On the consistency of hyper-parameter selection in value-based deep reinforcement learning

Johan Obando-Ceron[*1,2,3], **João G.M. Araújo**[*3], **Aaron Courville**[1,2], **Pablo Samuel Castro**[1,2,3]

Mila - Québec AI Institute[1]
Université de Montréal[2]
Google DeepMind[3]

Abstract

Deep reinforcement learning (deep RL) has achieved tremendous success on various domains through a combination of algorithmic design and careful selection of hyper-parameters. Algorithmic improvements are often the result of iterative enhancements built upon prior approaches, while hyper-parameter choices are typically inherited from previous methods or fine-tuned specifically for the proposed technique. Despite their crucial impact on performance, hyper-parameter choices are frequently overshadowed by algorithmic advancements. This paper conducts an extensive empirical study focusing on the reliability of hyper-parameter selection for value-based deep reinforcement learning agents, including the introduction of a new score to quantify the consistency and reliability of various hyper-parameters. Our findings not only help establish which hyper-parameters are most critical to tune, but also help clarify which tunings remain *consistent* across different training regimes.

1 Introduction

Sequential decision making is generally considered an essential ingredient for generally capable agents. The ability to plan ahead and adapt to changing circumstances is synonymous with the concept of *agency*. For decades, the field of reinforcement learning (RL) has worked on developing methods, or agents, for precisely this purpose. This research has borne impressive results, such as developing agents which can play difficult Atari games (Mnih et al., 2015), control stratospheric balloons (Bellemare et al., 2020), control a tokamak fusion reactor (Degrave et al., 2022), among others. These are all examples of *deep reinforcement learning* (DRL), which combines the theory of reinforcement learning with the expressiveness and flexibility of deep neural networks.

The success of these methods built on years of academic research, where novel algorithms and techniques were introduced and showcased on academic benchmarks such as the ALE (Bellemare et al., 2012), MuJoCo (Todorov et al., 2012), and others. These benchmarks typically consist of a suite of environments that have varied transition and reward dynamics. Their common usage provides us with a familiarity which affords us a sense of interpretability, a consistency in evaluation that grants us a sense of reliability, and their variety yields a sense of generalizability. Unfortunately, this promise often fails to materialize: their reliability has been brought into question by numerous works which demonstrate their fickleness (Henderson et al., 2017; Agarwal et al., 2021), while there is a general sentiment that researchers have "overfit" to these benchmarks, bringing into question

*Authors contributed equally. Correspondence to jobando0730@gmail.com,[joaogui,psc]@google.com

their generalizability. A critical aspect to these challenges is the difficulty in training neural networks in an RL setting (Ostrovski et al., 2021; Lyle et al., 2022; Sokar et al., 2023).

Although the successes above built on prior methods, they were not taken "as is": it took large teams of researchers many months and lots of compute to adapt prior work to their specific problem. These adaptations include changes to the network architectures, designing reward functions to induce the desired behaviours, and careful tuning of the many hyper-parameters. This last point is indeed *essential* to the success of any DRL method: improper hyper-parameter choices can cause a theoretically sound method to drastically underperform, while careful hyper-parameter selection can dramatically increase the performance of an otherwise sub-optimal method.

As an example of this dichotomy, we examine how DER (van Hasselt et al., 2019), a method that has become a common baseline for the Atari 100k benchmark (Kaiser et al., 2019), came to be. DQN, considered to be the start of the field of DRL research, was introduced by showcasing its super-human performance on the ALE (Bellemare et al., 2012), a suite of 57 Atari 2600 games. This suite became one of the most popular benchmarks on which to evaluate new methods over 200 million environment frames[1]. A few years later, when Kaiser et al. (2019) introduced the SiMPLe algorithm as a sample-efficient method, they argued for evaluating it only on 100k agent actions[2] with a subset of 26 games, so as to properly test the sample-efficiency of new methods. The authors demonstrated that their proposed method outperformed Rainbow (Hessel et al., 2018), the state-of-the-art method of the time. In response, van Hasselt et al. (2019) introduced Data Efficient Rainbow (DER), which outperformed SiMPLe even though it was the same Rainbow algorithm, but *with a careful tuning of the hyper-parameters for the* 100*k training regime*.

One could argue that the hyper-parameters of Rainbow were overly-tuned to the 200M benchmark, while the hyper-parameters of DER were overly-tuned to the 100k benchmark. More importantly, what this story highlights is that, despite careful evaluation it is quite likely that a new method *will not work as intended when deployed on a different environment from which it was trained on*, and that a significant amount of hyper-parameter tuning will be necessary. This flies in the face of the supposed generalizability of DRL academic research, and makes it difficult for groups without large computational budgets to successfully apply prior work to applied problems.

It thus behooves the community to develop a better understanding of the *transferability* and *consistency* of hyper-parameter selection across different training regimes, and to build a better shared understanding of the relative importance of the many possible hyper-parameters to tune. In this work, we take a stride towards this by conducting an exhaustive empirical investigation of the various hyper-parameters affecting DRL agents. We focus our attention on two value-based agents developed for the Atari 100k suite: DER mentioned above, and DrQ(ϵ), a variant of DQN that was optimized for the 100k suite. Although developed for the 100k suite, we also train these agents for 40M million environment frames. Our intent is to examine the transferability of various hyper-parameter choices across different training regimes. Specifically, we investigate: **Across data regimes:** Do hyper-parameters selected in the 100k regime work well in a larger data regime? **Across agents:** Do hyper-parameters selected for one agent work well in another? **Across environments:** Do hyper-parameters tuned in one set of environments work well in others?

In total, we investigated 12 hyper-parameters with different values for 2 agents over 26 environments, each for 5 seeds, resulting in a total of 108k independent training runs. This breadth of experimentation results in an overwhelming amount of data which complicates their analyses. We address this challenge in two ways: *(i)* We introduce a new score which provides us with an aggregate value for the considerations mentioned above. *(ii)* We provide an interactive website where others may easily navigate the large number of experimental figures we have generated.

[1] See (Machado et al., 2018) for more details on ALE evaluation standards.

[2] The standard for ALE agents is to use frame-skipping, where 4 environment frames occur for every agent action. This results in frustratingly confusing nomenclature, as 200M is specified in environment frames (or 500k agent actions), while 100k is specified in agent actions (or 400k environment frames).

The score provides us with a high-level overview of our findings, while the website grants us a fine-grained mechanism to analyze the results. We hope this effort provides the community with useful tools so as to develop not just better DRL algorithms, but better methodologies to evaluate their interpretability, reliability, and generalizability.

2 Background

The field of reinforcement learning studies algorithms for sequential decision-making problems. In these settings, an algorithm (or agent) interacts with an *environment* by transitioning between *states* and making action choices at discrete timesteps; the environment responds to each action by (possibly) changing the agent's state and yielding a numerical reward or cost. The goal of the agent is to maximize the cumulative rewards (or minimize the cost) throughout its lifetime. This is typically formalized as a Markov decision process (MDP) (Puterman, 2014) $\langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{X}$ is the set of states, $\mathcal{A}$ is the set of available actions, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$[3] is the transition function, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is a discount factor. An agent's behaviour is formalized by a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, whose *value* from any state $x \in \mathcal{X}$ is given by the Bellman recurrence $V^\pi(x) := \mathbb{E}_{a\sim\pi(x)} \left[\mathcal{R}(x, a) + \gamma \mathbb{E}_{x'\sim\mathcal{P}(x,a)} V^\pi(x')\right]$. Q-functions allow us to measure the value of taking any action $a \in \mathcal{A}$ from a state $x \in \mathcal{X}$ and following π afterwards: $Q^\pi(x, a) := \mathcal{R}(x, a) + \gamma \mathbb{E}_{x'\sim\mathcal{P}(x,a)} V^\pi(x')$. A policy π^* is considered optimal if for any policy π, $V^* := V^{\pi^*} \geq V^\pi$.

