**A logo of a university of hong kong

AI-generated content may be incorrect.**

**City University of Hong Kong**

**CS4486 Artificial Intelligence**

**Assignment 3**

# **Topic 6 Reinforcement Learning on Lunar Lander**

Zhu Zhitao

SID: 57846759

**1.Background**

Prospect of Lunar Lander Control

The Lunar Lander environment, powered by the Box2D physics engine, can simulate the control problem with relevance to real-world applications in aerospace and robotic systems, for instance, drones and unmanned helicopters. The task requires an agent to learn a control policy that can safely land a spacecraft on a designated landing pad, taking into account factors such as gravity, momentum, and fuel consumption. It might be struggling for traditional control methods to design and tune for these dynamic systems.

Reinforcement learning (RL) provides a promising alternative, enabling the agent to learn an optimal control policy through trial and error, directly from interactions with the environment. This approach eliminates the need for explicit modelling of the system dynamics, allowing for greater adaptability and potentially superior performance in complex and uncertain environments. In this report I consider to use two sub-methods in RL learning, the Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) to bypass the needs for explicit modelling of system dynamics, providing enhanced adaptability and superior performance in different environments.

**2.Description of Environment**

**2.1 Action Space**

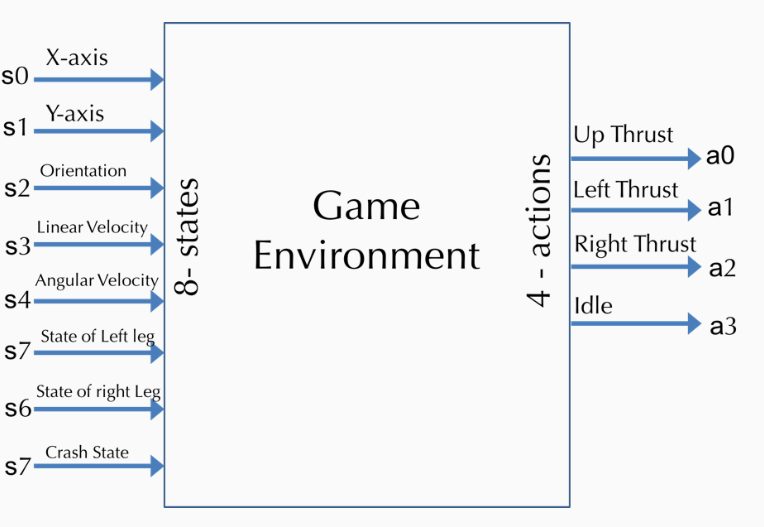
There are four discrete actions available: do nothing, fire left orientation engine, fire main engine, fire right orientation engine.

|  |  |
| --- | --- |
| Action | Result |
| 0 | Do nothing |
| 1 | Left thrust |
| 2 | Up thrust |
| 3 | Right thrust |

**2.2 Observation**

The state is an 8-dimensional vector: the coordinates of the lander in x & y, its linear velocities in x & y, its angle, its angular velocity, and two Booleans that represent whether each leg is in contact with the ground or not.

* X coordinate(float)
* Y coordinate(float)
* X linear velocity(float)
* Y linear velocity(float)
* Angular velocity(float)
* Angle of the Lunar Lander (Orientation)
* state of left leg(bool)
* state of right leg(bool)



**2.3 Reward**

Reward for moving from the top of the screen to the designated landing pad and coming to rest is about 150-180points. If the lander moves away from the landing pad, it loses reward. If the lander crashes, it receives an additional -100 points. If it comes to rest, it receives an additional +100 points. Each leg contact with the ground is +10 points. Firing the main engine is -0.3 points each frame. Firing a side engine is -0.03 points.

**3.Methods**

A diagram of a company

AI-generated content may be incorrect.

