Interpretable performance analysis towards offline reinforcement learning: A dataset perspective

Chenyang Xi [1 2] **Bo Tang** [1] **Jiajun Shen** [3 1] **Xinfu Liu** [2] **Feiyu Xiong** [1] **Xueying Li** [1]

Abstract

Offline reinforcement learning (RL) has increasingly become the focus of the artificial intelligent research due to its wide real-world applications where the collection of data may be difficult, time-consuming, or costly. In this paper, we first propose a two-fold taxonomy for existing offline RL algorithms from the perspective of exploration and exploitation tendency. Secondly, we derive the explicit expression of the upper bound of extrapolation error and explore the correlation between the performance of different types of algorithms and the distribution of actions under states. Specifically, we relax the strict assumption on the sufficiently large amount of state-action tuples. Accordingly, we provably explain why batch constrained Q-learning (BCQ) performs better than other existing techniques. Thirdly, after identifying the weakness of BCQ on dataset of low mean episode returns, we propose a modified variant based on top return selection mechanism, which is proved to be able to gain the state-of-the-art performance on various datasets. Lastly, we create a benchmark platform on the Atari domain, entitled RL easy go (RLEG), at an estimated cost of more than 0.3 million dollars. We make it open-source for fair and comprehensive competitions between offline RL algorithms with complete datasets and checkpoints being provided.

1. Introduction

1.1. Background and motivations

Reinforcement learning (RL) tries to figure out how intelligent agent ought to take actions under the interaction with environment such that the accumulative reward could be maximized, and becomes increasingly popular due to its wide real-world applications where data collection may be difficult, time-consuming, and costly.

[1]Alibaba Group, Hangzhou, China [2]School of Aerospace Engineering, Beijing Institute of Technology, Beijing, China [3]School of Electrical and Computer Engineering, Purdue University, Indiana, USA. Correspondence to: Xinfu Liu <lau.xinfu@gmail.com>.

In most studies concerned with reinforcement learning (RL) algorithms (Sutton & Barto, 2018), authors assume that an agent interacts with an online environment or simulator and learns from the ”dynamic” date set generated by updated policy of its own. However, when facing complex real world problems, it is a totally different case due to the extremely large data (including states and actions), which limits the applicability of online methodologies. As a consequence, offline RL (also known as batch RL in some researches) algorithms are well and rapidly developed especially in many practical scenarios where the explorations (actions of trial and errors) are extremely costly, e.g., robotics, E-commercials, manufactures. Especially, in E-commercial case, RL has been widely applied in different and challenging business scenarios, e.g., coupons delivery (Xiao et al., 2019), search engine (Hu et al., 2018), recommendations (Zhao et al., 2018), impression allocation (Cai et al., 2018), etc. Nonetheless, each update and iteration of algorithm already deployed online would introduce uncertainties to production system, which possibly contributes to an asset loss accident. In addition, performance improvement cannot be guaranteed since the training process of online algorithms (e.g. Nature DQN) is time-consuming.

However, we notice that there are no conclusive investigations and reviews for the effectiveness and applicability of existing offline RL algorithms, which no doubtedly would make readers confused when choosing algorithms given an offline dataset. For example, both batch-constrained Q-learning (BCQ) and random ensemble mixture (REM) are claimed to preform better than each other. However, the offline dataset of their experiments are basically different. In BCQ experiments, the offline dataset is generated by a partially trained DDPG (i.e. a medium oracle), while that of REM is generated by Nature DQN (i.e. a combination of starter, medium and complete oracle). Consequently, it is of great necessity to figure out the underlying principles such that fair comparisons could be made.

To address the above problems, we first propose a taxonomy, which divides the existed offline RL algorithms into two categories, exploitation-tentative algorithms (e.g. BCQ, (Fujimoto et al., 2019b), (Fujimoto et al., 2019a), best action imitation learning (BAIL) (Chen et al., 2020), bootstrapping error accumulation reduction (BEAR) (Kumar et al., 2019), safe Policy improvement with baseline bootstrapping (SPIBB) (Laroche et al., 2019)), and exploration-tentative algorithms (e.g. ensemble DQN, (Faußer & Schwenker, 2015), (Anschel et al., 2017), Bootstrapped-DQN, (Osband et al., 2016), C51, (Bellemare et al., 2017), Quantile Regression(QR)-DQN (Dabney et al., 2017), REM (Agarwal et al., 2020)).

1.2. Literature Review

1.2.1. Exploitation-tentative algorithms

In (Fujimoto et al., 2019b), authors claim that most off-policy algorithms would fail in offline setting due to extrapolation errors caused by erroneously estimating the unseen state-action pairs, and therefore proposed BCQ. In BCQ, when selecting actions that maximize Q value, they further eliminate actions which are unlikely to be selected by behavioral policy using a generative model. Experiments are made on offline dataset generated by deep deterministic policy gradient (DDPG).

The counterpart of (Fujimoto et al., 2019b) on discrete action space is (Fujimoto et al., 2019a) where discrete BCQ is claimed to be the optimal offline RL algorithm. However, its performance can only achieve to be equivalent to or a little bit higher than the one of noiseless policy, which is obtained based on the data set generated by partially trained DQN, i.e., a quite lower level of performance. Under the circumstance of limited data in offline setting, BCQ acts more likely to robust imitation learning algorithm ((Wang et al., 2017)). In contrast to the investigation in (Agarwal et al., 2020) where the scale of offline dataset is assumed to be large enough, the authors conclude that common off-policy Deep RL algorithms are not well suitable for offline learning tasks.

Similar to BCQ of continuous version, (Kumar et al., 2019) also imposes the constraints on the distribution of continuous action space in off-policy Q-learning cases. Authors identify bootstrapping error as key source of instability in existing off-policy RL algorithms, the performance of which could not be elevated merely through scaling up the the batch.

Different from imposing strict constraints on distributional similarities in BCQ, the basic mindset of BEAR is to make trade-off between concentrability coefficient (i.e., the parameter quantifying the degree to which current states and actions are out of distribution generated by behavioral policy) and suboptimality constant (i.e., the parameter quantifying the distance between the current policy and the optimal one). Compared with BCQ, the visitation distribution generated by current policies would not be too much similar to the batch data distribution in BEAR. Thus, BEAR can be treated as a robust variant of BCQ. However, it is not readily extended to the cases of discrete action space.

Instead focusing on generating similar state-action visitations, (Chen et al., 2020) tries to crack the problem from a imitation-learning perspective by selecting ”valuable” state-action pairs and episodes that contain enough information for learning a relatively optimal strategy with regards to higher returns. Supervised learning methodology has been applied for the derivation of an upper envelope where high return data lie nearby. Accordingly, the optimal strategy is obtained directly through imitation.

In (Laroche et al., 2019), authors judge the value of state-action pairs based on the number of occurrences. For a specified state-action pair (s, a), it will be accepted for further imitation learning process (in a greedy way) only if the number of occurrences $N(s, a)$ is more than a fixed threshold $N_{\wedge}(s, a)$ (calculated based on (Ghavamzadeh et al., 2016) and (Weissman et al., 2003)), otherwise behavioral policy would serves as a baseline.

Our discussion and review on exploitation-tentative offline RL algorithms can be concluded in Table. 1.

1.2.2. Exploration-tentative algorithm

C51 (Bellemare et al., 2017), as a comb form methodology, extends the Q-value to Q-distribution where value function is defined as the expectation of value distribution with multiple peaks. Ensemble-DQN (Anschel et al., 2017) is a simple extension of DQN that approximates the Q-values via an ensemble of parameterized Q-functions, i.e., multiple heads. It should be noted that each head independently estimates the Q-value with huber loss. The final loss is derived by simply taking average of all heads.

Bootstrapped-DQN (Osband et al., 2016) uses one of the Q-value estimates in each episode to improve the depth of exploration. The authors claim that bootstrapped neural networks are able to produce reasonable posterior estimates for regression. The basic mindset of REM (Agarwal et al., 2020) borrows from dropout mechanism. For five different outputs (Q-networks generated by a shared neural network), authors randomly assign the weights with sampling performance of the algorithm being lifted. In order to underscore the importance of randomness, the authors also make comparisons between their random mechanism and the average one, which is proved to be less optimal.

Among most of off-policy algorithms, QR-DQN is provably to be the best rather still underperform the policy with noise (Dabney et al., 2017). Again, it should be noted that although QR-DQN is not exclusively developed for offline setting, it is still able to achieve high performance given

Table 1. Comparison of exploitation-tentative offline RL algorithms

	C-BCQ	BAIL	D-BCQ	BEAR	SPIBB
Scenario	Continuous action space	Continuous action space	Discrete action space	Continuous action space	Countable (s, a) pairs
Constraints on (s,a) quantity	N/A	YES	YES	N/A	YES
Main idea	Maximizing the similarities between behavioral and trained policy	Imitate (s,a) pairs with high return	Extension of C-BCQ to discrete action space	Relax the constraints on distributional similarities	Safely improved based on behavioral policy
Pros	SOTA among exploitation-tentative algorithms	Same as C-BCQ	Better performance on random/less Oracle dataset	Readily implement	Safely bounded performance
Cons	Rely on Oracle dataset	Same as C-BCQ	Suboptimal performance		Hard to count N(s,a)

sufficiently large and complete data set.

Our discussion and review on exploration-tentative offline RL algorithms can be concluded in Table. 2.

Noticed that in all imitation-based offline RL algorithms, a strong assumption of the amount of (s, a) pairs, i.e., $N(s, a) > N_{\wedge}(s, a)$, $\forall(s, a) \in \mathcal{S} \times \mathcal{A}$, is a necessity. In fact the assumption is unreal in certain practical scenarios especially video games due to the high cost as we have discussed before.

1.3. Main contribution

From the above literature reviews, we notice that existing offline RL techniques have not been well concluded, and the applicability of them under various datasets has not been clearly stated either. Thus, readers might be confused about how to select the most appropriate algorithm when facing a brand new dataset generated by different behavioral polices or even unknown ones. In practical scenarios, trial and error is often costly and time consuming. Besides, the argue between self-claimed SOTA algorithms has not been well resolved due to the totally different behavior policies. To address all the mentioned concerns, we list our four-fold contributions as:

(1) We propose a taxonomy for existing offline RL algorithms from the perspective of exploration and exploitation tendency.

(2) Based on the derivation of upper bounded extrapolation error, we provably investigate the applicability of both types of algorithms on different datasets (in terms of different action distributions under states) and explain why BCQ performs better than existing techniques.

(3) We identify the limitation of BCQ, as its weak performance on datasets with low mean episode return. To bridge the gap, we propose a modified version by introducing a return-based data selection mechanism, which reaches better performance on various datasets.

(4) A benchmark of Atari domain is open sourced and most existing offline RL algorithms are included. We spend more than 0.3 million dollars on the experiments and collect all intermediate results incuding various datasets and the intermediate model checkpoints. The benchmark could be used for fair and comprehensive competitions between existing and future offline RL algorithms.

2. Extrapolation error-based applicability analysis and comparisons

In this section, we would analyze the applicability of both exploration and exploitation-tentative algorithms on different datasets from a perspective of extrapolation error.

Due to the inability of interacting with the environment, both types of offline RL algorithms we conclude before have to be suffered from the failure of learning as well as online ones. More specifically, when testing the offline trained model in real world, the mismatch between offline dataset and practical state-action visitations of the current policy would give rise to the extrapolation error, which introduces the performance gap between offline and online RL algorithms.

Besides, intuitively, due to he vital role that behavioral policy plays in offline RL training, the discrepancies of various offline datasets would contribute to the different extrapolation errors and therefore distinct performance of offline RL algorithms. The corresponding argue with respect to the state-of-the-art performance comes up referring to the one between above-mentioned REM and BCQ. How to provably explain and make a fair comparison is undoubtedly

Table 2. Comparison of exploration-tentative off-policy RL algorithms

	Ensemble DQN	Bootstrapped DQN	C51	QR-DQN	REM
Main idea	Taking average of single heads	Randomly select optimal strategy of higher possibility	Extend Q-value to Q-distribution	Same as C51	Convex combination of single heads
Pros	Easy to compute	Efficient training time	Innovated idea	Theoretical proof based	SOTA Offline RL methodology under Atari environment
Cons	Training process is not stable	N/A	No theoretical support	N/A	No theoretical support

a challenging task, which would be resolved in this section.

2.1. Can extrapolation error be completely eliminated?

In existing exploitation-tentative algorithms, researchers try to eliminate the extrapolation error by placing strict constraints on the distribution of (state-action) tuples with the help of generative model (e.g. (Fujimoto et al., 2019b), (Fujimoto et al., 2019a), (Kumar et al., 2019)), simply discarding the tuples of insufficient amount of occurrences (e.g. (Chen et al., 2020)) or replacing these tuples with behavioral policy as a safe baseline (e.g. (Laroche et al., 2019)). However, as we conclude in Table 1, a strong assumption on the sufficient visitations of selected tuples is necessary.

For exploration-tentative algorithms, extrapolation error exists as well but claims to completely vanish by assuming a big enough offline dataset with adequate diversity (e.g. 50 million tuples per game from Nature DQN (Agarwal et al., 2020)).

However, we notice that the assumptions concerned with the size of tuples in the offline dataset are too strong to be satisfied in practical scenarios (especially for some costly application such as E-commercial, robotics, etc). In addition, for large continuous state or action space, it is impossible to accurately count the amount of occurrences for each tuple.

