# An Optimistic Perspective on Offline Reinforcement Learning

**Rishabh Agarwal** [1] **Dale Schuurmans** [1 2] **Mohammad Norouzi** [1]

## Abstract

Off-policy reinforcement learning (RL) using a fixed offline dataset of logged interactions is an important consideration in real world applications. This paper studies offline RL using the DQN Replay Dataset comprising the entire replay experience of a DQN agent on 60 Atari 2600 games. We demonstrate that recent off-policy deep RL algorithms, even when trained solely on this fixed dataset, outperform the fully-trained DQN agent. To enhance generalization in the offline setting, we present Random Ensemble Mixture (REM), a robust $Q$-learning algorithm that enforces optimal Bellman consistency on random convex combinations of multiple $Q$-value estimates. Offline REM trained on the DQN Replay Dataset surpasses strong RL baselines. Ablation studies highlight the role of offline dataset size and diversity as well as the algorithm choice in our positive results. Overall, the results here present an optimistic view that robust RL algorithms used on sufficiently large and diverse offline datasets can lead to high quality policies. To provide a testbed for offline RL and reproduce our results, the DQN Replay Dataset is released at offline-rl.github.io.

## 1 Introduction

One of the main reasons behind the success of deep learning is the availability of large and diverse datasets such as ImageNet (Deng et al., 2009) to train expressive deep neural networks. By contrast, most reinforcement learning (RL) algorithms (Sutton & Barto, 2018) assume that an agent interacts with an online environment or simulator and learns from its own collected experience. This limits online RL's applicability to complex real world problems, where active data collection means gathering large amounts of diverse data from scratch per experiment, which can be expensive, unsafe, or require a high-fidelity simulator that is often difficult to build (Dulac-Arnold et al., 2019).

Offline RL concerns the problem of learning a policy from a fixed dataset of trajectories, without any further interactions with the environment. This setting can leverage the vast amount of existing logged interactions for real world decision-making problems such as robotics (Cabi et al., 2019; Dasari et al., 2019), autonomous driving (Yu et al., 2018), recommendation systems (Strehl et al., 2010; Bottou et al., 2013), and healthcare (Shortreed et al., 2011). The effective use of such datasets would not only make real-world RL more practical, but would also enable better generalization by incorporating diverse prior experiences.

In offline RL, an agent does not receive any new corrective feedback from the online environment and needs to generalize from a fixed dataset of interactions to new online interactions during evaluation. In principle, off-policy algorithms can learn from data collected by any policy, however, recent work (Fujimoto et al., 2019b; Kumar et al., 2019; Wu et al., 2019; Siegel et al., 2020) presents a discouraging view that standard off-policy deep RL algorithms diverge or otherwise yield poor performance in the offline setting. Such papers propose remedies by regularizing the learned policy to stay close to the training dataset of offline trajectories. Furthermore, Zhang & Sutton (2017) assert that a large replay buffer can even hurt the performance of off-policy algorithms due to its "off-policyness".

By contrast, this paper presents an optimistic perspective on offline RL that with sufficiently large and diverse datasets, robust RL algorithms, without an explicit correction for distribution mismatch, can result in high quality policies. The contributions of this paper can be summarized as:

- An offline RL setup is proposed for evaluating algorithms on Atari 2600 games (Bellemare et al., 2013), based on the logged replay data of a DQN agent (Mnih et al., 2015) comprising 50 million (observation, action, reward, next observation) tuples per game. This setup reduces the computation cost of the experiments considerably and helps improve reproducibility by standardizing training using a fixed dataset. The DQN Replay Dataset and our code[1] is released to enable offline optimization

[1]Google Research, Brain Team [2]University of Alberta. Correspondence to: Rishabh Agarwal <rishabhagarwal@google.com>, Mohammad Norouzi <mnorouzi@google.com>.

*Proceedings of the $37^{th}$ International Conference on Machine Learning*, Vienna, Austria, PMLR 119, 2020. 

[1]Open-source code at github.com/google-research/batch_rl.

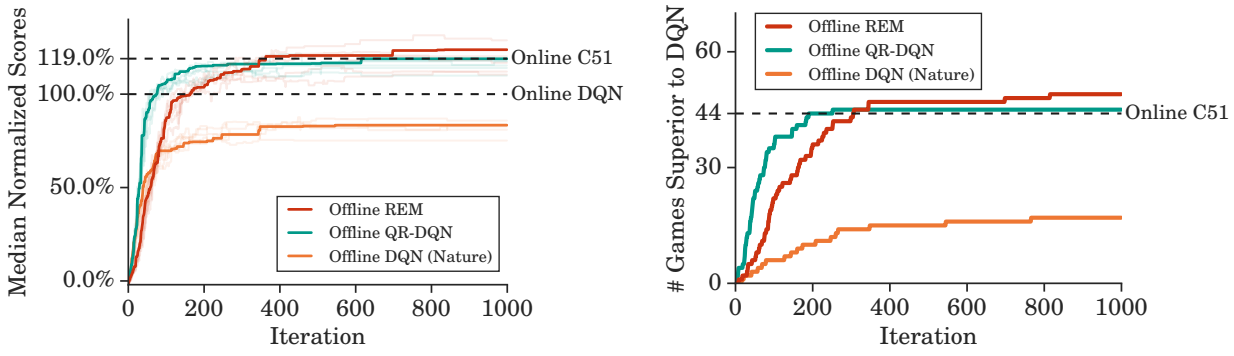



Figure 1: **Offline RL on Atari 2600.** (a) Median normalized evaluation scores averaged over 5 runs (shown as traces) across stochastic version of 60 Atari 2600 games of offline agents trained using the DQN replay dataset. (b) Number of games where an offline agent achieves a higher score than fully-trained DQN (Nature) as a function of training iterations. Each iteration corresponds to 1 million training frames. Offline REM outperforms offline QR-DQN and DQN (Nature). The comparison with online C51 gives a sense to the reader about the magnitude of the improvement from offline agents over the best policy in the entire DQN replay dataset.

of RL algorithms on a common ground.

- Contrary to recent work, we show that recent off-policy RL algorithms trained solely on offline data can be successful. For instance, offline QR-DQN (Dabney et al., 2018) trained on the DQN replay dataset outperforms the *best* policy in the DQN replay dataset. This discrepancy is attributed to the differences in offline dataset size and diversity as well as the choice of RL algorithm.
- A robust $Q$-learning algorithm called *Random Ensemble Mixture (REM)* is presented, which enforces optimal Bellman consistency on random convex combinations of multiple $Q$-value estimates. Offline REM shows strong generalization performance in the offline setting, and outperforms offline QR-DQN. The comparison with online C51 (Bellemare et al., 2017), a strong RL baseline illustrates the relative size of the gains from exploitation of the logged DQN data with REM.

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

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

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

$$Q^*(s,a) = \mathbb{E}\, R(s,a) + \gamma \mathbb{E}_{s' \sim P} \max_{a' \in \mathcal{A}} Q^*(s', a') \,. \quad (2)$$

To learn a policy from interaction with the environment, $Q$-learning (Watkins & Dayan, 1992) iteratively improves an approximate estimate of $Q^*$, denoted $Q_\theta$, 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 $\theta'$ are updated to the current $Q$-network parameters $\theta$ after a fixed number of time steps.

DQN (Mnih et al., 2013; 2015) parameterizes $Q_\theta$ with a convolutional neural network (LeCun et al., 1998) and uses $Q$-learning with a target network while following an $\epsilon$-greedy policy with respect to $Q_\theta$ for data collection. DQN minimizes the temporal difference (TD) error $\Delta_\theta$ 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:

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

where $l_\lambda$ is the Huber loss (Huber, 1964) given by

$$\ell_\lambda(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \lambda \\ \lambda(|u| - \frac{1}{2}\lambda), & \text{otherwise.} \end{cases} \quad (4)$$

$Q$-learning is an *off-policy* algorithm (Sutton & Barto, 2018) since the learning target can be computed without any consideration of how the experience was generated.

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

$$Z^*(s,a) \stackrel{D}{=} r + \gamma Z^*(s', \text{argmax}_{a' \in \mathcal{A}} Q^*(s',a')), \quad (5)$$
$$\text{where } r \sim R(s,a),\ s' \sim P(\cdot \mid s,a).$$

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., 2018) approximates the return density by using a uniform mixture of $K$ Dirac delta functions, *i.e.,*

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

QR-DQN 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 Reinforcement Learning

Modern off-policy deep RL algorithms (as discussed above) 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 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.

