National College of Ireland

Advancements in Steering Angle Prediction: Deep Learning Approaches for Self-Driving Car

MSc Research Project
MSc Artificial Intelligence

Rahul Goswami
Student ID: X23167572

School of Computing
National College of Ireland

Supervisor: Arundev Vamadevan

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Rahul Goswami
Student ID:	X23167572
Programme:	MSc Artificial Intelligence
Year:	2024
Module:	MSc Research Project
Supervisor:	Arundev Vamadevan
Submission Due Date:	12/12/2024
Project Title:	Advancements in Steering Angle Prediction: Deep Learning Approaches for Self-Driving Car
Word Count:	Approx 9000
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Advancements in Steering Angle Prediction: Deep Learning Approaches for Self-Driving Car

Rahul Goswami
X23167572

Abstract

Deep learning and computer vision have been driven forward, with time, as the demand in the self-driving automobile sector increases. Under this effort, three unique models will be built and tested for use in self-driving car applications. Training shall make use of the Udacity Self-Driving Car Simulator. Each of these shall be oriented toward making predictions of the steering angle, hence enabling a self-driving car to make its way through a course completely independently of any human intervention. Similarly, three models were built using this dataset of images from the center, left, and right camera with values for steering angle, throttle, brake and speed NVIDIA Model, J-Net Model, and Custom CNN Model.

Each model was developed under diverse architectural decisions with an eye on improving the accuracy of the steering predictions, taking into consideration various environmental conditions available in the simulator. With that extensive data preprocessing and augmentation were performed: cropping, resizing, changing brightness, and adding Gaussian blur; all this to make sure the dataset covers everything and is representative for most driving conditions. Later on, the performance of models trained based on a mean squared error metric and their generalization capability on previously unseen driving environments was measured. The primary aim of the study was to determine which among the three models-the NVIDIA Model, J-Net Model, or Custom CNN Model-provides the most consistent and accurate steering prediction to enable safe and reliable autonomous navigation.

Keywords: Autonomous Vehicles, Deep Learning, Steering Angle Prediction, NVIDIA Model, J-Net Model, Custom CNN Model, Udacity Simulator

1 Introduction

1.1 Background

Autonomous driving is basically the capability of vehicles to operate without any interference from the outside world, much like human cognitive capabilities using advanced technology like sensor systems, cameras, and machine learning algorithms. At the heart of it, deep learning helps the vehicle understand, make decisions, and control itself by finding detailed patterns within driving data Mozaffari et al. (2022).

Levels of automobile autonomy range from Level 0-all manual with no automation-to Level 5-fully automated. While Level 0 has full human control, Level 1 includes basic automation features, such as adaptive cruise control; Level 2 then adds features like automated steering and acceleration, but with continued human oversight. Conditional

automation in Level 3 lets drivers disengage but requires them to be ready to intervene. Level 4 involves high automation with minimal human input under certain conditions, and Level 5 represents full autonomy, requiring advanced sensor integration and machine learning to handle diverse driving scenarios.

This paper uses three different deep learning models: PilotNet, an improved Nvidia model, and a CNN-RNN Hybrid with LSTM layers that will provide the output for steering angle prediction. PilotNet is a model developed by Nvidia for the estimation of steering angles with convolutional layers on visual inputs. Data augmentation practices such as brightness adjustment, rotation, and zoom enhancements make this model more robust. Improved Nvidia model: Input pre-processing consists of YUV color transform and normalization; optimized over mean squared error loss on training.

Finally, the CNN-RNN hybrid utilizes LSTMs to capture temporal dependencies for the estimation of vehicle motion dynamics based on successive frames. All three models are trained on simulated data comprising images from front-facing cameras and corresponding steering angles. Their performance is evaluated using metrics such as MAE and RMSE to assess prediction accuracy. This thesis investigates these architectures, comparing their strengths and limitations in autonomous driving tasks Janai et al. (2020).

1.2 Problem Statement

With the rapid development of autonomous vehicles, a lot of different problems have arisen, especially with regard to the safety and security of the control over vehicle navigation. One of the most vital components in autonomous driving is the forecasting of steering angles, which affects directly the trajectory the vehicle will follow. While deep learning models have achieved much success in this field, accurate estimation of the steering angle in ever-changing environmental conditions remains a formidable challenge. Illumination, meteorological conditions, types of roadways, and other environmental variables will produce changes that tend to weaken and make such models even less reliable. Deep learning frameworks also need large training datasets necessary to capture the complex features of real driving scenarios.

The key problem explored in the given paper is related to enhancing the reliability and accuracy of the predictions of the steering angle in various driving scenarios to be used in autonomous vehicles. With the purpose of addressing this task, three deep learning architectures are suggested and implemented: a convolutional neural network-based PilotNet model, an enhanced Nvidia architecture, and a CNN-RNN combination. It aims to investigate which model has better performance in terms of steering angle prediction, under what driving environment, and what are the main factors improving model resilience under practical application.

The research question is: How can deep learning models be leveraged to improve the accuracy of steering angle predictions for autonomous vehicles? The sub-questions that will support this research are as follows:

• What are the comparative strengths and weaknesses of the PilotNet, enhanced Nvidia, and hybrid CNN-RNN models in terms of steering angle prediction accuracy?

• How does incorporating temporal dependencies using LSTM layers in the hybrid CNN-RNN model affect its steering angle prediction capabilities? The ultimate goal is to identify an effective approach to steering angle prediction, contributing to the development of safer and more reliable autonomous vehicle systems.

2 Related Work

Especially in the last decade, because of the innovations in the domain of artificial intelligence, mainly deep learning, there has been an unprecedented advancement of autonomous vehicles. The possible impacts of AVs-no fewer roadway accidents, improved flow of traffic, and better mobility-are life-changing. Their deployment, however, rests on surmounting some of the key technical hurdles, especially in fulfilling the requirements of precise estimations of steering angles, so vital for trajectory planning and vehicle control Badue et al. (2021).

Initial architectures for steering angle prediction were dominated by CNNs for feature extraction from images. These architectures were, in fact, good to process the front camera view images and capture critical roadway features, lane markings, and boundary information. While pioneering work like PilotNet by Bojarski et al. (2017) recognized the feasibility of end-to-end learning for steering tasks in autonomous driving, their shortfalls with respect to dynamic, unstructured environments gradually came out. PilotNet and similar models were sensitive in changing light conditions; thus, the models were not able to generalize very well on unseen situations.

Recent works have sought to improve these deficiencies by both architectural improvements and data-driven improvements. Nadella et al. (2024) explored the enhancement of CNN-based models with essential preprocessing steps such as transformation into the YUV color space, image normalization, and heavy augmentation of data. These preprocessing methodologies, when applied, increased the robustness of the models against poor environmental conditions such as glare and low visibility, increasing their generalization capability. In a related study, Rebuffi et al. (2021) have shown that enriching the training datasets through synthetic transformations-flipping, changing brightness, and adding Gaussian noise-significantly reduces overfitting and improves performance results.

