Deep Reinforcement Learning at the Edge of the Statistical Precipice

Rishabh Agarwal*
Google Research, Brain Team
MILA, Université de Montréal

Max Schwarzer
MILA, Université de Montréal

Pablo Samuel Castro
Google Research, Brain Team

Aaron Courville
MILA, Université de Montréal

Marc G. Bellemare
Google Research, Brain Team

Abstract

Deep reinforcement learning (RL) algorithms are predominantly evaluated by comparing their relative performance on a large suite of tasks. Most published results on deep RL benchmarks compare *point estimates* of aggregate performance such as mean and median scores across tasks, ignoring the statistical uncertainty implied by the use of a finite number of training runs. Beginning with the Arcade Learning Environment (ALE), the shift towards computationally-demanding benchmarks has led to the practice of evaluating only a small number of runs per task, exacerbating the statistical uncertainty in point estimates. In this paper, we argue that reliable evaluation in the few-run deep RL regime cannot ignore the uncertainty in results without running the risk of slowing down progress in the field. We illustrate this point using a case study on the Atari 100k benchmark, where we find substantial discrepancies between conclusions drawn from point estimates alone versus a more thorough statistical analysis. With the aim of increasing the field's confidence in reported results with *a handful of runs*, we advocate for reporting interval estimates of aggregate performance and propose performance profiles to account for the variability in results, as well as present more robust and efficient aggregate metrics, such as interquartile mean scores, to achieve small uncertainty in results. Using such statistical tools, we scrutinize performance evaluations of existing algorithms on other widely used RL benchmarks including the ALE, Procgen, and the DeepMind Control Suite, again revealing discrepancies in prior comparisons. Our findings call for a change in how we evaluate performance in deep RL, for which we present a more rigorous evaluation methodology, accompanied with an open-source library *rliable*[2], to prevent unreliable results from stagnating the field.

1 Introduction

Research in artificial intelligence, and particularly deep reinforcement learning (RL), relies on evaluating *aggregate* performance on a diverse suite of tasks to assess progress. Quantitative evaluation on a suite of tasks, such as Atari games [5], reveals strengths and limitations of methods while simultaneously guiding researchers towards methods with promising results. Performance of RL algorithms is usually summarized with a *point estimate* of task performance measure, such as mean and median performance across tasks, aggregated over independent training runs.

A small number of training runs (Figure 1) coupled with high variability in performance of deep RL algorithms [16, 17, 41, 68, 70], often leads to substantial statistical uncertainty in reported point

*Outstanding Paper Award. Correspondence to Rishabh <rishabhagarwal@google.com>.

[2] https://github.com/google-research/rliable

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

estimates. While evaluating more runs per task has been prescribed to reduce uncertainty and obtain reliable estimates [20, 41, 49], 3-10 runs are prevalent in deep RL as it is often computationally prohibitive to evaluate more runs. For example, 5 runs each on 50+ Atari 2600 games in ALE using standard protocol requires more than 1000 GPU training days [15]. As we move towards more challenging and complex RL benchmarks (*e.g.,* StarCraft [110]), evaluating more than a handful of runs will become increasingly demanding due to increased amount of compute and data needed to tackle such tasks. Additional confounding factors, such as exploration in the low-data regime, exacerbates the performance variability in deep RL – as seen on the Atari 100k benchmark [50] – often requiring many more runs to achieve negligible statistical uncertainty in reported estimates.

Ignoring the statistical uncertainty in deep RL results gives a false impression of fast scientific progress in the field. It inevitably evades the question: "Would similar findings be obtained with new independent runs under different random conditions?" This could steer researchers towards superficially beneficial methods [11, 12, 25], often at the expense of better methods being neglected or even rejected early [67, 74] as such methods fail to outperform inferior methods simply due to less favorable random conditions. Furthermore, only reporting point estimates obscures nuances in comparisons [85] and can erroneously lead the field to conclude which methods are *state-of-the-art* [63, 84], ensuing wasted effort when applied in practice [108]. Moreover, not reporting the uncertainty in deep RL results makes them difficult to reproduce except under the *exact* same random conditions, which could lead to a *reproducibility crisis* similar to the one that plagues other fields [4, 44, 78]. Finally, unreliable results could erode trust in deep RL research itself [45].

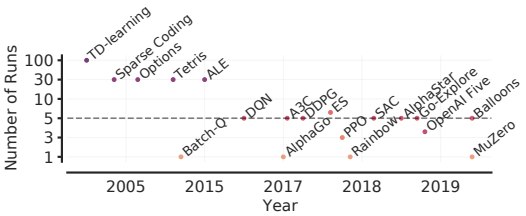


Figure 1: **Number of runs in RL over the years**. Beginning with DQN [75] on the ALE, 5 or less runs are common in the field. Here, we show representative RL papers with empirical results, in the order of their publication year: TD-learning [99], Sparse coding [100], Options [102], Tetris (CEM) [103], Batch-Q [31], ALE [5], DQN [75], AlphaGo [96], A3C [76], DDPG [62], ES [88], PPO [92], SAC [36], Rainbow [42], AlphaStar [110], Go-Explore [28], OpenAI Five [8], Balloon navigation [7] and MuZero [91].

In this work, we show that recent deep RL papers compare unreliable point estimates, which are dominated by statistical uncertainty, as well as exploit non-standard evaluation protocols, using a case study on Atari 100k (Section 3). Then, we illustrate how to reliably evaluate performance with only *a handful of runs* using a more rigorous evaluation methodology that accounts for uncertainty in results (Section 4). To exemplify the necessity of such methodology, we scrutinize performance evaluations of existing algorithms on widely used benchmarks, including the ALE [5] (Atari 100k, Atari 200M), Procgen [18] and DeepMind Control Suite [104], again revealing discrepancies in prior comparisons (Section 5). Our findings call for a change in how we evaluate performance in deep RL, for which we present a better methodology to prevent unreliable results from stagnating the field.

How do we reliably evaluate performance on deep RL benchmarks with only a handful of runs? As a practical solution that is easily applicable with 3-10 runs per task, we identify three statistical tools (Table 1) for improving the quality of experimental reporting. Since any performance estimate based on a finite number of runs is a *random variable*, we argue that it should be treated as such. Specifically, we argue for reporting aggregate performance measures using *interval estimates* via stratified bootstrap confidence intervals, as opposed to point estimates. Among prevalent aggregate measures, mean can be easily dominated by performance on a few outlier tasks, while median has high variability and zero performance on nearly half of the tasks does not change it. To address these deficiencies, we present more *efficient* and *robust* alternatives, such as *interquartile mean*, which are not unduly affected by outliers and have small uncertainty even with a handful of runs. Furthermore, to reveal the variability in performance across tasks, we propose reporting performance distributions across all runs. Compared to prior work [5, 83], these distributions result in *performance profiles* [26] that are statistically unbiased, more robust to outliers, and require fewer runs for smaller uncertainty.

2 Formalism

We consider the setting in which a reinforcement learning algorithm is evaluated on M tasks. For each of these tasks, we perform N independent runs[3] which each provide a scalar, *normalized score*

[3] A run can be different from using a fixed random seed. Indeed, fixing the seed may not be able to control all sources of randomness such as non-determinism of ML frameworks with GPUs (*e.g.,* Figure A.13).

Table 1: Our recommendations for reliable evaluation, easily applicable with a handful of runs. Refer to Section 4 for details about recommendations and Section 5 for their application to widely-used RL benchmarks.

Desideratum	Current Evaluation Protocol	Our Recommendation
Uncertainty in aggregate performance	**Point estimates** • Ignore statistical uncertainty • Hinder *results reproducibility*	Interval estimates via **stratified bootstrap confidence intervals**
Variability in performance across tasks and runs	**Tables with mean scores per task** • Overwhelming beyond a few tasks • Standard deviations often omitted • Incomplete picture for multimodal and heavy-tailed distributions	**Performance profiles** (*score distributions*) • Show tail distribution of scores on combined runs across tasks • Allow qualitative comparisons • Easily read any score percentile
Aggregate metrics for summarizing performance across tasks	**Mean** • Often dominated by performance on outlier tasks **Median** • Requires large number of runs to claim improvements • Poor indicator of overall performance: zero scores on nearly half the tasks do not affect it	**Interquartile Mean** (IQM) across all runs • Performance on middle 50% of combined runs • Robust to outlier scores but more statistically efficient than median To show other aspects of performance gains, report average *probability of improvement* and *optimality gap*.

$x_{m,n}$, $m = 1, \ldots, M$ and $n = 1, \ldots, N$. These normalized scores are obtained by linearly rescaling per-task scores[4] based on two reference points; for example, performance on the Atari games is typically normalized with respect to a random agent and an average human, who are assigned a normalized score of 0 and 1 respectively [75]. We denote the set of normalized scores by $x_{1:M,1:N}$.

In most experiments, there is inherent randomness in the scores obtained from different runs. This randomness can arise from stochasticity in the task, exploratory choices made during learning, randomized initial parameters, but also software and hardware considerations such as non-determinism in GPUs and in machine learning frameworks [116]. Thus, we model the algorithm's normalized score on the mth task as a real-valued random variable X_m. Then, the score $x_{m,n}$ is a realization of the random variable $X_{m,n}$, which is identically distributed as X_m. For $\tau \in \mathbb{R}$, we define the tail distribution function of X_m as $F_m(\tau) = \mathrm{P}(X_m > \tau)$. For any collection of scores $y_{1:K}$, the *empirical tail distribution function* is given by $\hat{F}(\tau; y_{1:K}) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}[y_k > \tau]$. In particular, we write $\hat{F}_m(\tau) = \hat{F}(\tau; x_{m,1:N})$.

The *aggregate performance* of an algorithm maps the set of normalized scores $x_{1:M,1:N}$ to a scalar value. Two prevalent aggregate performance metrics are the mean and median normalized scores. If we denote by $\bar{x}_m = \frac{1}{N}\sum_{n=1}^{N} x_{m,n}$ the average score on task m across N runs, then these aggregate metrics are $\mathrm{Mean}(\bar{x}_{1:M})$ and $\mathrm{Median}(\bar{x}_{1:M})$. More precisely, we call these *sample mean* and *sample median* over the task means since they are computed from a finite set of N runs. Since $\bar{x}_m$ is a realization of the random variable $\bar{X}_m = \frac{1}{N}\sum_{n=1}^{N} X_{m,n}$, the sample mean and median scores are *point estimates* of the random variables $\mathrm{Mean}(\bar{X}_{1:M})$ and $\mathrm{Median}(\bar{X}_{1:M})$ respectively. We call *true mean* and *true median* the metrics that would be obtained if we had unlimited experimental capacity ($N \to \infty$), given by $\mathrm{Mean}(\mathbb{E}[X_{1:M}])$ and $\mathrm{Median}(\mathbb{E}[X_{1:M}])$ respectively.

Confidence intervals (CIs) for a finite-sample score can be interpreted as an estimate of plausible values for the true score. A $\alpha \times 100\%$ CI computes an interval such that if we rerun the experiment and construct the CI using a different set of runs, the fraction of calculated CIs (which would differ for each set of runs) that contain the true score would tend towards $\alpha \times 100\%$, where $\alpha \in [0, 1]$ is the nominal coverage rate. 95% CIs are typically used in practice. If the true score lies outside the 95% CI, then a sampling event has occurred which had a probability of 5% of happening by chance.

[4] Often the average undiscounted return obtained during an episode (see Sutton and Barto [101] for an explanation of the reinforcement learning setting).

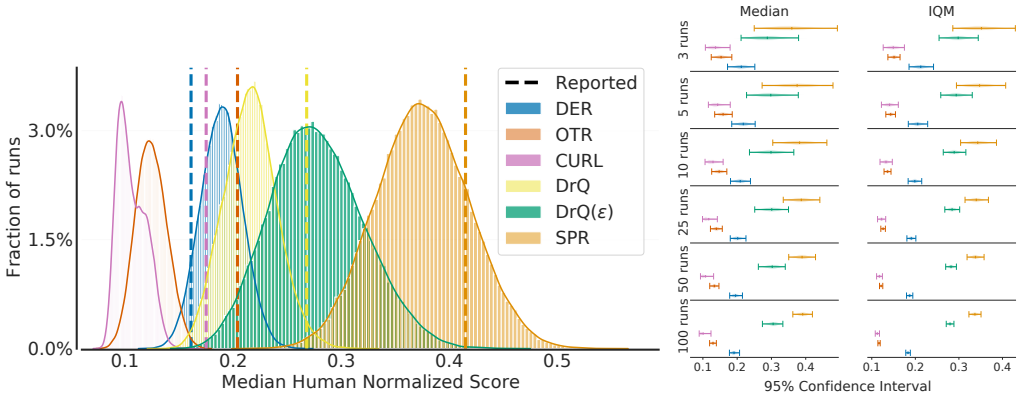


Figure 2: **Left**. **Distribution of median normalized scores** computed using 100,000 different sets of N runs subsampled uniformly with replacement from 100 runs. For a given algorithm, the sampling distribution shows the variation in the median scores when re-estimated using a different set of runs. The reported *point estimates* of median in publications, as shown by dashed lines, do not provide any information about the variability in median scores and severely overestimate or underestimate the expected median. We use the same number of runs as reported by publications: $N = 5$ runs for DER, OTR and DrQ, $N = 10$ runs for SPR and $N = 20$ runs for CURL. **Right**. **95% CIs** for median and IQM scores (Section 4.3) for varying N. There is a substantial uncertainty in median scores even with 50 runs. IQM has much smaller CIs than median. Note that when CIs overlap, properly accounting for uncertainty entails computing CIs for score differences (Figure A.15).

