Reducing Variance in Temporal-Difference Value Estimation via Ensemble of Deep Networks

Litian Liang [1] **Yaosheng Xu** [1] **Stephen McAleer** [2] **Dailin Hu** [1]
Alexander Ihler [1] **Pieter Abbeel** [3] **Roy Fox** [1]

Abstract

In temporal-difference reinforcement learning algorithms, variance in value estimation can cause instability and overestimation of the maximal target value. Many algorithms have been proposed to reduce overestimation, including several recent ensemble methods, however none have shown success in sample-efficient learning through addressing estimation variance as the root cause of overestimation. In this paper, we propose MeanQ, a simple ensemble method that estimates target values as ensemble means. Despite its simplicity, MeanQ shows remarkable sample efficiency in experiments on the Atari Learning Environment benchmark. Importantly, we find that an ensemble of size 5 sufficiently reduces estimation variance to obviate the lagging target network, eliminating it as a source of bias and further gaining sample efficiency. We justify intuitively and empirically the design choices in MeanQ, including the necessity of independent experience sampling. On a set of 26 benchmark Atari environments, MeanQ outperforms all tested baselines, including the best available baseline, SUNRISE, at 100K interaction steps in 16/26 environments, and by 68% on average. MeanQ also outperforms Rainbow DQN at 500K steps in 21/26 environments, and by 49% on average, and achieves average human-level performance using 200K ($\pm$100K) interaction steps. Our implementation is available at https://github.com/indylab/MeanQ.

[1]University of California, Irvine [2]Carnegie Mellon University [3]University of California, Berkeley. Correspondence to: Litian Liang and Roy Fox <{litianl1, royf}@uci.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

1. Introduction

Model-free reinforcement learning (RL) with deep learning has proven successful at achieving master-level performance in many sequential decision making problems (Silver et al., 2016; 2018; Mnih et al., 2015; Vinyals et al., 2019; Kalashnikov et al., 2018; Haarnoja et al., 2018b). RL algorithms learn a control policy that maximizes the expected discounted sum of future rewards (the *policy value*) through experience collected via interacting with an environment. Temporal-Difference (TD) (Sutton & Barto, 2018) is a principled approach to RL that maintains value estimates and iteratively improves them using *bootstrapped* targets that combine experienced short-term rewards and current estimates of future long-term values.

Q-learning (Watkins & Dayan, 1992) learns state–action value estimates (*Q function*) by minimizing the TD error between the estimates and the bootstrapped targets. Although tabular Q-learning asymptotically converges to the optimal policy (Tsitsiklis, 1994; Jaakkola et al., 1994), it is known to be positively biased and overestimate the value before convergence due to Jensen's inequality under value estimation uncertainty (Thrun & Schwartz, 1993; Fox et al., 2015). This bias is detrimental to efficient learning, because it propagates through bootstrapping and because it can cause further experience collection to use suboptimal actions that appear optimal. Deep Q-Networks (DQN) (Mnih et al., 2013; 2015) represent the Q function with a deep neural network to learn expressive policies in environments with high dimensional states. Unfortunately, such value function approximation can further exacerbate instability and overestimation (Thrun & Schwartz, 1993; Mahmood et al., 2015).

One way to reduce the variance (between-runs variability) and instability (within-run variability) of value estimation is to use an ensemble of estimators. Ensemble learning is well-studied in machine learning and is known for its property of reducing estimation variance and ability to capture epistemic (model) uncertainty, usually in a form of prediction variance. In the field of RL, ensemble RL methods (Wiering & van Hasselt, 2008) have been applied to improve exploration (Osband et al., 2016a;b; Chen et al., 2017; Fortunato et al.,

2017) and guide value or policy updates (van Hasselt et al., 2015; Fujimoto et al., 2018; Fox, 2019; Lan et al., 2020; Lee et al., 2021; Liang et al., 2021; Chen et al., 2021).

In this work, we propose a simple model-free ensemble method, called MeanQ, that reduces variance and instability of the TD target estimates by averaging an ensemble of neural network learners. We discuss similarities and differences between MeanQ and several closely related existing methods, including Anschel et al. (2016), and justify our specific design choices intuitively and empirically. We discuss the variance-reduction properties of MeanQ's target estimator and show empirically that, in some cases, instability is sufficiently reduced to eliminate the need for a target network (Mnih et al., 2015; Lillicrap et al., 2015; Kim et al., 2019), which removed the target lag and further improves sample efficiency.

In experiments in the Atari Learning Environment (ALE) domain (Bellemare et al., 2013), MeanQ significantly outperforms all baselines with which we compared, in a benchmark set of 26 environments. MeanQ outperforms the best available baseline, SUNRISE (Lee et al., 2021), at 100K interaction steps in 16/26 environments, and by 68% in normalized return averaged over all 26 environments. Compared with a commonly used baseline, Rainbow DQN (Hessel et al., 2018), MeanQ achieves higher returns at 500K steps in 21/26 environments, and 49% higher average normalized return. MeanQ also achieves average human-level performance in the 26 games using only 200K ($\pm$100K) interaction steps.

2. Preliminaries

We consider a Markov Decision Process (MDP) with probability $p(s'|s,a)$ to transition to state s' when taking in state s an action a in a finite action space. An agent controls the dynamical process using a policy π with probability $\pi(a|s)$ to take action a in state s, after which a reward $r(s,a)$ is observed.

The MDP and policy jointly induce a distribution p_π over the trajectory $\xi = (s_0, a_0, r_0, s_1 \ldots)$ in each *episode*. An RL algorithm should discover a control policy that maximizes the expected discounted return, $R(\xi) = \sum_{t\geq 0} \gamma^t r_t$, of trajectory ξ, where t is the time step in the interaction process, $0 \leq \gamma < 1$ is a discount factor, and $r_t = r(s_t, a_t)$. The value-to-go (Q value) of policy π, starting at state–action pair (s,a), is

$$Q_\pi(s,a) = \mathbb{E}_{\xi\sim p_\pi}[R(\xi)|s_0 = s, a_0 = a].$$

Value-based RL methods maintain a state–action value function (Q function) to infer a control policy and guide policy updates. Q-learning (Watkins & Dayan, 1992) learns the optimal value of each state–action by stochastically updating a tabular representation of Q, on experience (s,a,r,s'), by

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)),$$

to minimize the Temporal-Difference (TD) error in parentheses. Deep Q-Networks (DQN) (Mnih et al., 2013; 2015) learn a parametrized Q_θ value function by minimizing the squared error between Q_θ and the target value estimate, typically predicted by a lagging version $Q_{\bar\theta}$ of the current value function

$$\mathcal{L}(s,a,r,s';\theta) = (r + \gamma \max_{a'} Q_{\bar\theta}(s',a') - Q_\theta(s,a))^2.$$

The stochastic fitting process means that, before convergence of the algorithm, the Q values are uncertain value estimates. The $\max$ operator in the noisy target value estimate has been shown to introduce an overestimation bias (Thrun & Schwartz, 1993), known as the "winner's curse", due to Jensen's inequality

$$\mathbb{E}[\max_{a'} Q(s',a')] - \max_{a'} \mathbb{E}[Q(s',a')] \geq 0, \qquad (1)$$

where the expectation is over runs of the algorithm. Since the update is applied repeatedly through bootstrapping, it can iteratively increase the bias of the estimated Q values before convergence, and introduce instability into TD-learning algorithms as it propagates through the repeated Bellman update (Kumar et al., 2019; 2020) and neural-network extrapolation errors (van Hasselt et al., 2018; Lee et al., 2021).

3. Why Reduce Estimation Variance

To take useful update steps in parameter space, a sample-efficient value-based RL algorithm must be able to produce informative next-state target estimates, even in early stages of training when data is scarce and estimates are noisy. Unfortunately, stochasticity in the fitting process, including parameter initialization, environment dynamics, exploration, and replay sampling, leads to randomness in the parameters and the target estimates. This uncertainty can introduce bias via the Jensen gap as well as destabilize the training process, particularly under function approximation.

While the Jensen gap (1) cannot be fully characterized in terms of the variance of the value estimates, instead requiring extreme-value analysis, it is clear that target estimation variance generally contributes to overestimation bias in parametrized Q functions (Thrun & Schwartz, 1993; Kim et al., 2019; Duan et al., 2020). Reducing the variance of the value estimate can help reduce the resulting bias.

