ABSTRACT

There are several autonomous vehicles (AVs), often known as self-driving cars, on the roads today. In 2035, AVs are projected to make up to 25 percent of all vehicles worldwide. Steering angle prediction, which is crucial for autonomous vehicles' (AVs') management, has drawn interest from researchers, manufacturers, and insurance firms in the automotive sector. In the proposed method, image data taken from a front dash-cam has been taken as input and the steering angle(in degrees) at the particular instance is the output received. The model predicts the angle based solely upon the image data it receives. The core part of the model architecture is Convolutional Neural Network. The working of the model has further been tested in a simulation.

Keywords: Convolutional Neural Networks (CNN), Tensorflow, Autonomous Vehicles (AVs), Deep Learning (DL)

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 How Autonomous Vehicles are becoming a substitute for human drivers?

The National Motor Vehicle Crash Causation Survey (NMVCCS) [1], conducted from 2005 to 2007, gathered information about the factors that contributed to street accidents. It found that more than 90% of auto collisions had a significant human factor, 2% had a mechanical factor, and 1.3% involved natural factors (streets, environment, and so on). The control architecture of autonomous vehicles includes both longitudinal and lateral regulators. Lateral regulators limit steering control, altering course, and keeping a safe distance from accidents, whereas longitudinal regulators manage the vehicle's cruise speed [4]. AVs have a larger range and a chance to make lesser mistakes as compared to human drivers.

1.2 Advantages of steering angle prediction with Deep Learning approach

The capacity of Deep Learning[15] to handle unlabeled raw data insufficiently. While observing a scene, DL is logically insensitive to different ecological conditions, but it needs a vast volume of data for achieving high accuracy. The advantages of prediction of steering angle using the DL technique include its ability to handle faults, ability to spot errors quickly, and improved ability to manage erratic situations[2]. A camera that scans the road in front of an autonomous car determines the steering wheel's attitude. An extremely sparse training signal allows the CNN to learn important road details. The ability of CNN to recognise objects and other elements, including as lane markers, road boundaries, other vehicles and objects on the road, is a function that would be challenging for engineers to manually program[11].

1.3 Challenges of Deep Learning in steering wheel angle prediction in AVs

Scientists heavily trust synthetic datasets for investigating how to use DL to forecast the steering angle of AVs. It is challenging to guide experiments so that the results may be applied in the actual world when using generated datasets [3]. Even though the simulated environment simulates the circumstance in the actual world, unforeseen events can nonetheless happen there. CNN finds it difficult to drive at faster speeds because it reacts to unforeseen situations very slowly. High memory and processing expenditures are required for CNN [12]. There is still time to improve and reach the ideal performance for regulating control utilizing DL, even though it has not yet been reached.

Chapter 2
Literature Survey

In order to learn more about the approaches that are currently being utilised for classification, we conducted a literature review. In one of the published articles, the causes of wilt's start and its many stages were quantified and examined.

[6] In order to remove human intervention, Akhil Agnihotri, Prathamesh Saraf, Kriti Rajesh Bapnad created a CNN that uses the highly varied highlights from the dashcam camera. platforms with an RGBD camera for continuous location monitoring and an ultrasonic sensor for identifying impediments. The steering angle is output by the Raspberry Pi, which is then transformed to angular velocity for steering.

[7] Davide Del Testa, Daniel Dworakowski, and Mariusz Bojarski have prepared CNN to arrange rudimentary pixels from a single forward-looking camera directly to directing orders. They assert that complete framework modifications made all handling procedures appear differently when there was path planning, clear division, lane-marking, and control.

[8] An end-to-end steering angle controller with CNN-based closed-loop feedback for autonomous vehicles was created in this paper by Junekyo Jhung, Jaeyoung Moon, and other authors. This controller performs better than CNN-based methods. Their work illustrates how the suggested model may learn to identify steering wheel angles for the lateral control of self-driving cars using supervised pre-training and subsequent reinforced closed-loop feedback utilizing images from a dashcam camera.

[9] Despite a platform that makes use of a convolutional neural network (CNN), that uses images of a forward-looking camera for input and generates vehicle directing points as yield, Michael G. Bechtel and Elise McEllhiney were able to employ it. We dissect the registration capabilities of the Pi 3 using their foundation to support continual profound learning-based control of self-sufficient vehicles from beginning to end. They examined top-tier hold distributing memory move speed gagging methods on the Pi 3 to ensure the CNN duty.

[10] The earliest findings on creating a weighted N-version programming (NVP) plot to ensure the adaptability of AI-based guiding control calculations were provided by Ailec Wu and Abu Hasnat Mohammad Rubaiyat. The combination of yields from three extra Deep Neural Network (DNN) models will determine the recommended conspiracies.