Remark. Following Amrhein et al. [2], Romer [87], Wasserstein et al. [112], we recommend using confidence intervals for measuring the uncertainty in results and showing effect sizes (*e.g.,* performance improvements over baseline) that are compatible with the given data. Furthermore, we emphasize using statistical thinking but avoid statistical significance tests (*e.g.,* p-value < 0.05) because of their dichotomous nature (significant *vs.* not significant) and common misinterpretations [33, 35, 73] such as 1) lack of statistically significant results does not demonstrate the absence of effect (Figure 2, right), and 2) given enough data, any trivial effect can be statistically significant but may not be practically significant.

3 Case Study: The Atari 100k benchmark

We begin with a case study to illustrate the pitfalls arising from the naïve use of point estimates in the few-run regime. Our case study concerns the Atari 100k benchmark [50], an offshoot of the ALE for evaluating data-efficiency in deep RL. In this benchmark, algorithms are evaluated on only 100k steps (2-3 hours of game-play) for each of its 26 games, versus 200M frames in the ALE benchmark. Prior reported results on this benchmark have been computed mostly from 3 [39, 55, 59, 72, 89, 95] or 5 runs [50, 51, 53, 54, 64, 66, 86, 107, 115], and more rarely, 10 [65, 93] or 20 runs [56].

Our case study compares the performance of five recent deep RL algorithms, namely: (1) DER [107] and (2) OTR [51], (3) DrQ[5] [53], (4) CURL [56], and (5) SPR [93]. We chose these methods as representative of influential algorithms within this benchmark. Since good performance on one game can result in unduly high sample means without providing much information about performance on other games, it is common to measure performance on Atari 100k using sample medians. Refer to Appendix A.2 for more details about the experimental setup.

We investigate statistical variations in the few-run regime by evaluating 100 independent runs for each algorithm, where the score for a run is the average returns obtained in 100 evaluation episodes taking place after training. Each run corresponds to training one algorithm on each of the 26 games in Atari 100k. This provides us with 26×100 scores per algorithm, which we then subsample with replacement to 3–100 runs. The subsampled scores are then used to produce a collection of point estimates whose statistical variability can be measured. We begin by using this experimental protocol to highlight statistical concerns regarding median normalized scores.

High variability in reported results. Our first observation is that the sample medians reported in the literature exhibit substantial variability when viewed as random quantities that depend on a

[5] DrQ codebase uses non-standard evaluation hyperparameters. Instead, DrQ(ε) corresponds to DrQ with standard ε-greedy parameters [14, Table 1] in ALE. See Appendix for more details.

small number of sample runs (Figure 2, left). This shows that there is a fairly large potential for drawing erroneous conclusions based on point estimates alone. As a concrete example, our analysis suggests that DER may in fact be better than OTR, unlike what the reported point estimates suggest. We conclude that in the few-run regime, point estimates are unlikely to provide definitive answers to the question: "Would we draw the same conclusions were we to re-evaluate our algorithm with a different set of runs?"

Substantial bias in sample medians. The sample median is a biased estimator of the true median: $\mathbb{E}[\text{Median}(\bar{X}_{1:M})] \neq \text{Median}(\mathbb{E}[X_{1:M}])$ in general. In the few-run regime, we find that this bias can dominate the comparison between algorithms, as evidenced in Figure 3. For example, the score difference between sample medians with 5 and 100 runs for SPR (+0.03 points) is about 36% of its mean improvement over DrQ(ε) (+0.08 points). Adding to the issue, the magnitude and sign of this bias strongly depends on the algorithm being evaluated.

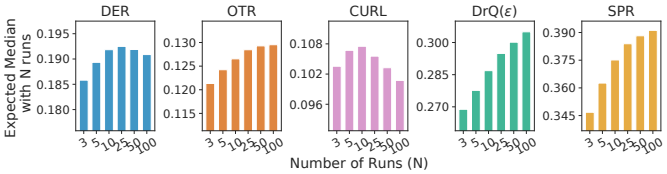


Figure 3: **Expected sample median** of task means. The expected score for N runs is computed by repeatedly subsampling N runs with replacement out of 100 runs for 100,000 times.

Statistical concerns cannot be satisfactorily addressed with few runs. While claiming improvements with 3 or fewer runs may naturally raise eyebrows, folk wisdom in experimental RL suggests that 20 or 30 runs are enough. By calculating 95% confidence interval[6] on sample medians for a varying number of runs (Figure 2, right), we find that this number is closer to 50–100 runs in Atari 100k – far too many to be computationally feasible for most research projects.

Consider a setting in which an algorithm is known to be better – what is the reliability of median and IQM (Section 4.3) for accurately assessing performance differences as the number of runs varies? Specifically, we consider two identical N-run experiments involving SPR, except that we artificially inflate one of the experiments' scores by a fixed fraction or *lift* of $+\ell\%$ (Figure 4). In particular, $\ell = 0$ corresponds to running the same experiment twice but with different runs. We find that statistically defensible improvements with median scores is only achieved for 25 runs ($\ell = 25$) and 100 runs ($\ell = 10$). With $\ell = 0$, even 100 runs are insufficient, with deviations of 20% possible.

Changes in evaluation protocols invalidates comparisons to prior work. A typical and relatively safe approach for measuring the performance of an RL algorithm is to average the scores received in their final training episodes [69]. However, the field has seen a number of alternative protocols used, including reporting the maximum evaluation score achieved during training [1, 3, 75] or across multiple runs [32, 47, 82]. A similar protocol is also used by CURL and SUNRISE [59] (Appendix A.4).

Results produced under alternative protocols involving maximum are generally incomparable with end-performance reported results. On Atari 100k, we find that the two protocols produce substantially different results (Figure 5), of a magnitude greater than the actual difference in score. In particular, evaluating DER with CURL's protocol results in scores far above those reported for CURL. In other words, this gap in evaluation procedures resulted in CURL being assessed as achieving a greater true median than DER, where our experiment gives strong support to DER being superior. Similarly, we find that a lot of SUNRISE's improvement over DER can be explained by the change in evaluation protocol (Figure 5). Refer to Appendix A.4 for discussion on pitfalls of such alternative protocols.

4 Recommendations and Tools for Reliable Evaluation

Our case study shows that the increase in the number of runs required to address the statistical uncertainty issues is typically infeasible for computationally demanding deep RL benchmarks. In this section, we identify three tools for improving the quality of experimental reporting in the few-run regime, all aligned with the principle of accounting for statistical uncertainty in results.

4.1 Stratified Bootstrap Confidence Intervals

We first reaffirm the importance of reporting interval estimates to indicate the range within which an algorithm's aggregate performance is believed to lie. Concretely, we propose using bootstrap CIs [29]

[6]Specifically, we use the m/n bootstrap [9] to calculate the interval between $[2.5^{th}, 97.5^{th}]$ percentiles of the distribution of sample medians (95% CIs).

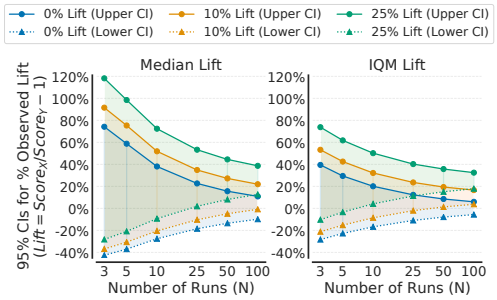


Figure 4: **Detecting score lifts**. **Left**. 95% CIs for observed lift with median scores, and **Right**. 95% CIs for observed lift with IQM (Section 4.3) when comparing SPR with an algorithm that performs $\ell\%$ better. IQM requires fewer runs than median for small uncertainty.

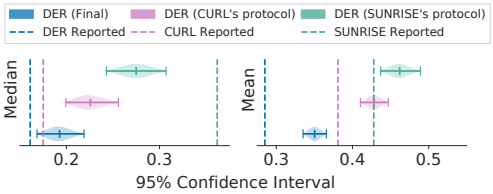


Figure 5: **Normalized DER scores** with non-standard evaluation protocols. Gains from SUNRISE and CURL over DER can mostly be explained by such protocols.

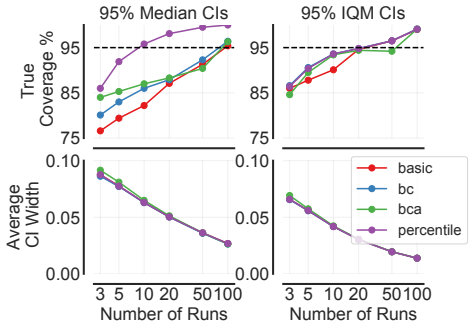


Figure 6: **Validating 95% Stratified Bootstrap CIs** for a varying number of runs for median and IQM scores for DER. The true coverage % is computed by sampling 10,000 sets of K runs without replacement from 200 runs and checking the fraction of 95% CIs that contains the true estimate approximation based on 200 runs. Note that we evaluate additional 100 runs for DER for an accurate point estimate. Percentile CIs has the best coverage while achieving a small width compared to other methods. Also, CI widths for IQM are much smaller than that of median. We also note that with 3 runs, bootstrap CIs underestimate the true 95% CIs and might require a larger nominal coverage rate to achieve true 95% coverage.

with stratified sampling for aggregate performance, a method that can be applied to small sample sizes and is better justified than reporting sample standard deviations in this context. While prior work has recommended using bootstrap CIs for reporting uncertainty in single task mean scores with N runs [16, 20, 41], this is less useful when N is small (Figure A.18), as *bootstrapping* assumes that re-sampling from the data approximates sampling from the true distribution. We can do better by aggregating samples across tasks, for a total of MN random samples.

To compute the stratified bootstrap CIs, we re-sample runs with replacement independently for each task to construct an empirical bootstrap sample with N runs each for M tasks from which we calculate a statistic and repeat this process many times to approximate the sampling distribution of the statistic. We measure the reliability of this technique in Atari 100k for variable N, by comparing the nominal coverage of 95% to the "true" coverage from the estimated CIs (Figure 6) for different bootstrap methods (see [30] and Appendix A.5). We find that percentile CIs provide good interval estimates for as few as $N = 10$ runs for both median and IQM scores (Section 4.3).

4.2 Performance Profiles

Most deep RL benchmarks yield scores that vary widely between tasks and may be heavy-tailed, multimodal, or possess outliers (*e.g.,* Figure A.14). In this regime, both point estimates, such as mean and median scores, and interval estimates of these quantities paint an incomplete picture of an algorithm's performance [24, Section 3]. Instead, we recommend the use of *performance profiles* [26], commonly used in benchmarking optimization software. While performance profiles from Dolan and Moré [26] correspond to empirical cumulative distribution functions without any uncertainty estimates, profiles proposed herein visualize the empirical tail distribution function (Section 2) of a random score (higher curve is better), with pointwise confidence bands based on stratified bootstrap.

By representing the entire set of normalized scores $x_{1:M,1:N}$ visually, performance profiles reveal performance variability across tasks much better than interval estimates of aggregate metrics. Although tables containing per-task mean scores and standard deviations can reveal this variability, such tables tend to be overwhelming for more than a few tasks.[7] In addition, performance profiles are robust to outlier runs and insensitive to small changes in performance across all tasks [26].

In this paper, we propose the use of a performance profile we call run-score distributions or simply *score distributions* (Figure 7, right), particularly well-suited to the few-run regime. A score distribution shows the fraction of runs above a certain normalized score and is given by

[7] In addition, standard deviations are sometimes omitted from tables due to space constraints.

