Value-Distributional Model-Based Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Quantifying uncertainty about a policy's long-term performance is key in sequential decision-making tasks. We study the problem from a Bayesian perspective, where the goal is to learn the posterior distribution over value functions induced by parameter (epistemic) uncertainty of the Markov decision process. Previous work restricts the analysis to a few moments of the distribution over values or imposes a particular distribution shape (e.g., Gaussians). Inspired by distributional reinforcement learning, we introduce a Bellman operator whose fixed-point is the value distribution function. Based on our theory, we propose Epistemic Quantile-Regression (EQR), a model-based algorithm that learns a value distribution function that can be used for policy optimization. Evaluation across several continuous-control tasks shows performance benefits with respect to established model-based and model-free algorithms.

1 Introduction

Reinforcement learning (RL) tackles optimal decision-making in an unknown Markov Decision Process (MDP) (Sutton and Barto, 2018). Uncertainty is at the heart of the RL problem: on one hand, aleatoric uncertainty refers to the stochasticity in the MDP transitions and the RL agent's action selection; on the other hand, *epistemic* uncertainty appears due to lack of knowledge about the MDP. During policy evaluation, both sources of uncertainty induce a distribution of possible returns, which should be considered for policy optimization. For instance, in high-stakes applications like medical treatments, accounting for aleatoric noise is key towards training risk-averse policies (Keramati et al., 2020). Similarly, effective exploration can be achieved by proper handling of epistemic uncertainty (Curi et al., 2020).

Two paradigms have emerged to capture uncertainty in the predicted outcomes of a policy. First, *distributional* RL (Bellemare et al., 2017) models the aleatoric uncertainty about returns, due to the inherent noise of the decision process. In contrast, *Bayesian* RL (Ghavamzadeh et al., 2015) captures the epistemic uncertainty about the unknown *expected* return of a policy — denoted as the *value* function — due to incomplete knowledge of the MDP. As such, the distribution over outcomes from each perspective has fundamentally different meaning and utility. If we care about effective exploration of *unknown* (rather than stochastic) outcomes, then Bayesian RL is the appropriate choice of framework (Osband et al., 2019).

In this paper, we focus on the Bayesian RL setting where a posterior distribution over possible MDPs induces a distribution over value functions. The posterior over values naturally models the epistemic uncertainty about the long-term performance of the agent, which is the guiding principle behind provably-efficient exploration (Strehl and Littman, 2008; Jaksch et al., 2010). An open question remains how to effectively model and learn the posterior distribution over value functions. We provide an answer by using tools from distributional RL in the Bayesian framework. The key idea is that, for

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

time-inhomogeneous MDPs with a tabular representation, the value distribution follows a Bellman equation from which we can derive an iterative estimation algorithm, that resembles methods from distributional RL. Based on this insight, we present a novel algorithm that uses a learned value distribution for policy optimization.

Our contribution. We introduce the value-distributional Bellman equation that describes the relationship between the value distributions over successive steps. Moreover, we show that the fixed-point of the associated Bellman operator is precisely the posterior value distribution. Then, leveraging tools from distributional RL, we propose a practical algorithm for learning the *quantiles* of the value distribution function. Finally, we propose Epistemic Quantile-Regression (EQR), a model-based policy optimization algorithm that learns a distributional critic and can flexibly accomodate any differential objective function of the learned value distribution (mean, exponential risk, CVaR, etc.)

1.1 Related work

Distributional RL. The treatment of the policy return as a random variable dates back to Sobel (1982), where it is shown that the higher moments of the return obeys a Bellman equation. More recently, distributional RL has emerged as a paradigm for modelling and utilizing the entire distribution of returns (Tamar et al., 2013; Bellemare et al., 2023), with real-world applications including guidance of stratospheric balloons (Bellemare et al., 2020) and super-human race-driving in simulation (Wurman et al., 2022). The distributional RL toolbox has expanded over the years with diverse distribution representations (Bellemare et al., 2017; Dabney et al., 2018b,a; Yang et al., 2019) and deeper theoretical understanding (Bellemare et al., 2019; Rowland et al., 2018; Lyle et al., 2019). In our core algorithm, we use quantile-regression (QR) by Dabney et al. (2018b) as a tool for learning the value — rather than return — distribution. Moreover, QR has been integrated with soft actor-critic (SAC) (Haarnoja et al., 2018) for improved performance (Wurman et al., 2022; Kuznetsov et al., 2020). At a high level, EQR combines model learning with quantile-regression, which is then integrated with SAC for policy optimization.

Bayesian RL. Model-free approaches to Bayesian RL directly model the distribution over values, e.g., with normal-gamma priors (Dearden et al., 1998), Gaussian Processes (Engel et al., 2003) or ensembles of neural networks (Osband et al., 2016). Instead, model-based Bayesian RL represents uncertainty in the MDP dynamics, which must then must be propagated to the value function. For instance, the PILCO algorithm by Deisenroth and Rasmussen (2011) learns a Gaussian Process (GP) model of the transition dynamics and integrates over the model's total uncertainty to obtain the expected values. In order to scale to high-dimensional continuous-control problems, Chua et al. (2018) use ensembles of probabilistic neural networks (NNs) to capture both aleatoric and epistemic uncertainty as first proposed by Lakshminarayanan et al. (2017). Both approaches propagate model uncertainty during policy evaluation and improve the policy via greedy exploitation over this model-generated noise.

Closest to our problem setting are approaches that explicitly model the value distribution function or statistics thereof. The uncertainty Bellman equation (UBE) offers a framework to estimate the *variance* of the value distribution (O'Donoghue et al., 2018; Zhou et al., 2020; Luis et al., 2023). Jorge et al. (2020) propose a principled backwards induction framework to estimate value distributions, with the caveat of assuming a Gaussian parameterization for practical implementation. Perhaps closest to our approach is the work by Dearden et al. (1999), which is a local sampling scheme that maintains a sample-based approximation of the value distribution, updated using a Bellman equation. While it does not assume a restrictive parametric form for the distribution, it ignores that samples from the value distribution at successive states are correlated through the Bellman equation; we make a similar approximation in our theory, see Section 3. In our work, rather than generating random samples of the value distribution, we keep track of a relevant set of statistics (Rowland et al., 2019), e.g., evenly spread quantiles, that have adequate coverage and representation power of the underlying distribution.

Mixed Approaches. Recent methods have combined distributional and Bayesian RL methods to capture both sources of uncertainty. The core idea is to train an ensemble of return-distributional critics, where each critic models aleatoric uncertainty, and the ensemble recovers epistemic uncertainty (Moskovitz et al., 2021; Eriksson et al., 2022). Our approach is fundamentally different: we leverage tools from distributional RL to model the epistemic uncertainty around *expected* returns, i.e., we

average over aleatoric noise. Moreover, our experiments show that our value representation with quantiles leads to substantial gains in performance over an ensemble of critics.

2 Background & Notation

In this section, we provide the relevant background and formally introduce the problem of value distribution estimation. We use upper-case letters to denote random variables and lower-case otherwise.

2.1 Markov Decision Processes

We consider an agent that acts in an infinite-horizon MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p, r, \gamma\}$ with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, unknown transition function $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ that maps states and actions to the set of probability distributions over $\mathcal{S}$, a known[1] and bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. The agent is equipped with an action-selection stochastic policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ defining the conditional probability distribution $\pi(a \mid s)$, $(s, a) \in \mathcal{S} \times \mathcal{A}$. Given an initial state $s \in \mathcal{S}$ and some policy π, the RL agent interacts with the environment and generates a random *trajectory* $T = \{S_h, A_h, R_h\}_{h=0}^{\infty}$, with $S_0 = s$ and for $t \geq 0$ we have $A_h \sim \pi(\cdot \mid S_h)$, $R_h = r(S_h, A_h)$, $S_{h+1} \sim p(\cdot \mid S_h, A_h)$.

2.2 Return-Distributional Reinforcement Learning

The *return* of a policy, denoted Z^π, is a random variable defined as the discounted sum of rewards along a trajectory, $Z^\pi(s) = \sum_{h=0}^{\infty} \left[\gamma^h R_h\right]$. The randomness in trajectories and returns originates from the inherent stochasticity of the environment dynamics and the policy, oftentimes called *aleatoric* uncertainty. A common objective for the RL agent is to maximize the *expected* return, where we average over this aleatoric noise to obtain a deterministic function known as the *value*. The value function of policy π under dynamics p, starting from $s \in \mathcal{S}$ is defined as a map $v^{\pi,p} : \mathcal{S} \to \mathbb{R}$ and is given by

$$v^{\pi,p}(s) = \mathbb{E}_T \left[\sum_{h=0}^{\infty} \gamma^h R_h \middle| S_0 = s, p \right], \tag{1}$$

where we explicitly condition on the dynamics p; although redundant in the standard RL setting, this notation will become convenient later on when we consider distribution over dynamics.