Reference	Methodology	Result	Limitations
A CNN Approach Towards Self-Driving Cars	CNN	The steering angle, which is transformed to angular velocity for steering, is the most changeable element that CNN learns for the camera input.	The network's robustness has to be improved.
Steering angle prediction for autonomous electric vehicles using end-to-end learning on the CNN platform	CNN	Comparing weighted model fusion to individual techniques, the average accuracy was 40%.	Accuracy of Model
Deep Learning for Automatic Steering Angle and Direction Prediction in Autonomous Driving	Deep Learning	Direction prediction was found right and it is the most efficient model with respect to other models so it is a good one.	It is necessary to enhance the discriminative sequential pattern mining algorithm.
Learning from Start to Finish for Self-Driving Cars	CNN	Learns significant features from sparse training steering angle	Accuracy of Model
For autonomous vehicles, an end-to-end steering controller with CNN-based closed loop feedback is used.	CNN	Achieved robust steering control even in settings with incomplete observation.	The simulation was unable to maintain sufficient lateral control while driving.
A low-cost autonomous vehicle powered by deep neural networks	CNN Raspberry pi3	Superior to other versions in terms of speed, and most economical	Overfitting issue, high power consumption

Table 2.2 Literature Survey Table

Chapter 3
Methodology

3.1 Dataset

The collection is derived from the Sullychen [13] dataset, which has 45406 photos in total. Picture 3.1 The dataset's images were captured using a dashcam with identified steering angles. The video is shown in continuous images. The data collection is later divided into train and test categories. Convolutional neural networks (CNN) are used to train the model once the dataset has been divided.

Figure 3.1 : Images record from a car dashcam with labeled steering angles[13]

3.2 Image Preprocessing

The dataset's images are split into training and test sets. We only use the lower (150 pixels) portion of the photos from each batch because we are only interested in roads. Then, for

simpler computation, the image was scaled from 455 X 150 to 200 X 66, and it was normalized (divided by 255) after that.

3.3 Model Architecture using CNN

The CNN model architecture is discussed in detail and how it is used in predicting the Steering Wheel Angle as shown in Figure 3.2 Figure 3.3.

3.3.1 Convolutional layer

The CNN layer takes an input volume and outputs a volume with a different shape thanks to the numerous filters it contains. Each filter extracts a distinct kind of feature and produces an activation map, and multiple activation maps are combined by stacking to make the output volume. Convolution is the process by which two real-valued features are combined to create a new real-valued role. Let f(t) and g(t) be two real valued functions, where t stands for consistent time.

3.3.2 Stride, Padding and Pooling

Stride: The size and movement of filters can vary. Stride specifies how an image filter should be applied to the image. In other words, stride is the number of pixels we repeatedly skip[5].

Padding: We have seen that convolution reduces the original image's height and weight, however there are occasions when we prefer the output image to have the same size as the input image. Therefore, placing 0 value pixels (neurons) outside of the original image—a process known as padding—will allow us to achieve this.

Pooling: Following the Convolution Operation, pooling is done. Average Pooling and Max Pooling are the two different types of pooling layers.

Max-pooling layer: The output records the maximum value of the window that is slid over the input. In terms of the max-pooling procedure,

$$\mathrm{P4}^{k} = MaxPool(C3^{k})P4^{k}_{i,j} = max\begin{pmatrix} C3^{k}_{(2i,2j)'} & C3^{k}_{(2i+1,2j)'} \\ C3^{k}_{(2i,2j+1)'} & C3^{k}_{(2i+1,2j+1)'} \end{pmatrix} \quad \textbf{(3.1)}$$

where (i, j) are indices of k th feature map of the output, and k is the feature map index[15].

A window is slid over the input in the average-pooling layer, which then stores the window's average value in the output. With 75% of the neurons being discarded, computation is reduced (assuming 2X2 filters with a stride of 2). Additionally, it strengthens feature detectors. There are no learning parameters in it; only hyperparameters like filter size and pooling type.

3.3.3 Activation Functions

Between the input that drives the current neuron and its output that drives the subsequent layer is a mathematical "gate" known as the activation function. It can be as simple as a step function that turns on and off the neuron output in accordance with a rule or threshold. There are many activation mechanisms, for instance, Relu, sigmoid, tanh, etc. Relu and tanh function, however, are utilized in this model. Rectified Linear Function is known as Relu. The maximum value is taken by the ReLu function. Therefore, it can be written as

$$f(x) = \max(x,0)$$

Using the rectifier function, we may make our images more nonlinear. A non-linear activation function called Tanh (Hyperbolic Tangent) compresses all of its inputs to the range [-1, 1].

$$f(x) = \frac{e^{x} - e^{-}x}{e^{x} + e^{-}x}$$

$$f'(x) = \frac{\delta f(x)}{\delta x} = 1 - (f(x))^{2} \quad \textbf{(3.2)}$$

We want to accomplish that since photos are inherently non-linear. Tanh's function is zero-centered, which allows gradients to move in any direction[17]. Any image you look at will show you that it has a lot of non-linear elements (e.g., the transition between pixels, the borders, the colors, etc.).

