What Matters for Maximizing Data Reuse In Value-based Deep Reinforcement Learning

Anonymous Authors[1]

Abstract

A key ingredient for successfully applying deep reinforcement learning to challenging tasks is the effective use of data at scale. Although originally deep RL algorithms achieved this by storing past experiences collected from a synchronous actor in an external replay memory (DQN; Mnih, 2013), follow-up works scaled training by collecting data asynchronously through distributed actors (R2D2; Kapturowski et al., 2018), and more recently by GPU-optimized parallelization (PQN; Gallici et al., 2024). We argue that DQN, PQN, and R2D2 constitute the outer vertices of a graph, whose edges correspond to the addition or removal of various training techniques, and whose interior has thus far remained unexplored. We conduct a thorough empirical study to populate this interior and develop an understanding of the relationship between the uncovered methods. Our empirical analyses demonstrate that maximizing data reuse involves addressing the deadly triad: Q-lambda rollouts for reducing the bias from bootstrapping, the use of LayerNorm for stabilizing function approximation, and parallelized data collection for mitigating off-policy divergence.

1. Introduction

A number of works have built on DQN (Mnih, 2013) to achieve strong performance in complex benchmarks (Hessel et al., 2018), advances in online representation learning (Castro et al., 2021), exploration (Osband et al., 2016), sample efficiency (Schwarzer et al., 2023), and compute efficiency (Kapturowski et al., 2018). These value-based algorithms leverage previously gathered (off-policy) training data for improved sample efficiency, a distinct advantage over policy-gradient methods that require gathering new (on-policy) data for each update (Sutton & Barto, 2018). Recently, however, GPU-optimization techniques have enabled new environment simulations achieving tens of thousands of steps per second (Matthews et al., 2024; Weng et al., 2022), which can dramatically speed up training RL agents. This computational scalability has shifted the practical balance, favoring policy gradient methods that can consume larger volumes of data at shorter wall-clock times.

Despite their advantages, value-based methods face persistent learning challenges due to the "deadly triad" of RL: the interplay between function approximation, bootstrapping, and off-policy learning (Sutton & Barto, 2018). This triad often induces severe learning instabilities, particularly when reusing off-policy data, as it amplifies the bias-variance trade-offs inherent to temporal difference (TD) learning. Consequently, value-based algorithms require careful hyper-parameter selections, such as small training batches, low replay ratios, and large replay buffers to maintain stability. This hyper-parameter sensitivity has resulted in significant challenges when attempting to leverage increased computational resources (Obando-Ceron et al., 2024).

These issues are more apparent with parallelized data collection, as used by Apex (Horgan et al., 2018), R2D2 (Kapturowski et al., 2018), and PQN (Gallici et al., 2024). These methods explore asynchronous and GPU-compiled training setups, necessitating a number of important design choices. For example, R2D2 uses an LSTM (Hochreiter & Schmidhuber, 1997) and prioritized replay (Schaul, 2015), while PQN discards target networks in favor of Layer Normalization (Lei Ba et al., 2016) and eliminates the replay buffer altogether. The learning targets also vary: DQN uses single-step updates (Q(0)), while R2D2 and PQN calculate multi-step updates over full rollouts. These differences illustrate how progress in value-based algorithms spans architecture, sampling regimes, and learning objectives. Indeed, these improvements are directly addressing deadly triad challenges by reducing the reliance on off-policy data (Gallici et al., 2024; Schaul, 2015) or improving function approximation through techniques such as layer normalization (Lei Ba et al., 2016).

In this work, we take a data-centric view of the deadly triad in deep RL, systematically tracing the evolution of a

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

representative subset of value-based algorithms. Starting with DQN, we revisit the design of modern methods such as R2D2 and PQN, uncovering algorithmic modifications that span improvements in sample and compute efficiency. To structure our investigation, we pose two key research questions that guide our analysis of value-based algorithms: **RQ1:** *What strategies can improve data reuse in high data replay ratio settings, where DQN typically becomes unstable?* **RQ2:** *What are the deadly triad considerations that make it difficult to effectively use a high data replay ratio?* Central to our contributions is the introduction of a novel metric, the Data Replay Ratio (DRR), which provides a more nuanced understanding of the data reuse properties inherent to value-based methods. This metric serves as a foundation for optimizing algorithmic efficiency and is a key highlight of our work.

Through rigorous evaluation on subsets of the Arcade Learning Environment (ALE) (Bellemare et al., 2013), we explore algorithmic modifications that interpolate between foundational and modern value-based methods. By systematically analyzing design choices and their impact, we provide a roadmap for developing scalable, efficient, and robust RL algorithms. Our study highlights promising directions for future research (see Appendix 6.4) and contributes open source implementations to enable further research in the field. A conceptual overview of our investigation is shown in Figure 2.

2. Background

Our work explores the progression from DQN (Mnih, 2013) to more compute- and sample-efficient value-based algorithms. This effort builds on multiple research directions aimed at optimizing data reuse, stabilizing learning dynamics, and addressing the fundamental challenges of the deadly triad. The deadly triad refers to the interconnected challenges that arise when training deep RL agents (Sutton & Barto, 2018). These challenges primarily stem from the need to **bootstrap** from (non-stationary) **off-policy** data while using deep networks as **function approximators**.

2.1. Bootstrapping Bias and Multi-Step Returns

Bootstrapping, where value estimates are updated based on prior estimates, introduces bias in TD learning which can accumulate and destabilize training. DQN (Mnih, 2013) uses Q(0), which estimates targets using one-step Bellman updates. Modern deep RL algorithms frequently use multi-step returns to balance bias and variance: R2D2 (Kapturowski et al., 2018) and Rainbow (Hessel et al., 2018) leverage n-step returns, while Agent57 (Badia et al., 2020) employs Retrace (Munos et al., 2016) to correct for off-policy errors.

Peng's Q(λ) (Peng & Williams, 1994; 1996) has demonstrated empirical success, even in scenarios previously deemed unstable (Kozuno et al., 2021). However, applying Q(λ) to off-policy data presents significant challenges due to the risk of stale updates, which can destabilize learning. To address this, corrective mechanisms like Retrace (Munos et al., 2016) or periodic recomputation of Q(λ) targets (Daley & Amato, 2019) have been proposed. Recently, PQN (Gallici et al., 2024) introduced a lightweight alternative, using λ-returns on parallel on-policy trajectories without an off-policy replay buffer. Our work extends this line of research by further investigating multi-step TD methods in different data reuse regimes.

2.2. Off-Policy Learning and Data Reuse

Off-policy learning is a central challenge in deep RL, as it enables better sample efficiency but also induces instability due to distributional shift (Sutton & Barto, 2018). DQN leverages an experience replay buffer, which allows the agent to reuse past experiences for training (Mnih, 2013). Several extensions on the replay buffer, such as prioritized experience replay (Schaul, 2015) and distributed replay schemes (Horgan et al., 2018; Kapturowski et al., 2018), have improved DQN's learning efficiency by biasing sampling toward informative transitions. However, maximizing data reuse introduces new challenges. While increased data reuse improves efficiency in supervised learning, excessive reuse in RL often leads to training collapse (Kumar et al., 2020; D'Oro et al., 2022; Sokar et al., 2023).

Our work investigates parallel data collection as a potential solution to mitigate learning collapse in high data reuse settings. By designing training settings that replicate DQN's data reuse properties while leveraging parallel environments, we explore whether data-centric modifications to DQN can alleviate instability in high data reuse settings.

2.3. Neural Network Function Approximation

The third challenge of the deadly triad arises from the use of deep neural networks as non-linear function approximators in value-based RL. Unlike learning with linear models, function approximation errors are often unstructured, rendering deep RL particularly susceptible to overestimation bias, representation collapse, and catastrophic forgetting (Sutton & Barto, 2018; Baird et al., 1995; Tsitsiklis & Van Roy, 1996).

DQN partially mitigates these issues with target networks, which stabilize bootstrapping updates by maintaining a slowly updated copy of the Q-network, and recent works have explored alternative stabilizers (Bhatt et al., 2019).

Plasticity loss and primacy bias in deep RL networks have been a barrier to improving deep RL algorithms (Nikishin et al., 2022). Proposed solutions include periodic network resets (Nikishin et al., 2022; Sokar et al., 2023), weight prun-

ing (Obando-Ceron et al., 2024a), and auxiliary penalties to prevent feature rank collapse (Kumar et al., 2020).

Layer normalization (Lei Ba et al., 2016) has been shown to improve training stability by addressing distributional drift in network activations (Lyle et al., 2024). PQN (Gallici et al., 2024) eliminates target networks entirely in favor of LayerNorm and achieves competitive performance without the need for a replay buffer.

Our work builds on these findings by investigating how architectural modifications, such as removing target networks and applying LayerNorm, affect the learning stability, particularly in combination with the modifications aforementioned to tackle the challenges of the deadly triad.

A more comprehensive discussion of our work in relation to existing research can be found in Appendix 6.3.

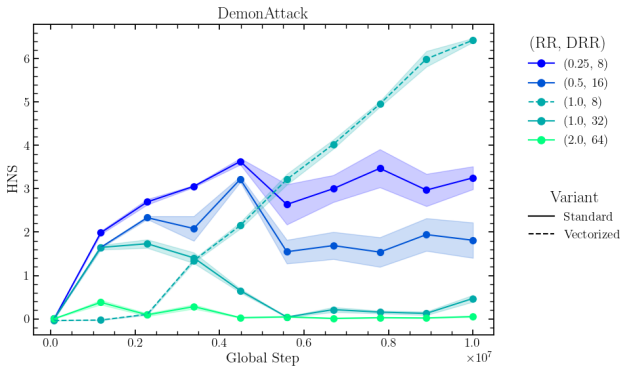


Figure 1. Previous studies suggest that increasing the RR leads to lower-performing policies, attributing this to learning collapse caused by plasticity loss, dying neurons, and feature rank collapse. We argue that this claim is not universally applicable, as the RR fails to account for the data-centric properties on which value-based methods are trained. By incorporating the DRR as a guiding factor, we demonstrate that it is possible to configure experiments with the same algorithm (e.g., DQN) that match the RR of configurations that fail to progress (i.e., RR=1) under different DRR conditions, simply by adjusting the number of vectorized environments and the batch size. Our results indicate that it is high DRR values, rather than RR, that contribute to performance collapse, and that the DRR can reveal training configurations that lead to superior performance

3. Improving Data Reuse in Value-Based RL: Research Questions and the Data Replay Ratio

We introduce the **Data Replay Ratio (DRR)**, a novel metric for analyzing the learning dynamics of value-based algorithms. Unlike the Replay Ratio (RR) commonly used in prior studies (Fedus et al., 2020; D'Oro et al., 2022), the DRR provides a more comprehensive framework for optimizing both sample and compute efficiency.

$$\text{RR} = \frac{\text{num\_grad\_steps}}{\text{train\_frequency}}, \quad \text{DRR} = \text{RR} \cdot \frac{\text{batch\_size}}{\text{num\_envs}}. \quad (1)$$

num_grad_steps controls how many batches to sample from the buffer while *train_frequency* controls how many environment steps are taken between batch samples. The DRR enables the exploration of alternative scaling strategies, such as increasing batch sizes or parallelizing environment rollouts, rather than solely adjusting the number of gradient steps per experience sample. We note that increasing the number of parallel environments has a similar impact on learning dynamics as decreasing the *train_frequency*, particularly in terms of the data generated per unit of computation. However, increasing parallel environments better aligns with recent advancements in compute-scalable value-based algorithms, as it allows for the efficient generation and consumption of data through massively vectorized environment simulations (Gallici et al., 2024; Kapturowski et al., 2018). Unlike simply reducing the replay ratio, which primarily controls the amount of data reused between updates, increasing parallel environments directly influences the diversity and coverage of data in the replay buffer, thereby affecting the stability and efficiency of learning. The DRR represents the ratio of data used for learning to the new data collected per step, providing insight into recent works on data-efficient RL (Riedmiller et al., 2022):

$$\begin{aligned} \text{DRR} &= \frac{\text{num\_grad\_steps} \cdot \text{batch\_size}}{\text{train\_frequency} \cdot \text{num\_envs}} \\ &= \frac{\text{num\_samples\_in\_update}}{\text{num\_samples\_collected}}. \end{aligned} \quad (2)$$

