Autonomous Vehicle Simulation in GTA-5

Project report submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Tushar Jain - 20UCS211

Under Guidance of
Mr. Vikas Bajpai

Department of Computer Science and Engineering
The LNM Institute of Information Technology, Jaipur

April 2023

The LNM Institute of Information Technology
Jaipur, India

CERTIFICATE

This is to certify that the project entitled Autonomous Vehicle Simulation in GTA-5, submitted by Tushar Jain (20UCS211) in partial fulfillment of the requirement of degree in Bachelor of Technology (B. Tech), is a bonafide record of work carried out by them at the Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur, (Rajasthan) India, during the academic session 2022-2023 under my supervision and guidance and the same has not been submitted elsewhere for award of any other degree. In my/our opinion, this report is of standard required for the award of the degree of Bachelor of Technology (B. Tech).

Date

Adviser: Mr. Vikas Bajpai

Acknowledgments

We would like to express our gratitude to our instructor Mr. Vikas Bajpai, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system. Without his constant support, motivation and guidance this project would not have been successful.

Abstract

Autonomous Vehicle is a highly appealing sphere exciting all of the mankind. Several factors governs the decision making for the vehicles making legacy artificial intelligence networks inefficient. With the ever changing environment for the vehicles to drive on, automobile industry required an control system able to adapt with the environment.
Advancement in Deep Learning Algorithms helped majorly in fulfilling the requirement. With the self-adapting capability of the deep learning algorithms or neural networks, engineers are trying to make a fully-autonomous car.
With the advancement of time several neural networks have be developed which made creation of autonomous car very much closer from dream to reality.
In this project, the objective is to understand the Transformer Network, a neural network said to currently the best architecture and understands its applicability in automobile industry. To achieve this objective, an autonomous vehicle will be simulated with its control system built on top of a transformer network.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Area of Work

Autonomous Vehicle are cars and trucks that can understand and navigate the environment with little to no help from a human driver. While consumers cannot yet buy a car that fully drives itself (Level 4 or 5 automation)[1], the technology is under development and improving quickly. The self-driving cars being worked on today have many different sensors and cameras to capture information about the world. So, a big challenge in the field of Computer Science and Engineering is to make a Artificial Intelligent network which takes the captured information and gives the best possible action a car should in that state of environment. My project aims to try to build a network capable of taking some of the actions and to able to drive a car efficiently.

1.2 Problem Addressed

An autonomous vehicle is made with the goal to make travel of an individual safer than non autonoumus ones. The environment being partially observable and undeterministic, an autonomous vehicle should be prepared to take action based on any outcome. So an artificial intelligent model which can take decision in any type of environment is desired. Thus we need a model that can learn and adapt on its own in any type of scenarios. This is why a neural network based model is ideal for the use case and forms the heart of any autonomous vehicle. Since driving a vehicle in not efficient, we need a simulator which can create a close-to real world environment. Grand Theft Auto - V fits this criteria perfectly. Thus the main objective is to design, train, and simulate a autonomous vehicle in GTA-V and try to make the model as efficient as possible.

1.3 Existing System

Currently, there are no legally operational and fully autonomous vehicles in the world. However, partially autonomous vehicles, such as cars and trucks with varying amounts of automation, from assistance for braking to aid changing lanes and parking, with some models even having a certain degree of automatic steering are running on the roads. Although it is still in its infancy, autonomous driving technology is becoming increasingly common and could radically transform our transport system.
Currently there are several companies worldwide competing in the race of automation in aotomobiles like, [2][3][4][5]

- **Tesla** has been rolling out its autonomous cars with Level 2 automation for some past years and has also rolled out a beta-version of its software to upscale the level of autonomy in its vehicles.
- **Waymo** operates commercial self-driving taxi services in Phoenix, Arizona and San Francisco, CA. In October 2020, it was the only self-driving commercial service that operates without safety backup drivers in the vehicle at that time.
- **TuSimple** has developed a fleet of 50+ trucks and 18+ contracted customers in the last years, claiming that its tech can reduce transportation costs by 30%. It is also backed by Nvidia and UPS, which is working with TuSimple to test its Level 4 autonomous trucks.
- **Zoox** is creating an entirely new autonomous vehicle targeted at the robo-taxi market. Zoox has applied the latest techniques in automotive, robotics and renewable energy to build a symmetrical, bi-directional battery-electric vehicle that solves for the unique challenges of autonomous mobility.
- **Aurora** recently announced it is working to apply its Aurora Driver platform (Level 4) to heavy-duty Class 8 trucks.

Chapter 2

Literature Review

2.1 Sensors in Autonomous Vehicles

With the help of sensors, autonomous vehicles ensure no human interaction is needed while driving. A wide range of sensors are used in autonomous vehicles to build reliable vision. The sensors help the self-driving vehicle to detect hurdles or blockages in the driving environment and to move without causing fatalities.
Below are some primary sensors used in autonomous vehicles:

2.1.1 Camera

Cameras used in autonomous cars are specialized image sensors that detect the visible light spectrum reflected from objects. Cameras are the best sensor solution to give an accurate visual representation of an autonomous vehicle's surroundings. In autonomous vehicles, cameras are fixed on all four sides—front, rear, right, and left—to give a 360° view. These cameras use wide and narrow fields of view to perceive both short-range wide view and long-range arrow view. Super-wide lenses are used in autonomous vehicles for capturing a panoramic view that assists with parking. However, accurate camera visuals fail to give information regarding the distance of objects from autonomous vehicles.

