Evaluation of DQN and Double DQN Algorithms in Flappy Bird Environment

Zhenyu Chen

Department of Electrical and Electronic Engineering, University of Manchester, Manchester, M13 9PL, United Kingdom
zhenyu.chen-4@student.manchester.ac.uk

Abstract. To solve problems involving sequential decision-making, Deep Reinforcement Learning (DRL) combines the advantages of Reinforcement Learning (RL) and Deep Learning (DL). These fusions are skilled at training agents for video games because they use neural networks to approximate nonlinear functions. The Deep Q Network (DQN), a key algorithm in this field, tends to overestimate Q-values despite its effectiveness. The Double Deep Q Network (Double DQN) was presented to address this. The Kera's framework in PyCharm is used in this study to examine the practical application and comparative analysis of DQN and Double DQN in the Flappy Bird game. To speed up its training, improvements were also made to the DQN model. The enhanced DQN performed better than the traditional DQN but less well than the Double DQN, according to the results. This study delves into the practical application and comparative analysis of DQN and Double DQN in the Flappy Bird game, utilizing the Kera's framework in PyCharm. Additionally, enhancements were made to the DQN model to expedite its training. Results affirmed that the enhanced DQN outperformed the conventional DQN but lagged the Double DQN. The training loss trajectory further substantiated Double DQN's superiority in mitigating overestimation issues.

Keywords: Reinforcement Learning, Deep Learning, Flappy Bird.

1 Introduction

The core of deep reinforcement learning (DRL) is the training of artificial agents to navigate complex decision spaces while interacting with their environments. The fundamental objective is to make decisions that maximize cumulative rewards over time, often with minimal initial knowledge of the environment [1,2]. DRL diverges from traditional reinforcement learning by incorporating deep neural networks to approximate intricate functions that map raw sensory inputs to optimal actions. This capacity enables DRL algorithms to effectively manage high-dimensional and continuous state spaces, rendering them particularly well-suited for real-world scenarios involving raw sensory data, such as images, audio, or text [3,4]. DRL's fundamental goal is to teach artificial agents how to maneuver through difficult decision spaces while interacting with their surroundings. Making judgments that maximize cumulative benefits over time is the

B. H. Ahmad (ed.), *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, Advances in Intelligent Systems Research 180,
https://doi.org/10.2991/978-94-6463-370-2_24

main goal, which is frequently accomplished with little initial environmental knowledge. By using deep neural networks to approximate complex functions that connect unprocessed sensory inputs to optimum behaviors, DRL departs from standard reinforcement learning. DRL algorithms are particularly well-suited for real-world scenarios requiring raw sensory input, such as photos, music, or text, because to their ability to manage high-dimensional and continuous state spaces well [5].

A well-known computer game called Flappy Bird gives players the seemingly straightforward but difficult task of navigating a bird through a series of pipes that are in its way [6]. The bird must pass across gaps created by the separation of these pipes to continue its journey. There are only two possible actions available to players, making the gameplay mechanics simple. They can either tap the screen to slightly raise the bird or they can let it alone and let gravity pull the bird downward. The bird's survival is the main goal, along with avoiding any collisions with the pipes and moving it as far as is practical. The player receives one point for each successfully navigating through a pipe gap, and their score is continuously displayed on a scoreboard. Flappy bird serves as a common example for basic study of applying DRL in teaching artificial intelligence to master video games.

The rest of paper compared the performance of both Deep Q Network (DQN) and Double DQN in the video game Flappy Bird.

2 Method

2.1 Game environment

The open-sourced environment named flappy bird is leveraged for evaluation. This environment is built by OPEN AI GYM and PYGAME library [6,7].

The flappy bird is a video game, in which the frames have high correlations with each. Four frames are stacked as input, and when storing relay memory, this work used four most recently experience is form (s, a, r, st + 1).

There is a trade-off between exploration and exploitation that must be managed because if one concentrates on exploration, it may take too long to identify a suitable course of action or if one concentrates on exploitation, it may become trapped in the local optimum. So, the greedy policy is applied. epsilon ε is set to be 0.1 to 0.0001 for exploration. Since in flappy bird has 4 frames as a state, so the agent will make the decision very often (about 0.03s per once). If the initial epsilon is very high, that will make the bird flap too much time and in that case the efficiency will be very slow.

Since the main goal for the flappy bird is to survive as long as possible, so the reward is set +0.1 for each step (four frames) the bird survives. If the bird is colliding with the pip, the bird will get -1 reward, and for each pip the bird passed, the agent will get +1 reward.

2.2 Q Learning

Both Deep Q-Learning and Double Deep Q-Learning are all based on the Q Learning [8]. Q Learning introduces Q table to store the Q value for each action in each state, and after each state Q Learning use an equation (below) to update the Q function.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma maxQ(s',a') - Q(s,a)] \quad (1)$$

In the equation, α is a parameter know as learning rate, which is decreasing by the time. By applying the bellman equation, the model can draw a Q table with Q value that is already known for each action in every state. For each state, the agent will choose the maximin Q value action, and after that state, update the Q table. If any real reward is not known, all the initial value in that table will be set to zero. Then the agent will choose randomly to collect data. For every update the Q value will be converged close to the real reward.