Thus, in the following context, we start with deriving an explicit upper bound on the extrapolation error, and investigate how it is affected by behavioral policy and corresponding offline dataset.

2.2. Preliminaries

We first introduce following useful lemmas and necessary assumptions for the ensuing proofs.

Lemma 2.1. *(Weissman et al., 2003) For* $\forall(s,a) \in \mathcal{S}\times\mathcal{A}$*,* $Q(s,a) \leq \frac{R_{max}}{1-\gamma}$

Lemma 2.2. *(Ghavamzadeh et al., 2016) For* $\forall(s,a) \in \mathcal{S}\times\mathcal{A}$*,* $||p_1(.|s,a) - p_2(.|s,a)||_1 \leq e(s,a)$ *where,* $e(s,a) = \sqrt{\frac{2}{N(s,a)} log \frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}}$

Assumption 2.1. *In offline dataset to be fed to training process, the number of all state-action pair occurrences satisfy* $N(s) = \sum_{a\in\mathcal{A}} N(s,a) = N, \forall s \in \mathcal{S}$*, i.e., although for each state* $s \in \mathcal{S}$*, the distribution of action* $a \in \mathcal{A}$ *may vary, the total amount of occurrence of each state would be the same.*

Compared with the strong assumption that $N(s,a) < N_{\wedge}(s,a), \forall(s,a) \in \mathcal{S}\times\mathcal{A}$ made in most previous research, the assumption 2.1 is much relaxed and therefore more realistic to be satisfied in practical scenarios, which also indicates that our work is not a trivial extension of previous work.

Assumption 2.2. *The agent is rational (i.e., aiming for a relatively better result or higher total rewards following the rules of environment) such that pure random policy (i.e., under each state, choosing each action with same probability) is always the worst one (i.e., the mean episode return is the lowest) among all candidate policies, while the policies of biased action distribution under each state would result in a better achievement. In addition, the more biased the distribution is, the higher reward is achieved.*

It is noted that we use *randomness* for capturing the degree of distributional bias in the following context, which would be further discussed in the ensuing sections. Assumption 2.2 reveals our insights about some practical scenarios (e.g., video games, E-commercial, robotics, etc) for applying RL-based methodologies, which reflects an underlying mindset that Oracle policy are mostly deterministic. We admit that there might be cases against this mind as shown in the following four-quadrant figure, through which we try to explain Assumption 2.2 from a perspective of the correlation between randomness and returns.

In Fig. 1, we take all possible cases into consideration. The 2nd and 4th quadrants conform the Assumption 2.2. Nonetheless, some cases might fall in 3rd quadrant due to some inaccurate description of fundamental elements, e.g., dynamics, rewards. Game players may misunderstand the goal and take completely opposite actions to optimal ones such that worst return is achieved. These special cases are not included in this paper and would be discussed separately.

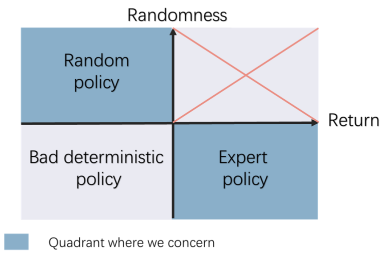


Figure 1. **Correlation between randomness and returns**

Assumption 2.3. *If a dataset is generated by a given policy π_b, then for $\forall s \in \mathcal{S}$, $N(s,a) = N(s)\pi_b(a|s)$*

In the following context, for notational convenience, we omit the arguments of $N(s)$, and simply denote it as N.

2.3. How does dataset affect the extrapolation error of exploration-tentative algorithms?

Proposition 1. *The extrapolation error $\epsilon_{s,a}$ is upper bounded by*

$$\begin{aligned}\bar{\epsilon}_{s,a} = (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}N^{-1/2} \\ [(\pi_b(a|s))^{-1/2} \\ +\gamma\sum_{s'}p_1(s'|s,a)\sum_{a'}\pi(a'|s')(\pi_b(a'|s'))^{-1/2} \\ +...+\gamma^{(n)}\sum_{s'}p_1(s'|s,a)\sum_{a'}\pi(a'|s')\sum_{s''}... \\ \sum_{s^{(n)}}p_1(s^{(n)}|s^{(n-1)},a^{(n-1)}) \\ \sum_{a^{(n)}}\pi(a^{(n)}|s^{(n)})(\pi_b(a^n|s^n))^{-1/2}+...]\end{aligned} \quad (1)$$

Inspired by the results of Proposition 1, we define the randomness of dataset as follow.

Definition 1. *The randomness of dataset is defined as*

$$q = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{A}}\frac{1}{\sqrt{\pi_b(a_i|s)}} \quad (2)$$

According to Definition 1, a pure random behavior policy having a highest value of q indicates the highest randomness of the corresponding datasets, i.e., uniform distribution of actions under each state.

Assumption 2.4. *The distribution of π is uniform, i.e., the possibilities of any type of trainer or player are same. Intuitively, it is also noted that the distribution of π is independent of the distribution of π_b.*

Now, we are ready to propose the following theorem.

Theorem 1. *Given the distribution of $\pi(a|s)$ follows assumption 2.4, the general term of extrapolation error, equation (3), reaches its minimum when $\pi_b(a|s)$ is pure random. Also, the more even the behavior policy $\pi_b(a|s)$ under each state is, the less value it would be.*

$$\sum_a \pi(a|s)\frac{R_{max}}{(1-\gamma)}N^{-1/2}(\pi_b(a|s))^{-1/2}, \forall s \in \mathcal{S} \quad (3)$$

Equivalently, the optimal behavioral policy is
$\pi^*(a|s) = \frac{1}{|\mathcal{A}|} = \underset{\pi_b\in\Pi}{argmin}\,\mathbb{E}_\pi[\sum_a \pi(a|s)(\pi_b(a|s))^{-\frac{1}{2}}]$, $\forall(a,s) \in \mathcal{A}\times\mathcal{S}$

According to Theorem 1, the performance of exploration-tentative algorithms would best when offline dataset is of the highest randomness. Besides, the returns would decrease along with the randomness descent of dataset.

2.4. How does dataset affect the extrapolation error of exploitation-tentative algorithms?

In this section, we continue to investigate influence of dataset on exploitation-tentative algorithms. As we have discussed in Section 1, most exploitation-tentative algorithms fall into two underlying mindsets represented by BCQ and BAIL respectively, i.e., generating similar state-action distribution, and selecting targeted state-action tuples. Thus, the following investigation is two-fold.

2.4.1. BCQ-LIKE ALGORITHMS

For BCQ-like algorithms, actions chosen for offline optimization must satisfy the constraint $G(a|s) > \tau$, where G is the generative model for selecting batch-constrained actions and quantitatively depends on the number of occurrences of tuple (s,a) in dataset. Given Assumption 2.1 being satisfied, we have $N(s,a) > N\tau$, where (s,a) are batch-constrained tuples.

Thus, we derive the upper bounded extrapolation error for BCQ-like algorithms, based on which we further give the sufficient condition for ensuring a lower upper boundary compared with exploration-tentative methods through Theorem 2.

Proposition 2. *The extrapolation error for BCQ, $\epsilon_{s,a}$, is upper bounded by*

$$\begin{aligned}\bar{\epsilon}_{s,a} = (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}(N\tau)^{-1/2} \\ [1+\gamma+...+\gamma^{(n)}+...]\end{aligned} \quad (4)$$

Theorem 2. *When $\tau > \frac{1}{|A|}$, The upper bound of extrapolation error for BCQ-like algorithms is strictly less than exploration-tentative methods.*

2.4.2. BAIL-LIKE ALGORITHMS

Instead of choosing (S, A) pairs with $N(S, A) > N\tau$, the underlying mechanism of BAIL is to select tuples with top returns given under each state for offline training. The data selection scheme can be either percentile-based or difference-based. As claimed by the author, the first one performs better, and thus is considered in this paper. For notational convenience and easier understanding, the percentile is still notated by τ for BAIL-like algorithms. The dataset after percentile-based selection is denoted as $\hat{\mathcal{D}}$, with the corresponding MDP $\hat{M}$. The number of occurrence of tuple (s, a) is denoted as $\hat{N}(S, A)$.

Intuitively, the distribution of $\hat{N}(s, a)$, $\forall(s, a) \in \mathcal{S} \times \mathcal{A}$ would be of greater variance and can barely be lower bounded by a specified value, such as $N\tau$ in BCQ-like case, due to the selection merely based on episode return. Thus, it seems quite challenged to derive an explicit expression as same as equation (15).

Assumption 2.5. *The expectation of each $\hat{N}(s, a)$ is $N(s, a)\tau$.*

However, given assumption 2.5, we are able to derive the upper bounded extrapolation error and its corresponding minimum for BAIL-like algorithms.

Proposition 3. *The expectation of extrapolation error for BAIL-like algorithms, $\epsilon_{s,a}$, is upper bounded by*

$$\begin{aligned}\mathbb{E}(\bar{\epsilon}_{s,a}) = (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}(N\tau)^{-1/2} \\ [(\pi_b(a|s)^{-1/2} + \gamma\sum_{s'} p_1(s'|s,a)\sum_{a'}\pi_b(a'|s')^{1/2} \\ + ... + \gamma^{(n)}\sum_{s'} p_1(s'|s,a)\sum_{a'}\pi_b(a'|s')\sum_{s''}... \\ \sum_{s^{(n)}} p_1(s^{(n)}|s^{(n-1)}, a^{(n-1)}) \\ \sum_{a^{(n)}}\pi_b(a^n|s^n)^{1/2} + ...]\end{aligned} \tag{5}$$

Proposition 4. *The general term of extrapolation error, equation (5), reaches its minimum when $\pi_b(a|s)$ is uniform, i.e., a pure random policy. Also, the more even the action distribution under each state, $\pi_b(a|s)$, is, the less value it would be.*

Specifically, the corresponding optimal value of extrapolation error would be

$$\begin{aligned}\mathbb{E}^*(\bar{\epsilon}_{s,a}) = (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}(N\tau)^{-1/2} \\ [|\mathcal{A}|^{-1/2} + \gamma|\mathcal{A}|^{1/2} + ... + \gamma^{(n)}|\mathcal{A}|^{1/2} + ...]\end{aligned} \tag{6}$$

On the other hand, when π_b is deterministic, the optimal value of extrapolation error is

$$\begin{aligned}\mathbb{E}^*(\bar{\epsilon}_{s,a}) = (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}(N\tau)^{-1/2} \\ [1 + \gamma + ... + \gamma^{(n)} + ...]\end{aligned} \tag{7}$$

2.5. Comparisons among algorithms

Compared with exploration-tentative algorithms, exploitation-tentative algorithms are most likely to have less extrapolation error. As for exploitation-tentative algorithms, from Proposition 4, we notice that the best case of BAIL-like algorithm is the same as the general case of BCQ-like algorithms. From the perspective of extrapolation error, BCQ-like algorithms provably performs better than the others.

3. Top Return Batch Constrained Q-learning (TR-BCQ)

According to the results of Section 2, exploitation-tentative algorithms would achieves less extrapolation error. However, less extrapolation error does not equivalent to be better overall performance (i.e., estimate Q-value ± extrapolation error). The redestimated Q-value is critical as well. Compared with BAIL-like algorithms, BCQ-like algorithms adopt extra off-policy optimization techniques for a higher estimated Q-value, and thus is expected to have a better overall performance. Besides, for the exploitation-tentative algorithms, a good "teacher" is of great importance due to their imitation-based essentials. Thus, we explore the weakness of BCQ as it would suffer from the dataset generated by the behavioral policy of low mean episode returns. To fix this shortcoming, we propose a variant of BCQ, named top return batch constrained Q-learning (TR-BCQ). For easy understanding and avoiding confusions, we will use "low-quality dataset" or directly "low dataset" for the same meaning of "dataset generated by the behavioral policy of low mean episode returns" in the following context.

The proposed TR-BCQ is basically consisted of two phases:

Phase 1 – Top return data selection: In this phase, we select tuples with high episode return based on a percentile parameter, ζ;

Phase 2 – Tuple visitation constrained optimization: In this phase, we proceed off-policy optimization on a visitation constrained batch of tuples.

Please refer to the pseudocode in the appendix.

We further explain advantages of our algorithm over original BCQ not merely on low-quality dataset, starting with the investigation of extrapolation errors as follow

$$|Q_{M}^{\pi_{BCQ}} - Q_{M^*}^{\pi_{BCQ}}| \leq \bar{\epsilon}_1, |Q_{\hat{M}}^{\pi_{TRCQ}} - Q_{M^*}^{\pi_{TRCQ}}| \leq \bar{\epsilon}_2 \quad (8)$$

where M indicates the underlying MDP of original offline dataset, M^* is the MDP in the realistic world, $\hat{M}$ represents the MDP of top-return selected offline dataset.

Proposition 5. *Using same off-policy optimization techniques, the relationship between the upper bound of extrapolation error generated on $\hat{M}$ and M is as follows:*

$$\mathbb{E}(\hat{\bar{\epsilon}}_2) = \mathbb{E}(\bar{\epsilon}_1)\zeta^{-1/2} \quad (9)$$

Additionally, since imitating the selected tuples with higher episode returns, the policy derived by TR-BCQ would achieve higher estimated Q-value compared with the original BCQ, i.e., $Q_{M}^{\pi_{BCQ}} < Q_{\hat{M}}^{\pi_{TRCQ}}$.