3.3.4 Optimizer

There are numerous optimizers, including Adagrad, Adadelta, Adam, and others. Adam is utilized in this project. The acronym for Adaptive Moment Estimation is Adam. This optimizer makes it possible to train the neural network quickly and effectively. Adam works on the premise that we don't want to roll so quickly just because we can hop over the local minima. For a thorough search, we want to slightly lower the velocity. Both an exponentially decaying average of the previous squared gradient and an exponentially decaying average of the preceding gradient are stored.

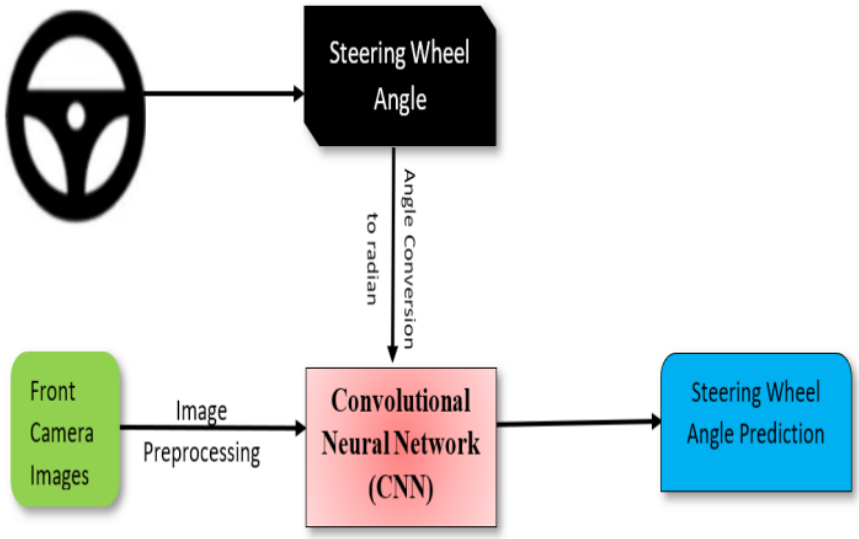

Figure 3.2: Figure describing how dash cam images and their corresponding steering wheel angle(in radians) are feeded into CNN to predict Steering Wheel Angle

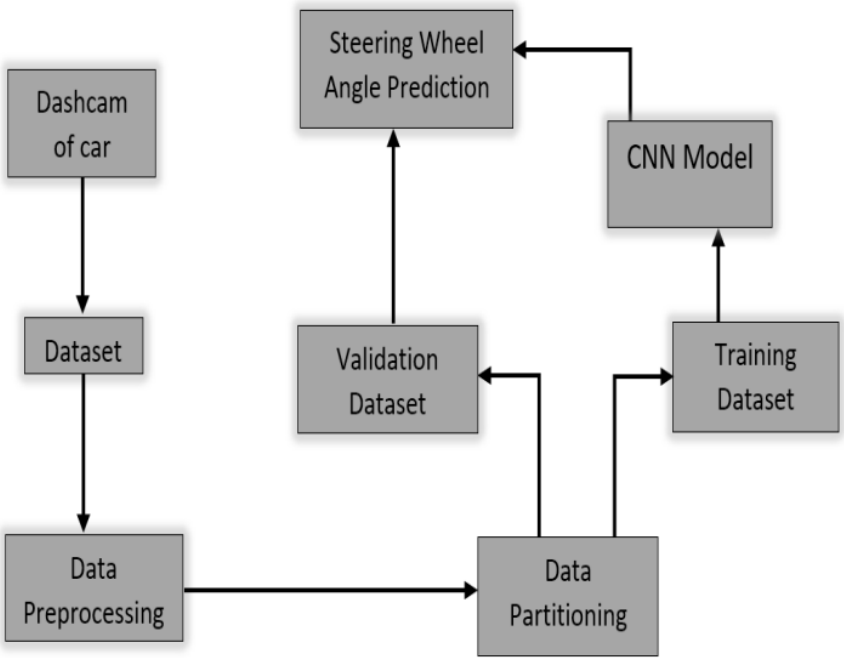

Figure 3.3: Overview of Model

Chapter 4

Implementation and Results

4.1 Environment

Python is used to implement the code. Tensorflow, Keras, and a few more frequently used Python libraries are available in the environment. A complete open-source machine learning platform is TensorFlow[12]. Researchers can advance the state-of-the-art in DL thanks to its extensive, adaptable ecosystem of tools, libraries, and community resources, while developers can simply create and deploy DL-powered apps. Keras is a high-level neural networks API that sits on top of TensorFlow.