2.3 Deep Q Learning

In Q learning, the Q table can only handle the finite state, while in the real world, the environment is complex, and many situations could be infinite that Q table cannot handle it [9]. In the case, Deep Q learning is induced a neural network to replace the Q table. Since neural networks have shown successful at simulating non-linear functions, they can be used in Q-learning to model the Q function in infinite states. This work can achieve deep Q-learning by combining a neural network with Q-learning. As a result, a shared loss could be generated and train a parameterized value function.

$$L(\theta) = \mathbb{E}[(r + \gamma max_{a'} Q(s',a',\theta') - Q(s,a,\theta))^2] \quad (2)$$

Now that the mentioned loss function is given, this work can update the weights of the network to roughly approach the Q function using stochastic gradient descent and back-propagation.

2.4 Double DQN

DQN has issues with overestimating approximation of function, yet performing well in DRL which employs high dimensions as input to help agent make choice in large-scale situations [10]. The DQN use the function

$$a_{opt} = argmaxQ(s,a;\theta_t) \quad (3)$$

to choose the max Q value for each action in each state. This will cause the Q value to converge slow because of the overestimation.

Double DQN (DDQN) is an improvement over the standard DQN algorithm. Double DQN addresses the overestimation (DQN) by using two networks, one for selecting the best action (the online network) and another one for evaluating that action (the target network). For two networks, the online network selects what it believes to be the

optimal action mentioned above equation. The target network then estimates the Q-value of taking that action at the next state. The target network:

$$Q(s,a) = r(s,a) + \gamma Q(s', argmax_a Q(s',a)) \quad (4)$$

Q (s, a) is the Q-value of current state-action pair. r is the immediate reward after acting a at the state s. γ is the discount factor. Q' (s', a) is the Q-value of next state-action pair from the target network. argmax Q (s', a) is the action that maximizes the Q-value at next state s', as determined by the online network. Since Q value calculated by target network is not necessarily the highest one, Double DQN can decrease the probability of making sub-optimal overoptimistic choice.

3 Result

In Fig. 1 and Fig. 2, the experiment set x-axis as episode the bird survive and y-axis as cumulated reward the bird can obtain for each game. In DQN diagram, the previous 1500 episodes, the agent perform not well but the overall trend could still be observed that the agent improves its performance as in Fig 2. In Double DQN, it could be observed that the agent performance is more stable, but overall is worse than DQN. There is an improvement, as the training episode is not enough, so the Double DQN is worse.

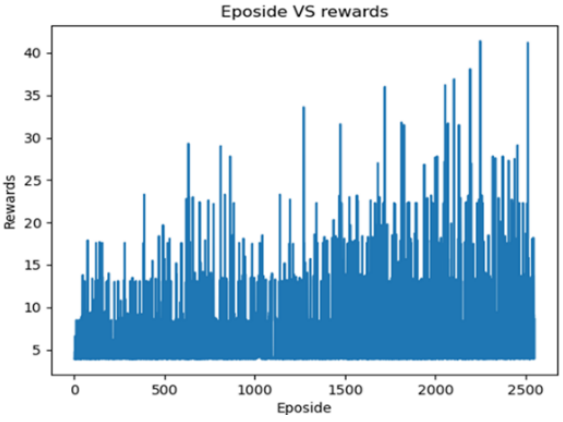

Fig. 1. Performance of DQN at first 3000 episodes (Figure credit: Original).

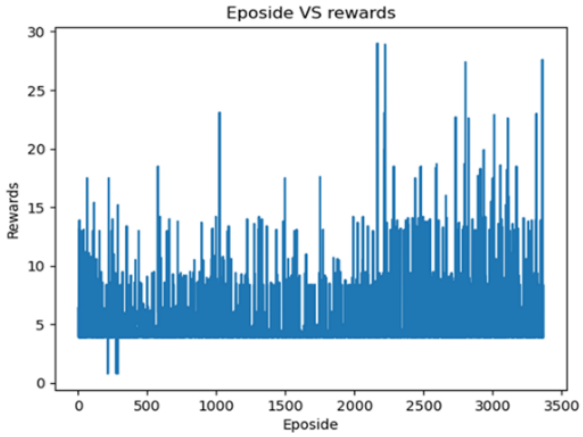

Fig. 2. Performance of DDQN at first 3000 episodes (Figure credit: Original).

However, in the Fig 3 and Fig 4, the training amount increased from 3 hundred thousand to 3 million. Agent in both DQN and Double DQN perform bad in episodes 20000 or more, but at the point between 20000 and 25000, there is a breakthrough for the flappy bird. The plot diagram of Double DQN looks very close to the DQN figure, which has the same trend. However, if comparing carefully of both figures. The Fig 4 converged slightly faster than the Fig 3.

