Self-Driving Car Simulation Using a Simple Neural Network

Ayush Chamoli
Student of graphic era hill university
Program: Bachelor of Technology
Branch: CSE
Sec: k1 / AI&ML. Roll No: 2318580 / 66

Abstract:

The advancement of self-driving car technologies has led to a growing interest in autonomous systems that can navigate complex environments without human intervention. This paper presents the design, development, and implementation of a self-driving car simulation using a simple neural network. The aim is to demonstrate the potential of neural networks, particularly basic feedforward architectures, in solving the problem of autonomous driving. By leveraging sensor data to detect road boundaries and nearby traffic, the network can make decisions such as steering, braking, and acceleration. In this research, we focus on creating a simulated environment where the neural network can be trained and tested for its ability to autonomously drive a car without colliding with obstacles or veering off the road. We evaluate the performance of the system under various conditions, demonstrating the simplicity and efficiency of neural networks in the context of self-driving cars.

Keywords: Neural Networks, Self-Driving Car, Autonomous Vehicles, Simulation, Decision Making, Reinforcement Learning, Car Simulation.

INRODUCTION

The concept of self-driving cars, also referred to as autonomous vehicles (AVs), has become a key area of research and development in the field of artificial intelligence (AI) and robotics. Autonomous vehicles are expected to revolutionize transportation by enhancing safety, improving traffic efficiency, and reducing human error. Self-driving cars rely heavily on advanced algorithms and deep learning models to make real-time decisions based on sensory input. These models interpret data from cameras, LIDAR sensors, GPS, and radar systems to navigate the vehicle through complex environments, which may involve obstacles, other vehicles, pedestrians, and traffic signals.

Despite the complexity of autonomous driving, early models relied on simpler systems, such as rule-based approaches, where specific pre-defined rules were used to handle various driving scenarios. However, with the introduction of machine learning techniques, particularly neural networks, the focus has shifted toward models that can learn from large datasets and make decisions based on experience. Modern deep learning systems have demonstrated remarkable capabilities, but they are computationally intensive and require large amounts of data for training. While these deep learning systems offer high performance, there is still potential in exploring simpler neural networks that can perform adequately in controlled environments.

This research investigates the application of a basic feedforward neural network for controlling a self-driving car in a simulation. The goal is to show that a relatively simple network can be trained to navigate predefined road networks and avoid obstacles with minimal computational resources. By simplifying the problem and environment, the study will also highlight how basic neural networks can contribute to the development of autonomous systems without the need for extensive computational power.

Literature Review:

The field of self-driving cars has experienced remarkable progress over the past decade, largely driven by advancements in deep learning techniques for perception and decision-making. Autonomous vehicles (AVs) are equipped with a range of sensors, including cameras, radar, and LIDAR, which collectively provide comprehensive data about the surrounding environment. This sensor data is processed through sophisticated algorithms that interpret road features, detect

obstacles, and predict the behavior of other vehicles and pedestrians. Several methodologies have been proposed for developing autonomous driving systems, ranging from traditional computer vision-based techniques to modern machine learning approaches.

Early autonomous systems primarily relied on rule-based or expert systems, where predefined sets of rules guided vehicle behavior. These systems combined sensor inputs with software algorithms to recognize objects and navigate roads. Although effective in structured and predictable environments, rule-based systems faced significant challenges in managing the complexities of real-world driving scenarios, such as dynamic traffic and unexpected road conditions. The limitations of these approaches led to the exploration of more adaptable and robust solutions.

With the rise of deep learning, particularly convolutional neural networks (CNNs), the capacity of self-driving systems to process and understand visual data has significantly improved. CNNs have become instrumental in tasks like object detection, lane recognition, and obstacle avoidance. However, deep learning models require extensive datasets for training and considerable computational power, which poses limitations for real-time performance in resource-constrained environments.

Another promising approach is reinforcement learning (RL), where an autonomous agent learns optimal actions through feedback from its environment in the form of rewards or penalties. RL has been successfully applied in autonomous driving research to teach cars how to make decisions by trial and error. Despite its flexibility and ability to train without labeled data, reinforcement learning demands substantial computational resources and large-scale simulations, making it challenging to implement in practical, real-world settings.

