COSE474-2024F: Final Project
"Develop game AI agent for Lunar Lander using Deep Reinforcement Learning Algorithms"

Kim JinHyeong

1. Introduction

1.1. Motivation

When video games first emerged, people in the world were captivated by them. However, there was another big event making people more captivated, that was the advent of Computer Bot. Before Bot, players were limited to play in fixed environments or comparing their scores with other players individually. The feeling of playing with or against other players simultaneously didn't exist. With the advent of Bot, even in its simplest form, players experienced a new feeling of competition against a virtual opponent. Yet, these early Bots, just following pre-programmed patterns without considering player actions, eventually lost their appeals due to their lack of adaptability and novelty.

As technology advanced, Game Bot also evolved. Game bots have come to be called Game AI Agents by having AI capabilities. Modern Game AI Agent, hereafter referred to as 'Agent', can now observe player actions, formulate strategies, and effortlessly achieve their objectives using various algorithms. One of the most famous examples of Agent is "AlphaGo", developed by DeepMind. AlphaGo's ability to analyze its opponent's moves and calculate its next-step made it to win the world champion Lee Sedol. Like this achievement, there is immense potential for further AI agent innovation. And here, this project aims to figure out Agent models by developing and estimating a new Agent.

1.2. Problem definition

Continuous decision

The Agent must decide the direction of Lander and whether or not to turn on the engine in that direction. These decisions must be made correctly at the right time and always until the Lander lands. To learn and make these decisions properly is a difficult challenge of this project.

Stability and Convergence

DRL algorithms can face instability and slow convergence due to the complexity of high-dimensional input spaces and the learning process. Ensuring the Agent continuously improves without oscillation or divergence is a key challenge.

By addressing these challenges, this project aims to contribute to the development of a Lunar Lander AI Agent and advance the understanding of game AI Agent development using DRL.

1.3. Contribution

This paper focuses to improve the existing training model used in Lunar Lander. We tried to improve the training model that can achieve better scores based on Lunar Lander's reward score.

2. Method

Significance / Novelty

Currently, the game development industry is increasingly demanding immersive experiences, and the role of intelligent AI bot is growing for this. The more vivid and performance-enhanced AI is, the more fun the game is. Developing AI agents with improved performance based on the Lunar Lander game environment can be a great help in the game development. Moreover, Lunar Lander's AI can provide modules to develop useful AI models in various physical simulation environments such as space exploration, autonomous driving, and parking. Most of the existing studies have used machine learning, such as Q-learning, which is relatively simple. In this study, we will learn and inspire models using deep learning using the neuron network to create AI agents that can learn well even in complex environments.

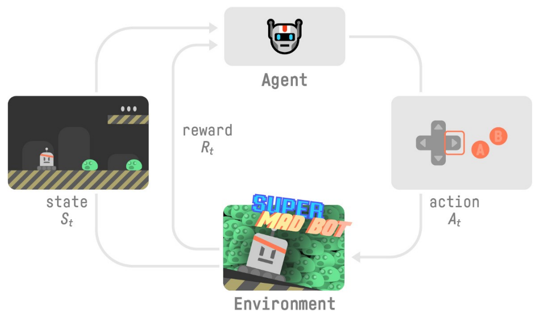


Figure 1. Process

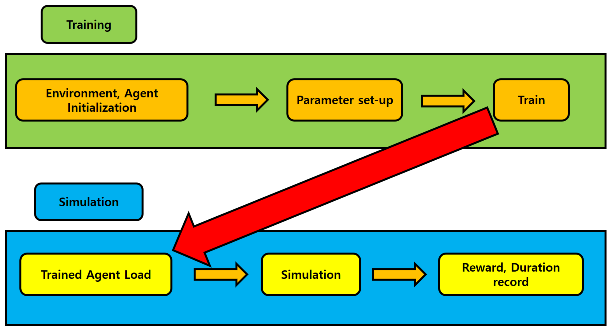


Figure 2. Model Pipeline

Recognizing the environment and receiving the state value first. According to the state, the agent performs the next action, and the environment is changed by the action. The state is newly determined according to the changed environment, and a reward is provided for each cycle. First, initialize the environment and agent for training, set-up the parameter, and then train. To determine the performance of the agent after training, the reward score and duration time are measured through simulation.