In contrast to learning the value function, *return-distributional* RL aims to learn the entire distribution of returns by leveraging the random variable *return-distributional* Bellman equation (Bellemare et al., 2017)

$$Z^\pi(s) \stackrel{D}{=} r(s, A) + \gamma Z^\pi(S'), \tag{2}$$

where $A \sim \pi(\cdot \mid s)$, $S' \sim p(\cdot \mid s, A)$ and $(\stackrel{D}{=})$ denotes equality in distribution, i.e., the random variables in both sides of the equation may have different outcomes, but they share the same probability distribution.

2.3 Bayesian RL

In this paper, we adopt a Bayesian perspective and model the unknown dynamics p as a random transition function P with some prior distribution $\Phi(P)$. As the agent acts in $\mathcal{M}$, it collects data $\mathcal{D}$ and obtains the posterior distribution $\Phi(P \mid \mathcal{D})$ via Bayes' rule. In what follows, we will assume $P \sim \Phi(P \mid \mathcal{D})$ and consider trajectories T defined as previously but with next-states as $S_{h+1} \sim P(\cdot \mid S_h, A_h)$. Notably, the sampling process of next states mixes two sources of uncertainty: the aleatoric noise, as with the original MDP, but also the uncertainty in P due to finite data, often called *epistemic* uncertainty. Consequently, the aleatoric and epistemic noise in trajectories propagates to the returns. We define the *random* value function of policy π under random dynamics P as

$$V^\pi(s) = v^{\pi,P}(s). \tag{3}$$

By (1), V^π is an expectation over the trajectories T *conditioned* on the random variable P, which means the aleatoric noise of trajectories is averaged out, but the epistemic uncertainty (due to the

[1] The theory results can be easily extended to unknown reward functions.

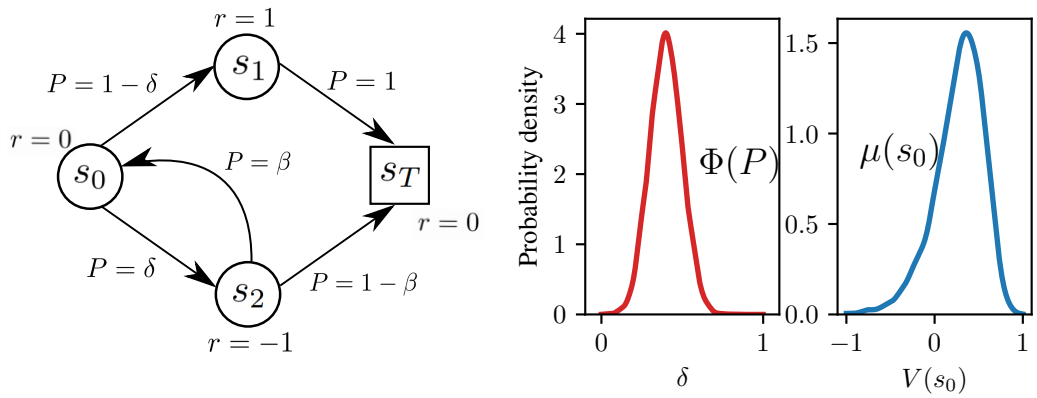

Figure 1: Example value distribution. **(Left)** Uncertain Markov Reward Process with a truncated Gaussian transition probability $\delta \sim \bar{\mathcal{N}}(\mu = 0.4, \sigma = 0.1)$ and a scalar (deterministic) $\beta \in [0, 1]$. For this example, we fixed $\beta = 0.9$. **(Middle)** The prior over MRPs, which corresponds directly to the distribution of δ. **(Right)** The corresponding distribution over values for state s_0.

conditioning on P) remains and is propagated to V^π. Intuitively, to obtain a sample from V^π is equivalent to sample from the posterior $\Phi(P \mid \mathcal{D})$ and calculate the corresponding expected return, i.e., the value. As such, the stochasticity of V^π decreases as we gather data and $\Phi(P \mid \mathcal{D})$ concentrates around the true transition function p.

The main focus of this paper is to study the *value-distribution*[2] function $\mu^\pi : \mathcal{S} \to \mathcal{P}(\mathbb{R})$, such that $V^\pi(s) \sim \mu^\pi(s), \forall s \in \mathcal{S}$. As such, μ^π represents the distribution of the *epistemic* noise around the *expected* return of a policy. Refer to Figure 1 for a simple example of a transition function prior and the value distribution it induces.

Our problem statement stands in clear contrast to the return-distributional RL setting introduced in Section 2.2, which models the *aleatoric* noise around the return of a policy. While both value and return distributions aim to obtain a richer representation of complex random variables, only the former characterizes the type of uncertainty which could be valuable for effective exploration of the environment.

3 The Value-Distributional Bellman Equation

In this section, we establish the theoretical backbone of iterative algorithms for learning the value-distribution function μ^π. We include formal proofs in Appendix A.

For the nominal transition kernel p, we can relate the values at subsequent time steps using the well-known Bellman equation

$$v^{\pi,p}(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{s',a} \pi(a \mid s) p(s' \mid s,a) v^\pi(s'), \tag{4}$$

which holds for any policy π and initial state $s \in \mathcal{S}$. It is straightforward to extend the Bellman-style recursion to account for the distributional nature of the values.

Proposition 1 (Random Variable Value-Distribution Bellman Equation)**.** *Let V^π be the random value function defined in* (3)*. Then, it holds that*

$$V^\pi(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{s',a} \pi(a \mid s) P(s' \mid s,a) V^\pi(s'), \tag{5}$$

for any policy π and initial state $s \in \mathcal{S}$.

[2]We focus on state-value functions for simplicity, but the results have a straightforward extension for state-action-value functions.

Note that the random variable value-distribution Bellman equation (5) differs from the random variable return-distribution Bellman equation (2) in that the former holds in strict equality, while the latter holds in the weaker notion of equality in distribution. The main caveat of (5) with respect to model-free distributional RL is that, in general, $P(s' \mid s, a)$ and $V^\pi(s')$ are correlated.

We now shift from discussing the random value function to focus instead on the value distribution function. The main challenge in establishing a recursion for learning μ^π is the dependency between $P(s' \mid s, a)$ and $V^\pi(s')$ in (5). We side-step the issue by restricting our study to a family of MDPs under which these random variables are independent, similar to previous work (O'Donoghue et al., 2018; Luis et al., 2023). All the results that follow in this section hold under:

Assumption 1 (Independent transitions)**.** *$P(s' \mid x, a)$ and $P(s' \mid y, a)$ are independent random variables if $x \neq y$.*

Assumption 2 (Acyclic MDP (O'Donoghue et al., 2018))**.** *For any realization $\tilde{p}$ of the random variable P, the MDP $\tilde{\mathcal{M}}$ with transition function $\tilde{p}$ is a directed acyclic graph, i.e., states are not visited more than once in any given episode.*

Assumption 1 holds naturally in the case of discrete state-action spaces with a tabular transition function, where there is no generalization. Assumption 2 is non-restrictive as any finite-horizon MDP with cycles can be transformed into an equivalent time-inhomogeneous MDP without cycles by adding a time-step variable h to the state-space. Since the state-space is finite-dimensional, for infinite-horizon problems we consider the existence of a terminal (absorbing) state which is reached in finite steps.

To establish a Bellman-style recursion defining the value distribution function, we will leverage a notion of pushforward distributions akin to Rowland et al. (2018). First, recall the matrix-vector notation of the standard Bellman equation: $v^{\pi,p} = r^\pi + \gamma p^\pi v^{\pi,p}$, where $r^\pi \in \mathbb{R}^{\mathcal{S}}$, $v^{\pi,p} \in \mathbb{R}^{\mathcal{S}}$ are vectors and $p^\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}_{[0,1]}$ is a so-called stochastic matrix whose entries are restricted to $[0, 1]$ and whose rows sum up to 1, i.e., they represent a discrete probability distribution. Then, we define a bootstrap function applied to value vectors:

$$b_{r,p,\gamma} : v \to r + \gamma p v, \tag{6}$$

for an arbitrary $r \in \mathbb{R}^{\mathcal{S}}$, $p \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}_{[0,1]}$ and $\gamma \in [0, 1)$. Applying $b_{r,p,\gamma}$ is a combination of adding r to a γ-scaled linear transformation of the original vector. Now, given a random vector $V \sim \nu$, we define the pushforward distribution $(b_{r,p,\gamma})_\# \nu$ to be the distribution resulting from applying the bootstrap function to every possible realization of V. Based on this pushforward operation, we can now propose a Bellman equation for the value distribution function μ^π.

