RANDOMIZED ENSEMBLED DOUBLE Q-LEARNING: LEARNING FAST WITHOUT A MODEL

Anonymous authors
Paper under double-blind review

A THEORETICAL RESULTS

A.1 TABULAR VERSION OF REDQ

In the tabular algorithm below, for clarity we use $G = 1$.

Algorithm 1 Tabular REDQ

Initialize $\left\{Q^i(s,a), s \in \mathcal{S}, a \in \mathcal{A}\right\}_{i=1}^N$,
repeat
  Choose $a \in \mathcal{A}$ based on $\left\{Q^i(s,a)\right\}_{i=1}^N$, observe r, s'
  Randomly choose a subset $\mathcal{M}$ of size M from $\left\{1,..,N\right\}$
  $y = r + \gamma \max_{a' \in \mathcal{A}} \min_{j \in \mathcal{M}} Q^j(s',a')$
  for $i = 1, \ldots, N$ **do**:
    $Q^i(s,a) \leftarrow Q^i(s,a) + \alpha(y - Q^i(s,a))$
  $s \leftarrow s'$
until end

Alternatively in Algorithm 2, at each iteration we could just update one of the Q^i functions.

A.2 PROOF OF THEOREM 1

We first prove the following lemma:

Lemma 1. *Let $X_1, X_2, \ldots$ be an infinite sequence of i.i.d. random variables. Let $F(x)$ be the cdf of X_m and let $\tau = \inf\{x : F(x) > 0\}$. Also let $Y_m = \min\{X_1, X_2, \ldots, X_m\}$. Then $Y_1, Y_2, \ldots$ converges to τ almost surely.*

Proof:

Let $F_m(x)$ be the cdf of Y_m. Since $X_1, ..., X_m$ are independent,

$$F_m(x) = 1 - [1 - F(x)]^m$$

For $x < \tau$, $F_m(x) = 0$ since $F(x) = 0$. For $x > \tau$, $F_m(x) \xrightarrow{m \to \infty} 1$. Therefore, Y_m weakly converges to τ.

Moreover, for each $\omega \in \Omega$, $\{Y_m(\omega)\}$ is a decreasing sequence. So $\{Y_m(\omega)\}$ either converges to a real number or $-\infty$. Therefore $Y_m \to Y$ almost surely for some random variable Y. Combined with the result that $Y_m \xrightarrow{d} \tau$, we can conclude that

$$Y_m \xrightarrow{a.s.} \tau$$

Proof of Theorem 1:

1. Let B_1, B_2 be two subsets of $\mathcal{N} = \left\{1, ..., N\right\}$ of size M. First of all, since $\{Q^j(s,a)\}_{j=1}^N$ are i.i.d for any $a \in \mathcal{A}$, $\min_{j \in B_1} Q^j(s,a)$ and $\min_{j \in B_2} Q^j(s,a)$ are identically distributed. Furthermore, since $Q^j(s,a)$ are independent for all $a \in \mathcal{A}$ and $1 \leq j \leq N$, $\left\{\min_{j \in B} Q^j(s,a)\right\}_{a \in \mathcal{A}}$ are

independent for any $B \subset \mathcal{N}$. Denote the distribution function of $\max_a \min_{j\in B_1} Q^j(s,a)$ as $F_1(x)$ and the distribution function of $\max_a \min_{j\in B_2} Q^j(s,a)$ as $F_2(x)$. Then for any $x \in \mathbb{R}$,

$$\begin{aligned} F_1(x) &= \mathbb{P}\big(\max_a \min_{j\in B_1} Q^j(s,a) \le x\big) = \mathbb{P}\big(\cap_{a\in\mathcal{A}} \{\min_{i\in B_1} Q^j(s,a) \le x\}\big) \\ &= \prod_{a\in\mathcal{A}} \mathbb{P}\big(\min_{j\in B_1} Q^j(s,a) \le x\big) = \prod_{a\in\mathcal{A}} \mathbb{P}\big(\min_{j\in B_2} Q^j(s,a) \le x\big) \\ &= \mathbb{P}\big(\max_a \min_{j\in B_2} Q^j(s,a) \le x\big) = F_2(x) \end{aligned}$$

Therefore, we have proved that $\max_a \min_{j\in B_1} Q^j(s,a)$ and $\max_a \min_{j\in B_2} Q^j(s,a)$ are identically distributed. Then

$$\begin{aligned} \mathbb{E}\big[Z_{M,N}\big] &= \gamma\mathbb{E}\big[(\max_a \min_{j\in\mathcal{M}} Q^j(s,a) - \max_a Q^\pi(s,a))\big] \\ &= \gamma\mathbb{E}\big[\frac{1}{\binom{N}{M}} \sum_{\substack{B\subset\mathcal{N} \\ |B|=M}} \max_a \min_{j\in B} Q^j(s,a)\big] - \gamma \max_a Q^\pi(s,a) \\ &= \gamma\bigg(\mathbb{E}\big[\max_a \min_{1\le j\le M} Q^j(s,a)\big] - \max_a Q^\pi(s,a)\bigg) \end{aligned}$$

which does not depend on N.

