DRIVER MANEUVER PREDICTION VIA FEATURE FUSION USING DEEP LEARNING

A Project Report

Submitted by

Mr. SARATH MOHAN

REG NO : TKM21MEAI08

SEMESTER : IV

In partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY
IN
Mechanical Engineering (Artificial Intelligence)

Under the guidance of
Dr. ADARSH S

Thangal Kunju Musaliar College of Engineering
Kerala

MAY 2023

DECLARATION

I undersigned hereby declare that the project report "DRIVER MANEUVER PREDICTION VIA FEATURE FUSION USING DEEP LEARNING", submitted for partial fulfillment of the requirements for the award of degree of Master of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Dr. Adarsh S . This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Kollam
Date: SARATH MOHAN

Thangal Kunju Musaliar College of Engineering
Centre for Artificial Intelligence

C E R T I F I C A T E

This is to certify that, this report titled ***DRIVER MANEUVER PREDICTION VIA FEATURE FUSION USING DEEP LEARNING*** is a bonafide record of the **Project** presented by **SARATH MOHAN (TKM21MEAI08)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **M.Tech in Mechanical Engineering (Artificial Intelligence)** in **APJ Abdul Kalam Technological University** .

Project Guide	Project Coordinator	Head of the Department
Dr. Adarsh S Professor Department of Civil Engineering	Prof. Chinnu Jacob Assistant Professor Centre for AI	Dr. Imthias Ahamed T P Professor Centre for AI

Internal Examiner	External Examiner

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my project a successful one. I thank him for steering me to the shore of fulfillment under his protective wings.

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of TKM College of Engineering for giving me an opportunity to present my project. I would like to thank **Dr. Imthias Ahamed T P**, Professor and Head of the Department, Centre for Artificial Intelligence, for his constant support and encouragement throughout the project work.

With a profound sense of gratitude, I would like to express my heartfelt thanks to my guide, **Dr. Adarsh S**, Professor, Department of Civil Engineering, and my Project Coordinator, **Prof. Chinnu Jacob**, Assistant Professor, Centre for AI, for their valuable suggestions, guidance and immense encouragement. I am grateful to **Prof. Sumod Sundar**, Assistant Professor, Centre for AI, for his valuable feedback and patience, which helped me to complete this project to the best of my abilities. I would like to thank **Mr. Sreejith Pai P S**, Project Manager and **Mr. Jithin M**, Project Mentor, Tata Elxsi for their expert guidance and cooperation. I also extend my thanks to the entire faculty and staff members of the Centre for AI, TKM College of Engineering, Kollam, who have encouraged me throughout this work.

I also express my thanks to my loving parents, sister and friends, for their support and encouragement in the successful completion of this project work.

SARATH MOHAN

Abstract

The eighth most common cause of death and the top cause of death for people aged 5-29 are traffic accidents. Automated vehicles and driver-assistance systems have emerged as a promising alternative to lower the number of fatalities in traffic accidents and provide a safer and more effective transportation system. Accurate driver maneuver prediction in a dynamic traffic scene remains a difficult topic due to its complexity, despite the considerable attention of researchers and industry. Driver maneuver prediction is a technique used by advanced driving assistance systems (ADAS) systems to give the driver early warnings and help. For instance, the system can inform the user if the driver is about to make a lane change without indicating. This project deals with accurately predicting the maneuver of the driver with the help of a deep learning model that utilizes feature fusion from various data. The dataset is simulated from the Car Learning to Act simulator (CARLA) from which driver face video, road video, and data from 12 sensors of the car are logged at 20 Hertz. The model incorporates U-shaped encoder-decoder network architecture (UNET) trained with the DRI(EYE)VE dataset to gauge the points of attention of the driver which is then used to extract the points of interest from the road video. A face landmark model is also incorporated by the model to retrieve essential features from the driver face video. These three features are fused and are fed into an long short term memory (LSTM) model for contextual learning and finally, the maneuver at a certain time ahead is classified. Experimental results have shown that the feature fusion model obtained an accuracy of 80.19%, an overall precision of 87.93%, an overall recall of 87.95%, and an F1 score of 87.80%.

Contents

1 Introduction **1**
- 1.1 Objectives . . . 2
- 1.2 Organization of the report . . . 2

2 Literature Review **3**
- 2.1 Summary . . . 4
- 2.2 Research gap . . . 5

3 Theoretical Background of the Algorithms **6**
- 3.1 Long Short Term Memory . . . 6
- 3.2 Convolutional Neural Networks . . . 7
- 3.3 Feature Fusion Techniques . . . 8
- 3.4 Robot Operating System . . . 9
- 3.5 Driver eye tracking . . . 10

4 Methodology **12**
- 4.1 Car Learning to Act (CARLA) . . . 12
- 4.2 Transfer learning models . . . 14
  - 4.2.1 UNET Model . . . 14
  - 4.2.2 Facial Landmark Model . . . 15
  - 4.2.3 Proposed Methodology . . . 16

5 Implementation Framework **18**
- 5.1 Datasets Used . . . 18
  - 5.1.1 DR(EYE)VE Dataset . . . 18
  - 5.1.2 CARLA SIMULATED DATASET . . . 18
- 5.2 Model Pipeline . . . 23
- 5.3 Overall Architecture . . . 25

6 Results and Discussion **27**
- 6.1 Results of driver eye attention model . . . 27
- 6.2 Results of driver maneuver prediction models . . . 29

7 Conclusion **33**

List of Figures

3.1 Structure of an LSTM unit 6
3.2 A visual representation of a convolutional layer 7
3.3 Concatenation of vectors of varying modalities into a single vecto 8
3.4 Framework of ROS used to synchronize dataflow from CARLA simulator . . 10
3.5 Example of eye attention data obtained from eye tracking equipment 11

4.1 CARLA simulator system architecture pipeline 12
4.2 UNET Model 14
4.3 FaceNet Model 15
4.4 Proposed Methodology 16

5.1 The three different types of videos available in DR(eye)VE 18
5.2 CARLA Setup 19
5.3 Example of driver face video frame 20
5.4 Example of road video frame 20
5.5 Sample of the 12 signals obtained from CARLA 21
5.6 Class proportions in the entire dataset 22
5.7 Correlation heatmap of the dataset 22
5.8 Sliding window and look ahead technique 23
5.9 Model pipeline 24
5.10 Overall Architecture 26