Lemma 1 (Value-Distribution Bellman Equation)**.** *The value distribution function μ^π obeys the Bellman equation.*

$$\mu^\pi = \mathbb{E}_P\left[(b_{r^\pi,P^\pi,\gamma})_\# \mu^\pi\right] \tag{7}$$

for any policy π.

Lemma 1 provides the theoretical backbone towards designing an iterative algorithm for learning the value distribution function. From (7) we can extract an operator that acts on arbitrary value distribution functions.

Definition 1. The value-distributional Bellman operator $\mathcal{T}^\pi : \mathcal{P}(\mathbb{R})^{\mathcal{S}} \to \mathcal{P}(\mathbb{R})^{\mathcal{S}}$ is defined by

$$\mathcal{T}^\pi \mu = \mathbb{E}_P\left[(b_{r^\pi,P^\pi,\gamma})_\# \mu\right] \tag{8}$$

Intuitively, the operator $\mathcal{T}^\pi$ corresponds to mixing pushforward distributions, where the pushforward itself involves shifting, scaling and linearly transforming the probability mass. The natural question that follows is whether we can establish convergence to μ^π by repeated applications of $\mathcal{T}^\pi$ starting from an arbitrary initial guess μ_0.

Our convergence result is an adaptation of the standard distributional RL analysis done in Bellemare et al. (2023). We adopt the supremum p-Wasserstein distance $\bar{w}_p$ to establish contractivity of the operator $\mathcal{T}^\pi$ (see Definition 2 in Appendix A).

Lemma 2. *The operator $\mathcal{T}^\pi$ is a γ-contraction with respect to $\bar{w}_p$ for all $p \in [1, \infty)$. That is, $\bar{w}_p(\mathcal{T}^\pi \mu, \mathcal{T}^\pi \mu') \leq \gamma \bar{w}_p(\mu, \mu')$ for all $\mu, \mu' \in \mathcal{P}(\mathbb{R})^{\mathcal{S}}$.*

Lemma 2 parallels similar results in standard RL and model-free distributional RL, in that it allows us to establish the convergence of iterated applications of $\mathcal{T}^\pi$ and characterize the operator's fixed-point.

Theorem 1. *Denote the space of value distribution functions with bounded support by $\mathcal{P}_B(\mathbb{R})^\mathcal{S}$. Given an arbitrary value distribution function $\mu_0 \in \mathcal{P}_B(\mathbb{R})^\mathcal{S}$, the sequence $\{\mu_k\}_{k=0}^\infty$ defined by $\mu_{k+1} = \mathcal{T}^\pi \mu_k$ for all $k \geq 0$ is such that $\bar{w}_p(\mu_k, \mu^\pi) \leq \gamma^k \bar{w}_p(\mu_0, \mu^\pi) \to 0$ as $k \to \infty$ for $p \in [1, \infty)$. That is, μ^π is the unique fixed-point of the value-distribution Bellman operator $\mathcal{T}^\pi$.*

Proof. From the contractivity of $\mathcal{T}^\pi$ (Lemma 2) and the fact that the operator maps $\mathcal{P}_B(\mathbb{R})^\mathcal{S}$ onto itself, then $\mathcal{T}^\pi$ is a *contraction mapping*, and by Banach's fixed-point theorem $\mathcal{T}^\pi$ admits a unique fixed-point which is the limiting value of the sequence $\{\mu_k\}_{k=0}^\infty$. Since $\mu^\pi = \mathcal{T}^\pi \mu^\pi$ holds by Lemma 1, then μ^π must be the unique fixed-point $\mathcal{T}^\pi$. □

In summary, Theorem 1 establishes that repeated applications of $\mathcal{T}^\pi$ from an arbitrary initial guess converges to the value distribution function μ^π. Inspired by this theoretical result, in the remaining sections we introduce and evaluate a practical algorithm for learning the value-distribution function.

4 Quantile-Regression for Value-Distribution Learning

In the previous section we described an iterative process that converges to μ^π starting from an arbitrary (bounded) value distribution. In practice, however, to implement such a recursion we must project the value distributions onto some finite-dimensional parameter space. Following the success of quantile distributional RL (Dabney et al., 2018b), we adopt the quantile parameterization. Let $\mathcal{V}_M$ be the space of quantile distributions with fixed M quantiles corresponding to quantile levels $\tau_i = (2i-1)/2M$. Define a parametric model $\theta : \mathcal{S} \to \mathbb{R}^M$, then the quantile distribution $\mu_\theta \in \mathcal{V}_M$ maps states to a uniform probability distribution supported on $\theta_i(s)$. That is, $\mu_\theta(s) = \sum_{i=1}^M \frac{1}{M} \delta\big(\theta_i(s)\big)$ given by a uniform mixture of Dirac deltas, where the particle $\theta_i(s)$ corresponds to the τ_i-quantile. With this parameterization, our aim now becomes to compute the so-called quantile projection of μ^π onto $\mathcal{V}_M$, given by

$$\Pi_{w_1} \mu^\pi := \operatorname*{argmin}_{\mu_\theta \in \mathcal{V}_M} w_1(\mu^\pi, \mu_\theta), \tag{9}$$

where w_1 is the 1-Wasserstein distance.

In order to solve (9), we follow closely the treatment by Rowland et al. (2023) of quantile-regression temporal-difference learning for return-distributions, and adapt it to instead work on value-distributions. The following loss function corresponds to the quantile-regression problem of estimating the quantile τ of the value distribution μ^π:

$$\mathcal{L}_s^{\tau,\pi}(v) = \mathbb{E}_P\Big[\big(\tau \mathbb{1}\{V^\pi(s) > v\} + (1-\tau)\mathbb{1}\{V^\pi(s) < v\}\big)\big|V^\pi(s) - v\big|\Big]. \tag{10}$$

It is an asymmetric convex loss function whose minimizer is the τ-quantile of μ^π, where quantile overestimation and underestimation errors are weighted by τ and $1-\tau$, respectively.

Our goal is to propose a practical algorithm to learn the value distribution function based on the quantile-regression loss (10). If we have access to samples of $V^\pi(s)$, denoted $\tilde{v}^\pi(s)$, then we can derive an unbiased estimate of the negative gradient of (10) and obtain the update rule

$$\theta_i(s) \leftarrow \theta_i(s) + \alpha\big(\tau_i - \mathbb{1}\{\tilde{v}^\pi(s) < \theta_i(s)\}\big), \tag{11}$$

where α is some scalar step size. One option to sample $V^\pi = \tilde{v}^\pi$ would be to first sample a model $P = \tilde{p}$ and then solve the corresponding Bellman equation. Instead, we use a computationally cheaper alternative and bootstrap like in temporal-difference learning, so that the samples are defined as

$$\tilde{v}^\pi(s) = r^\pi(s) + \gamma \sum_{s'} \tilde{p}^\pi(s' \mid s)\theta_J(s'), \tag{12}$$

where $J \sim \text{Uniform}(1, \ldots, M)$. Lastly, we reduce the variance of the gradient estimate by averaging over the values of J, which results in the update

$$\theta_i(s) \leftarrow \theta_i(s) + \frac{\alpha}{M} \sum_{j=1}^M \left(\tau_i - \mathbb{1}\left\{ r^\pi(s) + \gamma \sum_{s'} \tilde{p}^\pi(s' \mid s)\theta_j(s') < \theta_i(s) \right\} \right). \tag{13}$$

Algorithm 1 Epistemic Quantile-Regression (EQR)

Input: Posterior MDP Φ, policy π, number of quantiles M.
Randomly initialize estimates $\{\theta_i(s)\}_{i=1}^{M}$ for all $s \in \mathcal{S}$
repeat
  Sample $\tilde{p}$ from posterior Φ
  for $i = 1, \ldots, M$ **do**
    Update $\theta_i(s)$ with (13) for all $s \in \mathcal{S}$
until convergence
return $\{\theta_i(s)\}_{i=1}^{M}$