2. It follows from 1 that

$$\mathbb{E}\big[Z_{1,N}\big] = \gamma\bigg(\mathbb{E}\big[\max_a Q^1(s,a)\big] - \max_a Q^\pi(s,a)\bigg)$$

Since $\max_a Q^1(s,a) \ge Q^1(s,a')$ for all $a' \in \mathcal{A}$, we have

$$\mathbb{E}\big[\max_a Q^1(s,a)\big] \ge \mathbb{E}\big[Q^1(s,a')\big]$$

for all $a' \in \mathcal{A}$. Consequently,

$$\mathbb{E}\big[\max_a Q^1(s,a)\big] \ge \max_a \mathbb{E}\big[Q^1(s,a)\big] = \max_a Q^\pi(s,a)$$

Hence

$$\mathbb{E}\big[Z_{1,N}\big] = \gamma\bigg(\mathbb{E}\big[\max_a Q^1(s,a)\big] - \max_a Q^\pi(s,a)\bigg) \ge 0$$

3. Since $\max_a \min_{1\le j\le M} Q^j(s,a) \ge \max_a \min_{1\le j\le M+1} Q^j(s,a)$,

$$\begin{aligned} \mathbb{E}\big[Z_{M,N}\big] &= \gamma\bigg(\mathbb{E}\big[\max_a \min_{1\le j\le M} Q^j(s,a)\big] - \max_a Q^\pi(s,a)\bigg) \\ &\ge \gamma\bigg(\mathbb{E}\big[\max_a \min_{1\le j\le M+1} Q^j(s,a)\big] - \max_a Q^\pi(s,a)\bigg) = \mathbb{E}\big[Z_{M+1,N}\big] \end{aligned}$$

4. Let $F_a(x)$ be the cdf of $Q^j(s,a)$ and let $\tau_a = \inf\{x : F_a(x) > 0\}$. Here we assume the approximation error e^i_{sa} is non-trivial, which implies $\tau_a < Q^\pi(s,a)$. Note that τ_a can be equal to $-\infty$. Let

$$Y^M_a = \min_{1\le i\le M} Q^j(s,a)$$

From Lemma 1 we have Y^M_a converges to τ_a almost surely for each a. Because the action space is finite, it therefore follows that

$$Y^M = \max_a Y^M_a$$

converges almost surely to $\tau = \max_a \tau_a$. Furthermore, for each a we have

$$Y^M_a = \min_{1\le j\le M} Q^j(s,a) \ge \min_{1\le j\le M+1} Q^j(s,a) = Y^{M+1}_a$$

from which it follows that $Y^M \ge Y^{M+1}$. Thus $\{Y^M\}$ is a monotonically decreasing sequence. We also note that due to the assumption $e^i_{sa} \le c$ for all a and i, and because $Q^\pi(s,a)$ is finite

for all s and a, it follows that $Y^M \leq d$ for all M for a finite d. Thus $\{Y^M\}$ is a bounded-above, monotonically-decreasing sequence of random variables which converges almost surely to τ. We can therefore apply the monotone convergence theorem, giving

$$\begin{aligned}\mathbb{E}\big[Z_{M,N}\big] &= \gamma\bigg(\mathbb{E}\big[\max_a \min_{1\leq j\leq M} Q^j(s,a)\big] - \max_a Q^\pi(s,a)\bigg) \\ &= \gamma\bigg(\mathbb{E}[\max_a Y_a^M] - \max_a Q^\pi(s,a)\bigg) \xrightarrow{M\to\infty} \gamma\bigg(\max_a \tau_a - \max_a Q^\pi(s,a)\bigg) < 0,\end{aligned}$$

where the last inequality follows from $\tau_a < Q^\pi(s,a)$ for all actions a.

A.3 Proof of Theorem 2

For convenience, define $Y_B = \max_{a'} \min_{j\in B} Q^j(s',a')$ for any subset $B \subset \mathcal{N}$. The distribution of Y_B only depends on the cardinality of B; for a fixed M, let $v_M \triangleq \mathrm{Var}(Y_B)$ where $|B| = M$.

Proposition 1.

$$\mathrm{Var}(Y_{M,N}) \leq G_M(N)$$

for some function $G_M(N)$ satisfying

$$\lim_{N\to\infty} \frac{G_M(N)}{M^2 v_M/N} = 1$$

Consequently,

$$\lim_{N\to\infty} \mathrm{Var}(Y_{M,N}) = 0$$

Proof. Suppose $N > 2M$.

$$\begin{aligned}\mathrm{Var}(Y_{M,N}) &= \frac{\gamma^2}{\binom{N}{M}^2}\mathrm{Var}\Big(\sum_{\substack{B\subset\mathcal{N}\\|B|=M}} Y_B\Big) \\ &= \frac{\gamma^2 (M!)^2}{\big(\Pi_{i=0}^{M-1}(N-i)\big)^2}\Big[\sum_{B\subset\mathcal{N}} \mathrm{Var}(Y_B) + 2\cdot \sum_{\substack{B_1,B_2\subset\mathcal{N}\\B_1\neq B_2}} \mathrm{Cov}(Y_{B_1},Y_{B_2})\Big]\end{aligned}$$

Let $A = \sum_{\substack{B_1,B_2\subset\mathcal{N}\\B_1\neq B_2}} \mathrm{Cov}(Y_{B_1},Y_{B_2})$, which consists of

$$\begin{aligned}\binom{\binom{N}{M}}{2} &= \frac{1}{2}\cdot\frac{N!}{(N-M)!M!}\cdot\left(\frac{N!}{(N-M)!M!}-1\right) \\ &= \frac{1}{2(M!)^2}\cdot\Pi_{i=0}^{M-1}(N-i)^2 - \frac{N!}{2\cdot M!(N-M)!}\end{aligned}$$

terms. $\binom{\binom{N}{M}}{2}$ can be seen as a polynomial function of N with degree $2M$. The coefficient for the term N^{2M} is $\frac{1}{2(M!)^2}$. The coefficient for the term N^{2M-1} is $\frac{1}{2(M!)^2}\cdot(-2\sum_{i=0}^{M-1} i)$.