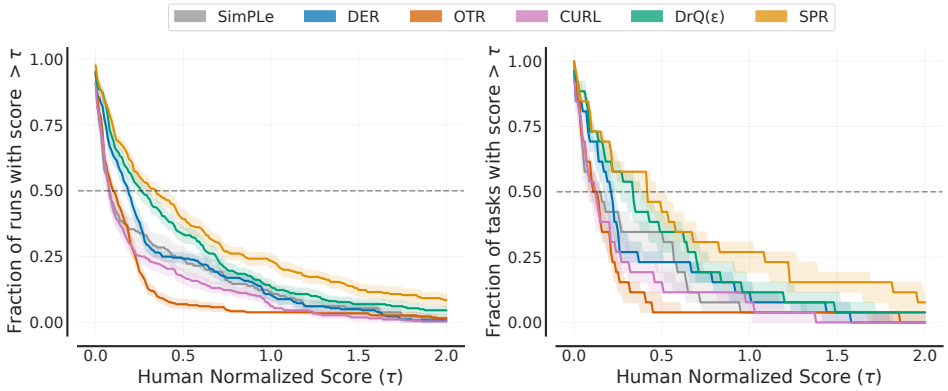


Figure 7: **Performance profiles on Atari 100k** based on score distributions **(left)**, which we recommend, and average score distributions **(right)**. Shaded regions show pointwise 95% confidence bands based on percentile bootstrap with stratified sampling. The profiles on the left are more robust to outliers and have smaller confidence bands. We use 10 runs to show the robustness of profiles with a few runs. For SimPLe [50], we use the 5 runs from their reported results. The τ value where the profiles intersect $y = 0.5$ shows the median while for a non-negative random variable, area under the performance profile corresponds to the mean.

$$\hat{F}_X(\tau) = \hat{F}(\tau; x_{1:M,1:N}) = \frac{1}{M}\sum_{m=1}^{M} \hat{F}_m(\tau) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{n=1}^{N} \mathbb{1}[x_{m,n} > \tau]. \tag{1}$$

One advantage of the score distribution is that it is an unbiased estimator of the underlying distribution $F(\tau) = \frac{1}{N}\sum_{m=1}^{M} F_m(\tau)$. Another advantage is that an outlier run with extremely high score can change the output of score distribution for any τ by at most a value of $\frac{1}{MN}$.

It is useful to contrast score distributions to average-score distributions, originally proposed in the context of the ALE [5] as a generalization of the median score. Average-score distributions correspond to the performance profile of a random variable $\bar{X}$, $\hat{F}_{\bar{X}}(\tau) = \hat{F}(\tau; \bar{x}_{1:M})$, which shows the fraction of tasks on which an algorithm performs better than a certain score. However, such distributions are a biased estimate of the thing they seek to represent. Run-score distributions are more robust than average-score distributions, as they are a step function in $1/MN$ versus $1/M$ intervals, and typically has less variance: $\sigma^2_X = \frac{1}{M^2 N}\sum_{m=1}^{M} F_m(\tau)(1 - F_m(\tau))$ versus $\sigma^2_{\bar{X}} = \frac{1}{M^2}\sum_{m=1}^{M} F_{\bar{X}_m}(\tau)(1 - F_{\bar{X}_m}(\tau))$. Figure 7 illustrates these differences.

4.3 Robust and Efficient Aggregate Metrics

Performance profiles allow us to compare different methods at a glance. If one curve is strictly above another, the better method is said to *stochastically dominate*[8] the other [27, 61]. In RL benchmarks with a large number of tasks, however, stochastic dominance is rarely observed: performance profiles often intersect at multiple points. Finer quantitative comparisons must therefore entail aggregate metrics.

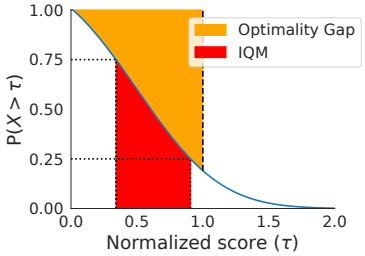


Figure 8: **Aggregate metrics**. For a non-negative random variable X, IQM corresponds to the red shaded region while optimality gap corresponds to the orange shaded region in the performance profile of X.

We can extract a number of aggregate metrics from score distributions, including median (mixing runs and tasks) and mean normalized scores (matching our usual definition). As we already argued that these metrics are deficient, we now consider interesting alternatives also derived from score distributions.

As an alternative to median, we recommend using the **interquartile mean** (IQM). Also called 25% trimmed mean, IQM discards the bottom and top 25% of the runs and calculates the mean score of the remaining 50% runs (=$\lfloor NM/2 \rfloor$ for N runs each on M tasks). IQM interpolates between mean and median across runs, which are 0% and almost 50% trimmed means

[8] A random variable X has stochastic dominance over random variable Y if $P(X > \tau) \geq P(Y > \tau)$ for all τ, and for some τ, $P(X > \tau) > P(Y > \tau)$.

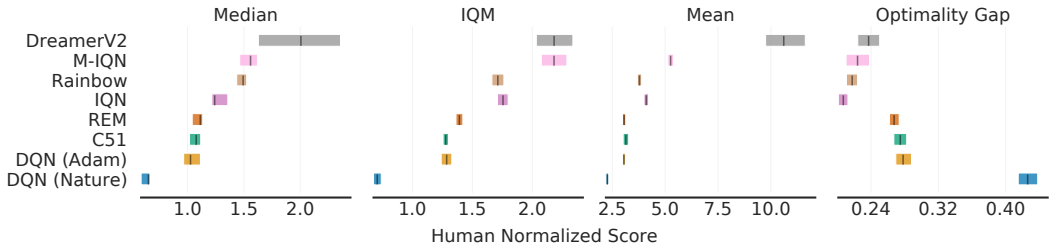


Figure 9: **Aggregate metrics on Atari 200M** with 95% CIs based on 55 games with sticky actions [69]. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling. IQM typically results in smaller CIs than median scores. Large values of mean scores relative to median and IQM indicate being dominated by a few high performing tasks, for example, DreamerV2 and M-IQN obtain normalized scores above 50 on the game JAMESBOND. Optimality gap is less susceptible to outliers compared to mean scores. We compare DQN (Nature) [75], DQN with Adam optimizer, C51 [6], REM [1], Rainbow [42], IQN [22], Munchausen-IQN (M-IQN) [109], and DreamerV2 [38]. All results are based on 5 runs per game except for M-IQN and DreamerV2 which report results with 3 and 11 runs.

respectively. Compared to sample median, IQM is a better indicator of overall performance as it is calculated using 50% of the combined runs while median only depends on the performance ordering across tasks and not on the magnitude except at most 2 tasks. For example, zero scores on nearly half of the tasks does not affect the median while IQM exhibits a severe degradation. Compared to mean, IQM is robust to outliers, yet has considerably less bias than median (Figure A.17). While median is more robust to outliers than IQM, this robustness comes at the expense of statistical efficiency, which is crucial in the few-run regime: IQM results in much smaller CIs (Figure 2 (right) and 6) and is able to detect a given improvement with far fewer runs (Figures 4 and A.15).

As a robust alternative to mean, we recommend using the **optimality gap**: the amount by which the algorithm fails to meet a minimum score of $\gamma = 1.0$ (orange region in Figure 8). This assumes that a score of 1.0 is a desirable target beyond which improvements are not very important, for example when the aim is to obtain human-level performance [*e.g.,* 3, 23]. Naturally, the threshold γ may be chosen differently, which we discuss further in Appendix A.7.

If one is interested in knowing how robust an improvement from an algorithm X over an algorithm Y is, another possible metric to consider is the average **probability of improvement** – this metric shows how likely it is for X to outperform Y on a randomly selected task. Specifically, $P(X > Y) = \frac{1}{M}\sum_{m=1}^{M} P(X_m > Y_m)$, where $P(X_m > Y_m)$ (Equation A.2) is the probability that X is better than Y on task m. Note that, unlike IQM and optimality gap, this metric does not account for the size of improvement. While finding the best aggregate metric is still an open question and is often dependent on underlying normalized score distribution, our proposed alternatives avoid the failure modes of prevalent metrics while being robust and requiring fewer runs to reduce uncertainty.

5 Re-evaluating Evaluation on Deep RL Benchmarks

Arcade Learning Environment. Training RL agents for 200M frames on the ALE [5, 69] is the most widely recognized benchmark in deep RL. We revisit some popular methods which demonstrated progress on this benchmark and reveal discrepancies in their findings as a consequence of ignoring the uncertainty in their results (Figure 9). For example, DreamerV2 [38] exhibits a large amount of uncertainty in aggregate scores. While M-IQN [109] claimed better performance than Dopamine Rainbow[9] [42] in terms of median normalized scores, their interval estimates strikingly overlap. Similarly, while C51 [5] is considered substantially better than DQN [75], the interval estimates as well as performance profiles for DQN (Adam) and C51 overlap significantly.

Figure 9 reveals an interesting limitation of aggregate metrics: depending on the choice of metric, the ordering between algorithms changes (*e.g.,* Median *vs.* IQM). The inconsistency in ranking across aggregate metrics arises from the fact that such metrics only capture a specific aspect of overall performance across tasks and runs. Additionally, the change of algorithm ranking between optimality gap and IQM/median scores reveal that while recent algorithms typically show performance gains relative to humans on average, their performance seems to be worse on games below human

[9] Dopamine Rainbow differs from that of Hessel et al. [42] by not including double DQN, dueling architecture and noisy networks. Also, results in [42] were reported using a single run without sticky actions.

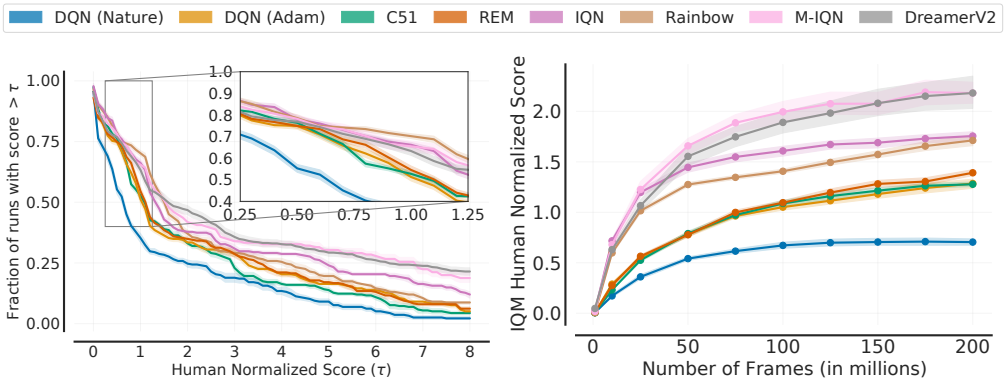
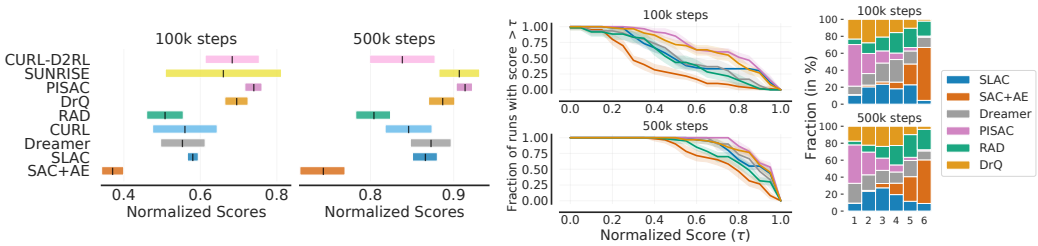


Figure 10: **Atari 200M evaluation**. **Left**. Score distributions using human-normalized scores obtained after training for 200M frames. **Right**. Sample-efficiency of agents as a function of number of frames measured via IQM human-normalized scores. Shaded regions show pointwise 95% percentile stratified bootstrap CIs.



(a) Mean Scores (= 1 - Optimality Gap) (b) Score Distributions. (c) Rank comparisons.

Figure 11: **DeepMind Control Suite evaluation** results, averaged across 6 tasks, on the 100k and 500k benchmark. We compare SAC+AE [114], SLAC [58], Dreamer [37], CURL [98], RAD [57], DrQ [53], PISAC [60], SUNRISE [59], and CURL-D2RL [97]. The **ordering** of the algorithms in the left figure is based on their claimed relative performance – all algorithms except Dreamer claimed improvement over at least one algorithm placed below them. **(a)** Interval estimates show 95% stratified bootstrap CIs for methods with individual runs provided by their respective authors and 95% studentized CIs for CURL, CURL-D2RL, and SUNRISE. Normalized scores are computed by dividing by the maximum score (=1000). **(b)** Score distributions. **(c)** The ith column in the rank distribution plots show the probability that a given method is assigned rank i, averaged across all tasks. The ranks are estimated using 200,000 stratified bootstrap re-samples.

performance. Since performance profiles capture the full picture, they would often illustrate why such inconsistencies exist. For example, optimality gap and IQM can be both read as areas in the profile (Figure 8). The performance profile in Figure 10 (left) illustrates the nuances present when comparing different algorithms. For example, IQN seems to be better than Rainbow for $\tau \geq 2$, but worse for $\tau < 2$. Similarly, the profiles of DreamerV2 and M-IQN for $\tau < 8$ intersect at multiple points. To compare sample efficiency of the agents, we also present their IQM scores as a function of number of frames in Figure 10 (right).