4.2 Implementation

A CSV file with steering wheel angle data is part of the dataset utilized in this model. The steering wheel angle, which is proportional to the inverse of the turning radius, is utilized as the output because the angles in the CSV file are the inverse of the turning radius. We used an 80:20 split to separate the dataset into the train and validation sets. In order to improve efficiency, the dataset should thereafter be loaded in batches, each of size 100. For training and optimization in this model, Adam Tanh is the activation function employed in this model. In this model, CNN architecture shown in Figure 4.1 has been utilized. Five fully connected layers and five convolutional layers with some padding and stride makeup this model.

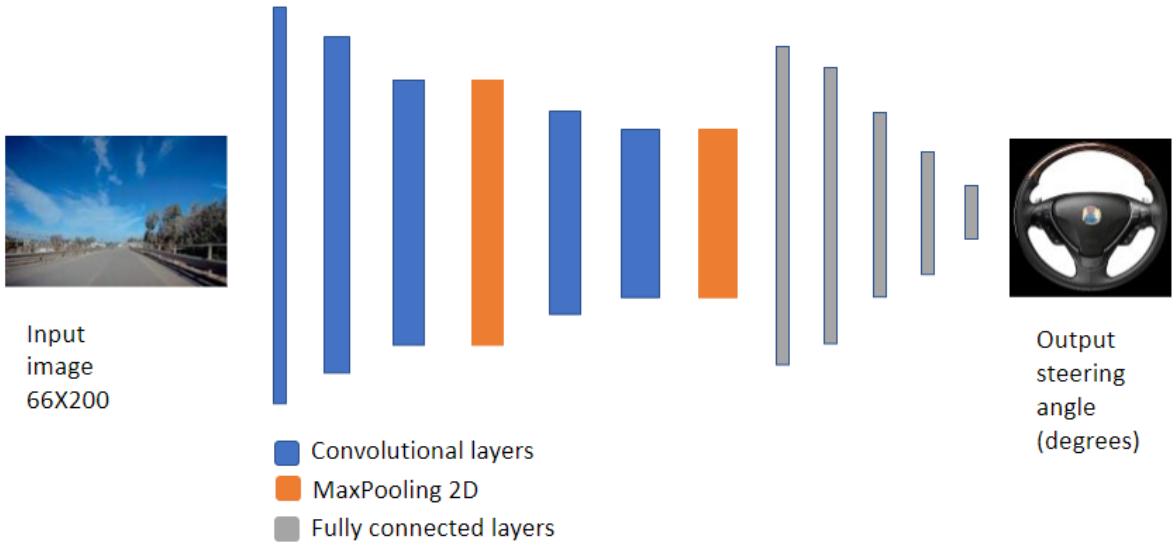

Figure 4.1: CNN architecture of Proposed Model. Around 27 million connections And 250 thousand characteristics make up the network

4.3 Results

4.3.1 CNN Model

The results showed that the model could recognise useful road components on its own. As can be seen in fig. 5.1, after 30 iterations the validation Mean Square Error (MSE) loss of the model shows a significant difference. The model is quite effective at mimicking the actions of a human driver.

4.3.2 CNN Model + Transfer Learning

Transfer learning[18] is a straightforward process that transfers the weights and information of a model that was trained on a large dataset to a smaller dataset. In order to employ the previously trained model in this model, we freeze the early convolution layers of the network and train only the final few layers to make predictions. But here, something goes wrong. Our deep neural networks' performance stagnates or worsens as we add more layers to them. This occurs as a result of the vanishing gradient issue[18]. Here, the model is trained using the Resnet50 model. The 50 layer Residual Network is called Resnet50.

