Benchmarking and Reliability in Reinforcement Learning

Gurjeet Singh
University of Padua
gurjeet.singh@studenti.unipd.it

Abstract

The absence of standardized methods and lack of adequate metrics for measuring Reliability and evaluating performances improvement of novel reinforcement learning (RL) is still a debated problem. We examine in this report past and recent approaches in order to give a summary of proposed solutions for addressing evaluation and comparison of different models. In particular, we describe metrics, methods, and statistical tests to follow for pertinent analysis, by describing their explanatory powers, usage, and related issues.

1 Introduction

In the recent years there has been a great growth in the area of Deep Reinforcement Learning (DRL); novel approaches and techniques are constantly being invented and applied in many different domains. But to effectively compare model performances with pre-existing approaches, advanced studies have to be carried out on evaluation methods. This is needed for detecting and improving their weaknesses, and enhancing results for further works.
Beside new reinforcement learning models, there have also been an increasing growth for developing benchmarks and environments. Thanks to diverse benchmarking tests we can detect different aspects of the agent and give a global picture of the learning by comparing them across different environments and algorithms.

Despite the already existing benchmarks, it is still hard to reproduce the results of Deep RL algorithms. At first, we have to recall that these approaches have to deal most commonly with complex environments, which present strong stochasticity that could result in high variance and instability in the learning. However, still most of the issues and main causes of an algorithm to fail are because of the variance present in the learning algorithm, or either because of the variance incorporated in the dynamics of the environment. Furthermore, as reported by Henderson et.al [11], the importance of reporting hyperparameters is essential to make the results of the experiments more reproducible. As they discussed, just few changes on the configuration of neural network methods can lead into completely different results. They have shown that many details of hyperparameters are not consistent and no ranges are provided, in addition, a lot of experimental details are not reported ending to un-reproducible experiments. Thus, being consistent and reporting the hyperparameters are extremely important to make the experiment reproducible.

Although more benchmarks are now released, there are no guidelines or standardized methods for fair comparisons among algorithms that have been defined yet. Moreover, if poorly tests are carried out, they can lead to misleading results. In addition, benchmarks are also used improperly and in a naive way for comparing different algorithms, indeed they should be exploited better, as we will see in this report.

Defining the appropriate metrics and statistical tests are required to diagnose and compare different models approaches. For this reason, we will discuss in this work some novel measures and statistical tests, proposed by Chan et al. [5], which are based on robust statistics to assess the reliability and reproducibility of reinforcement learning model that deal with instability in their results. Finally, based on these measures a fair comparing method to evaluate different models will be discussed using statistical test.

2 Method

In this section, we state past and recent measures and test methodologies proposed in the scientific literature to assess evaluation methods of RL algorithms. At first, we introduce some well-known and **commonly used metrics**. Secondly, we discuss and suggest some solutions for two major issues related to the previous metrics, which are **inconsistency across runs** and **multiple environments**. Besides these issues, a separate paragraph will discuss the evaluation of **model-based reinforcement learning (MBRL)** [14], which requires additional evaluations and particular attention. Finally, we present rigorous methods for evaluating RL models and measuring their **reliability** using **robust statistics and statistical tests**.
For each approach, we describe not only the proposed methods but also their usage, aims, explanatory power, and highlight issues related to the measures and their applications.

2.1 Common metrics

As mentioned in the Introduction in Reinforcement learning models reproducing an experiment is a challenging problem. To carry out simulations and replicate results in some experiments, we require specific metrics for evaluation. A common metric in RL algorithms is the average cumulative reward over some episodes T, i.e the average returns, and it is defined as:

$$G(A) = \mathbb{E}_{s_{t+1} \sim f} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]. \tag{1}$$

Other metrics like maximum reward achieved over a fixed number of time steps have also been used, but due to the unstable nature of many algorithms reporting them are inadequate. Even reporting only the average returns results to be misleading, in fact they have to be combined at least with the confidence intervals (CI). In such a manner, we are able to track the variance present in the average return. This measure can be helpful as a first evaluation for simple comparison among different algorithms. But most importantly, it allows to tune important hyperparameters of the algorithm.

