

# Revisiting Fundamentals of Experience Replay

William Fedus<sup>\*1,2</sup> Prajit Ramachandran<sup>\*1</sup> Rishabh Agarwal<sup>1</sup> Yoshua Bengio<sup>2,3</sup> Hugo Larochelle<sup>1,4</sup>  
 Mark Rowland<sup>5</sup> Will Dabney<sup>5</sup>

## Abstract

Experience replay is central to off-policy algorithms in deep reinforcement learning (RL), but there remain significant gaps in our understanding. We therefore present a systematic and extensive analysis of experience replay in Q-learning methods, focusing on two fundamental properties: the replay capacity and the ratio of learning updates to experience collected (*replay ratio*). Our additive and ablative studies upend conventional wisdom around experience replay — greater capacity is found to substantially increase the performance of certain algorithms, while leaving others unaffected. Counterintuitively we show that theoretically ungrounded, uncorrected  $n$ -step returns are uniquely beneficial while other techniques confer limited benefit for sifting through larger memory. Separately, by directly controlling the replay ratio we contextualize previous observations in the literature and empirically measure its importance across a variety of deep RL algorithms. Finally, we conclude by testing a set of hypotheses on the nature of these performance benefits.

to understand the interplay of learning algorithms and data-generating mechanisms in order to inform the design of better algorithms.

Earlier works investigating replay buffers have often focused on individual hyperparameters, such as the capacity of the replay buffer (Zhang & Sutton, 2017), which has typically been preserved since the seminal work in this field of Mnih et al. (2013; 2015). We begin by identifying that several such hyperparameters, such as buffer capacity and rate of data throughput, are interlinked in the manner that they affect experience replay, modifying both the amount of data available to an agent and the typical age of that data. This motivates a comprehensive set of experiments to understand the relative effects of modifying each property independently. We then make the surprising discovery that these effects depend critically on the presence of a particular algorithmic component,  $n$ -step returns, which has not previously been linked to experience replay. We conclude by examining several hypotheses to uncover why this link exists.

1. Understand relative effects of changing hyperparams.
2. Discover significance of  $n$ -step returns
3. Form Hypothesis

## 2. Background

We consider a Markov decision process  $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ , and denote the sequence of states, actions, and rewards experienced by an agent by  $(S_t)_{t \geq 0}$ ,  $(A_t)_{t \geq 0}$ , and  $(R_t)_{t \geq 0}$ , respectively. The central task of reinforcement learning is to find a policy  $\pi : \mathcal{S} \rightarrow \Delta_{\mathcal{A}}$  that maximizes the expected return

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t \geq 0} \gamma^t R_t \mid S_0 = s, A_0 = a \right],$$

for each initial state-action pair  $(s, a) \in \mathcal{S} \times \mathcal{A}$ . This problem is well studied, and a wide range of methods based on value iteration, policy iteration, and policy gradients have been built up over the course of decades (Bellman, 1957; Puterman, 1994; Bertsekas & Tsitsiklis, 1996; Kaelbling et al., 1996; Szepesvári, 2010; Sutton & Barto, 2018; François-Lavet et al., 2018). A prototypical method based on value iteration is Q-learning (Watkins & Dayan, 1992); in its most basic form, it maintains an estimate  $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  of the optimal value function, and given a sequence of transition tuples  $(s_t, a_t, r_t, s_{t+1})_{t \geq 0}$ , updates  $Q(s_t, a_t)$  towards

## 1. Introduction

Experience replay is the fundamental data-generating mechanism in off-policy deep reinforcement learning (Lin, 1992). It has been shown to improve sample efficiency and stability by storing a fixed number of the most recently collected transitions for training. However, the *interactions* of experience replay with modern algorithmic components of deep RL agents are still poorly understood. This lack of understanding impedes progress, as researchers are unable to measure the full impact of algorithmic changes without extensive tuning. We therefore present a large-scale study

<sup>\*</sup>Equal contribution <sup>1</sup>Google Brain <sup>2</sup>MILA, Université de Montréal <sup>3</sup>CIFAR Director <sup>4</sup>CIFAR Fellow <sup>5</sup>DeepMind. Correspondence to: William Fedus <liamfedus@google.com>.

arXiv:2007.06700  
 Analysis of ER focusing on:  
 - replay capacity  
 - replay ratio  
 Greater replay capacity helps some algos., does not change other algos.  
 n-step returns are uniquely beneficial for large memory  
 Q. What does it mean by "uncorrected" n-step returns?

Q. Def'n of upend?

the target  $r_t + \gamma \max_{a \in A} Q(s_{t+1}, a)$ , for each  $t \geq 0$ .

## 2.1. Deep reinforcement learning

In recent years, the field of *deep reinforcement learning* has sought to combine the classical reinforcement learning algorithms mentioned above with modern techniques in machine learning to obtain scalable learning algorithms. Deep Q-Networks (DQN) (Mnih et al., 2015) combine Q-learning with neural network function approximation and experience replay (Lin, 1992) to yield a scalable reinforcement learning algorithm that achieves superhuman performance on a range of games in the Arcade Learning Environment (Bellemare et al., 2013). Many further approaches have developed since, which like DQN can be understood as comprising *three fundamental units*:

- (i) a function approximation architecture;
- (ii) a learning algorithm;
- (iii) a mechanism for generating training data.

A range of innovations in all three areas have been developed since the introduction of the original DQN algorithm. A limited selection of these include architectures based on duelling heads (Wang et al., 2015) and various forms of recurrence (Hausknecht & Stone, 2015; Kapturowski et al., 2019), learning algorithms using auxiliary tasks (Jaderberg et al., 2016; Fedus et al., 2019), distributional variants of RL (Bellemare et al., 2017; Dabney et al., 2018), and the use of *prioritisation* in sampling from experience replay (Schaul et al., 2015).

A notable agent combining several such innovations is *Rainbow* (Hessel et al., 2018). An open-source implementation based on this agent is available in *Dopamine* (Castro et al., 2018), which has four main differences relative to the original DQN agent:

- **Prioritized Experience Replay (PER)** (Schaul et al., 2015): A scheme for sampling non-uniformly from the replay buffer that favors transitions with a high temporal-difference (TD) error. In contrast, DQN uniformly samples experience from the replay buffer.
- **$n$ -step returns**: Rather than training the action-value estimate  $Q(s_t, a_t)$  on the basis of the single-step temporal difference error  $r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$ , an  $n$ -step target  $\sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \max_a Q(s_{t+n}, a) - Q(s_t, a_t)$  is used, with intermediate actions generated according to a behavior policy  $\mu$ .
- **Adam optimizer** (Kingma & Ba, 2014): An improved first-order gradient optimizer which normalizes for first and second gradient moments, in contrast to the RMSProp optimizer used by DQN.
- **C51** (Bellemare et al., 2017): A distributional RL algorithm that trains an agent to make a series of predictions about the *distribution* of possible future returns, rather

than solely estimating the scalar expected return.

The Dopamine Rainbow agent differs from that of Hessel et al. (2018) by not including Double DQN updates (Van Hasselt et al., 2016), dueling heads (Wang et al., 2015), or noisy networks (Fortunato et al., 2018). For completeness, we provide a discussion of the details around these *algorithmic adjustments in the Appendix*.

## 2.2. Experience replay

A critical component of DQN-style algorithms is *experience replay* (Lin, 1992). The experience replay is a fixed-size buffer that holds the most recent transitions collected by the policy. It greatly improves the sample efficiency of the algorithm by enabling data to be reused multiple times for training, instead of throwing away data immediately after collection, and also improves the stability of the network during training.

The experience replay is typically implemented as a *circular buffer*, where the oldest transition in the buffer is removed to make room for a transition that was just collected. Transitions are sampled from the buffer at fixed intervals for use in training. The most basic *sampling strategy* used is uniform sampling, whereby each transition in the buffer is sampled with equal probability. Other sampling strategies, such as prioritized experience replay (Schaul et al., 2015), can be used instead of uniform sampling. While this is the most common implementation, other variations, such as a distributed experience replay buffer (Horgan et al., 2018), can be used.

Mnih et al. (2015) set the experience replay size to hold 1M transitions. This setting is often preserved in works building off DQN (Hessel et al., 2018). In this work, we hold other components of the algorithm fixed and study the effects of modifying various aspects of experience replay.

## 2.3. Related work

While the three principal aspects of DQN-based agents listed in Section 2.1 have individually received much attention, comparatively little effort has been spent on investigating the *interactions between design choices* across these areas; notable examples include the original Rainbow work (Hessel et al., 2018), as well as more recent work focused on  $\lambda$ -returns and replay (Daley & Amato, 2019). The principal aim of this work is to improve our understanding of the relationship between data generation mechanisms, in the form of experience replay, and learning algorithms.

