Deep Reinforcement Learning for 2D Flappy Brid Game

Yueyang Li[1, *]

[1]University of Wisconsin-Madison, Madison, WI, United States, 53703

ABSTRACT: In recent years, the prevailing of application of Deep Reinforcement Learning have granted the traditional game AI training a brand-new perspective. Google's Alpha Go agent might mark the beginning of the trend. While many 2D games have been researched for effective trained agent to gain extraordinary performance, Flappy Bird is among, perhaps, the most popular one that could demonstrate the effectiveness of trained AI. This research has successfully trained a efficient agent using Deep-Q network that could outperforms its human counterparts. Although previous trainings have also granted successful results, due to the optimization in the memory array to store previous states, the training time has been largely reduced, which is useful for future agent training optimization.

1. INTRODUCTION

Flappy Bird is a game that the players could only use their interactive devices to control the altitude of their game agents to void hitting certain objects in the interface. Players could only choose to do two options at each instant: whether to control their agents to go up by flapping or go down naturally by not pressing any button. The "up" option grant minor constant distance for the agent to go up, whereas the "down" option grants constant distance shift downwards by gravity.

Today, the blooming in the recent research in deep neural networks where multiple layers of nodes are implemented and utilized to construct more complicated modeling of raw datum. Since the Flappy Bird game is an infinite thread that might not terminate until certain conditions are triggered, it would be impossible to hard program all the steps in traditional way to control the behavior of the game avatar [1]. However, training agents that could take adaptive actions in each state might sufficiently solve the problem.

This research is based on the recent application of the Q-learning in training of various traditional 2D game agents. The goal of this project is to train efficiently an agent that could successfully and perfectly play the Flappy Bird game, that could prolong the existence of the avatar in the game in optimal time intervals. Moreover, the trained agent should be robust enough to not make unnecessary actions and maximize its survival time.

Training the agent to maximize its survival interval is challenging, since it is only provided with pixel information and the current score that the agent earns. Other information regarding the location of the pillars (pipes), the shape of the bird, or the shape of the pillars are not well informed, bringing exceedingly challenges for the agent to learn to avoid collision with the pillars.

2. METHODS

2.1. Model

This section will discuss the methods that are applied in this project. In each step, the agent could only take two options: to flap or not flap. In our model, it is represented using a parameter e, where e = 0 represents simply the bird drop, and e = 1 represents the information that the agent does not take any options. State in this model is represented by a list of collections of pixels (frames) and previous actions that the agent adopts are also considered as the input of this model. Detailed representation of the state could find it explanation in Eq.(1) [2].

$$s_t = (x_{t-hl+1}, \epsilon_{t-hl+1}, \dots, x_{t-1}, \epsilon_{t-1}, x_t) \quad (1)$$

where x_t is the input represented as pixels at *t* moment, e_t is the action taken at the t moment, and hl represents an dynamic array of most recent pixel input that we are keeping tracking of, which is a design that aims to reduce the space used by the unnecessary storage of states from t = 1 until all other consecutive states [3].

2.2. Q-Learning

The goal of reinforcement learning is to maximize the total reward. Therefore, Q-learning, might be the optimal choice for this scenario, which can be represented as the equation Eq.(2) [4].

$$(s, a) = r + \gamma \max_{a'} Q_i(s', a') \quad (2)$$

where s^0 and a^0 are the state and action for the next frame. *r* is the reward, 7 is represented as the discount factor, and Q_i (s^0, a^0) is the Q-value for (s, a) at i^{th} iteration. A sequence of input is applied to the function, which is *st,*

*Corresponding author. Email: egoistheresyus@outlook.com

*et, st+**1**,* et+i, and our current task is to remap this function, and transfer this function as a Convolutional Neural Network, and get updated accordingly in Eq.(2).

$$L = \sum_{s,a,r,s'} \left(Q(s,a;\theta) - \left(r + \gamma \max_{a'} Q(s',a';\theta^*) \right) \right)^2 \quad (3)$$

$$\nabla_\theta L = \sum_{s,a,r,s'} -2 \left(Q(s,a;\theta) - \left(r + \gamma \max_{a'} Q(s',a';\theta^*) \right) \right) \nabla_\theta Q(s,a;\theta) \quad (4)$$

One could easily notes that the Eq.(3) and Eq.(4) here are the loss function and the gradient function of Eq.(2) which is utilized to approximate this model [5, 6]. Here, it should be noticed that * is the parameter that is not renowned for the Q-value function. The discount factor y is set to 0.85, whereas transition probabilities along with gain are kept occlusive from the agent [7].

In addition to the Q-learning approach that is applied in this research, the setting of the reward in this game is also of vital importance. Intuitively, the reward should be set to the scores that the agent earn currently. However, this will lead to the fact that if the agent dies immediately after the startle of the training, this reward is exactly the same if the agent dies when it hits the first pillar. Therefore, it is better for us to divide the reward into two parts including Reward of Alive and Reward of Hitting Pipe [8].

2.3. Deep Q-Network

The Q-function in the experiment is approximated by a neural network, which takes 186 x 186 pixel representation of images as its input. Design of this convolutional neural network is divided into two parts, where the first layer comprises of a convolutional filter of size 16 x 16 with stride 8. The Algorithm of this Deep-Q Network could be observed as below in Algorithm 1.

