UNIVERSITY OF SUSSEX

2023-2024 Spring
Adaptive Systems - Assignment 2

Adaptive Navigation Algorithms for a 2D Self-Driving Car Simulation

Abstract. This report explores adaptive navigation algorithms in a 2D self-driving car simulation. The NEAT's (NeuroEvolution of Augmenting Topologies) role in evolving neural networks for collision avoidance and pathfinding is explained, by using feed-forward and recurrent neural networks. Then their performance across various circuits with their strengths and weaknesses is analyzed. Outcomes show that while recurrent neural networks outperform in complex environments, feed-forward networks are more efficient in straightforward scenarios. This outcome may lead future research directions, like extended simulations and increased environmental complexity.

Table of Contents

Introduction

Adaptive systems play a crucial role in autonomous vehicles, by enabling vehicles to navigate through complex environments and adapt to changing conditions with no human need. This project aim is to study adaptive navigation within a 2D self-driving car simulation, especially focusing on the development and evaluation of adaptive algorithms to improve pathfinding and collision avoidance.

Adaptive systems in autonomous vehicles utilize sensors and machine learning algorithms that can be modified according to the changing conditions on the road. In other words, adaptivity means system's ability to learn and modify its behavior based on environment changes such as traffic flow, and the road path[1]. Navigation and pathfinding are the key components of self-driving vehicles, they are directly related to their safety and performance. In this project, the simulation environment presents various road paths and moving obstacles to test the adaptive system's effectiveness. The objective is to examine how different algorithms and adaptive techniques influence the system's performance in terms of speed, accuracy, and collision avoidance[2].

The goal is to understand the behavior of adaptive systems in autonomous vehicles, with an emphasis on real-time decision-making and environmental adaptation. By simulating complex scenarios, the strengths and weaknesses of different adaptive strategies are analyzed, which offers insights into their potential implications in real-world autonomous vehicle systems.

Background Research

Adaptive navigation algorithms in autonomous cars utilize multiple sensor technologies and data processing methods to achieve safe and efficient navigation in complex environments. The combination of different sensors like LiDAR, GNSS and IMU with advanced algorithms is one of the crucial parts in the development of a real self-driving simulation. For example, studies on sensor fusion demonstrate how combining multiple sensors can improve localization and mapping accuracy. Techniques such as LiDAR odometry and NDT-based localization provide high precision, enabling adaptive navigation in diverse scenarios[3].

Simulation environments are the essential tools that are responsible for testing and refining the self-driving algorithms. Competitions like the SAE AutoDrive Challenge encourage participants to run simulations which help them to evaluate multiple algorithms and sensor setups and offer an opportunity to explore the new and innovative solutions in a controlled environment[4]. These simulators can be used to evaluate open-loop perception, closed-loop controls, and the generation of code for controls algorithms. In addition, synthetic data of sensor fusion and control system plays important role in the development of adaptive navigation algorithms.

Regulatory frameworks and calibration techniques are crucial for the development of adaptive navigation systems. Compliance with the General Data Protection Regulation (GDPR) help to direct technologies of self-driving towards data protection and user privacy [5]. Techniques like SLAM-based calibration provide accurate sensor data without the necessity for fiducial marks[6]. This flexibility allows for more reliable sensor setups and enhances the adaptability of self-driving car simulations in various environments.

Methods

Simulation Environment

Simulation environment is created by Pygame which is a Python based library for creating simulations and games. The resolution of the simulation is set to 1920x1080 pixels. The environment is consisting of the circuit layout, car dimensions, moving obstacles, and the starting points. There is also 2 different circuits available Austin and Silverstone which are real Formula 1 circuits. Each circuit has unique characteristics, which has an influence on how the car behaves, like the starting position, the locations of obstacles, and initial angle which determines the direction of the road. Since there were 2 different circuits and the starting direction was tested in both directions in each circuit, they followed 4 different paths in total.

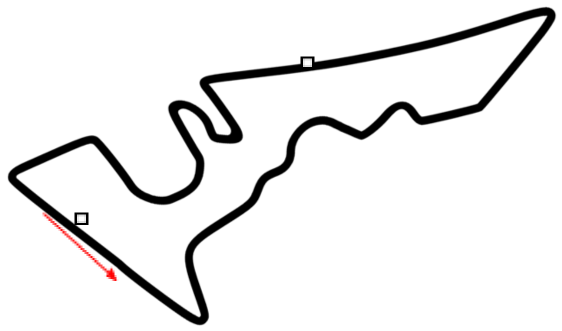


Figure 1: Austin Circuit with initial starting area[7]

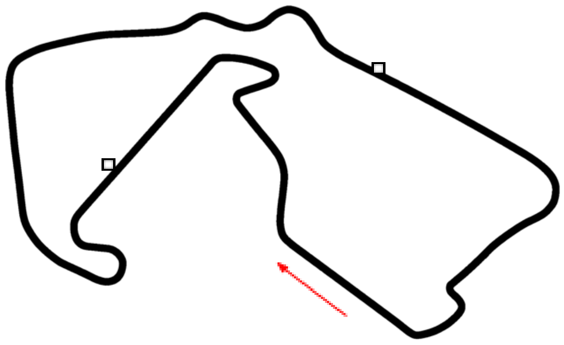


Figure 2: Silverstone Circuit with initial starting area[8]

