**The report of Mario game by Reinforcement Learning**

Zhou Qiang

Dongseo University

1. **Introduction**

Reinforcement learning is widely used in the gaming field to train AI agents to play games like humans. Agents interact with the game environment by observing the state of the game and executing actions, and receive rewards or punishments as feedback from the environment.

This report selects Mario, a classic game, as a case study to test the iterative evolution process through training agents. In training, the agent will determine the next action based on the current game state and observation information, and obtain rewards through interaction with the environment. By continuously adjusting the behavior of the agent and optimizing it based on feedback from reward values, the agent's behavior will gradually tend towards the optimal solution, enabling it to achieve the highest score in the game.

1. **Relate work**

In reinforcement learning, the goal of agents is to learn the optimal behavior by maximizing the cumulative reward value. During the training process, the agent passes the current "state" as input to the neural network, and through the calculation of the neural network, obtains corresponding "actions" and "rewards". The agent executes "actions" and interacts with the environment, receiving new states and rewards. Based on these feedback information, the agent updates the weight parameters in its neural network again to better predict the next "action".

Specifically, the agent uses an objective function that includes "rewards" and "actions" to evaluate its behavior, and uses gradient ascent to update the weight parameters in the neural network. In Mario's game, agents can use functions encapsulated within Gym to update "status" and "rewards" for more efficient training.

Through continuous iterative training, the agent's neural network gradually learns the optimal strategy, enabling it to achieve higher scores in the game.

1. **Task definition**

The task we are considering is to learn to play Mario, and the purpose of this game is to get Mario to the target point on the right as soon as possible.

Firstly, the agent should first be in a certain environment, recognize, understand, and learn to interact, and then derive appropriate behavior from a series of actions,

The environment is the first level for running Mario on Python. At the runtime of each game, the agent selects one action from a series of actions. We formalize this problem as follows:

• **State s**: The current observation represents the matrix of image frames, including Mario's position, obstacles, and monsters.

• **Action a**: A series of predefined actions, integers within the range of [0, 6]. They are {('NOOP '), ('right'), (right ',' A '), ('right', 'B'), ('right ',' A ',' B '), ('A'), ('left ')}.

• **Reward r**: The reward returned by the environment. The reward should be calculated based on Mario's movement distance to the right - game running time.

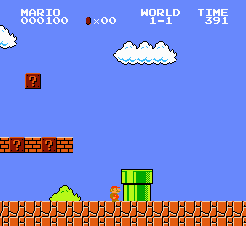


Figure 1 : Mario

1. **Approach**

This training used stable\_ The PPO algorithm in the baselines 3 library is used to train agents for Mario games. The PPO algorithm is a reinforcement learning algorithm based on policy gradients, which has the advantages of fast convergence speed and good performance, and has been widely applied in the field of reinforcement learning.

During the training process, the agent used the PPO algorithm to train 160000 times in the game environment. By recording each reward value during the training process, the training results can be visualized in TensorBoard. It can be observed that as the number of training sessions increases, the reward value obtained by the agent shows a steady upward trend, indicating that the agent's behavioral strategy is continuously optimized.

The PPO algorithm gradually learns the optimal strategy by continuously updating the strategy, enabling agents to achieve higher scores in the game. In this training, the performance of the PPO algorithm indicates its good effectiveness and feasibility in the training of Mario games.

**4.1.Hyperparameters Tune**

For parameters that do not need to be manually set through the training process, such as learning rate, discount factor, etc. The different values of hyperparameter will have an important impact on the training process and results.

In this training, optuna library is used to adjust hyperparameter. Optuna is an automatic hyperparameter optimization framework, which can automatically select the optimal hyperparameter combination according to the results of the objective function, thus accelerating the training process of the model. Through optuna, we can automatically search the best combination of hyperparameter to achieve better training results.

This is right for 'n'\_ steps'，'gamma'，'learning\_ rate‘，'clip\_ range'，'gae\_ Lambda 'for adjustment.

1. **Data and Experiment**

This experiment used the PPO algorithm to train agents for Mario games. Firstly, we conducted an experiment under default parameters to observe the performance and convergence of the agent during the training process. Then, we use optuna library to adjust the hyperparameter of PPO algorithm, and explore better hyperparameter combination to obtain better training effect.

By running models with different training times, it can be found that Mario can perform better with increasing training times.

At the same time, we can also record the final hyperparameter to provide reference for subsequent model optimization.

Through the observation and analysis of the experimental results, we can get the final game performance and convergence of the agent, as well as the impact of different hyperparameter combinations on the model performance.

1. **Discussion and Analysis**

In the training, Mario's environment has a total of 240 \* 256 pixels, which is too large a state space. It is actually difficult to obtain a suitable result in just over 100000 training sessions. However, from Figure 2, it can be seen that as the number of training sessions increases, the rewards obtained by the agent have significantly improved. This indicates that the agent has learned better strategies to complete game tasks.

