STRIVING FOR SIMPLICITY IN OFF-POLICY DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper advocates the use of offline (batch) reinforcement learning (RL) to help (1) isolate the contributions of exploitation vs. exploration in off-policy deep RL, (2) improve reproducibility of deep RL research, and (3) facilitate the design of simpler deep RL algorithms. We propose an offline RL benchmark on Atari 2600 games comprising all of the replay data of a DQN agent. Using this benchmark, we demonstrate that recent off-policy deep RL algorithms, even when trained solely on logged DQN data, can outperform online DQN. We present Random Ensemble Mixture (REM), a simple Q-learning algorithm that enforces optimal Bellman consistency on random convex combinations of multiple Q-value estimates. The REM algorithm outperforms more complex RL agents such as C51 and QR-DQN on the offline Atari benchmark and performs comparably in the online setting.

1 INTRODUCTION

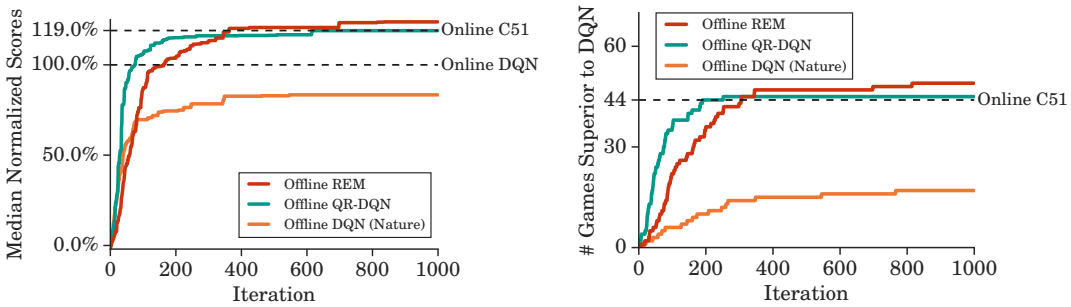
Deep neural networks have become a critical component of modern reinforcement learning (RL) (Sutton and Barto, 2018). The seminal work of Mnih et al. (2013; 2015) on deep Q-networks (DQN) has demonstrated that it is possible to train neural networks using Q-learning (Watkins and Dayan, 1992) to achieve human-level performance in playing Atari 2600 games (Bellemare et al., 2013) directly from raw pixels. Recent progress in mastering Go (Silver et al., 2016) and advances in robotic control (Levine et al., 2016; OpenAI et al., 2018; Kalashnikov et al., 2018) present additional supporting evidence for the enormous potential of deep RL.

Off-policy RL algorithms such as Q-learning are attractive because they disentangle data collection and policy optimization and offer more sample efficient solutions than on-policy algorithms (Sutton et al., 2000; Schulman et al., 2015; Mnih et al., 2016). Importantly, off-policy techniques can leverage the vast amount of existing offline logged data for real-world applications such as digital advertising (Strehl et al., 2010; Bottou et al., 2013), education (Mandel et al., 2014), and healthcare (Shortreed et al., 2011). Since online RL is often unsafe to deploy in the real world, offline RL algorithms are the only feasible solution for many practical decision making problems (Dulac-Arnold et al., 2019). Nevertheless, off-policy RL algorithms when combined with neural networks can be unstable or even divergent (Baird, 1995; Boyan and Moore, 1995; Tsitsiklis and Van Roy, 1997).

In the absence of theoretical guarantees for off-policy deep RL, recent advances (see Hessel et al. (2018) for an overview) are largely governed by empirical results on a popular benchmark suite of Atari 2600 games (Bellemare et al., 2013). Unfortunately, these empirical results are difficult to reproduce (Henderson et al., 2018) and the contribution of novel RL algorithms is often conflated with many other design choices (Clary et al., 2019; Khetarpal et al., 2018). Given the empirical nature of deep off-policy RL, it is crucial to come up with simpler and reproducible experimental settings and study the relative importance of different components of deep RL algorithms. It is also important to strive for finding successful RL algorithms that are as simple as possible.

This paper advocates the use of offline (batch) RL to help (1) isolate the contributions of *exploitation vs. exploration* in off-policy deep RL, (2) improve reproducibility of deep RL research, and (3) facilitate the design of simpler off-policy deep RL algorithms. To this end, we propose a new benchmark for offline optimization of RL agents on Atari 2600 games using all of the replay data of a DQN agent (Mnih et al., 2015). Using this benchmark, we investigate the following questions:

1. Is it possible to train successful Atari agents based solely on offline data?
2. Can one design simple and effective alternatives to intricate RL algorithms in the offline setting?
3. Are the insights gained from the offline setting useful for developing effective online algorithms?



(a) (b)

Figure 1: (a) Median normalized online evaluation scores averaged over 5 runs across 60 Atari 2600 games of offline agents trained using the DQN replay dataset (b) Number of games where an offline agent achieves a higher score than fully trained online DQN (Nature) as a function of training time. The online agents are trained for 200 iterations where each iteration corresponds to 1 million game frames.

The contributions of this paper can be summarized as:

- An offline RL benchmark is proposed for evaluating and designing RL algorithms on Atari 2600 games without exploration, based on the logged replay data of a DQN agent comprising 50 million (observation, action, reward, next observation) tuples per game. This reduces the computation cost of the experiments considerably and helps improve reproducibility of deep RL research by standardizing training using a fixed offline dataset. The replay dataset used in our experiments will be released to enable offline optimization of RL algorithms on a common ground.
- Contrary to recent work (Zhang and Sutton, 2017; Fujimoto et al., 2019), we find that the logged DQN data is sufficient for optimizing strong Atari agents offline without any environment interaction. For instance, QR-DQN (Dabney et al., 2018b) trained on the DQN replay dataset significantly outperforms online DQN (Mnih et al., 2015).
- A simple and novel Q-learning algorithm called *Random Ensemble Mixture (REM)* is presented, which enforces optimal Bellman consistency on random convex combinations of multiple Q-value estimates. The REM algorithm outperforms more complex RL algorithms such as C51 (Bellemare et al., 2017) and QR-DQN in the offline setting (Figure 1).
- We use the insights gained from the offline experiments to develop an online variant of REM, which performs comparably with online QR-DQN (Figure 4b).

2 Off-policy Reinforcement Learning

An interactive environment in reinforcement learning (RL) is typically described as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ (Puterman, 1994), with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a stochastic reward function $R(s, a)$, transition dynamics $P(s'|s, a)$ and a discount factor $\gamma \in [0, 1)$. A stochastic policy $\pi(\cdot \mid s)$ maps each state $s \in \mathcal{S}$ to a distribution (density) over actions.

For an agent following the policy π, the action-value function, denoted $Q^\pi(s, a)$, is defined as the expectation of cumulative discounted future rewards, *i.e.,*

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; s_0 = s, a_0 = a, s_t \sim P(\cdot \mid s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot \mid s_t), r_t \sim R(s_t, a_t)\right]. \tag{1}$$

The goal of RL is to find an optimal policy π^* that attains maximum expected return, for which $Q^{\pi^*}(s, a) \geq Q^\pi(s, a)$ for all π, s, a. The Bellman optimality equations (Bellman, 1957) characterize the optimal policy in terms of the optimal Q-values, denoted $Q^* = Q^{\pi^*}$, via:

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s', a') \;\middle|\; r \sim R(s,a),\; s' \sim P(\cdot \mid s, a)\right]. \tag{2}$$

To learn a policy from interaction with the environment, Q-learning (Watkins and Dayan, 1992) iteratively improves an approximate estimate of Q^*, denoted Q_θ, by repeatedly regressing the LHS of (2) to target values defined by samples from the RHS of (2). For large and complex state spaces, approximate Q-values are obtained using a neural network as the function approximator. To further stabilize optimization, a target network $Q_{\theta'}$ with frozen parameters may be used for

computing the learning target (Mnih et al., 2013). The target network parameters θ' are updated to the current Q-network parameters θ after a fixed number of time steps. DQN (Mnih et al., 2013; 2015) parameterizes Q_θ with a convolutional neural network (LeCun et al., 1998) and uses Q-learning with a target network while following an ϵ-greedy policy with respect to Q_θ for data collection. DQN minimizes the TD error using the loss $\mathcal{L}(\theta)$ on mini-batches of agent's past experience tuples, (s, a, r, s'), sampled from an experience replay buffer $\mathcal{D}$ (Lin, 1992) collected during training:

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\ell_\delta\left(Q_\theta(s,a) - r - \gamma \max_{a'} Q_{\theta'}(s',a')\right)\right], \tag{3}$$

where l_δ is the Huber loss (Huber, 1964) given by $\ell_\delta(u) = \begin{cases} \frac{1}{2}u^2 & \text{for } |u| \le \delta, \\ \delta(|u| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$.

Q-learning is an *off-policy* algorithm (Sutton and Barto, 2018) since the learning target can be computed without any consideration of how the experience was generated. In offline (batch) RL (Ernst et al., 2005; Riedmiller, 2005; Lange et al., 2012), we assume access to a fixed offline dataset of experiences $\mathcal{D}$, without any further interaction with the environment.

