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

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

If the heads Q_θ^i and Q_θ^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* 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_θ^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_{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 $(w_{\theta^*}^1,\cdots,w_{\theta^*}^K)$, *i.e.*, $Q(s,a) = \left(\sum_{k=1}^K \alpha_k^* w_{\theta^*}^k\right) \cdot f_{\theta^*}(s,a)$.

Let θ^I be such that $Q_{\theta^I}^k = Q_{\alpha^*,\theta^*}$ 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_θ^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 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.

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 ϵ_{Adam} when using Adam.

Online Agents. For online REM shown in Figure 1b, 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 ($\text{lr} = 0.00005$, $\epsilon_{\text{Adam}} = 0.001/32$).

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.

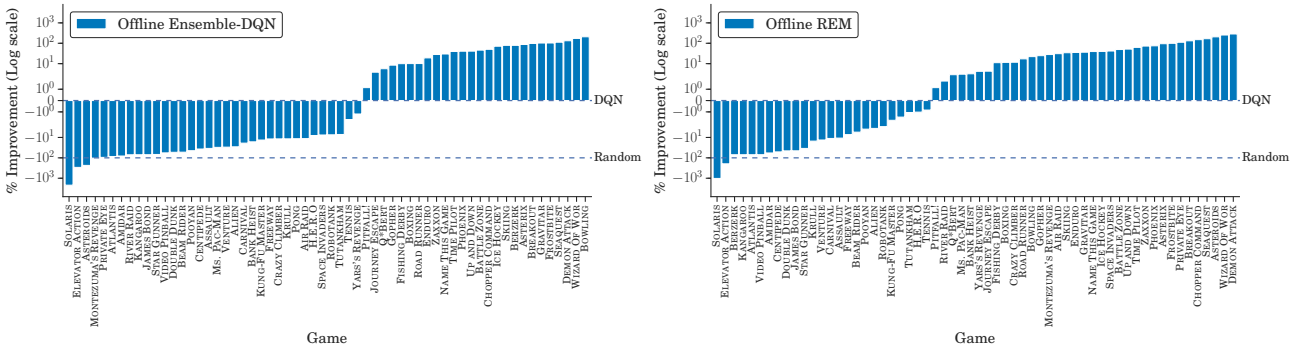
Hyperparameter	setting (for both variations)
Sticky actions	Yes
Sticky action probability	0.25
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Frame skip (Action repetitions)	4
Reward clipping	[-1, 1]
Terminal condition	Game Over
Max frames per episode	108K
Discount factor	0.99
Mini-batch size	32
Target network update period	every 2000 updates
Training steps per iteration	250K
Update period every	4 steps
Evaluation 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

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.

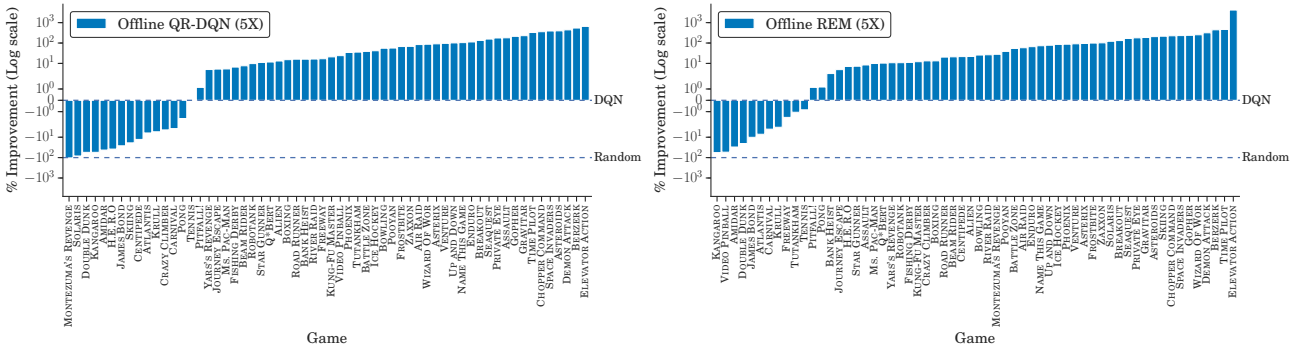
Offline agent	Median	$>$ DQN
DQN (Nature) ($1x$)	74.4%	10
DQN (Adam) ($1x$)	104.6%	39
Ensemble-DQN ($1x$)	92.5%	26
Averaged Ensemble-DQN ($1x$)	88.6%	24
QR-DQN ($1x$)	**115.0**%	**44**
REM ($1x$)	103.7%	35