2.2 Inconsistency across runs

The average return and the CI not only allow to calibrate major hyperparameters, but also to detect some pitfalls in the learning of the agent. In fact, it has been shown [11], that the variance between runs of a model can be enough to create statistically different distribution just by varying the random seed across trials or by applying different simulation on the test set. This issue can be easily detect by plotting the average returns with the relative confidence interval from different runs. Moreover, it can even happen by averaging several learning results together across different random seed. The major cause of this problem can be associated with the variance present in the environment or in the learning process, it can be mostly caused also by the chosen sample size that is not enough to have a consistent learning behaviour across different runs.
To overcome the inconsistency across runs, we can apply power analysis and statistical test in order to understand the required sample size such that performances have same distribution on different runs. An example of sample size analysis is bootstrap power analysis [19], as suggested by Henderson et.al [11], by running many bootstrap simulation we can determine the sample size required by looking at the percentage of the simulations result in statistically significant values, where in case of low percentage of significant values a large sample size is required in each run. Further details on

sample size studies applied to RL problems can also be found in the chapter 3 and 4 of Colas et.al [6], where they used Welch's t-test and power analysis methods to understand the sample size required by the model.

2.3 Inconsistency on Multiple Environments

Another type of inconsistency in performances of RL models relates to multi-environments tests, i.e when a model is compared on different implementations for the same agent tasks. Indeed, it has been shown firstly by Henderson [11], and lately by Helfenstein [10] that evaluation of models across different environments are barely addressed and tested. They have found that algorithms performances can differ significantly depending on the chosen environment, leading to misleading results when comparing different algorithms. Thus, it should be more considered not just to test the model on a single environments but also compare the learning of the agent on multiple environments in order to understand their variance and reliability.
In their papers, some of the causes of this issue have also been detected. They addressed two major factors, the presence of biases in the learning of the agent and the usages of biased preprocessing techniques applied on the environment. Indeed, in their experiments they found that biases in the agent can be introduced with reward scaling normalization layers, which can have a large effect in gradient based methods. Large and sparse output scale can result in neurons saturation and inefficiency in the learning, thus compressing the space of estimated expected returns in action value function based method using reward scaling can be very fruitful to agent. But results are inconsistent across environments causing failure to learn on unstable environment, where Deep Q-Value function approximator are used or where untuned reward scales are defined. To overcome this issues, a better approach would be to adaptively rescale reward targets with normalized stochastic gradient a made by Van Hasselt et.al [17].

2.4 Evaluation of Model-based Reinforcement Learning

All the previous issues and methods can be applied to any Reinforcement Learning algorithm, but when considering Model-based Reinforcement Learning (MBRL) [14] more tests are required to carry out. These type of algorithm have the potential to be significantly more sample efficient by modelling the transition dynamic. Thus, evaluation on the errors made in the transition dynamic and its effect on the policy learned have to be analyzed more carefully using the related loss function and metrics. Research in this particular field of RL has not been yet standardized. Indeed, it is common that experiments are carried out with self-designed environment, which are sometimes closed-source, leading to un-reproducible algorithms. For this reason, Wang et.al [18] have addressed this issue by proposing new benchmark environments designed for MBRL. They adapted many common environments of model-free RL for MBRL, because contrary to free model approaches, MBRL requires access to an analytical differentiable reward function, such that the gradient can be computed. Thus, in their work they adapted or approximated the reward functions in order to make them differentiable. Furthermore, based on their experiments, they have been able also to detect main causes of performance stagnation of models-based methods. The major one regards the Dynamic bottleneck, i.e when the learned dynamics are stuck at performance local minima significantly worse than using ground-truth dynamics. They address the cause of this issue to the accumulated prediction errors, since MBRL inevitably involves prediction on unseen states. Another factor is that the policy and the learning of dynamics are coupled, which makes the agents more prone to performance local-minima.

2.5 Reliability and robust statistics

As we have seen through different proposed approaches and issues, reliability is a major problem in RL algorithms. Recently, some efforts have been made into defining adequate metric to quantify different aspect of reliability. In fact, Chan et al. [5] have designed a statistical test for enabling rigorous comparisons of algorithms by using robust statistics. These metrics measure the reliability by looking at different aspects: reproducibility, stability, variability/dispersion, and risk.

