Flappy Bird Game Based on the Deep Q Learning Neural Network

Jiarui Gu [1,*,†], Yunhao Guo [2,†], Yushan Lam [3,†], Ziyou Benjamin Pu [4,†]

[1] Jinling highschool, Nanjing, China

[2.]Shanghai Wenlai International School, Shanghai, China

[3.]Shenzhen Hankai high school of mathematics and science, Shenzhen, China

[4.]Adcote School Shanghai, Shanghai, China

* Corresponding Author Email: 17011010127@stu.suse.edu

[†]These authors contributed equally.

Abstract. In the past decades, with the rapid development of technology, people discovered ways for machines to learn. Machines can be trained to recognize things, play games, create sounds, or find the best choices. There are several models and tools to train the machine to make the machine learn through supervised or not supervised, even independent learning. Through neural networks or other methods, the machine can be trained for many episodes. Making awards or punishments can make the machine construct ways to decide the most valuable solution. Among these functions AI can achieve, game is the most direct platform to apply machine learning, since awards and punishments can be applied very easily: through the score. In this study, we choose to use a Deep-Q-Learning Neural Network (DQN) to train our AI to achieve our goal: Using AI to play Flappy Bird through deep learning. Our task is different from other game training, such as navigating an AI to find the best solution in different choices. In this task, the player (i.e. bird) cannot get any award or punishment through a single action, but it can get an award by passing each obstacle. The goal of the AI is to pass as many obstacles as it can, by choosing to fly upward or idle.

Keywords: Deep Q, Flappy Bird, Machine Learning.

1. Introduction

Reinforcement learning is an important machine learning method, which can control the individual's autonomous actions in a certain environment, and through the interaction with the environment, the individual can continuously improve his or her own behavior. At present all kinds of games involve the use of some Artificial Intelligence (AI), some game developers think that, from simple to pursue and escape all movement patterns, as well as neural network and Genetic Algorithm (GA) is a part of game AI, AI general been in the game industry includes the limited state design and fuzzy state design of two parts, The guiding idea is to create the illusion that the AI is superior in the simplest way possible, using the least amount of resources. The use of reinforcement learning in games is important because games can quickly generate a large amount of state-action-reward data, which is a good training material for reinforcement learning.

For example, RLCard supports a variety of card game environments, such as Black-Jack, Leduc Hold'em and UNO etc. RLCard is proposed to make reinforcement learning better applied in incomplete information games and promote the development of reinforcement learning in multi-agent, large state and action space, and sparse reward fields. The interface is easy to get started with, especially for those with no knowledge of Game Theory. The authors also provide a single-agent environment where other players are programmed to use pre-trained models [1]. Also have an RL method including automatic battle feature and data skipping technique is proposed for implementing combat games. Through the experience of self-fighting, three different forms of agents can be drawn and fought against each other. This paper suggests improvements upon this self-adversarial algorithm by diversifying the opponent pool, and the proposed data skimming technique can increase the data efficiency and help exploration in large Spaces [2].

For example, Michelle McPartland and Marcus Gallagher applied reinforcement learning to 3D First Person Shooter (FPS) games [3]. In FPS games, the bots are required to deal with a wide array of tasks, such as find pathways, pick or use the objects in the surrounding environment, or attack the players. However, without the RL program, the bots can only move or direct the pathway as the parameters are designed, making their actions predictable to the player and making the game less realistic and interesting [4]. The group designed two tasks for the AI: navigation and item pickups in a maze-like map for bots [5]; bots combat against the others. The maps of the game have walls, items, and spawn points, and the bots need to identify the obstacles, find the items and pick them up, or shoot at enemies. In that program, the team used tabular Sarsa algorithm with eligibility traces for action-selection for the AI training. When a state-action pair occurs, the eligibility was set to 1.0. In the navigation test, a small reward (i.e. 0.000002) was given to the bot if it did not collide with the wall for each action, and a small penalty (i.e. -0.000002) was given to the bot if it collided with the wall. In the trials, the bot was set up in a 50mx50m map. The bot was outfitted with six sensors, divided into two groups of three. One sensor was immediately in front of the bot, one was 20 degrees to the left of the bot, and one was 20 degrees to the right of the bot. The first three sensors were used to identify whether or not there were any obstructions in the bot's path. The result provided by the obstacle sensors is either 0 or 1, with 0 indicating that there is no obstacle, 1 indicating that there is an impediment near to the bot (within four meters), and 2 indicating that there is an obstruction further away from the bot (within ten meters). The second set of sensors was used to determine the location of things in respect to the bot. For the combat task, the bot needs to fight against a state machine-controlled AI. The bot needs to maximize the kills and lower the deaths to get the highest possible award. The team trained the bot’s movement 5000 trials and the combat for 20 trials. In the end, the bot was able to navigate the maps and shoot at enemies.

