The Journey of AI in Mastering Street Fighter II

Optimizing Reward Functions for AI Training in Reinforcement Learning

Jing Wang, Alexander Moshchev,
Saarthak Kataria, Suchitra Nayak, Yiting Yuan

03.May 2024

Abstract

This project leverages the OpenAI Gym training an AI model using Proximal Policy Optimization (PPO)[1] with Convolutional Neural Network (CNN)[2] architecture. Initial trials with a simple reward function led to cowardice behavior in the model. By integrating advanced techniques from recent research, including a "penalty decay" mechanism[4], we significantly improved the AI's combat strategy, achieving a win rate of 99% in best-of-three matches. This project was showed in Github

1 Introduction

We focuses on training an AI to compete in the challenging game of Street Fighter II using reinforcement learning techniques. We constructed a tailored environment by using the OpenAI Gym framework. Our initial approach used a straightforward reward function, which unfortunately encouraged the AI to avoid confrontation—a behavior termed as "cowardice".

To overcome this problem, we drew inspiration from seminal works. Specifically, the hyperparameter settings discussed in "DIAMBRA"[3] provided a foundation for optimizing our model's learning parameters. Furthermore, the "penalty decay" mechanism proposed in "Mitigating Cowardice for Reinforcement Learning"[4] proved crucial. This mechanism effectively reduced the AI's aversion to confrontation by decreasing penalties over time, encouraging more aggressive tactics.

After updating the reward function, the effects were significant. Following the initial training, the AI dominated the first round of matches. However, as Street Fighter II features a fixed set of initial moves by the opponents in the first round, finding the optimal solution was straightforward. From the second round onwards, where the opponents' initial moves vary randomly, the AI's performance was less satisfactory, achieving only a 38% win rate in best-of-three matches.

To address these challenges, we restructured the training environment in two ways: 1) by using random initial moves from both the first and second rounds for training, and 2) by training across entire best-of-three matches, excluding the intermission screens. Method 1 achieved a win rate of 58%, while Method 2 led to a remarkable

improvement, achieving a win rate of 99%.

2 The base environment and reward function

To avoid "reinventing the wheel," the highly successful reinforcement learning API, Gym, was employed to establish the gaming environment. The initial phase of training focused on the first round of the game, where the opponent exhibits a predetermined set of initial moves. The environment will take an input of the action order and output the image and reward, which are calculated by the reward function(RF).

Like many other project selections, we also use the package stable-baselines3, which is based on PPO[1], to control the training process. The CNN[2] is used to analyze the image outcome from the game environment. Each step will be during 6 Rendering Screens (One Action). The model uses the last 32-step history as a memory.

The straightforward reward function is

$$\begin{aligned} +10: &\quad \text{win} \\ -10: &\quad \text{loss} \\ +1: &\quad \text{when the opponent loses health} \\ -1: &\quad \text{when the agent loses health} \end{aligned}$$

After 3 million iterations, the model still didn't get the right way. As the fig1 showed, the average lens of the steps is always increasing, from 200 to 300, and the reward also increases but too slowly. The model just tries to stall the game, avoiding penalties by dodging and waiting for an opportunity to increase chances of attacking. It termed as "cowardice"

