Evaluating Self-Driving Car Performance with RNNs and Genetic Algorithm Optimization

Gautam Sethia
Computer Science and Engineering
Indian Institute of Technology Ropar
Rupnagar, Punjab, 140001, India
2020csb1086@iitrpr.ac.in

Ojassvi Kumar
Computer Science and Engineering
Indian Institute of Technology Ropar
Rupnagar, Punjab, 140001, India
2020csb1187@iitrpr.ac.in

Prakhar Saxena
Computer Science and Engineering
Indian Institute of Technology Ropar
Rupnagar, Punjab, 140001, India
2020csb1111@iitrpr.ac.in

***Abstract*—In the realm of self-driving cars, addressing the challenges posed by dynamic environments and the inherent unpredictability of sensor data remains a pivotal concern. This research explores the performance of Recurrent Neural Networks (RNNs) and Artificial Neural Networks (ANNs) with Genetic Algorithm (GA) optimization for self-driving car control. Contrary to initial expectations, early indications reveal that while RNNs offer tailored solutions for dynamic and time-sensitive data, they may require a higher number of generations to converge compared to ANNs. However, the convergence of RNNs is notably smoother and more stable, making them well-suited for dynamic environments. Utilizing Unity environments, the study systematically evaluates these approaches across diverse tracks featuring challenging routes and dynamic obstacles.**

***Index Terms*—Self-driving cars, Recurrent Neural Networks (RNNs), Artificial Neural Networks (ANNs), Genetic Algorithms (GAs), and Unity simulation.**

I. Introduction

In the realm of self-driving cars, addressing the challenges of dynamic environments and high levels of randomness in sensor data remains a pivotal concern. Traditional neural networks, while promising, struggle in the face of the inherent unpredictability of dynamic settings.

Previous approaches incorporated Artificial Neural Networks (ANNs) for self-driving car control, coupled with Genetic Algorithms (GAs) for optimization [1]. This method sought optimal hyperparameters to enhance control aspects. Despite successes, it faced challenges like track-specific training and difficulties in handling dynamic environments.

In response, our research innovatively evaluates and compares the performance of Recurrent Neural Networks (RNNs) and Artificial Neural Networks (ANNs) with Genetic Algorithm (GA) optimization. Initially, our hypothesis was that RNNs, being specifically designed for sequential data processing, would outperform ANNs, particularly in the context of the continuous and time-sensitive nature of self-driving car sensor inputs. However, our findings reveal nuanced differences in their performance.

Contrary to our initial expectations, initial findings suggest that Recurrent Neural Networks (RNNs) may face certain limitations in specific scenarios. Although RNNs offer a specialized solution for handling dynamic and time-sensitive data, our results indicate a potentially higher requirement for generations to achieve convergence compared to Artificial Neural Networks (ANNs). Despite this, it is noteworthy that the convergence process for RNNs exhibits a smoother and more stable trajectory, showcasing their suitability for dynamic environments.

Employing Unity environments, our methodology systematically explores the intricacies of this approach, considering the performance of ANNs and RNNs with GA optimization across diverse Unity tracks featuring challenging routes and dynamic obstacles.

II. Literature Review

The landscape of self-driving car research has witnessed notable contributions from diverse studies, each shedding light on unique aspects of autonomous vehicle control. In the seminal work by Chaudhari et al. (2018) [2], the utilization of Convolutional Neural Networks (CNNs) for training AI cars in autonomous driving scenarios was introduced. Leveraging the virtual environment of GTA 5, this approach showcased the potential of CNNs in extracting meaningful information from an open-world game for training purposes.

Gupta et al. (2021) [3] provided a comprehensive survey on the theoretical foundations of self-driving vehicles, coupled with insights into recent implementations of deep learning for object detection and scene perception. This survey not only elucidated the current state of the art but also pinpointed challenges and open issues in the realm of self-driving car technology.