DeepMind Control Suite. Recent continuous control papers benchmark performance on 6 tasks in DM Control [104] at 100k and 500k steps. Typically, such papers claim improvement based on higher mean scores per task regardless of the variability in those scores. However, we find that when accounting for uncertainty in results, most algorithms do not consistently rank above algorithms they claimed to improve upon (Figure 11c and 11b). Furthermore, there are huge overlaps in 95% CIs of mean normalized scores for most algorithms (Figure 11a). These findings suggest that a lot of the reported improvements are spurious, resulting from randomness in the experimental protocol.

Procgen benchmark. Procgen [18] is a popular benchmark, consisting of 16 diverse tasks, for evaluating generalization in RL. Recent papers report mean PPO-normalized scores on this benchmark to emphasize the gains relative to PPO [92] as most methods are built on top of it. However, Figure 12 (left) shows that PPO-normalized scores typically have a heavy-tailed distribution making the mean scores highly dependent on performance on a small fraction of tasks. Instead, we recommend using normalization based on the estimated minimum and maximum scores on ProcGen [18] and reporting aggregate metrics based on such scores (Figure A.32). While publications sometimes make binary claims about whether they improve over prior methods, such improvements are inherently probabilistic. To reveal this discrepancy, we investigate the following question: "What is the

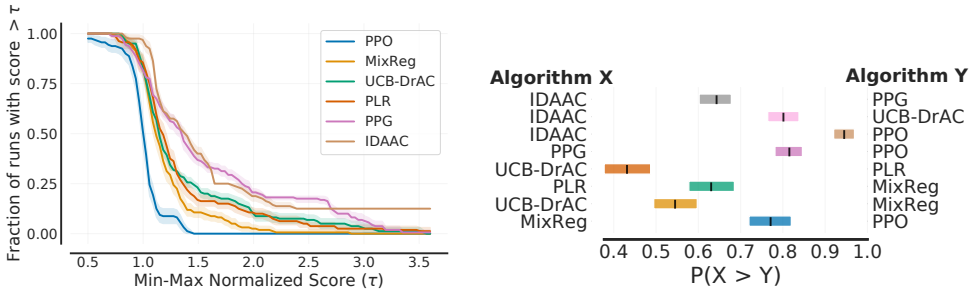


Figure 12: **Procgen evaluation** results based on easy mode comparisons [80] with 16 tasks. **Left**. Score distributions which compare PPO [92], MixReg [111], UCB-DrAC [81], PLR [48], PPG [19] and IDAAC [80]. Shaded regions indicate 95% percentile stratified bootstrap CIs. **Right**. Each row shows the probability of improvement, with 95% bootstrap CIs, that the algorithm X on the left outperforms algorithm Y on the right, given that X was claimed to be better than Y. For all algorithms, results are based on 10 runs per task.

probability that an algorithm which claimed improvement over a prior algorithm performs better than it?" (Figure 12, right). While this probability does not distinguish between two algorithms which uniformly improve on all tasks by 1% and 100%, it does highlight how likely an improvement is. For example, there is only a $40 - 50\%$ chance that UCB-DrAC [81] improves upon PLR [48]. We note that a number of improvements reported in the existing literature are only $50 - 70\%$ likely.

6 Discussion

We saw, both in our case study on the Atari 100k benchmark and with our analysis of other widely-used RL benchmarks, that statistical issues can have a sizeable influence on reported results, in particular when point estimates are used or evaluation protocols are not kept constant within comparisons. Despite earlier calls for more experimental rigor in deep RL [16, 20, 21, 41, 49, 83] (discussed in Appendix A.3), our analysis shows that the field has not yet found sure footing in this regards.

In part, this is because the issue of reproducibility is a complex one; where our work is concerned with our confidence about and interpretation of reported results (what Goodman et al. [34] calls *results reproducibility*), others [79] have highlighted that there might be missing information about the experiments themselves (*methods reproducibility*). We remark that the problem is not solved by fixing random seeds, as has sometimes been proposed [52, 77], since it does not really address the question of whether an algorithm would perform well under similar conditions but with different seeds. Furthermore, fixed seeds might benefit certain algorithms more than others. Nor can the problem be solved by the use of dichotomous statistical significance tests, as discussed in Section 2.

One way to minimize the risks associated with statistical effects is to report results in a more complete fashion, paying close attention to bias and uncertainty within these estimates. To this end, our recommendations are summarized in Table 1. To further support RL researchers in this endeavour, we released an easy-to-use Python library, `rliable` along with a Colab notebook for implementing our recommendations, as well as all the individual runs used in our experiments[10]. Again, we emphasize the importance of published papers providing results for all runs to allow for future statistical analyses.

A barrier to adoption of evaluation protocols proposed in this work, and more generally, rigorous evaluation, is whether there are clear incentives for researchers to do so, as more rigor generally entails more nuanced and tempered claims. Arguably, doing good and reproducible science is one such incentive. We hope that our findings about erroneous conclusions in published papers would encourage researchers to avoid fooling themselves, even if that requires tempered claims. That said, a more pragmatic incentive would be if conferences and reviewers required more rigorous evaluation for publication, *e.g.,* NeurIPS 2021 checklist asks whether error bars are reported. Moving towards reliable evaluation is an ongoing process and we believe that this paper would greatly benefit it.

Given the substantial influence of statistical considerations in experiments involving 40-year old Atari 2600 video games and low-DOF robotic simulations, we argue that it is unlikely that an increase in available computation will resolve the problem for the future generation of RL benchmarks. Instead, just as a well-prepared rock-climber can skirt the edge of the steepest precipices, it seems likely that ongoing progress in reinforcement learning will require greater experimental discipline.

[10]Colab: `bit.ly/statistical_precipice_colab`. Individual runs: `gs://rl-benchmark-data`.

Societal Impacts

This paper calls for statistical sophistication in deep RL research by accounting for statistical uncertainty in reported results. However, statistical sophistication can introduce new forms of statistical abuses and monitoring the literature for such abuses should be an ongoing priority for the research community. Moving towards reliable evaluation and reproducible research is an ongoing process and this paper only partly addresses it by providing tools for more reliable evaluation. That said, while accounting for uncertainty in results is not a panacea, it provides a strong foundation for trustworthy results on which the community can build upon, with increased confidence. In terms of broader societal impact of this work, we do not see any foreseeable strongly negative impacts. However, this paper could positively impact society by constituting a step forwards in rigorous few-run evaluation regime, which reduces computational burden on researchers and is "greener" than evaluating a large number of runs.

Acknowledgments

We thank Xavier Bouthillier, Dumitru Erhan, Marlos C. Machado, David Ha, Fabio Viola, Fernando Diaz, Stephanie Chan, Jacob Buckman, Danijar Hafner and anonymous NeurIPS' reviewers for providing valuable feedback for an earlier draft of this work. We also acknowledge Matteo Hessel, David Silver, Tom Schaul, Csaba Szepesvári, Hado van Hasselt, Rosanne Liu, Simon Kornblith, Aviral Kumar, George Tucker, Kevin Murphy, Ankit Anand, Aravind Srinivas, Matthew Botvinick, Clare Lyle, Kimin Lee, Misha Laskin, Ankesh Anand, Joelle Pineau and Braham Synder for helpful discussions. We also thank all the authors who provided individual runs for their corresponding publications. We are also grateful for general support from Google Research teams in Montréal and elsewhere.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

[2] Valentin Amrhein, Sander Greenland, and Blake McShane. Scientists rise up against statistical significance. *Nature*, 2019.

[3] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

[4] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature News*, 2016.

[5] 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*, 47:253–279, 2013.

[6] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

[7] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 2020.

[8] 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.

[9] Peter J Bickel, Friedrich Götze, and Willem R van Zwet. Resampling fewer than n observations: gains, losses, and remedies for losses. In *Selected works of Willem van Zwet*, pages 267–297. Springer, 2012.

[10] Mauro Birattari and Marco Dorigo. How to assess and report the performance of a stochastic algorithm on a benchmark problem: mean or best result on a number of runs? *Optimization letters*, 2007.

[11] Xavier Bouthillier, César Laurent, and Pascal Vincent. Unreproducible research is reproducible. In *International Conference on Machine Learning*, pages 725–734, 2019.

[12] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3, 2021.

[13] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[14] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

[15] Johan Samir Obando Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, 2021.

[16] Stephanie CY Chan, Samuel Fishman, Anoop Korattikara, John Canny, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *International Conference on Learning Representations*, 2020.

[17] Kaleigh Clary, Emma Tosch, John Foley, and David Jensen. Let's play again: Variability of deep reinforcement learning agents in atari environments. *arXiv preprint arXiv:1904.06312*, 2019.

[18] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[19] Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. *arXiv preprint arXiv:2009.04416*, 2020.

[20] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

[21] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms. *arXiv preprint arXiv:1904.06979*, 2019.

[22] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

[23] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Mostafa Dehghani, Yi Tay, Alexey A Gritsenko, Zhe Zhao, Neil Houlsby, Fernando Diaz, Donald Metzler, and Oriol Vinyals. The benchmark lottery. *arXiv preprint arXiv:2107.07002*, 2021.

[25] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[26] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[27] Rotem Dror, Segev Shlomov, and Roi Reichart. Deep dominance-how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[28] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[29] Bradley Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7:1–26, 1979.

[30] Bradley Efron. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 1987.

[31] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[32] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*. PMLR, 2018.

[33] Gerd Gigerenzer. Statistical rituals: The replication delusion and how we got there. *Advances in Methods and Practices in Psychological Science*, 1(2):198–218, 2018.

[34] Steven N Goodman, Daniele Fanelli, and John PA Ioannidis. What does research reproducibility mean? *Science translational medicine*, 8(341):341ps12–341ps12, 2016.

[35] Sander Greenland, Stephen J Senn, Kenneth J Rothman, John B Carlin, Charles Poole, Steven N Goodman, and Douglas G Altman. Statistical tests, p values, confidence intervals, and power: a guide to misinterpretations. *European journal of epidemiology*, 2016.

[36] 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*, pages 1861–1870. PMLR, 2018.

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

[38] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[39] Steven Hansen, Will Dabney, Andre Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2020.

[40] Nathaniel E Helwig. Bootstrap Confidence Intervals, 01 2021. URL http://users.stat.umn.edu/~helwig/notes/npboot-notes.html#bootstrap-confidence-intervals.

[41] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[42] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[43] Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. Muesli: Combining improvements in policy optimization. *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[44] John PA Ioannidis. Why most published research findings are false. *PLoS medicine*, 2(8):e124, 2005.

[45] Alex Irpan. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[46] 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.

[47] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[48] Minqi Jiang, Ed Grefenstette, and Tim Rocktäschel. Prioritized level replay. *International Conference on Machine Learning*, 2021.

[49] Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip Thomas. Evaluating the performance of reinforcement learning algorithms. In *International Conference on Machine Learning*, pages 4962–4973. PMLR, 2020.

[50] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[51] Kacper Kielak. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? *arXiv preprint arXiv:2003.10181*, 2020.

[52] Sergey Kolesnikov and Oleksii Hrinchuk. Catalyst. rl: a distributed framework for reproducible rl research. *arXiv preprint arXiv:1903.00027*, 2019.

[53] Ilya Kostrikov*, Denis Yarats*, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[54] Piotr Kozakowski, Lukasz Kaiser, Henryk Michalewski, Afroz Mohiuddin, and Katarzyna Kańska. Q-value weighted regression: Reinforcement learning with limited data. *arXiv preprint arXiv:2102.06782*, 2021.

[55] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *NeurIPS*, 32:10724–10734, 2019.

[56] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

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

[58] Alex Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33, 2020.

[59] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *International Conference on Machine Learning*, 2021.

[60] Kuang-Huei Lee, Ian Fischer, Anthony Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in rl. *Advances in Neural Information Processing Systems*, 2020.

[61] Haim Levy. Stochastic dominance and expected utility: Survey and analysis. *Management science*, 38(4): 555–593, 1992.