Zhang & Sutton (2017) study the effects of replay buffer size on performance of agents of varying complexity, noting hyperparameters of the replay buffer such as this are not well understood generally, partly as a result of the complexities of modern learning systems. They find that both smaller

and larger replay buffers are detrimental to performance on a set three tasks: a gridworld, LUNAR LANDER (Catto, 2011) and PONG from the Arcade Learning Environment (Bellemare et al., 2013). Liu & Zou (2018) also study the effects of replay buffer size and minibatch size on learning performance. Fu et al. (2019) report that agent performance is sensitive to the number of environment steps taken per gradient step, with too small or large a ratio also hindering performance. van Hasselt et al. (2019) vary this ratio in combination with batch sizes to obtain a more sample-efficient version of Rainbow. Beyond direct manipulation of these properties of the buffer, improving and understanding experience replay algorithms remains an active area of research (Pan et al., 2018; Schlegel et al., 2019; Zha et al., 2019; Novati & Koumoutsakos, 2019; Sun et al., 2020; Lee et al., 2019).

### 3. Disentangling experience replay

We conduct a detailed study of the ways in which the type of data present in replay affects learning. In earlier work, Zhang & Sutton (2017) studied the effects of increasing the size of the replay buffer of DQN. We note that in addition to increasing the diversity of samples available to the agent at any moment, a larger replay buffer also typically contains more off-policy data, since data from older policies remain in the buffer for longer. The behavior of agents as we vary the rate at which data enters and leaves the buffer is another factor of variation which is desirable to understand, as this is commonly exploited in distributed agents such as R2D2 (Kapturowski et al., 2019). Our aim will be to disentangle, as far as is possible, these separate modalities. To make these ideas precise, we begin by introducing several formal definitions for the properties of replay that we may wish to isolate, in order to get a better understanding behind the interaction of replay and learning.

#### 3.1. Independent factors of control

We first disentangle two properties affected when modifying the buffer size.

**Definition 1.** The replay capacity is the total number of transitions stored in the buffer.

By definition, the replay capacity is increased when the buffer size is increased. A larger replay capacity will typically result in a larger state-action coverage. For example, an  $\epsilon$ -greedy policy samples actions randomly with probability  $\epsilon$ , so the total number of random actions in the replay buffer with capacity  $N$  will be  $\epsilon N$  in expectation.

**Definition 2.** The age of a transition stored in replay is defined to be the number of gradient steps taken by the learner since the transition was generated. The age of the oldest policy represented in a replay buffer is the age of the

		Replay Capacity				
		100,000	316,228	1,000,000	3,162,278	10,000,000
Oldest Policy	25,000,000	250.000	79.057	25.000	7.906	2.500
	2,500,000	25.000	7.906	2.500	0.791	0.250
	250,000	2.500	0.791	0.250	0.079	0.025
	25,000	0.250	0.079	0.025	0.008	0.003

Figure 1. Replay ratio varies with replay capacity and the age of the oldest policy. The replay ratio for controlling different replay capacities (rows) and different ages of the oldest policy (columns). Bold values of 0.25 are the default replay ratio (one gradient update per four actions) used by Mnih et al. (2015).

Q. Can you directly control the age of the oldest policy?

oldest transition in the buffer.

The buffer size directly affects the age of the oldest policy. This quantity can be loosely viewed as a proxy of the degree of off-policyness of transitions in the buffer; intuitively, the older a policy is, the more likely it is to be different from the current policy. However, note that this intuition does not hold for all cases; e.g., if the acting policy cycles through a small set of policies.

Buffer size affects age of oldest policy

Age of oldest policy is a proxy to off-policyness of transitions

(Exception: policy space is small)

Whenever the replay buffer size is increased, both the replay capacity and the age of the oldest policy increase, and the relationship between these two independent factors can be captured by another quantity, the replay ratio.

If capacity increases, age of oldest policy increases

**Definition 3.** The replay ratio is the number of gradient updates per environment transition.

The replay ratio can be viewed as a measure of the relative frequency the agent is learning on existing data versus acquiring new experience. The replay ratio stays constant when the buffer size is increased because the replay capacity and the age of the oldest policy both increase. However, if one of the two factors is independently modulated, the replay ratio will change. In particular, when the oldest policy is held fixed, increasing the replay capacity requires more transitions per policy, which decreases the replay ratio. When the replay capacity is held fixed, decreasing the age of the oldest policy requires more transitions per policy, which also decreases the replay ratio.

This sentence doesn't makes sense

To fix one and change another, we need to change # transitions per policy

# transitions per policy is measured with "replay ratio"

In the hyperparameters established in (Mnih et al., 2015), the policy is updated every 4 environment steps collected, resulting in a replay ratio of 0.25. Therefore, for a replay capacity of 1M transitions, the oldest policy captured in the replay buffer is 250k gradient updates old. Figure 1 computes the resultant replay ratio when varying either the replay capacity or the age of the oldest policy. Quantities similar to the replay ratio have also been identified as important hyperparameters in several recent works on deep RL methods (Wang et al., 2016; Kapturowski et al., 2019; Hamrick et al., 2020; Lin & Zhou, 2020).

Q. What are these similar quantities? How are they different? Why did the authors not use these instead?

Goal: Disentangle various effects of ER

"Replay capacity" = # transitions in buffer

"Age of transition" = # gradient steps since transition was generated

"Age of oldest policy in buffer" = age of oldest transition

Better →

↓ Better

		Replay Capacity				
		100,000	316,228	1,000,000	3,162,278	10,000,000
Oldest Policy	25,000,000	-74.9	-76.6	-77.4	-72.1	-54.6
	2,500,000	-78.1	-73.9	-56.8	-16.7	28.7
	250,000	-70.0	-57.4	0.0	13.0	18.3
	25,000	-31.9	12.4	16.9	--	--

Figure 2. Performance consistently improves with increased replay capacity and generally improves with reducing the age of the oldest policy. Median percentage improvement over the Rainbow baseline when varying the replay capacity and age of the oldest policy in Rainbow on a 14 game subset of Atari. We do not run the two cells in the bottom-right because they are extremely expensive due to the need to collect a large number of transitions per policy.

### 3.2. Experiments

Environment: 14 games from ALE with sticky actions

We conduct experiments on the commonly-used Atari Arcade Learning Environment (Bellemare et al., 2013) with sticky actions (Machado et al., 2018). We focus on a subset of 14 games in order to reduce the computational burden of our experiments, as we aim to benchmark a sizeable grid of values for the two factors. The subset is chosen in a manner meant to reflect the diversity of environments across games (e.g., ensuring we have sparse, hard exploration games such as MONTEZUMA’S REVENGE). For each game, we run 3 different random seeds. We use Rainbow (Hessel et al., 2018) implemented in Dopamine (Castro et al., 2018) as our base algorithm in this study due to its strong performance among existing Q-learning agents.

Q. How did the

Agent: Rainbow (Dopamine)

Agent trained with different settings (capacity, age of oldest buffer)

In these experiments, we fix the total number of gradient updates and the batch size per gradient update to the settings used by Rainbow, meaning that all agents train on the same number of total transitions, although environment frames generated may vary due to controlling for the oldest policy in replay. Rainbow uses a replay capacity of 1M and an oldest policy of 250k, corresponding to a replay ratio of 0.25. We assess the cross product of 5 settings of the replay capacity (from 0.1M to 10M) and 4 settings of the oldest policy (from 25k to 25M), but exclude the two settings with the lowest replay ratio as they are computationally impractical due to requiring a large number of transitions per policy. The replay ratio of each setting is shown in Figure 1 and the Rainbow results in Figure 2. Several trends are apparent.

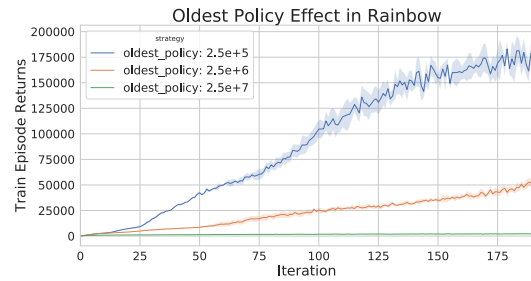
#### Increasing replay capacity improves performance.

While fixing the oldest policy, performance improves with higher replay capacity (rows in Figure 2). This general trend holds regardless of the particular value of oldest policy, though the magnitude of improvement is dependent on the setting of oldest policy. It may be that larger replay capacities improves the value function estimates due to having

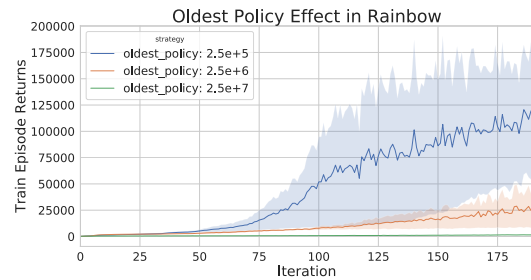
a larger state-action coverage, which can lower the chance of overfitting to a small subset of state-actions.

