CS482/682 Final Project Report Group 4

Reinforcement Learning in Different Games

Conghao Xiong/cxiong5, Nan Huo/nhuo1, Luchao Qi/lqi9, Shuxian Zhao/szhao35

1 INTRODUCTION

Nowadays, we care about how agents interact with the environment and perform some tasks under given circumstances. Since there are plenty of restrictions which are hard to satisfy in the real world, such as the high temperature, high pressure, and vacuum, so because of this, people tend to conduct experiments in virtual environments instead of in the real world. Therefore, how to embed this realistic situation into a virtual scene and then test the performance of our agent in this setting is of vital importance.

We implemented reinforcement learning method in toy problems at first: Cartpole Game[1] and Mountain Car Game[2]. Also, we trained agent to complete Super Mario Bros in OpenAI Gym automatically without a single death. After achieving this, we would then try to investigate on getting higher scores, completing faster, upgrading status of Mario Bros, breaking more number of bricks, smashing more number of enemies, etc.

Previous studies implemented Deep Q Network (DQN), Q learning, or some policy-based methods in Super Mario Bros [1]. But we found out that there are several new methods proposed recently for reinforcement learning, such as Deep Double Q Network (DDQN), Proximal Policy Optimisation (PPO) [2]. Our results show that PPO performs better than DQN in Super Mario Bros so far.

2 METHODS

Generally, all the training data are generated by the process that the agent plays the game many times automatically, which is called experience replay technique [3]. As for the environment, we use the environment provided by OpenAI Gym.

2.1 Cartpole & Mountain Car Game

The core method for CartPole is DQN [4], which uses a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output and DDQN [5], which uses two neural networks to learn and predict what action to take at every step; for Mountain Car is Deep Imitation Learning which is a deep network trained by episodes which are generated by pretrained agent. We finally implemented with several techniques such as skip connection. Then we use this trained agent to play the games lots of times to calculate the successful rates[3].

A Training We will focus on deep network of our model here. The network structure is: 3 linear layers with Relu as its activation function and 1 fully connected layer. And we have implemented several techniques: 1)Dropout, 2)Batch Normalisation, 3)Tuning batch size, learning rate. And we use the technique called 'experience replay'. That is we firstly pre-train our agent by Q learning(baseline method) to have certain successful rates of our agent. Then we use the pretrained agent to generate plenty of episodes as our training data which labeled with successful passing game or not. The deep network of the agent will trained from these training data.

B Evaluation We take the time of staying balanced for Cartpole Game as rewards, which means the longer the cartpole can stay balanced, the more rewards the game can get. The rewards policy of Mountain Car Game is -1 for each step until it successfully climb to the mountain which means the highest reward is -2, since the car can't climb up with single movement. We evaluate the Game's performance based on its successful rate, rewards. Intuitively, the method with higher successful rates and rewards are better.

2.2 Mario Game

A Introduction Super Mario Bros is divided into eight worlds, each of them containing four levels. In this case, word 1, level 1 (world 1-1) was chosen to train and test our agent.

B Evaluation In reinforcement learning, the model (called agent) interacts with environment by choosing from a set of possible actions (action space) that cause either positive or negative rewards from the environment. And we decided also to take scores as another metric for performance of the model. The scores are directly accessed by our program and this score does not include the flag bonus scores. Fig 1 shows that difference in the game clock between frames (**unit time in game**), velocity of agent, and death of agent are used for rewards calculation.

2.2.1 DQN for Mario

In figure 2, game states are defined as 4-D tensors containing the last four game states as grayscale matrices to detect agent

[1] https://gym.openai.com/envs/CartPole-v0/

[2] https://gym.openai.com/envs/MountainCar-v0/

[3] successful rate is the rate of successfully finishing the game (for instance, for one episode in Cartpole Game, the cartpole can stay more than 200 seconds is a success episode)

"movement" on the screen. For this reason, we implemented convolutional neural network (CNN) to train the agent.

2.2.2 PPO for Mario

PPO stands for Proximal Policy Optimisation. The structure for PPO is shown in figure 3, we take the current frame and the previous actions as input. We feed the current frame into four consecutive convolutional layers, and then concatenate it with the other side, which is passed through a linear layer. Then we perform sequence modelling on it, since it involves the previous actions. Then we feed the result into different linear layer to get values and next-step action separately.

A Training We have tried several techniques: 1).Dropout, 2).Batch Normalisation, 3).Skip Connections, 4).Fine Tuning, 5).Change the Structure of Network. Besides this, we try to use different set of actions for Marios, namely Simple Movement and Complex Movement [6]. Simple Movement does not allow complex left movement like action "left + A" or "left + B", in which "A" and "B" mean the button on Joypad, but Complex Movement does allow that.

3 RESULTS

3.1 Cartpole and Mountain Car

The successful rate of Q learning is remarkably lower than Deep network methods like DQN, DDQN. The maximum stable time for Q learning in Cartpole Game is 208 seconds. For DQN, maximum stable time is 499 seconds and the average stable time can be 232 seconds(shown in figure 4). For DDQN, higher success rate and more stable results can be shown(shown in figure 5). For Mountain Car, after changing the architecture of our network, we found that the training loss of original network(shown in figure 6) converge worse than the training loss of our better network(shown in figure 7) with several techniques implemented. The results and comparison are shown in figure [8,9].