A family of recent off-policy deep RL algorithms, which serve as a strong baseline in this paper, include Distributional RL (Bellemare et al., 2017; Jaquette, 1973) methods. Such algorithms estimate a density over returns for each state-action pair, denoted $Z^\pi(s,a)$, instead of directly estimating the mean $Q^\pi(s,a)$. Accordingly, one can express a form of distributional Bellman optimality as

$$Z^*(s,a) \stackrel{D}{=} r + \gamma Z^*(s', \text{argmax}_{a'\in\mathcal{A}}\, Q^*(s',a')), \qquad \text{where } r \sim R(s,a),\ s' \sim P(\cdot \mid s,a), \tag{4}$$

and $\stackrel{D}{=}$ denotes distributional equivalence and $Q^*(s',a')$ is estimated by taking an expectation with respect to $Z^*(s',a')$. C51 (Bellemare et al., 2017) approximates $Z^*(s,a)$ by using a categorical distribution over a set of pre-specified anchor points, and distributional QR-DQN (Dabney et al., 2018b) approximates the return density by using a uniform mixture of K Dirac delta functions, *i.e.,*

$$Z_\theta(s,a) := \frac{1}{K}\sum_{i=1}^{K} \delta_{\theta_i(s,a)}, \quad \text{and} \quad Q_\theta(s,a) = \frac{1}{K}\sum_{i=1}^{K} \theta_i(s,a)\,. \tag{5}$$

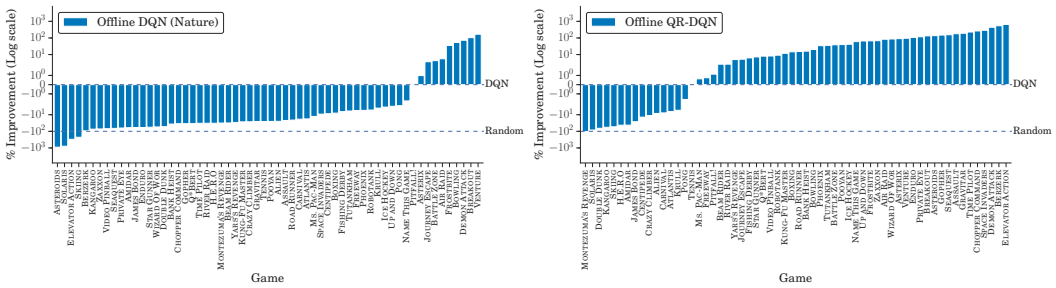
QR-DQN trains each θ_i for $1 \le i \le K$ to match the $\frac{2i-1}{2K}$-quantile of the target return density (RHS of (4)) using the Huber quantile regression loss (Koenker, 2005; Dabney et al., 2018b). QR-DQN, albeit complex, outperforms C51 and DQN and obtains state-of-the-art results on Atari 2600 games, among agents that do not exploit n-step updates (Sutton, 1988) and prioritized replay (Schaul et al., 2016). This paper avoids using n-step updates and prioritized replay to keep the empirical study simple and focused on deep Q-learning algorithms.

3 Offline Optimization of Reinforcement Learning Agents

Modern off-policy deep RL algorithms perform remarkably well on common benchmarks such as the Atari 2600 games (Bellemare et al., 2013) and continuous control MuJoCo tasks (Todorov et al., 2012). Such off-policy algorithms (Mnih et al., 2015; Lillicrap et al., 2015) are considered "online" because they alternate between optimizing a policy and using that policy to collect more data. Typically, these algorithms keep a sliding window of most recent experiences in a finite replay buffer (Lin, 1992), throwing away stale data to incorporate most fresh (on-policy) experiences. In principle, off-policy algorithms can learn from data collected by any policy, however, Zhang and Sutton (2017) assert that using a large replay buffer can significantly hurt the performance of Q-learning algorithms, since it can delay rare on-policy experiences that are importnat for policy learning.

This paper revisits offline off-policy RL and investigates whether off-policy deep RL agents trained solely on offline data can be successful. We advocate the use of offline RL to help isolate an RL algorithm's ability to *exploit* experience and generalize *vs.* its ability to *explore* effectively by collecting interesting new data. The offline RL setting removes design choices related to the replay buffer and exploration; therefore, it is much simpler to experiment with and reproduce than the typical online off-policy learning. Also, in the offline setting, we optimize an objective (*e.g.,* (3)) over a fixed dataset as opposed to a changing replay buffer of an online agent.

To facilitate a large-scale study of offline RL, we train several instances of DQN (Nature) agents (Mnih et al., 2015) on 60 Atari 2600 games for 200 million frames each, with a frame skip of 4 (standard



(a) (b)

Figure 2: Normalized performance improvement (in %) over online DQN (Nature), per game, of (a) offline DQN (Nature) and (b) offline QR-DQN trained using the DQN replay dataset for same number of gradient updates as online DQN. The normalized online score for each game is 0.0 and 1.0 for the worse and better performing agent among fully trained online DQN and random agents respectively.

protocol) and sticky actions enabled (Machado et al., 2018). On each game, we train 5 different agents with random initialization, and store all of the experience tuples of (observation, action, reward, next observation) encountered during training into 5 replay datasets of 50 million tuples each. These replay datasets are used for training standard off-policy RL agents, offline, without any interaction with the environment during training. We perform an online evaluation of the agents in the intervals of 1 million training frames, and report the best evaluation score for each agent, averaged over 5 runs.

3.1 Experiments and Results

Given the logged replay data of the DQN agent, it is natural to ask whether training an offline variant of DQN on this data will recover the performance of online DQN. Furthermore, whether more recent off-policy algorithms are able to exploit the DQN replay dataset more effectively than DQN. To investigate these questions, we train DQN (Nature) and distributional QR-DQN (Dabney et al., 2018b) agents, offline, on the DQN replay dataset for the same number of gradient updates as online DQN. We use the hyperparameters provided in the Dopamine baselines (Castro et al., 2018) for a standardized comparison (Section A.4) and report scores using a normalized scale (Section A.3). We note that the replay datasets include samples from all of the intermediate policies seen during the optimization of the DQN (Nature) agent. Accordingly, we compare the performance of offline agents against the best performing policy among the mixture of data collecting policies.

Results. We find that offline DQN underperforms online DQN on all except a few games while offline QR-DQN outperforms offline DQN and online DQN on most of the games (Figure 2). C51 (Bellemare et al., 2017) trained offline using the DQN replay dataset also considerably improves upon offline DQN (Figure A.4). These results suggest that DQN (Nature) is somewhat ineffective at exploiting off-policy data. However, offline QR-DQN's impressive performance demonstrates that it is possible to optimize strong Atari agents completely offline (see Figure A.6 for learning curves).

3.2 Offline Continuous Control Experiments

In contrast to our offline results on Atari 2600 games with discrete action spaces, discussed above, Fujimoto et al. (2019) find that standard off-policy deep RL agents are not effective on continuous control problems when trained offline, even when large and diverse replay datasets are used. The results of Fujimoto et al. (2019) are based on the evaluation of a standard continuous control agent, called DDPG (Lillicrap et al., 2015), and other more recent continuous control algorithms such as TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018) are not considered in their study.

Motivated by the so-called *final buffer* setting in Fujimoto et al. (2019) (Section A.2), we train a DDPG agent on continuous control MuJoCo tasks (Todorov et al., 2012) for 1 million time steps and store all of the experienced transitions. Using this dataset, we train standard off-policy agents including TD3 and DDPG completely offline. Consistent with our offline results on Atari games, offline TD3 significantly outperforms the data collecting DDPG agent and offline DDPG (Figure A.1).

4 Seeking Simpler Off-Policy RL Algorithms

Section 3.1 finds that recent distributional RL algorithms such as QR-DQN are effective at exploiting offline data. However, these algorithms are complicated as they require a generalization of Bellman

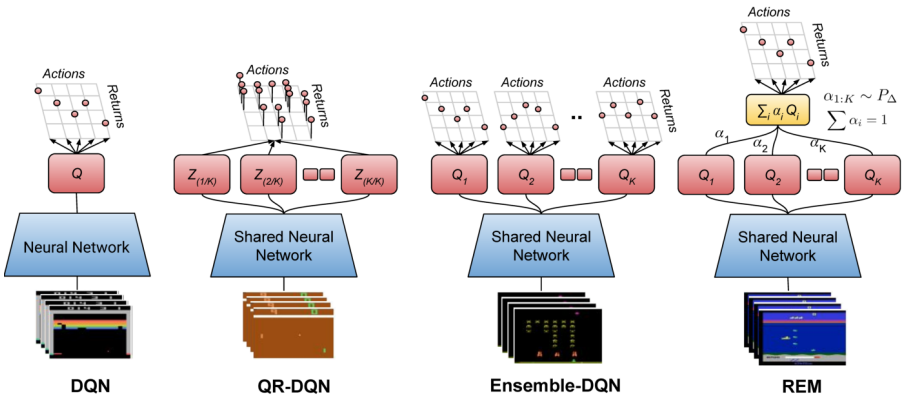


Figure 3: Neural network architectures for DQN, QR-DQN and the proposed expected RL variants, *i.e.,* Ensemble-DQN and REM, with the same multi-head architecture as QR-DQN. The individual Q-heads share all of the neural network layers except the final fully connected layer.

equations to distributions (4) and typically minimize a probability divergence measure from a distributional target (Rowland et al., 2018; Dabney et al., 2018b) instead of scalar TD errors. Moreover, despite recent efforts (Lyle et al., 2019; Bellemare et al., 2019), the source of the gains when using distributional RL algorithms remains unclear. We investigate whether one can develop simple and effective alternatives to more intricate RL algorithms in the offline setting. To this end, we study two deep Q-learning algorithms, Ensemble DQN and REM, which adopt ensembling to improve stability.