Algorithm 1

Initialize the lists of previous states
Initialize the Q-value function
Initialize the θ^* value to be θ
for game count = 1→ maxGames **do**
for frame 1 → t do **do**
extract x_t from raw pixel data and update s^t with e_t
if game terminates **then**
else
$y_i = r_i + \gamma max_{a'} Q(s', a'; \theta^*)$
end if
end for
end for

2.4. Stability of the algorithm

The algorithm is proven to be robust in overall sense. The process that the author used to increase the robustness of my algorithm is by applying the target network Q(a, t), where this formula is a ”shadow” isomorphism to the original Q-network. Every ticks/frames the program updates its DQN, the corresponding isomorphism, Q(a, t) is also updated accordingly, see in Eq.5 [5].

$$y_i = E_{s' \sim \varepsilon} \left[r + \gamma \max_{a'} \hat{Q}\left(s', a'; \hat{\theta}_{i-1}\right) \mid s, a \right] \quad (5)$$

By applying this method, the overall stability of the system is ensured.

2.5. Data Pre-processing

As stated previously, the data of the program that is using is a 186 x 186 grid (pixels) that represents the history input. The first step of prepossessing is a convolution layer with 16 filters of a size 16 x 16 with stride of 8. Next, the processed data is again processed according to another layer of 128 layers with 8 x 8 stride. Finally, the data is generated with respect to a single output of 512, where the greatest output Q-value will be taken.

3. EXPERIMENT AND RESULT

3.1. Parameters

The Flappy Bird game is running at a constant frame rate o 30fps here, and we are only keeping track of 5 previous states in the lists. As discussed previously, the discount factor Y is set to 0.85 and rewards of a single training: Reward of Alive = 1.2 and Reward of Hitting Pipe = -0.2.

3.2. Performance

The trained agent has excellent performance and sometimes could exceed the performance of human players. The result of the trained agent is compared with the results with human players and a baseline average performance. The average approach simply does the flap action every 2 frames to maintain the current altitude of the bird avatar. This approach is adopted because the uniform distribution of pillars.

The results of the training are listed in the Table 1 and Table 2. Table 1 is the comparison between the baseline approach and the performance of human along with trained agent. Table 2 is the comparison of the best performance of trained agent under different difficulties

level, where the trained agent still gained comparatively stable results.

Table 1. Score of performance comparison between human, baseline, and trained agent

Comparison			
difficulty	baseline approach	human	trained agent
easy	inf	inf	inf
medium	inf	inf	inf
extreme hard	1.3	20.23	92.3

Table 2. Score of performance, but in highest score comparison

Comparison			
Difficulty	Baseline approach	Human	Trained agent
Easy	inf	inf	inf
Medium	15	inf	inf
Extreme hard	3.2	78.65	224.45

3.3. Training time

Although our model might give the results of how well the agent performs after trained by the Deep-Q Network, the training time is still a significant factor that objectively evaluate our model. Even intuitively speaking, longer training time would yield better results [9], this is not valid in my testing situation. This paper uses the training rate as a standardized comparison factor to evaluate the effectiveness of the training, the results are shown in the Table 3. As it could be observed from the table, the effectiveness is positively related with the times of iterations of training. It is not the case that the training performance is reached with increasing times of training iterations. One potential solution of this issue might be increasing the complexity of the current model being applied [10].

Table 3. Score of Deep-Q Network effectiveness

Comparison			
Times of training	Easy	Medium	Hard
10000	1200	53.7	0.1
90000	1034	34.5	14.37
200000	341.3	41.5	67
300000	1451.6	2598.2	73.45

4. CONCLUSION

The Flappy Bird game is successfully played by the trained agent in this experiment. The trained avatar could reach a constant high score in different levels of difficulties even compared with human players, with stability and accuracy. However, the effectiveness of training is not guaranteed as more training does not grant better performance, which needs to be improved in the future research. This model could be enhanced by establishing more complicated feedback system and reward mechanism so that it could yields better and more stable performance in the future.

AUTHORS' CONTRIBUTIONS

This paper is independently completed by Yueyang Li.

ACKNOWLEDGMENTS

Thanks for professor Pietro Lio' for providing me constructional ideas and guidance that I could realize my idea, without their help, this paper could not be done smoothly.

REFERENCES

1. V. Mnih, K. Kavukcuoglu, D. Silver, et al., Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602, 2013.
2. N. Appiah and S. Vare, Playing flappybird with deep reinforcement learning, 2018.
3. A. Brandao, P. Pires, and P. Georgieva, Reinforcement learning and neuroevolution in flappy bird game, in Iberian Conference on Pattern Recognition and Image Analysis, Springer, 2019, pp. 225-236.
4. K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, A survey of deep reinforcement learning in video games, arXiv preprint arXiv:1912.10944, 2019.
5. K. Chen, Deep reinforcement learning for flappy bird, 2015.
6. J. Fan, Z. Wang, Y. Xie, and Z. Yang, A theoretical analysis of deep q-learning, in Learning for Dynamics and Control, PMLR, 2020, pp. 486-489.
7. L. S. Pilcer, A. Hoorelbeke, and A. Andigne, Playing flappy bird with deep reinforcement learning [c]

IEEE Transactions on Neural Networks, vol. 16, no. 1, pp. 285-286, 2015.

8. T. Hester, M. Vecerik, O. Pietquin, et al., Deep q-learning from demonstrations, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.
9. A. Ganesh, J. Charalel, M. D. Sarma, and N. Xu, Deep reinforcement learning for simulated autonomous driving, 2016.
10. T. Vu and L. Tran, Flapai bird: Training an agent to play flappy bird using reinforcement learning techniques, arXiv preprint arXiv:2003.09579, 2020.