6.1 Example of driver eye attention model prediction 27
6.2 Driver eye attention Model evaluation metrics graphs 28
6.3 Accuracy and loss graphs of driver face only model 29
6.4 Accuracy and loss graphs of road scenes only model 29
6.5 Accuracy and loss graphs of signal data only model 30
6.6 Accuracy and loss graphs of feature fusion model 30
6.7 Accuracies of the various models 31
6.8 Confusion matrix of feature fusion model 32

List of Tables

6.1 Performance parameters of driver eye attention model 28
6.2 Performance parameters of the various models 31

Chapter 1

Introduction

Road crashes are the 8th leading cause of death and the biggest killer of those aged 5-29[1]. To reduce road crash fatalities and have a safer and more efficient transportation system, automated vehicles and driver-assistance systems have become a promising solution. Despite the substantial attention of researchers and industry, accurate driver maneuver prediction in a dynamic traffic scene remains a challenging problem due to its complex nature. Driver maneuver prediction is an exciting field that has a lot of potential to improve road safety and traffic flow. By accurately predicting a driver's next move, we can identify potentially dangerous situations and alert the driver or autonomous vehicles to take preventive measures. This can significantly reduce the risk of accidents on the road.

Moreover, predicting driver maneuvers can also assist in developing better driver assistance systems that can provide proactive support to the driver and improve the overall driving experience. It can also help in planning efficient routes for autonomous vehicles or fleet management systems, resulting in reduced fuel consumption and better operational efficiency.

Real-time feedback provided to the driver based on the prediction can encourage safer and more efficient driving practices. Moreover, advanced driver assistance systems (ADAS) can use driver maneuver prediction to provide advanced warnings and assistance to the driver. For example, the system can detect if the driver is about to change lanes without signaling and provide an alert.

Predicting driver maneuvers can also enhance the features of various automotive technologies such as automatic gears, turn signals, anti-lock braking system, etc. Companies that manage large fleets of vehicles can also benefit from driver maneuver prediction by monitoring driver behavior and identifying potentially unsafe driving patterns. This can help in developing targeted driver training programs and reduce the risk of accidents.

Overall, driver maneuver prediction is a promising field that can revolutionize the way we drive and ensure a safer and more efficient future on the roads.

Although many strategies have been put out, there are still some research gaps that must be filled. The requirement for more varied and accurate datasets that can record a variety of driving scenarios, road conditions, and driver behaviour is one of the main problems. The creation and assessment of more reliable and accurate prediction models is hampered by the paucity of such datasets. In order to increase the precision of manoeuvre prediction, additional study is required on the integration of multimodal sensor data, such as camera, LiDAR, and radar. The creation of explicable models that can shed light on the prediction

system's decision-making process is another area that needs further research. This might improve the dependability and trustworthiness of automated driving systems. This work focuses on predicting the maneuver of the driver seconds before they make the choice in real time. Hence a novel method based on deep learning is proposed to infer the direction of the driver's maneuver seconds before he makes the prediction using the road scenes, driver face scenes and sensor readings from the vehicle.

1.1 Objectives

- To develop a deep learning model that predicts driver maneuvers from road video, driver face video and vehicle sensor data.
- To integrate the model in real-time ADAS system which would process the data and provide the necessary feedback to the vehicle's control system.

1.2 Organization of the report

The remainder of this report is organized into the following chapters: Chapter 2 reviews different traditional and deep learning techniques for driver maneuver detection. Chapter 3 includes the theoretical backgrounds of the various algorithms used in this project. Chapter 4 includes Methodology and details about various deep learning models used in the project. Detailed implementation framework described in Chapter 5. The experimental results are discussed in Chapter 6. Conclusions and possible future works of this study is presented in Chapter 7.

Chapter 2

Literature Review

In this section, several studies of driver maneuver prediction and trajectory prediction using deep learning and other techniques are discussed.

Ou et al.[1] proposed a recurrent neural network (RNN) algorithm that utilises both the road scene videos and driver face cam videos to make predictions about the next frames of the sequence. The RNN is able to fuse the information regarding driver observation actions and the driving environment. With new data labeling methods and effective sequential modelling approaches, the system is able to predict with high accuracy driving maneuvers shortly before the actual steering operations. The simulated database includes 700 maneuvers containing 274 lane changes, 131 turns, and 295 randomly sampled instances of going straight. Several cars were used for data collection in driving experiments, so the camera positions vary among samples. The dataset is annotated at the starting times when the cars touch lane marks (for lane change maneuvers) or cars start yaw (for turn maneuvers) and the ending times of steering actions.

Peng et al.[2] proposed a Driving Maneuver Early Detection (DMED) model that contains three major computational components, distance based representation of driving context, combined vehicle trajectory features and visual features, and a Long Short-Term Memory (LSTM)-based neural network that captures temporal dependencies of driving maneuvers. In addition to a general task, i.e. training a model to detect driving maneuvers based on partially observed evidence of maneuver events, the performance of the model was also evaluated on detecting driving maneuvers based on driving context observed ahead of the time of the driving maneuvers. The limitation of this work is that the performance of the proposed DMED model is not evaluated on different traffic scenarios respectively. Since the patterns of GPS noise and traffic objects could change significantly from one traffic scenario to another, it will be interesting to demonstrate the performance of the proposed model on different traffic scenarios.

Mahajan et al.[3] constructed a two-step prediction process from raw data observations: Initially, an unsupervised method is used for learning driving intentions in relation to lane keeping or lane changing, and the results are used for labeling the raw measurements. The labeled data then act as input for the prediction of lane driving intention using a deep learning model. The purpose is to develop a data-driven method, which is not concerned with manual labeling of data. This study bridges the gap in the literature by demonstrating that only with lateral movement data in relation to velocity and acceleration it is possible to distinguish whether a vehicle will carry out a lane change or not. The developed LSTM

model for predicting maneuvers over trajectories from different highway locations shows a significant performance and can detect lane change at least 3 s before the vehicle crosses the lane markings. The data recorded were obtained from a short highway segment, which consequently limits the number and nature of the identified maneuvers.