4.1 Ensemble-DQN

Ensemble-DQN is a simple extension of DQN that approximates the Q-values via an ensemble of parameterized Q-functions (Faußer and Schwenker, 2015; Osband et al., 2016; Anschel et al., 2017). Each Q-value estimate, denoted $Q^k_\theta(s,a)$, is trained against its own target $Q^k_{\theta'}(s,a)$, similar to Bootstrapped-DQN (Osband et al., 2016). The Q-functions are optimized using identical mini-batches in the same order, starting from different parameter initializations. The loss $\mathcal{L}(\theta)$ takes the form,

$$\mathcal{L}(\theta) \;=\; \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\ell_\delta\left(Q^k_\theta(s,a) - r - \gamma \max_{a'} Q^k_{\theta'}(s',a')\right)\right] \;, \tag{6}$$

where l_δ is the Huber loss. While Bootstrapped-DQN uses one of the Q-value estimates in each episode to improve exploration, in the offline setting, we are only concerned with the ability of Ensemble-DQN to exploit better and use the mean of the Q-value estimates for evaluation.

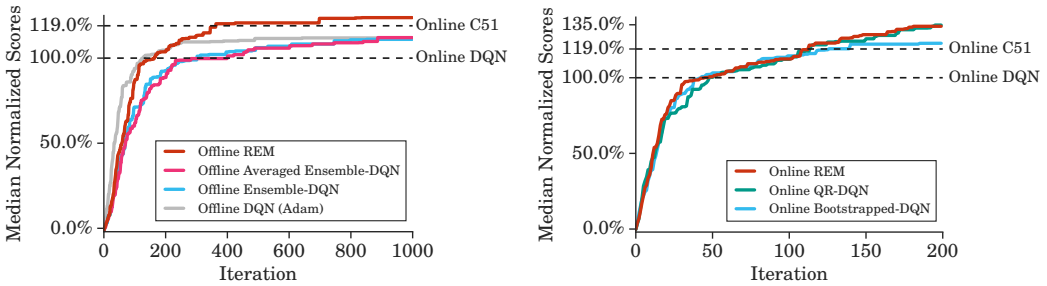
4.2 Random Ensemble Mixture (REM)

Increasing the number of models used for ensembling typically improves the performance of supervised learning models (Shazeer et al., 2017). This raises the question whether one can use an ensemble over an exponential number of Q-estimates in a computationally efficient manner. Inspired by dropout (Srivastava et al., 2014), we propose Random Ensemble Mixture for RL.

Random Ensemble Mixture (REM) uses multiple parameterized Q-functions to estimate the Q-values, similar to Ensemble-DQN. The key insight behind REM is that one can think of a convex combination of multiple Q-value estimates as a valid Q-value estimate itself. This is especially true at the fixed point, where all of the Q-value estimates have converged to an identical Q-function. Using this insight, we train a family of Q-function approximators defined by mixing probabilities on a $(K-1)$-simplex. Specifically, for each mini-batch, we randomly draw a categorical distribution α, which defines a convex combination of the K estimates to approximate the optimal Q-function. This approximator is trained against its corresponding target to minimize the TD error. The loss $\mathcal{L}(\theta)$ takes the form,

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\mathbb{E}_{\alpha_1,\ldots,\alpha_K\sim \mathrm{P}_\Delta}\left[\ell_\delta\left(\sum_k \alpha_k Q^k_\theta(s,a) - r - \gamma \max_{a'} \sum_k \alpha_k Q^k_{\theta'}(s',a')\right)\right]\right] \tag{7}$$

where P_Δ represents a probability distribution over the standard $(K-1)$-simplex $\Delta^{\mathrm{K}-1} = \{\alpha \in \mathbb{R}^{\mathrm{K}} : \alpha_1 + \alpha_2 + \cdots + \alpha_K = 1, \alpha_k \geq 0, k = 1,\ldots,K\}$. One can view $\mathcal{L}(\theta)$ as an infinite set



(a) (b)

Figure 4: Normalized scores averaged over 5 runs across 60 Atari 2600 games of (a) offline agents trained using the DQN replay dataset, and (b) agents trained online for 200 million game frames.

of Bellman optimality constraints corresponding to different mixture probability distributions. For action selection, we use the average of the K value estimates as the Q-function, *i.e.,* $Q(s,a) = \sum_k Q_\theta^k(s,a)/K$. REM is easy to implement and analyze (see Proposition 1), and can be viewed as a simple regularization technique for value-based RL. In our experiments, we use a very simple distribution P_Δ: we first draw a set of K values *i. i. d.* from Uniform (0, 1) and normalize them to get a valid categorical distribution, *i.e.,* $\alpha'_k \sim U(0,1)$ followed by $\alpha_k = \alpha'_k / \sum \alpha'_i$.

Proposition 1. *Consider the assumptions: (a) The distribution* P_Δ *has full support over the entire* $(K-1)$*-simplex. (b) Only a finite number of distinct* Q*-functions globally minimize the loss in* (3)*. (c)* Q^* *is defined in terms of the MDP induced by the data distribution* $\mathcal{D}$*. (d* Q^* *lies in the family of our function approximation. Then, at the global minimum of* $\mathcal{L}(\theta)$ (7) *for a multi-head* Q*-network:*

(i) Under assumptions (a) and (b), all the Q*-heads represent identical* Q*-functions.*
(ii) Under assumptions (a)–(d), the common global solution is Q^**.*

The proof of *(ii)* follows from *(i)* and the fact that (7) is lower bounded by the TD error attained by Q^*. The proof of part *(i)* can be found in the supplementary material.

4.3 Experiments and Results

Asymptotic performance of offline agents. In supervised learning, asymptotic performance matters much more than performance within a fixed budget of gradient updates. Similarly, for a given sample complexity, we prefer RL algorithms that perform the best as long as the number of gradient updates is feasible. Since the sample efficiency of the offline agents for a given dataset is fixed, we train them on the DQN replay dataset for five times as many gradient updates as online DQN.

Comparison with QR-DQN. QR-DQN modifies the DQN (Nature) architecture to output K values for each action using a multi-head Q-network and replaces RMSProp (Tieleman and Hinton, 2012) with Adam (Kingma and Ba, 2015) for optimization. To ensure a fair comparison with QR-DQN, we use the same multi-head Q-network as QR-DQN with $K = 200$ heads (Figure 3), where each head represents a Q-value estimate for REM and Ensemble-DQN. We also use Adam for optimization.

Additional Baselines. To isolate the gains due to Adam in QR-DQN and our proposed variants, we compare against a DQN baseline which also uses Adam. We also evaluate Averaged Ensemble-DQN, a variant of Ensemble-DQN proposed by Anschel et al. (2017), which uses the average of the predicted target Q-values as the Bellman target for training each parameterized Q-function. This baseline determines whether the random combinations of REM provide any significant benefit over simply using an ensemble of predictors to stabilize the Bellman target.

REM with separate Q-networks. To investigate whether the effectiveness of REM is linked to ensembling techniques and dropout, we compare offline REM with K Q-value estimates computed using K separate Q-networks *vs.* a multi-head Q-network for $K \in \{4, 16\}$.

Results. Figure 4a shows the comparison of the additional baselines with REM and Ensemble-DQN and Table 1 summarizes the asymptotic performance results. DQN with Adam noticeably bridges the gap in asymptotic performance between QR-DQN and DQN (Nature) in the offline setting. Offline Ensemble-DQN does not improve upon this strong DQN baseline showing that its naive ensembling approach is inadequate. Furthermore, Averaged Ensemble-DQN performs only slightly better than Ensemble-DQN. In contrast, REM exploits offline data more effectively than other agents, including

Table 1: Median normalized best evaluation scores (averaged over 5 runs) across 60 Atari 2600 games, measured as percentages and number of games where an agent achieves better scores than a fully trained online DQN (Nature) agent. The offline RL agents are trained using the DQN replay dataset for five times as many gradient updates as the online DQN agent. The online RL agents are trained for 200 million frames (standard protocol).

Offline agent	Median	> DQN
DQN (Adam)	111.9%	41
Ensemble-DQN	111.0%	39
Averaged Ensemble-DQN	112.1%	43
QR-DQN	118.9%	45
REM	**123.8%**	**49**

Online agent	Median	> DQN
C51	119.0%	44
Bootstrapped-DQN	122.8%	**53**
QR-DQN	**134.8%**	52
REM	134.0%	52

QR-DQN, when trained for more gradient updates. Surprisingly, using the DQN replay dataset, offline REM surpasses the gains from online C51, the first state-of-the-art distributional RL method on Atari games (Figure 1). REM with separate Q-networks performs better asymptotically and learns faster than REM with a multi-head Q-network in the offline setting (Figure A.5).

