Artificial Intelligence in Flappy Bird Game

Abstract- The oldest and very popular game, The Flappy bird which is no longer available at the play store but many games follow that rules and patterns and come up with another version like temple run and subway surface. We build an Agent which controls the bird and flew between the pipes safely and makes the bird live longer all done with Reinforcement learning which updates some variables by learning at each generation after trained it makes the bird immortal unless shut down the whole game. In calculation, the bird measures itself location / state at the game then looks for the next state, which state will be more affordable and leads to achieving reward, the reward is fly between pipes.

Index Terms— Agent, Reinforcement Learning, Flappy Bird game, Q-Learning, Values Iteration

Introduction:

Flappy bird is a very challenging game and measuring the distance by just looking at the bird and pipes makes it very hard for humans if it's just four to five times making the bird fly between pipes can be feasible but for unlimited time, for that purpose need to hire a candidate for that, but we don't need any candidate to do we just use deep Q networks for an agent to learn its moves and measures distance before making any move for that, there are some variables from that it calculates pips distance and bird distance and how high bird from the ground, to get reward there must be perfect move or state in the game for the bird to choose, if it passes the pipes perfectly it will get 1 point as a reward but if bird touches the ground or any pipe it will lead end game and -1000 as a reward.

Now-a-days Artificial Intelligence is a very powerful tool for making nearly perfect predictions so that we can train a model on a specific problem and then ask for results, as for this problem we use Deep Q-learning to learn in a game's environment and keep learning generation by generation until reaching a specific goal, which is in the form of rewards. To play the Flappy Bird game successfully and be able to achieve reward infinitely with undying or becoming immoral all will be done with training an Agent. the information Agent is provided with images and in further images inform of pixels of game and score which is dependent on rewards, and rewards can be negative and positive. There is no additional information is given to an Agent which is like what is the bird and what color it has and how pipes look like and it's color and where to find pipes and birds, these are very important pieces of information but it makes mode hard-coded so for that we are not providing that information, the Agent has to explore all by itself the representation of the game and the interactions between bird and pipes in Flappy Bird game.

Flappy Bird was a very famous game in the world, the game can be played with some of the simplest strategies. Create a gap between the bird and pipes that gap should be enough for the bird to move up by taping the space button, There is only Space used in the whole game. The bird moves downward with a certain amount of slop and it continuously moves downward unit tap the space button to push the bird up side to wards sky. The game will end if the bird touches the pipe or the ground. Thus, the flappy bird game will start and be played by an Agent or making action in the environment.