Hybrid architectures have marked a large step in the prediction of steering angles. While CNNs do an excellent job in spatial feature extraction, their inability to understand temporal dependencies hinders complete understanding of motion dynamics at different instants in time. In order to bridge this gap, researchers have integrated RNNs, especially LSTM layers, into CNN frameworks. Bachute and Subhedar (2021) illustrated that a hybrid CNN-RNN model enhanced the accuracy of a steering prediction by including sequential information from every frame in the video. These modes give quite good performance on dynamic scenes: sharp turns or abrupt lane shifts require sharp temporal awareness jae Lee and Ha (2020).

It becomes clear that, in the last few years especially, preprocessing and augmentation bear vital importance. For the given dataset, Saleem et al. (2021) showed that using Gaussian blur with histogram equalization enhances model robustness through eliminating noise by amplifying the necessary features. These are found to be quite effective for the simulation of real-world conditions of driving, along with some domain-specific data augmentation, such as random shadow addition Taylor and Nitschke (2018). The ability to do well with relatively small datasets is an important limitation with most works in autonomous driving.

Running parallel to these, the methodologies of sensor fusion have also been amply employed.

While camera frames remain the most common input for steering angle prediction, researchers have explored augmenting this with LIDAR, radar, and GPS for enhanced contextual understanding by the model. Liu et al. (2018) pointed out that making use

of multimodal inputs increases robustness for autonomous vehicle systems, especially in complex urban environments.

These methods help the models fill in some gaps that are visually there, occlusions, bad weather, among other things, and enhance precision in decision-making.

Despite such progress, there is still some way to go. Aljehane (2024) pointed out that hybrid CNN-RNN deep learning models, which are computationally complex, are barely applicable in real-time at least in resource-constrained setups. Large and varied training datasets are also viewed as one of the bottlenecks of this approach, since a simulated environment cannot quite afford the variability that real driving scenarios provide. Generative adversarial networks have also been used for the synthesis of data, and their integration might allow researchers to include more realistic driving scenarios into this dataset Rožanec et al. (2022). Apart from developing better architectures, significant work has been done in optimizing using different techniques. Among these techniques are quantization and pruning to reduce the computational load, hence allowing real-time use on edge devices. According to Nadella et al. (2024) quantized versions of the CNN models retain most of the predictive accuracy while reducing the time and memory taken for the inference by a huge factor. These are important steps for field deployments of the AV system with restricted computation capacity. As the domain keeps developing, generalization issues have increasingly cast their focus on domain adaptation methods. Li et al. (2024) investigated the utilization of unsupervised domain adaptation, which aligns the features of simulated environments with those of real driving data. So far, this method has proved very promising and efficient in narrowing the gap between synthetic and natural datasets, making models trained on simulation act effectively in the real world. Dong et al. (2023)The architectures that are surveyed here on deep learning for steering angle prediction, in chronological order, show the overall trend in AV development: from early CNN-based models via hybrid CNN-RNN frameworks toward even further models. While current models achieve remarkable accuracy in a controlled environment, future work has to be directed toward increasing robustness, scalability, and computational efficiency Faizi and Alsulaifanie (2023). This is foreseen to be based on deeper studies related to multimodal data fusion, sophisticated data augmentation methods, and lightweight model architectures suitable for edge deployment. It is this kind of innovation that has been instrumental in unleashing the full value proposition of AVs: making transportation autonomous, safe, reliable, and efficient.

2.1 Summary of Literature Review

Starting with CNN-based architectures, such as Nvidia's PilotNet, the bases of their application were shown to be very effective in extracting spatial features but struggled in dynamic and unstructured environments. Accordingly, improvements have been made via advanced preprocessing and data augmentation, hybrid CNN-RNN with LSTMs, with the addition of better robustness and temporal learning. Recently, domain adaptation, sensor fusion, and lightweight network architectures have further boosted model performance and applicability in real time. While there has been considerable improvement, both aspects, computing efficiency and data variability, are yet open and active areas of research.

3 Methodology

3.1 Approach 1 - Vision-Based Steering Prediction Using PilotNet Architecture

In this work, the architecture of PilotNet is followed to estimate the steering angle of an autonomous car from visual data; the data is sourced from the Udacity simulation environment. In designing high computational efficiency, the model processes RGB images of size 66x200, captured from the front-facing camera of the car. These are passed through convolutional layers responsible for the extraction of salient features like lane markings and road edges, followed by ReLU for learning complex representations. Flattening these into vectors, the features extracted are further processed by fully connected layers, which include an output layer that predicts a continuous steering angle.

PilotNet is purely an end-to-end architecture; hence, it minimizes the mean squared error between the predicted and actual steering angles that can effectively reproduce human driving without explicit feature engineering. This makes it really simple and efficient for regulated environments such as the Udacity Simulator devoid of pedestrians, moving vehicles, or even traffic lights, meaning the model could focus solely on lane detection and smooth steering control.

3.2 Approach 2 – Steering Control - Enhanced Nvidia Model

The Enhanced Nvidia Model predicts the steering angles of self-driving cars from visual information gathered from the Udacity simulator. An extension from Nvidia's original basic end-to-end architecture incorporates extra convolutional layers to enhance its performance and ability to outline key road features that facilitate better trajectory control. This model processes RGB images downscaled to 66x200 pixels using convolutional layers where lane markings and road edges are identified by the filters. Each convolutional layer is followed by an ReLU activation function that helps in identifying complex patterns.

Following the extraction of features, the output is flattened and passed through the fully connected layers to make predictions of the steering angle. Utilizing mean squared error for the loss function, the model minimizes the difference between predicted and actual steering angles while it learns from the raw pixel data input directly, without explicit feature engineering.

The Enhanced Nvidia Model proves to be more effective in those lane-following scenarios which do not involve many intricate factors like vehicles or pedestrians. The Udacity simulator shows that by adding more convolutional layers to the base architecture, the model can gain further finesse in understanding road features for precise and reliable steering in controlled environments.

3.3 Approach 3 – Hybrid CNN-RNN Model-Steering Angle Prediction

It proposes a hybrid model that couples the strengths of a Convolutional Neural Network with the advantages of a Recurrent Neural Network using LSTM units for the exact steering angle prediction of an autonomous vehicle. Every step in the implementation of this model is explained, including the training and data collection using Udacity's simulator. The proposed architecture will hence take into account both the spatial and

temporal features of driving comprehensively for the prediction of steering Ngoc et al. (2023).

In addition, the CNN portion of the model receives input images from only one single front-facing camera and calculates its spatial features. In particular, the input images are cropped to 66 x 200 pixels and then pass through a stack of convolutional layers, each one followed by a ReLU activation that allows the model to learn nonlinear relationships in the data and understand the road marks and lane boundaries. After those convolutional layers, batch normalization and dropout provide better training stability and normalize the output from each layer so that it wouldn't go to overfit.