Solving for the equations discussed above would require access to both $\mathcal{R}$ and $\mathcal{P}$, which are usually unknown. Instead, RL typically assumes the agent has access to transitions $\tau := (x, a, r, x') \in \mathcal{X} \times \mathcal{A} \times \mathbb{R} \times \mathcal{X}$, arising from interactions with the environment. Given such a transition, Q-learning (Watkins & Dayan, 1992) updates its estimate of Q via: $Q_{t+1}(x, a) \leftarrow Q_t(x, a) + \alpha TD(Q, \tau)$, where α is a learning rate and TD is the *temporal-difference error*, given by $TD(Q_t, \tau) := r + \gamma \max_{a'\in\mathcal{A}} Q_t(x', a') - Q_t(x, a)$. If the state and action spaces are small, one can store all the Q-values in a table of size $|\mathcal{X}| \times |\mathcal{A}|$. For most problems of interest, however, state spaces are very large (and possibly infinite). In these cases, one can use a function approximator, such as a neural network, parameterized by θ: $Q_\theta \approx Q$. Indeed, in order to achieve super-human performance on the Arcade Learning Environment (ALE) (Bellemare et al., 2012), Mnih et al. (2015) used a neural network consisting of three convolutional layers (Conv layers), followed by two multi-layer perceptrons (Dense layers) with $|\mathcal{A}|$ outputs in the final layer (representing the Q-value estimates for each action). With the exception of the final layer, a ReLU non-linearity follows each layer.

Updating Q_θ thus corresponds to updating the parameters θ, which may be done by using optimization algorithms such as Adam (Kingma & Ba, 2015) to minimize the temporal-difference error. At a high-level, this yields an update of the form: $\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\tau\sim\mathcal{D}} TD(Q_{\theta_t}, \tau)$. The expectation can be approximated using a batch of m transitions drawn from a distribution $\mathcal{D}$, which can be computed efficiently on specialized hardware such as GPUs and TPUs. Additionally, Mnih et al. (2015) argued that using $\bar{\theta}$, a less-frequently updated copy of the parameters, when computing TD helps with training stability. A common approach introduced by Mnih et al. (2015) is to clip the rewards at $(-1, 1)$. The TD term thus becomes: $TD(Q_\theta, \tau) := clip(r, (-1, 1)) + \gamma \max_{a'\in\mathcal{A}} Q_{\bar{\theta}}(x', a') - Q_\theta(x, a)$.

Although DQN benchmarked on the 57 ALE games with the same set of hyper-parameters, Anschel et al. (2017) demonstrated that in some environments it can result in degraded performance. A number of papers have proposed improvements to increase stability and performance, which Hessel et al. (2018) combined into a single agent they called *Rainbow*. Specifically, they combined DQN with double Q-learning (van Hasselt et al., 2016), prioritized experience replay (Schaul et al., 2016), dueling networks (Wang et al., 2016), multi-step learning (Sutton, 1988), noisy nets (Fortunato et al., 2018), and distributional reinforcement learning (Bellemare et al., 2017).

[3] $\Delta(X)$ denotes a distribution over the set X.

3 THC Score

Statistical metrics play a crucial role in assessing and evaluating the performance of DRL algorithms. They provide valuable insights into the strengths and weaknesses of different approaches, guiding researchers and practitioners in the development of more effective reinforcement learning systems. For example, some the metrics focus on the mean reward obtained by an agent per time step (Average Reward), the percentage of episodes in which the agent achieves a predefined goal or task (success rate) among others (Agarwal et al., 2021; Chan et al., 2019; Henderson et al., 2017).

Measuring the transferability/consistency of hyper-parameters in DRL is challenging, as existing metrics fall short in capturing the nuanced aspects of how well hyper-parameter settings generalize across different environments or agents. Developing such a metric would enhance the ability to systematically compare and select hyper-parameter configurations that exhibit robust performance across a range of application domains.

To understand the consistency of hyper-parameters we focus on their ranking consistency across experimental settings. Put another way: if a given hyper-parameter value is optimal/pessimal in a setting, is it still optimal/pessimal in another? And so we analyse, for each hyper-parameter, whether their values lead to the same ranking order for different experimental settings, where the ranking is on final performance.

We compute ranking agreement for three setups: 1**) Varying algorithms** while keeping the environment and data regime fixed (e.g. when proposing a new value-based algorithm but not having enough compute to run a comprehensive hyper-parameter search). 2**) Varying environments** while keeping the algorithm and data regime fixed (e.g. when using a state of the art algorithm in a new domain). 3**) Varying data regimes** while keeping the environment and algorithm fixed (e.g. when adapting a new algorithm to a new data regime (van Hasselt et al., 2019)). Concretely, our desire is to have a metric that yields a high value score would indicate that the hyper-parameter in question is *important*, in the sense that it will likely require retuning; conversely, a low score suggests the hyper-parameter value can likely be kept as is.

Kendall's Tau (Kendall, 1938) and Kendall's W (Kendall & Smith, 1939) are natural choices, but these metrics were developed for situations where the rankings were based on a single score, instead of a range of possible scores, and they can result in degenerate values when two settings have similar performance or when two settings alternate between optimal and pessimal rankings. For these reasons, we introduce the **T**uning **H**yperparameter **C**onsistency (**THC**) score. Consider a set of n hyper-parameters $\{H_1, \ldots, H_n\}$, each with its set of values $\{\{h_{11}, h_{12}, \ldots, h_{1m_1}\}, \ldots, \{h_{n1}, h_{n2}, \ldots, h_{nm_n}\}\}$ (e.g. hyper-parameter H_i has m_i values). The THC score involves three computations: (i) rankings for each hyper-parameter setting (Algorithm 1); (ii) normalized peak-to-peak value for each hyper-parameter setting (Eqn. 1 below); and (iii) overall THC score for the hyper-parameter (see Eqn. 2 below).

If we run multiple independent runs for each hyper-parameter setting h_{ij}, we can compute the mean μ_{ij} and standard deviation σ_{ij} for these runs[4]. For each hyper-parameter setting h_{ij} we then compute an initial ranking r'_{ij} based on the upper bound $(\mu_{ij} + \sigma_{ij})$, with the lower bound $(\mu_{ij} - \sigma_{ij})$ used to break ties. We then define a set containing hyper-parameter settings with overlapping values:

$$I_{ij} := \{k | (\mu_{ij} - \sigma_{ij} < \mu_{ik} + \sigma_{ik} \text{ and } \mu_{ij} - \sigma_{ij} > \mu_{ik} - \sigma_{ik}) \\ \text{or} \\ (\mu_{ij} + \sigma_{ij} > \mu_{ik} - \sigma_{ik} \text{ and } \mu_{ij} + \sigma_{ij} < \mu_{ik} + \sigma_{ik})\}$$

The final ranking of each hyper-parameter is $r_{ij} = \frac{\sum_{k \in I_{ij}} r'_{ik}}{|I_{ij}|}$, as Algorithm 1 details. These rankings are for *one* training regime; however, as mentioned in the introduction, we are interested in quantifying the *consistency* of a hyper-parameter H across varying training regimes.

[4]One may also use confidence intervals instead of standard deviations.