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**



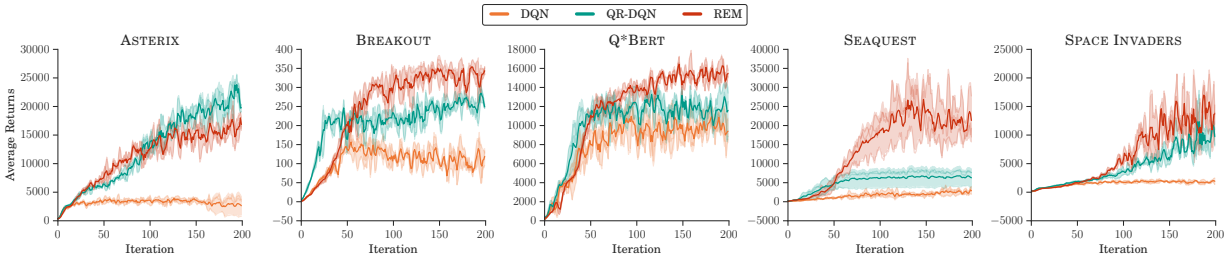
(a) (b)

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.

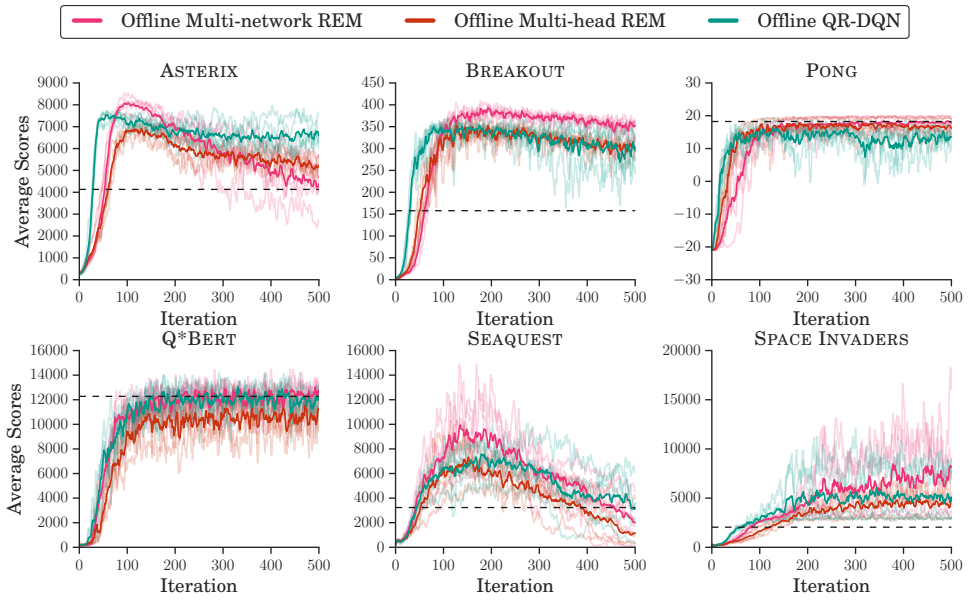


(a) (b)

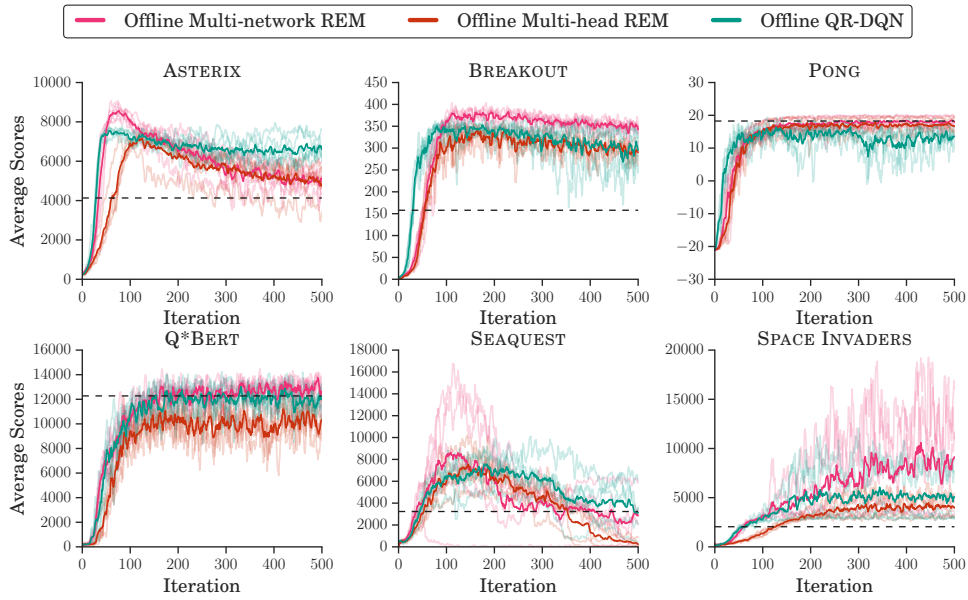
Figure A.2: Normalized Performance improvement (in %) over online DQN (Nature), per game, of (a) offline QR-DQN (5X) (b) offline REM (5X) trained using the DQN replay dataset for five times as many gradient steps as online DQN. The normalized online score for each game is 0.0 and 1.0 for the worse and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