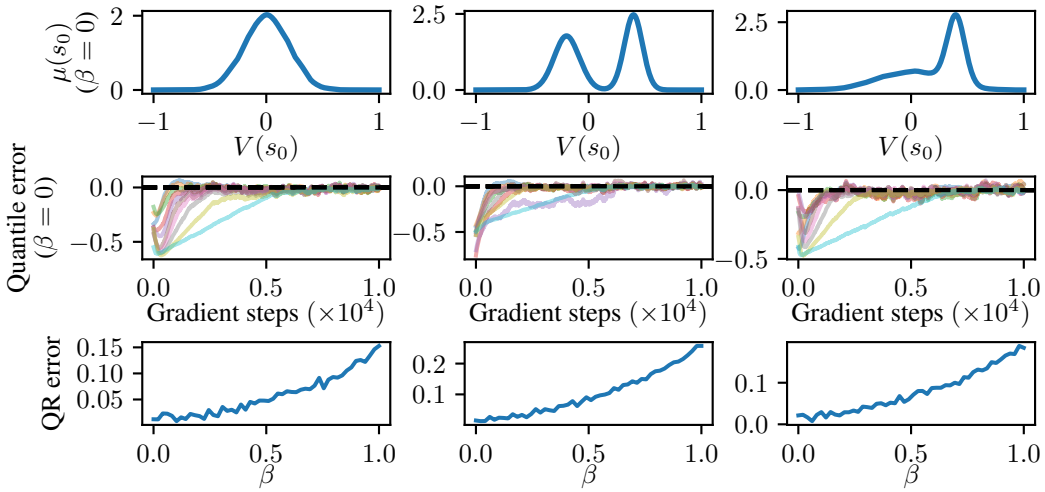

Figure 2: Performance of quantile-regression for value-distribution learning in the example MDP of Figure 1. The parameter β controls the covariance between $V(s_0)$ and $P(s_2|s_0)$; the covariance increases with β and is zero for $\beta = 0$. **(Top)** Value distributions (Gaussian, bi-modal and heavy-tailed) generated by different prior distributions of the parameter δ. **(Middle)** Evolution of the per-quantile estimation error between the true quantile projection $\Pi_{w_1}\mu(s_0)$ and the estimate $\mu_\theta(s_0)$; for $\beta = 0$ our algorithm oscillates around the true quantile projection. **(Bottom)** 1-Wasserstein metric between the true quantile projection and the estimate μ_θ after 10^4 gradient steps, as a function of the correlation parameter β. As β moves from zero to one, the regression error increases and the algorithm no longer converges to the true quantiles, although the error is relatively small.

We introduce EQR in Algorithm 1 to iteratively learn the value distribution function μ^π. From an arbitrary initial guess of quantiles, we sample an MDP from the posterior and update the quantiles using (13) for all states until convergence.

We validate our theoretical results by running Algorithm 1 in the simple MDP of Figure 1 for diverse posterior distributions. The results are summarized in Figure 2: when the assumptions hold ($\beta = 0$), we validate the algorithm converges to the true quantile projection of the value distribution. As we move β from zero to one, the covariance between $V(s_0)$ and $P(s_2|s_0)$ — a term ignored by our theory — increases, which induces an approximation error in EQR.

5 Policy Optimization with Value Distributions

In this section we combine EQR with SAC (EQRSAC) to obtain a model-based reinforcement learning algorithm that leverages a value-distributional critic for policy optimization. The key ingredients of our method are: (1) an ensemble-based posterior over MDPs, (2) a quantile-distributional critic network that models the M-quantile function $\theta(s, a)$ and (3) a policy network π_ϕ trained to optimize some differentiable function of the critic. A full algorithmic description of EQRSAC is included in Appendix B.

Posterior Dynamics. We adopt the baseline architecture from MBPO (Janner et al., 2019) and the implementation from Pineda et al. (2021), where the posterior MDP, denoted Γ_ψ, is represented as an ensemble of N neural networks trained via supervised learning on the environment dataset $\mathcal{D}$ to predict the mean and variance of a Gaussian distribution over next states and rewards. We use Γ_ψ to populate an experience replay buffer $\mathcal{D}_{\text{model}}$ with model-consistent k-step rollouts; that is, we use a consistent ensemble member throughout a rollout, rather than randomizing the model per-step like in MBPO. We believe that model-consistent rollouts are important to disentangle aleatoric and epistemic uncertainties, as argued by Chua et al. (2018).

Critic. We train the critic on mini-batches drawn from $\mathcal{D}_{\text{model}}$, using the entropy regularization loss from SAC with temperature α and replacing the quantile regression loss with the quantile Huber loss $\rho_\tau(u)$ from Dabney et al. (2018b) (see Appendix B.1)

$$\mathcal{L}_{\text{critic}}(\theta) = \mathbb{E}_{(S,A)\sim\mathcal{D}_{\text{model}}}\left[\mathbb{E}_{(\hat{R},\hat{P})\sim\Gamma_\psi}\left[\sum_{i=1}^{M}\mathbb{E}_J\left[\rho_{\tau_i}\big(\mathcal{T}\theta_J(S,A)-\theta_i(S,A)\big)\right]\right]\right], \tag{14}$$

where the target quantiles $\mathcal{T}\theta_j$ are defined as

$$\mathcal{T}\theta_j(s,a) = \hat{R}(s,a) + \gamma\,\mathbb{E}_{(S',A')\sim\hat{P}(\cdot|s,a),\pi_\phi}\big[\theta_j(S',A') - \alpha\log\pi_\phi(A' \mid S')\big]. \tag{15}$$

The expectation in (15) is approximated by generating transition tuples (s', a') using the policy and the sampled dynamics from Γ_ψ. Typically, model-based algorithms like MBPO only use data in the mini-batch to compose critic targets, rather than leveraging the learned dynamics model for better approximation of expectations.

Actor. The policy is trained to maximize some differentiable utility function f of the quantile critic, in addition to the entropy regularization term from SAC,

$$\mathcal{L}_{\text{actor}}(\phi) = \mathbb{E}_{S\sim\mathcal{D}_{\text{model}}}\Big[\mathbb{E}_{A\sim\pi_\phi}\big[f(\theta(S,A)) - \alpha\log\pi_\phi(A \mid S)\big]\Big]. \tag{16}$$

Let $\bar{\theta}(s,a)$ and $\sigma_\theta(s,a)$ be the mean and standard deviations of the quantiles, respectively. Then, we consider two concrete utility functions: the classical mean objective $f_{\text{mean}}(\theta(s,a)) = \bar{\theta}(s,a)$ and an objective based on optimism in the face of uncertainty $f_{\text{ofu}} = \bar{\theta}(s,a) + \sigma_\theta(s,a)$.

6 Experiments

In this section we evaluate EQRSAC in environments with continuous state-action spaces. Implementation details and hyperparameters are included in Appendix B.

6.1 Baselines

We consider the following baselines, which all share a common codebase and hyperparameters: SAC (Haarnoja et al., 2018), MBPO (Janner et al., 2019), Quantile-regression MBPO (QRMBPO), which replaces the standard critic with a distributional one; and Q-uncertainty SAC (QUSAC) from Luis et al. (2023), which trains an ensemble of critics to model the posterior over values and optimizes its policy using the mean prediction of the ensemble. More details about these baselines are provided in Appendix B.3. For EQRSAC we consider two variants: EQRSAC-mean, which uses f_{mean} and EQRSAC-OFU, which uses f_{ofu}.

6.2 DM Control Benchmark

We conduct an experiment in a subset of 14 continuous-control tasks from the DeepMind Control Suite (Tunyasuvunakool et al., 2020). In Figure 3, we plot the results for four environments ranging from small (cartpole) to mid/large (quadruped) observation spaces. Our method significantly improves performance over previous model-based algorithms in these environments. Moreover, in the full benchmark, EQRSAC achieves the best (or comparable) final performance in 11 out of 14 tasks (see Appendix E). Table 1 shows the mean/median scores across all tasks after 250 episodes of training, which demonstrates the performance gain achieved by our algorithm. Training runs and final scores for all environments are included in Appendices D and E, respectively.