The work by Song et al. (2021) [4] took a distinctive approach by incorporating curriculum reinforcement learning to train high-speed autonomous racecars for overtaking maneuvers in Gran Turismo Sport. This novel methodology showcased advantages over standard reinforcement learning, providing a nuanced perspective on training strategies.

In the domain of neuroevolution, Maresso (2018) [5] presented a generalized process for training neural networks, demonstrating emergent behavior in controlling video games or simulations. This approach highlighted the potential of neuroevolutionary techniques in imparting adaptive behavior to self-driving car agents without the need for predefined training data or hard-coded behaviors.

Garnica et al. (2020) [6] explored the realm of autonomous virtual vehicles using FNN-GA and Q-learning in a Unity 2D environment. The study delved into the intricacies of these algorithms, emphasizing their dependence on track designs, evaluation functions, and reward systems.

Fuchs et al. (2021) [7] showcased super-human performance in Gran Turismo Sport using deep reinforcement learning, presenting an autonomous car racing policy that achieved high performance in time trial settings without human intervention. This work underscored the potential of deep reinforcement learning in achieving superior performance metrics.

Haromainy et al. (2023) [8] addressed the challenges of effectively configuring Recurrent Neural Networks (RNNs) for time series forecasting, specifically in the context of stock price prediction. They propose the use of Genetic Algorithms (GAs) for optimizing the hyperparameters of RNNs, such as the number of hidden layers and the learning rate. The results show that the optimized RNN outperforms the baseline, achieving a lower Root Mean Squared Error (RMSE), indicating improved accuracy in predicting stock prices.

Despite these commendable advancements, existing literature reveals certain gaps. There is a limited exploration of dynamic environments characterized by high levels of randomness in self-driving car scenarios. Additionally, the dependence on hit-and-trial methods for weight optimization, the necessity for track-specific training requiring retraining for different environments, and challenges associated with real-time traffic scenarios and overtaking maneuvers represent significant gaps in the current body of knowledge.

Emerging trends in the literature point toward the integration of neuroevolution techniques for training neural networks in self-driving scenarios, an exploration of curriculum reinforcement learning for enhanced training strategies, and the application of FNN-GA and Q-learning in Unity 3D for autonomous vehicle simulations. These trends hint at a dynamic and evolving field, poised for further exploration and advancements.

III. Preliminaries

Following are the foundational concepts and methodologies that underpin our approach for evaluating self-driving car performance through the integration of Recurrent Neural Networks (RNNs) and Genetic Algorithm (GA) optimization.

A. *Neural Networks in Autonomous Driving*

B. *Recurrent Neural Networks (RNNs) for Sequential Data*

RNNs are a class of neural networks designed to handle sequential data, making them well-suited for the dynamic nature of self-driving car sensor inputs. Unlike feedforward neural networks, RNNs maintain an internal memory state, allowing them to capture temporal dependencies in sequential data. This capability is crucial for understanding the evolving state of the environment over time.

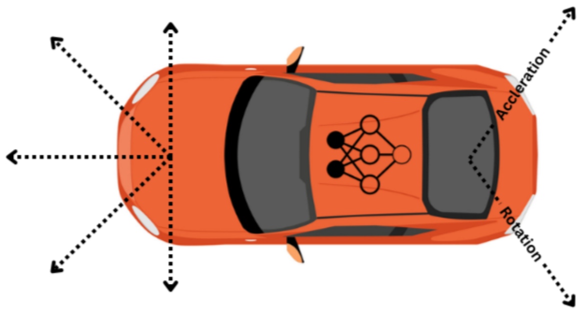

Fig. 1. A car with its different parts. The sensors give information to the neural network that can dictate the values of the wheels.

C. *Genetic Algorithms (GAs) for Optimization*

Genetic Algorithms, inspired by natural selection, optimize neural network architectures and parameters iteratively. They evolve populations of solutions through selection, crossover, and mutation processes, ideal for determining optimal configurations in complex problems like self-driving car applications.