#### Reducing the oldest policy improves performance.

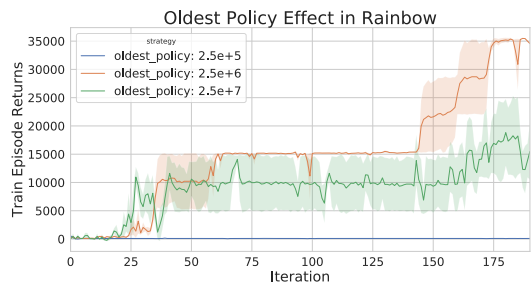
When fixing the replay capacity, performance tends to improve as the age of the oldest policy decreases (columns in Figure 2). We visualize the training curves for three settings on three games in Figure 3. Using the heuristic that the age of oldest policy is a proxy for off-policy-ness, this result suggests that learning from more on-policy data may improve performance. As the agent improves over the course of training, it spends more time in higher quality (as measured by return) regions of the environment. Learning to better estimate the returns of high quality regions can lead to further gains.



(a) ASTERIX



(b) SEAQUEST



(c) PRIVATEEYE

#### Performance generally improves when trained on data from more recent policies.

Training curves for three games each over a sweep of three oldest policy parameters (2.5e5, 2.5e6 and 2.5e7). Performance generally improves significantly with reduced oldest policies except in sparse-reward games such as PRIVATEEYE.

However, an exception to this trend is seen in the 10M replay capacity setting where the performance drops when moving from an age of 2.5M to 250k. This aberration is explained by a drop in scores of two specific games (see Figure 4), MONTEZUMA’S REVENGE and PRIVATEEYE, which are considered sparse-reward hard-exploration environments (Bellemare et al., 2016). Agents that only sample from the newest policies do not seem to be able to find the sparse reward (see Figure 3(c)).

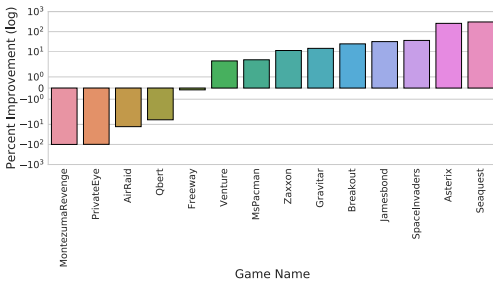


Figure 4. Sparse-reward games benefit from data generated by older policies. Median relative improvement of a Rainbow agent with a 10M replay capacity and 250k oldest policy compared to one with 2.5M oldest policy. Decreasing the age of the oldest policy improves performance on most games. However, performance drops significantly on the two hard exploration games, which bucks the trend that data from newer policies is better.

*How is this offset from point 4?*

**Increasing buffer size with a fixed replay ratio has varying improvements.** When the replay ratio is fixed while the buffer size is modified, there is an interplay between the improvements caused by increasing the replay capacity and the deterioration caused by having older policies in the buffer. The magnitude of both effects depends on the particular settings of these quantities. Generally, as the age of the oldest policy increases, the benefits from increasing the replay capacity are not as large.

### 3.3. Generalizing to other agents

The experiments in the previous subsection are conducted using the Dopamine Rainbow algorithm but we now test whether experience replay behaves similarly in other Q-learning variants. In particular, we test if increases in the replay capacity improve performance with the original DQN algorithm (Mnih et al., 2015).

We maintain the default Dopamine hyperparameters (Castro et al., 2018) specifically tuned for DQN and increase the replay capacity from 1M to 10M. We consider two experimental variants: fixing either the replay ratio or the oldest policy. Fixing the replay ratio corresponds to the standard setting of increasing the buffer size hyperparameter. Fixing the oldest policy requires adjusting the replay ratio so that the replay buffer always contains policies within a certain

number of gradient updates. The results are presented in Table 1.

Table 1. DQN does not improve with larger replay capacities, unlike Rainbow. Relative improvements of increasing replay capacity from 1M to 10M for DQN and Rainbow. Relative improvements are computed with respect to performance of the corresponding agent with a 1M replay capacity. Either the replay ratio or the oldest policy is held fixed when increasing the replay capacity.

Agent	Fixed replay ratio	Fixed oldest policy
DQN	+0.1%	-0.4%
Rainbow	+28.7%	+18.3%

Surprisingly, providing a DQN agent with an order of magnitude larger memory confers no benefit regardless of whether the replay ratio or the oldest policy is held fixed. These results stand in contrast to the dynamics of the Rainbow agent which demonstrates consistent improvements with increased replay capacity. We also note the fixed replay ratio result disagrees with the conclusion in Zhang & Sutton (2017) that larger replay capacity is detrimental – we instead observe no material performance change.

This result calls into question which differences between these two value-based algorithms are driving the distinct responses to an increased replay buffer size. In the next section, we perform a large scale study to determine which algorithmic components enable Rainbow to take advantage of a larger replay capacity.

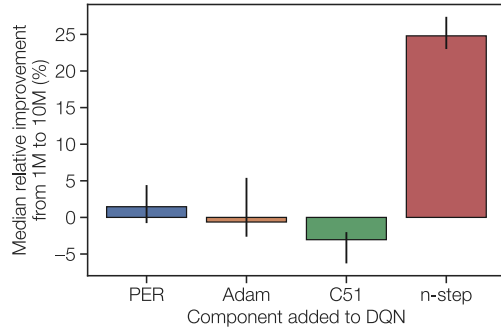
## 4. What components enable improving with a larger replay capacity?

As described in Section 2.1, the Dopamine Rainbow agent is a DQN agent with four additional components: prioritized experience replay (Schaul et al., 2015),  $n$ -step returns, Adam (Kingma & Ba, 2014), and C51 (Bellemare et al., 2017). We therefore seek to attribute the performance difference under larger replay capacities to one or more of these four components. To do so, we study agents built from a variety of subsets of components, and measure whether these variant agents improve when increasing the replay capacity. In these studies we specifically measure the relative improvement upon increasing the replay capacity, not which variant achieves the highest absolute return.

### 4.1. Additive and ablation experiments

We begin with an additive study where we add a single component of Rainbow to the DQN algorithm, resulting in four agent variations. We independently compute the relative performance difference when increasing the replay capacity from 1M to 10M for each variant. When increasing

the replay capacity, we fix the replay ratio, and revisit the case of fixing the oldest policy later in the section. We evaluate across a set of 20 games that is a superset of the 14 games used in the previous section. The results are shown in Figure 5.

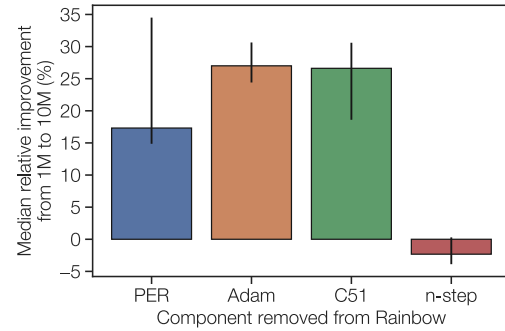


**Figure 5. Adding  $n$ -step to DQN enables improvements with larger replay capacities.** Median relative improvement of DQN additive variants when increasing replay capacity from 1M to 10M. Bars represent 50% percentile improvement and the lower and upper bound of the error line is denoted by 25% and 75% percentiles, respectively.

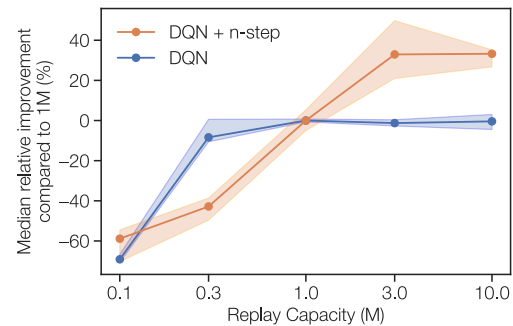
The only additive variant that materially improves with larger replay capacity is the DQN agent with  $n$ -step returns. From this we hypothesize that  $n$ -step returns uniquely confer a benefit with larger replay capacity. As a test of this hypothesis, removing the  $n$ -step returns from the Rainbow agent should inhibit this ablative variant from improving with a larger replay capacity. Furthermore, for  $n$ -step returns to be the *sole* influential component, the ablative versions of the other three components (PER, Adam, C51) must still show improvements with a larger replay capacity. We present the result of this ablative Rainbow experiment in Figure 6.

As predicted, a Rainbow agent stripped of  $n$ -step returns does not benefit with larger replay capacity, while the Rainbow agents stripped of other components still improve. These results suggest that  $n$ -step returns are *uniquely* important in determining whether a Q-learning algorithm can improve with a larger replay capacity. Another surprising finding is that prioritized experience replay does not significantly affect the performance of agents with larger memories; intuitively, one might expect prioritized experience replay to be useful in selecting relevant experience for the learner as the replay buffer size grows. Further detail at the per-game level is provided in Appendix B.