In the case of learning an approximate Q function, such as a neural network, variance in the target estimate (that is, estimate variability between runs) can also manifest as instability (that is, variability throughout a single run of

the learning process). One possible reason is that update steps with similar experiences, which should be similar, can become inconsistent in the presence of high target variance. A target network (Mnih et al., 2015; Lillicrap et al., 2015) regularizes the estimates by separating the learner's parameters from those responsible for producing target estimates. The target network uses a delayed copy or a Polyak-Rupert (exponential window) average of the parameters, effectively preventing it from changing too erratically.

Target networks have been shown to successfully stabilize off-policy TD learning (van Hasselt et al., 2018). However, by introducing a lagging network for value estimation, target networks introduce additional bias to the target estimate in exchange for lower variance. In many algorithms, the target network update rate is a hyperparameter that requires careful tuning, and can be domain-dependent.

By averaging an ensemble of Q networks, MeanQ reduces the target estimate variance. Empirically, this reduces both the overestimation bias and the instability that this variance can cause. Thus MeanQ enjoys better sample efficiency, which is even further improved by using up-to-date target estimates, since a lagging target network is shown to be unneeded for stability in MeanQ.

4. Related Work

4.1. Off-Policy RL

Off-policy RL algorithms improve sample efficiency by reusing environment interactions experienced by different policy (Fujimoto et al., 2018; Haarnoja et al., 2018a; Hessel et al., 2018). Off-policy learning is thus considered a promising direction for scaling RL to meet the needs of the real world (Levine, 2021). Rainbow DQN (Hessel et al., 2018) has been shown to perform well on the Atari Learning Environment benchmark (Bellemare et al., 2013) by combining a set of techniques ("Rainbow techniques") that are empirically successful at improving sample efficiency, including Double Q-learning (van Hasselt, 2010; van Hasselt et al., 2015), dueling networks (Wang et al., 2016), prioritized experience replay (Schaul et al., 2015), distributional value estimates (Bellemare et al., 2017), and noisy network exploration (Fortunato et al., 2017).

4.2. Stablizing Q-Learning

Direct bootstrapping with a learned parametrized function approximator can cause instability and overestimation (van Hasselt, 2010; van Hasselt et al., 2015; Fujimoto et al., 2018; Song et al., 2018; Kim et al., 2019; Kumar et al., 2019; 2020). To alleviate these effect, Double Q-learning (van Hasselt, 2010; van Hasselt et al., 2015) decorrelates the action optimization and value estimation, replacing the overestimation due to Jensen's inequality with a moderate underestimation. Twin-Q (Fujimoto et al., 2018) attempts to reduce overestimation more directly, by taking the minimum target value over two learner networks. Soft-maximal value targets, often involving the mellow-max operator (Ziebart, 2010; Rubin et al., 2012; Fox et al., 2015; Asadi & Littman, 2017; Kim et al., 2019), can reduce both the bias (Fox, 2019) and variance (Kim et al., 2019) of target value estimates. A weighted TD error has also been proposed to handle uncertainty in the training signal caused by error propagation through the self-referential update structure (Kumar et al., 2020) and by model uncertainty due to model expressiveness and limited data (Lee et al., 2021).

4.3. Ensemble Learning in RL

Ensemble learning, namely training a set of more than one learners for the same task, is often used in RL to guide exploration, and in some methods to improve target value estimates. The statistics of the ensemble can be used to assess model uncertainty or to produce a lower-variance estimate, compared to a single estimator. These benefits have been utilized in many RL techniques, such as in measuring the error accumulation in a learned dynamics model in model-based RL (Chua et al., 2018), evaluating a temperature for softer maximization in uncertain states (Fox, 2019; Liang et al., 2021), lowering the target estimate variance (Anschel et al., 2016; An et al., 2021), biasing exploration toward novel states (Chen et al., 2017; Osband et al., 2016a; Lee et al., 2021), down-weighting loss for uncertain target values (Lee et al., 2021), and alleviating estimation bias by pessimistically estimating the target as the minimum over ensemble predictions (Fujimoto et al., 2018; Lan et al., 2020). Sufficient ensemble diversity is crucial in all of these methods (Sheikh et al., 2022).

While all ensemble methods are similarly motivated, approaches closely related to our own are Averaged-DQN and Ensemble-DQN (Anschel et al., 2016) and EBQL (Peer et al., 2021). Like MeanQ, these methods use an ensemble mean as the target estimate, with Ensemble-DQN and EBQL maintaining an explicit ensemble and Averaged-DQN reusing past snapshots of the network parameters. Both Ensemble-DQN and EBQL train all ensemble members with the same experience mini-batch and mean target values, unlike MeanQ which has each member sample independently from a shared replay buffer. The significance of these design choices is further discussed in Section 5.2.

Deep Exploration methods, such as RLSVI (Osband et al., 2016b), collect training data by selecting an ensemble member to interact with the environment throughout each episode. The ultimately deployed policy has the same property, unlike MeanQ which selects greedy actions using the ensemble mean in both exploration and deployment. Consistently with this difference, the methods also differ in how

they generate target values, with RLSVI bootstrapping each member from its own value estimates and MeanQ averaging the ensemble for all targets. Thus, both methods collect data relevant to their targets and evaluated policies, which has been shown imperative for successful training (Ostrovski et al., 2021). We find empirically that this latter consideration outweighs the potential benefit of independent experience through Deep Exploration; that off-policy training with a replay buffer (Mnih et al., 2015) provides sufficient experience diversity in MeanQ; and that the correlation between ensemble members introduced by the shared replay buffer is alleviated by independent sampling from the buffer.

SUNRISE (Lee et al., 2021) and MeanQ differ only in how they compute the TD error. SUNRISE bootstraps each ensemble member from its corresponding target network, while MeanQ uses the ensemble mean. SUNRISE also rescales the TD error based on an ensemble-induced uncertainty measure of the target value.

5. MeanQ

We present MeanQ, a simple and sample-efficient ensemble-based RL algorithm. In this section, we discuss in detail each step of the algorithm and compare it with existing methods. Finally, we discuss how MeanQ can be combined with several RL techniques that have been shown useful in sample-efficient learning of Q networks.

5.1. Updating Q Ensemble

MeanQ estimates state–action values with an ensemble of K Q networks. As in DQN, each network $k = 1, \ldots, K$, parametrized by θ_k, takes a state s as input, and outputs a vector predicting $Q_{\theta_k}(s, a)$ for each action a. The Q-network architecture allows maximizing the Q function over actions and finding a maximizing action in two operations that require it: (1) computing target values for optimization; (2) selecting a greedy action for rolling out the trained agent, whether to collect more data or to evaluate. In MeanQ, the maximum is taken over the mean value of the K ensemble members:

$$V_\theta(s) = \max_a \{\operatorname*{mean}_k Q_{\theta_k}(s, a)\}.$$

In particular, this $V_\theta(s)$ is used for the target value when taking gradient steps to minimize the square TD error for each estimator θ_k, with respect to target values computed by separate target networks $\bar{\theta}_1, \ldots, \bar{\theta}_K$

$$\mathcal{L}(s, a, r, s'; \theta_k) = \left(r + \gamma V_{\bar{\theta}}(s') - Q_{\theta_k}(s, a)\right)^2.$$

5.2. Decorrelating Ensemble Members

The motivation for using an ensemble mean is that, when the members are not fully correlated, the mean has lower variance. We expect this variance reduction to provide the benefits discussed in Section 3, including decreasing the overestimation bias in $V_{\bar{\theta}}(s')$ due to Jensen's inequality.

Effectively reducing variance requires keeping the ensemble members as uncorrelated as possible. However, complete independence of the members is impossible; in order to make use of the improved value estimates, they must affect each other's target values. MeanQ strikes a balance between keeping sources of estimate stochasticity independent when it can be done efficiently, and otherwise allowing their dependence. The complete MeanQ method is presented in Algorithm 1.

The first source of estimate stochasticity is the initialization of each Q network (Osband et al., 2016a), which is naturally done independently for each of the K ensemble members.

When rolling out an exploration policy, it is possible for each Q_{θ_k} to form its own ϵ-greedy exploration policy. However, that policy would not reflect our best available policy, resulting in suboptimal exploration. It would also differ from the evaluated policy

$$\pi_\theta(s) = \arg\max_a \{\operatorname*{mean}_k Q_{\theta_k}(s, a)\},$$

which can be harmful (Ostrovski et al., 2021). We therefore use the ensemble mean for ϵ-greedy exploration, by selecting $\pi_\theta(s)$ in state s with probability $1 - \epsilon$ and a uniform action otherwise.