		Dispersion (D)	Risk (R)
DURING TRAINING	**Across Time (T) (within training runs)**	IQR* within windows, after detrending	**Short-term:** CVaR† on first-order differences **Long-term**: CVaR† on Drawdown
	Across Runs (R)	IQR* across training runs, after low-pass filtering.	CVaR† across runs
AFTER LEARNING	**Across rollouts on a Fixed Policy (F)**	IQR* across rollouts for a fixed policy	CVaR† across rollouts for a fixed policy

Figure 1: Summary of proposed reliability metrics [5]

These properties can be detected by measuring dispersion, and risk of the average rewards. The former one, measure the width of the distribution and it is inferred using the **Inter-quartile range (IQR)**, i.e the difference between the 75th and 25th percentile, $IQR(X) = Q_3(X) - Q_1(X)$, this metric does not require assuming normality of the distribution and it is suitable for asymmetric distribution. Instead, Risk can be derived by quantifying the heaviness and extend of the lower tail of the distribution, which give us the worst-case scenarios of the performances. To measure it **Conditional Value at Risk (CVaR)** [1] can be used and it is defined as:

$$CVaR_\alpha(X) = \mathbb{E}\left[X | X \leq Var_\alpha(X)\right] \quad \alpha \in (0, 1), \tag{2}$$

where VaR_α is the value at Risk, i.e the α-quantile of distribution X and it is defined as

$$VaR_\alpha(X) = -inf\{x \in \mathbb{R} : F_X(x) \geq \alpha\}. \tag{3}$$

By applying the previous metrics on different phases we can extract several information, as summarized by the following Table 1. At first we distinguish two phases, during training and after learning. Within training runs (across time, i.e within each run) we check the stability across time by looking for monotonic improvement in the measures. The IQR across time is applied using a sliding window on detrended training, i.e $y_{t'} = y_t - y_{t-1}$, to not capture long term trend and isolate higher frequency variability. Instead, short term risk measures the most extreme short-term drop over time, by applying the $CVaR_\alpha$ on a distribution obtained by the differences of one point of evaluation to the next one normalized by the distance between time-point $CVaR_\alpha\left(\frac{X_t - X_{t+n}}{n}\right)$, to make it invariance to evaluation frequency. On the other hand, long term risk is obtained by looking at the draw-down at time T, which describes the drop in performance relative to the highest peak so far this way we capture drop over longer timescale but also unusual short term drops.
By contrast, when analyzing the model across runs, we should have reproducible performances and consistency. Here the variability describes the algorithm's sensitivity to random seed which includes initialization of weights, and sensitivity to multiple environments. The $CVaR$ during this phase gives the expected worst performances across all the runs. Once fixing the policy, i.e after learning, the two measures allow to measure the reproducibility of the algorithm by comparing it with the results across runs during the training phase.

Thanks to these reliability metrics, we are able to quantify different aspects of an algorithm and they can help to pinpoint their specific strengths and weaknesses. But when we need to compare different models and consider multiple environments, the metrics can not be compared directly. The results can have different ranges and distribution of reward across multiple environments, thus a suggested approach is to compute the median range of performance. In addition, for a fair comparison of multiple algorithms, ranking method of the agent with related statistical test has been proposed by Chan et.al [5]. They designed a pair permutation test applied after ranking the algorithm for each agent's task on per run-metrics or across run. The permutation test is based on bootstrap sampling on the ranking values, and by applying the difference between the mean ranking across tasks of the two algorithms on each resample we can obtain a distribution over the ranking, assuming under the null hypothesis that runs or episodes are exchangeable. Thus, from the retrieved distribution we

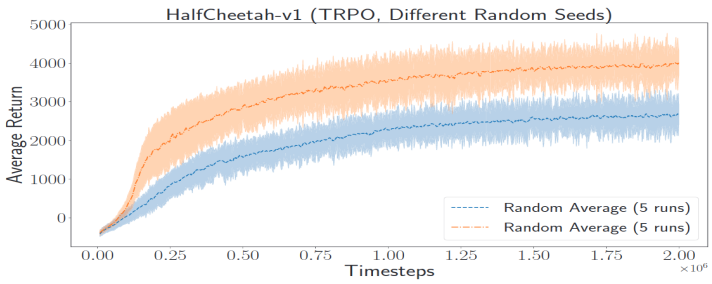

Figure 2: TRPO performance on HalfCheetah-v1 and run inconsistency [11]

