Hyperparameter Optimization for Deep Reinforcement Learning: An Atari Breakout Case Study

Ken Zheng

ABSTRACT

In reinforcement learning (RL), a subfield of machine learning, we train systems to perform complex tasks through trial and error. In RL, an agent interacts with an environment, taking actions that generate a cumulative reward. If a series of actions generates a high cumulative reward, those actions will be favorable in the future. Some applications of RL include improving the performance of self-driving cars, improving the performance of large language models, and playing games like Go. While playing games might not have a direct impact on the real world, systems like AlphaGo have helped improve our understanding of RL that aids in more real-world applications. This study uses game completion as a test bed to better understand the underlying mechanisms behind RL, specifically the effects of tuning hyperparameters on a model's performance. A Deep Q-Learning (DQN) model architecture was chosen for this analysis, and we tuned batch size, learning rate, exploration rate, and discount factor. We hypothesized optimizing these hyperparameters would increase the cumulative reward. These hyperparameters were tuned to maximize the score in the game of Atari Breakout. We found that altering the discount factor to be greater than or less than one results in a much less effective model, whereas tuning hyperparameters that were changed caused little change to the performance. The results of this study can be used to improve the performance of future RL models. All code to reproduce results in this study is available at: https://github.com/BobyWoby/Reinforcement-Learning.

INTRODUCTION

Machine Learning (ML) is a part of computer science that uses data and algorithms to make predictions on data (1). There are 3 subcategories of machine learning: supervised,

unsupervised, and reinforcement (2). Supervised machine learning models are trained on labeled data sets (1), Unsupervised models look for patterns in unlabeled datasets, and reinforcement models improve their accuracy through trial and error (1). Reinforcement learning (RL) mimics the way that humans learn through trial and error, and is broadly separated into 2 categories, model-based, and model-free algorithms (3). Model based methods utilize neural networks to understand the rules of the environment they're in, while model-free methods will just try different actions and remember which ones returned good results (4). In this study, we focused on model-based RL, which used Deep Q-Learning (DQN) to predict the reward for each action taken by an agent. Following a sequence of actions taken in the training environment, the agent will receive a cumulative reward (4). A deep neural network approximates a policy function for the agent's actions. This policy function is trained by the agent's actions to maximize the cumulative reward. DQN is useful for many different applications, especially in real-world applications, where the final reward isn't immediately available, such as swarm system control (5), and self-driving cars (6).

The DQN algorithm leverages a neural network to try and predict the reward of an action, given the current state. This neural network (policy network) is trained to yield the highest cumulative reward. The reward is expressed by the Bellman Equation:

$$Q(s, a) = r + \gamma \, max_a Q(s', a),$$

Where $Q(s, a)$ is the true (unknown) final cumulative reward, s is the current state, a is an action, is the current cumulative reward, γ is the discount factor, and s' is the next state.

The Bellman Equation allows for a long-term reward to be calculated from the current reward and a discounted value of the expected future reward, which can be predicted using the neural network (7). Within these RL systems, there are external configuration variables that control the training process called hyperparameters, with γ being one such example of a hyperparameter. These hyperparameters affect how the system gets trained. Examples include the learning rate of the neural network and the frequency in which the system leverages the neural network (epsilon). While these variables aren't the same as the biases and weights that are within the neural network, they still affect the performance of the RL algorithm.

The process of hyperparameter optimization is a crucial component in refining machine learning models, particularly in RL (8). It involves systematically exploring and adjusting these settings to find the optimal combination that enhances the performance of the RL algorithm. To accomplish this task, many use automated hyper-parameter optimization (HPO). HPO automatically optimizes the hyperparameters of a model to remove humans from the loop (9).

The main objective of this analysis was to determine the extent to which tuned hyperparameters would improve an RL algorithm's performance as opposed to literature-based values (7). The hyperparameters chosen for this study were batch size, the starting step for epsilon decay, the starting epsilon, the learning rate, and .

RESULTS

Using a Gymnasium-based Atari Breakout game environment (9), we aimed to optimize the performance of our DQN model through hyperparameter tuning. The model trained with an 84x84 pixel image of the game, cropped to only show the play area to decrease computation time and to focus the model on the relevant portions of the input data.

The preprocessed data was fed in batches to a DQN based RL model to reinforce long-term rewards, as it is trained based on the cumulative reward of the batch, rather than a single action and instantaneous reward. Using the input data, the model was trained to predict the cumulative reward of an action, and the algorithm picked the action that maximized this predicted cumulative reward according to the Bellman equation.

