Let's Play Again: Variability of Deep Reinforcement Learning Agents in Atari Environments

Kaleigh Clary
University of Massachusetts Amherst
Amherst, MA 01003
kclary@cs.umass.edu

Emma Tosch
University of Massachusetts Amherst
Amherst, MA 01003
etosch@cs.umass.edu

John Foley
Smith College
Northampton, MA 01060
jjfoley@smith.edu

David Jensen
University of Massachusetts Amherst
Amherst, MA 01003
jensen@cs.umass.edu

Abstract

Reproducibility in reinforcement learning is challenging: uncontrolled stochasticity from many sources, such as the learning algorithm, the learned policy, and the environment itself have led researchers to report the performance of learned agents using aggregate metrics of performance over multiple random seeds for a single environment. Unfortunately, there are still pernicious sources of variability in reinforcement learning agents that make reporting common summary statistics an unsound metric for performance. Our experiments demonstrate the variability of common agents used in the popular OpenAI Baselines repository. We make the case for reporting post-training agent performance as a distribution, rather than a point estimate.

1 Introduction

Deep reinforcement learning faces substantial and unusual challenges in evaluation and reproducibility [5, 6]. Based on reports of common evaluation practices in the field, many RL researchers train a few candidate models[1] and then report performance using some aggregate function of the scores of the trained agents.[2] This aggregate can be taken from the learning curves of the model or from scores collected in post-training episodes.[3] Such metrics are often used for comparison against reported benchmark results. Underlying this practice is the expectation that the average reward provides a meaningful summary of agent performance when operating in the environment.

Researchers know that randomness in the training algorithm of a deep RL agent can have a large impact on the agent's ability to effectively learn a policy. For example, an agent with poorly seeded weights, or poorly chosen random actions early in training, may find its way to a local optimum and never achieve high reward. With this context in mind, deep RL researchers often train multiple agents with different random seeds to account for this variability in training.

Unfortunately, careful selection of random seeds during training does not necessarily translate to careful consideration of reporting metrics for reproducibility. Reporting and analyzing the choice of random seed is critical for reproducing a training procedure, but it is also critical for assessing

---

[1]We use the term model to refer to the learned component of a policy.
[2]We use *score* to mean any measure of performance used to compare the performance of different RL agents.
[3]Evaluation of online learning methods is beyond the scope of this work.

the variability inherent in the final evaluation of an agent. Reproducibility in RL is increasingly a cause for concern, due to the growing complexity of training, evaluation, and agent architecture. The process of selecting and reporting random seeds represents a potential bias in the training of RL agents and requires closer inspection, as there is high potential for researchers to unknowingly run afoul of the *multiple comparisons procedure*.

To our knowledge, this paper is the first to explore variability of individual deep RL models. We will use the term *seeded model* to refer to the specific model specified by hyperparameters and model random seed. We constrain our analysis to models operating in Atari environments [1] and leave a broader analysis to future work.

We show empirically that seeded models have diverse performance distributions, demonstrate how this can affect model selection, and point to commonly used techniques for correcting for variability.

2 Problems with Reproducibility and Reporting Practices

There has been much recent interest in improving reproducibility in deep reinforcement learning [5, 6, 7]. One major challenge to reproducibility is the effect of uncontrolled variability during either training or evaluation. Recent work has attempted to isolate the effects of various sources of variability in RL, examining variability due to differences in algorithm implementation, hyperparameter selection, environment stochasticity, network architectures, and random seed selection [5]. We focus on reproducibility and variability as it pertains to the effects of erroneously reporting point estimates, and specifically analyze the implications of work on the effects of random seeds. A common practice is to extract a single sample of performance from each seeded model from the learning curve of that model and report it as representative of a learned agent's behavior [2, 5]. However, without characterizing the variability of a seeded model and and specifying how the single model was selected, it is not clear that a single sample is sufficient to describe the behavior of that model.

Model selection and random seeds. Most commonly, researchers select one model from among many, based on which model has the highest *mean* score across some number of trials (each associated with a different random seed) [3, 5, 9]. Less commonly, researchers select one model from among many based on which model has the highest *maximum* score across some number of trials [9, 11].