While previous works (Schmidt & Schmied, 2021; Clark et al., 2024) have proposed similar strategies to improve sample efficiency, their focus has been limited to specific algorithms like Rainbow (Hessel et al., 2018) and the data reuse properties have not been explicitly quantified. The DRR facilitates a broader analysis across a wider range of value-based algorithms, offering insights into their evolution and scalability. To address critical gaps in understanding data reuse in value-based algorithms, we pose two central research questions:

RQ1: *What strategies can improve data reuse in high data replay ratio settings, where DQN typically becomes unstable?*

RQ2: *What are the deadly triad considerations that make it difficult to effectively use a high data replay ratio?*

3.1. RQ1: Improving Data Reuse in High Data Replay Ratio Settings

In the original DQN setup, *num_grad_steps* = 1, *train_frequency* = 4, *batch_size* = 32 and *num_envs* = 1, yield-

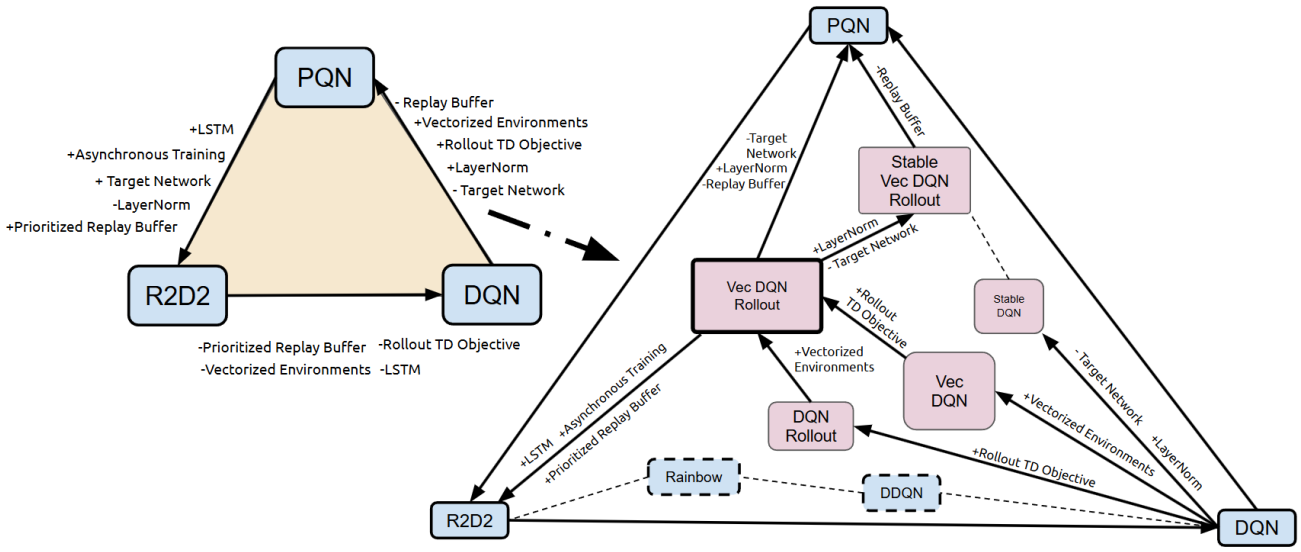


Figure 2. **Conceptual overview of our study**. Modern developments in value-based deep RL algorithms constitute the outer vertices of a graph where edges correspond to adding/removing training techniques. In our work, we systematically trace the evolution of this algorithmic family from the canonical DQN algorithm by means of a composable exploration of data-centric training techniques. Table 4 provides a complete overview of all algorithmic variants used in our experiments throughout the paper.

.

ing RR $= 0.25$ and DRR $= 0.25 \times \frac{32}{1} = 8$. However, increasing *num_grad_steps* to 2 and reducing *train_frequency* to 1 (resulting in RR $= 2$ and DRR $= 64$) has been shown to destabilize learning entirely (D'Oro et al., 2022; Sokar et al., 2023). Throughout the paper, we refer to these unstable conditions as High Data Replay Ratio (HDRR) settings.

With the DRR, we extend this analysis by incorporating batch size and the number of parallel environments into the discussion, allowing for a deeper understanding of how these factors interplay with stability. We hypothesize that DQN's vulnerability to HDRR is tied to its reliance on small batch sizes (susceptible to priming and overfitting) and limited buffer diversity caused by using a single environment instance. Thus, claims about HDRR universally destabilizing DQN are context-dependent and linked to this fragile configuration.

To address these limitations, we propose and evaluate algorithmic modifications informed by the DRR. These include scaling batch sizes, increasing the number of environments, and designing healthier data regimes that improve data reuse without sacrificing stability (Figures 4 and 20).

3.2. RQ2: Addressing the Challenges of the Deadly Triad

For RQ2, we focus on mitigating challenges posed by the deadly triad (i.e. bootstrapping from off-policy data with neural network function approximators).

First, we explore less biased TD objectives. While Q(0) (Sutton & Barto, 2018; Mnih, 2013) remains a widely used method, we hypothesize that its bias can exacerbate poor data reuse, reducing both performance and sample efficiency. To address this, we adopt λ-returns (Peng & Williams, 1994), which incorporates multi-step returns with reduced bias and provide a flexible interpolation of Q(0) and Monte Carlo returns (Sutton & Barto, 2018). We modify the replay buffer to store fixed-length rollouts (see Figure 21), allowing the computation of Q(λ) targets from sequences rather than individual transitions.

Secondly, for RQ1, we test Q(λ) in both low and HDRR settings combined with single and parallel environments to mitigate off-policy divergence.

Finally, we explore improved non-linear approximation by means of parameter normalization and re-considering the use of target networks in deep value-based algorithms.

3.3. Building a Comprehensive Map of Value-Based Algorithms

To address RQ1 and RQ2, we focus on a set of representative value-based algorithms: DQN, R2D2 (Kapturowski et al., 2018), and PQN (Gallici et al., 2024), chosen for their relevance across the spectrum of scalability, stability, and data usage in Q-learning. Our interest lies specifically in algorithms that extend or modify canonical Q-learning by leveraging different mechanisms for temporal credit assignment, stability, and compute efficiency. Starting from

PQN, a recent value-based algorithm designed to be highly scalable and compute-efficient by removing replay buffers and target networks, we systematically trace back its core components: the use of parallel environments, rollout-based objectives, and off-policy updates. This naturally connects it to earlier methods like R2D2, which similarly incorporates rollout objectives and parallel data collection (albeit with replay and target networks), and further back to DQN, the canonical instantiation of value-based learning via TD(0). By situating these algorithms along a consistent design axis, we construct a unified experimental framework for analyzing learning dynamics through the lens of the DRR. This framing enables us to identify and compare stability and sample efficiency challenges under a shared metric, laying the groundwork for general principles in scalable value-based RL (see Figure 2).

4. Empirical analyses

Our empirical analyses are designed to address the RQs outlined in Section 3, focusing on evaluating the data reuse properties and learning dynamics of various value-based deep RL algorithms. In particular, our aim is to "fill in" the interior of the simplex formed with DQN, R2D2, and PQN as the outer vertices (see Figure 2, left). As such, we will be referring to Figure 2 (right) to indicate which path in the completed triangle is being discussed.

Guided by the challenges of the deadly triad and toward maximizing data reuse, we explore three orthogonal axes: (1) the data sampling regime, investigating how parallel data collection impacts the presence of off-policy data in the replay buffer; (2) the use of rollout TD methods, such as Q(λ), to address bootstrapping bias; and (3) improvements to function approximation through normalization and the removal of target networks. These axes yield algorithmic variants that interpolate between DQN, R2D2, and PQN: (1) **Vec-DQN**, which augments DQN with parallel environments (Figure 4); (2) **DQN-Rollout**, which integrates Q(λ) TD estimates into single-environment DQN; (3) **Vec-DQN-Rollout** as the combination of (1) and (2); and (4) **Stable DQN** and **Stable Vec-DQN-Rollout**, incorporating layer normalization and removing target networks. Table 4 provides a complete overview of all algorithmic variants used in our experiments throughout the paper.

4.1. Experimental setup

We conduct our experiments using eight representative games from the Arcade Learning Environment (ALE) benchmark (Bellemare et al., 2013), together with 3 partially-observable tasks from the VizDoom suite (Kempka et al., 2016). Environment details are provided in Appendix 6.1. In addition to reporting accumulated returns during training, our analyses also include the mean, median and interquantile mean (IQM) of the human-normalized scores (HNS) with 95% stratified bootsrap confidence intervals (Agarwal et al., 2021) obtained with 3 independent runs for each experiment configuration. Although it is beyond the scope of our work, we provide a preliminary investigation of data reuse for continuous-action algorithms in Appendix 6.7, as our investigation traces the evolution of value-based algorithms from PQN, which naturally connects to discrete-control algorithms.

4.2. The impact of parallel data collection

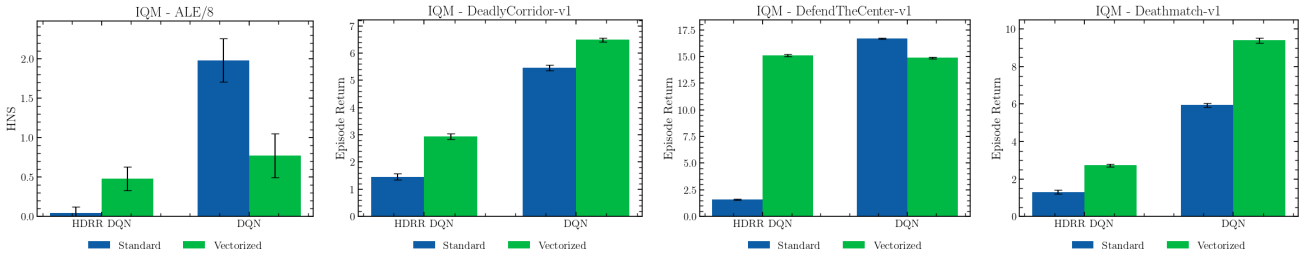
We start by investigating the effect of parallel data collection, focusing on the transition from DQN to Vec-DQN in Figure 2 (right), where the DRR provides a framework for analyzing data reuse properties. As noted in Section 3, DQN typically performs well with a DRR of 8 but experiences learning collapse under HDRR conditions. We investigate whether parallelized data collection can address this issue by designing a training configuration that replicates DQN's data reuse by using parallel environments and larger training batches (see Figure 4). Specifically, we evaluate whether this data regime can mitigate the learning collapse observed in HDRR-DQN. The resulting methods, Vec-DQN, HDRR-DQN, and HDRR-Vec-DQN are illustrated in Figure 20. In Figure 3a, we compare the performance of DQN, HDRR-DQN, Vec-DQN, and HDRR-Vec-DQN. Results indicate that with the simple Q(0) objective, configurations utilizing a single environment instance and small learning batches yield higher aggregated performance.