Nevertheless, due to the uncertainty introduced by $\bar{\epsilon}_2$, TR-BCQ might not gain a better performance. Meanwhile, we notice that the extrapolation error rather changes slightly along with ζ, e.g., $\mathbb{E}(\hat{\bar{\epsilon}}_2) = 1.29\mathbb{E}(\bar{\epsilon}_1)$ when $\zeta = 60\%$, i.e., tuples of top 40% episode returns are selected. It indicates that we are able to achieve higher overall performance (i.e., estimate Q-value $\pm$ extrapolation error) by increasing extrapolation error a little bit in return for higher estimate Q-value.

It should be noted that although the name of proposed algorithm contains "BCQ", BCQ is not the only option for Phase 2. More advanced extrapolation error-insensitive techniques are applicable for further improvement, yet which is out of the scope of this paper.

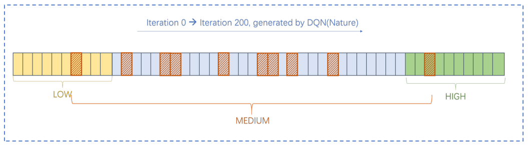


Figure 2. **Datasets of Tri-level Quality**

4. Experiments on Atari 2600 Games

4.1. Experiment setup

We generate the dataset through an online DQN agent from scratch in the Atari 2600 Games. Considering better understanding of readers and space limitation, we list experimental results towards ten randomly selected games in this section. More results can refer to Appendix.

4.2. The benchmark platform: RL easy go

Fig. 2 indicates the iterations of different episode returns during the whole training process, and how we divide the dataset into three subsets, named "low", "medium", and "high" according to their mean episode returns. We named our benchmark platform RL easy go (RLEG), which enables a lighter and faster evaluation of off line RL experiments. Given Assumption 2.2 being satisfied, we are able to derive an equivalence between "low return" with "high randomness"

The reasons why we do not directly quantify randomness is that since considering continuous state space, we are exposed with uncountable states. Thus, the results of performance would strongly rely on the granularity of discretized states. Although the more fine granularity is, the more accurate the result would be, and corresponding computational overhead would be undesirable.

4.3. Performance of existing offline RL algorithms on various datasets

In this section, we will show the experimental results of exploration and exploitation-tentative algorithms on "low", "medium" and "high" quality datasets.

In Fig. 3, we can clearly distinguish the monotonically trend of performance, which are deceasing and increasing for exploration-tentative and exploitation-tentative algorithms respectively, along with dataset randomness (represented in the form of return). Refer to the statistical comparison results in table 3 for a more clear insight. We notice that exploration-tentative algorithms perform better on "low" dataset, i.e., high randomness and low-quality dataset, which conforms to the analysis in section 2.3. Besides, exploitation-tentative algorithms outperform on "high" dataset than exploration-tentative candidates. On "medium" datasets, exploitation-tentative algorithms behaves best on half games. Based on the above results, we would recommend giving top priority to exploitation-tentative algorithms on high-quality dataset, while trying both types of algorithms on low-quality dataset. Admittedly, defining the quality of an offline dataset is empirical, our suggestion is to take both the mean episode return and the action distribution into accounts.

4.4. Comparison between TR-BCQ with the best of existing algorithms

Fig. 4 shows the performance of TR-BCQ under different percentages of data selection in Phase 1, which is one of the most critical hyperparameters and can be set in some heuristic ways. As shown in Fig. 4, it is not always better to train with as much data as possible. By choosing an appropriate ζ, the proposed algorithm is able to achieve higher online performance. From Fig. 4, TR-BCQ outperforms original

Table 3. Comparison on dataset quality

		Algorithms					
		BCQ	BAIL	DQN	MultiHeadDQN	Quantile	REM
Trend of Performance with Dataset Quality	Increase	41.667%	43.333%	15.000%	6.667%	5.000%	6.667%
	Decrease	21.667%	21.667%	56.667%	66.667%	48.333%	53.333%
Num of Best Scores	Low	11.667%	16.667%	0.000%	10.000%	55.000%	6.667%
	Medium	25.000%	25.000%	0.000%	5.000%	41.667%	3.333%
	High	33.667%	20.000%	1.667%	8.333%	30.000%	3.333%

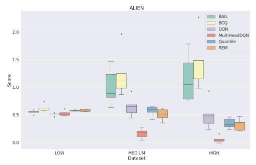
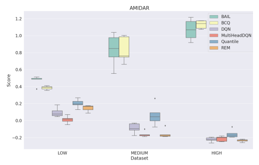
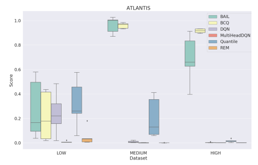
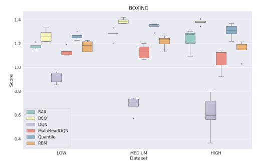
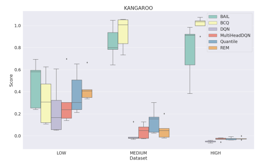
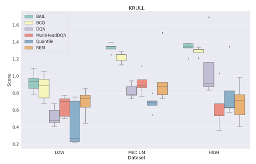
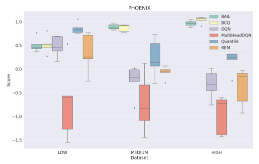
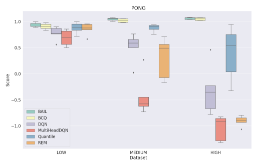
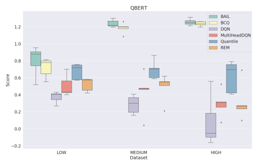
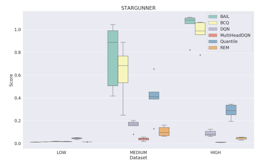

(a) Alien

(b) Amidar

(c) Atlantis

(d) Boxing

(e) Kangaroo

(f) Krull

(g) Phoenix

(h) Pong

(i) Qbert

(j) StarGunner

Figure 3. **Comparison for Different Dataset Qualities.** The mean score of top 20 iterations are used for fair comparison. The boxplot indicates scenarios of 5 different random seeds.

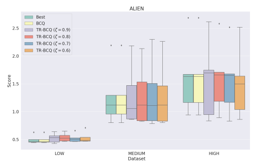
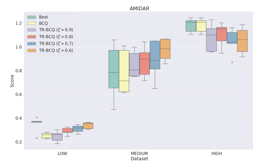
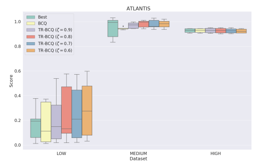
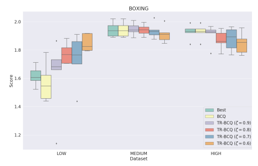
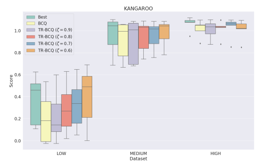
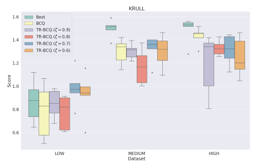
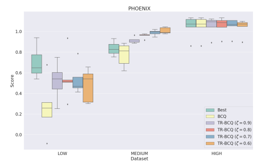
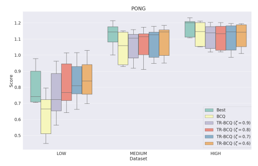
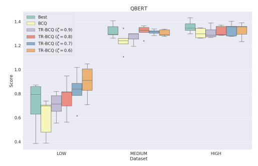
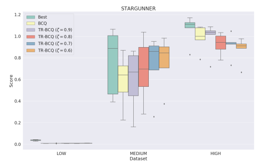

(a) Alien

(b) Amidar

(c) Atlantis

(d) Boxing

(e) Kangaroo

(f) Krull

(g) Phoenix

(h) Pong

(i) Qbert

(j) StarGunner

Figure 4. **Performance of TR-BCQ and SOTA algorithms on datasets of different qualities** In low, meidum, and high dataset, TR-BCQ performs best on 70%, 50%, and 30% of the games respectively.

BCQ a lot in “low” dataset, while less in “high” dataset. This implies that for dataset with lower quality, TR-BCQ is an indeed method with controllable extrapolation error, even if the number of data sets is greatly reduced. On the other hand, with the mean episode return increasing along with the dataset quality, the extrapolation error is of greater importance in dataset with higher quality.

Overall, we recommend TR-BCQ for “low” and “medium” dataset and any exploitation-tentative or simply imitation-based algorithms for “high” dataset.

5. Conclusion and Future Work

In this paper, we start with proposing a two-fold taxonomy for existing offline RL methodologies, i.e., exploration and exploitation-tentative algorithms. Then, with the help of derived upper bound of extrapolation error, we explore and prove the dependence of algorithm performance on dataset, especially for the action distribution for each state. From such a dataset perspective, although BCQ is provably better than the other, we identify its weak performance on dataset of low mean episode returns. Accordingly, we propose a modified BCQ based on a top return-based data selection mechanism. Our experimental results indicates that our algorithm could reach the best performance on various datasets. At last, a benchmark platform is created on the Atari domain, where we further open-source all datasets and checkpoints as a fair and comprehensive environment for competition between offline RL techniques.

Along with our top return-based data selection, some tuples are discarded and therefore is not used for constructing the underlying MDP. How to make full use of these data would be worth to investigated. Besides, for generalizing the proposed algorithms, we will extend it to the case of continuous state space and action space.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In International Conference on Machine Learning, pp. 104–114. PMLR, 2020.

Anschel, O., Baram, N., and Shimkin, N. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In International Conference on Machine Learning, pp. 176–185. PMLR, 2017.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707.06887, 2017.

Cai, Q., Filos-Ratsikas, A., Tang, P., and Zhang, Y. Reinforcement mechanism design for e-commerce. In Proceedings of the 2018 World Wide Web Conference, pp. 1339–1348, 2018.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K. Bail: Best-action imitation learning for batch deep reinforcement learning. Advances in Neural Information Processing Systems, 33, 2020.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. arXiv preprint arXiv:1710.10044, 2017.

Faußer, S. and Schwenker, F. Neural network ensembles in reinforcement learning. Neural Processing Letters, 41 (1):55–69, 2015.

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. In International Conference on Machine Learning, pp. 2052–2062. PMLR, 2019b.

Ghavamzadeh, M., Petrik, M., and Chow, Y. Safe policy improvement by minimizing robust baseline regret. Advances in Neural Information Processing Systems, 29: 2298–2306, 2016.

Hu, Y., Da, Q., Zeng, A., Yu, Y., and Xu, Y. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 368–377, 2018.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In Advances in Neural Information Processing Systems, pp. 11784–11794, 2019.

Laroche, R., Trichelair, P., and Des Combes, R. T. Safe policy improvement with baseline bootstrapping. In International Conference on Machine Learning, pp. 3652–3661. PMLR, 2019.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. Advances in neural information processing systems, 29:4026–4034, 2016.

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

Wang, Z., Merel, J., Reed, S., Wayne, G., de Freitas, N., and Heess, N. Robust imitation of diverse behaviors. arXiv preprint arXiv:1707.02747, 2017.

Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., and Weinberger, M. J. Inequalities for the l1 deviation of the empirical distribution. Hewlett-Packard Labs, Tech. Rep, 2003.

Xiao, S., Guo, L., Jiang, Z., Lv, L., Chen, Y., Zhu, J., and Yang, S. Model-based constrained mdp for budget allocation in sequential incentive marketing. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 971–980, 2019.

Zhao, X., Xia, L., Zhang, L., Ding, Z., Yin, D., and Tang, J. Deep reinforcement learning for page-wise recommendations. In Proceedings of the 12th ACM Conference on Recommender Systems, pp. 95–103, 2018.

A. Introduction of open-source benchmark: RL easy go (RLEG)

A.1. Workflow

The open sourced RLEG is a benchmark platform for offline reinforcement learning reproduction and evaluation, which consists of the following parts:

- Codes: We integrate all the source code of existing SOTA algorithms including BCQ, REM, Quantile, DQN, MultiHead-DQN, and BAIL into two frameworks based on our two-fold taxonomy (i.e., exploration- and exploitation-tentative). Besides, we open-source the source code of TR-BCQ under exploitation-tentative framework. We appreciate the open-source code of REM, BCQ, and BAIL. We build our code based on their work.
- Datasets: (i) All the checkpoints generated during the experiments; (ii) The tuples of the form (s, a, r, s', a', r', t) for low, medium, high-quality dataset. (iii) The mean episode return logs for data generation process and offline reinforcement learning process on the aforementioned algorithms. Researchers can check their own off line reinforcement learning algorithms on various dataset directly without running the costly data generation process.

