPG, and D4PG. We also demonstrated the effectiveness of our risk-sensitive RL framework through simulation experiments.

2 Background

We operate within the context of a standard MDP [42], characterized by the tuple $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$. Here, $\mathcal{S}$ and $\mathcal{A}$ represent the continuous state and action spaces, respectively. The transition probability density from one state to another, given an action, is denoted by $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ and is considered unknown. The reward function is represented by $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and $\gamma \in (0, 1)$ is the discount factor. A policy $\pi : \mathcal{S} \to \mathscr{P}(\mathcal{A})$ is a mapping from each state to a probability distribution over actions in $\mathcal{A}$. The set of all policies is denoted by Π.

The goal of standard RL is to maximize the expected sum of discounted rewards, given by:

$$\mathcal{J}(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t\right) \right], \tag{1}$$

with the initial state s_0 distributed according to $d_0(s)$.

For a given policy $\pi \in \Pi$, the *action-value function* $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as:

$$Q^\pi(s, a) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim P(\cdot \mid s_t, a_t), s_0 = s, a_0 = a.$$

The *Bellman operator* $\mathcal{T}^\pi$ and the *Bellman optimality operator* $\mathcal{T}^*$ are defined as:

$$\begin{aligned} \mathcal{T}^\pi Q(s, a) &:= \mathbb{E}\left[R(s, a)\right] + \gamma \mathbb{E}_{P,\pi}[Q\left(s', a'\right)], \\ \mathcal{T}^* Q(s, a) &:= \mathbb{E}\left[R(s, a)\right] + \gamma \max_{a'} \mathbb{E}_P[Q\left(s', a'\right)]. \end{aligned} \tag{2}$$

Applying either operator iteratively from some initial Q_0 will converge to its fixed point Q^π or Q^* at a geometric rate, as both operators are contractive [8].

2.1 Maximum Entropy Reinforcement Learning

MaxEnt RL diverges from standard RL by maximizing the sum of rewards while simultaneously maximizing the entropy of the policy. The objective function of MaxEnt RL is given by:

$$\mathcal{J}(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(R\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)\right) \right]. \tag{3}$$

where α is a temperature parameter. The *soft action-value function* $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ for a policy $\pi \in \Pi$ is defined as:

$$\begin{aligned} Q^\pi(s, a) := \mathbb{E}_\pi \left[R(s, a) + \sum_{t=1}^{\infty} \gamma^t [R(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)] \right], \\ a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim P(\cdot \mid s_t, a_t), s_0 = s, a_0 = a. \end{aligned}$$

By augmenting the standard Bellman operator $\mathcal{T}^\pi$ with an entropy regularization term, we define the *soft Bellman operator* $\mathcal{T}_S^\pi$ and the *soft optimality Bellman operator* $\mathcal{T}_S^*$ [37] as:

$$\begin{aligned}\mathcal{T}_S^\pi Q(s,a) &:= \mathbb{E}[R] + \gamma \mathbb{E}_{P,\pi}\left[Q(s',a') - \alpha \log \pi(a' \mid s')\right], \\ \mathcal{T}_S^* Q(s,a) &:= \mathbb{E}[R] + \gamma \mathbb{E}_P\left[\alpha \log(\| \exp(Q(s',\cdot)/\alpha)\|_1)\right].\end{aligned} \tag{4}$$

The soft Bellman operators retain the γ-contraction property of the original Bellman operators. The key distinction between $\mathcal{T}_S^*$ and $\mathcal{T}^*$ is that the unique fixed point of $\mathcal{T}_S^*$ corresponds to a unique stochastic policy in the softmax form of Q^*: $\pi^*(\cdot \mid s) \propto \exp(Q^*(s,\cdot)/\alpha)$.

Haarnoja et al. [22] introduced the Soft Actor-Critic (SAC) algorithm for learning policies in continuous action spaces with a MaxEnt objective function. SAC alternates between soft policy evaluation, implemented by repeatedly applying $\mathcal{T}_S^\pi$, and soft policy improvement until convergence is achieved.

The soft policy improvement is realized by minimizing the Kullback-Leibler divergence between the policy distribution and the exponential form of the soft action-value function:

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\cdot \mid s) \,\|\, \frac{\exp\left(Q^{\pi_{\text{old}}}(s,\cdot)/\alpha\right)}{\Delta^{\pi_{\text{old}}}(s)}\right), \tag{5}$$

where $\Delta^{\pi_{\text{old}}}$ is the partition function that normalizes the distribution.

2.2 Distributional Reinforcement Learning

Distributional RL extends MaxEnt RL by accounting for the randomness in accumulated discounted returns. The objective function of distributional RL aligns with traditional RL, aiming to maximize the discounted return as in Equation 1.

Let $\mathcal{Z}$ denote the action-value distribution space with finite moments. The *distributional Bellman operator* $\mathcal{T}_D^\pi : \mathcal{Z} \to \mathcal{Z}$ [4] is defined as:

$$\mathcal{T}_D^\pi Z(s,a) \overset{D}{:=} R(s,a) + \gamma Z(s',a'), s' \sim P(\cdot \mid s,a), a' \sim \pi(\cdot \mid s'), \tag{6}$$

where $U \overset{D}{:=} V$ indicates that random variables U and V have the same distribution.

In the control setting, the *distributional Bellman optimality operator* $\mathcal{T}_D^*$ is defined as:

$$\mathcal{T}_D^* Z(s,a) \overset{D}{:=} R(s,a) + \gamma Z(s',a^*), s' \sim P(\cdot \mid s,a), a^* \in \arg\max_{a'} \mathbb{E}[Z(s',a')]. \tag{7}$$

To measure the distance between value distributions, the p-Wasserstein distance is employed, defined as:

$$d_p(U,V) = \left(\int_0^1 \left|F_U^{-1}(\omega) - F_V^{-1}(\omega)\right|^p d\omega\right)^{1/p}, \tag{8}$$

where F_U and F_V are the cumulative distribution functions (CDFs) of two random variables U and V. For $Z_1, Z_2 \in \mathcal{Z}$, the supremum-p-Wasserstein metric $\bar{d}_p$ over value distributions is given by:

$$\bar{d}_p(Z_1,Z_2) = \sup_{s,a} d_p\left(Z_1(s,a), Z_2(s,a)\right). \tag{9}$$

The operator $\mathcal{T}_D^\pi$ is a γ-contraction in $\bar{d}_p$. Although $\mathcal{T}_D^*$ does not maintain the contractive property of any distributional metric, convergence to the optimal action-state function Q^* can still be attained by taking the expectation $\mathbb{E}[Z]$ [4].

A primary challenge in distributional RL is the approximation of value distributions. Quantile Distributional Reinforcement Learning (QDRL) has emerged as a popular method for distribution approximation [15, 16, 67]. In QDRL, the return distribution is projected onto a parameterized quantile distribution, expressed as $Z(s,a) = \frac{1}{K}\sum_{k=0}^{K-1} \delta_{z_k}(s,a)$, where δ_z is a Dirac measure, z_k is a quantile atom located at the τ_k-quantile with a total number

of K atoms. The d_1 distance between the quantile distribution and the target is minimized using the quantile regression loss:

$$z_k = \arg\min_q \mathbb{E}_Z \left[\left(\tau_k \mathbb{1}_{Z>q} + (1-\tau_k) \mathbb{1}_{Z \le q} \right) |Z - q| \right],$$

where $\mathbb{1}(\cdot)$ is the indicator function and q is the candidate location for the τ_k-quantile..

3 Distributional Soft Actor-Critic

In this section, we introduce our novel algorithm, the *Distributional Soft Actor-Critic* (DSAC). Initially, we propose a new Bellman operator and develop a policy iteration scheme known as Distributional Soft Policy Iteration (DSPI), which is proven to converge to the optimal value distribution. Subsequently, we detail our DSAC algorithm, which effectively implements DSPI within an Actor-Critic framework. The specifics for training the distributional critic using quantile regression and double learning are provided at the end of this section.

3.1 Distributional Soft Policy Iteration

DSAC is designed to optimize the policy to maximize the objective function outlined in Equation 3, in line with the principles of maximum entropy reinforcement learning (MaxEnt RL). Accounting for the inherent randomness in both rewards and actions, we define the *soft action-value distribution* Z^π for a policy $\pi \in \Pi$ as follows:

$$\begin{aligned} Z^\pi(s,a) &:= R(s,a) + \sum_{t=1}^{\infty} \gamma^t \left(R(s_t,a_t) - \alpha \log \pi(a_t \mid s_t) \right), \\ & a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim P(\cdot \mid s_t, a_t), s_0 = s, a_0 = a. \end{aligned} \tag{10}$$

We introduce the *distributional soft Bellman operator* $\mathcal{T}_{DS}^\pi$ as:

$$\begin{aligned} \mathcal{T}_{DS}^\pi Z(s,a) &:= R(s,a) + \gamma \left(Z\left(s',a'\right) - \alpha \log \pi(a' \mid s') \right), \\ & s' \sim P(\cdot \mid s,a), a' \sim \pi(\cdot \mid s'). \end{aligned} \tag{11}$$

This operator harmoniously combines the soft Bellman operator $\mathcal{T}_S^\pi$ from Equation 4 and the distributional Bellman operator $\mathcal{T}_D^\pi$ from Equation 6, inheriting the convergence property from its constituent operators.

Lemma 1. *The operator $\mathcal{T}_{DS}^\pi : \mathcal{Z} \to \mathcal{Z}$ is a γ-contraction with respect to the distance metric $\bar{d}_p$.*

Armed with this new operator, we can formulate an algorithm akin to soft policy iteration, termed *distributional soft policy iteration*. This algorithm is divided into two phases: *distributional soft policy evaluation* and *distributional soft policy improvement*. For any given policy π, the soft action-value distribution Z^π is obtained by iteratively applying $\mathcal{T}_{DS}^\pi$.

Lemma 2 (Distributional Soft Policy Evaluation). *Let $Z_{k+1} := \mathcal{T}_{DS}^\pi Z_k$ with $Z_0 \in \mathcal{Z}$. The sequence $\{Z_k\}$ will asymptotically converge to Z^π as $k \to \infty$.*

Once the value distribution is accurately evaluated, policy improvement is achieved by solving the optimization problem defined in Equation 5.

Lemma 3 (Distributional Soft Policy Improvement). *Let $Q^\pi(s,a) := \mathbb{E}[Z^\pi(s,a)]$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Given an old policy $\pi_{\text{old}} \in \Pi$ and a new policy π_{new} obtained by solving the problem in Equation 5, it holds that $Q^{\pi_{\text{old}}}(s,a) \le Q^{\pi_{\text{new}}}(s,a)$.*

The optimal policy π^* is derived from Q^*, the expectation of the optimal action-value distribution Z^*. This derivation involves a many-to-one mapping, as multiple distributions Z^* could yield identical expected values. However, since the entropy regularization term $\alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)$ in DSAC objective function $\mathcal{J}(\theta)$, extended from

SAC, is a convex function in $\pi^*(\cdot|s)$ at its optimality, this constraint in $\mathcal{J}(\theta)$ allows the optimal policy achieves its uniqueness at minimization for the current policy π_t [20]. Iteratively applying distributional soft policy evaluation and improvement will result in the unique optimal policy π^*.

The uniqueness of π^* implies the uniqueness of the Z^*. This distribution can be obtained by repeatedly applying he distributional soft Bellman operator $\mathcal{T}_{DS}^{\pi}$ to π_t. Moreover, to understand the convergence of the soft action-value distribution in the control setting, we define the *distributional soft Bellman optimality operator* $\mathcal{T}_{DS}^{*}$ as:

$$\mathcal{T}_{DS}^{*} = \mathcal{T}_{DS}^{\mu}, \quad \mu(\cdot \mid s) \propto \exp(\mathbb{E}[Z(s,\cdot)]/\alpha). \tag{12}$$

Theorem 1 (Convergence in the Control Setting). *Let $Z_{k+1} := \mathcal{T}_{DS}^{*} Z_k$ with $Z_0 \in \mathcal{Z}$. The sequence $\{Z_k\}$ will converge to Z^* as $k \to \infty$.*

Unlike $\mathcal{T}_{D}^{*}$, which leads to a uniform convergence of distributions to a set of optimal state-action distributions [4], the repeated application of $\mathcal{T}_{DS}^{*}$ results in a unique Z^*. This characteristic demonstrates that entropy in the objective function not only enhances exploration but also refines the estimation of the value distribution.