However, increasing the DRR generally results in diminished performance, with the non-vectorized versions collapsing entirely. Interestingly, vectorized (i.e. parallel) algorithms, which use larger learning batches to mimic the DRR of their equivalent single-environment versions (see Figure 4), exhibit greater robustness under HDRR regimes, avoiding complete performance collapse.

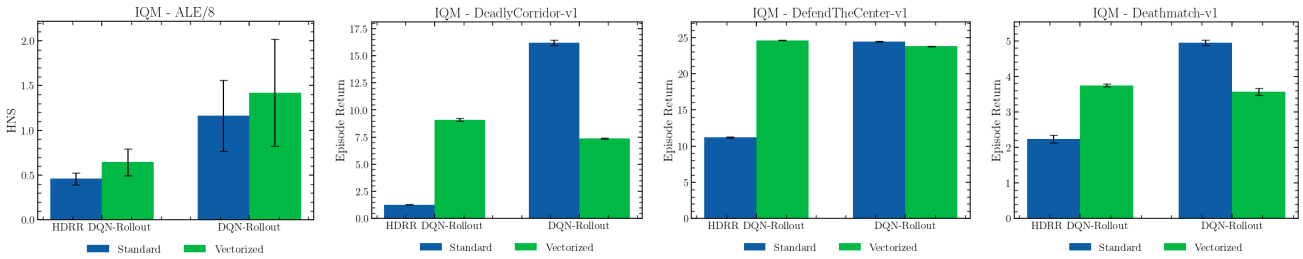
These findings suggest that small learning batches induce better optimization, although at the cost of increased sensibility to priming and overfitting. This observation aligns well with the findings by Obando Ceron et al. (2024). Conversely, larger batches mitigate overfitting by averaging gradients over more diverse data but are more prone to becoming stuck in local minima in the non-stationary optimization process of deep RL. To maintain consistency, we adjust the learning rate in vectorized settings based on the batch size increase, following a proportional scaling by the square root of the batch size.

4.3. The impact of rollouts

In Figure 3 we explore the use of rollouts to reduce biased updates by using λ-returns, which suggests that rollout TD methods are more compatible with vectorized envi-

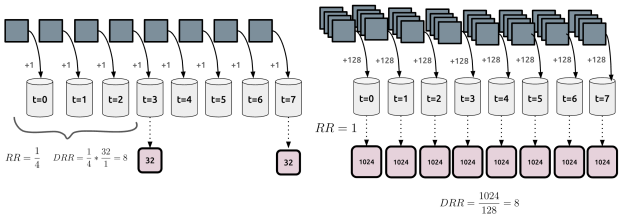


(a) Q(0)



(b) λ-returns

Figure 3. **IQM scores across different environments for Q(0) and Q(λ).** The first row presents results for Q(0), while the second row shows results for Q(λ). For ALE/8 (first column), we report the interquartile mean (IQM) of human-normalized scores (HNS), aggregated over 8 selected Atari games from the Arcade Learning Environment (ALE) with 100M training frames. The other three environments: Deadly Corridor, Defend the Center, and Deathmatch, are partially observable tasks from the ViZDoom suite, where we report IQM episode returns over 10M training frames. Results demonstrate that multi-step objectives like Q(λ) (rollout variants) mitigate the collapse seen under High Data Replay Ratio (HDRR) settings. Additionally, using parallel environments and larger batch sizes often yields better performance, even under HDRR, with configurations that outperform single-environment setups and previously collapse-prone RR values (e.g., RR=1), but with lower DRR values (DRR=8). This suggests that DRR better captures the data reuse properties of algorithms, enabling improved performance beyond traditional batch size and RR configurations.



(a) DQN (b) Vec-DQN

Figure 4. **Illustration of the Data Replay Ratio induced by different data sampling configurations**. Gray squares represent environment observations, where *Vec-* denotes vectorized (parallel) environments contributing observations to a shared buffer. Red squares represent learning batches.

ronments and larger batch sizes. This is evident from the similar median HNS and, more notably, a higher IQM. Additionally, while vectorized algorithms remain more robust under HDRR regimes (consistent with the findings from Section 4.2), the non-vectorized variants also avoid the complete collapse observed when using Q(0). These results suggest that less biased TD objectives, such as λ-returns, have the potential to stabilize the learning dynamics of value-based algorithms and mitigate the risks of collapse under high data reuse settings. The full set of learning curves is presented in Figures 14a and 14b.

In Appendix 6.2, we provide a detailed analysis of the learning dynamics induced during training. Our findings reveal notable correlations between several recently proposed learning metrics in deep RL and the HNS. Specifically, we observe that increasing the DRR generally leads to a decrease in the HNS across all value-based algorithms studied in this work. Furthermore, the presence of dormant neurons (Sokar et al., 2023) in the convolutional layers has a significant negative impact on performance. We discuss this important finding further in Appendix 6.4.

4.4. Addressing training instabilities

Gallici et al. (2024) provided theoretical arguments for the use of LayerNorm (Lei Ba et al., 2016) to compensate for the instabilities caused by bootstrapping from the online network, thus removing the need for target networks, even when using off-policy data; in their empirical evaluations, however, the authors focused on on-policy settings. We

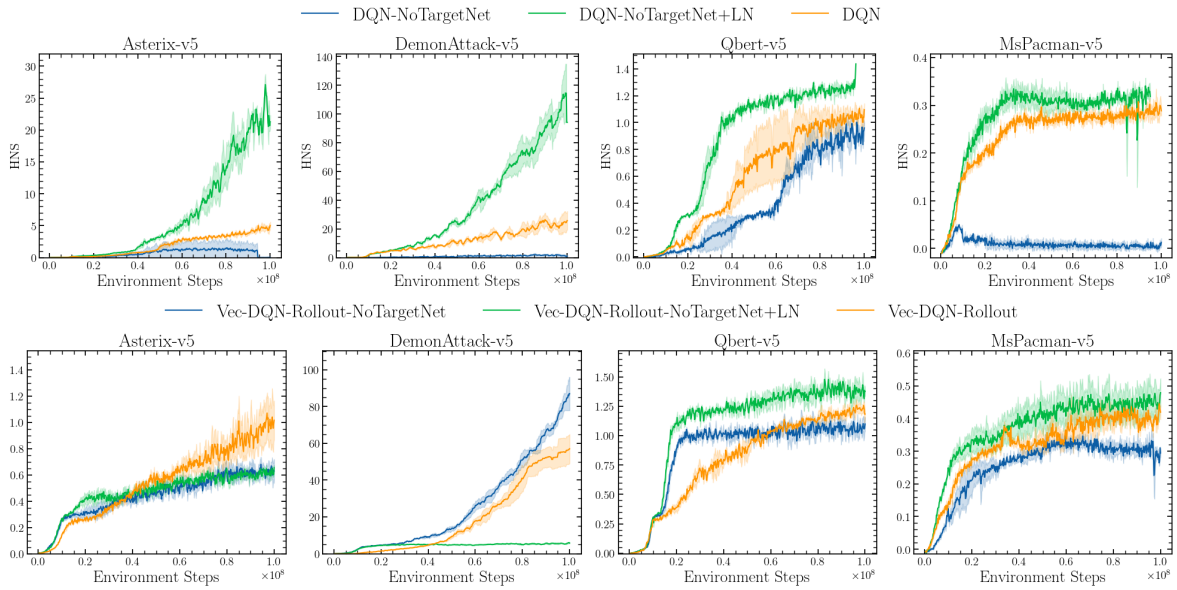


Figure 5. Mean human-normalized scores when using a target network, bootstrapping from the online network, and bootstrapping from the online network using LayerNorm. The **variant with LayerNorm consistently performs best with Q(0), demonstrating that improving function approximation with LayerNorm can eliminate the need for target networks**. Throughout the paper, we refer to this variant as **Stable DQN** (see Figure 2). However, with $Q(\lambda)$, performance becomes more environment-specific; in DemonAttack, the variant without LayerNorm or target networks achieves the best results. **This suggests that less biased TD methods like $Q(\lambda)$ can also allow safe bootstrapping from the online network in certain environments**.

evaluated its impact in off-policy methods in Figure 5, and find that LayerNorm is a viable alternative to target networks, consistent with the findings of Gallici et al. (2024). Our results show significant performance gains in favor of layer normalization compared to the original DQN algorithm, which uses target networks. Notably, DQN with Q(0) experiences a complete collapse in performance when the network is not normalized and target networks are absent (blue lines in Figure 5). The use of Q(λ) returns to estimate the TD targets seems to avoid this collapse (green lines in Figure 5). This result can be interpreted through the lens of the deadly triad, where improvements in function approximation gained from learned layer normalization result in safer bootstrapping from off-policy data, especially when utilizing highly-biased bootstrapped estimates from Q(0).

5. Closing the triangle

As shown in Figure 2, the parallel version of DQN-Rollout (Vec-DQN-Rollout) is just one step away from PQN (Gallici et al., 2024) and R2D2 (Kapturowski et al., 2018). Building on this connection, we study how these improvements impact parallel data collection, reuse, and overall efficiency.

5.1. Connecting to R2D2 and PQN

R2D2 (Kapturowski et al., 2018) was established as the state-of-the-art algorithm in the ALE at the time of its introduction. R2D2 is the foundational algorithm which serves as the basis of the current state-of-the-art algorithms in the ALE such as MEME (Kapturowski et al., 2022) and Agent57 (Badia et al., 2020). In our work, R2D2 emerges as Vec-DQN-Rollout augmented with an LSTM, a prioritized sampling scheme, and distributed training. We provide an implementation of R2D2 that faithfully reproduces the core components of the original method: (i) a recurrent Q-network, (ii) a prioritized replay buffer, and (iii) distributed computation.[1] Further implementation details are available in Appendix 6.5.

Similarly, PQN can be viewed as Vec-DQN-Rollout augmented with LayerNorm, and removing the target network and replay buffer.

5.2. The DRR frontier

Having established the connection to these algorithms, we investigate the learning dynamics induced by different DRR regimes. In asynchronous settings, and especially when using a prioritized replay buffer, the DRR cannot be trivially computed. Therefore, in our implementation of R2D2, we empirically track how often each data point in the buffer is sampled during training to estimate the DRR. We then study the task performance as a function of this empirical

[1]We will release code together with the R2D2 implementation with the final version of this work.

