Comparison of DQN and Double DQN With Dueling Architecture on Super Mario Bros

Du Han
School of Computer Science, McGill University
du.han@mail.mcgill.ca

Abstract

In this project, I compare Deep Q-Network (DQN) and Double Deep Q-Network (Double DQN), both augmented with Dueling Network architectures. DQN uses a target network to estimate maximum Q-values, which can lead to optimistic bias. Double DQN addresses this by using the online network to select actions and the target network to estimate their Q-values, reducing overestimation. The Dueling Network architecture, incorporated into both models, splits the network into streams for estimating A and V. Both of them are tested on the Super Mario Bros environment from the OpenAI Gym library. The results demonstrated that with dueling architecture, the difference between DQN and double DQN is not significant, implying that the dueling architecture minimizes the performance differences between DQN and Double DQN.

1 Introduction

This project compared the effectiveness of DQN and Double DQN with dueling architecture, specifically on a Super Mario Bros dataset, using the OpenAI Gym. As an intersection of my interests in video games and machine learning, this study aims to explore the DQN algorithm, which was covered but not deeply explored in lecture.

As a video game lover, I am always interested in how reinforcement learning could be used in video game environments. However, modern video games maybe too complex since it may include multilayer mechanics and evaluation of state and reward would be difficult. The choice of Super Mario Bros represents a blend of simplicity and complexity, making it an ideal candidate for this project. Comparing to existing works, this project focused on comparing the performance of different architectures of DQN on this specific environment, and I believe it would provide insights into RL field.

2 Background and Related Works

2.1 DQN

Deep Q-Network (DQN) is a approach of reinforcement learning that combine traditional Q-Learning with deep neural networks. Mnih et al. [2] first introduced DQN, demonstrating DQN is capable of mastering effective strategies directly from complex, high-dimensional sensory data through a comprehensive reinforcement learning process, and it could reach a performance level equivalent to a professional human game player across a collection of 49 games.

In addition to using of neural networks, one critical difference between DQN and regular Q-Learning algorithm is the use of target network, which is a clone of the online/policy network that helps stabilize training by providing a fixed set of parameters for calculating the target Q-values during updates. This network's weights are periodically updated to match weights of online/policy network.

2.2 Double DQN

Although the use of DQN was successful, there are still some issues such as "over-estimation" of Q-Values. One usual improvement is double DQN. In Double DQN, the action is selected based on the Q-values from the online/policy network, which is continuously updated, ensuring that the most current data influences action selection. Sewak [3] summarized that the 'over-estimation' problem of Q Values and the instability in the target values could be simultaneously overcame when using double DQN.

2.3 Dueling Architecture

The Dueling Network architecture introduces a further refinement by splitting the network into two distinct pathways: one to estimate the state value function $V(s)$, and the other to compute the advantage function $A(s, a)$ for each action. This bifurcation allows the network to better discern the relative importance of each action within a given state.

Wang et al. [4] showed that the Dueling DQN architecture significantly enhances policy evaluation, particularly when actions have similar values. The architecture decomposes the Q-function as

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in \mathcal{A}} A(s, a'; \theta, \alpha) \right),$$

and has been shown to outperform the state-of-the-art in the Atari 2600 domain, providing a more generalized learning across actions without altering the underlying reinforcement learning algorithm.

3 Methodology

3.1 Data Collection and Processing

The dataset utilized in this study is derived from the 'gym-super-mario-bros[1]' environment, which is part of the OpenAI Gym library. Note that version 0.17.3 of the gym library was used for compatibility.

Effective preprocessing of the environment is proved to be critical for getting better results in this project. Initially, I applied several wrappers commonly used for preprocessing Atari games in the OpenAI Baselines, which provided a standardized approach to modifying the game. Key modifications included frame skipping, frame stacking, grayscaling, resizing, and permuting the image data to match the input format expected by our neural network models. I also used some wrappers implemented by myself to modify the episode reset conditions. Since the reward of this environment is very unstable, I also smoothed the reward before saving it into the replay.