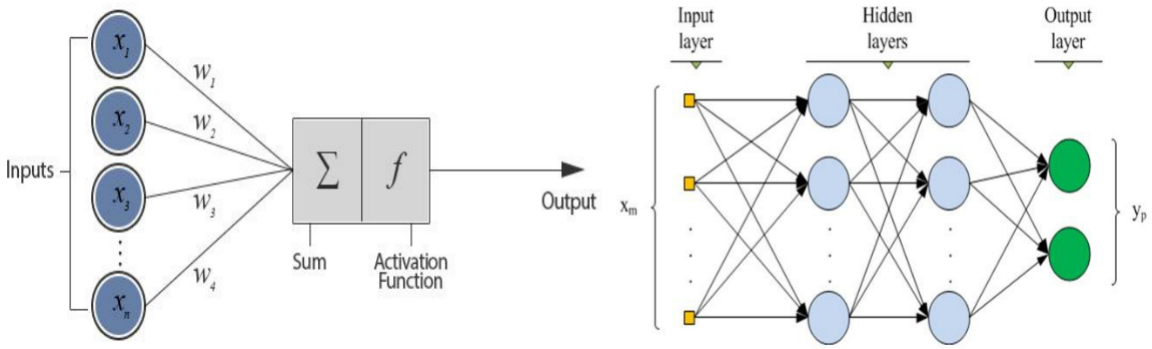
In contrast to deep learning, simpler neural network architectures, such as basic feedforward networks, have received less attention in autonomous vehicle research. These networks, which consist of an input layer, one or more hidden layers, and an output layer, have shown effectiveness in various robotic applications, including navigation and obstacle avoidance. While their simplicity limits their complexity and scalability, they offer the advantage of reduced computational requirements, making them suitable for less demanding, controlled environments.

This research aims to bridge the gap between the sophisticated methods of deep learning and the simplicity of basic neural networks. By demonstrating that a simple neural network can be effectively applied to self-driving car simulations in controlled settings, it highlights the potential of lightweight models to perform essential decision-making tasks. Such an approach offers an alternative perspective for developing efficient autonomous systems, particularly where computational resources are limited.

What is Artificial Neural Network?

An Artificial Neuron Network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes or learns, in a sense - based on that input and output. Artificial Neural Networks are moderately rough electronic models in light of the neural structure of the brain. The mind essentially gains for a fact. It is characteristic confirmation that a few issues that are past the extent of current computers are without a doubt reasonable by little strength effective bundles. This brain displaying additionally guarantees a less specialized approach to create machine arrangements. This new way to deal with processing additionally gives a more elegant corruption amid framework over-burden than its more conventional partners. These organically roused techniques for processing are thought to be the following significant progression in the registering business. Indeed, even basic creature brains are fit for capacities that are presently unimaginable for Computers. Computers do repetition things well, such as keeping records or performing complex math. Yet, Computers experience difficulty perceiving even basic examples considerably less summing up those examples of the past into activities without bounds. Presently, progresses in organic research guarantee an underlying comprehension of the regular intuition instrument. This examination demonstrates that brains store data as examples. Some of these examples are exceptionally entangled and permit us the capacity to perceive singular appearances from a wide range of edges. This procedure of putting away data as examples, using those examples, and after those tackling issues envelops another field in processing. This field, as said some time recently, does not use conventional programming but rather includes the production of enormously parallel systems and the preparation of those systems to take care of particular issues. This field additionally uses words altogether different from customary figuring, words like carry on, respond, self-sort out, learn, sum up, and overlook. At whatever point we discuss a neural system, we ought to all the more prevalently say —Artificial Neural Network (ANN), and ANN are Computers whose engineering is designed according to the brain. They normally comprise of many basic preparing units which are wired together in an unpredictable

correspondence arrange. Every unit or hub is astreamlined model of genuine neuron which sends off another flag or flames on the off chance that it gets an adequately solid