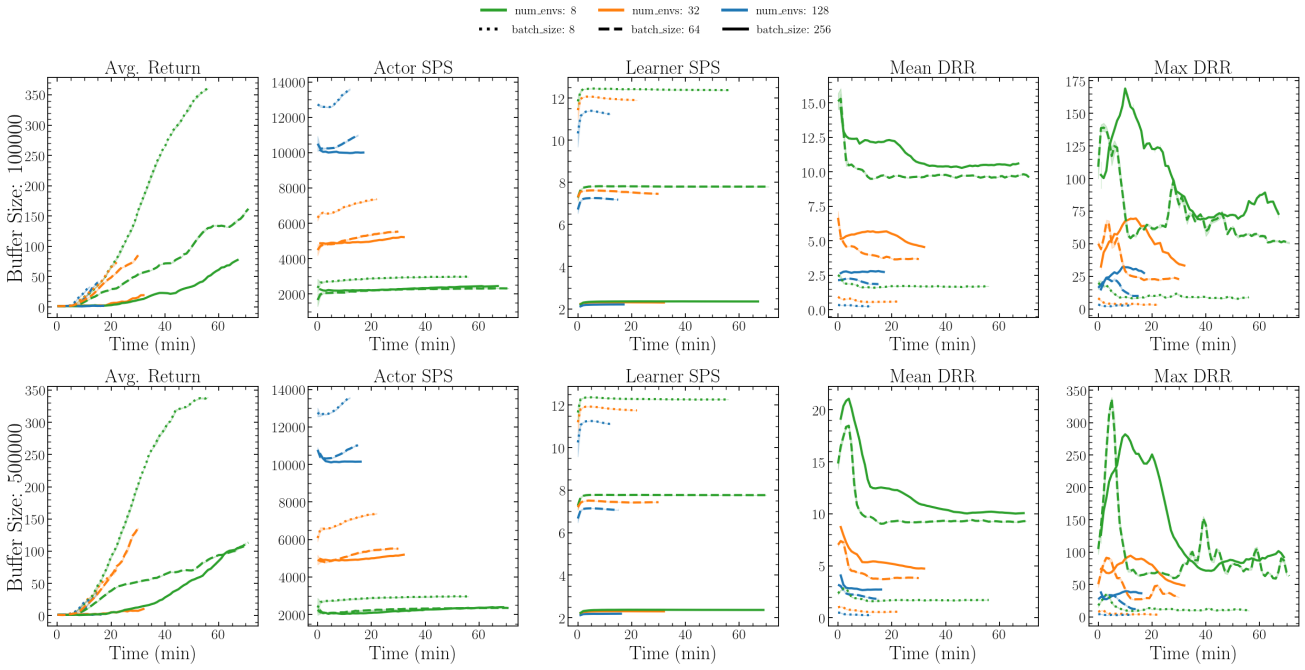


Figure 6. **Asynchronous training of R2D2 as a function of training time (40M frames)**. Variants using 128 parallel actors exhibit the fastest performance, achieving between 10k and 14k simulation Steps Per Second (SPS). However, these variants show lower DRR and returns because the learners generally do not have enough time to process all the data before it gets replaced in the buffer. In contrast, the variants using 8 actors in parallel are the slowest, taking around 1 hour to complete the experiments, but are also more sample efficient. Using 8 actors in parallel and small batches of size 8 yields the largest average returns given the fixed budget of environment steps.

DRR estimate. The results of the R2D2 training runs are presented in Figure 6.

As can be seen, there exists a spectrum of DRR, with a region that balances task performance with efficiency. Neither the lowest DRR (blue dotted line) nor the highest DRR (green solid line) configurations perform the best, suggesting that adjustments to the number of environments, batch size, and replay buffer size can increase DRR for improved sample efficiency. However, there is a limit beyond which further increases in DRR lead to decreased performance. This phenomenon is not exclusive to R2D2 but is observed across all value-based algorithms we study (see Figure 7). A key challenge moving forward is developing algorithms that can continue to extract the maximum information from data to learn effectively, even as DRR increases.

Next, we analyze PQN's data reuse properties by varying the DRR similar to our previous analysis of DQN in Figure 3a and R2D2 in Figure 6. Importantly, PQN trains exclusively from on-policy data batches collected using parallel environments. After a set number of gradient steps, the data is discarded rather than stored in a large replay buffer. This approach mirrors the training process of PPO (Schulman et al., 2017).

In this setting, the number of times each data point is used for training is dictated entirely by the num_epochs parameter, becoming equivalent to the DRR. Our results, illustrated in Figure 7, reveal several key insights:

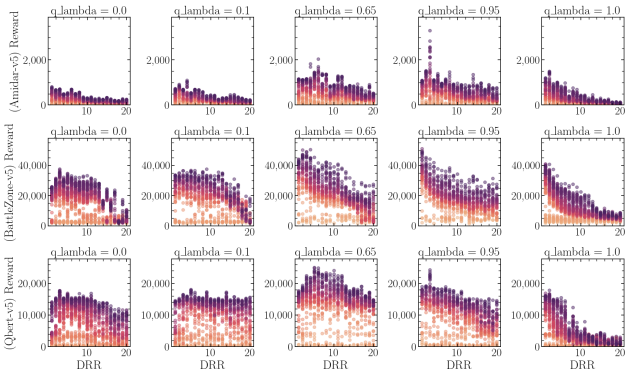


Figure 7. **PQN scores as the DRR increases (200M frames)**. Each row corresponds to a different game from the ALE which facilitates evaluating the generality of the results. Each column represents a different value of λ for assessing the impact of rollout TD objectives. The x-axis varies the number of update epochs (i.e. the DRR in PQN). The y-axis shows the mean score achieved in the games. Shades represent various stages of training, going from light (early) to dark (late).

As with R2D2, there exists a trade-off between sample

efficiency and learning collapse as DRR increases. Fewer training epochs generally result in diminished sample efficiency and reduced performance given a fixed budget of environment interactions, while excessively HDRR causes a collapse in learning dynamics, leading to performance losses. This trade-off is reflected in the bell-shaped patterns, particularly evident in Qbert ($q\_lambda = 0.65$).

Highly biased TD targets hinder data reuse. Figure 7 sheds light on both RQs. PQN, which utilizes parallel environments and a large batch size, experiences a drop in performance as DRR increases but avoids complete collapse, likely due to the stabilizing effect of large batches. (as observed in Figure 3a). However, highly biased TD targets, such as with $q\_lambda = 0.0$ or $q\_lambda = 0.1$, lead to substantial performance declines in HDRR settings. In contrast, targets with less bias, obtained using $q\_lambda = 0.65$ or $q\_lambda = 0.95$, allow for safer increases in DRR. Furthermore, when $q\_lambda = 1.0$, where TD estimates become Monte Carlo returns, we observe severe performance collapse in HDRR settings. This suggests that the trade-off between sample efficiency and learning collapse is influenced not only by the bias in TD estimates but also by their variance.

PQN's behavior in HDRR settings is environment-specific. For instance, in BattleZone, performance decreases nearly monotonically as the DRR increases, while in Amidar and Qbert, the performance decrease is less steep. This observation motivates the need for further exploration into the environment-specific properties that influence these differing patterns.

To complement our evaluation, we benchmarked PQN and R2D2 on the Atari-10 suite, which has been shown to strongly correlate with the full set of games in the ALE (Aitchison et al., 2023; Gallici et al., 2024). For a detailed analysis of the results, we refer the reader to Appendix 6.6.

Finally, our results highlight several promising research directions to advance progress in developing scalable and efficient deep value-based RL algorithms. These are outlined in detail in Appendix 6.4.

6. Conclusion

In this work, we revisited advancements in value-based deep RL algorithms since the introduction of the canonical DQN algorithm, focusing on the data reuse properties that distinguish this class of algorithms. Considering the issues of the deadly triad, we proposed modifications to value-based algorithms by redefining the data regime and introducing the Data Replay Ratio (DRR) as a more precise measure of sample efficiency compared to the commonly used Replay Ratio (RR). Unlike the RR, the DRR quantifies the usage of each individual data point during training.

Using the DRR, we proposed a data sampling regime that mimics the characteristics of the original DQN implementation but incorporates larger training batches and parallel environments. Our goal was to alleviate the issues of the deadly triad by achieving more stable bootstrapping from off-policy data. We defined two RQs to drive our empirical study and be able to tackle the deadly triad from a data-centric perspective, concretely by studying, larger training batches, parallel data collection, less biased TD methods and normalized networks.

Our results indicate that while larger batches trade off the performance benefits associated with the higher variance of smaller batches (Obando Ceron et al., 2024), they offer increased robustness to overfitting. Furthermore, less biased TD methods, such as Q(λ) in place of Q(0), generally improve data reuse and task performance. Finally, our empirical evidence supports the use of normalized networks as a viable alternative to target networks, even in off-policy training scenarios, aligning with the theoretical findings of Gallici et al. (2024).

Crucially, our study facilitates a clearer mapping of modern developments in value-based algorithms and their interconnections. To this end, we will release an open-source implementation of the R2D2 algorithm with the final version of our work, which emerged naturally in our framework as Vec-DQN-Rollout with the addition of an LSTM network, prioritized sampling, and asynchronous computation. We benchmarked R2D2 alongside PQN, a recent, simple, yet effective on-policy value-based algorithm, on multiple subsets of the ALE including the Atari-10 (Aitchison et al., 2023).

As advances in the software and hardware used to train deep networks continue to progress, RL researchers are empowered to tackle problems of increasing complexity with ever-growing models. Our investigation has highlighted that, even in these modern and large-scale settings, the canonical deadly triad continues to prove a central challenge to overcome. In order to develop robust and reliable agents, it is crucial that we develop a deep understanding of the challenges and trade offs that accompany training under these settings. Our work, which takes a data-centric perspective, is a step towards developing this understanding.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Aitchison, M., Sweetser, P., and Hutter, M. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pp. 421–438. PMLR, 2023.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pp. 507–517. PMLR, 2020.

Baird, L. et al. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the twelfth international conference on machine learning*, pp. 30–37, 1995.

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.

Bhatt, A., Palenicek, D., Belousov, B., Argus, M., Amiranashvili, A., Brox, T., and Peters, J. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. *arXiv preprint arXiv:1902.05605*, 2019.

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

Castro, P. S., Kastner, T., Panangaden, P., and Rowland, M. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34:30113–30126, 2021.

Clark, T., Towers, M., Evers, C., and Hare, J. Beyond the rainbow: High performance deep reinforcement learning on a desktop pc. *arXiv preprint arXiv:2411.03820*, 2024.

Daley, B. and Amato, C. Reconciling λ-returns with experience replay. *Advances in Neural Information Processing Systems*, 32, 2019.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International conference on machine learning*, pp. 3061–3071. PMLR, 2020.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Gallici, M., Fellows, M., Ellis, B., Pou, B., Masmitja, I., Foerster, J. N., and Martin, M. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

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 *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Hussing, M., Voelcker, C., Gilitschenski, I., Farahmand, A.-m., and Eaton, E. Dissecting deep rl with high update ratios: Combatting value overestimation and divergence. *arXiv e-prints*, pp. arXiv–2403, 2024.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Kapturowski, S., Campos, V., Jiang, R., Rakićević, N., van Hasselt, H., Blundell, C., and Badia, A. P. Human-level atari 200x faster. *arXiv preprint arXiv:2209.07550*, 2022.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE conference on computational intelligence and games (CIG)*, pp. 1–8. IEEE, 2016.

Kozuno, T., Tang, Y., Rowland, M., Munos, R., Kapturowski, S., Dabney, W., Valko, M., and Abel, D. Revisiting peng's q (lambda) for modern reinforcement learning. In *International Conference on Machine Learning*, pp. 5794–5804. PMLR, 2021.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Lee, H., Hwang, D., Kim, D., Kim, H., Tai, J. J., Subramanian, K., Wurman, P. R., Choo, J., Stone, P., and Seno, T. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. *arXiv preprint arXiv:2410.09754*, 2024.

Lei Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *ArXiv e-prints*, pp. arXiv–1607, 2016.

Li, A. A., Lu, Z., and Miao, C. Revisiting prioritized experience replay: A value perspective. *arXiv preprint arXiv:2102.03261*, 2021.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lyle, C., Zheng, Z., Khetarpal, K., Martens, J., van Hasselt, H., Pascanu, R., and Dabney, W. Normalization and effective learning rates in reinforcement learning. *arXiv preprint arXiv:2407.01800*, 2024.

Matthews, M., Beukman, M., Ellis, B., Samvelyan, M., Jackson, M., Coward, S., and Foerster, J. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*, 2024.

Mnih, V. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Nauman, M., Bortkiewicz, M., Miłoś, P., Trzciński, T., Ostaszewski, M., and Cygan, M. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. *arXiv preprint arXiv:2403.00514*, 2024a.