2.1.2 LiDAR

LiDAR uses laser beams (light waves) to determine the distance between two objects. In autonomous vehicles, LiDAR is mounted on top of vehicles and is rotated at high speed while emitting laser beams. The laser beams reflect from the obstacles and travel back to the device. The time taken for this to happen is used to determine the distance, shape, and depth of the obstacles surrounding the autonomous vehicle.
Even though LiDAR can catch the position, shape, size, and depth of an obstacle, they can

get glitched by fake echoes showing far objects as near objects and vice versa. LiDAR fails to distinguish between multiple copies of laser signals and shows non-existent obstacles to autonomous vehicles. LiDAR does not function well in rain, snow, or fog.

2.1.3 RADAR

The principle of operation for LiDAR and RADAR are the same, but instead of the light waves used in LIDAR, RADAR relies on radio waves. The time taken by the radio waves to return from the obstacles to the device is used for calculating the distance, angle, and velocity of the obstacle in the surroundings of the autonomous vehicle.
RADAR in autonomous vehicles operates at the frequencies of 24, 74, 77, and 79 GHz, corresponding to short-range radars (SRR), medium-range radars (MRR), and long-range radars (LRR), respectively. They each have slightly different functions:

- SRR technology enables blind-spot monitoring, lane-keeping assistance, and parking assistance in autonomous vehicles.
- MRR sensors are used when obstacle detection is in the range of 100-150 meters with a beam angle varying between 30° to 160°.
- The automatic distance control and brake assistance are supported by LRR radar sensors.

RADAR technology in autonomous vehicles operates with millimeter waves and offers millimeter precision. The utilization of millimeter waves in autonomous vehicular RADAR ensures high resolution in obstacle detection and centimeter accuracy in position and movement determination. Compared to other sensor technologies in autonomous vehicles, RADAR works reliably under low visibility conditions such as cloudy weather, snow, rain, and fog.

2.1.4 Microphone

A microphone is a transducer that converts sound into an electrical signal. Microphones are used to give hearing abilities to autonomous vehicles. In autonomous vehicles, microphones can be used to detect horns, emergency sirens, etc.
Microphones in autonomous vehicles are also used to identify the condition of road, the vehicle is driving on. It is shown that different types of roads (like wet, sandy, snowy) have different spectrum hence plays an important role in the identification.

2.2 Navigation Systems in Autonomous Vehicles

Autonomous vehicles can use navigation systems to geolocate with numerical coordinates (e.g. latitude, longitude) representing their physical locations in space. They can also navigate by

combining real-time GPS coordinates with other digital map data (e.g. via Google Maps). Geolocation data often varies around a five-meter radius. To compensate for imprecise GPS data, self-driving cars can use unique data-processing techniques like particle filtering to improve location accuracy. Furthermore, these systems can also be used to make efficient judgement for vehicle to take turn.
Some geolocation services used nowadays includes Indian Regional Navigation Satellite System (*IRNSS* or *NavIC*), United States' Global Positioning System (*GPS*), Russia's Global Navigation Satellite System (*GLONASS*), China's *BeiDou* Navigation Satellite System and the European Union's *Galileo*.

2.3 Cruise Control in Vehicles

With the help of modern systems, cruise control became an important part in the modern-day vehicles. It is a technique to maintain the speed of the car without the need of the driver to push the throttle. Cruise Control helped in enhancing the vehicle's stability and fuel efficiency. The Proportional-Integral-Derivative Controller or PID Controller is a widely adopted approach for cruise control due to its simplicity and effectiveness.

2.3.1 PID Controller

A PID controller is a widely used feedback control system in engineering. It maintains a desired output by continuously adjusting an input based on three components:

- **Proportional**: Directly proportional to the error between the desired and actual values, providing an immediate response to deviations.
- **Integral**: Accumulates past errors over time, addressing prolonged deviations and eliminating steady-state errors.
- **Derivative**: Predicts future errors by considering the rate of change, dampening rapid changes and improving stability.

By combining these components, a PID controller ensures a balance between responsiveness and stability, making it versatile for applications such as process control, automation, and notably, in systems like cruise control for vehicles.
The figure below describes the graph of how PID works and is better than P alone or PI. Assume y-axis as speed of vehicle and x-axis as time.