4.4 From Offline to Online Reinforcement Learning

Can we combine the insights gained from the offline setting with appropriate design choices (*e.g.,* exploration, replay buffer) to create effective online methods? In online RL, learning and data generation are tightly coupled, *i.e.,* an agent that learns faster also collects more relevant training data.

We ran online REM with K separate Q-networks (with $K = 4$ for computational efficiency) because of the improved convergence speed over multi-head REM in the offline setting. For data collection, we use ϵ-greedy with a randomly sampled Q-estimate from the simplex for each episode, similar to Bootstrapped DQN. To estimate the gains from the REM objective (7) in the online setting, we also evaluate Bootstrapped-DQN with identical modifications (*e.g.,* separate Q-networks) as online REM. Figure 4b and Table 1 show that REM performs on par with QR-DQN and considerably outperforms Bootstrapped-DQN (see Figure A.8 for learning curves). Based on these results, we conclude that our offline benchmark cen help facilitate the design of effective online off-policy RL algorithms.

5 Related work

Our work is motivated by whether the complexity of recent off-policy approaches (*e.g.,* Rainbow (Hessel et al., 2018)) is necessary and is similar in spirit to work by Rajeswaran et al. (2017), which attempts to demystify the complexity of on-policy deep RL for continuous control. Our work is mainly related to two subareas of RL known as batch RL and distributional RL. For the sake of clarity, we refer to batch RL as offline RL in other sections of the paper.

Batch Reinforcement Learning. While there has been increasing interest in batch RL (Lange et al., 2012) over the last few years (Jiang and Li, 2016; Farajtabar et al., 2018; Irpan et al., 2019), much of this focussed on off-policy policy evaluation, where the goal is to estimate the performance of a given fixed policy. Similar to (Ernst et al., 2005; Kalyanakrishnan and Stone, 2007; Jaques et al., 2019), we investigate batch off-policy learning, which requires learning a good policy given a fixed dataset. In our offline setup, we only assume access to samples from the behavior policy and focus on Q-learning methods without any importance sampling correction as opposed to (Swaminathan and Joachims, 2015; Liu et al., 2019). Recent work (Fujimoto et al., 2019; Kumar et al., 2019) documents that standard off-policy methods with fixed datasets fail on continuous control MuJoCo tasks. Opposed to Kumar et al. (2019), we focus on the offline setting with data collected from a diverse mixture of policies rather than a single Markovian behavior policy. Furthermore, in contrast to Fujimoto et al. (2019), our batch learning results on MuJoCo tasks and Atari 2600 games (Section 3) demonstrate that standard off-policy deep RL algorithms (*e.g.,* TD3 (Fujimoto et al., 2018), QR-DQN (Dabney et al., 2018b)) are quite effective when learning truly off-policy from large and diverse datasets. Zhang and Sutton (2017) show that large replay buffers can hurt the performance of simple Q-learning methods with weak function approximators; they attribute this to the "off-policyness" of a large buffer which might delay important transitions for learning. However, our results reveal that even with uniform sampling, effective deep Q-learning algorithms (*e.g.,* REM) can exploit completely off-policy logged DQN data on Atari 2600 games, given sufficient number of gradient updates.

Distributional Reinforcement Learning. Recently, distributional RL algorithms (*e.g.,* C51 (Bellemare et al., 2017), QR-DQN (Dabney et al., 2018b), IQN (Dabney et al., 2018a)) gained popularity due

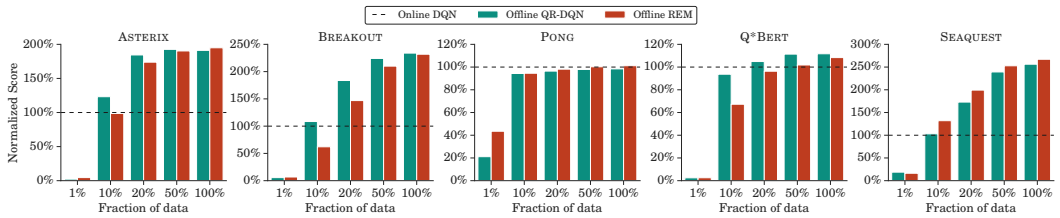


Figure 5: Normalized scores (averaged over 5 runs) of QR-DQN and multi-head REM trained offline on 5 Atari 2600 games for 5X gradient steps using only a fraction of the DQN replay dataset.

to their strong performance on Atari 2600 games. We observe that distributional algorithms (*e.g.,* QR-DQN, C51) are quite effective at exploitation and develop much simpler REM (Section 4.2), which is as successful as QR-DQN. In contrast to distributional RL, much of the theoretical results of Q-learning naturally extends to REM due to its similarity to Q-learning, *e.g.,* in the tabular setting, REM converges to Q^*, while popular distributional methods (*e.g.,* QR-DQN, C51) need not learn true return statistics corresponding to the underlying MDP (Rowland et al., 2019).

6 Offline RL Challenges: Sample Efficiency and Stability

Our offline learning results indicate that 50 million logged experience tuples per game from DQN (Nature) are sufficient to obtain good online performance on most of the Atari 2600 games. One may argue that the size of the fixed replay buffer and its composition play a key role in our empirical results (De Bruin et al., 2015). To study the role of the replay buffer size in the success of the offline agents, we perform an ablation experiment with variable replay buffer size.

We conducted offline experiments with reduced data obtained via randomly subsampling entire trajectories from the logged DQN experiences, thereby maintaining the same data distribution. Figure 5 presents the performance of the offline REM and QR-DQN agents with $N\%$ of the 50 million tuples in the DQN replay dataset where $N \in \{1, 10, 20, 50, 100\}$. As expected, performance tends to increase as the fraction of data increases. With $N \geq 10\%$, REM and QR-DQN still perform comparably to online DQN on most of these games. However, the performance deteriorates drastically for $N = 1\%$. These results pose an important challenge in off-policy deep RL: Can we create *sample-efficient* agents that squeeze the most performance gains out of a fixed batch of data?

The offline agents obtain much higher scores compared to the returns in DQN replay dataset on most of the games (Figure A.3). However, even with 50 million tuples, they exhibit overfitting on some games (Figure A.7), *i.e.,* after a sufficiently large number of gradient updates, their performance degrades. We also observe divergence w.r.t. Huber loss in our reduced data experiments with $N = 1\%$. Based on these findings, we posit that our offline benchmark can be used for designing techniques akin to regularization for *stable* off-policy deep RL algorithms with reasonable performance throughout learning (Garcıa and Fernández, 2015), a requirement for many practical RL applications.

7 Conclusions and Future Work

This work investigates off-policy deep RL using an offline RL benchmark on Atari 2600 games based on logged experiences of a DQN agent. We demonstrate that it is possible to learn policies with high returns that significantly outperform the policies used to collect replay data. We develop REM, a simple variant of ensemble Q-learning, that can effectively exploit large-scale off-policy data and shares much of the success of the more complicated distributional RL algorithms on Atari 2600 games. The proposed benchmark can serve as a testbed for designing simple, stable and sample-efficient RL algorithms and enable the research community to evaluate off-policy methods on a common ground.

For future work, investigating Rainbow components, *e.g.,* prioritized replay (Schaul et al., 2016), n-step updates (Sutton, 1988), double Q-learning (Van Hasselt et al., 2016), *etc.*, in the offline RL setting and incorporating them with REM are interesting directions. Experience replay-based algorithms can be more sample efficient than model-based approaches (Van Hasselt et al., 2019), and using the DQN replay dataset on Atari 2600 games for designing non-parametric replay models (Pan et al., 2018) and parametric world models (Kaiser et al., 2019) is quite promising for improving sample-efficiency. We also leave further investigation of the exploitation ability of distributional RL to future work. As mentioned by Sutton and Barto (2018, Chapter 11.10): "*The potential for off-policy learning remains tantalizing, the best way to achieve it still a mystery.*"

REFERENCES

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. *ICML*, 2017.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. *Machine Learning*, 1995.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *ICML*, 2017.

Marc G Bellemare, Nicolas Le Roux, Pablo Samuel Castro, and Subhodeep Moitra. Distributional reinforcement learning with linear function approximation. *arXiv preprint arXiv:1902.03149*, 2019.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*, 2013.

Justin A Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. *NeurIPS*, 1995.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *ArXiv:1812.06110*, 2018.

Kaleigh Clary, Emma Tosch, John Foley, and David Jensen. Let's play again: Variability of deep reinforcement learning agents in atari environments. *arXiv preprint arXiv:1904.06312*, 2019.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. *ICML*, 2018a.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. *AAAI*, 2018b.

Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. The importance of experience replay database composition in deep reinforcement learning. *Deep reinforcement learning workshop, NIPS*, 2015.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *JMLR*, 2005.

Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. *ICML*, 2018.

Stefan Faußer and Friedhelm Schwenker. Neural network ensembles in reinforcement learning. *Neural Processing Letters*, 2015.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *ICML*, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *ICML*, 2019.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 2015.

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.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. 2018.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *AAAI*, 2018.

PJ Huber. Robust estimation of a location parameter. *Ann. Math. Stat.*, 1964.

