Deep Reinforcement Learning at the Edge of the Statistical Precipice

Anonymous Author(s)
Affiliation
Address
email

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. However, only reporting point estimates ignores the statistical uncertainty implied by the use of a finite number of evaluation runs. Beginning with the Arcade Learning Environment (ALE), the shift towards computationally-demanding benchmarks has led to the practice of evaluating only a handful 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 assert reporting interval estimates of aggregate performance and propose performance distributions 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 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 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 [4], 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[1].

A small number of training runs (Figure 1) coupled with high variability in performance of deep RL algorithms [11, 12, 32, 54], often leads to substantial statistical uncertainty in reported point estimates. While evaluating more runs per task has been prescribed to reduce uncertainty and obtain reliable estimates [15, 32, 37], 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 [55] requires more than 1000 GPU training days. As we move towards more challenging and complex RL benchmarks (*e.g.,* StarCraft [82]), evaluating more than a handful of runs will

[1] A run can be different from using a fixed random seed as fixing the seed may not be able to control all sources of randomness such as randomness from non-determinism of ML frameworks with GPUs.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

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 [38] – often requiring many more runs to achieve negligible statistical uncertainty in reported estimates.

Ignoring the statistical uncertainty in results on deep RL benchmarks 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 [18], often at the expense of better methods being neglected or even rejected early [50, 53] as such methods fail to outperform inferior methods simply due to less favorable random conditions. Furthermore, reporting point estimates can erroneously lead the field to conclude which methods are *state-of-the-art* [9, 64], ensuing wasted effort and sometimes degradation in performance over existing methods when applied in practice [80]. 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 [3, 34, 59]. Finally, unreliable results could erode trust in deep RL research itself [35].

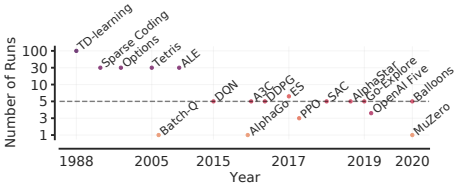


Figure 1: **Number of runs in RL over the years**. Beginning with DQN [57] on the ALE, 5 or less runs are common in the field. Here, we show representative RL papers with empirical results, in the order of their publication year: TD-learning [73], Sparse coding [74], Options [76], Tetris (CEM) [77], Batch-Q [23], ALE [4], DQN [57], AlphaGo [70], A3C [58], DDPG [51], ES [66], PPO [68], SAC [27], AlphaStar [82], Go-Explore [20], OpenAI Five [7], Balloon navigation [6] and MuZero [67].

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 [4] (Atari 100k, Atari 200M), Procgen [13] and DeepMind Control Suite [78], 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 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* such as bootstrap confidence intervals [21], 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 is independent of performance on all except at most 2 tasks. 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 [4, 63], these distributions result in *performance profiles* [19] 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 which each provide a scalar, *normalized score* $x_{m,n}$, $m = 1, \ldots, M$ and $n = 1, \ldots, N$. These normalized scores are obtained by linearly rescaling per-task scores[2] 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 [57]. 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,

[2]Often the average undiscounted return obtained during an episode (see Sutton and Barto [75] for an explanation of the reinforcement learning setting).

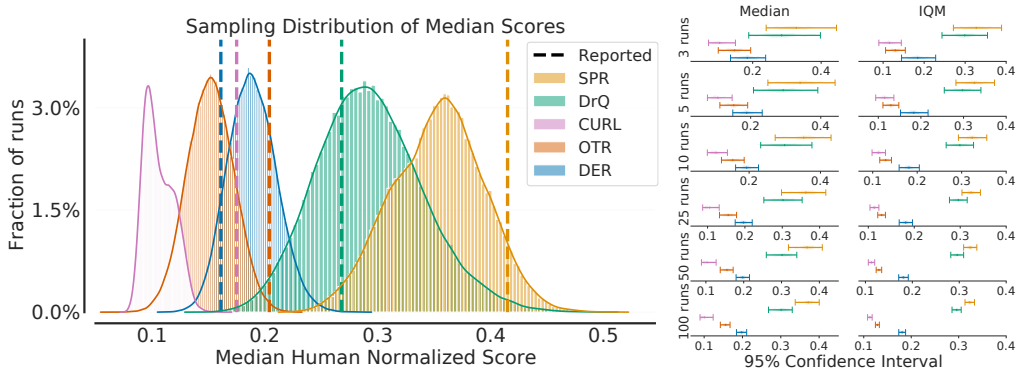


Figure 2: **Left**. **Distribution of median normalized scores** computed using 100,000 different sets of N runs subsampled uniformly with replacement from 100 runs. For a given algorithm, the sampling distribution shows the variation in the median scores when re-estimated using a different set of runs. The reported *point estimates* of median, 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 (Figure 8) for varying N. There is a substantial uncertainty in median scores even with 50 runs. IQM has much smaller CIs than median. While improvement from SPR over DER with 5 to 25 runs is not statistically significant, claiming "no improvement" would also be misleading as evaluating more runs indeed shows that the improvement is significant.

