Genetic algorithms based AI to play Flappy birds

Devishi Kesar
Computer science engineering
IIIT-Delhi
Delhi, India
devishi15024@iiitd.ac.in

Siddharth Arya
Computer science engineering
IIIT-Delhi
Delhi, India
siddharth14103@iiitd.ac.in

Abstract— **The aim of this project is to implement, compare and improve some algorithms, based on their performance on flappy bird. This would improve our understanding of the algorithms and implementation of algorithms in the field of computer games.**

I. Introduction

Reinforcement learning algorithms are one of the most popular choices when it comes to applying artificial intelligence for games. The main aim of this paper was to implement genetic algorithms for scoring with neural networks and compare the performance with other algorithms. All the algorithms were developed on the available execution of flappy birds in python by sourabhv[1].

II. Related Work

Deep Reinforcement Learning [2], introduced by DeepMind, was the first known model-free method which generalised well over multiple Atari games and performed better than humans. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and output is a value function estimating future rewards. There have been many attempts to solve various games like checkers[3], Go[4], 2048[5] etc. The attempt to solve flappy bird has also been done before using deep learning[6][7](Fig.1) and the code was replicated to compare performance.

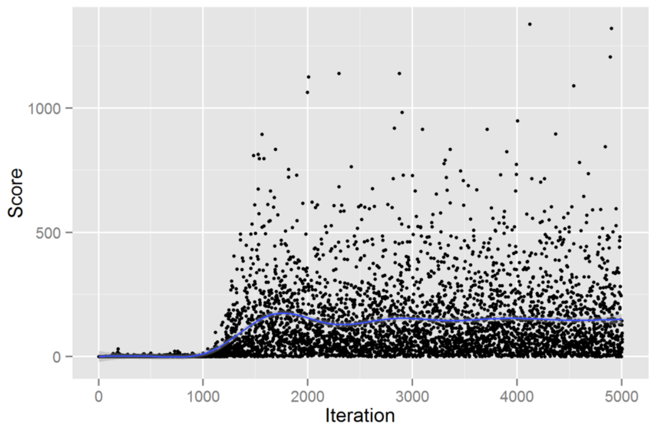

Fig. 1. Performance of q learning over iterations.

III. Design choices

- Genetic algorithm: It is a metaheuristic used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection. The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population ”evolves” toward an optimal solution.
- Deep Q Learning: It is a convulational neural network trained with a variant of Q learning. Through trials-and-errors, Q learning finds a Q-value for each state-action pair. This Q-value represents the desirability of an action given the current state. Over time, if the world is static (i.e. the physics or the cause-and-effects don’t change), the Q- values converge and the optimal policy of a given state is given by the action with the largest Q-value. Update Step:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t)+\alpha[r_t + \gamma max_a(Q(s_{t+1}, a))]$$

- The ant colony optimization algorithm (ACO): It is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.All solutions are ranked according to their reward. The amount of pheromone deposited is then weighted for each solution, such that solutions with more reward deposit more pheromone than the solutions with lesser reward. The trails are updated by: $\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^k$

IV. Results

A. Explanation

1) Genetic Algorithm: The algorithm has been developed over a neural network with three layers. The input layer consists of 2 nodes with the position of pipes and position of bird, the hidden layer consists of 3 nodes and the output has 1 node telling the bird whether to flap or not. The weight of the nodes are updated using the genetic algorithm. The implementation of genetic algorithm was done using tournament selection of 10 birds with 15 in every generation and top 5 selected as elite.The mutation rate was 20%. Our algorithm reached a score of greater than 1043 in just 34 generations.[Fig.2]

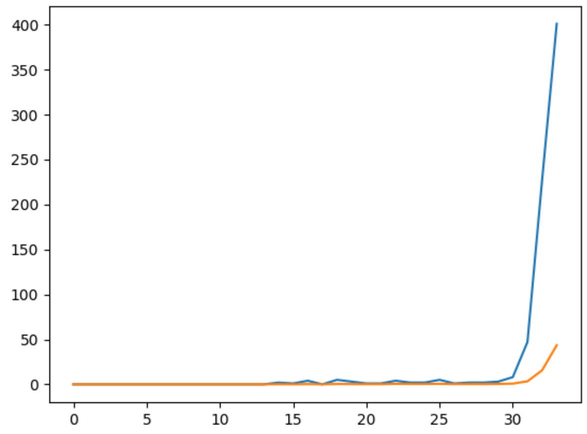

Fig. 2. Performance of genetic algorithm over generations.

2) Deep Q Learning: The Q learning algorithm took 4 days for training over GPU, the environment provides the position of pipes and bird and velocity of bird, etc as continous values, these were discretised by dividing the image into multiple frames and applying 1 action per frame. The input contains 3 inputs: pipe position, bird position, velocity of bird in particular direction; and 1 output node for deciding whether to flap or not in the frame. The game screens were preprocessed with the following steps: convert image to grayscale, resize image to 80x80 and stack last 4 frames to produce an 80x80x4 input array for network.The first layer convolves the input image with an 8x8x4x32 kernel at a stride size of 4. The output is then put through a 2x2 max pooling layer. The second layer convolves with a 4x4x32x64 kernel at a stride of 2. We then max pool again. The third layer convolves with a 3x3x64x64 kernel at a stride of 1. We then max pool one more time. The last hidden layer consists of 256 fully connected ReLU nodes. We initialize $\epsilon = 1$ and linearly anneal ϵ from 0.1 to 0.0001 over the course of the next 3,00,000 frames. [Fig.3] Since the model was trained for 1 frame per action, when tested the result was a linear increase for every second and pipe received(the line is a litle jagged as there is approximately 1 pipe in 2 seconds). [Fig.4] While when tested for 2 frames[Fig.5] and 4 frames per action(4fpa), the result was uneven as expected.

