# QUOTA: The Quantile Option Architecture for Reinforcement Learning

**Shangtong Zhang,**[*1] **Hengshuai Yao** [2]
[1] Department of Computing Science, University of Alberta
[2] Reinforcement Learning for Autonomous Driving Lab, Noah's Ark Lab, Huawei
shangtong.zhang@ualberta.ca, hengshuai.yao@huawei.com

## Abstract

In this paper, we propose the Quantile Option Architecture (QUOTA) for exploration based on recent advances in distributional reinforcement learning (RL). In QUOTA, decision making is based on quantiles of a value distribution, not only the mean. QUOTA provides a new dimension for exploration via making use of both *optimism* and *pessimism* of a value distribution. We demonstrate the performance advantage of QUOTA in both challenging video games and physical robot simulators.

## Introduction

Mean of the return has been the center for reinforcement learning (RL) for a long time, and there have been many methods to learn a mean quantity (Sutton 1988; Watkins and Dayan 1992; Mnih et al. 2015). Thanks to the advances in distributional RL (Jaquette 1973; Bellemare, Dabney, and Munos 2017), we are able to learn the full distribution, not only the mean, for a state-action value. Particularly, Dabney et al. (2017) used a set of quantiles to approximate this value distribution. However, the decision making in prevailing distributional RL methods is still based on the mean (Bellemare, Dabney, and Munos 2017; Dabney et al. 2017; Barth-Maron et al. 2018; Qu, Mannor, and Xu 2018). The main motivation of this paper is to answer the questions of how to make decision based on the full distribution and whether an agent can benefit for better exploration. In this paper, we propose the Quantile Option Architecture (QUOTA) for control. In QUOTA, decision making is based on all quantiles, not only the mean, of a state-action value distribution.

In traditional RL and recent distributional RL, an agent selects an action greedily with respect to the mean of the action values. In QUOTA, we propose to select an action greedily w.r.t. certain quantile of the action value distribution. A high quantile represents an *optimistic* estimation of the action value, and action selection based on a high quantile indicates an optimistic exploration strategy. A low quantile represents a *pessimistic* estimation of the action value, and action selection based on a low quantile indicates a pessimistic exploration strategy. (The two exploration strategies are related to risk-sensitive RL, which will be discussed later.) We first compared different exploration strategies in two Markov chains, where naive mean-based RL algorithms fail to explore efficiently as they cannot exploit the distribution information during training, which is crucial for efficient exploration. In the first chain, faster exploration is from a high quantile (i.e., an optimistic exploration strategy). However, in the second chain, exploration benefits from a low quantile (i.e., a pessimistic exploration strategy). Different tasks need different exploration strategies. Even within one task, an agent may still need different exploration strategies at different stages of learning. To address this issue, we use the option framework (Sutton, Precup, and Singh 1999). We learn a high-level policy to decide which quantile to use for action selection. In this way, different quantiles function as different options, and we name this special option the *quantile option*. QUOTA adaptively selects a pessimistic and optimistic exploration strategy, resulting in improved exploration consistently across different tasks.

We make two main contributions in this paper:

- First, we propose QUOTA for control in discrete-action problems, combining distributional RL with options. Action selection in QUOTA is based on certain quantiles instead of the mean of the state-action value distribution, and QUOTA learns a high-level policy to decide which quantile to use for decision making.
- Second, we extend QUOTA to continuous-action problems. In a continuous-action space, applying quantile-based action selection is not straightforward. To address this issue, we introduce *quantile actors*. Each quantile actor is responsible for proposing an action that maximizes one specific quantile of a state-action value distribution.

We show empirically QUOTA improves the exploration of RL agents, resulting in a performance boost in both challenging video games (Atari games) and physical robot simulators (Roboschool tasks)

In the rest of this paper, we first present some preliminaries of RL. We then show two Markov chains where naive mean-based RL algorithms fail to explore efficiently. Then we present QUOTA for both discrete- and continuous-action problems, followed by empirical results. Finally, we give an overview of related work and closing remarks.

[*] Work done during an internship at Huawei

## Preliminaries

We consider a Markov Decision Process (MDP) of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward "function" $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, which we treat as a random variable in this paper, a transition kernel $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and a discount ratio $\gamma \in [0, 1]$. We use $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ to denote a stochastic policy. We use $Z^\pi(s, a)$ to denote the random variable of the sum of the discounted rewards in the future, following the policy $\pi$ and starting from the state $s$ and the action $a$. We have $Z^\pi(s, a) \doteq \sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)$, where $S_0 = s, A_0 = a$ and $S_{t+1} \sim p(\cdot|S_t, A_t), A_t \sim \pi(\cdot|S_t)$. The expectation of the random variable $Z^\pi(s, a)$ is $Q^\pi(s, a) \doteq \mathbb{E}_{\pi,p,R}[Z^\pi(s, a)]$, which is usually called the state-action value function. We have the Bellman equation

$$Q^\pi(s, a) = \mathbb{E}[R(s, a)] + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')}[Q^\pi(s', a')]$$

In a RL problem, we are usually interested in finding an optimal policy $\pi^*$ such that $Q^{\pi^*}(s, a) \geq Q^\pi(s, a) \, \forall \, (\pi, s, a)$. All the possible optimal policies share the same (optimal) state action value function $Q^*$. This $Q^*$ is the unique fixed point of the Bellman optimality operator $\mathcal{T}$(Bellman 2013)

$$\mathcal{T}Q(s, a) \doteq \mathbb{E}[R(s, a)] + \gamma \mathbb{E}_{s' \sim p}[\max_{a'} Q(s', a')]$$