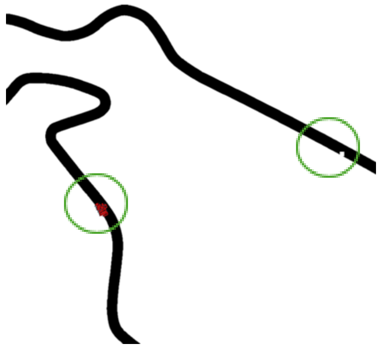


Figure 3: Silverstone Circuit top right obstacle

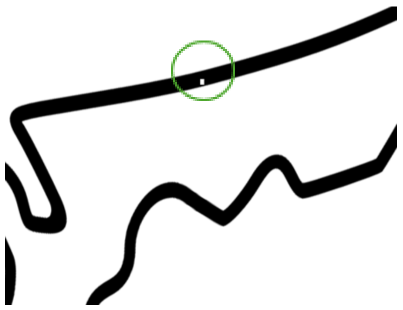
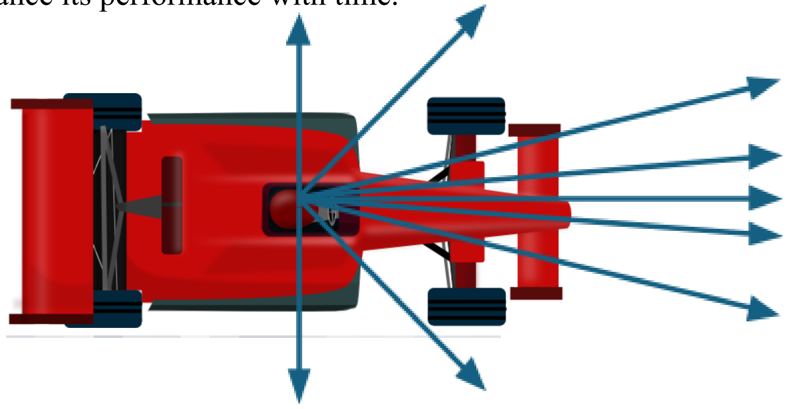


Figure 4: Austin circuit top right obstacle

There 2 different obstacles in each map that follows a straight path with an oscillating type motion. The positions are indicated in figure 1 and figure 2 respectively and the example obstacles are shown at figure 3 and figure 4.

Car Design and Control Systems

The car design uses a sprite representing a Formula 1 car, scaled to appropriate dimensions (15 x 15 pixel) for the simulation. The car has various attributes, such as position, speed, angle, and center point. It includes total of 9 sensors with -90, -45, 0, 45, 90, -15, 15, -7, 7 degree angles used to identify obstacles and boundaries. The control system is based on the neural network that determines the car's actions from the sensor inputs. The adaptive characteristic of the system allows it to learn and enhance its performance with time.



Algorithms and Pseudocode

In the project, the adaptive learning system is based on the NEAT (Neuro Evolution of Augmenting Topologies) algorithm, which is a popular approach to evolving neural networks. The algorithm adapts by making changes to the network's structure and weight, allowing it to learn from experiences.

Two different neural networks are used in this project to test the car's control system which are feed forward networks and recurrent neural networks. The feed forward network processes inputs through a straightforward path, without feedback loops to the network making it suitable for simpler decision-making tasks. Recurrent network's uses feedback loops making it suitable for more complex behaviors and memory-based decision-making.

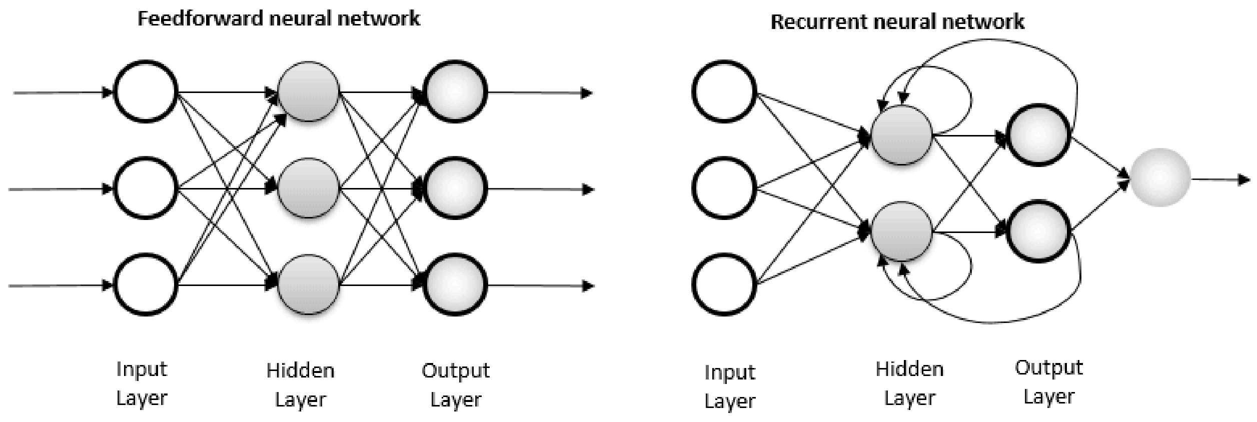


Figure 6:Feed forward and recurrent ANN architecture[10]

Table 1:Feed-Forward Neural Networks vs Recurrent Neural Networks[11]

Comparison Attribute	**Feed-forward Neural Networks**	**Recurrent Neural Networks**
Signal flow direction	Forward only	Bidirectional
Delay introduced	No	Yes
Complexity	Low	High
Neuron independence in the same layer	Yes	No
Speed	High	slow

While feed forward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network. That said, these weights are still adjusted in the through the processes of backpropagation and gradient descent to facilitate reinforcement learning.
[12].

