CAR RACING SIMULATION

Implementation of Neural Network and Genetic Algorithm_Jayoung Byeun / Tilman Kreis

HOCHSCHULE DER MEDIEN

INTRODUCTION

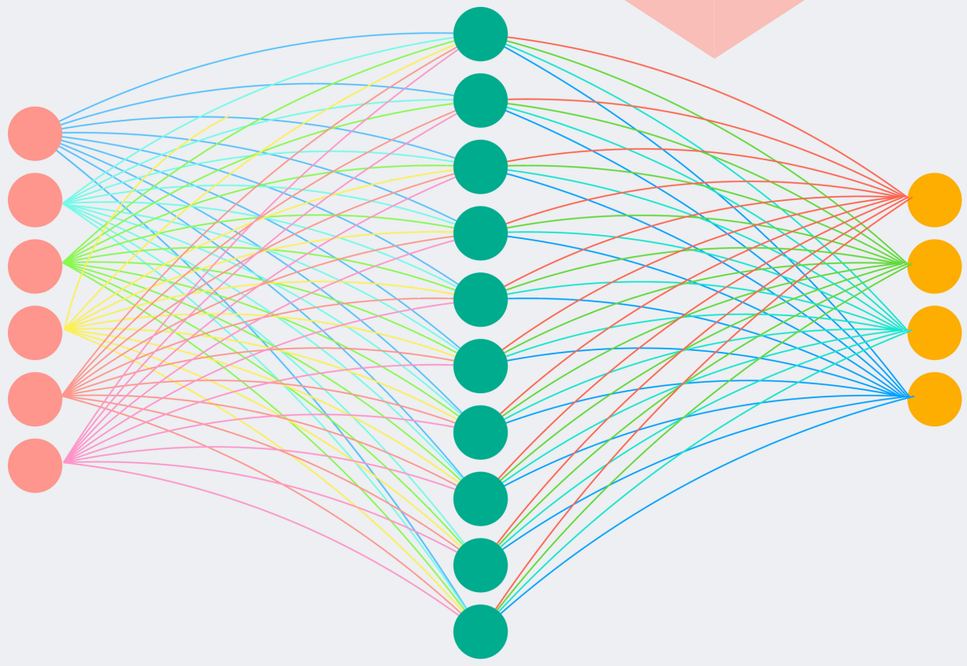
In this project, neural networks (NNs) learn to navigate a racetrack using a genetic algorithm inspired by the principles of natural selection and genetics. Over several generations, these networks evolve and adapt to improve their performance on the track.

IMPLEMENTATION

In each generation, the two best performing networks are selected as "parents" and their biases and weights are crossed and mutated, resulting in a new generation of similar networks. The selection function uses a score value derived from the distance traveled by the car.

```
top_cars = sorted(cars,
key=lambda x: x.score,
reverse=True)[:2]
```


```
def sigmoid(x):
    # Clip x to avoid overflow
    x = np.clip(x, -500, 500)
    return 1 / (1 + np.exp(-x))
```



The input layer is comprised of five nodes, each representing the car's distance to the edge of the track, along with one neuron dedicated to acceleration. Additionally, the network includes a hidden layer featuring 10 neurons and an output layer with 4 neurons representing the possible actions the network can undertake: acceleration, braking, right-turning, and left-turning.

Each neuron within the network possesses weights signifying the strength of its connection to other neurons. These weights play a crucial role in determining the network's output. Furthermore, a bias may be applied to a neuron, serving as a fixed offset value. Adjusting both weights and biases allows for the enhancement of the network's performance in each generation.