RLSB

Software Engineering for Artificial Intelligence

Reinforcement Learning Sustainability Benchmark

Report v1.0

Luca Strefezza*

June 30, 2024

*l.strefezza1@studenti.unisa.it

Abstract

This project explores the energy consumption of deep reinforcement learning (DRL) algorithms and their impact on the environment and business costs. Beginning with the development of Deep Q-Networks (DQN) by DeepMind, numerous algorithms have been proposed to enhance the performance of DRL agents. However, the energy implications of these improvements have not been extensively studied. This project aims to fill this gap by benchmarking the energy consumption and performance of several widely used DRL algorithms.

We train various reinforcement learning algorithms on the same task: Atari 100k, a widely recognized benchmark in the DRL community. The selected algorithms include both value-based methods (such as DQN and RAINBOW) and policy gradient methods (such as REINFORCE with baseline and PPO). Each algorithm is evaluated on 8 different Atari games, with 4 runs per game using different random seeds, to ensure statistically significant results.

To track performance and energy consumption, we utilize Weights and Biases, TensorBoard, and CodeCarbon. The development environment is based on CleanRL, a library that provides single-file implementations of some DRL algorithms, ensuring consistency and reproducibility.

The preliminary version of this report covers the context, goals, and methodological steps of the project. Future sections will present detailed results and analyses of the trade-offs between performance and energy consumption, contributing to a broader understanding of the sustainability implications of DRL technologies.

Contents

1 Context

This project addresses the energy consumption of deep reinforcement learning (DRL) solutions and their impact on the environment and business costs.

Beginning with the resurgence of the field following the development of *Deep Q-Networks* (DQN) by DeepMind in the early 2010s [1], there have been a number of algorithm proposals over time that with minor modifications to DQN or using a completely different paradigm (such as policy gradient methods) sought to improve the performance achieved by the learning agent.

Although the performances of the various solutions have been extensively studied and tracked, little effort has been directed toward understanding how the tweaks to the DQN introduced to improve performance impacted energy consumption, or what the cost of the alternative approaches developed was, per se and in comparison with previous solutions.

The motivation behind this project is to fill this gap by evaluating the trade-offs between performance and energy consumption for several widely used deep reinforcement learning (DRL) algorithms. Understanding these trade-offs is crucial for businesses and researchers who aim to optimize both performance and sustainability in their applications. This project aims to provide valuable insights into the energy efficiency of different DRL approaches, enabling informed decisions about their use in various contexts.

To reach this goal we train various reinforcement learning algorithms on the same task, the choice of which is discussed in section 3.2 on page 6. Section 3.1 on page 5 describes the selected algorithms, whose choice was made taking into account that DRL algorithms can be devided in two main categories: *value based* (i.e. algorithms based on the approximation of a value function, be it the state-value function or the action-value function) and *policy gradient*. The latter are methods that approximate directly the policy,

and includes as a special case the *actor-critic methods*, which approximate simultaneously a policy (said actor) and a value function (said critic).

2 Goals

The primary goal of this project is to benchmark the energy consumption and performance of various deep reinforcement learning algorithms. Specifically, we aim to:

1. evaluate the energy consumption of different DRL algorithms when trained on the same task;
2. compare the performance of these algorithms in terms of their ability to achieve high scores on the given task;
3. analyze the trade-offs between performance and energy consumption to identify the most efficient algorithms;
4. provide a comprehensive report that can guide practitioners in selecting the appropriate DRL algorithms based on specific use-case requirements.

By achieving these goals, the project will contribute to the broader understanding of the sustainability implications of deep reinforcement learning technologies.

3 Methodological Steps

The methodology used follows from the basic idea of this benchmark: to execute all the algorithms for the same number of environment interactions, so that we can compare the score they achieve and the energy consumption of each one of them. Additionally, a good comparison would be to take the score obtained by the lowest performer in this initial trial and re-train all the algorithms until they reach that score. This would allow us to compare how much time and energy each algorithm requires to achieve the same performance level. Unfortunately, time and resources constraints make retraining all algorithms unfeasible, so we will approximate this second comparison by using the returns from the logging of the training during the first trial. This logging includes the `global_step`, indicating the environment interaction we are at, and the `episodic_return`, which is the return of the episode (i.e., the score on which to compare), as well as all performance and power consumption data up to that point. By analyzing these logs, we will estimate how much time and energy each algorithm would take to reach the score obtained by the lowest performer in the initial trial.