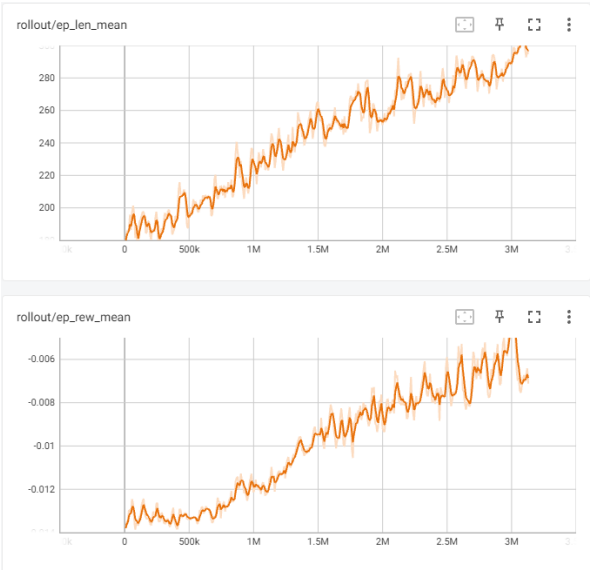

Figure 1: trained result with straightforward reward function

3 Optimaiz the Reward Function

the "DIAMBAR" and "Mitigating Cowardice for Reinforcement Learning" give us spiration. They redifine the reward funcion, make it flexisble and correlated with the health of the opponent and agent. This disgin of the reward function control the scalar of the reward and panishment. The situation that the model tries just to escape will be reduced.
The DIAMBAR focus the current change of the health, the reward function is:

$$\begin{aligned} +\delta hp_{opponent}: &\quad \text{opponent loses health} \\ -\delta hp_{agent}: &\quad \text{agent loses health} \end{aligned}$$

The Decay focus the end status of the game, use the exponnent to reduce the panishment and reward, the reward function is :

$$\begin{aligned} +\text{full } hp_{agent}^{\frac{\text{current } hp_{agent}}{\text{full } hp_{agent}}}: &\quad \text{win} \\ -\text{full } hp_{agent}^{\frac{\text{current } hp_{agent}}{\text{full } hp_{agent}}}: &\quad \text{loss} \end{aligned}$$

[1]Proximal Policy Optimization (PPO)[1] is a reinforcement learning algorithm designed for more stable and efficient policy updates by optimizing a specific objective function. It uses a truncated probability ratio to prevent too large updates, enhancing the robustness and performance of the algorithm.

[2]Convolutional Neural Network (CNN)[2] is a deep learning architecture especially effective for processing image data. It captures hierarchical spatial and temporal relationships in data through convolutional layers, making it highly effective in tasks such as image recognition, video processing, and natural language processing.

We took the combina of the both reward function and add a coeffiecent α, which is 3 in the training process, to involve the model been more aggressive. Our reward function is:

$$
\begin{aligned}
+\alpha \cdot \delta hp_{opponent} &: \text{opponent loses health} \\
-\delta hp_{agent} &: \text{agent loses health} \\
+\alpha \cdot \text{full } hp_{agent}^{\frac{\text{current } hp_{agent}}{\text{full } hp_{agent}}} &: \text{win} \\
-\text{full } hp_{agent}^{\frac{\text{current } hp_{agent}}{\text{full } hp_{agent}}} &: \text{loss}
\end{aligned}
$$

The trained result is shown in Figure 2. The steps start to reduce from 2 million to 3 million, and the reward has a huge increase at the same time. The reward converges at 3 million with 1 (the maximal reward is 176(full health) $\cdot$ 3 $*$ 2/1000(normalization) = 1.056), and the steps converge in 50. The model found the solution to kill the first-round game.

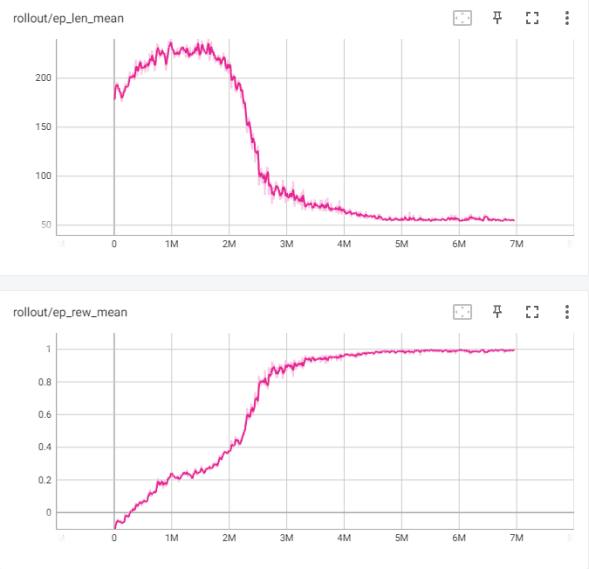

Figure 2: trained result with optimized reward function

4 compete the entire match

In Street Fighter II, you must win with the rule "best of three". From the second round, the opponent's initial movement is not stable; it is random from a set of movements. The model trained only with round one can not handle this. The 6.5 million steps model, which kills the first round in a very short time, loses the second match every time. It is highly customer for the first round, not general at all. When we back up the model to 2.5 million steps(average reward = 0.68), which is still not overfitting for the first round, the result is still not satisfied. The win rate of this model is only 38%.