Martin et al.[4] developed a model that employs a machine vision-based framework to classify driver's gaze into context rich zones of interest and model driver's gaze behavior by representing gaze dynamics over a time period using gaze accumulation, glance duration, and glance frequencies. The model features three major aspects: one is the spatio-temporal features to represent the gaze dynamics, second is in defining the model as the average of the observed instances, third is in the design of the metric for estimating fitness of model. The spatio-temporal feature descriptor composed of gaze accumulation, glance duration and glance frequency are powerful tools to capture the essence of recurring driver gaze dynamics. The study shows that condensing gaze dynamics into durations and frequencies leads to recurring patterns based on driver activities. Furthermore, modelling these patterns show predictive powers in maneuver detection up to a few hundred milliseconds a priori.

Castignani et al.[5] developed a model utilizes sensor data obtained from smartphone placed in the vehicle. This data is used to create a multivariate normal representation using maximum likelihood estimator. The models trains and predicts in real time. Rather than detecting maneuvers based on fixed thresholds or supervised learning methods requiring labelled driving data, the system allows the driver style to be dynamically fitted in a multivariate Gaussian model that is frequently updated in order to adapt to changing driving conditions. The model is implemented and deployed in a mobile application and have collected driving traces from more than 4,800 users. The results confirm that the model adapts well to different road conditions and device types. The main advantage of such a system is that the model is computed for each individual mobile device, vehicle and driver, avoiding any dependency on a priori training data.

Khairdoost et al.[6] developed a deep learning method based on Long Short-Term Memory (LSTM) which utilizes data on the driver's gaze and head position as well as vehicle dynamics data. They applied this approach on real data collected during drives in an urban environment in an instrumented vehicle. In comparison with previous input output hidden markov model (IOHMM) techniques that predicted three maneuvers including left/right turns and driving straight, their prediction model is able to anticipate two more maneuvers.

2.1 Summary

The works mentioned propose different approaches for predicting driving maneuvers using various techniques such as recurrent neural networks, machine vision-based frameworks, and deep learning methods. Some of the proposed models utilize both road scene videos and driver face cam videos to make predictions, while others use sensor data obtained from smartphones or data on the driver's gaze and head position as well as vehicle dynamics data. The proposed models aim to detect driving maneuvers based on partially observed evidence of maneuver events or the driving context observed ahead of the time of the driving maneuvers. The main advantage of some models is that they adapt well to different road conditions and device types, while others show significant performance in predicting maneuvers up to a few hundred milliseconds prior.

2.2 Research gap

Overall, these studies show significant advancements in the development of deep learning-based models for maneuver detection in driving scenarios. However, there are still some research gaps that need to be addressed.

- Most of the studies have been evaluated on a limited set of driving scenarios, and there is a need to assess the generalization of the models to different traffic conditions and environments.
- The models proposed in these studies mainly rely on camera and sensor data, and there is a need to investigate the integration of other sources of information such as GPS, traffic information, and weather conditions.
- Most of the studies focus on detecting maneuvers such as lane changes and turns, and there is a need to explore the detection of more complex maneuvers such as merging and overtaking.
- There is a need to evaluate the real-world feasibility of these models, including their computational efficiency, scalability, and compatibility with existing driving assistance systems.

Chapter 3

Theoretical Background of the Algorithms

This chapter presents information on the background for the research presented in this thesis. It introduces the concepts and technologies relevant to the applications of driver maneuver prediction system. These include description of deep learning methods such as long short term memory (LSTM), convolutional neural networks (CNN), and feature fusion techniques. This is followed by explanation of data extraction and synchronization techniques used in advanced driver assistance systems (ADAS) such as robot operating system (ROS) and eye tracking technology.

3.1 Long Short Term Memory

Long Short-Term Memory (LSTM) is an artificial neural network architecture developed to tackle the shortcomings of Recurrent Neural Networks (RNNs) in processing sequential data. Hochreiter and Schmidhuber [7] first introduced LSTM in 1997, and since then, it has become a popular model for sequence prediction tasks such as speech recognition and natural language processing. The primary challenge with traditional RNNs is the vanishing gradient problem, which makes it difficult to learn long-term dependencies. LSTM networks overcome this problem by incorporating a memory cell that can selectively forget or update information over an extended period. Figure 3.1 illustrates an LSTM unit.

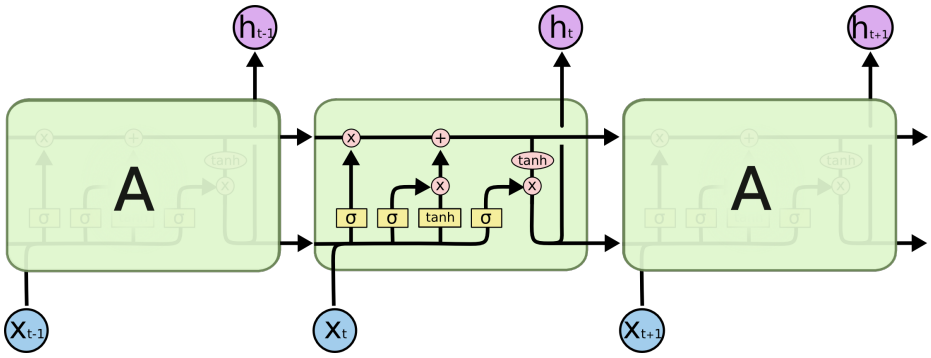


Figure 3.1: Structure of an LSTM unit [8]

An LSTM cell consists of three gates and a memory cell that control the flow of information. The forget gate determines which information to discard, the input gate decides what information to update, while the output gate determines what information to output. LSTM cells can be stacked to create a deep LSTM network that processes complex sequential data effectively. Training an LSTM network involves computing gradients of the loss function with respect to the network parameters using an optimization algorithm such as stochastic gradient descent. In conclusion, LSTM is a powerful neural network architecture for processing sequential data that has demonstrated success in various sequence prediction tasks and is widely used in industry and academia.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of artificial neural network that are commonly used for image recognition, computer vision, and natural language processing tasks. CNNs are inspired by the structure of the human visual cortex and use a combination of convolutional layers and pooling layers to learn hierarchical representations of input data.