Q-learning networks typically use a replay buffer to store experience and draw mini-batches for training. This diversifies each mini-batch by mixing in it steps from different episodes or disparate times in an episode. The next design choice for MeanQ is whether ensemble members use individual replay buffers (which could be experienced by π_θ or $\pi_{\theta_k} = \arg\max_a Q_{\theta_k}$) or share a single buffer. For the same amount of total exploration, a shared buffer is more diverse, and we find it empirically beneficial.

Using a shared experience policy and a shared replay buffer further correlates the ensemble members. To alleviate this negative effect and help the members evolve more independently, MeanQ has each member sample its mini-batches independently from the replay buffer. A downside, however, is computational efficiency. Computing targets for each member requires us to evaluate all members' value estimates, Q_{θ_k}, at each state s' present in any of the members' mini-batches, performing $O(K^2)$ target Q function evaluations for $O(K)$ gradient updates. For small K (5 in our experiments), this increase is not prohibitive, and is offset by the resulting improvement in sample efficiency,

particularly in settings where environment interactions are more expensive than function evaluations.

MeanQ is closely related to Averaged-DQN (Avg-DQN) and Ensemble-DQN (Ens-DQN) (Anschel et al., 2016) and Ensemble Bootstrapped Q-Learning (EBQL) (Peer et al., 2021), which are similarly motivated. For computational reasons, these methods were designed to have members share mini-batch data (Ens-DQN and EBQL) or simply reuse recent snapshots of a single network (Avg-DQN). We find empirically that these choices degrade performance. In our experiments (Section 6.3), we test Ens-DQN by using the same replay samples across members; intuitively, this makes the members more correlated and so reduces variance less. Our experiments indicate that, while Ens-DQN provides only modest benefits over Avg-DQN, by modifying the ensemble training procedure to that of MeanQ we are able to improve sample efficiency significantly over Ens-DQN.

Finally, for prioritized experience replay (Schaul et al., 2015), the priority weights that control the sampling distribution are computed using the previous TD error for each data point. Independently sampling for each ensemble member enables us to also keep separate priorities, so that each member samples experience that is most relevant to its own value estimates. To this end, MeanQ computes the priority weights for each ensemble member using its own previous TD errors.

5.3. Combining with Existing Techniques

Our proposed method can be combined with several existing techniques for improving deep Q-learning, namely Rainbow DQN (Hessel et al., 2018) and UCB exploration (Chen et al., 2017) as discussed below, and doing so further improves its performance.

Distributional target estimation. A useful modification to standard Q-learning is to have the network output a *distribution* over values for each action (Bellemare et al., 2017). This value distribution is typically approximated by a categorical distribution over a fixed finite set of possible return values, called atoms. A distributional MeanQ target estimate can be computed as an average over the members' estimates of the probability mass of each atom

$$p_{s'} = \operatorname*{mean}_k p_{\bar{\theta}_k}(s', a^*)$$
$$a^* = \arg\max_{a'} \{\operatorname*{mean}_k z^\intercal p_{\bar{\theta}_k}(s', a'), \}$$

where z are the locations of the return value's discrete support, similarly to Bellemare et al. (2017). Finally, we take member k's loss function to be the cross-entropy loss between $p_{\theta_k}(s, a)$ and $p_{s'}$, with the latter projected onto the support z. Implementation details are given in Appendix A.

Algorithm 1 MeanQ

Initialize K Q networks Q_{θ_k} for all k
Initialize K target Q networks $Q_{\bar{\theta}_k}$, $\bar{\theta}_k \leftarrow \theta_k$ for all k
Initialize replay memory D to capacity N
Initialize prioritization $p_k(b|D)$ for all k
for $t = 1, \ldots, T$ **do**
  Sample action a_t according to exploration policy π_θ
  Observe $r_t \leftarrow r(s_t, a_t)$
  Sample $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
  Store transition (s_t, a_t, r_t, s_{t+1}) in D
  $s_t \leftarrow s_{t+1}$
  for $k = 1, \ldots, K$ **do**
    Sample B transitions $(s_b, a_b, r_b, s'_b) \sim D$
    $V_{s'_b} = \max_{a'_b}\{\text{mean}_k\, Q_{\bar{\theta}_k}(s'_b, a'_b)\}$
    $y_b = \begin{cases} r_b & \text{if } s'_b \text{ is a terminal state} \\ r_b + \gamma V_{s'_b} & \text{otherwise} \end{cases}$
    $\mathcal{L}(\theta_k) = \text{mean}_b\, \mathcal{L}_b(\theta_k)$,
    where $\mathcal{L}_b(\theta_k) = (y_b - Q_{\theta_k}(s_b, a_b))^2$
    Update θ_k using $\nabla_{\theta_k}\mathcal{L}(\theta_k)$
    Update $p_k(b|D)$ using $\sqrt{\mathcal{L}_b(\theta_k)}$
  end
  Every T_{target} steps: update $\bar{\theta}_k \leftarrow \theta_k$ for all k
end

UCB exploration. Ensembles have previously been used to assist in crafting an exploration policy (Auer et al., 2002; Osband et al., 2016a; Chen et al., 2017). Since MeanQ already has an ensemble, it is easy to combine it with any of these exploration methods. In our experiments, we combine MeanQ with the exploration policy used in Lee et al. (2021)

$$\pi(s) = \arg\max_a \{\operatorname*{mean}_k Q_{\theta_k}(s, a) + \lambda \operatorname*{std}_k Q_{\theta_k}(s, a)\},$$

where mean_k and std_k are the ensemble mean and standard deviation and $\lambda > 0$ is a hyperparameter.

Other techniques. MeanQ can also trivially incorporate dueling network (Wang et al., 2016) and noisy exploration (Fortunato et al., 2017), which have been observed to improve sample efficiency. MeanQ can also incorporate Double DQN (van Hasselt, 2010), itself a size-2 ensemble method, but this would halve the effective ensemble size, and is therefore note done in our experiments.

6. Experiment Results

6.1. Setup

We evaluate MeanQ on several discrete control benchmark environments in Atari (Bellemare et al., 2013). We compare with existing algorithms, including: (1) SUNRISE (Lee et al., 2021), a model-free value-based ensemble RL method that is, to our knowledge, the state-of-the-art on

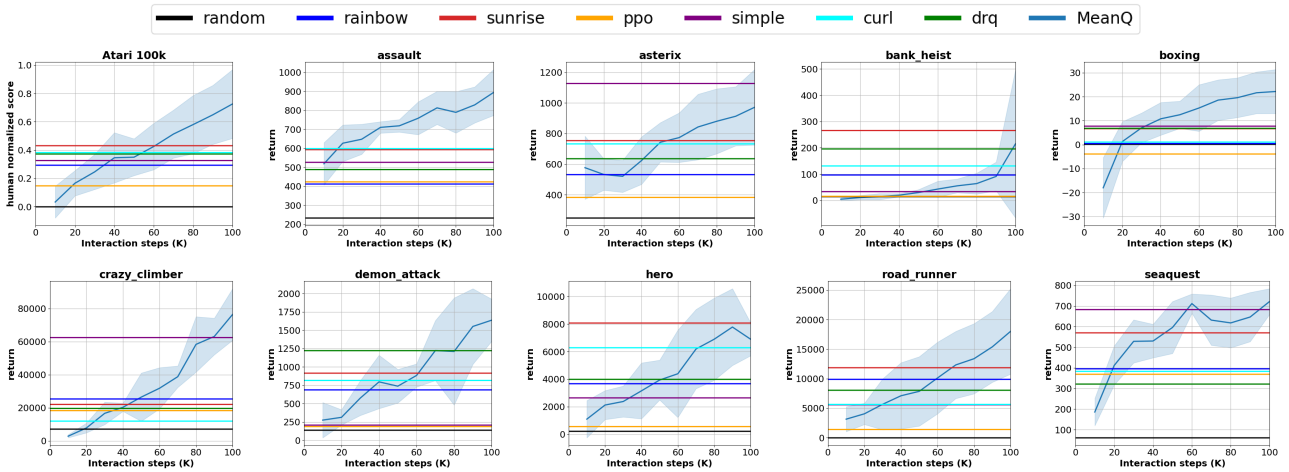

Figure 1. Performance of MeanQ with Rainbow techniques, compared with 6 existing algorithms. Mean expected undiscounted return over 5 runs is shown (shaded: 1 standard deviation), averaged over 26 Atari environments and in 9 individual ones. Baseline results of different algorithms at 100K interactions are rendered as horizontal lines and cited from: Rainbow (van Hasselt et al., 2019); SUNRISE (Lee et al., 2021); PPO, SimPLe, CURL, and DrQ (Kaiser et al., 2019). Full results in all 26 environments are available in Appendix C.

this benchmark; (2) SimPLe (Kaiser et al., 2019), a model-based method; (3) PPO (Schulman et al., 2017); (4) Rainbow DQN (Hessel et al., 2018) with two hyperparameter settings (van Hasselt et al., 2019; Kaiser et al., 2019); (5) CURL (Laskin et al., 2020); and (6) DrQ (Kostrikov et al., 2021). We also compare with human-level performance, as reported by Kaiser et al. (2019). For MeanQ, we use the version extended with the techniques of Section 5.3 and the same hyperparameter configuration as SUNRISE (Lee et al., 2021), with two exceptions: no multi-step targets are used (compared with 3 steps in SUNRISE) and no target network is used. To remain comparable with SUNRISE, all our evaluations use an ensemble size $K = 5$. Our MeanQ implementation is described in full detail in Appendix A and available at `https://github.com/indylab/MeanQ`.

