Mario Fast Learner: Fast and Efficient Solutions for Super Mario Bros

Chen Lin(✉)

South China University of Technology, Guangzhou 510641, Guangdong, China
201920127742@mail.scut.edu.cn

Abstract. Super Mario Bros (SMB) are popular video games. Reinforcement learning has solved various problems including robot control and the game of Go. This article focuses on reinforcement learning methods for Super Mario Bros (SMB) games. Previous methods could solve all available SMB single player open source levels by using reinforcement learning methods. The article summarizes that previous evaluation metrics include reward function, loss function and the arrival of the endpoint flag but these metrics cannot fully judge the quality of the policies. The article analyzes the difficulties for agents to complete SMB levels and points out the problems that need to be solved. To solve the problems, the article proposes a new judging metric for SMB games called 100 recent accuracy. The article propose a solution to speed up the training procedure and improve the experimental results. According to the experimental results, the new solution has good experimental performance under the new evaluation metrics proposed in this article.

Keywords: Super Mario Bros · Reinforcement learning · Policy optimization · Evaluation indicator · Accelerated training

1 Introduction

This article focuses on Super Mario Bros [1] (SMB), which are popular video games since 1985. Researchers have studied SMB games from multiple perspectives. One article in 2009 [2] uses simple single-neuron model and non-linear approximators and gets more than 60% accuracy. Another paper in 2009 [3] introduces a reinforcement learning method for solving SMB games. The method could solve easy levels but not hard ones. The article in 2012 [4] uses grammatical evolution to evolve levels. An article in 2013 [5] compares methods for generating game controllers and finds that a method based on neuroevolution performs best both in terms of the instrumental similarity measure and in phenomenologic AI evolution by human spectators. For mathematical analysis, one article in 2016 [6] proves that solving SMB games is PSPACE-complete and the problem is weakly NP-hard if the levels can be represented using run-length encoding.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13369, pp. 94–103, 2022.
https://doi.org/10.1007/978-3-031-10986-7_8

Reinforcement learning has achieved great application results in various fields and it focuses on solving real-world problems. The most famous reinforcement learning achievement is Alpha Go [7] in the area of the game of Go. The mu-zero method [8] that Deepmind proposed in 2020 could solve atari, go, chess and shogi with superhuman performance. One article in 2019 [9] uses residual reinforcement learning methods for robot control work. Another article in 2022 [10] gives a method for robot hand-eye calibration. For theoretical analysis, there are studies about explainability of Deep Reinforcement Learning [11] and prototypical representations Reinforcement Learning [12].

This article focuses on reinforcement learning methods that can solve SMB games. One article in 2020 [13] use an method based on reinforcement learning and could solve SMB levels with reliable results. Another article in 2021 [14] proposes a method for generating levels and testing levels with the help of reinforcement learning. There is an article in 2021 [15] proposes a new method for reward shaping and uses the method to gain higher reward.

According to the research analysis, researchers have demonstrated that reinforcement learning methods can handle SMB basic problems.
This article poses two problems and tries to solve them:

1. How can different policies be judged when the policy could already bring the agent to the end flag?
2. How to speed up the training process?

Based on the problems this article proposed, the contributions are:

1. The article proposes a new evaluation indicator called 100 recent accuracy. The new indicator represents the accuracy of the policy for getting to the endpoint flag. By observing the accuracy, it is easier to tell the quality of the policy.
2. The article proposes accelerated training techniques to save the training time. The flag factor could indicate if the policy has already brought the agent to the endpoint flag. When flag factor is true, the following training procedure can be accelerated and the accuracy could be higher.
3. The article proposes a new format of the loss function which adds a factor called critic_discount. The new loss function could balance the different components by adjusting parameters. With the flag_factor, a new format of reward function is given.

Section 2 discusses the background. Section 3 presents the details of the proposed method. The fourth section presents the experimental results.

2 Background

2.1 Reinforcement Learning

Reinforcement learning focuses on the interaction between the agent and the environment. It can be assumed that the interactive process fits the framework of Markov Decision Process (MDP). At time t, the agent takes action a_t in state s_t based on policy π and obtains a reward r_t. The cumulative reward is defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ where $\gamma \in (0,1]$ is the discount factor. The goal of reinforcement learning is to find an optimal policy to maximize the cumulative reward. There are value function $V_\pi(s_t) = E(R_t|s_t)$ to calculate the average cumulative reward in state s_t and action value function $Q_\pi(s_t, a_t) = E(R_t|s_t, a_t)$ to calculate the average cumulative reward in state s_t and action a_t. There is advantage function $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$ for evaluating how good the action is with respect to the average action.