Offline RL, in contrast to online RL, describes the fully off-policy setting of learning using a fixed dataset of experiences, without any further interactions with the environment. 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. The offline RL setting removes design choices related to the replay buffer and exploration; therefore, it is simpler to experiment with and reproduce than the online setting.

Offline RL is considered challenging due to the *distribution mismatch* between the current policy and the offline data collection policy, *i.e.,* when the policy being learned takes a different action than the data collection policy, we don't know the reward that should be provided. This paper revisits offline RL and investigates whether off-policy deep RL agents trained solely on offline data can be successful without correcting for distribution mismatch.

## 4 Developing Robust Offline RL Algorithms

In an online RL setting, an agent can acquire on-policy data from the environment, which ensures a virtuous cycle where the agent chooses actions that it thinks will lead to high rewards and then receives feedback to correct its errors. Since it is not possible to collecting additional data in the offline RL setting, it is necessary to reason about generalization using the fixed dataset. We investigate whether one can design robust RL algorithms with an emphasis on improving generalization in the offline setting. Ensembling is commonly used in supervised learning to improve generalization. In this paper, 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 & 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,

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

where $l_\lambda$ 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 mod-

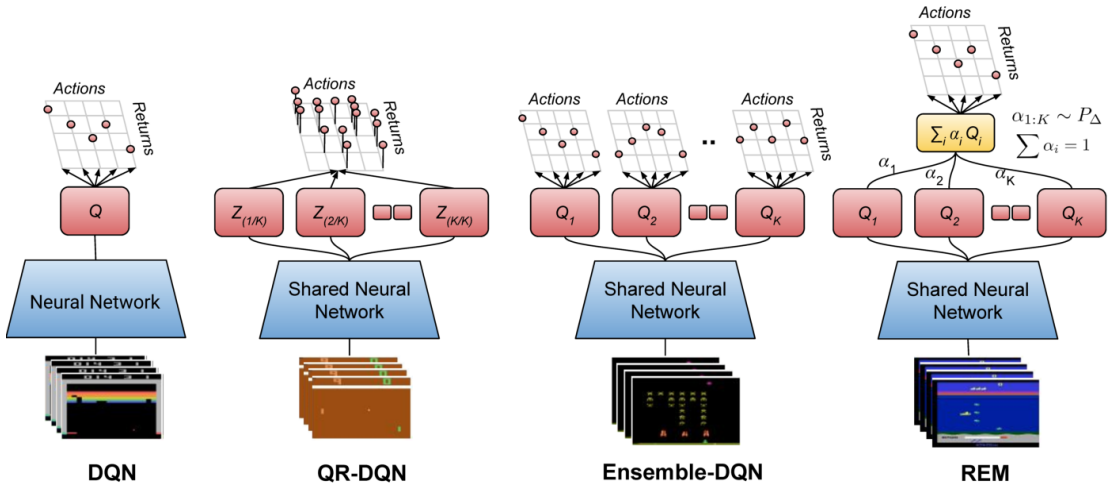



Figure 2: Neural network architectures for DQN, distributional QR-DQN and the proposed expected RL variants, *i.e.,* Ensemble-DQN and REM, with the same multi-head architecture as QR-DQN. The individual $Q$-heads share all of the neural network layers except the final fully connected layer. In QR-DQN, each head (red rectangles) corresponds to a specific quantile of the return distribution, while in the proposed variants, each head approximates the optimal $Q$-function.

els (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 off-policy 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 $Q$-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 $\alpha$, 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,

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[ \mathbb{E}_{\alpha \sim \mathrm{P}_\Delta} \left[ \ell_\lambda \left( \Delta^\alpha_\theta(s,a,r,s') \right) \right] \right], \quad (7)$$

$$\Delta^\alpha_\theta = \sum_k \alpha_k Q^k_\theta(s,a) - r - \gamma \max_{a'} \sum_k \alpha_k Q^k_{\theta'}(s',a')$$

where $\mathrm{P}_\Delta$ represents a probability distribution over the standard $(K-1)$-simplex $\Delta^{\mathrm{K}-1} = \{\alpha \in \mathbb{R}^{\mathrm{K}} : \alpha_1 + \alpha_2 + \cdots + \alpha_K = 1, \alpha_k \geq 0, k = 1, \ldots, K\}$.

REM considers $Q$-learning as a constraint satisfaction problem based on Bellman optimality constraints (2) and $\mathcal{L}(\theta)$ can be viewed as an infinite set of 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.,* $Q(s,a) = \sum_k Q^k_\theta(s,a)/K$. 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 $\mathrm{P}_\Delta$: 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* $\mathrm{P}_\Delta$ *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.

## 5 Offline RL on Atari 2600 Games

To facilitate a study of offline RL, we create the DQN Replay Dataset by training 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 protocol) and sticky actions (Machado et al., 2018) (with 25% probability, the agent's previous action is executed instead of the current action). On each game, we train 5 different agents with random initialization, and store all of the tuples of (observation, action, reward, next observation) en-

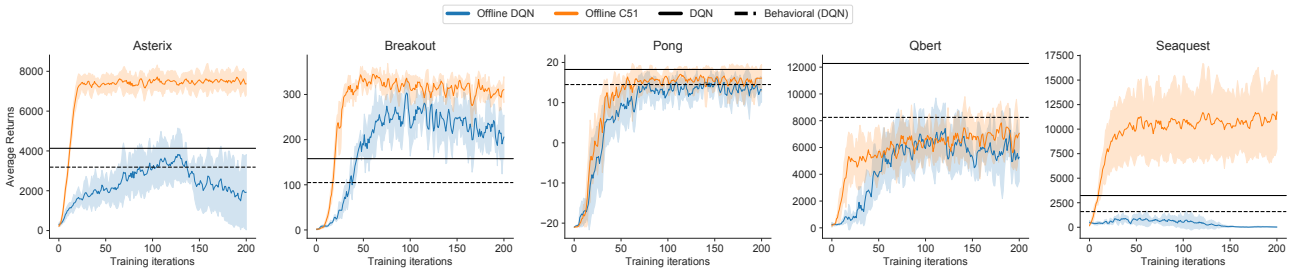
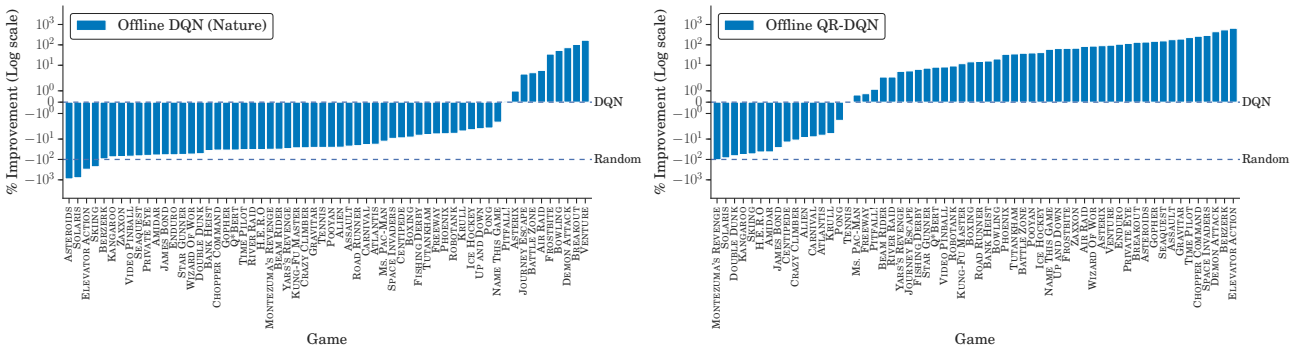



Figure 3: **Offline Agents on DQN Replay Dataset.** Average online scores of C51 and DQN (Nature) agents trained offline on 5 games with sticky actions for the same number of gradient steps as the online DQN agent. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation. The solid and dashed horizontal line show the performance of best policy (*i.e.,* fully-trained DQN) and the average behavior policy in the DQN Replay Dataset respectively.




(a) (b)

Figure 4: **Offline QR-DQN *vs.* DQN (Nature).** Normalized performance improvement (in %) over fully-trained 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 100% and 0% for online DQN and random agents respectively.

countered during training into 5 replay datasets per game resulting in a total of 300 datasets.