6.2. Evaluation

Our results in Atari environments are summarized in Figure 1. The full results in all 26 environments are available in Appendix C. The experiment results show that MeanQ significantly outperforms all baselines with which we compared. MeanQ outperforms the best available baseline, SUNRISE, at 100K interaction steps in 16/26 environments. Scaling and shifting the returns such that random-policy performance is 0 and human-level performance is 1, to allow averaging across environments, MeanQ outperforms SUNRISE by 68% on average over 5 runs in each of the 26 environments. MeanQ also achieves higher returns than Rainbow DQN at 500K steps in 21/26 environments, and 49% higher average normalized return. Averaged over 5 runs in each environment, MeanQ achieves average human-level performance in the 26 games using only 200K ($\pm$100K) interaction steps. We note that, since our method is compatible with many more existing techniques which we did not evaluate, such as contrastive learning (Laskin et al., 2020), augmentation (Kostrikov et al., 2021), and various model-based methods, it has the potential for further performance improvement.

6.3. Effect of Target Network

In this experiment, we compare DQN and MeanQ with and without a lagging target network. To isolate the effects on MeanQ itself, we experiment without including any of the techniques described in Section 5.3 ("vanilla MeanQ"). To study the effect of variance on overestimation bias and policy performance, it is also important not to use a distributional return representation (Bellemare et al., 2017), which curtails overestimation due to the clipping effect of projecting the value distribution onto a fixed discrete support. Exploration is ϵ-greedy with linear annealing of ϵ. The detailed modification in hyperparameter configuration for this experiment is available in Appendix B.

To study the algorithmic behavior, we measure a set of statistics during training:

1. **Performance.** We measure performance by the undiscounted sum of rewards $\sum_t r_t$, averaged over 20 evaluation trajectories after every 10K interaction steps of the exploration policy.
2. **Estimation bias.** We measure the value estimation bias in the initial state s_0. In this experiment, a prediction of the learner network $V(s_0) = \max_a Q_\theta(s_0, a)$ is queried from the learner. For ensemble methods,

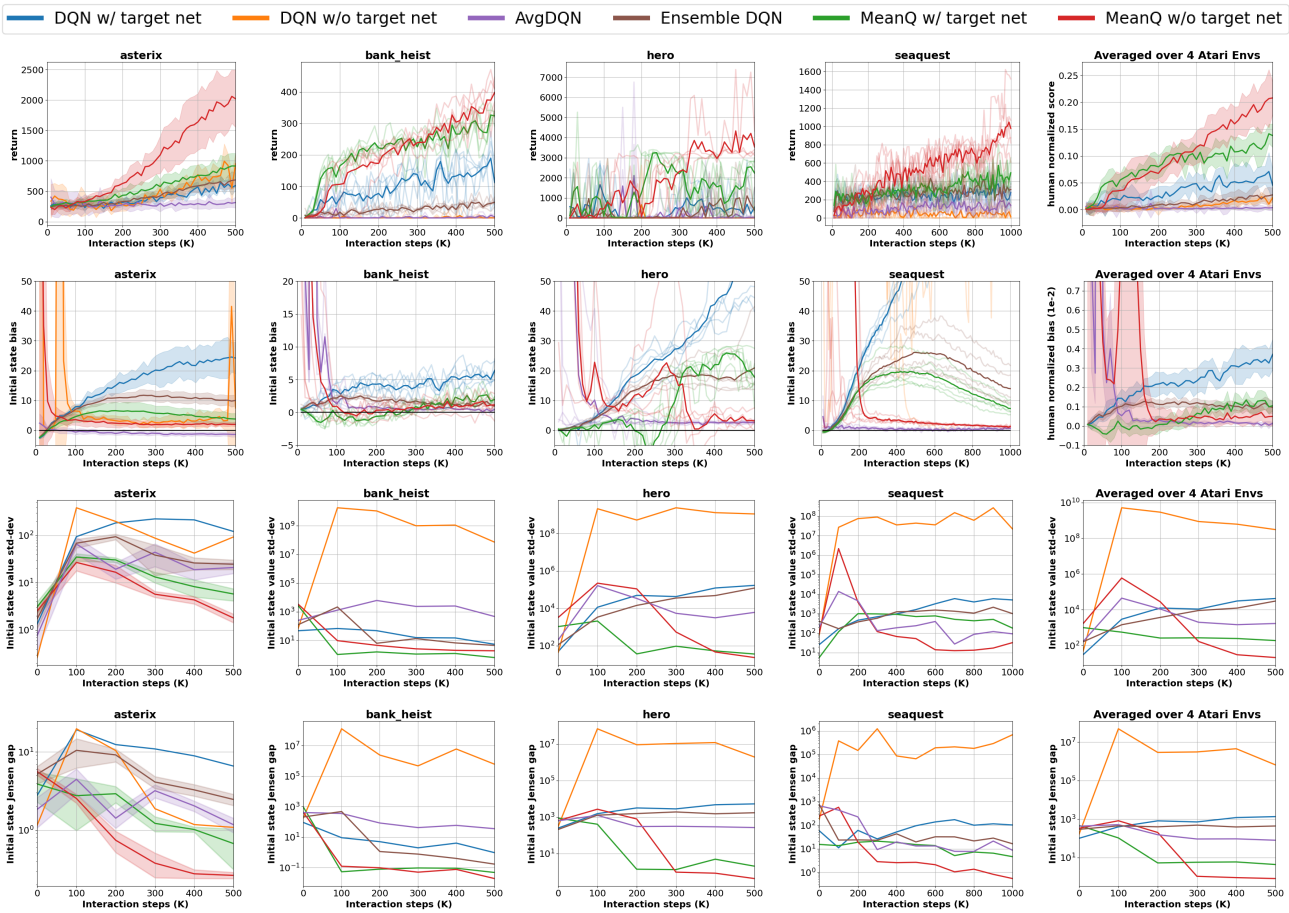

Figure 2. Performance, initial state estimation bias, standard deviation, and Jensen gap of DQN (with and without target network), Averaged-DQN, Ensemble-DQN, and MeanQ (with and without target network). Expected returns and biases, standard deviation, and Jensen gaps are over 5 runs; except in ensemble methods in the Asterix environment where 4 groups for 5 runs each are used to plot standard deviations (shaded; for details see text). Bias in DQN without target network is often above the plotted range. In MeanQ, estimation variance is reduced by the ensemble averaging, eliminating the need for a target network. See Section 6.3 for further analysis.

$V(s_0) = \max_a\{\text{mean}_k Q_{\theta_k}(s_0, a)\}$ is queried. We then report the empirical mean of $V(s_0) - \sum_t \gamma^t r_t$ over 20 evaluation trajectories after every 10K interactions.

3. **Estimation variance.** We measure the variance of value estimates by reporting the standard deviation of the initial state value $V(s_0)$ over 5 runs of each algorithm after every 100K interactions. Scaling and shifting algorithm performance such that random-policy performance is 0 and human-level is 1, the standard deviation is normalized by this standardized performance to obtain the relative standard deviation.[1] The relative standard deviation is averaged over 50 resets of s_0.
4. **Jensen gap.** We measure the Jensen gap $\mathbb{E}[\max_a Q(s_0, a)] - \max_a \mathbb{E}[Q(s_0, a)]$, with the expectation estimated empirically in 5 runs of each algorithm after every 100K interactions. The Jensen gap is normalized by standardized performance, and averaged over 50 resets of s_0.

[1] This normalization method is only approximate, since value estimates are discounted and performance is undiscounted.