After the feature extraction, the output of CNN is flattened into a one-dimensional vector for feeding into the fully connected layer so that the spatial information will be represented right for further temporal analysis. The output from the fully-connected layer at this stage will be reshaped and used as the input for the LSTM layer. After that, through the LSTM, it captures the temporal relations of these frames, enabling the model to understand what temporal progressions can most likely happen in a visual context. This would be a way to enable the model to capture, far ahead on the road, the temporal information that will eventually allow it to make suitable decisions relating to direction; hence, it is useful in making predictions by understanding the sequence of frames Wu et al. (2019)

The proposed LSTM module will extract frame-to-frame interdependencies by skillfully controlling the flow, with the help of three types of gates, recognizing both short-term and long-term interdependencies.

This will consequently allow the model to predict future road characteristics and therefore make the steering prediction more reliable. Finally, the output from the LSTM layer is fed through a final fully connected layer that outputs the predicted steering angle. The hybrid CNN-RNN network is therefore trained in an end-to-end fashion using MSE to compare predicted steering angles against ground truth values of human driving. Therefore, the proposed architecture has both spatial and temporal pattern learning capabilities, having a CNN and an LSTM section hence capable of driving with informed decisions in a sequential environment such as the one from Udacity's Simulator. That proves to be a very effective combination not only for obtaining the static configuration of the roadway but also the temporal dynamics. The hybrid model thus has been quite apt for following lanes in the absence of any interaction with traffic, pedestrians, or other complicating factors.

4 Design Specification

Here, we discuss the design of the three models utilized in this research—PilotNet, the enhanced Nvidia architecture, and the hybrid CNN-RNN model. We also describe the methodology approach adopted for model training, evaluation, and deployment.

4.1 PilotNet Model

The PilotNet model is a CNN proposed by Nvidia, (Bojarski et al. 2016), which is designed for an end-to-end learning process in the task of autonomous driving. More precisely, the PilotNet model is supposed to predict the steering angle by an end-to-end learning process in autonomous driving. In this study, the PilotNet model was trained on

data from the Udacity simulator and it processes road images to infer the best steering response. It uses a series of convolutional layers to extract features such as lane markings and curvature of the road. Following these are activation functions such as ReLU that allow the model to learn complex nonlinear relationships inherent in the input. With the input traversing multiple convolutional layers, higher-level features capture more abstract aspects of the driving environment. Subsequent to convolutional layers, there comes a flattening layer that condenses the obtained feature maps to one-dimensional vectors, further to be processed over fully connected layers. The fully connected layers combine these spatial features and produce a single output; in this case, that would be the predicted steering angle. In its architecture, PilotNet is trained end-to-end to minimize the MSE present between predicted and actual steering angles so that it successfully emulates human driving patterns. As the model was trained on the Udacity simulator, it focuses mainly on learning lane following without considering the complications of traffic or obstacles, and thus is feasible in the simple, controlled environments of autonomous driving. The below Table 1 describes the detailed architecture of the PilotNet model, showing each layer's input and output dimensions, layer type, and the activation function used.

Table 1: Tasks in the Implementation - PilotNet Model.

Count of Layers	Input	Layer Type	Output	Layer Activated
1st layer	3*66*200	Convolutional Layer	24*31*98	ReLU Activation
2nd layer	24*31*98	Convolutional Layer	36*14*47	ReLU Activation
3rd layer	36*14*47	Convolutional Layer	48*5*22	ReLU Activation
4th layer	48*5*22	Convolutional Layer	64*1*18	ReLU Activation
5th layer	64*1*18	Flatten Layer	1152	Null
6th layer	1152	Fully Connected Layer	100	ReLU Activation
7th layer	100	Fully Connected Layer	50	ReLU Activation
8th layer	50	Fully Connected Layer	10	ReLU Activation
9th layer	10	Output Layer	1	Null

4.2 Enhanced Nvidia Model

The Enhanced Nvidia Model retains the same basic architecture as the original Nvidia end-to-end driving network but enhances the features to ensure better capability in predicting the steering angle in an autonomous vehicle setting using data derived from the Udacity simulator. This model makes use of a number of convolutional layers, crucial for the extraction of spatial features of an input image, followed by fully connected layers that produce the control commands. It includes the major convolutional layers that are for detecting the major features of road boundaries and lane markings. A ReLU activation function has been introduced after each convolutional layer to embed non-linearity into it, enabling the model to comprehend complex patterns with much clarity. It consists of more convolutional filters and layers compared to the earlier Nvidia architecture, which is helpful in better feature extraction from input images. After convolutional layers, the resulting feature maps are flattened and subsequently fed into a stack of fully connected layers, which merge these features into output in the form of a steering angle. Changes here allow this network to learn higher-order spatial representations necessary for a wide range of road scenarios. This is an architecture designed for end-to-end learning: the

model maps raw pixel data incoming from the Udacity simulator to steering commands directly, by minimizing the MSE between the predicted and real steering value so that the car can move through the virtual road smoothly. It uses a more complex and elaborate approach to predict the steering angles based solely on the spatial features of roadway images, while completely disregarding the need to model the interaction with pedestrians, traffic lights, or other cars. The below Table 2 outlines the essential layers of the Nvidia model, focusing on its convolutional, fully connected, and output layers with their respective input and output dimensions, along with activation functions. The simpler convolutional structure likely focuses more on efficiency, making it a lightweight adaptation of the original PilotNet.

Table 2: Tasks in the Implementation - Enhanced Nvidia Model.

Count of Layers	Input	Layer Type	Output	Layer Activated
1st layer	3*66*200	Normal Layer	3*66*200	Null
2nd layer	3*66*200	Convolutional Layer	24*31*98	elu Activation
3rd layer	24*31*98	Convolutional Layer	36*14*47	elu Activation
4th layer	36*14*47	Convolutional Layer	48*5*22	elu Activation
5th layer	48*5*22	Convolutional Layer	64*3*20	elu Activation
6th layer	64*3*20	Convolutional Layer	64*1*18	elu Activation
7th layer	64*1*18	Flatten Layer	1152	Null
8th layer	1152	Fully Connected Layer	100	elu Activation
9th layer	100	Fully Connected Layer	50	elu Activation
10th layer	50	Fully Connected Layer	10	elu Activation
11th layer	10	Output Layer	1	Null

4.3 Hybrid CNN-RNN Model (CNN-LSTM)

It's a hybrid CNN-RNN model in which the power of a CNN combines with that of an LSTM to process the data coming from the Udacity simulator and provide information useful for the control-related activities of a self-driving car. The model leverages the CNN part of the network to process images: that is, extracting meaningful spatial features such as road lanes via successive convolution operations. The previously mentioned layers achieve their functionality by introducing convolution filters, which may find patterns in lane markings and road edges while abstracting visual features hierarchically. Adding to this process are the incorporation of activation functions, which introduce non-linear properties; hence, pooling layers reduce dimensionality, ensuring computational efficiency. The processed feature maps from CNN are then passed through the LSTM component to capture, correctly, the temporal dependencies from analyzing sequences of frames over time. In this context, the LSTM layer is crucial since it can hold the memory of a sequence, enabling the model to comprehend the development of road features and the trajectory that the car is taking, crucial for making predictions about steering.