There is gym [16] , an open source reinforcement learning environment on Github designed by OpenAI that is popular for reinforcement learning experimentation.

2.2 Super Mario Bros Games

Super Mario Bros Games [1] originates in 1985 and are popular games in the world. There is a github edition SMB environment [18] based on gym environment [16]. The gym-super-mario-bros environment is written in the Python programming language and can be used on Windows and Linux. The environment includes 32 single player levels.

2.3 Leading Reinforcement Learning Methods

The leading reinforcement learning methods are the methods that based on proximal policy optimization theory. The original method [17] was proposed in 2017. The key concept of PPO is the clipped surrogate objective:

$$L^{clip} = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between the new probability $\pi_\theta(a_t|s_t)$ and the old probability $\pi_{\theta_{old}}(a_t|s_t)$. PPO methods are robust and it is easy to tune PPO parameters in different environments.

2.4 Problems of Previous Methods

Previous work has solved 31 levels of gym-super-mario-bros [18] where the only failed stage 7-4 is incomplete in the original environment. However, there are problems about the previous methods.

First, the agent could get to the endpoint flag but its performance is not easy to measure. Therefore, new evaluation criteria should be given.

Second, parameters need to be tuned in different stages of SMB games. It is hard to find the optimal parameters for different stages. Whatsmore, it is not clear whether parameters need to be added.

Third, it takes a long time to train the policy even with multiprocessing techniques. The main reason is that the sampling process of SMB games takes a long time.

Therefore, it is necessary to propose a new judging indicator for SMB policies. The proposed solutions are shown in the next section.

3 Proposed Methods

To solve the aforementioned SMB problems, there are three parts in the proposed solutions, namely, recent accuracy evaluation, accelerated training skills and target function update.

3.1 Use Accuracy Metrics

The goal of SMB games is to reach the endpoint flag. However, recent studies [13–15] focus on two aspects:

1. Whether the agent could get to the final flag or not.
2. How to make the reward higher.

Based on such considerations, it is necessary to add an accuracy indicator.

The proposed method adds 100 recent accuracy to judge the accuracy of the policy. The formula is:

$$accuracy = length(a)/100 \tag{2}$$

where a is a queue that contains the episodes that the agent could reach the endpoint flag during the last 100 episodes. Formally, the formula is as follows:

$$localflag = \begin{cases} True & \text{reached the flag in the current episode} \\ False & \text{did not reach the flag in the current episode} \end{cases} \tag{3}$$

It is indeed trivial in supervised learning, but novel in SMB games with Reinforcement Learning methods of recent years. For accuracy, the proposed methods use it in two parts: Training goal and reward shaping.

1. Use it as the training goal. The goal is to make the accuracy higher during the training process. And by observing the 100 recent accuracy, it is easier to judge whether the policy is getting better or not. The accuracy indicator is used in the test during the experiments.

2. Add accuracy to reward function. We add the accuracy information to the reward function in specific episodes that the agent could reach the flag. The reward function for that episode is:

$$reward = reward + min(accuracy, int(local\_flag\_get)) \tag{4}$$

By adding the accuracy component, it could make the agent learn faster for successful episode.

3.2 Accelerated Training Solution

It is obvious that the agent should reach the final flag as fast as possible. In reality, balancing the ability to reach the end with enhanced exploration is the key to successful training procedure. In previous studies, clipping the gradient of the norm is a technique via pytorch to avoid gradient explosion. However, according to the baseline results of 100 recent accuracy, mario can reach the endpoint flag but the following procedure shows that mario may not tend to get to the flag. To achieve the goal, we add a flag check point.

$$flag = \begin{cases} 1 & \text{has reached the flag} \\ 0 & \text{has not reached the flag} \end{cases} \tag{5}$$

With the flag information, the proposed methods have two points of improvement:

1. Use the flag information to determine the clip grad norm parameter used during training procedure. The flag information is used as clipping parameter to clip grad norm:

$$max\_grad\_norm = 1 + flag * flag\_factor \tag{6}$$

   where flag_factor is a hyperparameter to be tuned which determines how fast the policy should learn after specific successful episode.
2. Use the flag get information to limit the local steps sampling procedure.