B. Missing proofs

B.1. Proof of Proposition 1

First, we express the extrapolation error as:

$$\begin{aligned}
\epsilon_{s,a} &= Q_\pi^{MDP_1}(s,a) - Q_\pi^{MDP_2}(s,a) \\
&= \sum_{s'}[(p_1(s'|s,a) - p_2(s'|s,a)]r(s,a,s') + \sum_{s'} p_1(s'|s,a)\gamma \sum_{a'} \pi(a'|s')(Q_1(s',a') - Q_2(s',a')) + \\
&\quad \sum_{s'}(p_1(s'|s,a) - p_2(s'|s,a))\gamma \sum_{a'} \pi(a'|s') Q_2(s',a') \\
&\leq R_{max} \sum_{s'}[(p_1(s'|s,a) - p_2(s'|s,a)] + \sum_{s'} p_1(s'|s,a)\gamma \sum_{a'} \pi(a'|s')(Q_1(s',a') - Q_2(s',a')) + \\
&\quad \sum_{s'}(p_1(s'|s,a) - p_2(s'|s,a))\gamma \sum_{a'} \pi(a'|s') \frac{R_{max}}{(1-\gamma)} \\
&= \frac{R_{max}}{(1-\gamma)} e(s,a) + \gamma \sum_{s'} p_1(s'|s,a) \sum_{a'} \pi(a'|s') \frac{R_{max}}{(1-\gamma)} e(s',a') + ... + \\
&\quad \gamma^{(n)} \sum_{s'} p_1(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} ... \sum_{s^{(n)}} p_1(s^{(n)}|s^{(n-1)}, a^{(n-1)}) \sum_{a^{(n)}} \pi(a^{(n)}|s^{(n)}) \frac{R_{max}}{(1-\gamma)} e(s^{(n)}, a^{(n)}) + \\
&\quad ...
\end{aligned} \tag{10}$$

Given Lemma 2.2 and Assumption 2.1 to be satisfied, equation (1) could be readily derived.

B.2. Proof of Theorem 1

The main problem we focus now turns to be the following expression:

$$\sum_a \pi(a|s) \frac{R_{max}}{(1-\gamma)} N^{-1/2} (\pi_b(a|s))^{-1/2}, \forall s \in \mathcal{S}$$

We need to prove that no matter what distribution $\pi(a|s)$ is, the expression reaches its minimum when $\pi_b(a|s)$ is uniform. Also, the more even the distribution $\pi_b(a|s)$ is, the less expression value would be.

The result for π_b when $|\mathcal{A}| = 2$ will be given firstly. Then the generalized proof will be presented.

Case: $n = 2$

$$\begin{aligned} \min \quad & \mathbb{E}[\sum_{a\in\mathcal{A}} \frac{p_i}{\sqrt{q_i}}] \\ s.t. \quad & q_1 + q_2 = 1 \end{aligned} \tag{11}$$

To solve this problem,

$$\begin{aligned} q &= \arg\min \frac{1}{(n-1)!}|_{n=2} \int_0^1 \frac{p_1}{\sqrt{q_1}} + \frac{1-p_1}{\sqrt{q_2}} dp_1 \\ &= \arg\min \frac{1}{2\sqrt{q1}} + \frac{1}{2\sqrt{q2}} \\ s.t. \quad & q_1 + q_2 = 1 \end{aligned} \tag{12}$$

The solution is $q_1 = q_2 = \frac{1}{2}$ for Equation 12.

Consider the most general case as follow:

$$\begin{aligned} \mathbb{E}(\pi|\pi_b) \quad &= \frac{1}{(n-1)!}\int_0^1 dp_1 \int_0^{1-p_1} dp_2 ... \int_0^{1-p_1-p_2-...-p_{n-2}} \sum_{i=1}^{n-1} \frac{p_i}{\sqrt{q_i}} + \frac{p_n}{\sqrt{q_n}} dp_{n-1} \\ &= \frac{1}{(n-1)!}\int_0^1 dp_1 \int_0^{1-p_1} dp_2 ... \int_0^{1-p_1-p_2-...-p_{n-2}} \sum_{i=1}^{n-1} \frac{p_i}{\sqrt{q_i}} + \frac{1-p_1-p_2-...-p_{n-1}}{\sqrt{q_n}} dp_{n-1} \end{aligned} \tag{13}$$

We rewrite the above equation as

$$\mathbb{E}(\pi|\pi_b) = \sum_{i=1}^{n} \phi(p_i) \frac{1}{\sqrt{q_i}} \tag{14}$$

where $\phi(p_i) = \frac{1}{(n-1)!}\int_0^1 dp_1 \int_0^{1-p_1} dp_2 \cdots \int_0^{1-p_1-p_2-...-p_{n-2}} p_i dp_{n-1}$.

We notice that $\phi(p_i) = \int_0^1 dp_1 \int_0^{1-p_1} \cdots \int_0^{1-p_1-p_2-\cdots-p_{i-1}} \frac{(n-i)!}{(1-p_1-p_2-\cdots-p_{i-1})^{n-i}} dp_i$, and for each $i \in [1, n]$, $\phi(p_i)$ are same. This fact is also intuitively understandable, since all p_i, $i = 1, 2, \cdots, n$ are independent and identically distributed (i.i.d), we have $\phi(p_1) = \phi(p_2) = \cdots = \phi(p_n) = \mathbb{E}(p_i)$.

B.3. Proof of Proposition 2

In order to derive the upper bound of extrapolation error for BCQ, $\bar{\epsilon}_{s,a}$, we adopt zoom method using the result of Proposition 1 and $N(s,a) > N\tau$ as

$$\begin{aligned} \epsilon_{s,a} = (&\frac{R_{max}}{(1-\gamma)} e(s,a) + \gamma \sum_{s'} p_1(s'|s,a) \sum_{a'} \pi(a'|s') \frac{R_{max}}{(1-\gamma)} e(s',a') + ...+ \\ &\gamma^{(n)} \sum_{s'} p_1(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} ... \sum_{s^{(n)}} p_1(s^{(n)}|s^{(n-1)}, a^{(n-1)}) \sum_{a^{(n)}} \pi(a^{(n)}|s^{(n)}) \frac{R_{max}}{(1-\gamma)} e(s^{(n)}, a^{(n)}) + \\ &...)|_{N(s,a)>N\tau} \\ \le & (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2} [\frac{R_{max}}{(1-\gamma)} (N\tau)^{-1/2} + \gamma \sum_{s'} p_1(s'|s,a) \sum_{a'} \pi(a'|s') \frac{R_{max}}{(1-\gamma)} (N\tau)^{-1/2} + ...+ \\ &\gamma^{(n)} \sum_{s'} p_1(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} ... \sum_{s^{(n)}} p_1(s^{(n)}|s^{(n-1)}, a^{(n-1)}) \sum_{a^{(n)}} \pi(a^{(n)}|s^{(n)}) \frac{R_{max}}{(1-\gamma)} (N\tau)^{-1/2} + ...] \\ = & (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2} \frac{R_{max}}{(1-\gamma)} [(N\tau)^{-1/2} + \gamma (N\tau)^{-1/2} + ... + \gamma^{(n)} (N\tau)^{-1/2} + ...] \\ = & (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2} \frac{R_{max}}{(1-\gamma)} (N\tau)^{-1/2} [1 + \gamma + ... + \gamma^{(n)} + ...] \end{aligned} \tag{15}$$

B.4. Proof of Theorem 2

Given $\tau > \frac{1}{|A|}$, the extrapolation bound for BCQ satisfies that

$$\bar{\epsilon}_{s,a} \leq (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}(\frac{N}{|A|})^{-1/2}[1+\gamma+...+\gamma^{(n)}+...] \tag{16}$$

From Theorem 1, the minimum bound of exploration-tentative algorithm is achieved when π_b is uniform, i.e., $\pi(\cdot|s) = \frac{1}{|A|}$. Thus, the minimal extrapolation bound for exploration-tentative algorithm is

$$\begin{aligned}\bar{\epsilon}_{s,a} = (2log(\frac{|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}))^{1/2}\frac{R_{max}}{(1-\gamma)}N^{-1/2} \\ [(\frac{1}{|A|})^{-1/2} + \gamma\sum_{s'} p_1(s'|s,a)\sum_{a'}\pi(a'|s')(\frac{1}{|A|})^{-1/2} + ... + \gamma^{(n)}\sum_{s'} p_1(s'|s,a)\sum_{a'}\pi(a'|s')\sum_{s''}... \\ \sum_{s^{(n)}} p_1(s^{(n)}|s^{(n-1)},a^{(n-1)})\sum_{a^{(n)}}\pi(a^{(n)}|s^{(n)})(\frac{1}{|A|})^{-1/2} + ...]\end{aligned} \tag{17}$$

Note that not all terms of equation 17 exists in equation 16, because only (s,a) pairs that satisfy $N(s,a) > \frac{1}{|A|}$ are selected in BCQ. Therefore, Theorem 2 holds.

B.5. Proof of Proposition 5

In the first phase, because we process the data selection, for any (s,a) pair under the batch constraint, $\hat{N}(s,a) \leq N(s,a)$. Probabilistically, $\mathbb{E}(\hat{N}(s,a)) = \mathbb{E}(N(s,a))\zeta$

Thus, for every term in equation (15), the extrapolation error would increase after data selection, the expectation of which turns out to be

$$\mathbb{E}(\hat{\bar{\epsilon}}_{s,a}) = \mathbb{E}(\bar{\epsilon}_{s,a})\zeta^{-1/2} \tag{18}$$

C. Pseudocode of TR-BCQ

Please refer Algorithm 1.

D. Additional Experiments

D.1. Data Generation

Please refer Fig. 5.

D.2. Learning curves of all 60 Atari 2600 games on poor dataset

Please refer Fig. 6 and Fig. 7.

D.3. Learning curves of all 60 Atari 2600 games on medium dataset

Please refer Fig. 8 and Fig. 9.

D.4. Learning curves of all 60 Atari 2600 games on high dataset

Please refer Fig. 10 and Fig. 11.

D.5. Comparison on different datasets

Please refer Fig. 12 - Fig. 19

Algorithm 1 TR-BCQ Algorithm

```
Input: Offline dataset tuples $(\mathcal{S}, \mathcal{A}, \mathcal{S}', \mathcal{R}, \mathcal{T}, \mathcal{G})$ , data selection percentile $\zeta$, and number of iterations $T$;
Output: Policy $\pi$;
Initialization: Q-network $Q_\theta$, generative model $G_\omega$ and target network $Q_{\theta'}$.
                Phase 1: Top Return-based Data selection
a) Sort the tuples by $\mathcal{G}$.
b) Select top $(1-\zeta)$ percentage of tuples.
                Phase 2: tuple visitation constrained Q-learning
For $t = 1$ to $T$ {
a) Selecting the max valued action with $Q_\theta$
    $$a^{'} = \underset{a' | G_\omega(a'|s')/max\hat{a}G_\omega(\hat{a}|s') > \tau}{argmax} Q_\theta(s^{'}, a^{'})$$
b) Evaluating with $Q_{\theta'}$
    $$\theta \leftarrow \underset{\theta}{argmin} \sum_{\{s,a,r,s'\} \in \{\mathcal{S},\mathcal{A},\mathcal{R},\mathcal{S}'\}} \mathcal{L}(\theta)$$
where $\mathcal{L}(\theta) = l_{\mathcal{K}}(r + \gamma Q_{\theta'}(s^{'}, a^{'}) - Q_\theta(s, a))$
c) Behavioral cloning with $G_\omega$
    $$\omega \leftarrow \underset{\omega}{argmin} - \sum_{(s,a) \in \tau-\text{constrained dataset}} logG_\omega(s|a)$$
}
d) Update target network $Q_{\theta'}$ with $\theta^{'} \leftarrow \theta$
```