Note that Y_{B_1} and Y_{B_2} are independent if $B_1 \cap B_2 = \varnothing$. The total number of different pairs (B_1, B_2) such that $B_1 \cap B_2 = \varnothing$ is

$$\binom{N}{2M}\cdot\binom{2M}{M}\cdot\frac{1}{2} = \frac{1}{2(M!)^2}\cdot\frac{N!}{(N-2M)!} = \frac{1}{2(M!)^2}\cdot\Pi_{i=0}^{2M-1}(N-i)$$

This is again a polynomial function of N with degree $2M$. The coefficient of the term N^{2M} is $\frac{1}{2(M!)^2}$. The coefficient of the term N^{2M-1} is $\frac{1}{2(M!)^2}\cdot(-\sum_{i=0}^{2M-1} i)$. So the number of non-zero terms in A is at most

$$\begin{aligned}&\frac{1}{2(M!)^2}\cdot\Pi_{i=0}^{M-1}(N-i)^2 - \frac{N!}{2\cdot M!(N-M)!} - \frac{1}{2(M!)^2}\cdot\Pi_{i=0}^{2M-1}(N-i) \\ =&\frac{M^2}{2(M!)^2}\cdot N^{2M-1} + O(N^{2M-2})\end{aligned}$$

Moreover, by Cauchy-Schwarz inequality, for any $B_1, B_2 \subset \mathcal{N}$

$$\mathrm{Cov}(Y_{B_1}, Y_{B_2}) \leq \sqrt{\mathrm{Var}(Y_{B_1}) \cdot \mathrm{Var}(Y_{B_2})} = v_M$$

Therefore,

$$A \leq [\frac{M^2}{2(M!)^2} \cdot N^{2M-1} + O(N^{2M-2})] v_M$$

which implies

$$\begin{aligned}
\mathrm{Var}(Y_{M,N}) &= \frac{\gamma^2 (M!)^2}{\left(\Pi_{i=0}^{M-1}(N-i)\right)^2} \Big[\sum_{B \subset \mathcal{N}} \mathrm{Var}(Y_B) + 2 \cdot \sum_{\substack{B_1, B_2 \subset \mathcal{N} \\ B_1 \neq B_2}} \mathrm{Cov}(Y_{B_1}, Y_{B_2}) \Big] \\
&= \frac{\gamma^2 (M!)^2}{\left(\Pi_{i=0}^{M-1}(N-i)\right)^2} \Big[\sum_{B \subset \mathcal{N}} \mathrm{Var}(Y_B) + 2A \Big] \\
&\leq \gamma^2 \big[M^2 \cdot \frac{N^{2M-1}}{\Pi_{i=0}^{M-1}(N-i)^2} + O(\frac{1}{N^2}) \big] v_M \xrightarrow{N \to \infty} 0
\end{aligned}$$

Moreover,

$$\lim_{N \to \infty} \frac{M^2 v_M / N}{\big[M^2 \cdot \frac{N^{2M-1}}{\Pi_{i=0}^{M-1}(N-i)^2} + O(\frac{1}{N^2}) \big] v_M} = \lim_{N \to \infty} \frac{\Pi_{i=0}^{M-1}(N-i)^2}{N^{2M}} = 1$$

□

A.4 Proof of convergence of tabular REDQ

Assuming that the step size satisfies the standard Robbins-Monro conditions, it is easily seen that the tabular version of REDQ converges with probability 1 to the optimal Q function. In fact, for our Weighted scheme, where we take the expectation over all sets of size M, the convergence conditions in Lan et al. (2020) are fully satisfied.

For the randomized case, only very minor changes are needed in the proof in Lan et al. (2020). Note that in the case of REDQ, the underlying deterministic target is:

$$F\left(Q^1, Q^2, \ldots, Q^N\right)(s,a) = r(s,a) + \gamma \sum_{s'} p\left(s' \mid s, a\right) \sum_{\substack{B \subset \mathcal{N} \\ |B| = M}} \frac{1}{\binom{N}{M}} \max_{a'} \min_{j \in B} Q^j(s', a')$$

Let $\mathcal{T}$ be the operator that concatenates N identical copies of F, so that $\mathcal{T}$: $\mathbb{R}^{S \times A \times N} \to \mathbb{R}^{S \times A \times N}$ where S and A are the cardinalities of the state and action spaces, respectively. It is easy to show that the operator $\mathcal{T}$ is a contraction with the l_∞ norm. The stochastic approximation noise term is given by

$$\omega(s,a) = R - r(s,a) + \gamma \Big[\max_{a'} \min_{j \in B} Q^j(s', a') - \sum_{s'} p\left(s' \mid s, a\right) \sum_{\substack{B \subset \mathcal{N} \\ |B| = M}} \frac{1}{\binom{N}{M}} \max_{a'} \min_{j \in B} Q^j(s', a') \Big]$$

It is straightforward to show

$$\mathbb{E}\left[\omega^2(s,a) \mid \mathcal{F}_{\text{past}}\right] \leq \mathrm{Var}\left(R \mid S = s, A = a\right) + \max_{1 \leq i \leq N} \max_{s', a'} \left(Q^i(s', a')\right)^2 \tag{1}$$

As in Lan et al. (2020), it follows from the contraction property and (1) that REDQ converges with probability 1 to the optimal Q function (Tsitsiklis, 1994; Bertsekas & Tsitsiklis, 1996).