The main building block of a CNN is the convolutional layer, which applies a set of filters to the input data to extract local features. Each filter is a small matrix that slides over the input data, performing element-wise multiplication and summation at each position to produce a feature map. By stacking multiple convolutional layers on top of each other, the network can learn increasingly abstract representations of the input data.

In addition to convolutional layers, CNNs also use pooling layers to downsample the feature maps and reduce the dimensionality of the input. Pooling can be performed using various methods, such as max pooling, which selects the maximum value within a window, or average pooling, which computes the average value within a window. Figure 3.2 illustrates an example of 2D convolution.

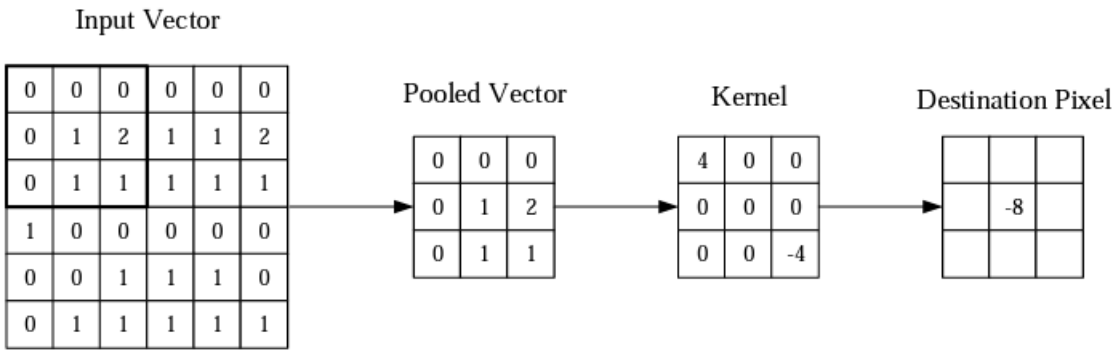


Figure 3.2: A visual representation of a convolutional layer [9]

One advantage of CNNs over other types of neural networks is their ability to learn translation-invariant features. This means that the same features can be detected regardless of their position within the image, making CNNs robust to small variations and transformations in the input data. Another advantage is their ability to learn spatial hierarchies of

features, which allows them to capture complex patterns and relationships in the input data.

Training a CNN involves computing the gradients of a loss function with respect to the network parameters and updating them using an optimization algorithm such as stochastic gradient descent. The gradients are computed using the backpropagation algorithm, which involves propagating the error through the network and updating the weights in the opposite direction of the gradient.

CNNs have achieved state-of-the-art results in many computer vision tasks, including image classification, object detection, and image segmentation. They have also been applied to natural language processing tasks such as sentiment analysis and text classification.

In conclusion, Convolutional Neural Networks (CNNs) are a powerful type of neural network that use convolutional layers and pooling layers to learn hierarchical representations of input data. CNNs have achieved significant success in a variety of computer vision and natural language processing tasks, and their ability to learn translation-invariant and spatially hierarchical features make them an important tool for many applications.

3.3 Feature Fusion Techniques

Feature fusion is the process of combining different types of features from multiple sources to enhance the performance of machine learning models[10]. Feature fusion techniques have been widely used in computer vision, speech recognition, and natural language processing tasks, where multiple modalities of input data are available, as shown in Figure 3.3.

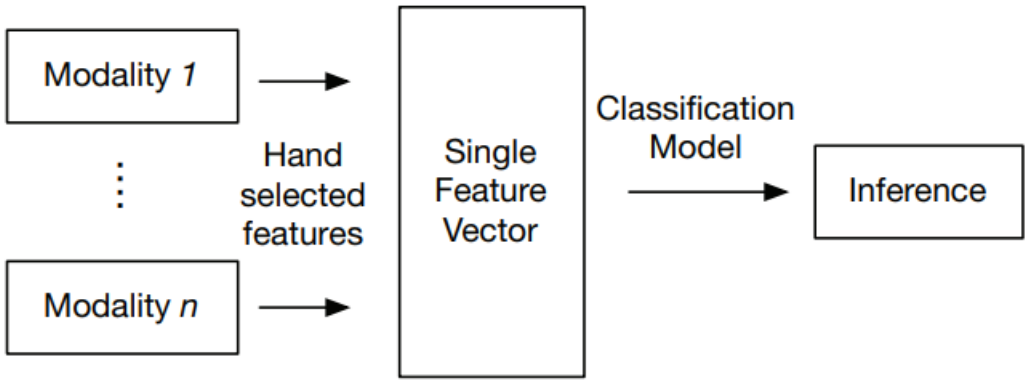


Figure 3.3: Concatenation of vectors of varying modalities into a single vector[11]

One common technique for feature fusion is early fusion, where the features from different sources are concatenated together and fed as input to the model. For example, in image recognition tasks, early fusion can be used to combine color and texture features from an image. This approach is straightforward and easy to implement, but it may not always be effective if the different feature types have different scales or distributions.

Another technique for feature fusion is late fusion, where the features from each source are processed separately, and the outputs are combined at a later stage. For example, in speech recognition tasks, late fusion can be used to combine features extracted from different

frequency bands. This approach allows for more flexibility in the modeling process and can better handle feature variations across sources.

A third technique for feature fusion is attention-based fusion, where the model learns to dynamically weight the contributions of each feature source based on their relevance to the task at hand. Attention mechanisms have been widely used in natural language processing tasks, where the model must attend to different parts of the input sequence to make accurate predictions.

Recently, deep learning techniques such as neural architecture search and transfer learning have also been used for feature fusion. Neural architecture search can automatically discover optimal architectures for combining features from different sources, while transfer learning can leverage pre-trained models to extract features from one modality and use them as input to another modality.

Feature fusion techniques have been successfully applied in a variety of tasks such as image recognition, speech recognition, and natural language processing. They can help to capture more informative representations of the input data, reduce the impact of noise and outliers, and improve the overall performance of machine learning models.

3.4 Robot Operating System

Robot Operating System (ROS) is an open-source robotics middleware suite that provides a set of tools and libraries to develop and operate robotic systems [12]. It was first developed by Willow Garage in 2007 and is now maintained by the Open Robotics organization. ROS has become one of the most widely used platforms for developing robotics applications and has a large and active community of developers.