Each game replay dataset is approximately 3.5 times larger than ImageNet (Deng et al., 2009) and include samples from all of the intermediate (diverse) policies seen during the optimization of the online DQN agent. Figure A.4 shows the learning curves of the individual agents used for data collection as well as the performance of best policy found during training (which we use for comparison).

**Experiment Setup**. The DQN Replay Dataset is used for training RL agents, offline, without any interaction with the environment during training. We use the hyperparameters provided in Dopamine baselines (Castro et al., 2018) for a standardized comparison (Appendix A.4) and report game scores using a normalized scale (Appendix A.3). Although the game replay datasets contain data collected by a DQN agent improving over time as training progresses, we compare the performance of offline agents against the best performing agent obtained after training (*i.e.,* fully-trained DQN). The evaluation of the offline agents is done online for a limited number of times in the intervals of 1 million training frames. For each game, we evaluate the 5 offline agents trained (one per dataset), using online returns, reporting the best performance averaged across the 5 agents.

### 5.1 Can standard off-policy RL algorithms with no environment interactions succeed?

Given the logged replay data of the DQN agent, it is natural to ask how well an offline variant of DQN solely trained using this dataset would perform? Furthermore, whether more recent off-policy algorithms are able to exploit the DQN Replay Dataset more effectively than offline DQN. To investigate these questions, we train DQN (Nature) and QR-DQN agents, offline, on the DQN replay dataset for the same number of gradient updates as online DQN.

Figure 4 shows that offline DQN underperforms fully-trained online DQN on all except a few games where it achieves much higher scores than online DQN with the same amount of data and gradient updates. Offline QR-DQN, on the other hand, outperforms offline DQN and online DQN on most of the games (refer to Figure A.4 for learning curves). Offline C51 trained using the DQN Replay Dataset also considerably improves upon offline DQN (Figure 3). Offline QR-DQN outperforms offline C51.

Table 1: **Asymptotic performance of offline agents**. Median normalized scores (averaged over 5 runs) across 60 games and number of games where an offline agent trained using the DQN Replay Dataset achieves better scores than a fully-trained DQN (Nature) agent. Offline DQN (Adam) significantly outperforms DQN (Nature) and needs further investigation. Offline REM surpasses the gains (119%) from fully-trained C51, a strong online agent.

| Offline agent | Median | > DQN |
|---|---|---|
| DQN (Nature) | 83.4% | 17 |
| 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** |

These results demonstrate that it is possible to optimize strong Atari agents offline using standard deep RL algorithms on DQN Replay Dataset without constraining the learned policy to stay close to the training dataset of offline trajectories. Furthermore, the disparity between the performance of offline QR-DQN/C51 and DQN (Nature) indicates the difference in their ability to exploit offline data.

### 5.2 Asymptotic performance of offline RL agents

In supervised learning, asymptotic performance matters 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 for an offline dataset is fixed, we train offline agents for 5 times as many gradient updates as 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 & Hinton, 2012) with Adam (Kingma & 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 2), 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 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.

**Results**. The main comparisons are presented in Figure 1. Table 1 shows the comparison of baselines with REM and Ensemble-DQN. Surprisingly, DQN with Adam 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 QR-DQN, when trained for more gradient updates. Learning curves of all offline agents can be found in Figure A.5.

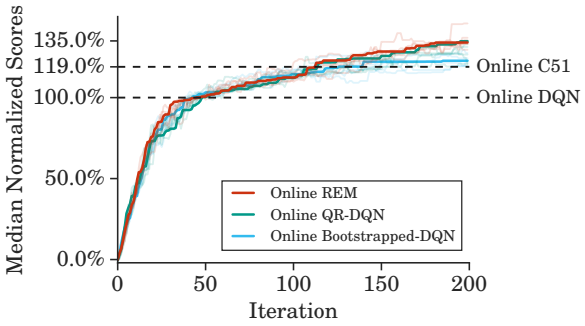



Figure 5: **Online REM *vs.* baselines**. Median normalized evaluation scores averaged over 5 runs (shown as traces) across stochastic version of 60 Atari 2600 games of online agents trained for 200 million frames (standard protocol). Online REM with 4 $Q$-networks performs comparably to online QR-DQN. Please refer to Figure A.6 for learning curves.

**Hypothesis about effectiveness of REM**. The gains from REM over Averaged Ensemble-DQN suggest that the effectiveness of REM is due to the noise from randomly ensembling Q-value estimates leading to more robust training, analogous to dropout. Consistent with this hypothesis, we also find that offline REM with separate $Q$-networks (with more variation in individual $Q$-estimates) performs better asymptotically and learns faster than a multi-head $Q$-network (Figure A.3). Furthermore, randomized ensembles as compared to a minimum over the $Q$-values for the target perform substantially better in our preliminary experiments on 5 games and requires further investigation.

### 5.3 Does REM work in the online setting?

In online RL, learning and data generation are tightly coupled, *i.e.,* an agent that learns faster also collects more relevant data. We ran online REM with 4 separate $Q$-networks because of the better convergence speed over multi-head REM in the offline setting. For data collection, we use $\epsilon$-greedy with a randomly sampled $Q$-estimate from the simplex for each episode, similar to Bootstrapped DQN. We follow the standard online RL protocol on Atari and use a fixed replay buffer of 1M frames.

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 5 shows that REM performs on par with QR-

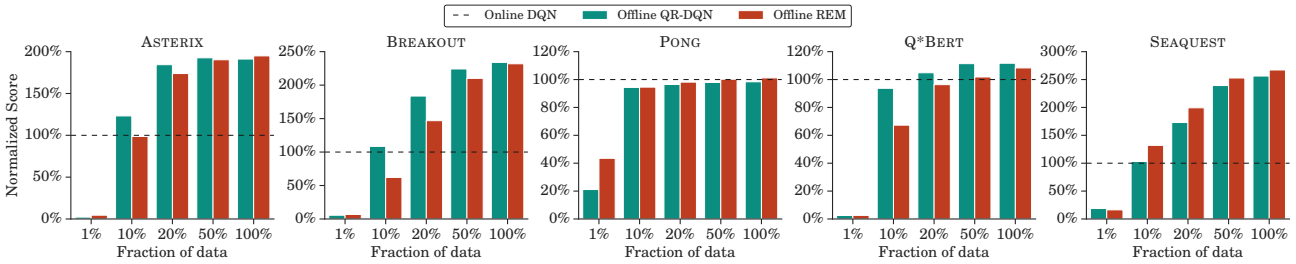



Figure 6: **Effect of Offline Dataset Size.** Normalized scores (averaged over 5 runs) of QR-DQN and multi-head REM trained offline on stochastic version of 5 Atari 2600 games for 5X gradient steps using only a fraction of the entire DQN Replay Dataset (200M frames) obtained via randomly subsampling trajectories. With only 10% of the entire replay dataset, REM and QR-DQN approximately recover the performance of fully-trained DQN.

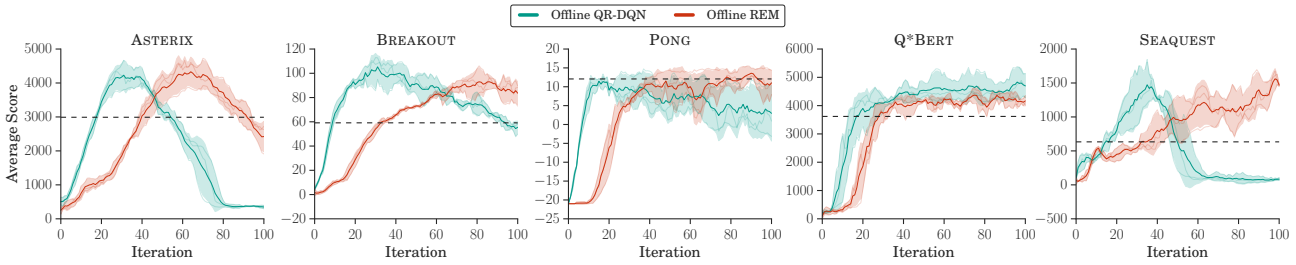



Figure 7: **Offline RL with Lower Quality Dataset.** Normalized scores (averaged over 3 runs) of QR-DQN and multi-head REM trained offline on stochastic version of 5 Atari 2600 games for 5X gradient steps using logged data from online DQN trained only for 20M frames (20 iterations). The horizontal line shows the performance of best policy found during DQN training for 20M frames which is significantly worse than fully-trained DQN. We observe qualitatively similar results to offline setting with entire replay dataset.