D. *Integration of RNNs and ANNs with GAs*

Our model integrates RNNs and ANNs with GAs to leverage their respective strengths. RNNs excel in processing sequential data, aligning with the time-sensitive nature of self-driving car sensor inputs. ANNs handle complex patterns within non-sequential data. GAs automate the optimization of RNN and ANN architectures, exploiting their adaptability in exploring and optimizing solution spaces.

E. *Unity Environment for Simulation*

Experimentation utilizes the Unity environment for simulating self-driving car scenarios. Unity offers a versatile platform to create realistic virtual environments, enabling assessment under various conditions. It facilitates the development of challenging tracks with dynamic obstacles, providing a robust testing ground for the ANN-GA and RNN-GA model adaptability and efficiency.

IV. Methodology

This section outl ines the algorithmic framework, workflow, and architectural intricacies of the proposed model designed to enhance self-driving car performance through the synergistic integration of Neural Networks and Genetic Algorithm (GA) optimization.

A. *The Car*

The simulation involves the navigation of a car through a virtual environment, comprising a road, static objects, and dynamic elements. The starting position marks the point where cars spawn at the beginning of each generation. Crucially, the car must remain on the road throughout the journey. Deviating

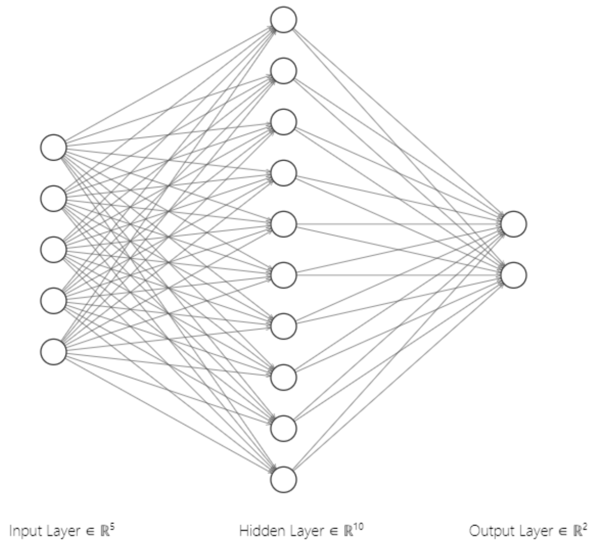

Fig. 2. ANN Architecture.

from the road or colliding with objects results in the car being marked as 'eliminated', causing the simulation to restart.

The car consists of three essential components: sensors, a neural network, and wheels, as depicted in Figure 1. The sensors actively gather information about the environment, measuring distances from surrounding objects and gauging the proximity of the car to the road edges. Subsequently, this sensor data is relayed to the neural network. Following the reception of sensor input, the neural network undergoes calculations based on its unique implementation, assigning values to each of the two back wheels. These assigned values determine the speed and the direction in which the wheels move, ultimately influencing the car's trajectory within the simulated environment.

B. Architecture of the ANN

The Artificial Neural Network (ANN) in our model is structured with five neurons in the input layer, each corresponding to a specific sensor. A hidden layer follows, comprising ten neurons. The network concludes with two output neurons with tanh activation, where one dictates acceleration, and the other governs the degree of rotation. This architecture allows the network to process information from the sensors, perform calculations in the hidden layer, and generate output values that influence the car's acceleration and steering within the simulated environment.

C. Architecture of the RNN

The architecture of the Recurrent Neural Network (RNN) closely mirrors that of the Artificial Neural Network (ANN), with a key distinction: in the hidden layer of the RNN, each neuron establishes a recurrent connection with itself. This unique feature allows the RNN to incorporate memory and capture temporal dependencies, distinguishing it from the static nature of traditional feedforward neural networks like ANNs.

