A Comparison of DQN and Dueling DQN in A Super Mario Environment

Xiangwei Fu

Department of Computer Science, University of Liverpool, Liverpool, L697ZX, United Kingdom
sgxfu5@liverpool.ac.uk

Abstract. Reinforcement learning (RL) has made significant advancements in training artificial agents to play video games. Among various RL algorithms, the deep Q-network (DQN) based on Q-learning has shown outstanding performance in this domain. Recently, dueling DQN, as an enhancement to the standard DQN, has garnered significant attention in the research community. However, there remains a need for a comprehensive and detailed comparison between DQN and dueling DQN, specifically in the context of the Super Mario game environment, as well as an examination of their performance differences. This article aims to investigate and compare the advantages and disadvantages of DQN and dueling DQN in the Super Mario game. It seeks to explore the potential reasons underlying the observed differences in their respective performances. The evaluation is conducted over 3000 epochs, during which the final scores achieved by dueling DQN are observed to be slightly higher than those achieved by DQN. By conducting a rigorous and systematic analysis, this research is conducted for improving the the understanding of the nuances and performance disparities between DQN and dueling DQN in the specific context of the Super Mario game. The obtained results will shed light on the potential benefits and drawbacks of each algorithm, providing insights for further advancements and improvements in RL-based gaming agents.

Keywords: Reinforcement Learning, Deep Q-network, Super Mario Game.

1 Introduction

Reinforcement learning (RL) has emerged as a prominent and active research area, distinguished from traditional supervised and unsupervised learning paradigms. In contrast to these methods, RL places a primary focus on the dynamic interaction between an environment and its corresponding agents. By actively exploring the environment, RL agents continually acquire feedback in the form of rewards, which serve as signals to refine their decision-making processes and optimize their actions in order to maximize cumulative rewards. One of the core characteristics of RL is the incorporation of sequential decision-making within a Markov Decision Process framework. RL agents operate in an environment with partially observable states and must make a series of actions over time to accomplish long-term missions. The

B. H. Ahmad (ed.), *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, Advances in Intelligent Systems Research 180,
https://doi.org/10.2991/978-94-6463-370-2_25

temporal dimensionality introduces challenges related to the trade-off between short-term rewards and long-term objectives, necessitating the use of sophisticated algorithms to balance exploration and exploitation strategies [1].

The distinctive feature of RL is that it is adaptive, with agents continuously learning from their experiences to enhance performance. RL agents attempt to estimate the ideal policy or value function through the use of a variety of learning techniques, such as value-based methodologies or policy gradients, while relying on the environment's feedback to inform their choices. With high uncertainty and randomness, RL can handle complicated issues thanks to its iterative learning process.

RL techniques have been thoroughly studied in a variety of domains, such as operational research, simulation-based optimization, evolutionary computing, and multi-agent systems, including games [1].

Playing video games is the most simple and affordable method of observing agent performance among these applications. As a result, this article decides to use gaming to test the Deep Q network (DQN) and competing DQN algorithms. The gaming environment is based on a set of RL research tools called OpenAI Gym. It has a website where people may share their findings and assess the efficacy of algorithms, as well as a growing database of benchmark challenges that highlight a common user interface [2].

This library encompasses a robust package that integrates reward strategies and state representations, thus serving as a suitable framework for RL training. Within the OpenAI Gym library, the Super Mario environment was specifically chosen as a testbed due to its characteristic of exhibiting linear state changes over time. Although the presence of traps within the game is deterministically governed rather than stochastic, their abundance poses a formidable challenge for agents, necessitating multiple training iterations to attain high scores. In this context, the dynamic manifestation of traps as the agent traverses the environment provides a distinct opportunity to discern the contrasting capabilities of the dueling DQN and DQN algorithms [3,4]. In the context of experimental design, this study adopts both DQN and dueling DQN methodologies to train on identical Super Mario game levels over a duration of 3000 iterations. The ultimate criterion for evaluating the comparative efficacy of these two approaches rests upon the obtained final scores. Subsequently, it is discerned that dueling DQN exhibits a marginal performance advantage over DQN within this experimental setting.