As one final control, we check that DQN with  $n$ -step returns still improves with larger replay capacity if the oldest policy is held fixed, rather than the replay ratio being held fixed. Figure 7 shows that the DQN +  $n$ -step algorithm is able to



**Figure 6. Removing  $n$ -step from Rainbow prevents improvements with larger replay capacities.** Median relative improvement of Rainbow ablative variants when increasing replay capacity from 1M to 10M. Bars represent 50% percentile improvement and the lower and upper bound of the error line is denoted by 25% and 75% percentiles, respectively.



**Figure 7. DQN +  $n$ -step improve with larger replay capacity if oldest policy is fixed.** 25<sup>th</sup>, 50<sup>th</sup>, and 75<sup>th</sup> percentile relative improvement of DQN and DQN +  $n$ -step when increasing replay capacity while the oldest policy is fixed at 250k.

consistently improve when increasing the replay capacity from the highly tuned default of 1M while the standard DQN does not. When given less data, DQN with  $n$ -step can perform worse, an observation that we revisit in Section 5.2.

Taken together, these results suggest that  $n$ -step is a critical factor for taking advantage of larger replay sizes. This is unexpected. Uncorrected  $n$ -step returns are not theoretically justified for off-policy data because they do not correct for differences between the behavior and target policies, but they are still used due to convenience and their empirical benefits. The experiments show that extending this theoretically unprincipled approach into a regime where issues may be further exacerbated<sup>1</sup> is essential for performance.

<sup>1</sup>Larger buffers under a fixed replay ratio will contain data from older policies which potentially increases the discrepancy between the old behavior and current agent.

## 4.2. $n$ -step for massive replay capacities

These results suggest that  $n$ -step returns should be used when increasing replay capacity. However, our previous results only consider replay capacities up to 10M, which is a fraction of the 200M total transitions collected over the entire course of training. It may be the case that  $n$ -step is no longer as beneficial, or even harmful, as the replay capacity increases. This degradation may happen because when fixing the replay ratio, which is the most common setting used in practice, the age of the oldest policy will increase alongside the replay capacity. As demonstrated in Section 3.2, the exact settings of each factor controls the magnitude of degradation caused by an increase in oldest policy and the magnitude of improvement caused by an increase in replay capacity. Furthermore, the uncorrected nature of  $n$ -step returns may hurt performance in regimes of high off-policyness.

Therefore to test the limits of the hypothesis that  $n$ -step returns are useful in large capacity regimes with high levels of off-policyness, we turn to the logical extreme — offline deep reinforcement learning (Agarwal et al., 2020). In offline reinforcement learning, a learner is trained only using data collected from another agent. All data from the original agent is preserved unlike the typical case in online reinforcement learning where older experience is evicted from the replay buffer. This also represents a worst-case scenario with respect to off-policyness because the learner cannot interact with the environment to correct its estimates. We use the settings of Agarwal et al. (2020) where for each game in the same subset of games used in previous experiments, a DQN agent collects a data set of 200M frames which are used to train another agent. We train two variants of DQN with  $n$ -step returns, and compare each setting of  $n$  against the online DQN agent used to generate the data. The results are presented in Figure 8.

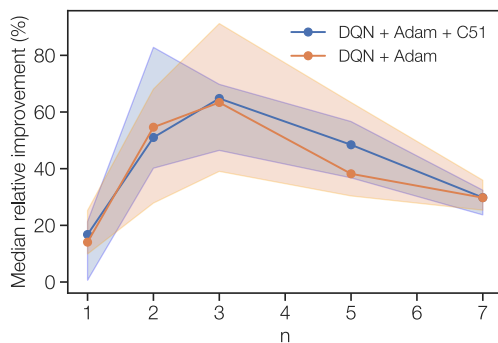


Figure 8. Agents improve with  $n$ -step in the offline batch RL setting. 25<sup>th</sup>, 50<sup>th</sup>, and 75<sup>th</sup> percentile relative improvement of each offline agent over the DQN used to gather the training data. Using  $n > 1$  improves performance.

Even in this challenging task, using  $n > 1$  consistently improves performance for both agents. The shape of the curve when varying  $n$  depends on the particular agent that is used, but setting  $n = 3$ , which is the value used in all previous experiments, performs well. These results further validate the hypothesis that  $n$ -step is beneficial when increasing the replay capacity.

What  $n$  to use?

## 5. Why is $n$ -step the enabling factor?

In the previous section, we showed empirically that  $n$ -step returns modulates whether DQN can take advantage of larger replay capacities. In this section, we attempt to uncover the mechanism that links these two seemingly unrelated components together. In the hypotheses that we evaluate, we find that one plays a partial role in the linkage between  $n$ -step and replay capacity.

### 5.1. Deadening the deadly triad

Function approximation of Q-values, bootstrapping, and off-policy learning have been identified as the deadly triad (Sutton & Barto, 2018; van Hasselt et al., 2018) of properties that, when combined, can negatively affect learning or even cause divergence. van Hasselt et al. (2018) suggest that  $n$ -step returns work well because they make the magnitude of the bootstrap smaller, making divergence less likely. Recall that the  $n$ -step target is  $\sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \max_a Q(s_{t+n}, a)$  where  $\gamma \in [0, 1]$  is the discount factor and  $\gamma^n$  is the contraction factor. The smaller the contraction factor, the less impact the bootstrap  $\max_a Q(s_{t+n}, a)$  has on the target.

When the replay capacity is increased while keeping the replay ratio fixed, the transitions in the buffer come from older policies, which may increase the off-policyness of the data and, according to the deadly triad, destabilize training. Thus, one may hypothesize that the supposed stability offered by  $n$ -step is required to counter the increased off-policyness produced by a larger replay capacity.

We test this hypothesis by applying a standard 1-step update to DQN with the same contractive factor as an  $n$ -step update:  $r_t + \gamma^n \max_a Q(s_{t+1}, a)$ ; this is equivalent to simply reducing the discount factor, although we note that it also changes the fixed point of the algorithm. If the contractive factor is the key enabler, using DQN with the modified update should be able to improve with increased replay capacity. However, we find empirically that there is no improvement with an increased replay capacity when using the smaller contractive factor in a 1-step update. Furthermore, even if the oldest policy is fixed, which should control off-policyness, DQN does not improve with a larger capacity (see Figure 7). These results suggests that the hypothesis that the stability improvements of  $n$ -step that arise from the lower contrac-

tion rate do not explain the importance of  $n$ -step in taking advantage of larger replay capacities.

## 5.2. Variance reduction

One can view  $n$ -step returns as interpolating between estimating Monte Carlo (MC) targets,  $\sum_{k=0}^T \gamma^k r_{t+k}$ , and single-step temporal difference (TD) targets,  $r_t + \gamma \max_a Q(s_{t+1}, a)$ . It balances between the low bias but high variance of MC targets, and the low variance but high bias of single-step TD targets. The variance of MC targets comes from stochasticity of rewards and environmental dynamics, whereas the bias of single-step TD targets comes from using an imperfect bootstrap to estimate future returns.

An increase in replay capacity might provide a means of mitigating the additional variance of  $n$ -step returns, relative to single-step TD targets. The increased variance of the  $n$ -step target increases the learning algorithm’s sensitivity to changes in the replay buffer data. Whilst an increased replay capacity will not affect the variance of the learner’s target ascribable to minibatch sampling from a fixed replay buffer, it will affect the diversity of transitions which the buffer contains. Thus, in scenarios where the data-generating policy is rapidly changing, for example, a small replay buffer may undergo wild shifts in the type of data it contains, which may have a particularly pronounced effect on higher variance  $n$ -step methods. In contrast, a larger buffer may moderate the effects of fluctuations in the data-generating policy.

This brief analysis provides a testable hypothesis: in an environment with less variance in returns, the gains from increasing the replay capacity should be reduced. The variance of returns in the Atari domain can be reduced by turning off *sticky actions*. Sticky actions (Machado et al., 2018) cause the previously taken action to be repeated with some probability – increasing the stochasticity of the transition dynamics – which in turn increases the variance of returns.

We test this hypothesis by running 1-,3-,5- and 7-step versions of DQN on the ALE with and without sticky actions, and report results in Figure 9. As predicted by the hypothesis, the relative improvements in the absence of sticky actions are consistently less than than the relative improvements with sticky actions present. Furthermore, the difference of improvements between sticky actions and no sticky actions increases with  $n$ , which is predicted by the hypothesis given that variance also increases with  $n$ . However, even when removing stochasticity, using  $n$ -step returns still shows improvements with increased capacity, indicating that whilst there is some evidence for the hypothesis presented here, it can only play a partial part in explaining the effectiveness of  $n$ -step updates and their ability to make use of larger buffers.