Algorithm 1 Compute rankings

Require: Multiple runs for various settings of hyper-parameter H_i: $\{h_{i1}, h_{i2}, \ldots, h_{im_i}\}$, aggregate metrics μ_i: $\{\mu_{i1}, \mu_{i2}, \ldots, \mu_{im_i}\}$ and measure of spread σ_i: $\{\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{im_i}\}$

for i in $1 \ldots n$ **do**
  $r'_i = \text{argsort}(\mu_i + \sigma_i)$ ▷ Gets the index of each value as if the array was sorted
  $\mu'_i, \sigma'_i = \mu_i[r'_i], \sigma_i[r'_i]$ ▷ Sorted versions of aggregate and spread metrics
  for j in $1 \ldots m_i$ **do**
    $u_j = \text{binary\_search}(\mu'_i - \sigma'_i, \mu_{ij} + \sigma_{ij})$ ▷ highest rank whose lower bound overlaps with j
    $l_j = \text{binary\_search}(\mu'_i + \sigma'_i, \mu_{ij} - \sigma_{ij})$ ▷ lowest rank whose upper bound overlaps with j
  end for
  $\mathbf{r_i} = \frac{\mathbf{u+l}}{\mathbf{2}}$ ▷ The average rank in $l_j, l_j + 1, \ldots, u_j$ is the average of l_j and u_j
end for

Consider four training regimes A, B, C, D, and let $\{\mathfrak{R}^A, \ldots, \mathfrak{R}^D\}$ denote their respective rankings. For each hyper-parameter value $h_x \in H$ we compute its normalized "peak-to-peak"[5] value $\overline{\text{ptp}}$, which quantifies its variance in ranking, as follows: First compute the ptp value $\text{ptp}(h_x) = \max\left(\{\mathfrak{R}^A(h_x), \ldots, \mathfrak{R}^D(h_x)\}\right) - \min\left(\{\mathfrak{R}^A(h_x), \ldots, \mathfrak{R}^D(h_x)\}\right)$, then normalize:

$$\overline{\text{ptp}}(h_x) = \frac{\text{ptp}(h_x)}{\sum_{h_y \in H} \text{ptp}(h_y)} \tag{1}$$

Notably, hyper-parameter settings that have consistent rankings across training regimes will have a normalized ptp value of zero. Finally, the THC score for hyper-parameter H is defined as:

$$\text{THC}(H) = \frac{\sum_{h_x \in H} \overline{\text{ptp}}(h_x)}{|H|}. \tag{2}$$

This score will result in low values for hyper-parameters whose varying settings have consistent ranking across various training regimes, and high values when these rankings vary. Intuitively, *hyper-parameters with high values will most likely require re-tuning when switching training regimes.* See Appendix A for more examples of computing the score, as well as the source code provided with this submission.

4 Hyper-parameters considered

We describe the set of hyper-parameters explored in this work, with the values used for each listed in Appendix C. Unless otherwise specified, these are examined for both Conv and Dense layers.

Activation functions: Non-linear activation functions are a fundamental part of neural networks, as their removal effectively turns the network into a linear function approximator. While various activation functions have been proposed (Devlin et al., 2018; Elfwing et al., 2018; Dauphin et al., 2017), there have been few works comparing their performance (Shamir et al., 2020); to the best of our knowledge, there are no previous examples of such a comparison in the RL setting.

Normalization: Normalization plays an important role in supervised learning (Tan & Le, 2019; Xie et al., 2017) but is relatively rare in deep reinforcement learning, with a few exceptions (Gogianu et al., 2021; Bhatt et al., 2019; Arpit et al., 2019; Silver et al., 2017). We explore *batch normalization* (Ioffe & Szegedy, 2015) and *layer normalization* (Ba et al., 2016).

Network capacity: "Scaling laws" have been central to the growth of capabilities in large language/vision models, but have mostly eluded reinforcement learning agents, with a few exceptions (Schwarzer et al., 2023; Taiga et al., 2022; Farebrother et al., 2022; Ceron et al., 2024b;a; Farebrother

[5] Inspired by numpy's peak-to-peak function numpy.ptp (Harris et al., 2020).

et al., 2024). To investigate the impact of network size, we vary the *depth* (e.g. the number of hidden layers) and the *width* (e.g. the number of neurons of each hidden layer).

Optimizer hyper-parameters: We explore three hyper-parameters of Adam (Kingma & Ba, 2015), which has become the standard optimizer used by most: *learning rate*, *epsilon* and *weight decay*. *Learning rate* determines the step size at which the algorithm adjusts the model's parameters during each iteration. ϵ represents a small constant value that is added to the denominator of the update rule to avoid numerical instabilities. *Weight decay* adds a penalty term to the loss function during training that discourages the model from assigning excessively increasing weight magnitudes.

ϵ-greedy exploration: ϵ-greedy exploration is a simple and popular exploration technique which picks actions greedily with probability $1-\epsilon$, and a random action with probability ϵ. Traditionally, experiments on the ALE use a linear decay strategy to decay ϵ from 1.0 to its target value.

Reward clipping: Most ALE experiments clip rewards at $(-1, 1)$ (Mnih et al., 2015).

Discount factor: The multiplicative factor γ discounts future rewards and its importance has been observed in a number of recent works (Amit et al., 2020; Hessel et al., 2019; Gelada & Bellemare, 2019; Van Seijen et al., 2019; François-Lavet et al., 2015; Schwarzer et al., 2023).

Replay buffer: DRL agents store past experiences in a replay buffer, to sample from during learning. The *replay capacity* parameter refers to the amount of data experiences stored in the buffer. It is common practice to only begin sampling from the replay buffer when a minimum number of transitions have been stored, referred to as the *minimum replay history*.

Batch size: The number of stored transitions that are sampled for learning at each training step.

Update horizon: Multi-step learning (Sutton, 1988) computes the temporal difference error using multi-step transitions, instead of a single step. DQN uses a single-step update by default, whereas Rainbow chose a 3-step update (Hessel et al., 2018). The update horizon has been argued to trade-off between the bias and the variance of the return estimate (Kearns & Singh, 2000).

Target Update periods: Value based agents often employ an online and a *target* Q-network, the latter which is updated less frequently by directly syncing (or Polyak-averaging) from the online network; the *target updated period* determines how frequently this occurs.

Update periods: The online network parameters are updated after every *update period* environment steps, with a value of 4 used in standard ALE training.

Number of atoms: In distributional reinforcement learning (Bellemare et al., 2017), the output layer predicts the distribution of the returns for each action a in a state s, instead of the mean $Q^\pi(s, a)$. A popular approach is to model the return as a categorical distribution parameterized by a certain number of 'atoms' over a pre-specified support.

5 Experimental results

As mentioned in the introduction, there already exist two data regimes for evaluating agents on the ALE suite: the (low-data regime) 100k (Kaiser et al., 2019) and the original 200M benchmark (Mnih et al., 2015). The 100k benchmark includes only 26 games from the original suite, so we focus on these for our evaluation. For computational considerations, we follow Graesser et al. (2022) and use 40M million environment frames as our large-data regime. We use the settings of DrQ(ϵ) (introduced by Agarwal et al. (2021) as an improvement over the DrQ of Yarats et al. (2021)), and Data Efficient Rainbow (DER) introduced by van Hasselt et al. (2019). All experiments were run on a Tesla P100 GPU and took around 2-4 hours (100k) and 1-2 days (40M) per run. Both algorithms are implemented in the Dopamine library (Castro et al., 2018). Since the 100k setting is cheaper, we evaluated a larger set of hyper-parameter values there and manually picked the most informative subset for running in the 40M setting. For all our experiments we ran 5 independent seeds and followed the guidelines suggested by Agarwal et al. (2021) for more statistically meaningful