1. Initialize the environment with "env.reset()" for the first time
2. In each action, a random action was taken through the same code as "np.random.randint()"
3. After that, the final score is obtained in the "totalreward += reward" method and the performance is evaluated.

3. Experiments

3.1. Dataset

We used the 'Lunar Lander v3' environment provided by Gymnasium for the game environment of Lunar Lander. To develop the agent, We used Juliankappler's Lunarlander training agent and dataset. Agent was defined through agent_class. The agent was learned through training_agent, and the learned agent was stored in training_agent. And We simulated the agent through run_agent. During the training process, a total of 100 runs were executed using 100 episodes.

3.2. Computer Resource & Experimental Design

The experiments were conducted using Google Colab free version, a cloud-based Jupyter Notebook platform.

- GPU: NVIDIA Tesla T4 (provided by Colab).
- CPU: Intel Xeon Processor (Colab default configuration).
- RAM: 12 GB (Colab Free version resources).
- Operating System: Ubuntu 20.04 (Colab backend).
- Deep Learning Framework: PyTorch
- Additional Libraries: Gymnasium, pygame, numpy, itertools, Matplotlib

For initialization, the dimensionality of the state space and the number of possible actions were provided as parameters.

3.3. Experimental Design

In the process of training the agent, the base training model used Linear and ReLU as activation functions. However, in the case of the existing ReLU, if the value is less than 0, the slope becomes 0 and there is a problem that activation is not possible at all. So, We plan to use GELU instead of ReLU to achieve better performance.

3.4. Results

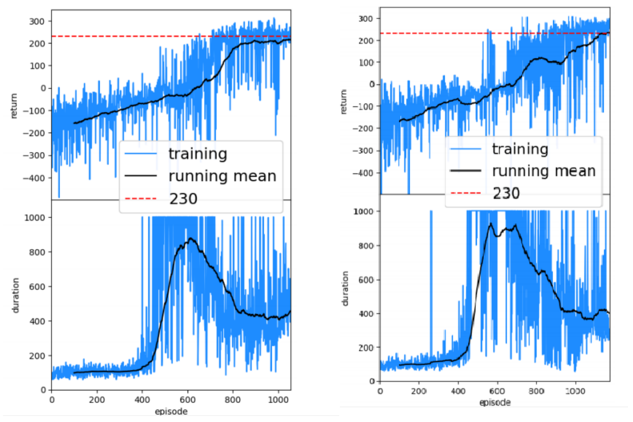


Figure 3. Result Comparison

The left is the result of the base training model, and the right is the result of the improved training model. The x-axis is the number of episodes, and the y-axis is the reward score value and duration time value. When comparing the above plot, it can be seen that the base training model did not exceed the score 230, but the improved training model reached 230. In other words, the improved training model can obtain better scores in the Lunar Lander game.

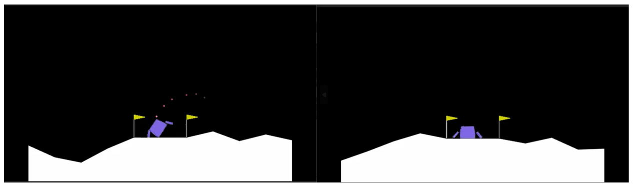
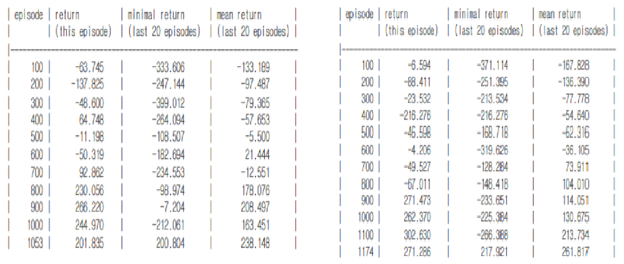