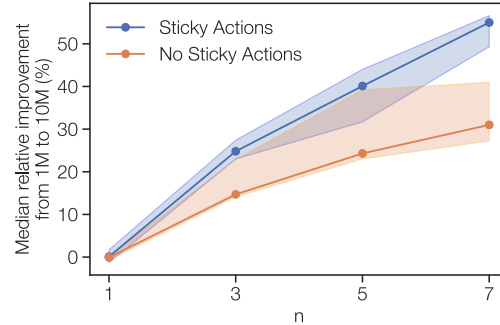


Figure 9. Turning off sticky actions reduces gains from increasing replay capacity. 25<sup>th</sup>, 50<sup>th</sup>, and 75<sup>th</sup> percentile relative improvements from increasing the replay capacity from 1M to 10M for DQN agents with  $n$ -step when toggling sticky actions. Larger  $n$  benefit more from increased replay capacity, but agents trained in an environment without sticky actions benefit less than agents trained in an environment with sticky actions.

## 5.3. Further multi-step and off-policy methods

Our investigation has focused specifically on the effects of  $n$ -step returns, as one of the key aspects of the Rainbow agent. These findings naturally open further questions as to the interaction between experience replay and more general classes of return estimators based on multi-step, off-policy data, such as variants of  $Q(\lambda)$  (Watkins, 1989; Peng & Williams, 1994; Sutton et al., 2014; Harutyunyan et al., 2016), TreeBackup (Precup et al., 2000) and Retrace (Munos et al., 2016), which we believe will be interesting topics for future work.

## 6. Discussion

We have conducted an in-depth study of how replay affects performance in value-based deep reinforcement learning agents. The summary of our contributions are:

- Disentangling the effects of *replay capacity* and *oldest policy*, finding that increasing replay capacity and decreasing the age of the oldest policy improves performance;
- Discovering that  $n$ -step returns are uniquely critical for taking advantage of an increased replay capacity;
- Benchmarking  $n$ -step returns in the massive replay capacity regime, and finding that it still provides gains despite the substantial off-policyness of the data;
- Investigating the connection between  $n$ -step returns and experience replay, and finding that increasing the replay capacity can help mitigate the variance of  $n$ -step targets, which partially explains the improved performance.

Taking a step back, this can be interpreted as an investigation into how two of the principal aspects of deep RL

agents described in Section 2.1, namely learning algorithms and data generating mechanisms, interact with one another. These two aspects are inextricably linked; the data on which an algorithm is trained clearly affects what is learned, and correspondingly what is learned affects how the agent interacts with the environment, and thus what data is generated. We highlight several fundamental properties of the data generating distribution: (a) Degree of on-policy-ness (how close is the data-generating distribution to the current policy being evaluated?); (b) State-space coverage; (c) Correlation between transitions; (d) Cardinality of distribution support.

Practically, these aspects may be difficult to control independently, and the typical algorithmic adjustments we can make affect several of these simultaneously; two examples of such adjustments are the replay capacity and replay ratio investigated in this paper. We emphasize that these practically controllable aspects of an agent may also have differing effects on the data distribution itself depending on the precise architecture of the agent; for example, in a distributed agent such as R2D2 (Kapturowski et al., 2019), decreasing the replay ratio by increasing the number of actors will lead to changes in both (a) and (c) above, whilst in a single-actor agent such as DQN, changing the replay ratio by altering the number of environment steps per gradient step will also change (b).

These issues highlight the *entanglement* that exists between these different properties of the data-generating mechanism at the level of practical algorithmic adjustments, and motivates further study into how these properties can be disentangled. This direction of research is particularly important with regard to obtaining agents which can effortlessly scale with increased availability of data. More broadly, this work opens up many questions about the interaction of replay and other agent components, the importance of  $n$ -step returns in deep RL, and off-policy learning, which we expect to be interesting subjects for future work.

## Acknowledgements

We’d like to thank Carles Gelada and Jacob Buckman for many lively discussions trying to understand early empirical results. In addition, we thank Dale Schuurmans for theoretical insights and Sylvain Gelly for advice on conducting a hard-nosed scientific study. We had several helpful discussions with the Google Brain RL team, in particular, Dibya Ghosh and Marlos Machado. Finally, we would like to thank Georg Ostrovski for extensive comments on an earlier draft of this work.

## References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. *In-*

*ternational Conference on Machine Learning (ICML)*, 2020.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Neural Information Processing Systems (NIPS)*, 2016.

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*, 47:253–279, 2013.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Bellman, R. *Dynamic Programming*. Dover Publications, 1957.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996. ISBN 1886529108.

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

Catto, E. Box2D: A 2D physics engine for games, 2011.

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

Daley, B. and Amato, C. Reconciling  $\lambda$ -returns with experience replay. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Fedus, W., Gelada, C., Bengio, Y., Bellemare, M. G., and Larochelle, H. Hyperbolic discounting and learning over multiple horizons. *arXiv*, 2019.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018.

Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep Q-learning algorithms. In *International Conference on Machine Learning (ICML)*, 2019.

- Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Pfaff, T., Weber, T., Buesing, L., and Battaglia, P. W. Combining Q-learning and search with amortized value estimates. In *International Conference on Learning Representations (ICLR)*, 2020.
- Harutyunyan, A., Bellemare, M. G., Stepleton, T., and Munos, R.  $Q(\lambda)$  with off-policy corrections. In *International Conference on Algorithmic Learning Theory (ALT)*, 2016.
- Hausknecht, M. and Stone, P. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium Series*, 2015.
- Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.
- Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2018.
- Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2016.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
- Kapturowski, S., Ostrovski, G., Dabney, W., Quan, J., and Munos, R. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.
- Lee, S. Y., Sungik, C., and Chung, S.-Y. Sample-efficient deep reinforcement learning via episodic backward update. In *Neural Information Processing Systems (NeurIPS)*, 2019.
- Lin, K. and Zhou, J. Ranking policy gradient. In *International Conference on Learning Representations (ICLR)*, 2020.
- Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
- Liu, R. and Zou, J. The effects of memory replay in reinforcement learning. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 478–485. IEEE, 2018.
- 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*, 61:523–562, 2018.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv*, 2013.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.
- Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2016.
- Novati, G. and Koumoutsakos, P. Remember and forget for experience replay. In *Remember and Forget for Experience Replay*, 2019.
- 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. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.
- Peng, J. and Williams, R. J. Incremental multi-step q-learning. In *International Conference on Machine Learning (ICML)*, 1994.
- Precup, D., Sutton, R. S., and Singh, S. P. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning (ICML)*, 2000.
- Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.
- Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2015.
- Schlegel, M., Chung, W., Graves, D., Qian, J., and White, M. Importance resampling for off-policy prediction. In *Neural Information Processing Systems (NeurIPS)*, 2019.
- Sun, P., Zhou, W., and Li, H. Attentive experience replay. In *AAAI Conference on Artificial Intelligence*, 2020.

- Sutton, R., Mahmood, A. R., Precup, D., and Hasselt, H. A new  $q(\lambda)$  with interim forward view and monte carlo equivalence. In *International Conference on Machine Learning (ICML)*, 2014.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.
- Szepesvári, C. *Algorithms for reinforcement learning*. Morgan & Claypool Publishers, 2010.
- Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double Q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.
- van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. In *Deep Reinforcement Learning Workshop, NeurIPS*, 2018.
- van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? In *Neural Information Processing Systems (NeurIPS)*, 2019.
- Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2015.
- Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.
- Watkins, C. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.
- Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- Zha, D., Lai, K.-H., Zhou, K., and Hu, X. Experience replay optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.
- Zhang, S. and Sutton, R. S. A deeper look at experience replay. *Deep Reinforcement Learning Symposium, NIPS*, 2017.

## APPENDICES: Revisiting Fundamentals of Experience Replay

### A. Experimental details

#### A.1. The Dopamine Rainbow agent

Our empirical investigations in this paper are based on the Dopamine Rainbow agent (Castro et al., 2018). This is an open source implementation of the original agent (Hessel et al., 2018), but makes several simplifying design choices. The original agent augments DQN through the use of (a) a distributional learning objective, (b) multi-step returns, (c) the Adam optimizer, (d) prioritized replay, (e) double Q-learning, (f) duelling architecture, and (g) noisy networks for exploration. The Dopamine Rainbow agent uses just the first four of these adjustments, which were identified as the most important aspects of the agent in the original analysis of Hessel et al. (2018).

#### A.2. Atari 2600 games used

A 14 game subset was used for the grid measuring the effects of varying replay capacity and oldest policy. A 20 game subset, which is comprised of the 14 games used for the grid with 6 additional games, was used for all other experiments.

**14 game subset:** AIR RAID, ASTERIX, BREAKOUT, FREEWAY, GRAVITAR, JAMES BOND, MONTEZUMA'S REVENGE, MS. PACMAN, PRIVATE EYE, Q\*BERT, SEAQUEST, SPACE INVADERS, VENTURE, ZAXXON.

**20 game subset:** The 14 games above in addition to: ASTEROIDS, BOWLING, DEMON ATTACK, PONG, WIZARD OF WOR, YARS' REVENGE.

Why  
these  
14/20?