To address these challenges, we first use a random initial start from both the first and second rounds for training. The trained result is still not good. The reward grows slowly. Even after 10 million steps, the reward is still not above 0.5 (the maximum is 1.054). The curve has a big zic zac, which is because the environment is too random. The win rate finally reached 68%.

Another way to generalize our model is to use the entire match to train it. The theoretical maximal reward points is 2.108 with two wins, or 2.465 one loss and 2 wins. The second maximal value of the reward is quite not possible to achieve, so we ignore this situation. The model solves the first round at 4 million, and the model domain the match starts at 5 million steps. In the end, the 10 million steps got a win rate of 98% with an average reward of 1.745.

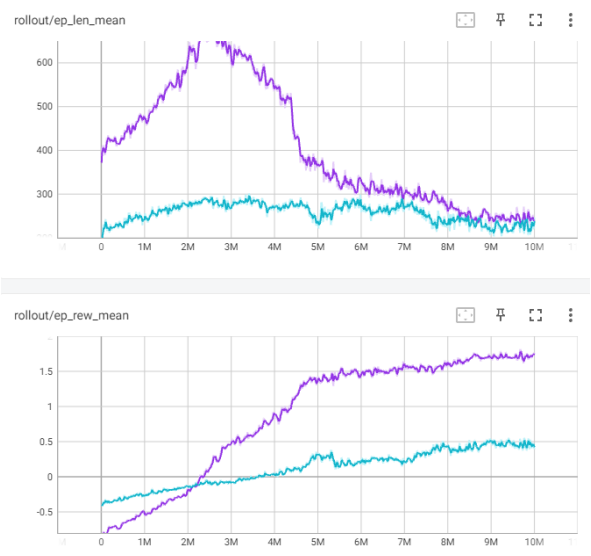

Figure 3: general model trained curve
puple line with entire match
blue line with random round

In the first round of fixation, the model is

able to quickly learn the behavioral pattern of the opponent and efficiently build a strategy to react to this behavioral pattern. This initial fast learning provides a solid foundation for the model to maintain better performance in subsequent training, even in the face of more randomness.
Although the "random" model faces different starting rounds in each training session, which theoretically helps to enhance the generalization ability of the model, in practice, frequent random initial conditions may lead to instability problems in the learning process. In contrast, the "ENTIRE" model starts with a fixed first round, which helps the model to establish a more stable learning path and thus shows more adaptability and stability when encountering stochastic conditions in the subsequent rounds.

5 Conclution

In this project, we tested the impact of different reward functions on AI training and found that a continuous and distinctive reward function is crucial for model training. Additionally, we attempted various model generalization strategies. By training with data from entire matches, we successfully raised the model's win rate to 98%. This approach allowed the AI to comprehensively learn and adapt to different stages of the game, especially in subsequent rounds where opponent behavior becomes more random. In contrast, the training method using random rounds faced issues with unstable learning and low learning rates. These challenges highlight areas that require further exploration and improvement in future research. Overall, a well-designed reward function and comprehensive training data are vital for enhancing AI performance in complex environments.

References

[1] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv preprint arXiv:1707.06347.

[2] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278-2324.

[3] DIAMBRA Arena, "A New Reinforcement Learning Platform for Research and Experimentation", The valuable summary of the experience in setting hyperparameters for deep reinforcement learning models in fighting games, which was of great help to the training process of this project.

[4] Steve Bakos, Heidar Davoudi *Mitigating Cowardice for Reinforcement Learning Agents in Combat Scenarios* Mitigating Cowardice in Reinforcement Learning, The "penalty decay" mechanism proposed in this paper effectively solved the "cowardice" problem (always avoiding opponents and not daring to even try attacking moves).