Pseudocode is a tool that can help to demonstrate the algorithm's process and the car's control functions. Below is a simplified pseudocode representation of the car's control system:

```
Initialize neural network with NEAT configuration
For each generation:
  For each car in the population:
    Collect sensor data from the environment
    Use neural network to determine car's actions
    Update car's position based on the neural network output
    Check for collisions with boundaries or obstacles
    Calculate fitness based on distance traveled and other metrics
    If car collides or completes a lap, end its simulation
  Select the fittest cars to reproduce
  Evolve the neural network for the next generation
```

Adaptive Learning and Real-Time Decision-Making

The adaptive learning system runs reinforcement learning algorithm to decide the car's driving behavior. The fitness function gives rewards for the distance traveled by the car, with a value of +1 for each pixel moved forward. Collisions with obstacles or boundaries suffer a penalty of -10. This makes the car to navigate efficiently and avoid collisions, as maximizing the total reward is the objective of the learning process. This system evolves with each generation of the AI so that it is able to enhance its navigation and route-finding capacities. The system's real-time decision-making is the key to its success, where the neural network generates the specific actions based on the sensor data. The neural network consists of 9 inputs representing car sensors, 7 hidden layers, and 4 outputs representing turn right, turn left, speed up, and slow down.

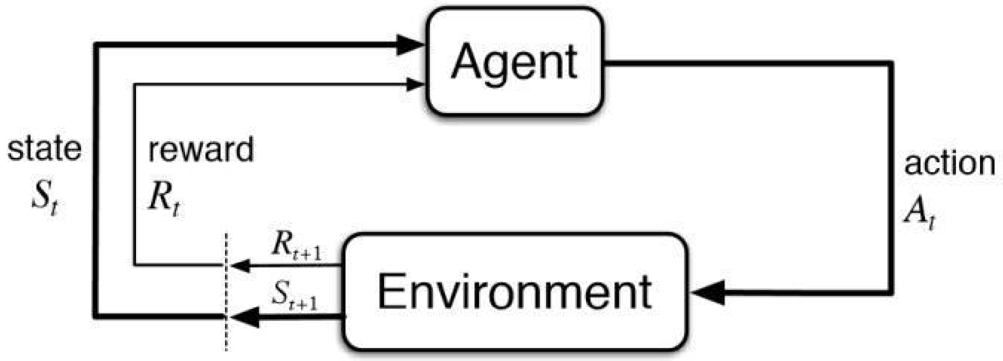


Figure 7: Flowchart of reinforcement learning[13]

Third-Party Libraries and Mathematical Models

The most important third-party library used in the simulator is Pygame which provides the graphical capabilities and event handling for the environment of the simulation. The NEAT library used for evolving neural networks with a configuration file which specifies the network parameters. The mathematical models used in the simulation involve the equations of motion for the car's control system and the sensor detection logic.

Equations of Motion for Car Control System: The car's motion can be modeled by using the very basic kinematic equations. Let x and y be the position of the car, v the velocity, and θ the orientation angle. The car's dynamics can then be described by the following equations:

$$\dot{x} = v\times \cos cos\, \theta \qquad \dot{y} = v\times \sin sin\, \theta \qquad \dot{\theta} = \omega$$

in which $\dot{x}$ and $\dot{y}$ are the rates of change of the car's in x and y axis, $\dot{\theta}$ is the rate of change of the car's orientation angle, and ω is the angular velocity. Additionally, the car's velocity v and angular velocity ω are controlled by the car's acceleration and steering commands, denoted by u_a and u_ω, respectively:

$$\dot{v} = u_a \qquad \dot{\omega} = u_\omega$$

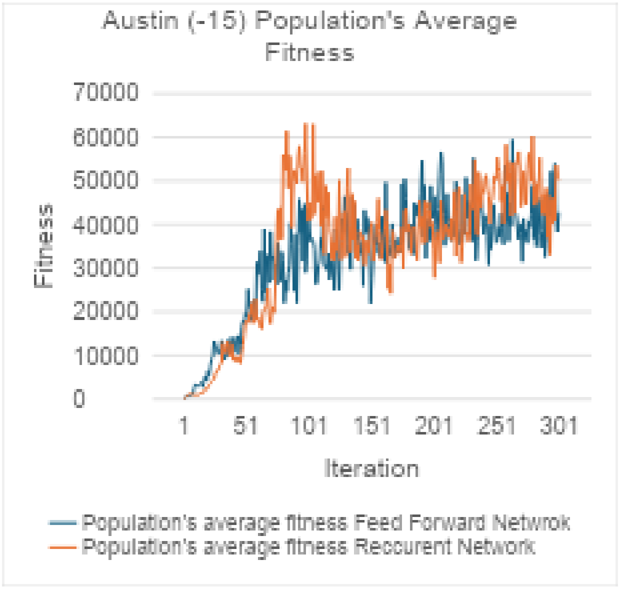
Sensor Detection Logic: The sensor detection logic involves determining the distance between the car and obstacles or boarders detected by its sensors. Let $\left(x_C, y_C\right)$ be the car's position and $\left(x_O, y_O\right)$ be the position of an obstacle. The distance d between the car and the obstacle can be calculated by using the Euclidean distance formula:

$$d = \sqrt{\left(x_O - x_C\right)^2 + \left(y_O - y_C\right)^2}$$

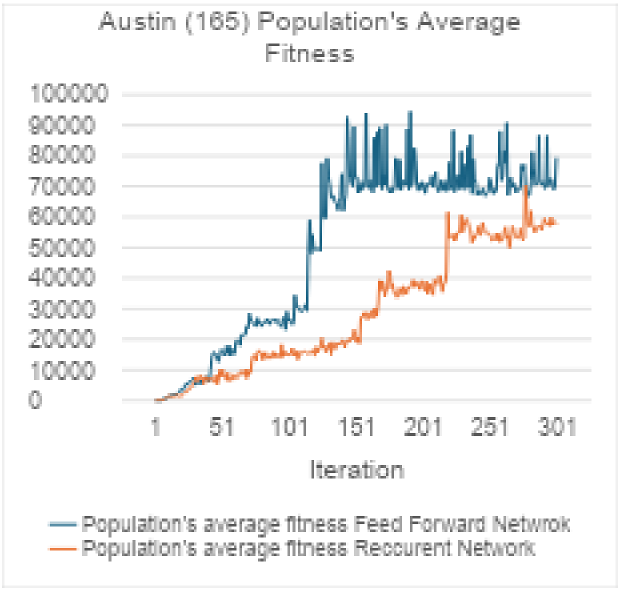
Results and Analyses

To evaluate the performance of the NEAT implementation, a series of tests were designed for recurrent and feed forward neural networks on different f1 circuits. The aim is to determine how each network adapted to different conditions and which parameters had the most significant impact on the network's performance. A total of eight tests were conducted, with four tests for each circuit. The tests were conducted with both recurrent neural networks (RNNs) and feed forward neural networks (FNNs), each of them running minimum of 300 iterations. For every test the data gathered consisted of the number of laps completed by each car, lap times of the cars, population's average fitness, standard deviation, best fitness, mean adjusted fitness, mean genetic distance, total extinctions, and generation time. The population in these tests consisted of 60 members.

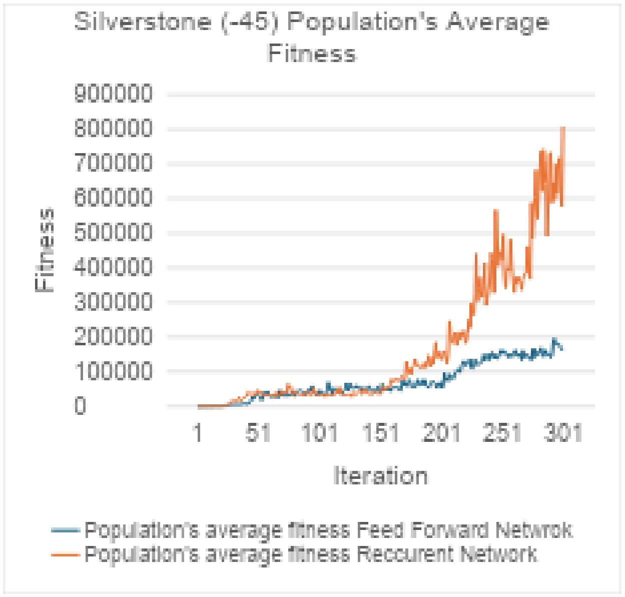
Below, there are four graphs. These graphs display the average fitness values for both Feed Forward Networks and Recurrent Networks, with each have different map and the initial race angle values which indicate the discretion of the cars starting point. These values provide insights into the performance of these networks and NeuroEvolution of Augmenting Topologies (NEAT) under different conditions.



Austin (-15) Population's Average Fitness
70000
60000
50000
40000
30000
20000
10000
0
Fitness
1
51
101
151
201
251
301
Iteration
Population's average fitness Feed Forward Netwrok
Population's average fitness Reccurent Network



Austin (165) Population's Average Fitness
100000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
1
51
101
151
201
251
301
Iteration
Population's average fitness Feed Forward Netwrok
Population's average fitness Reccurent Network



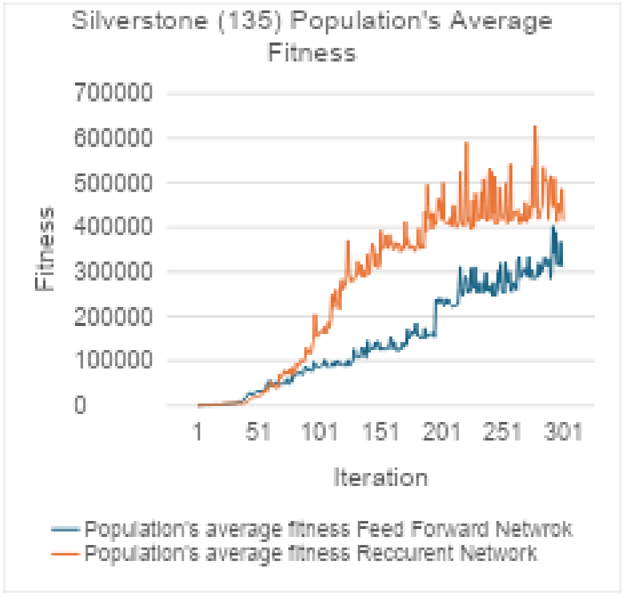


Figure 8: Average Fitness Comparisons of Silverstone and Austin Circuits with different neural networks