DQN and considerably outperforms Bootstrapped-DQN. This shows that we can use the insights gained from the offline setting with appropriate design choices (*e.g.,* exploration, replay buffer) to create effective online methods.

## 6 Important Factors in Offline RL

**Dataset Size and Diversity**. Our offline learning results indicate that 50 million tuples per game from DQN (Nature) are sufficient to obtain good online performance on most of the Atari 2600 games. Behavior cloning performs poorly on the DQN Replay Dataset due to its diverse composition. We hypothesize that the size of the DQN Replay Dataset and its diversity (De Bruin et al., 2015) play a key role in the success of standard off-policy RL algorithms trained offline.

To study the role of the offline dataset size, we perform an ablation experiment with variable replay size. We train offline QR-DQN and REM with reduced data obtained via randomly subsampling entire trajectories from the logged DQN experiences, thereby maintaining the same data distribution. Figure 6 presents the performance of the offline REM and QR-DQN agents with $N\%$ of the 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 $N = 1\%$.

To see the effect of quality of offline dataset, we perform another ablation where we train offline agents on the first 20 million frames in the DQN Replay Dataset – a lower quality dataset which roughly approximates exploration data with suboptimal returns. Similar to the offline results with the entire dataset, on most Atari games, offline REM and QR-DQN outperform the best policy in this dataset (Figure 7 and Figure A.7), indicating that standard RL agents work well with sufficiently diverse offline datasets.

**Algorithm Choice**. Even though continuous control is not the focus of this paper, we reconcile the discrepancy between our findings (Section 5.1) and the claims of Fujimoto et al. (2019b) that standard off-policy methods fail in the offline continuous control setting, even with large and diverse replay datasets. The results of Fujimoto et al. (2019b) 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. (2019b) (Appendix 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. Con-

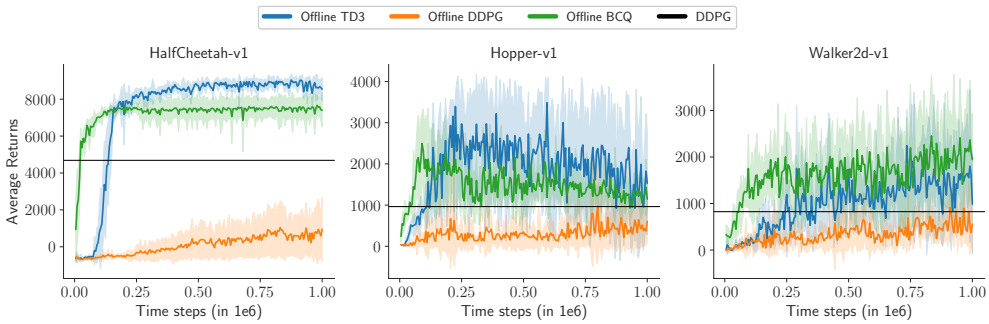



Figure 8: **Offline Continuous Control Experiments**. We examine the performance of DDPG, TD3 and BCQ (Fujimoto et al., 2019b) trained using identical offline data on three standard MuJoCo environments. We plot the mean performance (evaluated without exploration noise) across 5 runs. The shaded area 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). Similar to our results on Atari 2600 games, we find that recent off-policy algorithms perform quite well in the offline setting with entire DDPG replay data.

sistent with our offline results on Atari games, offline TD3 significantly outperforms the data collecting DDPG agent and offline DDPG (Figure 8). Offline TD3 also performs comparably to BCQ (Fujimoto et al., 2019b), an algorithm designed specifically to learn from offline data.

## 7 Related work

Our work is mainly related to batch RL [2] (Lange et al., 2012). Similar to (Ernst et al., 2005; Riedmiller, 2005; Jaques et al., 2019), we investigate batch off-policy RL, which requires learning a good policy given a fixed dataset of interactions. In our offline setup, we only assume access to samples from the behavior policy and focus on $Q$-learning methods without any form of importance correction, as opposed to (Swaminathan & Joachims, 2015; Liu et al., 2019).

Recent work (Fujimoto et al., 2019b; Kumar et al., 2019; Wu et al., 2019; Siegel et al., 2020) reports that standard off-policy methods trained on fixed datasets fail on continuous control environments. Fujimoto et al. (2019a) also observe that standard RL algorithms fail on Atari 2600 games when trained offline using trajectories collected by a single partially-trained DQN policy. The majority of recent papers focus on the offline RL setting with dataset(s) collected using a single data collection policy (*e.g.,* random, expert, *etc*) and propose remedies by regularizing the learned policy to stay close to training trajectories. These approaches improve stability in the offline setting, however, they introduce additional regularization terms and hyper-parameters, the selection of which is not straightforward (Wu et al., 2019). REM (Section 4.2) is orthogonal to these approaches and can be easily combined with them.

This paper focuses on the offline RL setting on Atari 2600 games with data collected from a large mixture of policies seen during the optimization of a DQN agent, rather than a single Markovian behavior policy. Our results demonstrate that recent off-policy deep RL algorithms (*e.g.,* TD3 (Fujimoto et al., 2018), QR-DQN (Dabney et al., 2018)) are effective in the offline setting with sufficiently large and diverse offline datasets, without explicitly correcting for the distribution mismatch between the learned policy and the offline dataset. We suspect that the suboptimal performance of offline DQN (Nature) is related to the notion of off-policyness of large buffers established by Zhang & Sutton (2017). However, robust deep $Q$-learning algorithms such as REM are able to effectively exploit DQN Replay, given sufficient number of gradient updates. Section 6 shows that dataset size and its diversity, as well as choice of the RL algorithms significantly affect offline RL results and explains the discrepancy with recent work. Inspired by our work, Cabi et al. (2019) successfully apply distributional RL algorithms on large-scale offline robotic datasets.

## 8 Future Work

Since the *(observation, action, next observation, reward)* tuples in DQN Replay Dataset are stored in the order they were experienced by online DQN during training, various data collection strategies for *benchmarking* offline RL can be induced by subsampling the replay dataset containing 200 million frames. For example, the first $k$ million frames from the DQN Replay Dataset emulate exploration data with suboptimal returns (*e.g.,* Figure 7) while the last $k$ million frames are analogous to near-expert data with stochasticity. Another option is to randomly subsample the entire dataset to create smaller offline datasets (*e.g.,* Figure 6). Based on the popularity and ease of experimentation on Atari 2600 games, the DQN Replay Dataset can be used for benchmarking offline RL in addition to continuous control setups such as BRAC (Wu et al., 2019).

As REM simply uses a randomized $Q$-ensemble, it is straightforward to combine REM with existing $Q$-learning methods including distributional RL. REM can be used for

[2] To avoid confusion with batch *vs.* minibatch optimization, we refer to batch RL as offline RL in this paper.

improving value baseline estimation in policy gradient and actor-critic methods (Weng, 2018). Using entropy regularization to create correspondence between $Q$-values and policies (*e.g.,* SAC (Haarnoja et al., 2018)) and applying REM to $Q$-values is another possibility for future work. REM can also be combined with behavior regularization methods such as SPIBB (Laroche et al., 2019) and BCQ (Fujimoto et al., 2019b) to create better offline RL algorithms.

Our results also emphasize the need for a rigorous characterization of the role of *generalization* due to deep neural nets when learning from offline data collected from a large mixture of (diverse) policies. We also leave further investigation of the exploitation ability of distributional RL as well as REM to future work.

Analogous to supervised learning, the offline agents exhibit overfitting, *i.e.,* after certain number of gradient updates, their performance starts to deteriorates. To avoid such overfitting, we currently employ online policy evaluation for early stopping, however, "true" offline RL requires offline policy evaluation for hyperparameter tuning and early stopping. We also observe divergence w.r.t. Huber loss in our reduced data experiments with $N = 1\%$. This suggests the need for offline RL methods that maintain reasonable performance throughout learning (Garcıa & Fernández, 2015).

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 another promising direction for improving sample-efficiency in RL.

## 9 Conclusion

This paper studies offline RL on Atari 2600 games based on logged experiences of a DQN agent. The paper demonstrates that standard RL methods can learn to play Atari games from DQN Replay Dataset, better than the best behavior in the dataset. This is in contrast with existing work, which claims standard methods fail in the offline setting. The DQN Replay Dataset can serve as a benchmark for offline RL. These results present a positive view that robust RL algorithms can be developed which can effectively learn from large-scale offline datasets. REM strengthens this optimistic perspective by showing that even simple ensemble methods can be effective in the offline setting.

