**Project Proposal: Flappy Bird using Deep Q-learning**

**Team Members:** Radium Zhang(Student ID: 5408080; Email: [zhan5821@umn.edu](mailto:zhan5821@umn.edu) );

**Introduction:**

In this era of evolving technologies, Reinforcement Learning is an emerging branch of Artificial Intelligence that has been used to play Atari games and great success can be seen in [1] and [2]. Here, we want to design an agent that can automate the flappy bird game. Earlier, the problem of automating games like flappy bird has been solved by parsing the images of the game through a convolutional neural network and obtaining the states as an output from the convolutional neural network. Our aim is to train the agent without using images but using the states which we obtain from the OpenAI gym based game engine and evaluate the performance by comparing it with existing image based algorithms which are treated as a benchmark during this study.

**Problem Statement:**

Flappy Bird is a game that involves navigating a bird through a bunch of obstacles using reinforcement learning. In this project, we will use an open AI and Pygame based game engine to train the agent to take the right action at each instance of the game through Deep Q-learning under an optimal policy.

Because in Q-learning algorithm producing and updating a Q-table can become ineffective in big state space environments and this can lead to two issues: First, the amount of memory required to save and update that table would increase as the number of states increases. Second, the amount of time required to explore each state to create the required Q-table would be unrealistic. Therefore, Deep Q-learning algorithm, which implements a Neural Network that takes a state and approximates Q-values for each action based on that state, arises.

**Model Formulation**

At each time instant t, the actions which bird can take are to flap (a=1) or not to flap (a=0) i.e the player can either press up key to make the bird jump upward or not pressing any key consequent to which the bird will descent at a constant rate and dies. As the bird dies when it hits the pipe or the edges of the screen, a negative award can be associated to the crashing of the bird and a positive reward if it passes through the gap between the obstacles. This way it imitates the human player’s behavior who tries to avoid dying and maximize the total score.

We attain the goal of maximizing the expected total discounted rewards in the task by defining a value function V for each state under policy π as follows:

Vπ = E[ ∑ 𝝲k Rt+k+1 | St = s, At = a ]

Also, the V can be retrospectively updated by just observing the rewards and value function of successive time steps as follows until we reach the destination:

V(S) ⟵ V(S) + α[ R + Ɣ.V(S’)-V(S) ]

**Plan and expected result:**

First, we plan to implement different architectures of neural networks and collect a comparative analysis of training performance and test performance of the agent using deep Q-learning approach. We would expect that the best result achieved above will do better than the image based algorithm in [3]. More specifically, we would compare the training time to show the advantage.

Furthermore, we plan to investigate the effect of exploration on computation time and learning time. More specifically, we will implement some of the other strategies other than epsilon greedy such as decaying epsilon greedy, boltzmann distribution softmax search etc. to see if we can learn faster compared to uniform random search over actions.

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning.arXiv preprint arXiv:1312.5602, 2013.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.

[3] Naveen Appiah,Sagar Vare.Playing FlappyBird with Deep Reinforcement Learning