We refer to this phenomenon as the *smoothing effect* of entropy in distribution estimation, which will be further illustrated through a simple experiment in Section 5.1. Although we have established the convergence of the value distribution, $\mathcal{T}_{DS}^{*}$ is not a contraction. However, entropy still plays a constructive role in preserving the contractive property, that is, it aids in bringing the updated distribution closer to the optimal distribution. A detailed discussion on this aspect is reserved for the appendix.

The proofs for the aforementioned theoretical results are provided in Appendix A.

3.2 The DSAC Algorithm

The Distributional Soft Actor-Critic (DSAC) algorithm is structured around an Actor-Critic framework, comprising a distributional soft value network, denoted as $Z_\tau(s,a;\theta)$, which serves as the critic, and a stochastic policy network $\pi(a \mid s;\phi)$, which is the actor. We begin by detailing the training process for the critic.

The *quantile function* F_Z^{-1} for a random variable Z is defined as the inverse of its cumulative distribution function (CDF) $F_Z(z) = Pr(Z < z)$. Mathematically, this is expressed as $F_Z^{-1}(\tau) := \inf\{z \in \mathbb{R} : \tau \le F_Z(z)\}$, where τ represents the quantile fraction. In the subsequent discussion, we use $Z_\tau := F_Z^{-1}(\tau)$.

For a given state s and action a, the action-value distribution is approximated by a set of quantile fractions $\{\tau_i\}_{i=0,\dots,N}$, with $\tau_0 = 0, \tau_N = 1$, $\tau_i < \tau_j, \forall i < j$, and $\tau_i \in [0,1], i = 0,\dots,N$. The midpoint of each pair of consecutive quantile fractions is $\hat{\tau}_i = (\tau_i + \tau_{i+1})/2$. Quantile approximation methods include QR-DQN [16], where the value distribution is approximated by a group of trainable quantile values at fixed quantile fractions. IQN [15] randomly samples the quantile fractions from a uniform distribution. FQF [67] introduces a trainable proposal network to generate τ. Later on, NC-QR-DQN [69] was proposed to solve the quantile crossing issue. NDQFN [70] learns a baseline quantile value and then adds non-negative increments to generate monotonic quantile values. A newer method SPL-DQN [29] learns continuous quantile functions represented by monotonic rational quadratic spline. DSAC accommodates these methods flexibly as a modular choice. See appendix B for ablation results on QR-DQN, IQN and FQF.

According to Equation 11, the pairwise temporal difference (TD) error between two quantile fractions $\hat{\tau}_i$ and $\hat{\tau}_j$ is given by:

$$\delta_{ij}^{t} = r_t + \gamma \left[Z_{\hat{\tau}_i}\left(s_{t+1}, a_{t+1}; \bar{\theta}\right) - \alpha \log \pi(a_{t+1} \mid s_{t+1}; \bar{\phi}) \right] - Z_{\hat{\tau}_j}\left(s_t, a_t; \theta\right), \tag{13}$$

where $\bar{\theta}$ and $\bar{\phi}$ represent the parameters of the target action-value distribution network and target policy, respectively. The target networks are softly updated to maintain stability during training.

We adopt quantile regression to train the $Z_\tau(s,a;\theta)$ network by minimizing the weighted pairwise Huber regression loss across quantile fractions. This choice is consistent with prior works (QR-DQN, NC-QR-DQN, FQF,

Algorithm 1 DSAC update

Parameter: N, κ
Input: $s, a, r, s', \gamma \in (0, 1)$
Generate quantile fractions $\tau_i, i = 0, \ldots, N, \tau_j, j = 0, \ldots, N$
Update Quantile Value Network $Z(s, a; \theta_k)$
Get next actions for calculating target $a' \sim \pi(\cdot \mid s'; \bar{\phi})$
for $i = 0$ **to** $N - 1$ **do**
  for $j = 0$ **to** $N - 1$ **do**
    $y_i = \min_{k=1,2} Z_{\hat{\tau}_i}\left(s', a'; \bar{\theta}_k\right)$
    $\delta_{ij}^k = r + \gamma\left[y_i - \alpha \log \pi(a' \mid s'; \bar{\phi})\right] - Z_{\hat{\tau}_j}\left(s, a; \theta_k\right), k = 1, 2$
  end for
end for
$\mathcal{J}_Z(\theta_k) = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(\tau_{i+1} - \tau_i)\varrho_{\hat{\tau}_j}^{\kappa}\left(\delta_{ij}^k\right), k = 1, 2$
Update θ_k with $\nabla\mathcal{J}_Z(\theta_k), k = 1, 2$
Update $\bar{\theta}_k \leftarrow \iota\theta_k + (1 - \iota)\bar{\theta}_k, k = 1, 2$
Update Policy Network $\pi(a \mid s; \phi)$
Get new actions with re-parameterized samples $\tilde{a} \sim \pi(\cdot \mid s; \phi)$
$Q(s, \tilde{a}) = \sum_{i=0}^{N-1}\left(\tau_{i+1} - \tau_i\right)\min_{k=1,2} Z_{\hat{\tau}_i}\left(s, \tilde{a}; \theta_k\right)$
$\mathcal{J}_\pi(\phi) = \alpha \log(\pi(\tilde{a} \mid s; \phi)) - Q(s, \tilde{a})$
Update ϕ with $\nabla\mathcal{J}_\pi(\phi)$
Update $\bar{\phi} \leftarrow \iota\phi + (1 - \iota)\bar{\phi}$

NDQFN) to ensure fair comparison across baselines. The Huber quantile regression loss [25], with a threshold κ, is defined as:

$$\varrho_\tau^\kappa\left(\delta_{ij}\right) = \left|\tau - \mathbb{1}_{\delta_{ij}<0}\right| \frac{\mathcal{L}_\kappa\left(\delta_{ij}\right)}{\kappa}, \text{ with}$$
$$\mathcal{L}_\kappa\left(\delta_{ij}\right) = \begin{cases} \frac{1}{2}\delta_{ij}^2, & \text{if } \left|\delta_{ij}\right| \le \kappa \\ \kappa\left(\left|\delta_{ij}\right| - \frac{1}{2}\kappa\right), & \text{otherwise.} \end{cases}$$

The objective function for the quantile value network $Z_\tau(s, a; \theta)$ is then:

$$\mathcal{J}_Z(\theta) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(\tau_{i+1} - \tau_i)\varrho_{\hat{\tau}_j}^{\kappa}\left(\delta_{ij}^t\right), \tag{14}$$

where each $\varrho_{\hat{\tau}_j}$ is weighted by the target distribution fractions $\tau_{i+1} - \tau_i$.

DSAC addresses the overestimation issue common in off-policy continuous action RL algorithms when calculating the target value. By extending the double learning concept from TD3 to DSAC, we employ two networks with the same structure, parameterized by $\theta_k, k = 1, 2$, and trained to fit a conservative target. For two quantile fractions $\hat{\tau}_i$ and $\hat{\tau}_j$, the TD-error is redefined as:

$$y_i^t = \min_{k=1,2} Z_{\hat{\tau}_i}\left(s_{t+1}, a_{t+1}; \bar{\theta}_k\right),$$
$$\delta_{ij}^{t,k} = r_t + \gamma\left(y_i^t - \alpha \log \pi(a_{t+1} \mid s_{t+1}; \bar{\phi})\right) - Z_{\hat{\tau}_j}\left(s_{t+1}, a_{t+1}; \theta_k\right).$$

The policy network $\pi(a \mid s; \phi)$ is trained by adapting the SAC algorithm using the parameterized quantile function. The action-value function is derived by taking the expectation over the quantile values:

$$Q(s, a; \theta) = \sum_{i=0}^{N-1} (\tau_{i+1} - \tau_i) \, Z_{\hat{\tau}_i}(s, a; \theta) \,. \tag{15}$$

In the double variant DSAC, $Q(s, a; \theta) = \min_{k=1,2} Q(s, a; \theta_k)$. To solve the minimization problem in Equation 5, SAC samples actions using a re-parameterized policy neural network $f(s, \epsilon; \phi)$, with ϵ being a noise vector drawn from a fixed distribution, such as a standard spherical Gaussian. The original problem is then solved using gradient descent with the objective:

$$\mathcal{J}_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}}[\alpha \log \pi(f(s_t, \epsilon_t; \phi) \mid s_t) - Q(s_t, f(s_t, \epsilon_t; \phi); \theta)],$$

where $\mathcal{D}$ represents the transitions replay buffer. The complete algorithm is outlined in Algorithm 1.

4 A Risk-sensitive RL Framework

The distribution of returns encompasses a wealth of information beyond mere expectations, enabling the consideration of diverse risk metrics. In this section, we expand the foundational DSAC to encompass risk-sensitive reinforcement learning (RL) settings, presenting a unified framework capable of optimizing the majority of conventional risk metrics. We designate this risk-sensitive variant of DSAC as RDSAC.

4.1 Risk-sensitive Policy Learning

While risk-sensitive RL has been extensively investigated in the literature [13, 40, 52], the direct estimation of most risk measures poses a challenge due to the loss of linearity in Bellman's equation [42]. Our approach provides a unified framework that optimizes policies under different risk metrics in complex, continuous control environments. Leveraging the distribution of returns, it approximates the value function under a risk measure.

A *risk measure* $\rho : \mathcal{Z} \to \mathbb{R}$ is a function that maps an uncertain outcome Z to a real number. The expectation is considered a risk-neutral measure function, denoted as $\rho[\cdot] = \mathbb{E}[\cdot]$. Beyond expectation, a multitude of risk preferences can be articulated through risk measures, as discussed in Section 4.2.

In our approach, we focus on the risk associated with the reward function, rather than the risk of the entropy. To address external rewards and the entropy bonus separately, we partition Z^π into two components: the reward distribution Z_R^π and the entropy distribution Z_H^π:

$$\begin{aligned} Z_R^\pi(s, a) &\overset{D}{:=} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t), \\ Z_H^\pi(s, a) &\overset{D}{:=} -\sum_{t=1}^{\infty} \gamma^t \log \pi(a_t \mid s_t), \\ a_t &\sim \pi(\cdot \mid s_t), s_{t+1} \sim P(\cdot \mid s_t, a_t), s_0 = s, a_0 = a. \end{aligned} \tag{16}$$

We establish $Z^\pi \overset{D}{=} Z_R^\pi + \alpha Z_H^\pi$, with Z_R^π and Z_H^π satisfying their respective distributional Bellman equations:

$$\begin{aligned} \mathcal{T}_R^\pi Z(s, a) &\overset{D}{:=} R(s, a) + \gamma Z(s', a'), \\ \mathcal{T}_H^\pi Z(s, a) &\overset{D}{:=} \gamma (Z(s', a') - \alpha \log \pi(a' \mid s')), \\ s' &\sim P(\cdot \mid s, a), a' \sim \pi(\cdot \mid s'). \end{aligned} \tag{17}$$

Consequently, we can train Z_R^π and Z_H^π using quantile regression. In RDSAC, we employ two quantile networks to parameterize Z_R^π and Z_H^π, sharing parameters except for the final layer of the network.

The risk-sensitive policy is derived by optimizing the following objective:

$$\mathcal{J}_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},a\sim\pi_\phi}[\alpha \log \pi\left(a \mid s_t;\phi\right) - \alpha\mathbb{E}[Z_H\left(s_t,a;\theta_H\right)] - \rho[Z_R\left(s_t,a;\theta_R\right)]]$$

where θ_R and θ_H represent the network parameters for Z_R^π and Z_H^π, respectively. By jointly optimizing the entropy and risk value functions, RDSAC integrates risk measures into training while maintaining a diverse policy landscape.

4.2 Common Risk Measures

We detail several prevalent risk measures below, elucidating how RDSAC can optimize them. Initially, we introduce a category of risk measures known as *distorted expectation*, owing to their uniform approximation form. To showcase the adaptability of RDSAC, we also discuss mean-semideviation, a measure that does not fall under the distorted expectation category. MSD penalizes only downside deviations from the mean, making it suitable for risk-averse decision-making scenarios with asymmetrical outcome distributions.