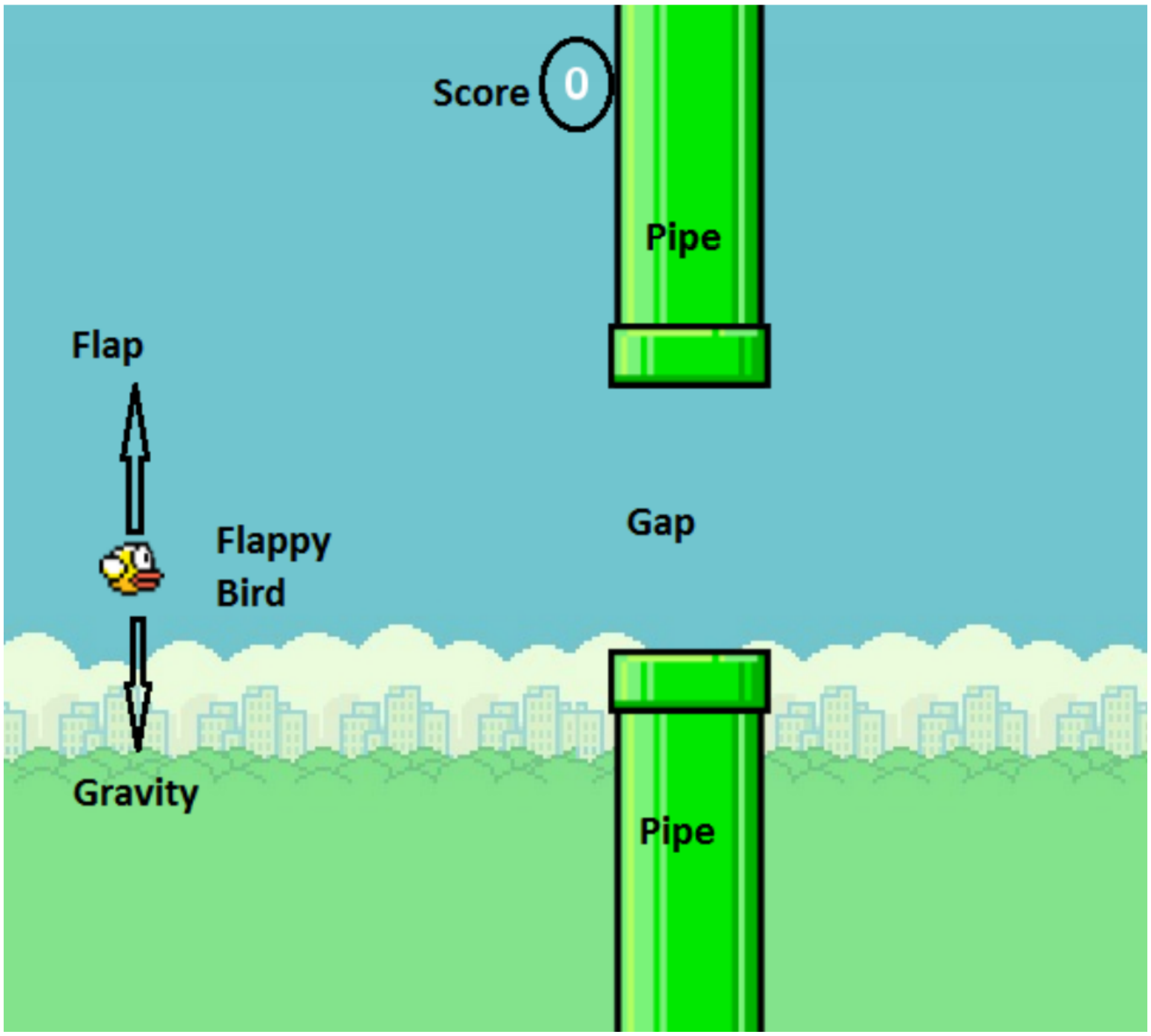
As this game has rules to lose the game or achieve a low number of scores so as there is a way to get the highest scores. According to achieve the highest score it is only possible if the bird flies between numerous pipes as conceivable if the bird has passed a lot of pipes it will lead this to succeed as the highest score. If it gets to land on any pipe or on the ground, According to this situation, the game closed.

Implementation:

Flappy Bird- The Flappy Bird is a game that is implemented in python and in the game, there have some objects which are very important because the whole game depends on them where the first one is the bird, the pipes, and the last ground. The rules of the game are the bird has to fly always although by default the bird moves downward with a certain amount of slop and it continuously moves downward unit tap the space button to push the bird up side to wards sky. The whole game depends on a single button and that is space if we press space the bird will jump and get some amplitude in the game environment, only the button we have to press in a very situation where the bird may possibly crash with the pipes or on the ground. So the game will end if the bird touches the pipe or the ground.

Bird- The bird has the vision to cross all pipes and ground without landing any of them, so for that, an Agent is developed for that which give an action of execution of button to the bird get some amplitude.

Pipes- Appears anywhere in the environment but only on the upper side of the window and on the ground and they appear in a form of pair and always have some space between them for the bird to fly from it. It has a green color with long or short length but with a fixed width, pair appear randomly at same X-axis of upper and lower if bird touch any of pipe the game will end and the final score will be revealed.



Flappy bird Game - Schematics

Method of Implemented Algorithm- The bird takes a jump also known as a flap, For the implementation of Deep Q Learning we have to compute all inputs and label them with numbers so for this in the form of action if an Agent wants to make the bird flap so the action state will be 1 (i.e : $a = 1$) and the second state for the bird is do not flap, in that case, the bird will dropping towards the ground or on a pipe which is in the form of doing nothing and letting the bird decreasing its amplitude so the action state will be 0 (i.e : $a = 0$). The Flappy Bird game updates with the sequence of frames and each frame represented the position of the bird and the pipes, were also shown the bird took action on the previous state, and that action led the bird to the helping form passes through pipes or to crash with the objects. The game is filming, frame by frame in the form of pixels so getting the sequence of states.