B HYPERPARAMETERS AND IMPLEMENTATION DETAILS

Since MBPO builds on top of a SAC agent, to make our comparisons fair, meaningful, and consistent with previous work, we make all SAC related hyperparameters exactly the same as used in the MBPO paper (Janner et al., 2019). Table 1 gives a list of hyperparameter used in the experiments. For all the REDQ curves reported in the results section, we use a Q network ensemble size N of 10. We use a UTD ratio G of 20 on the four MuJoCo environments, which is the same value that was used in the MBPO paper. Thus most of the hyperparameters are made to be the same as in the MBPO paper to ensure fairness and consistency in comparisons.

For all the algorithms and variants, we also first obtain 5000 data points by randomly sampling actions from the action space without making parameter updates. In our experiments we found that using a high UTD from the very beginning with a very small amount of data can easily lead to complete divergence on SAC-20. Sampling a number of random datapoints at the start of training is also a common technique that has been used in previous model-free as well as model-based works (Haarnoja et al., 2018; Fujimoto et al., 2018; Janner et al., 2019).

Table 1: REDQ hyperparameters

Parameter	Value
Shared	
optimizer	Adam (Kingma & Ba, 2014)
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
target smoothing coefficient (ρ)	0.005
replay buffer size	10^6
number of hidden layers for all networks	2
number of hidden units per layer	256
mini-batch size	256
nonlinearity	ReLU
random starting data	5000
REDQ	
ensemble size N	10
in-target minimization parameter M	2
UTD ratio G	20
OFENet	
random starting data	20,000
OFENet number of pretraining updates	100,000
OFENet UTD ratio	4

For the REDQ-OFE experiments, we implemented a minimal version of the OFENet in the original paper, with no batchnorm layers. We use the recommended hyperparameters as described in the original paper (Ota et al., 2020). Compared to REDQ without OFENet, the main difference is we now first collect 20,000 random data points (which is accounted for in the training curves), and then pre-train the OFENet for 100,000 updates, with the same learning rate and batch size. We then train OFENet together with REDQ agent, and the OFENet uses a UTD ratio of 4. We tried a simple hyperparameter search on Ant with 200,000, 100,000 and 50,000 pre-train updates, and learning rates of 1e-4, 3e-4, 5e-4, and a OFENet UTD of 1, 4 and 20. However, the results are not very different. It is possible that better results can be obtained through a more extensive hyperparameter search or other modifications.

B.1 EFFICIENT CALCULATION OF TARGET FOR WEIGHTED VERSION OF REDQ

In section 4 we provided experimental results for the Weighted version of REDQ. Recall that in this version, instead of sampling a random subset $\mathcal{M}$ in the target, we average over all subsets B in

$\{1, \ldots, N\}$ of size M:

$$y = r + \gamma \frac{1}{\binom{N}{M}} \sum_{B} \left(\min_{i \in B} Q_{\phi_{\text{targ},i}}\left(s', \tilde{a}'\right) \right)$$

In practice, however, we do not need to sum over all N choose M subsets. Instead we can re-order the indices so that

$$Q_{\phi_{\text{targ},i}}\left(s', \tilde{a}'\right) \leq Q_{\phi_{\text{targ},i+1}}\left(s', \tilde{a}'\right)$$

for $i = 1, \ldots, N-1$. After the re-ordering, we can use the identity:

$$\frac{1}{\binom{N}{M}} \sum_{\mathcal{M} \subset \mathcal{N}} \min_{i \in \mathcal{M}} Q_i(s', a') = \frac{1}{\binom{N}{M}} \sum_{i=1}^{N-M+1} \binom{N-i}{M-1} Q_{\phi_{\text{targ},i}}\left(s', \tilde{a}'\right)$$

C SAMPLE EFFICIENCY COMPARISON FOR REDQ, SAC AND MBPO

The sample efficiency claims made in the main paper are based on Table 2 and Table 3. Table 2 shows that compared to naive SAC, REDQ is much more sample efficient. REDQ reaches 3500 on Hopper with 8x sample efficiency, and reaches 5000 for Ant and Humanoid with 5x and 3.7x sample efficiency. After adding OFE, this becomes more than 7x on Ant and Humanoid. If we average all the numbers for the four environments, then REDQ is 5.0x as sample efficient, and 6.4x after including OFE results.

Table 3 compares REDQ to SAC and MBPO. As in the MBPO paper, we train for 125K for Hopper, and 300K for the other three environments (Janner et al., 2019). The numbers in Table 3 show the performance when trained to half and to the full length of the MBPO training limits. When averaging the numbers, we see that REDQ reaches 4.5x and 2.1x the performance of SAC at 150K and 300K. REDQ is also stronger than MBPO, with 1.4x and 1.1x the performance of MBPO at 150K and 300K. If we include the results of REDQ-OFE, then the numbers become 5.5x and 2.3x the SAC performance at 150K and 300K, and 1.8x and 1.2x the MBPO performance at 150 and 300K.

