Zachary Bowyer and Jonathan Olderr

Qian Mao

CS-457 Artificial Intelligence

20 May 2022

Final Project Report

For our final project we created an AI to play the classic game of snake utilizing a neural networks and genetic algorithm. The inputs for this neural network were the food’s x/y position and checkers if the snake would die given its next move is north, south, east, or west. These inputs then feed through two hidden layers (with 20 nodes) and return a probability for its next movement (N, S, E, W), with the highest probability being the move taken. Each edge connecting between nodes between layers has an assigned bias and weight, with the weight controlling the strength of the connection and the bias a threshold for a node to signal another. Usually, the wight and biases are determined with back propagation, but for this project we implemented a genetic fitness algorithm. This algorithm takes the most successful model the AI is running on and implement a “mutation”. These mutations are alterations to the weights/biases, and if the mutated model performs better, keep the model, otherwise disregard it. This method works much slower than backpropagation as there are many weights/biases to be accounted for, thus much training time is required. Another implementation was a starvation timer, in which if the snake takes too long, its overall score is decremented (this score increases by the snake eating food). This had to be implemented because otherwise the bot learned to just go around in a circle, not increasing its score, but also not risking a chance of losing.

We would have our computers running for days with just the bot training constantly only to see the AI make small advancements. This was a result of the genetic algorithm making those small changes to the weights/biases. There was also the challenge of getting the right inputs for the neural network. Much of this project was testing inputs which may work such as loading in the whole screen, loading in an array of positions corresponding to the snake’s body, distance from head to tail, number of foods eaten, etc. But to see if any changes were sufficient in improving the AI, we must let it train for about a day to see any results, resulting in a testing bottleneck. Since we used neural networks and a genetic algorithm, we were stuck with these slow testing speeds, it would have been faster to use Q-learning or a min/max. For Further research in improving this bot, it would be interesting to see how using backpropagation might make training time faster/slower and how much improvement the bot might see from that training time.