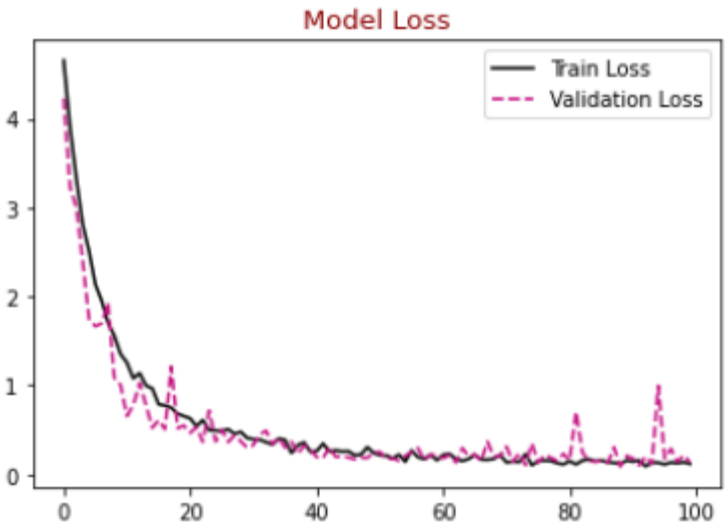

Figure 4.2(a) : Loss is plotted on the yl axis and Epochs in the x axis

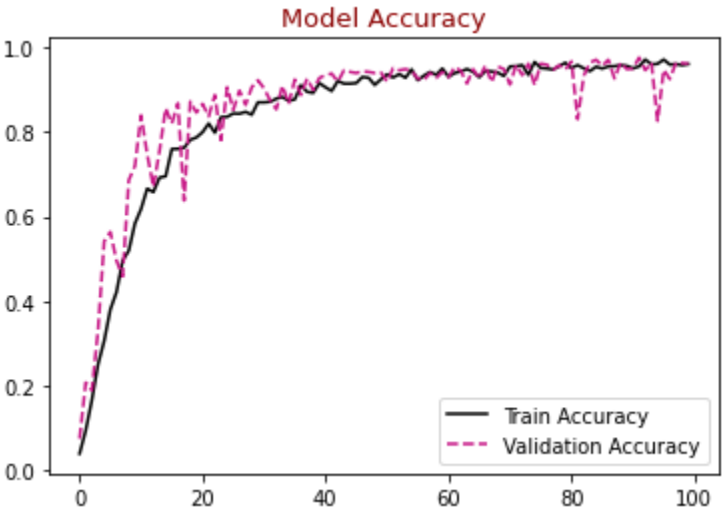

Figure 4.2(b) : Loss is plotted on the y axis and Epochs in the x axis

4.3.3 Comparison of Models

The loss curve will become static as epochs increase, as can be seen in the figure, as the initial loss is reduced. The Vanishing gradient issue is to blame for this. The vanishing gradients problem in a multilayer perceptron may manifest as a sluggish rate of model improvement during training and possibly as premature convergence, where additional training results in no further progress. If we looked at the weight changes during training, we would see more change (more learning) happening in the layers closer to the output layer and less change happening in the layers closer to the input layer. Due to this, we will choose the standard CNN model in this instance over the CNN+transfer learning approach (Figure 4.2).

4.3.4 Code Implementation

The simulator uses dashcam footage of a human-driven car that has been previously recorded to produce images from our dataset. Our dataset included these. These test movies are time-synchronized with the human driver's recorded steering commands. This shows that the CNN picked up on useful road elements all by itself, using merely the steering angle of a person as a training input. It has never been explicitly trained to recognise road outlines. The next figures, Figures 4.4, Figure 4.5, Figure4.6, and Figure 4.7, display the model's predictions.