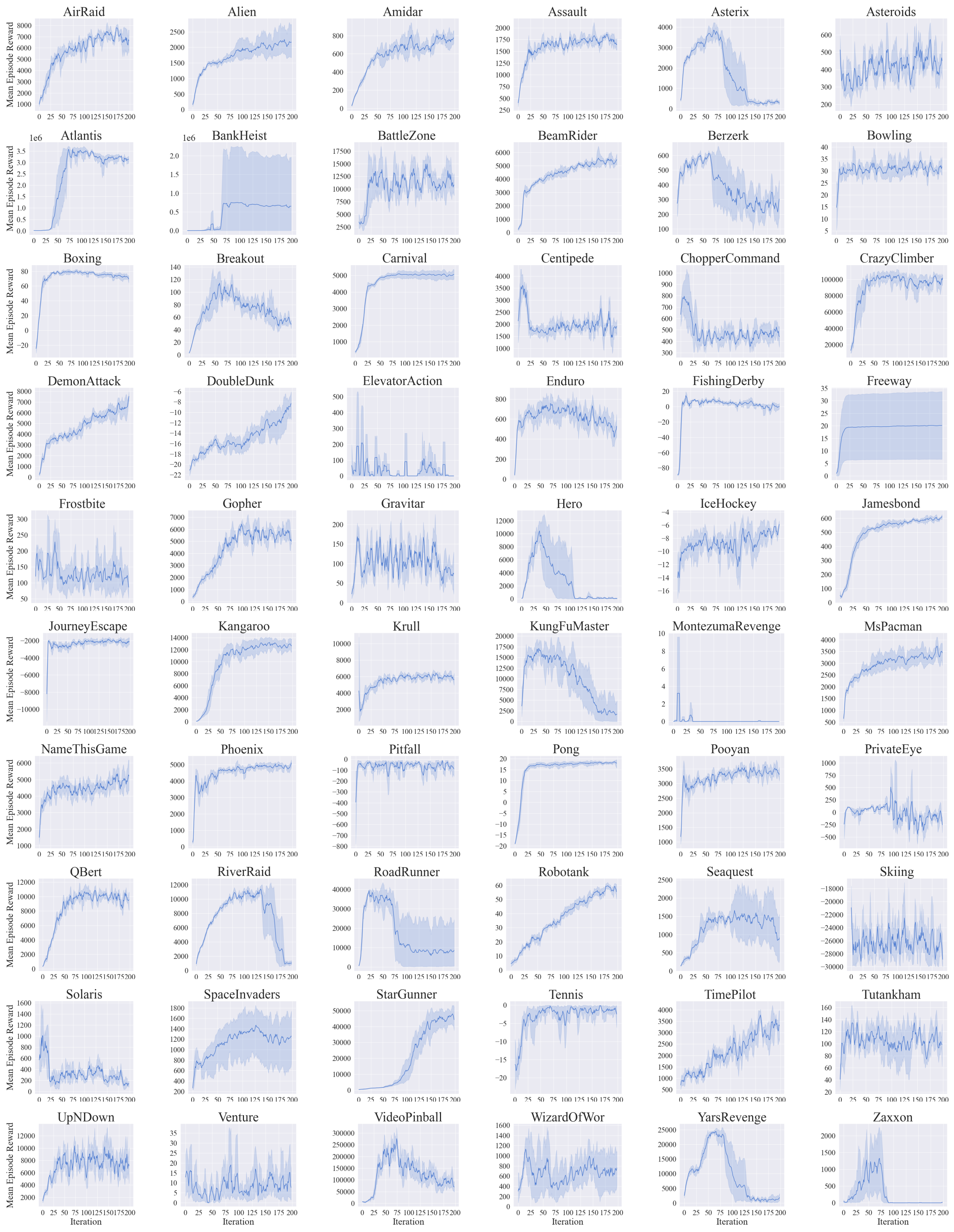


Figure 5. **Data Generation (by DQN)**

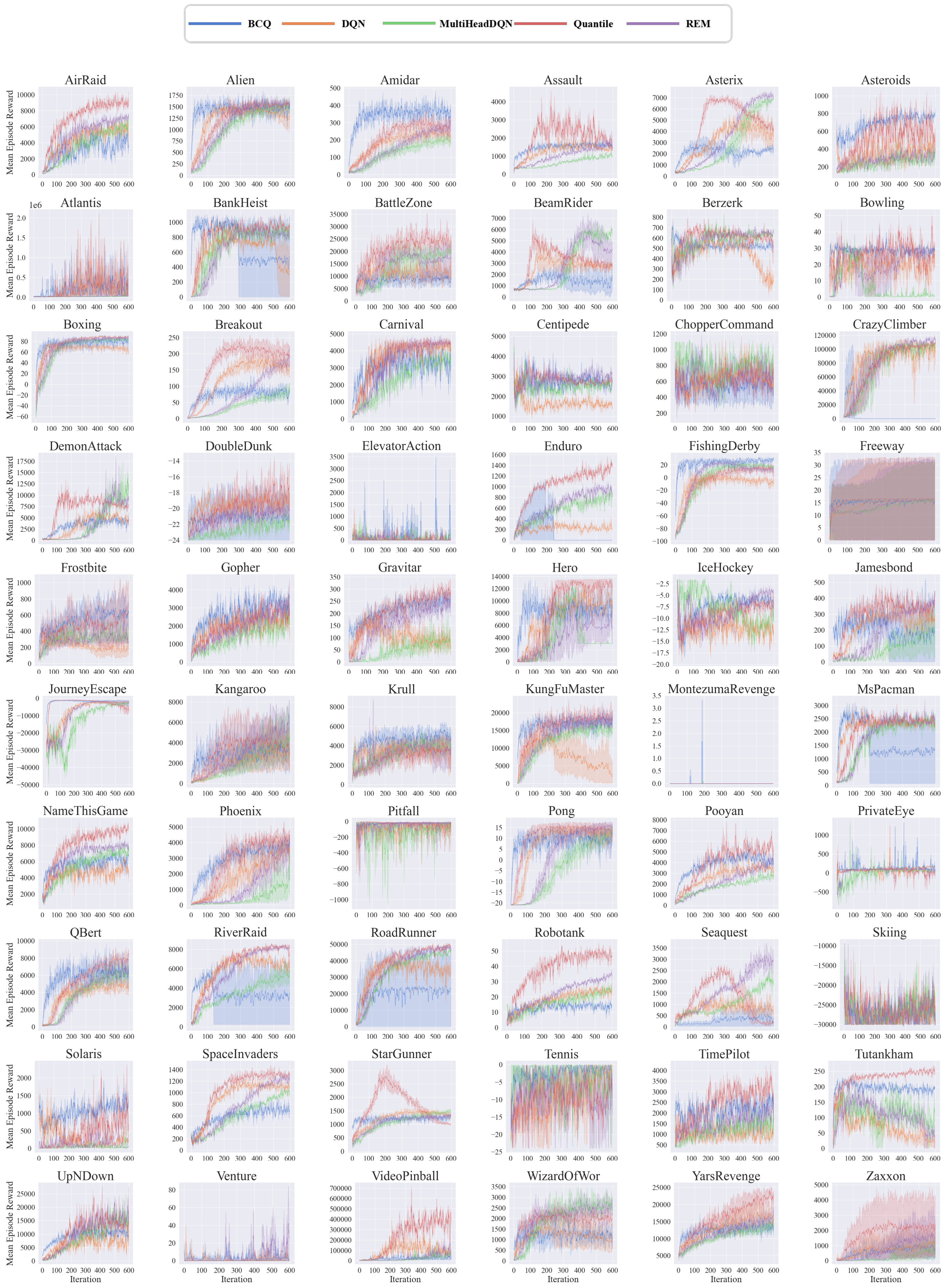


Figure 6. **Learning curves of all** 60 **Atari** 2600 **games on poor dataset**

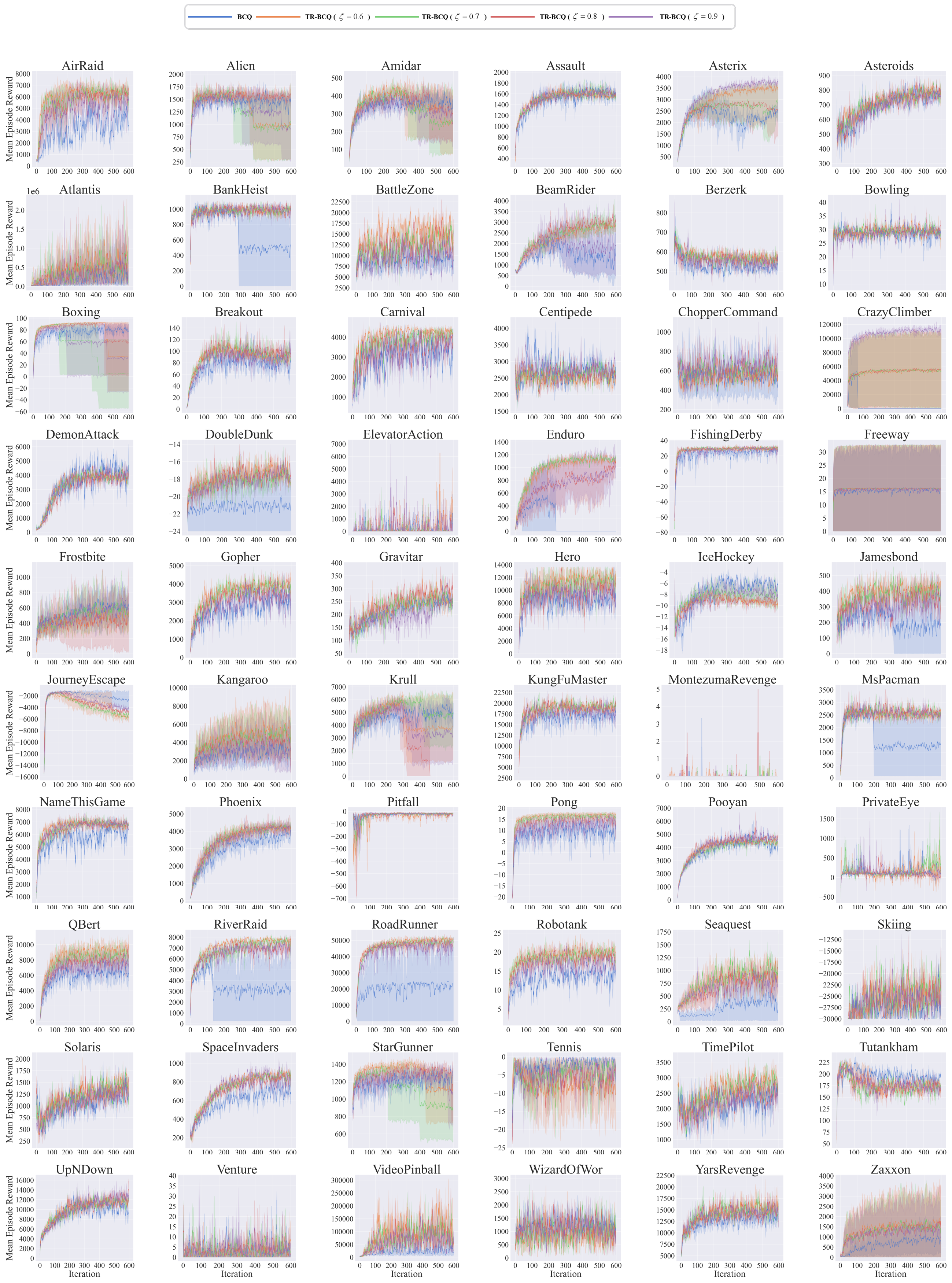


Figure 7. **Learning curves of all** 60 **Atari** 2600 **games on poor dataset**

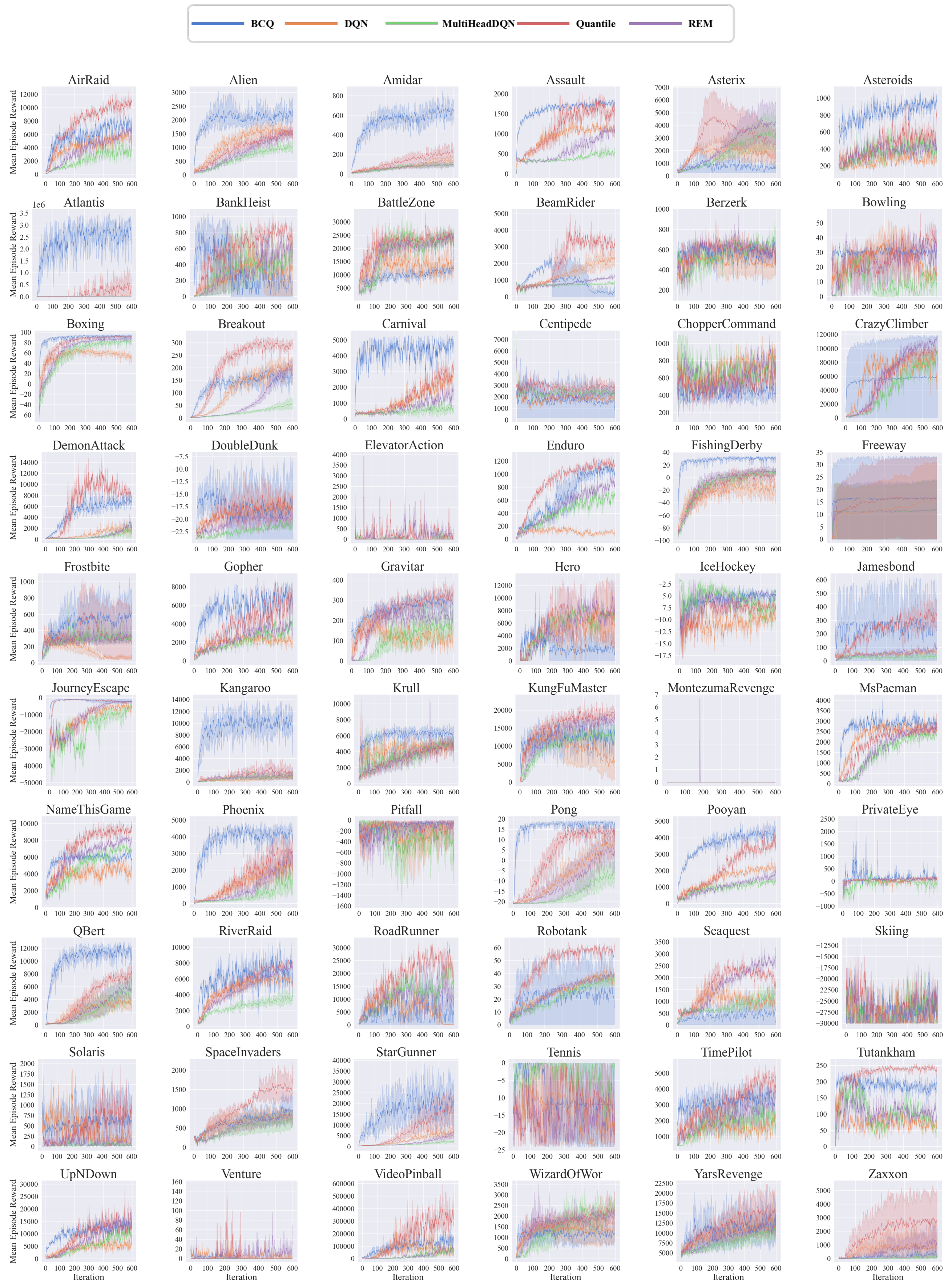


Figure 8. **Learning curves of all** 60 **Atari** 2600 **games on medium dataset**