randomized initial parameters, but also hardware considerations such as non-determinism in GPUs and in machine learning frameworks. 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 a 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 common 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* 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 statistic can be interpreted as an estimate of plausible values for the true statistic. 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 statistic 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 statistic lies outside the 95% CI, then a sampling event has occurred which had a probability of 5% of happening by chance.

Remark. Following Amrhein et al. [2], Wasserstein et al. [84], we recommend the use of 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 [24, 26, 56] such as 1) lack of statistically significant results does not demonstrate the absence of effect, 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 [38], a popular offshoot of the ALE for evaluating data-efficiency in deep RL. In this benchmark, algorithms are evaluated on

only 100k steps (two hours of real-time 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 [30, 43, 47] or 5 runs [38, 39, 41, 42, 52, 65, 79, 87], and more rarely, 10 [31, 69] or 20 runs [44].

Our case study compares the performance of five recent deep RL algorithms, namely: (1) DER [79] and (2) OTR [39], (3) DrQ [41], (4) CURL [44], and (5) SPR [69]. We chose these methods as representative of influential algorithms within this benchmark; SPR, in particular, is currently reported as state-of-the-art on Atari 100k. 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 the Appendix for more details.

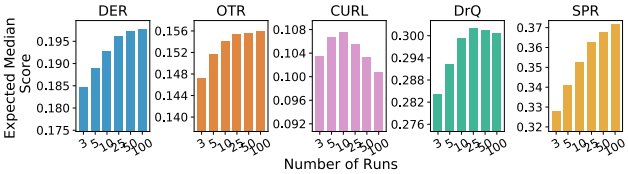


Figure 3: **Expected sample median scores**. The expected score for N runs is computed by repeatedly subsampling N runs with replacement out of 100 runs for 100,000 times.

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 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 43% of its mean improvement over the previous state-of-the-art, DrQ (+0.07 points). Adding to the issue, the magnitude and sign of this bias strongly depends on the algorithm being evaluated.

Statistical concerns cannot be satisfactorily addressed with few runs. While claiming statistical power from 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 intervals (CIs)[3] 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.

In a separate experiment, we consider two identical N-run experiments involving SPR, except that we artificial inflate one of the experiments' score by a fixed fraction or *lift* of $+\ell\%$ (Figure 4). In particular, $\ell = 0$ corresponds to running the same experiment twice. 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 [55]. However, the field has seen a number of alternative protocols used, including reporting the maximum evaluation score achieved during training [1, 57] or across multiple runs. A similar protocol is also used by CURL and SUNRISE [47] (see Appendix).

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,

[3] Specifically, we use the m/n bootstrap [8] to calculate the interval between $[2.5^{th}, 97.5^{th}]$ percentiles of the distribution of sample medians.

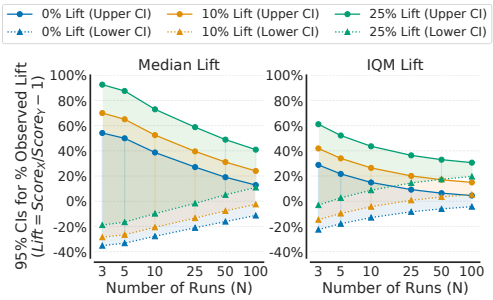


Figure 4: **Detecting score lifts**. **Left**. 95% CIs for observed lift with median scores, and **Right**. 95% CIs for observed lift with IQM (Section 4) when comparing SPR with an algorithm that performs ℓ% better.

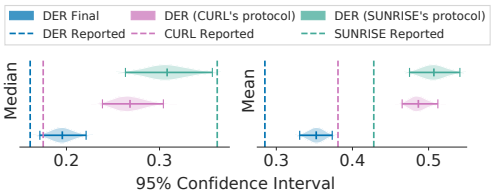


Figure 5: **Normalized DER scores** with non-standard evaluation protocols. Gains from SUNRISE and CURL over DER can mostly be explained by such protocols.