Traditionally neural network was used to refer as network or circuit of biological neurons, but modern usage of the term often refers to ANN. ANN is mathematical model or computational model, an information processing paradigm i.e. inspired by the way biological nervous system, such as brain information system. ANN is made up of interconnecting artificial neurons which are programmed like to mimic the properties of m biological neurons. These neurons working in unison to solve specific problems. ANN is configured for solving artificial intelligence problems without creating a model of real biological system. ANN is used for speech recognition, image analysis, adaptive control etc. These applications are done through a learning process, like learning in biological system, which involves the adjustment between neurons through synaptic connection. Same happen in the ANN.

Working of Artificial Neural Network :

The other parts of the —art of using neural networks revolve around the myriad of ways these individual neurons can be clustered together. This clustering occurs in the human mind in such a way that information can be processed in a dynamic, interactive, and self-organizing way. Biologically, neural networks are constructed in a three-

dimensional world from microscopic components. These neurons seem capable of nearly unrestricted interconnections. That is not true of any proposed, or existing, man-made network. Integrated circuits, using current technology, are two-dimensional devices with a limited number of layers for interconnection. This physical reality restrains the types,

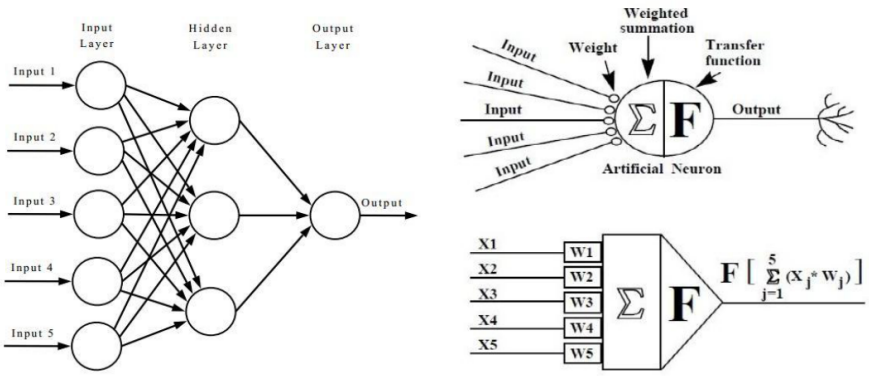


Figure 9:- A Simple Neural Network Diagram &Functions of an Artificial Neuron

Essentially, all simulated neural systems have a comparative structure or topology. In that structure a portion of the neurons interfaces to this present reality to get its sources of info. Different neurons give these present reality the system's yields. This yield may be the specific character that the system believes that it has checked or the specific picture it supposes is being seen. All whatever remains of the neurons are escaped see. Be that as it may, a neural system is more than a bundle of neurons. Some early analysts attempted to just associate neurons in an arbitrary way, without much achievement. Presently, it is realized that even the brains of snails are organized gadgets. One of the least demanding approaches to outline a structure is to make layers of components. It is the gathering of these neurons into layers, the associations between these layers, and the summation and exchange works that involves a working neural system. The general terms used to portray these attributes are normal to all systems. Despite the fact that there are helpful systems which contain just a single layer, or even one component, most applications require systems that contain in any event the three typical sorts of layers - input, covered up, and yield. The layers of info neurons get the information either from information records or straightforwardly from electronic sensors progressively

applications. The yield layer sends data specifically to the outside world, to an auxiliary computer handle, or to different gadgets, for example, a mechanical control framework. Between these two layers can be many concealed layers. These inward layers contain a considerable lot of the neurons in different interconnected structures. The sources of info and yields of each of these shrouded neurons just go to different neurons. In many systems every neuron in a concealed layer gets the signs from the majority of the neurons in a layer above it, normally an information layer. After a neuron plays out its capacity it passes its yield to the greater part of the neurons in the layer underneath it, giving an encourage forward way to the yield. (Note: in segment 5 the drawings are turned around, data sources come into the base and yields turn out the top.)