Table 2: Sample efficiency comparison of SAC and REDQ. The numbers show the amount of data collected when the specified performance level is reached. The last two columns show how many times REDQ and REDQ-OFE are more sample efficient than SAC in reaching that performance.

Score	SAC	REDQ	REDQ-OFE	REDQ faster	REDQ-OFE faster
Hopper at 3500	933K	116K	-	8.04	-
Walker2d at 3500	440K	141K	-	3.12	-
Ant at 5000	771K	153K	106K	5.04	7.27
Humanoid at 5000	945K	255K	134K	3.71	7.05

Table 3: Performance comparison of REDQ, REDQ-OFE, MBPO and SAC. The numbers show the performance achieved when the specific amount of data is collected. The last two columns show the ratio of REDQ or REDQ-OFE performance compared to SAC and MBPO performance.

Amount of data	SAC	MBPO	REDQ	REDQ/SAC	REDQ/MBPO
Hopper at 62K	594	1919	3278	5.52	1.71
Hopper at 125K	2404	3131	3517	1.46	1.12
Walker2d at 150K	760	3308	3544	4.66	1.07
Walker2d at 300K	2556	3537	4589	1.80	1.30
Ant at 150K	1245	4388	4803	3.86	1.09
Ant at 300K	2485	5774	5369	2.16	0.93
Humanoid at 150K	674	1604	2641	3.92	1.65
Humanoid at 300K	1633	4199	4674	2.86	1.11
Amount of data	SAC	MBPO	REDQ-OFE	OFE/SAC	OFE/MBPO
Ant at 150K	1245	4388	5524	4.44	1.26
Ant at 300K	2485	5774	6578	2.65	1.14
Humanoid at 150K	674	1604	5011	7.43	3.12
Humanoid at 300K	1633	4199	5309	3.25	1.26

D NUMBER OF PARAMETERS COMPARISON

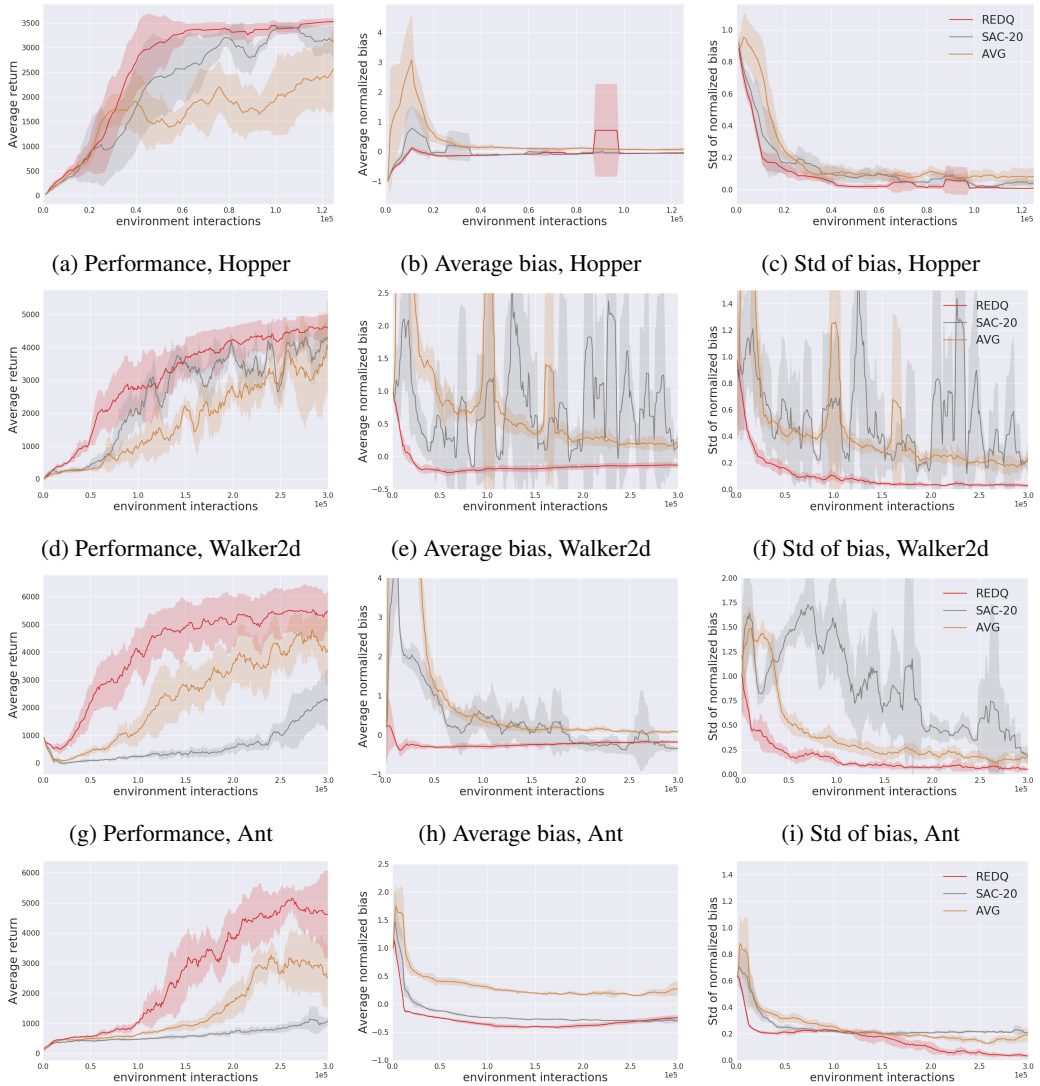
Table 4 gives the number of parameters for MBPO, REDQ and REDQ-OFE, for all four environments. As discussed in the main paper, REDQ uses fewer parameters than MBPO for all four environments: between 26% and 70% as many parameters depending on the environment. After adding OFENet, REDQ still uses fewer parameters than MBPO, with 80% and 35% as many parameters on Ant and Humanoid. In particular, it is surprising that REDQ-OFE can achieve a much stronger result on Humanoid with much fewer parameters.