D. Genetic Algorithm

- Initialization: The genetic algorithm commences with the initialization of the initial population of neural networks with random weights.
- Generation Loop: The algorithm executes iterations over multiple generations until it reaches the predefined maximum number of generations specified.
- Evaluation: For each generation, the performance of individual neural networks is assessed. The *Death* method evaluates each network's fitness based on its performance, resetting the environment for subsequent evaluations.
- Repopulation and Selection: After evaluating the population, the algorithm selects the best-performing individuals as parents for the next generation. The algorithm selects high-fitness individuals, performing crossover, and mutation, and generates a new population.
- Crossover: Crossover, a genetic operator, combines the genetic information of two parents to create new individuals (children). The Crossover method randomly exchanges genetic material (weights and biases) between parent neural networks to produce offspring.
- Mutation: Mutation introduces random changes in genetic material to maintain diversity. The *Mutate* method introduces random changes in weights and biases of child neural networks.
- Termination: The algorithm iterates through generations until it reaches the predefined maximum number of generations specified.
- Results and Analysis: At the conclusion of the algorithm, the performance and characteristics of the best-performing neural network in the final generation are analyzed.

E. Fitness Equation

The fitness value of a neural network indicates its performance. For our solution, the fitness function depends on the following :

- Total distance travelled by the car (D)
- Average speed of the car (V)
- Average sensor values (S)

The fitness score (F) is calculated as:

$$F = D \times w_d + V \times w_v + S \times w_s$$

where w_d, w_v, w_s are the respective weights for the three parameters.

F. Model Workflow

- Data Flow: Input data includes the configuration of the hyper-parameters for the genetic algorithm. Output data comprises fitness values across generations and the final evolved neural network.
- Roles of Components: The *GeneticAlgorithmManager* orchestrates the genetic algorithm's workflow. It manages

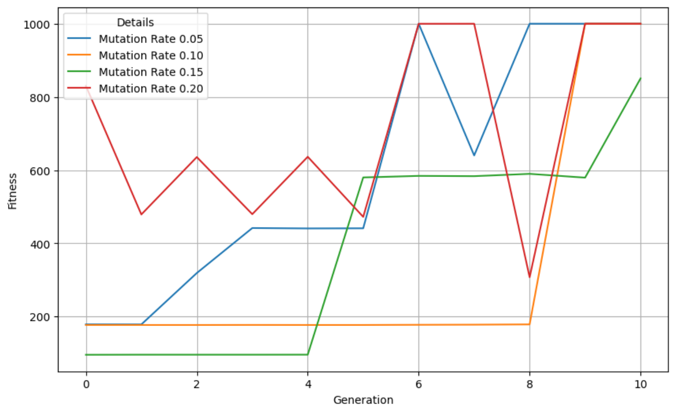

Fig. 3. Fitness over Generations for different Mutation rates (ANN).

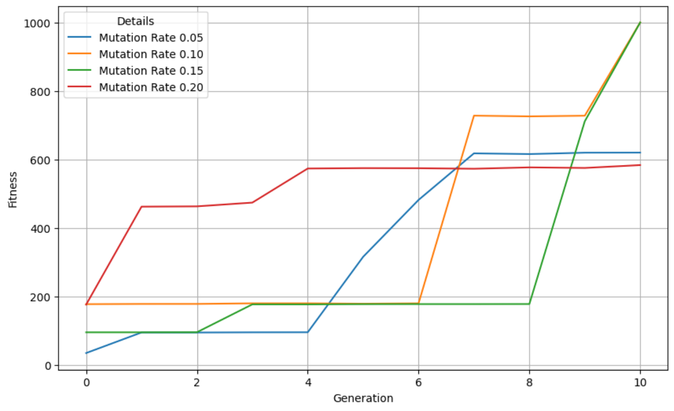

Fig. 5. Fitness over Generations for different Mutation rates (RNN).