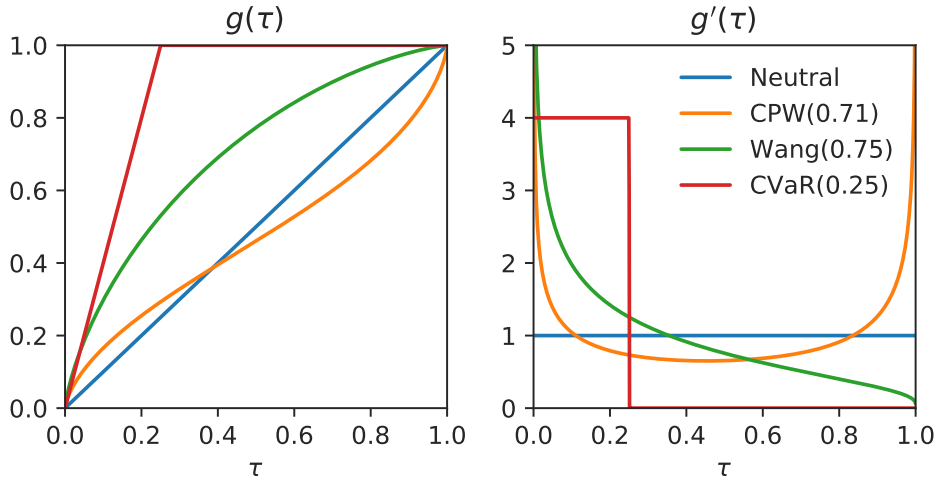


Fig. 1. Different distortion functions and their derivations.

Distorted Expectation. Distorted expectation is a risk weighted expectation of value distribution under a specific distortion function [2]. By saying a *distortion function*, we mean a non-decreasing function $g : [0,1] \to [0,1]$ satisfying $g(0) = 0$ and $g(1) = 1$. The distorted expectation of Z under g is defined as $\rho_{\mathrm{DE}}[Z] = \int_0^1 F_Z^{-1}(\tau)dg(\tau)$.

Different distortion functions encapsulate distinct risk propensities. We enumerate some prevalent distortion functions as follows:

- **CVaR** The *Conditional Value at Risk* (CVaR) quantifies risk as the conditional expectation of losses exceeding (rewards below) a specified quantile β [13, 45]. For a random variable Z with the CDF $F_Z(z)$ and a confidence level $\beta \in [0,1]$, CVaR is expressed as $\rho_{\mathrm{CVaR}}[Z] = \mathbb{E}\left[Z \mid Z \le F_Z^{-1}(\beta)\right]$. The corresponding distortion function is defined as $g(\tau) = \min\{\tau/\beta, 1\}$. Estimating CVaR precisely is complex, as only β fraction of the data can influence the CVaR value.
- **Wang** The *Wang* distortion risk measure [65] is given by $g(\tau) = \Phi(\Phi^{-1}(\tau) + \beta)$, where Φ and Φ^{-1} represent the standard Normal CDF and its inverse, respectively. $\beta > 0$ indicates risk aversion, while $\beta < 0$ suggests

risk seeking. Unlike CVaR, Wang distributes weights across the entire τ interval and modulates the weight intensity based on β.

- **CPW** *Cumulative Probability Weighting* (CPW) parameterization is defined as $g(\tau) = \tau^\beta / \left(\tau^\beta + (1-\tau)^\beta\right)^{\frac{1}{\beta}}$. It originates from cumulative prospect theory [63]. CPW is sensitive to the extremities when $\tau \to 0$ or $\tau \to 1$, mirroring human decision-makers. Tversky and Kahneman [63] determined that $\beta = 0.71$ closely aligns with human subjects. CPW is not strictly risk-seeking or risk-averse but represents a hybrid.

We propose two methods for approximating distorted expectation with sampling. First, drawing inspiration from inverse transform sampling, we have $\rho_{\text{DE}}[Z] = \int_0^1 F_Z^{-1}(g^{-1}(\iota))d\iota$, where $\iota = g(\tau)$. Thus, we can estimate the distorted action-value as:

$$\hat{\rho}_{\text{DE}}[Z(s,a)] = \sum_{i=0}^{N-1} (\tau_{i+1} - \tau_i)\, Z_{g^{-1}(\hat{\tau}_i)}(s,a;\theta).$$

This method is viable when g^{-1} has a closed-form solution, such as with CVaR and Wang. Second, recognizing that $\rho_{\text{DE}}[Z] = \int_0^1 g'(\tau) F_Z^{-1}(\tau) d\tau$, it is evident that the expectation is distorted by $g'(\tau)$. Consequently, the distorted action-value can be approximated as:

$$\hat{\rho}_{\text{DE}}[Z(s,a)] = \sum_{i=0}^{N-1} (\tau_{i+1} - \tau_i) g'(\hat{\tau}_i) Z_{\hat{\tau}_i}(s,a;\theta).$$

This approach is primarily utilized for CPW.

Mean-Semideviation. Variance-related metrics form another significant class of risk measures [41, 56, 60]. While variance is an intuitive measure of uncertainty, fluctuation, and robustness by incorporating higher moment information, semideviation [38] distinguishes itself by capturing the variation on only one side of the return distribution. Specifically, the square root of semivariance [30, 32] is known as the semideviation (SD). For risk-averse policies, we penalize the downside semideviation as $\mathbb{SD}[Z] := \mathbb{E}[(Z - \mathbb{E}[Z])_-^2]^{1/2}$, where $(\cdot)_- = \min(\cdot, 0)$. Rather than optimizing SD directly, we often consider a mean-semidiviation (MSD) objective $\rho_{\text{MSD}}[Z] = \mathbb{E}[Z] - \beta\mathbb{SD}[Z]$, which balances the mean with semidiviation.

As pointed out by Bellemare et al. [4], the distributional Bellman operator $\mathcal{T}_D^\pi$ is a contraction in variance, which means that convergence in the value distribution space also leads to a good estimation of variance. Under the distributional Bellman operator $\mathcal{T}_D^\pi$, the variance can be approximated as:

$$\hat{\rho}_{\text{SD}}[Z(s,a)] = \sqrt{\sum_{i=0}^{N-1} (\tau_{i+1} - \tau_i) \left[Z_{\tau_i}(s,a;\theta) - Q(s,a)\right]_-^2}$$

the MSD can be approximated as:

$$\hat{\rho}_{\text{MSD}}[Z(s,a)] = Q(s,a) - \beta\sqrt{\sum_{i=0}^{N-1} (\tau_{i+1} - \tau_i) \left[Z_{\tau_i}(s,a;\theta) - Q(s,a)\right]_-^2}$$

where $Q(s,a) = \mathbb{E}[Z(s,a)]$ is the expected value of the return distribution.

5 Experiments

In this section, we conduct experiments to answer the following questions:

- Are MaxEnt RL and distributional RL better together than they are alone?
- Why would incorporating the policy entropy into distribution learning be helpful?

- Can the agent learn in high-stack, complex environments?
- How the policy performs with different risk measure?

Three groups of experiments are designed to address these questions above.

5.1 Toy Example

We give a toy example to illustrate the benefits of entropy in distributional RL. Here is a chain environment (see Figure 2), which contains N states. For each state, the agent can choose either a_0 (going right) or a_1 (going up). Both actions result in a noise reward $\mathcal{N}(0, \sigma)$, but the episode ends immediately after taking a_1. When the agent arrives at N-th state and goes right, it will achieve a final noisy reward $1 - \mathcal{E}(\lambda)$. In this paper, we set $\sigma = 0.1$ and $\lambda = 0.5$. Obviously, the optimal policy is keeping going right.

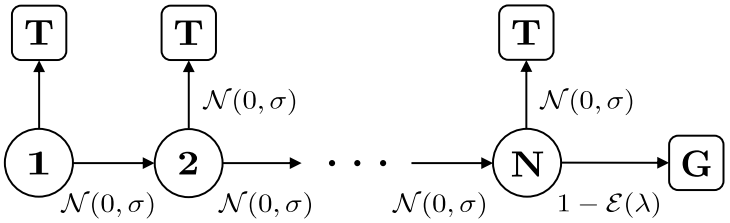


Fig. 2. A chain environment.

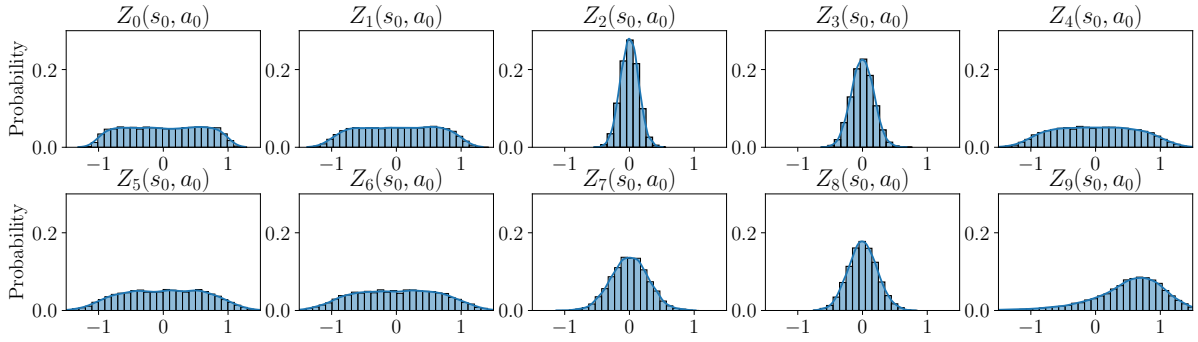
We consider distributional learning with or without entropy regularization respectively. For distributional RL without entropy regularization, the target distribution is taken from the greedy actions directly (see Equation 7). Its counterpart, distributional RL with entropy regularization (see Equation 12), targets the distribution as a mixed one under a softmax policy (the optimal policy with entropy regularization). We evaluate the two methods in the chain task above to show the benefit of entropy regularization in distribution learning, and visualize the target distributions of each iteration in Figure 3.

Smoothing involves modifying an objective function or constraints to make them more tractable, particularly with noisy functions. The *smoothing effect* has been observed in the expected RL with entropy on the loss optimization landscape in high dimensional environments [1]. We observes the smoothing effect on target action-value distributions in distributional RL. Since the standard policy learning process only considers the expectation of the whole distribution, the target is sensitive to the noise when Q-values of next actions are similar [4]. That phenomenon will do harm to the distributional learning, such as quantile regression. By contrast, the entropy regularized policy is stochastic and stable to the noise in Q estimates, leading to distribution updates with small shifts in training.

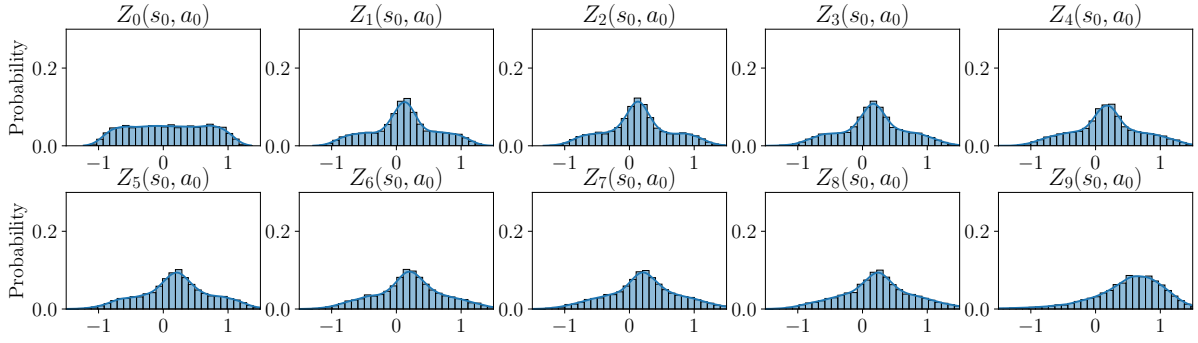
We illustrate such smoothing effect in Figure 3 that our target distributions are smoother with the entropy regularization. As demonstrated in toy example, before adding entropy, the distribution Z_k may incur abrupt shifts (Figure 3a). Examples of the abrupt shifts include from Z_1 to Z_2, from Z_3 to Z_4 and from Z_6 to Z_7. After adding entropy in distributional RL, the learning distribution Z_k exhibited small shifts from the previous steps (Figure 3b). We see that adding entropy leads to faster convergence.