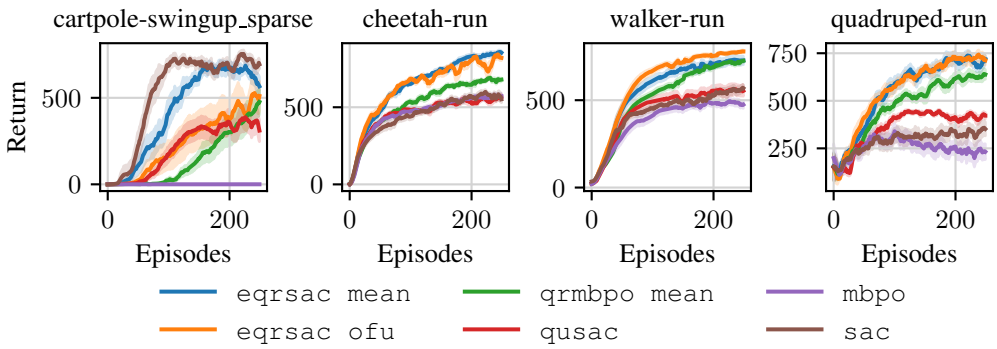

Figure 3: Performance in four DeepMind Control tasks. The returns are smoothened by a moving average filter and we report the mean and standard error over 10 random seeds. EQRSAC significantly improves performance with respect to the model-based baselines. In the full benchmark, EQRSAC achieves the best or comparable final performance in 11 out of 14 tasks and the best mean/median scores (see Table 1).

Table 1: Mean and median scores in the DM control benchmark after 250 episodes (or 250K environment steps). We report the mean and standard error over 10 random seeds.

	`sac`	`mbpo`	`qrmbpo`	`qusac`	`eqrsac mean`	`eqrsac ofu`
mean	556 ± 16	333 ± 26	543 ± 8	473 ± 6	$\mathbf{605 \pm 13}$	$\mathbf{624 \pm 12}$
median	596 ± 26	406 ± 7	659 ± 13	459 ± 14	$\mathbf{750 \pm 13}$	$\mathbf{748 \pm 11}$

6.3 Discussion

We observe a clear gap in performance between MBPO and QRMBPO, which supports the observations from Lyle et al. (2019) and reinforces their hypothesis that distributional critics boost performance under non-linear function approximation.

The gap between QUSAC and the distributional methods (QRMBPO / EQRSAC) indicates that the quantile representation of values leads to more sample-efficient learning compared to the ensemble-based approach. Moreover, training one distributional critic is typically less computationally intensive than training an ensemble of standard critics.

Most interestingly, the performance difference between QRMBPO and EQRSAC suggests that our loss function (14) is instrumental towards sample-efficient learning, especially in environments with sparse rewards like cartpole swing-up (see also fish-swim and finger-spin in Appendix D). As such, our theory provides a solid guideline on how to integrate model-based RL architectures with distributional RL tools, which goes beyond simply using a distributional critic with established algorithms like MBPO.

7 Conclusions

We investigated the problem of estimating the distribution of values, given parameter uncertainty of the MDP. We proposed the value-distributional Bellman equation and extracted an operator whose fixed-point is precisely the distribution of values. Leveraging tools from return-distributional RL, we designed Epistemic Quantile-Regression, an iterative procedure for estimating quantiles of the value distribution. We applied our algorithm in a simple MDP and validated the convergence properties prescribed by our theory and assessed its limitations once the main assumptions are violated. Lastly, we scaled-up EQR using neural networks and combine it with SAC for policy optimization, which resulted in a novel model-based deep RL algorithm that trains a critic to predict quantiles of the value distribution and optimizes the actor on any differentiable function of the learned quantiles. Our approach was benchmarked in several continuous-control tasks from the DeepMind Control suite and showed improved sample-efficiency and final performance compared to several baselines.

References

Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. In *International Conference on Machine Learning*, volume 70, pages 449–458. PMLR, August 2017.

Marc G. Bellemare, Nicolas Le Roux, Pablo Samuel Castro, and Subhodeep Moitra. Distributional Reinforcement Learning with Linear Function Approximation. In *International Conference on Artificial Intelligence and Statistics*, volume 89, pages 2203–2211. PMLR, April 2019.

Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning. In *Advances in Neural Information Processing Systems*, volume 33, pages 14156–14170. Curran Associates, Inc., 2020.

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit Quantile Networks for Distributional Reinforcement Learning. In *International Conference on Machine Learning*, volume 80, pages 1096–1105. PMLR, July 2018a.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional Reinforcement Learning with Quantile Regression. In *AAAI Conference on Artificial Intelligence*, pages 2892–2901. AAAI Press, 2018b.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-Learning. In *AAAI Conference on Artificial Intelligence*, pages 761–768, 1998. American Association for Artificial Intelligence.

Richard Dearden, Nir Friedman, and David Andre. Model Based Bayesian Exploration. In *Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999. Morgan Kaufmann Publishers Inc.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *International Conference on Machine Learning*, pages 465–472, 2011.

Yaakov Engel, Shie Mannor, and Ron Meir. Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning. In *International Conference on Machine Learning*, pages 154–161. AAAI Press, 2003.

Hannes Eriksson, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis. SENTINEL: taming uncertainty with ensemble based distributional reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, volume 180, pages 631–640. PMLR, August 2022.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, volume 80, pages 1587–1596. PMLR, July 2018.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, volume 80, pages 1861–1870. PMLR, July 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications, January 2019.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research*, 11(4), 2010.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Emilio Jorge, Hannes Eriksson, Christos Dimitrakakis, Debabrota Basu, and Divya Grover. Inferential Induction: A Novel Framework for Bayesian Reinforcement Learning. In *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, volume 137, pages 43–52. PMLR, December 2020.

Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. Being Optimistic to Be Conservative: Quickly Learning a CVaR Policy. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 4436–4443, April 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. In *International Conference on Machine Learning*, volume 119, pages 5556–5566. PMLR, July 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Carlos E. Luis, Alessandro G. Bottero, Julia Vinogradska, Felix Berkenkamp, and Jan Peters. Model-Based Uncertainty in Value Functions. In *International Conference on Artificial Intelligence and Statistics*, volume 206, pages 8029–8052. PMLR, April 2023.

Clare Lyle, Marc G. Bellemare, and Pablo Samuel Castro. A Comparative Analysis of Expected and Distributional Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 4504–4511, July 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*, December 2013.

Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical Optimism and Pessimism for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 12849–12863. Curran Associates, Inc., 2021.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The Uncertainty Bellman Equation and Exploration. In *International Conference on Machine Learning*, pages 3836–3845, 2018.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research*, 20:1–62, 2019.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. MBRL-Lib: A Modular Library for Model-based Reinforcement Learning. *arXiv:2104.10159 [cs, eess]*, April 2021.

Mark Rowland, Marc Bellemare, Will Dabney, Remi Munos, and Yee Whye Teh. An Analysis of Categorical Distributional Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pages 29–37. PMLR, April 2018.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Remi Munos, Marc G. Bellemare, and Will Dabney. Statistics and Samples in Distributional Reinforcement Learning. In *International Conference on Machine Learning*, volume 97, pages 5528–5536. PMLR, June 2019.

Mark Rowland, Rémi Munos, Mohammad Gheshlaghi Azar, Yunhao Tang, Georg Ostrovski, Anna Harutyunyan, Karl Tuyls, Marc G. Bellemare, and Will Dabney. An Analysis of Quantile Temporal-Difference Learning, January 2023.

Matthew J Sobel. The Variance of Discounted Markov Decision Processes. *Journal of Applied Probability*, 19(4):794–802, 1982.

Alexander L Strehl and Michael L Littman. An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*, volume 7. MIT Press, 2018.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Temporal Difference Methods for the Variance of the Reward To Go. In *International Conference on Machine Learning*, pages 495–503. PMLR, 2013.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, February 2022.

Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Qi Zhou, HouQiang Li, and Jie Wang. Deep Model-Based Reinforcement Learning via Estimated Uncertainty and Conservative Policy Optimization. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 6941–6948, April 2020.

A Theory Proofs

Proposition 1 (Random Variable Value-Distribution Bellman Equation)**.** *Let V^π be the random value function defined in* (3)*. Then, it holds that*

$$V^\pi(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{s',a} \pi(a \mid s) P(s' \mid s,a) V^\pi(s'), \tag{5}$$

for any policy π and initial state $s \in \mathcal{S}$.