**3.1 DQN**

Our Deep Q-Network is designed with a fully connected architecture and consists of 3 linear layers. Each layer contributes uniquely to the network's functionality:

1. **Layer 1: Input Processing**
   * **Type**: Fully Connected (Linear) Layer.
   * **Input**: The state of the environment (size = 8).
   * **Output**: Hidden features (size =: 64).
   * **Activation**: ReLU is applied to introduce non-linearity.
2. **Layer 2: Feature Extraction**
   * **Type**: Fully Connected (Linear) Layer.
   * **Input**: The features produced by Layer 1 (size = 64).
   * **Output**: Refined features (size = 64).
   * **Activation**: ReLU is applied to enhance feature representation.
3. **Layer 3: Action Prediction**
   * **Type**: Fully Connected (Linear) Layer.
   * **Input**: Processed features from Layer 2 (size = 64).
   * **Output**: Q-values for each possible action (size = 4).
   * **Activation**: None

After processing through these layers, the final output is a set of Q-values representing the expected rewards for each action. These values guide the agent's decisions during its interactions with the environment.

A diagram of a network

AI-generated content may be incorrect.

**3.2 Dueling DQN**

The Dueling DQN architecture, proposed by Wang et al. in 2015, introduces improvement over the standard DQN by modifying the way Q-values are calculated. This architecture is consists of two fully connected layers that process the state to extract shared features, followed by two separate streams:

* 1. Advantage Stream: It is a fully connected layer calculates the advantages (A (s, a)) for each action, allowing the model to evaluate the importance of different actions when given a state.
* 2. Value Stream: Another fully connected layer estimates the value (V(s)) of the current state independently from the specific actions.

The final Q-Value Combination:



A diagram of a diagram of a square and square object

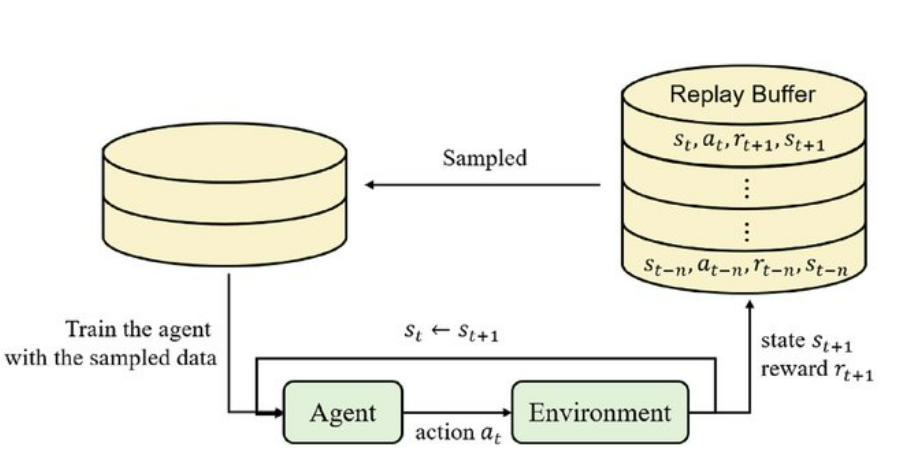
AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **DQN** | **Dueling DQN** |
| **Q-Value Calculation** | **Using direct computations** | **Separates Q-values into 1.state value and 2.action advantage.** |
| **Architectural Streams** | **Single stream** | **Two streams: state value and advantage.** |
| **Normalization** | **None.** | **Normalizes advantages to avoid potential redundancy.** |
| **Strengths** | **Simple and effective for uniform environments.** | **Could handles state-action importance asymmetry better.** |

**4. Boosting technique**

**4.1 Replay Buffer**

The Replay buffer works by storing transitions (state, action, reward, next\_state, done) collected during interaction with the environment and assigning a priority to each experience based on its importance. During training, it samples diverse batches of data using prioritized experience replay, where transitions with higher learning potential are more likely to be included. According to research by Schaul et al. (2015), prioritized replay can enhance learning stability and improve model convergence without increasing computational complexity significantly.



