Monotonic Quantile Network for Worst-Case Offline Reinforcement Learning

Chenjia Bai, Ting Xiao, Zhoufan Zhu, Lingxiao Wang, Fan Zhou, Animesh Garg, Bin He, *Member, IEEE*, Peng Liu, and Zhaoran Wang

***Abstract*— A key challenge in offline reinforcement learning (RL) is how to ensure the learned offline policy is *safe*, especially in safety-critical domains. In this article, we focus on learning a distributional value function in offline RL and optimizing a worst-case criterion of returns. However, optimizing a distributional value function in offline RL can be hard, since the crossing quantile issue is serious, and the distribution shift problem needs to be addressed. To this end, we propose monotonic quantile network (MQN) with conservative quantile regression (CQR) for risk-averse policy learning. First, we propose an MQN to learn the distribution over returns with non-crossing guarantees of the quantiles. Then, we perform CQR by penalizing the quantile estimation for out-of-distribution (OOD) actions to address the distribution shift in offline RL. Finally, we learn a worst-case policy by optimizing the conditional value-at-risk (CVaR) of the distributional value function. Furthermore, we provide theoretical analysis of the fixed-point convergence in our method. We conduct experiments in both risk-neutral and risk-sensitive offline settings, and the results show that our method obtains safe and conservative behaviors in robotic locomotion tasks.**

***Index Terms*— Monotonic quantile network (MQN), offline reinforcement learning (RL), quantile regression, risk-sensitive learning.**

I. Introduction

DEEP reinforcement learning (DRL) [1] has achieved remarkable success in a variety of tasks, including video games [2], [3], robotic manipulation [4], and so on [5], [6], [7]. In most applications, DRL needs a large amount of interacting data generated from simulators [8] to train the high-capacity neural networks. Due to the requirement of large samples, it remains challenging for applying DRL in real-world applications, such as bin packing [9], healthcare [10], and industrial robotics [11]. In the real world, it is usually costly and risky to obtain interacting experiences without a simulator. A possible solution is developing DRL algorithms that can learn policies from the previously collected dataset, which is typically generated by one or more *behavior policies*. Recently, *offline RL* [12] becomes a promising paradigm in learning policies from static *offline datasets* without interacting with the environment. However, directly applying off-policy algorithms to offline RL often has large extrapolation errors and causes overestimation for out-of-distribution (OOD) actions [13]. The reason is the behavior policy of the offline dataset, and the policy learned by RL algorithms often has large *distribution shift*. Current offline RL methods aim to solve this problem through policy constraints [14], [15], [16], conservative value functions [17], [18], [19], and uncertainty-based pessimism [20], [21], [22]. These methods focus on various penalizations to regularize the behavior of OOD actions. However, most methods consider a risk-neutral learning setting [23] and do not address the risk-sensitive problem in offline RL.

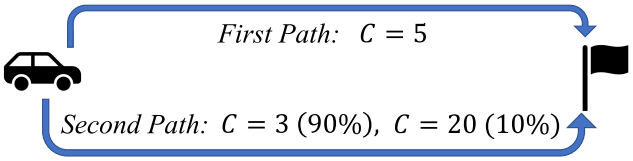

Fig. 1. Illustration example. The agent has two paths toward the goal. The first path takes $C = 5$ min. The second path takes $C = 3$ min with a probability of 90% and takes $C = 20$ min with a probability of 10% (in case of traffic jam). We want to consume less time and consider the *expected return* as $R = -C$. Then, the expected returns of two paths are $R_1 = -5$ and $R_2 = -3 \times 0.9 - 20 \times 0.1 = -4.7$, respectively.

One key challenge for offline RL is how to ensure the agent makes *safe* decisions, especially in safety-critical applications, such as autonomous driving [24]. Considering a simple driving problem shown in Fig. 1, the agent has two paths, and the corresponding expected returns are -5 and -4.7, respectively. The standard RL algorithms only consider the expectation of return and choose actions to maximize it. In this example, the agent will choose the second path, since the corresponding expected return is higher. However, choosing the second path

Manuscript received 7 December 2021; revised 31 May 2022 and 26 August 2022; accepted 12 October 2022. This work was supported by the Shanghai AI Laboratory, Shanghai, China. *(Corresponding author: Ting Xiao.)*

Chenjia Bai is with the Shanghai AI Laboratory, Shanghai 200232, China (e-mail: baichenjia@pjlab.org.cn).

Ting Xiao is with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China (e-mail: xiaoting@ecust.edu.cn).

Zhoufan Zhu and Fan Zhou are with the School of Statistics and Management, Shanghai University of Finance and Economics, Shanghai 200433, China (e-mail: tylerzzf@163.sufe.edu.cn; zhoufan@mail.shufe.edu.cn).

Lingxiao Wang and Zhaoran Wang are with the Department of Industrial Engineering and the Department of Management Sciences, Northwestern University, Evanston, IL 60208 USA (e-mail: lingxiaowang2022@u.northwestern.edu; zhaoranwang@gmail.com).

Animesh Garg is with the Vector Institute, University of Toronto, Toronto, ON M5S 0A5, Canada (e-mail: garg@cs.toronto.edu).

Bin He is with the School of Electronic and Information Engineering, Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University, Shanghai 201210, China (e-mail: hebin@tongji.edu.cn).

Peng Liu is with the Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China (e-mail: pengliu@hit.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3217189.

Digital Object Identifier 10.1109/TNNLS.2022.3217189

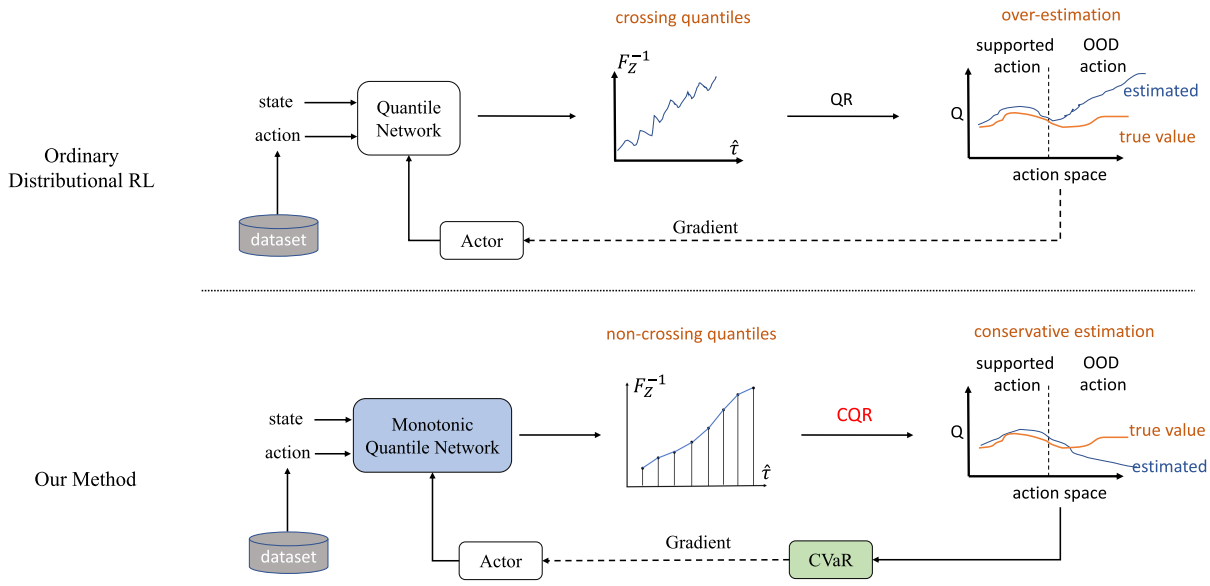

Fig. 2. Illustration of the ordinary distributional RL algorithm and the proposed method. The differences between these two methods include the following: 1) the ordinary distributional RL has a crossing quantile problem, while our method generates monotonic quantiles via an MQN; 2) the ordinary distributional RL has overestimation for OOD actions in an offline setting, while our method enforces pessimism for OOD actions via CQR; and 3) in a risk-sensitive setting, we perform risk-averse policy training by optimizing the CVaR objective of the value distribution.

is *risky* and may take a very long-time waiting (i.e., $C = 20$) with a small probability. Thus, maximizing the expected return does not prevent catastrophic events, and it is better to model the whole *return distribution* that contains more information about the inherent randomness of reward and transition function [25]. With a learned return distribution, we can obtain risk-sensitive policies by optimizing the varying level of risks.

Previous methods often study the risk-sensitive learning in an online RL setting, and the agent is trained to avoid visiting risky states and actions when interacting with the environment [26], [27]. Nevertheless, in offline RL, learning a safe policy is more challenging, since the offline dataset can be generated by some risky policy, and the agent is forced to learn a *safe* policy from the given *unsafe* experiences. The traditional control theories also study the risk-sensitive learning problem, while they often consider a linear stochastic control system and exponential quadratic cost [28], which is different from offline RL setting considered in this study. Other safety-critical learning algorithm also has assumptions of kernel-based transition dynamics [29], physical constraints [30], or studies in hybrid fuzzy systems [31].

In this article, we aim to answer the following questions.

How to perform risk-sensitive policy learning for offline RL in large-scale tasks? To answer this question, we have the following new developments. First, we need a stable distributional value function to measure risks. To this end, we propose a monotonic quantile network (MQN) to learn an incremental structure to enforce a monotonic constraint over different quantile fractions, which guarantees the validity of non-crossing quantile distribution [32] and is important with limited training samples in offline RL. MQN uses continuous quantiles, while previous non-crossing methods only support fixed quantiles. Theoretically, we show MQN with quantile regression converges to a fixed point. We highlight that MQN ensures that the quantile function increases monotonically, which is essential for our theoretical analysis. Second, to address the distribution shift problem in offline RL, we extend the conservative Q learning (CQL) [17] to distributional offline RL setting and perform conservative quantile regression (CQR) to penalize the quantile function for OOD actions, which is essential for offline distributional RL. We prove that MQN with CQR-update converges to a lower bound on the quantile value of the return distribution. Third, with the MQN critic and CQR update, we train a risk-averse policy by optimizing the worst case of the distributional value function. Fig. 2 compares the difference between our method and the ordinary distribution RL algorithm.

We theoretically show that our method obtains a lower bound of the true quantiles and converges to a fixed point. We demonstrate the effectiveness of our method in various robotic locomotion tasks. We train risk-neutral [23] and risk-averse policies in standard and risk-sensitive datasets for deep data-driven RL (D4RL) tasks [33]. The results show that our method provides safe and conservative behaviors and performs better than state-of-the-art methods.

II. Background

A. Offline RL

We consider an episodic Markov decision process (MDP) represented by $(\mathcal{S}, \mathcal{A}, T, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition function, T is the episode length, $|r| \leq R_{\max}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. In online RL, the agent interacts with the environment by taking actions through the policy $\pi(a|s)$. The return of (s_t, a_t) is defined as $R_t = \sum_{i=t}^{T-1} \gamma^{i-t} r_i$. The Q function represents the expected return as $Q^\pi(s_t, a_t) := \mathbb{E}_\pi[R_t]$. In offline RL, the policy is learned from an offline dataset $\mathcal{D} := \{s, a, r, s'\}$ without any interactions

with the environment. The Q function is estimated by sampling experience from $\mathcal{D}$. However, $\mathcal{D}$ only contains limited transitions, which cannot cover the entire state-action space of the true environment. As a result, the policy evaluation in offline RL becomes an empirical estimation of the transition function, as follows:

$$\widehat{\mathcal{T}}^{\pi}\hat{Q}(s,a) := \hat{r}(s,a) + \gamma\, \mathbb{E}_{s'\approx\widehat{P}(\cdot|s,a),a'\approx\pi\left(a'\middle|s'\right)}\left[\hat{Q}\left(s',a'\right)\right] \quad (1)$$

where $\widehat{P}$ is the empirical estimation of P through $\mathcal{D}$, and $\widehat{\mathcal{T}}^{\pi}$ is the empirical Bellman operator. In policy evaluation, we have $\hat{Q}^{k+1} = \widehat{\mathcal{T}}^{\pi}\hat{Q}^{k}$.

In offline RL, the policy evaluation in (1) is unstable because of *distribution shift*, which is caused by the different marginal state distributions of the learned policy π and the *behavior policy* π_β, where π_β indicates the policy that generates the offline dataset $\mathcal{D}$. In offline RL, since the learned policy π is different from the behavior policy π_β, the greedy action a' selected by $\pi(\cdot|s')$ in (1) becomes an OOD action. The policy evaluation for OOD actions usually causes erroneously high Q values [15].