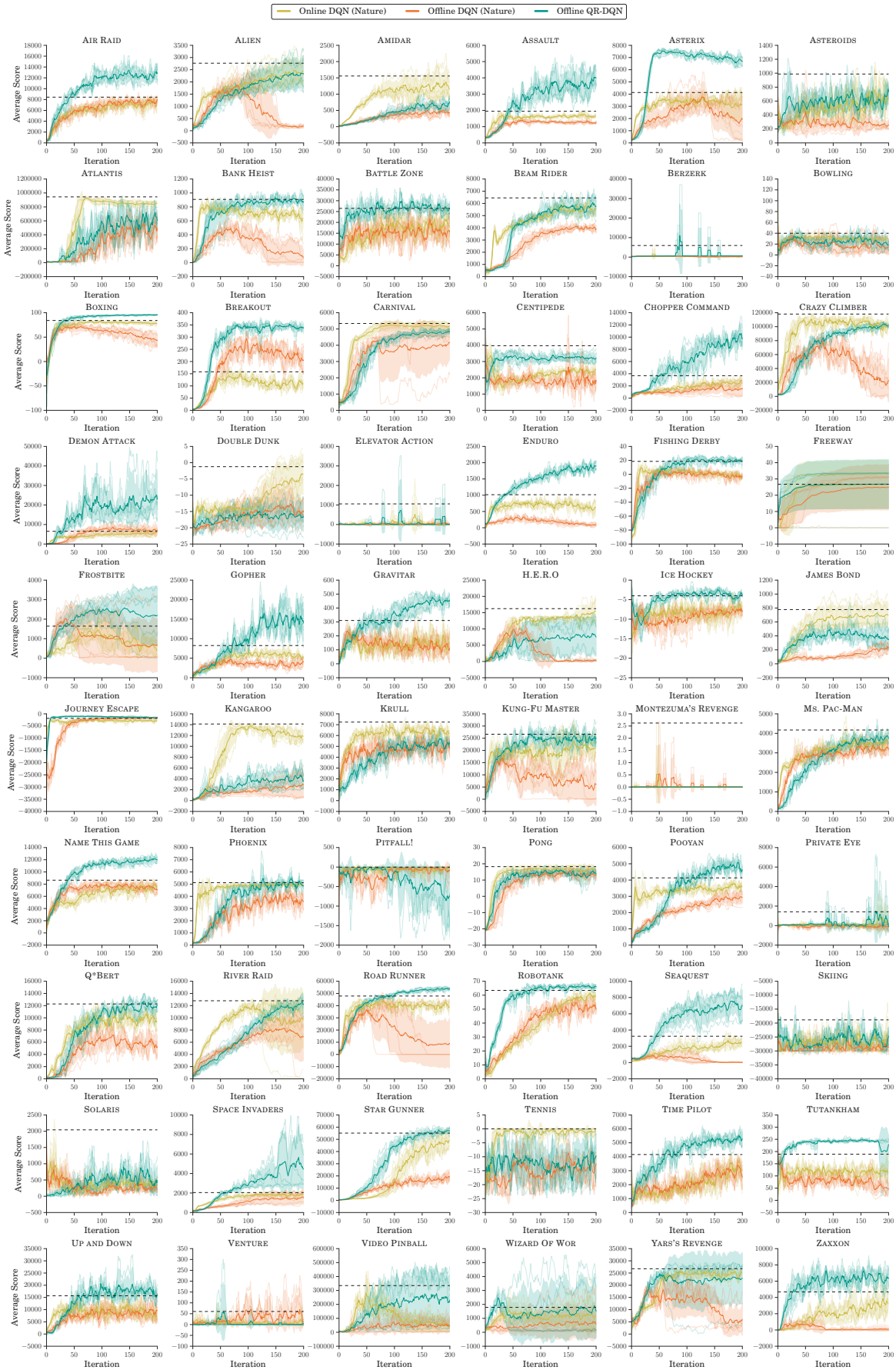


Figure A.4: Average evaluation scores across stochastic version of 60 Atari 2600 games for online DQN, offline DQN and offline QR-DQN trained for 200 iterations. The offline agents are trained using the DQN replay dataset. The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding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