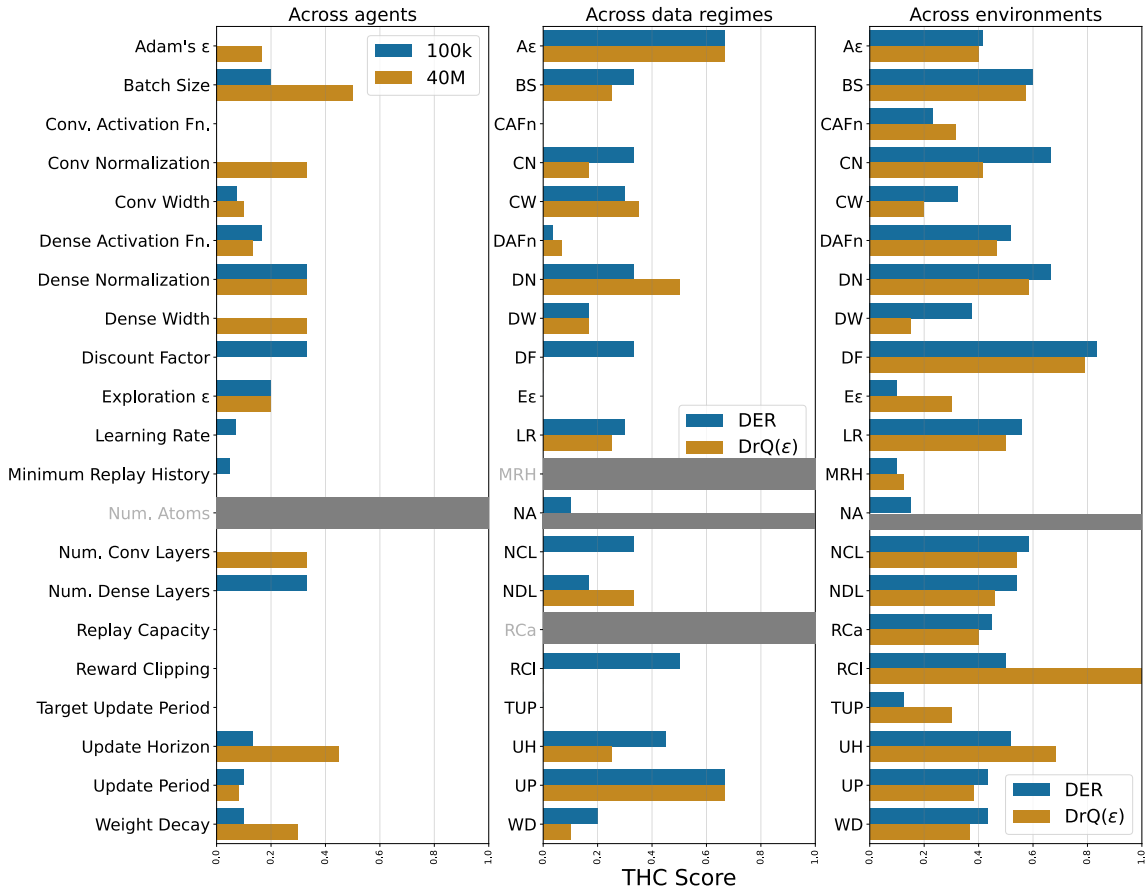

Figure 1: Tuning hyper-parameter Consistency (THC Score, see section 3) evaluated across agents (**left panel**), data regimes (**center panel**), and environments (**right panel**). Different colors indicate different data regimes (left panel) and different agents (center and right panels); grey bars/titles indicate hyper-parameters which are not comparable across the considered transfer settings.

comparisons. Specifically, we computed aggregate human-normalized scores and report interquantile mean (IQM) with 95% stratified bootstrap CIs.

In Figure 1 we present the computed THC score for all the hyper-parameters discussed in section 4, and we discuss their consistency across agents in Section 5.1, across data regimes in Section 5.2, and across environments in Section 5.3. More detailed discussions are provided in Appendix B and a set of interesting findings in Appendix D. It is worth recalling that higher THC scores indicate less consistency, which suggests a likely need to re-tune the respective hyper-parameters when changing training configurations.

5.1 Optimal hyper-parameters mostly Transfer Across Agents

We find that optimal hyper-parameters for DrQ(ϵ) agree quite often with DER, which is somewhat expected given that they're based on the same classical RL algorithm of Q-learning, and have the same number of updates in the same environments. Looking at THC values between the two agents for different data regimes we see that all values are below 0.5, and in the 100k regime tend to be even lower. Nevertheless, comparing the results of the two rows in figs. 3 and 4 demonstrate that there can still be strong differences between the two. In the 40M regime, the hyper-parameters with the highest THC are batch size and update horizon, consistent with the findings of Obando Ceron et al. (2023), where these two hyper-parameters proved crucial to boosting agent performance.

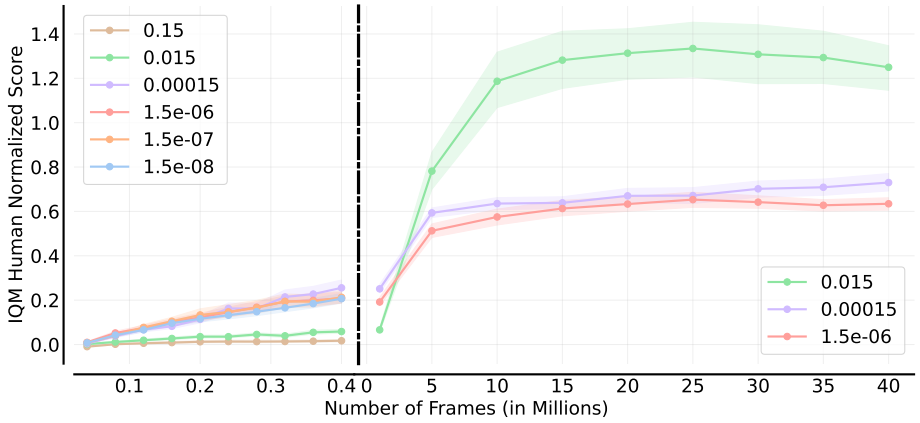

Figure 2: **Measured IQM of human-normalized scores on the** 26 100**k benchmark games, with varying Adam's** ϵ for DER. We evaluate performance at 100k agent steps (or 400k environment frames), and at 40 million environment frames. The ordering of the best hyper-parameters switches between the two data regimes.

5.2 Optimal hyper-parameters mostly do not Transfer Across Data Regimes

We find that optimal hyper-parameters for Atari 100k mostly do not transfer once you move to 40M updates, showing that even when keeping algorithms and environment constant one may still need to tune hyper-parameters should they change the amount of data their agent can train on. Of the hyper-parameters considered, *Adam's* ϵ and *update period* seem to be the most critical to re-tune (see Figure 2 for results on DER for Adam's ϵ). The results with Adam's ϵ are surprising, as the purpose of this hyper-parameter is mostly for numerical stability. The update period (as well as the update horizon) results are consistent with what is done in practice between these two data regimes (e.g. Rainbow uses an update period of 4 and an update horizon of 3, while DER uses 1 and 10, respectively).

5.3 Optimal hyper-parameters do not Transfer Across Environments

Our experiments show that hyper-parameters that perform well on some games lead to lackluster final performance in others. Indeed, in Figure 1 we can see that the THC score is highest when evaluating across environments. This strongly suggests that, when using an existing agent in a new environment, most of the hyper-parameters would need extra tuning. Figure 3 displays the results when varying batch size, where we can see that the rankings can sometimes be complete opposites across games (compare Kangaroo and Gopher).

6 A web-based appendix

We have run an extensive number of experiments (around 108k) for this work, which would render a traditional appendix unwieldy. Instead, we provide an interactive website[6] which facilitates navigating the full set of results[7]. Presenting empirical research results in this manner offers a range of benefits that enhance accessibility, engagement, and comprehension. This dynamic presentation allows readers to more easily make comparisons over different games, agents, and parameters.