Table 4: Number of parameters in millions. REDQ uses the same network structure and ensemble size for all four environments. The difference in the number of parameters comes from the fact that the environments have very different observation and action dimensions, which will affect the size of the input and output layers of the networks.

Algorithm	Hopper	Walker2d	Ant	Humanoid
MBPO	1.106M	1.144M	1.617M	7.087M
REDQ $N = 10$	0.769M	0.795M	1.066M	1.840M
REDQ-OFE $N = 10$	-	-	1.294M	2.460M

E ADDITIONAL RESULTS FOR REDQ, SAC-20, AND AVG

Due to lack of space, Figure 2 in Section 3 only compared REDQ with SAC-20 and AVG for the Ant environment. Figure 1 presents the results for all four environments. We can see that in all four environments, REDQ has much stronger performance and much lower std of bias compared to SAC-20 and AVG. Note in terms of average normalized bias, AVG is slightly closer to zero in Ant compared to REDQ, and SAC-20 is a bit closer to zero in Humanoid compared to REDQ; however, their std of normalized bias is consistently higher. This shows the importance of having a low std of the bias in addition to a close-to-zero average bias.



(a) Performance, Hopper (b) Average bias, Hopper (c) Std of bias, Hopper

(d) Performance, Walker2d (e) Average bias, Walker2d (f) Std of bias, Walker2d

(g) Performance, Ant (h) Average bias, Ant (i) Std of bias, Ant

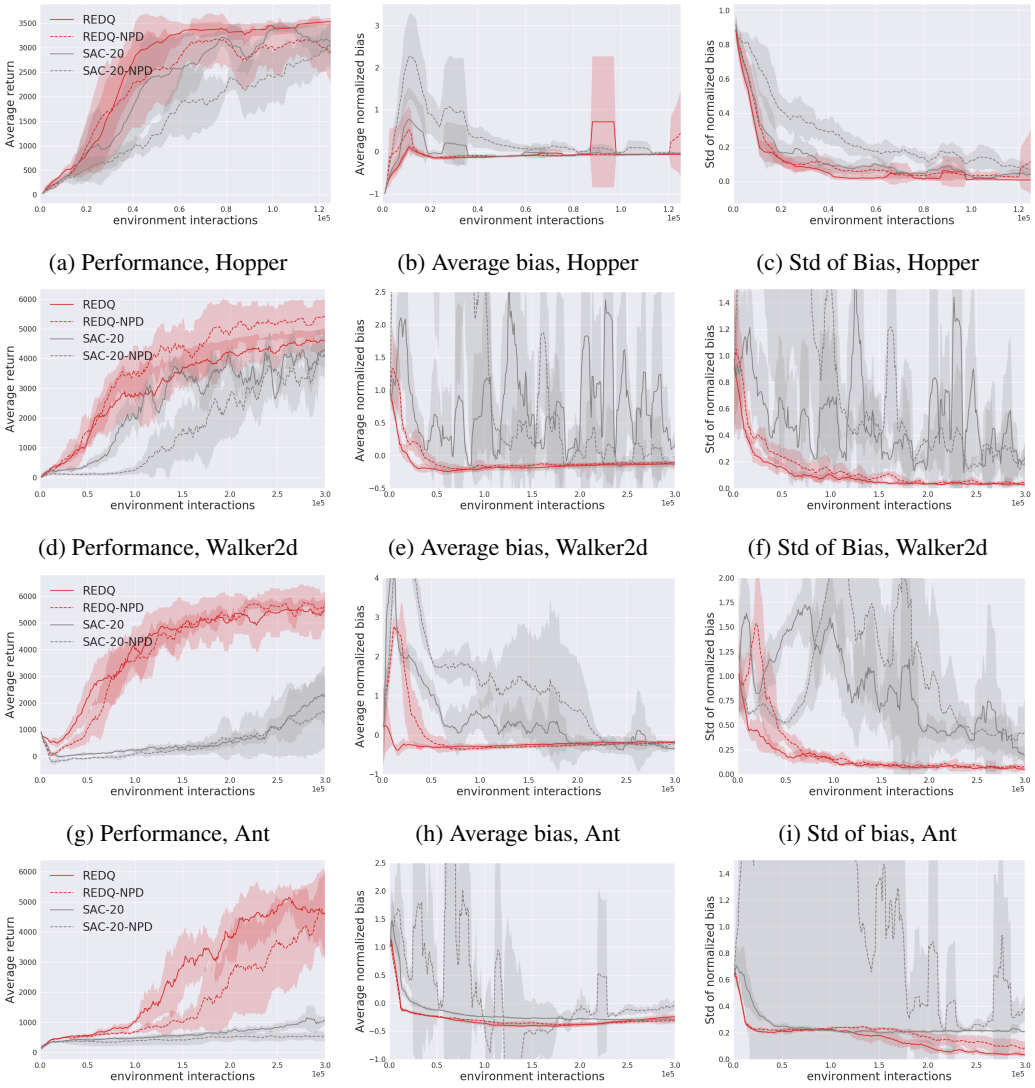
(j) Performance, Humanoid (k) Average bias, Humanoid (l) Std of bias, Humanoid

Figure 1: Performance, mean and std of normalized Q bias for REDQ, AVG, and SAC. All three algorithms have a UTD ratio of 20.