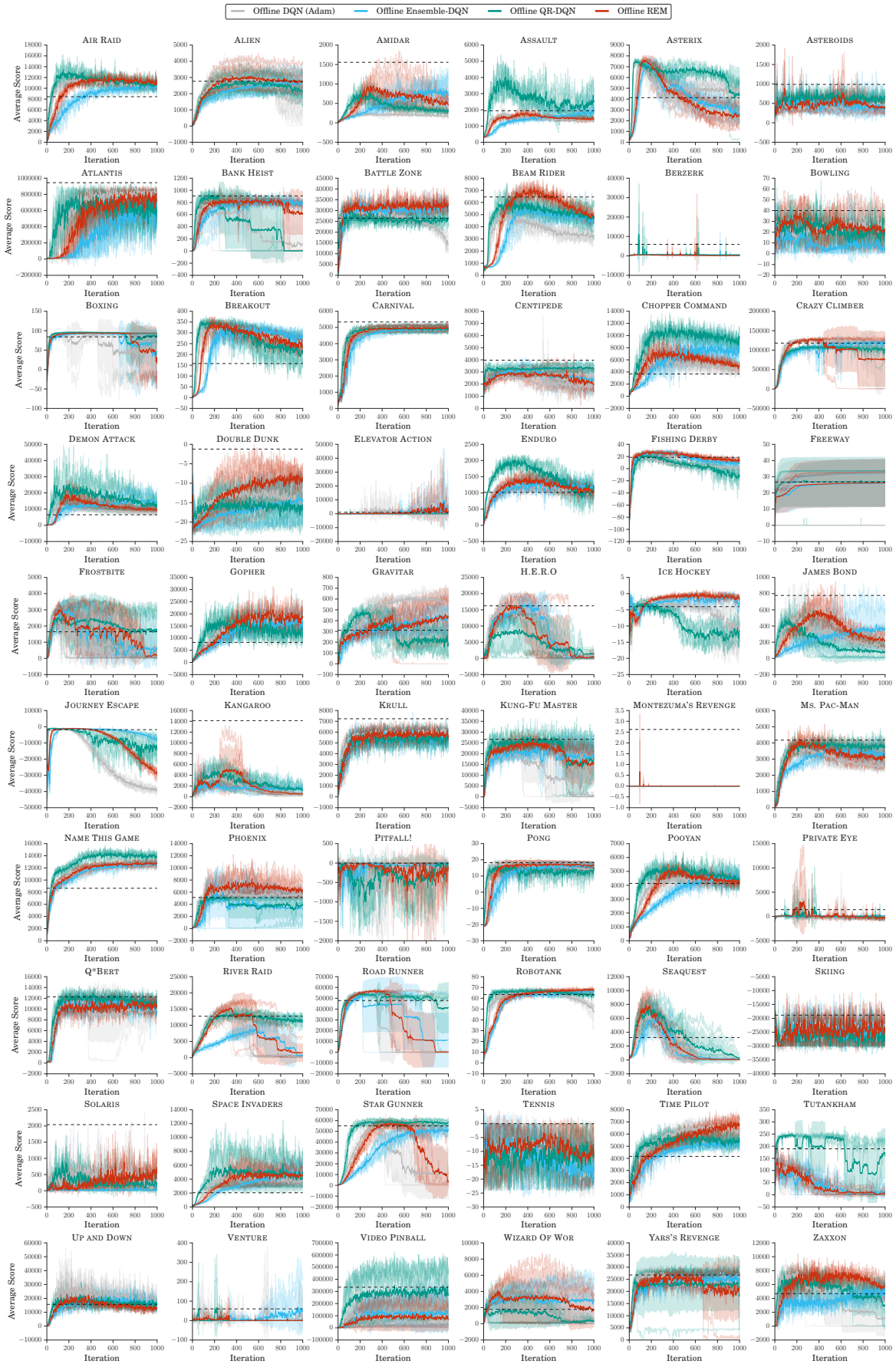


Figure A.5: Average evaluation scores across stochastic version of 60 Atari 2600 games of DQN (Adam), Ensemble-DQN, QR-DQN and REM agents trained offline using the DQN replay dataset. The horizontal line for online DQN show the best evaluation performance it obtains during training. All the offline agents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