Either approach can produce unexpected biases due to the high variability of scores. Using the mean score is an appropriate performance measure if the score distribution is well-behaved (e.g., Gaussian). However, if the performance distribution is multimodal, has fat tails, or has outliers, the mean alone can be a misleading summary of model performance. Using the maximum score can be extremely misleading if the different models produce score distributions with widely differing variance. In such a case, choosing based on the maximum score will favor models with high-variance distributions rather than models with highest expected value [8]. There also exists a second-order effect in both cases, because the expected distribution of either the mean or the maximum has variance itself, and selecting models based on a sample from that distribution (a specific mean or maximum) will tend to favor those models with high inherent variability.

Aggregates of model performance summarizing behavior across random seeds can also be affected by variability of seeded model performance. With no convergence guarantees, less control over the representation of state, and the sources of variability listed previously, it is unclear whether one should expect individual trained deep RL agents to exhibit well-behaved, narrow distributions of performance. Reporting over a sample of seeds, each potentially exhibiting variability, introduces an additional degree of freedom in the statistical analysis of model performance. Thus, it is important to characterize the variability of agent performance associated with different random seeds.

Background: Multiple Comparisons Procedures. Jensen & Cohen [8] define a *multiple comparison procedure* (MCP) as any process that generates multiple items (e.g., models), estimates a score (with some variability) for each item, and then selects the item with the maximum score. MCPs are common in machine learning, and they result in positive bias in the score of the selected item. This statistical effect of MCPs is the underlying reason for regularization methods such as complexity penalties and evaluation procedures such as cross-validation.

Failing to account for variability in model scores can lead to incorrect conclusions about model rankings. By selecting the models with the maximum score, it is likely that the maximum will be an outlier rather than an unbiased estimate of performance. Selection procedures using the mean

are also MCPs, as that mean score has variability (and possibly different variability depending on training approach, agent architecture, etc.). The bias of such estimates is magnified as the number of models increases, as the length or number of testing episodes decreases, and as the inherent variability of performance increases. Worse, if the inherent variability of models differs, then an MCP will often result in selecting the model with the highest variance, rather than the model with the highest expected score.

Such high variability of results makes performance evaluations vulnerable to cherry-picked point estimates of performance and even scrupulous researchers can run into the problems outlined above. This can result in incorrect conclusions when comparing learning methods and architectures, and it can misdirect individual and field-wide threads of research [4].

3 Complications due to Variability

We collected end-game scores of 100 games for each of 10 different seeded models and examined the resulting score distributions of each seeded model set.[4]

OpenAI Baselines Benchmarks. To create a suite of models, we replicate the experiments in OpenAI Baselines Benchmarks [3]. We train different models on 10 random seeds each for five ALE Atari environments[5]: Beam Rider, Breakout, Q*bert, Seaquest, and Space Invaders. We compare a2c [9], acktr [11], and ppo2 [10] models trained using OpenAI Baselines implementations with default hyperparameters. Learning curves for our models are provided in Appendix A.

Variability of Seeded Models. We report score distributions for four model sets from our suite in Figure 1. These results illustrate why it is best practice to test and account for seeded model variability in random seed and model selection.

- **Stationary Performance Distributions across Random Seeds.** In Figure 1a, we see that each random seed exhibits similar performance distributions. Sampling a small number of times from each of these random seeds may look like variability between random seeds, when in fact the distribution of scores is due to variability of the seeded model.
- **Fat-tailed Distributions.** The score distribution for Breakout models shown in Figure 1b covers nearly the entire range of possible scores in the first level of Breakout. Reporting the mean, standard error of 209.3 ± 3.2 gives an inaccurate basis of comparison between Breakout models.
- **Multimodal Distributions.** The distribution of Q*bert scores shown in 1d is bimodal[6] in seeded model performance, while Space Invaders scores (Figure 1c) are bimodal when taking together the performance of all seeded models. This distinction is lost if we do not examine variability both within and between seeded models.

Degenerate Model Selection. The mean for a2c on Q*bert (2425.4 ± 53.8) is not a reasonable expectation for model performance; it is among the least densely observed scores. The wide distribution of scores in ppo2 on Breakout means that the mean is not very informative. Using the max to choose models in any of these model sets does not guarantee a high-performing model. This means that not only is reporting difficult, but even selecting a good model for production use requires knowledge of within-model variance.

4 Discussion

These results show that seeded models can also exhibit variability in Atari deep RL. In light of this, we argue for reporting model performance as a distribution in place of point estimates. Even the mean with standard error is inappropriate when the distribution is bimodal. Note that generating the distribution of scores for a seeded model simply requires running the trained model for some

---

[4]Our code is available at: `https://github.com/kclary/variability-RL`.