Proof. We proceed similarly as the standard Bellman equation proof shown by Bellemare et al. (2023). First, the random trajectories $\tilde{T}$ have two properties endowed by the Markov decision process: time homogeneity and the Markov property. Informally speaking, time homogeneity states that the trajectory from a given state s is independent of the time k at which the state is visited, while the Markov property states that trajectories starting from s are independent of states, actions or rewards encountered before s (c.f. Bellemare et al. (2023) Lemmas 2.13, 2.14 for a formal definition). In the domain of random variables, these properties imply that two trajectories starting from the same initial state s are equally distributed regardless of past history.

From the definition (3) we decompose the random value into the immediate reward and the value at the next state:

$$V^\pi(s) = \mathbb{E}_{\tilde{T}}[R_0 | S_0 = s,\, P] + \gamma\, \mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_0 = s,\, P\right]. \tag{17}$$

For the first term, the only random variable remaining is A_0, so we rewrite it as

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_0 = s,\, P\right]. \tag{18}$$

For the second term, we apply the tower property of expectations

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\left[\mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_0 = s,\, A_0,\, S_1,\, P\right] \middle| S_0 = s,\, P\right]. \tag{19}$$

By the Markov property,

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\left[\mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_1,\, P\right] \middle| S_0 = s,\, P\right]. \tag{20}$$

By time homogeneity, the inner expectation is exactly equal to the random variable $V^\pi(S_1)$, after a change of variable in the infinite sum index

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\big[V^\pi(S_1) \big| S_0 = s,\, P\big]. \tag{21}$$

Lastly, the remaining random variable is S_1, for which we can explicitly write its probability distribution, concluding the proof

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) V^\pi(s'). \tag{22}$$

□

Lemma 1 (Value-Distribution Bellman Equation)**.** *The value distribution function μ^π obeys the Bellman equation.*

$$\mu^\pi = \mathbb{E}_P\big[(b_{r^\pi, P^\pi, \gamma})_\# \mu^\pi\big] \tag{7}$$

for any policy π.

Proof. In matrix-vector format, the random-variable value-distributional Bellman equation is expressed as

$$V^\pi = r^\pi + \gamma P^\pi V^\pi. \tag{23}$$

Let $\mathfrak{D}_P(\cdot)$ be a function that extracts the probability distribution of a random variable belonging to the probability space of P. Then, Applying $\mathfrak{D}_P(\cdot)$ to both sides we get

$$\mu^\pi = \mathfrak{D}_P(r^\pi + \gamma P^\pi V^\pi). \tag{24}$$

Now we use the fact that any marginal probability can be written as the expected value of a conditional probability. In our case, by conditioning on the transition function, we obtain:

$$\mu^\pi = \mathbb{E}_P\big[\mathfrak{D}_P(r^\pi + \gamma P^\pi V^\pi \mid P^\pi)\big], \tag{25}$$

but given that $P(s' \mid s, a)$ and $V^\pi(s')$ are independent under our assumptions, then conditioning on P^π means that the distribution of the matrix-vector product $P^\pi V^\pi$ is simply the distribution of applying a linear transformation on V^π. The result is that the conditional distribution can be interpreted as the pushforward

$$\mathfrak{D}_P(r^\pi + \gamma P^\pi V^\pi \mid P^\pi) = (b_{r^\pi, P^\pi, \gamma})_\# \mu^\pi, \tag{26}$$

which completes the proof. □

We adopt the supremum p-Wasserstein distance to establish contractivity of the operator $\mathcal{T}^\pi$.

Definition 2. For $p \in [1, \infty)$, the p-Wasserstein distance between two distributions ν, ν' is a metric $w_p : \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}) \to [0, \infty]$ defined by

$$w_p(\nu, \nu') = \left(\int_0^1 \left| F_\nu^{-1}(\tau) - F_{\nu'}^{-1}(\tau) \right|^p d\tau \right)^{1/p}, \tag{27}$$

where $F_{(\cdot)}^{-1}$ is the inverse cumulative distribution function. Furthermore, the supremum p-Wasserstein distance $\bar{w}_p$ between two value distribution functions $\mu, \mu' \in \mathcal{P}(\mathbb{R})^{\mathcal{S}}$ is defined by $\bar{w}_p(\mu, \mu') = \sup_{s \in \mathcal{S}} w_p(\mu(s), \mu'(s))$.

The supremum p-Wasserstein distance was proven to be a metric in $\mathcal{P}(\mathbb{R})^{\mathcal{S}}$ by Bellemare et al. (2017).

To prove that $\mathcal{T}^\pi$ is a contraction, we adopt the technique from Bellemare et al. (2023) that relies on the alternative definition of the p-Wasserstein distance in terms of couplings.

Definition 3. Let $\nu, \nu' \in \mathcal{P}(\mathbb{R})$ be two probability distributions over the reals. A coupling υ is a joint probability whose marginals are ν and ν', such that if $(V, V') \sim \upsilon$, then we have $V \sim \nu$ and $V' \sim \nu'$. Further, we denote $\Gamma(\nu, \nu')$ the set of all couplings of ν and ν'.

Intuitively, the coupling υ can be interpreted as a transport plan to move probability mass from one distribution to another. As such, the p-Wasserstein distance is defined as the cost of the optimal transport plan.

$$w_p(\nu, \nu') = \min_{\upsilon \in \Gamma(\nu, \nu')} \mathbb{E}_{(V, V') \sim \upsilon}\big[|V - V|^p\big]^{1/p} \tag{28}$$

With these definitions, we now proceed to prove the contraction of the Bellman operator.

Lemma 2. *The operator $\mathcal{T}^\pi$ is a γ-contraction with respect to $\bar{w}_p$ for all $p \in [1, \infty)$. That is, $\bar{w}_p(\mathcal{T}^\pi \mu, \mathcal{T}^\pi \mu') \leq \gamma \bar{w}_p(\mu, \mu')$ for all $\mu, \mu' \in \mathcal{P}(\mathbb{R})^{\mathcal{S}}$.*

Proof. Follows closely the proof of Proposition 4.15 by Bellemare et al. (2023). For each $s \in \mathcal{S}$, let $\upsilon^\star$ denote the optimal coupling w.r.t. the p-Wasserstein distance between some arbitrary pair of value distributions $\mu(s), \mu'(s) \in \mathcal{P}(\mathbb{R})$. Further, define a pair of random variables $(V(s), V'(s)) \sim \upsilon^\star$ such that $V(s) \sim \mu(s)$ and $V'(s) \sim \mu'(s)$. We assume the pair $(V(s'), V'(s'))$ is statistically independent of $P^\pi(s' \mid s)$ for any $s' \in \mathcal{S}$, akin to Assumptions 1 and 2.

Define new random variables $\tilde{V}(s) = r^\pi(s) + \gamma \sum_{s'} P^\pi(s' \mid s) V(s')$, $\tilde{V}'(s) = r^\pi(s) + \gamma \sum_{s'} P^\pi(s' \mid s) V'(s')$. By definition of the operator $\mathcal{T}^\pi$, we have that $\tilde{V}(s) \sim (\mathcal{T}^\pi \mu)(s)$ and $\tilde{V}'(s) \sim (\mathcal{T}^\pi \mu')(s)$, which means that the pair $(\tilde{V}(s), \tilde{V}'(s)) \sim \tilde{\upsilon}$ is a coupling between $(\mathcal{T}^\pi \mu)(s)$ and $(\mathcal{T}^\pi \mu')(s)$.

