Bigger, Better, Faster: Human-level Atari with human-level efficiency

Max Schwarzer 1 2 3 Johan Obando-Ceron 1 2 3 Aaron Courville 2 3 Marc G. Bellemare 1 2 3
Rishabh Agarwal 1 2 3 Pablo Samuel Castro 1

Abstract

We introduce a value-based RL agent, which we call BBF, that achieves super-human performance in the Atari 100K benchmark. BBF relies on scaling the neural networks used for value estimation, as well as a number of other design choices that enable this scaling in a sample-efficient manner. We conduct extensive analyses of these design choices and provide insights for future work. We end with a discussion about updating the goalposts for sample-efficient RL research on the ALE. We make our code and data publicly available.

1. Introduction

Deep reinforcement learning (RL) has been central to a number of successes including playing complex games at a human or super-human level, such as OpenAI Five (Berner et al., 2019), AlphaGo (Silver et al., 2016), and AlphaStar (Vinyals et al., 2019), controlling nuclear fusion plasma in a tokomak (Degrave et al., 2022), and integrating human feedback for conversational agents (Ouyang et al., 2022). The success of these RL methods has relied on large neural networks and an enormous number of environment samples to learn from – a human player would require tens of thousands of years of game play to gather the same amount of experience as OpenAI Five or AlphaGo. It is plausible that such large networks are necessary for the agent’s value estimation and/or policy to be expressive enough for the environment’s complexity, while large number of samples might be needed to gather enough experience so as to determine the long-term effect of different action choices as well as train such large networks effectively. As such, obtaining human-level sample efficiency with deep RL remains an outstanding goal.

Although advances in modern hardware enable using large networks, in many environments it may be challenging to scale up the number of environment samples, especially for real-world domains such as healthcare or robotics. While approaches such as offline RL leverage existing datasets to reduce the need for environment samples (Agarwal et al., 2020), the learned policies may be unable to handle distribution shifts when interacting with the real environment (Levine et al., 2020) or may simply be limited in performance without online interactions (Ostrovski et al., 2021). Thus, as RL continues to be used in increasingly challenging and sample-scarce scenarios, the need for scalable yet sample-efficient online RL methods becomes more pressing.

Figure 1: Environment samples to reach human-level performance, in terms of IQM (Agarwal et al., 2021b) over 26 games. Our proposed model-free agent, BBF, results in $5 \times$ improvement over SR-SPR (D’Oro et al., 2023) and at least $16 \times$ improvement over representative model-free RL methods, including DQN (Mnih et al., 2015b), Rainbow (Hessel et al., 2017) and IQN (Dabney et al., 2018). To contrast with the sample-efficiency progress in model-based RL, we also include DreamerV2 (Hafner et al., 2020), MuZero Reanalyse (Schrittwieser et al., 2021) and EfficientZero (Ye et al., 2021).

In this vein, we focus on the Atari 100K benchmark (Kaiser et al., 2017).
The RL problem is generally described as a Markov Decision Process (MDP) (Puterman, 2014), defined by the tuple \((S, A, P, R)\), where \(S\) is the set of states, \(A\) is the set of available actions, \(P: S \times A \rightarrow \Delta(S)\) is the transition function, and \(R: S \times A \rightarrow \mathbb{R}\) is the reward function. Agent behavior in RL can be formalized by a policy \(\pi: S \rightarrow \Delta(A)\), which maps states to a distribution of actions. The value of \(\pi\) when starting from \(s \in S\) is defined as the discounted sum of expected rewards: \(V^\pi(s) := \mathbb{E}_{\pi, P, R} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]\), where \(\gamma \in [0, 1)\) is a discount factor that encourages the agent to accumulate rewards sooner rather than later. The goal of an RL agent is to find a policy \(\pi^*\) that maximizes this sum: \(V^{\pi^*} \geq V^\pi\) for all \(\pi\).

While there are a number of valid approaches (Sutton & Barto, 1998), in this paper we focus on model-free value-based methods. Common value-based algorithms approximate the \(Q^*\)-values, defined via the Bellman recurrence: \(Q^*(s, a) := R(s, a) + \gamma \mathbb{E}_{s', \sim P(s, a)} Q^*(s', a')\). The optimal policy \(\pi^*\) can then be obtained from the optimal state-action value function \(Q^*\) as \(\pi^*(s) := \max_{a \in A} Q^*(s, a)\). A common approach for learning \(Q^*\) is the method of temporal differences, optimizing the Bellman temporal difference:

\[
\left( r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \right) - Q(s_t, a_t).
\]

We often refer to \(\left( r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \right)\) as the Bellman target.