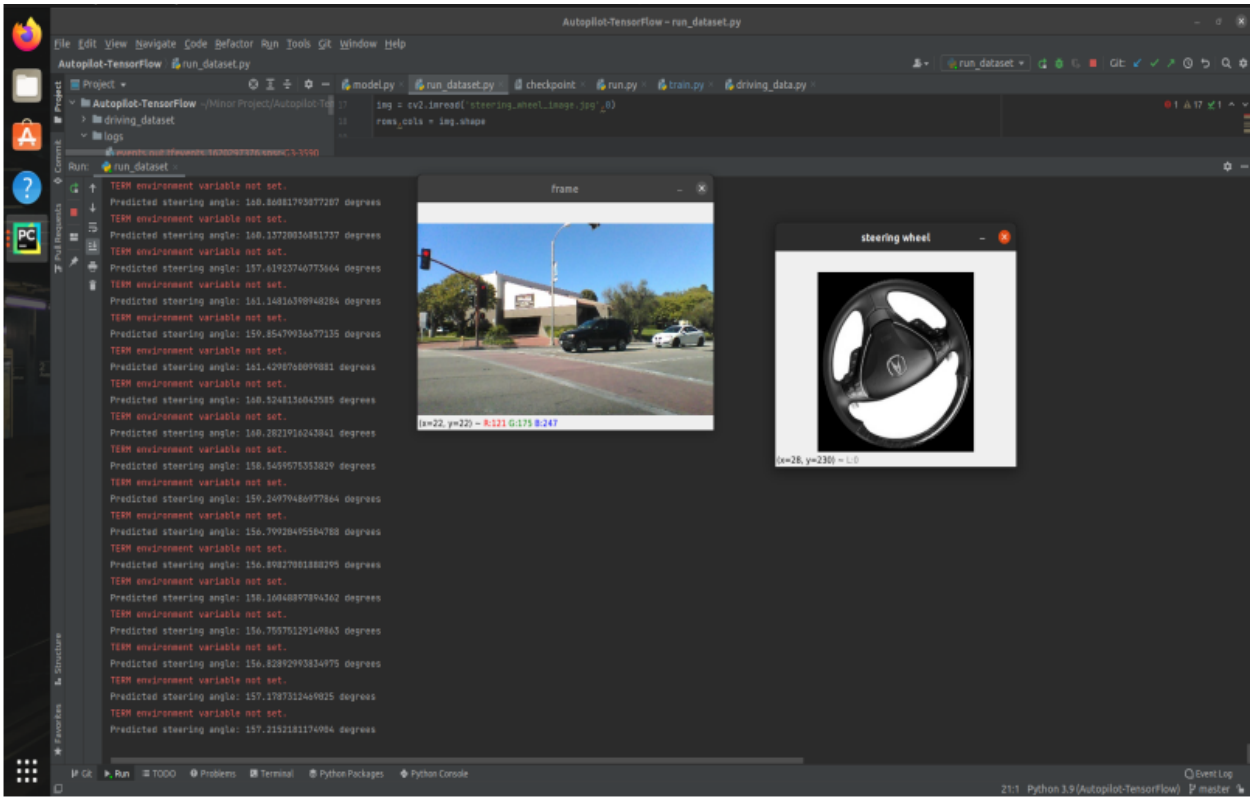

Figure 4.3: Model Prediction - 1

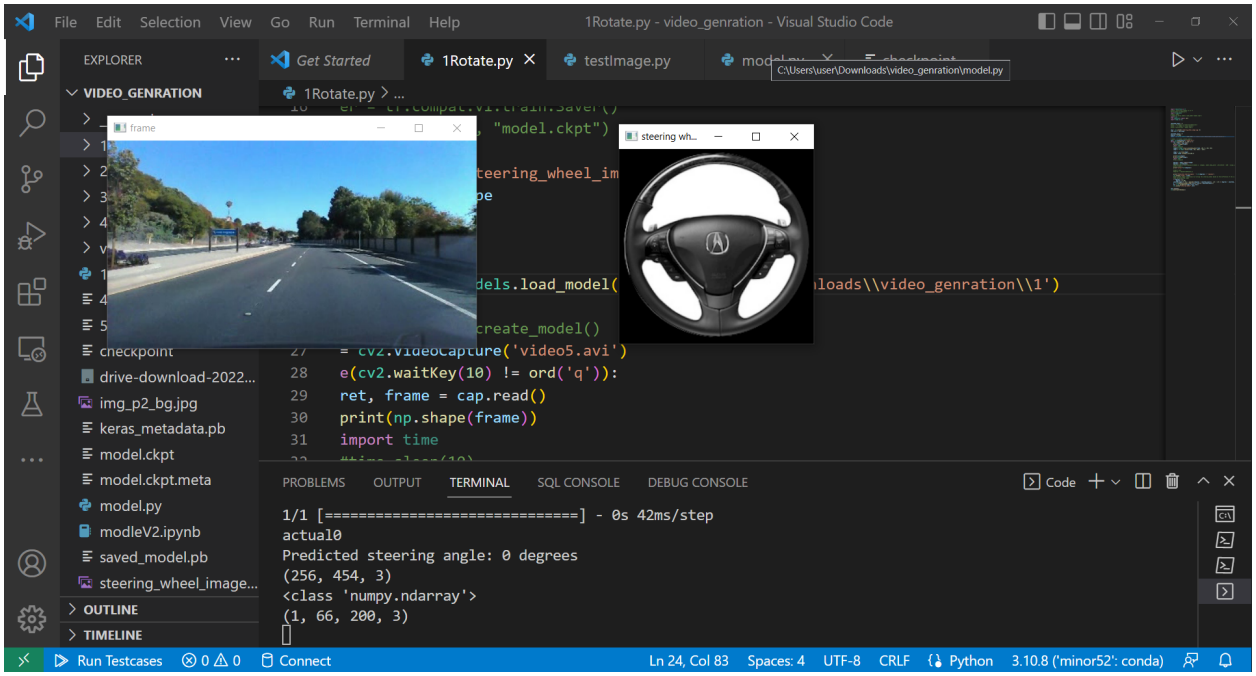

Figure 4.4: Model Prediction - 2

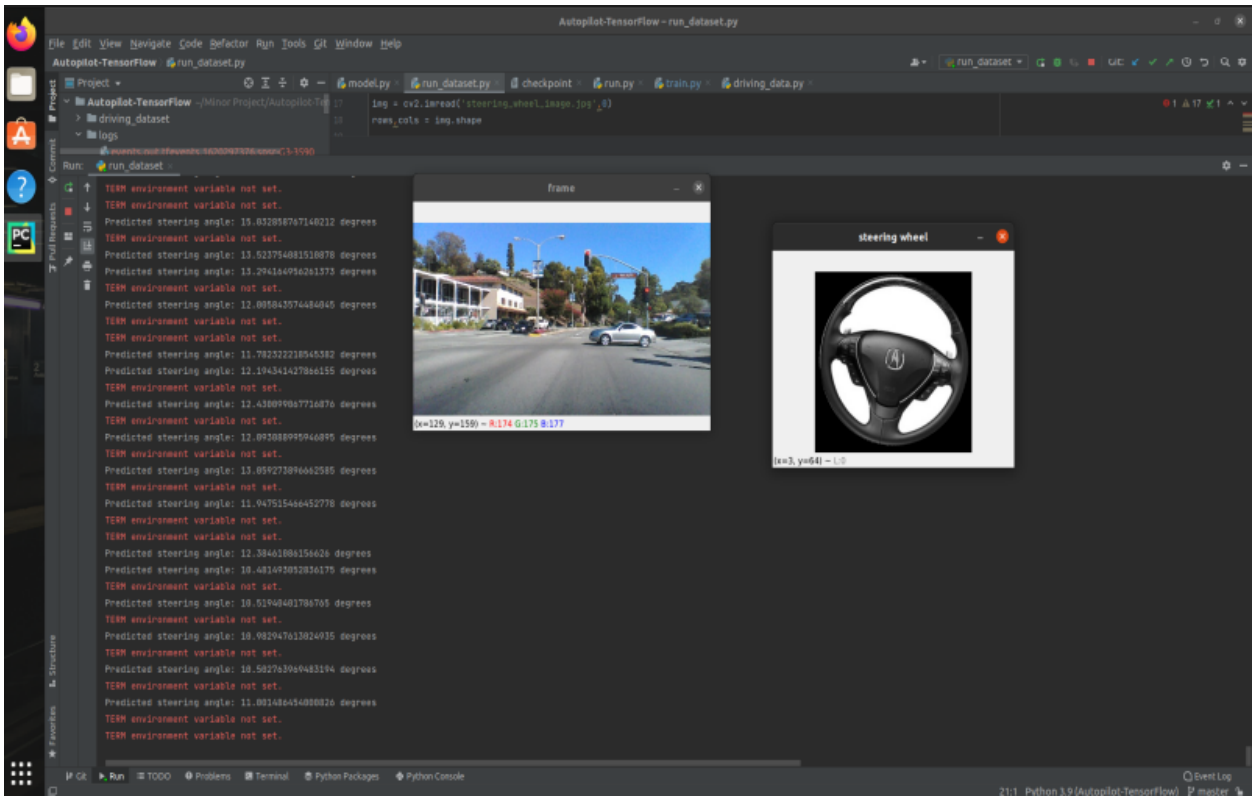

Figure 4.5: Model Prediction - 3

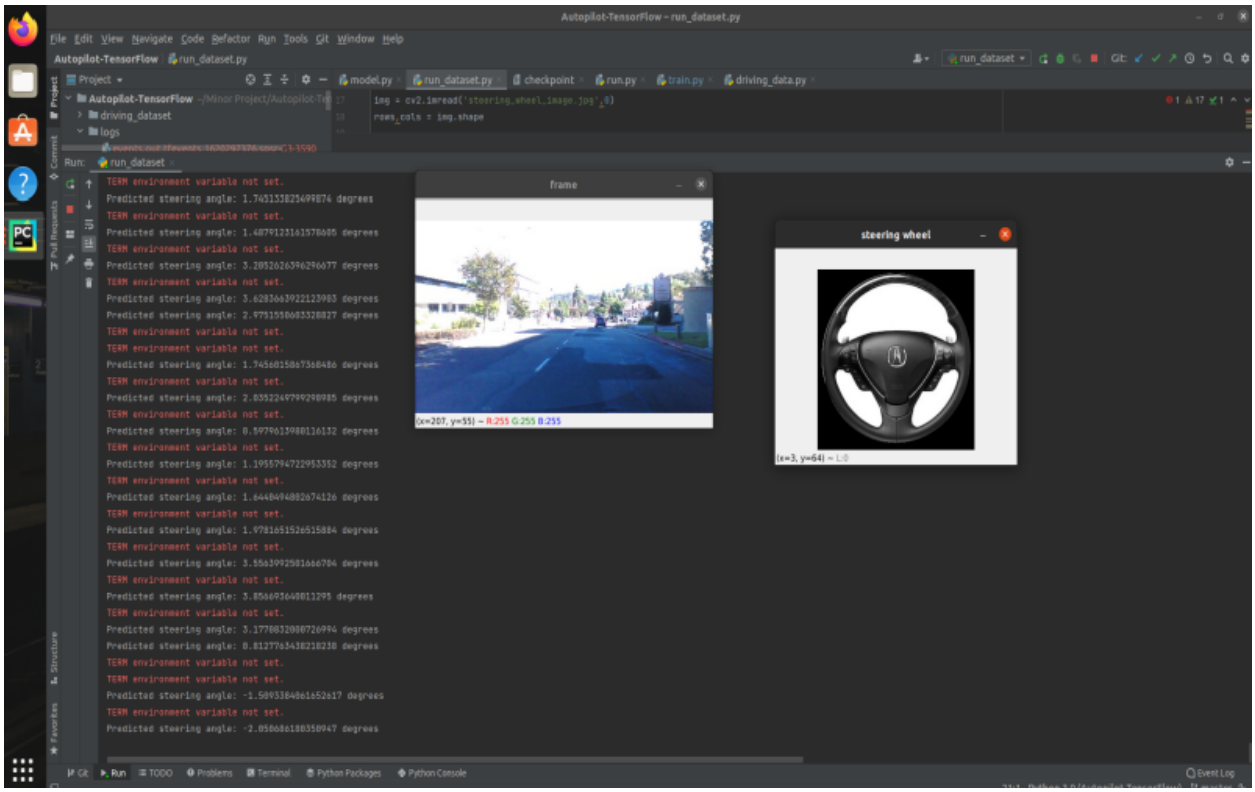

Figure 4.6: Model Prediction - 4

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

This study supports the notion that trained convolutional neural networks can be used to drive autonomous automobiles. This project demonstrated that CNNs is able to learn without being broken down into path planning, lane detection, semantic abstraction, and road identification or lane detection. From a sparse training signal, the CNN may learn about the characteristics of the road and the lanes.

In order to reduce the risky behavior of autonomous vehicles, this effort combined computer vision techniques with a CNN model. Setting up the environment took up a large portion of the implementation time. As a result of the results, we can conclude that deep learning is one of the most reliable methods for autonomous vehicles. As a result, we think that future projects would greatly benefit from the presentation of a framework that combines the advantages of computer vision-based methods with the computational power of the CNN model.

5.2 Future Scope

In order to make our model more reliable, we could incorporate more features like break and acceleration as well as more training data with various circumstances. Additionally, the dataset can be expanded using photographs with varying light levels. The accuracy of the model could be greatly boosted by using a hybrid model (CNN and RNN, for example) and could be significantly decreased by not using CNN in the physical car.