To optimize the hyperparameters, we utilized Optuna, a powerful hyperparameter optimization framework (11), for 100 trials. The hyperparameters tuned were γ (the discount factor), learning rate, starting epsilon (the initial exploration rate), and the step at which epsilon would start to decay. Optuna's objective was to identify the set of hyperparameters that maximized the final score, and the best performing set of hyperparameters were then compared to values found in the literature (7). Note that Optuna's objective of the final score differs from the RL algorithm's cumulative reward.

Figure 1: Reward vs Episodes Graph of the 2 Hyperparameter Sets. The plot of Mnih et al. illustrates the performance of the literature values during the model's training (left), while the plot to the right shows the performance of the tuned hyperparameters during training. The orange line shows the moving average performance of training over the last 100 episodes. The dashed line represents the highest average score for each of the training processes. The reward represents the in-game score that the model received on the corresponding episode. Below each reward vs episodes graph lies the final frame of a representative episode from the trained models, visualizing the improved performance of a tuned RL algorithm.

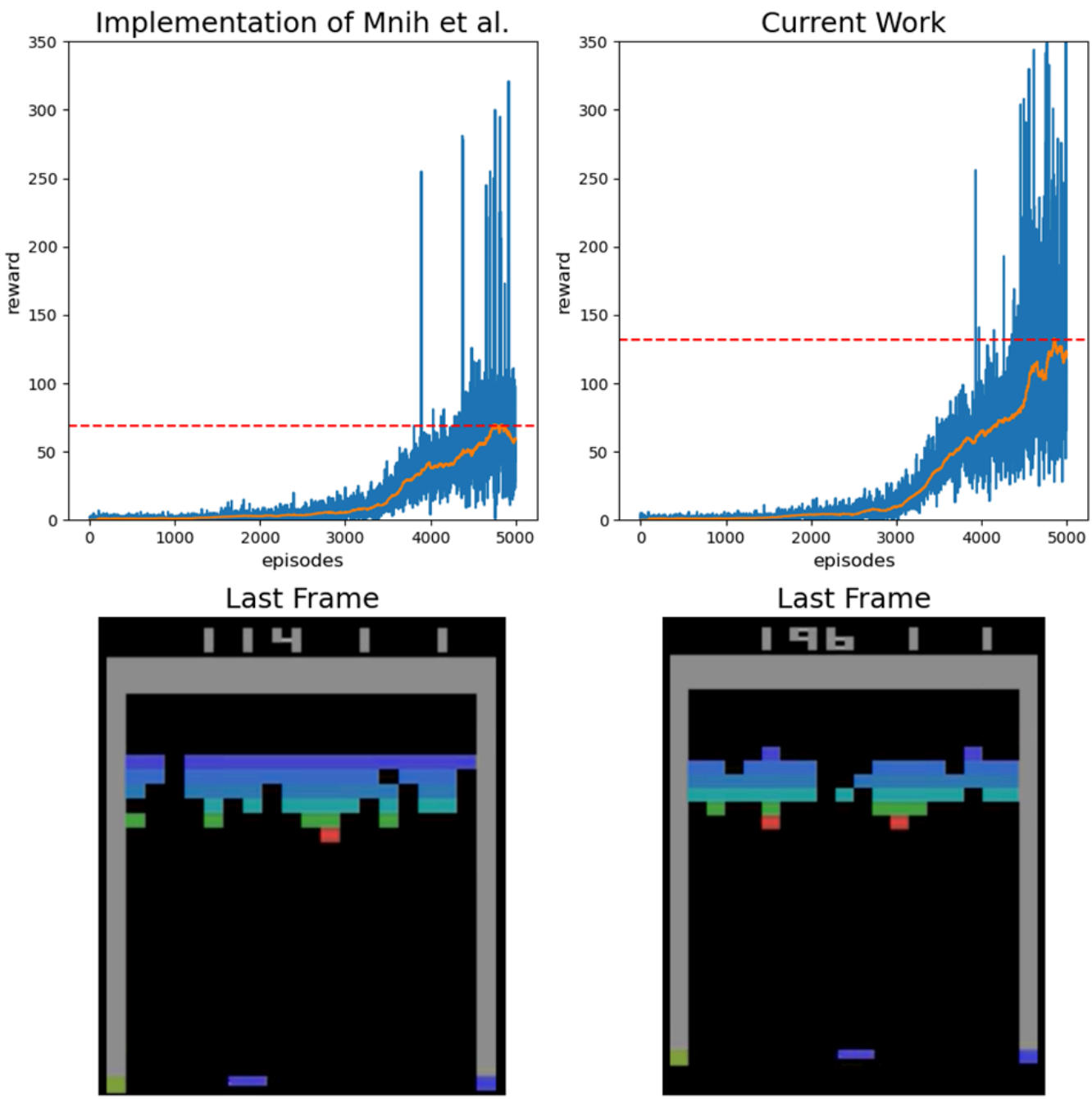