Alex Irpan, Kanishka Rao, Konstantinos Bousmalis, Chris Harris, Julian Ibarz, and Sergey Levine. Off-policy evaluation via off-policy classification. *NeurIPS*, 2019.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

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

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. *ICML*, 2016.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRL*, 2018.

Shivaram Kalyanakrishnan and Peter Stone. Batch reinforcement learning in a complex domain. *AAMAS*, 2007.

Khimya Khetarpal, Zafarali Ahmed, Andre Cianflone, Riashat Islam, and Joelle Pineau. Re-evaluate: Reproducibility in evaluating reinforcement learning algorithms. 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Roger Koenker. *Quantile Regression*. Cambridge University Press, 2005.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. *NeurIPS*, 2019.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. *Reinforcement learning*, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ArXiv:1509.02971*, 2015.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 1992.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

Clare Lyle, Pablo Samuel Castro, and Marc G Bellemare. A comparative analysis of expected and distributional reinforcement learning. *AAAI*, 2019.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 2018.

Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. *AAMAS*, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *ICML*, 2016.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, 2018.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *NeurIPS*, 2016.

Yangchen Pan, Muhammad Zaheer, Adam White, Andrew Patterson, and Martha White. Organizing experience: a deeper look at replay mechanisms for sample-based planning in continuous state domains. *IJCAI*, 2018.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. *NeurIPS*, 2017.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, 2005.

Mark Rowland, Marc G Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. *AISTATS*, 2018.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. *ICML*, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *ICLR*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. *ICML*, 2015.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*, 2017.

Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 2011.

David Silver, Aja Huang, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Alexander L. Strehl, John Langford, Lihong Li, and Sham Kakade. Learning from logged implicit exploration data. *NeurIPS*, 2010.

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

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

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *NeurIPS*, 2000.

Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *JMLR*, 2015.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *IROS*, 2012.

John N Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. *NeurIPS*, 1997.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. *AAAI*, 2016.

Hado Van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *NeurIPS*, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 1992.

Shangtong Zhang and Richard S. Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

A APPENDIX

A.1 PROOFS

Proposition 1. *Consider the assumptions: (a) The distribution* P_Δ *has full support over the entire* $(K-1)$*-simplex. (b) Only a finite number of distinct Q-functions globally minimize the loss in* (3)*. (c)* Q^* *is defined in terms of the MDP induced by the data distribution* $\mathcal{D}$*. (d)* Q^* *lies in the family of our function approximation. Then at the global minimum of* $\mathcal{L}(\theta)$ (7) *for multi-head Q-network :*

(i) Under assumptions (a) and (b), all the Q-heads represent identical Q-functions.
(ii) Under assumptions (a)–(d), the common convergence point is Q^**.*

Proof. Part *(i)*: Under assumptions (a) and (b), we would prove by contradiction that each Q-head should be identical to minimize the REM loss $\mathcal{L}(\theta)$ (7). Note that we consider two Q-functions to be distinct only if they differ on any state s in $\mathcal{D}$.

The REM loss $\mathcal{L}(\theta) = \mathbb{E}_{\alpha\sim P^\Delta}\left[\mathcal{L}(\alpha,\theta)\right]$ where $\mathcal{L}(\alpha,\theta)$ is given by

$$\mathcal{L}(\alpha,\theta) \;=\; \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\ell_\delta\left(\sum_k \alpha_k Q^k_\theta(s,a) - r - \gamma \max_{a'}\sum_k \alpha_k Q^k_{\theta'}(s',a')\right)\right]. \tag{8}$$

If the heads Q^i_θ and Q^j_θ don't converge to identical Q-values at the global minimum of $\mathcal{L}(\theta)$, it can be deduced using Lemma 1 that all the Q-functions given by the convex combination $\alpha_i Q^i_\theta + \alpha_j Q^j_\theta$ such that $\alpha_i + \alpha_j = 1$ minimizes the loss in (3). This contradicts the assumption that only a finite number of distinct Q-functions globally minimize the loss in (3). Hence, all Q-heads represent an identical Q-function at the global minimum of $\mathcal{L}(\theta)$.

Lemma 1. *Assuming that the distribution* P_Δ *has full support over the entire* $(K-1)$*-simplex* $\Delta^{\mathrm{K}-1}$*, then at any global minimum of* $\mathcal{L}(\theta)$*, the Q-function heads* Q^k_θ *for* $k = 1,\ldots,K$ *minimize* $\mathcal{L}(\alpha,\theta)$ *for any* $\alpha \in \Delta^{\mathrm{K}-1}$*.*

Proof. Let $Q_{\alpha^*,\theta^*} = \sum_{k=1}^K \alpha^*_k Q^k_{\theta^*}(s,a)$ corresponding to the convex combination $\alpha^* = (\alpha^*_1,\cdots,\alpha^*_K)$ represents one of the global minima of $\mathcal{L}(\alpha,\theta)$ (8) *i.e.,* $\mathcal{L}(\alpha^*,\theta^*) = \min_{\alpha,\theta}\mathcal{L}(\alpha,\theta)$ where $\alpha \in \Delta^{\mathrm{K}-1}$. Any global minima of $\mathcal{L}(\theta)$ attains a value of $\mathcal{L}(\alpha^*,\theta^*)$ or higher since,

$$\mathcal{L}(\theta) = \mathbb{E}_{\alpha\sim P^\Delta}\left[\mathcal{L}(\alpha,\theta)\right] \geq \mathbb{E}_{\alpha\sim P^\Delta}\left[\mathcal{L}(\alpha^*,\theta^*)\right] \geq \mathcal{L}(\alpha^*,\theta^*) \tag{9}$$

Let $Q^k_{\theta^*}(s,a) = w^k_{\theta^*}\cdot f_{\theta^*}(s,a)$ where $f_{\theta^*}(s,a) \in \mathbb{R}^D$ represent the shared features among the Q-heads and $w^k_{\theta^*} \in \mathbb{R}^D$ represent the weight vector in the final layer corresponding to the k-th head. Note that Q_{α^*,θ^*} can also be represented by each of the individual Q-heads using a weight vector given by convex combination α^* of weight vectors $(w^1_{\theta^*},\cdots,w^K_{\theta^*})$, *i.e.*, $Q(s,a) = \left(\sum_{k=1}^K \alpha^*_k w^k_{\theta^*}\right)\cdot f_{\theta^*}(s,a)$.

Let θ^I be such that $Q^k_{\theta^I} = Q_{\alpha^*,\theta^*}$ for all Q-heads. By definition of Q_{α^*,θ^*}, for all $\alpha \sim \mathrm{P}_\Delta$, $\mathcal{L}(\alpha,\theta^I) = \mathcal{L}(\alpha^*,\theta^*)$ which implies that $\mathcal{L}(\theta^I) = \mathcal{L}(\alpha^*,\theta^*)$. Hence, θ^I corresponds to one of the global minima of $\mathcal{L}(\theta)$ and any global minima of $\mathcal{L}(\theta)$ attains a value of $\mathcal{L}(\alpha^*,\theta^*)$.

Since $\mathcal{L}(\alpha,\theta) \geq \mathcal{L}(\alpha^*,\theta^*)$ for any $\alpha \in \Delta^{\mathrm{K}-1}$, for any θ^M such that $\mathcal{L}(\theta^M) = \mathcal{L}(\alpha^*,\theta^*)$ implies that $\mathcal{L}(\alpha,\theta^M) = \mathcal{L}(\alpha^*,\theta^*)$ for any $\alpha \sim \mathrm{P}_\Delta$. Therefore, at any global minimum of $\mathcal{L}(\theta)$, the Q-function heads Q^k_θ for $k = 1,\ldots,K$ minimize $\mathcal{L}(\alpha,\theta)$ for any $\alpha \in \Delta^{\mathrm{K}-1}$.

A.2 OFFLINE CONTINUOUS CONTROL EXPERIMENTS

We replicated the *final buffer* setup as described by Fujimoto et al. (2019): We train a DDPG (Lillicrap et al., 2015) agent for 1 million time steps three standard MuJoCo continuous control environments in OpenAI gym (Todorov et al., 2012; Brockman et al., 2016), adding $\mathcal{N}(0, 0.5)$ Gaussian noise to actions for high exploration, and store all experienced transitions. This collection procedure creates a dataset with a diverse set of states and actions, with the aim of sufficient coverage. Similar to Fujimoto et al. (2019), we train DDPG across 15 seeds, and select the 5 top performing seeds for dataset collection.

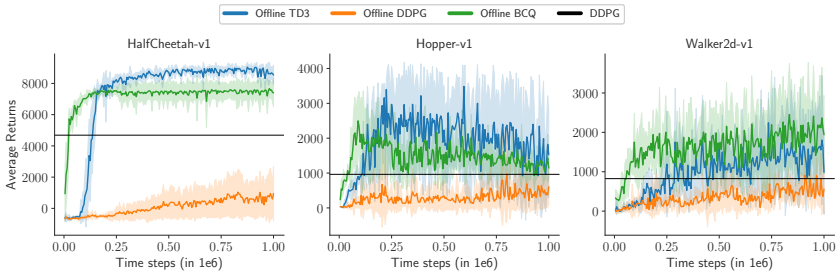


Figure A.1: We examine the performance of DDPG, TD3 and BCQ trained using identical offline data on three standard MuJoCo environments. We plot the mean performance (evaluated without exploration noise) across 5 runs. The shaded area represents a standard deviation. The bold black line measures the average return of episodes contained in the offline data collected using the DDPG agent (with exploration noise).