References

[1] Santokh Singh ,“Critical reasons for crashes investigated in the national motor vehicle crash causation survey,” Nat. Academies-Nat. Center Statist, 2015.

[2] M. Z. Islam, “Self-driving car: A step to the future of mobility,” BRACUniv., Dhaka, Bangladesh, Tech. Rep. 10361/11027, 2018.

[3] J. Woo, C. Yu, and N. Kim, “Deep reinforcement learning-based controller for path following of an unmanned surface vehicle,” Ocean Eng., vol. 183, pp. 155166, Jul. 2019.

[4] Denis Wolf , Carlos M. Massera ,Valdir Grassi Jr ,Fernando Osorio, “Longitudinal and lateral control for autonomous ground vehicles”,Conference: 2014 IEEE Intelligent Vehicles Symposium At: Dearborn, MI, USA Volume: 588 -593, June 2014.

[5] Mohamed A. A. Babiker, Mohamed A. O. Elawad,Azza H. M. Ahmed , ”Convolutional Neural Network for a Self-Driving Car in a Virtual Environment” International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE19), 2019.

[6] Agnihotri, Akhil Saraf, Prathamesh Bapnad, Kriti. “A Convolutional Neural Network Approach Towards Self-Driving Cars”, IEEE, 2019

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” arXiv:1604.07316, 2016.

[8] Jhung, I. Bae, J. Moon, T. Kim, J. Kim, and S. Kim, “End To-end steering controller with CNN-based closed-loop feedback for autonomous vehicles,” in Proc. IEEE Intell. Vehicles Symp. (IV), June. 2018, pp. 617–622.