5.2 Comparison with baselines

To evaluate our algorithm performance, we design a series of experiments to compare DSAC with some baselines for continuous control RL algorithms and test DSAC in different risk scenarios. We implement our algorithm based on *rlpyt* [57], a well-developed PyTorch [39] RL toolkit. All experiments are performed on a servers with 2 AMD EPYC 7702 64-Core Processor CPUs, 2x24-core Intel(R) Xeon(R) Platinum 8268 CPUs, and 8 Nvidia GeForce



(a) $Z_k(s_0, a_0)$ without entropy regularization



(b) $Z_k(s_0, a_0)$ with entropy regularization

Fig. 3. Visualization of the target state-action distributions $Z_k(s_0, a_0)$ distributions during first 10 iterative updates. s_0 is the initial state, the leftmost state in the toy example illustration. a_0 is the action of going right (optimal action).

RTX 2080 Ti GPUs. Hyper-parameters and implementation details are listed in Appendix B.1. The source code of our DSAC implementation[1] is available online.

We evaluate our algorithm with MuJoCo [62] and Box2d in OpenAI Gym [10]. We compare with two baselines for continuous control tasks: Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [19] and Soft Actor-Critic (SAC) [22]. To test whether the policy entropy is effective in distributional RL, we compare DSAC with two distributional RL methods, Sample-based Distributional policy gradient (SDPG) [54] and Distributed Distributional Deep Deterministic Policy Gradient algorithm (D4PG) [3], both distributional extension of DDPG. We also implement a distributional variant TD3 named Twin Delayed Deep Distributional Deterministic policy gradient (TD4), which includes techniques such as double learning to reduce the function approximation error. D4PG, SDPG, and TD4 adopt the same distributional critic as DSAC while excluding entropy information in training.

The results in Figure 4 and Table 1 show that DSAC outperforms other baselines. Moreover, in complex tasks such as Humanoid-v2 (which has a 17-dimensional action space) and BipealWalkerHardcore-v3 (hard for exploration), DSAC has significant advantages against other methods. Although TD4 also achieves sound results in many tasks like Walker2d-v3, DSAC has better performances in challenging environments with larger action spaces, which demonstrates the entropy regularized objective leads to better exploration. It needs to be pointed

[1] https://github.com/xtma/dsac

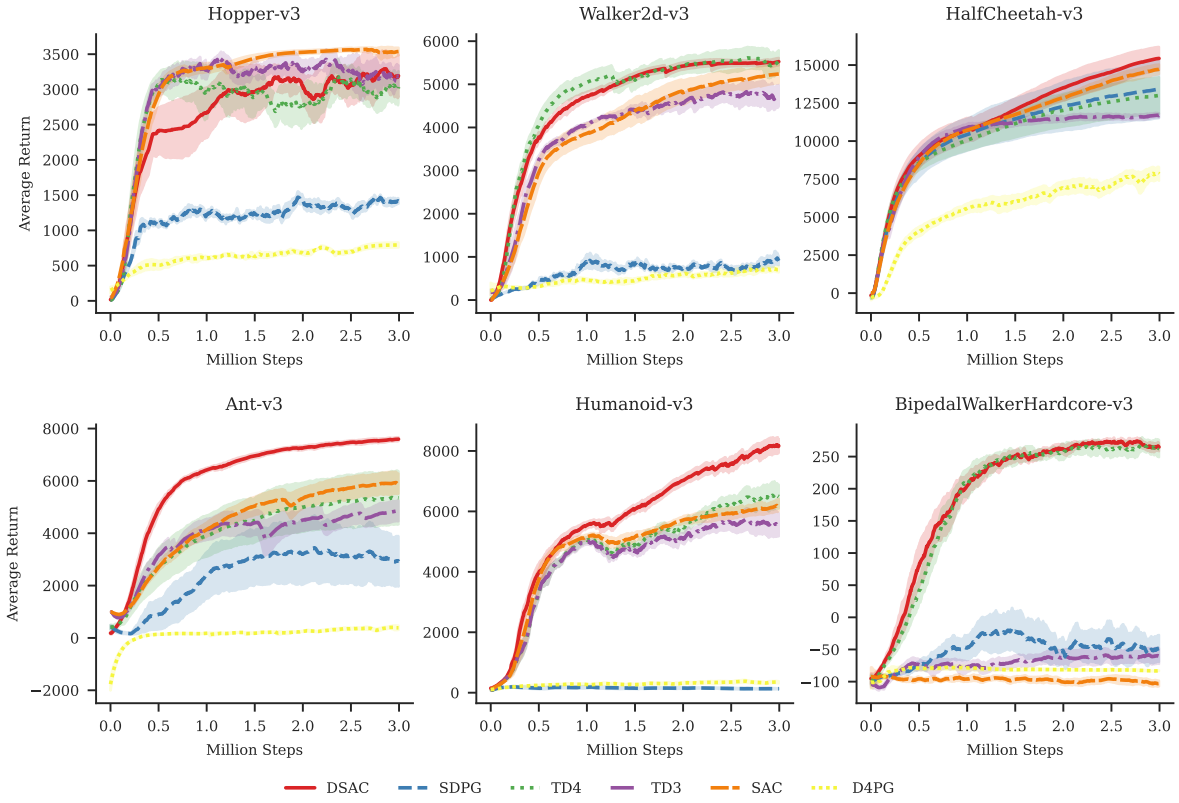


Fig. 4. Learning curves for the continuous control benchmarks in MuJoCo and Box2d. Each learning curve is averaged over 6 different random seeds and shaded by the half of their variance. All curves are smoothed for better visibility.

Table 1. Comparison of average maximal returns ± one standard deviation of 3 million training steps over 6 random seeds. The evaluations are performed every 5000 training steps in each trial over 10 episodes. "BWH" denotes BipedalWalkerHardcore.

Environment	DSAC	TD4	SDPG	SAC	TD3	D4PG
Hopper-v3	**4138 ± 113**	4034 ± 131	3712 ± 234	3634 ± 60	3814 ± 83	1543 ± 451
Walker2d-v3	5624 ± 182	**5797 ± 536**	4459 ± 956	5361 ± 442	5364 ± 565	2016 ± 346
HalfCheetah-v3	**15725 ± 1453**	13396 ± 2510	13865 ± 3731	15148 ± 632	12149 ± 419	9715 ± 763
Ant-v3	**7729 ± 142**	5791 ± 2155	5597 ± 1820	6387 ± 513	5224 ± 811	1732 ± 502
Humanoid-v3	**8673 ± 460**	7256 ± 961	475 ± 73	6547 ± 507	6119 ± 520	641 ± 174
BWH-v3	**318 ± 3**	318 ± 7	118 ± 124	−36 ± 11	−13 ± 16	5 ± 70

out that, unlike former solutions combining evolution strategy, implementing recurrent policy or using specific tricks, DSAC finishes BipealWalkerHardcore-v3 with an ideal score without any targeted changes. The results illustrate the effectiveness of modeling distributional information of both reward and action.

To help understand the effectiveness of distributional RL, we compare the difference of Q-values estimated by DSAC and SAC. As both algorithms utilize double learning for value evaluation, it is a natural way to compare the differences in the output values of the two networks. To be specific, we calculate normalized difference $\Delta Q = 2|Q_1 - Q_2|/|Q_1 + Q_2|$ over the last million steps. The results in Figure 5 show that DSAC has lower difference between the two network outputs. Reasonably, DSAC trains two Z networks, then takes the smaller of the two and then takes the expected value to get Q values:

$$Q_{DSAC}(s,a) = \mathbb{E}[\min(Z_\tau(s,a;\theta_1), Z_\tau(s,a;\theta_2))].$$

SAC, on the other hand, picks the min of the Q values:

$$Q_{SAC}(s,a) = \min(Q(s,a;\theta_1), Q(s,a;\theta_2))$$

where $Q(\cdot;\theta) = \mathbb{E}[Z(\cdot;\theta)]$. As we see $Q_{DSAC}(s,a) \leqslant Q_{SAC}(s,a)$, DSAC algorithm produces smaller estimates, thereby more accurate value networks.

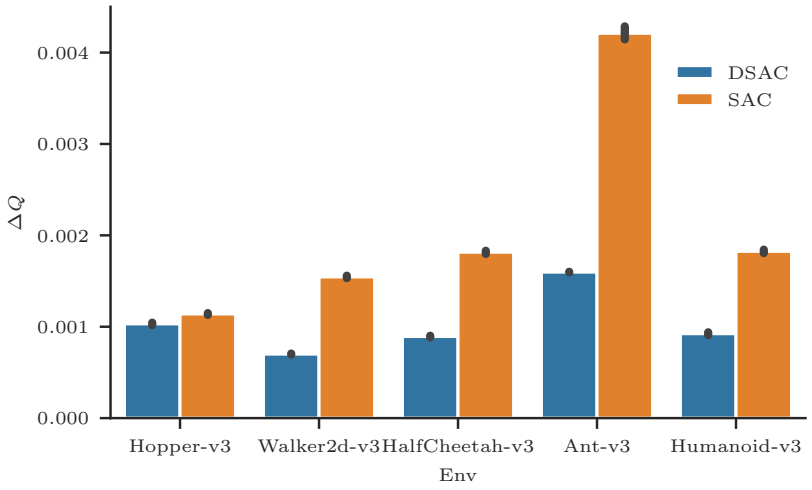


Fig. 5. Compare ΔQ of DSAC and SAC over the last million training steps in MuJoCo tasks.

5.3 Risk-sensitive Policies

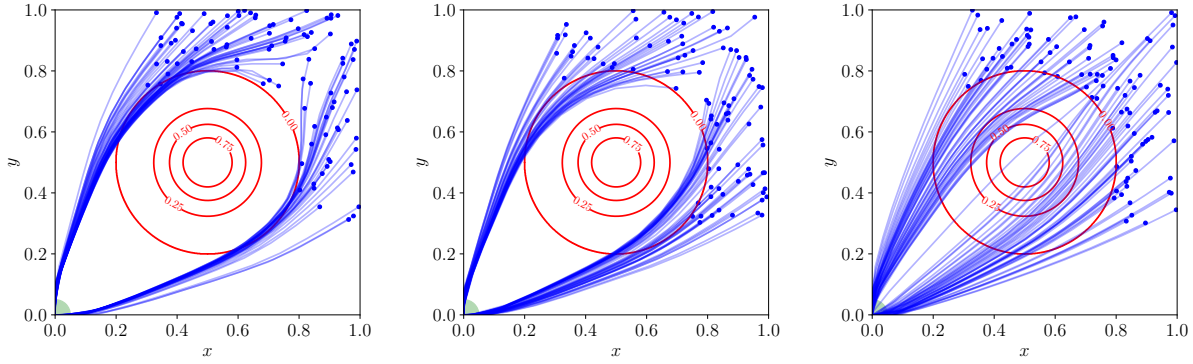
The standard MuJoCo environments are deterministic, which limits their applicability in scenarios involving risk. To assess the impact of risk-sensitive measures on agent performance, we utilize a set of tasks termed *risky robot navigation* [31].

Risky Mass Point. This task involves a mass point that must navigate to a target goal while avoiding a danger zone. Penalties are incurred upon entry, with the likelihood and severity increasing as the mass point approaches the center of the zone. In our experiments, the danger zone is a circular area with a radius of $d_0 = 0.3$, centered at $(0.5, 0.5)$. The risk of penalty is modeled by the function $p = p_0 e^{-4(d/d_0)^2}$, where $p_0 = 0.1$ represents the baseline risk and d is the distance to the zone's center. The penalty for entering the danger zone is set at 10. The mass point is considered to have reached the goal if it is within a circle centered at $(0, 0)$ with a radius of 0.05. The reward function is designed to encourage proximity to the goal and swift task completion, inversely related to

the distance to the goal and reduced by a constant factor. Initial states are randomly selected from $U(0.3, 1)$ for both x and y coordinates, excluding the danger zone.

Table 2. Comparison of the final performance of 200 thousand training steps over 5 random seeds in Risk Mass Point. The evaluation of each trial is over 100 episodes.

