CITS3001 Project Report

Luyang Chen 23426352
Qiya Yu 22957353

Introduction:

Project Goal:

The primary objective of this project is to implement and compare two different AI algorithms: Deep Q-Networks (DQN) and rule-based AI. Our goal is to train an AI agent capable of achieving higher scores in the Super Mario game and to compare the performance of these two algorithms in this task, understanding their strengths and limitations.

Method Used:

We have chosen DQN as a deep reinforcement learning algorithm and rule-based AI as an alternative approach. DQN is renowned for its ability to learn and optimize strategies in complex environments, while rule-based AI relies on game rules and heuristic methods for decision-making. Both methods have their respective advantages in theory and practice, and we will discuss them in detail in this report.

Report Structure:

This report will be organized as follows: Analyzation of both DQN and rule-based method, then the performance comparison of these two methods and a conclusion at last.

Analyzation:

In this section, we delve deeper into the heart of our project—analyzing the performance and intricacies of the two distinct AI algorithms, Deep Q-Networks (DQN) and the rule-based approach. Our journey begins by dissecting the inner workings of each method, shedding light on their strengths, weaknesses, and their practicality when applied to the complex and dynamic environment of the Super Mario game.

Analyzation of DQN:

Baseline:

The Baseline I used for this DQN implementation is from https://github.com/xiaoyou-bilibili/gym_super_mario. Which is using a CNN network and its architecture consists of three convolutional layers followed by Rectified Linear Unit (ReLU) activation functions. These layers progressively extract hierarchical features from the raw game frames, with each layer reducing the spatial dimensions of the feature maps and increasing the number of channels. The resulting feature maps are then fed into the fully connected layers of the DQN for further processing and Q-value estimation. This architecture is designed to effectively capture the relevant information required for making intelligent decisions in the Super Mario game environment.

Algorithm explanation:

The principle of DQN algorithm is to use neural network to substitute the Q-map when we are in this mario scenario which will have too many dimensions if we build the Q-map directly with the pictures and pixels in the game. So we use the neural network to extract the features of images and give these features as the input of our Q-learning method. The model of feature extraction is in models.py and the Q-learning method implementation is in helpers.py.

Core function explanation in helpers.py:

compute_td_loss Function (DQN Loss Computation):

This function calculates the temporal difference (TD) loss for the Deep Q-Network (DQN) algorithm, facilitating the learning process. It updates the Q-network by comparing predicted Q-values with the expected Q-values derived from experiences stored in a replay buffer.

update_epsilon Function:

This function updates the exploration parameter epsilon over episodes using an epsilon-greedy policy. It starts with a high exploration rate (eps_start) and decays exponentially towards a minimum value (eps_final) over time.

Adjustment from baseline:

The goal of the Baseline of the Mario game is to simply pass the levels one by one, but our goal for this project is to let Mario get higher score when he pass the level, we have to change the policy of the reward giving to the agent and focus the performance on World 1 Level 1.

Policy from the baseline:

- Give the agent a reward +400 when he gets the flag

Our adjustment:

- Reduce the reward of getting the flag from +400 to +350
- Add 40 reward for each enemy that Mario eliminated.
- Add 55 reward for each coins Mario get.
- Minus 400 reward per death

Comparison from the Baseline:

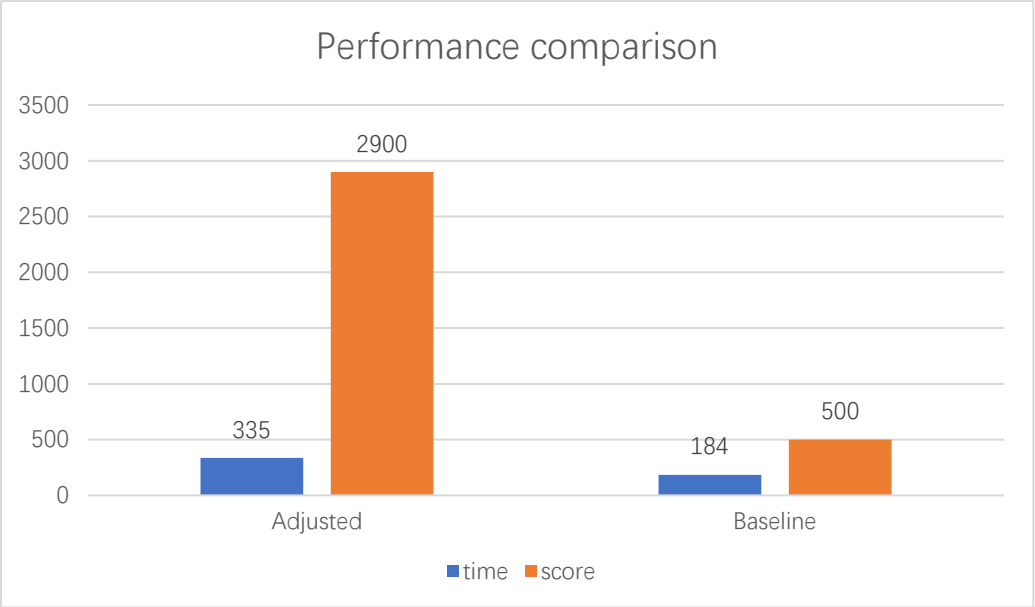
The Baseline implementation, which focus only on got the flag, got 500 score but for time is only 184s.

```
(mario) D:\UWA\CITS3001\project\xy_gym_super_mario\gym_super_mario>python evalation.py
WE GOT THE FLAG!!!!!!
{'coins': 1, 'flag_get': True, 'life': 2, 'score': 500, 'stage': 1, 'status': 'small', 'time': 184, 'world': 1, 'x_pos': 3161, 'y_pos': 122}
326.74999999999585
```