Starting from Definition 3, by definition of the p-Wasserstein distance

$$w_p^p\big((\mathcal{T}\mu)(s), (\mathcal{T}\mu')(s)\big) \leq \mathbb{E}_P\left[\left|\tilde{V}(s) - \tilde{V}'(s)\right|^p\right]. \tag{29}$$

Plugging the definition of the random variables,

$$= \mathbb{E}_P\left[\left|r^\pi(s) + \gamma\sum_{s'} P^\pi(s' \mid s)V(s') - r^\pi(s) - \gamma\sum_{s'} P^\pi(s' \mid s)V'(s')\right|^p\right] \tag{30}$$

By re-arrangement of terms

$$= \gamma^p\,\mathbb{E}_P\left[\left|\sum_{s'} P^\pi(s' \mid s)(V(s') - V'(s'))\right|^p\right]. \tag{31}$$

Since $f(x) = |x|^p$ is convex for $p \geq 1$, then by Jensen's inequality

$$\leq \gamma^p\,\mathbb{E}_P\left[\sum_{s'} P^\pi(s' \mid s)\big|(V(s') - V'(s'))\big|^p\right]. \tag{32}$$

By linearity of expectation

$$= \gamma^p \sum_{s'} \mathbb{E}_P\Big[P^\pi(s' \mid s)\big|(V(s') - V'(s'))\big|^p\Big]. \tag{33}$$

By the independence assumption on P^π, the expectation of the product becomes the product of expectations

$$= \gamma^p \sum_{s'} \mathbb{E}_P\big[P^\pi(s' \mid s)\big]\,\mathbb{E}_P\Big[\big|(V(s') - V'(s'))\big|^p\Big]. \tag{34}$$

Since the supremum of non-negative values is greater or equal than any convex combination of them

$$\leq \gamma^p \sup_{s'} \mathbb{E}_P\Big[\big|(V(s') - V'(s'))\big|^p\Big]. \tag{35}$$

By definition of the supremum p-Wasserstein distance

$$= \gamma^p \bar{w}_p^p(\mu, \mu'). \tag{36}$$

Taking supremum on the left-hand side and taking the p-th root on both sides completes the proof. □

Lemma 2 parallels similar results in standard RL and model-free distributional RL, in that it allows us to establish the convergence of iterated applications of $\mathcal{T}^\pi$ (Theorem 1).

B Implementation Details

B.1 Quantile Huber Loss

We adopt the quantile Huber loss from Dabney et al. (2018b) in order to train the distributional critic. The Huber loss is given by

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & \text{otherwise} \end{cases}, \tag{37}$$

and the quantile Huber loss is defined by

$$\rho_\tau^\kappa(u) = \big|\tau - \delta(u < 0)\big|\mathcal{L}_\kappa(u). \tag{38}$$

For $\kappa = 0$, we recover the standard quantile regression loss, which is not smooth as $u \to 0$. In all our experiments we fix $\kappa = 1$ and to simplify notation define $\rho_\tau^1 = \rho_\tau$.

Algorithm 2 Epistemic Quantile-Regression with Soft Actor-Critic (EQRSAC)

1: Initialize policy π_ϕ, MDP ensemble Γ_ψ, quantile critic θ, environment dataset $\mathcal{D}$, model dataset $\mathcal{D}_{\text{model}}$, utility function f.
2: Warm-up $\mathcal{D}$ with rollouts under π_ϕ
3: global step $\leftarrow 0$
4: **for** episode $t = 0, \ldots, T-1$ **do**
5: **for** E steps **do**
6: **if** global step $\% \; F == 0$ **then**
7: Train Γ_ψ on $\mathcal{D}$ via maximum likelihood
8: **for** each MDP dynamics in Γ_ψ **do**
9: **for** L model rollouts **do**
10: Perform k-step rollouts starting from $s \sim \mathcal{D}$; add to $\mathcal{D}_{\text{model}}$
11: Take action in environment according to π_ϕ; add to $\mathcal{D}$
12: **for** G gradient updates **do**
13: Update $\{\theta_i\}_{i=1}^{M}$ with mini-batches from $\mathcal{D}_{\text{model}}$, via SGD on (14)
14: Update π_ϕ with mini-batches from $\mathcal{D}_{\text{model}}$, via SGD on (16)
15: global step $\leftarrow$ global step $+1$

B.2 EQRSAC Algorithm

A detailed execution flow for training an EQRSAC agent is presented in Algorithm 2. Further implementation details are now provided.

Model learning. We use the `mbrl-lib` Python library from Pineda et al. (2021) to train N neural networks (Line 7). Our default architecture consists of four fully-connected layers with 200 neurons each (for the Quadruped environments we use 400 neurons to accomodate the larger state space). The networks predict delta states, $(s' - s)$, and receives as inputs normalizes state-action pairs. The normalization statistics are updated each time we train the model and are based on the training dataset $\mathcal{D}$. We use the default initialization that samples weights from a truncated Gaussian distribution, but we increase by a factor of 2 the standard deviation of the sampling distribution.

Capacity of $\mathcal{D}_{\text{model}}$. The capacity of the model buffer is computed as $k \times L \times F \times N \times \Delta$, where Δ is the number of model updates we want to retain data in the buffer. That is, the buffer is filled only with data from the latest Δ rounds of model training and data collection (Lines 6-10).

Critic Loss. The distributional critic is updated in Line, for which we use the loss function (14). To approximate the target quantiles (15), we use the learned generative model and the policy to generate transition tuples (r, s', a'). More specifically, each (s, a) pair in a mini-batch from $\mathcal{D}_{\text{model}}$ is repeated X times and passed through the environment model, thus generating X predictions (r, s'). Then, every s' prediction is repeated Y times and passed through π_ϕ, thus obtaining XY next state-action pairs (s', a'). This generated data is finally used in (15) to better approximate the expectation. In our experiments we use $X = Y$ and keep their product as a hyperparameter controlling the total amount of samples we use to approximate the expectation.

B.3 Baselines

In this section we provide more details about the baselines algorithms. We use a single codebase for all experiments and share architecture components amongst baselines whenever possible. The execution of experiments for all baselines follows the workflow of Algorithm 2.

SAC. We base our implementation on the open-source repository `https://github.com/pranz24/pytorch-soft-actor-critic` and allow for either model-free or model-based data buffers for the agent's updates, as done in `mbrl-lib`. The implementation uses typical design choices like target networks (Mnih et al., 2013), clipped double Q-learning (Fujimoto et al., 2018) and automatic entropy tuning (Haarnoja et al., 2019)

MBPO. Our MBPO implementation differs slightly from Janner et al. (2019) in that: (1) it only uses $\mathcal{D}_{\text{model}}$ to update the actor and critic, rather than mixing in data from $\mathcal{D}$; (2) it uses a fixed rollout length k, instead of having an adaptive scheme. With respect to EQRSAC, MBPO collects data differently: instead of collecting k-step rollouts under each model of the ensemble, it does so by

uniformly sampling a new model per-step of the rollout. We maintain the original model-buffer capacity of MBPO, calculated as $M \times F \times k \times \Delta$ (i.e., a factor of N smaller than EQRSAC).

QRMBPO. Simply replaces the critic in MBPO with a quantile-distributional critic, trained on the standard quantile-regression loss from Wurman et al. (2022), but using data from $\mathcal{D}_{\text{model}}$,

$$\mathcal{L}_{\text{critic}}^{\text{qrmbpo}}(\theta) = \mathbb{E}_{(S,A,S',R)\sim\mathcal{D}_{\text{model}}}\left[\left[\sum_{i=1}^{M}\mathbb{E}_J\left[\rho_{\tau_i}\left(\mathcal{T}\theta_J^{\text{qrmbpo}}(R,S,A) - \theta_i(S,A)\right)\right]\right]\right], \tag{39}$$

where the target quantile is defined as

$$\mathcal{T}\theta_j^{\text{qrmbpo}}(r,s',a) = r + \gamma\big(\theta_j(s',a') - \alpha\log\pi_\phi(a' \mid s')\big), \tag{40}$$

and $a' \sim \phi_\phi(a' \mid s')$. Importantly, (39) differs from (14) in that it only uses transitions sampled from the buffer. The objective function for the actor is fixed to be f_{mean}.

QUSAC. It collects data as in EQRSAC, but stores the N model-consistent rollouts in N separate buffers (while EQRSAC uses a single buffer). Then it trains an ensemble of N standard critics on the corresponding N model-buffers. As such, it interprets the ensemble of critics as samples from the value distribution. The actor loss function uses the mean prediction of the critic, mirroring QRMBPO and EQRSAC-mean.

C Hyperparameters

Table 2: Hyperparameters for DeepMind Control Suite. In red, we highlight the only deviations of the base hyperparameters across all environments and baselines.