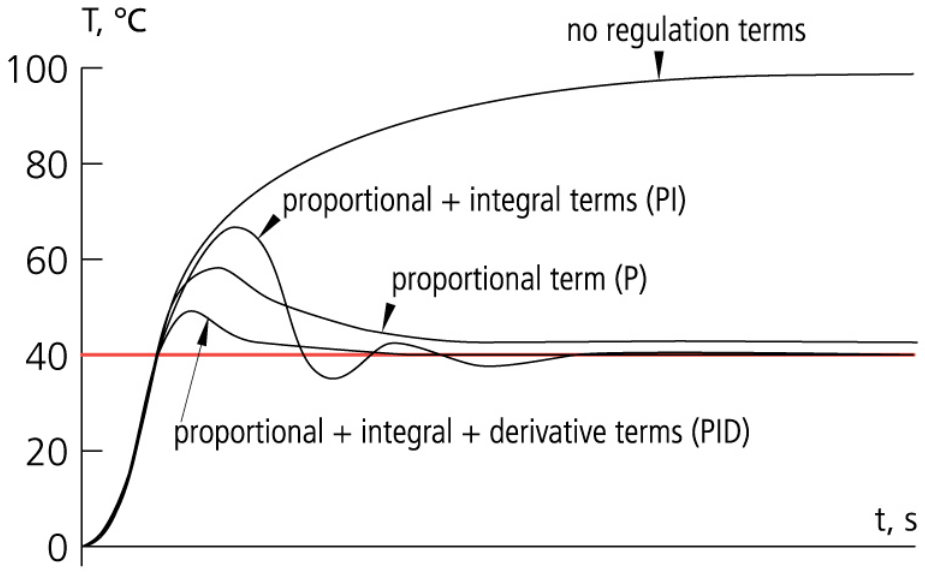

FIGURE 2.1: Difference between P, PI, and PID

2.3.2 Math in PID Controller

Initially,

$$PreviousError = 0$$

$$Integral = 0$$

$$TargetSpeed = SpeedAtWhichCarShouldMoveInCruiseControl$$

$$Delta = TimeBetweenEachIteration$$

$$K_p, K_i, K_d = Hyperparameters$$

In each iteration,

$$CurrentSpeed = SpeedAtWhichCarIsMoving$$

$$CurrentError = TargetSpeed - CurrentSpeed$$

$$P = K_p * CurrentError$$

$$Integral = Integral + CurrentError * Delta$$

$$I = K_i * Integral$$

$$RateOfChangeOfCurrentError = CurrentError - PreviousError$$

$$PreviousError = CurrentError$$

$$D = K_d * RateOfChangeOfCurrentError$$

$$ControlSignal = P + I + D$$

Control Signal is the value of throttle in each iteration, where negative means braking.

2.4 Vehicle Control with Deep Learning

Artificial Intelligence is the major backbone of the autonomous vehicles which enables vehicle to recognize and react to their environment in real time, allowing them to safely navigate. This is called **Perception**, the ability, while driving, to process and identify road data — from street signs to pedestrians to surrounding traffic.
In the earlier stages, Machine Learning was used to enable vehicles to perceive their environment. But as the technology increases, Deep Learning or use of Neural Networks turns out to be the better approach for the classification.

2.4.1 Machine Learning vs Deep Learning

2.4.1.1 Machine Learning

Machine Learning is an application of artificial intelligence that includes algorithms that parse data, learn from that data, and then apply what they've learned to make informed decisions.
In machine learning, human intervention is needed to extract the features based on input to make the accurate predictions. Furthermore, if an AI algorithm returns an inaccurate prediction, then an engineer has to step in and make adjustments.

2.4.1.2 Deep Learning

Deep Learning is subfield of machine learning that structures algorithms in layers to create an "artificial neural network" that can learn and make intelligent decisions on its own.
A deep learning model is designed to continually analyze data with a logical structure similar to how a human would draw conclusions. To complete this analysis, deep learning applications use a layered structure of algorithms called an **Artificial Neural Network**. The design of an artificial neural network is inspired by the biological network of neurons in the human brain, leading to a learning system that's far more capable than that of standard machine learning models

Figure 2.1 shows the distinction between algorithms based on Machine Learning and Deep Learning.

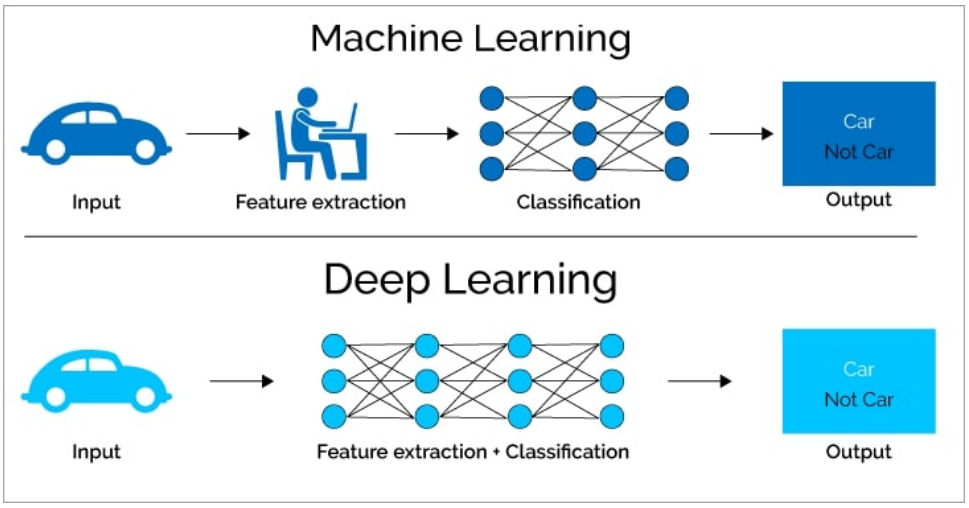

FIGURE 2.2: Difference Between Machine Learning and Deep Learning

2.4.1.3 Machine Learning or Deep Learning for Autonomous Vehicles

In case of autonomous vehicles, rather than requiring a manually written set of rules for the car to follow, such as "stop if you see red," Neural Networks enable vehicles to learn how to navigate the world on their own using sensor data. This makes Deep Learning algorithms optimal as the decision maker for the autonomous vehicle.

2.4.2 Deep Learning Algorithms

In the field of computer vision, over the period of time, several algorithms or neural networks are designed like Convulational Neural Networks, Generative Adversarial Networks, Transformers. With the advancement in technologies, advancement in accuracy of each neural network is done with transformers currently as the algorithms making best predictions.