CQL [17] is a principled method to solve the distribution shift problem by penalizing the value function of OOD actions, as follows:

$$\hat{Q}^{k+1} = \arg\min_{Q}\alpha\, \mathbb{E}_{s\approx\mathcal{D},a\approx\pi}[Q(s,a)] + \mathcal{L}\left(Q, \widehat{\mathcal{T}}^{\pi}\hat{Q}^{k}\right) \quad (2)$$

where the first term is the value function of OOD actions, and the second term is the temporal difference (TD) error. In CQL, the OOD actions are sampled from the learned policy π, since π can be very different from the behavior policy π_β. Our method follows the principle of conservative estimation in a risk-sensitive setting. The proposed CQR algorithm is an extension of CQL with distributional value function. Penalizing the value function of OOD actions regularizes the extrapolation behavior of quantiles and reduces the overestimation in policy evaluation. Then, we perform risk-averse policy training based on conservative quantile estimation to control risks.

B. Quantile Regression

In order to perform risk-sensitive learning, it is necessary to model the return distribution Z^{π} instead of Q^{π}, as illustrated in Fig. 1. Then, Q^{π} becomes the expectation of the return distribution as $Q^{\pi} = \mathbb{E}[Z^{\pi}]$. RL algorithms that aim to learn the return distribution Z^{π} are called *distributional RL* [25], which has been demonstrated to have improved performance in complex tasks.

In policy evaluation, distributional RL uses *Wasserstein distance* [34] to measure the distance between return distributions. Z^{k+1} is obtained by iteratively minimizing $Z^{k+1} = \arg\min_Z W_p(Z, \mathcal{T}^{\pi}Z^{k})$, where W_p is the p-Wasserstein distance. In the following, we denote F_Z as the cumulative density function (cdf) of the distribution Z [i.e., $F_Z(z) = \Pr(Z \le z)$] and F_Z^{-1} as the inverse cdf (i.e., quantile function) [35] of the distribution Z. According to the definition of p-Wasserstein distance, we have the following:

$$W_p\left(Z, \mathcal{T}^{\pi}Z^{k}\right) = \left(\int_0^1 \left|F_Z^{-1}(w) - F_{\mathcal{T}^{\pi}Z^{k}}^{-1}(w)\right|^p dw\right)^{1/p}. \quad (3)$$

The p-Wasserstein metric measures the optimal transport between distributions and has been widely used in machine learning.

Quantile regression deep Q-network (QR-DQN) [36] estimates the quantile function of the value distribution through *quantile regression* [37] and then minimizes the Wasserstein distance to the distributional Bellman target as in (3). In QR-DQN, the return distribution is parameterized by Z_θ, which maps a state-action pair to a uniform mixture supported on $\{\theta_i(s,a)\}_{i=1}^{N}$, as follows:

$$Z_\theta(s,a) = \sum_{i=0}^{N-1}(\tau_{i+1} - \tau_i)\delta_{\theta_i}(s,a) \quad (4)$$

where $\tau_i = i/N$, and δ_{θ_i} denotes a Dirac at θ_i. In QR-DQN, δ_{θ_i} is assigned a fixed quantile target $F_Z^{-1}(\hat{\tau}_i)$, where $\hat{\tau}_i = (\tau_{i+1} + \tau_i)/2$. The p-Wasserstein metric can be calculated by the pairwise distance between Z and $\mathcal{T}^{\pi}Z^{k}$.

Implicit quantile network (IQN) [26] extends the discrete quantile fractions to a continuous function as $\tau \approx U([0,1])$, where U is the uniform distribution. Then, quantile function $Z_\tau := F_Z^{-1}(\tau)$ maps quantile fraction τ to the quantile value $Z_\tau(s,a)$, where $Z_\tau(s,a)$ can be considered as a sample from the implicitly defined return distribution $Z(s,a)$. In this study, we follow IQN and use uniformly sampled quantile fractions. Meanwhile, our method can also use fixed fractions [36] or trainable proposal fractions [38]. Due to the distribution shift problem in offline RL, the distributional value function cannot be directly trained by offline datasets that have limited state-action coverage. As a result, we address this problem by penalizing the quantile values through CQR, which enforces conservatism for OOD actions.

C. Risk-Sensitive Learning

From the perspective of distributional RL, the standard RL algorithms only utilize the mean of the return distribution $\mathbb{E}[Z(s,a)]$ to learn a policy. Thus, a natural extension is to learn a *risk-sensitive* policy that utilizes the distribution information provided by $Z(s,a)$. *Risks* refer to the *aleatoric uncertainty* over possible future outcomes. The risks are measured by return distribution in this study.

In distributional RL with continuous quantiles, the policy is trained to maximize some utility function $\eta : [0,1] \to [0,1]$, as follows:

$$\pi(s) = \arg\max_{a}\mathbb{E}_{\tau\approx U([0,1])}\left[Z_{\eta(\tau)}(s,a)\right]. \quad (5)$$

If η is an identity function, the policy is *risk neutral*, since it maximizes the expectation of return distribution as in standard RL. Also, we can modify η mapping to be a distortion risk measure to learn a risk-sensitive policy (either *risk seeking* or *risk averse*). Existing distortion functions include cumulative probability weighting (CPW) [39], Wang [40], conditional value-at-risk (CVaR) [41], and so on.

In this study, we focus CVaR measure, since it can be easily implemented and works efficient empirically. For a random variable X, the VaR at level $\eta \in (0,1)$ is defined as follows:

$$\mathrm{VaR}_\eta(X) := \inf\{\Pr(X \le \xi) \ge \beta\}. \quad (6)$$

Since F is the cdf of X, VaR can also be defined as follows:

$$\mathrm{VaR}_\eta(X) := \inf\{\xi | F(\xi) \geq \eta\} = F^{-1}(\eta) \tag{7}$$

which is equal to the definition of quantile function. VaR is not a coherent risk measure. CVaR is an extension of VaR and is defined as the conditional mean over the tail distribution in VaR. Formally

$$\mathrm{CVaR}_\eta(X) := \mathbb{E}\big[X \big| X \leq \mathrm{VaR}_\eta(X)\big] \tag{8}$$

which is used to measure the bottom η percentile in distributional offline RL. CVaR has also been widely used in the financial industry to control risks.

III. Proposed Method

Our algorithm aims to solve the following optimization problem:

$$\pi(s) = \arg\max_a \mathbb{E}_{(s,a,r,s')\ approx\mathcal{D}}\big[\mathrm{CVaR}_\eta(Z(s,a))\big] \tag{9}$$

where the experiences (s, a, r, s') are sampled from a fixed dataset $\mathcal{D}$, CVaR is the risk measure, and Z is the value distribution.

To solve this problem, the following hold: 1) we need a stable distributional value function without crossing quantile, and we propose MQN to address this issue; 2) we need to solve the distribution shift problem in offline RL, and we propose CQR to penalize the quantiles of OOD actions; 3) we need an end-to-end training method to optimize the risk-averse policy; and 4) furthermore, we provide theoretical analysis of the fixed point convergence of our method.

In the following, we denote the quantile fractions as $\tau_1, \tau_2, \ldots, \tau_N$, which indicates the cumulative probabilities associated with such a distribution (that is, the discrete values taken on by the cdf). We will also write $\tau_0 = 0$ to simplify notation.

A. Monotonic Quantile Network

For distributional RL with neural network approximation, the parameterized quantile projection θ_i for $F_Z^{-1}(\hat{\tau}_i)$ is estimated by minimizing the p-Wasserstein metric. With $\tau_i \leq \tau_{i+1}$, we have the following:

$$W_p(Z, Z_\theta) = \left(\sum_{i=0}^{N-1} \int_{\tau_i}^{\tau_{i+1}} \left|F_Z^{-1}(w) - \theta_i\right|^p dw\right)^{1/p}. \tag{10}$$

It follows that:

$$\theta_i = Z_{\hat{\tau}_i} \approx F_Z^{-1}(\hat{\tau}_i) \quad \text{where} \quad \hat{\tau}_i = \frac{\tau_{i+1} + \tau_i}{2} \tag{11}$$

is a unique minimizer for $W_p(Z, Z_\theta)$ if F_Z^{-1} is a continuous function [36, Lemma 2]. Then, we have $Z_{\hat{\tau}_i} \leq Z_{\hat{\tau}_{i+1}}$ holds for all $\hat{\tau}_i$, since both the cdf $F_Z(\cdot)$ and inverse cdf $F_Z^{-1}(\cdot)$ are defined as monotonically increasing functions.

However, since we use a deep neural network to approximate $Z_{\hat{\tau}}$, the constraint of $Z_{\hat{\tau}_i} \leq Z_{\hat{\tau}_{i+1}}$ is not satisfied naturally, especially in offline RL where the training data are limited, and environment has high-dimensional states and actions. If the estimation of $Z_{\hat{\tau}_i} > Z_{\hat{\tau}_{i+1}}$ with $\hat{\tau}_i < \hat{\tau}_{i+1}$, the estimated quantile distribution is not monotonically increasing, and the quantile values become *crossing* among adjacent fractions. This issue is named *crossing quantile* in statistics [42]. In Fig. 3, we give an example of the crossing issue in offline RL training. We train a distributional RL agent in *Hopper* task with the offline dataset from D4RL [23]. We use the ordinary QR-DQN network without any monotonicity constraints and record the quantile values in the training process. In Fig. 3 (left), we visualize a typical quantile distribution for a specific state-action pair. We find the crossing quantile issue is serious, and 37.5% of quantiles are crossing. In Fig. 3 (right), we calculate the ratio of crossing quantiles in training. The crossing issue exists throughout the training process and is especially serious in the early stage of training.

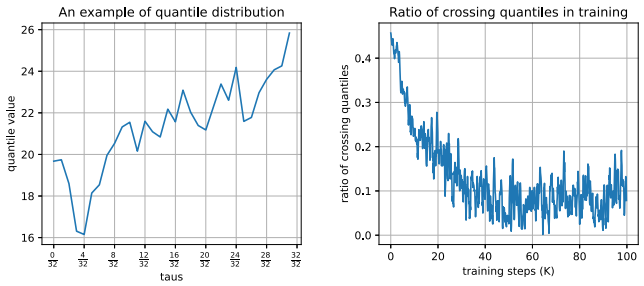

Fig. 3. Illustration of the crossing quantile issue in *Hopper task* in offline RL. We train a QR-DQN agent without monotonicity constraint. Left: we show the quantile function $F_Z^{-1}(\tau)$ of a specific (s, a) pair. We use $N = 32$ quantile fractions as $\tau \in \{(0/N), \ldots, (N-1/N)\}$ and calculate the corresponding $F_Z^{-1}(\tau)$. We find the quantile function is not monotonic, and about 37.5% of quantiles are crossing. Right: we record the ratio of crossing quantiles in the training process of QR-DQN. We find the crossing-quantile issue is extremely serious in the early stage of training, and nearly 50% of quantiles are crossing. The problem is alleviated with training, while it still exists at the end of training.

Since each quantile value is learned individually in training [see (10)] and the neural network approximation has a large solution space, if we do not add global constrain of monotonicity of quantiles in learning, the search space of the solution will be further enlarged. Meanwhile, since the estimation of θ_i usually has a large variance in approaching the true value, it becomes harder to obtain an optimal minimizer for the unconstrained p-Wasserstein metric with limited experiences. In offline RL, we find the crossing quantile issue hinders the performance and increases the instability in training.