$st = (xt - hist\_Len + 1 , at - hist\_Len + 1 , \dots , xt - 1 , at - 1 , xt)$ as Equation - (1)

$\nabla_{\theta_i} L_i (\theta_i) = E_{s , a \sim \rho (\cdot) ; s' \sim \varepsilon} [(r + \gamma * \max_{a'} Q (s' , a' ; \theta_{i-1}) - Q (s , a ; \theta_i)) \nabla_{\theta_i} Q (s , a ; \theta_i)]$ as Equatiom - (2)

Importantly equation - 1 describes the st, where s stands for state and the t with s stands for time which means state at each time, and input as in the form of the pixel so that defines with x and contain as with t represents the time which means capturing screen or the frame at a specific time xt. The action is represented with a and action at a specific time is represented with (at), hist_Len is a tuning parameter that represents the number of the very present frames of pictures to keep tracking of. It will help to decrease the use of the storage and the state space which is compared to saving the all frames and the all actions which have started from the time t was with the value 1 $t = 1$. The question appears here why we storing numbers of frames with numbers of actions as with a number of inputs x's, why not just save one frame with only action and input this is all because the agent we build need a flow of movement of the game as also the movement of the bird these all pieces of information is required for the agent to detect the velocity of the bird's movement is decreasing or increasing this can not be deduced from an only frame, the making decision for the bird to flap or not also depends on the velocity.

There is another factor which is DISCOUNT Factor which is represented with the γ symbols in the equation - 2, this factor is set up to 95/100 which is $0.95 = \gamma$. The Agent did not know about the rewards and as well the transition probabilities, because the Q learning is free of model it stores its values in a table, and before Q learning we can not directly predict the optimal value of the Q function. This will take some paragrams to explain.

Although we have to still mention the feature rewards to the flappy bird game this is because the idea of the reward is for the Agent to essentially achieve the maximum score of the game. The score starts with the value 0 and as the bird passes the pair of pipes, the score will be updated with the value 1. However, this can be potentially problematic as rewards will be paid out rarely. For example, if the bird touches the pipe or the ground in this reaction bird will die if this happens at the beginning of the game, and the reward will be the same if it dies on the pipe before crossing the pipe. So adding a reward for survival encourages agents to think in a similar way. Without this extra reward, the agent will eventually have to figure it out, but adding a reward called the Alive reward speeds up the learning process. There are a total of three awards: the Living Award, the Trumpet Award, and the Dead Award. An agent receives an Alive reward for every frame he is alive. Go through the pipes and reward the dead for death.

Q Learning- is the type of Reinforcement Learning in which it always looks for the maximization of optimal and the expected value which will be the total reward. So that we are using iterative update with the Bellman Equation as an.

$$Q_{i+1}(s, a) = E_{s'} \sim \varepsilon [r + \gamma \max_{a'} Q_i (s', a') \mid s, a] \text{ as Equation - (3)}$$

Let me explain equation 3, r is going to represent reward pretty simple, now comes s' which is s not it represents going to be state, Q_i (s , a) represents Q function which is about to calculate i-th iteration, and also shows that the iterative of i-th term is converges to the optimal Q function. Still, this is the base learning, if we prevent the base learning, the unseen states will generalize from the Q function which is the totally used for the approximations of function. We are using the Deep Q learning approach where we are setting up Neural-Network to estimated the Q function. About Convolution Neural-Network which is also famous with the name Deep Q Network (DQN). For training the approximation of Q function with the common loss_function.

$$L_i (\theta_i) = E_{s}, a \sim \rho (\cdot) [1 / 2 \ (y_i - Q (s , a ; \theta_i)) **2] \text{ as Equation - (4)}$$

For this equation 4, Q network have some parameters and θ_i are represents its parameters where the i and y_i iteration is the goal at the i-th iteration. The goal y_i is represented as:

$$y_i = E_{s'} \sim \varepsilon [r + \gamma \max_{a'} Q (s', a' ; \theta_{i-1}) \mid s, a] \text{ as Equation - (5)}$$

There are some parameters e = (s , a , r , s'), which are data points as for the linear regression and the recount of memory, The loss function in the gradient with regard to the weights as represented by equation 2. Another parameter ε greedy leads to control the explore and exploit problems in the Q learning algorithm. So we training with the select arbitrarily actions with the probabilities of ε and select the optimal actions as (a_{opt}= argument of $\max_{a'}$ Q (s , a').