Using this logged dataset, we train standard continuous control off-policy actor-critic methods namely DDPG and TD3 (Fujimoto et al., 2018) completely offline without any exploration. We also train a Batch-Constrained deep Q-learning (BCQ) agent, proposed by Fujimoto et al. (2019), which restricts the action space to force the offline agent towards behaving close to on-policy w.r.t. a subset of the given data. We use the open source code generously provided by the authors at `https://github.com/sfujim/BCQ` and `https://github.com/sfujim/TD3`. We use the hyperparameters mentioned in (Fujimoto et al., 2018; 2019) except offline TD3 which uses a learning rate of 0.0005 for both the actor and critic.

Figure A.1 show that offline TD3 significantly outperforms the behavior policy which collected the offline data as well as the offline DDPG agent. Noticeably, offline TD3 also performs comparably to BCQ, an algorithm designed specifically to learn from arbitrary, fixed offline data. While Fujimoto et al. (2019) attribute the failure to learn in the offline setting to extrapolation error (*i.e.,* the mismatch between the offline dataset and true state-action visitation of the current policy), our results suggest that failure to learn from diverse offline data may be linked to extrapolation error for only weak exploitation agents such as DDPG.

A.3 Score Normalization

The improvement in normalized performance of an offline agent, expressed as a percentage, over an online DQN (Nature) (Mnih et al., 2015) agent is calculated as: $100 \times (\text{Score}_{\text{Normalized}} - 1)$ where:

$$\text{Score}_{\text{Normalized}} = \frac{\text{Score}_{\text{Agent}} \; - \; \min(\text{Score}_{\text{DQN}}, \; \text{Score}_{\text{Random}})}{\max(\text{Score}_{\text{DQN}}, \; \text{Score}_{\text{Random}}) \; - \; \min(\text{Score}_{\text{DQN}}, \; \text{Score}_{\text{Random}})}. \tag{10}$$

Here, the scores are the mean evaluation scores averaged over 5 runs. We chose not to measure performance in terms of percentage of online DQN scores alone because a tiny difference relative to the random agent on some games can translate into hundreds of percent in DQN score difference. Additionally, the max is needed since DQN performs worse than a random agent on the games Solaris and Skiing.

A.4 Hyperparameters & Experiment Details

In our experiments, we used the hyperparameters provided in Dopamine baselines (Castro et al., 2018) and report them for completeness and ease of reproducibility in Table 2. As mentioned by Dopamine's GitHub repository, changing these parameters can significantly affect performance, without necessarily being indicative of an algorithmic difference. We will also open source our code to further aid in reproducing our results.

The Atari environments (Bellemare et al., 2013) used in our experiments are stochastic due to sticky actions (Machado et al., 2018), *i.e.,* there is 25% chance at every time step that the environment will execute the agent's previous action again, instead of the agent's new action. All agents (online or offline) are compared using the best evaluation score (averaged over 5 runs) achieved during training where the evaluation is done online every training iteration using a ϵ-greedy policy with $\epsilon = 0.001$. We report offline training results with same hyperparameters over 5 random seeds of the DQN replay data collection, game simulator and network initialization.

Table 2: The hyperparameters used by the offline and online RL agents in our experiments.

Hyperparameter	setting (for both variations)
Sticky actions	Yes
Sticky action probability	0.25
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Frame skip (Action repetitions)	4
Reward clipping	[-1, 1]
Terminal condition	Game Over
Max frames per episode	108K
Discount factor	0.99
Mini-batch size	32
Target network update period	every 2000 updates
Training steps per iteration	250K
Update period every	4 steps
Evaluation ϵ	0.001
Evaluation steps per iteration	125K
Q-network: channels	32, 64, 64
Q-network: filter size	8×8, 4×4, 3×3
Q-network: stride	4, 2, 1
Q-network: hidden units	512
Multi-head Q-network: number of Q-heads	200
Hardware	Tesla P100 GPU

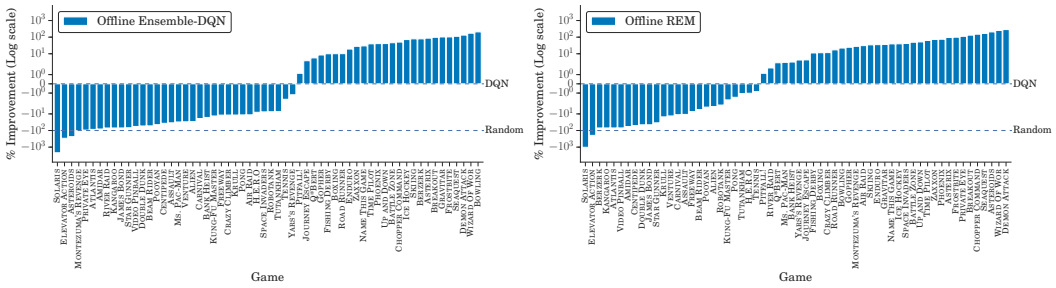
Hyperparameter	Online	Offline
Min replay size for sampling	20,000	-
Training ϵ (for ϵ-greedy exploration)	0.01	-
ϵ-decay schedule	250K steps	-
Fixed Replay Memory	No	Yes
Replay Memory size	1,000,000 steps	50,000,000 steps
Replay Scheme	Uniform	Uniform
Training Iterations	200	200 or 1000

DQN replay dataset collection. For collecting the offline data used in our experiments, we use online DQN (Nature) (Mnih et al., 2015) with the RMSprop (Tieleman and Hinton, 2012) optimizer. The DQN replay dataset, $\mathcal{B}_{\text{DQN}}$, consists of approximately 50 million experience tuples for each run per game corresponds to 200 million frames due to frame skipping of four, *i.e.,* repeating a selected action for four consecutive frames. Note that the total dataset size is approximately 15 billion tuples ($50 \ \frac{\text{million tuples}}{\text{agent}} * 5 \ \frac{\text{agents}}{\text{game}} * 60$ games).

Optimizer related hyperparameters. For existing off-policy agents, step size and optimizer were taken as published. Offline DQN (Adam) and all the offline agents with multi-head Q-network (Figure 3) use the Adam optimizer (Kingma and Ba, 2015) with same hyperparameters as online QR-DQN (Dabney et al., 2018b) ($\text{lr} = 0.00005$, $\epsilon_{\text{Adam}} = 0.01/32$). Note that scaling the loss has the same effect as inversely scaling ϵ_{Adam} when using Adam.

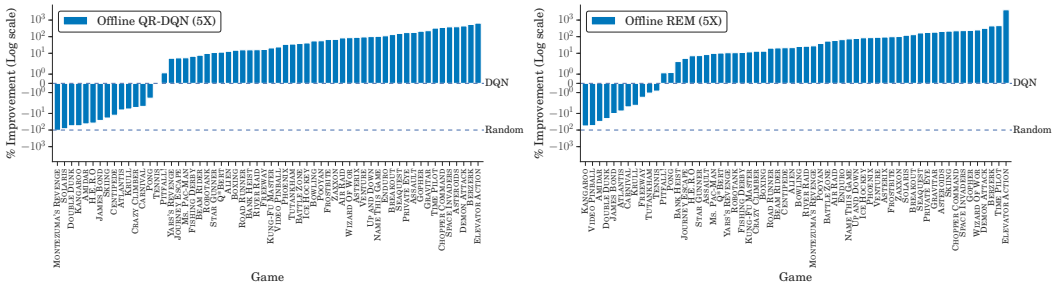
Online Agents. For online REM shown in Figure 4b, we performed hyper-parameter tuning over ϵ_{Adam} in $(0.01/32, 0.005/32, 0.001/32)$ over 5 training games (Asterix, Breakout, Pong, Q*Bert, Seaquest) and evaluated on the full set of 60 Atari 2600 games using the best setting ($\text{lr} = 0.00005$, $\epsilon_{\text{Adam}} = 0.001/32$). Online REM uses 4 Q-value estimates calculated using separate Q-networks where each network has the same architecture as originally used by online DQN (Nature). Similar to REM, our version of Bootstrapped-DQN also uses 4 separate Q-networks and Adam optimizer with identical hyperaparmeters ($\text{lr} = 0.00005$, $\epsilon_{\text{Adam}} = 0.001/32$).

A.5 Additional Plots & Tables



(a) (b)

Figure A.2: Normalized Performance improvement (in %) over online DQN (Nature), per game, of (a) offline Ensemble-DQN and (b) offline REM trained using the DQN replay dataset for same number of gradient steps as online DQN. The normalized online score for each game is 0.0 and 1.0 for the worse and better performing agent among fully trained online DQN and random agents respectively.



(a) (b)

Figure A.3: Normalized Performance improvement (in %) over online DQN (Nature), per game, of (a) offline QR-DQN (5X) (b) offline REM (5X) trained using the DQN replay dataset for five times as many gradient steps as online DQN. The normalized online score for each game is 0.0 and 1.0 for the worse and better performing agent among fully trained online DQN and random agents respectively.