[62] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[63] Jimmy Lin, Daniel Campos, Nick Craswell, Bhaskar Mitra, and Emine Yilmaz. Significant improvements over the state of the art? a case study of the ms marco document ranking leaderboard. *arXiv preprint arXiv:2102.12887*, 2021.

[64] Guoqing Liu, Chuheng Zhang, Li Zhao, Tao Qin, Jinhua Zhu, Li Jian, Nenghai Yu, and Tie-Yan Liu. Return-based contrastive representation learning for reinforcement learning. In *International Conference on Learning Representations*, 2021.

[65] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021.

[66] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[67] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[68] Nicolai A Lynnerup, Laura Nolling, Rasmus Hasle, and John Hallam. A survey on reproducibility by evaluating deep reinforcement learning algorithms on real-world robots. In *Conference on Robot Learning*, 2020.

[69] 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*, 2018.

[70] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[71] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[72] Muhammad Rizki Maulana and Wee Sun Lee. Ensemble and auxiliary tasks for data-efficient deep reinforcement learning. *arXiv preprint arXiv:2107.01904*, 2021.

[73] Blakeley B McShane, David Gal, Andrew Gelman, Christian Robert, and Jennifer L Tackett. Abandon statistical significance. *The American Statistician*, 2019.

[74] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

[75] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[76] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[77] Prabhat Nagarajan, Garrett Warnell, and Peter Stone. Deterministic implementations for reproducibility in deep reinforcement learning. *arXiv preprint arXiv:1809.05676*, 2018.

[78] Harold Pashler and Eric-Jan Wagenmakers. Editors' introduction to the special section on replicability in psychological science: A crisis of confidence? *Perspectives on psychological science*, 7(6):528–530, 2012.

[79] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:2003.12206*, 2020.

[80] Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. *International Conference on Machine Learning*, 2021.

[81] Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

[82] David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

[83] Ben Recht. Benchmarking Machine Learning with Performance Profiles, 03 2018. URL http://www.argmin.net/2018/03/26/performance-profiles/.

[84] Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, 2017.

[85] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *International conference on machine learning*, 2017.

[86] Jan Robine, Tobias Uelwer, and Stefan Harmeling. Smaller world models for reinforcement learning. *arXiv preprint arXiv:2010.05767*, 2020.

[87] David Romer. In praise of confidence intervals. In *AEA Papers and Proceedings*, volume 110, pages 55–60, 2020.

[88] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[89] Rohan Saphal, Balaraman Ravindran, Dheevatsa Mudigere, Sasikant Avancha, and Bharat Kaul. Seerl: Sample efficient ensemble reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1100–1108, 2021.

[90] Tom Schaul, Georg Ostrovski, Iurii Kemaev, and Diana Borsa. Return-based scaling: Yet another normalisation trick for deep rl. *arXiv preprint arXiv:2105.05347*, 2021.

[91] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

[92] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[93] 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*, 2021.

[94] Thibault Sellam, Steve Yadlowsky, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Turc, Jacob Eisenstein, Dipanjan Das, et al. The multiberts: Bert reproductions for robustness analysis. *arXiv preprint arXiv:2106.16163*, 2021.

[95] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[96] 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.

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

[98] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136v2*, 2020.

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

[100] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 1996.

[101] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2nd edition, 2018.

[102] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

[103] István Szita and András Lörincz. Learning tetris using the noisy cross-entropy method. *Neural computation*, 2006.

[104] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[105] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. Is deep reinforcement learning really superhuman on atari? leveling the playing field. *arXiv preprint arXiv:1908.04683*, 2019.

[106] John W Tukey. A survey of sampling from contaminated distributions. *Contributions to probability and statistics*, pages 448–485, 1960.

[107] Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *NeurIPS*, 2019.

[108] Gaël Varoquaux and Veronika Cheplygina. How i failed machine learning in medical imaging–shortcomings and recommendations. *arXiv preprint arXiv:2103.10292*, 2021.

[109] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[110] 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*, 2019.

[111] Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *arXiv preprint arXiv:2010.10814*, 2020.

[112] Ronald L. Wasserstein, Allen L. Schirm, and Nicole A. Lazar. Moving to a world beyond "p < 0.05". *The American Statistician*, 2019.

[113] Bernard L Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[114] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[115] Jinhua Zhu, Yingce Xia, Lijun Wu, Jiajun Deng, Wengang Zhou, Tao Qin, and Houqiang Li. Masked contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2010.07470*, 2020.

[116] Donglin Zhuang, Xingyao Zhang, Shuaiwen Leon Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. *arXiv preprint arXiv:2106.11872*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix A.1.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.2
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Apache License, Version 2.0
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Open-source notebook and data

Colab notebook for producing and analyzing performance profiles, robust aggregate metrics, and interval estimates based on stratified bootstrap CIs, as well as replicating the results in the paper can be found at `bit.ly/statistical_precipice_colab`.

Individual runs for Atari 100k. We released the 100 runs per game for each of the 6 algorithms in the case study in a public cloud bucket at `gs://rl-benchmark-data/atari_100k`.

Individual runs for ALE, Procgen and DM Control. For ALE, we used the individual runs from Dopamine [14] baselines except for DreamerV2 [38], REM [1] and M-IQN [109], for which the individual run scores were obtained from the corresponding authors. We release all the individual run scores as well as final scores for ALE at `gs://rl-benchmark-data/ALE`. The Procgen results were obtained from the authors of IDAAC [80] and MixReg [48] and are released at `gs://rl-benchmark-data/procgen`. For DM Control[11], all the runs were obtained from the corresponding authors and are released at `gs://rl-benchmark-data/dm_control`.

See `agarwl.github.io/rliable` for a website for the paper.

A.2 Atari 100k: Additional Details and Results

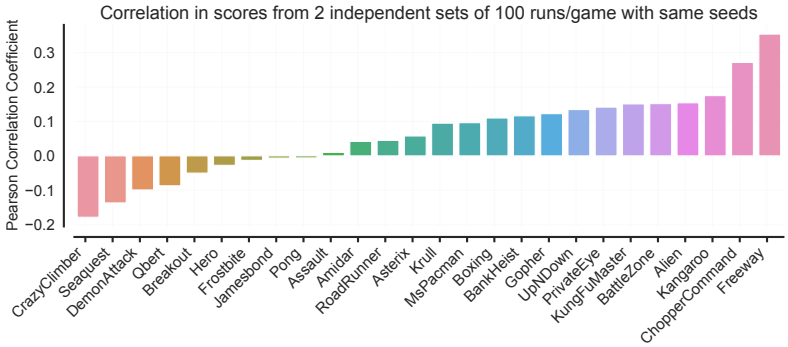


Figure A.13: **Runs can be different from using fixed random seeds**. We find that correlation between two sets of 100 runs of DER on Atari 100k using the same set of random seeds, that is, using a fixed random seed per run for Python, NumPy and JAX, is quite small. Small values of correlation coefficient highlight that fixing seeds does not ensure deterministic results due to non-determinism in GPUs. Similarly, setting random seed in PyTorch ensures reproducibility only on the same hardware.

Code. Due to unavailability of open-source code for DER, and OTR for Atari 100k, we re-implemented these algorithms using Dopamine [14], a reproducible deep RL framework. For CURL and SPR, we used the open-source code released by the authors while for DrQ, we used the source-code obtained from the authors. Our code for Atari 100k experiments is open-sourced as part of the Dopamine library under the `labs/atari_100k` folder. We also released a JAX [13] implementation of the full Rainbow [42] in Dopamine.

Hyperparameters. All algorithms build upon the Rainbow [42] architecture and we use the exact same hyperparameters specified in the corresponding publication unless specified otherwise. Akin to DrQ and SPR, we used n-step returns with $n = 10$ instead of $n = 20$ for DER. DrQ codebase uses non-standard evaluation hyperparameters, such as a 5% probability of selecting random actions during evaluation (ε-`eval`$= 0.05$). DrQ(ε) differs DrQ in terms of using standard ε-greedy parameters [14, Table 1] including training ε decayed to 0.01 rather than 0.1 and evaluation ε set to 0.001 instead of 0.05. Refer to the gin configurations in `labs/atari_100k/configs` for more details.

[11] Dreamer [37] results on DM control, obtained from the corresponding author, are based on hyperparameters tuned for sample-efficiency. Compared to the original paper [37], the actor-critic learning rates were increased to $3e-4$, the amount of free bits to 1.5, the training frequency, and the amount of pre-training to 1k steps on 10k randomly collected frames. The imagination horizon was decreased to 10.

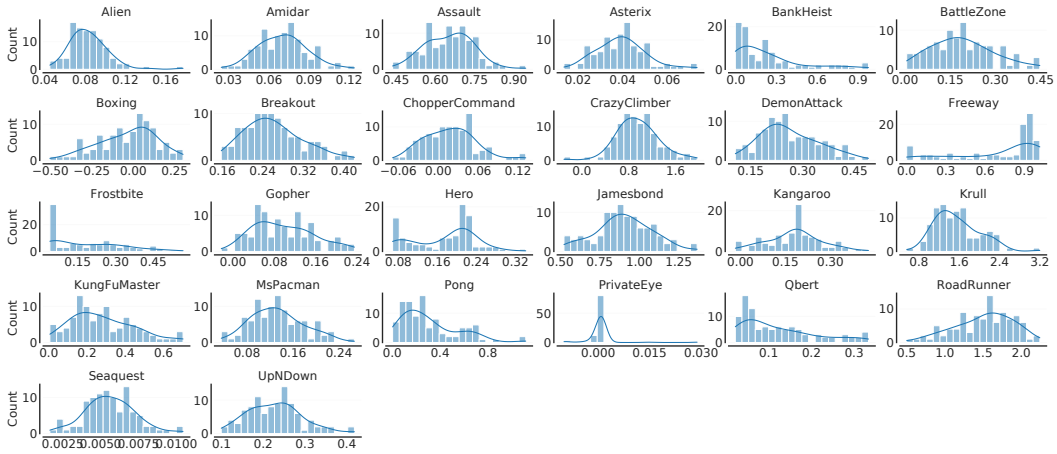


Figure A.14: **Per-game score distributions**. Histogram plot with kernel density estimate of human-normalized scores of DER on 26 games in the Atari 100k benchmark. Each histogram plot is based on 100 runs per game. For most games, the distributions are either skewed (*e.g.,* KUNGFUMASTER), heavy-tailed (*e.g.,* BANKHEIST, FROSTBITE) or multimodal (*e.g.,* HERO)

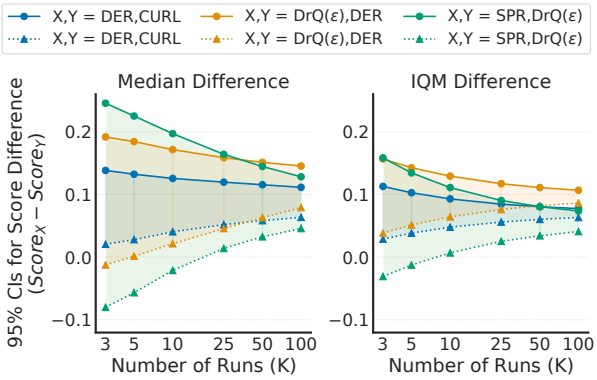


Figure A.15: **Detecting score differences**. **Left**. 95% CIs for differences in median scores. **Right**. 95% CIs for differences in IQM scores. Median requires many more runs than IQM for small uncertainty.

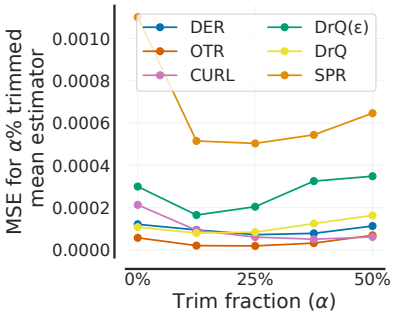


Figure A.16: **Statistical Efficiency of IQM**. Efficiency of an estimator is typically measured in terms of its mean squared error (MSE). We estimate MSE for trimmed estimators with 10 runs by subsampling 20,000 sets of 10 runs with replacement from 100 runs.

Compute. For the case study on Atari 100k, we used Tesla P100 GPUs for all the runs. Each run spanned about 3-5 hours depending on the algorithm, and we ran a total of 100 runs / game $\times$ 26 games/algorithm $\times$ 6 algorithms = 15,600 runs. Additionally, we ran an additional 100 runs per game for DER to compute a good approximation of point estimates for aggregate scores, which increases the total number of runs by 2600. Overall, we trained and evaluated 18,200 runs, which roughly amounts to 2400 days – 3600 days of GPU training.