With tabular representation, we can use Q-learning (Watkins and Dayan 1992) to estimate $Q^*$. The incremental update per step is

$$\Delta Q(s_t, a_t) \propto r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \quad (1)$$

where the quadruple $(s_t, a_t, r_{t+1}, s_{t+1})$ is a transition. There are lots of research and algorithms extending Q-learning to linear function approximation (Sutton and Barto 2018; Szepesvári 2010). In this paper, we focus on Q-learning with neural networks. Mnih et al. (2015) proposed Deep-Q-Network (DQN), where a deep convolutional neural network $\theta$ is used to parameterize $Q$. At every time step, DQN performs stochastic gradient descent to minimize

$$\frac{1}{2}(r_{t+1} + \gamma \max_a Q_{\theta^-}(s_{t+1}, a) - Q_\theta(s_t, a_t))^2$$

where the quadruple $(s_t, a_t, r_{t+1}, s_{t+1})$ is a transition sampled from the replay buffer (Lin 1992) and $\theta^-$ is the target network (Mnih et al. 2015), which is a copy of $\theta$ and is synchronized with $\theta$ periodically. To speed up training and reduce the required memory of DQN, Mnih et al. (2016) further proposed the $n$-step asynchronous Q-learning with multiple workers (detailed in Supplementary Material), where the loss function at time step $t$ is

$$\frac{1}{2}(\sum_{i=1}^{n} \gamma^{i-1} r_{t+i} + \gamma^n \max_a Q_{\theta^-}(s_{t+n}, a) - Q_\theta(s_t, a_t))^2$$

### Distributional RL

Analogous to the Bellman equation of $Q^\pi$, Bellemare, Dabney, and Munos (2017) proposed the distributional Bellman equation for the state-action value distribution $Z^\pi$ given a policy $\pi$ in the policy evaluation setting,

$$Z^\pi(s, a) \overset{D}{=} R(s, a) + \gamma Z^\pi(s', a')$$
$$s' \sim p(\cdot|s, a), a' \sim \pi(\cdot|s)$$

where $X \overset{D}{=} Y$ means the two random variables $X$ and $Y$ are distributed according to the same law. Bellemare, Dabney, and Munos (2017) also proposed a distributional Bellman optimality operator for control,

$$\mathcal{T}Z(s, a) \doteq R(s, a) + \gamma Z(s', \arg\max_{a'} \mathbb{E}_{p,R}[Z(s', a')])$$
$$s' \sim p(\cdot|s, a)$$

When making decision, the action selection is still based on the expected state-action value (i.e., $Q$). Since we have the optimality, now we need an representation for $Z$. Dabney et al. (2017) proposed to approximate $Z(s, a)$ by a set of quantiles. The distribution of $Z$ is represented by a uniform mix of $N$ supporting quantiles:

$$Z_\theta(s, a) \doteq \frac{1}{N} \sum_{i=1}^{N} \delta_{q_i(s,a;\theta)}$$

where $\delta_x$ denote a Dirac at $x \in \mathbb{R}$, and each $q_i$ is an estimation of the quantile corresponding to the quantile level (a.k.a. quantile index) $\hat{\tau}_i \doteq \frac{\tau_{i-1} + \tau_i}{2}$ with $\tau_i \doteq \frac{i}{N}$ for $0 \leq i \leq N$. The state-action value $Q(s, a)$ is then approximated by $\frac{1}{N} \sum_{i=1}^{N} q_i(s, a)$. Such approximation of a distribution is referred to as quantile approximation. Those quantile estimations (i.e., $\{q_i\}$) are trained via the Huber quantile regression loss (Huber and others 1964). To be more specific, at time step $t$ the loss is

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{i'=1}^{N} \Big[\rho_{\hat{\tau}_i}^{\kappa}\big(y_{t,i'} - q_i(s_t, a_t)\big)\Big]$$

where $y_{t,i'} \doteq r_t + \gamma q_{i'}\big(s_{t+1}, \arg\max_{a'} \sum_{i=1}^{N} q_i(s_{t+1}, a')\big)$ and $\rho_{\hat{\tau}_i}^{\kappa}(x) \doteq |\hat{\tau}_i - \mathbb{I}\{x < 0\}|\mathcal{L}_\kappa(x)$, where $\mathbb{I}$ is the indicator function and $\mathcal{L}_\kappa$ is the Huber loss,

$$\mathcal{L}_\kappa(x) \doteq \begin{cases} \frac{1}{2}x^2 & \text{if } x \leq \kappa \\ \kappa(|x| - \frac{1}{2}\kappa) & \text{otherwise} \end{cases}$$

The resulting algorithm is the Quantile Regression DQN (QR-DQN). QR-DQN also uses experience replay and target network similar to DQN. Dabney et al. (2017) showed that quantile approximation has better empirical performance than previous categorical approximation (Bellemare, Dabney, and Munos 2017). More recently, Dabney et al. (2018) approximated the distribution by learning a quantile function directly with the Implicit Quantile Network, resulting in further performance boost. Distributional RL has enjoyed great success in various domains (Bellemare, Dabney, and Munos 2017; Dabney et al. 2017; Hessel et al. 2017; Barth-Maron et al. 2018; Dabney et al. 2018).