Name	**Value**
General	
T - # episodes	250
E - steps per episode	10^3
Replay buffer $\mathcal{D}$ capacity	10^5
Warm-up steps (under initial policy)	5×10^3
SAC	
G - # gradient steps	10
Batch size	256
Auto-tuning of entropy coefficient α?	Yes
Target entropy	$-\text{dim}(\mathcal{A})$
Actor MLP network	2 hidden layers - 128 neurons - Tanh activations
Critic MLP network	2 hidden layers - 256 neurons - Tanh activations
Actor/Critic learning rate	3×10^{-4}
Dynamics Model	
N - ensemble size	5
F - frequency of model training (# steps)	250
L - # model rollouts per step	400
k - rollout length	5
Δ - # Model updates to retain data	10
Model buffer $\mathcal{D}_{\text{model}}$ capacity (EQRSAC)	$M \times F \times k \times \Delta(\times N) = 5 \times 10^6 (25 \times 10^6)$
Model MLP network (quadruped)	4 layers - 200 (400) neurons - SiLU activations
Learning rate	1×10^{-3}
Quantile Network	
M - # quantiles	51
# (s', a') samples (EQRSAC only)	25

D DM Control Learning Curves

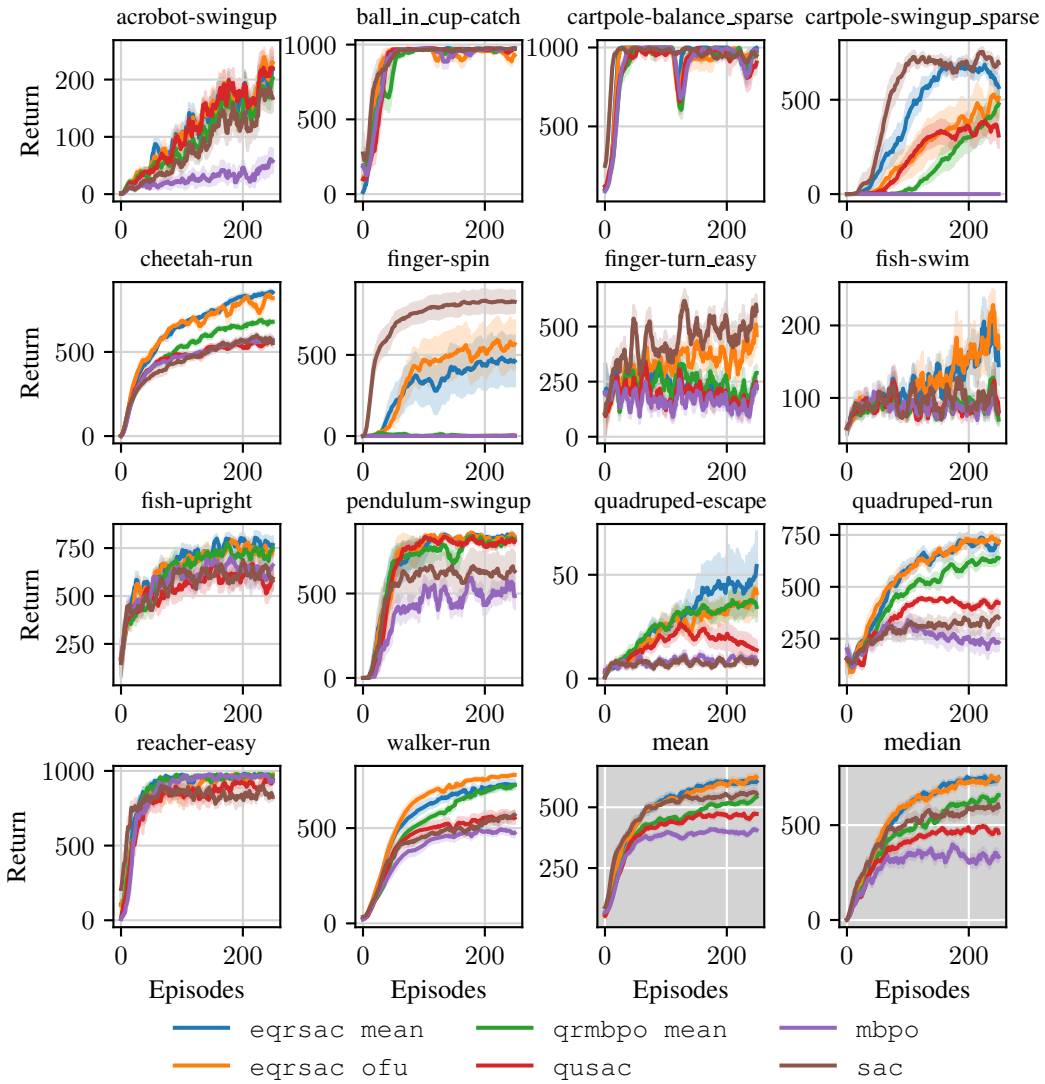

Figure 4: DeepMind Control benchmark of 14 continuous-control tasks. The returns are smoothened by a moving average filter and we report the mean and standard error over 10 random seeds.

E DM Control Final Scores

Table 3: Scores in DeepMind Control benchmark after 250 episodes (or 250K environment steps). For each environment, we report the mean and standard error scores over 10 random seeds.

Environment	`sac`	`mbpo`	`qrmbpo`	`qusac`	`eqrsac mean`	`eqrsac ofu`
acrobot-swingup	167.7 ± 22.4	57.7 ± 19.9	$\mathbf{202.3 \pm 18.7}$	$\mathbf{220.3 \pm 30.8}$	$\mathbf{217.8 \pm 18.5}$	$\mathbf{229.4 \pm 17.4}$
ball-in-cup-catch	$\mathbf{972.6 \pm 3.3}$	$\mathbf{972.5 \pm 1.7}$	$\mathbf{974.4 \pm 2.3}$	$\mathbf{972.8 \pm 2.8}$	$\mathbf{977.2 \pm 1.4}$	928.7 ± 18.5
cartpole-balance-sparse	949.5 ± 18.7	985.6 ± 9.7	977.0 ± 22.2	904.5 ± 37.3	$\mathbf{997.8 \pm 2.1}$	968.0 ± 15.2
cartpole-swingup-sparse	$\mathbf{693.8 \pm 27.2}$	0.1 ± 0.1	476.2 ± 72.6	310.6 ± 64.3	566.4 ± 54.4	510.6 ± 86.9
cheetah-run	551.6 ± 22.6	571.4 ± 19.3	679.1 ± 10.7	567.4 ± 16.8	$\mathbf{854.0 \pm 11.7}$	820.0 ± 18.4
finger-spin	$\mathbf{827.5 \pm 68.1}$	0.1 ± 0.1	1.2 ± 1.1	3.2 ± 2.6	567.1 ± 146.7	461.9 ± 154.1
finger-turn-easy	$\mathbf{571.3 \pm 31.3}$	220.0 ± 20.0	289.8 ± 34.1	220.0 ± 20.0	221.3 ± 19.9	460.5 ± 58.3
fish-swim	79.9 ± 10.9	80.6 ± 10.0	70.0 ± 8.7	83.8 ± 10.0	$\mathbf{145.1 \pm 27.3}$	$\mathbf{168.3 \pm 20.4}$
fish-upright	579.4 ± 50.8	660.5 ± 59.5	$\mathbf{749.9 \pm 29.1}$	591.9 ± 53.5	$\mathbf{766.2 \pm 45.3}$	$\mathbf{735.0 \pm 23.5}$
pendulum-swingup	631.0 ± 111.7	484.5 ± 76.4	$\mathbf{819.2 \pm 17.2}$	$\mathbf{808.6 \pm 19.7}$	$\mathbf{834.4 \pm 15.7}$	$\mathbf{833.7 \pm 16.0}$
quadruped-escape	8.0 ± 1.2	8.8 ± 1.8	$\mathbf{34.5 \pm 7.1}$	13.7 ± 4.3	$\mathbf{54.2 \pm 16.7}$	$\mathbf{41.1 \pm 11.4}$
quadruped-run	352.0 ± 36.4	232.3 ± 41.2	638.5 ± 26.0	421.9 ± 16.8	$\mathbf{719.8 \pm 19.0}$	$\mathbf{712.5 \pm 23.6}$
reacher-easy	824.6 ± 21.9	474.3 ± 20.8	968.6 ± 9.8	943.1 ± 13.8	931.2 ± 21.5	$\mathbf{977.9 \pm 2.5}$
walker-run	568.9 ± 19.1	474.3 ± 20.8	725.5 ± 10.8	553.8 ± 30.9	727.4 ± 24.3	$\mathbf{779.3 \pm 7.9}$
mean	555.6 ± 16.2	333.1 ± 25.8	543.3 ± 7.9	472.5 ± 6.3	$\mathbf{605.3 \pm 13.0}$	$\mathbf{623.7 \pm 12.4}$
median	595.6 ± 26.4	406.1 ± 7.4	658.6 ± 13.1	459.4 ± 14.4	$\mathbf{750.1 \pm 13.3}$	$\mathbf{748.3 \pm 11.4}$