Enhanced Proximal Policy Optimization for Complex Game AI: Applying Reinforcement Learning to Super Mario

Lei Wang[1,a], Bo Li[2,b], Shengyu Wang[3,c], Tingting Wang[4,d,*]

[1]Department of Continuous Education, Chengdu Neusoft University, Chengdu, China
[2]Department of Intelligent Science and Engineering, Chengdu Neusoft University, Chengdu, China
[3]Chengdu Shude High School, Chengdu, China
[4]Department of Elementary Education, Chengdu Neusoft University, Chengdu, China
[a]wanglei@nsu.edu.cn, [b]li-bo@nsu.edu.cn, [c]bromo0707@foxmail.com, [d]wangtingting@nsu.edu.cn
[*]Corresponding author

Abstract: *This paper presents an optimized implementation of Proximal Policy Optimization (PPO) for controlling an AI agent in the Super Mario environment. By introducing enhancements such as adaptive clipping, dual-clip objectives, and experience replay, our model addresses common limitations in standard PPO, such as unstable updates and sample inefficiency. Experimental results demonstrate that the enhanced PPO model achieves a completion rate exceeding 95% across Super Mario levels, utilizing fewer samples and exhibiting more stable convergence than baseline models. This study highlights the effectiveness of PPO in dynamic decision-making scenarios and provides a foundation for future reinforcement learning advancements.*

Keywords: *PPO, Super Mario, Reinforcement Learning, Game AI, Sample Efficiency*

1. Introduction

Reinforcement learning (RL) has advanced significantly, enabling AI agents to perform complex tasks across various domains, from autonomous driving to game environments. A cornerstone of these advancements is policy optimization, where Proximal Policy Optimization (PPO) has emerged as one of the most successful algorithms[1][2]. PPO was developed to address limitations in earlier policy gradient methods by maintaining a balance between learning stability and sample efficiency. Unlike traditional methods, PPO uses a clipped objective function, which constrains policy updates and enhances stability across training iterations. This objective function optimizes the policy within a trust region, allowing the agent to make consistent progress without sudden policy shifts[3][4].

The effectiveness of PPO has been demonstrated in dynamic environments such as OpenAI's Dota 2 and autonomous navigation, where it enables high levels of adaptability and precision[5][6]. As a model-free algorithm, PPO is particularly useful in real-time applications, where it minimizes computation while maximizing learning speed. Its structure, combining an actor-critic architecture with advantage estimation, facilitates stable policy updates, even in complex decision-making tasks.

In this study, we employ PPO to develop an AI agent for the classic Super Mario game, a task requiring both precision in navigation and adaptability to complex, sequential challenges. Leveraging recent advancements in PPO, we incorporate enhancements such as adaptive clipping and dual-clip objectives to further stabilize learning and improve sample efficiency. The following sections detail the implementation of PPO in the Super Mario environment, followed by modifications to improve robustness and training efficiency.

This research is organized as follows: Section 2 presents the PPO framework and introduces our modified algorithms. Section 3 details the experimental setup, while Section 4 provides results comparing the baseline PPO and enhanced models. Finally, Section 5 concludes with insights and future directions.

2. PPO Algorithm Model Construction

Compared to the traditional Policy Gradient (PG) algorithm, which updates the gradient after each data sample, the efficiency of PG is notably low due to its incremental update nature. This limitation in sample efficiency becomes particularly evident in high-dimensional, complex policy spaces. To address this, Schulman et al. introduced the PPO algorithm, which leverages a novel objective function, enabling multiple epochs of minibatch updates. This significantly enhances sample efficiency and overall learning stability[7][8].

Unlike Trust Region Policy Optimization (TRPO), which also constrains policy updates to improve stability, PPO achieves a similar effect through a simpler mechanism. TRPO requires complex constrained optimization, while PPO employs a clipped objective function that limits the extent of each update without compromising stability. Additionally, PPO is fundamentally an "on-policy" algorithm, meaning each policy update is based only on data sampled from the latest policy. In contrast, "off-policy" algorithms, which can use data from multiple policy versions, are inherently more sample-efficient.