Figure 1: Reward vs Episodes Graph of the 2 Hyperparameter Sets. The plot of Mnih et al. illustrates the performance of the literature values during the model's training (left), while the plot to the right shows the performance of the tuned hyperparameters during training. The orange line shows the moving average performance of training over the last 100 episodes. The dashed line represents the highest average score for each of the training processes. The reward represents the in-game score that the model received on the corresponding episode. Below each reward vs episodes graph lies the final frame of a representative episode from the trained models, visualizing the improved performance of a tuned RL algorithm.

After running for 5000 episodes each, the literature based hyperparameters (7) achieved a final moving average of 96.99, that is, for the last 100 episodes of training, the machine gained an average score of 96.99 per game (Figure 1). The tuned machine performed significantly better, achieving a final moving average of 132.51 (Figure 1).

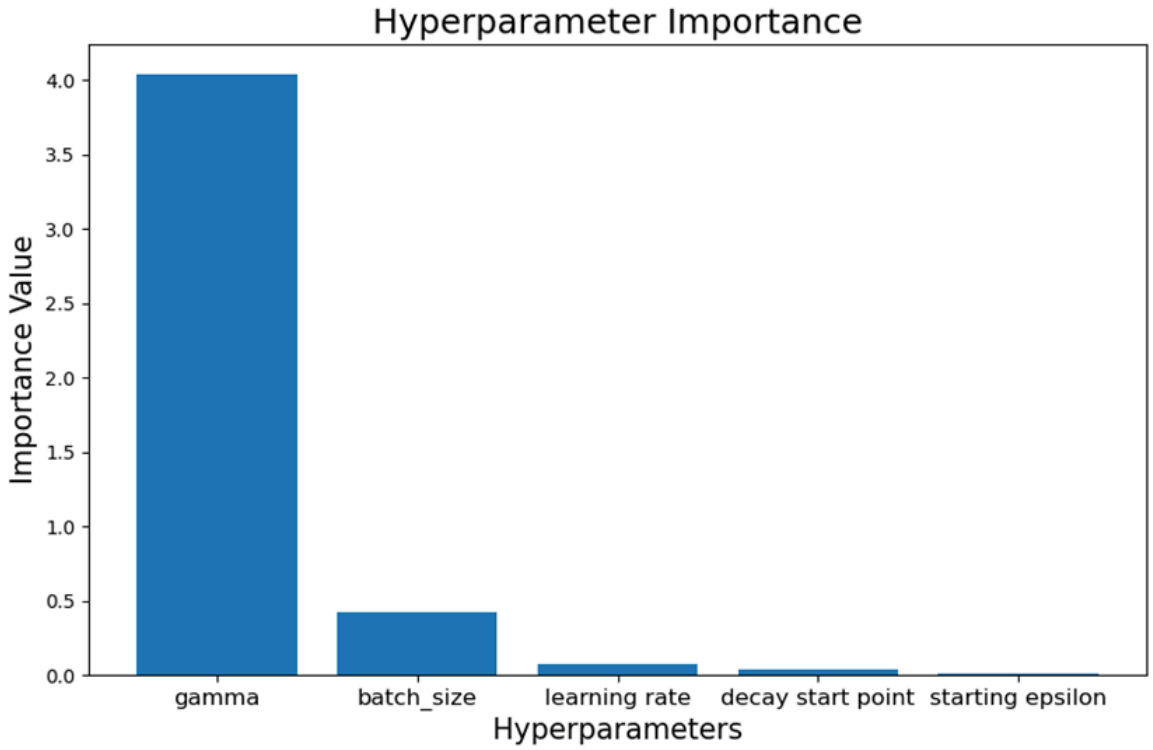

Figure 2: Hyperparameter Importance Using the fANOVA Algorithm. The Y-axis shows the calculated fANOVA importance of the hyperparameter and the X-axis shows the hyperparameter list. The figure shows that the most important hyperparameter by far is gamma.