Deep Q-Network- The

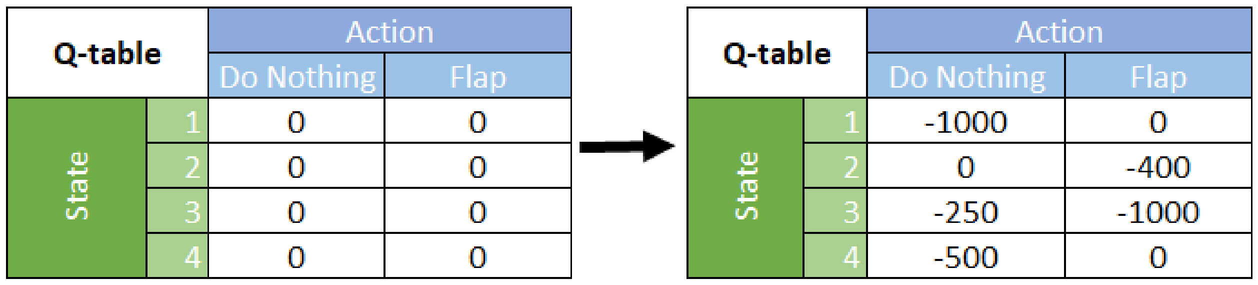
The Q function which we structured is estimated on the convolutional neural network, so the parameters taken as input to this Q network is an 84x84 history Length image as input and has one output for each possible operation. The first layer is a 32-filter 8×8 convolutional layer of 4 steps, which go along with a type of situation like hit or miss. After the first layer, there is another layer which comes after which is 4 × 4 convolutional layers which have around 64 filters from two stages, followed by another block of straight lines. The third convolutional layer has a 64 3 × 3 filter from stage 1 followed by a modified block of lines. This is followed by a completely join tier with the 512 as outputs, followed by an output tier (completely join) with one output for each task. To choose the best course of action, we take the action that works best Q value ($a_{opt} = \arg\max_{a'} Q (s , a')$).

Results:

Training time- Q learning is a type of machine learning which as no model where the techniques are commonly used to achieve the optimal actions of the Agent in a specific frame or state. Where an agent performs an action in a specific frame as a state, it will receive the reward as chosen action with did not end the game and also the penalty in the form of choosing the action which led to the end of the game. The Agent has an aim to achieve the highest reward from overall rewards so that potential future rewards must also be considered when performing the action. For loose birds, the agent is a bird that is an action that can do nothing or flap.

- Mark x as a vertically measure the distance to the pipes which are about to come.
- Mark y as a vertically measure the distance to the bottom pipe which is about to come.
- Currently, the parameter of y and the bird with the velocity v.
- Mark y1 as the gap between both pipes top and bottom which are about to come.

For the first time as reached to the new state, the values will be initialized of the Q function, so the agent will note as the default actions on the states and received a reward, and if the generation has been ended as the bird touch with the pipe, the Agent will start performing actions with the maximum Q value, it will considered the most rewarded value of the action. The Q value will be improves the value which is going to be entered or stored into the given cells in a state × action Q table.



Q-table		Action	
		Do Nothing	Flap
State	1	0	0
	2	0	0
	3	0	0
	4	0	0

Q-table		Action	
		Do Nothing	Flap
State	1	-1000	0
	2	0	-400
	3	-250	-1000
	4	-500	0