Overall, the paper indicates the potential of offline RL for creating a *data-driven* RL paradigm where one could pretrain RL agents with large amounts of existing diverse datasets before further collecting new data via exploration, thus creating *sample-efficient* agents that can be deployed and continually learn in the real world.

## Acknowledgements

We thank Pablo Samuel Castro for help in understanding and debugging issues with the Dopamine codebase and reviewing an early draft of the paper. We thank Andrew Helton, Christian Howard, Robert Dadashi, Marlos C. Machado, Carles Gelada, Dibya Ghosh and Liam Fedus for helpful discussions. We also acknowledge Marc G. Bellemare, Zafarali Ahmed, Ofir Nachum, George Tucker, Archit Sharma, Aviral Kumar, William Chan and anonymous ICML reviewers for their review of the paper.

## References

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

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*, 2013.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. *ICML*, 2017.

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

Bottou, L., Peters, J., Quiñonero-Candela, J., Charles, D. X., Chickering, D. M., Portugaly, E., Ray, D., Simard, P., and Snelson, E. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Cabi, S., Colmenarejo, S. G., Novikov, A., Konyushkova, K., Reed, S., Jeong, R., Żołna, K., Aytar, Y., Budden, D., Vecerik, M., et al. A framework for data-driven robotics. *arXiv preprint arXiv:1909.12200*, 2019.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. Dopamine: A research framework for deep reinforcement learning. *ArXiv:1812.06110*, 2018.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. *AAAI*, 2018.

Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., Singh, S., Levine, S., and Finn, C. Robonet: Large-scale multi-robot learning. *CoRL*, 2019.

De Bruin, T., Kober, J., Tuyls, K., and Babuška, R. The importance of experience replay database composition in deep reinforcement learning. *Deep reinforcement learning workshop, NIPS*, 2015.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *JMLR*, 2005.

Faußer, S. and Schwenker, F. Neural network ensembles in reinforcement learning. *Neural Processing Letters*, 2015.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *ICML*, 2018.

Fujimoto, S., Conti, E., Ghavamzadeh, M., and Pineau, J. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019a.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. *ICML*, 2019b.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *JMLR*, 2015.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

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

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

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

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

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

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. *NeurIPS*, 2019.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. *Reinforcement learning*, 2012.

Laroche, R., Trichelair, P., and Combes, R. T. d. Safe policy improvement with baseline bootstrapping. *ICML*, 2019.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

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:1509.02971*, 2015.

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

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

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*, 2015.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. *NeurIPS*, 2016.

Pan, Y., Zaheer, M., White, A., Patterson, A., and White, M. Organizing experience: a deeper look at replay mechanisms for sample-based planning in continuous state domains. *IJCAI*, 2018.

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

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

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *ICLR*, 2016.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*, 2017.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., and Murphy, S. A. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 2011.

Siegel, N., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. *ICLR*, 2020.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Strehl, A. L., Langford, J., Li, L., and Kakade, S. Learning from logged implicit exploration data. *NeurIPS*, 2010.

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

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

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

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

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. *IROS*, 2012.

Van Hasselt, H., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *NeurIPS*, 2019.

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

Weng, L. Policy gradient algorithms. *lilianweng.github.io/lil-log*, 2018. URL https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., and Darrell, T. Bdd100k: A diverse driving video database with scalable annotation tooling. *CVPR*, 2018.

Zhang, S. and Sutton, R. S. 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* $\mathrm{P}_\Delta$ *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

$$\mathcal{L}(\alpha,\theta) = \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\ell_\lambda\left(\Delta_\theta^\alpha(s,a,r,s')\right)\right], \quad (8)$$
$$\Delta_\theta^\alpha = \sum_k \alpha_k Q_\theta^k(s,a) - r - \gamma \max_{a'} \sum_k \alpha_k Q_{\theta'}^k(s',a')$$

If the heads $Q_\theta^i$ and $Q_\theta^j$ 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 $\alpha_i Q_\theta^i + \alpha_j Q_\theta^j$ 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* $\mathrm{P}_\Delta$ *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* $Q_\theta^k$ *for* $k = 1,\ldots,K$ *minimize* $\mathcal{L}(\alpha,\theta)$ *for any* $\alpha \in \Delta^{\mathrm{K}-1}$.

**Proof**. Let $Q_{\alpha^*,\theta^*} = \sum_{k=1}^K \alpha_k^* Q_{\theta^*}^k(s,a)$ 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,

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

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

Let $\theta^I$ be such that $Q_{\theta^I}^k = Q_{\alpha^*,\theta^*}$ for all $Q$-heads. By definition of $Q_{\alpha^*,\theta^*}$, 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, $\theta^I$ 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 $\theta^M$ 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 $Q_\theta^k$ 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. (2019b): 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. (2019b), we train DDPG across 15 seeds, and select the 5 top performing seeds for dataset collection.

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. (2019b), 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; 2019b) except offline TD3 which uses a learning rate of 0.0005 for both the actor and critic.

Figure 8 shows 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. (2019b) 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:

$$\text{Score}_{\text{normalized}} = \frac{\text{Score}_{\text{Agent}} - \text{Score}_{\text{min}}}{\text{Score}_{\text{max}} - \text{Score}_{\text{min}}}, \quad (10)$$
$$\text{Score}_{\text{min}} = \min(\text{Score}_{\text{DQN}}, \text{Score}_{\text{Random}})$$
$$\text{Score}_{\text{max}} = \max(\text{Score}_{\text{DQN}}, \text{Score}_{\text{Random}})$$

Here, $\text{Score}_{\text{DQN}}$, $\text{Score}_{\text{Random}}$ and $\text{Score}_{\text{Agent}}$ 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 $\epsilon$-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.

**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 & 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 2) use the Adam optimizer (Kingma & Ba, 2015) with same hyperparameters as online QR-DQN (Dabney et al., 2018) ($\text{lr} = 0.00005$, $\epsilon_{\text{Adam}} = 0.01/32$). Note that scaling the loss has the same effect as inversely scaling $\epsilon_{\text{Adam}}$ when using Adam.

**Online Agents**. For online REM shown in Figure 1b, we performed hyper-parameter tuning over $\epsilon_{\text{Adam}}$ 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 ($\text{lr} = 0.00005$, $\epsilon_{\text{Adam}} = 0.001/32$).

**Wall-clock time for offline experiments.** The offline experiments are approximately 3X faster than the online experiments for the same number of gradient steps on a P100 GPU. In Figure 1, the offline agents are trained for 5X gradient steps, thus, the experiments are 1.67X slower than running online DQN for 200 million frames (standard protocol). Furthermore, since the offline experiments do not require any data generation, using tricks from supervised learning such as using much larger batch sizes than 32 with TPUs / multiple GPUs would lead to a significant speed up.

**Additional Plots & Tables**

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

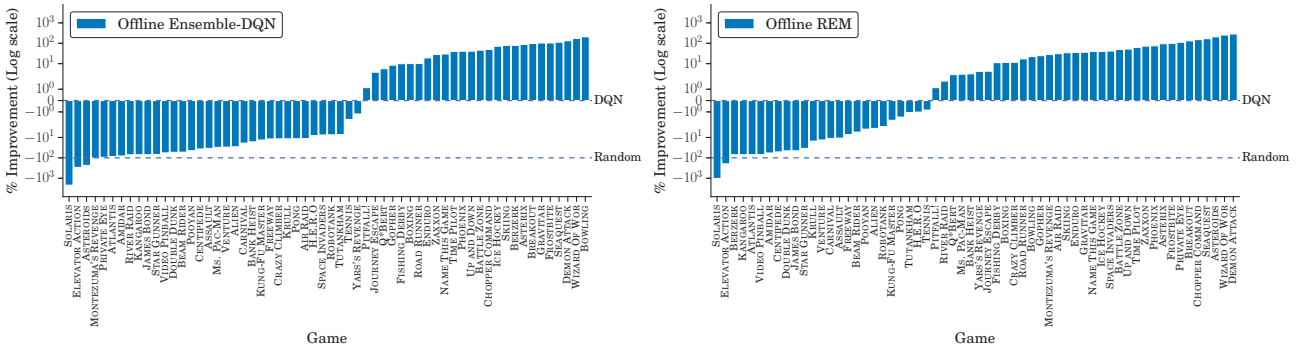
| Hyperparameter | setting (for both variations) |
|---|---|
| Sticky actions | Yes |
| Sticky action probability | 0.25 |
| Grey-scaling | True |
| Observation down-sampling | (84, 84) |
| Frames stacked | 4 |
| Frame skip (Action repetitions) | 4 |
| Reward clipping | [-1, 1] |
| Terminal condition | 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 $\epsilon$ | 0.001 |
| Evaluation steps per iteration | 125K |
| $Q$-network: channels | 32, 64, 64 |
| $Q$-network: filter size | $8 \times 8$, $4 \times 4$, $3 \times 3$ |
| $Q$-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 $\epsilon$ (for $\epsilon$-greedy exploration) | 0.01 | - |
| $\epsilon$-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 |

Table 3: Median normalized scores (Section A.3) across stochastic version of 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.