### B. Additive and ablative studies

#### B.1. DQN additions

We provide game-level granularity on the performance of each supplemented DQN agent in Figure 10.

#### B.2. Rainbow ablations

We provide game-level granularity on the performance of each ablated Rainbow agent in Figure 11.

### C. Error analysis for rainbow grid

We provide an error analysis for each of the elements in Figure 2 (reproduced here as Figure 12) by providing the 25% and 75% percentile improvements for each combination of replay capacity and oldest policy. These results are given in Figure 13.

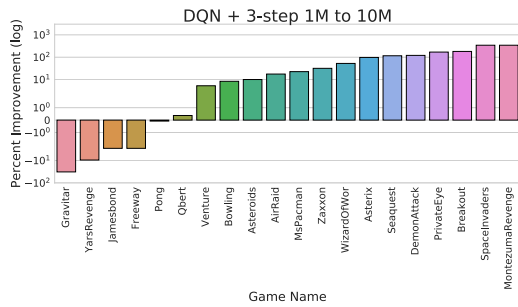
We present an alternative view of the data using a bootstrap estimation technique. Instead of fixing the seeds for both the baseline agent and our new agent at each cell, we sample, with replacement, the seeds. We carry out this procedure repeatedly and report the mean and standard deviations in Figure 14.

### D. Replay buffer size

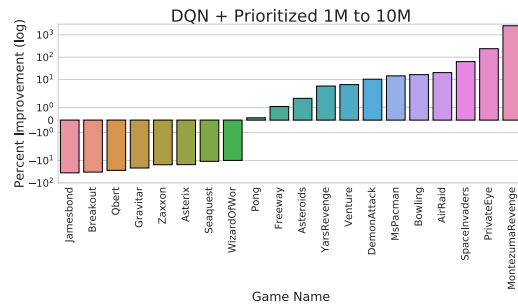
We provide a different perspective on the data from Figure 2 in Figure 15, illustrating a general relationship between replay ratio and performance improvement. We provide game-level granularity on the performance of Rainbow with varying buffer sizes in Figure 17. In Figure 16 we also gives results for varying replay buffer size and age of oldest policy for DQN, 3-step DQN, and Rainbow.

### E. Batch RL learning curves

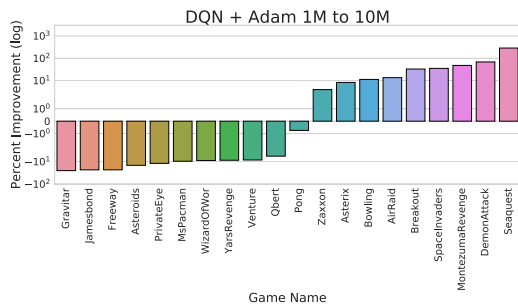
In Figures 18 and 19, we provide learning curves for the batch RL agents described in Section 4.2.



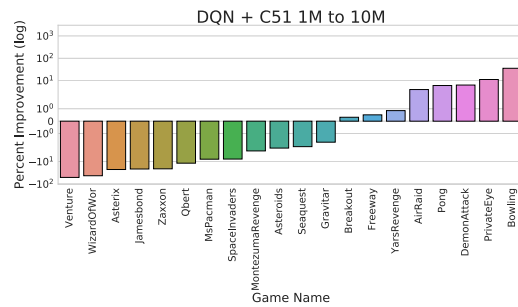
(a) DQN + 3-step provides a median performance change of +24.8%.



(b) DQN + PER provides a median performance change of +1.5%.



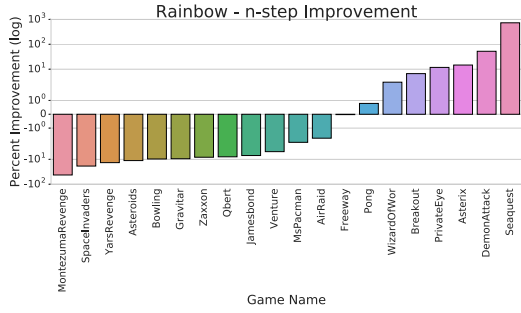
(c) DQN + Adam provides a median performance change of -0.6%.



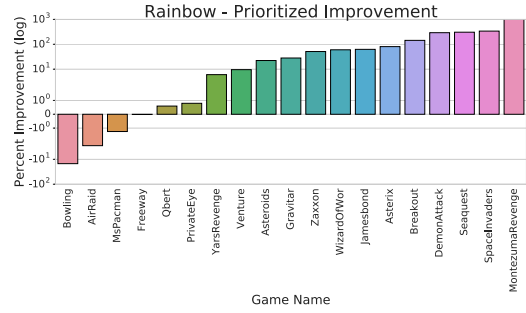
(d) DQN + C51 provides a median performance change of -3.0%.

Figure 10. Only DQN with  $n$ -step improves with increased capacity. DQN with an additional component results at a per-game level, measuring performance changes when increasing replay capacity from 1M to 10M.

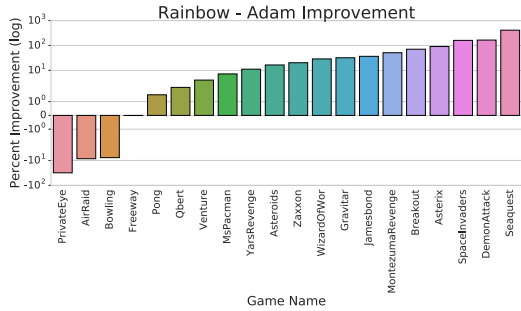
## Revisiting Fundamentals of Experience Replay



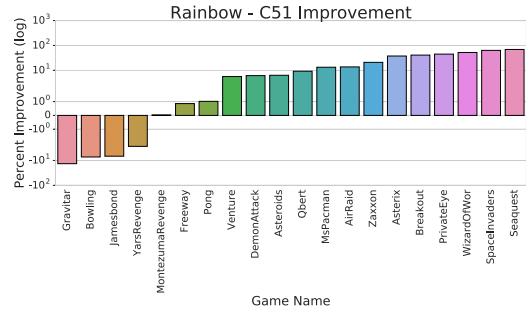
(a) The performance difference for a Rainbow agent without n-step returns when increasing the replay buffer size from 1M to 10M. We find that the resultant agent does not benefit from larger replay buffer sizes, reporting a median performance decrease of 2.3%. This implies the importance of n-step returns.



(b) The performance difference for a Rainbow agent without prioritized experience replay when increasing the replay buffer size from 1M to 10M. Even without prioritization, the algorithm still benefits +17.3%.



(c) A Rainbow agent without Adam optimizer has a median performance increase of +27.0% when the replay buffer size is increased from 1M to 10M.



(d) The performance difference for a Rainbow agent without C51 optimizer when increasing the replay buffer size from 1M to 10M. Median performance change of +26.6%.

Figure 11. Rainbow ablation results at a per-game level.

		Replay Capacity				
		100,000	316,228	1,000,000	3,162,278	10,000,000
Oldest Policy	25,000,000	-74.9	-76.6	-77.4	-72.1	-54.6
	2,500,000	-78.1	-73.9	-56.8	-16.7	28.7
	250,000	-70.0	-57.4	0.0	13.0	18.3
	25,000	-31.9	12.4	16.9	--	--

Figure 12. Performance consistently improves with increased replay capacity and generally improves with reducing the age of the oldest policy. We reproduce the median percentage improvement over the Rainbow baseline when varying the replay capacity and age of the oldest policy in Rainbow on a 14 game subset of Atari.

		Replay Capacity				
		100,000	316,228	1,000,000	3,162,278	10,000,000
Oldest Policy	25,000,000	-78.2	-77.6	-78.8	-76.6	-58.5
	2,500,000	-82.1	-74.2	-64.9	-18.8	26.8
	250,000	-74.6	-58.0	-1.0	8.8	13.2
	25,000	-37.5	5.6	14.5	--	--

		Replay Capacity				
		100,000	316,228	1,000,000	3,162,278	10,000,000
Oldest Policy	25,000,000	-74.1	-76.3	-77.3	-66.6	-53.2
	2,500,000	-77.5	-70.1	-56.5	-15.2	29.5
	250,000	-67.0	-54.8	1.1	16.2	20.3
	25,000	-26.2	18.7	18.6	--	--

Figure 13. 25% (left) and 75% (right) percentile performance improvement over the Rainbow baseline when the replay capacity and age of the oldest policy in Rainbow on a 14 game subset of Atari.