Metric	Average	CVaR(0.25)	CVaR(0.1)	Min
Neutral	−7.20 ± 0.14	−10.53 ± 0.44	−13.61 ± 0.91	−19.25 ± 0.59
CVaR(0.25)	−7.13 ± 0.08	**−9.59 ± 0.21**	−11.04 ± 0.41	−16.34 ± 0.49
CVaR(0.1)	−7.35 ± 0.09	−9.76 ± 0.15	**−10.80 ± 0.20**	**−14.17 ± 0.80**
Wang(0.75)	**−7.12 ± 0.09**	−9.67 ± 0.27	−11.28 ± 0.51	−16.95 ± 1.24
CPW(0.71)	−7.16 ± 0.11	−10.21 ± 0.30	−12.89 ± 0.61	−19.11 ± 1.30
MSD(1)	−7.20 ± 0.10	−9.66 ± 0.20	−11.01 ± 0.34	−15.33 ± 0.98



(a) CVaR(0.1) (b) CVaR(0.25) (c) Neutral

Fig. 6. Visualization of sampled trajectories in Risk Mass Point. For each risk measure, we choose its best seed in terms of average return. The results of other risk measures can be found in Figure 10 in Appendix.

Risky Ant. To evaluate the algorithm's capability in complex navigation tasks requiring control, we introduce an Ant agent in a larger environment. The Ant replaces the mass point, and the task retains the core elements of the Risky Mass Point. The target area is a circular region with a radius of 0.5 on the outer edge of a 10×10 playground. The danger zone, centered at $(5, 5)$, has a radius of $d_0 = 3$. The risk of penalty and its calculation remain consistent with the previous task, but the penalty is increased to 200. The reward function now includes a velocity bonus to incentivize rapid task completion, shifting from a constant factor. Initial states are sampled from $U(0, 7)$.

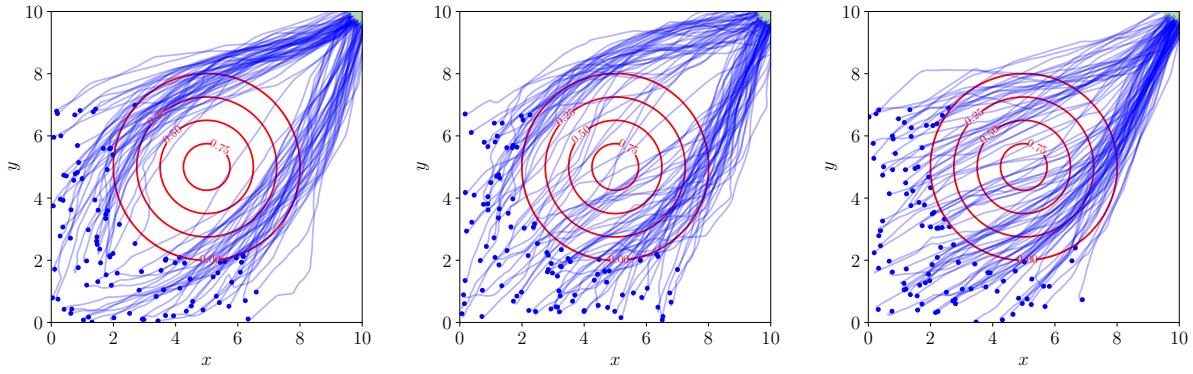
For each task, we execute the RDSAC algorithm with various risk measures, selecting one or two representative parameters for each. We assess the learned policies over 100 episodes, calculating the average return, CVaR(0.25), CVaR(0.1), and minimum values to compare their final performances. The results are detailed in Table 2 and Table 3.

In the Risky Mass Point experiments, all risk measure policies successfully navigate the task with comparable average returns. Notably, the Wang(0.75) measure demonstrates superior average performance. Both CVaR(0.1) and CVaR(0.25) excel under their respective metrics, with CVaR(0.1) also achieving the highest minimum value.

This is expected, as these measures align with the task's objectives. To further elucidate their behavior, we present selected trajectories in Figure 6.

Table 3. Comparison of the final performance of 3 million training steps over 5 random seeds in Risky Ant. The evaluation of each trial is over 100 episodes.

Metric	Average	CVaR(0.25)	CVaR(0.1)	Min
Neutral	−441.52 ± 33.79	−671.20 ± 75.88	−812.97 ± 111.65	−1409.28 ± 363.10
CVaR(0.25)	−427.09 ± 8.62	−646.40 ± 16.44	−775.53 ± 24.83	−1122.70 ± 89.03
CVaR(0.1)	−920.45 ± 375.10	−1370.94 ± 589.30	−1622.94 ± 673.30	−2087.99 ± 644.82
Wang(0.75)	**−383.50 ± 10.52**	**−554.48 ± 23.01**	**−650.78 ± 35.91**	**−946.51 ± 157.67**
CPW(0.71)	−423.44 ± 19.23	−644.12 ± 27.95	−778.95 ± 27.19	−1244.84 ± 99.60
MSD(1)	−396.79 ± 10.38	−590.98 ± 19.72	−717.52 ± 30.66	−1276.15 ± 33.27



(a) CVaR(0.1) (b) CVaR(0.25) (c) Neutral

Fig. 7. Visualization of sampled trajectories in Risk Ant. For each risk measure, we choose its best seed in terms of average return. The results of other risk measures can be found in Figure 11 in Appendix.

Risky Mass Point is a task with simple dynamics, while Risky Ant is more complicated as the agent needs to learn control policy and navigation together. As a result, the overly conservative measure like CVaR(0.1) fails in finishing the navigation task within the training time. A major reason is that a risk-averse objective hinders the exploration since CVaR(0.1) puts too little weight on the part of distribution with larger rewards. In comparison, CVaR(0.25) is less conservative than CVaR(0.1) and accomplished the task while maintaining good risk aversion in training. The sampled trajectories of Risky Ant are shown in Figure 7.

Wang(0.75) balances the return and risk more softly and achieves the best performance in all the metrics. It implies that a softer risk measure might be preferable for complex control tasks to ensure sufficient exploration in training. Meanwhile, MSD(1) also performs well in terms of both return and risk metrics and may be worth trying in other tasks. CPW(0.71) is better than Neutral, but worse than all other measures. Therefore, based on this task, Wang and MSD are recommended as risk measures.

6 Conclusions, Discussion and Future Works

Our proposed algorithm, DSAC (Distributional Soft Actor-Critic), merges MaxEnt RL and distributional RL, considering the randomness present in actions and discounted returns. Under this integration, we introduce a Bellman operator that retains the contractive properties characteristic of MaxEnt RL and at the same time incorporates the flexibility of distributional RL. DSAC surpasses existing baselines across a few continuous control benchmarks. Furthermore, we extended DSAC into a framework for handling diverse risk measures. Our framework Risk-Sensitive Distributional Soft Actor-Critic (RDSAC) balances risk and reward through multiple risk measures in two risk-averse tasks.

One future work to further show the effectiveness of our unified framework is to include a comparison study with prior works that can also handle multiple risk measures [15, 55]. A natural extension to our work is to replace the quantile regression loss in DSAC with a classification-based objective using cross-entropy over discrete bins, which has been shown to yield more expressive representations of value function and provide robustness against noisy target values. [18]. Another promising direction is to extend DSAC and RDSAC to model uncertainty over reward functions, allowing sample-based inference in tasks with implicit or human-derived objectives [28]. Advancing DRL will require richer analysis of the learned distribution, including their representational capacity, distributional metrics, its interaction with value estimation, and the role of parameterization choices, such as categorical, quantile-based, or mixture models [6, 12, 47, 58].

In conclusion, distribution contains the full information of a random variable. Considering the distribution of discounted returns in RL can help treat various optimization objectives, such as expected objective or risk objectives. How to develop efficient optimization algorithms for distributional RL is a promising research direction. The DSAC and RDSAC algorithms in this paper offer a robust foundation for future research and practical applications, particularly in domains where the trade-off between risk and reward is critical.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (2022YFA1004600), the National Natural Science Foundation of China (72342006, 72371253, 72461160315), the Guangdong Basic and Applied Basic Research Foundation (2023A1515012492, 2023B1515040001), the Regional Joint Foundation of Guangdong (2022A1515110725), and the Guangdong Province Key Laboratory of Computational Science at the Sun Yat-sen University. The work described in this paper was also partially supported by InnoHK initiative, the Government of the HKSAR, and Laboratory for AI-Powered Financial Technologies. This work was also generously supported by an NSF grant ECCS 2419564.

References

[1] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. 2019. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*. PMLR, 151–160.

[2] Alejandro Balbás, José Garrido, and Silvia Mayoral. 2009. Properties of distortion risk measures. *Methodology and Computing in Applied Probability* 11, 3 (2009), 385.

[3] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. 2018. Distributed distributional deterministic policy gradients. *ArXiv preprint* abs/1804.08617 (2018).

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

[5] Marc G Bellemare, Nicolas Le Roux, Pablo Samuel Castro, and Subhodeep Moitra. 2019. Distributional reinforcement learning with linear function approximation. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 2203–2211.

[6] Marc G. Bellemare, Nicolas Le Roux, Pablo Samuel Castro, and Subhodeep Moitra. 2019. Distributional reinforcement learning with linear function approximation. arXiv:1902.03149 [cs.LG] https://arxiv.org/abs/1902.03149

[7] Dimitri P. Bertsekas. 1995. *Dynamic Programming and Optimal Control* (1st ed.). Athena Scientific.

[8] Dimitri P. Bertsekas and John N. Tsitsiklis. 1995. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, Vol. 1. IEEE, 560–564.

[9] Cristian Bodnar, Adrian Li, Karol Hausman, Peter Pastor, and Mrinal Kalakrishnan. 2019. Quantile QT-Opt for Risk-Aware Vision-Based Robotic Grasping. *ArXiv preprint* abs/1910.02787 (2019).

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2018. Openai gym. *ArXiv preprint* abs/1606.01540 (2018).

[11] Shicong Cen, Chen Cheng, Yuxin Chen, Yuting Wei, and Yuejie Chi. 2022. Fast global convergence of natural policy gradient methods with entropy regularization. *Operations Research* 70, 4 (2022), 2563–2578.

[12] Yunho Choi, Kyungjae Lee, and Songhwai Oh. 2019. Distributional Deep Reinforcement Learning with a Mixture of Gaussians. In *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 9791–9797.

[13] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. 2015. Risk-sensitive and robust decision-making: a CVaR optimization approach. In *Advances in Neural Information Processing Systems*. 1522–1530.

[14] Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. 2020. A distributional code for value in dopamine-based reinforcement learning. *Nature* 577, 7792 (2020), 671–675.

[15] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. 2018. Implicit Quantile Networks for Distributional Reinforcement Learning. In *International Conference on Machine Learning*. 1104–1113.

[16] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[17] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. 2021. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE transactions on neural networks and learning systems* 33, 11 (2021), 6584–6598.

[18] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. 2024. Stop Regressing: Training Value Functions via Classification for Scalable Deep RL. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 13049–13071. https://proceedings.mlr.press/v235/farebrother24a.html

[19] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*. 1582–1591.

[20] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. 2019. A Theory of Regularized Markov Decision Processes. In *International Conference on Machine Learning*. 2160–2169.

[21] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3389–3396.

[22] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*. 1856–1865.

[23] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *ArXiv preprint* abs/1812.05905 (2018).

[24] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*. 2944–2952.

[25] Peter J. Huber. 1964. Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics* 35, 1 (1964), 492–518.

[26] Geoffrey E. Hinton Jimmy Lei Ba, Jamie Ryan Kiros. 2016. Layer Normalization. *arXiv preprint* abs/1607.06450 (2016).

[27] Sergey Levine. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *ArXiv preprint* abs/1805.00909 (2018).

[28] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025. Inference-Time Scaling for Generalist Reward Modeling. arXiv:2504.02495 [cs.CL] https://arxiv.org/abs/2504.02495

[29] Yudong Luo, Guiliang Liu, Haonan Duan, Oliver Schulte, and Pascal Poupart. 2022. Distributional Reinforcement Learning with Monotonic Splines. In *International Conference on Learning Representations*.

[30] Xiaoteng Ma, Shuai Ma, Li Xia, and Qianchuan Zhao. 2022. Mean-Semivariance Policy Optimization via Risk-Averse Reinforcement Learning. *J. Artif. Intell. Res.* 75 (2022), 569–595.

[31] Yecheng Jason Ma, Dinesh Jayaraman, and Osbert Bastani. 2021. Conservative offline distributional reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 1471, 13 pages.

[32] Harry M Markowitz. 1959. *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons, New York.

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

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *ArXiv preprint* abs/1312.5602 (2013).