At its core, ROS is a set of software libraries and tools that provide a communication infrastructure between different components of a robotic system. These components can be sensors, actuators, controllers, or any other module that makes up the robotic system. ROS provides a way for these components to communicate with each other, share data, and coordinate their actions.

Figure 3.4 illustrates the ROS bridge connections between CARLA and the database. One of the key features of ROS is its support for distributed computing. A ROS system can be composed of multiple nodes running on different machines, each responsible for a specific task. These nodes can communicate with each other using a publish-subscribe messaging system, where nodes can publish data to a topic and other nodes can subscribe to that topic to receive the data. This allows for a highly modular and flexible architecture, where different nodes can be added or removed from the system as needed.

ROS also provides a set of tools for visualization, simulation, and debugging of robotic systems. For example, RViz is a 3D visualization tool that allows users to visualize the sensor data, robot models, and other objects in the environment. Gazebo is a physics-based simulator that allows users to test their robotic systems in a virtual environment before deploying them on a physical robot.

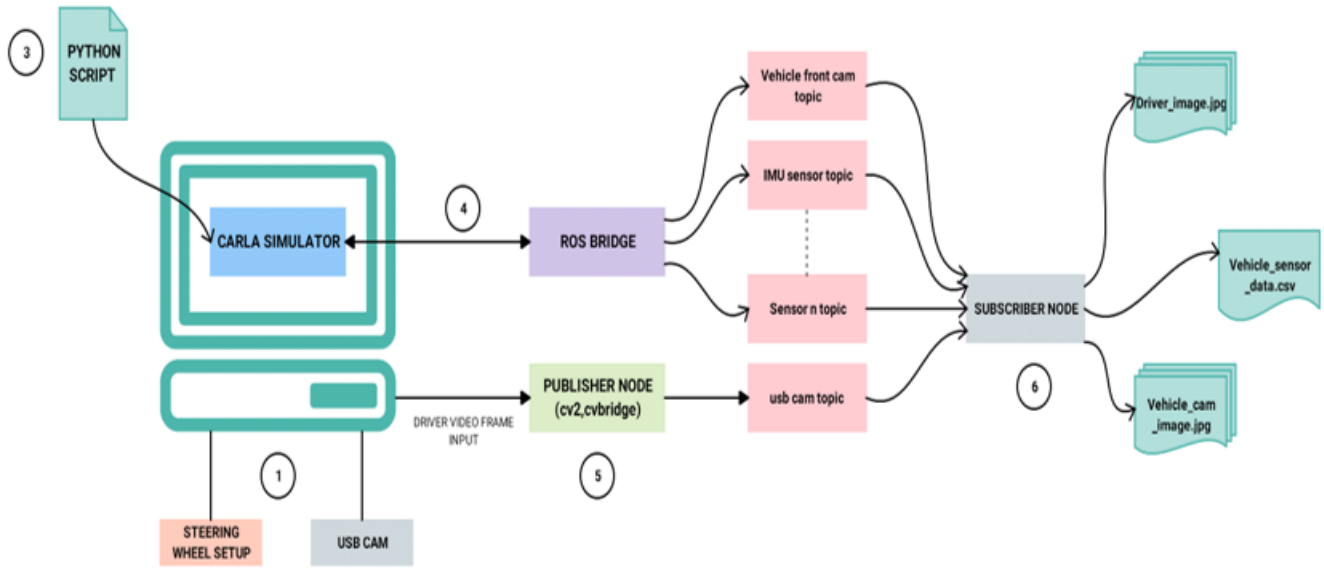


Figure 3.4: Framework of ROS used to synchronize dataflow from CARLA simulator[13]

ROS has been used in a wide range of robotics applications, from industrial automation to service robots and research platforms. It has become a de facto standard in many areas of robotics and has a large and active community of developers contributing to its development and maintenance. ROS also supports a wide range of programming languages, including C++, Python, and Java, making it accessible to developers with different backgrounds and expertise.

3.5 Driver eye tracking

Driver eye tracking is the process of monitoring a driver's eye movements and gaze behavior while driving a vehicle. This technology has the potential to enhance driver safety by detecting driver drowsiness, distraction, and other dangerous driving behaviors [14]. Eye tracking is a non-intrusive method of monitoring driver behavior, as it does not require the driver to wear any additional equipment or sensors.

Driver eye tracking systems use cameras and image processing techniques to detect and track the driver's eye movements and gaze behavior. These systems typically use infrared cameras that can operate in low-light conditions, allowing them to track eye movements even in the dark. The cameras capture images of the driver's eyes, which are then analyzed by the system to determine the direction of gaze and the duration of eye fixations.

One of the primary applications of driver eye tracking is detecting driver drowsiness. Drowsy driving is a major safety concern, as it can lead to accidents and fatalities. Driver eye tracking systems can detect signs of drowsiness, such as prolonged eye closures, changes in eye movements, and reduced blink rates. When drowsiness is detected, the system can issue alerts to the driver, such as audible warnings or vibrations, to alert the driver to take a

break or pull over. Figure 3.5 shows how the driver's eye attention heatmap is plotted over the road scene video frame.

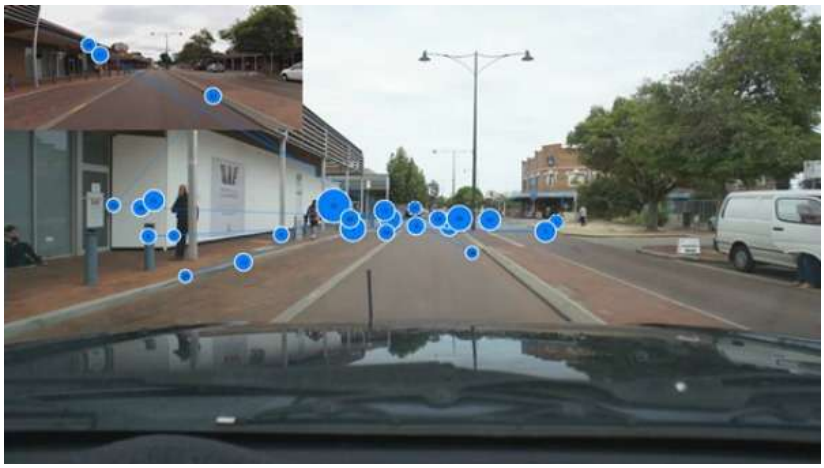


Figure 3.5: Example of eye attention data obtained from eye tracking equipment[13]

Driver eye tracking can also be used to detect driver distraction. Distraction is a common cause of accidents, and can occur when the driver is using a mobile device, adjusting the radio, or engaging in other non-driving activities. Driver eye tracking systems can detect when the driver's gaze is directed away from the road for an extended period of time, and issue alerts to encourage the driver to focus on the road.