The quantities above are evaluated in initial states s_0, because these are states that all algorithms must learn to evaluate in every run. Other states may not have the same reach probabilities across algorithms and runs, and comparing them is less indicative of the actual algorithmic behavior.

Figure 2 summarizes the empirical estimates of performance, bias, variance, and Jensen gap of 6 algorithms: DQN (Mnih et al., 2015), DQN without a target network (Mnih et al., 2013), Avg-DQN and Ens-DQN (Anschel et al., 2016), MeanQ, and MeanQ without a target network. Rows 1 and

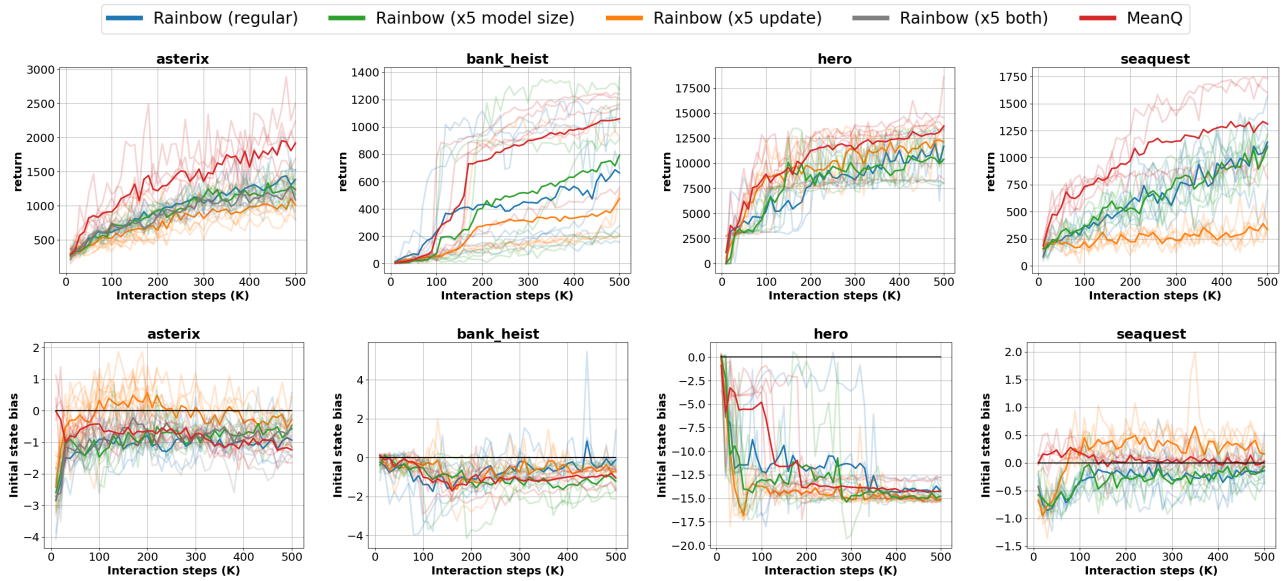

Figure 3. Performance of MeanQ with Rainbow techniques and UCB exploration, compared with several variations of Rainbow that resemble aspects of MeanQ: either a 5 times larger neural network, or 5 times more frequent network updates, or both in Asterix.

2 show the mean performance and bias over 5 runs, as well as all runs (faded); except in ensemble methods in the Asterix environment, where the mean and 1 standard deviation (shaded) over 20 runs are shown; and the average over 4 environments in the last column, where the mean and 1 standard deviation (shaded) are shown. Rows 3 and 4 show the standard deviation and Jensen gap over 5 runs; except in ensemble methods in Asterix, where the 20 runs are split into 4 groups and the mean and 1 standard deviation (shaded) of the within-group statistics are shown. The results suggest the following findings:

1. Across all methods and variations, higher estimation variance and higher Jensen gap are positively correlated.
2. Reproducing known results, a target network (Mnih et al., 2015) in DQN significantly reduces overestimation bias (often too high to plot) and estimation variance (van Hasselt et al., 2018).
3. MeanQ outperforms DQN, with and without a target network. MeanQ also has lower estimation variance, Jensen gap, and overestimation bias.
4. As in DQN, removing the target network in MeanQ tends to cause a mild increase in estimation variance, overestimation bias, and Jensen gap, and decrease in performance in early-stage training. However, in a reversal of the findings in DQN, MeanQ without a target network quickly overtakes MeanQ with a target network in terms of normalized variance and Jensen gap, which translates into a significant bias reduction and performance gain.
5. While Avg-DQN and, to a lesser extent, Ens-DQN reduce the variance and bias compared with DQN, this does not generally correspond to performance gain. We note that it is easier to keep bias low when performance is low, because bias is relative to returns and because values are easier to estimate when near-constant.
6. MeanQ with and without a target network does better than Ens-DQN in all four considered metrics (with the exception of bias with a target network). This demonstrates the importance of independent replay sampling and prioritization for learning neural network ensembles.

6.4. Ensembles vs. Larger Model and More Updates

In this experiment, we rule out two alternative hypotheses for the source of performance improvement in MeanQ. One could wonder whether the improvement could come from the ensemble being an effectively larger model or from performing more frequent value updates per interaction step. In this section, we empirically answer these questions in the negative.

Larger model. We experiment with Rainbow DQN with the same hyperparameters and a neural network architecture equivalent to the ensemble used in MeanQ. This network consists of 5 parallel networks, each taking input s and outputting $Q_{\theta_k}(s, a)$. The final layer is a fixed mean over $Q_{\theta_k}(s, a)$ with no additional parameters. The results in

Figure 3 show no significant improvement compared to Rainbow DQN with a standard network.

More frequent updates. We experiment with Rainbow DQN with the same hyperparameters, except with 5 times as many gradient steps per interaction step. The results in Figure 3 show, in all environments but in some more than others, a drop in performance, compared with the baseline hyperparameters, caused by performing gradient steps as frequently in Rainbow DQN as in MeanQ with 5 ensemble members. An experiment in the Asterix environment (Figure 3, left) also shows no benefit from combining both ablations.

These experiments reaffirm our motivation of reducing value estimate variance through ensemble averaging, and the significant role this plays in improving sample efficiency. MeanQ's improvements cannot be attributed to the larger model or more gradient updates.

7. Conclusion

In this paper, we introduce MeanQ, a simple ensemble RL method that uses the ensemble mean for target value estimates. Three key design choices, which set MeanQ apart from the extensive prior study of ensemble RL, are: (1) a shared exploration policy, likewise computed from the ensemble mean; (2) a shared replay buffer for all ensemble members; and (3) independent replay sampling from the buffer. While (1) and (2) can further correlate the ensemble members, beyond their use for each other's updates, we find that (1) is necessary to collect relevant experience, (2) helps experience diversity, and (3) is sufficient to maintain ensemble diversity. Averaging a sufficiently diverse ensemble reduces the variance of target value estimates, which in turn mitigates overestimation. Variance reduction in MeanQ also stabilizes training enough to obviate the lagging target network, which eliminates another source of estimation bias. Our experiment results show improved performance over record-holding baseline algorithms, as well as ablative versions.

Our empirical study furthers our understanding of the role of variance in value-based reinforcement learning, but much remains unknown about the interplay between variance, bias, exploration, and the deadly triad of off-policy bootstrapping with function approximation. Of particular interest is the necessity of nonlinear function approximation for the benefit of MeanQ to manifest, as learning in linear and tabular representations cannot benefit from a simple mean. We expect the community's further investigation to benefit from the simple and effective method proposed here, which can lend itself to more theoretical and empirical analyses, and serve as a stronger baseline for future work.

Acknowledgements

The authors would like to thank Sameer Singh and Kimin Lee for their help.

The research work of authors Dailin Hu and, partly, Roy Fox is funded by the Hasso Plattner Foundation.

References

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Anschel, O., Baram, N., and Shimkin, N. Deep reinforcement learning with averaged target DQN. *CoRR*, abs/1611.01929, 2016. URL http://arxiv.org/abs/1611.01929.

Asadi, K. and Littman, M. L. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pp. 243–252. PMLR, 2017.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

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

Chen, X., Wang, C., Zhou, Z., and Ross, K. W. Randomized Ensembled Double Q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Duan, J., Guan, Y., Ren, Y., Li, S. E., and Cheng, B. Addressing value estimation errors in reinforcement learning with a state-action return distribution function. *CoRR*, abs/2001.02811, 2020. URL http://arxiv.org/abs/2001.02811.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. *CoRR*, abs/1706.10295, 2017. URL http://arxiv.org/abs/1706.10295.