To balance the exploration and exploitation, the baseline method uses local steps threshold which regulates the multiprocessing procedure. For each episode, the sampling procedure ends when the agent reaches maximum local steps in previous methods. In the proposed methods, the local steps factor decreases after the agent could drive the agent to the endpoint flag. The formula is as follows:

$$local\_steps = num\_local\_steps * (1 - flag * flag\_factor\_steps) \tag{7}$$

The goal of this formula is to speed up the sampling process. The flag_get_factor could be adaptive and changes in different environments.

The algorithm procedure is shown in Algorithm 1.

Algorithm 1. Fast SMB algorithm

```
Input: network parameters $\theta_{old}$
Output: new network parameters $\theta$
 1: for $i \in [0, max\_episodes - 1]$ do
 2:     local steps changes according to flag_get condition
 3:     for $i \in [0, local\_steps - 1]$ do
 4:         Sample from environment
 5:     end for
 6:     for $i \in [0, num\_epochs - 1]$ do
 7:         for $j \in [0, batchsize - 1]$ do
 8:             Optimize loss function wrt $\theta$ according to flag_get condition
 9:         end for
10:     end for
11: end for
```

3.3 Target Function Update

The loss function in previous methods [17] is composed of actor loss and critic loss with additional entropy loss:

$$Loss = Loss_{actor} + Loss_{critic} - \beta * Loss_{entropy} \tag{8}$$

However, it is not a good idea to treat actor loss and critic loss equally due to their difference. Based on this analysis, we use a new loss function format:

$$Loss = Loss_{actor} + \phi * Loss_{critic} - \beta * Loss_{entropy} \tag{9}$$

In the loss function, L_{actor} means the actor loss, L_{critic} means the critic loss and $L_{entropy}$ the entropy loss respectively. The algorithm adds a new item called critic discount to balance the actor loss and critic loss. There is no need to add another actor discount factor because there is already β in the loss function. The parameters can be better tuned in different environments due to its robustness.

4 Experiments

4.1 Baseline with Accuracy Check

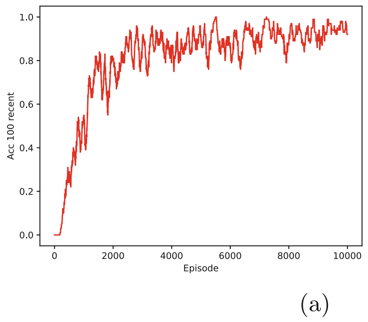
The basic experiment protocol is to use PPO and GAE. There is already an article using PPO and GAE [13] as mentioned before. The article has given clear description of training the agent with PPO methods and is open sourced on Github. The code the experiment uses is based on the paper.

The experiments includes different worlds and different stages. The experimental levels include world 1-1 and 1-2 in World One and 2-1 and 2-2 in World Two. The basic experiment settings are shown in Table 1.

Table 1. Basic experiment settings

Multiprocess	gamma	lr	critic_discount	num_batch
8	0.9	1e−4	1	500

The experimental results of 1-1 and 1-2 with clipping parameter of 1 are shown in Fig. 1. The experimental results of 2-1 and 2-2 in world 2 are shown in Fig. 2.



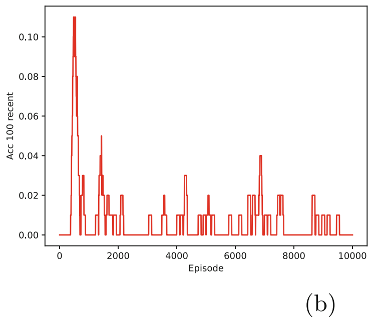
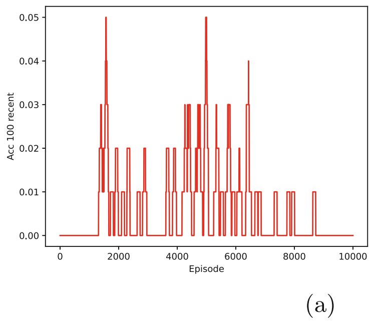


Fig. 1. 1-1 and 1-2 original results



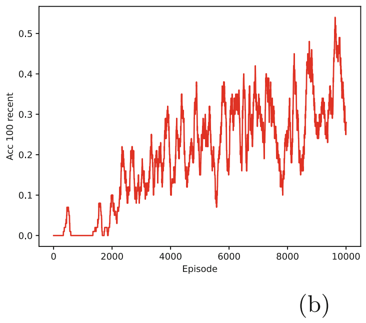


Fig. 2. 2-1 and 2-2 clip 1 results

Table 2 shows the differences in settings for different experiments.