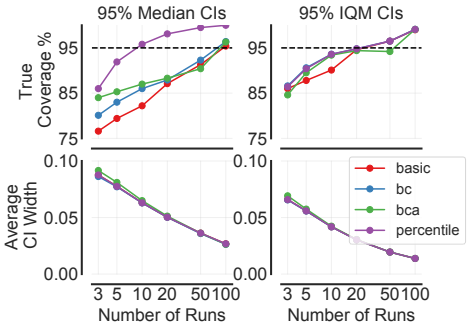


Figure 6: **Validating 95% Stratified Bootstrap CIs** for a varying number of runs for median and IQM scores for DER. The true coverage % is computed by sampling 20,000 sets of K runs without replacement from 200 runs and checking the fraction of 95% CIs that contains the true estimate approximation based on 200 runs. Percentile CIs has the best coverage while achieving a small interval width to other methods. Also, CI widths for IQM are 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.

without care this protocol 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. Empirically, 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 nearly all of SUNRISE's improvement over DER can be explained by the change in evaluation protocol (Figure 5).

4 Accounting for Uncertainty in Deep RL Evaluation

Our case study shows how increasing the number of runs alone cannot address the issues posed by statistical uncertainty, at least not when applied to 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.

Interval estimates. 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 [21] 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 results with N runs [11, 15, 32], this is less useful when N is small, 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 (percentile, basic, bias-corrected (BC) and bias-corrected and accelerated (BCa); see [22] and Appendix). We find that percentile intervals provide good interval estimates for as few as $N = 10$ runs for both median and IQM scores (defined below).

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. Instead, we recommend the use of *performance*

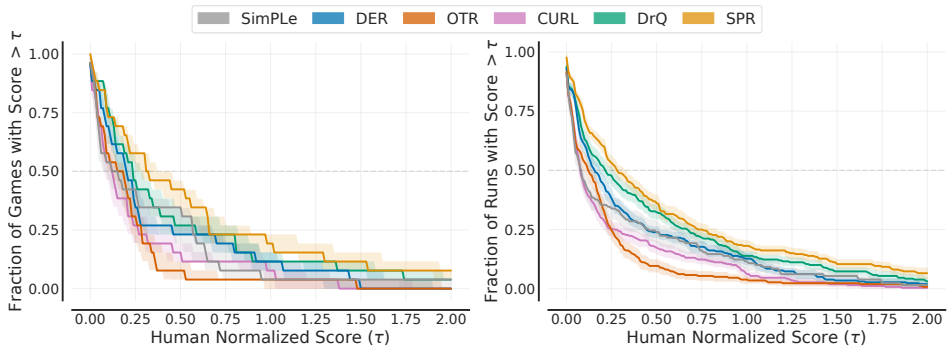


Figure 7: **Performance profiles on Atari 100K** based on average score distributions **(left)**, and score distributions **(right)**. Shaded regions show pointwise 95% confidence bands based on percentile bootstrap with stratified sampling. The profiles on the right are more robust to outliers and have smaller confidence bands. We use 10 runs to show the robustness of profiles with a few runs. For SimPLe [38], 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.

profiles [19], commonly used in benchmarking optimization software. Performance profiles visualize the empirical tail distribution function (Section 2) of a random score.

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.[4] In addition, performance profiles are robust to outlier runs and insensitive to small changes in performance across all tasks [19].

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

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

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 [4] 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: $\sigma^2_X = \frac{1}{M^2 N}\sum_{i=m}^{M} F_m(\tau)(1 - F_m(\tau))$ 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.

Another alternative [63] is to replace scores in a performance profile 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 [85]. However, (1) this profile is 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.

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*[5] the other [49]. In RL benchmarks with a large number of tasks, however, stochastic domi-

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

[5] 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, $P(X > \tau) > P(Y > \tau)$.

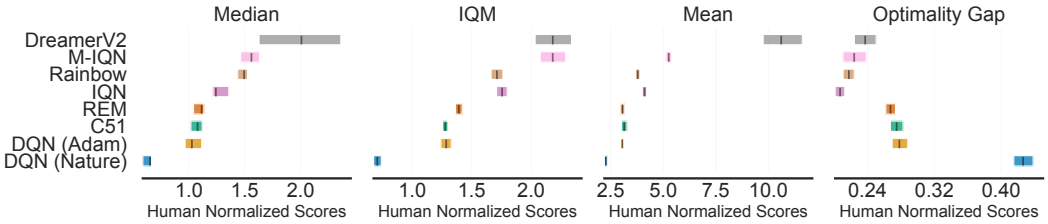


Figure 9: **Aggregate metrics on Atari 200M** with 95% CIs based on 55 games with sticky actions [55]. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling. IQM typically results in smaller CIs than median 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) [57], DQN with Adam optimizer, C51 [5], REM [1], Rainbow [33], IQN [17], Munchausen-IQN (M-IQN) [81], and DreamerV2 [29]. All results are based on 5 runs per game except for M-IQN and DreamerV2 which report results with 3 and 11 runs.

nance is rarely observed: performance profiles often intersect at multiple points. Finer quantitative comparisons must therefore entail aggregate metrics.