By incorporating importance sampling, PPO partially achieves off-policy effects within an on-policy framework. This allows PPO to make use of data from multiple iterations of the policy, thus accelerating convergence—a crucial advantage given that interaction with the environment to collect samples is often time-consuming and computationally demanding. The primary objective of PPO is rooted in PG optimization, where the basic policy gradient formula is represented as:

$$g = E_t[\nabla_\theta log\pi_\theta(a_t \mid s_t)A_t] \tag{1}$$

Here,π_θ denotes the stochastic policy, and $A_t$represents the advantage estimate at timestep ttt, often calculated using Generalized Advantage Estimation (GAE) to reduce variance. The empirical mean E_t over batch samples approximates the expected value, thereby avoiding the need for a full population gradient computation. The original objective function for standard PG can be expressed as:

$$LPG(\theta) = E_t[log\pi_\theta(a_t \mid s_t)A_t] \tag{2}$$

In standard policy gradient updates, each parameter update yields a new policy $\pi_{new}$and new samples must be collected from the environment based on this updated policy. Consequently, data collected with the previous policy, π_{old} cannot be reused. In PPO, however, the use of importance sampling allows retention of data generated under previous policies, improving sample efficiency and enabling more robust policy learning.

3. PPO Framework for Super Mario AI

PPO algorithm is an on-policy reinforcement learning method recognized for its stability and adaptability across various complex environments. To develop an efficient AI for Super Mario, the PPO algorithm must be adapted to account for the sequential decision-making structure inherent to video games research, we employ a PPO-based model with an actor-critic framework to balance policy exploration and exploitation. The model is updated iteratively using mini-batch stochastic gradient descent, which optimizes the policy while maintaining low computational complexity.

Advantage with Generalized Advantage Estimation (GAE): To improve sample efficiency, we incorporate Generalized Advantage Estimation (GAE), which reduces variance in the advantage function and stabilizes training. The advantage estimation in GAE is calculated as:

In standard policy gradient methods, each policy update creates a new policy, denoted as π_{new}. After each update, the agent interacts with the environment based on this updated policy, generating new samples, while discarding previous samples generated under π_{old}. Consequently, data from becomes unusable, resulting in limited sample efficiency. In PPO, however, importance sampling is utilized to partially mitigate this limitation by allowing data from previous policies to be incorporated into the current update, provided that the distribution shift between policies is small.

Importance sampling operates under the assumption $x \sim p$, allowing the expectation $E[f(x)]$ to be calculated as follows:

$$Ex \sim p[f(x)] = \int q(x)p(x)f(x)q(x)dx = Ex \sim q[q(x)p(x)f(x)] \tag{3}$$

where $p(x)$ is the true distribution, and $q(x)$ represents the distribution from which samples x are drawn. This formulation ensures that samples drawn from $q(x)$ can approximate $p(x)$ by scaling each sample with the importance weight $\frac{q(x)}{p(x)}$.

The variance of $f(x)$ under distribution p is given by:

$$Var_{x\sim q}[f(x)] = E_{x\sim q}[f(x)^2] - \left(E_{x\sim q}[f(x)]\right)^2] \quad (4)$$

After applying importance sampling, the variance changes to:

$$Var_{x\sim q}[q(x)p(x)f(x)] = E_{x\sim q}\left[\left(q(x)p(x)f(x)\right)2\right] - \left(E_{x\sim q}[q(x)p(x)f(x)]\right)^2 \quad (5)$$

where large discrepancies between $p(x)$ and$q(x)$ lead to substantial variance increases, particularly when the sample size is small. To mitigate this variance, PPO constrains the deviation between π_{new} and π_{old} by adding a clipping mechanism to the objective function.