To address this problem, we propose to add global monotonic constraints for quantile functions. In MQN, we do not predict $F_Z^{-1}(\hat{\tau}_i)$ directly. Instead, we predict the *quantile gap* between two adjacent quantile fractions. We denote G_i as the gap between quantile fractions $\hat{\tau}_i$ and $\hat{\tau}_{i+1}$, as follows:

$$G_Z(\hat{\tau}_i) := F_Z^{-1}(\hat{\tau}_{i+1}) - F_Z^{-1}(\hat{\tau}_i). \tag{12}$$

For given quantile fractions as $\boldsymbol{\tau} = \{\tau_0, \ldots, \tau_i, \ldots, \tau_{N-1}\}$, we first calculate the midpoint quantile fractions as $\hat{\boldsymbol{\tau}} = \{\hat{\tau}_0, \ldots, \hat{\tau}_i, \ldots, \hat{\tau}_{N-1}\}$ by following (11). For each $\hat{\tau}_i$, $F_Z^{-1}(\hat{\tau}_i)$ is the minimizer of quantile projection defined in (10). Then, for a specific state-action pair (s, a) and give quantile $\hat{\tau}_j \in \hat{\boldsymbol{\tau}}$, we have

$$F_Z^{-1}(\hat{\tau}_j) = F_Z^{-1}(\hat{\tau}_0) + \sum_{i=0}^{j-1} G_Z(\hat{\tau}_i) \tag{13}$$

by following the definition of G_i in (12). Our method ensures the monotonic increasing of the quantile function $F_Z^{-1}(\cdot)$, since

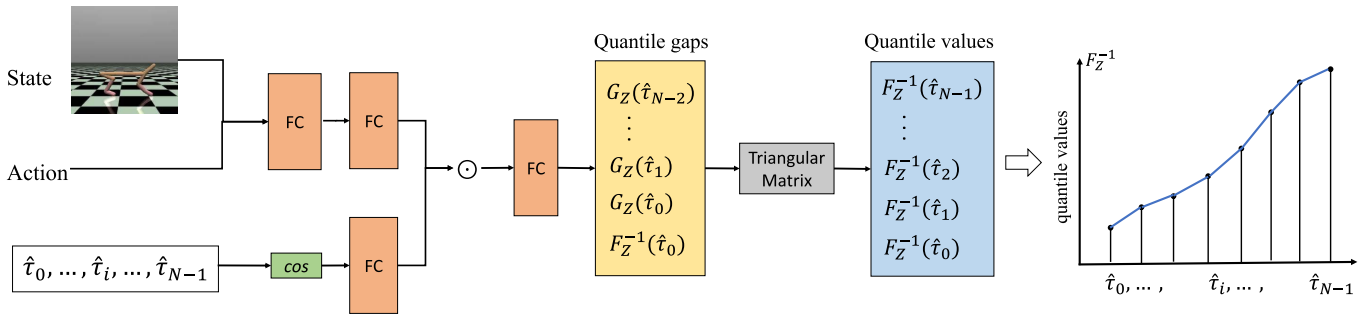

Fig. 4. Architecture of MQN. MQN consists of several FC layers, a cosine embedding layer, an element-wise product, and an upper triangular matrix. We use ReLU to ensure the non-negativity of the predicted $F_Z^{-1}(\hat{\tau}_0)$ and $G_Z(\hat{\tau}_i)$. MQN uses the cosine function to extract the feature of $\hat{\tau}_i$ and then predicts quantile gaps between adjacent quantile fractions based on the state, action, and quantiles. A triangular matrix is used to obtain monotonic quantile estimations based on quantile gaps. MQN finally generates non-crossing quantile estimations for given a $(s, a, \hat{\boldsymbol{\tau}})$. The whole network can be trained in an end-to-end manner through CQR.

we can use activation functions, such as rectified linear unit (ReLU), to make both $F^{-1}(\hat{\tau}_0)$ and $G_Z(\hat{\tau}_i)$ are non-negative. Then, it follows that $F_Z^{-1}(\hat{\tau}_{i+1}) \geq F_Z^{-1}(\hat{\tau}_i)$ always holds, since $\hat{\tau}_{i+1} > \hat{\tau}_i$, and the gap $G_Z(\hat{\tau}_i)$ is non-negative. The architecture of the proposed MQN is given in Fig. 4.

In MQN, we parameterize the quantile network as a function of $(s, a, \hat{\boldsymbol{\tau}})$. For each state-action pair, we concatenate them and use several fully connected (FC) layers to obtain the high-level features as $\psi(s, a)$. For each $\hat{\tau}_i$, we use a cosine function to map $\hat{\tau}_i$ to an embedding vector $\phi(\hat{\tau}_i) \in \mathbb{R}^{n_\tau}$, as follows:

$$\phi(\hat{\tau}_i) = \cos(\pi \cdot j \cdot \hat{\tau}_i), \quad j = 0, 1, \ldots, n_\tau - 1 \tag{14}$$

where we use $n_\tau = 64$ by following IQN [26]. Then, an element-wise product $\odot$ is used to merge the embeddings of state-action pair and quantile fractions, which forces the interactions between the $\psi(s, a)$ and $\phi(\hat{\tau}_i)$ features.

Based on the merged features of (s, a) and $\hat{\boldsymbol{\tau}}$, MQN gives N predictions that represent $F_Z^{-1}(\hat{\tau}_0)$ and $G_Z(\hat{\tau}_i), i = 0, \ldots, N-2$. Then, we follow (13) to calculate each quantile value. In MQN, such operation can be implemented through an upper triangular matrix as follows:

$$\begin{bmatrix} 1 & \cdots & 1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 1 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} G_Z(\hat{\tau}_{N-2}) \\ \vdots \\ G_Z(\hat{\tau}_2) \\ G_Z(\hat{\tau}_1) \\ G_Z(\hat{\tau}_0) \\ F_Z^{-1}(\hat{\tau}_0) \end{bmatrix}$$

$$= \begin{bmatrix} F_Z^{-1}(\hat{\tau}_0) + \sum_{i=0}^{N-2} G_Z(\hat{\tau}_i) \\ \vdots \\ F_Z^{-1}(\hat{\tau}_0) + \sum_{i=0}^{1} G_Z(\hat{\tau}_i) \\ F_Z^{-1}(\hat{\tau}_0) + G_Z(\hat{\tau}_0) \\ F_Z^{-1}(\hat{\tau}_0) \end{bmatrix}$$

$$= \left[F_Z^{-1}(\hat{\tau}_{N-1}),\ \ldots,\ F_Z^{-1}(\hat{\tau}_2),\ F_Z^{-1}(\hat{\tau}_1),\ F_Z^{-1}(\hat{\tau}_0)\right]^\top.$$

MQN generates monotonic quantile estimations for each state-action pair, as shown in Fig. 4. As a result, we solve the crossing quantile issue in offline distributional RL. MQN is trained by stochastic gradient descent (SGD) in an end-to-end manner through CQR, which we discussed in the following.

B. Conservative Quantile Regression

In offline RL, directly performing quantile regression in a fixed dataset usually fails because of the distribution shift problem. The policy evaluation in offline RL tends to have significant extrapolation errors for OOD actions.

For typical RL algorithms, CQL [17] solves this problem by penalizing the value function of OOD actions. In our work, we extend this method for distributional Q learning. Specifically, we penalize the quantile value of (s, a_{ood}) when performing quantile regression in offline RL training. In each training step, we randomly sample a quantile fraction $\tau_c \approx U(0, 1)$ and penalize the quantile estimation for (s, a_{ood}). The loss function for CQR becomes

$$L_{\text{CQR}} = \alpha \mathbb{E}_{\tau_c, a_{\text{ood}}}\left[F^{-1}_{Z(s,a_{\text{ood}})}(\tau_c)\right] + \mathcal{L}_{\text{QR}}\left(Z, \widehat{\mathcal{T}}^\pi \hat{Z}^k\right)$$

with

$$s \approx \mathcal{D}, \quad \tau_c \approx U(0, 1), \quad a_{\text{ood}} \approx \pi(\cdot|s) \tag{15}$$

where the first term performs OOD penalization, the second term $\mathcal{L}_{\text{QR}}$ is the ordinary quantile regression loss, and α is a scale factor. In (15), we sample a_{ood} from the learned policy π, since π is different from π_β of $\mathcal{D}$ and concentrates on the behaviors of the training agent.

In the following, we denote the operator that minimizes L_{CQR} in the offline setting as $\widetilde{\mathcal{T}}^\pi \tilde{Z}$. We define $\bar{d}_p$ as the p-Wasserstein distance to measure the maximum gap between two cdfs, as follows:

$$\bar{d}_p(Z_1, Z_2) = \sup_{s,a} W_p(Z_1(s, a), Z_2(s, a)).$$

Then, we have the following theorem to prove that $\widetilde{\mathcal{T}}^\pi \tilde{Z}$ converges to a unique fixed point $\tilde{Z}^\pi(s, a)$.

Lemma 1: For an MDP with finite state and action spaces, $\widetilde{\mathcal{T}}^\pi$ is a γ contraction in $\bar{d}_\infty$ as follows:

$$\bar{d}_\infty\left(\Pi_{W_1} \widetilde{\mathcal{T}}^\pi Z_1, \Pi_{W_1} \widetilde{\mathcal{T}}^\pi Z_2\right) \leq \gamma\, \bar{d}_\infty(Z_1, Z_2) \tag{16}$$

where Z_1, Z_2 are value distributions, and Π_{W_1} is the quantile projection defined in (10).

The proof is given in the Appendix. Then, according to Banach's fixed point theorem, repeatedly applying $\widetilde{\mathcal{T}}\tilde{Z}$ converges to a unique fixed point $\tilde{Z}^\pi$. Meanwhile, since $\bar{d}_p \le \bar{d}_\infty$, the convergence occurs for $p \in [1, \infty]$.

Since we use MQN to learn the quantile function, both the quantile function and the cdf are guaranteed to increase monotonically. Without loss of generality, we assume that $\forall s, a$, $F_{Z^\pi(s,a)}$ is ζ-strongly monotone, i.e., $F'_{Z^\pi(s,a)}(x) \ge \zeta$. Then, the following lemma connects the infinity norm between cdfs and quantile functions.

Lemma 2: Suppose two cdfs satisfies $\|F - G\|_\infty \le \epsilon$, then $\|F^{-1} - G^{-1}\|_\infty \le \epsilon/\zeta$ holds for quantile functions.

The proof is given in the Appendix. We highlight that the proposed MQN is essential for the assumption of ζ-strongly monotone, and Lemma 2 holds. Then, we denote the *conservative* quantile function $F^{-1}_{\tilde{Z}^\pi}(\hat{\tau})$ as the fixed point by iteratively minimizing $\mathcal{L}_{\text{CQL}}$. The following theorem shows $F^{-1}_{\tilde{Z}^\pi}(\hat{\tau})$ lower bounds the *true* quantile distribution $F^{-1}_{Z^\pi}(\hat{\tau})$.

Theorem 1: Assuming the behavior policy $\pi_\beta(a|s) \ge 0$. For $\forall s \approx \mathcal{D}$, a, with probability $\ge 1 - \delta$, the Z-distribution obtained by minimizing (15) satisfies

$$F^{-1}_{\tilde{Z}^\pi}(\hat{\tau}) \le F^{-1}_{Z^\pi}(\hat{\tau}) - C(\alpha, p, \delta, \gamma, \zeta) \tag{17}$$

where $C(\alpha, p, \delta, \gamma, \zeta) \ge 0$ with a sufficiently large α.

The proof is given in the Appendix. Intuitively, since $\tilde{Z}^\pi(s,a)$ has an additional penalty for OOD actions, the quantiles of $\tilde{Z}^\pi(s,a)$ lower bounds the expected quantiles of empirical distribution $\hat{Z}^\pi(s,a)$. Meanwhile, according to the concentration properties of the Bellman operator, we prove that the quantiles of $\tilde{Z}^\pi(s,a)$ lower bounds the quantiles of the true value distribution $Z^\pi(s,a)$ with a sufficiently large α, which provides conservative estimation for OOD actions and prevents overestimation in offline RL.

In practice, we use $p = 2$ in calculating the quantile regression $\mathcal{L}_{\text{QR}}$ by following previous works [26], [36]. We sample two independent sets of quantile fractions (namely, $\hat{\boldsymbol{\tau}}$ and $\hat{\boldsymbol{\tau}}'$) for the current value distribution $Z(s,a)$ and target value distribution $\mathcal{T}^\pi Z(s,a)$, respectively. For two randomly drawn fractions $\hat{\tau}_i \in \hat{\boldsymbol{\tau}}$ and $\hat{\tau}_j \in \hat{\boldsymbol{\tau}}$ and the current policy π, the corresponding pairwise TD error [36] for transition (s, a, r, s') is

$$\delta_{ij} = r + \gamma\, Z_{\hat{\tau}_j}\big(s', \pi(s')\big) - Z_{\hat{\tau}_i}(s,a). \tag{18}$$

In practice, $Z_{\hat{\tau}_j}(s', \pi(s'))$ is represented by an independent target network, which is soft updated toward the main MQN. We minimize the Huber quantile regression [43] as follows:

$$\rho^\kappa_{\hat{\tau}}(\delta_{ij}) = \big|\hat{\tau} - \mathbb{1}\{\delta_{ij} \le 0\}\big| \frac{\mathcal{L}_\kappa(\delta_{ij})}{\kappa}$$

with

$$\mathcal{L}_\kappa(\delta_{ij}) = \begin{cases} \dfrac{1}{2}\delta^2_{ij}, & \text{if } |\delta_{ij}| \le \kappa \\ \kappa\left(|\delta_{ij}| - \dfrac{1}{2}\kappa\right), & \text{otherwise} \end{cases} \tag{19}$$

where we set the threshold $\kappa = 1$. Then, the quantile regression loss is calculated by averaging the pairwise loss as follows:

$$\mathcal{L}_{\text{QR}}\big(Z, \widehat{\mathcal{T}}\hat{Z}^k\big) = \frac{1}{N^2}\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\big(\hat{\tau}_{j+1} - \hat{\tau}_j\big)\rho^\kappa_{\hat{\tau}}(\delta_{ij}) \tag{20}$$

where each ρ^κ_{ij} is weighted by the fraction length $\hat{\tau}_{j+1} - \hat{\tau}_j$. When minimizing $\mathcal{L}_{\text{QR}}$, we sample $s \in \mathcal{D}$ and $a_{\text{ood}} \approx \pi(\cdot|s)$ and randomly sample quantile functions to calculate the penalty term $F^{-1}_Z(s, a_{\text{ood}})$. We minimize our overall loss function $\mathcal{L}_{\text{CQR}}$ by combining the penalty term and $\mathcal{L}_{\text{QR}}$.