Comparing performance of two algorithms. When confidence intervals (CIs) overlap for two random variables X and Y overlap, we estimate the 95% CIs for $X - Y$ to account for uncertainty in their difference (Figure A.15). For example, when using 5 runs, the median score improvement from DrQ(ε) over DER is estimated to lie within $(0.01, 0.21)$ while that of SPR over DrQ lies within $(-0.09, 0.18)$. Furthermore, while improvement from SPR over DER with 5 to 15 runs is not statistically significant, claiming "no improvement" would be misleading as evaluating more runs indeed shows that the improvement is significant.

Analyzing efficiency and bias of IQM. Theoretically, trimmed means, are known to have higher statistical efficiency for mixed distributions and heavy-tailed distributions (Cauchy distribution), at the cost of lower efficiency for some other less heavily-tailed distributions (normal distribution) than mean, as shown by the seminal work of Tukey [106]. Empirically, on Atari 100k, IQM provides good statistically efficiency among trimmed estimators across different algorithms (Figure A.16) as well as has considerably small bias than median (*c.f.* Figure A.17 *vs.* Figure 3).

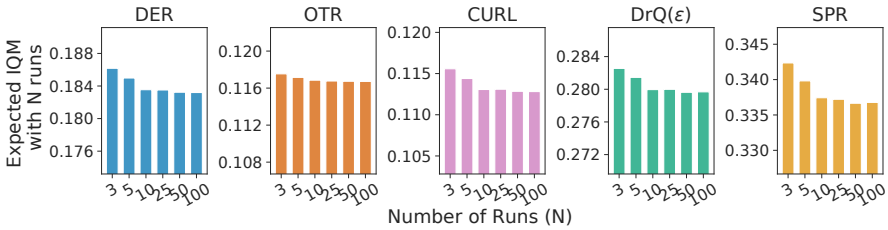


Figure A.17: **Negligible bias in IQM scores**. Expected IQM scores with varying number of runs. The expected score for N runs is computed by repeatedly subsampling N runs with replacement out of 100 runs for 100,000 times. Compared to expected median score differences (Figure 3), the difference in expected IQM scores with 3 runs and 100 runs is typically an order of magnitude smaller. For example, the expected median differences for SPR is 0.05 points while expected IQM differences are only 0.006 points.

A.3 Related work on rigorous evaluation in deep RL

While prior work [41, 46, 68] highlights various reproducibility issues in policy-gradient methods, this paper focuses specifically on the reliability of evaluation procedures on RL benchmarks and provides an extensive analysis on common deep RL algorithms on widely-used benchmarks.

For more rigorous performance comparisons on a single RL task, Colas et al. [21], Henderson et al. [41] provide guidelines for statistical significance testing while Colas et al. [20] focuses on determining the minimum number of runs needed for such comparisons to be statistically significant. Instead, this paper focuses on reliable comparisons on a suite of tasks and mainly recommends reporting stratified bootstrap CIs due to the dichotomous nature and wide misinterpretation of statistical significance tests (see Remark in Section 2). Colas et al. [20, 21], Henderson et al. [41] also discuss bootstrap CIs but for reporting single task mean scores – however, 3-5 runs is a small sample size for bootstrapping: on Atari 100k, for achieving true coverage close to 95%, such CIs require at least 20-30 runs per task (Figure A.18) as opposed to 5-10 runs for stratified bootstrap CIs for aggregate metrics like median, mean and IQM (Figure A.19).

Chan et al. [16] propose metrics to measure the reliability of RL algorithms in terms of their stability during training and their variability and risk in returns across multiple episodes. While this paper focuses on reliability of evaluation itself, performance profiles showing the tail distribution of episodic returns, applicable for even a single task with multiple runs, can be useful for measuring reliability of an algorithm's performance.

Jordan et al. [49] propose a game-theoretic evaluation procedure for "complete" algorithms that do not require any hyperparameter tuning and recommend evaluating between 1,000 to 10,000 runs per task to detect statistically significant results. Instead, this work focuses on reliably evaluating performance obtained after the hyperparameter tuning phase, even with just a handful of runs. That said, run-score distributions based on runs with different hyperparameter configurations might reveal sensitivity to hyperparameter tuning.

An alternative to *score distributions*, proposed by Recht [83], is to replace scores in a performance profile [26] by the probability that average task scores of a given method outperforms the best method (among a given set of methods), computed using the Welsh's t-test [113]. However, this profile is (1) also a biased estimate, (2) less robust to outlier runs, (3) is insensitive to the size of performance differences, *i.e.,* two methods that are uniformly 1% and 100% worse than the best method are assigned the same probability, (4) is only sensible when task score distributions are Gaussian, as required by Welsh's t-test, and finally, (5) the ranking of methods depends on the specific set of methods being compared in such profiles.

A.4 Non-standard Evaluation Protocols Involving Maximum

Even when adequate number of runs are used, the use of non-standard evaluation protocols can result in misleading comparisons. Such protocols commonly involve the insertion of a maximum operation inside evaluation, *across* or *within* runs, leading to a positive bias in reported scores compared to the standard approach without the maximum.

One seemingly reasonable but faulty argument [10] for maximum across runs is that in a real-world application, one might wish to run an stochastic algorithm A for N runs and then select the best result. However, in this case, we are not discussing A but another algorithm A^N, which evaluates N

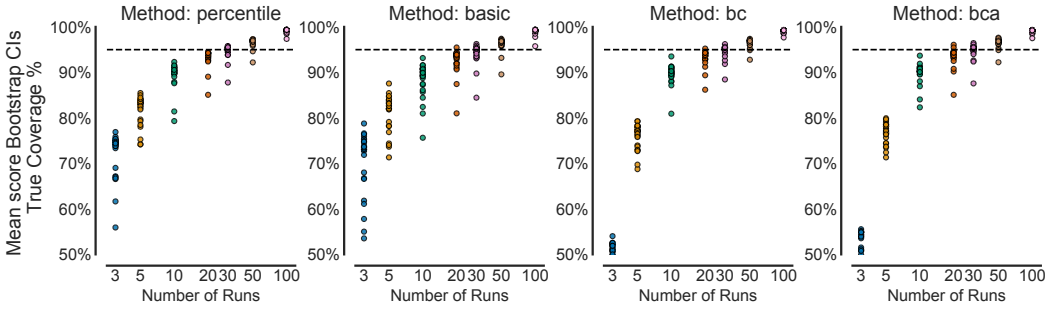


Figure A.18: **Validating 95% bootstrap CIs for per-game mean scores** for a varying number of runs for DER, shown as a scatter plot where each point corresponds to one of the 26 games in Atari 100k. For a given game, the true coverage % is computed by sampling 10,000 sets of K runs without replacement from 200 runs and checking the fraction of 95% CIs that contains the true mean score for that game based on 200 runs. For many games, the true coverage for per-game CIs is below the nominal coverage of 95% even with 30 runs per game.

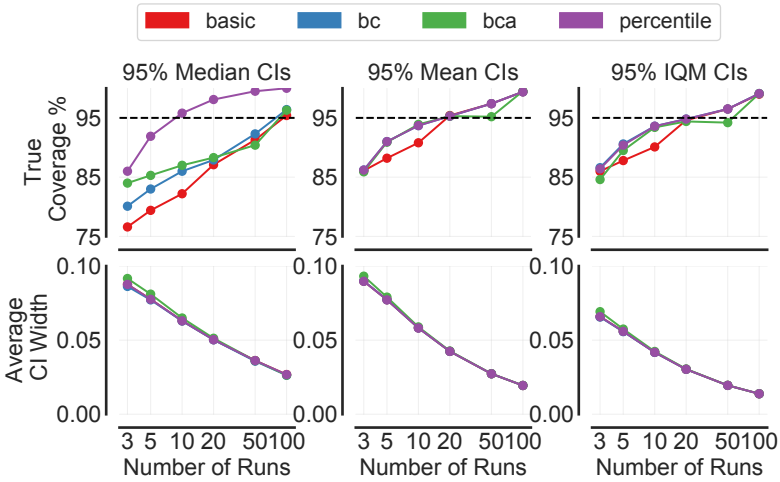


Figure A.19: **Validating 95% stratified bootstrap CIs for aggregate scores** for a varying number of runs. We show CIs for median, mean and IQM scores, aggregated using scores across 26 games, for DER. The true coverage % is computed by sampling 10,000 sets of K runs without replacement from 200 runs and checking the fraction of 95% CIs that contains the true estimate approximation based on 200 runs. Please note that coverage above 95%, even with 50+ runs, is likely due to approximation error in the true estimate using finite runs.

random runs of A. If we are interested in A^N, taking maximum over N runs only considers a single run of A^N. Since A^N is itself stochastic, proper experimental methodology requires multiple runs of A^N. Furthermore, because learning curves are not in general monotonic, results produced under the maximum-during-training protocol are in general incomparable with end-performance reported results. In addition, such protocols introduces an additional source of positive statistical bias, since the maximum of a set of random variables is a biased estimate of their true maximum.

On Atari 100k, CURL [56] and SUNRISE [59] used non-standard evaluation protocols. CURL reported the maximum performance over 10 different evaluations during training. As a result, natural variability in both evaluation itself and in the agent's performance during training contribute to overestimation. Applying the same procedure to CURL's baseline DER leads to scores far above those reported for CURL (Figure 5, "DER (CURL's protocol)"). In the case of SUNRISE, the maximum was taken over eight hyperparameter configurations separately for each game, with three runs each. We simulate this procedure for DER (also SUNRISE's baseline), using a dummy hyperparameter. We find that a lot of SUNRISE's improvement over DER can be explained by this evaluation scheme (Figure 5, "DER (SUNRISE's protocol)").

A.5 Bootstrap Confidence Intervals

Bootstrap CIs for a real parameter θ are based on re-sampling with replacement from a fixed set of K samples to create a bootstrap sample of size K and compute the bootstrap parameter θ^* and repeating

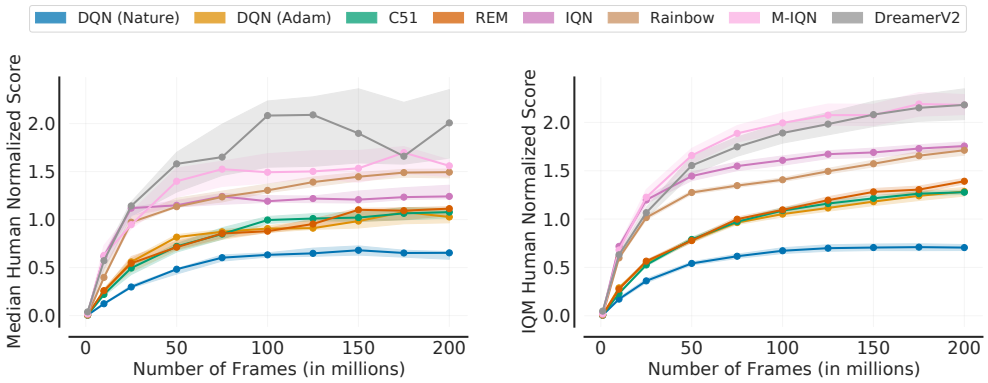


Figure A.20: **Comparing Median vs IQM on Atari 200M**. Sample-efficiency of agents as a function of number of frames measured via median **(left)** and IQM **(right)** human-normalized scores. Shaded regions show pointwise 95% percentile stratified bootstrap CIs. IQM results in significantly smaller CIs than median scores.

this process a numerous to create the bootstrap distribution over θ^*. In this paper, we evaluate the following non-parametric methods for constructing CIs for θ using this bootstrap distribution:

1. **Basic** bootstrap, also known as the reverse percentile interval, uses the empirical quantiles from the bootstrap distribution of the parameter $\widehat{\delta} = \hat{\theta} - \theta$ for defining the $\alpha \times 100\%$ CI: $(2\widehat{\theta} - \theta^*_{(\alpha/2)}, 2\widehat{\theta} - \theta^*_{(1-\alpha/2)})$, where $\theta^*_{(1-\alpha/2)}$ denotes the $1-\alpha/2$ percentile of the bootstrapped parameters θ^* and $\widehat{\theta}$ is the empirical estimate of the parameter based on finite samples.
2. **Percentile** bootstrap. The percentile bootstrap proceeds in a similar way to the basic bootstrap, using percentiles of the bootstrap distribution, but with a different formula: $(\theta^*_{(1-\alpha/2)}, \theta^*_{(\alpha/2)})$ for defining the $\alpha \times 100\%$ CI.
3. **Bias-corrected** (bc) bootstrap – adjusts for bias in the bootstrap distribution.
4. **Bias-corrected and accelerated** (bca) bootstrap, by Efron [29], adjusts for both bias and skewness in the bootstrap distribution. This approach is typically considered to be more accurate and has better asymptotic properties. However, we find that it is not as effective as percentile methods in the few-run deep RL regime.