In this study, the center of research we would like to dig more deeply into is about flappy bird with deep Q network. Deep Q network is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. And Flappy Bird is a simple game that the player controls a bird to get through obstacles. By clicking the screen, the bird will fly upward, and it will receive a constant downward force. The goal is to let the bird get through as much obstacles as the player can.

2. Methodology

2.1. The motivation of Flappy Bird

Flappy Bird is an arcade-style mobile game, developed by a Vietnamese, Dong Nguyen. The game is simple to play, which is controlling the bird without hitting the pipes and keeps flying. The game was first released in 2013 and soon become the most popular free games in App Store in 2014 [6].

Dong Nguyen is a programmer from Vietnam, birthed in 1985. At his age of 19, he found that the online games at that times on phones were too complicated. The game developer was inspired by the act of bouncing a ball, and tried to continue bouncing it for as long as possible. Hence, in 2012, he created a game titled “Flappy Bird” in only 3 days. In the game, Nguyen introduced the infinite time duration and the score of a gameplay is roughly measured by the time that the bird keeps flying [7].

2.2. Gameplay

During the game player controls a bird, attempting to fly up and down without hitting the green pipes that have equal-size gaps at random heights. The bird will fall down automatically until the player taps to the screen, then it will rise. Whenever the player passes through between the gap of pipes, the players will be awarded 1 point. At the same time, colliding with pipes or fall down to the ground will end the game immediately [8].

2.3. The DQN Algorithm

We present an arbitrary problem, where an agent has to choose a series of actions {a}, which will maximise some reward r. At the center is the realization that if a function Q(s,a) is able to determine the maximum reward given the current state s and the current chosen action a, we can simply select the policy that maximises Q(s,a). This is the namesake of Q learning [9].

Suppose we already have the optimal algorithm, and we select action a regarding state s, leading to a new state s'. Since this is the optimal algorithm, the reward for s 'is also the largest when compared to all other states that directly follow s. This leads to the Bellman equation:

$$Q^*(s,a) = r_s + \gamma Q^*(s',a') \tag{1}$$

Where, r_s is the reward directly given due to action a. Due to the law of diminishing returns, future rewards are discounted by a factor γ every episode. We can also use the equation to find the function Q by using the equation to iteratively update Q:

$$Q_{i+1}(s,a) = r_s + \gamma Q_i(s',a') \tag{2}$$

This can be interpreted as starting from a final state and consecutively backtracking to earlier states.

$$\begin{aligned} Q_i(s,a) = r_s + \gamma Q_{i-1}(\mathrm{s'},\mathrm{a'}) = r_s + \gamma r_{s'} + \gamma^2 Q_{i-2}(\mathrm{s''},\mathrm{a''}) \\ = r_s + \gamma r_{s'} + \gamma^2 r_{s''} + \gamma^3 Q_{i-3}(\mathrm{s'''},\mathrm{a'''}) \ldots \end{aligned} \tag{3}$$

Since the series of states s, s', s …''are likely to bear few similarities between the individual policies that the Machine-learning algorithm generates, naïvely implementing this method may fail to converge within an adequate time limit in the general scenario. Therefore, we use a neural network to approximate Q as they are universal function approximators. We arrive at the standard Q-learning algorithm.