Another application of driver eye tracking is monitoring driver behavior in autonomous vehicles. As autonomous vehicles become more prevalent, it will become increasingly important to monitor the driver's attention and readiness to take control of the vehicle. Driver eye tracking systems can detect when the driver's attention is not focused on the road, and issue alerts to prompt the driver to take control of the vehicle.

Driver eye tracking has the potential to improve driver safety and reduce accidents on the road. However, there are also privacy concerns associated with the use of this technology. It is important to ensure that driver eye tracking systems are used responsibly and with appropriate safeguards in place to protect the privacy of drivers.

Chapter 4

Methodology

This chapter describes the detailed methodology of the driver maneuver prediction technique.

4.1 Car Learning to Act (CARLA)

CARLA (Car Learning to Act) is an open-source simulator for autonomous driving research that was developed by the Robotics and Perception Group at the University of Torino, Italy. It provides a high-fidelity, realistic environment for testing and developing autonomous driving systems, allowing researchers to validate their algorithms in a safe and controlled environment before deploying them on real-world vehicles [15].

One of the key features of CARLA is its high level of realism, which is achieved through the use of detailed 3D graphics, realistic physics simulations, and accurate representations of the road network and surrounding environment. This allows researchers to test their algorithms in a variety of scenarios, including urban and rural environments, highways, and even off-road terrain. Figure 4.1 represents the overall layout of the CARLA simulator.

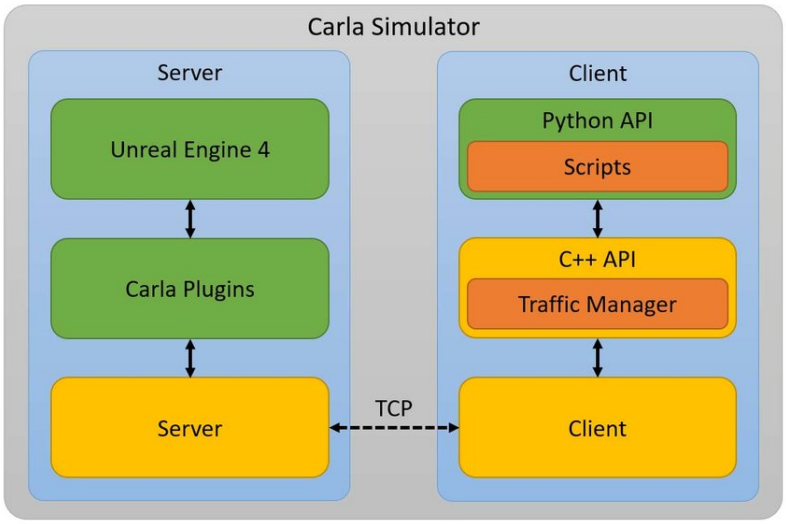


Figure 4.1: CARLA simulator system architecture pipeline [16]

CARLA also provides a flexible and extensible platform for developing and testing a wide range of autonomous driving algorithms, from perception and localization to planning and control. It supports a variety of sensor modalities, including lidar, radar, and cameras, as well as communication protocols such as ROS and MQTT.

In addition to its simulation capabilities, CARLA also provides a suite of tools for data analysis and visualization, including a built-in recorder for collecting sensor data, a replay system for replaying recorded scenarios, and a Python API for interacting with the simulator and analyzing data.

The CARLA simulator has been used in a wide range of research projects, including perception and control algorithms for autonomous vehicles, pedestrian detection and tracking, and even urban planning and traffic management. Its high level of realism and flexibility make it a valuable tool for both academic and industry researchers in the field of autonomous driving.

Some advantages of using CARLA simulator to other simulators are:

- Open-source: CARLA is an open-source simulation platform that allows researchers and developers to customize and extend the simulator to fit their specific needs.
- High-fidelity simulation: CARLA offers a high-fidelity simulation environment, with accurate physics models and realistic graphics, which enables researchers to test and evaluate their algorithms in a realistic environment.
- Multi-agent simulation: CARLA supports multi-agent simulation, which allows multiple agents to operate and interact with each other in the same environment. This feature is particularly useful for developing and testing autonomous driving algorithms.
- Large-scale simulation: CARLA can simulate large-scale environments, which enables researchers to evaluate the scalability of their algorithms in real-world scenarios.
- Modular architecture: CARLA has a modular architecture, which means that researchers and developers can easily add and remove components of the simulator to suit their specific needs.
- Python API: CARLA provides a Python API, which makes it easy for researchers and developers to interact with the simulator and develop their algorithms using the Python programming language.

4.2 Transfer learning models

4.2.1 UNET Model

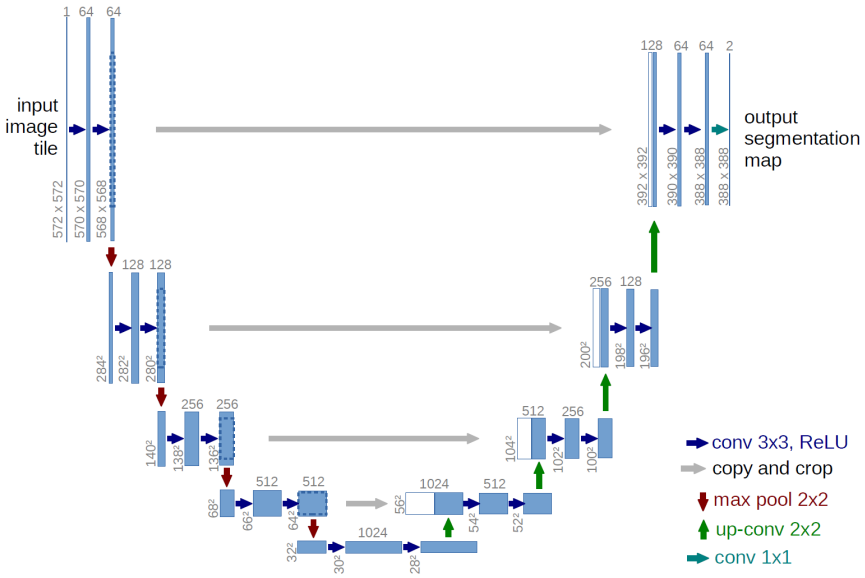


Figure 4.2: UNET Model [17]