**5. Experiment and Performance**

**5.1 Metrics Chosen**

Since the Lunar Lander game is in box2d engine, a pyplot demonstrating the relationship between number of episodes and scores obtained is sufficient to evaluate the performance.

**5.2 Some testing before/on training**

Before training the reinforcement learning agents for the Lunar Lander environment, some pre-training experiments were made to explore potential solutions and better understand the dynamic environment. These tests aimed to evaluate different methods and identify the most promising algorithms.

Initially, the **Proximal Policy Optimization (PPO)** algorithm was used to construct a baseline solution for the task. PPO demonstrated significant computational efficiency, completing training in approximately 30% of the time required by methods such as DQN or Dueling DQN. However, except for its rapid convergence, the average score achieved by PPO is considerably lower, highlighting the challenge in optimizing the policy for the Lunar Lander's discrete action space. The stochastic nature of PPO introduced greater exploration, but this came at the cost of reduced precision in action selection, which impacted overall performance.

Given the need for higher accuracy and robust decision-making, alternative methods were sought. This led to the implementation of **Deep Q-Networks (DQN)** and **Dueling DQN**, which utilize Q-value optimization to make deterministic decisions. These methods showed promising results in handling the task's reward structure, allowing for more stable performance and higher average scores, albeit with longer training times compared to PPO.

The findings from these initial experiments emphasize tradeoff between computational efficiency and performance quality. As a result, the focus shifted to further refining DQN and Dueling DQN, leveraging their deterministic action selection capabilities and Q-value optimization for tackling the Lunar Lander's complex dynamics.

**6. Experiment Results**

**Training with 1000 episodes**

|  |  |
| --- | --- |
| Method | Running Time (Approximate) |
| DQN | 24 minutes |
| Double DQN | 30 minutes |
| Dueling DQN | 28 minutes |
| Double Dueling DQN | 32 minutes |
| PPO | 9 minutes |

**DQN**

Episode 0 Average Score: -89.32

Episode 100 Average Score: -186.91

Episode 200 Average Score: -164.19

Episode 300 Average Score: -120.70

Episode 400 Average Score: -99.535

Episode 500 Average Score: -50.92

Episode 600 Average Score: 30.298

Episode 700 Average Score: 87.40

Episode 800 Average Score: 110.63

Episode 900 Average Score: 117.81

Episode 1000 Average Score: 124.71

A blue line graph with numbers

AI-generated content may be incorrect.

**Double DQN**

Episode 0 Average Score: -246.18

Episode 100 Average Score: -165.82

Episode 200 Average Score: -159.99

Episode 300 Average Score: -131.34

Episode 400 Average Score: -108.13

Episode 500 Average Score: -69.820

Episode 600 Average Score: 1.7153

Episode 700 Average Score: 66.67

Episode 800 Average Score: 72.65

Episode 900 Average Score: 83.96

Episode 1000 Average Score: 147.74

Score obtained: 88.25697263982065

A blue line graph with numbers

AI-generated content may be incorrect.

**Dueling DQN**

Episode 0 Average Score: -248.36

Episode 100 Average Score: -125.30

Episode 200 Average Score: -25.893

Episode 300 Average Score: 9.9725

Episode 400 Average Score: 15.65

Episode 500 Average Score: 91.04

Episode 600 Average Score: 107.33

Episode 700 Average Score: 112.84

Episode 800 Average Score: 154.91

Episode 900 Average Score: 197.98

Episode 1000 Average Score: 199.00

Score obtained: 109.86581999119802

A blue line graph with numbers

AI-generated content may be incorrect.