To supplement this algorithm, past experiences of the agent are stored in a memory stack, termed a "replay buffer". Between episodes of direct gameplay, the agent is also trained with data from the replay buffer, which helps reinforce past lessons lea3rnt. The memory stack also presents an opportunity for the agent to explore the environment.

2.4. Application of the DQN Method to Flappy Bird

We obtained our Flappy Bird environment from the Github of Lin Yen Chen [10]. The environment also includes several handles, enabling us to pass data to the game and retrieve data from the game more easily. Due to time limitations, we could only run the Deep Q-learning apparatus for about 250 games. The Deep Q-learning apparatus is trained with consecutive images of the game's visual interface. The original game's background was removed.

For the network itself, a convolution of stride 4 is first applied, followed by a 2x2 max-pool layer. Finally, 2 convolutions of strides 2 and 1 were performed, resulting in 1600 data points, which are directly weighted and combined. The result is interpreted by applying a relu activation function. This final result can then be used to control the movement of the Bird.

After each convolution, ReLU was applied to cleave insignificant data. The mean squared error function was used to evaluate the performance of the network, with error reduction optimized by the Adam Optimizer available in TensorFlow.

3. Results and discussion

3.1. Experimental data

For each game played by the DQN agent, the final score S_F obtained by the agent was recorded. The recorded data can be seen in Figure 1, where it is plotted against the number of games played, N. When N reaches 263, the agent has noticeably improved, as we see in the generally increasing trend.

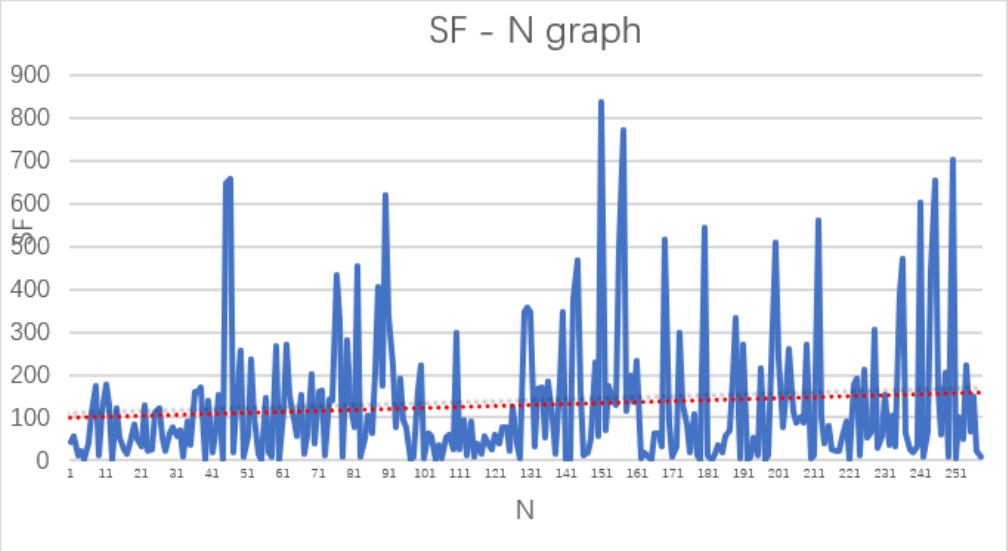

Figure 1. Final Score s_F against number of games N

As we have been provided with sufficient environments, libraries and models, we now conduct assays to discover the best Machine Learning apparatus by which acceptable and sustainable errors may be obtained. Applying our apparatus to the Flappy Bird environment, we find that the training process is unstable but still manages to progress.

The Figure 1 displays a general increasing trend, and the concentration of high-S_F attempts (defined as $s_F \geq 450$) generally increases with the number of games. The maximum s_F obtained was 839. However, dips to low scores such as 0 or 5 were not infrequent.