Mnih et al. (2015a) introduced the agent DQN by combining temporal-difference learning with deep networks, and demonstrated its capabilities in achieving human-level performance on the Arcade Learning Environment (ALE) (Bellemare et al., 2013). They used a network consisting of 3 convolutional layers and 2 fully connected layers, parameterized by \(\theta\), to approximate \(Q\) (denoted as \(Q_\theta\)). We will refer to this architecture as the CNN architecture. Most of the work in value-based agents is built on the original DQN architecture.

2. Background

The RL problem is generally described as a Markov Decision Process (MDP) (Puterman, 2014), defined by the tuple \((S, A, P, R)\), where \(S\) is the set of states, \(A\) is the set of available actions, \(P: S \times A \rightarrow \Delta(S)\) is the transition function, and \(R: S \times A \rightarrow \mathbb{R}\) is the reward function. Agent behavior in RL can be formalized by a policy \(\pi: S \rightarrow \Delta(A)\), which maps states to a distribution of actions. The value of \(\pi\) when starting from \(s \in S\) is defined as the discounted sum of expected rewards: \(V^\pi(s) := \mathbb{E}_{\pi, P, R} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]\), where \(\gamma \in [0, 1)\) is a discount factor that encourages the agent to accumulate rewards sooner rather than later. The goal of an RL agent is to find a policy \(\pi^*\) that maximizes this sum: \(V^{\pi^*} \geq V^\pi\) for all \(\pi\).

Figure 2: Comparing Atari 100K performance and computational cost of our model-free BBF agent to model-free SR-SPR (D’Oro et al., 2023), SPR (Schwarzer et al., 2021), DrQ (eps) (Kostrikov et al., 2020) and DER (Van Hasselt et al., 2019) as well as model-based” EfficientZero (Ye et al., 2021) and IRIS (Micheli et al., 2023). (Left) BBF achieves higher performance than all competitors as measured by interquartile mean human-normalized over 26 games. Error bars show 95% bootstrap CIs. (Right) Computational cost vs. Performance, in terms of human-normalized IQM over 26 games. BBF results in 2× improvement in performance over SR-SPR with nearly the same computational-cost, while results in similar performance to model-based EfficientZero with at least 4× reduction in runtime. For measuring runtime, we use the total number of A100 GPU hours spent per environment.
Figure 3: Scaling network widths for both ResNet and CNN architectures, for BBF, SR-SPR and SPR at replay ratio 2, with an Impala-based ResNet (left) and the standard 3-layer CNN (Mnih et al., 2015b) (right). We report interquantile mean performance with error bars indicating 95% confidence intervals. On the x-axis we report the approximate parameter count of each configuration as well as its width relative to the default (width scale = 1).

agent, and we discuss a few of these advances below which are relevant to our work.

Hessel et al. (2018) combined six components into a single agent they called Rainbow: prioritized experience (Schaul et al., 2016), n-step learning (Sutton, 1988), distributional RL (Bellemare et al., 2017), double Q-learning (van Hasselt et al., 2016), dueling architecture (Wang et al., 2016) and NoisyNets (Fortunato et al., 2018b). Hessel et al. (2018) and Ceron & Castro (2021) both showed that Multi-step learning is one of the most crucial components of Rainbow, in that removing it caused a large drop in performance.

In n-step learning, instead of computing the temporal difference error using a single-step transition, one can use n-step targets instead (Sutton, 1988), where for a trajectory \((s_0, a_0, r_0, s_1, a_1, \cdots)\) and update horizon \(n\): \(R_t^{(n)} := \sum_{k=0}^{n-1} \gamma^k r_{t+k+1}\), yielding the multi-step temporal difference: \(R_t^{(n)} + \gamma^n \max_{a'} Q_{\theta}(s_{t+n}, a') - Q_{\theta}(s_t, a_t)\).

Most modern RL algorithms store past experiences in a replay buffer that increases sample efficiency by allowing the agent to use samples multiple times during learning, and to leverage modern hardware such as GPUs and TPUs by training on sampled mini-batches. An important design parameter is the replay ratio, the ratio of learning updates to online experience collected (Fedus et al., 2020a). It is worth noting that DQN uses a replay ratio of 0.25 (4 environment interactions for every learning update), while some sample-efficient agents based on Rainbow use a value of 1.