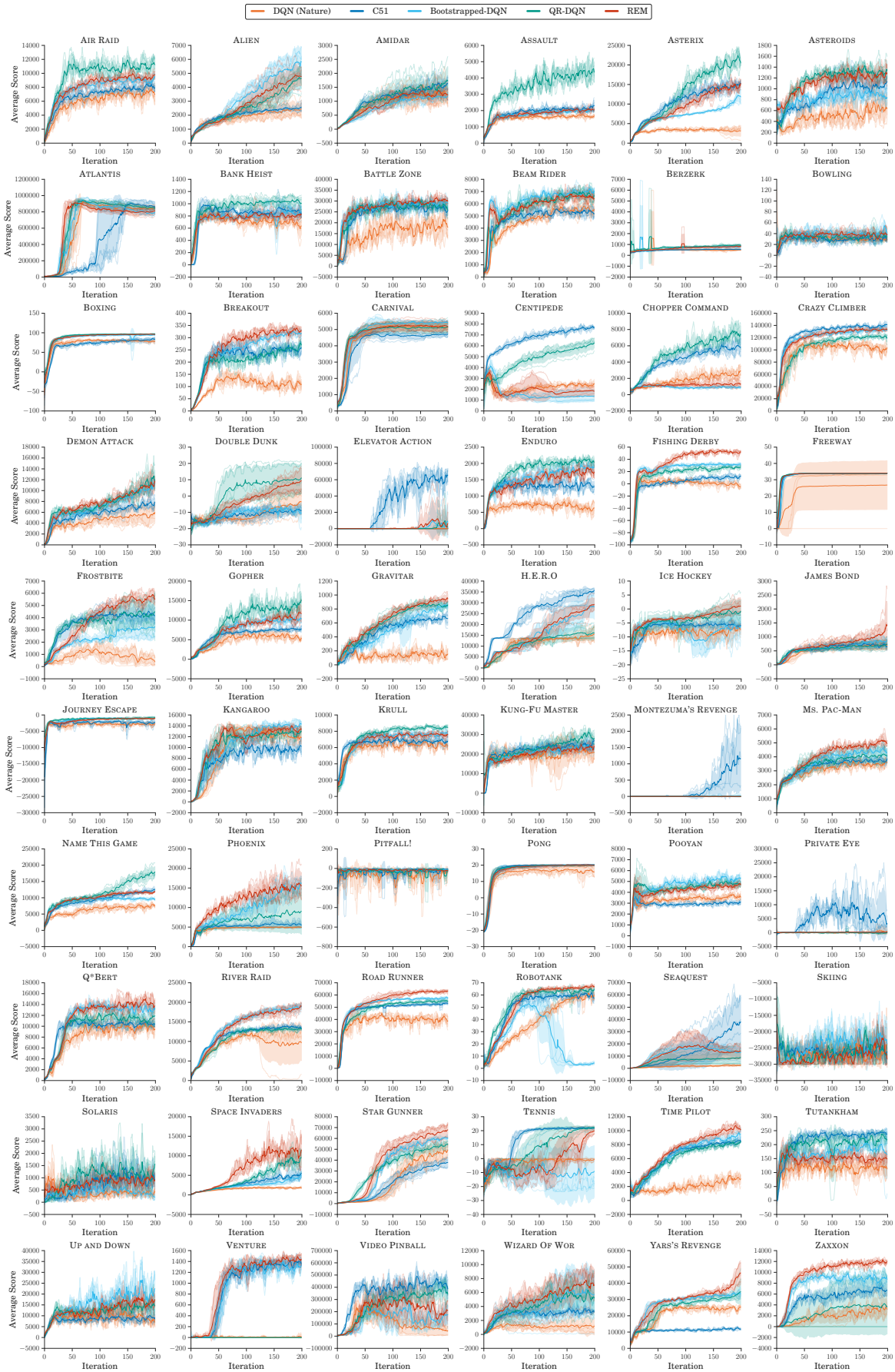


Figure A.6: **Online results.** Average evaluation scores across stochastic version of 60 Atari 2600 games of DQN, C51, QR-DQN, Bootstrapped-DQN and REM agents trained online for 200 million game frames (standard protocol). The scores are averaged over 5 runs (shown as traces) and smoothed over a sliding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