[9] M. G. Bechtel, E. Mcellhiney, M. Kim, and H. Yun, “Deepika: A low cost deep neural network-based autonomous car,” in Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA), Aug. 2018, pp. 11–21.

[10] A. Wu, A. H. M. Rubaiyat, C. Anton, and H. Alemzadeh, “Model fusion:

Weighted N-version programming for resilient autonomous vehicle steering control," in Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISS CREW), Oct. 2018, pp. 144–145.

[11] M. Abe and W. Manning, "Steering system and vehicle dynamics," in Vehicle Handling Dynamics, M. Abe andW. Manning, Eds. Oxford, U.K.: ButterworthHeinemann, 2009, ch. 5, pp. 149164.

[12] TensorFlow https://www.tensorflow.org/ as accessed on 19-10-22

[13] Sully Chen, "SullyChen/driving-datasets: A collection of labeled car driving datasets" (github.com) url: https://github.com/SullyChen/driving-datasets, Sep 24 2021, as accessed on 14-10-22.

[14] A. V. Joshi, "Machine Learning and Artificial Intelligence". Cham, Switzerland: Springer, 2019.

[15] M. A. Wani, "Advances in Deep Learning, vol. 57". Cham, Switzerland: Springer, 2020.

[16] Tensorboard :https://www.tensorflow.org/tensorboard as accessed on 15-10-22

[17] Exacct Blog, "Activation Functions and Optimizers for Deep Learning Models" : https://www.exxactcorp.com/blog/Deep-Learning/activation-functions-and-optimizers-for-deep-learning-models, Nov 26, 2019, as accessed on 04-11-22

[18] Sunitha, Chunyan, "Vanishing gradient problem": https://en.wikipedia.org/wiki/Vanishing_gradient_problem, 17 Jan 2021 as accessed on 10-11-22