can finally apply the statistical test and p-value thresholding for rejecting or not the observed value, using the common Benjamini-Yekutieli [3] method to control the False Discovery Rate (FDR). The statistical test can be summarized as follow:

$$
\begin{aligned}
&S_{MetricRanking}(A, B) = MetricRanking(A) - MetricRanking(B) \\
&H_0 : S_{MetricRanking}(A, B) = 0 \\
&H_1 : S_{MetricRanking}(A, B) \neq 0 \\
&(A', B') \in P(A \cup B) \quad S_{MetricRanking}(A', B').
\end{aligned}
\tag{4}
$$

Here, $MetricRanking$ is the mean of reliability metric ranking across tasks. A and B are sets of performance measurements (e.g average return) for the first and second algorithms. Bootstrap sampling is applied on the permutations of $S_{MetricRanking}(A', B')$ statistics to obtain a distribution and finally apply the statistical test on the observed value.
The authors of this method have been able to found, in their experimental analysis, that reliability is a separate dimension, which needs to be inspected separately from mean or median performance, i.e two algorithms may have similar median performance, but may nonetheless significantly differ in reliability. Additionally, they have proved that reliability along one axis (e.g across time) does not necessarily correlate with reliability on the other two axes (e.g across runs or roll-outs).
These proposed metrics and statistical test have been made available as an open-source library[1] with a few required hyperparameters that have to be reported in the experiments to ensure reproducibility.

3 Experiments

From the methods described in the previous section, different experiments have been carried out by each author in order to prove their results and highlight the issues present in some algorithms. For this reason, we report in this paper some meaningful results to give some insights and usage of the metrics and methods discussed previously.

Inconsistency across runs As mentioned in the first part of the Method section, the confidence interval is the simplest measure that has been considered when using RL models. Although its simplicity, it can be used to tune major hyperparameters of an algorithm and can highlight also the learning progress of the agent, but even some of its pitfalls. An example of its usage can be seen in the following Figure 2 from Henderson et.al work, where we see the instability of the TRPO [15] algorithm on the HalfCheetah-v1 agent of OpenAI Gym environment [4], using the same hyperparameter configuration proposed in the original paper. This result shows the inconsistency of the algorithms, even by averaging the models across two different sets of runs. As we previously see, this issue can be overcome by applying power analysis tests to estimate the required sample size for a consistent learning behaviour and avoid statistically different distributions of the average return for the same algorithm across runs.

[1] https://github.com/google-research/rl-reliability-metrics

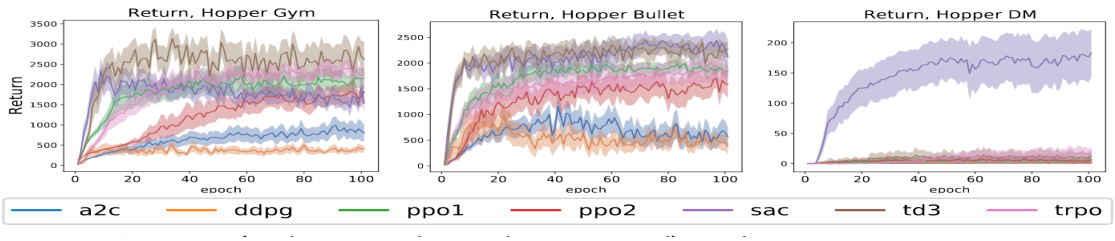

Figure 3: Environment inconsistency for different free-model RL algorithms on Hopper agent [10]