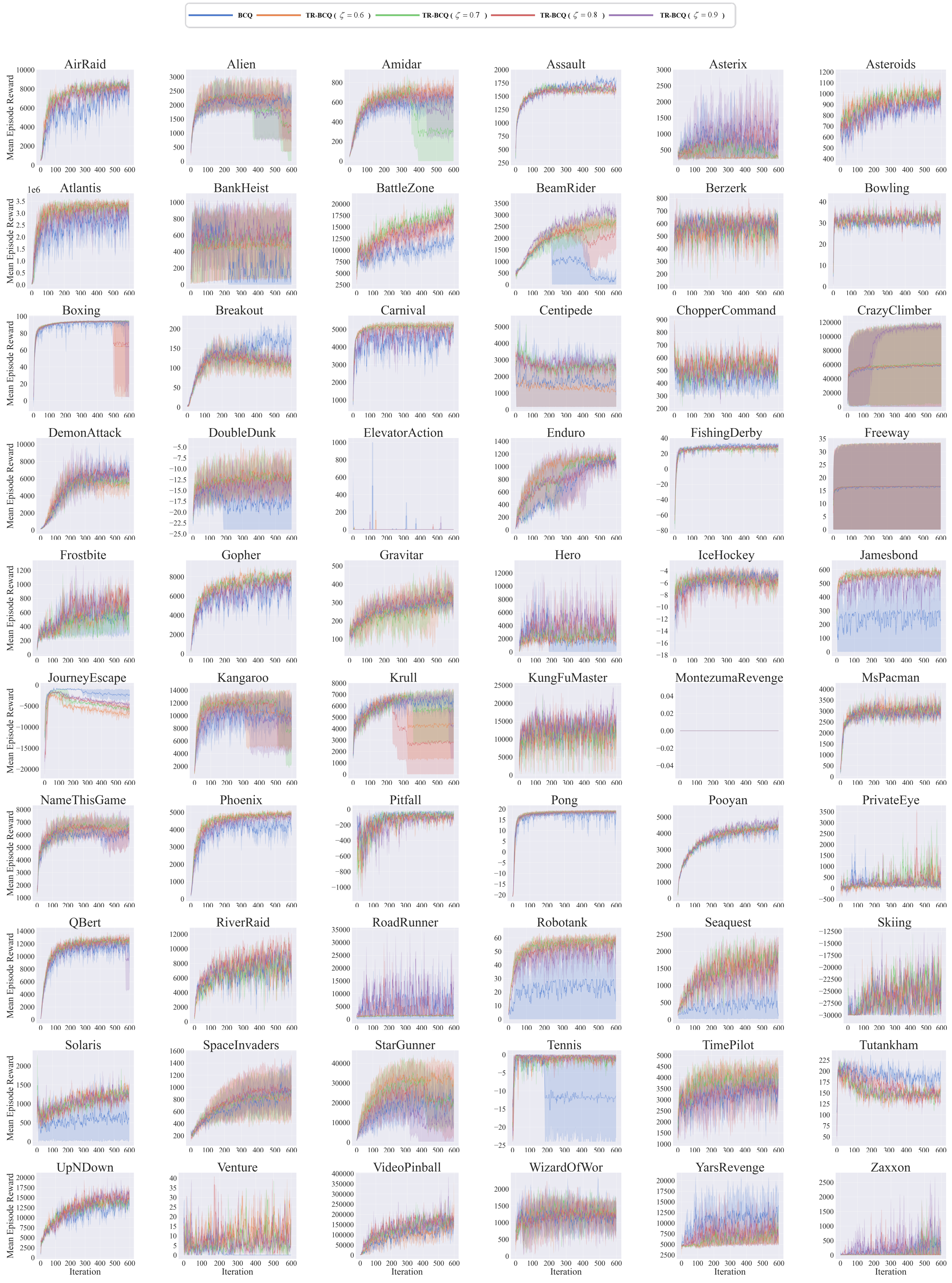


Figure 9. **Learning curves of all 60 Atari 2600 games on medium dataset**

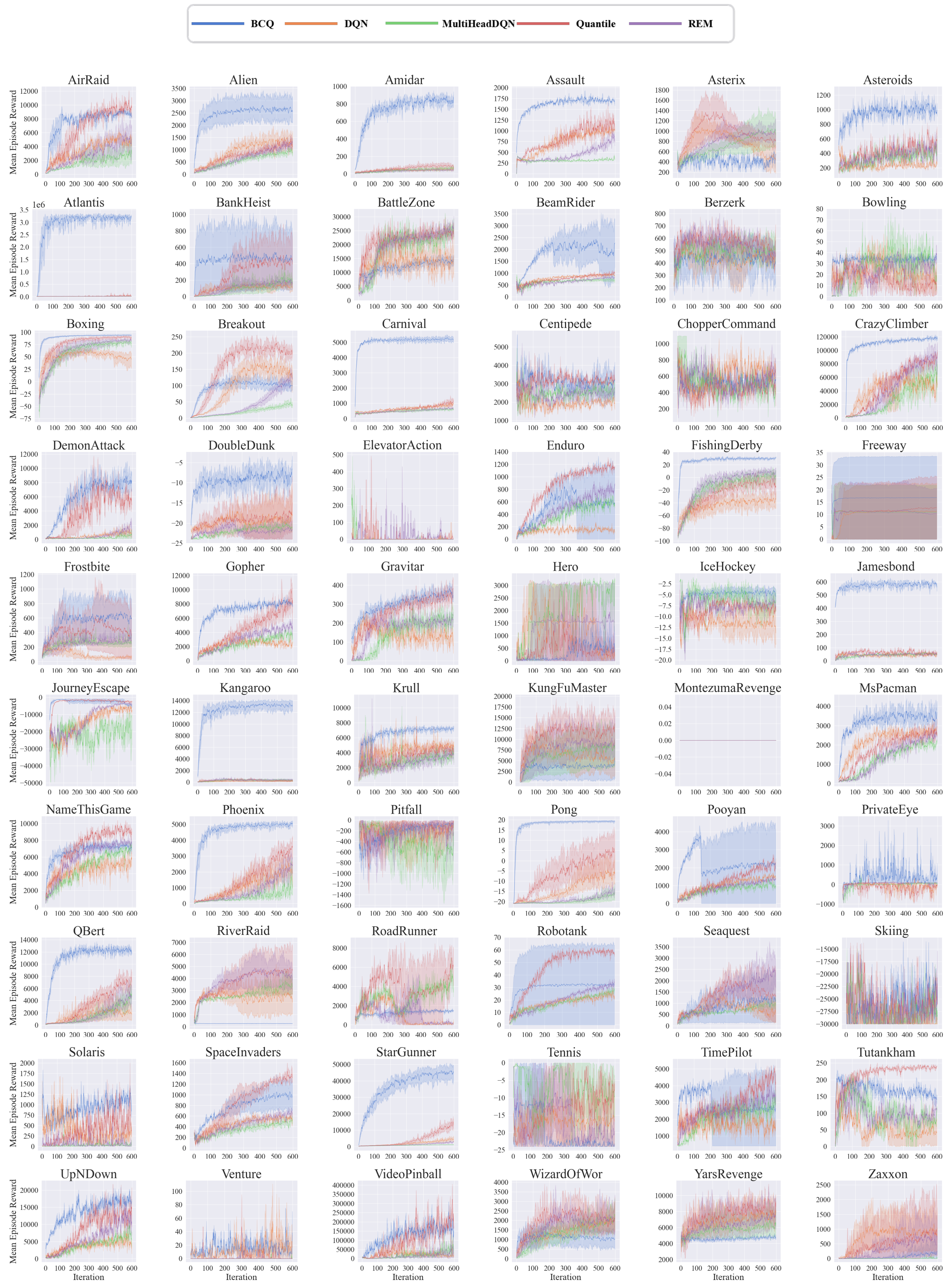


Figure 10. **Learning curves of all** 60 **Atari** 2600 **games on high dataset**