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

[36] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. 2010. Nonparametric return distribution approximation for reinforcement learning. In *International Conference on Machine Learning*. 799–806.

[37] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. 2017. Bridging the gap between value and policy based reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 2772–2782.

[38] Włodzimierz Ogryczak and Andrzej Ruszczyński. 1999. From Stochastic Dominance to Mean-Risk Models: Semideviations as Risk Measures. *European Journal of Operational Research* 116, 1 (1999), 33–50. https://doi.org/10.1016/S0377-2217(98)00167-2

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 8024–8035.

[40] LA Prashanth, Michael C Fu, et al. 2022. Risk-Sensitive Reinforcement Learning via Policy Gradient Search. *Foundations and Trends® in Machine Learning* 15, 5 (2022), 537–693.

[41] L. A. Prashanth and Mohammad Ghavamzadeh. 2016. Variance-constrained actor-critic algorithms for discounted and average reward MDPs. *Machine Learning* 105, 3 (2016), 367–417.

[42] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[43] Chao Qu, Shie Mannor, and Huan Xu. 2019. Nonlinear distributional gradient temporal-difference learning. In *International Conference on Machine Learning*. 5251–5260.

[44] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. 2013. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[45] R. Tyrrell Rockafellar and Stanislav Uryasev. 2000. Optimization of Conditional Value-at-Risk. *The Journal of Risk* 2, 3 (2000), 21–41. https://doi.org/10.21314/JOR.2000.038

[46] Mark Rowland, Marc Bellemare, Will Dabney, Remi Munos, and Yee Whye Teh. 2018. An Analysis of Categorical Distributional Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*. 29–37.

[47] Mark Rowland, Marc G. Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. 2018. An Analysis of Categorical Distributional Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*. https://api.semanticscholar.org/CorpusID:4861830

[48] Mark Rowland, Robert Dadashi, Saurabh Kumar, Remi Munos, Marc G Bellemare, and Will Dabney. 2019. Statistics and Samples in Distributional Reinforcement Learning. In *International Conference on Machine Learning*. 5528–5536.

[49] John Schulman, Xi Chen, and Pieter Abbeel. 2017. Equivalence between policy gradients and soft q-learning. *ArXiv preprint* abs/1704.06440 (2017).

[50] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

[51] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv preprint* abs/1707.06347 (2017). https://arxiv.org/abs/1707.06347

[52] Yun Shen, Michael J. Tobia, Tobias Sommer, and Klaus Obermayer. 2014. Risk-sensitive reinforcement learning. *Neural Computation* 26, 7 (2014), 1298–1328.

[53] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[54] Rahul Singh, Keuntaek Lee, and Yongxin Chen. 2022. Sample-based distributional policy gradient. In *Learning for Dynamics and Control Conference*. PMLR, 676–688.

[55] Rahul Singh, Qinsheng Zhang, and Yongxin Chen. 2020. Improving robustness via risk averse distributional reinforcement learning. In *Learning for Dynamics and Control*. PMLR, 958–968.

[56] Matthew J. Sobel. 1982. The variance of discounted Markov decision processes. *Journal of Applied Probability* 19, 4 (1982), 794–802.

[57] Adam Stooke and Pieter Abbeel. 2019. rlpyt: A Research Code Base for Deep Reinforcement Learning in PyTorch. *ArXiv preprint* abs/1909.01500 (2019).

[58] Ke Sun, Yingnan Zhao, Yi Liu, Enze Shi, Yafei Wang, Aref Sadeghi, Xiaodong Yan, Bei Jiang, and Linglong Kong. 2022. Interpreting Distributional Reinforcement Learning: Regularization and Optimization Perspectives. https://api.semanticscholar.org/CorpusID:246473235

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

[60] Aviv Tamar, Dotan Di Castro, and Shie Mannor. 2012. Policy gradients with variance related risk criteria. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 1651–1658.

[61] Emanuel Todorov. 2008. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*. IEEE, 4286–4292.
[62] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*.
[63] Amos Tversky and Daniel Kahneman. 1992. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty* 5, 4 (1992), 297–323.
[64] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.
[65] Shaun S. Wang. 2000. A class of distortion operators for pricing financial and insurance risks. *Journal of risk and insurance* 67, 1 (2000), 15–36.
[66] Li Xia. 2016. Optimization of Markov decision processes under the variance criterion. *Automatica* 73 (2016), 269–278.
[67] Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2019. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems*. 6190–6199.
[68] Shangtong Zhang and Hengshuai Yao. 2019. QUOTA: The quantile option architecture for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5797–5804.
[69] Fan Zhou, Jianing Wang, and Xingdong Feng. 2020. Non-crossing quantile regression for deep reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1334, 11 pages.
[70] Fan Zhou, Zhoufan Zhu, Qi Kuang, and Liwen Zhang. 2021. Non-decreasing Quantile Function Network with Efficient Exploration for Distributional Reinforcement Learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 3455–3461. https://doi.org/10.24963/ijcai.2021/476 Main Track.
[71] Zhengyuan Zhou, M Bloem, and N Bambos. 2018. Infinite Time Horizon Maximum Causal Entropy Inverse Reinforcement Learning. *IEEE Trans. Automat. Control* 63, 9 (2018), 2787–2802.
[72] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008.*

A Proofs

Before our proofs, we first review several useful properties of the metric d_p:

$$d_p(aU, aV) \leq |a| d_p(U, V) \tag{P1}$$

$$d_p(A + U, A + V) \leq d_p(U, V) \tag{P2}$$

$$d_p(AU, AV) \leq \|A\|_p d_p(U, V), \tag{P3}$$

where a is a scale and A is a random variable independent of U, V. An additional lemma is required, which is called *Partition lemma* by Bellemare et al. [4]:

$$d_p(U, V) \leq \sum_i d_p(A_i U, A_i V), \tag{P4}$$

where $A_i(\omega) \in \{0, 1\}$ and for any ω there is exactly one A_i with $A_i(\omega) = 1$.

We introduce the following inequality to deal with the relationship between policies and their action-value functions. Consider $\pi_\xi \in \mathbb{R}^{|\mathcal{A}|}$ parameterized by the softmax transform of $\xi \in \mathbb{R}^{|\mathcal{A}|}$ such that

$$\pi_\xi(i) = \frac{\exp \xi(i)}{\sum_j \exp \xi(j)}, 1 \leq i \leq |\mathcal{A}|.$$

For any two vectors π_{ξ_1} and π_{ξ_2}, we have,

$$\| \log \pi_{\xi_1} - \log \pi_{\xi_2} \|_\infty \leq 2 \| \xi_1 - \xi_2 \|_\infty. \tag{S1}$$

See Equation 68 in [11] for the proof.

In addition, we define a new transition operator P_π to simplify the notation in the later proofs:

$$P_\pi Z(s, a) \stackrel{D}{:=} Z(s', a'), s' \sim P(\cdot \mid s, a), a' \sim \pi(\cdot \mid s).$$

Further, we denote $P^* = P_{\pi^*}$.

A.1 Lemma 1

Lemma 1. *$\mathcal{T}_{DS}^{\pi} : \mathcal{Z} \to \mathcal{Z}$ is a γ-contraction in $\bar{d}_p$.*

Proof. Let $Z_1, Z_2 \in \mathcal{Z}$ denote two action-value distributions. For any $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have

$$\begin{aligned}
&d_p\big(\mathcal{T}_{DS}^{\pi} Z_1(s,a), \mathcal{T}_{DS}^{\pi} Z_2(s,a)\big) \\
&\quad = d_p\big(R(s,a) + \gamma P_\pi \left[Z_1(s,a) - \alpha \log \pi(a' \mid s')\right], R(s,a) + \gamma P_\pi \left[Z_2(s,a) - \alpha \log \pi(a' \mid s')\right]\big) \\
&\quad \overset{(i)}{\leq} d_p\left(\gamma P_\pi Z_1(s',a'), \gamma P_\pi Z_2(s',a')\right) \\
&\quad \overset{(ii)}{\leq} \gamma d_p\left(P_\pi Z_1(s',a'), P_\pi Z_2(s',a')\right) \\
&\quad \leq \gamma \sup_{s',a'} d_p\left(Z_1\left(s',a'\right), Z_2\left(s',a'\right)\right)
\end{aligned}$$

where (i) follows by P2, (ii) follows by P1. By definition of $\bar{d}_p$, we have

$$\begin{aligned}
\bar{d}_p\big(\mathcal{T}_{DS}^{\pi} Z_1, \mathcal{T}_{DS}^{\pi} Z_2\big) &= \sup_{s,a} d_p\big(\mathcal{T}_{DS}^{\pi} Z_1(s,a), \mathcal{T}_{DS}^{\pi} Z_2(s,a)\big) \\
&\leq \gamma \sup_{s',a'} d_p\left(Z_1\left(s',a'\right), Z_2\left(s',a'\right)\right) \\
&= \gamma \bar{d}_p\left(Z_1, Z_2\right).
\end{aligned}$$

□

A.2 Lemma 2

Lemma 2 (Distributional Soft Policy Evaluation). *Let $Z_{k+1} := \mathcal{T}_{DS}^{\pi} Z_k$ with $Z_0 \in \mathcal{Z}$. The sequence Z_k will converge to Z^π as $k \to \infty$.*

Proof. As we have proved that $\mathcal{T}_{DS}^{\pi}$ is a γ-contraction, policy evaluation can be obtained by repeatedly applying $\mathcal{T}_{DS}^{\pi}$. □

A.3 Lemma 3

Lemma 3 (Distributional Soft Policy Improvement). *Let $Q^\pi(s,a) := \mathbb{E}[Z^\pi(s,a)]$, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$. With $\pi_{\text{old}} \in \Pi$ and π_{new} as the solution of problem defined in Equation 5, we have $Q^{\pi_{\text{old}}}(s,a) \leq Q^{\pi_{\text{new}}}(s,a)$.*

Proof. The proof of this lemma has a similar idea to that of the soft policy improvement in [22].

For any policy $\pi \in \Pi$ and its soft action-value distribution Z^π, we define the soft action-value by taking the expectation,

$$Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)] = \mathbb{E}[R(s,a)] + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')}\left[Z^\pi(s',a') - \alpha \log \pi(a' \mid s')\right].$$

Let denote a policy $\pi_{\text{old}} \in \Pi$ and its soft action-value $Q^{\pi_{\text{old}}}$. We can obtain a new policy $\pi_{\text{new}} \in \Pi$ by solving the minimization problem defined in Equation 5,

$$\pi_{\text{new}}(\cdot \mid s) = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\cdot \mid s) \,\|\, \exp\left(Q^{\pi_{\text{old}}}(s,\cdot)/\alpha - \log \Delta^{\pi_{\text{old}}}(s)\right)\right).$$

Since π_{new} is the solution to the minimization problem above, it must be

$$\begin{aligned}
&\mathbb{E}_{a \sim \pi_{\text{new}}(\cdot|s)}\left[\log \pi_{\text{new}}(a \mid s) - Q^{\pi_{\text{old}}}(s,a)/\alpha + \log \Delta^{\pi_{\text{old}}}(s)\right] \\
&\qquad \leq \mathbb{E}_{a \sim \pi_{\text{old}}(\cdot|s)}\left[\log \pi_{\text{old}}(a \mid s) - Q^{\pi_{\text{old}}}(s,a)/\alpha + \log \Delta^{\pi_{\text{old}}}(s)\right],
\end{aligned}$$

which can be reduced as follows for partition function $\log \Delta^{\pi_{\text{old}}}$ is only related to s,

$$\mathbb{E}_{a\sim\pi_{\text{new}}(\cdot|s)}\left[\alpha \log \pi_{\text{new}}\left(a \mid s\right) - Q^{\pi_{\text{old}}}\left(s, a\right)\right] \leq \mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s)}\left[\alpha \log \pi_{\text{old}}\left(a \mid s\right) - Q^{\pi_{\text{old}}}\left(s, a\right)\right].$$

Thus, we expand the soft Bellman equation and have

$$\begin{aligned}
Q^{\pi_{\text{old}}}\left(s_t, a_t\right) &= \mathbb{E}[R(s_t, a_t)] + \gamma \mathbb{E}_{\substack{s_{t+1}\sim P(\cdot|s_t,a_t)\\ a_{t+1}\sim\pi_{\text{old}}(\cdot|s_{t+1})}}\left[Q^{\pi_{\text{old}}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_{\text{old}}\left(a_{t+1} \mid s_{t+1}\right)\right] \\
&\leq \mathbb{E}[R(s_t, a_t)] + \gamma \mathbb{E}_{\substack{s_{t+1}\sim P(\cdot|s_t,a_t)\\ a_{t+1}\sim\pi_{\text{new}}(\cdot|s_{t+1})}}\left[Q^{\pi_{\text{old}}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_{\text{new}}\left(a_{t+1} \mid s_{t+1}\right)\right] \\
&\vdots \\
&\leq Q^{\pi_{\text{new}}}\left(s_t, a_t\right).
\end{aligned}$$

□

A.4 Theorem 1

Theorem 1 (Convergence in the control setting). *Let $Z_{k+1} := \mathcal{T}^*_{DS} Z_k$ with $Z_0 \in \mathcal{Z}$. The sequence Z_k will converge to Z^* as $k \to \infty$.*

To prove this lemma, we first prove some auxiliary lemmas as follows.