3.2 Super Mario Bros

For DQN model, the final passing rate is 41% and passing time is 92 unit time in level 1. The highest score that the DQN model can reach is 700. For PPO model, the passing rate is 78% and the passing time is 65 seconds. The highest score is 1300. The reported scores do not contain the flag bonus scores. The result figure is shown here on figure 10.

Simple Movement vs Complex Movement: Passing time is relatively higher for Complex Movement. For passing rate, the Complex one has better result. Both perform better than DQN method on metrics (passing time, scores, etc.) we set.

Also we investigated some special tasks like getting more coins instead of just passing the game. We achieved this by resetting the reward mechanism to give more priority on getting more coins. We ended up with a score of 3200, which is much higher than the original score (1300).

4 CONCLUSION/ DISCUSSION

4.1 Cartpole and Mountain Car

We tried different layers of our deep network and tried different techniques like different optimizers, different weight decay values. Then we performed ablation analysis as follows: (1) Generally, the performance becomes worse when having too many layers. We think that it's easier to go to a local optimum and encounter gradient vanishing with too many layers; (2) Adam optimizer converges faster than SGD. However, SGD is easier to compute in each iteration because it has smaller batch size. (3) As for the weight decay value (one of parameters in the optimizer), when it becomes larger, the training process will become harder to converge. The purpose of weight decay is to avoid over-fitting, so it will definitely add more bias to the model.

Then we compared DDQN and DQN for Cartpole Game. DDQN uses the same network structure as DQN (shown in figure 4 and figure 5). DDQN uses two identical neural network models. One learns during the experience replay, just like DQN does, and the other one is a copy of the last episode of the first model. The Q-value is actually computed with this second model. The index of the highest Q-value can be found from the main model and that index can be used to obtain the action from the second model. This makes the results of DDQN to be more stable than the results of DQN. [7].

4.2 Super Mario Bros

We noticed that the PPO method was better than DQN in all the metrics like passing rate and the completion time, etc. It took much less time for PPO to complete world 1-1 compared to DQN. In other levels like world 1-2, world 1-3, etc., both models performed weaker compared to their performance in world 1-1, respectively. However, PPO still performed better than DQN in other levels on average. We believe that, since PPO model preserve previous information during iteration, PPO model is more robust than DQN model in this case. As for aforementioned DDQN, previous studies have constructed DDQN model for Super Mario Bros [4]. DDQN could be more customized regarding rewards calculation and agent action control. However, it costs around 150 unit time to complete world 1-1 in this case, which is slower than DQN. We believe this is the trade-off between getting more coins and smaller completion time. For further analysis, we could improve the network architecture by changing input layers to process more frames so that we could capture more information about the "movement" of agent. Also we could perform some special tasks like passing the game as soon as possible, maintain the status of fireball by changing rewards calculation. For instance, we could impose penalties on time consuming or we could increase rewards when Mario agent becomes "Fire Mario agent"[5] in game, etc.

[4] https://github.com/davidchiou/DDQN_Mario

[5] Fire Mario Bros is an unique state upgraded from Mario Bros who have the unique ability to throw fireballs.

5 ACKNOWLEDGEMENTS

We really enjoyed the fun of course as well as the exciting project. We are grateful to Dr. Silvio Ami. Thank you so much for giving us valuable course and all the useful advice you offered to us. We really appreciate your help. It's your advice that help us to choose 2 toy problems (i.e. The Cartpole Game and Mountain Car Game) to warm up instead of going straight to the difficult 'Reinforcement Learning of Super Mario Game' and suggested us to focus on Deep Learning part. Otherwise we may waste too much time on non-deep-learning things and may stuck in training Super Mario Game ended up learning far less than what we get so far.

REFERENCES

[1] Diego Perez et al. "Evolving behaviour trees for the mario ai competition using grammatical evolution". In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2011, pp. 123–132.

[2] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[3] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[4] Hao Yi Ong, Kevin Chavez, and Augustus Hong. "Distributed deep Q-learning". In: *arXiv preprint arXiv:1508.04186* (2015).

[5] Brian Ning, Franco Ho Ting Ling, and Sebastian Jaimungal. "Double Deep Q-Learning for Optimal Execution". In: *arXiv preprint arXiv:1812.06600* (2018).