The website's main page presents aggregate IQM results for all hyper-parameters investigated in both data regimes (e.g. Figure 2), while sub-pages present detailed performance comparisons when sliced by game (Figure 3 presents a subset of this) and hyper-parameter (Figure 4 presents a subset

[6]Website available at *https://consistent-hparams.streamlit.app/*.
[7]Website repository at *https://github.com/Consistent-Website*.

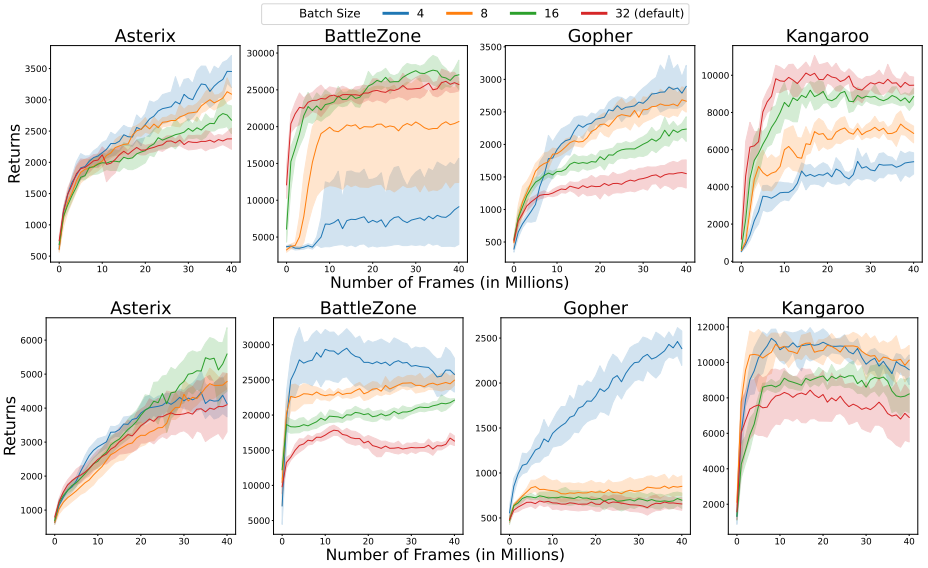

Figure 3: **Measured returns with varying batch size** for DrQ(ϵ) (top) and DER (bottom) at 40M environment frames for four representative games, demonstrating that the ranking of the hyper-parameter values can drastically change from one game to the next. All results averaged over 5 seeds, shaded areas represent 95% confidence intervals.

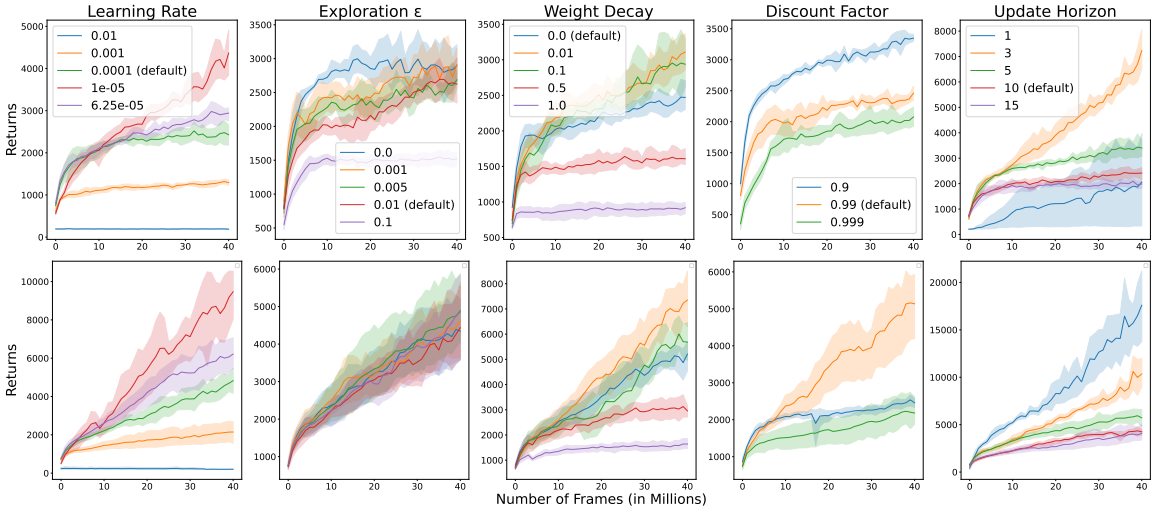

Figure 4: **Measured returns with various hyper-parameter variations on Asterix** for DrQ(ϵ) (top) and DER (bottom) at 40M environment frames. Displaying eight representative hyper-parameters, enabling per-game analyses for hyper-parameter selection.

of this). The added level of granularity provided by the sub-pages can be crucial for understanding the specific strengths and weaknesses of an algorithm in various scenarios. All results averaged over 5 seeds, shaded areas represent 95% confidence intervals.

7 Related work

While RL as a field has seen many innovations in the last years, small changes to the algorithm or its implementation can have a big impact on its results (Engstrom et al., 2020; Araújo et al., 2021). Deep reinforcement learning approaches are often notoriously sensitive to their hyperparamaters

and demonstrate brittle convergence properties (Haarnoja et al., 2018). This is particularly true for off-policy approaches that use a replay buffer to leverage past experiences (Duan et al., 2016).

Henderson et al. (2017) investigate the effects of existing degrees of variability between various RL setups and their effects on algorithm performance. Although restricted to the domain of existing environments, Henderson et al. (2017) propose more robust performance estimators for RL learning algorithms. Islam et al. (2017) and Huang et al. (2022) have shown the difficulty in reproducing policy gradient algorithms due to the variance. Andrychowicz et al. (2020) did a deep dive in algorithmic choices on policy-based algorithms. Their analyses covered differences in hyper-parameters, algorithms, and implementation details.

In an effort to consolidate innovations in deep RL, several papers have examined the effect of smaller design decisions like the loss function or policy regularization for on-policy algorithms Andrychowicz et al. (2020), DQN agents (Ceron & Castro, 2021), imitation learning (Hussenot et al., 2021) and offline RL (Paine et al., 2020; Lu et al., 2021). AutoRL methods, on the other hand, have focused on automating and abstracting some of these decisions (Parker-Holder et al., 2022; Eimer et al., 2023) by using data-driven approaches to learn various algorithmic components or even entire RL algorithms (Co-Reyes et al., 2021; Lu et al., 2022). All these works have demonstrated that hyperparameters in deep reinforcement learning warrant more attention from the research community than they currently receive. Underreported tuning practices can distort algorithm evaluations, and overlooked hyperparameters may lead to suboptimal performance.

8 Discussion

One of the central challenges in reinforcement learning research is the non-stationarity during training in the inputs (due to self-collected data) and targets (due to bootstrapping). This is in direct contrast with supervised learning settings, where datasets and labels are typically fixed throughout training. This non-stationarity may be largely to blame for some of the ranking inconsistencies observed under different training regimes (e.g. Figure 2), and why different hyper-parameter tunings are required for different settings (e.g. DER versus Rainbow).

Hyper-parameters are commonly tuned on a subset of environments (e.g. 3-5 games) and then evaluated on the full suite. Our findings suggest that this approach may not be the most rigorous, as hyper-parameter selection can vary dramatically from one game to the next (c.f. figs. 3 and 4). While aggregate results (e.g. IQM) provide a succinct summary of performance, they unfortunately gloss over substantial differences in the individual environments. If our hope as researchers is to be able to use these algorithms beyond academic benchmarks, understanding these differences is *essential*, in particular in real-world applications such as healthcare and autonomous driving.