These lines of correspondence starting with one neuron then onto the next are imperative parts of neural systems. They are the paste to the framework. They are the associations which give a variable quality to information. There are two sorts of these associations. One causes the summing instrument of the following neuron to include while alternate causes it to subtract. In more human terms one energizes while alternate hinders. A few systems need a neuron to hinder alternate neurons in a similar layer. This is called parallel restraint. The most widely recognized utilization of this is in the yield layer. For instance in content acknowledgment if the likelihood of a character being a "P" is .85 and the likelihood of the character being a "F" is .65, the system needs to pick the most noteworthy likelihood and repress all the others. It can do that with parallel hindrance. This idea is likewise called rivalry. Another sort of association is input. This is the place the yield of one layer courses back to a past layer. A case of this is appeared in Figure 10.

Methodology & Experimentations:

Simulation Environment:

For this research, a simulation environment was developed using JavaScript, HTML5, and the HTML Canvas API. This virtual environment replicates a simple road network where the self-driving car

must navigate while avoiding collisions with road boundaries and other vehicles. The road is represented as a series of connected lines or polygons, while traffic is simulated by moving vehicles represented as polygons. The environment includes dynamic road features such as curves, intersections, and varying traffic density.

Neural Network Model:

The neural network used in this study is a simple feedforward network with an input layer, a hidden layer, and an output layer. The input layer consists of sensor data that is generated by rays cast from the car's current position. These rays detect the distance between the car and various road features or traffic obstacles. The hidden layer consists of a small number of neurons that process the sensory input, while the output layer generates control commands for the car's movement (steering, acceleration, and braking).

The architecture of the network is as follows:

- **Input Layer:** Contains sensory input data, such as the distances to the left and right road boundaries, as well as the distance to any traffic or obstacles in the path.
- **Hidden Layers:** A single hidden layer with a small number of neurons processes the sensory input and determines the appropriate action for the vehicle.
- **Output Layer:** The output layer produces control signals that govern the car's movement, such as turning left or right, accelerating, or applying the brakes.

Training Process:

The neural network was trained using supervised learning, where the car's movements were manually labeled as correct or incorrect based on the environment. A series of sample scenarios were created in the simulation, and the network was trained to navigate these scenarios by adjusting the weights and biases of the network. The training data consisted of sensor readings and the corresponding desired vehicle actions.

The network was trained iteratively, using an error-correction method to adjust the weights of the network during each training session. A loss function, such as mean squared error, was used to calculate the difference between the predicted actions and the actual actions, allowing the network to update its weights accordingly.

Evaluation:

The performance of the trained neural network was evaluated by testing it in various driving scenarios. These scenarios included different road layouts, varying traffic conditions, and road curves. The success criteria were based on the network's ability to avoid collisions, stay within the road boundaries, and successfully navigate through intersections. The system was tested for robustness and generalization by using it in new, unseen scenarios to determine how well it performed in environments beyond the training set.

4.Results & Discussions:

Results:

The neural network was able to successfully navigate a variety of road networks, staying within the boundaries and avoiding collisions with obstacles. In scenarios with minimal traffic, the vehicle could maintain a straight path without any difficulty. In more complex scenarios, where traffic density was higher, the vehicle adjusted its path to avoid collisions and navigate through tight spaces. The training process was effective, and the network showed reasonable generalization capabilities in environments not seen during training.

One of the key findings was that the simple neural network performed reasonably well under controlled conditions, even though it was much less complex than state-of-the-art deep learning models. While the network was not capable of handling extreme traffic situations or making complex decisions (such as predicting the movement of other vehicles), it was effective for basic navigation tasks ed iteratively, using an error-correction method to adjust the weights of the network during each training session. A loss function, such as mean squared error, was used to calculate the difference between the predicted actions and the actual actions, allowing the network to update its weights accordingly.

Discussions:

Although the neural network performed well in the simulation, several limitations were identified. The primary limitation was the network's inability to handle highly dynamic environments. For example, in scenarios with high-speed traffic or pedestrians, the network was unable to predict future movements or respond to sudden changes in the environment. This is a limitation of simple neural networks, which are not designed to handle temporal dependencies or complex decision-making tasks.