Table 2. Experiment settings

Number	World	Stage	Clip grad norm	Other settings
1	1	1	1	Base
2	1	2	1	Base
3	2	1	1	Base
4	2	2	1	Base

The experimental results show that the initial scheme has driven the agent to reach the endpoint flag. But the results are not satisfactory due to the poor accuracy of recent episodes.

4.2 New Method

According to the algorithm, the flag indicator is used in various ways.

For better experimentation, the flag_factor is set to 19 of 1-1 and 1-2 which means that the maximum gradient norm is 20 after the policy could reach the flag. The experiment settings are shown in Table 3. The results are shown in Fig. 3. The results of 4-1 and 5-1 are shown in Fig. 4. The results of 6-1 and 7-1 are shown in Fig. 5. From the experimental results, the new method does not reduce the number of episodes required to reach the endpoint flag for the first time. The method we propose improves the agent's ability to consistently reach the endpoint flag. Based on the accuracy results, the policy works better after it obtains the ability to reach the endpoint flag.

Table 3. New experiment settings

Multiprocess	gamma	lr	critic_discount	num_batch
8	0.9	1e−4	0.5	500

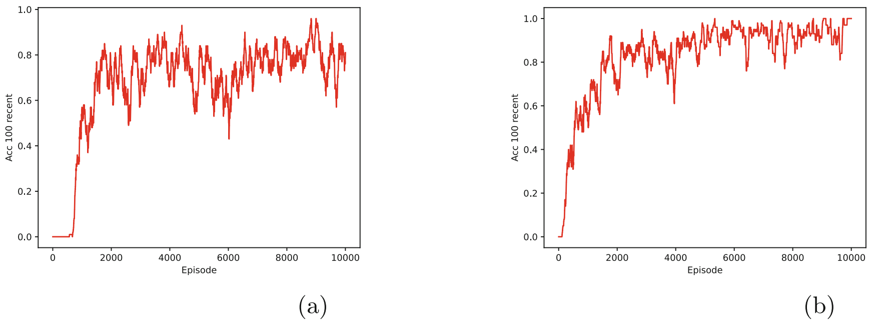
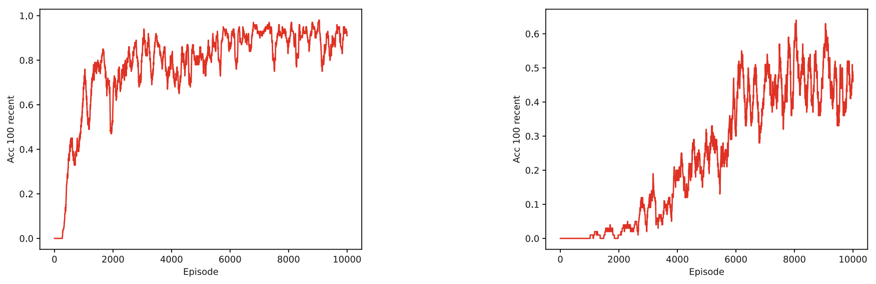
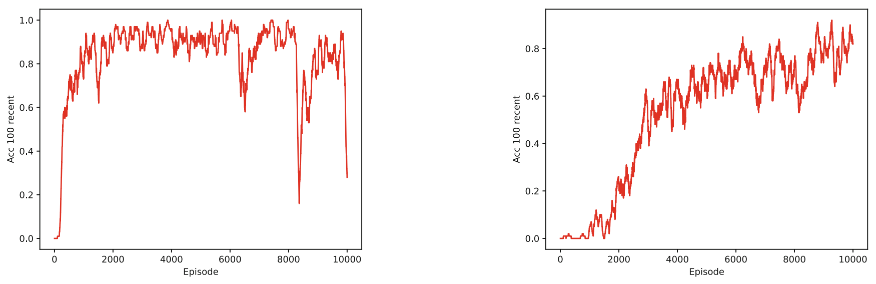


Fig. 3. 1-1 and 1-2 clip 20 results



(a) (b)

Fig. 4. 4-1 and 5-1 clip 20 results



(a) (b)

Fig. 5. 5-1 and 6-1 clip 20 results

On the other hand the parameter $flag\_factor\_steps$ is set to 0.9 in the experiment of the new method. And it could save 10% time during the sampling process after the flag get factor is True.