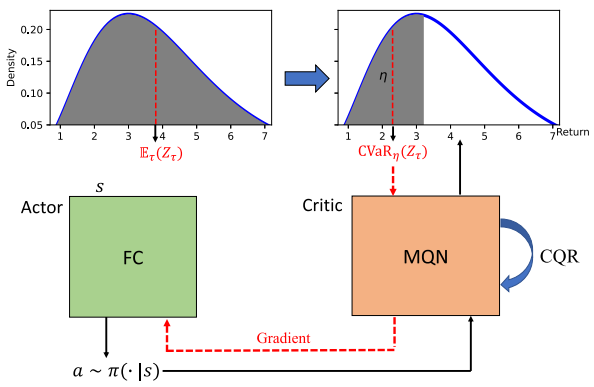

Fig. 5. Illustration of risk-averse policy training with CVaR. The actor contains several FC layers, and the critic is the proposed MQN. Top: we show two same return distributions. The left part calculates the average return, and the right part calculates CVaR_η return, where η is the area of the shadow part. CVaR_η measures the bottom η percentile of the value distribution. Bottom: with quantile estimation, CVaR_η can be calculated by averaging the quantile values by sampling $\tau_i \le \eta$. The gradient of CVaR_η is propagated through MQN to update the parameters of the policy network.

C. Risk-Averse Policy Learning

In this section, we introduce the risk-averse policy optimization in our approach. As an actor-critic framework, the proposed MQN is the critic network. We use CQR to train the critic to learn non-crossing and conservative quantile estimation $Z_\tau(s,a)$ for $\tau \in [0,1]$. Based on the quantile estimation, we learn an actor $\pi(a|s)$ by maximizing some risk-averse criteria of the value distribution. We choose CVaR as the risk criterion in our approach, which measures the risk through the following form:

$$\text{CVaR}_\eta = \eta\tau$$

where η is the parameter. For sampled quantile fractions, CVaR applies a linear transformation and modifies the distribution of quantile fractions from $\tau \approx U([0,1])$ to $U([0,\eta])$. For quantile function $Z_\tau(s,a)$, the risk-averse Q function is calculated under CVaR_η risk measure as follows:

$$\text{CVaR}_\eta(Z(s,a)) = \mathbb{E}_{\tau \approx U[0,\eta]}[Z_\tau(s,a)] = \frac{1}{K}\sum_{k=0}^{K-1} Z_{\eta(\tau)}(s,a)$$

where we use K samples to estimate the risk-averse Q function. As we illustrated in Fig. 5, CVaR measures the bottom η percentile of the value distribution, which is worst case of possible future outcomes. Then, the policy will learn how to perform well in such worst-case scenarios. In risk-sensitive learning, it is desirable to maximize the worst case instead of

Algorithm 1 MQN With CQR for Risk-Sensitive Offline RL

Initialize the actor-critic network, and the offline dataset $\mathcal{D}$, hyper-parameters α, η, and learning rate.
while not converged **do**
  Sample mini-batch transitions (s, a, r, s') from $\mathcal{D}$
  # *Critic Training*
  Calculate the monotonic quantile distribution $Z(s, a)$ and target value distribution $\widehat{\mathcal{T}}Z(s, a)$
  Sample $a_{\text{ood}} \sim \pi$ to calculate $F^{-1}_{Z(s,a_{\text{ood}})}(\tau)$.
  Train the critic by minimizing $\mathcal{L}_{\text{CQR}}$ through SGD
  # *Actor Training*
  Sample an action from π and calculate $\text{CVaR}_\eta(Z_\tau)$
  Train the actor by minimizing the CVaR measure
end while

the average-case performance, since the value distribution is usually heavy tailed. Then, the actor loss for training policy is defined as follows:

$$\mathcal{L}_{\text{policy}} = -\mathbb{E}_{s\in\mathcal{D},a\approx\pi}\big[\text{CVaR}_\eta(Z_\tau(s,a))\big]. \tag{21}$$

In MQN with quantile estimation, CVaR_η can be easily calculated by averaging the quantile values by sampling $\tau_i \le \eta$. The risk-averse policy learned by minimizing $\mathcal{L}_{\text{policy}}$ provides more robust policies by minimizing long-tail risks, which reduces the likelihoods of unsafe behaviors in offline RL.

We show the algorithmic description of our approach in Algorithm 1. In the offline setting, the transitions are sampled from $\mathcal{D}$. For critic training, we use MQN to calculate the value distribution, target value distribution, and sampled quantile of OOD action in $\mathcal{L}_{\text{CQR}}$ and then perform gradient descent to train the critic. For actor training, we follow Fig. 5 to calculate CVaR_η of the state-action pair, where the action is sampled from the learned policy. Then, we maximize CVaR_η to train the risk-averse policy. The code of our approach is available at https://github.com/Baichenjia/MQN-offline.

IV. Related Work

Distributional RL aims to learn the value distribution rather than its expectation. C51 [25], [44] represents the value distribution through fixed locations with categorical probabilities. QR-DQN [36] uses fixed quantile estimation with continuous probabilities and performs quantile regression as a surrogate objective for Wasserstein distance. IQN [26] and fully parameterized quantile function (FQF) [38] extend QR-DQN by uniformly sampled or self-trained quantiles fractions. The crossing quantile issue has been addressed in non-crossing QR-DQN (NC-QR-DQN) [32] and non-decreasing quantile function network (NDQFN) [45], while both works need fixed supported quantiles. In contrast, we learn continuous quantile fractions in an actor-critic framework through MQN. The continuous quantile function makes our method more flexible in optimizing CVaR objective CVaR_η in a risk-sensitive setting, where η can be any real number in $[0, 1]$. Meanwhile, previous research focuses on online RL setting, while we study the risk-sensitive learning in offline RL settings.

Offline RL algorithms typically rely on policy constraints, uncertainty-based penalization, and conservative value functions to address the distribution shift problem. The policy constraint methods restrict the learned policy to be close to the behavior policy through behavior cloning (BC) [13], [16], divergence measurement [14], [15], [46], and advantage-weighted constraints [47], [48]. The uncertainty-based methods use bootstrapping [20], [21], [49] and Dropout [22] to measure the uncertainty for pessimism. Conservative value functions, such as CQL [17], conservative offline distributional reinforcement learning (CODAC) [18], and conservative data-sharing (CDS) [50], minimize the Q values of OOD samples to prevent overestimations. Our method is related to CQL in the sense that both methods enforce conservatism in Q learning. However, we perform quantile regression rather than the ordinary Q function to measure risks and perform risk-averse policy learning.

Distributional value function has been used for offline settings very recently. Random ensemble mixture (REM) [51] directly applies QR-DQN in offline RL setting without considering the distribution shift problem with large-scale Atari datasets. Such a problem has been studied very recently [1], and the authors find that exploratory data allow vanilla off-policy RL algorithms, without any offline-specific modifications, to outperform or match state-of-the-art offline RL algorithms on downstream tasks. We leave the extension in sufficient diverse dataset in the future. Nevertheless, in an ordinary offline dataset with limited samples, it is necessary to enforce pessimism to prevent overestimation. Offline risk-averse actor-critic (ORAAC) [33] combines IQN with BC [13] as policy constraints, and CODAC [18] proposes to learn conservative quantile function to penalize the OOD actions. Compared with these two methods, the proposed MQN ensures the learned quantile function is non-crossing, which is essential for our theoretical analysis and also increases the empirical performance.

Risk-sensitive RL is related to robust MDPs [52]. Most previous methods aim to choose safe actions in interacting with the environment [53]. In offline RL, since we cannot modify the interaction stage to prevent unsafe exploration, we instead focus on learning a risk-averse policy in training. Distributional RL combined with CVaR has been used for risk-sensitive learning in IQN [26], distributional soft actor-critic (DSAC) [54], [55], and worst-case policy gradient [56]. The traditional control theories also study the risk-sensitive learning problem. However, they often consider a linear stochastic control system and exponential quadratic cost [28], which are unnecessary in our algorithm. We apply neural network to model the underlying dynamics and CVaR objective to optimize worst-case returns.

V. Experimental Results

In this section, we first conduct experiments in standard D4RL tasks [23] with a risk-neutral policy (i.e., set $\eta = 1.0$ in CVaR_η). Then, we set $\eta < 1.0$ and compare our approach with other baselines in risk-sensitive offline D4RL tasks. Finally, we provide visualization and ablation studies of our method.

A. Risk-Neutral Experiments

The D4RL domain includes three environments, namely, *HalfCheetah*, *Hopper*, and *Walker2d*. Each environment has

TABLE I

PERFORMANCE COMPARISON ON MULTI-JOINT DYNAMICS WITH CONTACT (MUJOCO) TASKS FROM THE STANDARD D4RL. WE REPORT THE AVERAGE NORMALIZED SCORE AND THE STANDARD DEVIATION FOR EACH TASK AFTER 1M TRAINING STEPS OVER FIVE SEEDS. THE HIGHEST PERFORMING SCORES ARE HIGHLIGHTED. THE SCORES OF BASELINES ARE ADOPTED FROM THE CORRESPONDING PAPERS

	Task Name	SAC	REM	BC	BCQ	BEAR	MOPO	CQL	ORAAC	CODAC	MQN-CQR (ours)
Random	halfcheetah	30.5	-0.8	2.1	2.2	25.5	**35.4**	**35.4**	13.5	34.6 ± 1.3	32.6 ± 2.9
	hopper	11.3	3.4	9.8	9.2	9.5	11.7	10.8	9.8	11.0 ± 0.4	**13.2** ± 0.6
	walker2d	4.1	6.9	1.6	4.8	6.7	13.6	7.0	3.2	18.7 ± 4.5	**22.6** ± 6.1
medium	halfcheetah	-4.3	-0.8	36.1	35.4	38.6	42.3	44.4	41.0	**46.3** ± 1.0	45.1 ± 1.5
	hopper	0.8	0.7	29.0	37.1	47.6	31.2	58.0	14.8	70.9 ± 11.5	**94.7** ± 13.2
	walker2d	0.9	0.2	6.6	19.2	33.2	17.8	79.2	27.3	**82.0** ± 0.5	80.0 ± 0.5
Medium Replay	halfcheetah	-2.4	6.6	38.4	29.5	36.2	**53.1**	46.2	30.0	44.0 ± 0.8	45.3 ± 7.9
	hopper	3.5	27.5	11.8	15.7	25.3	67.5	48.6	16.3	**100.2** ± 1.0	95.6 ± 18.5
	walker2d	1.9	12.5	11.3	8.3	10.8	39.0	26.7	28.0	33.2 ± 17.6	**52.3** ± 16.7
Medium Expert	halfcheetah	1.8	0.7	35.8	59.2	51.7	63.3	62.4	24.0	**70.4** ± 19.4	**71.1** ± 4.9
	hopper	1.6	0.8	111.9	66.6	4.0	23.7	111.0	18.2	112.0 ± 1.7	**113.0** ± 0.5
	walker2d	1.9	-0.1	11.3	19.1	10.8	44.6	98.7	28.2	106.0 ± 4.6	**112.1** ± 8.9
Expert	halfcheetah	-1.9	4.1	107.0	106.0	108.2	0.1	104.8	102.4	106.7 ± 0.3	**109.1** ± 0.2
	hopper	0.7	0.8	109.0	107.6	110.3	108.3	109.9	101.1	110.0 ± 3.0	**111.3** ± 0.1
	walker2d	-0.3	1.0	125.7	128.5	106.1	105.6	**153.9**	104.5	91.9 ± 25.8	113.1 ± 1.5
	Average	3.3	4.2	43.2	43.2	41.6	43.8	66.5	37.5	69.2	**74.1**

five dataset types, including *random*, *medium*, *medium replay*, *medium expert*, and *expert*, leading to a total of 15 task setups. The different types indicate different behavior policies in generating the dataset. The *random*, *medium*, and *expert* datasets are generated by single policies with different policy levels. The *medium-replay* dataset is the replay buffer in training a medium-level policy, thus containing experiences from different policies. The *medium-expert* dataset is a mixed dataset generated by a medium policy and an expert policy.