Nikishin et al. (2022) showed that the networks used by deep RL agents have a tendency to overfit to early experience, which can result in sub-optimal performance. They proposed a simple strategy consisting of periodically resetting the parameters of the final layers of DQN-based agents to counteract this. Building on this promising work, D’Oro et al. (2023) added a shrink-and-perturb technique for the parameters of the convolutional layers, and showed that this allowed them to scale the replay ratio to values as high as 16, with no performance degradation.

3. Related Work

Sample-Efficient RL on ALE: Sample efficiency has always been an import aspect of evaluation in RL, as it can often be expensive to interact with an environment. Kaiser et al. (2020) introduced the Atari 100K benchmark, which has proven to be useful for evaluating sample-efficiency, and has led to a number of recent advances.

Kostrikov et al. (2020) use data augmentation to design a sample-efficient RL method, DrQ, which outperformed prior methods on Atari 100K. Data-Efficient Rainbow (DER) (Van Hasselt et al., 2019) and DrQ(\(\epsilon\)) (Agarwal et al., 2021b) simply modified the hyperparameters of existing model-free algorithms to exceed the performance of existing methods without any algorithmic innovation.

Schwarzer et al. (2021) introduced SPR, which builds on Rainbow (Hessel et al., 2017) and uses a self-supervised tem-
EfficientZero (Ye et al., 2021), an efficient variant of MuZero (Schrittwieser et al., 2020), learns a discrete-action latent dynamics model from environment interactions, and selects actions via lookahead MCTS in the latent space of the model. Micheli et al. (2023) introduce IRIS, a data-efficient agent that learns in a world model composed of an autoencoder and an auto-regressive Transformer.

Scaling in Deep RL: Deep neural networks are useful for extracting features from data relevant for various downstream tasks. Recently, there has been interest in the scaling properties of neural network architectures, as scaling model size has led to commensurate performance gains in applications ranging from language modelling to computer vision.

Based on those promising gains, the deep RL community has begun to investigate the effect of increasing the model size of the function approximator. Sinha et al. (2020) and Ota et al. (2021) explore the interplay between the size, structure, and performance of deep RL agents to provide intuition and guidelines for using larger networks. Kumar et al. (2022) find that with ResNets (up to 80 million parameter networks) combined with distributional RL and feature normalization, offline RL can exhibit strong performance that scales with model capacity. Taiga et al. (2023) show that generalization capabilities on the ALE benefit from higher capacity networks, such as ResNets. Cobbe et al. (2020) and Farebrother et al. (2023) demonstrate benefits when scaling the number of features in each layer of the ResNet architecture used by Impala (Espeholt et al., 2018), which motivated the choice of feature width scaling in this work. Different from these works, our work focus on improving sample-efficiency in RL as opposed to offline RL or improving generalization in RL.

In the context of online RL, Hafner et al. (2023) demonstrate that increased dynamics model size, trained via supervised learning objectives, leads to monotonic improvements in the agent’s final performance. Recently, AdA (Team et al., 2023) scales transformer encoder for a Muesli agent up to 265M parameters. Interestingly, AdA required distillation from smaller models to bigger models to achieve this scaling, in the spirit of reincarnating RL (Agarwal et al., 2022). However, it is unclear whether findings from above papers generalize to scaling typical value-based deep RL methods in sample-constraint settings, which we study in this work.

4. Method

The question driving this work is: How does one scale networks for deep RL when samples are scarce? To investigate this, we focus on the well-known Atari 100K benchmark (Kaiser et al., 2020), which includes 26 Atari 2600 games of diverse characteristics, where the agent may perform only 100K environment steps, roughly equivalent to two hours of human gameplay\(^2\). As we will see, naively scaling networks can rarely maintain performance, let alone improve it.

The culmination of our investigation is the Bigger, Better,
BBF: Human-level Atari with human-level efficiency

Faster agent, or BBF in short, which achieves super-human performance on Atari 100K with about 6 hours on single GPU. Figure 2 demonstrates the strong performance of BBF relative to some of the best-performing Atari 100K agents: EfficientZero (Ye et al., 2021), SR-SPR (D’Oro et al., 2023), and IRIS (Micheli et al., 2023). BBF consists of a number of components, which we discuss in detail below.

Our implementation is based on the Dopamine framework (Castro et al., 2018) and uses mostly already previously-released components. For evaluation, we use reliable (Agarwal et al., 2021b) and in particular, the interquartile mean (IQM) metric, which is the average score of the middle 50% runs combined across all games and seeds.