## Deterministic Policy

Silver et al. (2014) used a deterministic policy $\mu : \mathcal{S} \rightarrow \mathcal{A}$ for continuous control problems with linear function approximation, and Lillicrap et al. (2015) extended it with deep networks, resulting in the Deep Deterministic Policy Gradient (DDPG) algorithm. DDPG is an off-policy algorithm. It has an actor $\mu$ and a critic $Q$, parameterized by $\theta^{\mu}$ and $\theta^{Q}$ respectively. At each time step, $\theta^{Q}$ is updated to minimize

$$\frac{1}{2}(r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1})) - Q(s_t, a_t))^2$$

And the policy gradient for $\theta^{\mu}$ in DDPG is

$$\nabla_a Q(s_t, a)|_{a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s_t)$$

This gradient update is from the chain rule of gradient ascent w.r.t. $Q(s_t, \mu(s_t))$, where $\mu(s_t)$ is interpreted as an approximation to $\arg\max_a Q(s_t, a)$. Silver et al. (2014) provided policy improvement guarantees for this gradient.

## Option

An option (Sutton, Precup, and Singh 1999) is a temporal abstraction of action. Each option $\omega \in \Omega$ is a triple $(\mathcal{I}_\omega, \pi_\omega, \beta_\omega)$, where $\Omega$ is the option set. We use $\mathcal{I}_\omega \subseteq \mathcal{S}$ to denote the initiation set for the option $\omega$, describing where the option $\omega$ can be initiated. We use $\pi_\omega : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ to denote the intra-option policy for $\omega$. Once the agent has committed to the option $\omega$, it chooses an action based on $\pi_\omega$. We use $\beta_\omega : \mathcal{S} \rightarrow [0, 1]$ to denote the termination function for the option $\omega$. At each time step $t$, the option $\omega_{t-1}$ terminates with probability $\beta_{\omega_{t-1}}(S_t)$. In this paper, we consider the call-and-return option execution model (Sutton, Precup, and Singh 1999), where an agent commits to an option $\omega$ until $\omega$ terminates according to $\beta_\omega$.

The option value function $Q_\Omega(s, \omega)$ is used to describe the utility of an option $\omega$ at state $s$, and we can learn this function via Intra-option Q-learning (Sutton, Precup, and Singh 1999). The update is

$$\begin{aligned} &Q_\Omega(s_t, \omega_t) \leftarrow Q_\Omega(s_t, \omega_t) + \\ &\alpha\Big(r_{t+1} + \gamma\big(\beta_{\omega_t}(s_{t+1}) \max_{\omega'} Q_\Omega(s_{t+1}, \omega') \\ &\quad + (1 - \beta_{\omega_t}(s_{t+1}))Q_\Omega(s_{t+1}, \omega_t)\big) - Q_\Omega(s_t, \omega_t)\Big) \end{aligned}$$

where $\alpha$ is a step size and $(s_t, a_t, r_{t+1}, s_{t+1})$ is a transition in the cycle that the agent is committed to the option $\omega_t$.

# A Failure of Mean

We now present two simple Markov chains (Figure 1) to illustrate mean-based RL algorithms can fail to explore efficiently.

Chain 1 has $N$ non-terminal states and two actions $\{$*LEFT*, *UP*$\}$. The agent starts at the state 1 in each episode. The action *UP* will lead to episode ending immediately with reward 0. The action *LEFT* will lead to the next state with a reward sampled from a normal distribution $\mathcal{N}(0, 1)$. Once the agent runs into the $G$ terminal state, the episode will end with a reward $+10$. There is no discounting. The optimal policy is always moving left.

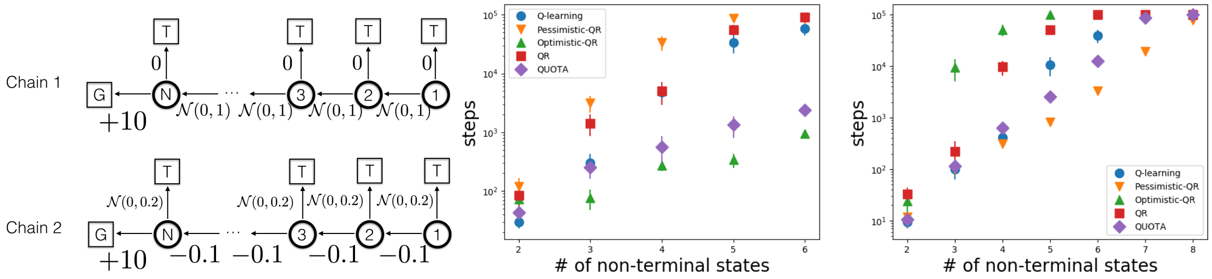
We first consider tabular Q-learning with $\epsilon$-greedy exploration. To learn the optimal policy, the agent has to reach the $G$ state first. Unfortunately, this is particularly difficult for Q-learning. The difficulty comes from two aspects. First, due to the $\epsilon$-greedy mechanism, the agent sometimes selects *UP* by randomness. Then the episode will end immediately, and the agent has to wait for the next episode. Second, before the agent reaches the $G$ state, the expected return of either *LEFT* or *UP* at any state is 0. So the agent cannot distinguish between the two actions under the mean criterion because the expected returns are the same. As a result, the agent cannot benefit from the $Q$ value estimation, a mean, at all.

Suppose now the agent learns the distribution of the returns of *LEFT* and *UP*. Before reaching the state $G$, the learned action-value distribution of *LEFT* is a normal distribution with a mean 0. A high quantile level of this distribution is greater than 0, which is an optimistic estimation. If the agent behaves according to this optimistic estimation, it can quickly reach the state $G$ and find the optimal policy.