For implementation, we use soft actor-critic (SAC) from rlkit codebase[1] as the basic actor-critic algorithm and use the same hyperparameters as rlkit for SAC. We choose $\alpha = 10.0$ for the conservative penalty in CQR. The learning rate for actor and critic is set to be $3e-5$ and $3e-4$, respectively. We use $N = 32$ quantile fractions to represent the value distribution. In MQN, each FC layer contains 256 units with ReLU activations. The batch size is 256. We refer to our released code[2] for the details.

In the standard D4RL dataset, we compare our approach with several state-of-the-art methods, including the following.

1) *SAC [58]:* Using SAC to learn a policy offline dataset.
2) *BC:* Performing BC in the offline dataset.
3) *REM [51]:* Using REM to enforce optimal Bellman consistency on random convex combinations of multiple Q-value estimates.
4) *Batch-Constrained Q-Learning (BCQ) [13]:* Regularizing the output of the actor to force the agent toward behaving close to the offline dataset.
5) *Bootstrapping Error Accumulation Reduction (BEAR) [15]:* Using maximum mean discrepancy (MMD) distance to constrain the learned policy to be close to the behavior policy.
6) *Model-Based Offline Policy Optimization (MOPO) [59]:* In a model-based setting, MOPO measures the uncertainty through ensemble dynamics and uses the uncertainty to penalize the rewards of OOD actions.
7) *CQL [17]:* Performing conservative value estimation by penalizing the value function of OOD actions.
8) *ORAAC [33]:* Learning a distributional value function and using BCQ [13] to constrain the distance between the learned policy and the behavior policy.
9) *CODAC [18]:* Combining CQL [17] with distributional RL to perform CQR.

In each task, we train offline RL methods for one million time steps by sampling batches from the offline dataset. Our method is trained for five random seeds, and then, we evaluate each trained policy by interacting with the environment for 100 episodes, resulting 500 evaluated episodes totally for each task. We remark that the online interaction is only used for evaluation, while the training process is purely offline in our experiments. We calculate the normalized return through

$$\text{normalized score} = \frac{\text{score} - \text{random score}}{\text{expert score} - \text{random score}} \tag{22}$$

where the expert score is obtained by training an expert SAC agent in an online RL setting [23]. The expert and random scores can be found in the official repository of D4RL.[3]

We show the result comparison in Table I. Most scores of baselines are adopted from their original paper. For tasks that are not reported in some baselines, we run their official implementations to train the policy and use the same evaluation method to obtain the scores. All baseline methods are trained for the same time steps.

According to Table I, REM performs significantly worse than BC and other offline RL methods, which signifies that solely relying on distributional RL is insufficient to address the distribution shift problem in the offline setting. CQL and CODAC are the strongest baselines in standard D4RL tasks. Our method performs conservative learning to handle overestimation and learns the monotonic quantile distribution, thus outperforming other distributional-based methods that may contain crossing quantiles. Our method performs the best

[1]https://github.com/rail-berkeley/rlkit

[2]https://github.com/Baichenjia/MQN-offline

[3]https://github.com/rail-berkeley/d4rl/blob/master/d4rl/infos.py

TABLE II
COMPARISON OF DT AND OUR METHOD IN D4RL. THE SCORES OF DT ARE ADOPTED FROM THE ORIGINAL PAPER [57]

Dataset	Task	DT [57]	MQN-CQR (ours)
Medium	Halfcheetah	42.6	**45.1**
	Hopper	67.6	**94.7**
	Walker2d	74.0	**80.0**
Medium-Replay	Halfcheetah	36.6	**45.3**
	Hopper	82.7	**95.6**
	Walker2d	**66.6**	52.3
Medium-Expert	Halfcheetah	**86.8**	71.1
	Hopper	107.6	**113.0**
	Walker2d	108.1	**112.1**
	Average	74.73	**78.8**

in 9 out of 15 tasks and performs better than CODAC and CQL evaluated by the average performance of all tasks.

Beyond offline RL algorithms, we compare our method to decision Transformer (DT) [57] that converts offline RL to a sequence prediction problem. We compare the performance on the medium, medium-replay, and medium-expert datasets. According to Table II, our method outperforms DT on average scores. We find DT performs better than our method on Walker2d-medium-replay and Halfcheetah-medium-expert, which signifies that DT is also a strong baseline in offline settings through supervised learning.

B. Risk-Sensitive Experiments

We use stochastic D4RL tasks released in ORAAC [33] for risk-sensitive experiments. Incorporating stochasticity in the standard task aims to inject risks into the original reward. As we discussed in the path-selection example in Fig. 1, since the second path may have a very long-time waiting, a risk-averse policy will not choose this path. In stochastic D4RL tasks, the reward function is modified by adding some *catastrophic events* with small probabilities, which give meaningful assessments of risks. A risk-averse policy will try to avoid risks by optimizing the worst-case returns in training.

In stochastic D4RL tasks, we choose the *medium*, *medium-replay*, and *expert* datasets in risk-sensitive experiments. The original reward function $r(s,a)$ in these datasets is modified to risk-sensitive reward function $\bar{r}(s,a)$ in the following ways.

1) In *Halfcheetah*, the new reward is set to be
$$\bar{r}(s,a) = r(s,a) - q\,\mathbb{1}_{\upsilon\geq\bar{\upsilon}}\mathcal{B}_{0.1}$$
where $\mathcal{B}$ indicates the Bernoulli distribution. With a probability of 10%, the reward function is penalized by a risk event (i.e., $\upsilon \geq \bar{\upsilon}$). υ denotes the forward velocity, and $\bar{\upsilon}$ is a threshold velocity ($\bar{\upsilon} = 4, q = 70$ for the medium/medium-replay datasets, and $\bar{\upsilon} = 10, q = 70$ for the expert dataset). The risk event $\upsilon \geq \bar{\upsilon}$ indicates a high-velocity event that may cause damage to the robots, and we want to learn a policy that is risk averse to this catastrophic event.
2) In *Walker2d* and *Hopper*, the new reward is set to be
$$\bar{r}(s,a) = r(s,a) - p\,\mathbb{1}_{|\theta|\geq\bar{\theta}}\mathcal{B}_{0.1}$$
where θ is the pitch angle, and $\bar{\theta}$ is a threshold angle ($\bar{\theta} = 0.5$, $p = 30$ for *Walker2d* task, and $\bar{\theta} = 0.5$, $p = 50$ for *Hopper* task). Since $\theta \geq 2\bar{\theta}$ causes the robot falls and the episode terminates, the risk event is set to be $\theta \geq \bar{\theta}$.

In evaluation, we report both the average score and the "worst-0.1" score. Specifically, we evaluate the trained policy for 100 episodes and record the accumulated rewards for each episode as $\boldsymbol{R} = \{\bar{R}_1, \bar{R}_2, \ldots, \bar{R}_{100}\}$; then, we sort these rewards and calculate the 0.1-percentile score of $\boldsymbol{R}$ as the "worst-0.1" score. This metric represents the worst case of returns, and we want the risk-averse policy to be good in the worst cases.

In risk-sensitive experiments, we use ORAAC [33] and CODAC [18] as baselines, since they are able to perform risk-sensitive learning by using risk measures. Other baselines can only optimize the expected return, since they do not learn the return distribution. We use CQL [17] as a baseline in a risk-sensitive setting, since it performs the best in non-distributional methods in standard D4RL tasks. For risk-sensitive methods, we set $\eta = 0.1$ and optimize $\text{CVaR}_{0.1}$ criteria to train the policies. In Table III, we show the result comparison of all methods. In Halfcheetah and Hopper tasks, we find all baselines have significant performance drop compared with the standard D4RL version. In contrast, our method significantly outperforms other methods and obtains reasonable performance in Halfcheetah and Hopper tasks. Our method ensures the non-crossing property of quantile functions, thus performing more stable in the difficult risk-sensitive settings. Surprisingly, we find CQL performs well in walker2d tasks; one possible reason is optimizing, and CVaR is somewhat high variance, since the value distribution can be complex.

In addition, we run the trained agent in medium-replay tasks for several episodes and record the risk events that are represented by $\upsilon \geq \bar{\upsilon}$ in Halfcheetah and $\theta \geq \bar{\theta}$ in walker2d. In Table IV, we show the ratio of the occurrence of the violation. Our method has the lowest chance to encounter risk events. We also report the average velocity and angle in evaluation. A lower velocity or smaller angle indicates the agent performs safer in the corresponding tasks. We give the supplementary videos at https://sites.google.com/view/mqncqr.

C. Visualization and Ablation

1) MQN Constraint: We visualize the value distribution in the training process. In each training step, we calculate the following: 1) 10%-high quantile value as $Z_{\text{high}} = \mathbb{E}_{\tau\approx U([0.9,1])}[Z_\tau]$; 2) 10%-low quantile value as $Z_{\text{low}} = \mathbb{E}_{\tau\approx U([0,0.1])}[Z_\tau]$; and 3) the mean quantile value as $Z_{\text{mean}} = \mathbb{E}_{\tau\approx U([0,1])}[Z_\tau]$ for the training batches. Ideally, we have $Z_{\text{low}} < Z_{\text{mean}} < Z_{\text{high}}$, since the quantile function is monotonic. In Fig. 6, we visualize $Z_{\text{low}}, Z_{\text{mean}}, Z_{\text{high}}$ of CODAC and our method in Hopper medium and medium-replay tasks. We find such a relationship does not always hold in CODAC, especially in the early stage of training. In the medium-expert dataset, the quantile distribution in CODAC is crossing, and we observe $Z_{\text{low}} \geq Z_{\text{high}}$ in the early stage of training, which hurts performance and leads to suboptimal policies. In contrast, our method ensures the non-crossing property of quantile functions through MQN and

TABLE III

RESULT COMPARISON OF MEAN SCORE AND WORST-0.1 SCORE ON MUJOCO TASKS FROM THE STOCHASTIC D4RL

		Medium		Medium-Replay		Expert	
		Mean	Worst-0.1	Mean	Worst-0.1	Mean	Worst-0.1
halfcheetah	**CQL**	2.5 ± 2.4	2.1 ± 2.3	4.0 ± 2.7	2.4 ± 2.4	1.7 ± 2.4	0.6 ± 2.6
	ORAAC	5.2 ± 2.4	3.0 ± 2.6	4.7 ± 2.3	3.2 ± 2.5	7.1 ± 2.6	3.1 ± 2.8
	CODAC	5.0 ± 2.9	2.0 ± 2.7	5.4 ± 2.7	4.2 ± 2.7	6.7 ± 3.3	3.5 ± 3.3
	MQN-CQR (ours)	**28.6** ± 0.6	**28.2** ± 0.4	**26.3** ± 0.7	**25.6** ± 0.7	**89.6** ± 1.3	**37.6** ± 18.2
hopper	**CQL**	27.6 ± 6.6	21.9 ± 5.6	6.4 ± 2.6	0.0 ± 2.5	36.4 ± 2.4	27.8 ± 4.7
	ORAAC	31.6 ± 2.4	24.2 ± 3.7	27.5 ± 3.3	16.8 ± 10.5	36.2 ± 11.1	24.2 ± 12.1
	CODAC	31.8 ± 9.3	30.6 ± 9.0	48.3 ± 1.6	45.2 ± 3.7	39.7 ± 2.9	30.9 ± 3.7
	MQN-CQR (ours)	**59.0** ± 13.9	**37.3** ± 7.0	**77.0** ± 4.9	**54.0** ± 7.4	**109.5** ± 2.2	**86.3** ± 24.6
walker2d	**CQL**	**33.2** ± 1.9	**29.2** ± 5.4	1.6 ± 1.6	-1.4 ± 1.7	44.5 ± 0.8	40.7 ± 1.2
	ORAAC	24.7 ± 5.1	14.4 ± 7.6	4.8 ± 0.8	-1.6 ± 1.6	21.6 ± 4.4	2.3 ± 1.6
	CODAC	24.4 ± 6.9	19.6 ± 10.7	**9.8** ± 4.2	5.7 ± 5.0	**44.8** ± 0.9	**41.1** ± 0.6
	MQN-CQR (ours)	21.6 ± 7.8	9.1 ± 6.4	7.3 ± 0.6	**7.0** ± 0.7	17.4 ± 12.5	4.9 ± 1.1