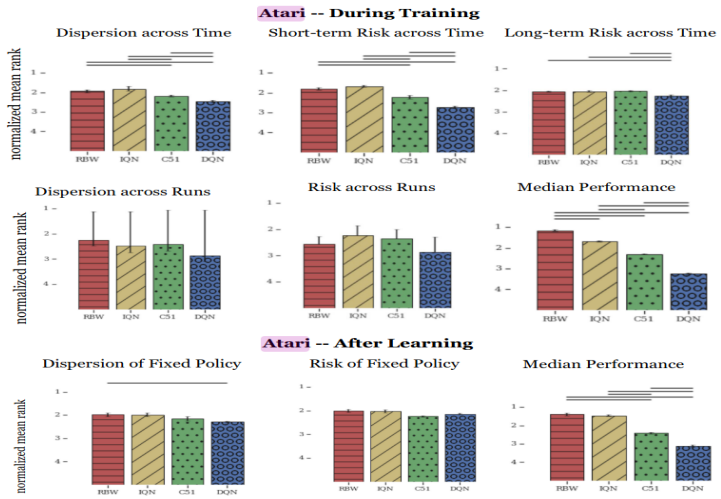

Figure 4: Ranking of reliability metrics on 60 Atari games, using DQN, IQN, C51, Rainbow [5]

Inconsistency on environments Another important type of assessment that is less likely done but should be considered more is to evaluate the consistency of the model in different environments. These type of tests have been studied by Helfenstein in his experimental works [10]. An example of his results can be seen in the following Figure 3. In this experiment, we see the performances of some free RL models for the Hopper agent on three different environments: OpenAI Gym [4], PyBullet [7], and DeepMind [16]. This figure shows the inconsistency of models across multiple environments, indeed for the same task, the algorithms performance can differ significantly depending on the chosen environment, as shown from the rewards and the instability of the algorithms in the plot. The SAC model [9] in all the scenarios has a very fast initial learning behaviour, but in the first environment its performances drop by ending as the third worst final undiscounted return, since it maximizes the immediate short-term reward. Instead, other algorithms take more in consideration the bonus given in the reward for staying alive. In PyBullet instead, SAC stays constantly higher than the others, and in DeepMind it is the only algorithm able to show some learning behaviour. Indeed, DeepMind is the hardest environment comparing all the others, because it does not give bonus on reward by keeping the agent alive.

Reliability In order to prove the separate dimension of reliability from mean and median performances we report an experiment made by Chan et.al using their novel approach based on robust statistics. The experiment involved a comparison of four different variant of Deep Q-Leaning algorithms (DQN, IQN, C51, Rainbow [13, 8, 2, 12]) on 60 Atari games. As shown in the Figure4, even though Rainbow performs significantly better than IQN in Median or Mean Performance, IQN performs numerically or significantly better than Rainbow on many of the reliability metrics. Thus, from this experiment we see that reliability comes as a different dimension. Further evidences and examples of reliability measures and statistical tests can be found in the supplemental material of their work.

4 Conclusion

In this paper, we have discussed different issues and aspects of reproducibility but also evaluation criteria on RL models. We described and summarized systematic evaluation and comparison that should be followed in an experiment. In particular, we have seen novel measures and tests based on robust statistics for evaluating the reliability of models, which is obscured to mean and median metrics. By combining all these methods and assessments, we are able to give more explanatory power to RL models results, in addition, they allow us to understand the strengths and weaknesses of the model, in order to improve existing algorithms, but also for suggesting future works for building robust, and reliable models.

References

[1] Carlo Acerbi and Dirk Tasche. Expected shortfall: A natural coherent alternative to value at risk. *Economic Notes*, 31(2):379–388, 2002.

[2] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017.

[3] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165 – 1188, 2001.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[5] Stephanie C. Y. Chan, Sam Fishman, John F. Canny, Anoop Korattikara Balan, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. *ArXiv*, abs/1912.05663, 2020.

[6] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. 06 2018.

[7] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning, 2017.

[8] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1104–1113. PMLR, 2018.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[10] Felix Helfenstein. Benchmarking deep reinforcement learning algorithms. 2021.

[11] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2017. cite arxiv:1709.06560Comment: Accepted to the Thirthy-Second AAAI Conference On Artificial Intelligence (AAAI), 2018.

[12] 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. 2017. cite arxiv:1710.02298Comment: Under review as a conference paper at AAAI 2018.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

[14] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *CoRR*, abs/2006.16712, 2020.

[15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[16] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

[17] Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4294–4302, Red Hook, NY, USA, 2016. Curran Associates Inc.

[18] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning, 2020.

[19] Ke-Hai Yuan and Kentaro Hayashi. Bootstrap approach to inference and power analysis based on three test statistics for covariance structure models. *British Journal of Mathematical and Statistical Psychology*, 56(1):93–110, 2003.