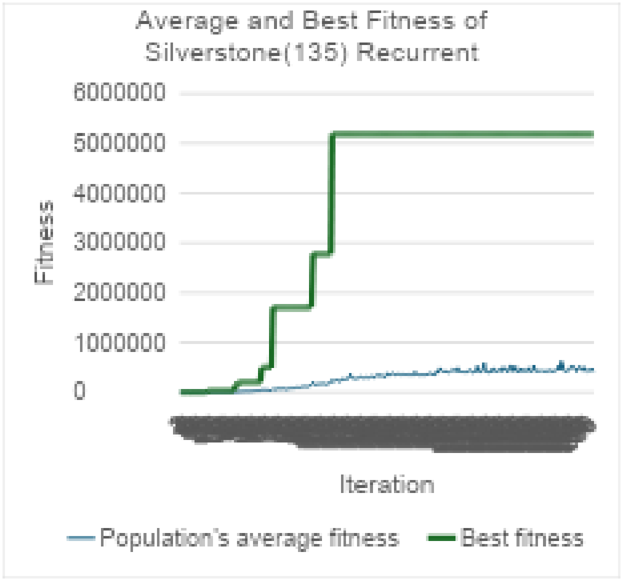
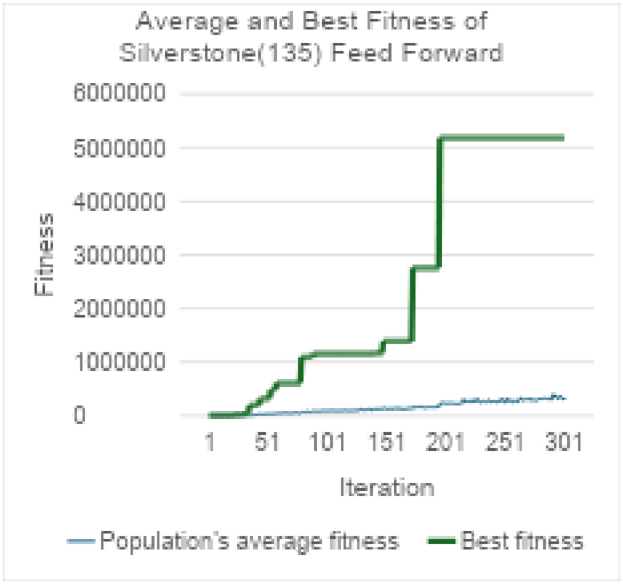
In analyzing the graphical data, it is obvious that Recurrent Network has better population's average fitness values and overall better performance compared to Feed Forward network on two out of the four test cases. Recurrent Network configuration, making use of its dynamic memory, demonstrates exceptional efficiency in the passage of the complicated routes. In the Austin circuit with a starting angle of -15 degrees the Recurrent Network demonstrates higher fitness scores at first 115 iterations. Nonetheless, as the test approach to the 300th iteration, the performance gap between the Recurrent Network and the Feed Forward Network is drastically narrowed, and eventually, the two networks become almost indistinguishable.

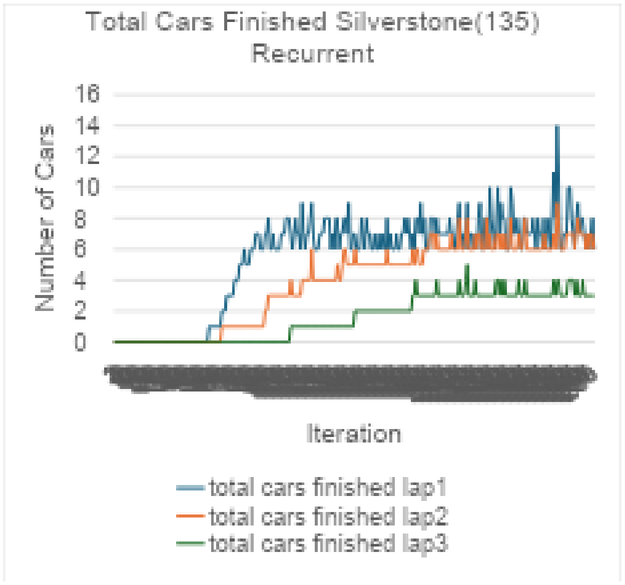
In the Austin circuit with the starting angle of 165 degrees the Feed Forward Network has been proved to be the clear winner and it has significantly outrun recurrent network. The cause of this is probably the fact that the Feed Forward Network has a simpler structure, which does not contain the feedback loops like in recurrent networks. Thus, the Feed Forward Network is advantageous when it comes to dealing with the Austin track, where it is more common to have direct input-output mappings. This architecture makes it possible for the network to process the data in a simplistic, linear way and hence the outcomes of the processing are uniform and quick. This is exactly why the Feed Forward Network reaches higher fitness in this circuit configuration that is capable of immediate responses to inputs, thus proving its effectiveness in these scenarios.

The performance differences between the Recurrent Network and the Feed Forward Network can be explained by the fact that these networks have different architectural features and how they process information. The Recurrent Network, which has dynamic memory, allows it to retain and reuse the information over time, making it ideal for scenarios with sequences. This architecture lets it take advantage of previous inputs, which make it better in complex circuits. This is the reason why it is the more efficient on the Silverstone circuit.

On the other hand, the Feed Forward Network processes inputs in a linear manner without any backward connections. This architecture suits scenarios where each input is independent from the previous ones, leading to more direct output calculations. The lack of memory mechanisms makes it less likely to overfitting to the past information, providing a clear advantage in cases where immediate reactions to inputs are critical, as observed in the Austin circuit with a 165-degree starting angle. Here, the Feed Forward Network's straightforward structure allows it to rapidly adapt to changing conditions, outperforming the Recurrent Network. There are also dynamic objects in the circuit which their positions changes as the car encounter with it and having instant decision making is beneficial where Feed forward networks has advantage.