[6] Tom Schaul et al. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[7] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

6 APPENDIX

Figure 1: Definition of Rewards for Super Mario

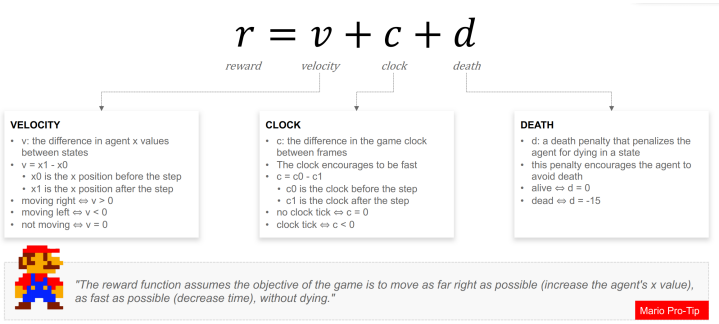

Figure 2: Simple Reinforcement Learning with CNN

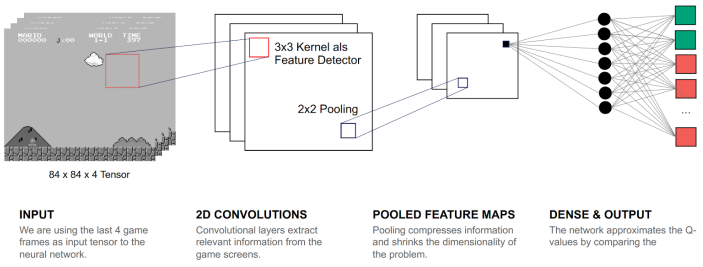

Figure 3: Architecture for PPO

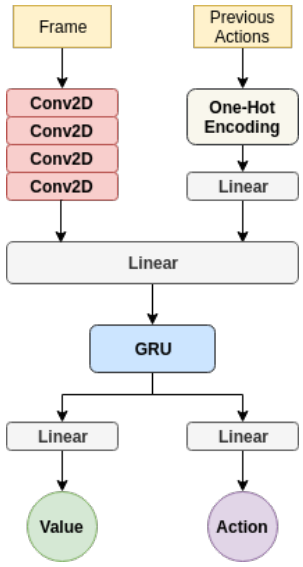

Figure 4: Scores for DQN on Cartpole Game

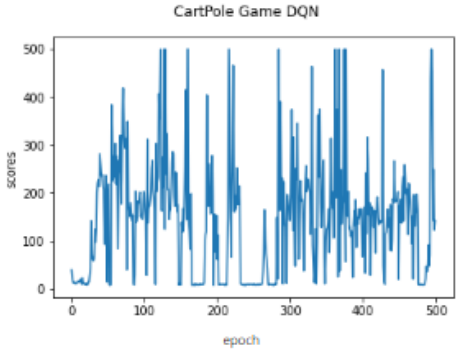

Figure 5: Scores for DDQN on Cartpole Game

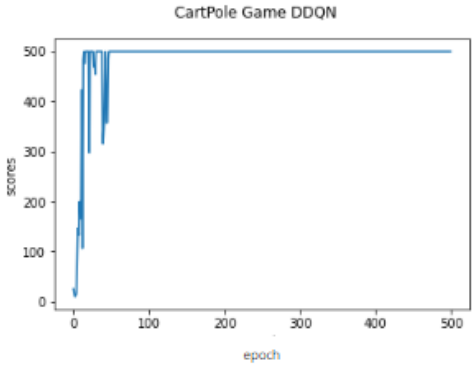

Figure 6: Original model: 3 layer neural network

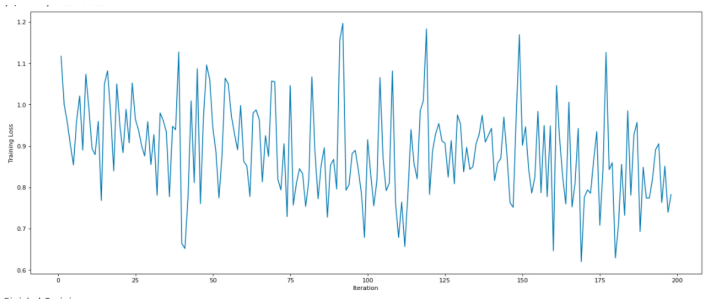

Figure 7: Better NN model

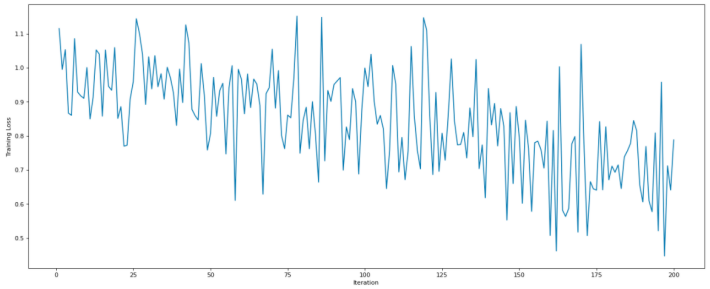

Figure 8: The Comparison of CartPole Game

Cartpole Game		
Model	Score/Stable Time (Max 499)	Success Rate
Q-learning	Max 208	0.37
Deep Q learning	Max 499	0.63
Double Deep Q learning	Max 499	0.95

Figure 9: The Comparison of Mountain Car Game

Mountain Car Game		
Model	Score (Max -2)	Success Rate
Q-learning	-151	0.5
Deep Imitation Learning	-85	0.86
Better Network	-57	0.94

Note: The score of Mountain Car Game will -1 *with each more step taken until success.*

Figure 10: The Comparison of Super Mario Game

Mario Bros Game			
Model	Passing Rate	Passing time (unit time)	Score
Deep Q Learning	41%	92	700
Proximal Policy	78%	65	1300