Nauman, M., Ostaszewski, M., Jankowski, K., Miłoś, P., and Cygan, M. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. *arXiv preprint arXiv:2405.16158*, 2024b.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Obando-Ceron, J., Araújo, J. G., Courville, A., and Castro, P. S. On the consistency of hyper-parameter selection in value-based deep reinforcement learning. *arXiv preprint arXiv:2406.17523*, 2024.

Obando Ceron, J., Bellemare, M., and Castro, P. S. Small batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Obando-Ceron, J., Courville, A., and Castro, P. S. In deep reinforcement learning, a pruned network is a good network. *arXiv preprint arXiv:2402.12479*, 2024a.

Obando-Ceron, J., Courville, A., and Castro, P. S. In value-based deep reinforcement learning, a pruned network is a good network. *arXiv preprint arXiv:2402.12479*, 2024b.

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

Peng, J. and Williams, R. J. Incremental multi-step q-learning. In *Proceedings of the International Conference on Machine Learning*, 1994.

Peng, J. and Williams, R. J. Incremental multi-step q-learning. *Machine Learning*, 22(1):283–290, 1996.

Riedmiller, M., Springenberg, J. T., Hafner, R., and Heess, N. Collect & infer-a fresh look at data-efficient reinforcement learning. In *Conference on Robot Learning*, pp. 1736–1744. PMLR, 2022.

Schaul, T. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schaul, T., Barreto, A., Quan, J., and Ostrovski, G. The phenomenon of policy churn. *Advances in Neural Information Processing Systems*, 35:2537–2549, 2022.

Schmidt, D. and Schmied, T. Fast and data-efficient training of rainbow: an experimental study on atari. *arXiv preprint arXiv:2111.10247*, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 32145–32168. PMLR, 2023.

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

Tsitsiklis, J. and Van Roy, B. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Weng, J., Lin, M., Huang, S., Liu, B., Makoviichuk, D., Makoviychuk, V., Liu, Z., Song, Y., Luo, T., Jiang, Y., et al. Envpool: A highly parallel reinforcement learning environment execution engine. *Advances in Neural Information Processing Systems*, 35:22409–22421, 2022.

Zhang, S., Yao, H., and Whiteson, S. Breaking the deadly triad with a target network. In *International Conference on Machine Learning*, pp. 12621–12631. PMLR, 2021.

Appendix

6.1. Environment Details

In this paper, we use subsets of the Arcade Learning Environment (ALE) (Bellemare et al., 2013) for our experiments. For evaluating the research questions, we use 8 representative games from ALE. To benchmark the most promising algorithmic variations identified in this study, alongside modern baselines such as PQN (Gallici et al., 2024) and R2D2 (Kapturowski et al., 2018), we use the Atari-10 subset (Aitchison et al., 2023). This subset comprises 10 games chosen for their strong correlation with evaluation results on the full set of 57 ALE games.

- **Eight representative ALE games**: Breakout, SpaceInvaders, Asterix, Pong, Qbert, DemonAttack, Seaquest, MsPacman
- **Atari-10**: Amidar, Bowling, Frostbite, KungFuMaster, RiverRaid, BattleZone, DoubleDunk, NameThisGame, Phoenix, Qbert

6.2. Learning Dynamics

We investigate the learning dynamics of the different value-based algorithms evaluated in Figures 3a and 3 by measuring several recently proposed learning metrics to uncover their relationships with the downstream task performance. These metrics include the emergence of dead neurons, computed as dormant neurons with $\tau = 0$ (Sokar et al., 2023; Nikishin et al., 2022), policy churn (Schaul et al., 2022), and representation rank (Kumar et al., 2020). The results are shown in Figure 19 and indicate that while there are some interesting correlations between learning metrics from different recent works, none result in a significantly strong relationship with HNS. Notable relations include:

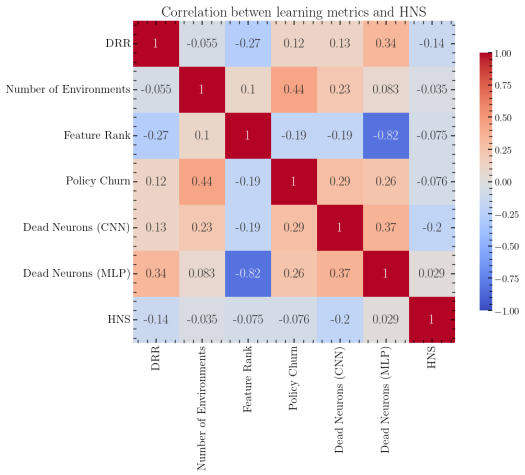


Figure 8. Correlation matrix with statistics aggregated over the 8 ALE environments used throughout the paper. This analysis captures linear relationships between widely used learning metrics, providing insights into the learning dynamics of deep RL algorithms.

- Higher DRR generally leads to worse performance, increased policy churn, a decrease in the rank of the representations (which causes a loss of the network's expressivity) (Kumar et al., 2020), and more dead neurons, especially in the penultimate fully-connected layer of the networks.
- Using a higher number of environments (i.e., also implying larger batch sizes) drastically reduces the number of dead neurons in the representations, at the cost of a slight increase in dead neurons in the convolutional layers. This allows the network to learn higher-rank representations. Notably, larger batches increase policy churn, which can help with exploration but also makes updates more unstable, as the greedy policy varies more abruptly.
- Feature rank does not seem to have a significant relation with HNS, though it is strongly (negatively) correlated with the presence of dead neurons in the representations. The appearance of dead neurons significantly diminishes the rank of the representations.

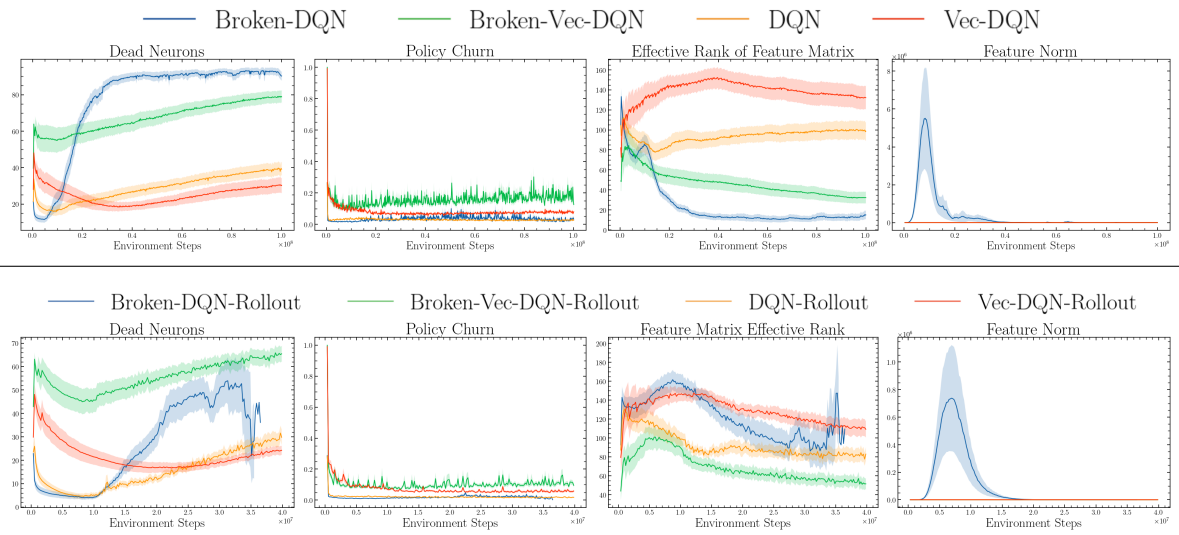


Figure 9. Learning dynamics during training for the algorithms evaluated in the RQs.

- Policy churn is correlated with several other metrics, as larger batches induce more churn, but also more dead neurons. However, the aggregate of all results indicates that there is no strong relationship between churn and HNS.
- Interestingly, dead neurons in the convolutional layers occur less frequently than in the penultimate fully-connected layer, but when they do appear, they have a more severe impact on task performance. This underscores the importance of improving the expressivity of convolutional architectures in online deep RL and empirically justifies the widespread adoption of the residual architecture, such as the Impala CNN (Clark et al., 2024; Castro et al., 2018), in recent literature. This architecture has shown significant performance gains across benchmarks in deep RL, despite the lack of rigorous studies explaining why.

We present the learning dynamics during training in Figure 9, which provide several insightful observations:

- HDRR leads to an increase in the number of dead neurons during training. Additionally, it results in a collapse of the representation rank, particularly when using Q(0) for optimization. In contrast, Q(λ) proves to be a more robust objective, helping to prevent this feature rank collapse.
- HDRR also causes a significant increase in the norm of the representations, especially in the early stages of training, which negatively impacts performance. Notably, this effect is mitigated when using larger training batches.
- Larger training batches tend to increase policy churn, resulting in more abrupt changes in the greedy policy. This not only affects the actions chosen during training but also influences the bootstrapping targets, thereby impacting the stability of the TD targets. This could explain the observed performance loss compared to smaller batches.

6.3. Discussion

The Deadly Triad in deep RL

The concept of the deadly triad, comprising function approximation, bootstrapping, and off-policy learning, has long been recognized as a potential source of instability in reinforcement learning algorithms (Sutton & Barto, 2018). This combination, when not managed carefully, can lead to divergence in value estimates and erratic learning behavior even with linear function approximators (Tsitsiklis & Van Roy, 1996; Baird et al., 1995).

Function approximation, particularly with complex architectures like deep neural networks, inherently introduces approximation errors. When bootstrapping is applied to these approximations, any errors are compounded across iterations, exacerbating divergence issues (Zhang et al., 2021; Van Hasselt et al., 2018). Off-policy learning further complicates this picture because the data distribution under which the agent learns may differ substantially from that induced by its current

policy (Kozuno et al., 2021). This divergence between data and policy has been a key focus in reinforcement learning research, motivating the development of techniques aimed at stabilizing learning while leveraging the benefits of off-policy data (Munos et al., 2016; D'Oro et al., 2022).

Deep reinforcement learning algorithms, such as Deep Q-Network (DQN) (Mnih, 2013) and its various extensions, have made significant progress in addressing the deadly triad. These algorithms incorporate mechanisms like experience replay (Li et al., 2021), target networks (Zhang et al., 2021), and carefully tuned update rules that help mitigate the instability typically induced by the triad (Schwarzer et al., 2023).

Despite these advances, the deadly triad remains a critical area of investigation in reinforcement learning research (Van Hasselt et al., 2018; Nauman et al., 2024a; Hussing et al., 2024). The interplay between theoretical insights and empirical evaluations continues to drive progress toward more robust and reliable algorithms (Gallici et al., 2024). Recently, Gallici et al. (2024) propose PQN, a purely on-policy value-based algorithm that explicitly deals with the cornerstone challenges of the deadly triad and achieves great performance empirically. PQN includes LayerNorm layers (Lei Ba et al., 2016) to stabilize function approximation, multi-step λ-returns to reduce the bootstrapping bias, and computes gradient updates only with on-policy data.

Parallel Reinforcement Learning

Parallel reinforcement learning (RL) has played a central role in scaling up learning processes by enabling faster data collection and more efficient policy updates. In these settings, multiple agents (actors) interact with environment replicas concurrently, with their experiences aggregated, either synchronously or asynchronously, to update a central learner. This decoupling improves sample efficiency, reduces temporal correlations, and accelerates convergence.