Chain 2 has the same state space and action space as Chain 1. However, the reward for *LEFT* is now $-0.1$ except that reaching the $G$ state gives a reward $+10$. The reward for *UP* is sampled from $\mathcal{N}(0, 0.2)$. There is no discounting. When $N$ is small, the optimal policy is still always moving left. Before reaching $G$, the estimation of the expected return of *LEFT* for any non-terminal state is less than 0, which means a Q-learning agent would prefer *UP*. This preference is bad for this chain as it will lead to episode ending immediately, which prevents further exploration.

We now present some experimental results of four algorithms in the two chains: Q-learning, quantile regression Q-learning (QR, the tabular version of QR-DQN), optimistic quantile regression Q-learning (O-QR), and pessimistic quantile regression Q-learning (P-QR). The O-QR / P-QR is the same as QR except that the behavior policy is always derived from a high / low quantile, not the mean, of the state-action value distribution. We used $\epsilon$-greedy behavior policies for all the above algorithms. We measured the steps that each algorithm needs to find the optimal policy. The agent is said to have found the optimal policy at time $t$ if and only if the policy derived from the $Q$ estimation (for Q-learning) or the mean of the $Z$ estimation (for the other algorithms) at time step $t$ is to move left at all non-terminal states.

All the algorithms were implemented with a tabular state representation. $\epsilon$ was fixed to $0.1$. For quantile regression, we used 3 quantiles. We varied the chain length and tracked the steps that an algorithm needed to find the optimal policy. Figure 1b and Figure 1c show the results in Chain 1 and Chain 2 respectively. Figure 1b shows the best algorithm for Chain 1 was O-QR, where a high quantile is used to derive the behavior policy, indicating an optimistic exploration strategy. P-QR performed poorly with the increase of the chain length. Figure 1c shows the best algorithm for Chain 2 was P-QR, where a low quantile is used to derive the behavior policy, indicating a pessimistic exploration strategy. O-QR performed poorly with the increase of the chain length. The mean-based algorithms (Q-learning and QR) performed



(a) (b) Chain 1 (c) Chain 2

Figure 1: (a) two Markov chains illustrating the inefficiency of mean-based decision making. (b)(c) show the required steps to find the optimal policy for each algorithm v.s. the chain length for Chain 1 and Chain 2 respectively. The required steps are averaged over 10 trials, and standard errors are plotted as error bars. For each trial, the maximum step is $10^5$.

poorly in both chains. Although QR did learn the distribution information, it did not use that information properly for exploration.

The results in the two chains show that quantiles influence exploration efficiency, and quantile-based action selection can improve exploration if the quantile is properly chosen. The results also demonstrate that different tasks need different quantiles. No quantile is the best universally. As a result, a high-level policy for quantile selection is necessary.

## The Quantile Option Architecture

We now introduce QUOTA for discrete-action problems. We have $N$ quantile estimations $\{q_i\}_{i=1,\dots,N}$ for quantile levels $\{\hat{\tau}_i\}_{i=1,\dots,N}$. We construct $M$ options $(M < N)$. For simplicity, in this paper all the options share the same initiation set $\mathcal{S}$ and the same termination function $\beta$, which is a constant function. We use $\pi^j$ to indicate the intra-option policy of an option $\omega^j, j = 1, \dots, M$. $\pi^j$ proposes actions based on the mean of the $j$-th window of $K$ quantiles, where $K \doteq N/M$. (We assume $N$ is divisible by $M$ for simplicity.) Here $K$ represents a window size and we have $M$ windows in total. To be more specific, in order to compose the $j$-th option, we first define a state-action value function $Q_{j|K}$ by averaging a local window of $K$ quantiles:

$$Q_{j|K}(s,a) \doteq \frac{1}{K} \sum_{k=(j-1)K+1}^{(j-1)K+K} q_k(s,a)$$

We then define the intra-option policy $\pi^j$ of the $j$-th option $\omega^j$ to be an $\epsilon$-greedy policy with respect to $Q_{j|K}$. Here we compose an option with a window of quantiles, instead of a single quantile, to increase stability. It appears to be a mean form, but it is not the mean of the full state-action value distribution. QUOTA learns the option-value function $Q_\Omega$ via Intra-option Q-learning for option selection. The quantile estimations $\{q_i\}$ is learned via QR-DQN.

To summarize, at each time step $t$, we reselect a new option with probability $\beta$ and continue executing the previous option with probability $1-\beta$. The reselection of the new option is done via a $\epsilon_\Omega$-greedy policy derived from $Q_\Omega$, where $\epsilon_\Omega$ is the random option selection probability. Once the current option $\omega_t$ is determined, we then select an action $a_t$ according to the intra-option policy of $\omega_t$. The pseudo code of QUOTA is provided in Supplementary Material.

### QUOTA for Continuous Control

**QR-DDPG** Quantile Regression DDPG (QR-DDPG, detailed in Supplementary Material) is a new algorithm by modifying DDPG's critic. Instead of learning $Q$, we learn the distribution $Z$ directly in the same manner as the discrete-action controller QR-DQN. Barth-Maron et al. (2018) also learned $Z$ instead of $Q$. However, they parameterized $Z$ through categorical approximation and mixed Gaussian approximation. To our best knowledge, QR-DDPG is the first to parameterize $Z$ with quantile estimations in continuous-action problems. We use QR-DDPG and DDPG as baselines.