**Base agent.** BBF uses a modified version of the recently introduced SR-SPR agent (D’Oro et al., 2023). Through the use of periodic network resets, SR-SPR is able to scale up its replay ratio (RR) to values as high as 16, yielding better sample efficiency. For BBF, we use RR=8 in order to balance the increased computation arising from our large network. Note that this is still very high relative to existing Atari agents – Rainbow and its data-efficient variant DER (Van Hasselt et al., 2019) use RR=0.25 and 1, respectively.

As we expect that many users will not wish to pay the computational costs of running at replay ratio 8, we also present results for BBF and ablations at replay ratio 2 (matching SPR). For all experiments we state which replay ratio is being used in the captions.

**Harder resets.** The original SR-SPR agent (D’Oro et al., 2023) used a shrink-and-perturb method for the convolutional layers where parameters were only perturbed 20% of the way towards a random target, while later layers were fully reset to a random initialization. An interesting result of our investigation is that using harder resets of the convolutional layers yields better performance. In our work, we move them 50% towards the random target, resulting in a stronger perturbation and improving results (see Figure 5). This may be because larger networks need more regularization, as we find that they reduce loss faster (Figure A.1).

**Larger network.** Scaling network capacity is one of the motivating factors for our work. As such, we adopt the Impala-CNN (Espeholt et al., 2018) network, a 15-layer ResNet, which has previously led to substantial performance gains over the standard 3-layer CNN architecture in Atari tasks where large amounts of data are available (Agarwal et al., 2022; Schmidt & Schmied, 2021). Additionally, BBF scales the width of each layer in Impala-CNN by 4×. In Figure 3, we examine how the performance of SPR, SR-SPR and BBF varies with different choices of scaling width, for both the ResNet and original CNN architectures. Interestingly, although the CNN has roughly 50% more parameters than the ResNet at each scale level, the ResNet yields better performance at all scaling levels for both SR-SPR and BBF.

What stands out from Figure 3 is that BBF’s performance continues to grow as width is increased, whereas SR-SPR seems to peak at 1-2× (for both architectures). Given that ResNet BBF performs comparably at 4× and 8×, we chose 4× to reduce the computational burden. While reducing widths beyond this could further reduce computational costs, this comes at the cost of increasingly sharp reductions in performance for all methods tested.

**Receding update horizon.** One of the surprising components of BBF is the use of an update horizon (n-step) that decreases exponentially from 10 to 3 over the first 10K gradient steps following each network reset. Given that we follow the schedule of D’Oro et al. (2023) and reset every

---

**Figure 5:** Evaluating the impact of removing the various components that make up BBF with RR=2 and RR=8. Reporting interquantile mean averaged over the 26 Atari 100k games, with 95% CIs over 15 independent runs.
40k gradient steps, the annealing phase is always 25% of training, regardless of the replay ratio. As can be seen in Figure 5, this yields a much stronger agent than using a fixed value of $n = 3$, which is default for Rainbow, or $n = 10$, which is typically used by Atari 100K agents like SR-SPR.

Our $n$-step schedule is motivated by the theoretical results of Kearns & Singh (2000) – larger values of $n$-step leads to faster convergence but to higher asymptotic errors with respect to the optimal value function. Thus, selecting a fixed value of $n$ corresponds to a choice between having either rapid convergence to a worse asymptote, or slower convergence to a better asymptote. As such, our exponential annealing schedule closely resembles the optimal decreasing schedule for $n$-step derived by Kearns & Singh (2000).

Increasing discount factor. Motivated by findings that increasing the discount factor $\gamma$ during learning improves performance (François-Lavet et al., 2015), we increase $\gamma$ from $\gamma_1$ to $\gamma_2$, following the same exponential schedule as for the update horizon. Note that increasing $\gamma$ has the effect of progressively giving more weights to delayed rewards. We choose $\gamma_1 = 0.97$, slightly lower than the typical discount used for Atari, and $\gamma_2 = 0.997$ as it is used by MuZero (Schrittwieser et al., 2021) and EfficientZero (Ye et al., 2021). As with the update horizon, Figure 5 demonstrates that this strategy outperforms using a fixed value.

Weight decay. We incorporate weight decay in our agent to curb statistical overfitting, as BBF is likely to overfit with its high replay ratio. To do so, we use the AdamW optimizer (Loshchilov & Hutter, 2019) with a weight decay value of 0.1. Figure 5 suggests the gains from adding weight decay are significant and increase with replay ratio, indicating that the regularizing effects of weight decay enhance replay ratio scaling with large networks.

