AI Driving Learning Simulator Using NEAT

Abdul Hadi Durrani(L215630) Nawall Aamer(L215620) Talha Imran(5657)

May 6, 2024

1 Introduction

With the help of Pygame for graphics and the NeuroEvolution of Augmenting Topologies (NEAT) algorithm, our project offers a 2D track learning driving simulator that lets users create personalised maps in MS Paint. The automobile learns the tracks on its own using NEAT's genetic algorithm, while Pygame offers immersive visual feedback. This study describes our simulator's architecture, implementation, and results, emphasising how game technology and artificial intelligence are combined to offer an interactive environment for investigating autonomous driving behaviours.

2 Preliminary Features

Our fundamental approach involved creating a white background against which collision detection could be effectively performed. We utilized this color contrast to detect collisions accurately. To visually represent the car, we employed an image of a car with a rectangular hit box, along with a predetermined starting point for the vehicle. The NEAT algorithm, central to our project's functionality, relied on a configuration file to govern the operation of the Feed Forward Neural Network (NN). This configuration file afforded us comprehensive control over the Neural Network's operation. Some key features facilitated by this file include:

- fitness criterion
- fitness threshold
- Pop size
- reset on extinction
- species fitness func
- max stagnation
- species elitism
- elitism
- survival threshold
- activation
- bias init
- weight init
- num hidden
- num input
- num output

3 Default values and our interpretation

The default configuration provided with the model offered a starting point, but its efficacy was contingent upon the car's sensory input. Initially, we adjusted the "num input" parameter to 3, anticipating forward, left, and right collisions. However, this simplistic approach yielded subpar results. Subsequently, we expanded our radar system to encompass five sensors, covering forward, left, and right collisions, as well as forward-left and forward-right perspectives. This adjustment markedly improved the model's ability to discern optimal turning points. Additionally, we initialized our training with a predefined generation size of 30 and an elitism value of 3. To encourage the emergence of improved solutions, we set the maximum stagnation threshold to 20, allowing newer generations ample opportunity to refine performance. Notably, these parameter values underwent iterative adjustments throughout our project, adapting to the varying complexities of the maps encountered (discussed further). Furthermore, for our fitness function we calculated the distance the car has travelled by keeping speed (initialized with 25) as our context. Finally one generation runs for around 20 secs and we have total 1000 generations after which the program terminates.

4 Difficulties

One of the primary challenges encountered in our project pertained to determining the corners of the hit box and detecting collisions. Given the criticality of accurately assessing these factors, an additional challenge emerged in determining the precise angle at which the car needed to turn for successful navigation. To address these issues, we leveraged trigonometric principles, employing cosine and sine formulas with respect to the car's center. These mathematical insights significantly aided in resolving both challenges. Furthermore, in our pursuit of optimizing input parameters for effective learning outcomes, we observed that the time frame within which the car learns to execute turns, particularly in relation to its speed, is relatively brief. To preemptively provide pertinent information to the Neural Network, we implemented an additional radar system to anticipate upcoming turns, thereby enhancing efficiency. Subsequently, challenges arose concerning the complexity of the network's hidden layers. Initially, to achieve improved performance, we commenced with a single hidden layer, which yielded satisfactory results for simpler maps. However, with the introduction of more intricate maps (Below Figure), our model encountered prolonged convergence times. To mitigate this, we undertook two key strategies. Firstly, we eliminated the hidden layer altogether, as all requisite inputs were already being provided. Secondly, recognizing that our output nodes encompassed four distinct functions (left turn, right turn, speed up, and slow down), we selectively removed the speed-related neurons for complex maps. This streamlined the decision-making process for the neural network, resulting in marked improvements and successful convergence across all map scenarios.

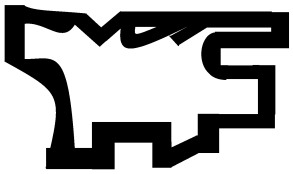

Figure 1: intricate Map

5 Final Evaluation Metrics

For the map depicted in Figure 2, we conducted simulations spanning 10 generations. In the initial generation (Figure 3), the average fitness stood at approximately 3238, with the highest fitness recorded at 14034. The generation lifespan, terminated either by extinction or collision, spanned 3.046 seconds. However, through successive iterations and mutations, notable improvements emerged by the tenth generation. Here, the average fitness surged to 1694971, with the best fitness peaking at 9418130. Impressively, the total execution time for this generation cycle reached 22.572 seconds. These results underscore a significant evolution in the model's performance, evidenced by the ability of the

fittest chromosomes to complete generations without succumbing to collisions. This substantiates our assertion that the model successfully learned the track, demonstrating its adaptability and learning capabilities within the simulated environment.

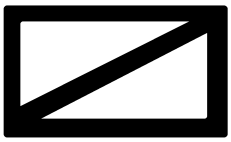

Figure 2: Map

```
****** Running generation 0 ******

Population's average fitness: 3237.86667 stdev: 4557.04898
Best fitness: 14034.00000 - size: (4, 20) - species 1 - id 2
Average adjusted fitness: 0.227
Mean genetic distance 1.218, standard deviation 0.262
Population of 30 members in 1 species:
   ID   age  size  fitness  adj fit  stag
  ====  ===  ====  =======  =======  ====
     1    0    30  14034.0    0.227     0
Total extinctions: 0
Generation time: 3.046 sec
```

Figure 3: First Generation

```
****** Running generation 9 ******

Population's average fitness: 1694970.86667 stdev: 3460721.69788
Best fitness: 9418130.00000 - size: (4, 19) - species 1 - id 211
Average adjusted fitness: 0.178
Mean genetic distance 1.442, standard deviation 0.472
Population of 30 members in 2 species:
   ID   age  size  fitness  adj fit  stag
  ====  ===  ====  =======  =======  ====
     1    9    10  9418130.0    0.164     6
     2    6    20  9418130.0    0.192     5
Total extinctions: 0
Generation time: 22.572 sec (15.817 average)
```

Figure 4: Tenth Generation

6 Conclusion

In conclusion, our project exemplifies the successful fusion of artificial intelligence and gaming technology in the development of a 2D track learning driving simulator. By leveraging Pygame for graphics and the NeuroEvolution of Augmenting Topologies (NEAT) algorithm, we have created a platform where users can craft personalized maps in MS Paint, with the car autonomously learning the tracks through NEAT's genetic algorithm. Our meticulous adjustments to the NEAT algorithm's configuration parameters, such as expanding the radar system and fine-tuning generation size and elitism values, have significantly enhanced the model's learning capabilities and adaptability to diverse map complexities. This project not only showcases the potential of AI in autonomous driving simulations but also opens avenues for applications in fields such as robotics, gaming, and education, where dynamic learning environments are crucial for enhancing AI-driven behaviors and decision-making processes.