**QUOTA** Given the distribution $Z$ of the state-action value approximated by quantiles, the main question now is how to select an action according to certain quantile. For a finite discrete action space, action selection is done according to an $\epsilon$-greedy policy with respect to $Q_{j|K}$, where we need to iterate through all possible actions in the whole action space. To get the action that maximizes a quantile of a distribution in continuous-action problems, we perform gradient ascent for different intra-option policies in analogy to DDPG. We have $M$ options $\{\omega^j\}_{j=1,\dots,M}$. The intra-option policy for the option $\omega^j$ is a deterministic mapping $\mu_j : \mathcal{S} \rightarrow \mathcal{A}$. We train $\mu_j$ to approximate the greedy action $\arg\max_a Q_{j|K}(s,a)$ via gradient ascent. To be more specific, the gradient for $\mu^j$ (parameterized by $\phi$) at time step $t$ is

$$\nabla_a Q_{j|K}(s_t,a)|_{a=\mu^j(s_t)} \nabla_\phi \mu^j(s_t)$$

To compute the update target for the critic $Z$, we also need one more actor $\mu^0$ to maximize the mean of the distribution (i.e., $\frac{1}{N}\sum_{i=1}^N q_i$) as it is impossible to iterate through all the actions in a continuous-action space. Note $\mu^0$ is the same as the actor of QR-DDPG. We augment QUOTA's option set with $\mu^0$, giving $M+1$ options. We name $\mu^j$ the *j-th quantile actor* $(j = 1, \dots M)$. QUOTA for continuous-action problems is detailed in Supplementary Material.

## Experiments

We designed experiments to study whether QUOTA improves exploration and can scale to challenging tasks. All the implementations are made publicly available. [1]

### Does QUOTA improve exploration?

We benchmarked QUOTA for discrete-action problems in the two chains. We used a tabular state representation and three quantiles to approximate $Z$ as previous experiments. Both $\epsilon$ and $\epsilon_\Omega$ were fixed at 0.1, $\beta$ was fixed at 0, which means an option never terminated and lasted for a whole episode. The results are reported in Figures 1b and 1c. QUOTA consistently performed well in both chains.

Although QUOTA did not achieve the best performance in the two chains, it consistently reached comparable performance level with the best algorithm in both chains. Not only the best algorithm in each chain was designed with certain domain knowledge, but also the best algorithm in one chain performed poorly in the other chain. We do not expect QUOTA to achieve the best performance in both chains because it has not used chain-specific knowledge. Those mean-based algorithms (Q-learning and QR) consistently performed poorly in both chains. QUOTA achieved more efficient exploration than Q-learning and QR.

### Can QUOTA scale to challenging tasks?

To verify the scalability of QUOTA, we evaluated QUOTA in both Arcade Learning Environment (ALE) (Bellemare et al. 2013) and Roboschool [2], both of which are general-purpose RL benchmarks.

**Arcade Learning Environment** We evaluated QUOTA in the 49 Atari games from ALE as Mnih et al. (2015). Our baseline algorithm is QR-DQN. We implemented QR-DQN with multiple workers (Mnih et al. 2016; Clemente, Castejón, and Chandra 2017) and an $n$-step return extension (Mnih et al. 2016; Hessel et al. 2017), resulting in reduced wall time and memory consumption compared with an experience-replay-based implementation. We also implemented QUOTA in the same way. Details and more justification for this implementation can be found in Supplementary Material.

We used the same network architecture as Dabney et al. (2017) to process the input pixels. For QUOTA, we added an extra head to produce the option value $Q_\Omega$ after the second last layer of the network. For both QUOTA and QR-DQN, we used 16 synchronous workers, and the rollout length is 5, resulting in a batch size 80. We trained each agent for 40M steps with frameskip 4, resulting in 160M frames in total. We used an RMSProp optimizer with an initial learning rate $10^{-4}$. The discount factor is 0.99. The $\epsilon$ for action selection was linearly decayed from 1.0 to 0.05 in the first 4M training steps and remained 0.05 afterwards. All the hyper-parameter values above were based on an $n$-step Q-learning baseline with synchronous workers from Farquhar et al. (2018), and all our implementations inherited these hyper-parameter values. We used 200 quantiles to approximate the distribution and set the Huber loss parameter $\kappa$ to 1 as used by Dabney et al. (2017). We used 10 options in QUOTA ($M = 10$) , and $\epsilon_\Omega$ was linearly decayed from 1.0 to 0 during the 40M training steps. $\beta$ was fixed at 0.01. We tuned $\beta$ from $\{0, 0.01, 0.1, 1\}$ in the game Freeway. The schedule of $\epsilon_\Omega$ was also tuned in Freeway.

We measured the final performance at the end of training (i.e., the mean episode return of the last 1,000 episodes) and the cumulative rewards during training. The results are reported in Figure 2. In terms of the final performance / cumulative rewards, QUOTA outperformed QR-DQN in 23 / 21 games and underperformed QR-DQN in 14 / 13 games. Here we only considered performance change larger than 3%. Particularly, in the 10 most challenging games (according to the scores of DQN reported in Mnih et al. (2015)), QUOTA achieved a 97.2% cumulative reward improvement in average.

In QUOTA, randomness comes from both $\epsilon$ and $\epsilon_\Omega$. However, in QR-DQN, randomness only comes from $\epsilon$. So QUOTA does have more randomness (i.e., exploration) than QR-DQN when $\epsilon$ is the same. To make it fair, we also implemented an alternative version of QR-DQN, referred to as QR-DQN-Alt, where all the hyper-parameter values were the same except that $\epsilon$ was linearly decayed from 1.0 to 0 during the whole 40M training steps like $\epsilon_\Omega$. In this way, QR-DQN-Alt had a comparable amount of exploration with QUOTA.