Below, there are four graphs. Two of which, shows the data of the best fitness and average fitness for the Silverstone circuit with a 135-degree starting angle. The other two show the number of cars that completed each lap for different neural networks.


Average and Best Fitness of Silverstone(135) Recurrent
6000000
5000000
4000000
3000000
2000000
1000000
0
Fitness
Iteration
Population's average fitness
Best fitness

Average and Best Fitness of Silverstone(135) Feed Forward
6000000
5000000
4000000
3000000
2000000
1000000
0
1
51
101
151
201
251
301
Fitness
Iteration
Population's average fitness
Best fitness



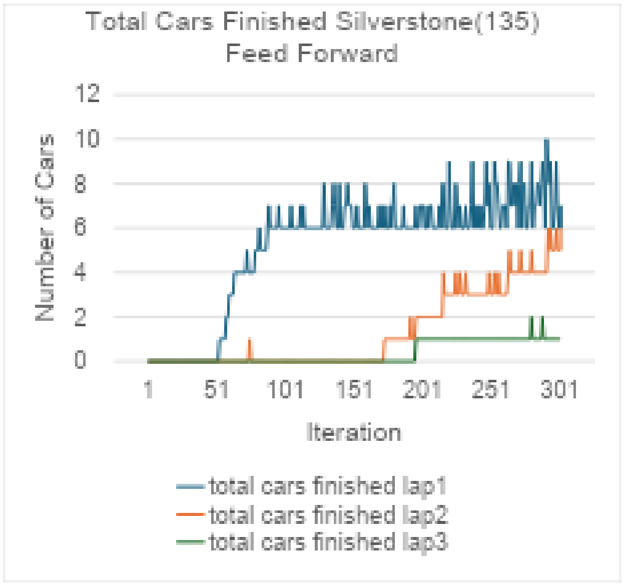


Figure 9:Silverstone (135) Recurrent Feed Forward network comparison

If analyzed Average and Best Fitness table (Figure 9), the fitness performance reveals that both Recurrent Networks and Feed Forward Networks reached the same peak fitness level of 5,184,000 within 300 iterations. However, the Recurrent Network achieved this point at the 110th iteration, while the Feed Forward Network reached it at the 195th iteration, which indicates a significant difference in adaptation speed.

Moreover, examining the first lap ever to be completed by each network, the Feed Forward Network completed the first lap at the 52nd iteration, whereas the Recurrent Network achieved it at the 59th iteration, indicating Feed Forward Network initially adapted more quickly. Similarly, the Feed Forward Network completed its first second lap at the 73rd iteration and its first third lap at the 195th iteration. On the other hand, The Recurrent Network completed its first second lap at the 67th iteration and its first third lap at the 110th iteration, showing that it required fewer iterations to master.

Despite both networks reaching the same best fitness level, their average fitness values at the 300th iteration differ, the Feed Forward Network holds an average fitness of 310,928.43556 and the Recurrent Network holds an average fitness of 310,928.43556. If examined for the total lap completions number in the 300th iteration, the Feed Forward Network had 6 cars completed the first lap, 5 cars completed the second lap, and 1 car completed the third lap. However, The Recurrent Network had 6 cars completed the first and second laps and has 3 cars completed the third lap.

Analyzing the fastest lap times in the 300th iterations, the Feed Forward Network's best lap times were 23.73 seconds for the first lap, 22.25 seconds for the second lap, and 21.81 seconds for the third lap. In contrast, the Recurrent Network's fastest lap times were 23.93 seconds for the first lap, 21.24 seconds for the second lap, and 20.90 seconds for the third lap.

Reviewing these analyses, the question of “Which model is better?” or the choice of which model to use eventually depends on the specific requirements of the task. If the objective is to determine which model could complete the first lap the fastest, a feedforward neural network would be the best choice, as it learns more quickly according to the results found in this project. However, if the goal is to cover the most distance and minimize collisions possibility, then the recurrent neural network would be the better solution. Despite considering the first 300 iterations, if all the test cases have more than 1000 iterations, the recurrent network would have better results. But if the iteration is let to infinity, then feed forward network world be better for new encounter since recurrent network would be overfitted. By utilizing the NEAT (Neuro Evolution of Augmenting Topologies) algorithm, the system was able to adapt appropriately regarding to the project requirements in both neural networks, effectively achieving the desired outcomes.

The outcomes of this results shows how the NEAT algorithm, used for the evolutionary framework in this project, lets the adaptive system to enhance itself over time. It affects the structure and behavior of neural networks, making them capable of collision avoidance and pathfinding in 2D self-driving car simulation.

Discussion

This project on adaptive navigation for 2D self-driving car simulation produced valuable information about the capabilities and limitations of adaptive systems. This part discusses the details of the findings, highlighting potential directions for further research and possible improvements to increase the adaptability of the system.

Implications for Future Research

Results of the study can be considered as the starting point for the further study, each of which is aimed at improving the understanding of adaptive systems and their application in autonomous vehicles. There are 3 possible further study that is going to be discussed.

Extended Simulations:

As discussed in the Results and Analyses section, Recurrent neural networks (RNNs) are described as being capable of retaining memory, which makes them quite useful for longer simulation periods. Furthermore, data observed demonstrates that RNNs show better performance than FNNs in the long iterations. To gain insight into this phenomenon, simulation with longer duration can be done. This would provide a more comprehensive assessment of the networks' ability to adapt, evolve, and improve their performance across extended sessions. In addition, this would reveal how the memory skills of RNNs contribute to their success in multi-dimensional, evolving environments. Having longer iterations may make the system converged to overfitting which should be considered while doing this future study.