2 Related Work

The distinctive feature of RL is that it is adaptive, with agents continuously learning from their experiences to enhance performance. RL agents attempt to estimate the ideal policy or value function through the use of a variety of learning techniques, such as value-based methodologies or policy gradients, while relying on the environment's feedback to inform their choices. With high uncertainty and randomness, RL can handle complicated issues thanks to its iterative learning process [5].The agent's behavior in Q-learning is determined by a policy that balances exploration and

exploitation. Initially, the agent explores the environment to gather information about state-action pairs, and as learning progresses, it exploits the learned Q-values to select actions with higher expected rewards. Q-learning employs the Bellman equation to iteratively update the Q-values, gradually approaching the optimal value function. The algorithm converges to the optimal policy, guaranteeing optimality and convergence under certain conditions [6]. Q-learning has shown success in various domains, especially in problems with discrete states and actions. However, it faces challenges in continuous state and action spaces, often requiring additional techniques like function approximation or deep learning to handle large and continuous state spaces effectively.

In order to broaden the scope of Q-learning, the DeepMind team unveiled DQN in 2013 [7]. DQN solves the issues posed by continuous state and high-dimensional action spaces by fusing Q-learning and deep learning. Environment observations are inputted into DQN, which then generates corresponding Q-values for every potential action. To do this, a Q-network is trained to resemble the ideal Q-value function [8,9]. The training process of DQN incorporates two key techniques: experience replay and target network. Experience replay involves storing and replaying experience samples of the agent's interactions with the environment. This method improves sample efficiency and data utilization, enabling the agent to learn from past experiences. The target network is employed to stabilize the training process. It involves periodically updating a separate network with the parameters of the Q-network, which reduces target value deviations during training.

Dueling DQN serves as an enhancement of the traditional DQN algorithm. It introduces a novel approach by decomposing the Q value function into two distinct components: the state value function, denoting the overall value of choosing any action in a given state, and the advantage function, quantifying the relative advantage of each action compared to others [10]. This decomposition allows Dueling DQN to grasp more knowledge of the underlying structure of the Q values and facilitates more accurate estimation. By separately modeling the state value and advantage, Dueling DQN effectively disentangles the value estimation process, enabling more efficient and precise value function approximation. This innovative approach not only improves the learning process but also enhances the algorithm's ability to generalize across different states and actions. Through the utilization of this decomposition technique, Dueling DQN strives to achieve more efficient and accurate Q value estimation, resulting in improved overall performance in RL tasks.

In previous research, researchers have investigated the comparative performance of DQN and Dueling DQN algorithms. Hessel et al. conducted a study comparing DQN with various enhanced algorithms, including Dueling DQN, Double DQN, and Noisy DQN. Their findings revealed that Dueling DQN achieved significantly higher scores compared to DQN and outperformed other DQN-based enhanced networks [8].

The DeepMind team, responsible for the development of Dueling DQN, also employed the Atari game environment to compare the DQN algorithm with Dueling DQN. Their results demonstrated that the Dueling DQN algorithm achieved higher scores in the game, highlighting its superior performance. Additionally, this study included visualizations to illustrate the differences between Dueling DQN and DQN

on the same image, further elucidating the advantages of the Dueling DQN algorithm [10].

Building upon these previous findings, this article aims to explore the performance disparities between Dueling DQN and DQN in a more complex environment, namely the Super Mario game. By examining their performance in this challenging environment, the study seeks to provide further insights into the differences and capabilities of Dueling DQN and DQN algorithms.

3 Method

3.1 DQN

The DQN algorithm was developed to handle decision-making problems in the context of Markov Decision Processes (MDPs), exploiting the advantages of RL and deep learning. The Q-value function, which is calculated by DQN utilizing deep neural networks, represents the expected cumulative reward in a particular state, received from an specific action.

A number of hidden layers are often present in the DQN, in addition to an output layer. Convolutional layers and fully connected layers are widely employed in the hidden layers to manage the raw input data gathered from the environment. The number of output nodes is equal to that of available actions, and each node in the layer reflects the Q-value for a certain action. The neural network layer that was used in this essay is displayed in Table 1.

Table 1. Architecture of the neural network used in this paper.

Layer	Output Shape	Param
Conv2d-1	[-1,32,20,20]	8224
Conv2d-2	[-1,64,18,18]	18496
Linear-3	[-1,512]	10617344
Linear-4	[-1,12]	6156

The neural network has two convolutional and two linear layers in total, as shown in the table. The neural network utilizes the q value as its final output before training to bring it closer to the desired q value. Each layer's activation function is a relu function.