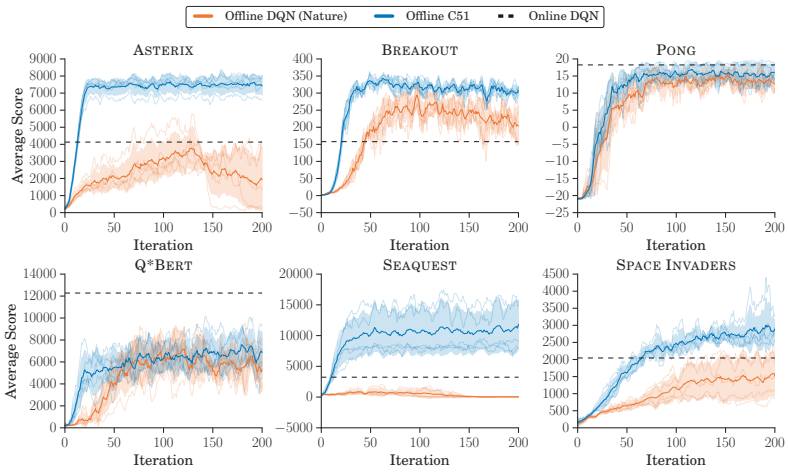
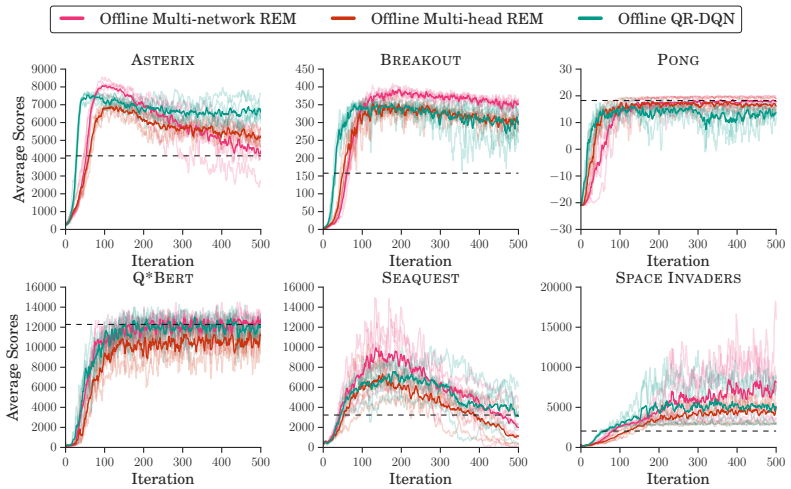
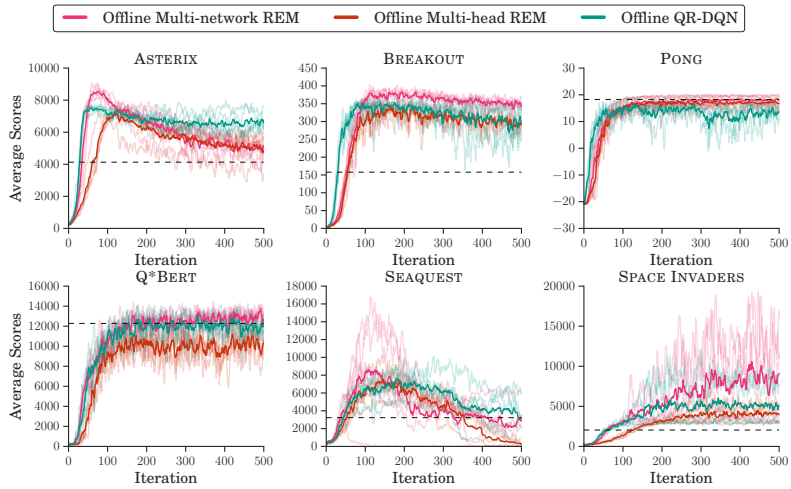


Figure A.4: **Offline C51 vs Offline DQN (Nature).** Average online scores of C51 and DQN (Nature) agents trained offline on 6 Atari 2600 games using the DQN replay dataset for the same number of gradient steps as the online DQN agent. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation.



(a) REM with 4 Q-value estimates ($K = 4$)



(b) REM with 16 Q-value estimates ($K = 16$)

Figure A.5: **REM with Separate Q-networks.** Average online scores of offline REM variants with different architectures and QR-DQN trained on 6 Atari 2600 games for 500 iterations using the DQN replay dataset. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation. The multi-network REM and the multi-head REM employ K Q-value estimates computed using separate Q-networks and Q-heads of a multi-head Q-network respectively and are optimized with identical hyperparameters. Multi-network REM improves upon the multi-head REM indicating that the more diverse Q-estimates provided by the separate Q-networks improve performance of REM over Q-estimates provided by the multi-head Q-network with shared features.

Table 3: Median normalized scores (Section A.3) across 60 Atari 2600 games, measured as percentages and number of games where an agent achieves better scores than a fully trained online DQN (Nature) agent. All the offline agents below are trained using the DQN replay dataset. The entries of the table without any suffix report training results with the five times as many gradient steps as online DQN while the entires with suffix ($1x$) indicates the same number of gradient steps as the online DQN agent. All the offline agents except DQN use the same multi-head architecture as QR-DQN.

Offline agent	Median	> DQN	Offline agent	Median	> DQN
DQN (Nature) ($1x$)	74.4%	10	DQN (Nature)	83.4%	17
DQN (Adam) ($1x$)	104.6%	39	DQN (Adam)	111.9%	41
Ensemble-DQN ($1x$)	92.5%	26	Ensemble-DQN	111.0%	39
Averaged Ensemble-DQN ($1x$)	88.6%	24	Averaged Ensemble-DQN	112.1%	43
QR-DQN ($1x$)	**115.0%**	**44**	QR-DQN	118.9%	45
REM ($1x$)	103.7%	35	REM	**123.8%**	**49**

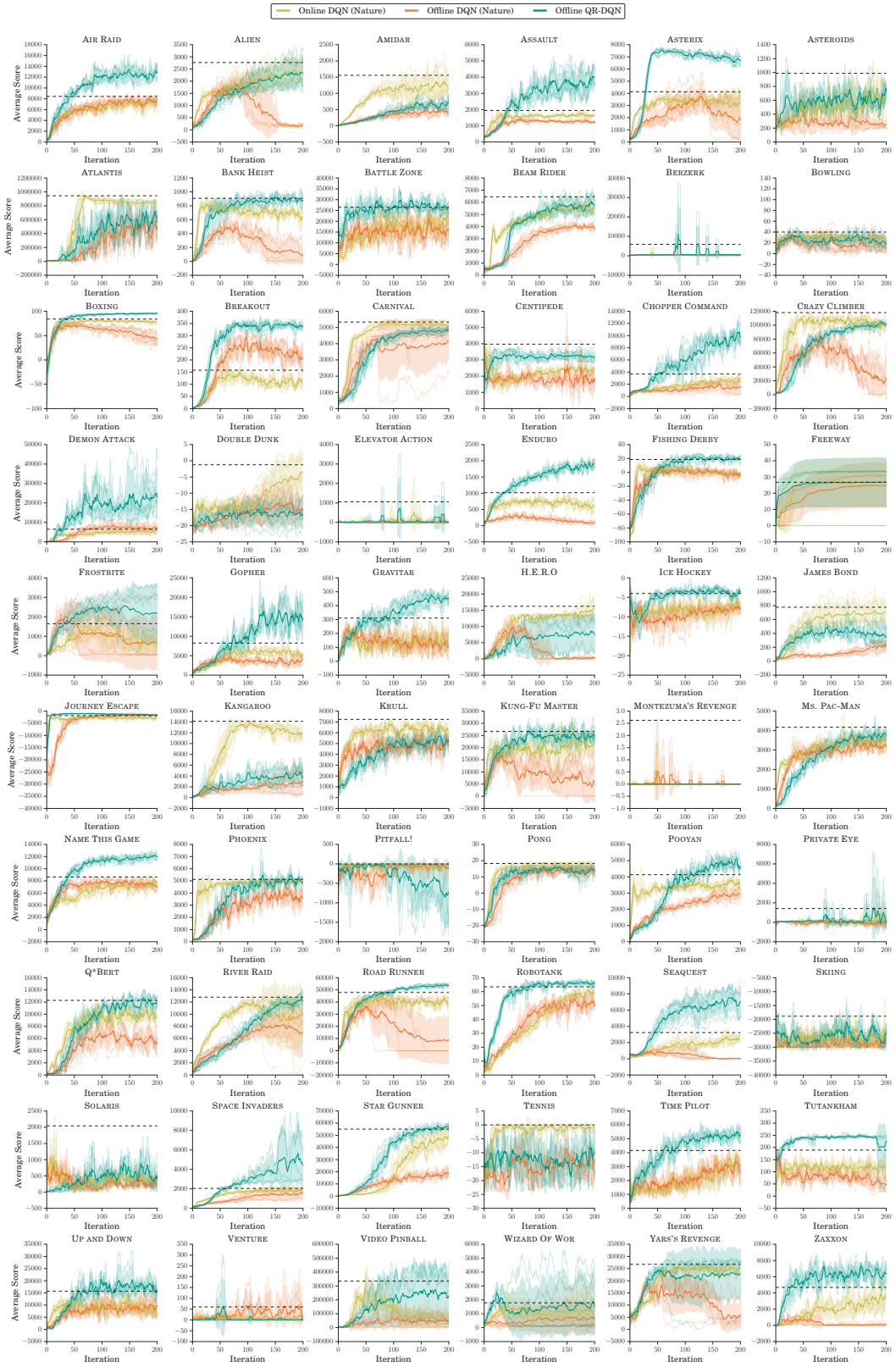


Figure A.6: Average evaluation scores across 60 Atari 2600 games for online DQN, offline DQN and offline QR-DQN trained for 200 iterations. The offline agents are trained using the DQN replay dataset. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation. The horizontal line shows the performance of the best policy (averaged over 5 runs) found during training of online DQN.