2.4.2.1 Transformer Networks

Transformer networks have been proven to achieve better accuracy in a variety of autonomous vehicle perception tasks when compared to convolutional neural networks running on embedded systems.
First transformer network (say, native transformer)[6] has 3 major components:

1. Positional Embeddings
2. Transformer Encoder
3. Transformer Decoder

This network was formed to performe tasks in the field of Natural Language Processing. After further research, a new transformer named **Vision Transformer**[7] was built to perform tasks related to Computer Vision.
In Vision Transformer, the Encoder has remained same but the algorithms for Positional Embeddings and Decoder were changed. For autonomous vehicles, since computer vision plays an important role, the model designing is inspired using Vision Transformer

2.4.2.2 Positional Embedder in Vision Transformer

As shown in the figure 2.2, in positional embedding layer, the image is first split into fixed-size patches, then each patch linearly embedded, add finally position embeddings are done, and the resulting sequence of vectors are fed to a standard Transformer encoder.

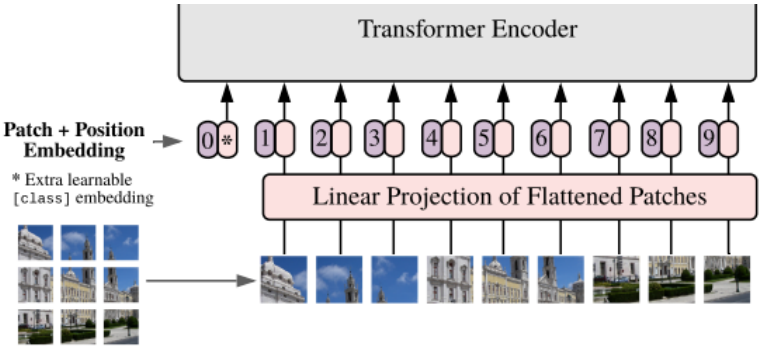

FIGURE 2.3: Positional Embedding Layer used in Vision Transformer

2.4.2.3 Encoder in Vision Transformer

The encoder is composed of a stack of N identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. A residual connection is employed around each of the two sub-layers, followed by Layer Normalization[8].
Figure 2.3 shows the encoder used in Vision Transformer.

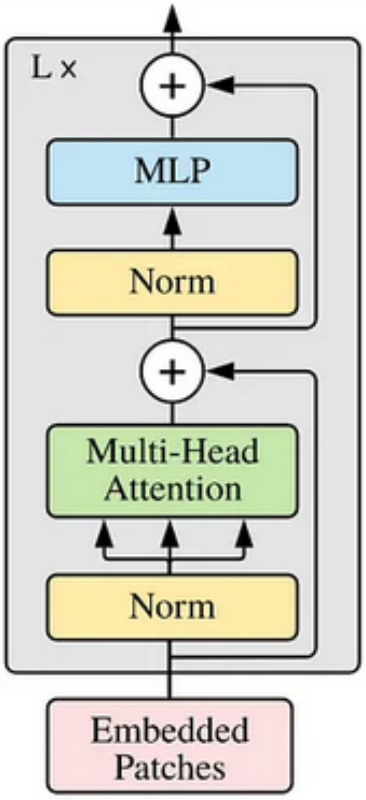

FIGURE 2.4: Encoder used in Vision Transformer

2.4.2.4 Multi-Head Attention in Transformers

Multi-head Attention is a module for attention mechanisms which runs through an attention mechanism several times in parallel. The independent attention outputs are then concatenated and linearly transformed into the expected dimension. Intuitively, multiple attention heads allows for attending to parts of the sequence differently (e.g. longer-term dependencies versus shorter-term dependencies).

$$MultiHead(Q, K, V) = [head_1, ..., head_h]W_o$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

In the above eqaution, W is the learning prameter and Attention is a mathematical function from Scaled Dot-Product Attention which is a single attention layer or a head in multi-head attention layer.
Scaled dot-product attention is an attention mechanism where the dot products are scaled down by $\sqrt{d_k}$. Formally we have a query Q, a key K and a value V and calculate the attention as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

With the help of Multi-Head Attention in Encoder, each position in the encoder can attend to all positions in the previous layer of the encoder. This played a mojor role in making Transformers more efficient and thus formed a backbone for the Transformers.

2.4.2.5 Multi-Layer Perceptron in Transformers

Multi layer perceptron (MLP) is a supplement of feed forward neural network. It consists of three types of layers—the input layer, output layer and hidden layer. The input layer receives the input signal to be processed. The required task such as prediction and classification is performed by the output layer. An arbitrary number of hidden layers that are placed in between the input and output layer are the true computational engine of the MLP. Similar to a feed forward network in a MLP the data flows in the forward direction from input to output layer.
The Multi-layer perceptron used in vision transformer contains GELU non-linearity[9]. Since the encoder block repeats, we have to be careful about the number of units in the hidden layer because the output dimension has to be compatible with the input for the next MultiHeadAttention layer.

2.4.2.6 Decoder in Vision Transformer

Decoder as per the paper is a normal classifier in which the tensor passed from the encoder are passed first through a Multi-Layer Perceptron and then with the help of softmax activation, classification was done.
Figure 2.4 describes the classifier used as decoder in the transformer.