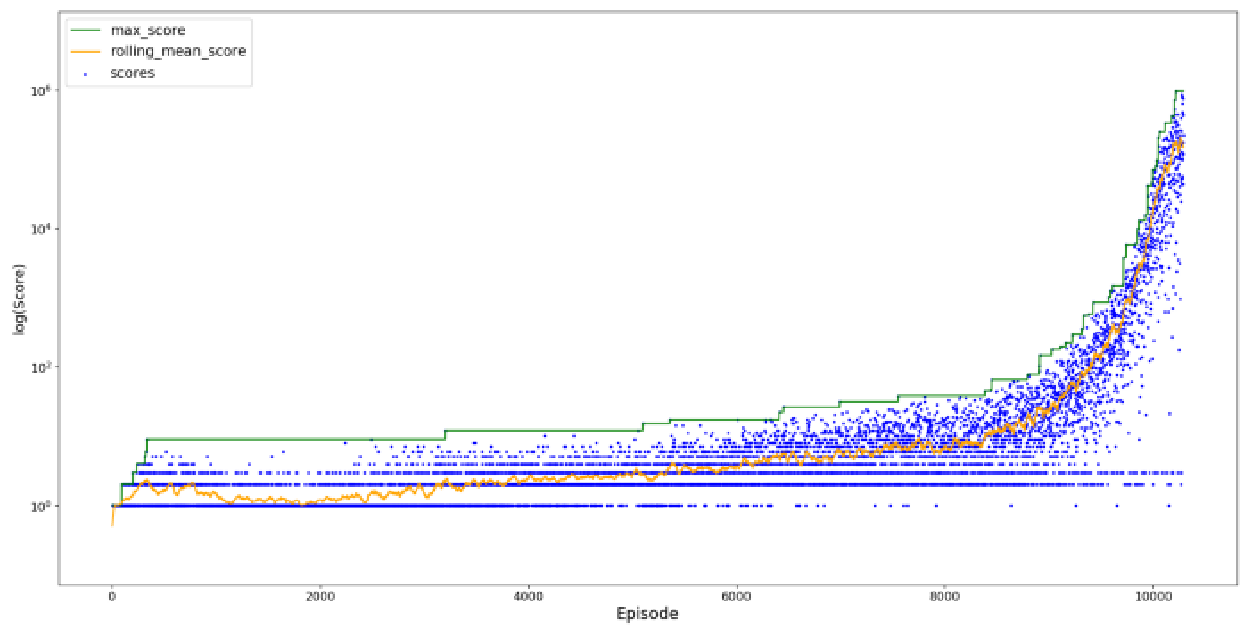
The Q table is updated as the agent takes different actions for different states. The next time the agent encounters state 1 it will flap, and for state 2 it will do nothing.

The Q table has values which are Q values as for the several parameters:

- Improve Q value as in the 2-D Array.
- Q [stat , act] = Q [stat , act] + α * (r + γ * np . max (Q [stat', :]) - Q [stat , act])
- Alpha α is the number of steps with the size committed for the previous information and the new information is extent about.
- Gamma γ committed with the rewards which will get in the future like $\gamma = 0$ represents the Agent will only about to examine its present reward.
- Q [s , a] is the present Q value.

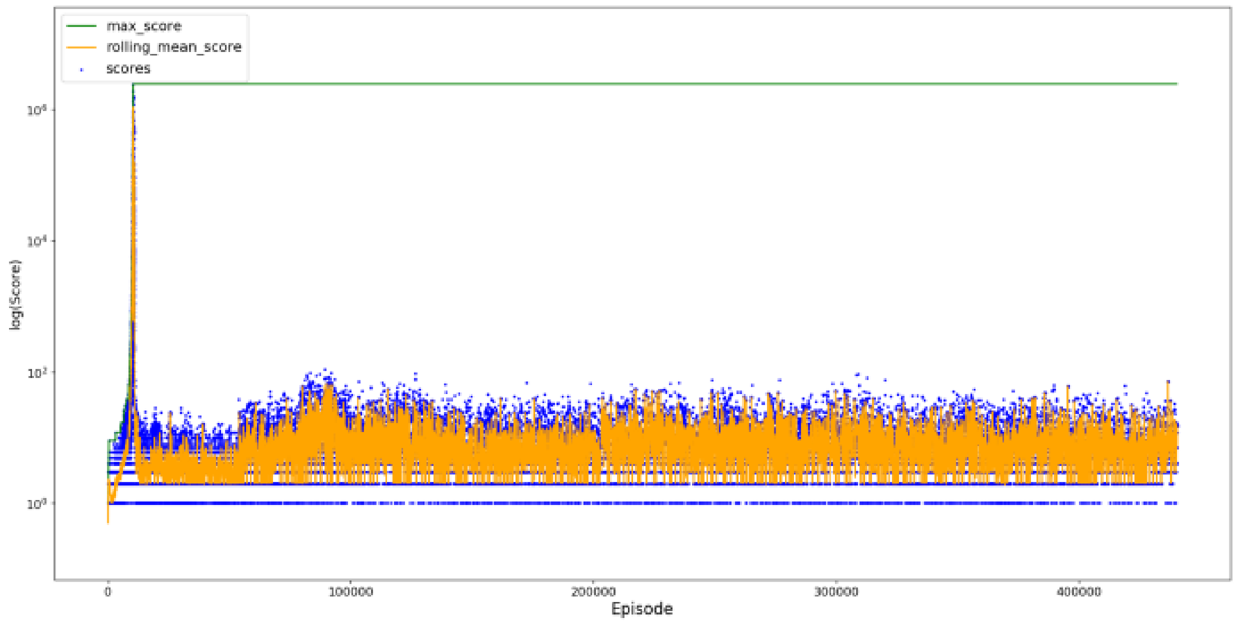
The praise feature became described to penalize -a thousand for loss of life and zero or else, kind of the agent's awareness is the get as excessive a rating as feasible. This guarantees which the praise feature have enough effect in the wake of every episode, vs an execution in which reward is going to be +1 to a rating growth approach that penalization has little to no effect.