5 Conclusion

This paper analyzed the problems of the previous SMB solutions. The paper proposed a new evaluation indicator named 100 recent accuracy for SMB games solutions. The new indicator was the key judging metric for the policies. The new indicator was used in neural network parameters update, local steps control and reward function reshaping. The experimental results indicated that the proposed methods could make the agent continue to get to the endpoint flag more frequently and speed up the training process. Future researches are needed to better apply the accuracy information to the training process.

References

1. Bros, S.M.: Nintendo entertainment system. Developed by Nintendo, Nintendo (1985)
2. Pedersen, C., Togelius, J., Yannakakis, G.N.: Modeling player experience in Super Mario Bros. In: 2009 IEEE Symposium on Computational Intelligence and Games, pp. 132–139 (2009). https://doi.org/10.1109/CIG.2009.5286482
3. Togelius, J., Karakovskiy, S., Koutnik, J., Schmidhuber, J.: Super Mario evolution. In: 2009 IEEE Symposium on Computational Intelligence and Games, pp. 156–161 (2009). https://doi.org/10.1109/CIG.2009.5286481
4. Shaker, N., Nicolau, M., Yannakakis, G.N., Togelius, J., O'neill, M.: Evolving levels for Super Mario Bros using grammatical evolution. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 304–311. IEEE (2012)
5. Ortega, J., Shaker, N., Togelius, J., Yannakakis, G.N.: Imitating human playing styles in Super Mario Bros. Entertain. Comput. **4**(2), 93–104 (2013)
6. Demaine, E.D., Viglietta, G., Williams, A.: Super Mario Bros. Is harder/easier than we thought. In: Demaine, E.D., Grandoni, F. (eds.) 8th International Conference on Fun with Algorithms, FUN 2016. Leibniz International Proceedings in Informatics (LIPIcs), vol. 49, pp. 13:1–13:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016). https://doi.org/10.4230/LIPIcs.FUN.2016.13. http://drops.dagstuhl.de/opus/volltexte/2016/5880
7. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
8. Schrittwieser, J., et al.: Mastering Atari, Go, chess and shogi by planning with a learned model. nature **588**(7839), 604–609 (2020)
9. Johannink, T., et al.: Residual reinforcement learning for robot control. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 6023–6029. IEEE (2019)
10. Zhang, R., Lv, Q., Li, J., Bao, J., Liu, T., Liu, S.: A reinforcement learning method for human-robot collaboration in assembly tasks. Robot. Comput. Integr. Manuf. **73**, 102227 (2022)
11. Heuillet, A., Couthouis, F., Díaz-Rodríguez, N.: Explainability in deep reinforcement learning. Knowl. Based Syst. **214**, 106685 (2021)
12. Yarats, D., Fergus, R., Lazaric, A., Pinto, L.: Reinforcement learning with prototypical representations. In: International Conference on Machine Learning, pp. 11920–11931. PMLR (2021)
13. Zhang, N., Song, Z.: Super reinforcement bros: playing Super Mario Bros with reinforcement learning (2020)
14. Shu, T., Liu, J., Yannakakis, G.N.: Experience-driven PCG via reinforcement learning: a Super Mario Bros study. In: 2021 IEEE Conference on Games (CoG), pp. 1–9. IEEE (2021)
15. Bougie, N., Ichise, R.: Fast and slow curiosity for high-level exploration in reinforcement learning. Appl. Intell. **51**(2), 1086–1107 (2020). https://doi.org/10.1007/s10489-020-01849-3
16. Brockman, G., et al.: OpenAI Gym (2016)
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
18. Kauten, C.: Super Mario Bros for OpenAI Gym. GitHub (2018). https://github.com/Kautenja/gym-super-mario-bros