The following sections outline the several key steps involved in the methodology adopted for this project.

3.1 Algorithms Selection

As stated, in our benchmark we consider both value-based methods and policy gradient methods. The selected algorithms are chosen to represent a wide range of approaches within both categories.

3.1.1 Value-Based Methods

Value-based methods are algorithms based on the approximation of a value function. The algorithms we will test in this category are:

- *Deep Q-Network (DQN)*: the first example of success in deep reinforcement learning, will serve as a sort of baseline for our benchmark.
- *RAINBOW* [2]: an advanced method that combines several improvements to the original DQN, that will also be tested individually to assess their individual contributions to energy consumption and performance. These are listed hereafter:
  - Double Q-Learning (Double DQN) [3];
  - Prioritized Experience Replay [4];
  - Dueling Network Architectures [5];
  - Multi-step Learning [6];
  - Distributional RL [7];
  - Noisy Nets [8];
- *Self-Predictive Representations (SPR)* [9]: a more advanced method introduced in recent research, which leverages self-predictive representations to improve efficiency.

3.1.2 Policy Gradient Methods

Policy gradient methods approximate the policy directly and include as a special case the actor-critic methods, which simultaneously approximate a policy and a value function. The algorithms we will test in this category are:

- *REINFORCE* [10, Chapter 13]: a basic policy gradient method, or its variant REINFORCE with baseline (also known as Vanilla Policy Gradient, VPG).
- *Proximal Policy Optimization (PPO)* [11]: a popular and efficient policy gradient method that uses a clipped objective to improve training stability.
- *Deep Deterministic Policy Gradient (DDPG)* [12]: an algorithm that combines policy gradients with deterministic policy updates for continuous action spaces.
- *Twin Delayed DDPG (TD3)* [13]: an improvement over DDPG that addresses function approximation errors through various techniques, such as delayed policy updates and target policy smoothing.

- *Data-Regularized Q (DRQ)* [14]: a method that incorporates data augmentation to regularize the training of Q functions, improving performance and stability.

3.2 Task Selection

Regarding the task on which to compare the algorithms, there were several suitable candidates: Atari 100k [15], one of the continuous control task of the DeepMind Control Suite, or one of the many other task (besides Atari) included in OpenAI Gymnasium (formerly Gym), and so on. After various tests and research we opted for the Atari 100k benchmark, a discrete task that consists of playing selected Atari games for only 100 000 environment interactions.

The reason for this choice is multifaceted. Atari 100k is a widely used benchmark in the DRL community, the wealth of prior research and baseline results available facilitates a more straightforward validation and comparison of our experimental results with those from other studies and algorithms. It is also well suited for evaluating the performance of almost all popular DRL algorithms, ensuring a comprehensive assessment. Additionally, Atari games provide a range of different challenges, including planning, reaction time, and strategy, making it a robust benchmark for assessing general DRL capabilities.

Moreover, the discrete nature of Atari 100k simplifies the implementation and comparison of algorithms, as continuous control tasks often require additional considerations and modifications. Finally, the 100 000 interactions limit strikes a balance between providing enough data for meaningful evaluation and being computationally feasible within our resource constraints, especially considering the large number of experiments required for each algorithm, as detailed in section 3.3.1.

These factors combined make Atari 100k a practical and effective choice for our benchmark, enabling us to achieve our project goals efficiently.

3.3 Experiment Setup

In this section we will address all the decisions made in the setup of the experiments.

3.3.1 Number of Runs

In determining how many runs to carry out during the experimentation and testing of a reinforcement learning algorithm, at least two fundamental aspects must be taken into account: the high variance of reinforcement learning, and thus its high susceptibility to randomness, and the evaluation of the generality of the algorithm, which must therefore be tested in several different environments in order to actually prove that it is capable of solving multiple problems and not just ultra-specialized on a single use-case.