We have conducted a large number of experiments to investigate the impact of various hyper-parameter choices. While the THC score (Figure 1) provides a high-level view of the transferability of hyper-parameter choices, our collective results suggest that a *single* set of hyper-parameter choices will never suffice to achieve strong performance across all environments. The ability to dynamically adjust hyper-parameter values during training is one way to address this; to properly do so would require quantifiable measures of environment characteristics that go beyond coarse specifications (such as sparse versus dense reward systems). The per-game results we present here may serve as an initial step in this direction. In Appendix D.3 we provide a fine-grained analysis of DER on Gopher as an example of the type of analyses enabled by our website. We hope our analyses, results, and website prove useful to RL researchers in developing robust and transferable algorithms to handle increasingly complex problems.

Acknowledgements

The authors would like to thank Jesse Farebrother, Gopeshh Subbaraj, Doina Precup, Hugo Larochelle, and the rest of the Google DeepMind Montreal team for valuable discussions during

the preparation of this work. Jesse Farebrother deserves a special mention for providing us valuable feed-back on an early draft of the paper. We thank the anonymous reviewers for their valuable help in improving our manuscript. 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), Jupyter Kluyver et al. (2016), Pandas McKinney (2013) and JAX Bradbury et al. (2018).

Broader Impact Statement

Although the work presented here is mostly academic, it aids in the development of more capable and reliable 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.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. In *International conference on machine learning*, pp. 269–278. PMLR, 2020.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*, 2020.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pp. 176–185. PMLR, 2017.

João Guilherme Madeira Araújo, Johan Samir Obando-Ceron, and Pablo Samuel Castro. Lifting the veil on hyper-parameters for value-baseddeep reinforcement learning. In *NeurIPS 2021 Workshop LatinX in AI*, 2021. URL https://openreview.net/forum?id=3UK39iaaVpE.

Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. *Advances in Neural Information Processing Systems*, 32, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, Vol. 47:253–279, 2012. cite arxiv:1207.4708.

Marc G. Bellemare, Will Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyun Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588:77 – 82, 2020.

Aditya Bhatt, Max Argus, Artemij Amiranashvili, and Thomas Brox. Crossnorm: Normalization for off-policy td reinforcement learning. *arXiv preprint arXiv:1902.05605*, 10, 2019.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

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.

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 (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1373–1383. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ceron21a.html.

Johan Samir Obando Ceron, Aaron Courville, and Pablo Samuel Castro. In value-based deep reinforcement learning, a pruned network is a good network. In *Forty-first International Conference on Machine Learning*. PMLR, 2024a. URL https://openreview.net/forum?id=seo9V9QRZp.

Johan Samir Obando Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob Nicolaus Foerster, Gintare Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep RL. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=X9VMhfFxwn.

Stephanie CY Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. *arXiv preprint arXiv:1912.05663*, 2019.

John D Co-Reyes, Yingjie Miao, Daiyi Peng, Esteban Real, Sergey Levine, Quoc V Le, Honglak Lee, and Aleksandra Faust. Evolving reinforcement learning algorithms. *arXiv preprint arXiv:2101.03958*, 2021.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 933–941. JMLR.org, 2017.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan D. Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, J-M. Moret, Seb Noury, Federico Pesamosca, David G. Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, B. Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin A. Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602:414 – 419, 2022.

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.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 9104–9149. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/eimer23a.html.

Stefan Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks : the official journal of the International Neural Network Society*, 107:3–11, 2018.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep rl. In *Forty-first International Conference on Machine Learning*. PMLR, 2024.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alexander Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. 2018.

Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*, 2015.

Carles Gelada and Marc G Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3647–3655, 2019.

Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pp. 3734–3744. PMLR, 2021.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 7766–7792. PMLR, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2017.

Matteo Hessel, Joseph Modayil, H. V. Hasselt, T. Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Matteo Hessel, Hado van Hasselt, Joseph Modayil, and David Silver. On inductive biases in deep reinforcement learning. *CoRR*, abs/1907.02908, 2019. URL http://arxiv.org/abs/1907.02908.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9 (03):90–95, 2007.

Léonard Hussenot, Marcin Andrychowicz, Damien Vincent, Robert Dadashi, Anton Raichuk, Sabela Ramos, Nikola Momchev, Sertan Girgin, Raphael Marinier, Lukasz Stafiniak, et al. Hyperparameter selection for imitation learning. In *International Conference on Machine Learning*, pp. 4511–4522. PMLR, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

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.

Michael J. Kearns and Satinder P. Singh. Bias-variance error bounds for temporal difference updates. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, pp. 142–147, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 155860703X.

M. G. Kendall. A New Measure Of Rank Correlation. *Biometrika*, 30(1-2):81–93, 06 1938. ISSN 0006-3444. doi: 10.1093/biomet/30.1-2.81. URL https://doi.org/10.1093/biomet/30.1-2.81.

M. G. Kendall and B. Babington Smith. The Problem of m Rankings. *The Annals of Mathematical Statistics*, 10(3):275 – 287, 1939. doi: 10.1214/aoms/1177732186. URL https://doi.org/10.1214/aoms/1177732186.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas Kluyver, Benjain Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In *IOS Press*, pp. 87–90. 2016. doi: 10.3233/978-1-61499-649-1-87.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35: 16455–16468, 2022.

Cong Lu, Philip J Ball, Jack Parker-Holder, Michael A Osborne, and Stephen J Roberts. Revisiting design choices in offline model-based reinforcement learning. *arXiv preprint arXiv:2110.04135*, 2021.

Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning dynamics and generalization in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14560–14581. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/lyle22a.html.

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

Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 1 edition, February 2013. ISBN 9789351100065. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1449319793.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. 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. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Johan Obando Ceron, Marc Bellemare, and Pablo Samuel Castro. Small batch deep reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 26003–26024. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/528388f1ad3a481249a97cbb698d2fe6-Paper-Conference.pdf.

Travis E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3): 10–20, 2007. doi: 10.1109/MCSE.2007.58.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=nPHA8fGicZk.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, et al. Automated reinforcement learning (autorl): A survey and open problems. *Journal of Artificial Intelligence Research*, 74: 517–568, 2022.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

T. Schaul, John Quan, Ioannis Antonoglou, and D. Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.

Max Schwarzer, Johan Samir Obando-Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30365–30380. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/schwarzer23a.html.

G. Shamir, Dong Lin, and Lorenzo Coviello. Smooth activations and reproducibility in deep networks. *ArXiv*, abs/2010.09931, 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 10 2017. doi: 10.1038/nature24270.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning*

Research, pp. 32145–32168. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sokar23a.html.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.

Adrien Ali Taiga, Rishabh Agarwal, Jesse Farebrother, Aaron Courville, and Marc G Bellemare. Investigating multi-task pretraining and generalization in reinforcement learning. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022. URL https://openreview.net/forum?id=NEtI9o7gtPL.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirthieth AAAI Conference On Artificial Intelligence (AAAI), 2016*, 2016. cite arxiv:1509.06461Comment: AAAI 2016.

Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1b742ae215adf18b75449c6e272fd92d-Paper.pdf.

Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Harm Van Seijen, Mehdi Fatemi, and Arash Tavakoli. Using a logarithmic mapping to enable lower discount factors in reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1995–2003, 2016.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

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*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

A Computing Tuning Hyperparameter Consistency (THC)

Computing the ranking between hyper-parameter values is non-trivial given the noise involved in Deep Reinforcement Learning Agents performances. We used 5 seeds to improve the robustness to noise of our results in this paper, but if we simply used the average performance the effects of noise would still be significant. As such our ranking is based on the Inter-Quantile Mean (IQM) (Agarwal et al., 2021) and its 95% confidence interval.