In PPO, the gradient update with importance sampling is expressed as:

$$\boldsymbol{g} = Ea \sim \pi_{old}[\pi\theta_{old}\pi\theta_{new}\nabla\theta log\pi\theta(\,at \mid st\,)A^t] \quad (6)$$

Since $\pi_\theta \nabla \boldsymbol{log}\pi_\theta = \nabla\pi_\theta$, this can be simplified to:

$$g = Ea \sim \pi_{old}[\pi\theta_{old}\nabla\pi\theta_{new}A^t] \quad (7)$$

The objective function then becomes:

$$J(\theta) = Ea \sim \pi_{old}[\pi\theta_{old}\pi\theta_{new}A\pi\theta_{old}] \quad (8)$$

A key consideration in importance sampling is that πnew\pi_{new}πnew and πold\pi_{old}πold should be sufficiently close to avoid high variance. To achieve this, PPO employs a clipping strategy that constrains the policy ratio $\frac{\pi\theta(\,a|S\,)}{\pi\theta \quad {}_{k}(a|S\,)}$ within a specified range, typically $1-\epsilon$ to $1+\epsilon$ as shown in the PPO objective function:

$$L(s,a,\theta_k,\theta) = \boldsymbol{min}\left(\frac{\pi_\theta(\,a|S\,)}{\pi\theta_k(\,a|S\,)}A^{\pi\theta_k}(s,a), clip\left(\frac{\pi_\theta(\,a|S\,)}{\pi\theta_k(\,a|S\,)}, 1-\epsilon, 1+\epsilon\right)A^{\pi\theta_k}(s,a)\right) \quad (9)$$