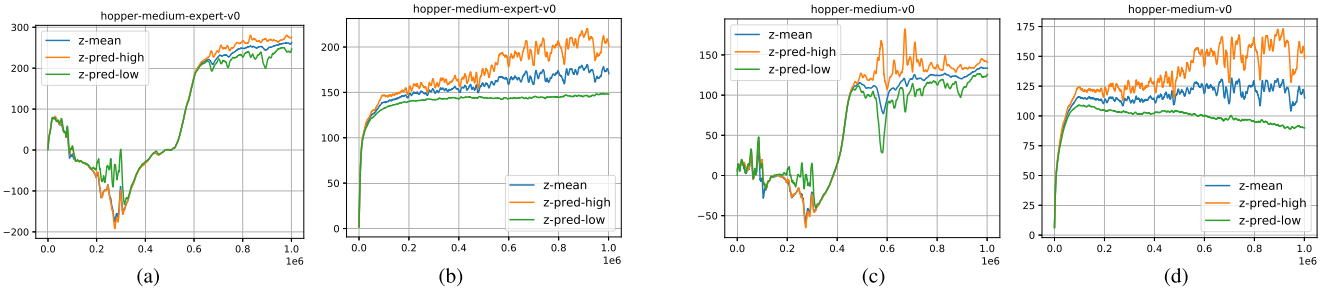

Fig. 6. Visualization of quantile distributions in (a) and (b) hopper-medium-replay and (c) and (d) hopper-medium. "z-mean" is the mean quantile value as $Z_{\text{mean}} = \mathbb{E}_{\tau \approx U([0,1])}[Z_\tau]$, "z-pred-high" is the 10%-high quantile value as $Z_{\text{high}} = \mathbb{E}_{\tau \approx U([0.9,1])}[Z_\tau]$, and "z-pred-low" is the 10%-low quantile value as $Z_{\text{low}} = \mathbb{E}_{\tau \approx U([0,0.1])}[Z_\tau]$. The value distributions of CODAC shown in (a) and (c) have crossing quantiles, which hurt performance and lead to suboptimal policies. Our method overcomes this issue through MQN, as shown in (b) and (d).

TABLE IV

VIOLATION RATIO COMPARISON IN MEDIUM-REPLAY TASK

	HalfCheetah		Walker2d	
	% Violation	Velocity	% Violation	Angle
MQN-CQR	**8**	**1.31**	**13**	**0.14**
CODAC	11	1.49	15	0.28
CQL	23	1.71	13	0.19
ORAAC	37	1.76	48	0.49

has $Z_{\text{low}} < Z_{\text{mean}} < Z_{\text{high}}$ in the training process. Similar results also occur in the medium dataset.

Furthermore, we add an ablation study to show the importance of MQN in stochastic environments. Specifically, we replace MQN with an ordinary quantile network used in IQN [26]. We train both methods with $\text{CVaR}_{0.1}$ objective in a risk-averse manner. Fig. 7 compares the average return in *Hopper-medium*, *Hopper-medium-replay*, and *Hopper-medium-expert* tasks. We find the proposed MQN network brings significant improvement compared with an ordinary quantile network. MQN provides non-crossing constraints in quantile regression to prevent the collapsed value distribution, thus improving the stability of the policy in stochastic environments with catastrophic events.

2) CQR Penalty: As defined in (15), the conservative penalty αF_Z^{-1} for OOD actions in CQR prevents the overestimation caused by distribution shift in offline training. We illustrate the Q value of (s, a) pairs sampled from the offline data, and (s, a_{random}) pairs that contain OOD actions uniformly sampled from the action space. The Q value is calculated by taking the expectation of quantile distribution.

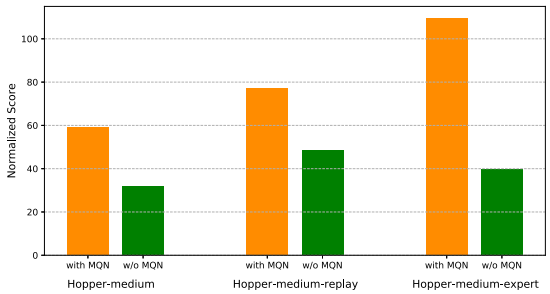

Fig. 7. Ablation study of MQN in stochastic D4RL with and without MQN. We perform comparison in Hopper task with medium, medium-replay, and medium-expert datasets. We report the averaged normalized score. The proposed MQN improves the performance in risk-sensitive policy learning.

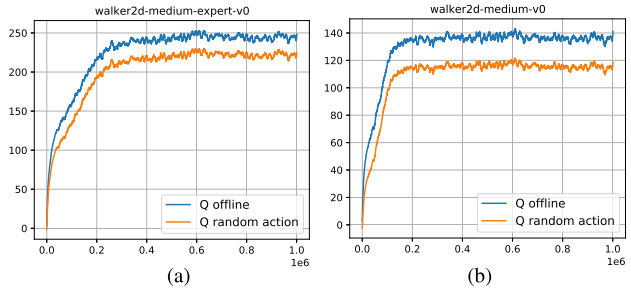

Fig. 8. Visualization of Q values in two D4RL tasks. Since we add penalty in quantile regression, the Q values of OOD actions are significantly lower than the Q values of offline data. (a) Walker2d-medium-expert. (b) Walker2d-medium.

As shown in Fig. 8, the Q values of (s, a_{random}) pairs are significantly lower than the Q values of offline data. Through

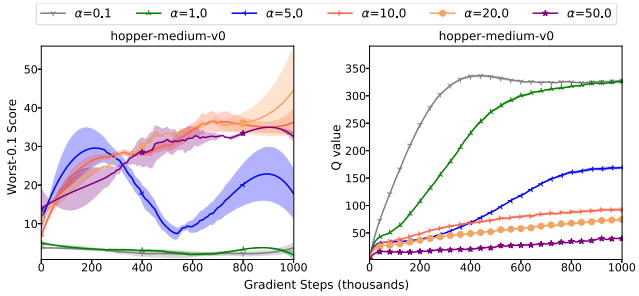

Fig. 9. Ablation study of the CQL penalty. We visualize the worst-0.1 scores and Q values in stochastic Hopper-medium-v0 task. Left: we show the evaluated score in the training process with different settings of α. The agent performs best when $\alpha \in \{10, 20\}$. Right: we show the Q values of (s, a) pairs sampled from the offline dataset. We find a larger α results in a more pessimistic value function.

TABLE V
COMPARISON OF WORST-0.1 SCORE WITH DIFFERENT CVAR SETTINGS IN STOCHASTIC D4RL

	$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.5$	$\eta = 0.9$
hopper-med	37.3 ± 7.0	31.3 ± 5.1	30.8 ± 0.9	29.3 ± 0.1
hopper-med-rep	54.0 ± 7.4	42.7 ± 2.5	31.4 ± 3.2	33.8 ± 0.9

CQR, the value estimation becomes conservative for OOD actions, which prevents the distribution shift problem in the offline setting. We conduct an ablation study of CQR penalty in Fig. 9. Since α controls the strength of CQL penalty in our method, we try different settings of α for the penalty term to show the importance of conservatism in *Hopper-medium* task. According to the results, we find a relatively large penalty (i.e., $\alpha \in \{10, 20\}$) performs the best in this task. A very small α (i.e., $\alpha \leq 5$) cannot prevent overestimation for OOD actions, which signifies that conservatism is crucial for quantile regression in offline setting. In addition, we find an extremely large α (i.e., $\alpha = 50$) results in an overly pessimistic value function and hinders performance.

3) CVaR Levels: The experiments of stochastic D4RL tasks (i.e., Table III) use a CVaR_η objective to optimize the worst-case return, where we use $\eta = 0.1$ in practice. We conduct an ablation study to compare the evaluated scores with various η settings in hopper-medium and hopper-medium-replay tasks, and the results are given in Table V. In stochastic D4RL tasks, since the risk events happen by following a Bernoulli distribution with probability of 0.1, we find it is necessary to set a small η to prevent the risk events. A large η leads to risk-neutral behavior and becomes less sensitive to the risks; thus, the evaluated scores decrease with the increase in η. Nevertheless, we find a larger η has a lower score variance, since it considers a larger part of value distribution and results in a lower variance in estimation. Developing methods for reducing the variance in estimating CVaR is an interesting future direction.

VI. CONCLUSION

In this article, we propose a risk-averse offline RL algorithm through MQN and CQR. Theoretically, we prove that our method converges to a fixed point and obtains a lower bound of the true value distribution. Empirically, we show our method obtains the state-of-the-art performance in risk-neutral D4RL tasks. In risk-sensitive tasks, all baselines have significant performance drop, while our method performs more stable and has the lowest chance to encounter risk events according to the recorded velocities and angles. As shown in ablation studies, the non-crossing constraint and conservatism are both crucial for risk-sensitive offline RL. Training different CVaR setting provides various risk-averse policies with continuous quantiles. In our future work, we wish to extend our method to risk-sensitive multitask datasets to solve more complex offline RL tasks.

APPENDIX A
PROOF OF LEMMA 1

Lemma 1: For an MDP with finite state and action spaces, $\widetilde{\mathcal{T}}^\pi$ is a γ contraction in $\bar{d}_\infty$ as follows:

$$\bar{d}_\infty\big(\Pi_{W_1}\widetilde{\mathcal{T}}^\pi Z_1, \Pi_{W_1}\widetilde{\mathcal{T}}^\pi Z_2\big) \leq \gamma\, \bar{d}_\infty(Z_1, Z_2)$$

where Z_1, Z_2 are value distributions, and Π_{W_1} is the quantile projection defined in (10).

Proof: The proof basically follows QR-DQN [36]. According to [36, Proposition 2], for the empirical Bellman operator $\widehat{\mathcal{T}}^\pi$, we have

$$\bar{d}_\infty\big(\Pi_{W_1}\widehat{\mathcal{T}}^\pi Z_1, \Pi_{W_1}\widehat{\mathcal{T}}^\pi Z_2\big) \leq \gamma\, \bar{d}_\infty(Z_1, Z_2).$$

As we will show in (32), the CQR operation uses an additional shift operator as $F^{-1}_{\widehat{\mathcal{T}}Z}(\tau) = F^{-1}_{\widehat{\mathcal{T}}Z}(\tau) - c_{\alpha,p}$. Since the shift operator is not non-expansion in $\bar{d}_p$, $\widetilde{\mathcal{T}}$ is also a γ contraction in $\bar{d}_\infty$. □

APPENDIX B
PROOF OF LEMMA 2

Lemma 2: Suppose two cdfs satisfies $\|F - G\|_\infty \leq \epsilon$, then $\|F^{-1} - G^{-1}\|_\infty \leq \epsilon/\zeta$ holds for quantile functions.

Proof: For the invertible cdf $y = F(x)$ and the continuously differentiable function $x = F^{-1}(y)$, we have

$$\big(F^{-1}\big)'(y) = \frac{1}{F'(x)} = \frac{1}{F'\big(F^{-1}(y)\big)} \leq \frac{1}{\zeta}$$

by following the assumption that the cdf is ζ-strongly monotone, and the derivation follows the inverse function theorem. Then, we have

$$\begin{aligned} F^{-1}(y) - G^{-1}(y) &= \int_{G^{-1}(y)}^{F^{-1}(y)} dx = \int_{F\left(G^{-1}(y)\right)}^{y} dF^{-1}(y') \\ &\leq \frac{1}{\zeta}\int_{F\left(G^{-1}(y)\right)}^{y} dy' \\ &= \frac{1}{\zeta}\big[G\big(G^{-1}(y)\big) - F\big(G^{-1}(y)\big)\big] \leq \frac{\epsilon}{\zeta} \end{aligned}$$

where the second step changes $y' = F(x)$, the third step follows the cdf function is ζ-strongly monotone, and the last step holds, since $\|F - G\|_\infty \leq \epsilon$. To conclude, we have

$$\max_y \big|F^{-1}(y) - G^{-1}(y)\big| = \big\|F^{-1} - G^{-1}\big\|_\infty \leq \epsilon/\zeta.$$