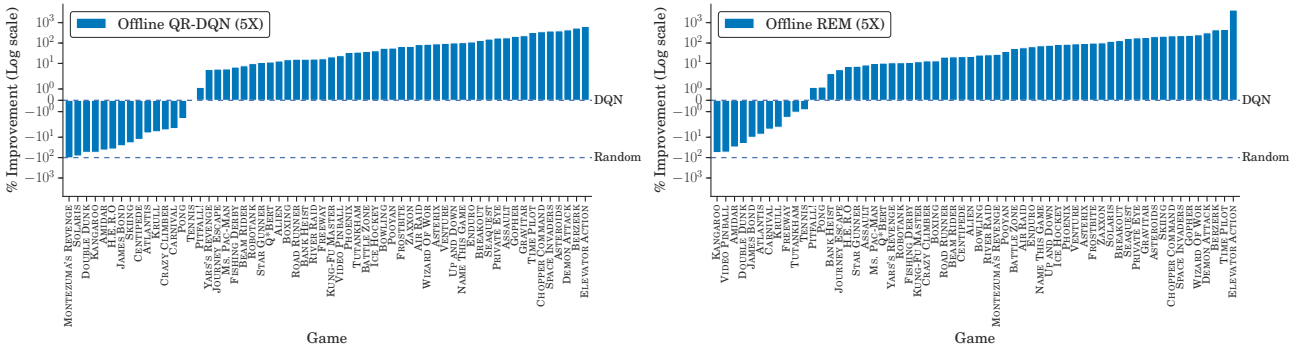
| Offline agent | Median | $>$ DQN |
|---|---|---|
| DQN (Nature) $(1x)$ | 74.4% | 10 |
| DQN (Adam) $(1x)$ | 104.6% | 39 |
| Ensemble-DQN $(1x)$ | 92.5% | 26 |
| Averaged Ensemble-DQN $(1x)$ | 88.6% | 24 |
| QR-DQN $(1x)$ | **115.0**% | **44** |
| REM $(1x)$ | 103.7% | 35 |

| Offline agent | Median | $>$ DQN |
|---|---|---|
| DQN (Nature) | 83.4% | 17 |
| 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** |



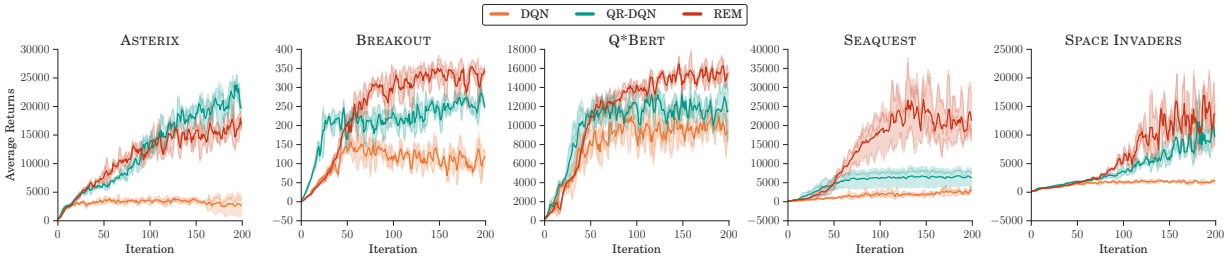

(a) (b)

Figure A.1: 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.

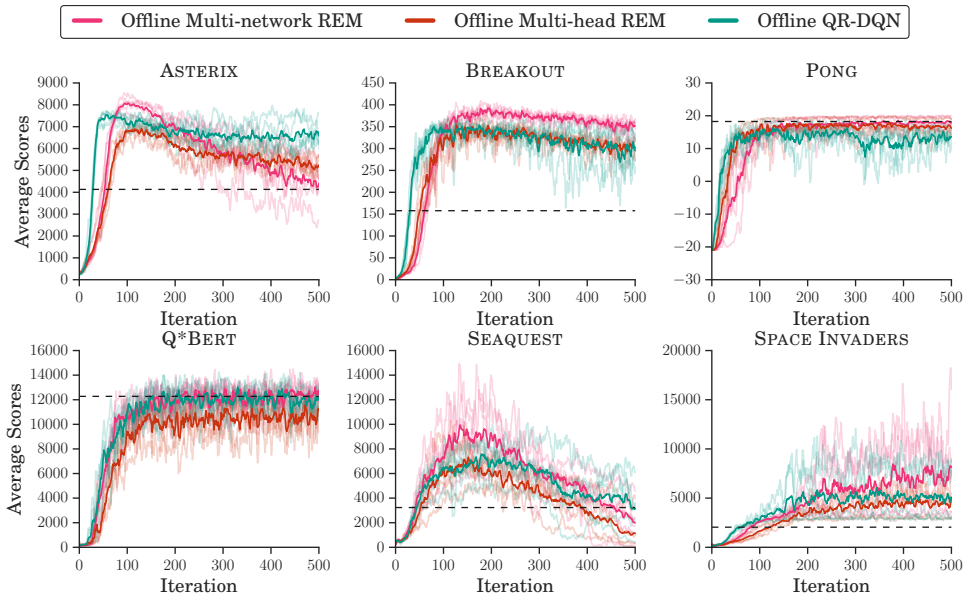



(a) (b)

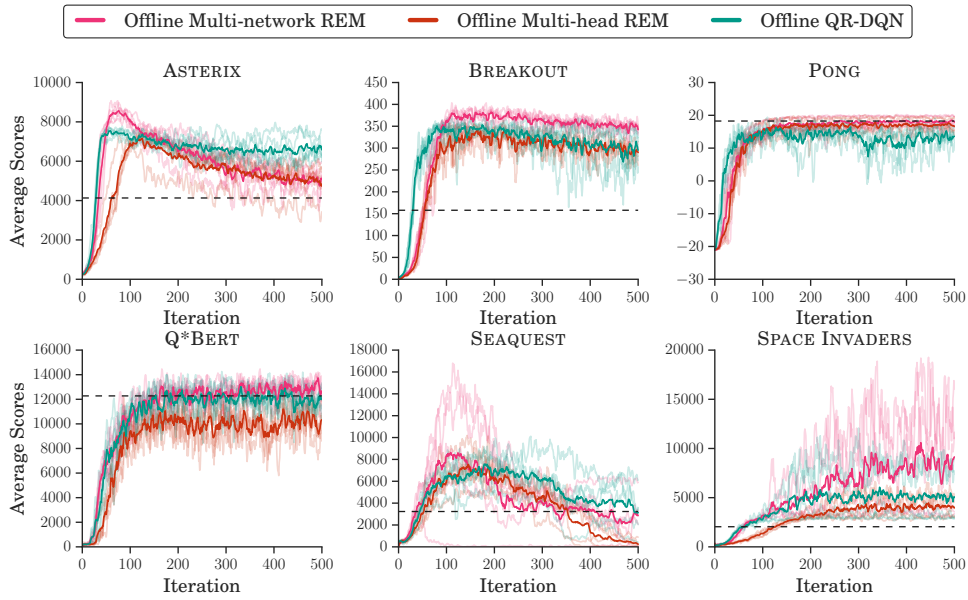
Figure A.2: Normalized Performance improvement (in %) over online DQN (Nature), per game, of (a) offline QR-DQN (5X) (b) offline REM (5X) trained using the DQN replay dataset for five times as many gradient steps as online DQN. The normalized online score for each game is 0.0 and 1.0 for the worse and better performing agent among fully trained online DQN and random agents respectively.




**Online REM *vs.* baselines**. Scores for online agents trained for 200 million ALE frames. Scores are averaged over 3 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.3: **REM with Separate $Q$-networks.** Average online scores of offline REM variants with different architectures and QR-DQN trained on stochastic version of 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.