First we sort the performance array in decreasing order based on the upper bound of the confidence interval for each hyper-parameter. Then we compute the rank of each hyper-parameter as the average between the lowest position (1-based) whose lower bound is less than or equal to the current hyper-parameter's performance upper bound and the highest position whose upper bound is greater than or equal to the current hyper-parameter's lower bound. Our choice of treating overlaps in performance by averaging the rankings comes from what is typically done when dealing with ties when computing Kendall's W and Kendall's τ, which are other commonly used metrics for inter-ranking agreement.

As an example imagine we are analyzing a hyper-parameter with 5 possible values, 1e-2, 1e-1, 1, 1e1, 1e2. We run all the experiments and get the following confidence intervals on their IQM ranges $(200, 300), (250, 350), (400, 600), (110, 220), (30, 70)$. After sorting them we're left with:

1. 1: (400, 600)
2. 10^{-1}: (250, 350)
3. 10^{-2}: (200, 300)
4. 10^{1}: (110, 220)
5. 10^{2}: (30, 70)

Then we can compute the ranks as:

1. $1 : \frac{1+1}{2} = 1$
2. $10^{-1} : \frac{2+3}{2} = 2.5$
3. $10^{-2} : \frac{2+4}{2} = 3$
4. $10^{1} : \frac{3+4}{2} = 3.5$
5. $10^{2} : \frac{5+5}{2} = 5$

An important feature of this method is that ranks needs not be integers. Now another relevant example is one where the 3 values, let's call them A, B, C, have completely overlapping intervals:

1. A: (200, 300)
2. B: (250, 350)
3. C: (180, 260)

In this case all of them will have the ranking $\frac{1+3}{2} = 2$, which shows how given our results we're unable to fully determine which one is the best or worst performing value for this hyper-parameter.

Here are two extra examples of computing the THC score.

1. We analyze a case with 2 hyper-parameters, A_1 and B_1, both with 3 values, being evaluated across 5 games (columns are ranks in a game):

$$r_{A_1} = \begin{bmatrix} 1 & 1 & 2 & 1 & 3 \\ 2 & 3 & 2 & 3 & 2 \\ 3 & 2 & 2 & 2 & 1 \end{bmatrix} \xrightarrow{\text{peak-to-peak}} ptp_{A_1} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} \xrightarrow{\text{Normalize}} ptp_{A_1} = \begin{bmatrix} 1.0 \\ 0.5 \\ 1.0 \end{bmatrix}$$
$$r_{B_1} = \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix} \qquad ptp_{B_1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \qquad ptp_{B_1} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.0 \end{bmatrix}$$

   Finally we average the values to get the THC for each hyper-parameter:

$$THC_{A_1} = \frac{2.5}{3} \approx 0.83333 \qquad THC_{B_1} = \frac{1.0}{3} \approx 0.33333 \tag{3}$$

   This example also shows an important property of THC, while a_1 seems to be consistently the best value for A, whereas b_1 and b_2 vary their position more often, the value of THC is higher for A then for B, since the largest change in performance for values of A is larger than the change for values of B. This is because THC considers the worst-case variance when assigning how important is tuning a given hyper-parameter.

2. Another example, now one hyper-parameter, A_2, has 4 possible values and the other, B_2, has 3, and we have 4 games.

$$r_{A_2} = \begin{bmatrix} 1 & 1 & 1 & 3 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 1 \\ 4 & 4 & 4 & 4 \end{bmatrix} \xrightarrow{\text{peak-to-peak}} ptp_{A_2} = \begin{bmatrix} 2 \\ 0 \\ 2 \\ 0 \end{bmatrix} \xrightarrow{\text{Normalize}} ptp_{A_2} = \begin{bmatrix} \frac{2}{3} \\ 0.0 \\ \frac{2}{3} \\ 0.0 \end{bmatrix}$$
$$r_{B_2} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2.5 & 2 & 3 & 2 \\ 2.5 & 3 & 2 & 3 \end{bmatrix} \qquad ptp_{B_2} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \qquad ptp_{B_2} = \begin{bmatrix} 0.0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

   And then average across the hyper-parameter values:

$$THC_{A_2} = \frac{\frac{4}{3}}{4} = \frac{1}{3} \qquad THC_{B_2} = \frac{1}{3} \tag{4}$$

   In this case we see that while A_2 has 2 hyper-parameter values with more variance in ranking then the 2 values of B_2 the fact that A_2 has more values overall than B_2 leads them to having the same THC value.

Finally it's worth pointing out that since the performances in the second case were more stable than in the first one their THC value was overall lower.

B Finer-grained experimental discussion

B.1 Optimal hyper-parameters do not Transfer Across Environments

1. For batch size in DrQ(ϵ)@40M we find that 4 is the optimal batch size for Asterix, Breakout, Gopher, and Seaquest, while being the worst value for effectively all the other games. See Figure 3

2. Convolutional width for DER@40M, 0.25 is the clear optimum in Assault, CrazyClimber, Roadrunner, Seaquest, and UpNDown, while leading to the worst performance in Breakout, Krull, and QBert

3. Dense layer width for DrQ(ϵ)@40M we see that 768 neurons per layer lead to best performance for Amidar, Assault, Hero, and Qbert, while most other games have 128 neurons as their optimal layer width. We see a similar mismatch for DER, though the games were 768 is optimal are different.

4. A discount factor of 0.99 is optimal for DER@40M in Alien, Amidar, Asterix, BankHeist, Breakout, Frostbite, Kangaroo, Kung Fu Master, QBert, RoadRunner, Seaquest, and UpNDown, but leads to pessimal performance in PrivateEye and non-optmimum in Assault, Boxing, ChopperCommand, CrazyClimber and many others.

B.2 Optimal hyper-parameters do not Transfer Across Data Regimes

1. Adam's ϵ, an often overlooked hyper-parameter, has optimal values $< 1.5 \cdot 10^{-4}$ for Atari 100k, while having optimal value of $1.5 \cdot 10^{-2}$ in the 40M setting. This result also begs for further research, as higher values of ϵ move Adam closer to SGD with momentum behaviour.

2. For Convolutional Width we find that the worst performing value for 100k, 0.25, is the optimal value when number of updates is 40M. Another important result given that it means one may want to effectively change the network architecture when the number of updates changes.

3. For normalization of the dense layers we see that while in the 100k regime Layer Norm leads to worse performance than no normalization, it is the best performing normalization once we move to the 40M regime.

4. For update horizon one can see that the best performing values are high, around 10, in the 100k regime, while lower values (as low as 1 for DER) are optimal in the 40M regime.

5. For update period we see that in the 100k regime a value of 6 is low performing and 1 is optimal, but once we move to the 40M regime we see an inversion, where 6 is substantially superior to 1.

C Hyper-parameters list

Default hyper-parameter settings for DER and DrQ(ϵ) across the environments. Table 1 shows the default values for each hyper-parameter across all the Atari games. In Table 2 we list all the possible values we explored for both agents. The values selection is informed by the recommendations provided by Araújo et al. (2021).

Table 1: Default hyper-parameters setting for DER and DrQ(ϵ) agents.