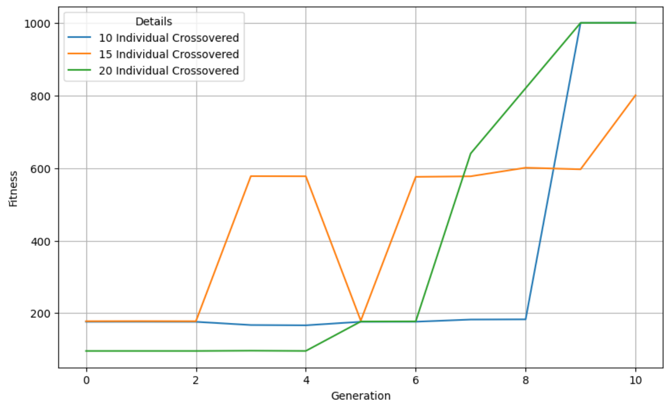

Fig. 4. Fitness over Generations for different Crossover rates (ANN).

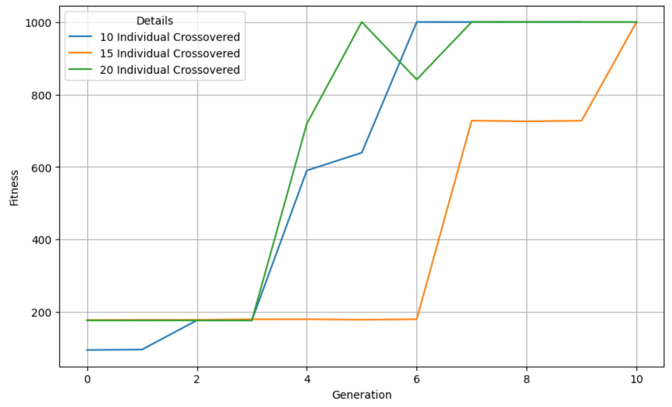

Fig. 6. Fitness over Generations for different Crossover rates (RNN).

the initialization, evaluation, selection, and evolution of neural networks. The neural network controls the movement of the car.

- Interactions: The genetic algorithm manages populations through crossover, mutation, and best selection operations and returns the fitness scores across generations and the best individual. Neural networks interact with the environment during the evaluation phase.

V. Experimentation

A. Experimental Setup

The experimentation environment employed Unity, providing a dynamic and realistic setting for the evaluation of self-driving car behaviors. The Unity environment consisted of a track with dynamic obstacles, introducing variability through random positions and movements. Simulation termination criteria included collisions with obstacles or track deviations, resulting in the lowest fitness score assignment for the affected member.

The complete codebase and setup instructions can be found at the following GitHub repository: **GARNNs**. This repository includes the entire code implementation, facilitating reproducibility and further exploration of the research methodology.

B. Mutation Rate

In the context of our experimentation within the Unity environment, Figure 3 illustrates the dynamic changes in fitness rates across generations specifically for Artificial Neural Networks (ANNs). Concurrently, Figure 5 visually represents the variations in fitness rates over successive generations, focusing on the performance of Recurrent Neural Networks (RNNs). Notably, the mutation rate, ranging from 0.05 to 0.2, is systematically adjusted to observe its impact on both the fitness rate and convergence.

Within each generation, our experimental setup comprises a total population of 30 individuals. Among these, 15 individuals with the highest fitness scores from the preceding generation are selected to continue, while the remaining 15 are generated through crossover operations. This evolutionary approach ensures a continuous exploration of the solution space. The simulation unfolds over a span of 10 generations, allowing for an in-depth analysis of the adaptive behaviors and optimization trends exhibited by the neural network architectures under consideration.

C. Crossover Rate

In a similar direction, the Figure 4 provides a graphical representation of how the fitness rate evolves across generations with variations in the crossover rate specifically for Artificial Neural Networks (ANNs). Similarly, Figure 6 depicts the analogous trends for Recurrent Neural Networks (RNNs).