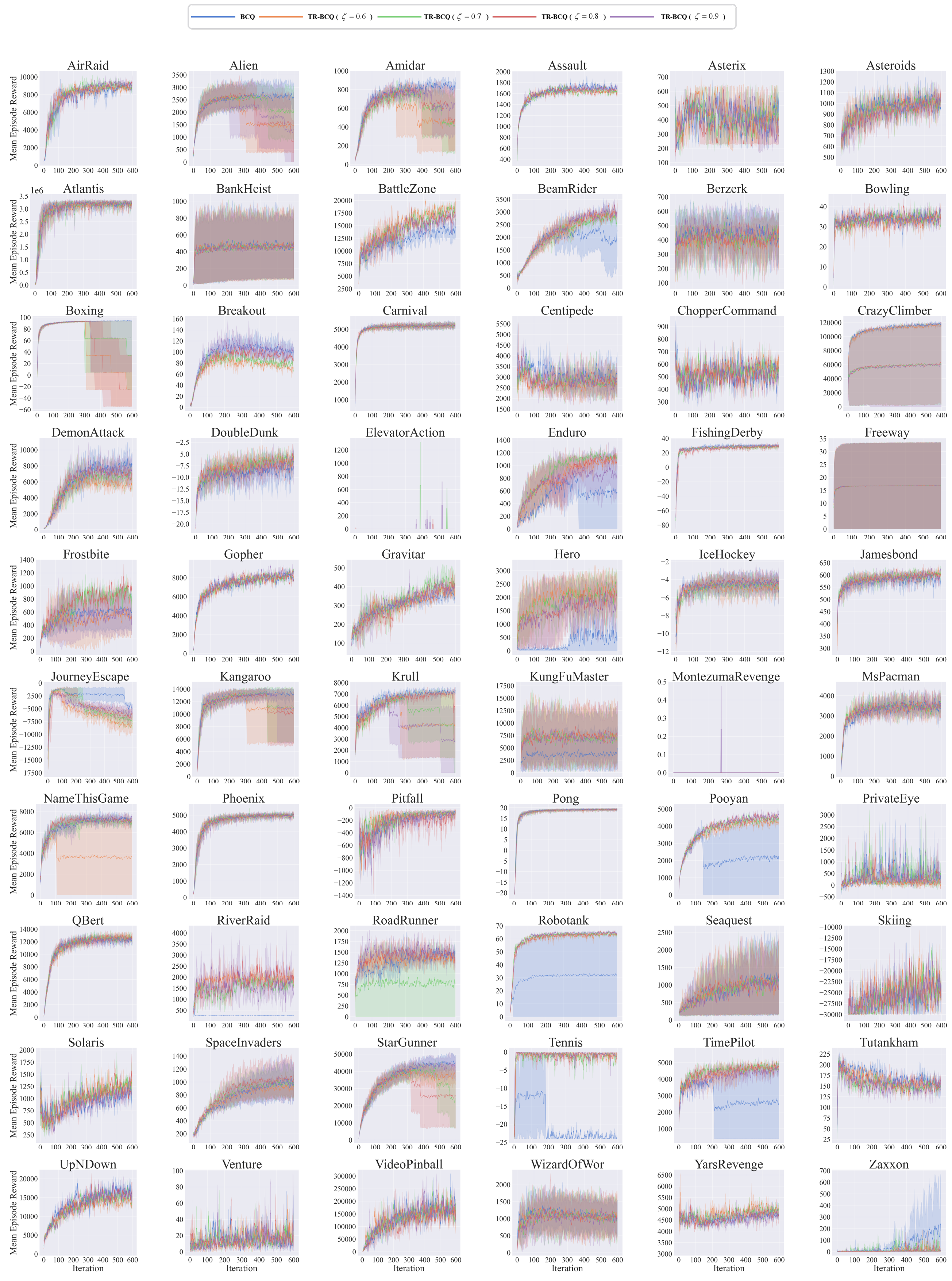


Figure 11. **Learning curves of all 60 Atari 2600 games on high dataset**

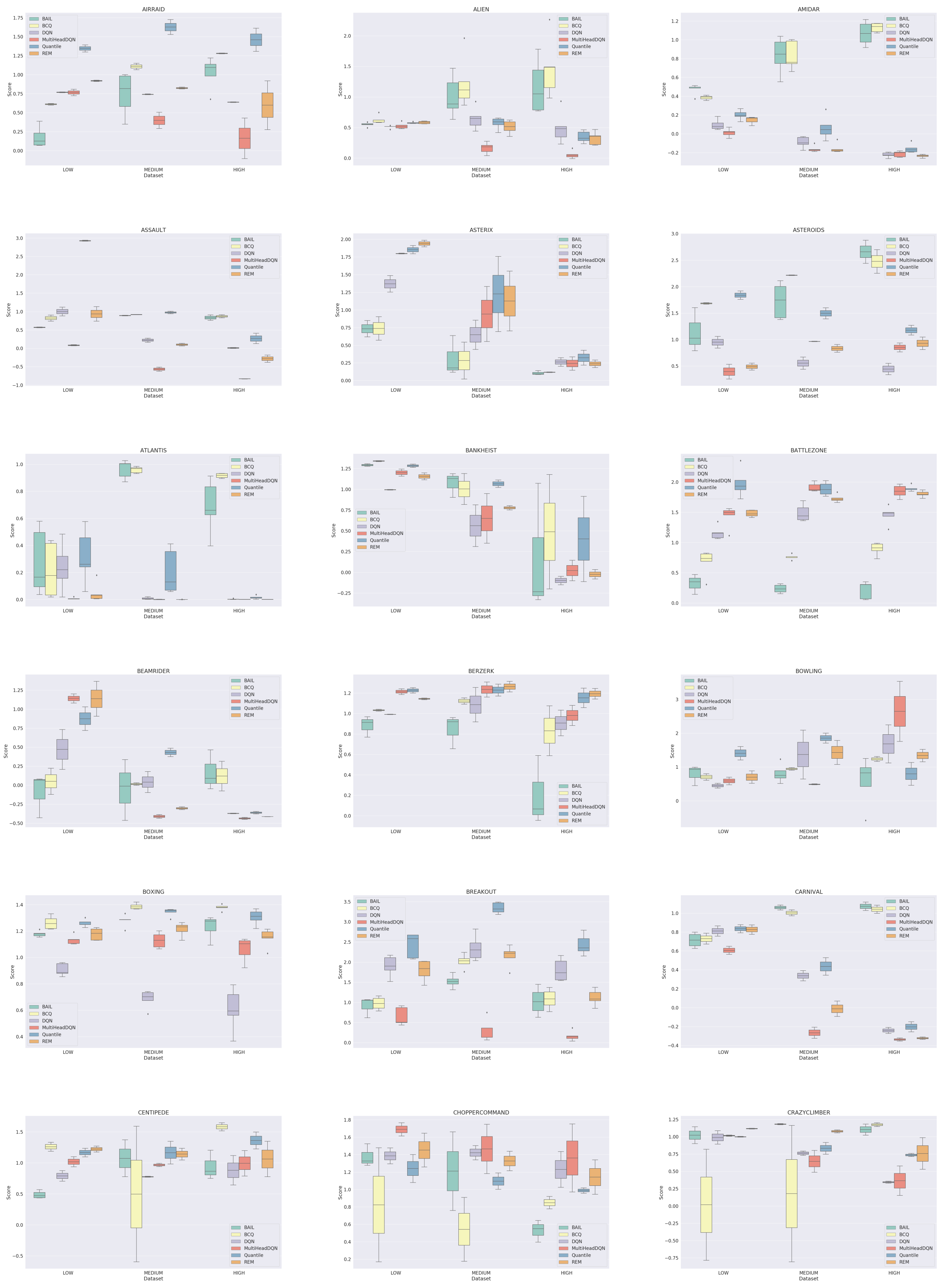


Figure 12. **Comparison between baselines on different datasets from Game Alien to Game CrazyClimber**

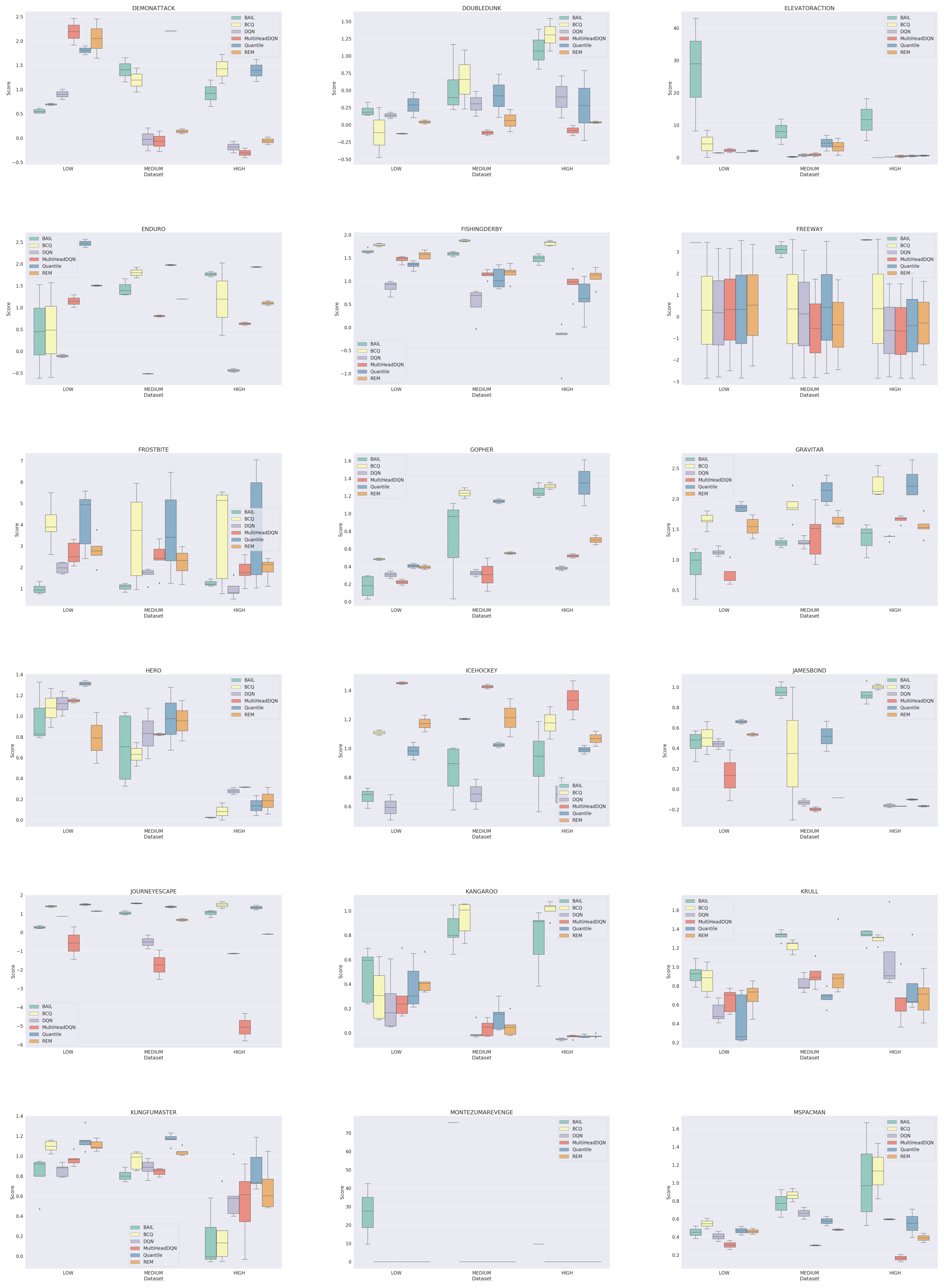


Figure 13. **Comparison between baselines on different datasets from Game DemonAttack to Game MsPacman**

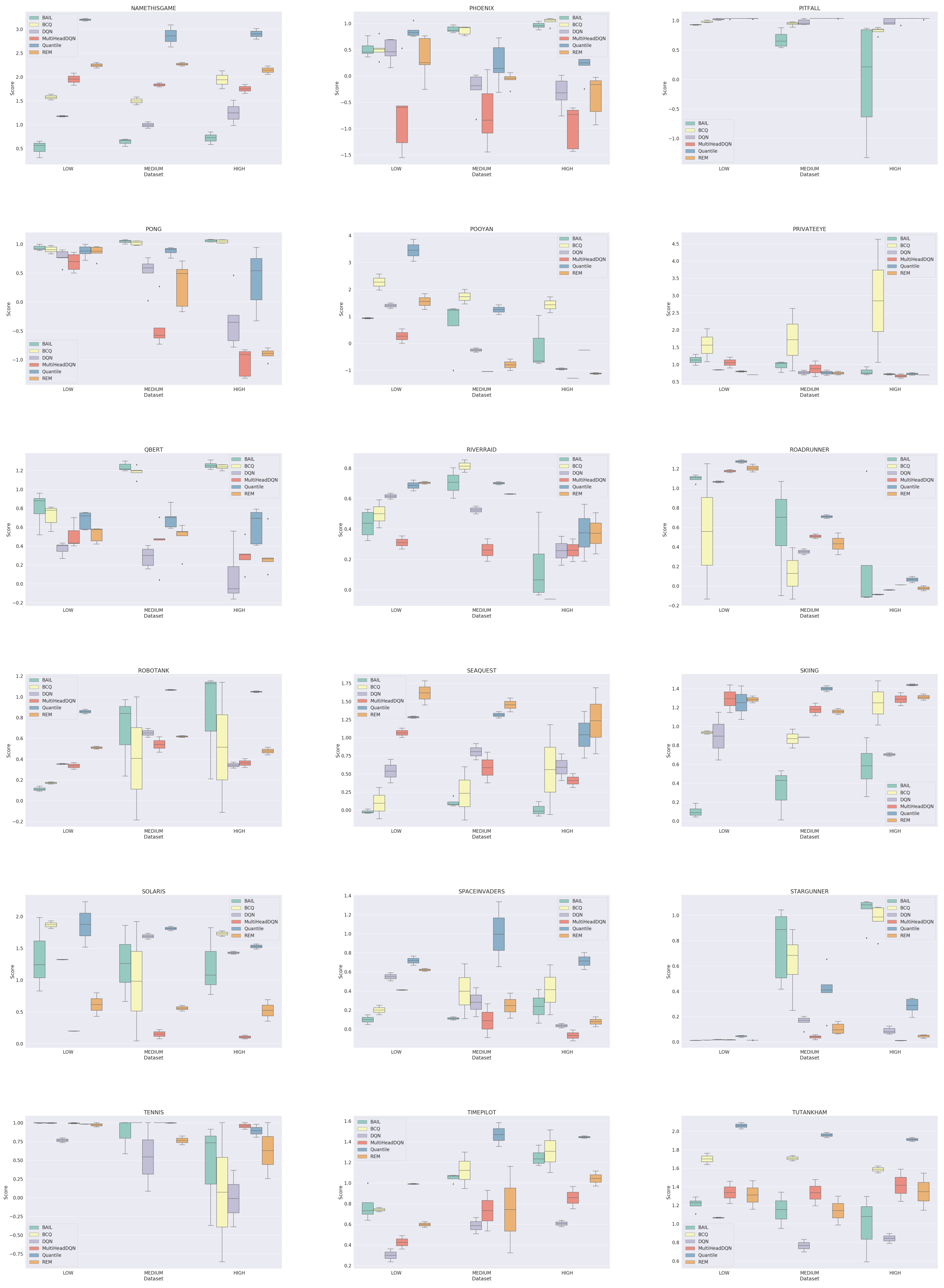


Figure 14. **Comparison between baselines on different datasets from Game NameThisGame to Game Tutankham**

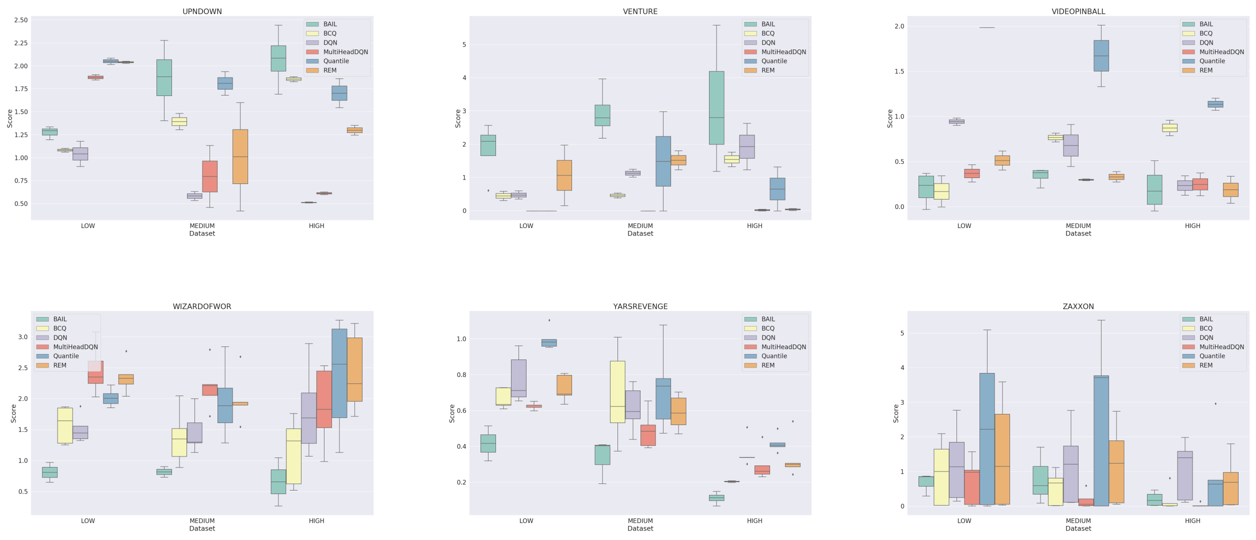


Figure 15. **Comparison between baselines on different datasets from Game UpNDown to Game Zaxxon**