Removing noisy nets. Finally, we found that NoisyNets (Fortunato et al., 2018a), used in the original SPR (Schwarzer et al., 2021) and SR-SPR, did not improve performance. This could be due to NoisyNets causing over-exploration due to increased policy churn (Schaul et al., 2022) from added noise during training, or due to added variance in optimization, and we leave investigation to future work. Removing NoisyNets results in large computational and memory savings, as NoisyNets creates duplicate copies of the weight matrices for the final two linear layers in the network, which contain the vast majority of all parameters: turning on NoisyNets increases the FLOPs per forward pass and the memory footprint by a factor of $2.5 \times$ and $1.6 \times$, respectively, which both increases runtime and reduces the number of training runs that can be run in parallel on a single GPU. Removing NoisyNets is thus critical to allowing BBF to achieve reasonable compute efficiency despite its larger networks. We found that this decision had no significant impact on task performance (see Figure A.2 in appendix).

5. Analysis

In light of the importance of BBF’s components, we discuss possible consequences of our findings for other algorithms.

The importance of self-supervision. One unifying aspect of the methods compared in Figure 2 is that they all use some form of self-supervised objective. In sample-constrained scenarios, like the one considered here, relying on more than the temporal-difference backups is likely to improve learning speed, provided the self-supervised losses are consistent with the task at hand. We test this by removing the SPR objective (inherited from SR-SPR) from BBF, and observe a substantial performance degradation (see Figure 5). It is worth noting that EfficientZero uses a self-supervised objective that is extremely similar to SPR, a striking commonality between BBF and EfficientZero.
Figure 8: Validating BBF design choices at RR=2 on 29 unseen games. While Atari 100K training set consists of 26 games, we evaluate the performance of various components in BBF on 29 validation games in ALE that are not in Atari 100K. Interestingly, all BBF components lead to a large performance improvement on unseen games. Specifically, we measure the % decrease in human-normalized IQM performance relative to the full BBF agent at RR=2.

Sample efficiency via more gradient steps. The original DQN agent (Mnih et al., 2015b) has a replay ratio of 0.25, which means a gradient update is performed only after every 4 environment steps. In low-data regimes, it is more beneficial to perform more gradient steps, although many algorithms cannot benefit from this without additional regularization (D’Oro et al., 2023). As Figure 6 confirms, performance of BBF grows with increasing replay ratio in the same manner as its base algorithm, SR-SPR. More strikingly, we observe a linear relationship between the performance of BBF and SR-SPR across all replay ratios, with BBF performing roughly 0.45 IQM above SR-SPR. While the direction of this relationship is intuitive given the network scaling introduced by BBF, its linearity is unexpected, and further investigation is needed to understand the nature of the interaction between replay ratio and network scaling.

One interesting comparison to note is that, although EfficientZero uses a replay ratio of 1.2, they train with a batch size that is 8 times larger than ours. Thus, their effective replay ratio is comparable to ours.

The surprising importance of target networks. Many prior works on Atari 100k, such as DrQ and SPR (Kostrikov et al., 2020; Schwarzer et al., 2021) chose not to use target networks, seeing them unnecessary or an impediment to sample efficiency. Later, D’Oro et al. (2023) re-introduced an exponential moving average target network, used both for training and action selection, and found that it improved performance somewhat, especially at high replay ratios. With network scaling, however, using a target network becomes a critical, but easy-to-overlook, component of the algorithm at all replay ratios (see Figure 7).

6. Revisiting the Atari 100k benchmark

A natural question is whether there is any value in continuing to use the Atari 100K benchmark, given that both EfficientZero and BBF are able to achieve human-level perfor-
Figure 10: **Sample efficiency progress on ALE**, measured via human-normalized IQM over 55 Atari games with sticky actions, as a function of amount of human gameplay hours, with BBF at RR=8. Shaded regions show 95% CIs.

**Overfitting on Atari 100K.** Another important consideration is that the Atari 100K benchmark uses only 26 of the 55 games from the full ALE suite, and it does not include sticky actions\(^1\) (Machado et al., 2018), which may make tasks significantly harder. Since we extensively benchmark BBF on Atari 100K, this raises the question of whether BBF works well on unseen Atari games and with sticky actions.