Parallel policy gradient algorithms often employ the actor-learner architecture. Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016) was a seminal work in this direction, showing that multiple asynchronous actor threads, each interacting with its own environment instance, can reduce sample correlation and improve exploration without requiring experience replay. A synchronous variant, A2C (Sutton & Barto, 2018), collects trajectories from parallel environments and performs batched updates to stabilize learning.

Later, asynchronous PPO variants combined PPO's clipped surrogate objective with parallel experience generation. IMPALA (Importance Weighted Actor-Learner Architecture) (Espeholt et al., 2018) scaled this further by decoupling acting and learning entirely: many CPU-based actors generate trajectories and send them to a GPU-based learner. To address the off-policy nature of the resulting data, IMPALA applies a V-trace correction to ensure stable learning.

These approaches share several key ideas:

- **Parallel data collection:** Multiple actors generate diverse trajectories in parallel, reducing variance in policy gradient estimates.
- **Actor-Learner architecture:** A central learner aggregates and applies gradient updates based on batches from distributed workers.
- **On-policy vs. off-policy trade-offs:** While A3C and A2C are fundamentally on-policy, IMPALA handles off-policy corrections arising from asynchronous updates.

Value-based methods have also benefitted significantly from parallelization. Distributed DQN variants such as APEX (Distributed Prioritized Experience Replay) (Horgan et al., 2018) and R2D2 (Recurrent Experience Replay in Distributed RL) (Kapturowski et al., 2018) use many actors to generate experience stored in a centralized replay buffer. The learner samples from this buffer, often with prioritization, to update the Q-network.

Recently, methods such as PQN (Parallel Q-Networks) (Gallici et al., 2024) and its variants have demonstrated that it is possible to achieve strong performance using purely on-policy data collection, even in value-based settings. In PQN, multiple actors each generate trajectories in parallel across distinct environment instances. These rollouts are then aggregated into a single large batch of on-policy experience, which is used to perform multiple gradient updates, similar to the minibatch and multi-epoch optimization procedure in PPO. After completing several epochs of training on this batch, the data is discarded, and a new set of rollouts is collected, with no persistent replay buffer or experience reuse.

Despite the lack of off-policy data reuse, PQN still achieves strong sample efficiency and competitive performance. This

finding is central to the motivation behind our work: we investigate what fundamental factors enable such efficiency in the absence of replay and explore whether value-based algorithms like PQN can be extended to support more effective data reuse. In particular, we seek mechanisms that allow more gradient steps per data point without suffering from the empirical collapse typically observed in off-policy value-based methods when data is overused.

Learning Dynamics Under High Data Reuse

A number of studies have identified several factors that impede effective data reuse in deep reinforcement learning. For example, the phenomenon of *dying neurons*, in which a large fraction of ReLU-activated neurons become inactive over time, has been reported as a significant barrier to sustained learning (Sokar et al., 2023). Concurrently, *representation collapse*, often measured by a decrease in the rank or diversity of internal features, further restricts a network's ability to capture new information and thus hinders data reuse (Lyle et al., 2024; Kumar et al., 2020). These issues limit the network's effective capacity by reducing the richness of the learned representations, ultimately leading to poorer generalization on novel data.

In addition to neuron inactivation and representational bottlenecks, *loss of plasticity* has emerged as a central challenge, where a network's ability to adapt to new data distributions gradually deteriorates (Nikishin et al., 2022). Research has shown that beyond the well-known problem of catastrophic forgetting, deep RL agents may also struggle to update their parameters effectively when overexposed to a fixed data distribution. Interventions such as plasticity injection (Nikishin et al., 2022; D'Oro et al., 2022) have been proposed to maintain plasticity by periodically reinitializing or perturbing less active units. These approaches indicate that a more dynamic strategy is needed to preserve the network's learning capability, which motivates our investigation into alternative metrics like the Data Replay Ratio (DRR) to capture the true impact of data reuse on learning dynamics.

What matters for maximizing data reuse

Our main contribution is the introduction of the Data Replay Ratio (DRR), a metric that offers a more faithful characterization of the effective data utilization in deep reinforcement learning experiments. We demonstrate that DRR captures important aspects of learning dynamics that are overlooked by the more commonly used Replay Ratio (RR). In particular, we present empirical scenarios in which different configurations, despite having identical RR values, lead to significantly different learning behaviors, which are accurately distinguished by their DRR values (see Figure 1).

Crucially, by modifying data-centric settings such as the number of parallel environments, batch size, and replay buffer capacity, we can induce variations in DRR while keeping RR constant. This allows us to isolate the effects of data reuse from other confounding factors. We observe that increasing the number of parallel environments and using larger batch sizes tends to mitigate the harmful effects of overfitting and priming in high-DRR regimes.

Additionally, DRR provides an accurate characterization of data reuse even in asynchronous settings, where the interaction between the actor and learner processes does not follow a fixed learning schedule. Instead, the rate at which data is generated and consumed varies dynamically, making traditional metrics less reliable. Our experiments with R2D2 in Figure 6 reveal key factors that influence DRR fluctuations in highly asynchronous and parallelized environments, providing insights into how data reuse evolves under such conditions.

Progress in value-based deep reinforcement learning

Recent advancements in value-based deep reinforcement learning RL have introduced a variety of new algorithms aimed at training agents to achieve superior performance in complex benchmarks. Many of these approaches address different orthogonal challenges in existing methods, leading to improvements in sample efficiency, stability, and generalization.

BBF (Schwarzer et al., 2023) enhances learning efficiency by incorporating dynamic schedules for key hyperparameters, such as the discount factor and n-step return, alongside self-predictive representations. These innovations enable BBF to achieve state-of-the-art sample efficiency, as demonstrated in the Atari100k benchmark. Similarly, BRO (Nauman et al., 2024b) improves sample efficiency in continuous control tasks by selectively scaling the critic's architecture in an actor-critic framework and applying layer normalization to the critic. Additionally, BRO incorporates mechanisms for optimistic exploration to enhance learning stability.

Further architectural innovations have been introduced by Simba (Lee et al., 2024), which leverages dynamic observation normalization, residual connections, and layer normalization to facilitate more efficient parameter scaling. Beyond architectural improvements, recent works such as ReDo (Obando-Ceron et al., 2024b) focus on fundamental learning

dynamics in value-based RL. ReDo specifically addresses the issue of dormant neurons by identifying and pruning them throughout training, resulting in sparser models that benefit from a simplicity bias, leading to improved generalization and sample efficiency.

The evolution of value-based RL methods can also be seen in the lineage from DQN (Mnih, 2013) to Rainbow (Hessel et al., 2018), and subsequently to Agent57 (Badia et al., 2020) and MEME (Kapturowski et al., 2022). These later approaches leverage learned meta-controllers to dynamically select policies for data collection. MEME further improves sample efficiency by integrating trust regions and multi-step returns with Q(λ), while also eliminating the need for a target network. However, both Agent57 and MEME are computationally expensive and currently lack accessible open-source implementations, limiting their practical adoption.

6.4. Future Work

Our study reveals several promising research directions to tackle the learning challenges identified:

Interconnections of normalization, bootstrapping instability, and temporal difference (TD) bias: Our findings suggest that while target networks can be replaced with layer normalization when using Q(0), removing them in favor of Q(λ) provides stability without performance loss. This aligns with the empirical insights of Kapturowski et al. (2022), where MEME optimizes a soft Q(λ) objective without target networks, relying on bootstrapping from the online network.

Advancing on-policy value-based algorithms: Develop methods that combine the compute efficiency of parallel algorithms without large replay buffers (like PQN) while efficiently leveraging the theoretical off-policy learning properties of value-based algorithms.

Batch size and DRR scheduling in value-based deep RL: Gain a deeper understanding of how batch size and Data Replay Ratio (DRR) affect performance. Explore how dynamically adjusting batch size—and consequently DRR-during training can balance the increased variance and exploration benefits of smaller batches with the stability, robustness, and improved sample reuse efficiency offered by larger batches. This opens the door to adaptive scheduling strategies that optimize data reuse and performance throughout learning.

Open-source implementations of state-of-the-art algorithms: Provide open-source implementations of competitive algorithms like MEME and Agent57. In this work, we release an open-source implementation of R2D2, featuring a recurrent Q network with an LSTM module, a prioritized replay buffer, and high-throughput asynchronous training. Since MEME and Agent57 are built on the R2D2 infrastructure, our implementation serves as a foundation for further open-source development of these algorithms. Additionally, it enables the analysis of DRR properties in asynchronous settings, facilitating a better understanding of these algorithms.

Improving convolutional architectures: Address the emergence of dormant neurons in convolutional layers, which our results show that significantly influence the task performance of value-based algorithms. This may explain the empirical success of adopting the ImpalaCNN residual architecture in deep RL (Castro et al., 2018; Clark et al., 2024). Further investigation is needed to understand the performance gains offered by different convolutional architectures such as squeeze-and-excitation networks (Hu et al., 2018), which have been utilized in algorithms like MEME.

6.5. R2D2 Open Source Implementation

R2D2 (Kapturowski et al., 2018) trains a recurrent Q-function using an LSTM network to extract representations that capture historical information. To train with off-policy data, R2D2 introduces a burn-in strategy to recover from stale LSTM states in the replay buffer. Additionally, R2D2 operates in a distributed setting, similar to using parallel environments but with the added complexity of asynchronous communication.

The state-of-the-art performance of R2D2 reported by Kapturowski et al. (2018) required 1e10 environment steps and 2M gradient updates, taking few days to complete the training program thanks to the extensive compute resources used. Note that a single-environment implementation of DQN (Mnih, 2013) can achieve better performance than R2D2 given a fixed budget of environment interaction steps (e.g. 200M frames) but it can take up to 4 full days of training (Castro et al., 2018). The latter illustrates the trade-off between sample-efficiency and wall-clock time in the asynchronous settings mentioned in the paper.

Similar to the original R2D2 implementation, we use batches of 64 rollouts during training, containing rollouts of length 80, along with a burn-in period of 40 steps. Kapturowski et al. (2018) reported an approximate speed of 65k simulation steps per second for data collection and 5 learning steps per second. In contrast, our best-performing R2D2 configuration achieves around 5k steps per second for data collection and 7 learning steps per second, likely resulting in a significantly higher DRR than the original implementation.

A significant difference between our implementation of R2D2 compared to the original one in Kapturowski et al. (2018) (other than the available computational resources used during training) is the multi-step TD method used for computing target returns. In our work, we used λ-returns due to consistency reasons as that is the same objective we use for all other discovered rollout algorithms (e.g. DQN-Rollout, Vec-DQN-Rollout, PQN). Kapturowski et al. (2018) used n-step returns instead (Sutton & Barto, 2018).

Table 1. Hyperparameters for R2D2.

Parameter	Value
num_envs	32
buffer_size	1,000,000
batch_size	64
rollout_length	80
start_e	1
end_e	0.01
exploration_fraction	0.1
learning_rate	2.5e-4
learning_starts	80,000
q_lambda	0.65
gamma	0.99
tau	1.0
target_network_frequency (learner steps)	500
burn_in_lstm_length	40
per_eta	0.9
per_alpha	0.9
per_beta	0.6

6.6. Benchmark

We evaluate the best-performing R2D2 configuration in Figure 6 and PQN on both the 8 representative ALE games used throughout the paper and the Atari-10 suite (Aitchison et al., 2023), training for the standard 200M environment steps (see Appendix 6.1 for details). We also include the DQN-NoTarget+LN variant, referred to as Stable DQN in Figure 2, which demonstrated superior performance among the Q(0)-based methods (see Figure 5). This evaluation, summarized in Table 2, provides valuable context for comparing the performance of various value-based baselines. Additionally, we provide the benchmark results in the Atari-10 in Table 3. In the 8 representative ALE games, PQN outperforms the other algorithms in 6 out of the 8 environments and is approximately nine times faster to train. These results position the newly proposed PQN algorithm as a highly competitive and efficient baseline for advancing value-based deep RL. For R2D2, the differences between our results and those reported by Kapturowski et al. (2018) can be attributed to variations in the training budget (200M vs. 1e10M frames) and the number of computational resources used (32 vs. 256 CPU actors). The results on the Atari-10 benchmark (Aitchison et al., 2023) are presented in Table 3. Notably, our R2D2 implementation achieves the highest performance in 3 games of the Atari-10 despite differences in compute availability, while PQN and DQN-NoTarget+LN achieve the best results in 5 and 2 games, respectively.