Fox, R. Toward provably unbiased temporal-difference value estimation. In *Optimization Foundations for Reinforcement Learning workshop (OPTRL @ NeurIPS)*, 2019.

Fox, R., Pakman, A., and Tishby, N. G-learning: Taming the noise in reinforcement learning via soft updates. *CoRR*, abs/1512.08562, 2015. URL http://arxiv.org/abs/1512.08562.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018b. URL http://arxiv.org/abs/1812.05905.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Jaakkola, T., Jordan, M. I., and Singh, S. P. On the convergence of stochastic iterative dynamic programming algorithms. *Neural computation*, 6(6):1185–1201, 1994.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Sepassi, R., Tucker, G., and Michalewski, H. Model-based reinforcement learning for atari. *CoRR*, abs/1903.00374, 2019. URL http://arxiv.org/abs/1903.00374.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018.

Kim, S., Asadi, K., Littman, M. L., and Konidaris, G. D. Deepmellow: Removing the need for a target network in deep q-learning. In *IJCAI*, 2019.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/c2073ffa77b5357a498057413bb09d3a-Paper.pdf.

Kumar, A., Gupta, A., and Levine, S. Discor: Corrective feedback in reinforcement learning via distribution correction. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18560–18572. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d7f426ccbc6db7e235c57958c21c5dfa-Paper.pdf.

Lan, Q., Pan, Y., Fyshe, A., and White, M. Maxmin q-learning: Controlling the estimation bias of q-learning, 2020.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Lee, K., Laskin, M., Srinivas, A., and Abbeel, P. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning, 2021.

Levine, S. Understanding the world through action. *CoRR*, abs/2110.12543, 2021. URL https://arxiv.org/abs/2110.12543.

Liang, L., Xu, Y., McAleer, S., Hu, D., Ihler, A., Abbeel, P., and Fox, R. Temporal-difference value estimation via uncertainty-guided soft updates. In *Deep Reinforcement Learning workshop (DRL @ NeurIPS)*, 2021.

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

Mahmood, A. R., Yu, H., White, M., and Sutton, R. S. Emphatic temporal-difference learning. *arXiv preprint arXiv:1507.01569*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

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. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016a. URL https://proceedings.neurips.cc/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf.

Osband, I., Roy, B. V., and Wen, Z. Generalization and exploration via randomized value functions. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2377–2386, New York, New York, USA, 20–22 Jun 2016b. PMLR. URL https://proceedings.mlr.press/v48/osband16.html.

Ostrovski, G., Castro, P. S., and Dabney, W. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 23283–23295, 2021.

Peer, O., Tessler, C., Merlis, N., and Meir, R. Ensemble bootstrapping for q-learning. In *International Conference on Machine Learning*, pp. 8454–8463. PMLR, 2021.

Rubin, J., Shamir, O., and Tishby, N. Trading value and information in mdps. In *Decision Making with Imperfect Decision Makers*, pp. 57–74. Springer, 2012.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Sheikh, H., Phielipp, M., and Boloni, L. Maximizing ensemble diversity in deep reinforcement learning. In *International Conference on Learning Representations*, 2022.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Song, Z., Parr, R. E., and Carin, L. Revisiting the softmax bellman operator: Theoretical properties and practical benefits. *CoRR*, abs/1812.00456, 2018. URL http://arxiv.org/abs/1812.00456.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Thrun, S. and Schwartz, A. Issues in using function approximation for reinforcement learning. In Mozer, M., Smolensky, P., Touretzky, D., Elman, J., and Weigend, A. (eds.), *Proceedings of the 1993 Connectionist Models Summer School*. Erlbaum Associates, June 1993.

Tsitsiklis, J. N. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

van Hasselt, H. Double q-learning. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning, 2015.

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018. URL http://arxiv.org/abs/1812.02648.

van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32:14322–14333, 2019.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pp. 1–5, 2019.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Wiering, M. A. and van Hasselt, H. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A. MeanQ with Rainbow Techniques and UCB Exploration

Results in Figures 1, 4, 5, and Tables 2, 3 are produced by the implementation of Algorithm 2 in https://github.com/indylab/MeanQ.

Algorithm 2 MeanQ with Rainbow techniques and UCB exploration

Initialize return distribution support z with L atoms spaced evenly from V_{min} to V_{max} by Δz
Initialize K parametrized return distribution p_{θ_k} for all k
Initialize replay memory D to capacity N
Initialize prioritization $p_k(i|D)$ for all k
for $t = 1, \ldots, T$ **do**
  Sample noisy network weights
  $a_t \leftarrow \arg\max_a \{\text{mean}_k\, Q_{\theta_k}(s,a) + \lambda\, \text{std}_k\, Q_{\theta_k}(s,a)\}$
  Observe $r_t \leftarrow r(s_t, a_t)$
  Sample $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
  Store transition (s_t, a_t, r_t) sequentially in D
  $s_t \leftarrow s_{t+1}$
  for $k = 1, \ldots, K$ **do**
    Sample B multi-step transitions $(s_b, a_b, r_b^{(0)}, ..., r_b^{(M-1)}, s_b^{(M)}) \sim D$
    $a^* = \arg\max_a \{\text{mean}_k\, z^\intercal p_{\theta_k}(s_b^{(M)}, a)\}$
    $p_{s_b^{(M)}} = \text{mean}_k\, p_{\theta_k}(s_b^{(M)}, a^*)$ (as a distribution vector over atoms)
    $c \leftarrow \vec{0}$
    $r_{\text{M-step}} = \sum_{m=0}^{M-1} \gamma^m r_b^{(m)}$
    for $j = 1, \ldots, L$ **do**
      $\hat{\mathcal{T}} z_j = \begin{cases} r_{\text{M-step}} & \text{if } s_b^{(M)} \text{ is a terminal state} \\ r_{\text{M-step}} + \gamma^M z_j & \text{otherwise} \end{cases}$
      $h_j = (\text{clip}[\hat{\mathcal{T}} z_j]_{V_{\text{min}}}^{V_{\text{max}}} - V_{\text{min}})/\Delta z$
      $l = \lfloor h_j \rfloor,\ u = \lceil h_j \rceil$
      $c_l \leftarrow c_l + p_{s_b^{(M)}}(u - h_j);\ c_u \leftarrow c_u + p_{s_b^{(M)}}(h_j - l)$
    end
    $\mathcal{L}(\theta_k) = \text{mean}_b\, \mathcal{L}_b(\theta_k)$, where $\mathcal{L}_b(\theta_k) = -c^\intercal \log p_{\theta_k}(s_b, a_b)$
    Update θ_k using $\nabla_{\theta_k} \mathcal{L}(\theta_k)$
    Update $p_k(b|D)$ using $\mathcal{L}_b(\theta_k)$
  end
end

B. Hyperparameters

The implementation of MeanQ is adapted from a publicly released implementation repository (https://github.com/Kaixhin/Rainbow). For results in Sections 6.2 and 6.4, our hyperparameters are the same as in Lee et al. (2021), except that no multi-step targets are used ($M = 1$ instead of 3 in Lee et al. (2021)). In Section 6.2, a target network is not used for MeanQ. MeanQ does not introduce additional hyperparameters. For results in Section 6.3, we list the modified hyperparameters compared with Section 6.2 in Table 1. The algorithms with "w/ target net" variant in Section 6.3 are using the same update frequency as in Section 6.2.

C. Performance of 26 Atari Environments

MeanQ is evaluated on the Atari Learning Environment (Bellemare et al., 2013). The evaluation policy is the greedy policy $\pi_\theta(s) = \arg\max_a \{\text{mean}_k\, Q_{\theta_k}(s,a)\}$. Evaluation returns after the first 100K interactions are recorded in Table 2 and Figure 4. Evaluation returns after the first 500K interactions are recorded in Table 3 and Figure 5.

Table 1. Hyperparameter of Vanilla MeanQ in Section 6.3

Hyperparameter	Value
Network architecture	same as Wang et al. (2016)
Learning rate α	6.25×10^{-5}
Epsilon greedy ϵ	Linear annealing from 1 to 0.1 from $t = 0$ to 200K
Interactions per gradient update	Seaquest: 3 ; Asterix, BankHeist, and Hero: 2
Multi-step (M)	1

Table 2. Performance at 100K interaction steps. The results for MeanQ show the average score and standard deviation (in parenthesis) of 5 runs. For Random, Rainbow ("Data-Efficient"), and SUNRISE, we cite the numbers reported in Lee et al. (2021). For PPO (Schulman et al., 2017), SimPLe (Kaiser et al., 2019), CURL (Laskin et al., 2020), and DrQ (Kostrikov et al., 2021), we cite the numbers reported in Kaiser et al. (2019). Best performance for each environment is shown in bold.