Increased Environmental Complexity:

One of the most important part of the autonomous vehicle performance is ability to adjust to the different and unstable environments. This research was carried out in a relatively controlled environment, where each track layout was predetermined, and the path of the obstacles were fixed. Further research focusing on increasing the environmental complexity, such as by introducing variable weather conditions with giving the car ability to slip, dynamic obstacles, and a more complicated track design. These modifications would create the unpredictability that is similar to the real-life driving situations. This is also important in the evaluating the system's resilience and flexibility in the face of unseen events.

Multi-Objective Optimization:

Autonomous driving systems are designed to balance many objectives like speed, safety and efficiency. A different future research topic may be exploring multi-objective optimization strategies. This involves developing adaptive algorithms that can simultaneously consider different objectives, allowing the system to make trade-offs based on current conditions. This approach could improve the system's overall performance, ensuring it meets safety standards while maximizing efficiency.

Adaptive System Analysis

The adaptive system used in this study showed flexibility and adaptability, which are the main features that ensure the safety of autonomous vehicles. The main focus of the reinforcement learning principle, which encourages distance traveling and discourages the collisions by reward and penalty, plays a central role in guiding the system's behavior. This real-time feedback loop enables the system to get the information, learn and further its actions based on environmental cues, and therefore, create a continuous improvement which can be seen in the results section.

On the other hand, the outcome also illustrates that recurrent neural networks, having ability to store the data, can be more vulnerable to overfitting overall, especially when they come across repetitive or limited scenarios. Furthermore, it arises the need to conduct a study on the methods to achieve both overfitting reduction and the continuity of recurrent neural networks. In contrast, feedforward neural networks outperforms in simpler linear problems and their use cases are limited to cases where instant reactions are key. This differentiation in performance underlines the importance of selecting the appropriate neural network architecture based on the complexity and nature of the driving circuit and moving obstacles.

Overall, findings from the experiment point to the adaptive system's ability to learn and respond to changes by adjusting its behavior. Furthermore, flexibility is one of the essential features for autonomous vehicle technology that requires the vehicle to navigate complex environments with a maximum level of safety and efficiency. By extending the scope of future research and refining parameter tuning, which means adding more parameters as well as adjusting them, the system's adaptability can be further improved, contributing to advancements in autonomous vehicle technology.

Connection to the Broader Field

The project is directly linked to the adaptive systems research field, as it is one of the new technologies that employs artificial intelligence (AI) and learning algorithms in autonomous cars. Therefore, adaptive systems that provide autonomous driving are based on complex AI integration across multiple subsystems, such as perception, planning and controls, in order to maintain safe and stable operations. Artificial intelligence powered testing and code analysis tools benefit the autonomous vehicle software by identifying bugs and weaknesses, which eventually lead to the safe and reliable software. These adaptive systems, which ensure predictive maintenance, real-time monitoring, and reduced downtime, and enhancing operational effectiveness[14].

Moreover, adaptive systems are crucial in the control of autonomous vehicles with AI-based motion forecasting and control algorithms that enable efficient navigation in different conditions. These adaptive algorithms optimize trajectories, minimize the computational requirements, and enhance the disturbance rejection. They are used in many situations such as warehouses and underwater, which demonstrates the adaptability and durability of these systems[15,16]. In this respect, the experiment's concentration on adaptive learning in autonomous vehicle simulations correlates with other research trends that point out AI's importance in enhancing safety, robustness, and efficiency in many applications.

Hypothesis

Finally, the experiment aimed to test the hypothesis which answers the following question "Whether different types of neural networks would show different levels of achievement in adaptive navigation algorithms within the 2D self-driving car simulation or not?". In particular, the purpose of the hypothesiswas to find out if the recurrent neural networks (RNNs) would show better adaptability using memory retention than the feedforward neural networks (FNNs) with a more straightforward approach, in a simulated environment which included complex road paths and moving obstacles.

Hypothesis Testing
The results from the experiment supported the hypothesis in several key areas:

Performance Over Time: Recurrent neural networks, which are notable by their ability to retain memory, demonstrated outperforms the feed forward network. The test data (Figure 8 and Figure 9) shows that, in general, RNNs outperformed FNNs, in the simulations if the duration was long enough, and the environment had more complex conditions.

Collision Avoidance and Pathfinding: RNNs demonstrated a higher ability to navigate through challenging paths and avoid collisions. This outcome can be interpreted that the feedback characteristic in RNNs allow the network to retain and reuse information, enhancing their adaptability in scenarios where memory-based decision-making is critical. But in some cases as the iteration goes infinity RNNs is overfitted which makes FNN better choice.

Feedforward Neural Networks: Although the hypothesis favored RNNs, looking at the results, FNNs performed well in simpler, more linear environments. This outcome proposes that FNNs, with its straightforward structure, which is better suited for tasks that require immediate reactions to inputs, making it advantageous in specific scenarios where a direct approach is beneficial like moving obstacles.

Implications
The experiment's results suggest that while RNNs generally outperform FNNs in complex environments if there is enough iteration in simulations, FNNs can still be valuable in specific contexts where simplicity and speed are critical because of the RNNs overfits. This supports the hypothesis that different neural network architectures have unique strengths, and their suitability depends on the nature of the task and environment.