Figure 4. Land Comparison

The left is the landing result of the base training model and the right is the improved training model. As can be seen from the figure, it can be seen that the landing of the base training model is unstable and the landing of the improved training model is more stable. In other words, the improved training model can land more stably in the Lunar Lander game.



```
| episode | return         | minimal return     | mean return        |
|         | (this episode) | (last 20 episodes) | (last 20 episodes) |
|---------------------------------------------------------------------
|     100 |      -63.745   |       -333.606     |      -133.189      |
|     200 |     -137.825   |       -247.144     |       -97.487      |
|     300 |      -48.600   |       -399.012     |       -79.365      |
|     400 |       64.748   |       -264.094     |       -57.653      |
|     500 |      -11.198   |       -108.507     |        -5.500      |
|     600 |      -50.319   |       -182.694     |        21.444      |
|     700 |       92.862   |       -234.553     |       -12.551      |
|     800 |      230.056   |        -98.974     |       178.076      |
|     900 |      266.220   |         -7.204     |       208.497      |
|    1000 |      244.970   |       -212.061     |       163.451      |
|    1053 |      201.835   |        200.804     |       238.148      |
```

```
| episode | return         | minimal return     | mean return        |
|         | (this episode) | (last 20 episodes) | (last 20 episodes) |
|---------------------------------------------------------------------
|     100 |       -6.594   |       -371.114     |      -167.828      |
|     200 |      -88.411   |       -251.395     |      -136.390      |
|     300 |      -23.532   |       -213.534     |       -77.778      |
|     400 |     -216.276   |       -216.276     |       -54.640      |
|     500 |      -46.598   |       -168.718     |       -62.316      |
|     600 |       -4.206   |       -319.626     |       -36.105      |
|     700 |      -49.527   |       -128.284     |        73.911      |
|     800 |      -67.011   |       -148.418     |       104.010      |
|     900 |      271.473   |       -233.651     |       114.051      |
|    1000 |      262.370   |       -225.384     |       130.675      |
|    1100 |      302.630   |       -266.388     |       213.734      |
|    1174 |      271.286   |        217.921     |       261.817      |
```

Figure 5. Return value Comparison

The left is the result of the base training model, and the right is the result of the improved training model. The return value was recorded and compared for each episode. It can be seen that the improved training model scored better than the base training model in almost all episodes.

3.5. Discussion

In this study, we compared the performance of the traditional ReLU activation function with the GELU activation function, which was utilized in the improved training model. While ReLU activation functions are widely used in deep learning models, they have a limitation when handling negative input values. Specifically, for input values less than zero, ReLU outputs a gradient of zero, which results in inactive neurons and can lead to issues such as the dying ReLU problem, where neurons fail to learn during training.

On the other hand, GELU provides a smoother approximation of the Gaussian error function, allowing for non-zero gradients even for negative input values. This characteristic enables GELU to retain more information from negative inputs, leading to better gradient flow during training and mitigating the issue of inactive neurons. By introducing GELU in our model, we observed that the activation function was more effective in learning from both positive and negative values, resulting in improved performance.

When evaluating the performance of the Lunar Lander agent, we saw a noticeable improvement in the training results. The model trained with GELU achieved higher scores and demonstrated more stable and successful landings compared to the base training model, which used ReLU. These results suggest that GELU's ability to provide non-zero gradients for negative inputs helped the agent learn more effectively, resulting in better overall performance.

In summary, replacing ReLU with GELU has demonstrated significant advantages in terms of both training stability and model performance. By addressing the limitations of ReLU and allowing the agent to learn more efficiently from negative inputs, GELU contributed to a more effective Lunar Lander agent, outperforming the base model in various key aspects.

4. Future Direction

In this study, the Lunar Lander environment provided by Gymnasium was somewhat basic and lacked complexity. For future work, we aim to enhance the environment by adding more challenges, such as time, fuel consumption, landing speed, and environmental factors. This will create a more engaging game experience. Along with these improvements, we will adapt the AI agent to handle the increased complexity, aiming to train a more capable and effective model for the enhanced environment.

References

Gymnasium. An api standard for reinforcement learning with a diverse collection of reference environments, 2024. https://gymnasium.farama.org/ [Accessed: 2024-10-10].

HuggingFace. Train your first deep reinforcement learning agent, 2022. https://huggingface.co/learn/deep-rl-course/unit1/hands-on [Accessed: 2024-12-05].

juliankappler. lunar-lander: Reinforcement learning algorithms for training an agent to play the game lunar lander, 2023. https://github.com/juliankappler/lunar-lander [Accessed: 2024-12-05].

Kwiatkowski, A., Towers, M., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., and Younis, O. G. *Computational Complexity of Machine Learning*. PhD thesis, Cornell University, 2024.

TensorFlow. Introduction to rl and deep q networks, 2023. https://www.tensorflow.org/agents/tutorials/0_intro_rl?hl=en [Accessed: 2024-10-15].