F REDQ AND SAC WITH AND WITHOUT POLICY DELAY

Note that in the REDQ pseudocode, the number of policy updates is always one for each data point collected. We set the UTD ratio for the policy update to always be one in order to isolate the effect of additional policy updates from Q updates. Note in this way, REDQ, SAC-20 and SAC-1 all take the same number of policy updates. This helps show that the performance gain mainly comes from the additional Q updates.

Having a lower number of policy updates can also be seen as a delayed policy update, or policy delay, and is a method that has been used in previous works to improve learning stability (Fujimoto et al., 2018). In this section we discuss how delayed policy update, or policy delay, impact the performance of REDQ and SAC (with UTD of 20). Figure 2 compares REDQ and SAC-20 with and without policy delay (NPD for no policy delay). We can see that having the policy delay consistently makes the bias and std of bias lower and more stable, although they have a smaller effect on REDQ than on SAC. Performance-wise SAC always gets a performance boost with policy delay, while REDQ sees improvement in Hopper and Humanoid, and becomes slightly worse in Walker2d and Ant. The results show that policy delay can be important under high UTD when the variance is not properly controlled. However, with enough variance reduction, the effect of policy delay is diminished, and in some cases having more policy update can give better performance.



(a) Performance, Hopper (b) Average bias, Hopper (c) Std of Bias, Hopper

(d) Performance, Walker2d (e) Average bias, Walker2d (f) Std of Bias, Walker2d

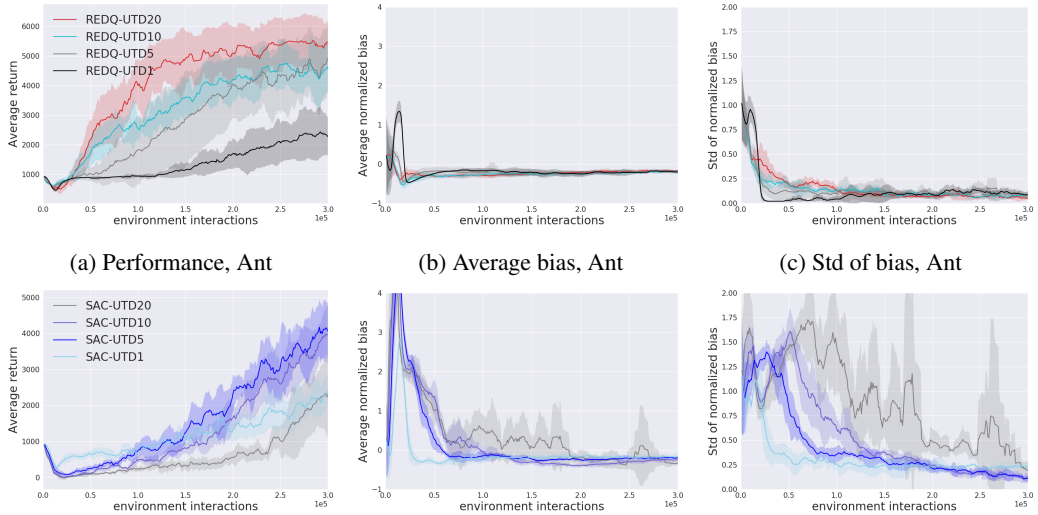
(g) Performance, Ant (h) Average bias, Ant (i) Std of bias, Ant

(j) Performance, Humanoid (k) Average bias, Humanoid (l) Std of bias, Humanoid

Figure 2: Performance, mean and std of normalized Q bias of REDQ and SAC, with and without policy delay.

G REDQ AND SAC WITH DIFFERENT UTD RATIOS

How do different UTD ratio values G impact the performance of REDQ and SAC? Figure 3 compares the two algorithms under UTD ratio values of 1, 5, 10 and 20 for the Ant environment. The results show that in the Ant environment, REDQ greatly benefits from larger UTD values, with UTD of 20 giving the best result. For SAC, performance improves slightly for UTD ratios of 5 and 10, but becomes much worse at 20. Looking at the normalized bias and the std of the bias, we see that changing the UTD ratio does not change the values very much for REDQ, while for SAC, we see that as the UTD ratio increases, both the mean and the std of the bias becomes larger and more unstable.