More technical details about bootstrap CIs can be found in [40]. We find that bootstrap CIs for mean scores per game (computed using N random samples) require many more runs than aggregate scores (computed using MN random samples) for achieving true coverage close to the nominal coverage of 95% (*c.f.* Figure A.18 *vs.* Figure A.19).

Number of bootstrap re-samples. Unless specified otherwise, for computing uncertainty estimates using stratified bootstrap, we use 50,000 samples for aggregate metrics and 2000 samples for pointwise confidence bands and average probability of improvement. Using larger number of samples then the above specified values might result in more accurate uncertainty estimates but would be slower to compute.

Stratified bootstrap over tasks and runs[12]. With access to only 1-2 runs per task, stratified bootstrapping can be done over tasks (Figure A.22), to answer the question: "If I repeat the experiment with a different set of tasks, what performance an algorithm is I expected to get?" It shows the sensitivity of the aggregate score to a given task and can also be viewed as an estimate of performance if we had used a larger unknown population of tasks [*e.g.,* 90, 94]. Compared to the interval estimates in Figure 9, bootstraping over tasks results in much larger uncertainty due to high variations in performance across different tasks (*e.g.,* easy vs hard exploration tasks).

A.6 Visualizing score distributions

Choice of Normalization. We used existing normalization schemes which are prevalent on benchmarks including human normalized scores for Atari 100k and ALE, PPO normalized scores and Min-Max normalized scores for Procgen, and Min-Max Normalized scores (minimum scores set

[12]Thanks to David Silver and Tom Schaul for suggesting stratified bootstrapping over tasks.

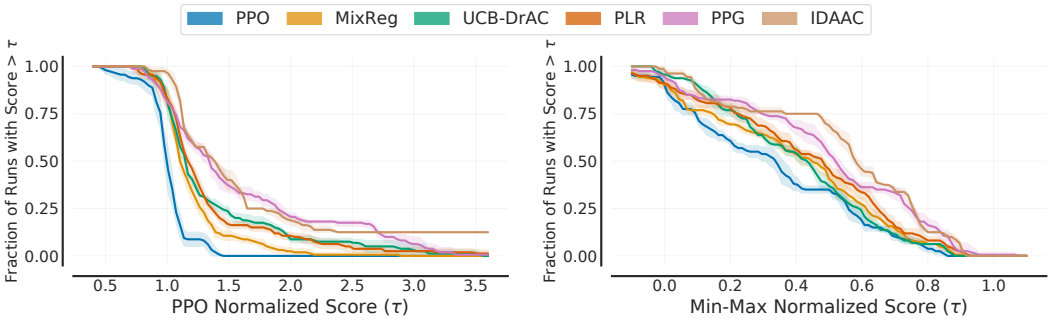


Figure A.21: **Score Distributions on the Procgen benchmark** [18] based on results in the easy mode setting [80]. Shaded regions indicate 95% CIs estimated using the percentile bootstrap with stratified sampling. We compare PPO [92], MixReg [111], UCB-DrAC [81], PLR [48], PPG [19] and IDAAC [80]. We recommend using min-max normalized scores as opposed to PPO normalized scores.

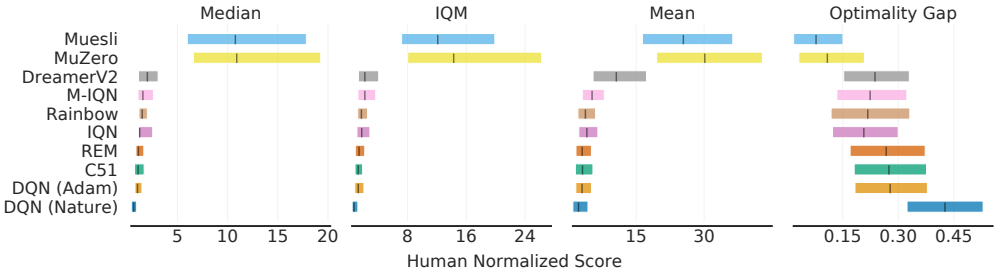


Figure A.22: **Stratified Bootstrap across tasks and runs**. Aggregate metrics on Atari 200M with 95% CIs based on 55 games with sticky actions [69]. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling across tasks and runs. MuZero [91] results use 1 run/game while Muesli [43] uses 2 runs/game, as provided by the corresponding authors. All other results are based on 5 runs per game except for M-IQN and DreamerV2 which report results with 3 and 11 runs. These estimates are much wider than that obtained via bootstrap over runs (Figure 9).

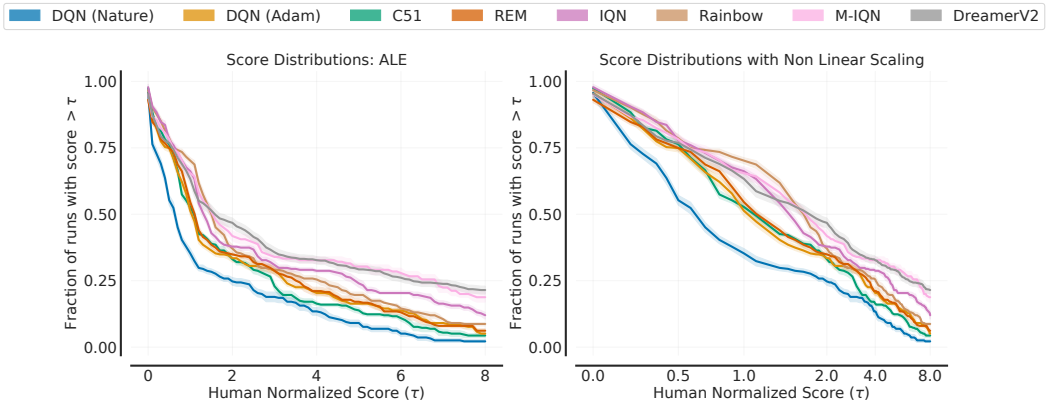


Figure A.23: **Score distributions with linear and with non-linear scaling** on Atari 200M. In the plots above, the x-axis is scaled such that spacing between any two τ values, τ_1 and τ_2, is proportional to the fraction of runs averaged across algorithms between those two τ values. This scaling shows the regions of the score distribution where most of the runs lie as opposed to comparing tail ends of the distribution. However, this scaling implies sub-linear utility of achieving higher scores, which may not be accurate as the utility depends on the difficulty of obtaining higher scores – it is much higher to obtain higher scores on hard exploration games. Furthermore, we cannot visually inspect mean/IQM scores based on the area under the curve due to the non-linear scaling.

to zero) scores for DM Control. We do not use record normalized scores for ALE (Figure A.27) in the main text as ALE results are reported by evaluating agents for 30 minutes of game-play as opposed to record scores which were obtained using game play spanning numerous hours (*e.g.*, Toromanoff et al. [105] recommend evaluating agents for 100 hours). Furthermore, we recommend

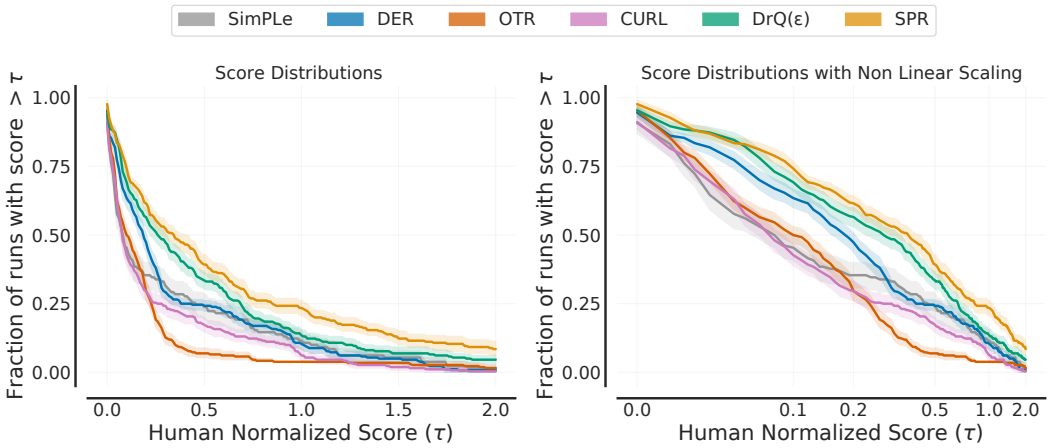


Figure A.24: **Score distributions with linear and with non-linear scaling** on Atari 100k. In the plots above, the x-axis is scaled such that spacing between any two τ values, τ_1 and τ_2, is proportional to the fraction of runs averaged across algorithms between those two τ values.

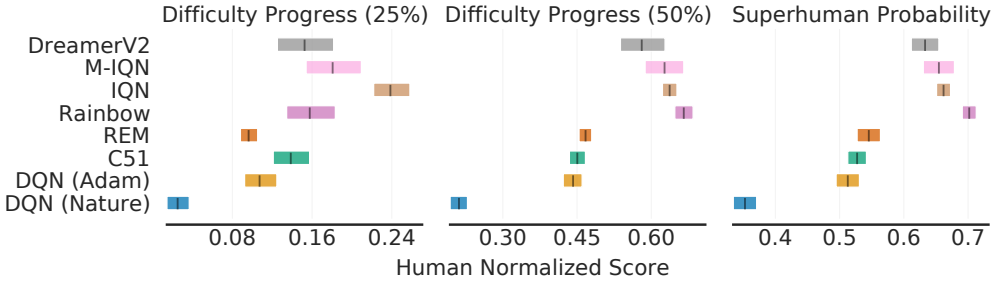


Figure A.25: **Alternative aggregate metrics on ALE** based on 55 games with 95% CIs. Higher metrics are better. The CIs are estimated using the percentile bootstrap with stratified sampling.

using Min-Max Normalized scores for Procgen instead of PPO Normalized scores (Figure A.21) to allow for comparisons to methods which do not build upon PPO [92].

Scaling x-axis in score distributions. Figure A.23 (right) and Figure A.24 (right) shows an alternative for visualizing score distributions where we simply scale the x-axis depending on the fraction of runs in a given region. This scaling more clearly shows the differences in algorithms by focusing on the regions where most of the runs lie[13].

A.7 Aggregate metrics: Additional visualizations and details

Alternative aggregate metrics. Different aggregate metrics emphasize different characteristics and no single metric would be sufficient for evaluating progress. While score distributions provide a full picture of evaluation results, we provide suggestions for alternative aggregate metrics to highlight other important aspects of performance across different tasks and runs.

- **Difficulty Progress**: One might be more interested in evaluating progress on the hardest tasks on a benchmark [3]. In addition to optimality gap which emphasizes all tasks below a certain performance level, a possible aggregate measure to consider is the mean scores of the bottom 25% of the runs (Figure A.25, left), which we call *Difficulty Progress* (DP-25).
- **Superhuman Probability**: We also recommend reporting *probability of being superhuman*, $P(X > 1)$, given by the number of runs above average human performance (Figure A.25, right) instead of number of games above average human performance [42, 93], a commonly used metric on ALE.

[13]Thanks to Mateo Hessel for suggesting this visualization scheme and the difficulty progress metric.

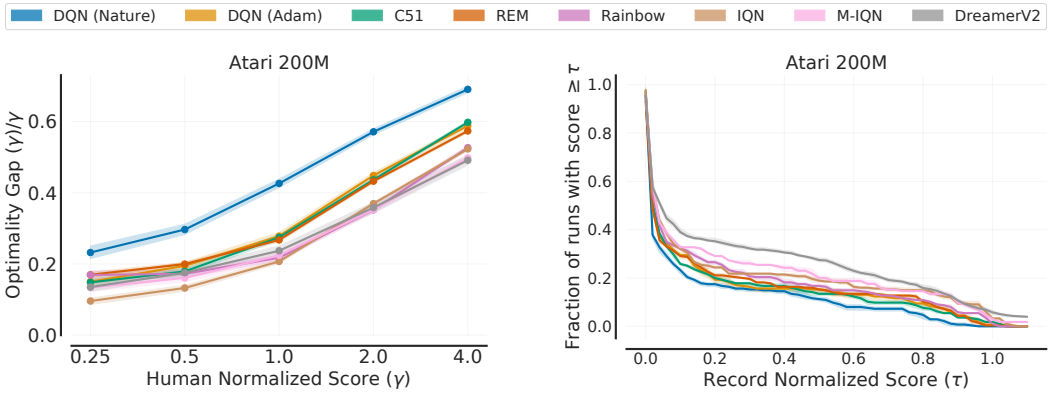


Figure A.26: Optimality gap (γ) divided by γ as a function of γ. Lower curves are better.