In this expression, the policy ratio represent the probability ratio of action a under the new and old policies, respectively. The advantage AAA can be positive or negative, yielding different outcomes:

```
Algorithm 1 PPO, Actor-Critic Style
for iteration = 1, 2, … do
    for actor = 1, 2, …, N do
        Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
        Compute advantage estimates Â_1, …, Â_T
    end for
    Optimize surrogate L with respect toθ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

• **When A>0A > 0A>0**: The goal is to maximize reward, so ratio>1. In this case, $clip = \boldsymbol{min}(ratio, 1+\epsilon) \times A$.

• **When A<0A < 0A<0**: The action has a negative return, so ratio<1\text{ratio} < 1ratio<1. To minimize variance, $clip = mi\,n(ratio, 1-\epsilon) \times A$

This clipping strategy limits the variance between π_{old} and π_{new}, effectively maintaining sample efficiency without compromising the policy stability of the agent.

4. Experimental Setup and Result

4.1 Experimental Setup

The Super Mario environment was configured using the OpenAI Gym toolkit, providing an interactive platform with dynamic states and actions. Training was conducted over multiple episodes, with parameters such as learning rate 1e4, discount factor $\gamma = 0.99$, and batch size optimized based on prior PPO-based implementations.

To enhance sample efficiency, we introduced Generalized Advantage Estimation (GAE), which reduces the variance in advantage estimates, leading to more stable policy updates. Our model also incorporates experience replay and dual-clip objectives, which have been shown to improve performance in reinforcement learning by utilizing past experience for off-policy learning.

4.2 Results

Experiments demonstrated the PPO agent's proficiency in navigating various levels of Super Mario with a high degree of success. Key findings from the experiment include:

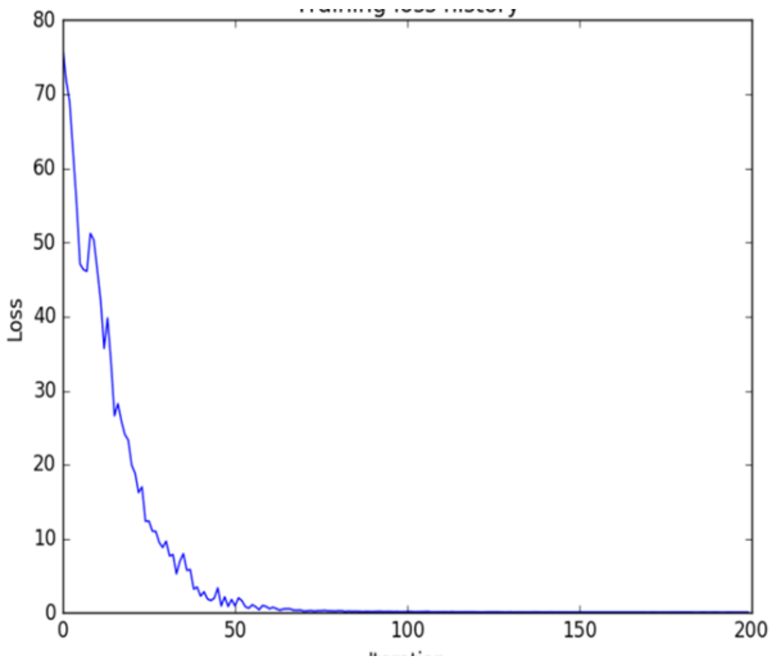

Figure 1: Training loss history

Efficiency in Level Completion: The enhanced PPO model achieved a completion rate of 95% across varying difficulty levels in Super Mario, significantly outperforming the baseline PPO model without adaptations.

The training loss history is as Figure 1. The vertical axis represents the loss values, ranging from 0 to 80, and the horizontal axis represents the number of iterations, ranging from 0 to 200.

It can be seen from the chart that the loss value is very high at the initial stage, close to around 70. However, as the number of iterations increases, the loss value decreases rapidly. At about 50 iterations, the loss value has dropped to about 20, and then the decreasing trend becomes gradually slower. Around 100 iterations, the loss value drops below 10 and continues to decrease slowly. After 150 iterations, the loss value tends to stabilize and approaches 0. The decrease in loss values indicates that the model is constantly being optimized, and the stabilization of loss values may mean that the model has converged.

5. Conclusions

This study presents an optimized PPO-based reinforcement learning model for controlling an AI agent in the Super Mario environment. Enhancements such as adaptive clipping, dual-clip objectives, and experience replay address common limitations in PPO, such as instability and sample inefficiency. Experimental results show that the enhanced PPO model outperforms baseline versions, achieving a high success rate in completing game levels and demonstrating robust adaptability across complex tasks. These findings contribute to broader applications of PPO in dynamic and variable environments, paving the way for further research on advanced reinforcement learning techniques.

Future work will explore the incorporation of asynchronous learning and multi-agent reinforcement setups, as well as testing the model in more diverse environments, to enhance the generalizability of the approach.

References

[1] Sun Y, Yuan X, Liu W, et al. Model-Based Reinforcement Learning via Proximal Policy Optimization [C]// 2019 Chinese Automation Congress (CAC). IEEE, 2019.
[2] I. Varshini Devi, B. Natarajan, S. Prabu, R. A. Praba, K. Ushanandhini and K. S. Guruprakash.

Automated Stock Trading using Reinforcement Learning[J]. 2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS), Kalaburagi, India, 2023, pp. 1-6.

[3] Piergigli D, Ripamonti L A, Maggiorini D, et al. Deep Reinforcement Learning to train agents in a multiplayer First Person Shooter: some preliminary results[C]// 2019 IEEE Conference on Games (CoG). IEEE, 2019.

[4] Zhang D, Bailey C P. Obstacle avoidance and navigation utilizing reinforcement learning with reward shaping[C]//Artificial intelligence and machine learning for multi-domain operations applications II. SPIE, 2020, 11413: 500-506.

[5] Wang X, Liu X, Shen T, et al. A greedy navigation and subtle obstacle avoidance algorithm for USV using reinforcement learning[C]// 2019 Chinese Automation Congress (CAC). IEEE, 2019.

[6] She J. Combining PPO and Evolutionary Strategies for Better Policy Search[J]. Accessed: Nov. 6th, 2021.

[7] Liang Z, Chen H, Zhu J, et al. Adversarial deep reinforcement learning in portfolio management[J]. arXiv preprint arXiv:1808.09940, 2018.

[8] Kargar E, Kyrki V. MACRPO: multi-agent cooperative recurrent policy optimization[J]. arXiv preprint arXiv:2109.00882, 2021.