The results of this project reveal the effects of different neural network methods on the autonomous vehicle in the simulation environment. The effects of different methods on different tracks and obstacles are seen. Thanks to the use of different tracks and methods, results can be compared across track types in the short term and long term.

Reference

[1] W. Black, P. Haghi, and K. Ariyur, “Adaptive Systems: History, techniques, problems, and perspectives,” *Systems*, vol. 2, no. 4, pp. 606–660, Nov. 2014. doi:10.3390/systems2040606

[2] “Model-free adaptive control of Uncertain Autonomous Systems,” Frontiers, https://www.frontiersin.org/research-topics/50724/model-free-adaptive-control-of-uncertain-autonomous-systems (accessed Apr. 26, 2024).

[3] Q. Li, J. P. Queralta, T. N. Gia, Z. Zou, and T. Westerlund, “Multi-sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments,” *Unmanned Systems*, vol. 08, no. 03, pp. 229–237, Jul. 2020. doi:10.1142/s2301385020500168

[4] N. Goel, “Simulation, scenarios, and self-driving – the SAE AutoDrive Way,” Student Lounge, https://blogs.mathworks.com/student-lounge/2021/03/11/simulation-scenarios-and-self-driving-the-sae-autodrive-way/ (accessed Apr. 26, 2024).

[5] Shahin Atakishiyev Mohammad Salameh Hengshuai Yao Randy Goebel Department of Computing Science et al., “Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for Future Research Directions,” ar5iv, https://ar5iv.labs.arxiv.org/html/2112.11561 (accessed Apr. 26, 2024).

[6] Christian Häne chaene@eecs.berkeley.edu Lionel Heng lionel_heng@dso.org.sg Gim Hee Lee gimhee.lee@nus.edu.sg Friedrich Fraundorfer fraundorfer@icg.tugraz.at Paul Furgale paul.furgale@mavt.ethz.ch Torsten Sattler sattlert@inf.ethz.ch Marc et al., “3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection,” ar5iv, https://ar5iv.labs.arxiv.org/html/1708.09839 (accessed Apr. 26, 2024).

[7] “F1 circuits 2014-2018 - Circuit of the Americas,” openclipart, https://openclipart.org/detail/291803/f1-circuits-20142018-circuit-of-the-americas (accessed Apr. 26, 2024).

[8] “Silverstone png,” cleanpng.com, https://www.cleanpng.com/png-silverstone-circuit-formula-one-information-diagra-915311/ (accessed Apr. 26, 2024).

[9] Makecodediy, “2d formula 1 cars,” OpenGameArt.org, https://opengameart.org/content/2d-formula-1-cars (accessed Apr. 26, 2024).

[10] Feed forward and recurrent Ann Architecture. | download scientific diagram, https://www.researchgate.net/figure/Feed-forward-and-recurrent-ANN-architecture_fig1_315111480 (accessed Apr. 26, 2024).

[11] GeeksforGeeks, “Difference between feed-forward neural networks and recurrent neural networks,” GeeksforGeeks, https://www.geeksforgeeks.org/difference-between-feed-forward-neural-networks-and-recurrent-neural-networks/ (accessed Apr. 26, 2024).

[12] “What are recurrent neural networks?,” IBM, https://www.ibm.com/topics/recurrent-neural-networks#:~:text=While%20feedforward%20networks%20have%20different,descent%20to%20facilitate%20reinforcement%20learning (accessed Apr. 26, 2024).

[13] “5 things you need to know about reinforcement learning,” KDnuggets, https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html (accessed Apr. 26, 2024).

[14] D. Garikapati and S. S. Shetiya, “Autonomous vehicles: Evolution of artificial intelligence and the current industry landscape,” *Big Data and Cognitive Computing*, vol. 8, no. 4, p. 42, Apr. 2024. doi:10.3390/bdcc8040042

[15] B. Li, X. Chen, T. Acarman, X. Li, and Y. Zhang, “Recent advances in motion planning and control of Autonomous Vehicles,” *Electronics*, vol. 12, no. 23, p. 4881, Dec. 2023. doi:10.3390/electronics12234881

[16] J. Yuan, Y. She, Y. Zhang, J. Xu, and L. Wan, “Research on L1 adaptive control of autonomous underwater vehicles with X-rudder,” *Journal of Marine Science and Engineering*, vol. 11, no. 10, p. 1946, Oct. 2023. doi:10.3390/jmse11101946

Bibliography

[1] “Welcome to neat-python’s documentation!,” NEAT, https://neat-python.readthedocs.io/en/latest/ (accessed Apr. 26, 2024).

[2] Adaptive Decision-Making for Autonomous Vehicles: A Learning-Enhanced Game-Theoretic Approach in Interactive Environments: https://arxiv.org/abs/2402.11467

[3] A NEAT REINFORCEMENT LEARNING JOURNAL FOR SELF-DRIVING AI: https://www.irjmets.com/uploadedfiles/paper/issue_6_june_2023/42451/final/fin_irjmets1687682258.pdf

[4] M. Wooldridge, An Introduction to MultiAgent Systems, 2nd ed. Hoboken, NJ, USA: Wiley, 2009.

[5] “Pygame front page,” Pygame Front Page - pygame v2.6.0 documentation, https://www.pygame.org/docs/ (accessed Apr. 26, 2024).

[6] S. Perla, N. N. K, and S. Potta, “Implementation of autonomous cars using machine learning,” *2022 International Conference on Edge Computing and Applications (ICECAA)*, Oct. 2022. doi:10.1109/icecaa55415.2022.9936102