Additionally, the performance of the neural network was highly dependent on the quality of the sensor data. In real-world applications, the sensors of an autonomous vehicle may be subject to noise or inaccuracies, which could affect the performance of the network. More advanced methods, such as recurrent neural networks (RNNs) or reinforcement learning, may be required to handle more complex tasks such as real-time decision-making in unpredictable environments.

Despite these limitations, this research demonstrates that a simple neural network can be effective for basic self-driving tasks in controlled environments. This approach may be suitable for applications in specific use cases where the environment is relatively simple and predictable, such as autonomous vehicles operating in structured areas like campuses, warehouses, or industrial sites.

5. Conclusion:

In conclusion, this paper has successfully demonstrated the feasibility and effectiveness of utilizing a simple feedforward neural network to simulate the behavior of a self-driving car within a controlled and structured environment. The designed network efficiently processed sensor inputs, translating them into control outputs that enabled the car to navigate predefined roads, follow paths, and avoid various obstacles. This outcome highlights the potential of lightweight neural network models for real-time decision-making and control in autonomous systems, particularly when computational resources are limited or when low-latency responses are critical.

One of the key advantages observed in this approach is its relative simplicity, which allows for rapid training, lower computational overhead, and ease of implementation. The straightforward architecture of the feedforward network makes it an attractive choice for scenarios where advanced and resource-intensive models, such as convolutional neural networks or deep reinforcement learning frameworks, may not be practical or necessary. This demonstrates that even basic machine learning techniques, when appropriately applied, can deliver valuable results in the realm of autonomous navigation.

Despite its success, this approach has limitations that are important to address. The simulation environment used for training and testing was highly controlled, with simplified road structures, static obstacles, and predictable scenarios. As a result, the generalizability of the feedforward network's performance in real-world driving conditions remains constrained. Real-world environments present significantly more complexity, including dynamic obstacles, varying road conditions, traffic rules, and unpredictable human behavior, all of which pose challenges that exceed the capabilities of the current model.

Given these limitations, future work can build upon the foundational concepts demonstrated here by incorporating more advanced machine learning techniques. Reinforcement learning, for example, could enable the system to learn optimal driving policies through trial-and-error interactions with its environment, thereby improving its adaptability and robustness in more complex and dynamic settings. Additionally, the integration of convolutional neural networks could enhance the model’s ability to process visual data for tasks such as lane detection and object recognition. Combining these advanced methods with a modular, hybrid approach could significantly improve the performance and safety of autonomous systems.

In summary, this project serves as a proof-of-concept that a simple feedforward neural network can effectively simulate basic self-driving car behavior. It highlights the potential of lightweight models for real-time applications while recognizing the need for further advancements to handle the complexities of autonomous navigation in real-world scenarios. Continued research and innovation in this domain have the potential to revolutionize transportation, making it safer, more efficient, and increasingly autonomous.

6. References:

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zhang, X. (2016). *End to End Learning for Self-Driving Cars*. arXiv preprint arXiv:1604.07316.

Pomerleau, D. A. (1989). *ALVINN: An Autonomous Land Vehicle in a Neural Network*. Advances in Neural Information Processing Systems, 1, 305–313.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep Learning*. Nature, 521(7553), 436–444.

Schmidhuber, J. (2015). *Deep Learning in Neural Networks: An Overview*. Neural Networks, 61, 85–117.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*. arXiv preprint arXiv:1312.5602.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2021). *Deep Reinforcement Learning for Autonomous Driving: A Survey*. IEEE Transactions on Intelligent Transportation Systems, 22(6), 3416–3434.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

Ng, A. Y., & Russell, S. (2000). *Algorithms for Inverse Reinforcement Learning*. Proceedings of the Seventeenth International Conference on Machine Learning, 663–670.

Geiger, A., Lenz, P., & Urtasun, R. (2012). *Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3354–3361.