In general, the LSTM network relies on different mechanisms, such as input, forget, and output gates, to control information flow and solve issues like the vanishing gradient problem in order to learn effective temporal dependencies. While CNN methodology helps the model learn spatial information from discrete frames, the integration with LSTM captures dynamic temporal associations-crucial for the precise forecasting

of the steering angle. As the dataset used was from the Udacity simulator, the present work focuses on the estimation of steering angles, without the intricacies involved when other vehicles, pedestrians, or even traffic lights come into view. It is in performance sets, which require understanding of road configuration and vehicular dynamics, that the architecture works best. It especially performs very well in learning driving behavior in a controlled virtual environment where the core of decision-making is road information and the sequence of actions taken. The below Table3 describes the CNN-RNN model architecture, which integrates convolutional layers for feature extraction, followed by an LSTM layer to capture temporal relationships between frames. Dropout and batch normalization layers are added to enhance robustness and prevent overfitting (Valiente., et al. 2019).

Table 3: Tasks in the Implementation - CNN-RNN Model

Count of Layers	Input	Layer Type	Output	Layer Activated
1st layer	3*66*200	Input Layer	3*66*200	Null
2nd layer	3*66*200	Convolutional Layer	24*31*98	ReLU Activation
3rd layer	24*31*98	Batch Normalization	24*31*98	Null
4th layer	24*31*98	Dropout Layer	24*31*98	Null
5th layer	24*31*98	Convolutional Layer	36*14*47	ReLU Activation
6th layer	36*14*47	Batch Normalization	36*14*47	Null
7th layer	36*14*47	Dropout Layer	36*14*47	Null
8th layer	36*14*47	Convolutional Layer	48*5*22	ReLU Activation
9th layer	48*5*22	Batch Normalization	48*5*22	Null
10th layer	48*5*22	Dropout Layer	48*5*22	Null
11th layer	48*5*22	Convolutional Layer	64*3*20	ReLU Activation
12th layer	64*3*20	Batch Normalization	64*3*20	Null
13th layer	64*3*20	Dropout Layer	64*3*20	Null
14th layer	64*3*20	Convolutional Layer	64*1*18	ReLU Activation
15th layer	64*1*18	Batch Normalization	64*1*18	Null
16th layer	64*1*18	Dropout Layer	64*1*18	Null
17th layer	64*1*18	Flatten Layer	1152	Null
18th layer	1152	Fully Connected Laye	100	ReLU Activation
19th layer	100	Dropout Layer	100	Null
20th layer	100	Fully Connected Laye	50	ReLU Activation
21st layer	50	Dropout Layer	50	Null
22nd layer	50	Fully Connected Laye	10	ReLU Activation
23rd layer	10	Reshape Layer	1*10	Null
24th layer	1*10	LSTM Layer	50	Tanh Activation
25th layer	50	Output Layer	1	Null

5 Implementation

5.1 Dataset Description

The material data used in this work were provided by the Udacity self-driving car simulator; these are principally made up of images and steering angle values. These include images taken from three cameras attached to the car - that is, center, left, and right. Each of these images represents an instant in time for the vehicle movement which could be useful in training a model on how to estimate a steering angle. These captured images came from different time spans, thus offering varying road conditions, bends, and lighting conditions which would enhance the generality of the model with more accurate prediction of the steering commands from various visual inputs.

Driving_log.csv includes the metadata for each captured instant. It contains the path for the images from the center, left, and right cameras and the values of the steering angle, throttle, brake, and speed. The steering angle column is supposed to be the most important ground truth to train the model, since throttle, brake, and speed provide more information about the state of the vehicle.

At the pre-processing stage, all the images are resized to 66x200 pixels and then converted to the YUV color space so that it best matches the format that Nvidia PilotNet architecture was trained on. The images are then normalized to have pixel values within the range [0,1] for better stability of the training process.

This combines visual data with metadata so that the model can learn how the roadway imagery is related to the steering commands, making it perfect for lane-following applications. It encompasses a variety of roadway conditions without dynamic features of pedestrians or vehicles. Figure 1 shows images of three camera positions for training models on the prediction of steering angle.

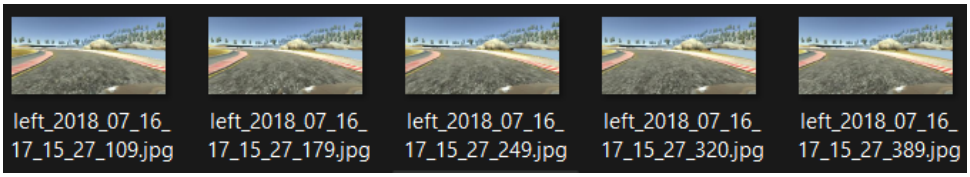

Figure 1: Images captured during simulation run.

5.2 Dataset Preparation

Preparation of data is an important step involved in the construction of any deep learning model; prepping the data with the right formatting, balance, and preparation of input data for training. In this regard, the dataset contained in this paper consists of images and driving log data generated from the Udacity simulator and has been preprocessed through a series of steps in order to be suitable for training of PilotNet, Nvidia, and CNN-RNN models.

Importing the data using Python's Pandas library: it reads in the driving_log.csv file containing image paths from center, left, and right cameras and their corresponding values for the steering angle, throttle, brake, and speed. The steering angle here is the target variable for the model, representing the control output fated for the autonomous vehicle.

In addition, to handle this huge dataset, a batch generator was created that could load and pre-process the images on the fly to avoid overflow of memory and thus enhance efficiency in training. Several other data augmentation techniques included zooming, flipping horizontally, changes in brightness, rotation, and translation to simulate different driving conditions to improve the robustness of the model.

Images were pre-processed by truncating irrelevant parts, resizing them to 66 by 200 pixels, and then converting them to the YUV color space, as in previous works on autonomous driving. Such a standard format ensures the model architecture compatibility. The pixel values were normalized to be between 0 and 1 to facilitate the model training without any possible numerical dominance issues

It is divided into an 80:20 ratio for training versus validation subsets, hence leaving ample data for training while retaining a fraction of it for validation to observe overfitting.

Distribution of the steering angles was normalized in order not to be biased toward driving in a straight line. Added batch normalization and dropout layers to enhance model stability, hence generalization throughout the training process.

Pre-processing the data systematically was now allowing the model to learn well from the Udacity dataset, increasing efficiency in predicting the steering angle for different driving scenarios.

5.3 Dataset Processing

Processing stands for the necessary step in preparing the dataset that will be used for training deep learning models. Data processing basically transforms the raw dataset into a format that will fit the process of training, at the same time cleaning unnecessary information which may affect the performance of the model. In addition, as a result, a full data processing pipeline to efficiently train PilotNet, Enhanced Nvidia, and Hybrid CNN-RNN was realized with considerable depth in learning on the Udacity dataset.