Environment	Random	PPO		SimPLe		Rainbow	CURL	DrQ	SUNRISE	MeanQ		Human
Alien	184.8	291.0	(40.3)	616.9	(252.2)	789.0	558.2	761.4	**872.0**	864.8	(309.7)	7128.0
Amidar	11.8	56.5	(20.8)	74.3	(28.3)	118.5	**142.1**	97.3	122.6	112.5	(28.3)	1720.0
Assault	233.7	424.2	(55.8)	527.2	(112.3)	413.0	600.6	489.1	594.8	**894.4**	(120.7)	742.0
Asterix	248.8	385.0	(104.4)	**1128.3**	(211.8)	533.3	734.5	637.5	755.0	971.5	(246.7)	8503.0
BankHeist	15.0	16.0	(12.4)	34.2	(29.2)	97.7	131.6	196.6	**266.7**	215.6	(282.4)	753.0
BattleZone	2895.0	5300.0	(3655.1)	4031.2	(1156.1)	7833.3	14870.0	13520.6	**15700.0**	13980.0	(3225.0)	37188.0
Boxing	0.3	-3.9	(6.4)	7.8	(10.1)	0.6	1.2	6.9	6.7	**22.2**	(9.2)	12.0
Breakout	0.9	5.9	(3.3)	**16.4**	(6.2)	2.3	4.9	14.5	1.8	13.0	(6.2)	30.0
ChopperCommand	671.0	730.0	(199.0)	979.4	(172.7)	590.0	1058.5	646.6	1040.0	**1118.8**	(330.5)	7388.0
CrazyClimber	7339.5	18400.0	(5275.1)	62583.6	(16856.8)	25426.7	12146.5	19694.1	22230.0	**76490.0**	(15610.2)	35829.0
DemonAttack	140.0	192.5	(83.1)	208.1	(56.8)	688.2	817.6	1222.2	919.8	**1636.8**	(287.6)	1971.0
Freeway	0.0	8.0	(9.8)	16.7	(15.7)	28.7	26.7	15.4	30.2	**30.4**	(0.9)	30.0
Frostbite	74.0	214.0	(10.2)	65.2	(31.5)	1478.3	1181.3	449.7	**2026.7**	1557.8	(1214.6)	4334.7
Gopher	245.9	246.0	(103.3)	596.8	(183.5)	348.7	669.3	598.4	654.7	**956.0**	(207.5)	2412.0
Hero	224.6	569.0	(1100.9)	2656.6	(483.1)	3675.7	6279.3	4001.6	**8072.5**	6893.2	(1192.8)	30826.0
Jamesbond	29.2	65.0	(46.4)	100.5	(36.8)	300.0	**471.0**	272.3	390.0	466.2	(80.1)	303.0
Kangaroo	42.0	140.0	(102.0)	51.2	(17.8)	1060.0	872.5	1052.4	2000.0	**5714.0**	(4012.9)	3035.0
Krull	1543.3	3750.4	(3071.9)	2204.8	(776.5)	2592.1	**4229.6**	4002.3	3087.2	3913.4	(185.7)	2666.0
KungFuMaster	616.5	4820.0	(983.2)	14862.5	(4031.6)	8600.0	14307.8	7106.4	10306.7	**17232.0**	(8019.1)	22736.0
MsPacman	235.2	496.0	(379.8)	1480.0	(288.2)	1118.7	1465.5	1065.6	**1482.3**	1032.0	(164.5)	6952.0
Pong	-20.4	-20.5	(0.6)	**12.8**	(17.2)	-19.0	-16.5	-11.4	-19.3	-15.4	(4.0)	15.0
PrivateEye	26.6	10.0	(20.0)	35.0	(60.2)	97.8	**218.4**	49.2	100.0	100.0	(0.0)	69571.0
Qbert	166.1	362.5	(117.8)	1288.8	(1677.9)	646.7	1042.4	1100.9	**1830.8**	1352.2	(571.9)	13455.0
RoadRunner	0.0	1430.0	(760.0)	5640.6	(3936.6)	9923.3	5661.0	8069.8	11913.3	**18012.5**	(7229.0)	7845.0
Seaquest	61.1	370.0	(103.3)	683.3	(171.2)	396.0	384.5	321.8	570.7	**721.8**	(62.4)	42055.0
UpNDown	488.4	2874.0	(1105.8)	3350.3	(3540.0)	3816.0	2955.2	3924.9	**5074.0**	2982.5	(560.9)	11693.0

Table 3. Performance at 500K interaction steps. The results for MeanQ show the average score and standard deviation (in parenthesis) of 5 runs. For PPO (Schulman et al., 2017) and Rainbow ("Canonical Rainbow") (Hessel et al., 2018) we cite the numbers reported in Kaiser et al. (2019). Best performance for each environment is shown in bold.

Environment	Random	PPO		Rainbow		MeanQ		Human
Alien	227.8	269.0	(203.4)	828.6	(54.2)	**1490.7**	(393.1)	7128.0
Amidar	5.8	93.2	(36.7)	194.0	(34.9)	**274.1**	(57.7)	1720.0
Assault	233.7	552.3	(110.4)	1041.5	(92.1)	**1522.4**	(131.1)	742.0
Asterix	210.0	1085.0	(354.8)	1702.7	(162.8)	**2234.5**	(419.2)	8503.0
BankHeist	14.2	641.0	(352.8)	727.3	(198.3)	**1060.2**	(255.0)	753.0
BattleZone	2360.0	14400.0	(6476.1)	19507.1	(3193.3)	**20770.0**	(2127.6)	37188.0
Boxing	0.1	3.5	(3.5)	**58.2**	(16.5)	36.8	(8.7)	12.0
Breakout	1.7	**66.1**	(114.3)	26.7	(2.4)	14.9	(5.6)	30.0
ChopperCommand	811.0	860.0	(285.3)	1765.2	(280.7)	**1816.2**	(322.0)	7388.0
CrazyClimber	10780.5	33420.0	(3628.3)	75655.1	(9439.6)	**107003.0**	(11795.9)	35829.0
DemonAttack	152.1	216.5	(96.2)	3642.1	(478.2)	**3988.2**	(341.7)	1971.0
Freeway	0.0	14.0	(9.8)	12.6	(15.4)	**32.6**	(0.2)	30.0
Frostbite	65.2	214.0	(10.2)	1386.1	(321.7)	**3110.0**	(1426.4)	4334.7
Gopher	257.6	560.0	(118.8)	1640.5	(105.6)	**2223.4**	(869.4)	2412.0
Hero	1027.0	1824.0	(1461.2)	10664.3	(1060.5)	**11928.8**	(2596.1)	30826.0
Jamesbond	29.0	255.0	(101.7)	429.7	(27.9)	**640.0**	(261.8)	303.0
Kangaroo	52.0	340.0	(407.9)	970.9	(501.9)	**13263.0**	(1542.6)	3035.0
Krull	1598.0	3056.1	(1155.5)	4139.4	(336.2)	**5923.9**	(352.7)	2666.0
KungFuMaster	258.5	17370.0	(10707.6)	**19346.1**	(3274.4)	18932.0	(6543.1)	22736.0
MsPacman	307.3	306.0	(70.2)	**1558.0**	(248.9)	1536.0	(304.5)	6952.0
Pong	-20.7	-8.6	(14.9)	19.9	(0.4)	**20.0**	(1.1)	15.0
PrivateEye	24.9	20.0	(40.0)	-6.2	(89.8)	**120.0**	(40.0)	69571.0
Qbert	163.9	757.5	(78.9)	4241.7	(193.1)	**10245.5**	(2640.8)	13455.0
RoadRunner	11.5	5750.0	(5259.9)	18415.4	(5280.0)	**38992.5**	(2890.7)	7845.0
Seaquest	68.4	692.0	(48.3)	**1558.7**	(221.2)	1331.8	(337.5)	42055.0
UpNDown	533.4	**12126.0**	(1389.5)	6120.7	(356.8)	8051.6	(2730.7)	11693.0