**Double Dueling DQN**

Episode 0 Average Score: -38.80

Episode 100 Average Score: -121.86

Episode 200 Average Score: -27.322

Episode 300 Average Score: -33.15

Episode 400 Average Score: -5.175

Episode 500 Average Score: 164.05

Episode 600 Average Score: 163.72

Episode 700 Average Score: 174.93

Episode 800 Average Score: 172.59

Episode 900 Average Score: 185.84

Episode 1000 Average Score: 199.89

Score obtained: 83.25946477119224

A blue line graph with numbers

AI-generated content may be incorrect.

**PPO**

Episode: 100 score: -139.94334598171542 avg score -165.87892887965702

Episode: 200 score: -20.569698074884116 avg score -208.11492645171344

Episode: 300 score: -82.85971500750334 avg score -92.06027323262461

Episode: 400 score: 53.03556348088961 avg score -16.258519828452535

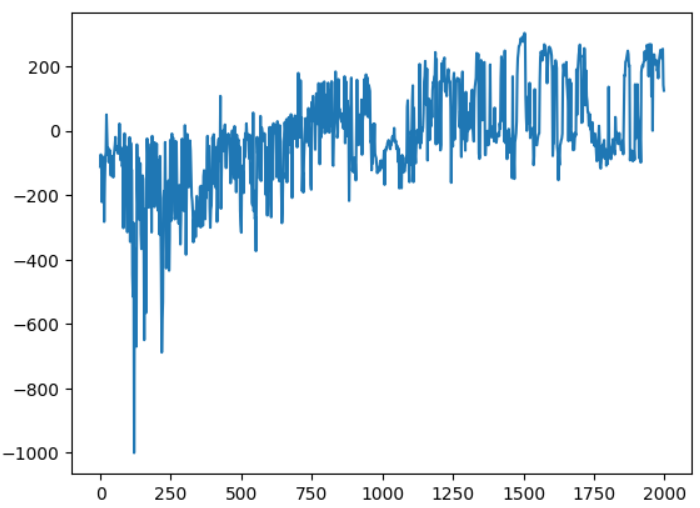
Episode: 500 score: -93.03156273152885 avg score 14.543368840621376

Episode: 600 score: -13.59196305604435 avg score -11.30499482863547

Episode: 700 score: 71.50012655609288 avg score 72.16276263529667

Episode: 800 score: 246.09074006192284 avg score 96.01500481614175

Episode: 900 score: -103.80737204646577 avg score 43.2933715514648

Episode: 1000 score: 124.7371626501414 avg score 92.80582899713117

**7.Findings and Discussions**

After the experiment, I found that Double Dueling DQN achieved the highest performance among the tested algorithms, with a final average score of 199.89, slightly outperforming Dueling DQN (199.00), significantly surpassing Double DQN (147.74), and standard DQN (124.71).

This highlights that combining both dueling architecture and double Q-learning provides substantial benefits for the Lunar Lander games.

I observe that the learning process showed a clear progression across all tested DQN variants. Initially, agents struggled with negative scores, reflecting frequent crashes and instability. Around episodes 400-500, agents began transitioning to achieving positive scores, which showcasing the ability to balance the lander and execute smooth touchdowns with increasing accuracy.

Among the variants:

* Double DQN showed steady improvement, with its score gradually increasing to positive territory by episode 400-500.
* The Standard DQN required more time to achieve competitive scores.
* Dueling DQN and Double Dueling DQN consistently exhibited faster learning and smoother improvements across training.

Although PPO demonstrated exceptional speed, completing training in just 9 minutes, its final performance was lower, with an average score of 92.80. This highlights the tradeoff between rapid training and precision performance in reinforcement learning tasks.

The standard DQN took around 24 minutes to train, while Double DQN, Dueling DQN, and Double Dueling DQN required 28-32 minutes due to their added complexity. Even though these advanced methods take longer, they delivered better results, making the extra training time worthwhile. Among them, Double Dueling DQN stood out as the top performer, offering a great balance between training efficiency and landing accuracy. For applications requiring highly accurate landings, I consider Double Dueling DQN as strongly recommended.

**8. References**

1. Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

2. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995-2003). PMLR.