In addressing the first aspect we can refer to the literature to get an idea of how many runs with different seeds are usually performed to alleviate this problem. If in the early days of RL (and not DRL) the number of runs stood at around 100 and in any case did not fall below 30, at least until the introduction of ALE (Arcade Learning Environment) [16] included, with the advent of DRL the number of runs was consistently

reduced to 5 or less because of the high cost in terms of time and resources per run. Although this has been the standard for years, a more recent work [17] has shown that this is the source of a problem. Practitioners use point estimates such as mean and median to aggregate performances scores across tasks to summarize the results of the various runs, but this metrics are not the best way to do so because they ignore the statistical uncertainty inherent in performing only a few runs.

In particular, the study points out that in the case of Atari at least 100 runs per environment are required to obtain robust results, a value that is, however, impractical in reality. In our case we will be forced to limit ourselves to 4 runs per environment, but it should be noted that this is a less significant problem for us, since we are not attempting to advance the state of the art performance of DRL algorithms, but have instead a focus on energy consumption, which should in any case remain constant regardless of the actual learning of the agent, which is instead related to randomness.

Despite this, we will still attempt to use, in addition to the more classic and popular metrics such as the point estimates mentioned above, other metrics suggested in [17] (like the *interquartile mean*), designed precisely to obtain more efficient and robust estimates and have small uncertainty even with a handful of runs, since they are not overly affected by outliers like the point estimates.

With regard to the second aspect, namely, testing the algorithms on a variety of environments to evaluate their generality, Atari 100k once again comes to our aid, being constituted by 26 games. Moreover, the Arcade Learning Environment, built on top of the Atari 2600 emulator Stella and used by gymnasium, includes over 55 games. Unfortunately, again, we do not have the time and/or computational resources to test on all the Atari 100k's 26 games or all the ones available in ALE, so we selected for the benchmark a representative subset of 8 Atari games, trying to choose games that cover a range of difficulties and styles. Obviously, with so few games because of the constraints just mentioned, an exhaustive selection is difficult, but we nonetheless tried to provide a balanced benchmark, ensuring that the selected games cover a range of challenges to effectively evaluate different algorithms, while still not being excessively difficult. This last requirement is due to basic DQN and its more simple extensions, which have some limitations in only 100k interactions (the team that introduced the DQNs trained its model on 2 million interactions to achieve interesting results).

Here are the 8 selected games, completed with a rationale for their inclusion:

- *Alien* - moderate difficulty, good for testing exploration and strategy;
- *Amidar* - requires planning and quick decision-making;
- *Assault* - tests reflexes and targeting accuracy;
- *Boxing* - simple but requires precise control and timing;
- *Breakout* - classic game, good for testing control;
- *Freeway* - simple yet tests quick decision-making under pressure;

- *Ms. Pac-Man* - classic maze game, tests navigation and evasion;
- *Pong* - simple and well-understood, great for baseline comparisons.

So, to summarize, each algorithm will be evaluated on 8 different Atari games, with 4 runs per game using different random seeds, for a total of 32 trainings per algorithm. This approach, with appropriate metrics, ensures that our results are statistically significant and account for the inherent variability in RL training processes.

3.3.2 Data Collection

Collecting comprehensive and accurate data is crucial for evaluating both the performance and energy consumption of the algorithms. We employ several tools and services to ensure robust data collection and analysis.

To track the performance metrics, we use both online and local tools. The online service *Weights and Biases* is used for real-time monitoring and storage of experimental data. This platform allows for easy sharing and collaboration, as well as providing powerful visualization and analysis tools. Locally, we use *TensorBoard*, which integrates seamlessly with our training workflows and offers detailed insights into the training process through its rich set of visualizations.

In addition to tracking performance metrics, monitoring energy consumption is the key aspect of the project. For this we use *CodeCarbon*, a tool designed to measure the carbon footprint of computing activities. CodeCarbon is integrated in our training scripts to provide real-time tracking of energy usage, a crucial metric for comparing the energy efficiency of the different algorithms.