The U-Net model is a convolutional neural network architecture that is commonly used for image segmentation tasks[6]. As shown in Figure 4.2, the architecture comprises of a symmetric expanding path that permits exact localisation and a contracting path to capture context. The U-Net model has been shown to achieve high accuracy in various image segmentation tasks, including biomedical image segmentation and segmentation of natural images. The U-Net model is designed to work with small amounts of training data, which makes it suitable for tasks where annotated training data is limited. The U-Net model can handle images of different resolutions, which makes it useful for tasks where images of different sizes need to be segmented.

The UNET architecture has several advantages over other models, including:

- Better performance for semantic segmentation: The UNET architecture is specifically designed for semantic segmentation tasks, making it more efficient and accurate than other models that are not specifically designed for this task.
- Fewer parameters: UNET has fewer parameters than other popular architectures such as VGG-16 and ResNet, which means that it can be trained more quickly and with less memory usage.
- More efficient memory usage: The UNET architecture utilizes skip connections to improve the memory efficiency of the model by reducing the number of layers needed for accurate segmentation.

4.2.2 Facial Landmark Model

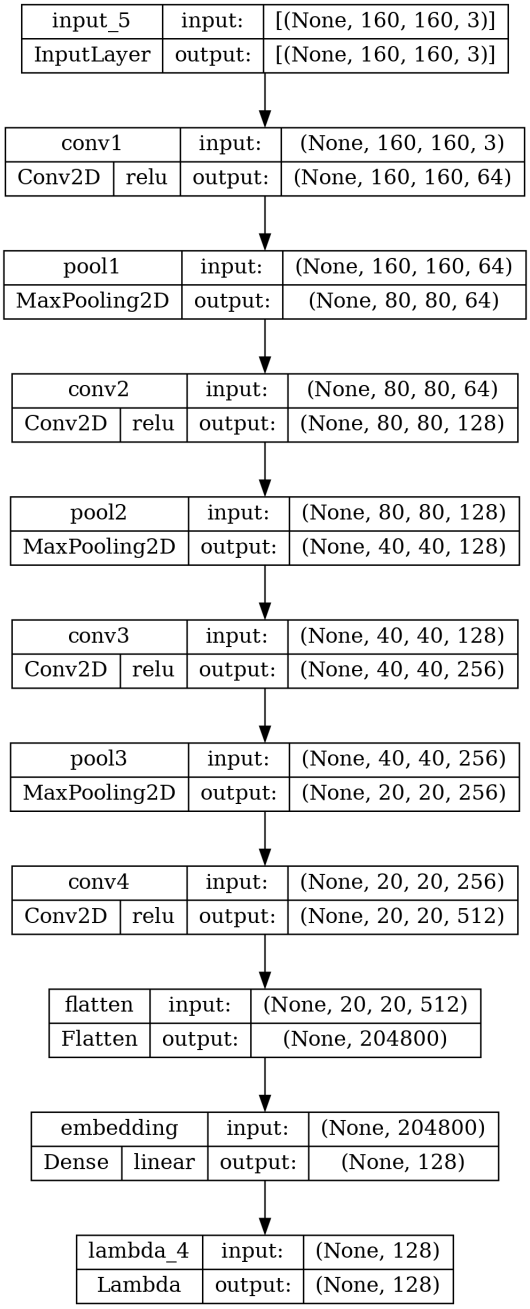


Figure 4.3: FaceNet Model [18]

FaceNet is a deep learning model developed by Google researchers for face recognition tasks. The model is based on a deep convolutional neural network architecture that learns to map facial features into a high-dimensional space. It is designed to generate a compact and meaningful embedding of a face image in a high-dimensional feature space. This embedding can be used to compare two face images and determine whether they belong to the same person

or not.

As shown in Figure 4.3, the FaceNet model uses a convolutional neural network (CNN) to extract features from a face image, followed by a triplet loss function that encourages similar faces to be closer together in the feature space and dissimilar faces to be farther apart. It takes three face images as input: an anchor image, a positive image (belonging to the same person as the anchor), and a negative image (belonging to a different person). The loss function encourages the distance between the anchor and positive embeddings to be smaller than the distance between the anchor and negative embeddings by a margin. This helps the model learn discriminative features that are useful for face recognition.

One of the key innovations of FaceNet is the use of a siamese network architecture, which allows the model to compare two face images by computing the distance between their embeddings in the feature space. This makes the model robust to variations in lighting, pose, and facial expression [19].

FaceNet achieves state-of-the art performance on face recognition benchmarks such as Labeled Faces in the Wild(LFW) and YouTube Faces(YTF), with accuracies of over 99% and 95%, respectively and has been widely adopted in various applications, including security systems, access control, and social media.