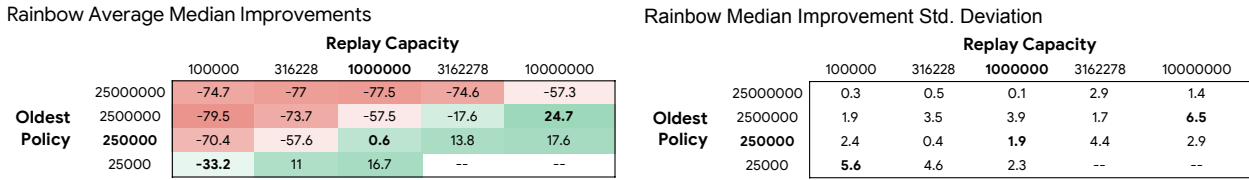


Figure 14. **Bootstrap estimate of variance for each cell.** For each cell, rather than using the same 3 seeds for the baseline and the same 3 seeds for each agent of each cell, we consider sampling seeds with replacement. The left grid shows the mean median improvement and the right grid shows the standard deviation of the median improvement.

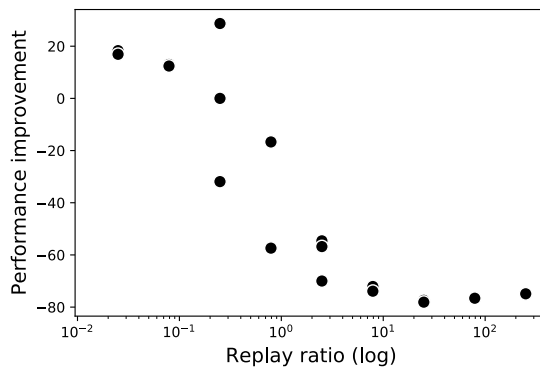
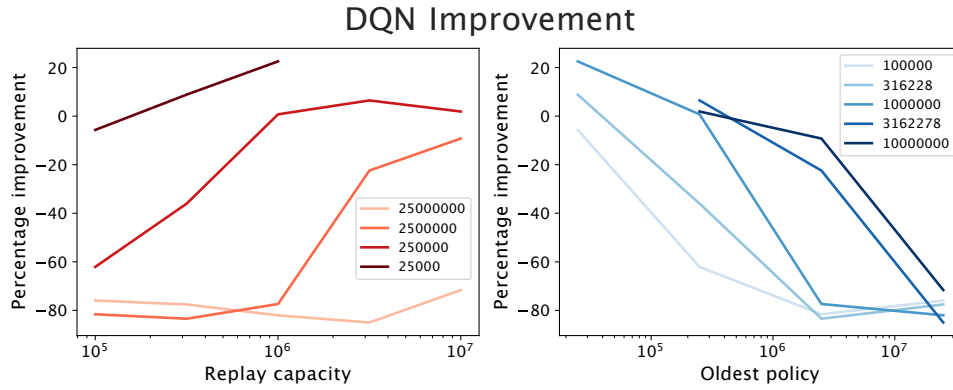
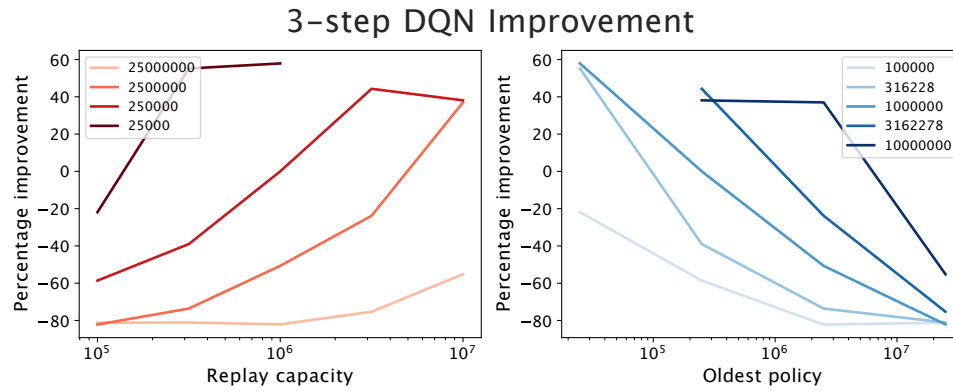


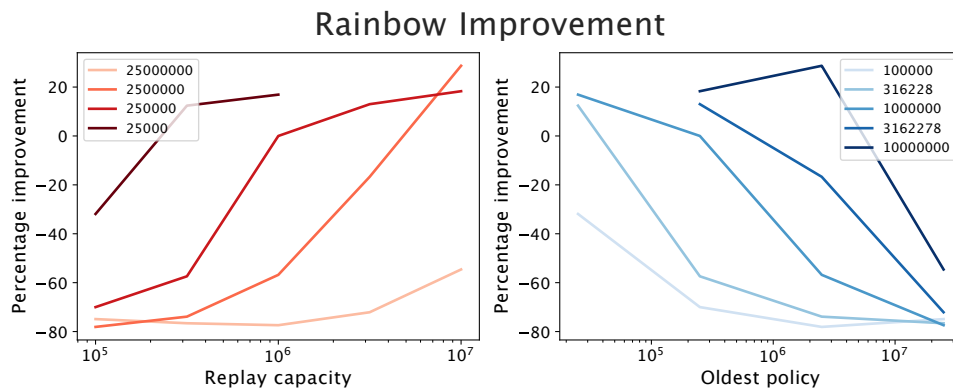
Figure 15. **Performance improvements increase as replay ratio drops.** We plot the results of Figure 2 with respect to the replay ratio, which shows the general trend.



(a) DQN does not show benefits of replay larger than IM, unless older policies are used



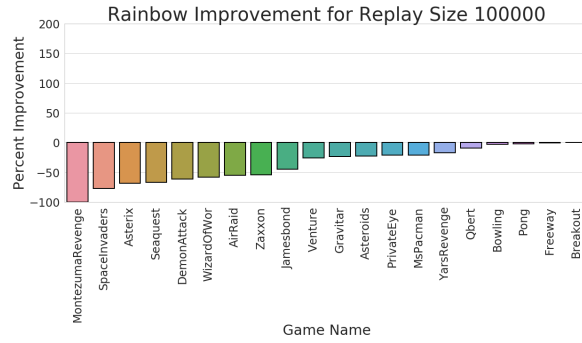
(b) 3-step DQN continues to improve with larger replay capacity



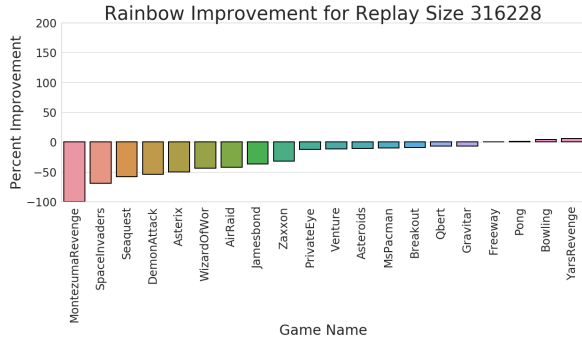
(c) Rainbow continues to improve with larger replay capacity

Figure 16. Performance improves with increased replay capacity and reduced oldest policy age.

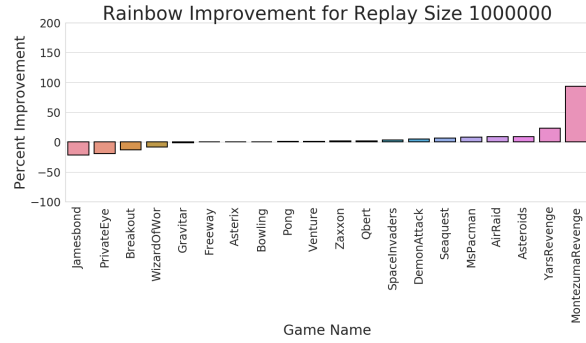
## Revisiting Fundamentals of Experience Replay



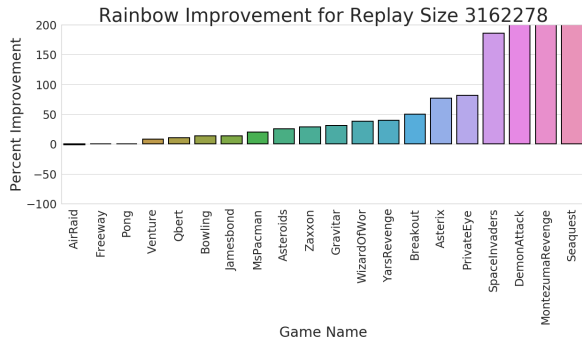
(a) 1M to 100K buffer. Median improvement: -24.6%.



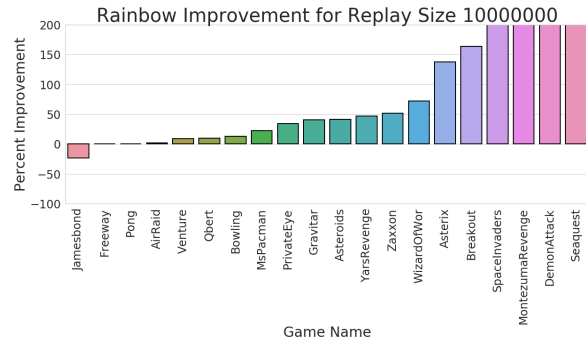
(b) 1M to 316K buffer. Median improvement: -11.8%.