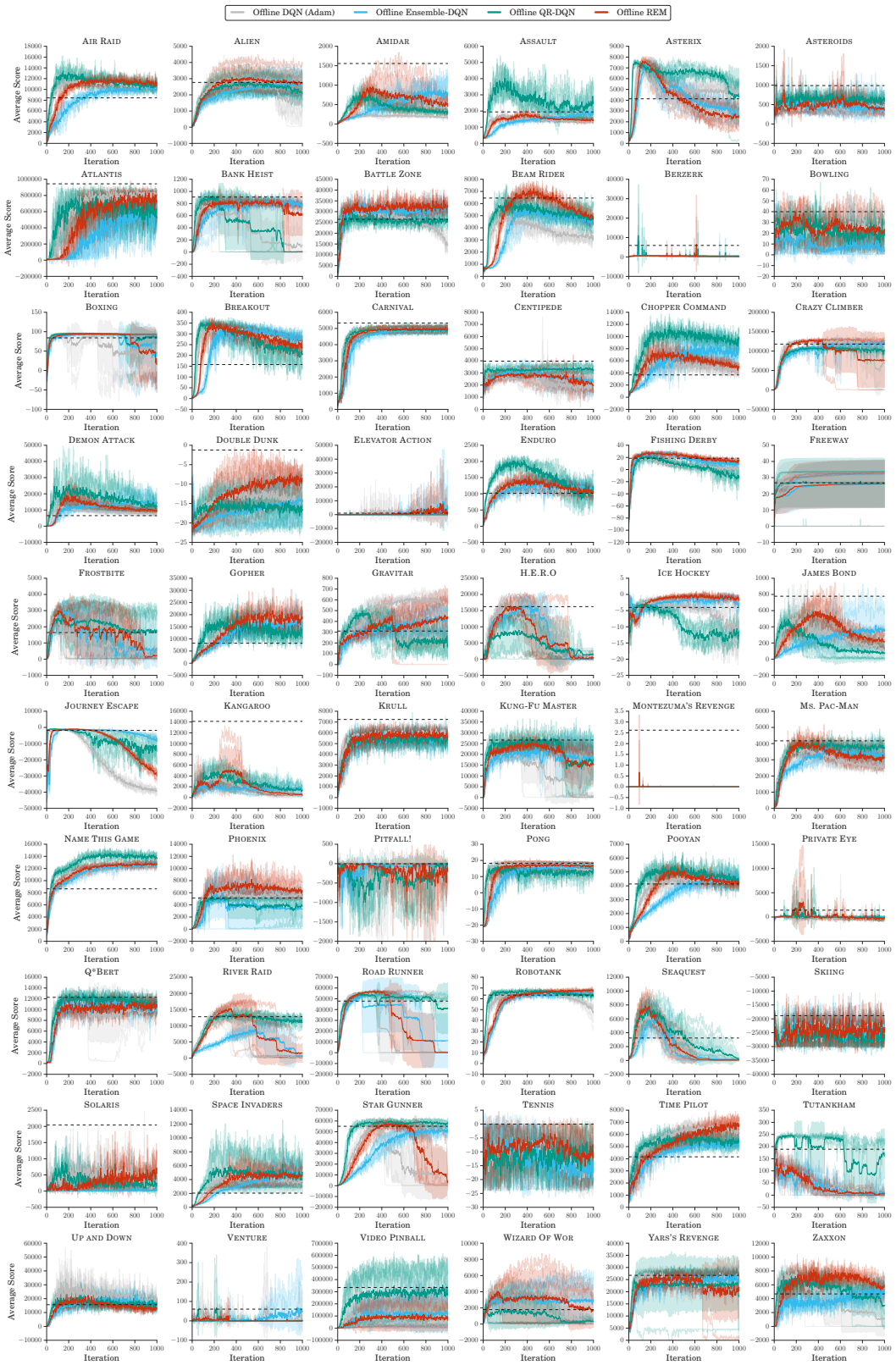


Figure A.7: Average evaluation scores across 60 Atari 2600 games of DQN (Adam), Ensemble-DQN, QR-DQN and REM agents trained offline using the DQN replay dataset. The horizontal line for online DQN show the best evaluation performance it obtains during training. All the offline agents except DQN use the same multi-head architecture with $K = 200$ heads. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation.

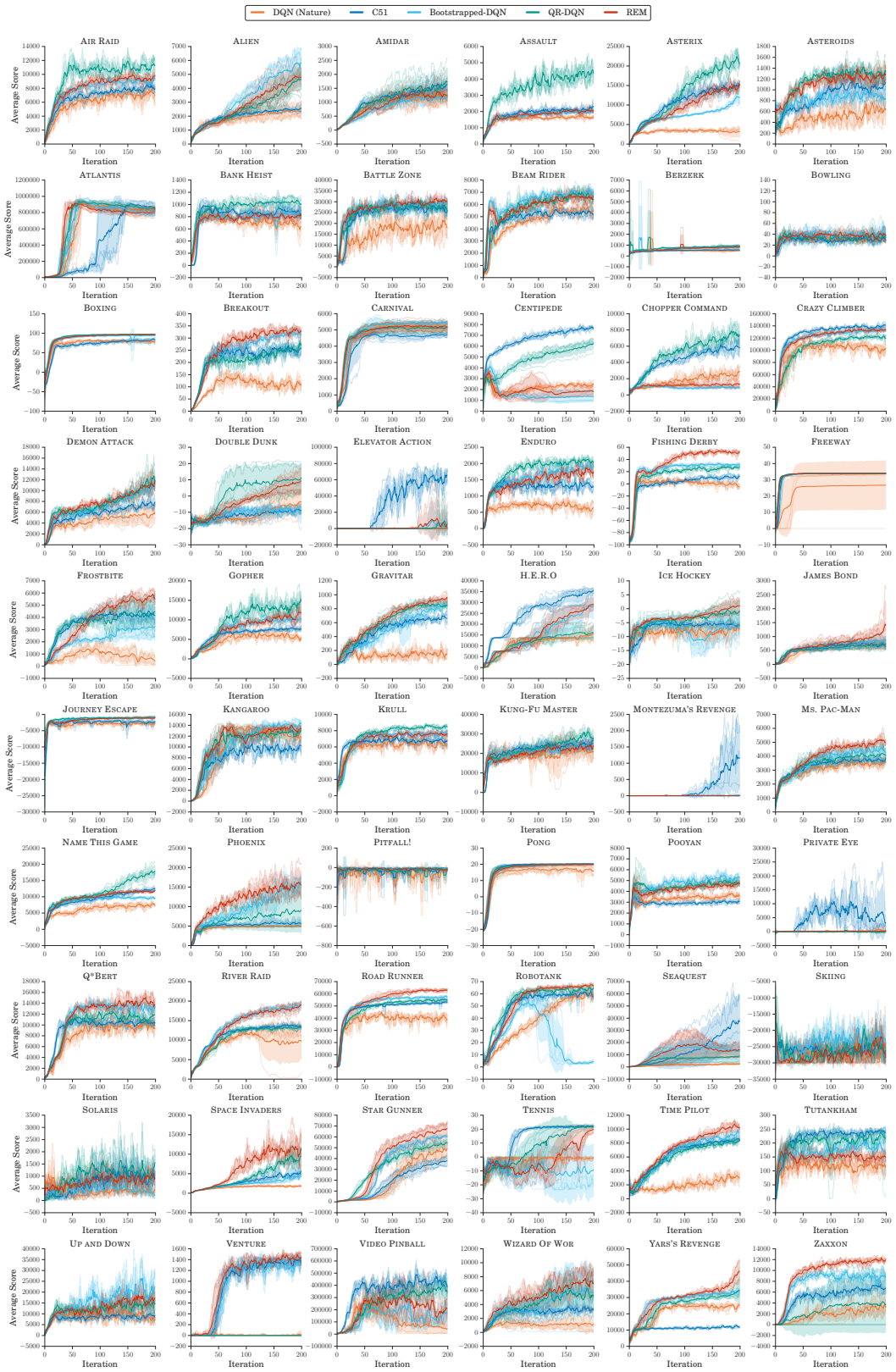


Figure A.8: Average evaluation scores across 60 Atari 2600 games of DQN, C51, QR-DQN, Bootstrapped-DQN and REM agents trained online for 200 million game frames (standard protocol). The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation.