# Fast and Data-Efficient Training of Rainbow: an Experimental Study on Atari

**Dominik Schmidt**
TU Wien
e11809917@student.tuwien.ac.at

**Thomas Schmied**
TU Wien
e1553816@student.tuwien.ac.at

## Abstract

Across the Arcade Learning Environment, Rainbow achieves a level of performance competitive with humans and modern RL algorithms. However, attaining this level of performance requires large amounts of data and hardware resources, making research in this area computationally expensive and use in practical applications often infeasible. This paper's contribution is threefold: We (1) propose an improved version of Rainbow, seeking to drastically reduce Rainbow's data, training time, and compute requirements while maintaining its competitive performance; (2) we empirically demonstrate the effectiveness of our approach through experiments on the Arcade Learning Environment, and (3) we conduct a number of ablation studies to investigate the effect of the individual proposed modifications. Our improved version of Rainbow reaches a median human normalized score close to classic Rainbow's, while using 20 times less data and requiring only 7.5 hours of training time on a single GPU. We also provide our full implementation including pre-trained models.

## 1 Introduction

In 2013, the Deep Q-Networks (DQN) algorithm [Mnih et al., 2013] kicked off a flurry of developments in the classic Atari reinforcement learning (RL) benchmark. In this benchmark, the agent is tasked with learning to play Atari 2600 games solely based on pixel inputs and rewards. While recent methods such as Agent57, R2D2, NGU, and MuZero [Badia et al., 2020a, Kapturowski et al., 2019, Badia et al., 2020b, Schrittwieser et al., 2020] can now achieve above-human level performance in most or all games, a number of critical issues remain. Among these are the often vast data, time, and compute requirements for training these agents. Agent57, the only RL algorithm to achieve super-human performance on all 57 Atari games, required in the order of a century (78 billion frames) of gameplay experience in order to beat the last game, Atari Skiing [Badia et al., 2020a].

Rainbow [Hessel et al., 2018], introduced in 2017 and itself based on DQN, represents an important milestone in the development of the above-mentioned agents, acting as a foundation for Agent57 and other algorithms [Badia et al., 2020a, Kapturowski et al., 2019]. In the past, Rainbow has also served as a useful baseline for the development of new environments [Cobbe et al., 2020, Nichol et al., 2018]. In this paper, we revisit the Rainbow algorithm and review some of the original design choices. By combining Rainbow with some more recent techniques and by simplifying the implementation, we seek to strike a good balance between performance, data efficiency, training time, and implementation complexity.

Aside from the Atari benchmark, reinforcement learning has further achieved remarkable accomplishments in other domains, such as beating the human champions in some of the world's most difficult board games [Silver et al., 2016, 2017, 2018], as well as in strategic real-time combat video games [Vinyals et al., 2019b, Berner et al., 2019]. Other prominent demonstrations of the effectiveness of

Deep Reinforcement Learning Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021).

RL are optimizing the energy consumption of data centers [Lazic et al., 2018], dexterous robotic manipulation of physical objects [Akkaya et al., 2019, Andrychowicz et al., 2020], and designing more efficient chip layouts [Mirhoseini et al., 2020] and neural architectures [Zoph and Le, 2017, Pham et al., 2018]. In these domains too, however, RL-based approaches can suffer from the issues of lacking data efficiency and requiring large amounts of compute resources, among others. Indeed, some of the most remarkable results achieved in these tasks required up to hundreds or even thousands of years worth of simulated environment interaction experience [Vinyals et al., 2019b,a, Berner et al., 2019, Akkaya et al., 2019]. This severely limits the applicability of RL to many real-world tasks, where gathering such large amounts of experience is often infeasible, and the required hardware resources impose a substantial financial burden.

One prominent stream of research that addresses these issues is data-efficient RL. This area refers to a set of algorithmic improvements to the RL agent that aim to enable it to learn from fewer environment interaction steps and shorten the training time the RL agent requires in order to learn the desired behavior. In recent years, this area has started to receive lots of attention within the RL community. Therefore, we discuss the literature on data-efficient RL in more detail in Section 2.

Other, more practical approaches attempt to explicitly shorten the time required to reach the learning objective by speeding up the RL training process itself. This can be achieved, for example, by employing simple tricks that are commonly used in other areas of machine learning but have previously rarely been used in RL. These include techniques to improve hardware utilization, mixed-precision training [Micikevicius et al., 2018] and methods that aim to stabilize the training process.

## 1.1 Contributions

In this work, we integrate techniques from both streams of research into the established Rainbow agent while re-evaluating a number of Rainbow's original design choices. Among others, we investigate what effect a larger batch size, a bigger network, a higher learning rate, spectral normalization, and mixed precision have on the training speed and the RL agent's overall performance. While most of these tricks/enhancements have been studied individually in the context of RL, to the best of our knowledge, our work is the first to study them jointly. We discuss their appearance in the literature in more detail in Section 2.

To study these improvements, we rely on the well-established Atari benchmark. The Arcade Learning Environment (ALE) has long been an important challenge for RL research [Bellemare et al., 2013, Machado et al., 2018] and has particularly entered the spotlight since the introduction of Deep Q-Networks (DQN) [Mnih et al., 2013, 2015]. DQN combines Q-Learning with deep convolutional neural networks (CNNs) and experience replay to learn to play Atari games at human-level performance, solely from pixels and with no prior human knowledge. Since its inception, a variety of enhancements to the original DQN architecture have been proposed [van Hasselt et al., 2016, Wang et al., 2016, Schaul et al., 2016, Sutton and Barto, 2018, Fortunato et al., 2018, Bellemare et al., 2017], many of which have since been integrated into a single unified algorithm, Rainbow-DQN [Hessel et al., 2018]. The introduction of Rainbow represents a critical milestone in the development of RL algorithms, and it has ever since been one of the best-performing and most established algorithms for Atari.

Overall, we make the following contributions in this paper:

- We propose an improved version of Rainbow, seeking to reduce wall-clock training time and required hardware resources while improving data efficiency and maintaining competitive performance.
- We empirically demonstrate the effectiveness of our approach on the Atari benchmark, achieving a median human normalized score close to that of classic Rainbow-DQN while using 20 times less data.
- We conduct a number of ablation studies to investigate the effect of the proposed modifications individually and in aggregate.

## 2 Related Work

This section covers some recent directions of research in deep reinforcement learning that are orthogonal to our work.

### 2.1 Data-efficient RL: the algorithmic solution

One prominent stream of research related to our work is data-efficient RL. Data-efficient RL aims to learn from fewer environment interaction steps by leveraging the information at the agent's disposal more effectively. Recently, this has become an attractive area of research, as current RL algorithms can be incredibly data-inefficient. In fact, data inefficiency has been identified as one of the major obstacles towards widespread adoption of RL in the real world [Dulac-Arnold et al., 2019, 2020]. Different solutions have been proposed to mitigate this limitation. Three of the most promising ones are (1) self-supervised learning, (2) model-based RL, and (3) data augmentation.

**Self-supervised learning.** Self-supervised methods in RL aim to learn more effective representations of the environment by optimizing a self-supervised auxiliary objective in addition to the primary RL objective. Self-supervised learning (SSL) is the driving force behind recent advances in NLP [Devlin et al., 2018, Brown et al., 2020], computer vision [Chen et al., 2020, Grill et al., 2020] and speech recognition [Baevski et al., 2020]. Only recently, SSL has started to receive widespread attention from the RL community and shown to improve data efficiency considerably. Prominent examples are Unreal [Jaderberg et al., 2017], CURL [Laskin et al., 2020a], SPR [Schwarzer et al., 2020], and SGI [Schwarzer et al., 2021].

**Model-based methods.** Model-based RL algorithms, in contrast, learn an explicit model of the environment and then use this model to derive a good policy by simulating experience, a process known as planning [Sutton and Barto, 2018]. A major advantage, again, is more data-efficient learning, but these methods have historically proven difficult in environments with large and complex observation spaces [Moerland et al., 2020]. However, Schrittwieser et al. [2020] recently introduced MuZero, a model-based RL algorithm that achieves state-of-the-art performance on the Arcade Learning Environment [Bellemare et al., 2013]. Other prominent examples of model-based RL methods include World Models [Ha and Schmidhuber, 2018], SimPle [Kaiser et al., 2019] and Dreamer [Hafner et al., 2019, 2020]. A detailed overview of the model-based RL literature is given by Moerland et al. [2020].

**Data augmentation.** Another option to improve data efficiency in RL is data augmentation, a well-established technique in many ML domains, most notably for image-based tasks [Perez and Wang, 2017, Shorten and Khoshgoftaar, 2019]. Data augmentation aims to artificially increase the amount of available training data by modifying existing observations in some meaningful way. For RL, this simple technique has been shown to improve data efficiency considerably, as demonstrated by RAD [Laskin et al., 2020b] and DrQ [Yarats et al., 2020].

### 2.2 Speeding up RL training: the practical solution

Both the approaches discussed in this section, as well as the data-efficient methods from the previous section, deal with the same fundamental question: how to accelerate the training of reinforcement learning agents. On the one hand, data-efficient methods implicitly shorten the training time by cutting down the number of required training samples. The methods discussed in this section, on the other hand, seek to shorten the required training time without fundamentally modifying the learning algorithm. These methods are not unique to deep RL but rather are generally applicable to many deep learning algorithms. We do not cover approaches that speed up training through parallelization since these simply trade hardware requirements for training time and thus do not alleviate the financial burden of training large models.

**Larger batch sizes.** For image-based RL tasks, agents are typically trained with relatively small batch sizes [Mnih et al., 2015, Hessel et al., 2018]. However, prior work has demonstrated that, on the procgen and ALE benchmarks, increasing the batch size can speed up the training time considerably [Cobbe et al., 2020, Stooke and Abbeel, 2018].

**Bigger networks.** Compared to other branches of deep learning, the networks employed by deep RL agents are relatively shallow [Mnih et al., 2015]. Replacing the standard network architecture with bigger networks has proven a fruitful direction in prior work [Espeholt et al., 2018, Cobbe et al., 2020, Sinha et al., 2020, Bjorck et al., 2021].

**Spectral normalization.** Recently, Gogianu et al. [2021] proposed to apply Spectral Normalization (SN), a technique that originates from the literature on Generative Adversarial Networks (GANs) [Miyato et al., 2018, Kurach et al., 2019], to the RL setting. The authors observed that a Categorical DQN agent (C51) [Bellemare et al., 2017] augmented with SN achieves similar results as a full Rainbow DQN implementation [Hessel et al., 2018].

**Mixed precision.** Another prominent and well-established option to accelerate the training of neural networks is mixed precision, initially proposed by Micikevicius et al. [2018]. In RL, Lam et al. [2019] and Björck et al. [2021] were the first to integrate low/mixed precision and saw substantial improvements in training time.

### 2.3 RL & the Atari benchmark: a lasting relationship

The Arcade Learning Environment (ALE) [Bellemare et al., 2013, Machado et al., 2018] is a lasting and indispensable element of the RL researcher's toolbox. It is also the focus of our work. Since its inception, hundreds of RL algorithms have been developed, and the achieved scores have only increased over time. The first model-free architecture to raise public interest on the Atari benchmark was Deep Q-Networks (DQN) [Mnih et al., 2013, 2015]. Several enhancements to DQN have been proposed. including Double DQN [van Hasselt et al., 2016], Dueling DQN [Wang et al., 2016], Prioritized Experience Replay [Schaul et al., 2016], Multi-step Bootstrapping [Sutton and Barto, 2018], Noisy Nets [Fortunato et al., 2018], and Distributional RL [Bellemare et al., 2017].

Rainbow DQN, the architecture we leverage in this work, integrates all these enhancements into a single unified framework and shows that they are largely compatible [Hessel et al., 2018]. A slightly modified version of Rainbow introduced by van Hasselt et al. [2019] and referred to as Data-efficient Rainbow trades off reduced computational efficiency in exchange for significantly improved data efficiency.

R2D2 [Kapturowski et al., 2019] and Agent57 [Badia et al., 2020a], distributed RL algorithms, follow in the lineage of Rainbow. Similar distributed training strategies were employed by Gorila [Nair et al., 2015], Ape-X [Horgan et al., 2018], IMPALA [Espeholt et al., 2018] and Reactor [Gruslys et al., 2018]. Like IMPALA, R2D2 also uses recurrent neural networks to take advantage of longer state histories. Agent57 builds off of the methods developed in Kapturowski et al. [2019] but combines them with two intrinsic reward-based exploration mechanisms: the curiosity-based Random Network Distillation [Burda et al., 2019] and Never Give Up exploration [Badia et al., 2020b], aimed at increasing long and short term state-space coverage, respectively. Additionally, a UCB bandit [Sutton and Barto, 2018] algorithm is used to control the amount of exploration and the value of the discount factor in each of the distributed actors.

## 3 Preliminaries

In this section, we briefly discuss the theoretical foundations of this work.

### 3.1 The RL framework

**Markov decision process.** The fundamental goal of RL is to learn how to behave in an unknown environment in order to maximize a scalar reward signal. In each step of this sequential decision-making task, the agent observes the environment's current state $s_t$, performs some action $a_t$, transitions to the next state $s_t$, and obtains a reward $r_t$ [Sutton and Barto, 2018]. We assume the standard formulation of the Markov Decision Process (MDP) with $\langle \mathcal{S}, \mathcal{A}, R, P \rangle$ where

- $\mathcal{S}, \mathcal{A}$ are state and action space, respectively
- $R \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function that assigns each transition from state $s_t$ to state $s_{t+1}$ via action $a_t$ a real valued reward $r_t = R(s_t, a_t, s_{t+1})$
- $P \colon \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the transition probability function that specifies a conditional probability $p(s_{t+1}|s_t, a_t)$ of transitioning into state $s_{t+1}$ after executing action $a_t$ in state $s_t$.

**The Goal of RL.** The goal the agents pursues is to maximize its return, $G_t = \sum_{k=t+1}^{T} R_k$, the sum of all collected rewards. To achieve this, it learns a policy $\pi$ that determines its behavior. The policy $\pi$ maps the perceived states to actions $a \sim \pi(a \mid s)$. The policy $\pi$ can be either learned directly via the policy gradient or indirectly via the Bellman equations. RL algorithms that directly learn the policy are referred to as policy-based methods whereas algorithms that learn value functions are known as value-based methods [Sutton and Barto, 2018].

**Bellman equations.** The state-value function $v_\pi(s) = \mathbb{E}_{a\sim\pi}[r(s,a) + \gamma v_\pi(s')]$ and action value function $q_\pi(s,a) = \mathbb{E}[r(s,a) + \gamma \mathbb{E}_{a'\sim\pi}[q_\pi(s',a')]]$ determine how good it is to be in a certain state and how good it is to perform a certain given a certain state, respectively [Sutton and Barto, 2018].

### 3.2 Q-learning: from tables to neural networks

**Tabular Q-learning** Q-Learning is a value-based RL algorithm that explicitly maintains a table of Q-estimates for state-action pairs [Watkins and Dayan, 1992]. More generally, it falls under temporal difference (TD) learning methods as it grounds value estimates for the current time step in estimates at future steps. The Q-estimates are repeatedly updated according to: $q(s,a) \leftarrow q(s,a) + \alpha[r + \gamma \max_a q(s',a) - q(s,a)]$. While Q-Learning works well for small problems, storing the table becomes intractable for larger state/action spaces [Sutton and Barto, 2018].

**Deep Q-Network.** DQN, proposed by Mnih et al. [2015], employ neural networks to represent the Q-table. In contrast to tabular Q-learning, DQN scales large state and action spaces. Furthermore, Mnih et al. [2015] introduced two additional enhancements to Q-Learning that are essential for good performance when using function approximation, experience replay as well as a separate target network that is updated periodically. Therefore, the learning objective becomes:

$$J(\theta) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}}[(r + \gamma \max_{a'} q_{\theta^-}(s,a') - q_\theta(s,a))^2] \tag{1}$$

where $\mathcal{D}$ represents the experience replay buffer, and $\theta$ and $\theta^-$ are the neural networks that parameterize the Q-functions.

## 4 Our Approach

Our main goals in this work are to reduce Rainbow's considerable hardware, training time, and data requirements while maintaining a similar level of performance as the original agent. We achieve these goals by combining a large Q-network architecture with an efficient high-throughput implementation as well as by accelerating and stabilizing training through extensive hyper-parameter tuning and the use of spectral normalization. Furthermore, we reduce the implementation complexity of Rainbow by removing the distributional RL component. While distributional RL was essential for good performance in Hessel et al. [2018] when training for over 40M frames, our improvements in learning speed — leading to lower overall required training time — made distributional RL less vital.

Our implementation is available at `https://github.com/schmidtdominik/Rainbow` and includes a complete and highly customizable framework for preprocessing, training, and evaluation. We further provide integrations for OpenAI's `gym`, `procgen`, and `gym-retro` environments, as well as pretrained models for all 53 tested Atari games.

In the rest of this section, we discuss the tricks we apply and describe the individual components of our approach. First, we describe our evaluation methodology. Then, we describe the network architecture we employ in this work and show how different variants thereof compare. Furthermore, we discuss how we make use of spectral normalization and again compare a few variants. Also, we address the tricks we employ to improve the hardware utilization of our agent.

### 4.1 Evaluation methodology

We evaluate our approach against the same set of 54 Atari games that were used in Hessel et al. [2018], excluding the game *Surround* as it is not available via OpenAI `gym`. To this end, we closely follow the evaluation procedure from Mnih et al. [2015] and Hessel et al. [2018]. All evaluation runs lasted for 500k frames and each individual episode was no longer than 108k frames. The only modification we made was that we performed the evaluation runs after training had concluded by periodically saving model snapshots during training and later loading them for evaluation. Lastly, we re-evaluated the best-performing snapshot for each game. Each experiment was performed with three random seeds.

Due to the high computational cost of training on the whole ALE, we limited our ablation studies to a subset of 5 games (Asterix, Beam Rider, Freeway, Seaquest, and Space Invaders) as recommended in Machado et al. [2018] and Bellemare et al. [2013]. This selection includes two human-optimal games, two score-exploitable games, and one sparse reward game [Taïga et al., 2020]. Overall, this selection intends to capture the large variety of the ALE in a small number of games.

We encountered one issue with comparability that occurs since Hessel et al. [2018] trained their agent for 200M frames while we only trained for 10M frames: the frequency of evaluation snapshots is the same (every 1M frames), but the total number is not. This means that for some games where the performance fluctuates wildly during training, taking a larger number of agent snapshots could increase the likelihood of taking at least one snapshot at a point of good performance. To support this hypothesis, we separately stored and evaluated twice the number of agent snapshots (every 500k training frames) and observed an approximately 10% increase in apparent performance. This suggests that evaluation results from agents trained on the ALE and evaluated on a lower number of snapshots than previous research may underestimate the true performance of their agent.

## 4.2 Larger and deeper Q-Network Architecture

First, we replace the small dueling network architecture, as introduced in Mnih et al. [2013] and combined with dueling DQN in Wang et al. [2016], with the both larger and deeper IMPALA CNN [Espeholt et al., 2018]. More specifically, we employ the large variant of the IMPALA CNN (see Figure 1 in the supplementary material) with twice the number of channels, as modified in Cobbe et al. [2020]. We additionally add a size $6 \times 6$ adaptive max-pooling layer between the network's convolutional and fully connected parts. This simple yet powerful modification makes it straightforward to use our implementation with inputs of different resolutions (such as games provided by `procgen` or `gym-retro`) without affecting the number of parameters in the network. Like Cobbe et al. [2020] and Espeholt et al. [2018], we found that using this architecture substantially increased learning speed (both in terms of wall-clock time and training steps), sample efficiency, and final overall performance.

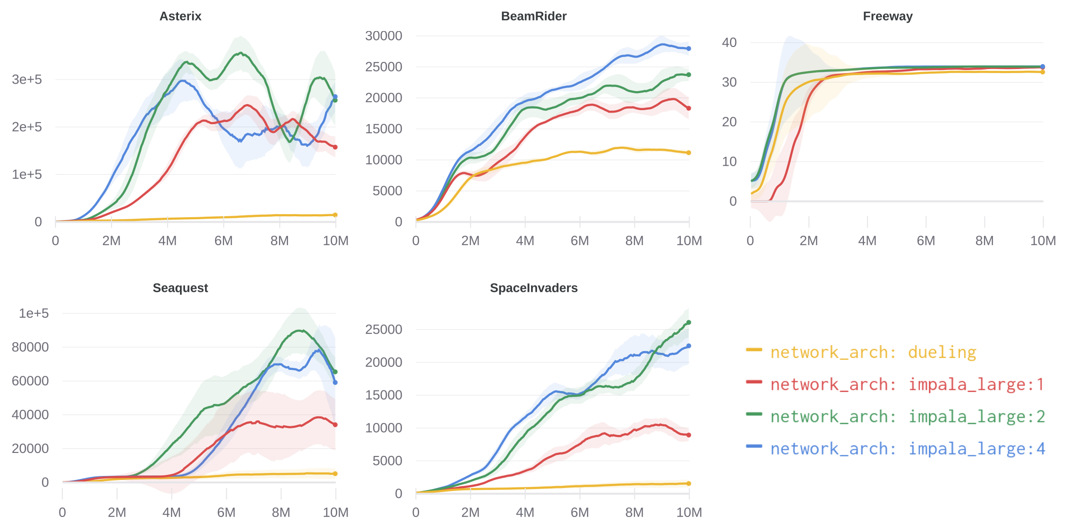

**Figure 1:** Comparison of the different network architectures. *Dueling* refers to the "Nature" dueling architecture used in Hessel et al. [2018]. Each curve shows the median over three seeds of the 100-episode running average of episode returns as a function of environment interactions.

In our experiments, we compared values of 1, 2 and 4 for the channel multiplier. The results of these experiments are shown in Figure 1. We found that a value of 2 yielded a good trade-off between data and computational efficiency. Consequently, we selected this network architecture for our subsequent runs.

### 4.3 Spectral Normalization

Spectral normalization (SN), commonly used to stabilize the training of discriminators in GANs [Miyato et al., 2018], is a method for controlling the Lipschitz constant of linear operators such as convolutional or dense linear layers in neural networks. Spectral normalization can be efficiently approximated via the power iteration method [Miyato et al., 2018, Gogianu et al., 2021].

In our experiments, we compared three different variants of applying spectral normalization to the convolutional layers in the IMPALA CNN:

- *none* – no spectral normalization is performed.
- *all* – SN is applied to both convolutional layers in all six residual blocks.
- *last* – SN is applied to all convolutional layers in the final two residual blocks.

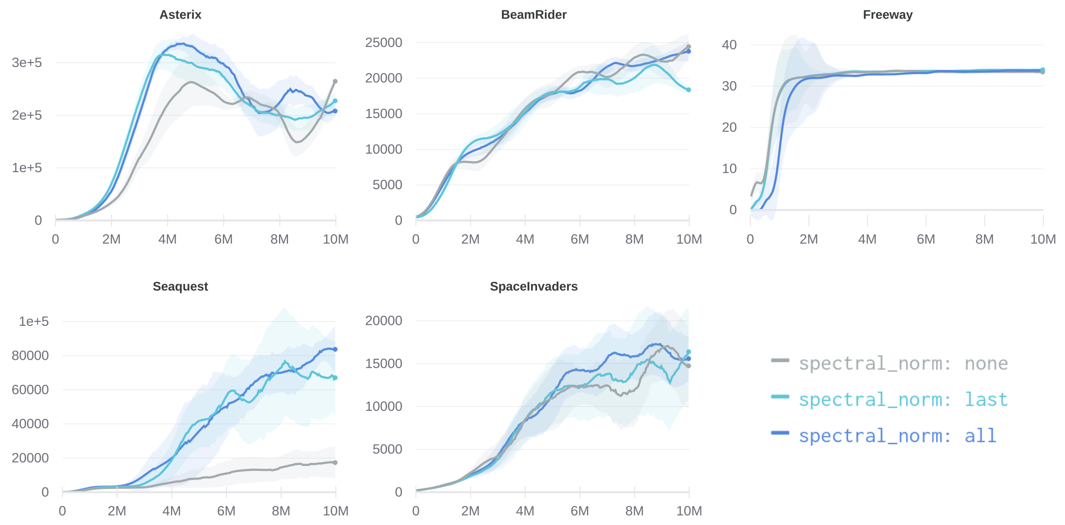

**Figure 2:** Comparison of the three variants of spectral normalization. Each curve shows the median over three seeds of the 100-episode running average of episode returns as a function of environment interactions.

Furthermore, we experimented with a fourth variant that applied SN to the final (noisy) linear layers only. However, we quickly dismissed this variant due to bad overall performance. We hypothesize that the drop in performance resulted from interference between spectral normalization and Noisy-Nets DQN exploration.

Figure 2 shows the comparison of the three variants of spectral normalization over five games. Even though the *last* variant was marginally faster in terms of training throughput, we eventually settled on the *all* variant as it slightly outperformed the former.

Overall, we observed that spectral normalization has the largest effect at the beginning of training across a number of games. It significantly reduces the time until initial learning progress is made. This effect was most strongly observable in games such as *Breakout*, *Seaquest* and *Tennis* where initial progress can be particularly slow.

### 4.4 Improving Hardware Utilization

We additionally employ several practical modifications to the training process that aim to decrease the wall-clock training time — and thus the required computational budget — by maximizing hardware utilization and training throughput:

- Similarly to Cobbe et al. [2020], we increase the batch size from 32 to 256. As suggested in Stooke and Abbeel [2018], we accordingly adjust the $\epsilon$ hyper-parameter for the Adam optimizer to $0.005/b$, where $b$ is the batch size.
- Environment interactions need to be computed sequentially and can thus not be parallelized across the time dimension. Thus, to batch environment simulation, we maintain $e = 64$ instances of the environment and take one step of this vectorized environment for every $k = 2$ training steps. In our implementation, the parameters $b$ and $e$ can be freely chosen so as to maximize GPU and CPU utilization and the agent's performance. The average number of times each environment transition is sampled from the replay buffer $\frac{bk}{e} = 8$ remains fixed by setting $k$ accordingly.
- We perform environment simulation steps and training steps in parallel to minimize idle time on the GPU.
- We use mixed-precision training as provided by PyTorch's `amp` package.

In aggregate, these modifications decrease the training time by a factor of 3.2, from 24 to approximately 7.5 hours, for training on 10M frames on a single Nvidia RTX 3090 GPU. Interestingly, the increased batch size also considerably improves the performance on several games. Prior work has observed a similar effect and attributed it to decreased gradient variance and thus more stable training behavior [Cobbe et al., 2020]. Figure 3 shows the individual contributions of the modifications compared to baseline.

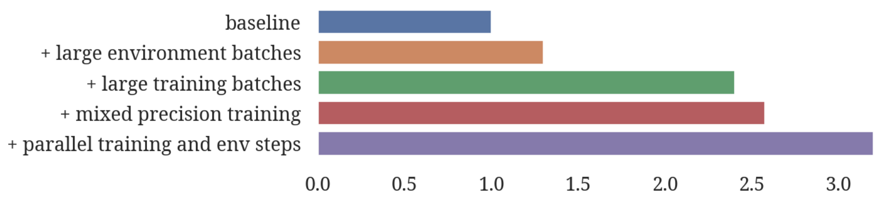

**Figure 3:** Cumulative improvement in training throughput compared to the baseline.

## 4.5 Hyperparameter Tuning

Apart from the batch size, network size, and choice of spectral normalized layers, we also performed a hyper-parameter search over learning rates and compared the Huber and mean-squared error (MSE) loss functions.

Unlike Obando-Ceron and Castro [2021] we did not observe better performance through the use of the MSE loss. However, the search over learning rates did prove fruitful. Increasing the learning rate by a factor of 4 (to the same value used in DQN) significantly speeds up learning in our experiments, both with the increased batch size and without. In addition, it considerably improves the final performance the agent achieves. We hypothesize that the reason Hessel et al. [2018] settled on the lower learning rate was that the benefit of better convergence when using a lower learning rate outweighed the disadvantage of slower learning when training for much longer periods of time.

The full set of hyperparameters is listed in Appendix A in the supplementary material.

## 4.6 Distributional RL

Finally, in our experiments removing the distributional RL component from Rainbow did not significantly affect our agent's final performance or learning speed when trained for 10M frames. This is in line with observations from the ablation studies in Hessel et al. [2018] that showed the main benefit of distributional RL materialized only after training for about 40M frames. Thus, considering the negligible performance improvement and substantial implementation complexity, we decided not to include distributional RL in our final agent.

Similarly, the evaluation curves in Hessel et al. [2018] indicate that other components of Rainbow such as Noisy-Nets DQN and Dueling DQN may only be important when training for long periods. This may suggest that — given future improvements in data efficiency, leading to lower required training time — it might be possible to further simplify the Rainbow agent with no penalties to performance, in particular when working with a limited computational budget.

Additionally, we investigated replacing the C51 [Bellemare et al., 2017] variant of distributional RL with QR-DQN [Dabney et al., 2018]. However, in our experiments, we could not observe a significant difference.

## 5 Results

Table 1 compares the final aggregate scores of our approach to the ones of the original Rainbow-DQN agent, as well as to standard DQN. The full scores and learning curves for each of the 53 games are available in Table 1 and Figure 2 in the supplementary material.

Each of our training runs took approximately 7.5 hours on a single Nvidia RTX 3090 GPU or 10 hours on a single Nvidia 2080Ti GPU. These requirements make performing larger numbers of experiments feasible, even with a more modest compute budget.

Our final agent was trained for only 10 million environment transitions and achieves a median human normalized score (HNS) of 205.7, reaching above human-level performance on 39 out of 53 games. In contrast, classic Rainbow achieves a median HNS of 70 (estimated) and 231 after training for 10 and 200 million transitions, respectively. Overall, their score after 200M transitions is equivalent to super-human performance on 40 out of 53 games. The exact scores for classic Rainbow at 10M frames were not published in Hessel et al. [2018]. Therefore, they were estimated from the provided evaluation curves. In addition, published results in Castro et al. [2018] for the "Dopamine" Rainbow implementation serve as another reference point (see Table 1, HNS of 50.4 for 10M and 145.8 for 200M). In any case, our approach outperforms the scores at 10M by a big margin and achieves scores more similar to the ones obtained after 200M steps.

| | **Rainbow [Castro et al., 2018]** | **DQN [Mnih et al., 2013]** | **Rainbow [Castro et al., 2018]** | **Rainbow [Hessel et al., 2018]** | **Ours** |
|---|---|---|---|---|---|
| **training frames** | 10M | 200M | 200M | 200M | 10M |
| **mean HNS** | 166.9 | 224.2 | 1161.6 | 1624.9 | 1174.2 |
| **median HNS** | 50.4 | 79.3 | 145.8 | 231.0 | 205.7 |
| **# games above human** | 19 | 24 | 38 | 40 | 39 |

**Table 1:** Comparison of human normalized scores averaged over the 53 tested games (and 3 random seeds) for our agent, two implementations of Rainbow, and DQN. Random and human scores used for computing the HNS were taken from Mnih et al. [2015] where available, otherwise from Badia et al. [2020a]. The full results are provided in the supplementary material.

## 6 Conclusion

In this paper, we addressed one of the fundamental questions in reinforcement learning: how to accelerate the training of RL agents? To achieve this, we employed a variety of tricks that speed up the RL training procedure and empirically investigated how they affect training speed as well as overall performance on the Atari benchmark. While most of these enhancements have been studied individually in the context of RL, our work is, to the best of our knowledge, the first to study them jointly. Our improved training procedure significantly reduces the training time and data required by the RL agent while maintaining competitive performance. Furthermore, we conducted a number of ablations to understand what effect the individual modifications have on the training process. Overall, the results we presented may serve as a practical guide for speeding up the training of RL agents.

## Acknowledgments and Disclosure of Funding

We would like to thank the TU Wien DataLab for providing the majority of the needed compute resources and the CleanRL team for their baseline implementation of DQN [Huang et al., 2021].

## References

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*. PMLR, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *8th International Conference on Learning Representations, ICLR 2020*, 2020b.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*, 2020.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*. PMLR, 2017.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Johan Björck, Xiangyu Chen, Christopher De Sa, Carla P. Gomes, and Kilian Q. Weinberger. Low-precision reinforcement learning: Running soft actor-critic in half precision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 980–991. PMLR, 2021. URL `http://proceedings.mlr.press/v139/bjorck21a.html`.

Johan Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning. *arXiv preprint arXiv:2106.01151*, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL `http://arxiv.org/abs/1812.06110`.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*. PMLR, 2020.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 2892–2901, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv preprint arXiv:2003.11881*, 2020.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 2018.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, 2018.

Florin Gogianu, Tudor Berariu, Mihaela Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: An optimisation perspective. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3734–3744. PMLR, 2021. URL `http://proceedings.mlr.press/v139/gogianu21a.html`.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc G. Bellemare, and Rémi Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, 2018.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html`.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2020.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 3215–3222, 2018.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, and Jeff Braga. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *arXiv preprint arXiv:2111.08819*, 2021. URL `https://arxiv.org/abs/2111.08819`.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=SJ6yPD5xg`.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2019.

Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590. PMLR, 2019.

Maximilian Lam, Sharad Chitlangia, Srivatsan Krishnan, Zishen Wan, Gabriel Barth-Maron, Aleksandra Faust, and Vijay Janapa Reddi. Quantized reinforcement learning (quarl). *arXiv preprint arXiv:1910.01055*, 2019.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 2020a. URL `http://proceedings.mlr.press/v119/laskin20a.html`.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33, 2020b.

Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle. Data center cooling using model-predictive control. *Advances in Neural Information Processing Systems*, 31:3814–3823, 2018.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *J. Artif. Intell. Res.*, 61:523–562, 2018.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *CoRR*, abs/2006.16712, 2020. URL `https://arxiv.org/abs/2006.16712`.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively parallel methods for deep reinforcement learning. *CoRR*, abs/1507.04296, 2015. URL `http://arxiv.org/abs/1507.04296`.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *CoRR*, abs/1804.03720, 2018. URL `http://arxiv.org/abs/1804.03720`.

Johan Samir Obando-Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 1373–1383. PMLR, 2021.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, and et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, Dec 2020.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining reward-free representations for data-efficient reinforcement learning. In *Self-Supervision for Reinforcement Learning Workshop - ICLR 2021*, 2021. URL `https://openreview.net/forum?id=o5z9Le5drua`.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. D2rl: Deep dense architectures in reinforcement learning. *arXiv preprint arXiv:2010.09163*, 2020.

Adam Stooke and Pieter Abbeel. Accelerated methods for deep reinforcement learning. *CoRR*, abs/1803.02811, 2018. URL `http://arxiv.org/abs/1803.02811`.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018.

Adrien Ali Taïga, William Fedus, Marlos C. Machado, Aaron C. Courville, and Marc G. Bellemare. On bonus based exploration methods in the arcade learning environment. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016.

Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 2019.

Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019a.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019b.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, 2016.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=r1Ue8Hcxg`.