We also benchmarked QR-DQN-Alt in the 49 Atari games. In terms of the final performance / cumulative rewards, QUOTA outperformed QR-DQN-Alt in 27 / 42 games and underperformed QR-DQN-Alt in 14 / 5 games, indicating a naive increase of exploration via tuning the $\epsilon$ dose not guarantee a performance boost.

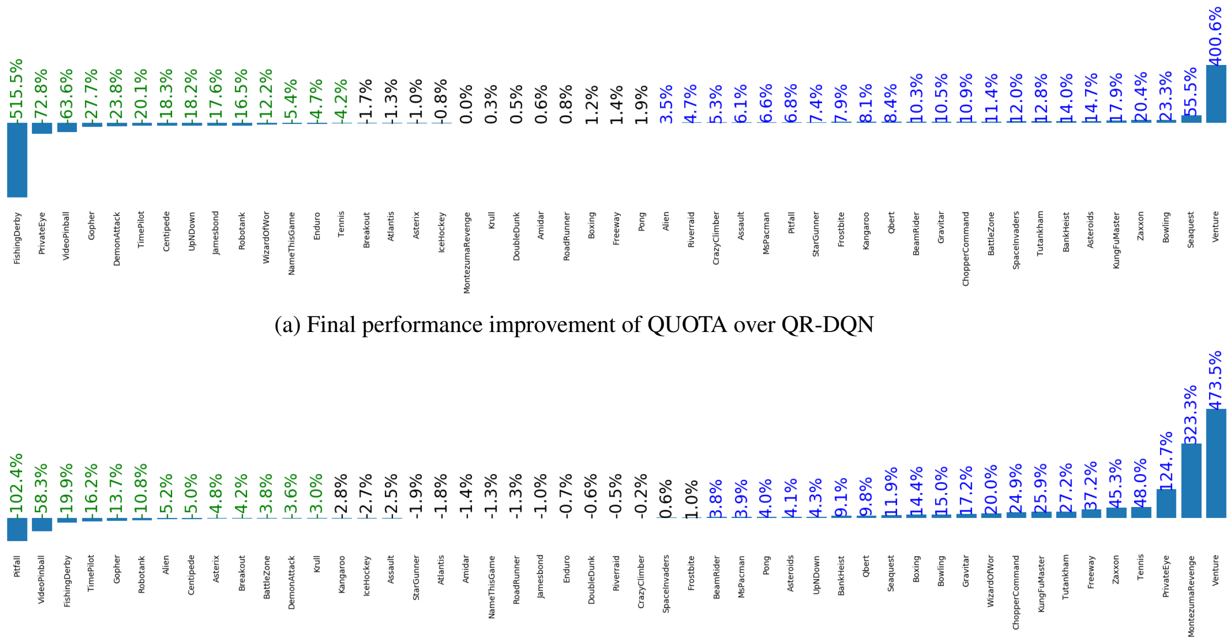
All the original learning curves and scores are reported in Supplementary Material.

**Roboschool** Roboschool is a free port of Mujoco [3] by OpenAI, where a state contains joint information of a robot and an action is a multi-dimensional continuous vector. We consider DDPG and QR-DDPG as our baselines, implemented with experience replay. For DDPG, we used the same hyper-parameter values and exploration noise as Lillicrap et al. (2015), except that we found replacing $ReLU$ and $L_2$ regularizer with $tanh$ brought in a performance boost in our Roboschool tasks. Our implementations of QUOTA and QR-DDPG inherited these hyper-parameter values. For QR-DDPG, we used 20 output units after the second last layer of the critic network to produce 20 quantile estimations for the action value distribution. For QUOTA, we used another two-hidden-layer network with 400 and 300 hidden units to compute $Q_\Omega$. (This network is the same as the actor network in DDPG and QR-DDPG.) We used 6 options in total, including one option that corresponds to the actor maximizing the mean value. $\epsilon_\Omega$ was linearly decayed from 1.0 to 0 in the whole 1M training steps. $\beta$ was fixed at 1.0, which means we reselected an option at every time step. We tuned

[1] https://github.com/ShangtongZhang/DeepRL

[2] https://blog.openai.com/roboschool/

[3] http://www.mujoco.org/



(a) Final performance improvement of QUOTA over QR-DQN

(b) Cumulative rewards improvement of QUOTA over QR-DQN

Figure 2: Improvements in Atari games. Numbers indicate improvement and are computed as $(\text{score}_{\text{QUOTA}} - \text{score}_{\text{QR-DQN}})/|\text{score}_{\text{QR-DQN}}|$, where scores in (a) are the final performance (averaged episode return of the last 1,000 episodes at the end of training), and scores in (b) are cumulative rewards during training. All the scores are averaged across 3 independent runs. Bars above / below horizon indicate performance gain / loss.

$\beta$ from $\{0, 0.01, 0.1, 1\}$ in the game Ant. We trained each algorithm for 1M steps and performed 20 deterministic evaluation episodes every 10K training steps.

The results are reported in Figure 3. QUOTA demonstrated improved performance over both DDPG and QR-DDPG in 5 out of the 6 tasks. For the other six tasks in Roboschool, all the compared algorithms had large variance and are hard to compare. Those results are reported in Supplementary Material.