Lemma 4. *Let $Z : \mathcal{A} \to \mathcal{Z}$ denote a distribution function that maps actions to return distributions. For two stochastic policy π_1, π_2, we denote $Z_i \overset{D}{:=} Z(a), a \sim \pi_i$. We have $d_p(Z_1, Z_2) \leq D_{\text{TV}}(\pi_1 \| \pi_2) B$, where D_{TV} is the total variation divergence and $B := \sup_{Z\in\mathcal{Z}} \|Z\|_\infty < \infty$.*

Proof. First, we denote a new random variable $W := \mathbb{1}_{a_1=a_2}, a_i \sim \pi_i(\cdot)$, while similarly, $\bar{W} := \mathbb{1}_{a_1\neq a_2}$. We have

$$\begin{aligned}
d_p(Z_1, Z_2) &= d_p((W + \bar{W})Z_1, (W + \bar{W})Z_2) \\
&\overset{(i)}{\leq} d_p(WZ_1, WZ_2) + d_p(\bar{W}Z_1, \bar{W}Z_2) \\
&\overset{(ii)}{\leq} d_p(\bar{W}Z_1, \bar{W}Z_2) \\
&\overset{(iii)}{\leq} \Pr(a_1 \neq a_2) d_p(\bar{W}Z_1, \bar{W}Z_2) \\
&\overset{(iv)}{\leq} D_{\text{TV}}(\pi_1 \| \pi_2) B,
\end{aligned}$$

where (i) follows by P4, (ii) follows by the fact $d_p(Z(a), Z(a)) = 0$, (iii) follows by P3, and (iv) follows by the definitions of $D_{\text{TV}}(\pi_1 \| \pi_2)$ and B. □

Lemma 5. *Let π_k, Q_k denote the policy and the action-value function in k-th iteration. Similarly, let denote π^*, Q^* in the optimal case. We have*

$$D_{\text{KL}}(\pi_k(\cdot \mid s) \| \pi^*(\cdot \mid s)) \leq 2 \left\|Q_k(s, a) - Q^*(s, a)\right\|_\infty / \alpha.$$

Proof. By the definition of D_{KL},

$$\begin{aligned}
D_{\text{KL}}(\pi_k(\cdot \mid s) \| \pi^*(\cdot \mid s)) &= \sum_a \pi_k(a|s) \left[\log \pi_k(a|s) - \log \pi^*(a|s)\right] \\
&\leq \left\|\pi_k(a|s)\right\|_1 \left\|\log \pi_k(a|s) - \log \pi^*(a|s)\right\|_\infty \\
&\leq 2 \left\|Q_k(s, a) - Q^*(s, a)\right\|_\infty / \alpha,
\end{aligned}$$

where the last inequality follows by S1.

Proof of Theorem 1.

$$
\begin{aligned}
&d_p\left(Z_{k+1}(s,a), Z^*(s,a)\right) = d_p\left(\mathcal{T}^*_{DS} Z_k(s,a), \mathcal{T}^*_{DS} Z^*(s,a)\right) \\
&\quad = d_p\left(R(s,a) + \gamma P_{\pi_k}[Z_k(s',a') - \alpha \log \pi_k(a' \mid s')], R(s,a) + \gamma P^*[Z^*(s',a') - \alpha \log \pi^*(a' \mid s')]\right) \\
&\quad \overset{(i)}{\leq} \gamma d_p\left(P_{\pi_k}[Z_k(s',a') - \alpha \log \pi_k(a' \mid s')], P^*[Z^*(s',a') - \alpha \log \pi^*(a' \mid s')]\right) \\
&\quad \overset{(ii)}{\leq} \underbrace{\gamma\, d_p\left(P_{\pi_k}[Z_k(s',a') - \alpha \log \pi_k(a' \mid s')], P_{\pi_k}[Z^*(s',a') - \alpha \log \pi_k(a' \mid s')]\right)}_{E_1} + \\
&\qquad \underbrace{\gamma\, d_p\left(P_{\pi_k}[Z^*(s',a') - \alpha \log \pi_k(a' \mid s')], P_{\pi_k}[Z^*(s',a') - \alpha \log \pi^*(a' \mid s')]\right)}_{E_2} + \\
&\qquad \underbrace{\gamma\, d_p\left(P_{\pi_k}[Z^*(s',a') - \alpha \log \pi^*(a' \mid s')], P^*[Z^*(s',a') - \alpha \log \pi^*(a' \mid s')]\right)}_{E_3},
\end{aligned}
$$

where (i) follows P1 and P2, and (ii) follows the triangle inequality as d_p is a metric.

Similar with the result of Lemma 1, we have

$$
\begin{aligned}
E_1 &\leq \bar{d}_p(Z_k, Z^*). \\
E_2 &\overset{(i)}{\leq} \alpha d_p(\log \pi_k(a' \mid s'), \log \pi^*(a' \mid s')) \\
&\overset{(ii)}{\leq} \alpha \| \log \pi_k(a' \mid s') - \log \pi^*(a' \mid s') \|_\infty \\
&\overset{(iii)}{\leq} 2 \| Q_k(s',a') - Q^*(s',a') \|_\infty,
\end{aligned}
$$

where (i) follows by P1 and P2, and (iii) follows by S1.

$$
\begin{aligned}
E_3 &\overset{(i)}{\leq} D_{\mathrm{TV}}(\pi_k(\cdot \mid s) \| \pi^*(\cdot \mid s)) B \\
&\overset{(ii)}{\leq} B\sqrt{\| Q_k(s,a) - Q^*(s,a) \|_\infty / \alpha},
\end{aligned}
$$

where (i) follow our lemma above, and (ii) follows the fact that $D_{\mathrm{TV}}(p\|q) \leq \sqrt{D_{\mathrm{KL}}(p\|q)/2}$.

Combine all the inequalities above, we have

$$
d_p\left(Z_{k+1}(s,a), Z^*(s,a)\right) \leq \gamma \left[\bar{d}_p(Z_k, Z^*) + 2C_k + B\sqrt{C_k/\alpha}\right],
$$

where $C_k = \max_{s',a'} \|Q_k(s',a') - Q^*(s',a')\|_\infty$. Since soft Bellman operator $\mathcal{T}^*_S$ is a γ-contraction, we have $C_k \leq \gamma^k C_0$. Thus, we infer that

$$
\begin{aligned}
\bar{d}_p\left(Z_{k+i+1}, Z^*\right) &\leq \gamma^i \bar{d}_p(Z_k, Z^*) + \sum_{j=0}^{i} \gamma^j \left(2C_{k+j} + B\sqrt{C_{k+j}/\alpha}\right) \\
&\leq \gamma^i B + \frac{1}{1-\gamma}\left(2C_k + B\sqrt{C_k/\alpha}\right).
\end{aligned}
$$

For any $\epsilon > 0$, we can take k and i large enough to make $\bar{d}_p\left(Z_{k+i+1}, Z^*\right) < \epsilon$. We thus have completed this proof. □

Proposition 1. *$\mathcal{T}^*_{DS}$ is not a contraction.*

Proof. Next, we consider the similar example in [4] and analyze the effect of adding entropy into the distributional optimality operator.

There are two states s_1, s_2 and a unique transition from s_1 to s_2, where s_2 has two actions leads to different return distributions. The distributions are shown in Table 4. In the table, we use a tuple $(Pr(z), z)$ to express the probability of an atom and its value, and denote $\diamond = 1 + e^{\frac{\epsilon}{\alpha}}$ in shorthand.

By the definition of $\bar{d}_1$, we have

$$\bar{d}_1(Z, Z^*) = d_1(Z(s_2, a_2), Z^*(s_2, a_2)) = 2\epsilon,$$
$$\bar{d}_1(\mathcal{T}^*_{\mathcal{DS}} Z, Z^*) = d_1(Z(s_1), Z^*(s_1)) = \epsilon + (1-\epsilon)\frac{e^{\frac{\epsilon}{\alpha}} - 1}{1 + e^{\frac{\epsilon}{\alpha}}}.$$

The size relationship between $\bar{d}_1(Z, Z^*)$ and $\bar{d}_1(\mathcal{T}^*_{\mathcal{DS}} Z, Z^*)$ depends on both ϵ and α. When $\epsilon = 0.1$ and $\alpha = 1$, $\bar{d}_1(Z, Z^*) < \bar{d}_1(\mathcal{T}^*_{\mathcal{DS}} Z, Z^*)$. However, $\bar{d}_1(Z, Z^*) > \bar{d}_1(\mathcal{T}^*_{\mathcal{DS}} Z, Z^*)$ if $\epsilon = 0.1$ and $\alpha = 1$, which reveals $\mathcal{T}^*_{DS}$ is a non-expansion. The result shows that although there is a positive impact on convergence of the distributional optimality operator, introducing entropy in policy objective can not make it a contraction.

Table 4. A Counterexample of Contraction

-	s_1	s_2, a_1	s_2, a_2
Z^*	$\left(\frac{e^{\frac{\epsilon}{\alpha}}}{2\diamond}, -1 + \alpha\log(\diamond)\right), \left(\frac{1}{\diamond}, \alpha\log(\diamond)\right), \left(\frac{e^{\frac{\epsilon}{\alpha}}}{2\diamond}, 1 + \alpha\log(\diamond)\right)$	(1,0)	$(\frac{1}{2}, \epsilon - 1), (\frac{1}{2}, \epsilon + 1))$
Z	$\left(\frac{e^{\frac{\epsilon}{\alpha}}}{2\diamond}, -1 + \alpha\log(\diamond)\right), \left(\frac{1}{\diamond}, \alpha\log(\diamond)\right), \left(\frac{e^{\frac{\epsilon}{\alpha}}}{2\diamond}, 1 + \alpha\log(\diamond)\right)$	(1,0)	$(\frac{1}{2}, -\epsilon - 1), (\frac{1}{2}, -\epsilon + 1)$
$\mathcal{T}^*_{\mathcal{DS}} Z$	$\left(\frac{1}{2\diamond}, -\epsilon - 1 + \alpha\log(\diamond)\right), \left(\frac{e^{\frac{\epsilon}{\alpha}}}{\diamond}, -\epsilon + \alpha\log(\diamond)\right), \left(\frac{1}{2\diamond}, -\epsilon + 1 + \alpha\log(\diamond)\right)$	(1,0)	$(\frac{1}{2}, \epsilon - 1), (\frac{1}{2}, \epsilon + 1))$

□

B Implementation Details

B.1 Quantile fraction Generation

We consider quantile regression for return distribution approximation in DSAC. Though we do not constraint which method to employ in quantile fraction generation, a modular experiment is provided. Out of six quantile function generation methods introduced in section 1 introduction, we adapt three earlier methods, QR-DQN, IQN and FQF, into DSAC and evaluate them on the MuJoCO and Box2d environments. We show the results in Figure 8. Based on the experiment results, we choose random for quantile fraction generation, as it has better performance than fix and fewer parameters than net. The authors acknowledge our limitation that the later improved methods NC-QR-DQN, NDQFN and SPL-DQN were not included in this ablation studies.

Quantile Regression DQN (QR-DQN) Dabney et al. [16] first introduced quantile regression to improve the distribution approximation. By replacing the fix value atoms in C51 [4], the value distribution is approximated by a group of trainable quantile values at fixed quantile fractions. With the basic idea of QR-DQN, quantile fractions are given by a group of fix values as $\tau_i = i/N, i = 0, \ldots, N$.