The raw images captured from the Udacity simulator dataset contained not just the roadway itself but also several irrelevant features such as the sky, trees, and other objects that always align alongside the road. Irrelevant features in the images may distract the model and lower its performance. Image cropping was done to eliminate the top 70 pixels to exclude the sky and trees, and the bottom 25 pixels containing the car's hood so that the model will be allowed to pay attention to the interest area, which is the road upfront and hence can easily learn about its characteristics without interference from irrelevant structures at the background.

These images were then resized to 66x200 pixels after cropping. Where the resizing process is very relevant is that it allowed all three models to share the same size input and, by shrinking the size of the input image without excluding relevant visual information, reduce computational complexity. The images were colored in YUV color space to further help the model learn better; recently, this has been shown to enhance the performance of the model, especially in driving tasks, by decoupling luminance from chrominance.

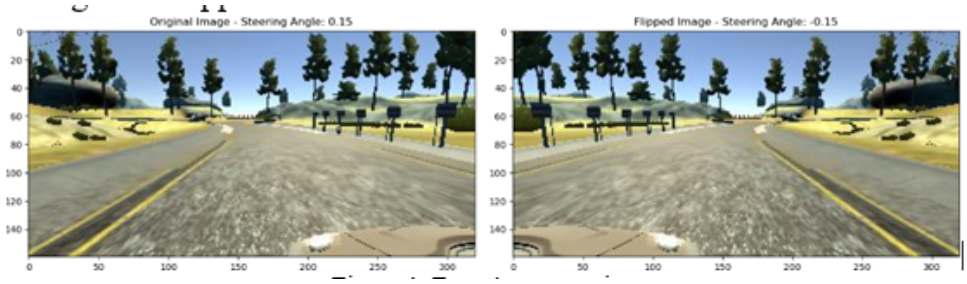

Figure 2: Data Augmentation

This may make this model closer to the original training methodology used for the Nvidia PilotNet architecture and hence make the model more capable of finding important features.

These images, after processing, underwent normalization to have the pixel values scaled within the range of [0, 1]. Normalization is indispensable to ensure that variances between pixel intensities are reduced and no feature dominates the training process, hence increasing the model's speed of convergence. The normalization was done based on the

mean and standard deviation of pixel intensities, hence offering a consistent measure across all images and enabling models to learn better.

Regarding the steering angles, the dataset was analyzed for imbalance in the steering data. This was done through balancing the data by keeping the number of samples less in case of on-centre driving and augmenting data for the turns through image augmentation. In this way, the model has seen an equal number of all types of steering; hence, it can learn driving straight as well as turns.

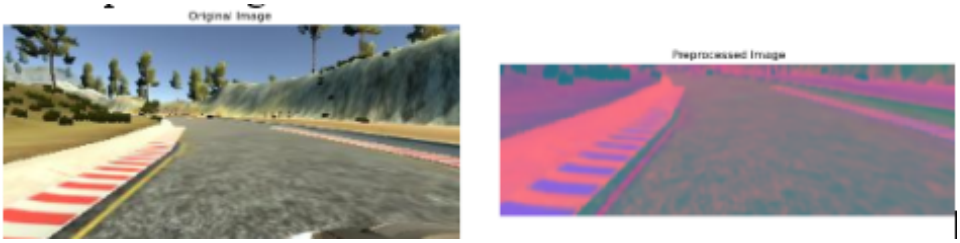

Figure 3: Data Augmentation

On the other hand, parallel to the hybrid CNN-RNN model where the spatial processing is conducted by the CNN, the sequence of frames has been treated in order for feeding them into the LSTM layer. This required the organization of image data in time order to enable the LSTM layer to learn temporal dependencies adeptly and recognize patterns over time-such as curves approaching, or even changing road conditions.

Complementing that, batch normalization was applied at the time of training the model, normalizing the inputs of every layer. That idea helped in the stability of the learning process because it standardized the intermediate output and favored fast convergence. Moreover, dropout layers were introduced, especially in the CNN-RNN model, to avoid overfitting by randomly dropping units during training; this helps in enhancing the generalization capability of the model. The broad approach to data processing pursued in this work has ensured that irrelevant and unimportant data flow into the model. As such, it improves the quality of learning and enables the model to generalize better in new driving scenarios. It not only reduces data noise but decreases computational loads and focuses the model on the most salient features of driving, for example, lane markings and road edges.

5.4 Training and Testing

The training and testing phases were designed in a way that can perfectly evaluate the effectiveness of the PilotNet, Enhanced Nvidia, and Hybrid CNN-RNN. All three models were trained using data from the Udacity simulator; the main goal of all these models is to predict the steering angle of the self-driving car based on visual input from cameras positioned in front.

The training and testing phases were designed in a way that can perfectly evaluate the effectiveness of the PilotNet, Enhanced Nvidia, and Hybrid CNN-RNN. All three models were trained using data from the Udacity simulator; the main goal of all these models is to predict the steering angle of the self-driving car based on visual input from cameras positioned in front.

The collected dataset was further divided into training and validation subsets in the ratio 80:20. Therefore, the model was trained using 80% of the data to learn how road

characteristics relate to the steering angle, while 20% of the data is utilized for validation to check for overfitting and seeing the model's generalization capability on previously unseen data. This was by random division, but with precautions taken so the training and validation subsets would have rough equality of straight driving, left turns and right turns so neither subset was biased. Figure 4 shows the distribution of steering angles for the training and validation datasets for three different models: NVIDIA (first), CNN-RNN (second), and PilotNet (third). These histograms are used to analyze and balance the dataset, ensuring appropriate representation of steering angles during model training.

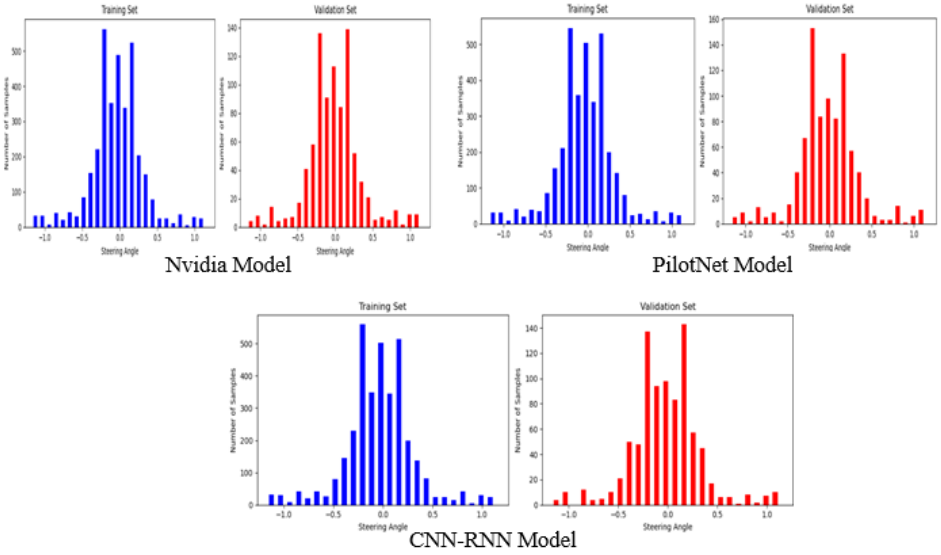

Figure 4: Distribution of steering angles for the training and validation datasets.