Table 2. Scores on the 8 representative ALE games (200M frames)

Environment	PQN	DQN-NoTarget+LN	R2D2
Asterix	**331,062**	226,677	9,000
Breakout	**518,46**	393,62	422,89
DemonAttack	171,992	**976,311**	50,912
MsPacman	**4,480**	2,609	2,734
Pong	**20.86**	**20.74**	**20.55**
Qbert	**22,900**	18,683	14,841
Seaquest	**54,489**	10,803	20,770
SpaceInvaders	**30,427**	2,522	11,341
DRR	2	8	~ 4
Training Time (hours)	2.69	26.34	20.07

Table 3. Atari-10 scores (200M frames)

Environment	PQN	R2D2	DQN-NoTarget+LN
Amidar	1,409	**1,614**	1,063
BattleZone	58,140	**68,000**	29,340
Bowling	**60.86**	49.75	41.81
DoubleDunk	-8.48	**3.00**	-19.39
Frostbite	4,058	4,650	**7,712**
KungFuMaster	19,114	26,200	**26,880**
NameThisGame	**18,784**	2,540	11,877
Phoenix	**71,933**	4,116	22,629
Qbert	**22,900**	18,683	14,841
Riverraid	**24,985**	7,830	16,618
DRR	4	4	8
Training Time (hours)	2.69	24.07	26.34

6.7. Extended Evaluation in Continuous Control Tasks

For this study, our choice of DQN, PQN, and R2D2 is directly motivated by the recent introduction of PQN, a value-based algorithm designed to be sample-efficient and stable while entirely avoiding the use of a replay buffer, thus disallowing data reuse. Interestingly, despite PQN being theoretically off-policy, there is no strong justification for completely discarding the replay buffer. This observation naturally motivates our exploration: can we improve over PQN by explicitly reintroducing and controlling data reuse?

To assess the generality of our claims beyond discrete-action environments, we conducted additional experiments in continuous control domains using SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018), and DDPG (Lillicrap et al., 2015) across the standard benchmarks of Hopper, Walker2d, and Humanoid, averaging the results over 3 independent runs. These experiments serve two main purposes:

- Validate whether the core phenomenon observed in our main analysis, namely that high data reuse (HDRR) degrades learning performance, extends to continuous control settings.
- Test the explanatory power of DRR versus RR in these domains, and whether the learning collapse under HDRR is similarly observed and mitigated by appropriate data regimes (e.g., larger batch sizes, more parallel environments).

Interestingly, TD3 exhibits a strong capacity for data reuse, benefitting from training configurations that lead to higher DRR values. However, interpreting the learning dynamics remains challenging. For instance, in the Humanoid environment, high DRR (HDRR) settings with vectorized environments and large training batches perform similarly to low DRR settings with small training batches. Conversely, the vectorized version with small batches performs as poorly as the non-vectorized version with larger batches. Despite these variations, the configuration with the highest DRR generally yields the best performance for TD3 across these environments. This observation motivates further investigation into the effects of delayed updates and double clipping mechanisms in TD3, which may help maximize data reuse.

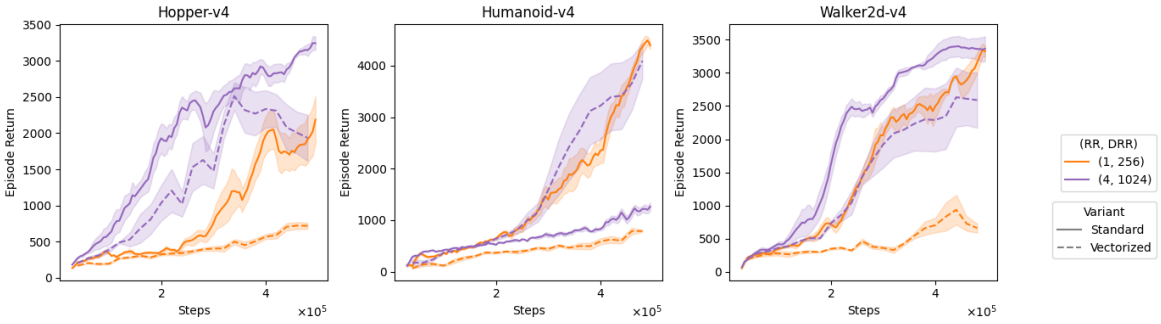


Figure 10. DRR vs. performance in TD3 across the Hopper, Walker2d, and Humanoid environments.

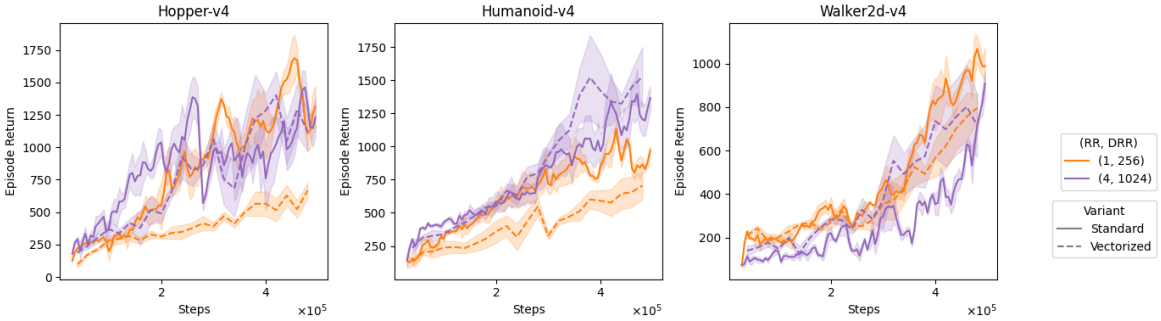


Figure 11. DRR vs. performance in DDPG across the Hopper, Walker2d, and Humanoid environments.

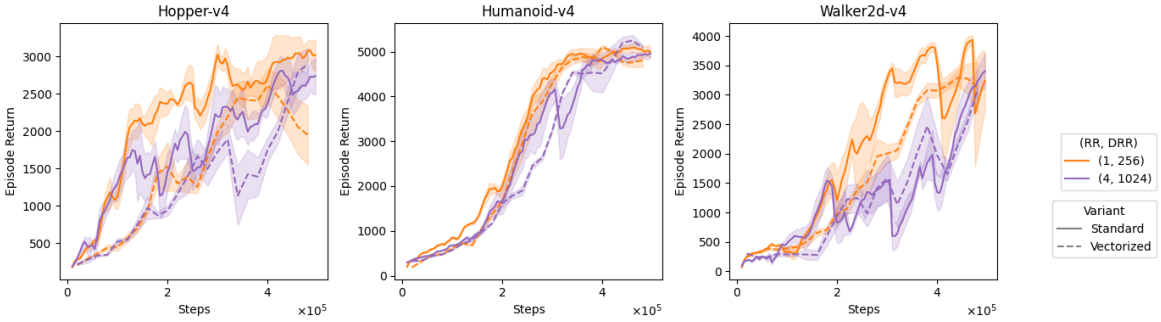


Figure 12. DRR vs. performance in SAC across the Hopper, Walker2d, and Humanoid environments.

Figure 13. DRR vs. performance in continuous control tasks (Hopper, Walker2d, Humanoid) for different algorithms: TD3, DDPG, and SAC.

For SAC and DDPG, the results are inconclusive and highly dependent on the specific environment. As mentioned earlier, the main focus of our paper is to trace the algorithmic decisions that led to the development of modern algorithms like PQN, which deploy deep value-based methods in purely on-policy settings, achieving strong performance across various baselines. In doing so, we specifically targeted discrete control problems. However, the findings presented in this section also provide valuable insights that can inspire future research on purely on-policy and highly regularized algorithms for continuous control tasks.

6.8. Algorithmic Variants

Algorithm	Vectorized	Target Net	LayerNorm	TD Target	Replay Buffer	Double Q-Learning	LSTM
DQN	✗	✓	✗	Q(0)	✓	✗	✗
Double DQN	✗	✓	✗	Q(0)	✓	✓	✗
Rainbow	✗	✓	✗	N-step	✓	✓	✗
PQN	✓	✗	✓	Q(λ)	✗	✗	✗
R2D2	✓	✓	✗	N-step	✓	✓	✓
DQN-Rollout	✗	✓	✗	Q(λ)	✓	✗	✗
Stable-DQN-Rollout	✗	✗	✓	Q(λ)	✓	✗	✗
Vec-DQN	✓	✓	✗	Q(0)	✓	✗	✗
Stable-Vec-DQN	✓	✗	✓	Q(0)	✓	✗	✗
Vec-DQN-Rollout	✓	✓	✗	Q(λ)	✓	✗	✗
Stable-Vec-DQN-Rollout	✓	✗	✓	Q(λ)	✓	✗	✗

Table 4. Component-wise configuration of the value-based algorithms studied in this work across architectural and algorithmic dimensions.

6.9. PQN Hyperparameters

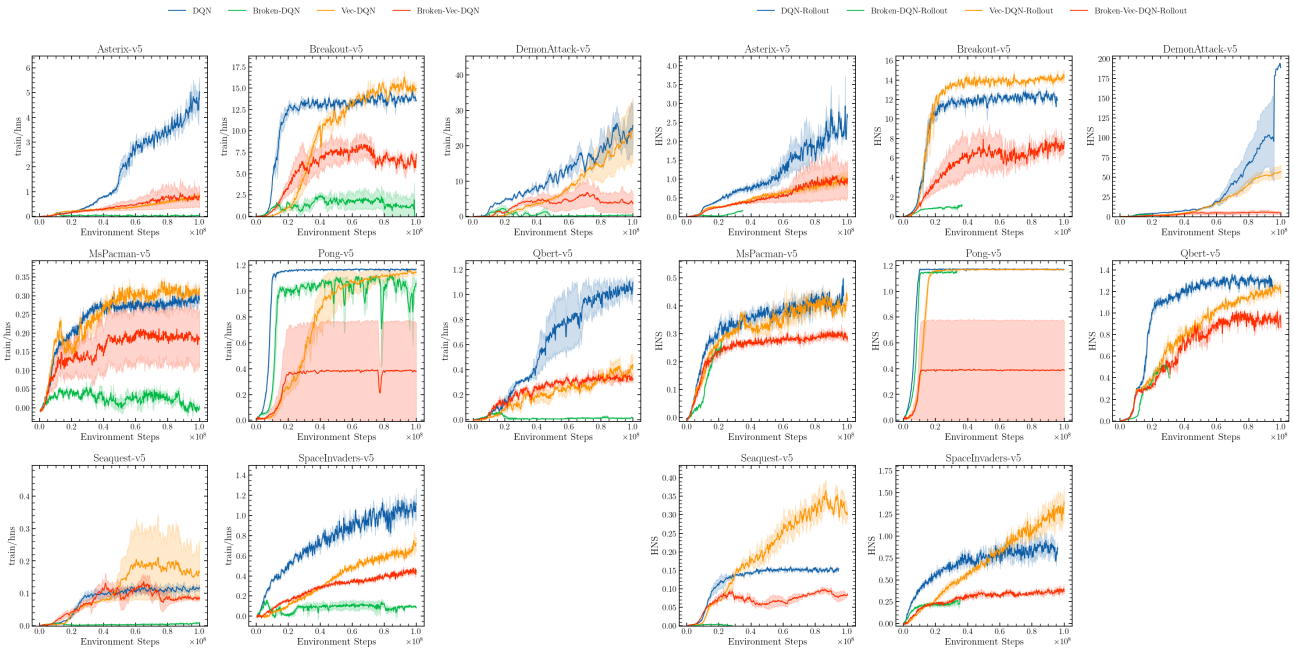
We use the implementation of PQN provided in CleanRL[2] and the hyperparameters proposed by Gallici et al. (2024).

