# Playing Flappy Bird with Reinforcement Learning

**Douglas Trajano**
Pontifical Catholic University of Rio Grande do Sul - PUCRS
School of Technology. Porto Alegre, Brazil
douglas.trajano@edu.pucrs.br

### Abstract

Flappy Bird is an electronic game created in 2013. The objective in the game is to earn as many points as possible by controlling a bird, without letting it crash into the pipes. The Flappy Bird environment uses OpenAI Gym API. In this work, we will implement reinforcement learning algorithms such as Deep Q-Network (DQN) and Proximal Policy Gradient (PPO) that will be used to train the agent to play the game.

## Introduction

The Flappy Bird was released in May 2013, but it received a sudden rise in popularity in early 2014 becoming a viral hit.

The game was developed by Vietnamese programmer Dong Nguyen. It has simple gameplay, the player controls a bird, attempting to fly between green pipes without hitting them. The Flappy Bird received poor reviews from some critics, who criticized its high level of difficulty and alleged plagiarism in graphics and game mechanics, while other reviewers found it addictive.

Flappy Bird was removed from both the App Store and Google Play by its creator on February 10, 2014. He claims that he felt guilt over what he considered to be its addictive nature and overuse.


Figure 1: Flappy Bird Game



Learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of reinforcement learning (RL). The basic idea behind many reinforcement learning algorithms is to estimate the action-value function. The goal of the agent (powered by a reinforcement learning algorithm) is to interact with the emulator by selecting actions in a way that maximizes future rewards.(Mnih et al. 2013)

We will develop some reinforcement learning algorithms that will be used to train our agents in the OpenAI Gym environment. OpenAI Gym is a toolkit for reinforcement learning research. It includes a growing collection of benchmark problems that expose a common interface.(Brockman et al. 2016)

## Approach

Our technical approach consists of two parts. The first part is the definition of the environment. The environment that will be explored in this project is provided by OpenAI Gym. The second part is the reinforcement learning algorithm, which will be implemented by ourselves.

### Environment

The Flappy Bird environment was developed by Gabriel Nogueira and is publicly available on GitHub [1], it also can be installed using PyPI [2]. It was developed in Python and uses the OpenAI Gym API.

The state observation is composed of the horizontal distance to the next pipe and the difference between the player's y position and the next hole's y position.

Two actions are available: do nothing and jump.

### RL Algorithms

The algorithms are responsible for learning the policy to solve the problem, it will be used to take actions in the environment. The reinforcement learning (RL) algorithms were developed in Python, we developed a base class with a random policy, it also provides the API to implement the RL algorithms. The RL algorithms are:

[1] https://github.com/Talendar/flappy-bird-gym

[2] https://pypi.org/project/flappy-bird-gym/

**Deep Q-Network (DQN)** combines Q-Learning with deep neural networks to let RL work for complex, high-dimensional environments, like video games, or robotics. A critical component of DQN-style algorithms is the memory buffer known as experience replay, it holds the most recent transitions collected by the policy.(Fedus et al. 2020)

Two different approaches of the memory buffer will be developed and tested.

- **Experience Replay (ER)**: The most basic sampling strategy, it uses uniform sampling, whereby each transition in the buffer is sampled with equal probability.(Fedus et al. 2020)
- **Prioritized Experience Replay (PER)**: Extends experience replay function by learning to replay memories where the real reward significantly diverges from the expected reward, letting the agent adjust itself in response to developing incorrect assumptions.(Schaul et al. 2015)

**Proximal Policy Optimization (PPO)** is a policy gradient method that trains a stochastic policy in an on-policy way. Also, it utilizes the actor-critic method. The actor maps the observation to action and the critic gives an expectation of the rewards of the agent for the observation given. Firstly, it collects a set of trajectories for each epoch by sampling from the latest version of the stochastic policy. Then, the rewards-to-go and the advantage estimates are computed to update the policy and fit the value function. The policy is updated via a stochastic gradient ascent optimizer, while the value function is fitted via some gradient descent algorithm. This procedure is applied for many epochs until the environment is solved. (Schulman et al. 2017)

## Implementation

We developed the reinforcement learning algorithms in Python using PyTorch and Tensorflow. The source code of this project is available on GitHub [3]. The code is divided into two parts:

### Trainer

The trainer is responsible for training the agents. In the training process, the environment provides observations and the agent takes actions for each observation. The Flappy Bird environment provides a reward 1 for each time step, except when the bird crashes the ground or the pipe.