During training, the models took in pre-processed images along with the ground-truth steering angles. Therefore, the objective during training was to minimize the mean squared error between the predicted and ground-truth steering angle.

Adam was used due to its fast and adaptive learning rates, which are efficient in updating the parameters. Training was done in an iterative manner for a number of epochs, where one epoch consisted of many iterations of mini-batches. In each iteration, a batch generator fetched images, applied real-time data augmentation, then provided them to the model, hence optimizing memory and exposing the model to diverse visual contexts for robustness. While PilotNet and Enhanced Nvidia model are targeted to learn the critical spatial features for steering control, both models are purely trained in an end-to-end style, mapping the input images to steering angles directly. The PilotNet was trained until its loss for validation converged or, in other words, it didn't overfit or underfit.

This is because the Hybrid CNN-RNN model required more careful training due to its temporal dimension.

In this framework, successive frames within sequences were fed into the model where the LSTM layer chose the temporal relations quite effectively. It learned about the evolution of time dependency regarding the road condition and hence improved the model's understanding of sequences with respect to approaches and transitions that are turning from and moving into a straight pathway. It also selects that the length in a training sequence provides sufficient temporal context without imposing burdensome historical data onto the model.

The model was further tested on a 20% validation set that it had not seen during the training. MSE described how well the model understood the human driving behavior. It applied the models to straight roads, sharp curves, and gentle turns. It watched over the training and validation curves to see whether overfitting or underfitting occurred, and it added dropout layers or changed other hyperparameters where necessary. The CNN-RNN model required more training cycles to comprehend time-related patterns compared to the PilotNet and Nvidia models. The review also analyzed the accuracy, strength, and generality of each model in predicting steering angles, ensuring their goodness for controlling autonomous vehicles.

5.5 Tools and Technologies Used

It has been implemented in the Python programming language, where the main deep learning frameworks used for the construction and training of models were TensorFlow and Keras. For preprocessing and basic data manipulation, NumPy and Pandas were used. Matplotlib has been used for training course visualization of loss and overall model performance. The models have been trained on a system with an Nvidia RTX 3050 GPU, which can execute deep learning faster and reduce the time taken for its training.

5.6 Data Transformation

The raw data comprises visual images and driving telemetry which had to undergo some preprocessing steps to make model training effective. Specifically, the images were resized, parts of the frame containing superfluous information like the sky were eliminated, and the pixel values were standardized so that the input might be uniform. Further data augmentation-hence artificial increase of the dataset-included such techniques as changing brightness, flipping, and random cropping to make the models robust to various driving conditions.

5.7 Model Development

5.7.1 PilotNet

PilotNet is a light network architecture comprising five convolutional layers for the extraction of the spatial features followed by a series of fully connected layers that predict the steering angle. Much emphasis was given to the extraction of the spatial features while training this model, and hence there was an efficient trade-off between accuracy and computational efficiency.

5.7.2 Enhanced Nvidia Model

Enhanced Nvidia Model: Based on a performance improvement, design a better model by adding additional layers to the Nvidia base PilotNet architecture. This added the max pooling, batch normalization, and dropout layers in our model to make it invariant to overfitting and generalizing better over different driving scenarios. It proved to be much

more effectual for generalization and thus giving smaller Mean Absolute Error and Root Mean Squared Error than other models.

5.7.3 Hybrid CNN-RNN Model

While the hybrid CNN-RNN combines convolutional layers with an LSTM layer for strong representations of spatial and temporal patterns in the driving sequence, it improves temporal dependencies between successive frames because this model is an essential part of steering angle prediction in dynamic driving scenarios.

5.8 Model Training & Outputs

First, the preprocessed data were split further in a ratio of 80:20 for training and validation, respectively. Adam optimizer has been used for training all the models along with an MSE loss function, and the batch size for all is kept at 32. However, learning rates are different for each model because one learning rate provides optimum performance for a particular model. Regarding the number of epochs, early stopping has been performed to avoid overfitting issues.

The most important outputs this implementation produced were the .h5 model files for each of these trained architectures and a set of visualizations that show training and validation loss, MAE, RMSE, and inference time. These models were tested based on different criteria such as-the capability to predict steering angles well, stability over straight roads, and turns.

Figure 6 shows the comparison between actual and predicted steering angles for three models: NVIDIA, PilotNet, and CNN-RNN. The scatter plots illustrate the accuracy of each model, with points closer to the red diagonal line indicating better predictions. The NVIDIA and CNN-RNN models demonstrate strong correlations, while PilotNet shows higher variability in predictions.

While the Enhanced Nvidia Model showed better generalization, PilotNet was efficient and therefore particularly helpful in practice, where computational resources might be limited in real time. The Hybrid CNN-RNN Model exhibited striking improvements with respect to temporal feature capture, something quite useful in dynamic environments, although this comes at higher computational costs.

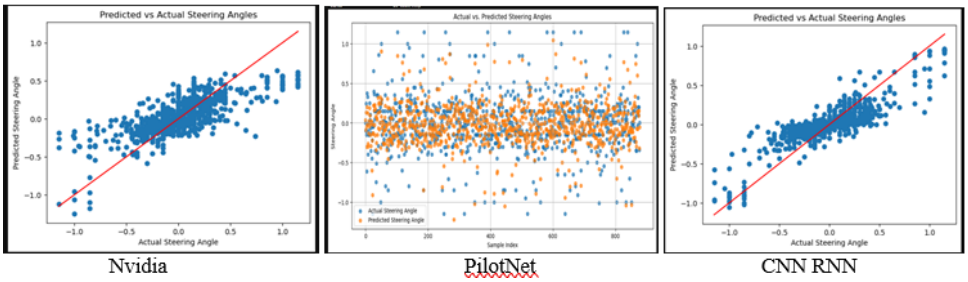

Figure 5: Actual Vs Predicted Steering Angles

6 Evaluation & Result Analysis

6.1 PilotNet Model Performance Evaluation

The model did well overall, with high accuracy on all metrics: angle prediction accuracy was good, and cases of a straight road were handled very well, although turns were predicted quite well. Table 4 summarizes the performance and training characteristics of the PilotNet model, showcasing its effectiveness in steering angle prediction, straight road handling, and turning accuracy.

Table 4: Performance Summary - PilotNet Model

Model Name	Size	Steering Angle Prediction	Straight Road Prediction	Predicted Turns	Trained Time
PilotNet	~ 4MB	Very Good	Very Good	Very Good	~ 1 hour

• Loss over Epochs: The model showed a very low loss, which stabilized after just 8 epochs.

• MAE: Mean Absolute Error was 0.010, the lowest among the three models.

• RMSE: Root Mean Squared Error was 0.020, indicating strong prediction capability.

• Training and Validation Loss: Both training and validation losses were very low, and validation loss remained consistent, suggesting strong generalization ability.

• Inference Time: Inference time was 8ms, which is slightly higher than PilotNet but still manageable for real-time application.