The learning target of DQN is to reduce the mean squared error between the estimated Q-values and the intended Q-values. The goal Q-values are computed using a separate target network with predetermined parameters to increase stability throughout training. The formula for updating the Q value is as follows.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q\left(s_{t+1}, a'\right) - Q(s_t, a_t) \right] \tag{1}$$

$Q(s_t, a_t)$ indicates the expected cumulative reward for the current state and action pair, and the Q-value of performing an action in the current state. α is the learning

rate, which determines how large an update will be. Each update has a smaller impact when the learning rate is low, and each update has a bigger impact when the learning rate is high. r_{t+1} represents the immediate reward obtained after taking an action a_t in the current state s_t, which is the reward feedback received when performing an action in the environment. γ is used as a discounting factor to balance the weight given to both current and future benefits. More focus is placed on present benefits with a smaller discount factor, and more attention is placed on future prizes with a greater discount factor. $\max_a Q\left(s_{t+1}, a^{'}\right)$ represents the action with the largest Q value in the next state s_{t+1}, it essentially indicates that the agent selects the following state using the current Q value function. The best thing a state can do.

The term on the right side of the formula represents the highest Q value of the ensuing state after discounting the immediate benefit obtained by acting in the current state and taking that action's Q value into account. This is expected to lead to the identification of the perfect policy as the Q-value function finally approximates it ideal form. By iteratively updating the Q-value function, the Q-learning algorithm can decide the optimum strategy for the RL challenge, allowing the intelligent agent to select the best course of action in the environment.The value functions that were discussed in the section before are objects with a high dimension. This work may use a DQN, $Q(s, a; \theta)$ with parameterθ, to approximate them. At iteration i, this work optimizes the following series of loss functions to estimate this network:

$$L(\theta) = E\left[\left(r + \gamma \max_a Q\left(s^{'}, a^{'}; \theta^{-}\right) - Q(s, a; \theta)\right)^2\right] \tag{2}$$

$L(\theta)$ is the mean square error between the goal Q value and the predicted Q value. E reflects the expected value, which entails adding up or averaging the samples from each potential pair of state-actions. r represents immediate reward, which is the reward feedback obtained by the agent after performing action a in state s. $s^{'}$ represents the next state. $a^{'}$ depicts the course of action chosen in the next state. γ is used to balance the weighting of current and future benefits. $Q(s, a; \theta)$ is the output of the DQN, representing the Q-value estimate for taking action a in state s. $Q\left(s^{'}, a^{'}; \theta^{-}\right)$ is the output of the target network, which represents the Q value estimation of taking action $a^{'}$ under the next state $s^{'}$, where θ^{-} means fixed the target network parameters, where the parameters of a fixed, independent target network are represented by. This work might attempt to utilize conventional Q-learning to learn the network $Q(s, a; \theta)$'s parameters live. In actual use, this estimator does poorly. The use of gradient descent to update the online network $Q(s, a; i)$ while freezing the target network's parameters for a predetermined number of iterations was a significant breakthrough [11].

3.2 Dueling DQN

The main difference between Dueling DQN and DQN is how the Q value is calculated. The following formula explains how Dueling DQN and DQN differ from one another. The DQN Q value function is expressed as follows:

$$Q(s,a) = f_\theta(s,a) \tag{3}$$

One of them, and the parameter, is the DQN neural network function. State and action are inputted directly, and the appropriate Q value is output.

Dueling DQN's Q value function is written as:

$$Q(s,a) = V_\theta(s) + A_\theta(s,a) - \frac{1}{|\mathcal{A}|} \; {}_{a'} A_{heta}(s,a') \tag{4}$$

Among them, $V_\theta(s)$ is the state value function, which symbolizes the entire worth of state s; $A_\theta(s,a)$ is the advantage function, which weighs each action's benefit compared to other ones. Both the state value function and the advantage function are parameterized by the neural network function f_θ. When they are combined, Dueling DQN can more accurately evaluate the relationship between state and action and calculate the Q-value. This separation makes dueling DQN more accurate and efficient when learning state-action value functions.

With the exception of a few extra nodes added at the output layer to calculate the state value and benefits, Dueling DQN's network structure matches that of DQN. Nevertheless, throughout the network's general training phase, the mean square error of the Q value is optimized.

The neural networks of fighting DQN and DQN differ in the interim. The Q value and the V value are separated from the Dueling DQN output. As demonstrated in Fig 1.

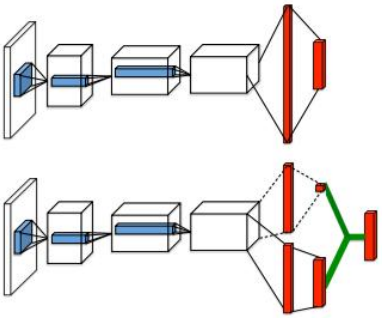

Fig. 1. Architecture of DQN and Dueling DQN (Figure credit: Original).

4 Result

This study compares the performance of DQN and Dueling DQN over the course of 3000 epochs. Additionally, the study explores the impact of two hyperparameters, learning rate and batch size, on the performance of both algorithms. The goal is to achieve better results by investigating the performance differences under different hyper-parameter settings.