Figure A.27: Score distributions using record normalized scores.

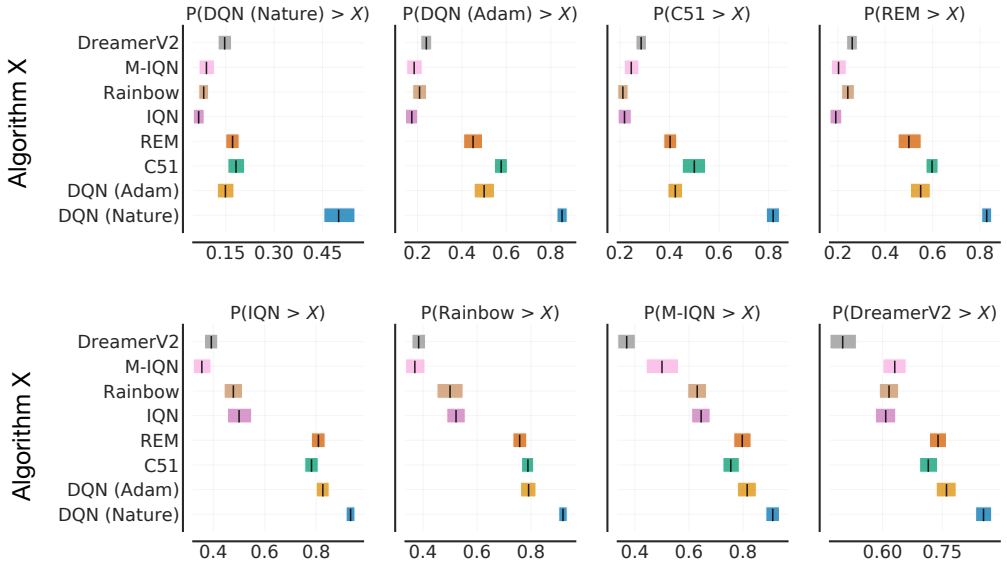


Figure A.28: **Average Probability of Improvement on ALE**. Each subplot shows the probability of improvement of a given algorithm compared to all other algorithms. The interval estimates are based on stratified bootstrap with independent sampling with 2000 bootstrap re-samples

Choice of γ for optimality gap. When using min-max normalized scores or human-normalized scores, setting a score threshold of $\gamma = 1$ is sensible as it considers performance on games below maximum performance or human performance respectively. If there is no preference for a specific threshold, an alternative is to consider a curve of optimality gap as the threshold is varied, as shown in Figure A.26, which shows how far from optimality an algorithm is given any threshold – a small value of optimality gaps for all achievable score thresholds is desirable.

Probability of improvement. To compute the probability of improvement for a task m for algorithms X and Y with N and K runs respectively, we use the Mann-Whitney U-statistic [71], that is,

$$P(X_m > Y_m) = \frac{1}{NK} \sum_{i=1}^{N} \sum_{j=1}^{K} S(x_{m,i}, y_{m,j}) \quad \text{where} \quad S(x, y) = \begin{cases} 1, & \text{if } y < x, \\ \frac{1}{2}, & \text{if } y = x, \\ 0, & \text{if } y > x. \end{cases} \tag{A.2}$$

Please note that if the probability of improvement is higher than 0.5 and the CIs do not contain 0.5, then the results are statistically significant. Furthermore, if the upper CI is higher than a threshold of 0.75, then the results are said to be statistically meaningful as per the Neyman-Pearson statistical testing criterion by Bouthillier et al. [12]. We show the average probability of improvement metrics

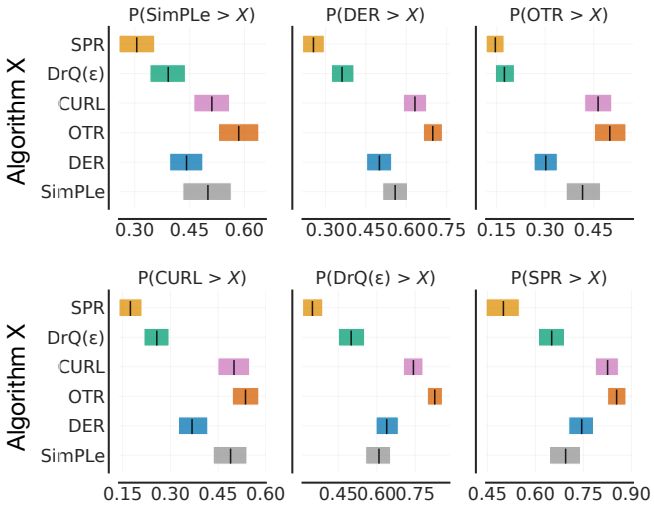


Figure A.29: **Average Probability of Improvement on Atari 100k**. Each subplot shows the probability of improvement of a given algorithm compared to all other algorithms. The interval estimates are based on stratified bootstrap with independent sampling with 2000 bootstrap re-samples.

for Atari 100k and ALE in Figure A.29 and Figure A.28. These estimates show how likely an algorithm improves upon another algorithm.

Aggregate metrics on Atari 100k, Procgen and DM Control as well as ranking on individual tasks on DM Control are visualized in Figures A.30–A.33.

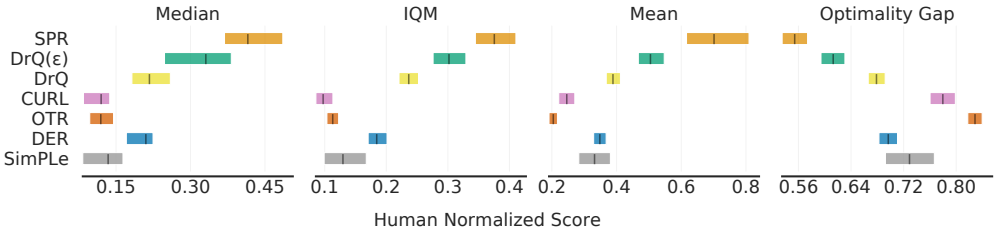
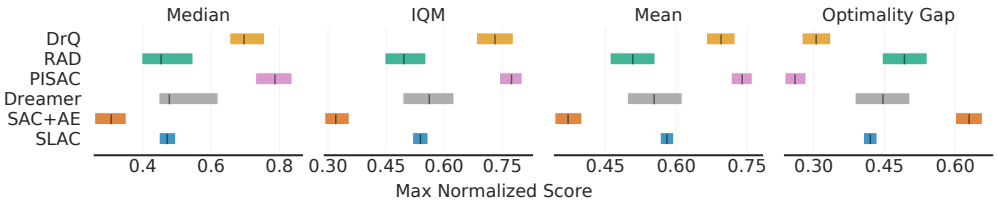
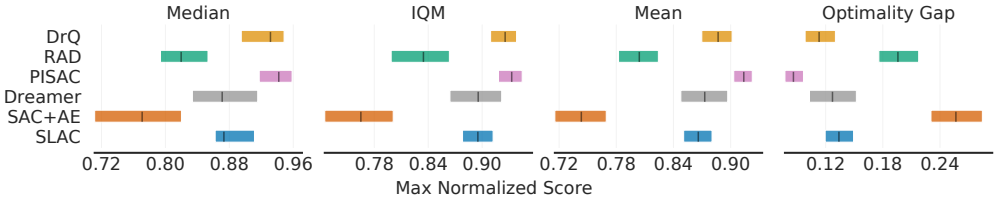


Figure A.30: **Aggregate metrics on Atari 100k** based on 26 games with 95% CIs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling. All results are based on **10 runs per game** except SimPLe, for which we use the 5 runs from their reported results. IQM results in smaller CIs than median scores while optimality gap results in smaller CIs than mean scores. Mean scores are higher than IQM and median scores, indicating that they might be dominated by performance on outlier tasks.

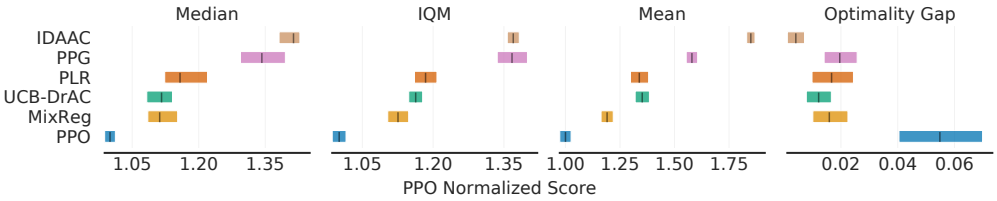


(a) 100k step benchmark.

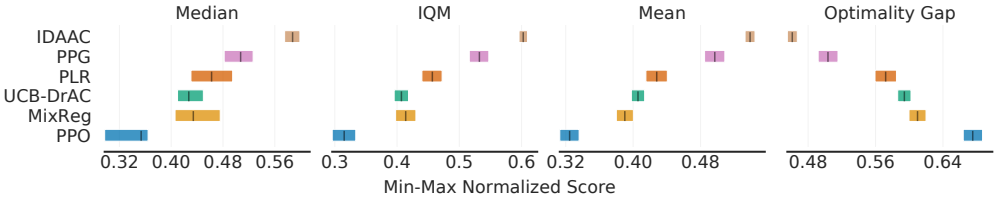


(b) 500k step benchmark.

Figure A.31: **Aggregate metrics on DM Control** based on 6 tasks with 95% CIs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling with 50,000 bootstrap resamples. All results are based on 10 runs per game. All scores are bounded above by 1, so 1 - optimality gap corresponds to mean scores.

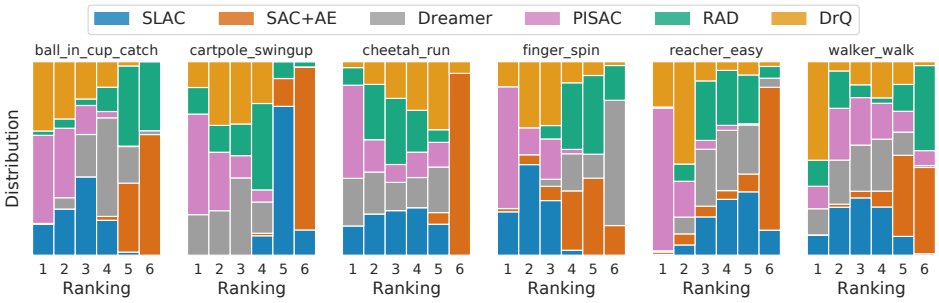


(a) Aggregate metrics based on **PPO normalized scores**. Mean is dominated by outliers while median has large CIs compared to IQM. All algorithms perform better than PPO, resulting in a small optimality gap.

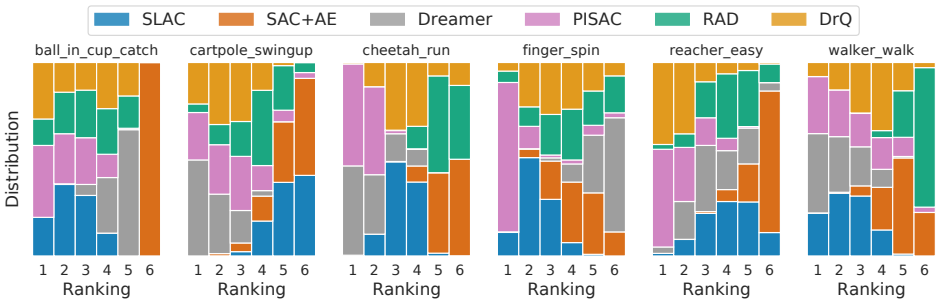


(b) Aggregate metrics based on **min-max normalized scores**. IQM results in smaller CIs than median scores. With min-max normalization, scores are below 1, so optimality gap corresponds to 1 - mean scores.

Figure A.32: **Aggregate metrics on Procgen** based on 16 tasks with 95% CIs. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling with 50,000 bootstrap resamples. We compare PPO [92], MixReg [111], UCB-DrAC [81], PLR [48], PPG [19] and IDAAC [80]. All results are based on 10 runs per game.



(a) 100k step benchmark.



(b) 500k step benchmark.

Figure A.33: **Ranking on individual tasks on DM Control** 100k and 500k step benchmark. The ith column in the rank distribution plots show the probability that a given method is assigned rank i, when compared to other methods. These distributions are estimated using stratified bootstrap with 200,000 repetitions. We observe that no single algorithm consistently ranks above other algorithms on all tasks, making comparisons difficult without aggregating results across tasks.