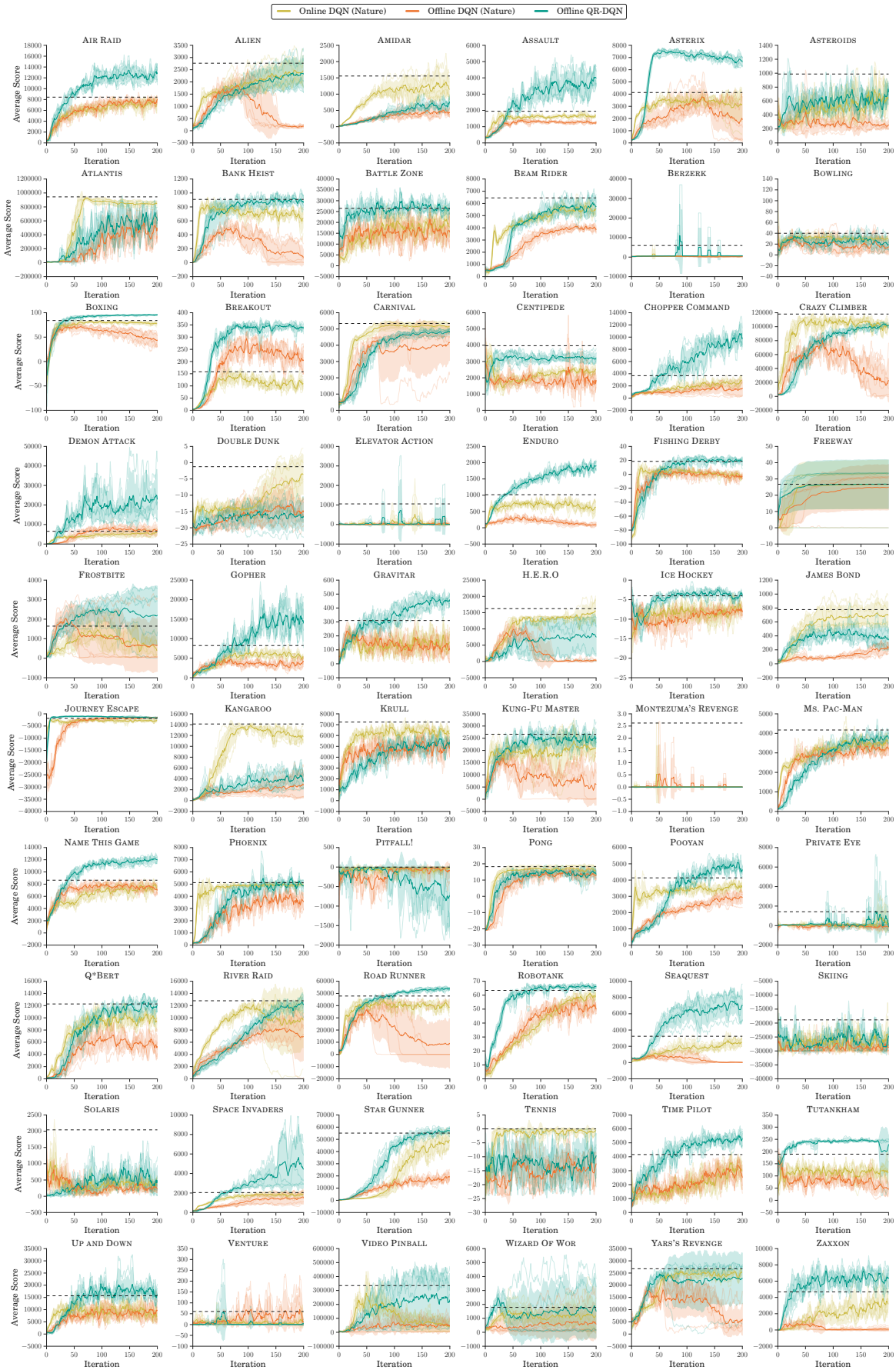



Figure A.4: Average evaluation scores across stochastic version of 60 Atari 2600 games for online DQN, offline DQN and offline QR-DQN trained for 200 iterations. The offline agents are trained using the DQN replay dataset. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding 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.

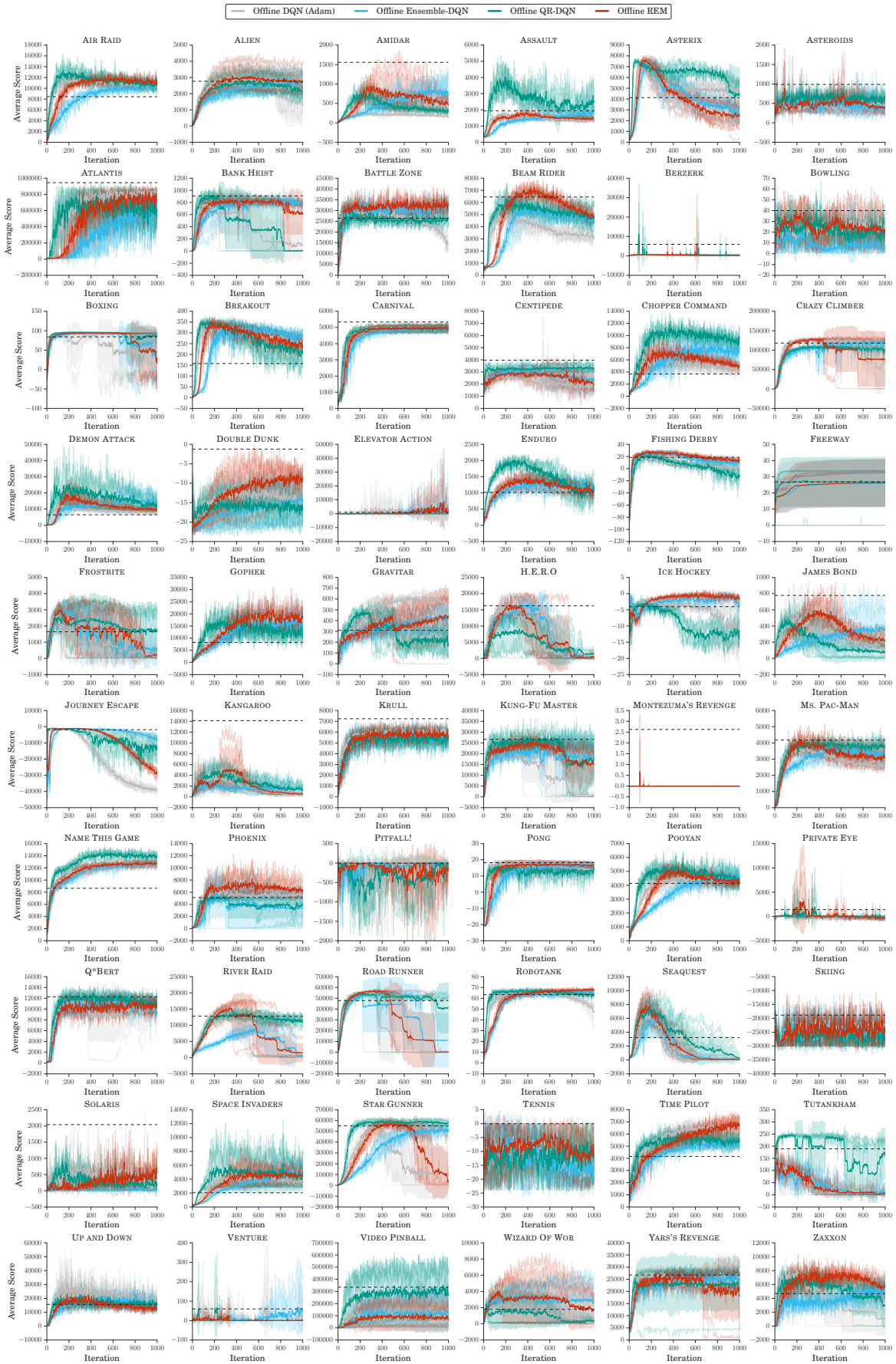



Figure A.5: Average evaluation scores across stochastic version of 60 Atari 2600 games of DQN (Adam), Ensemble-DQN, QR-DQN and REM agents trained offline using the DQN replay dataset. The horizontal line for online DQN show the best evaluation performance it obtains during training. All the offline agents 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.