The metrics we collect include:

- *Global Step:* indicates the number of environment interactions during training.
- *Episodic Return:* the score achieved in each episode, providing a measure of the algorithm's performance.
- *Loss:* tracks the optimization process, giving insight into the learning dynamics of the algorithm.
- *Value Estimates:* such as Q-values or value function estimates, offering insight into the agent's decision-making process.
- *Policy Entropy:* measures the randomness in the policy and how much it differs from the previous one, useful for understanding exploration behavior and how much room for improvement is still left.
- *Learning Rate:* the rate at which the model learns, especially if it changes during training.
- *Energy Consumption:* the amount of energy used during training, tracked by CodeCarbon, allowing us to evaluate the energy efficiency of each algorithm.

Weights and Biases facilitates the aggregation and visualization of these metrics across multiple runs and environments, making it easier to compare results and draw meaningful conclusions. TensorBoard provide supplementary local visualizations to help diagnose any issues during training and ensure the integrity of the collected data.

By using these tools in tandem, we aim to collect a comprehensive dataset that covers both the performance and energy consumption aspects of the algorithms, ensuring a thorough evaluation aligned with the goals of our project.

3.3.3 Development and Execution Environment

The development and execution environment for the project involves both hardware and software. The hardware used for training the algorithms is constituted by the processor *11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz*, the graphic card *NVIDIA GeForce GTX 1050 Ti*, and *16GB* of RAM.

On the software side, after careful considerations and some testing with other alternatives like OpenAI's *Spinning Up*, we chose to base the implementation of the project on *CleanRL* [18]. As the authors states, CleanRL is an open-source library that provides high-quality single-file implementations of Deep Reinforcement Learning algorithms. It provides an environment already complete with most dependencies a project like ours might need (like Gymnasium), has a straightforward codebase, and already integrates tools like Weights and Biases and TensorBoard, that help log metrics, hyperparameters, videos of an agent's gameplay, dependencies, and more.

The single-file implementation philosophy of CleanRL aims to make reinforcement learning research more accessible and reproducible and make the performance-relevant details easier to recognize. By consolidating every algorithm codebase into single files, it simplifies the understanding and modification of algorithms, which is particularly beneficial for both educational purposes and rapid prototyping, even though it comes at the cost of losing modularity and duplicating some code.

We leverage CleanRL's existing implementations where available, tweaking them to meet the specific requirements of our benchmarks. When an implementation for a particular algorithm is not available, we develop it from scratch, trying to adhere to CleanRL's philosophy and implementation principles. This approach ensures consistency and comparability across all tested algorithms.

In the end, the environment for our experiments should be efficient and easily reproducible, facilitating the accurate evaluation of both performance and energy consumption of various deep reinforcement learning algorithms.

References

[1] V. Mnih *et al.*, *Playing atari with deep reinforcement learning*, 2013. arXiv: 1312.5602 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1312.5602.

[2] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[3] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv preprint arXiv:1509.06461*, 2016.

[4] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[5] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[6] J. Peng and R. J. Williams, "Incremental multi-step q-learning," in *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 226–232.

[7] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," *arXiv preprint arXiv:1707.06887*, 2017.

[8] M. Fortunato *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[9] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," *arXiv preprint arXiv:2007.05929*, 2020.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[12] T. P. Lillicrap *et al.*, *Continuous control with deep reinforcement learning*, 2019. arXiv: 1509.02971 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1509.02971.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, PMLR, 2018, pp. 1587–1596.

[14] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv preprint arXiv:2004.13649*, 2020.

[15] L. Kaiser *et al.*, *Model-based reinforcement learning for atari*, 2024. arXiv: 1903.00374 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1903.00374.

[16] 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*, vol. 47, pp. 253–279, Jun. 2013, ISSN: 1076-9757. DOI: 10.1613/jair.3912. [Online]. Available: http://dx.doi.org/10.1613/jair.3912.

[17] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, *Deep reinforcement learning at the edge of the statistical precipice*, 2022. arXiv: 2108.13264 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2108.13264.

[18] S. Huang *et al.*, *Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms*, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1342.html.