### Agents

The agents are responsible for choosing actions for a given observation. We developed a base agent that provides the basic functions that agents should use to interact with the environment, for example, the act function receives the observation state and returns the selected action, in the base agent, the act function provides random actions based on the number of available actions. Both algorithms that we developed extend the base agent. The DQN Agent uses the experience replay (ER) or prioritized experience replay (PER), also a neural network is used to approximate the Q-function, our implementation of DQN also has an epsilon-greedy policy to handle the trade-off between exploration and exploitation. Our PPO Agent uses a Proximal Policy Optimization (PPO), a policy gradient method that uses the actor-critic method to train the policy. PPO Agent also has a buffer that uses Generalized Advantage Estimation (GAE-Lambda). We developed DQN algorithm using TensorFlow and PPO algorithm using Pytorch.

[3] https://github.com/DougTrajano/drl-flappy-bird

## Related work

In (Mnih et al. 2013) the researchers developed the Deep Q-Network (DQN) algorithm. In this study, the agent is trained from the screen images of Atari games and it produced results far above human results. (Alp and Guzel 2019a) trained DQN to play Flappy Bird, the average score for the algorithm is 3.3, and the human score is 4.25. In (Alp and Guzel 2019b) two algorithms were trained: Deep Q-Network (DQN) and Asynchronous Advantage Actor-Critic (A3C), and the results said that the A3C resulted in much faster training because it uses its own reward function. The (Vu and Tran 2020) showed that the SARSA and Q-Learning algorithms can be used to learn the Flappy Bird game. New algorithms of policy gradient methods were introduced in (Schulman et al. 2017), the most relevant method is proximal policy optimization (PPO) that we will implement and test in this work.

## Experiments

In this experiment, we want to see if one of the two reinforcement learning algorithms that we developed can learn to play the Flappy Bird game properly.

### Trainer hyperparameters

The following hyperparameters are used for training all the agents.

- Number of episodes (n_episodes): 60,000
- Early stopping (early_stop): 120
- Maximum number of time steps per episode (max_timestep): None

### DQN Agent

The following hyperparameters are used in the DQN agent:

- Dimension of each observation (state_size): 2
- Quantity of available actions (action_size): 2
- Random seed (seed): 1993
- Hidden units in the network (nb_hidden): (64, 64)
- Learning rate for the optimizer (learning_rate): 0.0005
- Size of the memory (memory_size): 100,000
- Prioritized replay memory (prioritized_memory): False
- Size of the batch to train the network (batch_size): 64
- Discount factor (gamma): 0.99
- Interpolation parameter for target network (tau): 0.001

- Small value used in the priority update (small_eps): 0.03
- Number of steps before updating the target network (update_every): 4
- Use epsilon-greedy action selection (epsilon_enabled): True
- Starting value of epsilon, for epsilon-greedy action selection (epsilon_start): 1.0
- Minimum value of epsilon (epsilon_end): 0.01
- Decay rate for epsilon (epsilon_decay): 0.99995

The training session was not stopped by the early stopping policy, and the agent was trained for 60,000 episodes. Figure 2 shows the training process of the DQN agent. We can see that in 40,000 episodes, the agent achieved the best score (116.23) of the training session, but it isn't enough to stop the training.

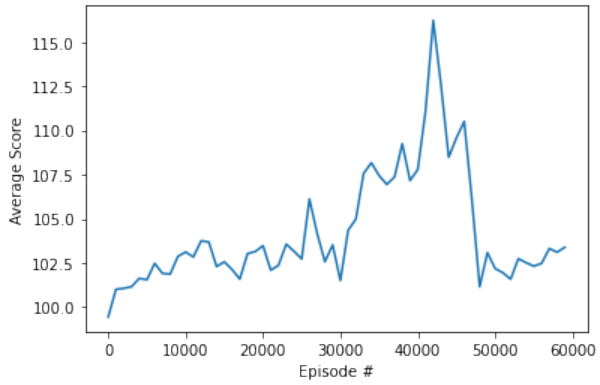


Figure 2: DQN - Average Scores by episodes

The DQN Agent (with 60,000 episodes) was tested in 10 trials as described in Table 1. The **Trial** column represents the number of the test, the **Env score** column represents the scores provided by the OpenAI Gym environment, and the **Skipped pipes** column represents the number of pipes skipped by the bird.

| Trial | Env score | Skipped pipes |
|---|---|---|
| 1 | 150 | 2 |
| 2 | 114 | 1 |
| 3 | 137 | 1 |
| 4 | 137 | 1 |
| 5 | 119 | 1 |
| 6 | 121 | 1 |
| 7 | 120 | 1 |
| 8 | 109 | 0 |
| 9 | 151 | 2 |
| 10 | 114 | 1 |

Table 1: DQN - Results table

It seems that the DQN Agent learned a little bit as in the best case, the bird was able to skip only 2 pipes. The average score was 127.2, and the average number of pipes skipped was 1.1.