The implementation afther our adjustment, after trained for over 3000 times, we got 2900 scores but it cost more time of 335. An intresting thing is that, when we add this reward of getting coins and eliminate enemies, instead of simply heading to flag and go to right side of the screen straight forward, as it is shown in the following picture, the agent begin to control the Mario turn back to left to collect the coin if it has pass that question mark block earlier.

```
Episode 3136 - Reward: 221.95, Best: 342.95, Average: 141.061 Epsilon: 0.01
(mario) D:\UWA\CITS3001\project\xy_gym_super_mario\gym_super_mario>python evalation.py
WE GOT THE FLAG!!!!!!
{'coins': 3, 'flag_get': True, 'life': 2, 'score': 2900, 'stage': 2, 'status': 'small', 'time': 335, 'world': 1, 'x_pos': 3161, 'y_pos': 103}
324.949999999999
```

So, we get this performance comparison graph from our implementation and Baseline, we can see the score we get is a lot more than what the Baseline get.



Analyzation of Rule-based AI:

Rule-Based Agents significantly differ from DQN methodologies, it operates based on predefined rules and heuristics, the basic idea is jump over the enemies and obstacles, and deal with some special situation. Additionally, rule-based agent has no ability to learn from the operations, it only follows the prerequisite rules.

Performance on Predefined Levels

Our rule-based agent performed consistently on World 1 levels 1. However, for the rest levels of world 1 and other worlds, it has low performance due to the lots of new situations like dynamic obstacle, enemy, and different background like underwater. The agent would need additional heuristics and the capability to discern variations in world colors and features.

Efficiency & Training

Unlike DQN require extensive training, rule-based agent operates without any. While this could be the advantageous for rapidly developing an agent, however, this comes at the expense of adaptability. However, agent require a large amount of analysis at each step, which will have a certain impact on the running speed, and potentially affects the development efficiency of the rule-based agent.

Performance Comparison Between two types of Agents

Level Completion with/without Additional Training

DQN agent was able to compete 5 different levels but with heavy trainings.

Rule based agent was able to complete 5 different levels without additional training, while must with an accuracy rule system. Our rule-based agent currently can only pass the World 1 level 1.

Training Iterations

For the DQN agent, it's noteworthy that it requires a significant number of training iterations to become competitive in the game. Specifically, in the case of "World 1 Level 1," the DQN agent needs approximately 3200 training iterations, with each iteration taking a considerable amount of computation time, roughly around 12 hours. (On my machine with GPU)
In contrast, the rule-based agent doesn't require explicit training. It can execute actions immediately but needs to adhere to predefined accuracy rules.

Time to Complete World 1-1

Because of the agent will try to collect coins the missed, it will control the Mario to go backward to collect the coins, so the time spent is lot more than rule-based agent spent. The average Time spent for rule-based agent is 70 after observing the time count in the upper right corner.

Deaths During Training

There were no training deaths for DQN. Every time it dead I make the agent restart from the beginning.
There were no training deaths, but the agent will start game after 3 deaths.

Marks and Coins

The agent of DQN get highest 2900 marks in World 1-1, which focus a lot on collecting coins and eliminating enemies.
The agent gets an average of 1500 marks and collected 3 coins in World 1-1, as its primary focus was on moving right and avoiding enemies and obstacles.

Conclusion:

In conclusion, the DQN agent demonstrates its potential for high performance in the game after extensive training iterations, achieving higher marks and demonstrating the ability to adapt its strategy based on rewards. However, this comes at the cost of significant computational resources and time. On the other hand, the rule-based agent provides a more straightforward approach, completing levels efficiently without additional training, albeit constrained by predefined accuracy rules. The choice between these two approaches depends on the trade-off between training time and the desired level of adaptability in the game environment.