The agent got trained with around about ten thousand of episodes When α has around 0.7 and γ has a 0.95, and with not any investigation which is about to score around 1 million scores and this was really something and its y-axis was recorded, so in a while the gradually got increases with a lot of points which can be seen easily and with clarity.



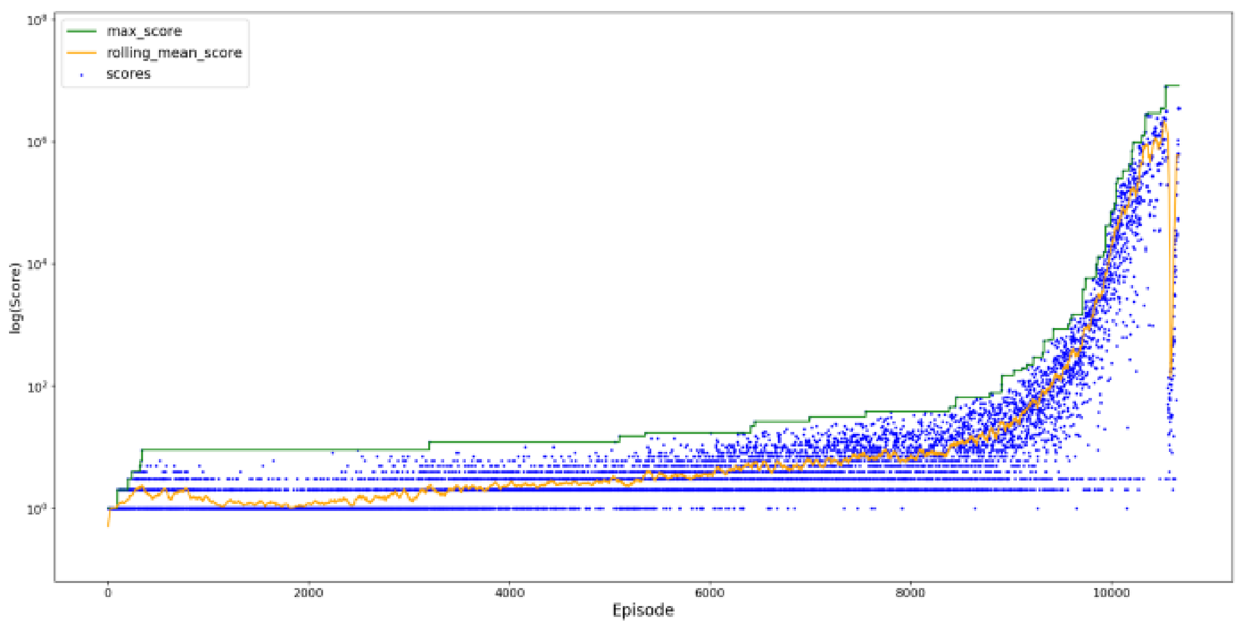
The agent is about to learn how to grow.

Agents accomplished very well as the training initiating, but as it takes longer time to improve its strategies which are basically values which it improves further as it explores more. But the problem is that it takes too much time and not just time also with lots of computations to get to the scenario where the agent dies, and after that, you can only learn from that death once. An agent can either implement a replay experience to overcome a complex scenario, or it can try multiple executions until it gets stays with the carry on with loop (here we fixed it to thousands of tries). If you fail after completing a difficult scenario, you start over from the beginning to avoid increasing your maximum score.