(c) 1M to 1M buffer. Median improvement: 1.6%.



(d) 1M to 3M buffer. Median improvement: 30.2%.



(e) 1M to 10M buffer. Median improvement: 41.0%.

Figure 17. Rainbow replay buffer effects at a per-game level.

## Revisiting Fundamentals of Experience Replay

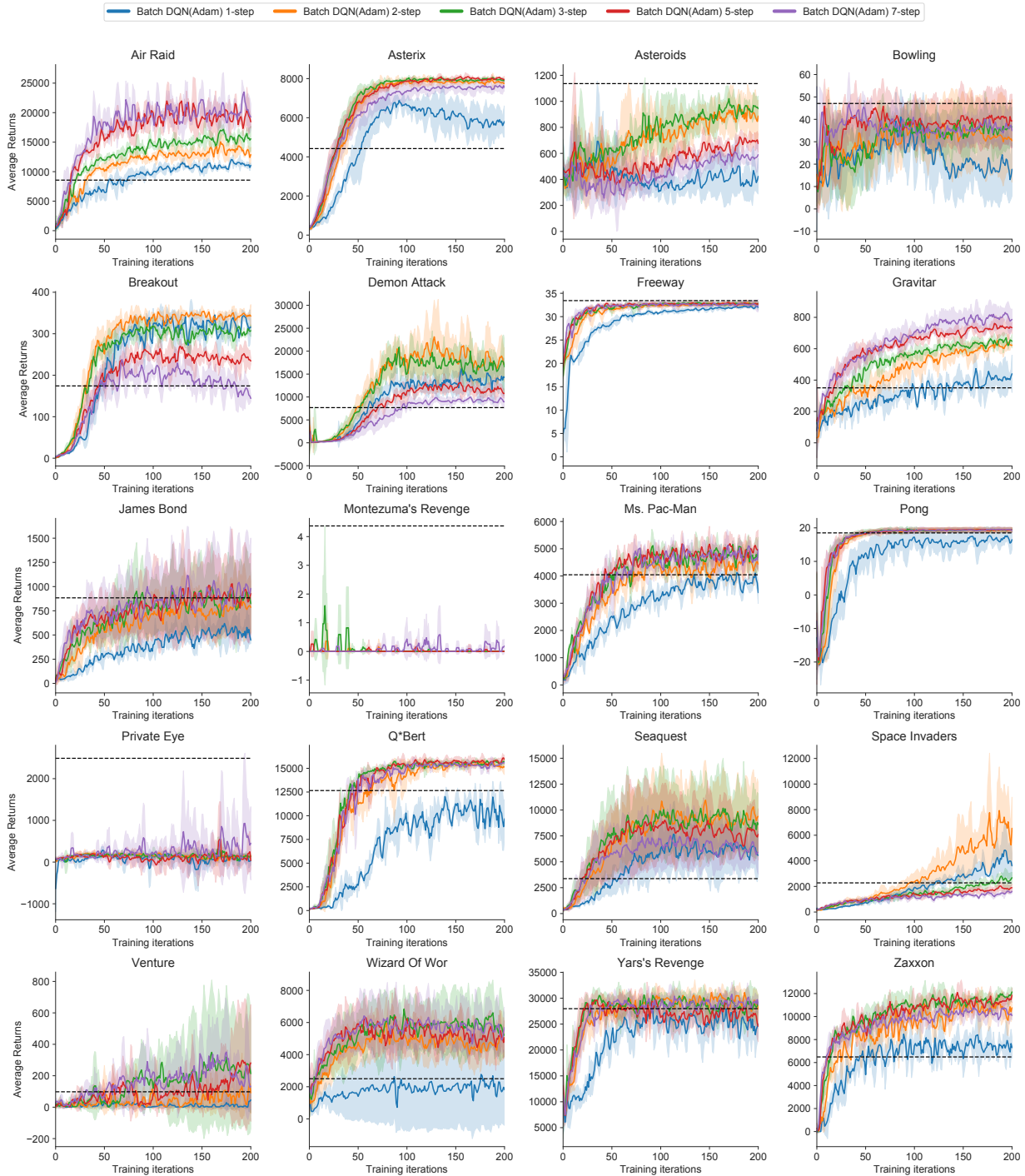


Figure 18. Average evaluation scores across 20 Atari 2600 games of batch DQN (Adam) agent with different  $n$ -step horizons trained offline using the DQN replay dataset (Agarwal et al., 2020). The horizontal line shows the evaluation performance of a fully-trained online DQN agent. The scores are averaged over 3 runs (shown as traces) and smoothed over a sliding window of 3 iterations and error bands show standard deviation.

## Revisiting Fundamentals of Experience Replay

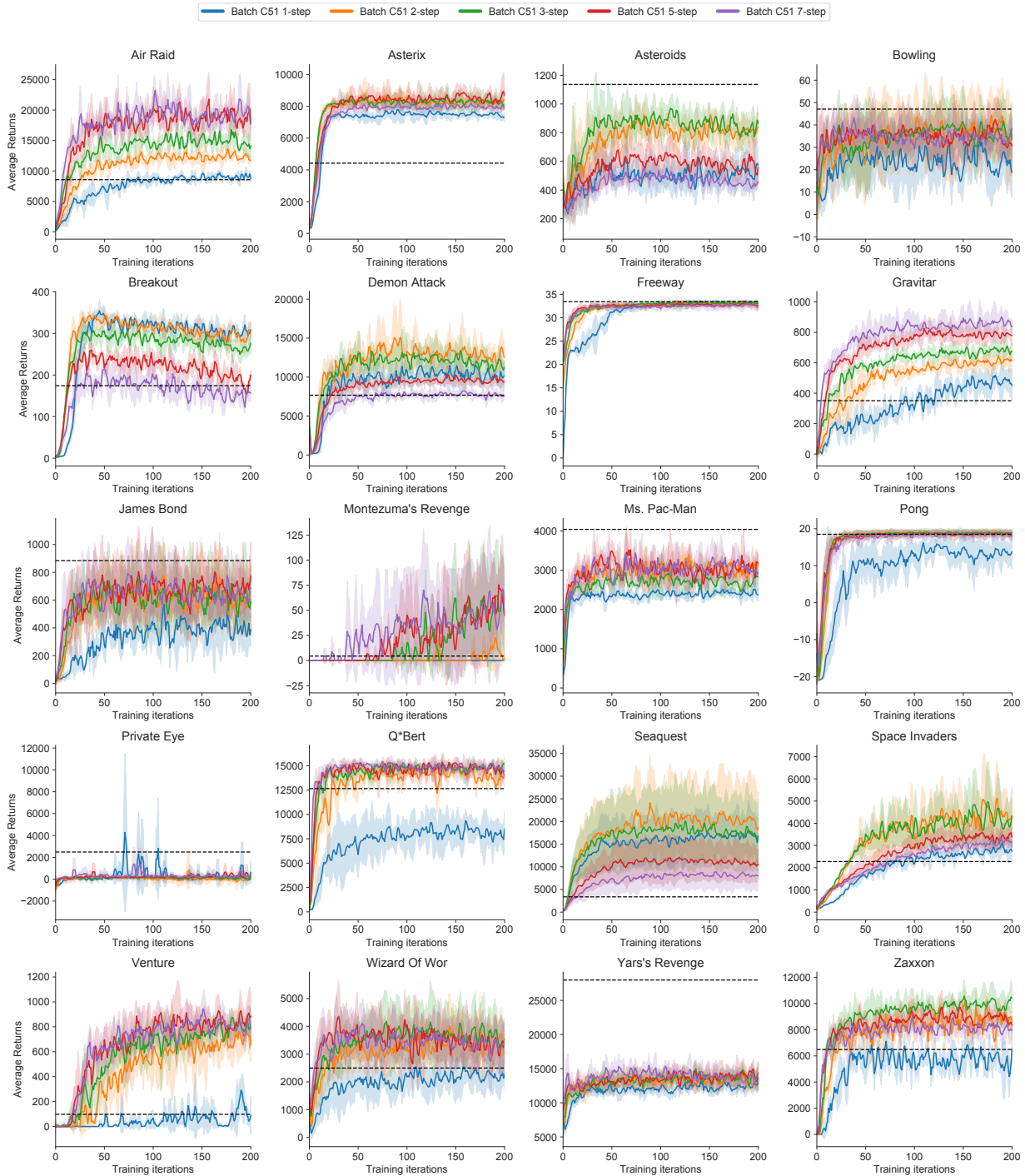


Figure 19. Average evaluation scores across 20 Atari 2600 games of a batch C51 agent with different  $n$ -step horizons trained offline using the DQN replay dataset (Agarwal et al., 2020). The horizontal line shows the evaluation performance of a fully-trained online DQN agent. The scores are averaged over 3 runs (shown as traces) and smoothed over a sliding window of 3 iterations and error bands show standard deviation.