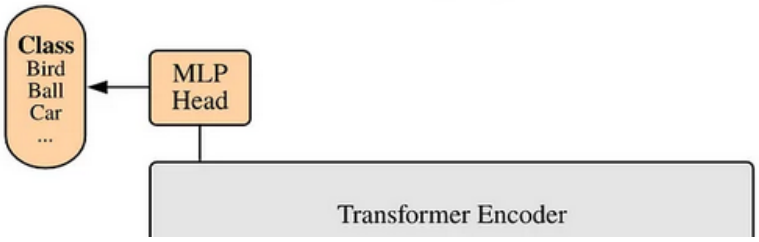

FIGURE 2.5: Decoder used in Vision Transformer

Chapter 3

Proposed Work

3.1 Workflow

I designed the following workflow for the project:

3.1.1 Formulation of problem statement

The problem statement as I described will be to form a autonomous vehicle. This will include creation of dataset, neural network training, and testing that neural network in the simulator (GTA-V).

3.1.2 Creating dataset

Due to the shortage to dataset, dataset needs to be created while manually controlling the vehicle in the simulator.

3.1.3 Data Preprocessing

Some data preprocessing is needed to be convert the dataset into neural network training ready data.

3.1.4 Designing Neural Network

A neural network needs to be designed which will be inspired from the Transformer Neural Networks.

3.1.5 Training Neural Network

Using the created dataset and designed model, the neural network needs to be trined and tested from the dataset.

3.1.6 Simulation in GTA-V

Once the neural network with training reaches efficient accuracy, with the help of the neural network, the vehicle needs to be driven autonomously in the simulator.

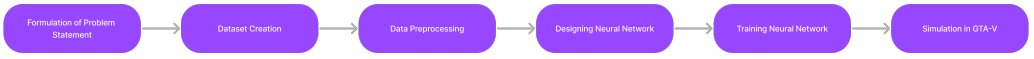

FIGURE 3.1: Project Workflow

3.2 Dataset

Since there is shortage of dataset, I have created a program to generate a dataset while a user plays the game. The data recorded consists of an image and 4 required labels. // A total of 10016 images and labels are recorded in the code.

3.2.1 Image

Image of resolution 1920 X 1080 is recorded while manually controlling vehicle in the simulator. Figure 3.1 shows a image recorded image.

FIGURE 3.2: Recorded Image

3.2.2 Labels

The dataset comprises of four labels:

- ***Throttle Value*** is the value of the acceleration the car is in. Throttle value 1.0 means forward acceleration, 0.0 means backward acceleration and 0.5 means no acceleration.
- ***Throttle Flag*** is a boolean value which states whether there is an acceleration.
- ***Steering Value*** is the value of the steering of the car. Steering value 1.0 means left steering, 0.0 means right steering and 0.5 means no steering.
- ***Steering Flag*** is a boolean value which states whether there is an steering.

3.3 Data Preprocessing

3.3.1 Generating Navigation Map

The Navigation System inbuilt in the game is extracted from the recorded image for coordinates (28,873) to (300,1045). Then since only the purple region is required to give the path, it is colored white ann the rest part is colored black. Then the image is resized to 128 X 128.

3.3.2 Generating Speed Map

Using a modification in GTA-V, it is possible to have a digital speedometer which is just above the Navigation Map. From there, the speed is cropped from (125,843) to (173,869). Then the image is made binary. Then the extracted image is resized to 128 X 128.

3.3.3 Resizing Image

Since the map and speed is extracted from the original image, the region in image containing map and speed is masked to not contail it in the image. The region masked is from (20,839) to (300,1060). Then the image is resized to 256 X 256 to make the image computationally feasible for Neaural Network training.
Figure 3.2 shows the processed images, map, speed.

FIGURE 3.3: Processed Images

3.3.4 Data Analysis

Since the data has been generated recently, it has to be analysed to confirm that all attributes are in balanced. The dataset comprises of three throttle and steering values combining them forms 9 different types of labels. The following are the unique labels:

- 0.0_0.0
- 0.0_0.5
- 0.0_1.0
- 0.5_0.0
- 0.5_0.5
- 0.5_1.0
- 1.0_0.0
- 1.0_0.5
- 1.0_1.0

The total number of data of each attributed has to be calculated to validate that the dataset is balanced. Figure 3.4 shows that the data is unbalanced.