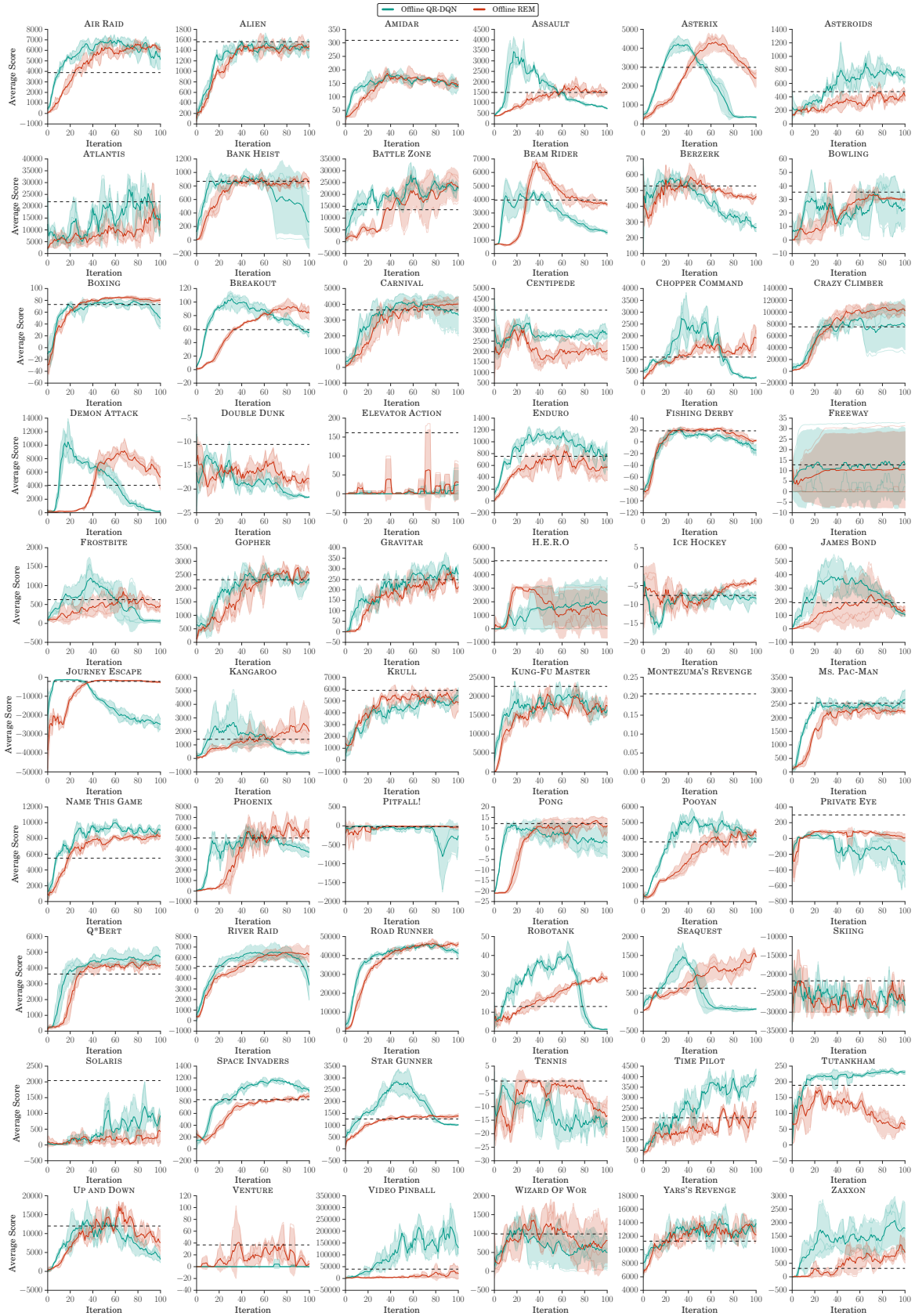


Figure A.7: **Effect of Dataset Quality.** Normalized scores (averaged over 3 runs) of QR-DQN and multi-head REM trained offline on stochastic version of 60 Atari 2600 games for 5X gradient steps using logged data from online DQN trained only for 20M frames (20 iterations). The horizontal line shows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