Reinforcement Learning Term Project

Snake game genetic algorithm

Weseem Ahmed

**What is snake game?**

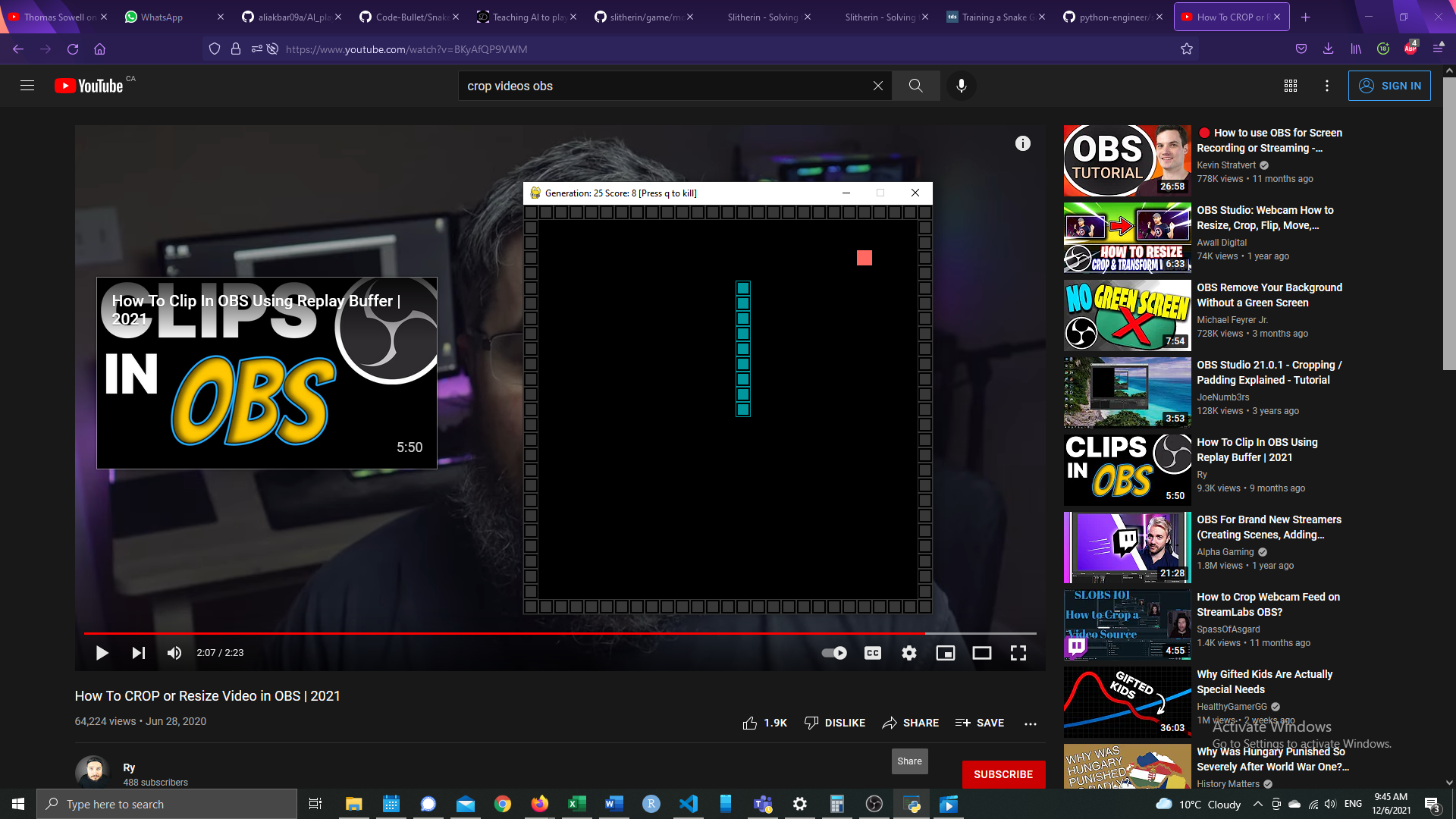
*Snake* is a classic game with a simple premise: a “snake” existing in an environment must try to eat an apple and grow. Once the snake reaches the apple, it grows a little bigger and a new apple appears in the environment which the snake must eat again. This process repeats until one of the following events occur:

1. The snake leaves the environment (ie. Hits a wall)
2. The snake takes too long to reach the apple and dies of hunger
3. The snake accidentally eats itself (by hitting its own tail)
4. The snake grows so large that there is no physical room in the environment left for an apple to grow

**Interacting with the environment**

The snake can interact within its environment via four possible actions (left, right, up, and down). In addition to its actions, it can perceive the apple’s location by looking in eight directions at each step (figure 1). By looking into the environment, the snake also perceives the location of the walls around it, as well as its tail.

Figure 1: The snake looks in eight directions before deciding its next action



Therefore, getting the snake from its current location to the apple can be simplified into a pathfinding problem of which there are multiple ways to approach it, each with their benefits and drawbacks. Typically, the shortest distance between the snake’s head and the apple would be the most efficient, but longer snakes will often bite their own tail using this process. Taking the longest distance possible (ie. The Hamiltonian distance) would solve this issue but would take longer to process. In this project, I opt to use a neural network to initialize the “brain” based on a genetic algorithm to let the snake “learn” the best way to reach the apple.

**Constructing the neural network**

The purpose of the neural network is to provide our snake with a starting point in making decisions.

The neural network itself is composed of three layers: a single input, hidden, and output layers. The input layer has 24 nodes (eight observational directions for each of the apple, snake’s body, and the wall). The output layer consists of four nodes, one for each possible direction. Finally, the hidden layer is composed of 18 nodes, although this number can be fine-tuned in future iterations.

The neural network is initialized with uniform random weights between . We use a relu activation function and softmax in the final layer. From here, the neural net is trained on a genetic algorithm to “evolve” snakes over time.

**Evolution in action**

Genetic algorithms handle transforming basic, primitive creatures into truly impressive specimens. We have a population of 500 snakes existing in any given generation (a total of 50 generations in our case). The “genes” of the snakes are randomly generated via the network weights. The snakes are let loose in their environment and are judged by their fitness.[[1]](#footnote-1) The top ten percent along with the bottom one percent of snakes of each generation are selected to “breed” and form the children snakes of the next generation. For variety, “mutations” are introduced each generation. This adds evolutionary pressure to select and breed the most successful snakes every generation.

**How did it perform?**

There is a clear difference in performance between early and later generations. Typically, snakes from generations 1-10 manage to eat up to five apples before crashing into a wall. However, progress improves markedly from generation 12-15 with double digit apples. The leading cause of death around generations 15-17 appear to be dying of hunger, perhaps as the snake grapples with learning how to navigate its environment with a longer body.

This dilemma becomes even more clear from generation 20 onwards as the leading cause of death is almost always crashing against its own body (figures 2&3). Nevertheless, later generation snakes are much more successful and eat upwards of 80 apples before biting themselves. Movement patterns at this stage become banal, with the snake looping continuously in circles to reach the next apple rather than deftly moving in its environment. A lack of variety in maneuverability will almost always lead to the snake running into its tail, especially at longer lengths. Better pathfinding algorithms will be needed to mitigate this behaviour and better navigate around a growing body.