[5]Researchers often insert randomly repeated actions to make Atari environments more stochastic. Our experiments use the ALE *NoFrameskip-v4* Atari environments, which do not include these stochastic variations [1].

[6]The modality exhibited by these models is not replicated in Q*bert agents trained with acktr or ppo2, so this distribution is likely not a consequence of the environment. For all three distribution plots, see Appendix B.
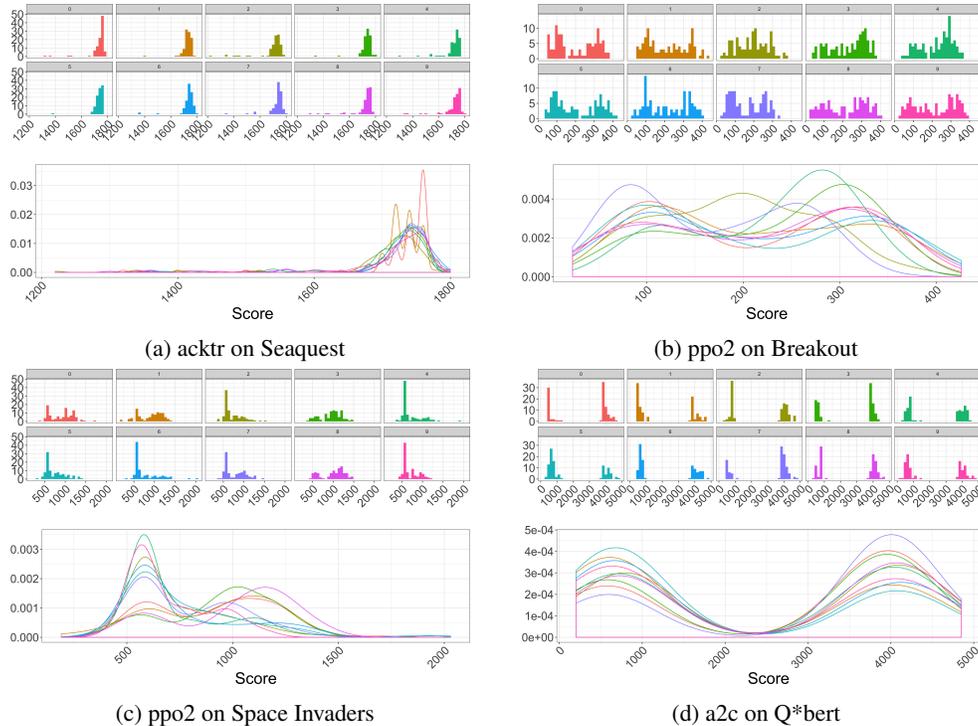
Figure 1: A selection of seeded models (each seed represented as a different color) on a subset of Atari games featured in the OpenAI Baselines benchmarks. Each agent-environment combination depicts histograms from 10 seeds. Each histogram represents the frequencies of scores for 100 trials of the given seed. Below the histograms are kernel density estimates for the distributions of scores. Some distributions have greater variability between seeds than others.

number of trial episodes. This is far less resource intensive than training a larger number of models on new random seeds. To more accurately characterize the expected performance of models trained with particular hyperparameter settings, we suggest reporting the performance distributions of a few different random seeds to demonstrate a range of score performance across two axes of variability for a given set of hyperparameters.

To choose a model with expected maximum score among several seeded models for production or publication, use best practices and avoid reporting cherry-picked scores of outlier random seeds. Multiple comparisons procedures are a well-studied process in statistics, and there are several ways to correct for this variability when making a selection. Resampling scores from the model with highest score, Bonferroni adjustment of score samples, and cross-validation over partitions of model scores are each methods used to account for variability in MCPs. For a more complete discussion of these and other MCP adjustment methods, see Jensen & Cohen [8].

Difficulty in finding random seeds and hyperparameters that seed reasonable model performance in deep RL has sometimes been described as *instability*. Referring to these phenomena as instability implies that this behavior is somehow errant and unexpected. Our experiments imply that, at least in Atari environments, variability in seeded model performance is expected behavior, and commonly reported evaluation measures are insufficient for characterizing that variability. Using community standard implementations of algorithms and the deep networks that back them, we find that even seeded models operating in controlled environments can still exhibit wild variability in performance.