## A Full Results

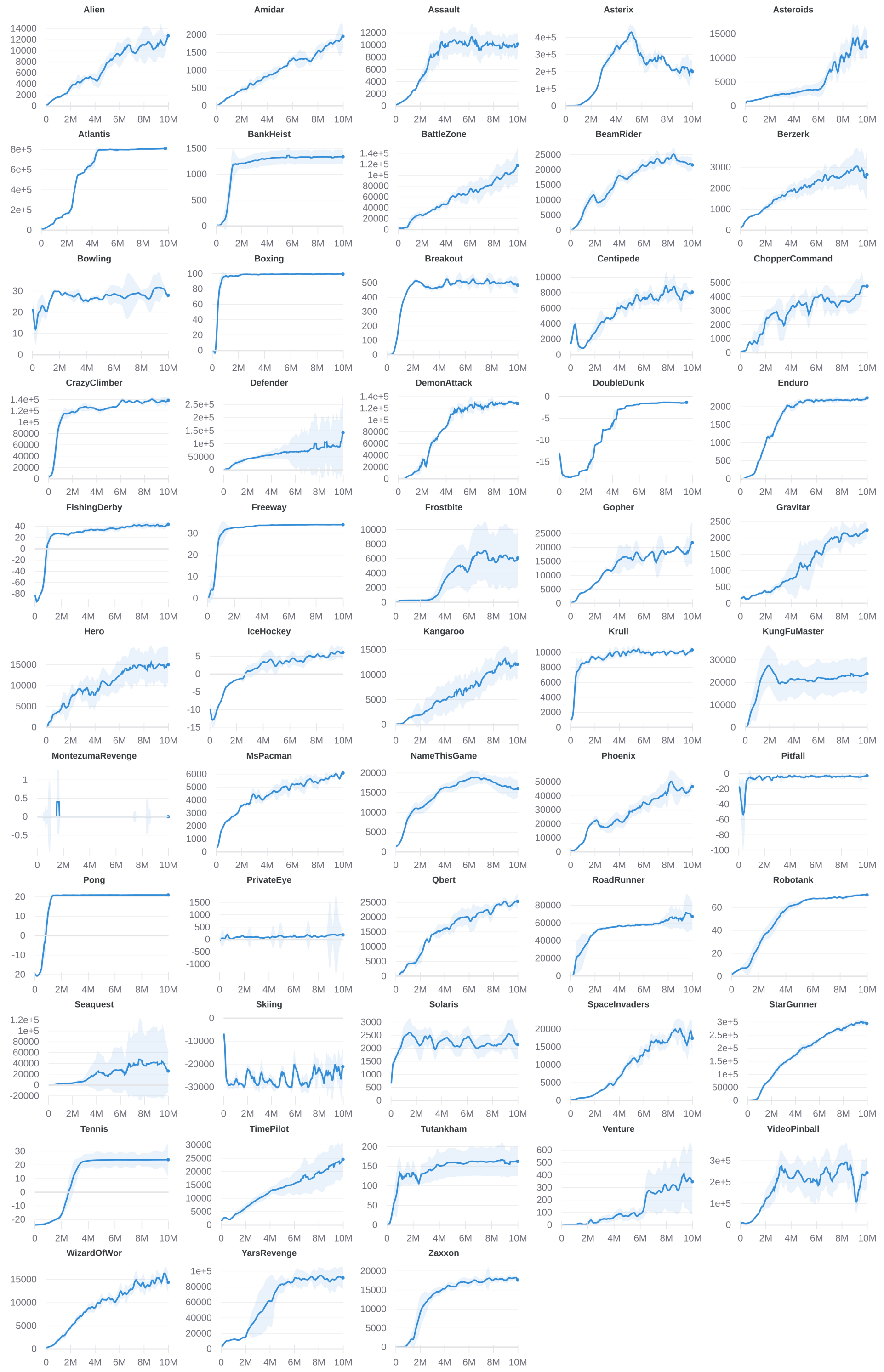

**Figure 4:** Learning curves for our Rainbow implementation for each of the selected Atari environments. Each curve represents the median 100-episode running average of episode returns over three seeds.

| | Uniform Random Policy | Human | Rainbow [Castro et al., 2018] | DQN [Mnih et al., 2013] | Rainbow [Castro et al., 2018] | Rainbow [Hessel et al., 2018] | Ours |
|---|---|---|---|---|---|---|---|
| **Alien** | 227.8 | 6,875.4 | 1,241.6 | 3,069.3 | 3,456.9 | 9,491.7 | 12,508.0 |
| **Amidar** | 5.8 | 1,675.8 | 348.3 | 739.5 | 2,529.1 | 5,131.2 | 2,071.2 |
| **Assault** | 222.4 | 1,496.4 | 1,483.1 | 3,358.6 | 3,228.8 | 14,198.5 | 10,709.5 |
| **Asterix** | 210.0 | 8,503.3 | 2,862.7 | 6,011.6 | 18,366.6 | 428,200.3 | 346,758.2 |
| **Asteroids** | 719.1 | 13,156.7 | 809.9 | 1,629.3 | 1,483.5 | 2,712.8 | 12,345.9 |
| **Atlantis** | 12,850.0 | 29,028.1 | 170,363.8 | 85,950.0 | 802,548.0 | 826,659.5 | 812,825.9 |
| **BankHeist** | 14.2 | 734.4 | 904.5 | 429.7 | 1,075.0 | 1,358.0 | 1,411.1 |
| **BattleZone** | 2,360.0 | 37,800.0 | 22,448.7 | 26,300.0 | 40,060.6 | 62,010.0 | 112,652.3 |
| **BeamRider** | 363.9 | 5,774.7 | 5,434.9 | 6,845.9 | 6,290.5 | 16,850.2 | 26,398.8 |
| **Berzerk** | 123.7 | 2,630.4 | 456.1 | 585.6 | 833.4 | 2,545.6 | 3,388.0 |
| **Bowling** | 23.1 | 154.8 | 44.1 | 42.4 | 42.9 | 30.0 | 40.9 |
| **Boxing** | 0.1 | 4.3 | 76.3 | 71.8 | 98.6 | 99.6 | 99.7 |
| **Breakout** | 1.7 | 31.8 | 49.9 | 401.2 | 120.1 | 417.5 | 537.6 |
| **Centipede** | 2,090.9 | 11,963.2 | 4,877.8 | 8,309.4 | 6,509.9 | 8,167.3 | 8,368.2 |
| **ChopperCommand** | 811.0 | 9,881.8 | 2,447.4 | 6,686.7 | 12,337.5 | 16,654.0 | 4,208.0 |
| **CrazyClimber** | 10,780.5 | 35,410.5 | 107,805.4 | 114,103.3 | 145,389.3 | 168,788.5 | 140,712.0 |
| **Defender** | 2,874.5 | 18,688.9 | | 23,633.0 | | 55,105.0 | 169,929.6 |
| **DemonAttack** | 152.1 | 3,401.3 | 2,943.1 | 9,711.2 | 17,071.3 | 111,185.2 | 131,657.2 |
| **DoubleDunk** | -18.6 | -15.5 | -18.6 | -18.1 | 22.1 | -0.3 | -1.2 |
| **Enduro** | 0.0 | 309.6 | 1,680.3 | 301.8 | 2,200.2 | 2,125.9 | 2,266.0 |
| **FishingDerby** | -91.7 | 5.5 | 14.5 | -0.8 | 41.8 | 31.3 | 42.1 |
| **Freeway** | 0.0 | 29.6 | 32.1 | 30.3 | 33.7 | 34.0 | 34.0 |
| **Frostbite** | 65.2 | 4,334.7 | 2,647.0 | 328.3 | 8,207.7 | 9,590.5 | 5,282.1 |
| **Gopher** | 257.6 | 2,321.0 | 4,399.6 | 8,777.4 | 10,641.1 | 70,354.6 | 25,606.8 |
| **Gravitar** | 173.0 | 2,672.0 | 208.2 | 306.7 | 1,271.8 | 1,419.3 | 2,107.3 |
| **Hero** | 1,027.0 | 25,762.5 | 10,468.5 | 19,950.3 | 46,675.2 | 55,887.4 | 15,377.2 |
| **IceHockey** | -11.2 | 0.9 | -5.2 | -1.6 | -0.2 | 1.1 | 6.6 |
| **Kangaroo** | 52.0 | 3,035.0 | 5,579.6 | 6,740.0 | 12,748.3 | 14,637.5 | 11,498.7 |
| **Krull** | 1,598.0 | 2,394.6 | 5,980.4 | 3,804.7 | 4,066.0 | 8,741.5 | 10,324.3 |
| **KungFuMaster** | 258.5 | 22,736.2 | 19,195.5 | 23,270.0 | 26,475.1 | 52,181.0 | 27,444.4 |
| **MontezumaRevenge** | 0.0 | 4,367.0 | 0.8 | 0.0 | 500.0 | 384.0 | 0.0 |
| **MsPacman** | 307.3 | 15,693.4 | 2,399.3 | 2,311.0 | 3,861.0 | 5,380.4 | 5,981.7 |
| **NameThisGame** | 2,292.3 | 4,076.2 | 9,023.2 | 7,256.7 | 9,025.8 | 13,136.0 | 19,819.0 |
| **Phoenix** | 761.4 | 7,242.6 | 4,925.3 | 8,485.2 | 8,545.4 | 108,528.6 | 60,954.5 |
| **Pitfall** | -229.4 | 6,463.7 | -1.4 | -286.1 | -19.8 | 0.0 | -1.8 |
| **Pong** | -20.7 | 9.3 | 15.8 | 18.9 | 20.2 | 20.9 | 21.0 |
| **PrivateEye** | 24.9 | 69,571.3 | 89.2 | 1,787.6 | 21,333.6 | 4,234.0 | 253.8 |
| **Qbert** | 163.9 | 13,455.0 | 6,861.5 | 10,595.8 | 17,382.9 | 33,817.5 | 25,712.4 |
| **RoadRunner** | 11.5 | 7,845.0 | 34,454.9 | 18,256.7 | 54,662.1 | 62,041.0 | 81,831.7 |
| **Robotank** | 2.2 | 11.9 | 21.7 | 51.6 | 65.5 | 61.4 | 70.7 |
| **Seaquest** | 68.4 | 20,181.8 | 1,646.0 | 5,286.0 | 9,903.4 | 15,898.9 | 63,724.4 |
| **Skiing** | -17,098.1 | -4,336.9 | -23,886.9 | -13,062.3 | -28,707.6 | -12,957.8 | -22,076.8 |
| **Solaris** | 1,236.3 | 12,326.7 | 1,429.0 | 3,482.8 | 1,582.7 | 3,560.3 | 2,877.6 |
| **SpaceInvaders** | 148.0 | 1,652.3 | 769.0 | 1,975.5 | 4,130.9 | 18,789.0 | 28,098.6 |
| **StarGunner** | 664.0 | 10,250.0 | 1,536.9 | 57,996.7 | 57,908.7 | 127,029.0 | 310,403.7 |
| **Tennis** | -23.8 | -8.9 | -2.3 | -2.5 | -0.2 | 0.0 | 15.9 |
| **TimePilot** | 3,568.0 | 5,925.0 | 2,960.6 | 5,946.7 | 12,050.5 | 12,926.0 | 31,333.2 |
| **Tutankham** | 11.4 | 167.6 | 203.8 | 186.7 | 239.1 | 241.0 | 167.0 |
| **Venture** | 0.0 | 1,187.5 | 17.2 | 380.0 | 1,528.9 | 5.5 | 437.1 |
| **VideoPinball** | 16,256.9 | 17,297.6 | 30,298.4 | 42,684.1 | 466,895.0 | 533,936.5 | 269,619.0 |
| **WizardOfWor** | 563.5 | 4,756.5 | 2,727.9 | 3,393.3 | 7,878.6 | 17,862.5 | 15,518.6 |
| **YarsRevenge** | 3,092.9 | 54,576.9 | 10,536.7 | 18,089.9 | 45,542.0 | 102,557.0 | 98,908.0 |
| **Zaxxon** | 32.5 | 9,173.3 | 4,521.1 | 4,976.7 | 14,603.0 | 22,209.5 | 18,832.6 |
| | | | | | | | |
| **training frames** | | | 10M | 200M | 200M | 200M | 10M |
| **mean HNS** | 0 | 100 | 166.9 | 224.2 | 1161.6 | 1624.9 | 1174.2 |
| **median HNS** | 0 | 100 | 50.4 | 79.3 | 145.8 | 231.0 | 205.7 |
| **# games above human** | 0 | 0 | 19 | 24 | 38 | 40 | 39 |