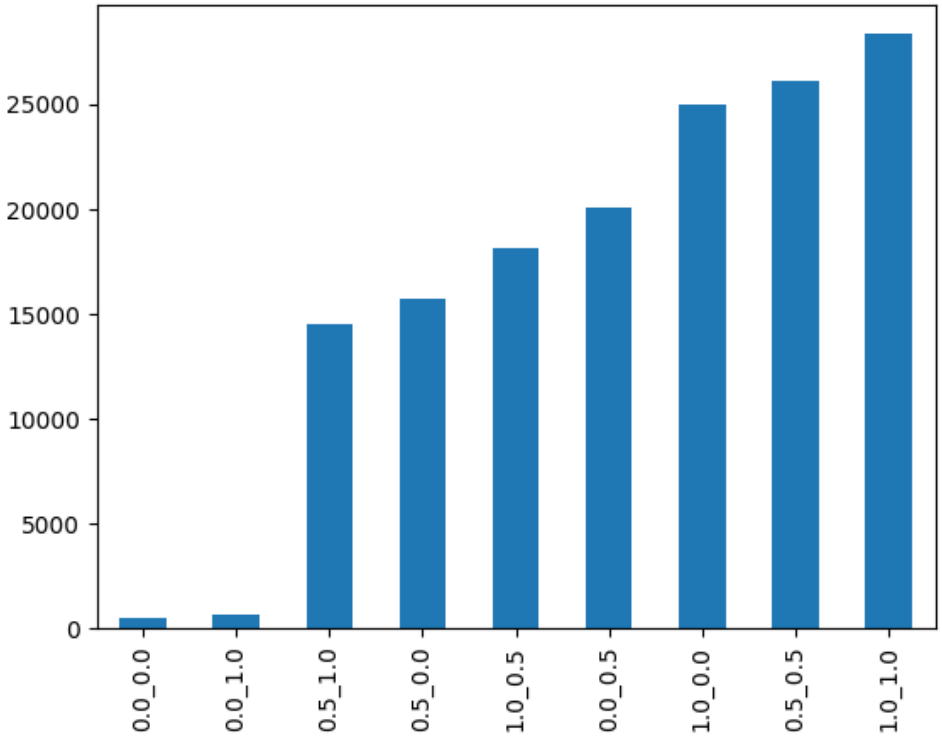

FIGURE 3.4: Bar Graph of the Dataset Generated

3.3.5 Data Augmentation

Since the data was found unbalanced, the following augmentation techniques are applied to increase the data of each labels:

- Increasing/Decreasing Brightness
- Increasing/Decreasing Contrast
- Horizontal Rotation of Image
- Horizontal Rotation of Radar
- Blurring
- Noising
- Bootstrapping

Figure 3.5 shows the data after applying these techniques.

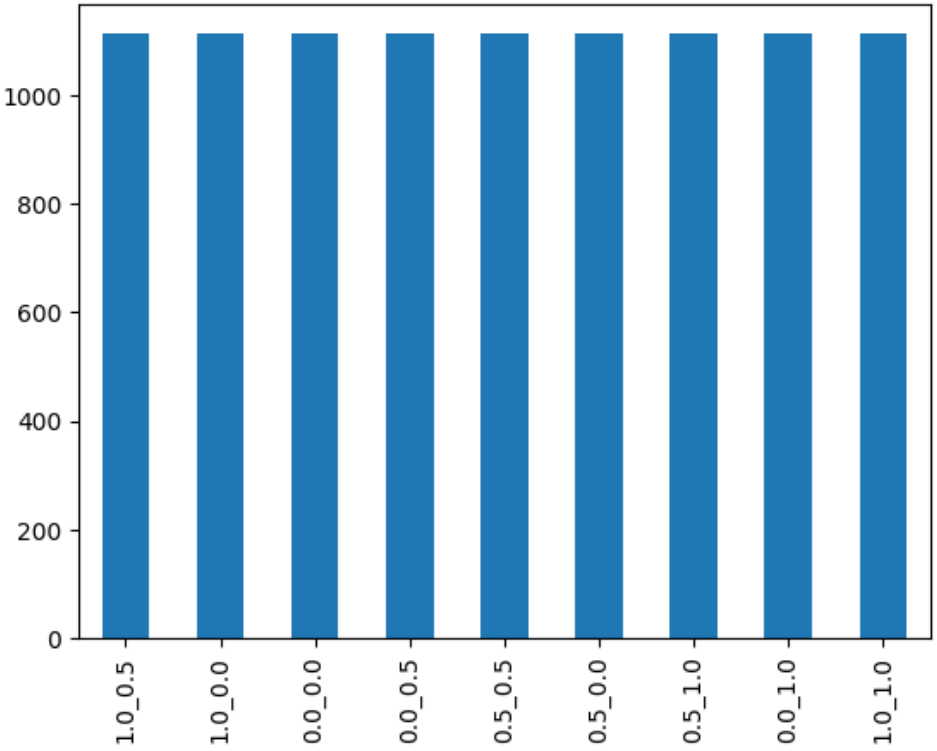

FIGURE 3.5: Bar Graph of the Augmented Dataset

3.3.6 Collecting generate images and labels

Then the processed images, radar and labels are pushed in an array to be saved as .npz file with images and radar saved as X and labels saved as y. A total of four such datasets are formed for training and fine-tuning. Datasets comprised of 10K, 20K, 30K, and 40K data each. The shape of:

- X is [n, 2]
- y is [n, 4]
- First index of tranpose of X is (n, 256, 256, 3)
- Second index of tranpose of X is (n, 128, 128, 3)

where, n is the number of data in the dataset.

3.4 Model Designing

With the reference from Vision Transformer[7] and Native Transformer[6] and transformations as per the dataset, I have devised a unique transformer, named DRAN.
Below is the description of the DRAN.

3.4.1 Patch Embedding

As mentioned in the vision transformer, the image before being passed to the transformer encoder needs to be converted into patches and positional embedding needs to be added to the generated patches.
For this, with the help of some online blogs, I have devised an algorithm for Patch Embedding as shown in figure 3.3.