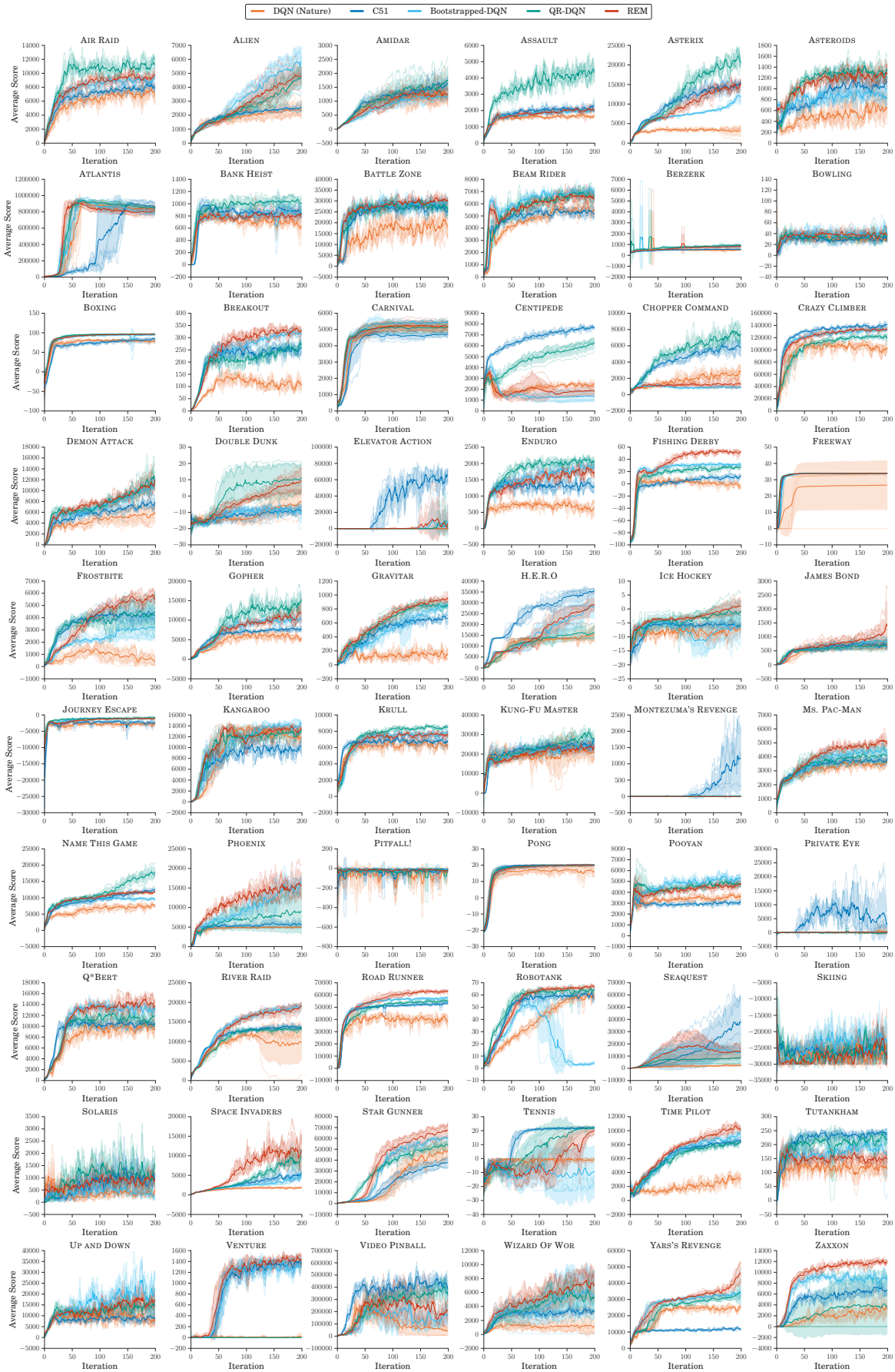



Figure A.6: **Online results.** Average evaluation scores across stochastic version of 60 Atari 2600 games of DQN, C51, QR-DQN, Bootstrapped-DQN and REM agents trained online for 200 million game frames (standard protocol). The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding window of 5 iterations and error bands show standard deviation.

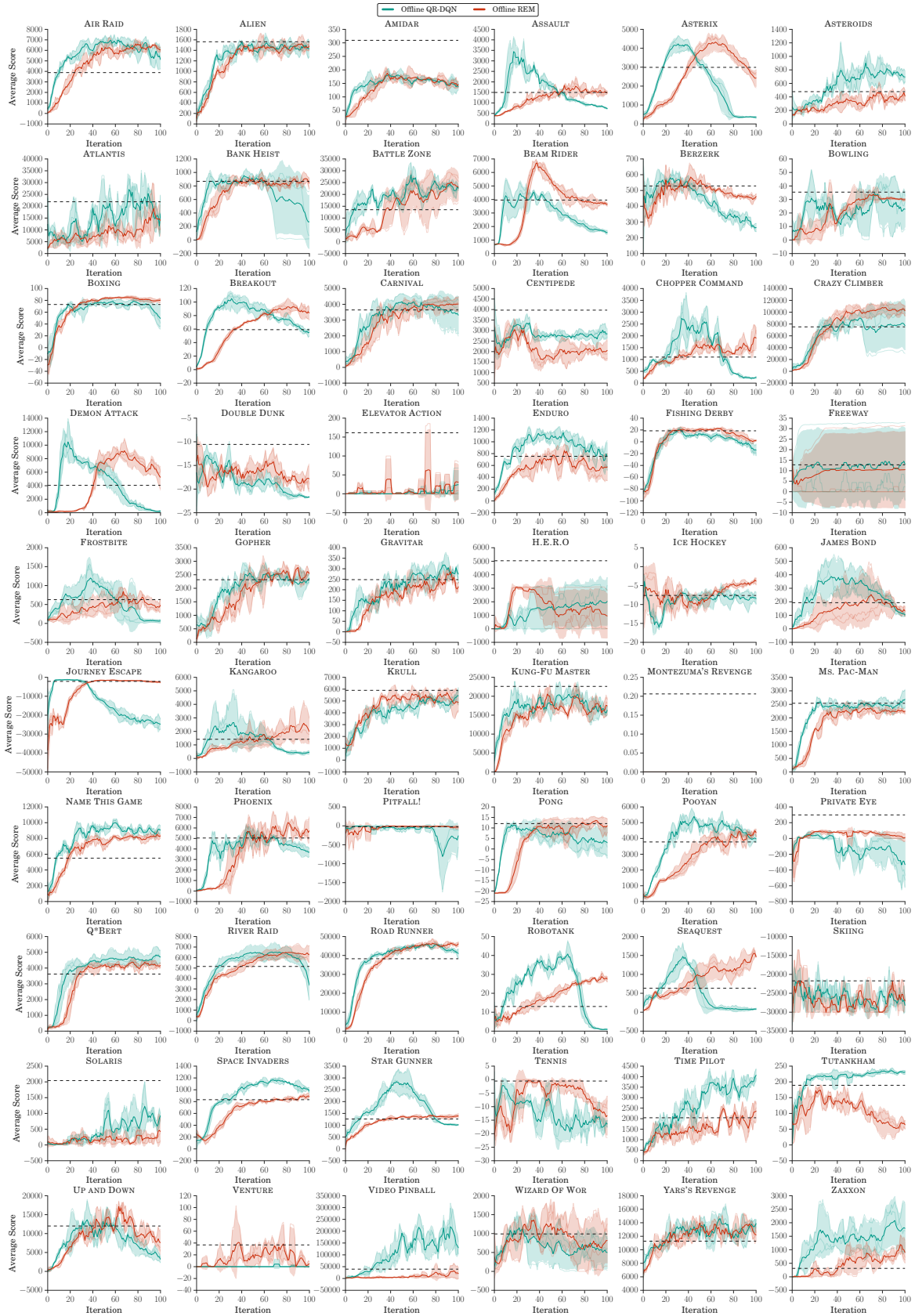



Figure A.7: **Effect of Dataset Quality.** Normalized scores (averaged over 3 runs) of QR-DQN and multi-head REM trained offline on stochastic version of 60 Atari 2600 games for 5X gradient steps using logged data from online DQN trained only for 20M frames (20 iterations). The horizontal line shows the performance of best policy found during DQN training for 20M frames which is significantly worse than fully-trained DQN.