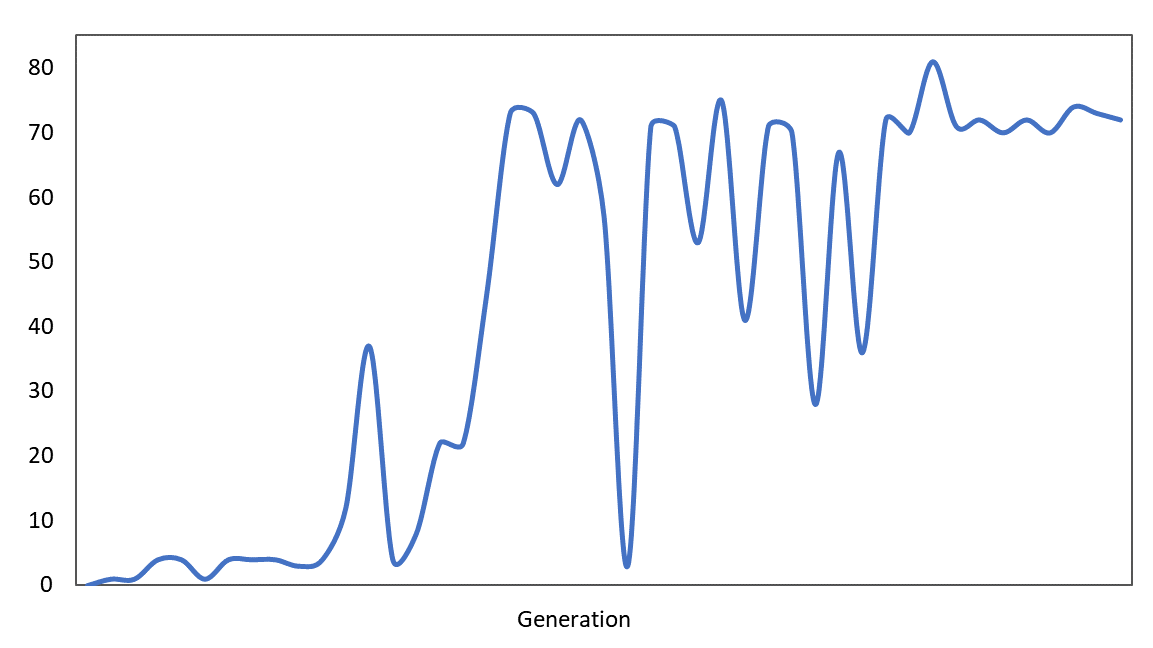
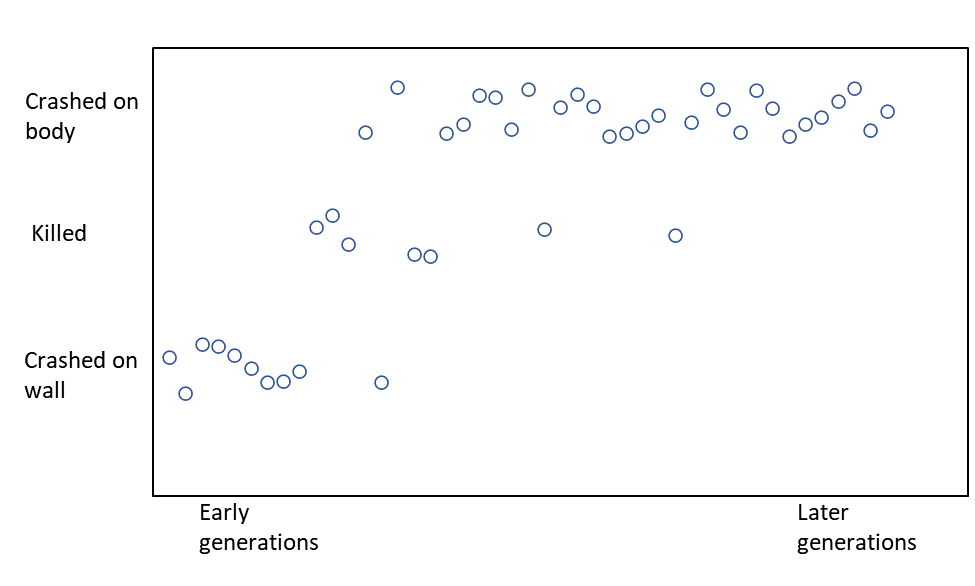
Figure 2: How many apples did snakes eat over 50 generations?

Figure 3: Snakes’ leading cause of death depends on the generation



References

Akbar, Ali (2019).

Code Bullet (2018).

Liu, Davide (2020).

Surma, Greg (2018).

Clark, Thomas Hikaru (2021).

1. Fitness in this case is simply how many apples the snake ate (ie. Total snake length). [↑](#footnote-ref-1)