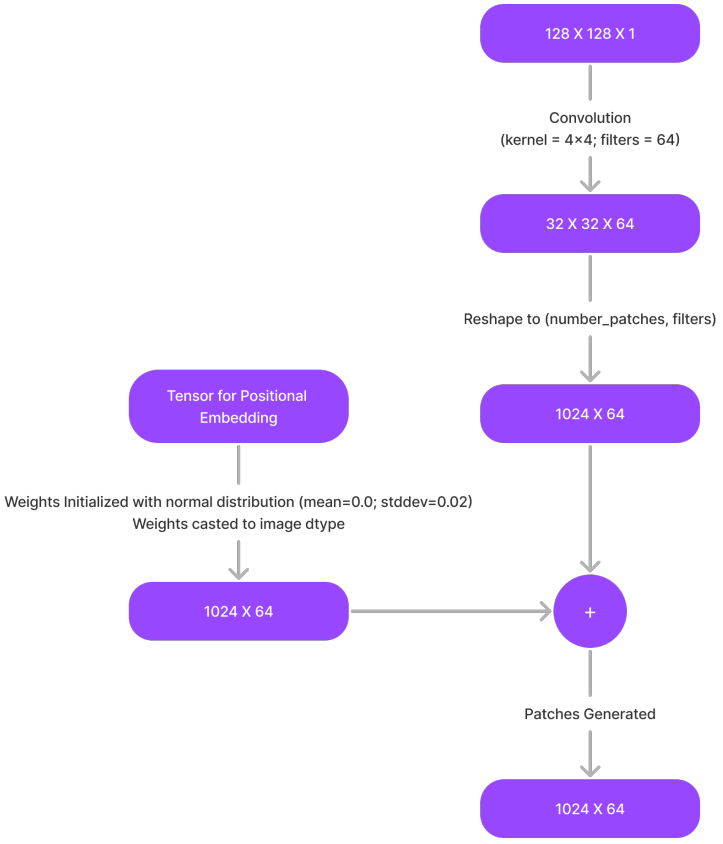

FIGURE 3.6: Patch Embedding Layer

3.4.2 Transformer Encoder

Encoder used in this transformer in a stack of 6 small identical units as explained in the original native transformer.

3.4.3 Model Decoder

There will be two transformer encoders in the final model so after some operations, data from two encoders will be combined and then further operations will be performed for generating output.
Figure 3.4, 3.5, 3.6 represents the structure of the decoder.

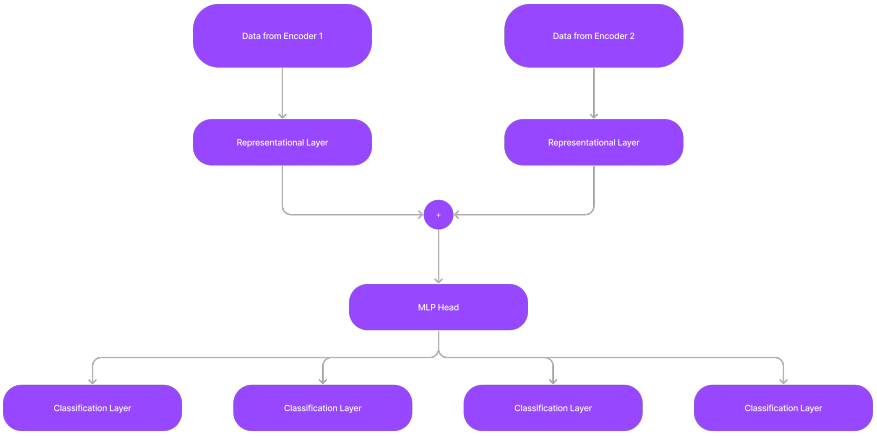

FIGURE 3.7: DRAN Decoder

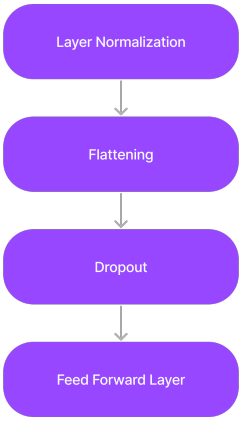

FIGURE 3.8: Representational Layer

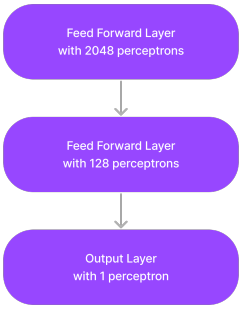

FIGURE 3.9: Classification Layer

3.4.4 Final Model

The main DRAN neural network is made with two input images one of the game camera recording, and other of the map processed in section 3.2.1.
There are 4 output labels of the neural network namely, Throttle Value, Throttle Flag, Steering Value, Steering Flag.
The final model designed is shown in figure 3.7 below.

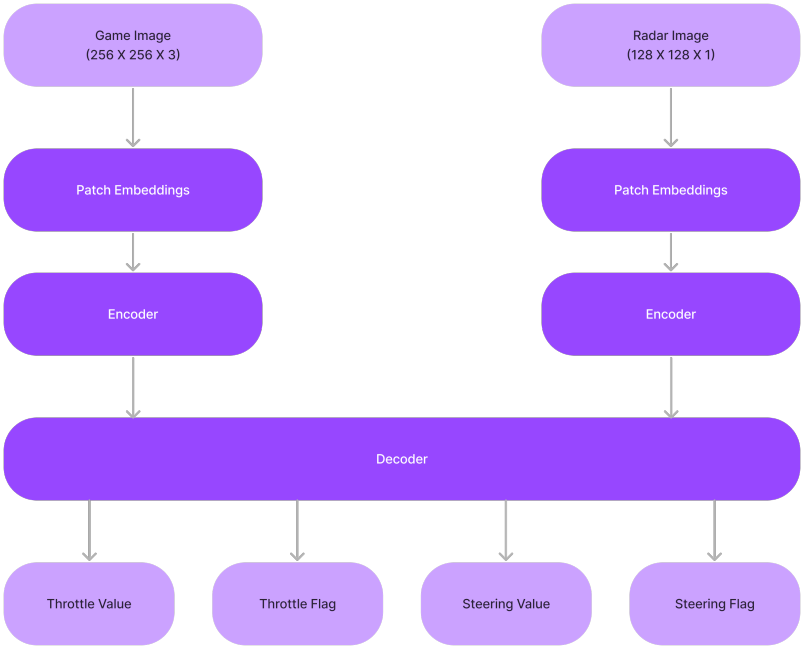

FIGURE 3.10: DRAN Neural Network

3.5 Model Preparation

Using the above model design, a model has to be built and compiled. The model will be built in TensorFlow 2 using Keras API.