4.1 Comparison in 3000 Epochs

The performance comparison of Dueling DQN and DQN with 0.0001 learning rate and 256 batch size is shown in Fig. 2. The graph shows the scores averaged across 100 epochs, giving information on patterns in overall performance. According to the figure, Dueling DQN and DQN both display performance patterns that are similar and have an upward tendency over the course of 3000 epochs. Notably, a considerable improvement has been seen in the first 500 epochs.

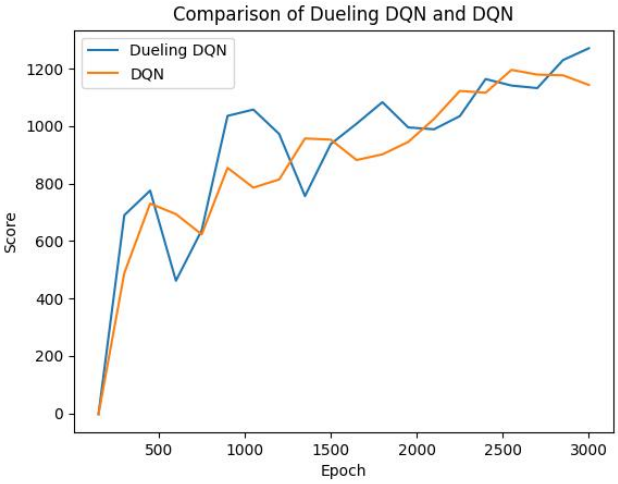

Fig. 2. Result comparison of Dueling DQN and DQN (Figure credit: Original).

The key difference between these two algorithms lies in their performance after the initial improvement. DQN shows a steady and continuous improvement after the initial 500 epochs, whereas Dueling DQN experiences greater fluctuations in performance after a sharp increase near the 1000-epoch mark.

Overall, the performance of Dueling DQN slightly outperforms DQN over the 3000 epochs. Specifically, during the first 500 epochs, Dueling DQN exhibits a higher growth rate compared to DQN. In the subsequent epochs, although Dueling DQN shows larger performance fluctuations, it consistently achieves higher scores compared to DQN. This performance advantage of Dueling DQN persists even after the 3000 epochs.

4.2 Comparison of Hyperparameters

In this study, the focus was on investigating the impact of two important hyperparameters, learning rate and batch size, on the performance of Dueling DQN and DQN algorithms. The analysis was conducted over a period of 2000 epochs, and the average scores of the last 100 epochs were recorded and displayed in Table 2.

Table 2. Performances comparison of various learning rate.

Learning rate	0.0001	0.001	0.01	0.1
Dueling DQN	1034.74	640.34	780.16	767.66
DQN	904.84	648.64	803.16	386.67

Table 2 specifically highlights the results related to different learning rates ranging from 0.0001 to 0.1. The findings demonstrate that the learning rate has distinct effects on the performance of both algorithms. Notably, the optimal performance was observed when taking 0.0001 as learning rate for both Dueling DQN and DQN, with Dueling DQN achieving a score of 1034.74 and DQN obtaining a score of 904.84.

Interestingly, the analysis revealed that Dueling DQN's performance deteriorated significantly when the learning rate increased to 0.001, reaching a score of 640.34. On the other hand, DQN's performance exhibited its lowest point with a score of 386.67 when the learning rate was 0.1. These results suggest that DQN is more sensitive to changes in the learning rate compared to Dueling DQN.

The variations in the algorithms' sensitivity to learning rate may stem from the differences in their architectural design. The unique nature of Dueling DQN, which separates the value and advantage functions, might contribute to its more stable performance across various learning rate values.

Table 3. Performances comparison of various batch size.

Batch Size	64	128	256	512
Dueling DQN	772.86	810.64	944.36	1200.90
DQN	916.22	777.87	883.67	885.21

The performance variance between Dueling DQN and DQN for various batch sizes is shown in Table 3. As the batch size grows, there is a discernible improvement in the average score for Dueling DQN. Particularly, the average score is 772.86 when the batch size is 64. The average score slightly climbs to 810.64 as the batch size is increased to 128. The average score significantly rises to 944.36 when the batch size is increased further to 256. The batch size of 512 results in the greatest average score, which is 1200.90. On the other hand, DQN's performance under various batch sizes exhibits greater unpredictability. Across the various batch sizes, the DQN average scores vary. The average scores for batch sizes 64 and 128 are 916.22 and 777.87, respectively, with batch size 64 recording the higher score. The average score slightly decreases to 883.67 when the batch size is raised to 256. The average score stays largely unchanged at 885.21, therefore increasing the batch size to 512 has little discernible impact.