□

APPENDIX C
PROOF OF THEOREM 1

Lemma 3: For a Bellman operator $\mathcal{T}^\pi$ and $\tau \in [0, 1]$, if the quantile function after one-step Bellman operator satisfies

$$F_Z^{-1}(\tau) \geq F_{\mathcal{T}^\pi Z}^{-1}(\tau) + b \tag{23}$$

then the fixed point Z^π of $\mathcal{T}^\pi Z$ satisfies

$$F_Z^{-1}(\tau) \geq F_{Z^\pi}^{-1}(\tau) + \frac{b}{1-\gamma}. \tag{24}$$

Proof: Since we have $F_Z^{-1}(\tau) \geq F_{\mathcal{T}^\pi Z}^{-1}(\tau) + b$, then $F_{\mathcal{T}^\pi Z}(F_Z^{-1}(\tau) - b) \geq \tau$ holds by applying $F_{\mathcal{T}^\pi Z}$ function in both sides. By substituting $\tau = F_Z(x + b)$, we have

$$F_{\mathcal{T}^\pi Z}(x) \geq F_Z(x + b). \tag{25}$$

Then, we substitute $y = x/\gamma$ for x and obtain

$$F_{\mathcal{T}^\pi Z}(x/\gamma) \geq F_Z(x/\gamma + b).$$

Then

$$F_{\gamma \mathcal{T}^\pi Z}(x) \geq F_{\gamma Z}(x + \gamma b). \tag{26}$$

By following the law of $F_Z(z) = \int F_X(z - y) dF_Y(y)$ if $z = x + y$, we apply additional Bellman operator as follows:

$$\begin{aligned}
&F_{\mathcal{T}^\pi(\mathcal{T}^\pi Z(s,a))}(x) \\
&= \sum_{s'} P\left(s' \middle| s, a\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} F_{r+\gamma \mathcal{T}^\pi Z(s',a')}(x) \\
&= \sum_{s'} P\left(s' \middle| s, a\right) \mathbb{E}_{\pi(a'|s')} \int F_{\gamma \mathcal{T}^\pi Z(s',a')}(x - r) dF_r(r) \\
&\geq \sum_{s'} P\left(s' \middle| s, a\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} \int F_{\gamma Z(s',a')}(x - r + \gamma b) dF_r(r) \\
&= \sum_{s'} P\left(s' \middle| s, a\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} F_{r+\gamma Z(s',a')}(x + \gamma b) \\
&= F_{\mathcal{T}^\pi Z(s,a)}(x + \gamma b)
\end{aligned}$$

where the inequality in the third line follows (26). Then

$$F_{\mathcal{T}^\pi Z}^{-1}\left(F_{\mathcal{T}^\pi(\mathcal{T}^\pi Z)}(x)\right) \geq x + \gamma b.$$

By setting $F_{\mathcal{T}^\pi(\mathcal{T}^\pi Z)}(x) = \tau$, we have

$$F_{\mathcal{T}^\pi Z}^{-1}(\tau) \geq F_{\mathcal{T}^\pi(\mathcal{T}^\pi Z)}^{-1}(\tau) + \gamma b = F_{(\mathcal{T}^\pi)^2 Z}^{-1}(\tau) + \gamma b. \tag{27}$$

By recursive applying (23) and (27), we have

$$\begin{aligned}
F_Z^{-1}(\tau) &\geq F_{\mathcal{T}^\pi Z}^{-1} + b \\
&\geq F_{(\mathcal{T}^\pi)^2 Z}^{-1} + \gamma b + b \\
&\geq F_{(\mathcal{T}^\pi)^3 Z}^{-1} + \gamma^2 b + \gamma b + b \\
&\geq F_{(\mathcal{T}^\pi)^k Z}^{-1} + \sum_{i=0}^{k-1} \gamma^i b.
\end{aligned}$$

With $k \to \infty$, we have

$$F_Z^{-1}(\tau) \geq F_{(\mathcal{T}^\pi)^\infty Z}^{-1} + \sum_{i=0}^{\infty} \gamma^i b = F_{(\mathcal{T}^\pi)^\infty Z}^{-1} + \frac{b}{1-\gamma}.$$

Then, since $(\mathcal{T}^\pi)^\infty Z = Z^\pi$, we have $F_Z^{-1}(\tau) \geq F_{Z^\pi}^{-1} + (b/(1-\gamma))$, which concludes our proof. □

Theorem 1: Assuming the behavior policy $\pi_\beta(a|s) \geq 0$. For $\forall s \approx \mathcal{D}$, a, with probability $\geq 1 - \delta$, the Z-distribution obtained by minimizing (15) satisfies

$$F_{\widehat{Z}^\pi}^{-1}(\hat{\tau}) \leq F_{Z^\pi}^{-1}(\hat{\tau}) - C(\alpha, p, \delta, \gamma, \zeta) \tag{28}$$

where $C(\alpha, p, \delta, \gamma, \zeta) \geq 0$ with a sufficiently large α.

Proof: First, we rewrite (2) as follows:

$$\begin{aligned}
\mathcal{L}_{\text{CQR}} &= \alpha \mathbb{E}_{\tau_c, a}\left[F_Z^{-1}(\tau_c)\right] + \mathcal{L}_{\text{QR}}\left(Z, \widehat{\mathcal{T}}^\pi \hat{Z}^k\right) \\
&= \alpha \mathbb{E}_{\tau_c, a}\left[F_Z^{-1}(\tau_c)\right] \\
&\quad + \mathbb{E}_{\mathcal{D}}\left(\int_0^1 \left|F_Z^{-1}(\tau) - F_{\widehat{\mathcal{T}}^\pi \hat{Z}}^{-1}(\tau)\right|^p d\tau\right)^{1/p}.
\end{aligned} \tag{29}$$

Since the solution $F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}(\tau)$ minimizes (29), we set the gradient of (29) to be zero, as follows:

$$\begin{aligned}
\nabla \mathcal{L}_{\text{CQR}} &= \int_0^1 \alpha \pi(a|s) + \frac{\pi_\beta(a|s)}{p} \nabla \left|F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}(\tau) - F_{\widehat{\mathcal{T}}^\pi \hat{Z}}^{-1}(\tau)\right|^p d\tau \\
&= \int_0^1 \alpha \pi(a|s) + c_s \pi_\beta(a|s) \left|F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}(\tau) - F_{\widehat{\mathcal{T}}^\pi \hat{Z}}^{-1}(\tau)\right|^{p-1} d\tau \\
&= 0
\end{aligned} \tag{30}$$

where we denote $c_s = \text{sign}(F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}(\tau) - F_{\widehat{\mathcal{T}}^\pi \hat{Z}}^{-1}(\tau))$. To ensure $\nabla \mathcal{L}_{\text{CQR}}$ is zero, the integrand must be zero. Considering if $c_s = 1$, then the integrand must greater than zero; thus, $c_s = -1$ holds. Then, we have

$$\alpha \pi(a|s) - \pi_\beta(a|s) \left|F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}(\tau) - F_{\widehat{\mathcal{T}}^\pi \hat{Z}}^{-1}(\tau)\right|^{p-1} = 0. \tag{31}$$

Then, we have

$$F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}(\tau) = F_{\widehat{\mathcal{T}}^\pi \hat{Z}}^{-1}(\tau) - c_{\alpha,p} \tag{32}$$

where $c_{\alpha,p} = (\alpha \pi / \pi_\beta)^{(1/(p-1))}$. Through (32), we connect $F_{\widetilde{\mathcal{T}}\tilde{Z}}^{-1}$ and $F_{\widehat{\mathcal{T}}\hat{Z}}^{-1}$, where $\widehat{\mathcal{T}}^\pi \hat{Z}$ is the empirical estimation of the true Z^π in given $\mathcal{D}$. Since Z^π is a fixed point, we have $Z^\pi = \mathcal{T}^\pi Z^\pi$. Next, we bound the error between $\widehat{\mathcal{T}}^\pi Z$ and $\mathcal{T}^\pi Z$.

We first derive the difference between the cdf of $F_{\widehat{\mathcal{T}}^\pi Z}$ and $F_{\mathcal{T}^\pi Z}$. Then, we derive the difference between of quantile function $F_{\widehat{\mathcal{T}}^\pi Z}^{-1}$ and $F_{\mathcal{T}^\pi Z}^{-1}$.

Following the distribution of Bellman operator, we have the following:

$$\begin{aligned}
&F_{\widehat{\mathcal{T}}^\pi Z}(x) - F_{\mathcal{T}^\pi Z}(x) \\
&= \sum_{s'} \widehat{P}\left(s' \middle| s, a\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} F_{\hat{r}+\gamma Z(s',a')}(x) \\
&\quad - \sum_{s'} P\left(s' \middle| s, a\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} F_{r+\gamma Z(s',a')}(x) \\
&\leq \sum_{s'} \widehat{P}\left(s' \middle| s, a\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} \left(F_{\hat{r}+\gamma Z(s',a')}(x) - F_{r+\gamma Z(s',a')}(x)\right) \\
&\quad + \sum_{s'} \left(\widehat{P}\left(s' \middle| s, a\right) - P\left(s' \middle| s, a\right)\right) \mathbb{E}_{\pi\left(a' \middle| s'\right)} F_{r+\gamma Z(s',a')}(x)
\end{aligned} \tag{33}$$

where we separate the error caused by empirical reward $\hat{r}$ and empirical transition function $\widehat{P}$. For the first term in (33),

we have

$$\begin{aligned}
&\sum_{s'} \widehat{P}\left(s'|s,a\right)\mathbb{E}_{\pi\left(a'|s'\right)}\left(F_{\hat{r}+\gamma Z(s',a')}(x) - F_{r+\gamma Z(s',a')}(x)\right)\\
&= \sum_{s'} \widehat{P}\left(s'|s,a\right)\mathbb{E}_{\pi\left(a'|s'\right)}\int [F_{\hat{r}}(r) - F_r(r)] f_{\gamma Z(s',a')}(x-r)dr\\
&\leq \sum_{s'} \widehat{P}\left(s'|s,a\right)\|F_{\hat{r}}(r) - F_r(r)\|_\infty\\
&= \|F_{\hat{r}}(r) - F_r(r)\|_\infty \qquad (34)
\end{aligned}$$

where the first equality holds by following the law of cdf as $F_Z(z) = \int F_X(z-y)dF_Y(y)$, the second inequality holds, since the cdf of any function is less than 1, and third inequality holds, since the sum of the transition probabilities is 1. Then, for the second term in (33), we have

$$\begin{aligned}
&\sum_{s'}\left(\widehat{P}\left(s'|s,a\right) - P\left(s'|s,a\right)\right)\mathbb{E}_{\pi\left(a'|s'\right)}F_{r+\gamma Z(s',a')}(x)\\
&\quad\leq \left\|\widehat{P}(\cdot|s,a) - P(\cdot|s,a)\right\|_1 \cdot \left\|\mathbb{E}_{\pi\left(a'|s'\right)}F_{r+\gamma Z(s',a')}(x)\right\|_\infty\\
&\quad= \left\|\widehat{P}(\cdot|s,a) - P(\cdot|s,a)\right\|_1 \qquad (35)
\end{aligned}$$

where we follow Holder's inequality. Then, we use the similar assumption as [60] about the presence of sampling error in the reward function and transition dynamics. $\forall s, a \in \mathcal{D}$, with high probability $\geq 1-\delta$, the following relationship holds by following Dvoretzky–Kiefer–Wolfowitz (DKW) inequality:

$$\begin{aligned}
\|F_{\hat{r}}(r) - F_r(r)\|_\infty &\leq \frac{c_{r,\delta}}{\sqrt{|\mathcal{D}|}}\\
\left\|\widehat{P}(\cdot|s,a) - P(\cdot|s,a)\right\|_1 &\leq \frac{c_{P,\delta}}{\sqrt{|\mathcal{D}|}} \qquad (36)
\end{aligned}$$

where $c_{r,\delta} = ((1/2)\ln(2/\delta))^{1/2}$ and $c_{P,\delta} = (2|\mathcal{S}|\ln(2/\delta))^{1/2}$, and $|\mathcal{D}|$ is the number of samples in the offline dataset. A larger $\mathcal{D}$ results in a smaller difference between the empirical distribution and the true distribution. Under this assumption, by adding (34) and (35), the bound in (33) becomes

$$\begin{aligned}
F_{\widehat{\mathcal{T}}^\pi Z}(x) - F_{\mathcal{T}^\pi Z}(x) &\leq \|F_{\hat{r}}(r) - F_r(r)\|_\infty\\
&\quad + \left\|\widehat{P}(\cdot|s,a) - P(\cdot|s,a)\right\|_1\\
&\leq \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}}. \qquad (37)
\end{aligned}$$