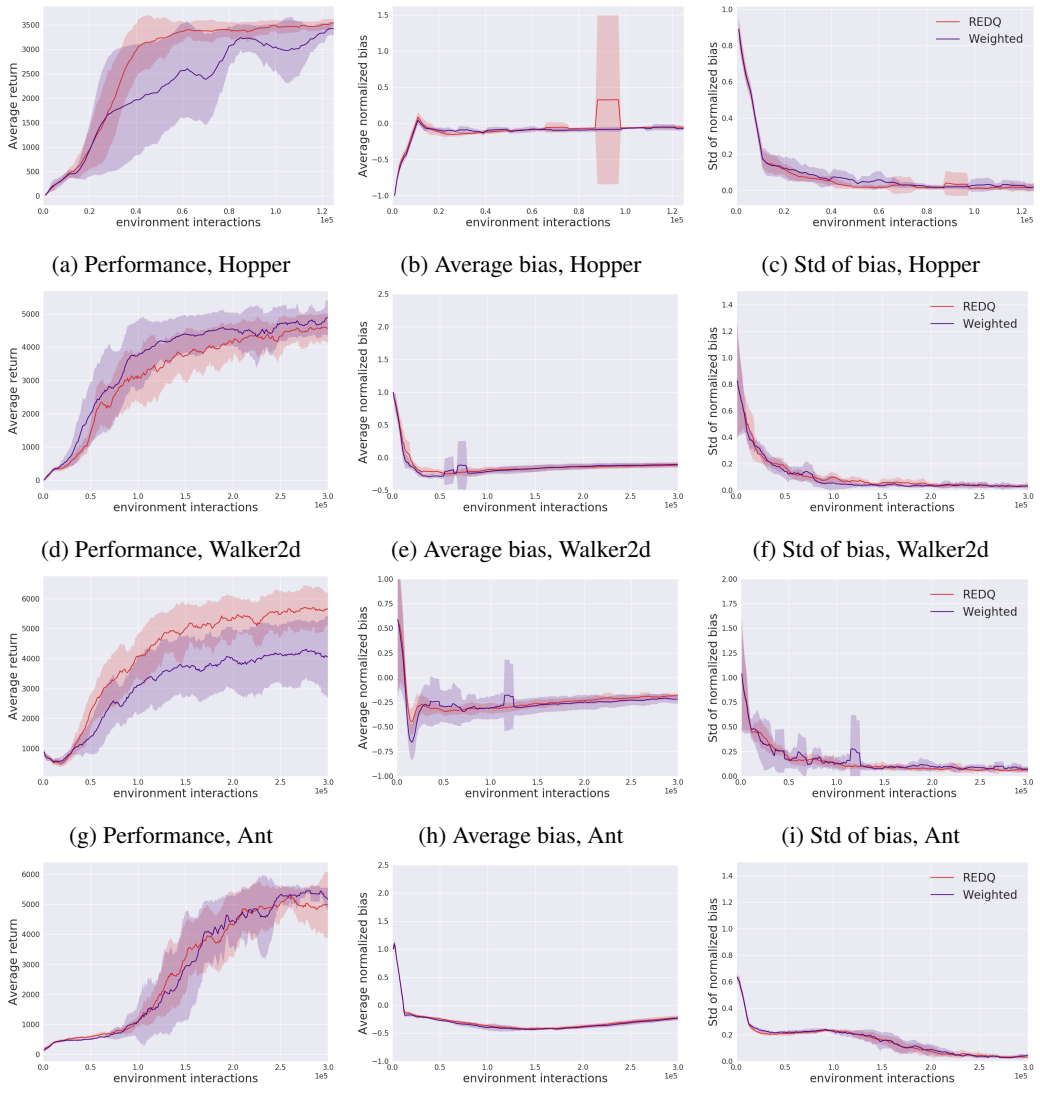
(a) Performance, Ant (b) Average bias, Ant (c) Std of bias, Ant

(d) Performance, Ant (e) Average bias, Ant (f) Std of bias, Ant

Figure 3: Performance, mean and std of normalized Q bias for REDQ, and SAC, with different UTD ratios, in Ant environment.

H ADDITIONAL RESULTS FOR WEIGHTED VARIANT

In this section we provide additional results for the Weighted variant. Figure 4 shows the performance and bias comparison on all four environments. Results show that Weighted and REDQ have similar average bias and std of bias. In terms of performance, Weighed is worse in Ant and Hopper, similar in Humanoid and slightly stronger in Walker2d. Overall REDQ seems to have stronger performance and is more robust. Randomness in the networks might help alleviate overfitting in the early stage, or improve exploration, as shown in previous studies (Osband et al., 2016; Fortunato et al., 2018). This can be important since positive bias in Q learning-based methods can sometimes help exploration. This is commonly referred to as optimistic initial values, or optimism in the face of uncertainty (Sutton & Barto, 2018; Brafman & Tennenholtz, 2002). Thus conservative Q estimates in recent algorithms can lead to the problem of pessimistic underexploration (Ciosek et al., 2019). An interesting future work direction is to study how robust and effective exploration can be achieved without relying on optimistic estimates.



(a) Performance, Hopper (b) Average bias, Hopper (c) Std of bias, Hopper

(d) Performance, Walker2d (e) Average bias, Walker2d (f) Std of bias, Walker2d

(g) Performance, Ant (h) Average bias, Ant (i) Std of bias, Ant

(j) Performance, Humanoid (k) Average bias, Humanoid (l) Std of bias, Humanoid

Figure 4: Performance, mean and std of normalized Q bias for REDQ and Weighted.

REFERENCES

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, pp. 1787–1798, 2019.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *ICLR*, 2018.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

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*, pp. 12519–12530, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pp. 4026–4034, 2016.

Kei Ota, Tomoaki Oiki, Devesh K Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? *arXiv preprint arXiv:2003.01629*, 2020.

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

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