The mean value of s_F was 130.95. The median was 70, which is significantly lower than the mean. The 1st quartile was 23 and 3rd quartile was 172.5. The 1st quartile is very low compared to the 3rd quartile and the average, showing a high variance in the score. The standard deviation of the scores is 156.4, which also shows that the data is wide spread. The mode was interestingly 1 appearing 7 times in total.

3.2. Analysis of the experimental results

To further analyse the results, we apply a linear regression to the S_F – N graph reveals a slightly upwards-sloping line. The gradient is 0.2277, or about an increment of 1 in S_F for every increment of 5 in N. This is a reasonably good result.

The figure also displays clear oscillatory movement. The score seems to generally increase, eventually reaching a relatively high score before suffering a sharp drop. This reflects the learning curve of the DQN: The DQN updates its strategy in nearly the same fashion each game, which initially leads to higher scores and higher confidence. However, once the strategy is pushed beyond a limit, its fitness sharply drops. The DQN then changes its strategy in a more substantial manner and begins to improve upon the newer strategy as before, effectively starting a new learning curve for the DQN. However, for each such learning curve, the score generally rises across a period.

The oscillatory method of learning explains the median being much smaller than the mean. For a period, it takes a long time for the agent to learn and stay in the low score, so that the scores are more concentrated around lower values. However, the scores obtained in a period generally improve upon the scores in the previous period. In the long run, the median and average score will become higher while the oscillatory pattern described above will remain.

The result of the network trained shows that, the Convolutional Q-network is able to decide whether the bird should jump with reasonable loss. It would be desirable if the variance of the scores could be decreased, limiting the oscillatory behavior of the agent's performance and hence raising the reliability of the agent. This may require changes to the model.

4. Conclusion

In this work, we proposed using deep Q-learning to train the models to pass as many obstacles as it can, by choosing to fly upward or downwards in the game ”Flappy bird”.

We developed the Image Translation Process and Deep Q-Learning Algorithm to improve the performance of the training model. During the experiments, we remove the game background to allow a faster convergence. At first, in the beginning of learning process, the data is nearly close to 0. After 50 thousand times of continuous experiments and data recording, the bird can fly through the third pipes. As the model can control the birds alive forever, 2 million attempts have been taken, which is about 30h of data experiments. In the future, we plan to adapt the proposed model to serve more reinforcement learning tasks

References

[1] Daochen Z et al. RLCard: A Toolkit for Reinforcement Learning in Card Games [C]. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20) Demonstrations Track.

[2] Oh I. et al. Creating Pro-Level AI for a Real-Time Fighting Game Using Deep Reinforcement Learning [J], in IEEE Transactions on Games, vol. 14, no. 2, pp. 212-220, 2022.

[3] Jones J, Benefits of Genetic Algorithms in Simulations for Game Designers [D]. Thesis, School of Informatics, University of Buffalo, Buffalo, USA, 2003.

[4] Manslow J. Using Reinforcement Learning to Solve AI Control Problems [B], in AI Game Programming Wisdom 2, S. Rabin, (Editor). Hingham, USA: Charles River Media, 2004.

[5] Michelle M. Learning to be a Bot: Reinforcement Learning in Shooter Games School of Information Technology and Electrical Engineering [R], University of Queensland St Lucia, Australia, 2008.

[6] Williams R. What is Flappy Bird? The game taking the App Store by storm [R], The Daily Telegraph. Archived from the original on January 30, 2014. Retrieved January 30, 2014.

[7] Heney E. Chocolate Lab Apps [R]. Archived from the original on February 6, 2014.

[8] Crecente B. Polygon [R]. Archived from the original on June 30, 2015. Retrieved June 13, 2015.

[9] Mnih V. Playing atari with deep reinforcement learning [R]. arXiv preprint arXiv:1312.5602.

[10] Chen L. Deep Learning Flappy Bird [R]. Github, 2022.