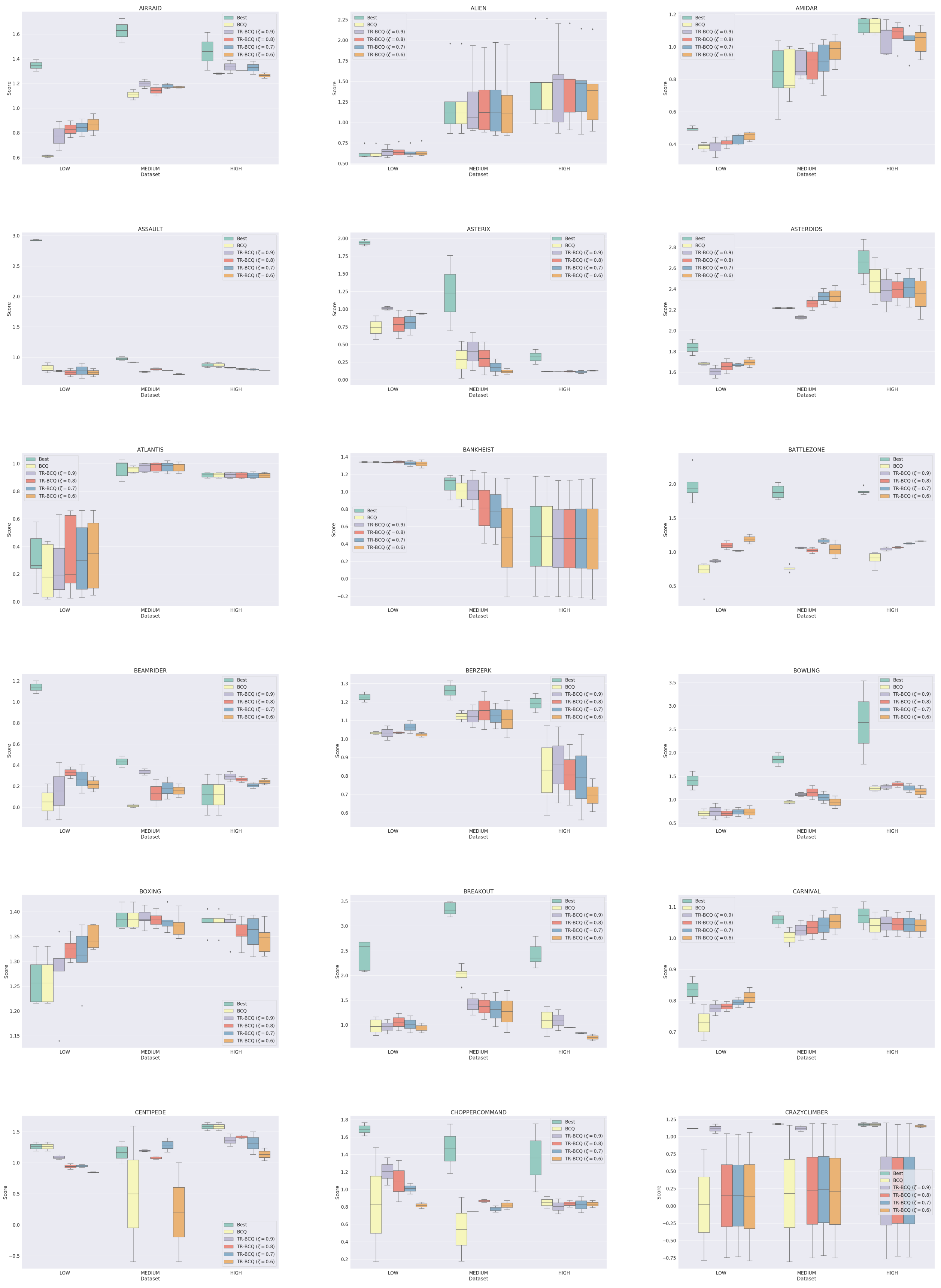


Figure 16. **Comparison between TR-BCQ and the best baselines on different datasets from Game Alien to Game CrazyClimber**

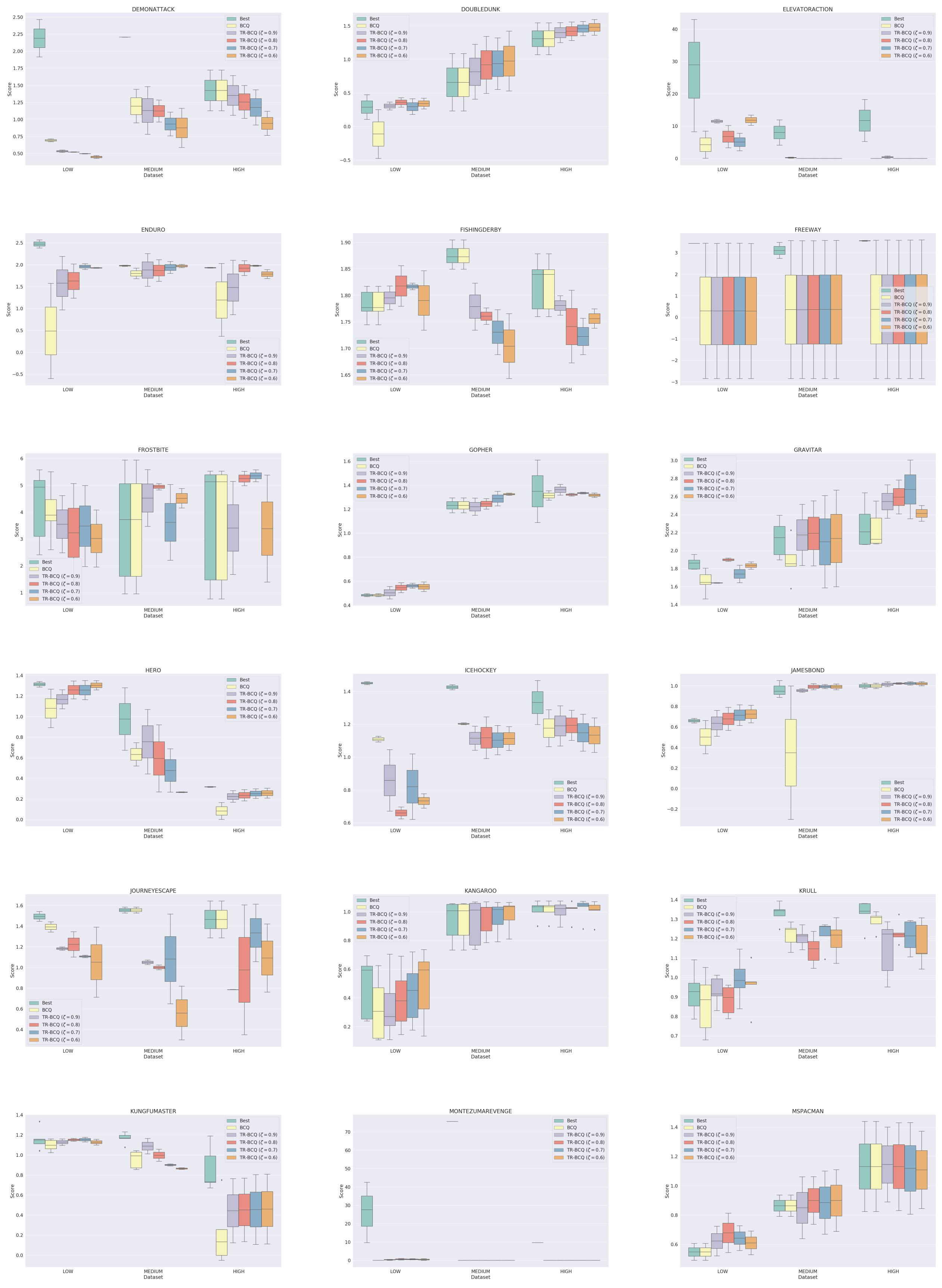


Figure 17. **Comparison between TR-BCQ and the best baselines on different datasets from Game DemonAttack to Game MsPacman**

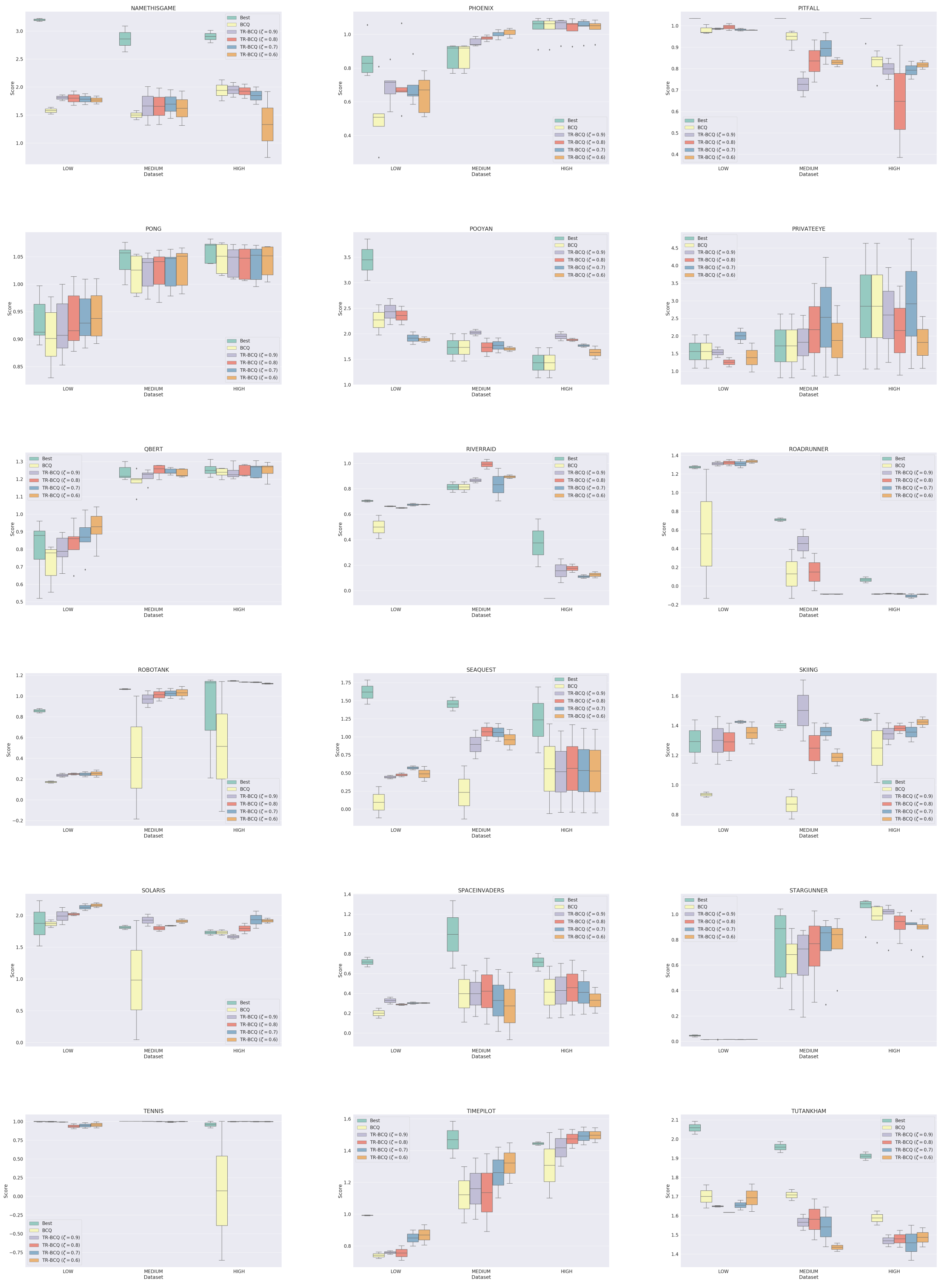


Figure 18. **Comparison between TR-BCQ and best baselines on different datasets from Game NameThisGame to Game Tutankham**

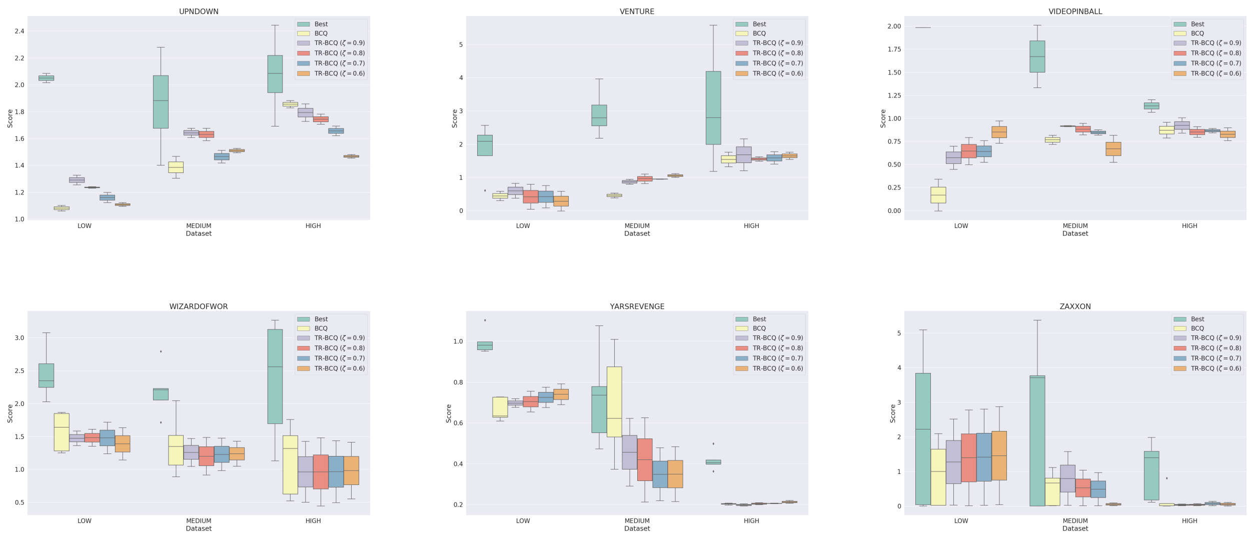


Figure 19. **Comparison between TR-BCQ and best baselines on different datasets from Game UpNDown to Game Zaxxon**