Table 5. Hyperparameters for PQN.

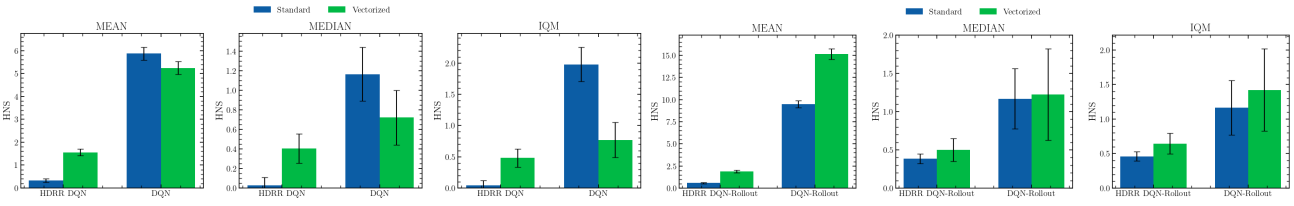
Parameter	Value
learning_rate	2.5e-4
num_envs	128
num_steps	32
anneal_lr	True
gamma	0.99
num_minibatches	32
update_epochs	2
max_grad_norm	10.0
start_e	1
end_e	0.01
exploration_fraction	0.10
q_lambda	0.65

[2] https://github.com/vwxyzjn/cleanrl/blob/master/cleanrl/pqn_atari_envpool.py

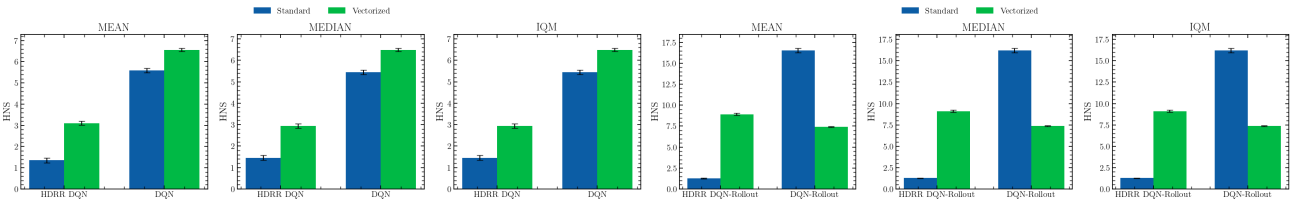
6.10. Learning Curves



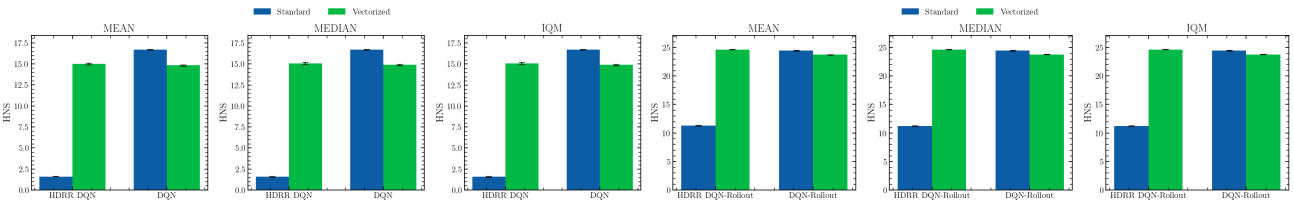
(a) Q(0) (b) Q(λ)



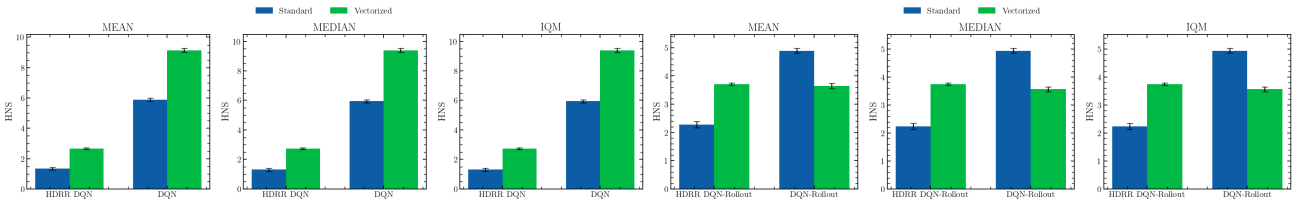
(a) ALE/8 - Q(0) (b) ALE/8 - Q(λ)



(a) DeadlyCorridor - Q(0) (b) DeadlyCorridor - Q(λ)



(a) DefendTheCenter - Q(0) (b) DefendTheCenter - Q(λ)



(a) Deathmatch - Q(0) (b) Deathmatch - Q(λ)

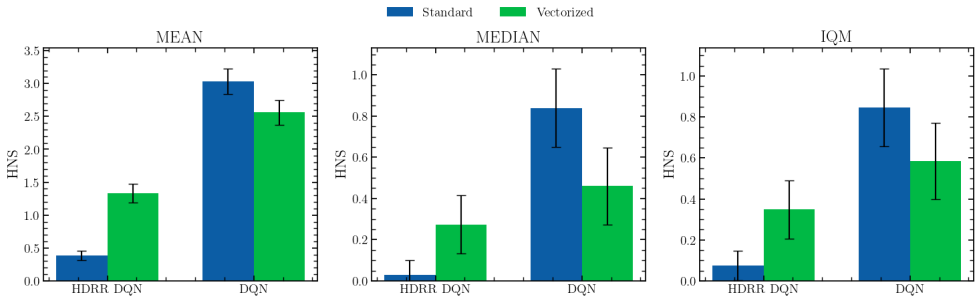
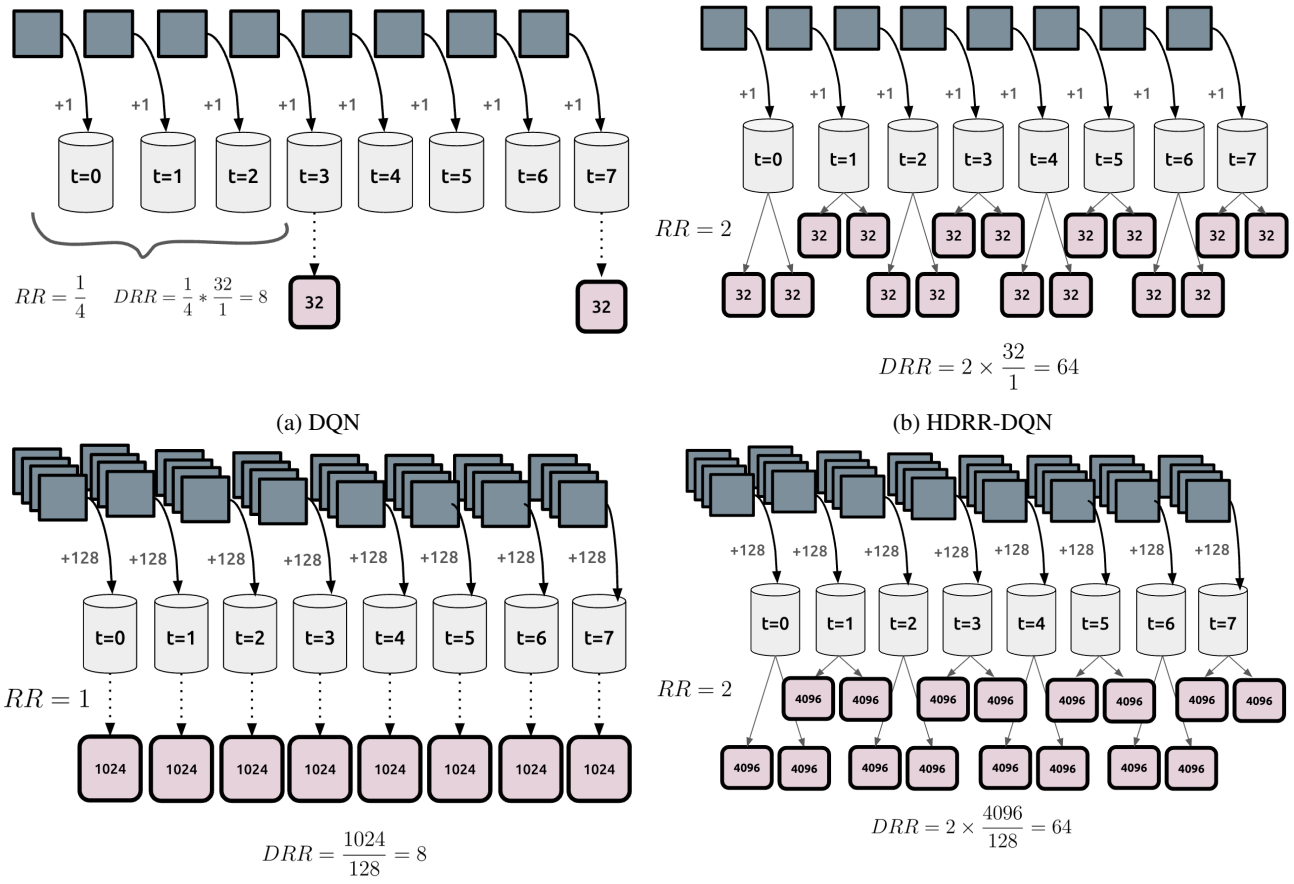


Figure 19. Evaluation of Double DQN (DDQN) in Atari10. While DDQN is designed to mitigate the TD bootstrapping bias and potentially improve data reuse, our results for DDQN show similar behavior to the DQN results in Figure 3. Specifically, small batch sizes remain highly sensitive to HDRR settings, often resulting in complete collapse. This issue can be mitigated by using vectorized environments and larger batch sizes. However, smaller batches can offer performance improvements due to their high variance updates, which may facilitate better exploration within the parameter space.

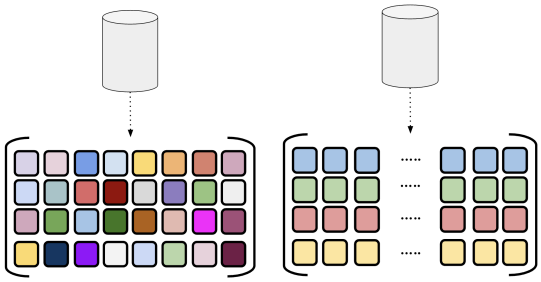


(a) DQN (b) HDRR-DQN

(c) Vec-DQN (d) HDRR-Vec-DQN

Figure 20. Illustration of the DRR across different data sampling configurations. Gray squares represent environment observations, where *Vec-* denotes multiple environments contributing observations to a shared buffer. Red squares depict learning batches.

6.11. Diagrams



(a) Q(0) (b) Q(λ)

Figure 21. Different data sampling procedures when training with Q(0) vs Q(λ).