Implicit Quantile Network (IQN) Dabney et al. [15] extended the fixed quantile fractions to uniform samples and proposed implicit quantile value network (IQN). With infinite sampled quantile fractions, IQN is able to approximate the full quantile function. IQN uniformly samples the quantile fractions and takes the expectation

in training. As we fix the number of quantile fractions and keep them in ascending order, we adapt the sampling as $\tau_0 = 0, \tau_i = \epsilon_i / \sum_{i=0}^{N-1} \epsilon_i$ where $\epsilon_i \sim U[0,1], i = 1, \ldots, N$.

Fully parameterized Quantile Function (FQF) Instead of sampling the quantile fractions uniformly, FQF parameterizes both the quantile values and quantile fractions. In addition to the quantile value network, FQF uses an additional fraction proposal network $q(s, a; \omega)$, which has a softmax output layer and cumulatively sums the outputs as quantile fractions. The gradient of ω is given by the gradient of W_1 with respect to τ, $\frac{\partial W_1}{\partial \tau_i} = 2Z_{\tau_i}(s, a; \theta) - Z_{\hat{\tau}_i}(s, a; \theta) - Z_{\hat{\tau}_{i-1}}(s, a; \theta)$, where W_1 denotes 1-Wasserstein error between the approximated and true quantile functions. The quantile proposal network in FQF is a two-layer 128-unit fully connected network with learning-rate set as 1e-5.

B.2 Hyper-Parameter Setting

Hyper-parameters used in experiments and environment specific parameters are listed in Table 6.

Instead of using fixed the temperature parameter α to balance reward and entropy, SAC has a variant [23] which introduces a mechanism of fine-tuning α to achieve target entropy adaptively. While our algorithm does not conflict with this parameter adaptive method, we use fixed α suggested in original SAC paper in our experiments to reduce irrelevant factors. For all experiments, we clip the rewards within $[-10, 10]$ to reduce the variation of value outputs, which is essential for BipedalWalkerHardcore-v3.

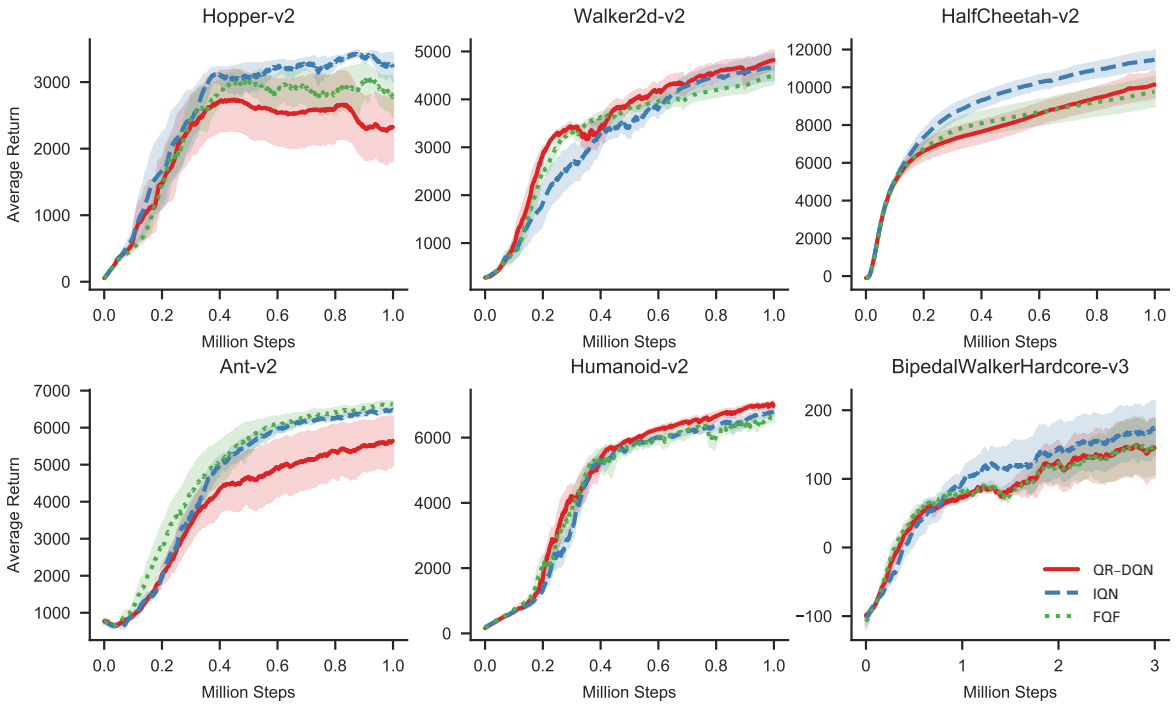


Fig. 8. Comparison of quantile fraction generation methods in MuJoCo and Box2d. Each learning curve is averaged over 5 different random seeds and shaded by the half of their variance. All curves are smoothed for better visibility.

Table 5. Hyper-parameters - Environments

Environment	Temperature Parameter	Max episode length
Hopper-v2	0.2	1000
Walker2d-v2	0.2	1000
HalfCheetah-v2	0.2	1000
Ant-v2	0.2	1000
Humanoid-v2	0.05	1000
BipedalWalkerHardcore-v3	0.01	2000
Risky Mass Point	0.2	100
Risky Ant	0.2	200

Table 6. Hyper-parameters - Algorithms

Hyper-parameter	Value
Shared among all algorithms	
Policy network learning rate	3e-4
(Quantile) Value network learning rate	3e-4
Optimizer	Adam
Discount factor	0.99
Target smoothing	5e-3
Batch size	256
Replay buffer size	1e6
Minimum steps before training	1e4
Shared among TD3 & TD4	
Target policy noise	0.2
Clip target policy noise	0.5
Policy update period	2
Shared among DSAC & TD4 & SDPG	
Number of quantile fractions	64
Quantile fraction embedding size	128
Huber regression threshold	1
Sample size	100

B.3 Network Structure

For SAC and TD3, we utilize two-layer fully connected networks with 256 hidden units for both value and policy network and ReLU as activation function. DSAC has the same structure of policy network $\pi_\phi(a \mid s; \phi)$.

In order to embed quantile fraction τ into the quantile value network, we borrow the idea of implicit representations from IQN [15]. With infinite sampled quantile fractions, IQN is able to approximate the full quantile function. IQN uniformly samples the quantile fractions and takes the expectation in training. As we fix the number of quantile fractions and keep them in ascending order, we adapt the sampling as $\tau_0 = 0, \tau_i = \epsilon_i / \sum_{i=0}^{N-1} \epsilon_i$ where $\epsilon_i \sim U[0,1], i = 1, \ldots, N$. With the element-wise (Hadamard) product (denoted as $\odot$) of state feature $\psi(s,a)$ and embedding $\varphi(\tau)$, the quantile values are given by $Z_\tau(s,a) = Z(\psi(s,a) \odot (1 + \varphi(\tau)); \theta)$. After studying

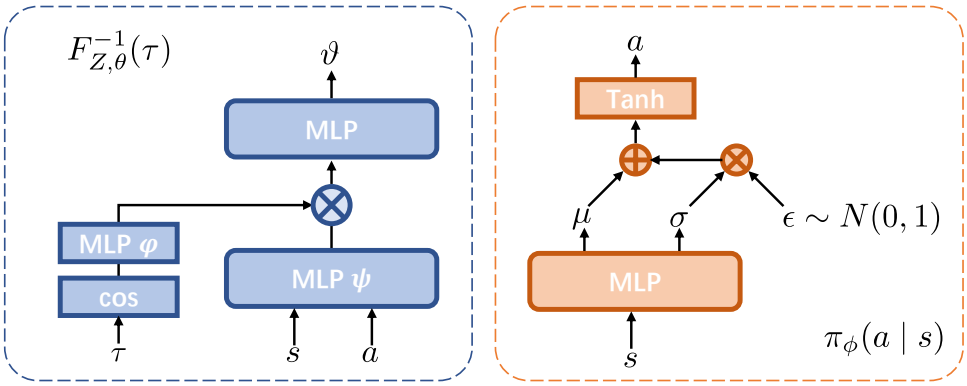


Fig. 9. Network structures of DSAC.

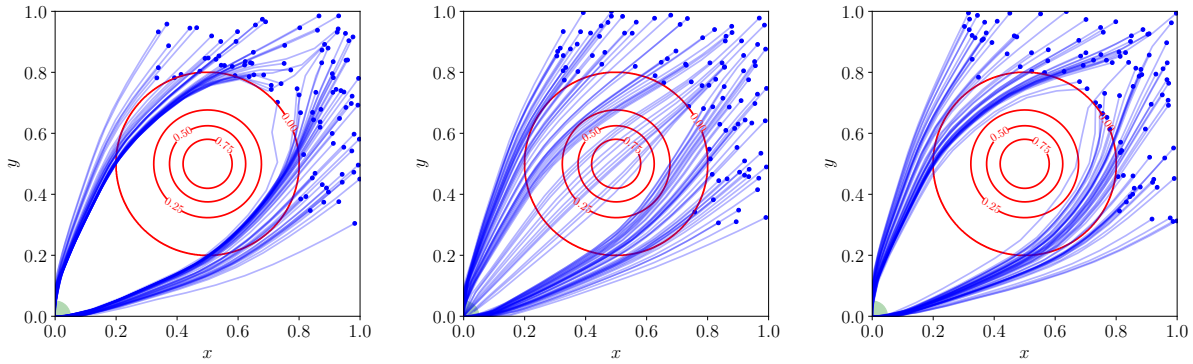
different ways for embedding τ, IQN suggests an embedding formula with $\cos(\cdot)$,

$$\varphi_j(\tau) := \mathrm{ReLU}\left(\sum_{i=0}^{n-1} \cos(i\pi\tau) w_{ij} + b_j\right), \tag{18}$$

where n is the embedding dimension and w_{ij}, b_j are network parameters.

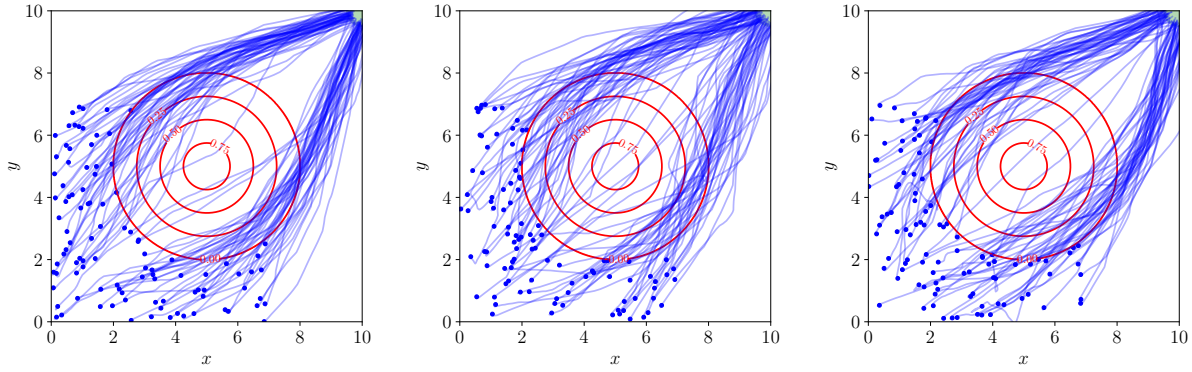
We use one-layer 256-unit fully connected network for $\psi(s,a)$ and one-layer 64-unit fully connected network for $\varphi(\tau)$ in Equation 18, followed with one-layer 256-unit fully connected network for their multiplied values $\psi(s,a) \odot (1+\varphi(\tau))$. Moreover, since magnitudes of quantile values vary hugely, we implement layer normalization [26] to the hidden activation of networks.

B.4 Additional Results



(a) Wang(0.75) (b) CPW(0.71) (c) MSD(1)

Fig. 10. Visualization of sampled trajectories in Risk Mass Point. For each risk measure, we choose its best seed in terms of average return.



(a) Wang(0.75) (b) CPW(0.71) (c) MSD(1)

Fig. 11. Visualization of sampled trajectories in Risk Ant. For each risk measure, we choose its best seed in terms of average return.

Received 1 November 2022; revised 3 November 2024; accepted 28 April 2025