3.2 Experimental Setup

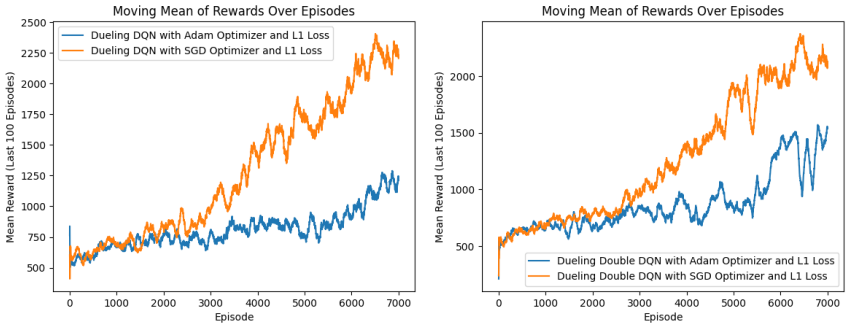
The experiments were conducted using n1-standard-8 machine on Google Colab Enterprise, with NVIDIA_TESLA_T4 GPU accelerator. Each experiment takes 10 to 20 hours on this configuration and weights of trained models are saved.

3.3 Algorithm Training

The training of our algorithm began with an in-depth analysis of existing code found on platforms like Kaggle and GitHub. This initial step was crucial for understanding current methodologies and gaining insights into potential improvements. Building on this foundational knowledge, we then crafted our own algorithm from the ground up, ensuring a tailored fit to the unique demands of our research.

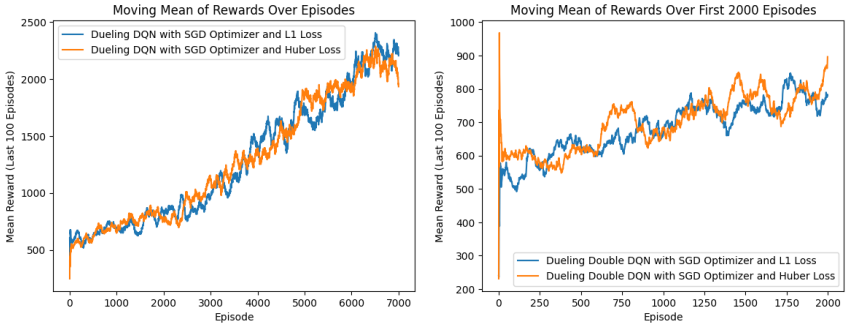
The programming was carried out in Python notebook, using the functionalities offered by pytorch. During the development phase, we addressed specific challenges, such as calculation of temporal difference, ensuring our algorithm was robust and effective.

Our algorithmic framework was architected with modularity, comprising distinct classes that correspond to different DQN agent configurations. This modularity facilitated the exploration of different configurations and integration of diverse loss functions and optimizers.



(a) dueling dqn (b) double dueling dqn

Figure 1: moving rewards of dqn with adam optimizer and sgd optimizer



(a) dueling dqn (b) double dueling dqn

Figure 2: moving rewards of dqn with adam optimizer and sgd optimizer

4 Experiments

4.1 Comparing Adam and SGD optimizer for Dueling DQN

In this experiment, two models were trained: Dueling DQN with Adam optimizer and Dueling DQN with SGD optimizer, both employing L1 Smooth Loss. Training rewards were recorded. To facilitate a clearer comparative analysis, the moving mean of rewards for the last 100 episodes was computed and depicted in a single figure.

As shown in Figure 1(a), the Dueling DQN model utilizing the SGD optimizer demonstrates a quicker increase and achieves higher rewards more rapidly when compared to the model using the Adam optimizer, as indicated by its higher reward curve.

4.2 Comparing Adam and SGD optimizer for Dueling Double DQN

Following the experiment 4.1, the experiment was extended to Dueling Double DQN models, maintaining identical conditions and a similar figure is plotted.