Fortunately, it does. In Figure 9, we compare the performance of BBF on all 55 games with sticky actions, and show that sticky action do not significantly harm performance. We do observe that the held-out games not included in the Atari 100k set are significantly more challenging than the 26 Atari 100k games (see Figure 11) – but this is even more true for baselines such as DQN (Nature) that did not use sticky actions at RR=8 for 100k steps approximately matches DQN (Nature) with 500 times more training data on each set. While we find that the 29 games not included in the Atari 100k setting are significantly harder than the 26 Atari 100k games, we see no evidence that BBF has overfitted to Atari 100k compared to DQN.

**Data Scaling** Prior works have indicated that many sample-efficient RL algorithms plateau in performance when trained for longer than they were originally designed for (e.g., Agarwal et al., 2022). To examine this phenomenon, we train BBF, SPR and SR-SPR at replay ratio 2 out to one million environment steps (Figure 12), keeping all parameters unchanged (including conducting resets as normal past 100k steps). We observe that SPR and SR-SPR experience stagnating performance, with SR-SPR’s advantage over SPR fading by 1M steps. BBF, however, remains consistently ahead of both, matching DQN’s final performance before 200k environment steps and matching Rainbow’s performance at 20M environment steps by 1M steps. We note that this experiment costs only 2.5 times more than training at replay ratio 8 to 100k steps, so we encourage other researchers to run similar experiments.

Additionally, we note in Figure 13 that it is possible to compare algorithms even with extremely small amounts of data, such as 20k or 50k steps, by which point BBF at replay ratio 2 (even with sticky actions enabled) outperforms most recently proposed algorithms (Robine et al., 2023; Micheli et al., 2023; Hafner et al., 2023), which did not use sticky actions. We thus suggest that compute-constrained groups consider this setting, as training BBF at replay ratio 2 for 40k environment steps takes only half of an A100 for 1 hour.

---

\(^1\)With 25% probability, the environment will execute the previous action again, instead of the agent’s executed action.
7. Discussion and Future Work

We introduced BBF, an algorithm that is able to achieve super-human level performance on the ALE with only 2-hours of gameplay. Although BBF is not the first to achieve this milestone, it is able to do so in a computationally efficient manner. Furthermore, BBF is able to better handle the scaling of networks and replay ratios, which are crucial for network expressivity and learning efficiency. Indeed, Figure 3 suggests that BBF is better-able to use over-parameterized networks than prior agents.

The techniques necessary to achieve this result invite a number of research questions for future work. Large replay ratios are a key element of BBF’s performance, and the ability to scale them is due to the periodic resets incorporated into the algorithm. These resets are likely striking a favourable balance between catastrophic forgetting and network plasticity. An interesting avenue for future research is whether there are other mechanisms for striking this balance that perhaps are more targeted (e.g. not requiring resetting the full network, as was recently explored by Sokar et al. (2023)).

We remarked on the fact that all the methods compared in Figure 2 use a form of self-supervision. Would other self-supervised losses (e.g. (Mazoure et al., 2020; Castro et al., 2021; Agarwal et al., 2021a)) produce similar results? Surprisingly, Li et al. (2022) argue that self-supervision from pixels does not improve performance; our results seem to contradict this finding.

Recent attention has shifted towards more realistic benchmarks (Fan et al., 2022) but such benchmarks exclude the majority of researchers outside certain resource-rich labs, and may require an alternative paradigm (Agarwal et al., 2022). One advantage of the Atari 100k benchmark is that, while still a challenging benchmark, it is relatively cheap compared to other benchmarks of similar complexity. However, despite its apparent saturation, scientific progress can still be made on this benchmark if we expand its scope. We hope our work provides a solid starting point for this.

Overall, we hope that our work inspires other researchers to continue pushing the frontier of sample efficiency in deep RL forward, to ultimately reach human-level performance across all tasks with human-level or superhuman efficiency.
References


Acknowledgements. Many thanks to Ross Goroshin, Georg Ostrovski, and Gopeshh Subbaraj for their feedback on an earlier draft of this paper. The authors would like to thank the anonymous reviewers for useful discussions and feedback on this paper. We would also like to thank the Python community (Van Rossum & Drake Jr, 1995; Oliphant, 2007) for developing tools that enabled this work, including NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007) and JAX (Bradbury et al., 2018).

Societal impact. Although the work presented here is mostly academic, it aids in the development of more capable autonomous agents. While our contributions do not directly contribute to any negative societal impacts, we urge the community to consider these when building on our research.