We developed the Prioritized Experience Replay (PER) to be used with Deep Q-Network (DQN) algorithm, but it takes a lot of time to train and it is not used in the paper.

## PPO Agent

The following hyperparameters are used in the PPO agent:

- Dimension of each observation (state_size): 2
- Quantity of available actions (action_size): 2
- Random seed (seed): 1993
- Size of the memory (memory_size): 100,000
- Hidden units in the network (nb_hidden): (64, 64)
- Discount factor (gamma): 0.99
- Lambda for GAE-Lambda (lam): 0.97
- KL divergence between target and current policy (target_kl): 0.01
- Learning rate for the policy optimizer (policy_lr): 0.0003
- Learning rate for the value function optimizer (value_lr): 0.001
- Number of iterations to train the policy (train_policy_iters): 10
- Number of iterations to train the value function (train_value_iters): 10
- Clipping ratio for the policy objective (clip_ratio): 0.2
- Use epsilon-greedy action selection (epsilon_enabled): True
- Starting value of epsilon, for epsilon-greedy action selection (epsilon_start): 1.0
- Minimum value of epsilon (epsilon_end): 0.01
- Decay rate for epsilon (epsilon_decay): 0.995

The training session was stopped when the average score of the last 100 episodes was higher than 120 in 36,029. Figure 2 shows the training process of the PPO agent.

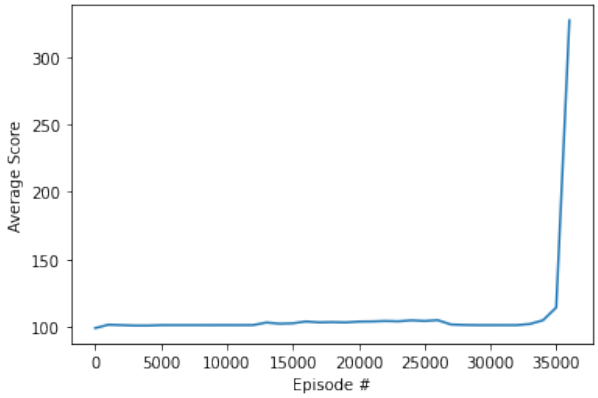


Figure 3: PPO - Average Scores by episodes

We can see that nearly to 35,000 episodes, the agent started to learn more quickly and finished the training achieving the stop condition (median of the last 100 scores higher than 120).

The PPO Agent (with 36,029 episodes trained) was tested in 10 trials as described in Table 2. The **Trial** column represents the number of the test, the **Env score** column represents the scores provided by the OpenAI Gym environment, and the **Skipped pipes** column represents the number of pipes skipped by the bird.

| Trial | Env score | Skipped pipes |
|---|---|---|
| 1 | 840 | 20 |
| 2 | 692 | 16 |
| 3 | 174 | 2 |
| 4 | 527 | 12 |
| 5 | 342 | 7 |
| 6 | 506 | 11 |
| 7 | 770 | 18 |
| 8 | 897 | 21 |
| 9 | 367 | 7 |
| 10 | 1062 | 26 |

Table 2: PPO - Results table

The PPO Agent performs better than the DQN Agent, in the best case, the bird was able to skip 26 pipes. The average score was 511.5, and the average number of pipes skipped was 14.

## Conclusion

Deep Reinforcement Learning (DRL) is a powerful tool for solving complex, high-dimensional environments. It can be used for video games, robotics, and many other tasks. In this paper, we developed two DRL algorithms: Deep Q-Network (DQN) and Proximal Policy Optimization (PPO). We tested the algorithms on the Flappy Bird game and showed that the PPO algorithm can learn to play the game properly. For future work, we plan to add hyperparameter tuning and other algorithms to the paper. We also want to run DQN with Prioritized Experience Replay (PER) with more time, and compare the results. We also can change the environment to use the screen of the game as the observation space.

## References


Alp, E. C., and Guzel, M. 2019a. Playing flappy bird via asynchronous advantage actor critic algorithm.

Alp, E. C., and Guzel, M. S. 2019b. Playing flappy bird via asynchronous advantage actor critic algorithm. *arXiv preprint arXiv:1907.03098*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.

Fedus, W.; Ramachandran, P.; Agarwal, R.; Bengio, Y.; Larochelle, H.; Rowland, M.; and Dabney, W. 2020. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, 3061–3071. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Vu, T., and Tran, L. 2020. Flapai bird: Training an agent to play flappy bird using reinforcement learning techniques. *arXiv preprint arXiv:2003.09579*.