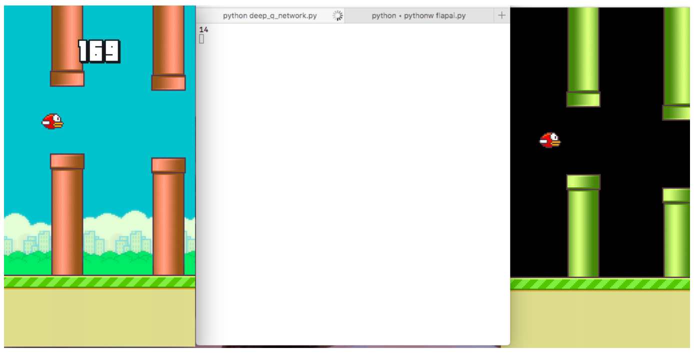

Fig. 3. Q learning.

V. Future work

A. To be done

We are planning to create an ACO(ant colony optimization algorithm) for further performance evaluation.

B. Analysis

We would further analyse the performance of the algorithms after implementing the final algorithm of ACO which would complete the aim of the project. The evaluation between the algorithms was done on the basis of best score for a particular time frame.

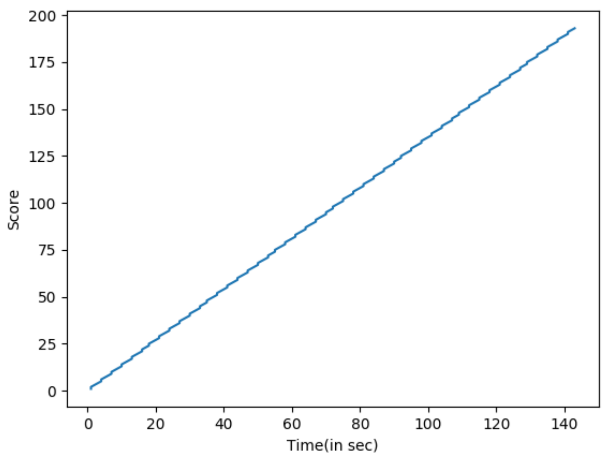

Fig. 4. Deep Q model 1fpa.

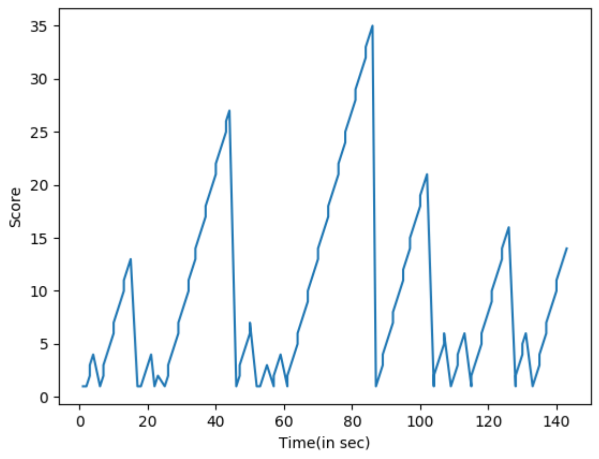

Fig. 5. Deep Q model 2fpa.

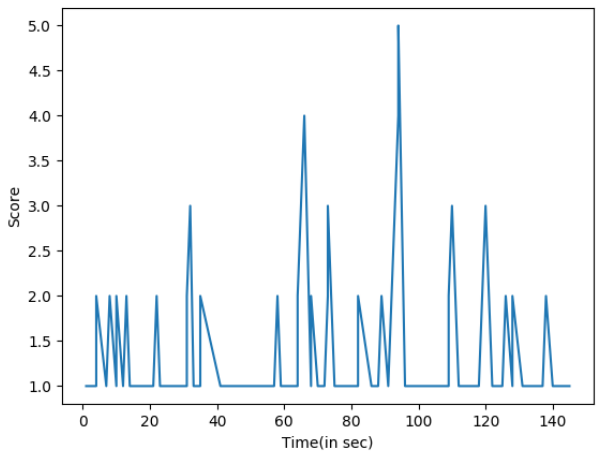

Fig. 6. Deep Q model 4fpa.

References

[1] https://github.com/sourabhv/FlapPyBird

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. CoRR, abs/1312.5602, 2013.

[3] http://sciencenetlinks.com/science-news/science-updates/checkers-solved/

[4] http://blogs.discovermagazine.com/crux/2016/01/27/artific intelligence-go-game/#.WgcARxOCzBI I. J. Computer Network and Information Security, 2012, 7, 12-18

[5] http://blog.datumbox.com/using-artificial-intelligence-to-solve-the-2048-game-java-code/

[6] https://github.com/yanpanlau/Keras-FlappyBird

[7] https://github.com/yenchenlin/DeepLearningFlappyBird