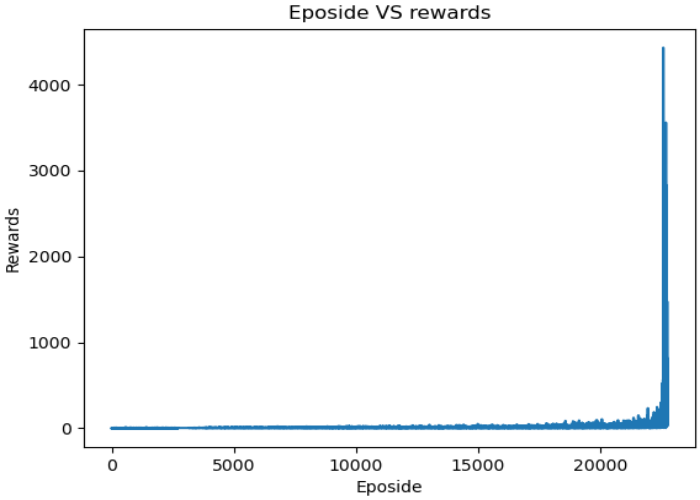

Fig. 3. Performance of DQN at first 30000 episodes (Figure credit: Original).

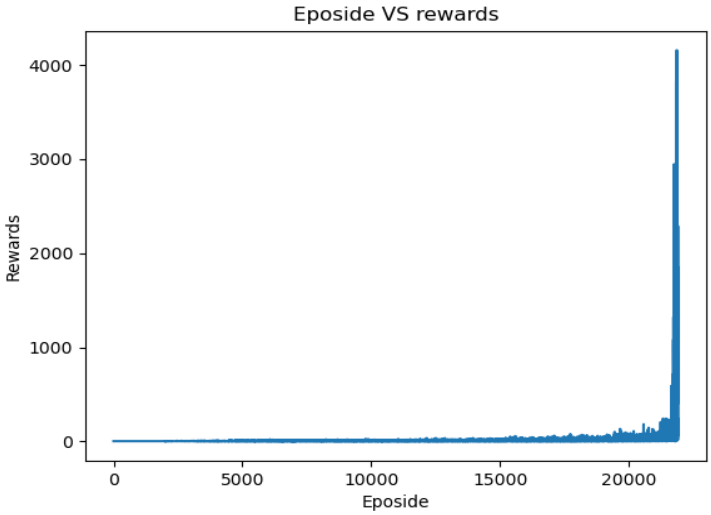

Fig. 4. Performance of DDQN at first 30000 episodes (Figure credit: Original).

4 Discussion

DRL has greatly benefited by the development of the reinforcement learning algorithms DQN (Deep Q-Network) and Double DQN (Double Deep Q-Network). DQN can learn sophisticated tactics in a variety of situations since it approximates the Q-values of actions in each state using a neural network. However, DQN has a propensity to overestimate Q-values, which can result in subpar training practices. By separating the processes of action selection and value evaluation, Double DQN successfully overcomes this constraint, lowering overestimations and stabilizing the learning process. Based on the experiment result, the Double DQN converged earlier as shown in the Fig 4, this shows that the Double DQN can solve the overestimation.

In the future, both DQN and Double DQN hold substantial potential for diverse applications. Their ability to handle high-dimensional state spaces and complex action sets makes them suitable for robotics, autonomous systems, and video game AI.

5 Conclusion

In the project, using DQN and Double DQN to train the agent to play flappy bird is successful, but it needs to train for around 3 million timestep to see a big improve. The result shows that Double DQN can handle the overestimation problem in DQN, but it only converged slightly faster than in DQN. Overall, the DRL is implemented in playing flappy bird. Both DQN and DDQN can make agent a master for playing this game (with big amount of training data to support).

References

1. Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 34(6), 26-38 (2017).
2. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. An introduction to deep reinforcement learning. Foundations and Trends in Machine Learning, 11(3-4), 219-354 (2018).
3. Fang, W., Pang, L., & Yi, W. N. Survey on the application of deep reinforcement learning in image processing. Journal on Artificial Intelligence, 2(1), 39-58 (2020).
4. Latif, S., Cuayáhuitl, H., Pervez, F., Shamshad, F., Ali, H. S., & Cambria, E. A survey on deep reinforcement learning for audio-based applications. Artificial Intelligence Review, 56(3), 2193-2240 (2023).
5. Wang, H. N., Liu, N., Zhang, Y. Y., Feng, D. W., Huang, F., Li, D. S., & Zhang, Y. M.. Deep reinforcement learning: a survey. Frontiers of Information Technology & Electronic Engineering, 21(12), 1726-1744 (2020).
6. Flappy Bird Homepage, URL: http://flappybird.io/, Last accessed 2023/09/01
7. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. Openai gym. arXiv preprint arXiv:1606.01540 (2016).
8. Watkins, C. J., & Dayan, P. Q-learning. Machine learning, 8, 279-292 (1992).
9. Fan, J., Wang, Z., Xie, Y., & Yang, Z. A theoretical analysis of deep Q-learning. In Learning for dynamics and control, 486-489 (2020).
10. Van Hasselt, H., Guez, A., & Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence, 30(1), 1-8 (2016).