	Atari	
Hyper-parameter	DER	DrQ(ϵ)
Adam's(ϵ)	0.00015	0.00015
Batch Size	32	32
Conv. Activation Function	ReLU	ReLU
Convolutional Normalization	None	None
Convolutional Width	1	1
Dense Activation Function	ReLU	ReLU
Dense Normalization	None	None
Dense Width	512	512
Discount Factor	0.99	0.99
Exploration ϵ		
Learning Rate	0.0001	0.0001
Minimum Replay History		
Number of Atoms	51	0
Number of Convolutional Layers		
Number of Dense Layers	2	2
Replay Capacity	1000000	1000000
Reward Clipping	True	True
Update Horizon	10	10
Update Period	1	1
Weight Decay	0	0

Table 2: Hyper-parameters settings for DER and DrQ(ϵ) agents

Hyper-parameter	Values
Adam's(ϵ)	1, 0.5, 0.3125, 0.03125, 0.003125, 0.0003125, 3.125e-05, 3.125e-06
Batch Size	4, 8, 16, 32, 64
Conv. Activation Function	ReLU, ReLU6, Sigmoid, Softplus, Soft sign, SiLU, Log Sigmoid, Hard Sigmoid, Hard SiLU, Hard tanh, ELU, CELU, SELU, GELU, GLU
Convolutional Normalization	None, BatchNorm, LayerNorm
Convolutional Width	0.25, 0.5, 1, 2, 4
Dense Activation Function	ReLU, ReLU6, Sigmoid, Softplus, Soft sign, SiLU, Log Sigmoid, Hard Sigmoid, Hard SiLU, Hard tanh, ELU, CELU, SELU, GELU, GLU
Dense Normalization	None, BatchNorm, LayerNorm
Dense Width	32, 64, 128, 256, 512, 1024
Discount Factor	0.1, 0.5, 0.9, 0.99, 0.995, 0.999
Exploration ϵ	0, 0.001, 0.005, 0.01, 0.1
Learning Rate	10, 5, 2, 1, 0.1, 0.01, 0.001, 0.0001, 1e-05
Minimum Replay History	125, 250, 375, 500, 625, 750, 875, 1000
Number of Atoms	11, 21, 31, 41, 51, 61
Number of Convolutional Layers	1, 2, 3, 4
Number of Dense Layers	1, 2, 3, 4
Replay Capacity	
Reward Clipping	True, False
Target Update Period	10, 25, 50, 100, 200, 400, 800, 1600
Update Horizon	1, 2, 3, 4, 5, 8, 10
Update Period	1, 2, 3, 4, 8, 10, 12
Weight Decay	0, 0.01, 0.03, 0.1, 0.5, 1

D Interesting Miscellaneous Findings

There were a couple of interesting findings from our experiments which are out of scope for this paper, but which may warrant further exploration in the future.

D.1 High Values of Weight Decay Can Be Optimal

We found that for DER at 40 Million environment frames having a weight decay of 0.1 was the overall best choice, and that for many games like Gopher and Boxing the optimal value was 0.5, an uncommonly high value for the hyperparameter.

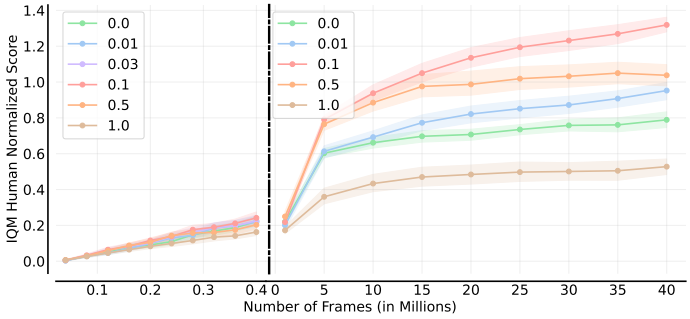

Figure 5: **Measured IQM of human-normalized scores on the 26 100k benchmark games, with varying Weight Decay** for DER. We evaluate performance at 100k agent steps (or 400k environment frames), and at 40 million environment frames. At 40 million frames 0.1 is on average optimal, with 0.5 being at second place and the standard value of 0.0 being in fourth.

D.2 Higher Values of Adam's ϵ can improve Performance

In our experiments we found that both DrQ(ϵ) and DER can benefit from a 100 times higher value of Adam's ϵ than what is commonly used. This is somewhat perplexing, as using such a high value of epsilon leads Adam to behave closer to SGD than to its common behaviour in other settings.

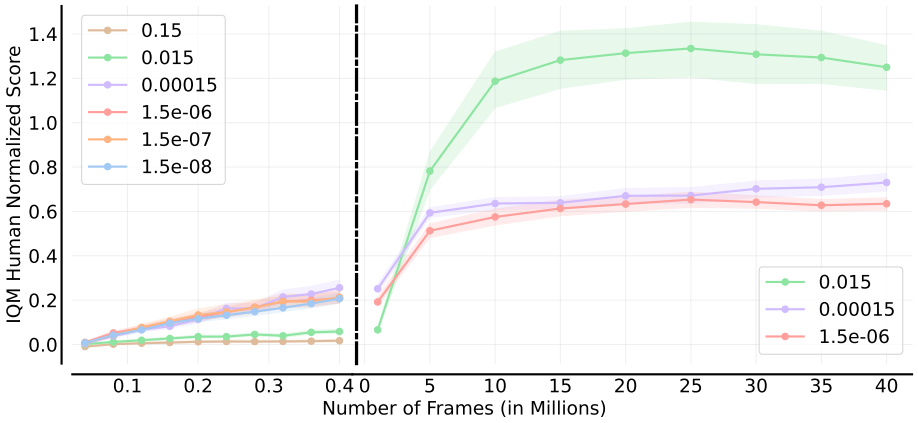

Figure 6: **Measured IQM of human-normalized scores on the 26 100k benchmark games, with varying Adam's ϵ** for DER. We evaluate performance at 100k agent steps (or 400k environment frames), and at 40 million environment frames.

D.3 Example Analysis: DER on Gopher

Finally we found that in a specific experimental setting, DER with 40 million frames on Gopher, whose optimal hyperparameters are very different from what is commonly observed in other applications of Deep Learning, and in some cases quite different even from the optimal values when using DER with 40 million environment frames in other Atari games. Not only that, but also we observed that often the difference in performance between the counter-intuitive optimal hyper-parameter and the standard is significant, leading to multiple-fold improvement in returns. For example in Gopher specifically we find that:

- For DER the standard value of update horizon is 10, but in the case of Gopher using an update horizon of 1 leads to roughly a 28 times improvement in performance.
- In Gopher a Weight Decay of 0.5 lead to a 5-fold increase of returns when compared to the standard value of 0.
- While the standard value of the Discount Factor is 0.99, for Gopher we see a 4.5 times improvement in performance when using a lower value of 0.9
- The optimal batch size we found was 4, which is relatively small compared to the standard of 32, and goes against the common Deep Learning practice of increasing batch sizes to increase performance. Changing batch size to 4 leads to a 4.5-fold increase in returns
- Finally, we recall the previous sub-section on Adam's ϵ and see that Gopher also benefits from an uncommonly high value of the hyperparameter, though here the performance gap is smaller, being closer to a 2x increase compared to the considerable differences discussed previously.

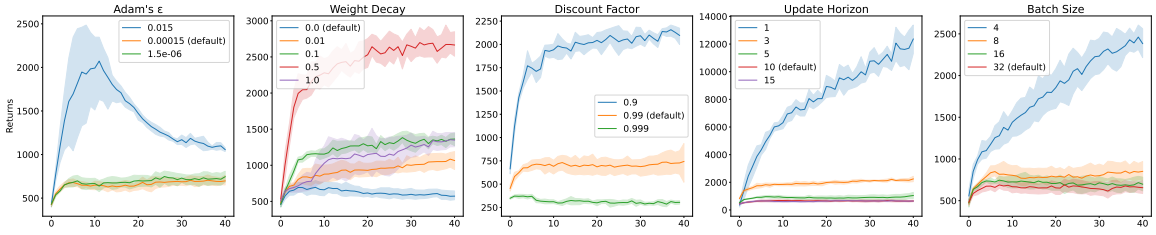

Figure 7: **Learning Curves of DER on Gopher at 40M frames as we vary Adam's ϵ, Weight Decay, Discount Factor, Update Horizon, and Batch Size**