To investigate the influence of crossover operations, we systematically vary the number of individuals undergoing crossovers, ranging from 10 to 20. This parameter adjustment allows us to observe the impact of crossover rates on the adaptive capabilities and convergence behavior of both ANNs and RNNs. It is important to note that, throughout these experiments, the mutation rate is held constant at 0.1. The simulation unfolds over 10 generations, offering a comprehensive view of how different crossover rates affect the fitness rates and overall performance of the neural network models.

VI. Results and Discussions

The plots indicate that lower mutation rates contribute to increased stability in both Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs), as expected, given that higher mutation rates can induce more significant individual changes. The convergence rates for both architectures appear comparable, with instances suggesting that ANNs may converge faster in specific scenarios. Notably, the plots for RNNs exhibit smoother trajectories, indicating that the impact of random changes does not lead to catastrophic effects on the neural network. This enhanced stability in RNNs is attributed to their inherent memory, which allows them to capture temporal dependencies and maintain continuity during optimization processes. Similarly, in the context of variations in crossover rates, similar stability patterns emerge, yet RNNs demonstrate a tendency to converge faster in this scenario. However, the models exhibit a faster convergence when subjected to larger crossover rates, and this phenomenon can be attributed to the increased scope for randomness introduced by higher crossover rates.

It is important to note that the initial weights for the neural networks are generated randomly at the onset of each simulation. This introduces a source of variability across every simulation, impacting the starting conditions for the evolutionary process. The randomness in the initial weights contributes to the diversity of the initial population and can play a significant role in shaping the trajectories of convergence for ANNs and RNNs.

VII. Conclusions and Future Work

In concluding our study on the integration of Recurrent Neural Networks (RNNs) and Artificial Neural Networks (ANNs) with Genetic Algorithm (GA) optimization for self-driving car control, we find that RNNs, tailored for dynamic data, exhibit a potential drawback in requiring more generations for convergence compared to ANNs. Despite this, the observed smoother and more stable convergence of RNNs underscores their suitability for dynamic environments. Future research should delve deeper into optimizing mutation rates and other hyperparameters for enhanced convergence efficiency in both architectures. Further exploration across a diverse set of dynamic scenarios and the integration of real-world data could provide a more practical understanding. Investigating hybrid models that leverage the strengths of RNNs and ANNs represents a promising avenue for advancing self-driving car performance, laying the foundation for future studies refining the application of neural network architectures in autonomous vehicles.

References

[1] Sharma, Singhal et al. ”Car Racing Game AI Using Reinforcement Learning.” IPEC Journal of Science & Technology, Vol. 02 (01), June 2023.

[2] Chaudhari, Raj, et al. ”Autonomous driving car using convolutional neural networks.” 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE, 2018.

[3] Gupta, Abhishek, et al. ”Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues.” Array 10 (2021): 100057.

[4] Song, Yunlong, et al. ”Autonomous overtaking in gran turismo sport using curriculum reinforcement learning.” 2021 IEEE international conference on robotics and automation (ICRA). IEEE, 2021.

[5] Maresso, Brian. Emergent behavior in neuroevolved agents. Diss. University of Wisconsin–Whitewater, 2018.

[6] Garnica, Carlos Andrés, Juan Carlos Barrero, and Miryam Liliana Chaves. ”Autonomous virtual vehicles with FNN-GA and Q-learning in a video game environment.” 2020 Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI). IEEE, 2020.

[7] Fuchs, Florian, et al. ”Super-human performance in gran turismo sport using deep reinforcement learning.” IEEE Robotics and Automation Letters 6.3 (2021): 4257-4264.

[8] Muhammad Muharrom Al Haromainy, Dwi Arman Prasetya, Anggraini Puspita Sari. ”Improving Performance of RNN-Based Models With Genetic Algorithm Optimization For Time Series Data.” TIERS Information Technology Journal (2023).