From these results, it can be inferred that Dueling DQN is more sensitive to changes in batch size compared to DQN. It demonstrates consistent performance improvements with larger batch sizes. However, the performance of DQN is less impacted by changes in batch size, showing less stability in terms of average scores across different batch sizes. It is worth noting that the optimal batch size may vary depending on the specific problem and dataset. Further investigation and

experimentation are required to identify the most suitable batch size for achieving optimal performance for both Dueling DQN and DQN algorithms.

In previous studies, Dueling DQN has shown superior performance compared to DQN. However, the results presented in this paper did not yield similar findings. Several factors could contribute to this discrepancy, such as differences in the number of epochs or variations in hyperparameter settings. It is important to note that the performance of deep RL could be sensitive to these experimental choices.

Future research should focus on adjusting the number of epochs and systematically comparing more hyperparameter configurations. By exploring different choices in these settings, a better understanding of the optimal conditions for achieving superior performance can be obtained. This process will help to identify the appropriate trade-offs between computational resources and the learning ability of the algorithm.

Additionally, conducting a comprehensive sensitivity analysis on hyperparameters, such as exploration-exploitation trade-offs, and network architectures, will be crucial for further improving the performance of the algorithm. By meticulously adjusting and fine-tuning these parameters, it could gain insights into how they impact the convergence speed and stability of the algorithm, ultimately aiming for better results.

5 Conclusion

In this study, an analysis of the performance of Dueling DQN and DQN algorithms was conducted 3000 epochs in a specific task and compare the result of two different hyper-parameter. Through careful experimentation and evaluation, the performance of both algorithms was observed and documented. The obtained results deviated from previous studies, suggesting potential differences in the experimental setup or hyperparameter configurations. Throughout the 3000 epochs, the performance of the algorithms, Dueling DQN and DQN, was carefully analyzed. Initially, both algorithms faced difficulties in learning an effective policy, leading to a period of stagnation in their performance. However, after this initial phase, both algorithms started to show rapid improvement and continued to increase their scores. By the end of the 3000 epochs, they achieved scores close to 1300 points. When comparing the specific hyperparameter settings of learning rate and batch size, it is evident that they influenced the performance of the algorithms differently. Under a learning rate of 0.0001 with 256 batch size, Dueling DQN significantly improved and achieved the highest score. However, DQN performed best when the batch size was 128 and the learning rate was 0.001. Interestingly, Dueling DQN exhibited a faster rate of improvement compared to DQN, especially around the 500-epoch mark. This surge in performance could be attributed to the specific hyperparameter settings of Dueling DQN, which may have allowed it to learn more effectively in the given task. Despite not achieving similar results as previous studies, this research provides valuable insights into the potential factors influencing the performance of Dueling DQN and DQN algorithms. The differences in the observed patterns may be attributed to the number of epochs and variations in hyperparameter settings.

Future work should focus on adjusting the number of epochs and conducting a systematic exploration of hyperparameters to further refine and improve the performance of Dueling DQN in the given task. By critically evaluating and fine-tuning these factors, it is possible to gain a deeper insight of the algorithm's capabilities and optimize its performance.

References

1. Torrado, R. R., Bontrager, P., Togelius, J., Liu, J., & Perez-Liebana, D. Deep reinforcement learning for general video game ai. In 2018 IEEE Conference on Computational Intelligence and Games (CIG), 1-8 (2018).
2. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. Openai gym. arXiv preprint arXiv:1606.01540 (2016).
3. Bellman, R. Dynamic programming and Lagrange multipliers. Proceedings of the National Academy of Sciences, 42(10), 767-769 (1956).
4. Watkins, C. J., & Dayan, P. Q-learning. Machine learning, 8, 279-292 (1992).
5. Lyu, L., Shen, Y., & Zhang, S. The Advance of reinforcement learning and deep reinforcement learning. In 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), 644-648 (2022).
6. Pandey, D., & Pandey, P. Approximate q-learning: An introduction. In 2010 second international conference on machine learning and computing, 317-320 (2010).
7. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).
8. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI conference on artificial intelligence, 32, 1 (2018).
9. Sewak, M., & Sewak, M. Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence. Deep Reinforcement Learning: Frontiers of Artificial Intelligence, 95-108 (2019).
10. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. Dueling network architectures for deep reinforcement learning. In International conference on machine learning, 1995-2003 (2016).
11. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. Human-level control through deep reinforcement learning. nature, 518(7540), 529-533 (2015).