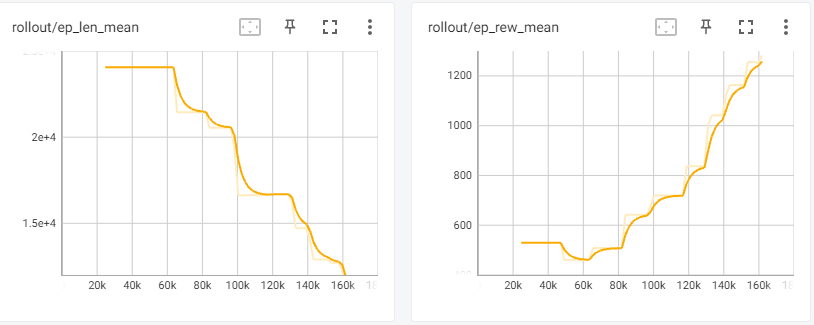


Figure 2 : PPO Train result

Figure 3 shows that as the training frequency increases, the relevant parameters are constantly adjusted.

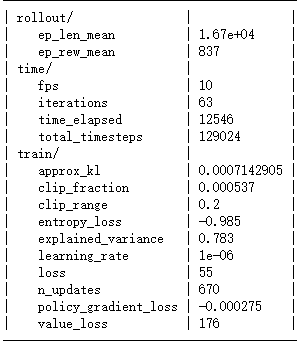
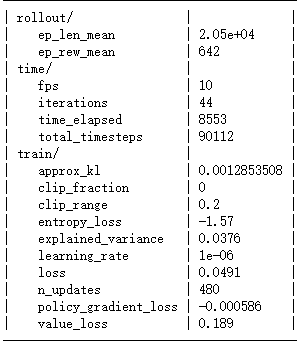
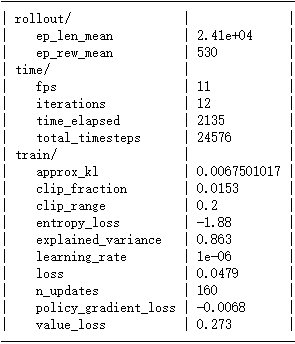


Figure 3 : PPP Train result

Figure 4 shows the positions that Mario can advance to for different training times models.

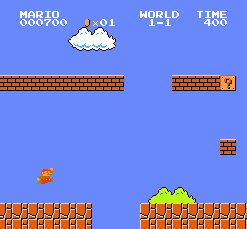
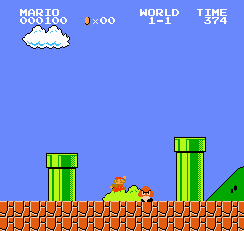
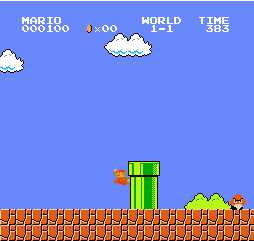


Figure 4 : The results of 5000, 30000, and 160000 models, respectively

The adjustment of hyperparameter has an important impact on the training effect of agents. When adjusting hyperparameter, it is necessary to fully consider the limitations of training time and computing resources of the model. As can be seen from Figure 5, in this experiment, we found that although hyperparameter combination can improve the model performance, it requires more training time and computing resources, so we need to balance between time and resources.

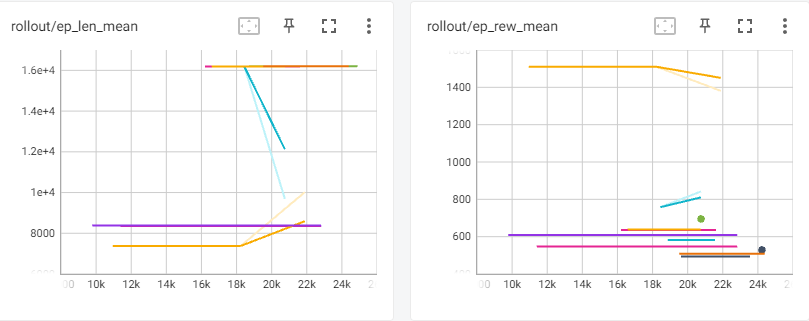


Figure 5 : Hyperparameter Tune

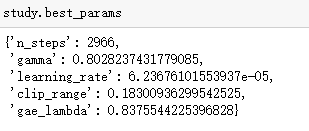


Figure 6 : Best Parmes

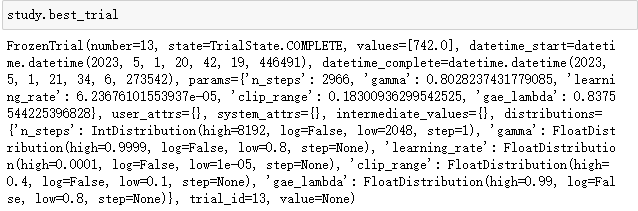


Figure 7 : Best trial