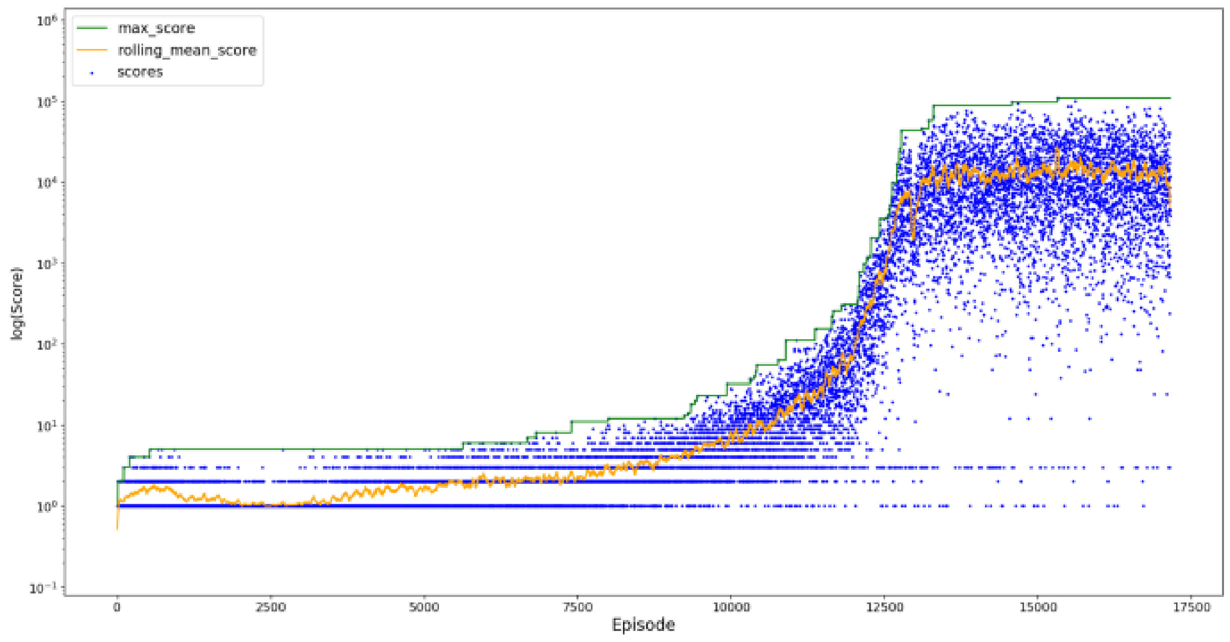
Q table is loading and the training got initialized but with the duplicated experience from the max score.

The agent's performance improves because it can initially be learning from quite complex scenarios, even so, the accomplished descending with quite fast below the last time maxima. This situation is also called severe forgetfulness, whichever is often primary to a wobble in the Agent's accomplished as they fail to learn and re-learn the actions which about to that and that will be optimal. Before learning the same situation continuously which is leading to failure, the Agent learns too much and forgets the previous generalizable Q values. Alpha decay is added as the agent continues to learn from rarer scenarios as it learns to retain the information it initially learned to overcome fatal oblivion. Also, the Agent many times got stuck in the loop which is about to attempts which have been reduced around 50, and a "playback buffer" is created with all actions taken during playback of the experience. Q table is then refreshed based on this mini-batch and picks up 5 trials after the playback experience is over.



Bird learns too well and has a good memory.