The tuning process revealed that the hyperparameter with the largest effect on the model's performance was the γ value, with a hyperparameter importance of 0.42 (Figure 2). We observe in the data that γ has to be below 1 to achieve optimal results, which is in line with other work (8). This scalar value, which discounts future rewards to decide how much importance is given to the cumulative reward compared to the instantaneous reward, plays a crucial role in the model's ability to prioritize long-term gains effectively. The other parameters, starting epsilon, learning rate, and the epsilon decay start point, had much lower hyperparameter importance values, 0.04, 0.07, 0.01, and 0.04 respectively (Figure 2).

DISCUSSION

As shown by our experiment, there is a substantial impact of hyperparameter tuning on the effectiveness of RL models, which agrees with other papers(13). Our study reveals that investing time and resources into optimizing hyperparameters leads to considerable improvements in model performance and learning efficiency, enhancing the practicality of RL systems. This advancement is particularly crucial for high-stakes applications such as autonomous driving, financial forecasting, and natural language processing (14-15). For instance, in autonomous driving, improved RL models can lead to more accurate and quicker decision-making, directly enhancing vehicle safety and operational efficiency (16). In finance, optimized RL models can refine predictive accuracy, potentially improving investment strategies and risk management (15). Similarly, for large language models, better-tuned RL algorithms contribute to faster and more accurate language processing, enhanced user interactions and application performance (17).

Our findings underscore the practical value of hyperparameter tuning in making RL systems more viable and effective for real-world applications. But it is important to note that while our results support our hypothesis that hyperparameter tuning enhances RL model performance, they do not provide a definitive explanation of the underlying mechanisms, and further research is needed to explore the underlying mechanisms by which hyperparameter tuning affects learning dynamics and to provide deeper insights into the optimization process and its broader implications for RL applications.

Several factors and limitations could have influenced our result. One major factor is the choice of the RL environment used in our experiment. Different environments have varied levels of complexity, which may result in different sensitivities to hyperparameter tuning. While our results might be robust for our chosen environment, different environments may see differing effects of hyperparameter tuning. Another possible limitation is the way in which our experiment measured performance. While the average reward is a commonly used metric, incorporating additional measures such as convergence speed and sample efficiency may provide a more holistic evaluation of the model's performance.

Several scientific questions remain unanswered. For example, in our experiment, we mostly focused on hyperparameters external to the actual neural network, such as the batch size and gamma values, but one important question for future research would be how would tuning the hyperparameters internal to the network, such as the number of hidden layers and the number of nodes within each layer. Another area for future research is investigating hyperparameter tuning across different RL algorithms and environments. This could help establish more generalized guidelines for hyperparameter tuning that can be applied across various settings. Furthermore, looking at the impact of hyperparameter tuning on long term stability and adaptability could provide valuable insights for real world applications.

MATERIALS AND METHODS

A Gymnasium environment (11) playing Atari Breakout was created. The original 210x160 RGB image was cropped to an 84x84 pixel image around the game area in grayscale. Each transition between frames, consisting of a state, an action, the next game state, and the individual reward, was stored in a memory replay buffer. The input to the DQN network grabbed a random sample of the batch size from this memory buffer to feed into the neural network. PyTorch (18) was used as the machine learning framework for this experiment, and the neural networks used in this experiment utilized convolutional neural networks (CNNs). To explore as many different strategies as possible, randomness was added to the process. The policy neural network was used a random percentage of the time. The percentage decreased as the training process continued, related to epsilon in this experiment. After predicting the value of the next state, the machine would try to optimize the DQN network using the Bellman equation. The optimizer used was Adam (19) with a loss function of Smooth L1 Loss (20). To tune the various hyperparameters for the experiment, an external library called Optuna (12) was chosen with the fANOVA hyperparameter importance algorithm (21), running for 100 trials, with the number of episodes for each trial being set to 2500. The implementation used for this paper can be found at https://github.com/BobyWoby/Reinforcement-Learning.git.

REFERENCES

1. Udousoro, I. C. “Machine Learning: A Review”. *Semiconductor Science and Information Devices*, vol. 2, no. 2, Oct. 2020, pp. 5-14, doi:10.30564/ssid.v2i2.1931.

2. Li, Yuxi. *Deep Reinforcement Learning: An Overview*. arXiv:1701.07274, arXiv, 25 Nov. 2018. *arXiv.org*, https://doi.org/10.48550/arXiv.1701.07274.