A. Additional Results

<table>
<thead>
<tr>
<th>Games</th>
<th>Random</th>
<th>Human</th>
<th>DER</th>
<th>DrQ(ε)</th>
<th>SPR</th>
<th>IRIS</th>
<th>SR-SPR</th>
<th>EfficientZero</th>
<th>BBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alien</td>
<td>227.8</td>
<td>7127.7</td>
<td>802.3</td>
<td>865.2</td>
<td>841.9</td>
<td>420.0</td>
<td>1107.8</td>
<td>808.5</td>
<td>1173.2</td>
</tr>
<tr>
<td>Amidar</td>
<td>5.8</td>
<td>1719.5</td>
<td>125.9</td>
<td>137.8</td>
<td>179.7</td>
<td>143.0</td>
<td>203.4</td>
<td>148.6</td>
<td>244.6</td>
</tr>
<tr>
<td>Assault</td>
<td>222.4</td>
<td>742.0</td>
<td>561.5</td>
<td>579.6</td>
<td>565.6</td>
<td>1524.4</td>
<td>1088.9</td>
<td>1263.1</td>
<td>2098.5</td>
</tr>
<tr>
<td>Asterix</td>
<td>210.0</td>
<td>8503.3</td>
<td>535.4</td>
<td>763.6</td>
<td>962.5</td>
<td>853.6</td>
<td>903.1</td>
<td>25557.8</td>
<td>3946.1</td>
</tr>
<tr>
<td>BankHeist</td>
<td>14.2</td>
<td>753.1</td>
<td>185.5</td>
<td>232.9</td>
<td>345.4</td>
<td>53.1</td>
<td>531.7</td>
<td>351.0</td>
<td>732.9</td>
</tr>
<tr>
<td>BattleZone</td>
<td>2360.0</td>
<td>37187.5</td>
<td>8977.0</td>
<td>10165.3</td>
<td>14834.1</td>
<td>13074.0</td>
<td>17671.0</td>
<td>13871.2</td>
<td>24459.8</td>
</tr>
<tr>
<td>Boxing</td>
<td>0.1</td>
<td>12.1</td>
<td>-0.3</td>
<td>9.0</td>
<td>35.7</td>
<td>70.1</td>
<td>45.8</td>
<td>52.7</td>
<td>85.8</td>
</tr>
<tr>
<td>Breakout</td>
<td>1.7</td>
<td>30.5</td>
<td>9.2</td>
<td>19.8</td>
<td>19.6</td>
<td>83.7</td>
<td>25.5</td>
<td>414.1</td>
<td>370.6</td>
</tr>
<tr>
<td>ChopperCommand</td>
<td>811.0</td>
<td>7387.8</td>
<td>925.9</td>
<td>844.6</td>
<td>946.3</td>
<td>1565.0</td>
<td>2362.1</td>
<td>1117.3</td>
<td>7549.3</td>
</tr>
<tr>
<td>CrazyClimber</td>
<td>10780.5</td>
<td>35289.4</td>
<td>34508.6</td>
<td>21539.0</td>
<td>59324.2</td>
<td>45544.1</td>
<td>83940.2</td>
<td>58431.8</td>
<td>13341.4</td>
</tr>
<tr>
<td>DemonAttack</td>
<td>152.1</td>
<td>1971.0</td>
<td>627.6</td>
<td>1321.5</td>
<td>517.6</td>
<td>2034.4</td>
<td>2814.4</td>
<td>13003.9</td>
<td>13341.4</td>
</tr>
<tr>
<td>Freeway</td>
<td>0.0</td>
<td>29.6</td>
<td>20.9</td>
<td>20.3</td>
<td>19.3</td>
<td>31.1</td>
<td>25.4</td>
<td>21.8</td>
<td>25.5</td>
</tr>
<tr>
<td>Frostbite</td>
<td>65.2</td>
<td>4334.7</td>
<td>871.0</td>
<td>1014.2</td>
<td>1170.7</td>
<td>259.1</td>
<td>2584.8</td>
<td>296.3</td>
<td>2384.8</td>
</tr>
<tr>
<td>Gopher</td>
<td>257.6</td>
<td>2412.5</td>
<td>467.0</td>
<td>621.6</td>
<td>660.6</td>
<td>2236.1</td>
<td>712.4</td>
<td>3260.3</td>
<td>1331.2</td>
</tr>
<tr>
<td>Hero</td>
<td>1027.0</td>
<td>30826.4</td>
<td>6226.0</td>
<td>4167.9</td>
<td>5858.6</td>
<td>7037.4</td>
<td>8524.0</td>
<td>9315.9</td>
<td>7818.6</td>
</tr>
<tr>
<td>Jamesbond</td>
<td>29.0</td>
<td>302.8</td>
<td>275.7</td>
<td>349.1</td>
<td>366.5</td>