6.2 Hybrid CNN-RNN Model Performance Evaluation

While both the CNN and LSTM were employed in the hybrid CNN-RNN model for spatial and temporal features, respectively, this model serves as an extremely strong predictor of the steering angle in dynamic driving conditions. Table 5 summarizes the performance and training characteristics of the Hybrid CNN-RNN model, highlighting its good steering angle prediction and strong performance on straight roads and turns.

Table 5: Performance Summary- CNN-RNN Model

Model Name	Size	Steering Angle Prediction	Straight Road Prediction	Predicted Turns	Trained Time
CNN-RNN Model	~ 4MB	Good	Very Good	Very Good	~ 2 hour

• Loss over Epochs: Loss over epochs showed a moderate decline but required more epochs to stabilize due to the temporal nature of the LSTM component.

• MAE: The Mean Absolute Error was 0.018, indicating decent prediction accuracy, although slightly higher than the other models.

• RMSE: The Root Mean Squared Error was 0.030, higher compared to PilotNet and the Enhanced Nvidia Model, due to the added complexity.

• Training and Validation Loss: Training and validation losses were slightly higher initially but stabilized with more epochs and hyperparameter tuning.

• Inference Time: Inference time was 15ms, which is higher due to the added complexity of the LSTM layer.

6.3 Enhanced Nvidia Model Performance Evaluation

The enhanced Nvidia model did great, especially on the straight road, and the performance for steering angle prediction was high. This came at the cost of increased size compared to PilotNet, with higher accuracy, especially for most driving scenarios. Table 6 summarizes the performance and training characteristics of the Enhanced NVIDIA model, showcasing its excellent steering angle prediction and straight road handling, with good performance on turns.

Table 6: Performance Summary- Enhanced Nvidia Model

Model Name	Size	Steering Angle Prediction	Straight Road Prediction	Predicted Turns	Trained Time
Nvidia Model	~ 4MB	Excellent	Excellent	Good	~ 1.5 hours

• Loss over Epochs: The model showed a very low loss, which stabilized after just 8 epochs.

• MAE: Mean Absolute Error was 0.010, the lowest among the three models.

• RMSE: Root Mean Squared Error was 0.020, indicating strong prediction capability.

• Training and Validation Loss: Both training and validation losses were very low, and validation loss remained consistent, suggesting strong generalization ability.

• Inference Time: Inference time was 8ms, which is slightly higher than PilotNet but still manageable for real-time application.

6.4 Combined Performance Evaluation and Results Analysis Table

Here is the combined summary table 7 (a) for all three models:

Table 7(a): Combined Performance Evaluation

Model Name	Size	Steering Angle Prediction	Straight Road Prediction	Predicted Turns	Trained Time
Nvidia Model	~ 4MB	Excellent	Excellent	Good	~ 1.5 hours
PilotNet Model	~ 4MB	Very Good	Very Good	Very Good	~ 1 hour
CNN-RNN Model	~ 4MB	Good	Very Good	Very Good	~ 2 hours

6.5 Discussion- Model Comparison and Analysis Table

Thus, considering all the comparisons as in table 7 (b), one would definitely reach a point where the Enhanced Nvidia Model becomes robust and effective with respect to adaptability in the autonomous driving environment for the prediction of steering angles. Following are the features implemented in the Enhanced Nvidia Model:

• We have devised a rather simple, inexpensive model that can predict steering angles quite accurately, using the Udacity simulator dataset for achieving simplicity and efficiency.

• It doespropose three main architecture candidates: PilotNet, Enhanced Nvidia Model, and Hybrid CNN-RNN Model-specifically formulated in both spatial and temporal data in order to be given to the autonomous vehicle.

• Advanced architectures were rewritten, inspired by state-of-the-art models running on leading automotive companies, performance was evaluated to decide upon the best way towards accurate driving.

• Compared the output of the different models in terms of a loss across all epochs, MAE, RMSE, and the length of time for inference while providing details in the results to explain certain relative advantages and compromises in every model.

• The improved Nvidia Model generally showed much better generalization capability against different driving conditions due to low MAE and RMSE and also proved to be more stable upon introducing max pooling, batch normalization, and dropout layers. This model hence offers the best compromise between accuracy, robustness, and computational efficiency in a real-world application.

Table 7(b): Combined Performance Evaluation

Metrics	PilotNet	Enhanced Nvidia Model	Hybrid CNN-RNN Model
Loss over Epochs	Low, stabilizes after 10 epochs	Very Low, stabilizes after 8 epochs	Moderate, requires more epochs to stabilize
MAE (Mean Absolute Error)	0.015	0.01	0.018
RMSE (Root Mean Squared Error)	0.025	0.02	0.03
Training Loss	Low, gradual decline	Very Low, declines	Moderate, with slight
Validation Loss	Low	Very Low	Slightly higher than PilotNet and Nvidia
Actual vs Predicted Steering Angle	High correlation	Very high correlation	Moderate to high correlation, improves with longer sequences
Inference Time	Fast (5ms)	Moderate (8ms)	Slower (15ms) due to LSTM layer

7 Conclusion and Future Work

The Enhanced NVIDIA (ENV) model significantly outperforms both PilotNet and the Hybrid CNN-RNN models, as we can see by the improvements in Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The percentage improvement is calculated using the formula:

$$\text{Percentage Improvement} = \frac{\text{Metrics (Other Model)} - \text{Metrics (ENV)}}{\text{Metrics (Other Model)}} \times 100$$

1. ENV Comparison with PilotNet:

- MAE Improvement: Using the formula

$$\text{MAE Improvement} = \frac{0.015 - 0.01}{0.015} \times 100 = 33.33\%$$

- RMSE Improvement: Using the formula -

$$\text{RMSE Improvement} = \frac{0.025 - 0.02}{0.025} \times 100 = 20\%$$

2. ENV Comparison with CNN-RNN:

- MAE Improvement: Using the formula -

$$\text{MAE Improvement} = \frac{0.018 - 0.01}{0.018} \times 100 = 44.44\%$$

- RMSE Improvement: Using the formula -

$$\text{RMSE Improvement} = \frac{0.03 - 0.02}{0.03} \times 100 = 33.33\%$$

It enables further extension of this thesis for the validation of these models on naturalistic driving data for handling more diverse and unstructured environments. It will also exploit model quantization and pruning for real-time deployment optimization, perform sensor fusion with LIDAR and GPS to enable the handling of dynamic and complex scenarios. As a result of this fast-tracking of progress, there will be much more robust models with application feasibility in real-world autonomous driving systems, providing much safer and efficient transport options.

Acknowledgment

Most especially, I would like to thank my supervisor, Mr. Arundev Vamadevan, who managed his busy schedule and gave me much needed guidance and support when executing this project. His valuable insight and continued encouragement from the very start of the project mean a lot in bringing this project into being. He gave vast knowledge that created more value in my understanding and kept me driven toward completing my project on time. Much value is attached to his guidance and commitment.