3. Kaelbling, L. P., et al. “Reinforcement Learning: A Survey.” *Journal of Artificial Intelligence Research*, vol. 4, May 1996, pp. 237–85. *www.jair.org*, https://doi.org/10.1613/jair.301.

4. Shakya, Ashish Kumar, et al. “Reinforcement Learning Algorithms: A Brief Survey.” *Expert Systems with Applications*, vol. 231, Nov. 2023, p. 120495. *ScienceDirect*, https://doi.org/10.1016/j.eswa.2023.120495.

5. Hüttenrauch, Maximilian, et al. “Deep Reinforcement Learning for Swarm Systems.” *Journal of Machine Learning Research*, vol 20, 19 Feb. 2019, p. 1, https://doi.org/10.48550/arXiv.1807.06613

6. Wang, Letian, et al. *Efficient Reinforcement Learning for Autonomous Driving with Parameterized Skills and Priors*. arXiv:2305.04412, arXiv, 7 May 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2305.04412.

7. Mnih, Volodymyr, et al. *Playing Atari with Deep Reinforcement Learning*. arXiv:1312.5602, arXiv, 19 Dec. 2013. *arXiv.org*, https://doi.org/10.48550/arXiv.1312.5602

8. Adaptive Discount Factor for Deep Reinforcement Learning in Continuing Tasks with Uncertainty - PMC. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9570626/. Accessed 10 Sept. 2024.

9. Kiran, Mariam, and Melis Ozyildirim. *Hyperparameter Tuning for Deep Reinforcement Learning Applications*. arXiv:2201.11182, arXiv, 26 Jan. 2022. *arXiv.org*, https://doi.org/10.48550/arXiv.2201.11182.

10. Yu, Tong, Zhu, Hong. *Hyper-Parameter Optimization: A Review of Algorithms and Applications.* arXiv:2003.05689 **,** arXiv, 12 Mar. 2020. *arXiv.org,* https://doi.org/10.48550/arXiv.2003.05689

11. Brockman, Greg, et al. *OpenAI Gym*. arXiv:1606.01540, arXiv, 5 Jun 2016. *arXiv.org*,

https://doi.org/10.48550/arXiv.1606.01540

12. Akiba, Takuya, et al. *Optuna: A Next-generation Hyperparameter Optimization Framework,* arXiv. 25 July 2019. *arXiv.org,* https://doi.org/10.48550/arXiv.1907.10902

13. Leeney, William, and Ryan McConville. *Uncertainty in GNN Learning Evaluations: The Importance of a Consistent Benchmark for Community Detection*. arXiv:2305.06026, arXiv, 25 Nov. 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2305.06026.

14. Li, Yuxi. *Reinforcement Learning Applications*. arXiv:1908.06973, arXiv, 19 Aug. 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1908.06973.

15. Charpentier, Arthur, et al. “Reinforcement Learning in Economics and Finance.” *Computational Economics*, vol. 62, no. 1, June 2023, pp. 425–62. *Springer Link*, https://doi.org/10.1007/s10614-021-10119-4.

16. Folkers, Andreas, et al. “Controlling an Autonomous Vehicle with Deep Reinforcement Learning.” *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2025–31. *IEEE Xplore*, https://doi.org/10.1109/IVS.2019.8814124.

17. Du, Yuqing, et al. “Guiding Pretraining in Reinforcement Learning with Large Language Models.” *Proceedings of the 40th International Conference on Machine Learning*, PMLR, 2023, pp. 8657–77. *proceedings.mlr.press*, https://proceedings.mlr.press/v202/du23f.html.

18. Paszke, Adam, et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. arXiv:1912.01703, arXiv, 3 Dec. 2019. *arXiv.org, https://doi.org/10.48550/arXiv.1912.01703*

19. Kingma, Diederik P., and Jimmy Ba. *“Adam: A Method for Stochastic Optimization.” arXiv.Org, 22 Dec. 2014, https://arxiv.org/abs/1412.6980v9.*

20. Girshick, Ross. *“Fast R-CNN.” arXiv.Org, 30 Apr. 2015, https://arxiv.org/abs/1504.08083v2.*

21. Hu, Linwei, et al. *Interpretable Machine Learning based on Functional ANOVA Framework: Algorithms and Comparisons*. arXiv:2305.15670, arXiv:2305.15670, arXiv, 25 May. 2023. *arXiv.org,* https://doi.org/10.48550/arXiv.2305.15670