**Table 2:** Evaluation scores for all 53 tested Atari games (averaged over 3 random seeds). Random and human scores are from Mnih et al. [2015] where available, otherwise from Badia et al. [2020a].

## B Hyperparameters

### B.1 Hyperparameters for Rainbow

This section lists the hyperparameters used in our experiments. Parameters that differ from the ones used in Hessel et al. [2018] are marked with an asterisk. As in previous work, the unit "frames" refers to the number of environment steps taken by the wrapped environment, including frame-skipping. For noisy-nets DQN we implemented the "factorized Gaussian noise" variant, with noise vectors generated on the GPU.

| **Parameter** | **Value** |
|---|---|
| Discount factor $\gamma$ | 0.99 |
| Q-target update frequency | 32,000 frames |
| Importance sampling $\beta_0$ for PER | 0.45 |
| $n$ in n-step bootstrapping | 3 |
| Initial exploration $\epsilon$ | 1.0 |
| Final exploration $\epsilon$ | 0.01 |
| *Exploration $\epsilon$ decay time | 500,000 frames |
| $\sigma_0$ for noisy linear layers | 0.5 |
| *Learning rate | 0.00025 |
| *Adam $\epsilon$ parameter | $0.005/\text{batch size}$ |
| Gradient clip norm | 10 |
| Loss function | Huber |
| *Batch size | 256 |
| *Parallel environments | 64 |
| Replay Buffer Size | 1M transitions |
| Training starts at | 80,000 frames |
| *Q-network architecture | IMPALA-large with 2x channels |

### B.2 Environment pre-processing hyperparameters

This section lists the settings for preprocessing environments from `gym`, `gym-retro` and `procgen`. All environments used a time limit of 108k frames (30 minutes of emulator time). Image downscaling was performed with area interpolation. For `gym` environments, we max-pooled consecutive frames and used 0-30 noop actions at the beginning of each episode as in Hessel et al. [2018].

| **Parameter** | **Environment** | **Value** |
|---|---|---|
| Grayscale | gym | yes |
| | retro | no |
| | procgen | no |
| Frame-skipping | gym | 4 |
| | retro | 4 |
| | procgen | 1 |
| Frame-stacking | gym | 4 |
| | retro | 4 |
| | procgen | 4 |
| Resolution | gym | $84 \times 84$ |
| | retro | $72 \times 96$ |
| | procgen | $64 \times 64$ |

## B.3 Network Architecture

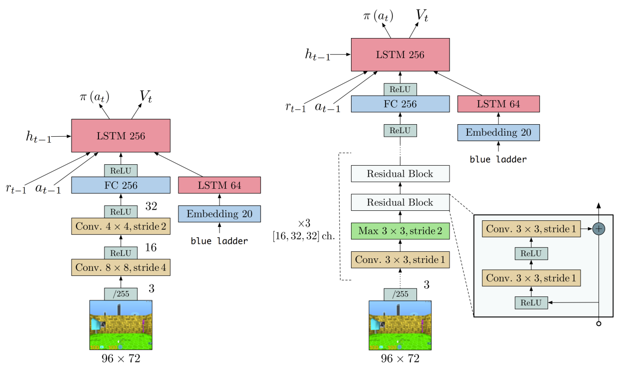

**Figure 5:** The unmodified small (left) and large (right) IMPALA CNN network architecture [from Espeholt et al., 2018].