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.

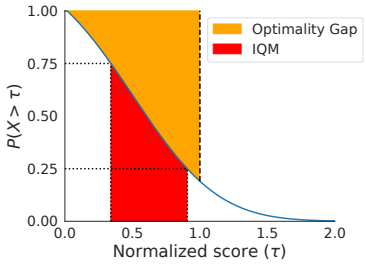


Figure 8: **Aggregate metrics**. For a non-negative random variable X, IQM corresponds to the red shaded region while optimality gap corresponds to the orange shaded region in the performance profile of X.

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 runs. IQM interpolates between mean and median, which are 0% and almost 50% trimmed means respectively. IQM is robust to outliers yet has considerably less bias than median. It is also a better indicator of overall performance than the median as it is calculated using 50% of the runs (versus at most two tasks). Our experiments show that 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.13).

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. Naturally, the threshold γ may be chosen differently based on the normalization scheme in use, which we discuss further in the Appendix.

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 $P(X_m > Y_m)$ 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 metrics is still an open question, our proposed alternatives are more robust and require fewer runs to reduce uncertainty than the ones currently used in practice.

5 Re-evaluating Evaluation on Deep RL Benchmarks

5.1 Arcade Learning Environment: Atari 200M

Training RL agents for 200M frames on the ALE [4, 55] 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 [29] exhibits a large amount of uncertainty in aggregate scores. While M-IQN [81] claimed better performance than Rainbow [33] in terms of median normalized scores, their interval estimates strikingly overlap. Similarly, while C51 [4] is considered substantially better

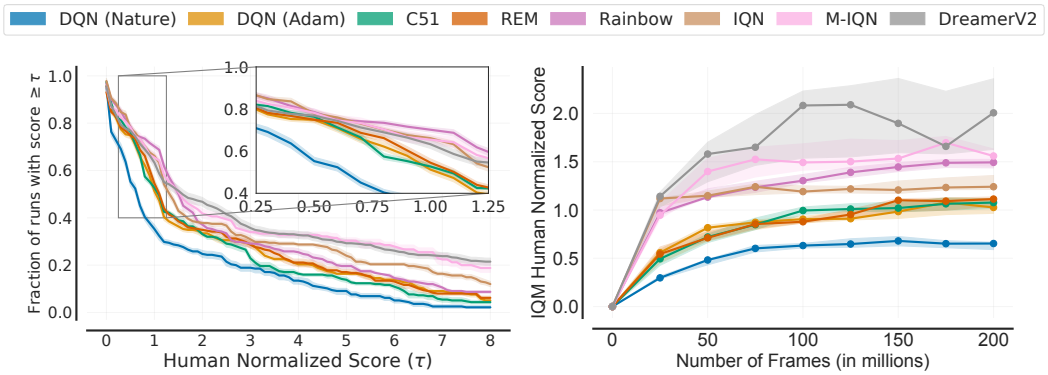


Figure 10: **Atari 200M evaluation**. **Left**. Score distributions using human-normalized scores obtained after training for 200M frames. **Right**. Sample-efficiency of agents as a function of number of frames measured via IQM human-normalized scores. Shaded regions show pointwise 95% percentile stratified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