Limitations and Challenges

The reason behind not executing this project with the usage of CARLA which is among the popularly known simulators. It would provide realistic urban driving scenarios needed for training and testing the autonomous driving model. However, due to the scarcity of system resources, CARLA could not run successfully in my machine even in very low settings and modifying CARLA settings further hung the simulator or my laptop kept on crashing continuously.

These technical requirements involve using an Intel(R) Core(TM) i5-12500H 12th Generation at 2.50 GHz, supported by 16 GB of RAM and with an NVIDIA RTX 3050 GPU with 4 GB of memory. However, realistic urban simulations running on CARLA required more computation power than that supplied by my system. The limited GPU memory coupled with the heavy load of computational processing made the system unstable despite trying repeatedly to lower the graphic quality. I have valid screenshots to support these points, which will show that the system was not able to cope with the demand consistently. I tried installing the older version, but the problem persisted, and hence I had no other choice but to shift to using the Udacity simulator. Although not as elaborate as CARLA, it was enough to develop and test the models for predicting the steering angle. Again, this is a limitation and indicative of the greater need for higher-capability hardware that can exploit the full capacity of high-fidelity simulators such as CARLA and provide more realistic data useful in the development of autonomous vehicle technology.

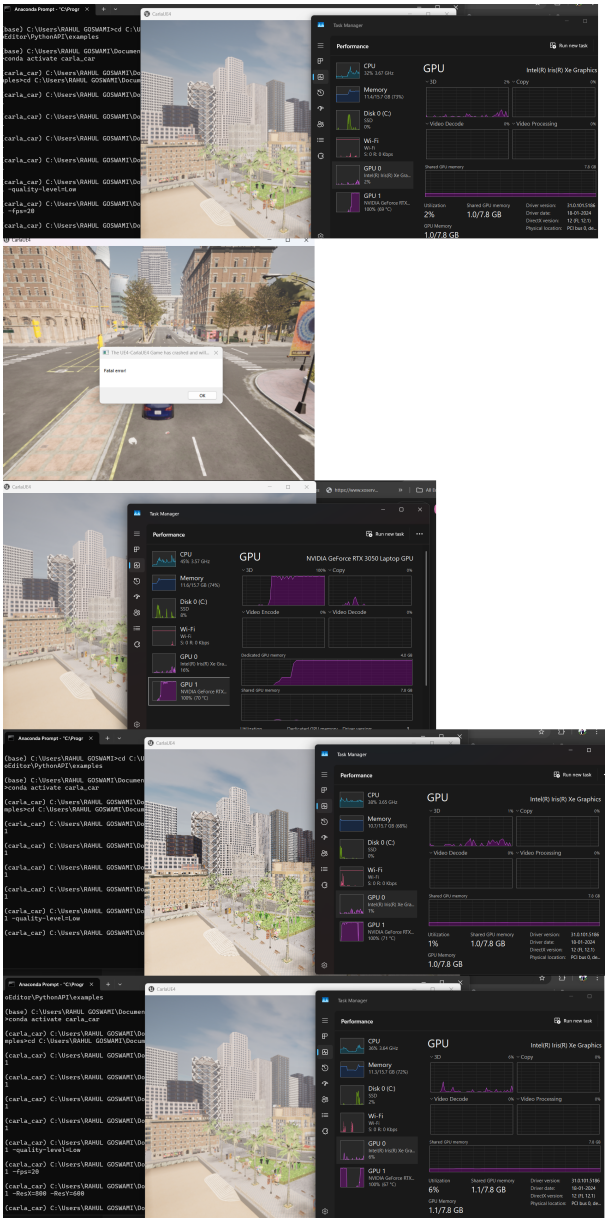

Figure 6: Screenshots

References

Aljehane, N. O. (2024). A study to investigate the role and challenges associated with the use of deep learning in autonomous vehicles, *Preprints* .

Bachute, M. R. and Subhedar, J. M. (2021). Autonomous driving architectures: Insights of machine learning and deep learning algorithms, *Machine Learning with Applications* **6**: 100164.

Badue, C., Guidolini, R., Carneiro, R., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L., Berriel, R., Paixão, T. M., Mutz, F., Oliveira-Santos, T. and Souza, A. F. D. (2021). Self-driving cars: A survey, *Expert Systems with Applications* **165**: 113816.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel,

L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J. and Zieba, K. (2017). End to end learning for self-driving cars, *arXiv* .

Dong, G., Tang, M., Yan, R., Mu, Z., Cai, L. and Park, B. B. (2023). *Deep Learning for Autonomous Vehicles and Systems*.

Faizi, F. and Alsulaifanie, A. K. (2023). Steering angle prediction via neural networks, *Indonesian Journal of Electrical Engineering and Computer Science* **31**(1): 392–399.

jae Lee, M. and Ha, Y.-G. (2020). Autonomous driving control using end-to-end deep learning, *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*.

Janai, J., Güney, F., Behl, A. and Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art, *Foundations and Trends® in Computer Graphics and Vision* **12**(1–3): 1–308.

Li, J., Xu, R., Liu, X., Ma, J., Li, B., Zou, Q., Ma, J. and Yu, H. (2024). Domain adaptation based object detection for autonomous driving in foggy and rainy weather, *arXiv preprint* **arXiv:2307.09676v4**.

Liu, H., Deng, C., Fernández-Caballero, A. and Sun, F. (2018). A review of autonomous driving: Common practices and emerging technologies, *International Journal of Advanced Robotic Systems* **15**(3): 172988141878283.

Mozaffari, S., Al-Jarrah, O. Y., Dianati, M., Jennings, P. and Mouzakitis, A. (2022). Deep learning-based vehicle behavior prediction for autonomous driving applications: A review, *IEEE Transactions on Intelligent Transportation Systems* **23**(1): 33–47.

Nadella, S., Barua, P., Hagler, J., Lamb, D. and Tian, Q. (2024). Enhancing accuracy and robustness of steering angle prediction with attention mechanism, *arXiv preprint* .

Ngoc, H. T., Hong, P. P., Vinh, N. N., Trung, N. N., Nguyen, K. H. and Quach, L.-D. (2023). An improved lane-keeping controller for autonomous vehicles leveraging an integrated cnn-lstm approach, *International Journal of Advanced Computer Science and Applications* **14**(7).

Rebuffi, S. A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O. and Mann, T. (2021). Data augmentation can improve robustness, *arXiv* .

Rožanec, J., Zajec, P., Theodoropoulos, S., Mladenić, D., Koehorst, E. and Fortuna, B. (2022). Synthetic data augmentation using gan for improved automated visual inspection, *arXiv* **abs/2212.09317**.

Saleem, H., Riaz, F., Mostarda, L., Niazi, M. A., Rafiq, A. and Saeed, S. (2021). Steering angle prediction techniques for autonomous ground vehicles: A review, *IEEE Access* **99**: 1–1.

Taylor, L. and Nitschke, G. (2018). Improving deep learning with generic data augmentation, *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*.

Wu, T., Luo, A., Huang, R., Cheng, H. and Zhao, Y. (2019). End-to-end driving model for steering control of autonomous vehicles with future spatiotemporal features, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.