As depicted in Figure 1(b), the Dueling Double DQN with the SGD optimizer exhibits a similar advantage in rewards, mirroring the faster increase and better performance observed with the SGD optimizer in the Dueling DQN setup. This results further proves the effectiveness of the SGD optimizer in these architectures.

4.3 Comparing SmoothL1 loss and Huber loss for Dueling and Dueling Double DQN

Additionally, a comparative analysis was conducted between L1 Smooth Loss and Huber Loss, with both Dueling DQN and Dueling Double DQN models utilizing the SGD optimizer. This aspect of the study was aimed at discerning the impact of loss function selection on model performance.

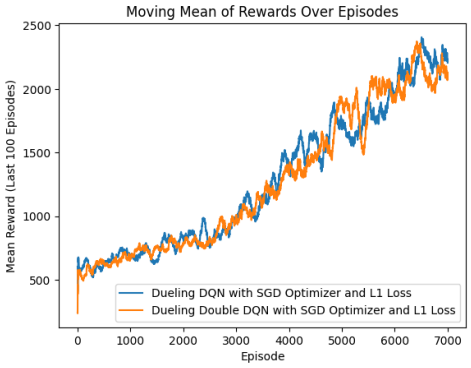


Figure 3: moving rewards of dqn and double dqn, with same optimizer and loss

As shown in Figure 2, it was observed that the performance metrics for models trained with L1 Smooth Loss were nearly identical to those trained with Huber Loss. This similarity was consistent across both the Dueling DQN and Dueling Double DQN architectures, suggesting that, within the context of this experiment, the choice between these two loss functions does not significantly affect the outcome.

4.4 Comparing DQN and Double DQN

To facilitate a direct comparison between Dueling DQN and Dueling Double DQN, both were trained using the SGD optimizer with L1 loss and identical hyperparameter settings for alpha and gamma.

As shown in Figure 3, the moving mean of rewards indicates a negligible difference in their performance. This could suggest that the "over-estimation" and instability traditionally addressed by Double DQN may already be solved by the Dueling architecture's separation of the Q-value into advantage (A) and state-value (V) streams, potentially making the Double DQN's additional complexity unnecessary in this context.

4.5 Comparing Training Duration of Different Models

By analyzing the runtime of each model, we observed consistent trends in training duration across different configurations. For both Dueling DQN and Dueling Double DQN, using the SGD optimizer resulted in approximately 20 hours of training time, irrespective of the loss function employed. In contrast, models utilizing the Adam optimizer required significantly less time, approximately 13 hours, to complete training. However, it is noteworthy that despite its longer training time, the SGD optimizer tended to yield better performance outcomes, as I showed before.

5 Conclusion and Future Works

In our comparative analysis of the dueling DQN and dueling double DQN architectures, we observed closely matched performance metrics between the two models. This observation underscores the pivotal role of optimizer selection in enhancing the efficacy of deep reinforcement learning algorithms. Notably, the selection of the optimizer appears to be more significant than the distinctions between DQN and Double DQN. This finding suggests that the optimization strategy may have a greater influence on the outcome, driving the performance and convergence rate of the learning process.

Future directions include addressing computational constraints that limited the scope of this study, where the cost of Tesla T4 GPU and large replay memory requirements limited the number of conducted experiments. More comprehensive trials are essential, such as exploring the impact of varying learning rates and discount factors, to enhance the robustness and generalizability of the findings. More validation experiments could also be deployed, and expanding computational capacity will be crucial for these additional experiments.

References

[1] Christian Kauten. Super Mario Bros for OpenAI Gym. GitHub, 2018. URL `https://github.com/Kautenja/gym-super-mario-bros`.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. doi: 10.1038/nature14236.

[3] Mohit Sewak. *Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence*, pages 95–108. 06 2019. ISBN 978-981-13-8284-0. doi: 10.1007/978-981-13-8285-7_8.

[4] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/wangf16.html`.