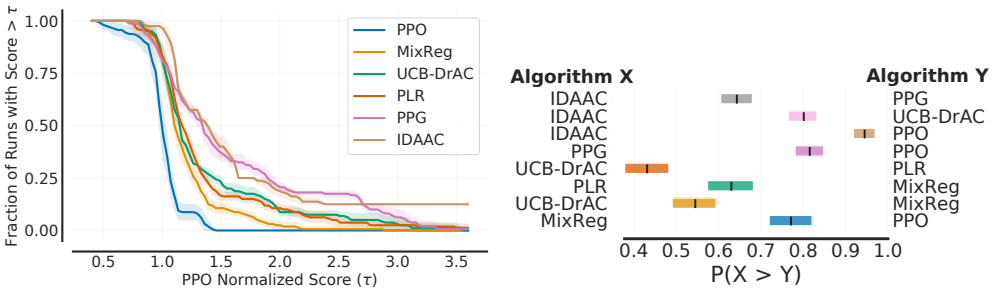


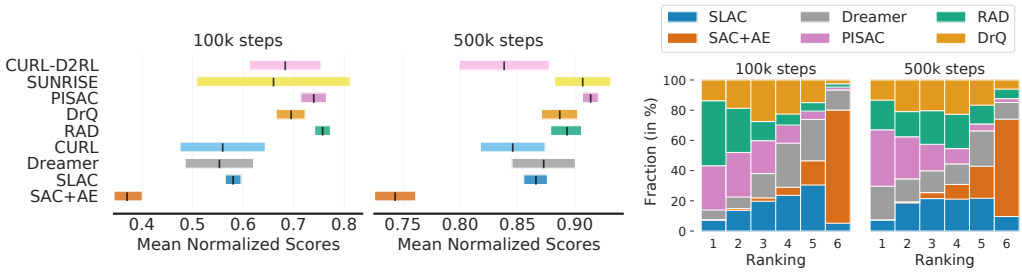
Figure 11: **Procgen evaluation** results based on easy mode comparisons [61]. **Left**. Score distributions which compare PPO [68], MixReg [83], UCB-DrAC [62], PLR [36], PPG [14] and IDAAC [61]. Shaded regions indicate 95% percentile stratified bootstrap CIs. **Right**. Each row shows the probability 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.

than DQN [57], 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). 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 performance. 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).

5.2 Procgen benchmark

Procgen [13] 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 [68] as most methods are built on top of it. However, Figure 11 (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 [13] and reporting aggregate metrics based on such scores (see Appendix).

While publications 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 probability that an algorithm which claimed improvement over a prior algorithm performs better than it?" (Figure 11, 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 [62]



(a) Mean Scores (= 1.0 - Optimality Gap). (b) Rank comparisons.

Figure 12: **DeepMind Control Suite evaluation** results, averaged across 6 tasks, on the 100K and 500K benchmark. We compare SAC+AE [86], SLAC [46], Dreamer [28], CURL [72], RAD [45], DrQ [40], PISAC [48], SUNRISE [47], and CURL-D2RL [71]. The ordering of the algorithms in the figures is based on their claimed relative performance – all algorithms except Dreamer claimed improvement over at least one algorithm placed below them. Individual runs for each method were provided by from their respective authors except for CURL, CURL-D2RL, and SUNRISE for which we used their reported scores. **(a)** Interval estimates show 95% stratified bootstrap CIs for methods with individual runs and 95% studentized CIs for the rest. Normalized scores are computed by dividing by the maximum score (=1000). **(b)** The ith column in the rank distribution plots show the probability that a given method is assigned rank i, averaged across tasks, when compared to other methods. These distributions are estimated using stratified bootstrap with 200,000 repetitions.

improves upon PLR [36]. We note that a number of improvements reported in the existing literature are only $50 - 70\%$ likely.

5.3 DM Control Suite

Recent continuous control papers benchmark performance on 6 tasks in DM Control [78] 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 12b). Furthermore, there are huge overlaps in 95% CIs of mean normalized scores for most algorithms (Figure 12a). These findings suggest that a lot of the reported improvements are spurious, resulting from randomness in the experimental protocol.

6 Discussion

We saw, both in our case study on the recent Atari 100k benchmark and with our analysis of other benchmarks commonly used by the RL community, 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 [11, 15, 16, 32, 37], 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. [25] calls *results reproducibility*), others [60] 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, since it does not really address the question of whether an algorithm would perform well under similar conditions but with different seeds, as well as 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 support RL researchers in this endeavour, we will release an easy-to-use Python library as well as a colab notebook for producing and analyzing performance profiles, robust aggregate metrics, and interval estimates, as well as all the runs used in our experiments. Again, we emphasize the importance of published papers providing results for all runs to allow for future statistical analyses.

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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

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

[20] 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.

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

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

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

[24] 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.

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

[26] 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.

[27] 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.

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

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

[30] 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.

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

[32] 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.

[33] 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.

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

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

[36] Minqi Jiang, Ed Grefenstette, and Tim Rocktäschel. Prioritized level replay. *arXiv preprint arXiv:2010.03934*, 2020.

[37] 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.

[38] 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.

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

[40] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ICLR*, 2021.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

[47] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *arXiv preprint arXiv:2007.04938*, 2020.

[48] 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.

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

[50] Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael H. Bowling. State of the art control of atari games using shallow reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2016.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

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

[62] 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.

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

[64] 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.

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

[66] 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.

[67] 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.

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

[69] 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.

[70] 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.

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

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

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

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

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

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

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

[78] 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.

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

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

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

[82] 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.

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

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

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

[86] 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.

[87] 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.

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]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix
   (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
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix
   (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
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]