We demonstrate the variability of seeded model performance in Atari environments. This variability impacts the conclusions one might draw from current reporting and evaluation practices. We recommend (1) running trained agents in the Atari environment for many episodes to collect samples of end-game score, (2) reporting this distribution of scores to characterize model performance, and (3) using statistical methods to adjust for seeded model variability when choosing among candidate models.
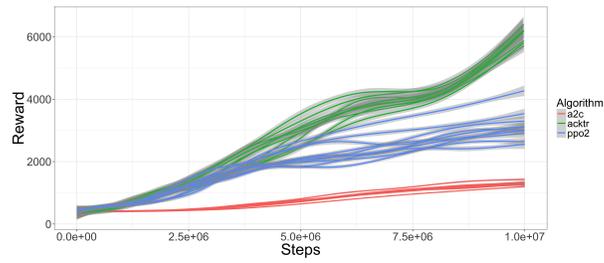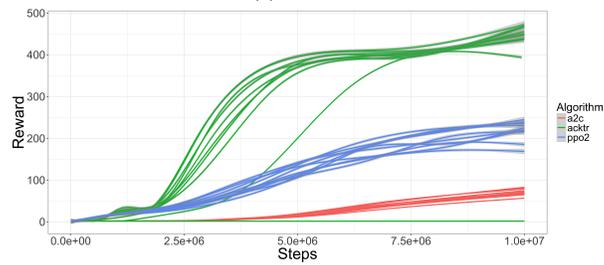
Acknowledgements

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013.

[2] Daniel Cohen, Scott M. Jordan, and W. Bruce Croft. Distributed evaluations: Ending neural point metrics. In *SIGIR; LND4IR*, 2018.

[3] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. OpenAI Baselines. `https://github.com/openai/baselines`, 2017.

[4] Andrew Gelman and Eric Loken. The Garden of Forking Paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. 2013.

[5] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. In *AAAI Conference on Artificial Intelligence (AAAI)*. arXiv preprint 1709.06560, 2017.

[6] Alex Irpan. Deep Reinforcement Learning Doesn't Work Yet. `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.

[7] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. In *Reproducibility in Machine Learning Workshop (ICML)*. arXiv preprint:1708.04133, 2017.

[8] David D. Jensen and Paul R. Cohen. Multiple Comparisons in Induction Algorithms. *Machine Learning*, 38(3):309–338, Mar 2000.

[9] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1928–1937. JMLR.org, 2016.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5279–5288. Curran Associates, Inc., 2017.
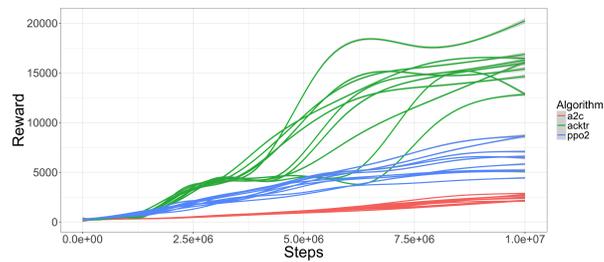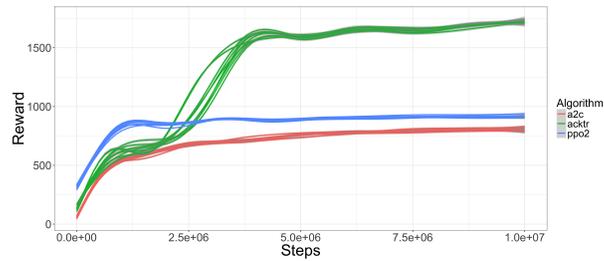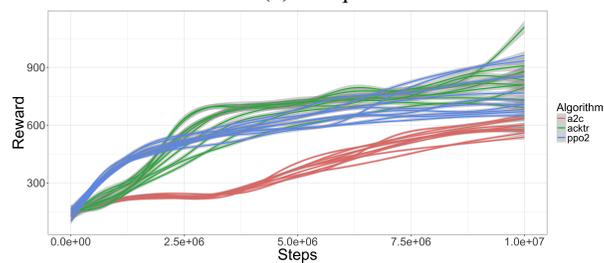
A   Learning Curves

(a) BeamRider

(b) Breakout

(c) Q*Bert

(d) Seaquest

(e) SpaceInvaders

Figure 2: Learning curves for each agent (a2c [9], acktr [11], ppo2 [10]) on each of the five Atari game environments featured in the OpenAI Baselines benchmarks.