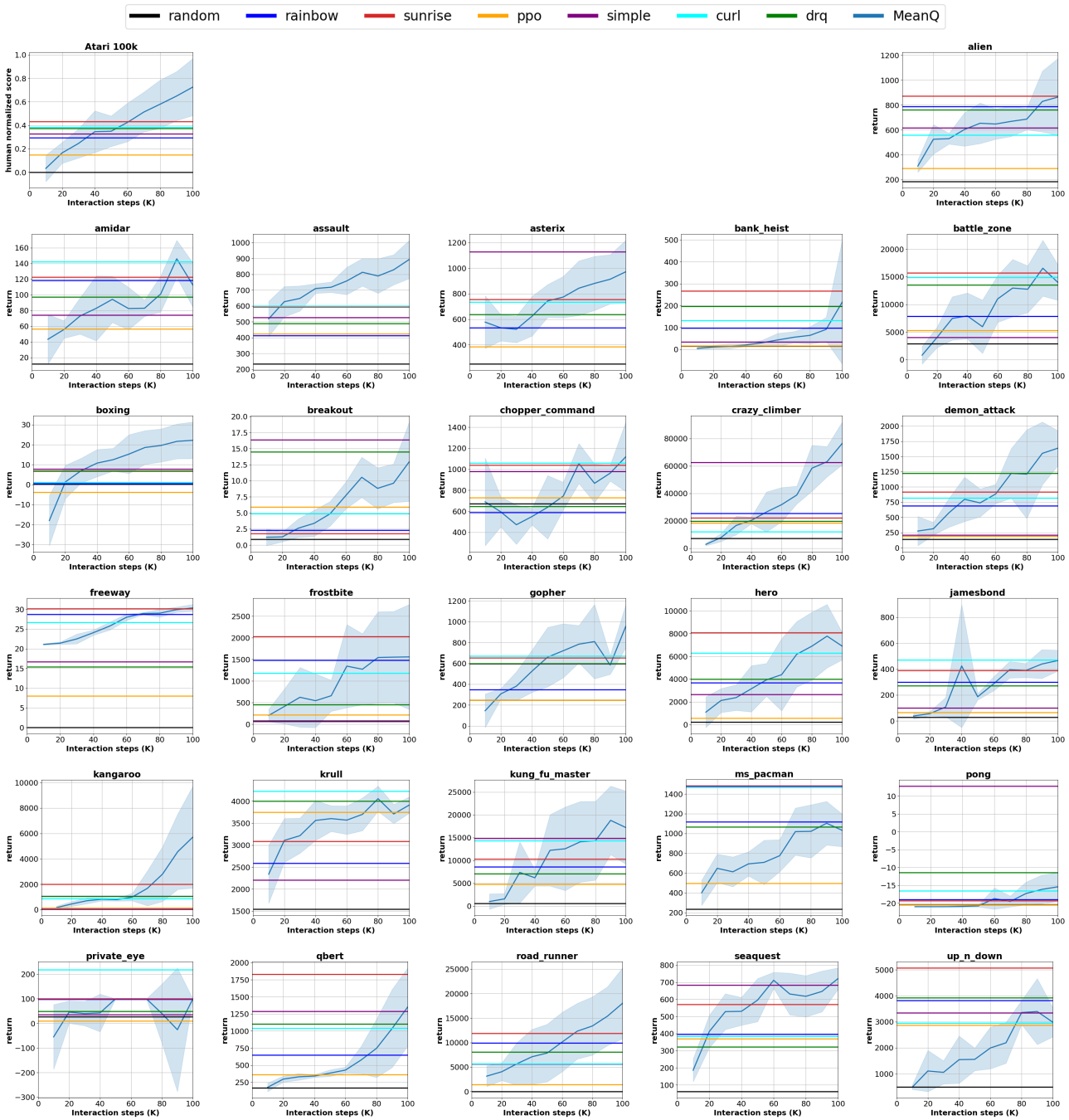

Figure 4. Comparison with baseline algorithms in 26 Atari environments at 100K interactions. The results for MeanQ show the average score and standard deviation (shaded) over 5 runs. Cited scores for baseline algorithms at 100K interactions are rendered as horizontal lines. For Random, Rainbow ("Data-Efficient") (van Hasselt et al., 2019), and SUNRISE (Lee et al., 2021), we cite the numbers reported in (Lee et al., 2021). For PPO (Schulman et al., 2017), SimPLe (Kaiser et al., 2019), CURL (Laskin et al., 2020), and DrQ (Kostrikov et al., 2021), we cite the numbers reported in Kaiser et al. (2019).

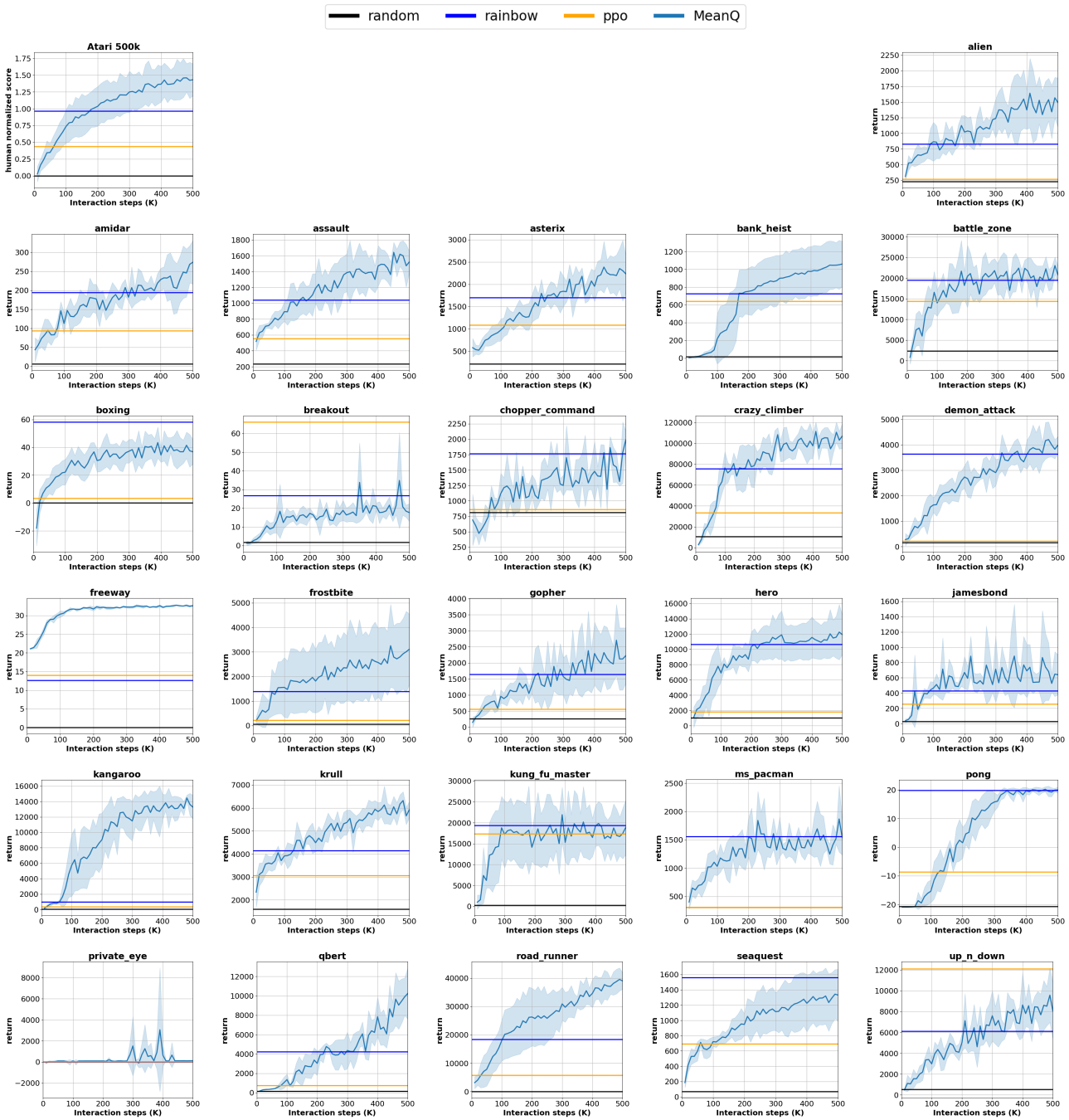

Figure 5. Comparison with baseline algorithms in 26 Atari environments at 500K interactions. The results for MeanQ show the average score and standard deviation (shaded) over 5 runs. Cited scores for baseline algorithms at 500K interactions are rendered as horizontal lines. For PPO (Schulman et al., 2017) and Rainbow ("Canonical Rainbow") (Hessel et al., 2018), we cite the numbers reported in Kaiser et al. (2019).