3.5.1 Model Building

The model is build considering the following hyperparameters:

- Number of Encoder Layers = 6
- Number of Heads in Multi-Head Attention Layer = 4
- Number of Patches = 64

Using the following hyperparameters, the model was build and saves as dranNN.h5

3.5.2 Model Compiling

Before training, the model has to be compiled. The following specifications were used to compile the model:

- Losses : Binary Crossentropy
- Optimizers: Adam
- Metrics:
  - Accuracy
  - Mean Squared Error

3.5.3 Model Callbacks

To assure proper training and visualization of trained model, the following callbacks are used:

- Model Checkpoint
- Tensorboard

3.5.4 Dataset Configuration

To train the model for the first time, dataset with 10K data will be used and the datset will be splitted between train and validate in a ration of 4:1.

Chapter 4

Simulation and Results

In response to unforeseen circumstances preventing the training of the intended model, an innovative solution has been implemented through the development of a simulator for vehicle cruise control. This alternative system employs Tesseract OCR (Optical Character Recognition) to convert speed data into numerical values. Subsequently, a PID (Proportional-Integral-Derivative) controller is utilized to determine the appropriate output action based on the disparity between the desired and actual speed. This output action is then translated into a corresponding key press, aligning with the desired speed adjustments.
The seamless integration of Tesseract OCR and PID control not only ensures accurate speed interpretation but also facilitates a responsive and efficient control mechanism for the vehicle. Notably, the flexibility of the code allows for the straightforward incorporation of the originally intended model, preserving the adaptability of the system to future enhancements. This strategic approach not only addresses the challenges posed by unforeseen obstacles but also lays the groundwork for future iterations, maintaining the system's potential for continuous improvement and evolution.

4.1 Final Result

Presenting a comprehensive video demonstration, the efficacy of the cruise control feature in GTA-V is meticulously showcased. The video underscores the seamless integration of Tesseract OCR and PID control, illustrating the system's nuanced ability to dynamically interpret in-game speed data through Tesseract OCR and execute precise adjustments via the PID controller. Audiences are invited to observe the virtual vehicle responding adeptly to diverse in-game scenarios, maintaining the desired speed with precision. The provided hyperlink facilitates direct access to a visual representation of the project's success, offering a firsthand perspective on the sophisticated control mechanism within the dynamic virtual realm of GTA-V. This video presentation serves not only as a visual complement to the technical intricacies outlined in the report but also as an engaging narrative of the cruise

control system's proficiency within the gaming environment. For an immersive demonstration, please follow the link: https://drive.google.com/file/d/1qhb-lCffd7_i3RC6IqrqWnJGbAAelngn/view?usp=sharing.

4.2 Other Important Links

- Link for the GitHub Organization where all the code of this project is: https://github.com/DRAN-Drive-Autonomous.
- Drive Link for the Analytics part: https://drive.google.com/drive/folders/1hKAnlzI5grYIPix5mnLZJZc28c-z5qWl?usp=sharing.

Chapter 5

Conclusions and Future Work

In this project, I discussed about many criterions used to design a autonomous vehicles, compared machine learning and deep learning for the task, discussed about cruise control and the mathematics involved behind it. This project involves developement of several programs, be it for data collection, preprocessing, analysing, augmentation, collecting data in a array. This gave insight about how difficult and crucial data collection is in making a Artificially Intelligent Model. Along with this, We discussed about different layers and their functionalities in Vision Transformer and a hands-on implementation of Vision Transformer is also done, though it could not be trained due to certain unseen circumstances. A hands-on implementation in developing a simple cruise control network for a vehicle was also done which gave us insights about how important system efficiency is in making a AI Network, as there was a gap of 0.5 seconds in each iteration in my system which greatly affected the performance. Finally, simulation of cruise control feature in GTA-V was done which helped in understanding the difference between theoretical and practical changes in running AI Models as their were several hyperparameter tuning which had to be done to make the car work as shown in video.

5.1 Scope of Future Work

Work that could be done in future:

- Training the model or build another model to acheive desired accuracy.
- Make a network to easily visualize and also efficienty predict the vehicle action.
- Training time-series model and compare the models.

Bibliography

[1] O.-R. A. D. O. Committee, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, apr 2021.

[2] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," 2019.

[3] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," 2017.

[4] L. Yang, Y. Zheng, X. Cai, H. Dai, D. Mu, L. Guo, and T. Dai, "A lstm based model for personalized context-aware citation recommendation," *IEEE Access*, vol. 6, pp. 59618–59627, 2018.

[5] T.-A. Teban, R.-E. Precup, E.-C. Lunca, A. Albu, C.-A. Bojan-Dragos, and E. M. Petriu, "Recurrent neural network models for myoelectricbased control of a prosthetic hand," in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 603–608, 2018.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, June 2017.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[8] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[9] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2020.