**Visualization** To understand option selection during training, we plot the frequency of the greedy options according to $Q_\Omega$ in different stages of training in Figure 4. At different training stage, $Q_\Omega$ did propose different quantile options. The quantile option corresponding to mean or median did not dominate the training. In fact, during training, the mean-maximizing options was rarely proposed by the high-level policy. This indicates that the traditional mean-centered action selection (adopted in standard RL and prevailing distributional RL) can be improved by quantile-based exploration.

## Related Work

There have been many methods for exploration in model-free setting based on the idea of *optimism in the face of uncertainty* (Lai and Robbins 1985). Different approaches are used to achieve this optimism, e.g., count-based methods (Auer 2002; Kocsis and Szepesvári 2006; Bellemare et al. 2016; Ostrovski et al. 2017; Tang et al. 2017) and Bayesian methods (Kaufmann, Cappé, and Garivier 2012; Chen et al. 2017; O'Donoghue et al. 2017). Those methods make use of *optimism* of the *parametric uncertainty* (i.e., the uncertainty from estimation). In contrast, QUOTA makes use of both *optimism* and *pessimism* of the *intrinsic uncertainty* (i.e., the uncertainty from the MDP itself). Recently, Moerland, Broekens, and Jonker (2017) combined the two uncertainty via the Double Uncertain Value Network and demonstrated performance boost in simple domains.

There is another line of related work in risk-sensitive RL. Classic risk-sensitive RL is also based on the intrinsic uncertainty, where a utility function (Morgenstern and Von Neumann 1953) is used to distort a value distribution, resulting in risk-averse or risk-seeking policies (Howard and Matheson 1972; Marcus et al. 1997; Chow and Ghavamzadeh 2014). The expectation of the utility-function-distorted valued distribution can be interpreted as a weighted sum of quantiles (Dhaene et al. 2012), meaning that quantile-based action selection implicitly adopts the idea of utility function. Particularly, Morimura et al. (2012) used a small quantile for control, resulting in a safe policy in the cliff grid world. Maddison et al. (2017) employed an exponential utility function over the parametric uncertainty, also resulting in a performance boost in the cliff grid world. Recently, Dabney et al. (2018) proposed various risk-sensitive policies by applying different utility functions to a learned quantile function. The high-level policy $Q_\Omega$ in QUOTA can be interpreted as a special utility function, and the optimistic and pessimistic exploration in QUOTA can be interpreted as risk-sensitive

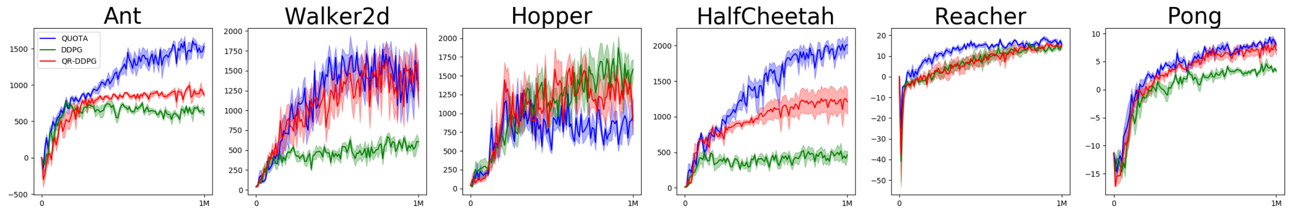


Figure 3: Evaluation curves of Roboschool tasks. The x-axis is training steps, and the y-axis is score. The curves are averaged over 5 independent runs, and standard errors are plotted as shadow.

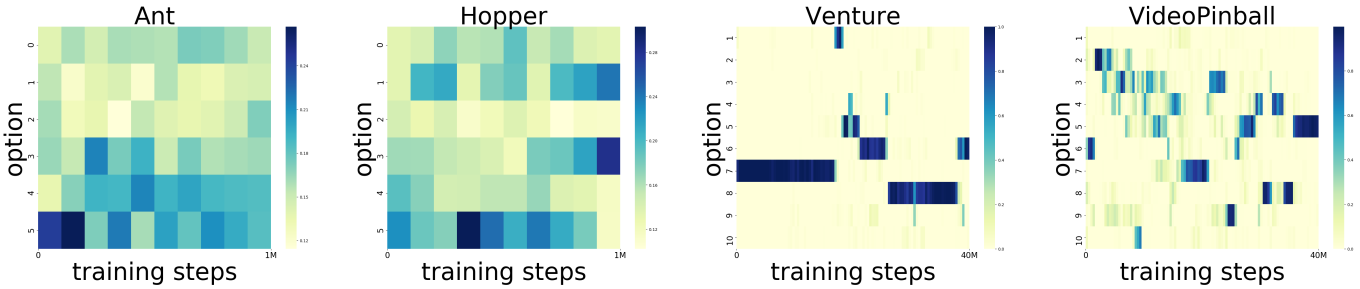


Figure 4: Frequency of the greedy option w.r.t. $Q_\Omega$. The color represents the frequency that an option was proposed by $Q_\Omega$ in different training stages. The frequencies in each column sum to 1. The darker a grid is, the more frequent the option is proposed at that time.

policies. However, the "utility function" $Q_\Omega$ in QUOTA is formalized in the option framework, which is *learnable*. We do not need extra labor to pre-specify a utility function.

Moreover, Tang and Agrawal (2018) combined Bayesian parameter updates with distributional RL for efficient exploration. However, improvements were demonstrated only in simple domains.

## Closing Remarks

QUOTA achieves an on-the-fly decision between optimistic and pessimistic exploration, resulting in improved performance in various domains. QUOTA provides a new dimension for exploration. In this optimism-pessimism dimension, an agent may be able to find an effective exploration strategy more quickly than in the original action space. QUOTA provides an option-level exploration.