B Complete Agent-Environment Histograms

The following five environments are featured in OpenAI Baseline's Atari benchmark (BeamRider, Breakout, Q*Bert, Seaquest, and SpaceInvaders). We chose three agents for evaluation: a2c [9], acktr [11], and ppo2 [10]. The OpenAI benchmark includes six agents; we decided against deepq due to the required training time, acer due to a bug that was present in the Baselines code at the time of evaluation, and trpo_mpi due to early issues during training related to MPI calls.

Each agent-environment combination depicts the histograms from 10 seeds. Each histogram represents the frequencies of scores for 100 trials of the given seed. Below the histograms are kernel density estimates for the distributions of scores. Some distributions have greater variability between seeds than others.
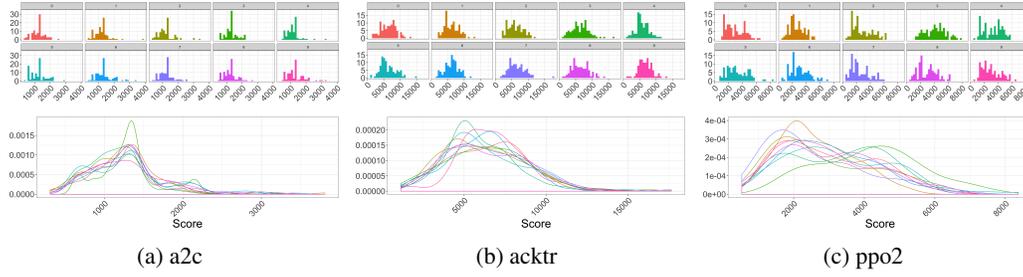

(a) a2c　　　　　　　　(b) acktr　　　　　　　　(c) ppo2
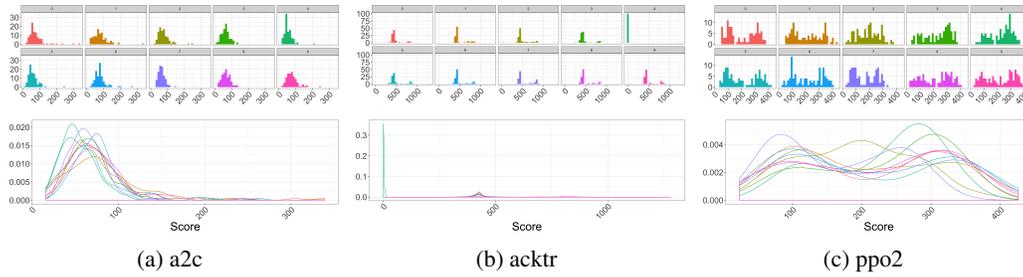
Figure 3: BeamRider Histograms

(a) a2c　　　　　　　　(b) acktr　　　　　　　　(c) ppo2

Figure 4: Breakout Histograms

(a) a2c　　　　　　　　(b) acktr　　　　　　　　(c) ppo2

Figure 5: Q*Bert Histograms

(a) a2c

(b) acktr
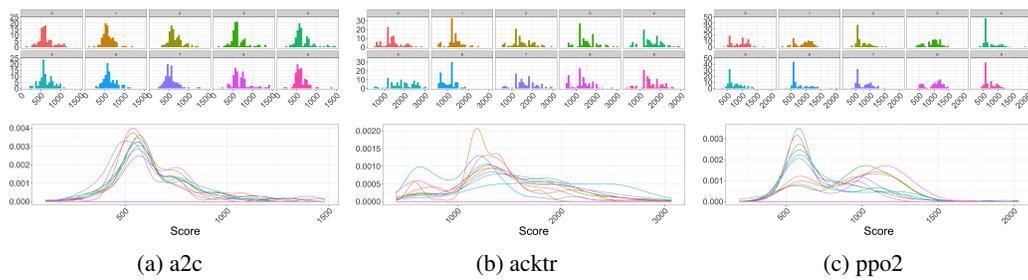
(c) ppo2

Figure 6: Seaquest Histograms

(a) a2c

(b) acktr

(c) ppo2

Figure 7: SpaceInvaders Histograms