Then, we follow Lemma 2 to convert bounds of cdfs to inverse cdfs. By using $\epsilon = ((c_{r,\delta} + c_{P,\delta})/\sqrt{|\mathcal{D}|})$, we have

$$F^{-1}_{\widehat{\mathcal{T}}^\pi Z}(x) - F^{-1}_{\mathcal{T}^\pi Z}(x) \leq \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}\cdot\zeta} \qquad (38)$$

where ζ is the maximum gradient of the quantile function. Then, we follow (32) to connect $\widetilde{\mathcal{T}}Z^\pi$ and $\mathcal{T}^\pi Z^\pi$ as follows:

$$\begin{aligned}
F^{-1}_{\widetilde{\mathcal{T}}Z^\pi}(\tau) &= F^{-1}_{\widehat{\mathcal{T}}^\pi Z^\pi}(\tau) - c_{\alpha,p} \leq F^{-1}_{\mathcal{T}^\pi Z^\pi}(x) + \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}\cdot\zeta} - c_{\alpha,p}\\
&= F^{-1}_{Z^\pi}(x) + \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}\cdot\zeta} - c_{\alpha,p}. \qquad (39)
\end{aligned}$$

By following Lemma 3 and setting $c_{\alpha,p} - ((c_{r,\delta} + c_{P,\delta})/(\sqrt{|\mathcal{D}|}\cdot\zeta)) = \beta$, we obtain the fixed point $\tilde{Z}^\pi$ of $\widetilde{\mathcal{T}}Z^\pi$ as follows:

$$\begin{aligned}
F^{-1}_{Z^\pi}(x) &\geq F^{-1}_{\widetilde{\mathcal{T}}Z^\pi}(x) + \left(c_{\alpha,p} - \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}\cdot\zeta}\right)\\
&\geq F^{-1}_{(\widetilde{\mathcal{T}})^\infty Z^\pi}(x) + \sum_{i=0}^{\infty}\gamma^i\left(c_{\alpha,p} - \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}\cdot\zeta}\right)\\
&= F^{-1}_{\tilde{Z}^\pi}(x) + \left(c_{\alpha,p} - \frac{c_{r,\delta} + c_{P,\delta}}{\sqrt{|\mathcal{D}|}\cdot\zeta}\right)\Big/(1-\gamma)\\
&= F^{-1}_{\tilde{Z}^\pi}(x) + C(\alpha, p, \delta, \gamma, \zeta) \qquad (40)
\end{aligned}$$

where we set $C(\alpha, p, \delta, \gamma, \zeta) = (c_{\alpha,p} - ((c_{r,\delta} + c_{P,\delta})/(\sqrt{|\mathcal{D}|}\cdot\zeta)))/(1-\gamma)$. $(\widetilde{\mathcal{T}})^\infty Z = \tilde{Z}^\pi$ is the fixed point of $\widetilde{\mathcal{T}}$ for any Z. To ensure $C(\alpha, p, \delta, \gamma, \zeta) \geq 0$, we need $c_{\alpha,p} - (c_{r,\delta} + c_{P,\delta})/(\zeta\sqrt{|\mathcal{D}|}) \geq 0$. Thus, if α is sufficiently large, we have $C(\alpha, p, \delta, \gamma, \zeta) \geq 0$, $\forall s, a$, which concludes our proof. □

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[2] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, Oct. 2019.

[3] C. Bai et al., "Dynamic bottleneck for robust self-supervised exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.

[4] C. Bai et al., "Addressing hindsight bias in multigoal reinforcement learning," *IEEE Trans. Cybern.*, early access, Sep. 8, 2021, doi: 10.1109/TCYB.2021.3107202.

[5] T. Yang et al., "Exploration in deep reinforcement learning: A comprehensive survey," 2021, *arXiv:2109.06668*.

[6] C. Bai et al., "Variational dynamic for self-supervised exploration in deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 1, 2021, doi: 10.1109/TNNLS.2021.3129160.

[7] C. Bai et al., "Principled exploration via optimistic bootstrapping and backward induction," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, Jul. 2021, pp. 577–587.

[8] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax—A differentiable physics engine for large scale rigid body simulation," in *Proc. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021, pp. 1–21.

[9] J. Zhang, B. Zi, and X. Ge, "Attend2Pack: Bin packing through deep reinforcement learning with attention," 2021, *arXiv:2107.04333*.

[10] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," 2019, *arXiv:1908.08796*.

[11] N. A. Lynnerup, L. Nolling, R. Hasle, and J. Hallam, "A survey on reproducibility by evaluating deep reinforcement learning algorithms on real-world robots," in *Proc. Conf. Robot Learn.*, 2020, pp. 466–489.

[12] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.

[13] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2052–2062.

[14] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," 2019, *arXiv:1911.11361*.

[15] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy Q-learning via bootstrapping error reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 11784–11794.

[16] S. Fujimoto and S. Shane Gu, "A minimalist approach to offline reinforcement learning," 2021, *arXiv:2106.06860*.

[17] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1179–1191.

[18] Y. Jason Ma, D. Jayaraman, and O. Bastani, "Conservative offline distributional reinforcement learning," 2021, *arXiv:2107.06106*.

[19] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "COMBO: Conservative offline model-based policy optimization," 2021, *arXiv:2102.08363*.

[20] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified Q-ensemble," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–12.

[21] C. Bai et al., "Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–29.
[22] Y. Wu et al., "Uncertainty weighted actor-critic for offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 11319–11328.
[23] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," 2020, *arXiv:2004.07219*.
[24] M. Henaff, A. Canziani, and Y. LeCun, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–20.
[25] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 449–458.
[26] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1096–1105.
[27] M. Rigter, B. Lacerda, and N. Hawes, "Risk-averse Bayes-adaptive reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1–13.
[28] M. P. Chapman, R. Bonalli, K. M. Smith, I. Yang, M. Pavone, and C. J. Tomlin, "Risk-sensitive safety analysis using conditional value-at-risk," *IEEE Trans. Autom. Control*, early access, Nov. 26, 2021, doi: 10.1109/TAC.2021.3131149.
[29] Y. Hu, W. Wang, H. Liu, and L. Liu, "Reinforcement learning tracking control for robotic manipulator with kernel-based dynamic model," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3570–3578, Sep. 2020.
[30] C. Liu and Y. L. Murphey, "Optimal power management based on Q-learning and neuro-dynamic programming for plug-in hybrid electric vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1942–1954, Jun. 2020.
[31] Q. Zhou, D. Zhao, B. Shuai, Y. Li, H. Williams, and H. Xu, "Knowledge implementation and transfer with an adaptive learning network for real-time power management of the plug-in hybrid vehicle," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5298–5308, Dec. 2021.
[32] F. Zhou, J. Wang, and X. Feng, "Non-crossing quantile regression for distributional reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15909–15919.
[33] N. A. Urpí, S. Curi, and A. Krause, "Risk-averse offline reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–7.
[34] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[35] A. Müller, "Integral probability metrics and their generating classes of functions," *Adv. Appl. Probab.*, vol. 29, no. 2, pp. 429–443, Jun. 1997.
[36] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–10.
[37] R. Koenker and K. Hallock, "Quantile regression," *J. Econ. Perspect.*, vol. 15, no. 4, pp. 143–156, 2001.
[38] D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, "Fully parameterized quantile function for distributional reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 6193–6202.
[39] A. Tversky and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," *J. Risk Uncertainty*, vol. 5, no. 4, pp. 297–323, 1992.
[40] S. S. Wang, "A class of distortion operators for pricing financial and insurance risks," *J. Risk Insurance*, vol. 67, no. 1, pp. 15–36, 2000.
[41] Y. Chow and M. Ghavamzadeh, "Algorithms for CVaR optimization in MDPs," 2014, *arXiv:1406.3339*.
[42] Y. Liu and Y. Wu, "Stepwise multiple quantile regression estimation using non-crossing constraints," *Statist. Interface*, vol. 2, no. 3, pp. 299–310, 2009.
[43] E. L. Lehmann, "Robust estimation in analysis of variance," *Ann. Math. Statist.*, vol. 34, no. 3, pp. 957–966, Sep. 1963.
[44] M. Hessel et al., "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 324–336.
[45] F. Zhou, Z. Zhu, Q. Kuang, and L. Zhang, "Non-decreasing quantile function network with efficient exploration for distributional reinforcement learning," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 3455–3461.
[46] I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum, "Offline reinforcement learning with Fisher divergence critic regularization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5774–5783.
[47] N. Y. Siegel et al., "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–21.
[48] Z. Wang et al., "Critic regularized regression," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–11.
[49] R. Yang, C. Bai, X. Ma, Z. Wang, C. Zhang, and L. Han, "RORL: Robust offline reinforcement learning via conservative smoothing," 2022, *arXiv:2206.02829*.
[50] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, S. Levine, and C. Finn, "Conservative data sharing for multi-task offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1–16.
[51] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 104–114.
[52] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.
[53] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative safety critics for exploration," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–127.
[54] X. Ma, L. Xia, Z. Zhou, J. Yang, and Q. Zhao, "DSAC: Distributional soft actor critic for risk-sensitive reinforcement learning," 2020, *arXiv:2004.14547*.
[55] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 8, 2021, doi: 10.1109/TNNLS.2021.3082568.
[56] Y. Charlie Tang, J. Zhang, and R. Salakhutdinov, "Worst cases policy gradients," 2019, *arXiv:1911.03618*.
[57] L. Chen et al., "Decision transformer: Reinforcement learning via sequence modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.
[58] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[59] T. Yu et al., "MOPO: Model-based offline policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 14129–14142.
[60] R. Keramati, C. Dann, A. Tamkin, and E. Brunskill, "Being optimistic to be conservative: Quickly learning a CVaR policy," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 4436–4443.

Chenjia Bai received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 2015, 2017, and 2022, respectively, all in computer science and technology.

He is currently a Researcher with the Shanghai AI Laboratory, Shanghai, China. His main research interests include reinforcement learning, deep learning networks, and robotics.

Ting Xiao received the master's degree in computer application technology and the Ph.D. degree from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, in 2016 and 2022, respectively.

She is currently a Lecturer with the East China University of Technology, Shanghai, China. Her research interests include computer vision, machine learning, and medical image analysis.

Zhoufan Zhu received the B.S. degree in mathematics and applied mathematics and the M.S. degree in statistics from Shanghai University, Shanghai, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree in statistics with the Shanghai University of Finance and Economics, Shanghai.

His research interests include reinforcement learning and high-dimensional time series analysis.

Lingxiao Wang received the B.S. degree from Nanyang Technological University, Singapore, in 2017. He is currently pursuing the Ph.D. degree with the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA.

His main research interests include reinforcement learning and machine learning.

Fan Zhou received the Ph.D. degree in biostatistics from the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, in 2019.

He is currently an Associate Professor with the School of Statistics and Management, Shanghai University of Finance and Economics, Shanghai, China. His work has been published in top-tier statistical journals and CS conferences, including the *Journal of the American Statistical Association*, *Biometrics*, *Nature Genetics*, Neural Information Processing Systems (NeurIPS), the International Joint Conference on Artificial Intelligence (IJCAI), and the IEEE International Conference on Data Mining (ICDM). His research interests include reinforcement learning, deep learning, spatial-temporal system, and causal inference.

Animesh Garg received the Ph.D. degree in industrial engineering and operations research from the University of California at Berkeley, Berkeley, CA, USA, in 2016.

He was a Post-Doctoral Researcher with the Stanford Artificial Intelligence Laboratory, Stanford, CA, USA. He is currently an Assistant Professor of Computer Science with the University of Toronto, Toronto, ON, USA. His research interests include understanding representations and algorithms to enable the efficiency and generality of learning for interaction in autonomous agents.

Bin He (Member, IEEE) received the B.S. degree in engineering machinery from Jilin University, Changchun, China, in 1996, and the Ph.D. degree in mechanical and electronic control engineering from Zhejiang University, Hangzhou, China, in 2001.

He held a Post-Doctoral Research appointment with the State Key Laboratory of Fluid Power Transmission and Control, Zhejiang University, from 2001 to 2003. He is currently a Professor with the Department of Control Science and Engineering, College of Electronics and Information Engineering, Tongji University, Shanghai, China. His current research interests include intelligent robot control, biomimetic microrobots, and wireless networks.

Peng Liu received the Ph.D. degree in microelectronics and solid-state electronics from the Harbin Institute of Technology (HIT), Harbin, China, in 2007.

He is currently a Professor with the School of Computer Science and Technology, HIT. His research interests include image processing, video analysis, pattern recognition, and the design of large-scale integrated circuits.

Zhaoran Wang received the Ph.D. degree from the Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA, in 2017.

He is currently an Assistant Professor with the Department of Industrial Engineering and Management Sciences and the Department of Electrical Engineering and Computer Science (by courtesy), Northwestern University, Evanston, IL, USA. His research interests include machine learning, optimization, statistics, game theory, and information theory. He has specific interests in making deep reinforcement learning more efficient both computationally and statistically, and scaling deep reinforcement learning to design and optimize societal-scale multiagent systems.