At first glance, QUOTA introduces three extra hyper-parameters, i.e., the number of options $M$, the random option probability $\epsilon_\Omega$, and the termination probability $\beta$. For $M$, we simply used 10 and 5 options for Atari games and Roboschool tasks respectively. For $\epsilon_\Omega$, we used a linear schedule decaying from 1.0 to 0 during the whole training steps, which is also a natural choice. We do expect performance improvement if $M$ and $\epsilon_\Omega$ are further tuned. For $\beta$, we tuned it in only two environments and used the same value for all the other environments. Involved labor effort was little. Furthermore, $\beta$ can also be learned directly in an end-to-end training via the termination gradient theorem (Bacon, Harb, and Precup 2017). We leave this for future work.

Bootstrapped-DQN (BDQN, Osband et al. 2016) approximated the distribution of the expected return via a set of $Q$ heads. At the beginning of each episode, BDQN uniformly samples one head and commits to that head during the whole episode. This uniform sampling and episode-long commitment is crucial to the deep exploration of BDQN and inspires us to set $\epsilon_\Omega = 1$ and $\beta = 0$. However, this special configuration only improved performance in certain tasks. Each head in BDQN is an estimation of $Q$ value. All heads are expected to converge to the optimal state action value at the end of training. As a result, a simple uniform selection over the heads in BDQN does not hurt performance. However, in QUOTA, each head (i.e., quantile option) is an estimation of a quantile of $Z$. Not all quantiles are useful for control. A selection among quantile options is necessary. One future work is to combine QUOTA and BDQN or other parametric-uncertainty-based algorithms (e.g., count-based methods) by applying them to each quantile estimation.

## Acknowledgment

Due to a technical mistake in submission, we failed to input the whole author list in this camera-ready version, which should be Shangtong Zhang, Borislav Mavrin, Linglong Kong, Bo Liu, Hengshuai Yao.

## References

Auer, P. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Barth-Maron, G.; Hoffman, M. W.; Budden, D.; Dabney, W.; Horgan, D.; Muldal, A.; Heess, N.; and Lillicrap, T. 2018. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*.

Bellman, R. 2013. *Dynamic programming*. Courier Corporation.

Chen, R. Y.; Sidor, S.; Abbeel, P.; and Schulman, J. 2017. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*.

Chow, Y., and Ghavamzadeh, M. 2014. Algorithms for cvar optimization in mdps. In *Advances in Neural Information Pprocessing Systems*.

Clemente, A. V.; Castejón, H. N.; and Chandra, A. 2017. Efficient parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1705.04862*.

Dabney, W.; Rowland, M.; Bellemare, M. G.; and Munos, R. 2017. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*.

Dabney, W.; Ostrovski, G.; Silver, D.; and Munos, R. 2018. Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*.

Dhaene, J.; Kukush, A.; Linders, D.; and Tang, Q. 2012. Remarks on quantiles and distortion risk measures. *European Actuarial Journal*.

Farquhar, G.; Rocktäschel, T.; Igl, M.; and Whiteson, S. 2018. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. *arXiv preprint arXiv:1710.11417*.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2017. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*.

Howard, R. A., and Matheson, J. E. 1972. Risk-sensitive markov decision processes. *Management Science*.

Huber, P. J., et al. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*.

Jaquette, S. C. 1973. Markov decision processes with a new optimality criterion: Discrete time. *The Annals of Statistics*.

Kaufmann, E.; Cappé, O.; and Garivier, A. 2012. On bayesian upper confidence bounds for bandit problems. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*.

Lai, T. L., and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*.

Maddison, C. J.; Lawson, D.; Tucker, G.; Heess, N.; Doucet, A.; Mnih, A.; and Teh, Y. W. 2017. Particle value functions. *arXiv preprint arXiv:1703.05820*.

Marcus, S. I.; Fernández-Gaucherand, E.; Hernández-Hernandez, D.; Coraluppi, S.; and Fard, P. 1997. Risk sensitive markov decision processes. In *Systems and Control in the Twenty-first Century*. Springer.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*.

Moerland, T. M.; Broekens, J.; and Jonker, C. M. 2017. Efficient exploration with double uncertain value networks. *arXiv preprint arXiv:1711.10789*.

Morgenstern, O., and Von Neumann, J. 1953. *Theory of games and economic behavior*. Princeton university press.

Morimura, T.; Sugiyama, M.; Kashima, H.; Hachiya, H.; and Tanaka, T. 2012. Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*.

O'Donoghue, B.; Osband, I.; Munos, R.; and Mnih, V. 2017. The uncertainty bellman equation and exploration. *arXiv preprint arXiv:1709.05380*.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*.

Ostrovski, G.; Bellemare, M. G.; Oord, A. v. d.; and Munos, R. 2017. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*.

Qu, C.; Mannor, S.; and Xu, H. 2018. Nonlinear distributional gradient temporal-difference learning. *arXiv preprint arXiv:1805.07732*.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction (2nd Edition)*. MIT press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine Learning*.

Szepesvári, C. 2010. *Algorithms for reinforcement learning*. Morgan and Claypool.

Tang, Y., and Agrawal, S. 2018. Exploration by distributional reinforcement learning. *arXiv preprint arXiv:1805.01907*.

Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, O. X.; Duan, Y.; Schulman, J.; DeTurck, F.; and Abbeel, P. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine Learning*.