<td>462.7</td>
<td>389.1</td>
<td>517.0</td>
<td>1129.6</td>
</tr>
<tr>
<td>Kangaroo</td>
<td>52.0</td>
<td>3035.0</td>
<td>581.7</td>
<td>1088.4</td>
<td>3617.4</td>
<td>838.2</td>
<td>3631.7</td>
<td>724.1</td>
<td>6614.7</td>
</tr>
<tr>
<td>Krull</td>
<td>1598.0</td>
<td>2665.5</td>
<td>3256.9</td>
<td>4402.1</td>
<td>3681.6</td>
<td>6616.4</td>
<td>5911.8</td>
<td>5663.3</td>
<td>8223.4</td>
</tr>
<tr>
<td>KungFuMaster</td>
<td>258.5</td>
<td>22736.3</td>
<td>6580.1</td>
<td>11467.4</td>
<td>14783.2</td>
<td>21759.8</td>
<td>18649.4</td>
<td>30944.8</td>
<td>18991.7</td>
</tr>
<tr>
<td>MsPacman</td>
<td>307.3</td>
<td>6951.6</td>
<td>1187.4</td>
<td>1218.1</td>
<td>1318.4</td>
<td>999.1</td>
<td>1574.1</td>
<td>1281.2</td>
<td>2008.3</td>
</tr>
<tr>
<td>Pong</td>
<td>-20.7</td>
<td>14.6</td>
<td>-9.7</td>
<td>-9.1</td>
<td>-5.4</td>
<td>14.6</td>
<td>2.9</td>
<td>20.1</td>
<td>16.7</td>
</tr>
<tr>
<td>PrivateEye</td>
<td>24.9</td>
<td>69571.3</td>
<td>72.8</td>
<td>3.5</td>
<td>86.0</td>
<td>100.0</td>
<td>97.9</td>
<td>96.7</td>
<td>40.5</td>
</tr>
<tr>
<td>Qbert</td>
<td>163.9</td>
<td>13455.0</td>
<td>1773.5</td>
<td>1810.7</td>
<td>866.3</td>
<td>745.7</td>
<td>4044.1</td>
<td>14448.5</td>
<td>4447.1</td>
</tr>
<tr>
<td>Roadrunner</td>
<td>11.5</td>
<td>7845.0</td>
<td>11843.4</td>
<td>11211.4</td>
<td>12213.1</td>
<td>9614.6</td>
<td>13463.4</td>
<td>17751.3</td>
<td>33426.8</td>
</tr>
<tr>
<td>Seaquest</td>
<td>68.4</td>
<td>42054.7</td>
<td>304.6</td>
<td>352.3</td>
<td>558.1</td>
<td>661.3</td>
<td>819.0</td>
<td>1100.2</td>
<td>1232.5</td>
</tr>
<tr>
<td>UpNDown</td>
<td>533.4</td>
<td>11693.2</td>
<td>3075.0</td>
<td>4324.5</td>
<td>10859.2</td>
<td>3546.2</td>
<td>112450.3</td>
<td>17264.2</td>
<td>12101.7</td>
</tr>
</tbody>
</table>

Games > Human

IQM (↑) 0.000 1.000 0.183 0.280 0.337 0.501 0.631 0.917 12
Optimality Gap (↓) 1.000 0.000 0.698 0.631 0.577 0.512 0.433 0.371 0.344
Median (↑) 0.000 1.000 0.189 0.313 0.396 0.289 0.685 1.116 0.917
Mean (↑) 0.000 1.000 0.350 0.465 0.616 1.046 1.272 1.945 2.247

Table A.1: Scores and aggregate metrics for BBF and competing methods across the 26 Atari 100k games. Scores are averaged across 50 seeds per game for BBF, 30 for SR-SPR, 5 for IRIS, 3 for EfficientZero, and 100 for others.
Figure A.1: Learning curves for BBF and SR-SPR at RR=2 with a ResNet encoder at various width scales, on the 26 Atari 100k games. Larger networks consistently have lower TD errors and higher gradient norms, and higher parameter norms, but only BBF translates this to higher environment returns. The large, systematic difference in TD error between BBF and SR-SPR is due to BBF’s use of a shorter update horizon, which makes each step of the TD backup easier to predict.
Figure A.2: BBF at RR=2 on the 26 Atari 100k tasks, with and without Noisy Nets.