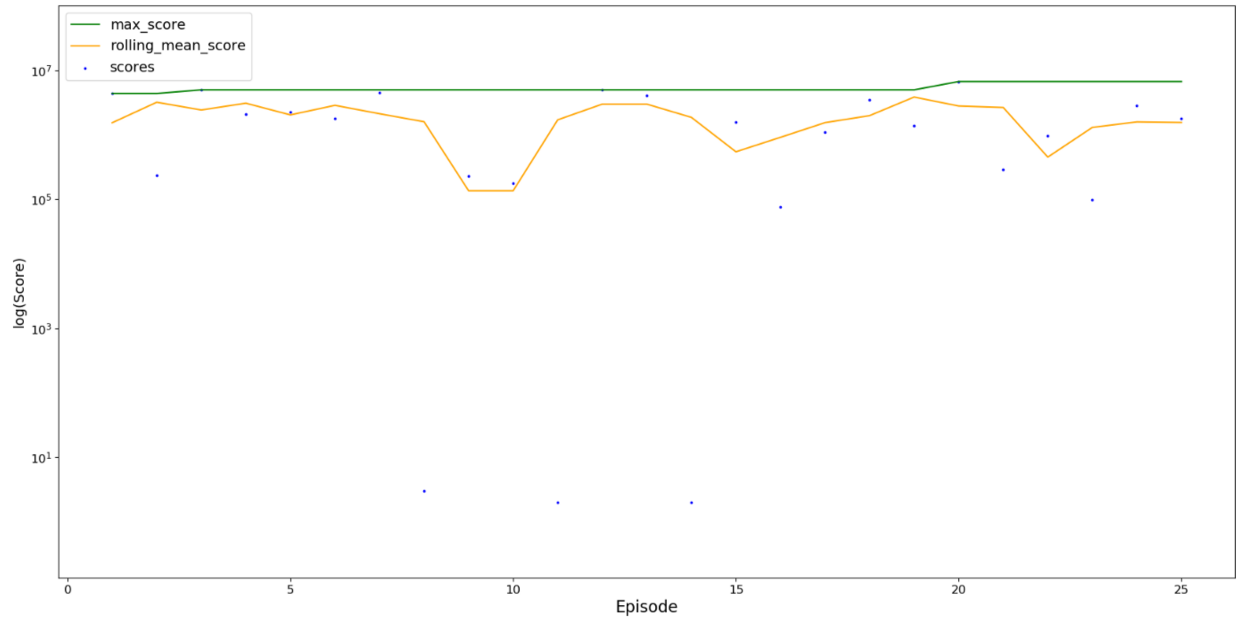
The Agent can earn nearly 10 million points. Performance degrades over 10,000 training sessions, but you can recover from early oblivion. Unless you spend extra time training that agent, you can expect the agent's performance to fluctuate as it learns and retrains the optimal action. As the Alpha is about to continue decreasing, this will be in the end allow the agent to stay strong and stable near the maxima.



Classic sigmoid growth curve.

Epsilon's Greedy Politics: We now test the newly trained agent by introducing a study ratio epsilon ε that allows us to investigate random behavior from the (0.1) to (0) afterward the (ten thousand) generations, and an alpha decay that decreases from the (0.7) to (0.1). 20,000 generations.

Epsilon's Greedy Politics: In this, we are going to test the recent agents which got trained which are going to represent a study ratio epsilon ε that allows us to investigate any task which is arbitrary until it decreases from (0.1) to (0) after the (ten thousand) generations, and the Alpha decaying that reduces the Alpha from (0.7) to (0), one time after (twenty thousand) times.



🐤 That's a good high score! (Image by author)

The agent plays well, always passing 100,000 and achieving the most rating above five million. It is capable of by skip maximum conditions, however, is not able to stay for all time whilst extra tough eventualities appear, which means it every so often dies pretty primary on.

- Steadiness: The value of alternative (standard deviation/mean) is (0.967). which is to be looking forward to after all in an arbitrary surrounding the agent, this is only fully stable after overcoming all scenarios and never dying.
- Average score: Average score 2 001 434. This is a strong result that an agent who excels in any human game can achieve consistently.
- Maximum Score: The highest score scored in twenty-five runs is around 6,720,279, which again surpasses the human game with the highest score of near to 10 million, the default maxima learning value.

CONCLUSION- We have been able to successfully play the Flappy Birds game and learn and calculate right on the pixel to achieve superhuman outcomes. Even so, the training was inconsistent in that quite large training will not as the matter of course agree with good performance. The model may be overfitted or forgotten, so it will be possible to study and solve this problem in the future. Another very important area for improvement is experience replication. Although we continuously sampled replay memories, some experiences have a greater impact on successful DQN learning than others. The ability to prioritize these experiences will lead to productivity, learning efficiency, and faster convergence. In addition, the background and calculations that have been removed from this game are to decrease bestrew and increase the chances of successful learning. The background improvement will really boost the Agent to act best in performance of choosing the actions. If we talk about our project it is nice to hear that our game with the implementation of Deep Reinforcement Learning is in the very right direction and it has great possibilities for different applications.