4.2.3 Proposed Methodology

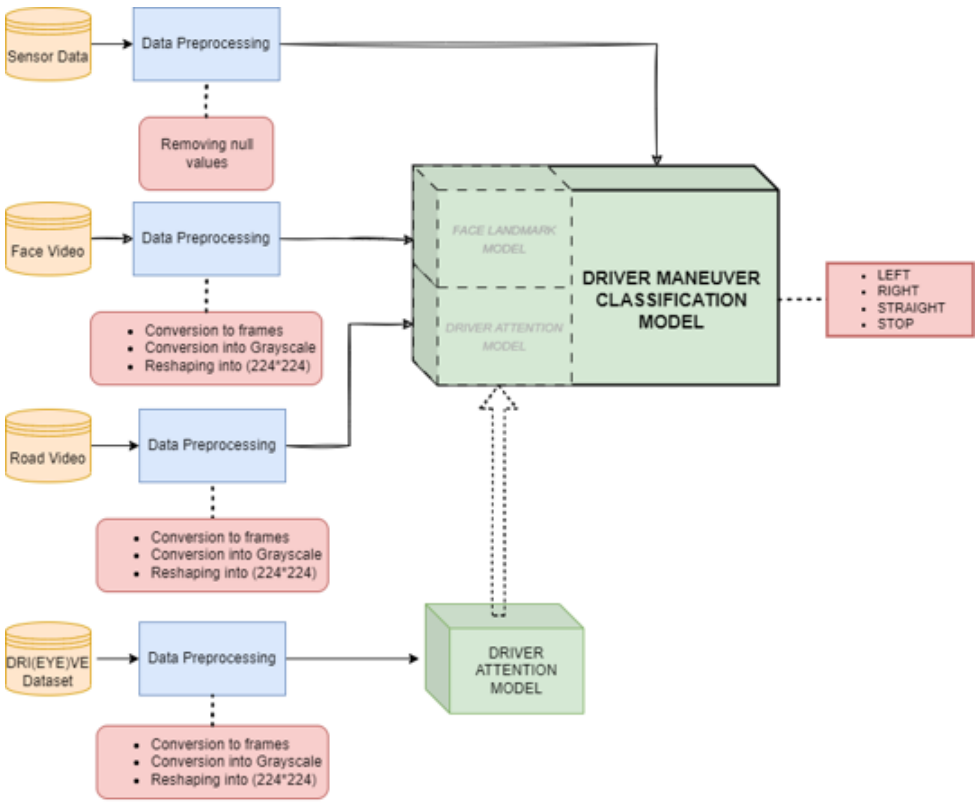


Figure 4.4: Proposed Methodology

The proposed system utilized multimodal driving signals (GPS, vehicle speed, heading, etc), video of the road scene ahead and video of driver's face, to understand the driving environment and learn the causal relation between driving environment and driving maneuver using deep learning algorithms.

Figure 4.4 presents an overview of the proposed DMD system. The DMD system is comprised of two computational modules, driving maneuvering feature extraction module and the driving maneuvering detection module. The driving maneuvering feature extraction module contains deep learning models for generating attention feature maps and driver face shift feature extraction. It generates a vector of the driver's attention features, driver's face shift features and vehicle signals, which is used as input to LSTM model.

The driving maneuver detection module uses a network of three LSTM layers to learn maneuver information from a sequence of feature vectors generated by the feature generation module in the time domain. Attention model and LSTM layers are updated with optimization method in training process [20].

Overall, the proposed DMD system offers a comprehensive framework for identifying driving manoeuvres through the use of multimodal data sources and deep learning algorithms. The system can successfully forecast driving manoeuvres in real-time by utilising the temporal relationships between driving signals and camera data, making it a viable tool for advanced driver assistance systems (ADAS) and autonomous vehicles.

Chapter 5

Implementation Framework

5.1 Datasets Used

5.1.1 DR(EYE)VE Dataset

DR(eye)VE is a large dataset of driving scenes for which eye-tracking annotations are available [21]. This dataset features more than 500,000 registered frames, matching ego-centric views (from glasses worn by drivers) and car-centric views (from roof-mounted camera). Using a precise eye tracking gadget, they captured the driver's gaze. We overlay gaze information on an HD video taken from a roof-mounted camera to favour the car's point of view. Eight subjects took it in turns to drive through various landscapes in order to rule out any potential biases. They carefully picked a subset of brief events in which the driver diverted his attention from the road to something else in the environment. Figure 5.1 shows an example of the DR(EYE)VE dataset.

Figure 5.1: The three different types of videos available in DR(eye)VE

The eye gaze annotations are particularly important for our task of driver maneuver prediction, as they provide a precise location of where the driver is looking at any given moment. This can be used to infer the driver's attention and intentions. One of the challenges of working with this dataset is the large size of the videos and the corresponding annotations. The dataset requires significant storage space and computational resources to process and analyze. Additionally, the eye gaze annotations are quite dense, with a total of over 400,000 annotations across the dataset. This requires careful preprocessing and filtering to ensure high quality data for training and evaluation.

5.1.2 CARLA SIMULATED DATASET

The open-source CARLA (Car Learning to Act) simulator is used in research on autonomous vehicles [22]. It is currently being maintained by the team at the Italian Institute of Tech-

nology (IIT), but it was originally created by the Computer Vision Centre (CVC) at the Universitat Autnoma de Barcelona (UAB) in Spain.

CARLA offers a platform to replicate a range of traffic situations, weather conditions, urban, suburban, and rural landscapes, and driving scenarios. Additionally, it provides an extensive selection of sensor simulators, including as cameras, lidars, radars, and GPS, that let academics test their planning, perception, and control algorithms.

In order to facilitate the creation, instruction, and validation of autonomous driving systems, CARLA has been built from the ground up. CARLA offers open digital assets (urban layouts, buildings, and vehicles) in addition to open-source technology and protocols that were developed for this purpose. The simulation platform offers customizable climatic conditions, full control over all static and dynamic characters, the creation of maps, and many other features. Currently about 60K datapoints are collected. The setup of the CARLA simulator used for this project is shown in Figure 5.2. For the dataset simulated for the project:

- Hardware configuration, includes calibrating the steering wheel, setting up a USB camera, and configuring a triple display setup.
- An automobile was spawned in the Carla simulator, equipped with the required sensors, and a python script was used to provide steering wheel control.
- The sensor data is saved as CSV file.
- Road images and webcam images are saved in JPG formats.
- Data is collected at the rate of 20 Hertz.

Figure 5.2: CARLA Setup

By offering a high-fidelity simulation environment that may assist academics and engineers in iterating and testing their algorithms in a secure and controlled environment,

CARLA promotes the development of autonomous driving algorithms. The software is constructed on top of the Unreal Engine [23], which renders the virtual environment realistically.

Researchers and developers can communicate programmatically with the simulation environment using CARLA's Python API. This enables the development of unique tools and scripts for data collecting, visualisation, and analysis. Figures 5.3 and 5.4 show the driver face camera output and the road video output respectively.

Figure 5.3: Example of driver face video frame

Figure 5.4: Example of road video frame

The 12 features logged from CARLA simulator are:

- Velocity
- Acceleration_linear_x
- Acceleration_linear_y
- Acceleration_linear_z
- Orientation_x
- Orientation_y
- Orientation_z
- Orientation_w
- Throttle
- Steer
- Brake
- Gear

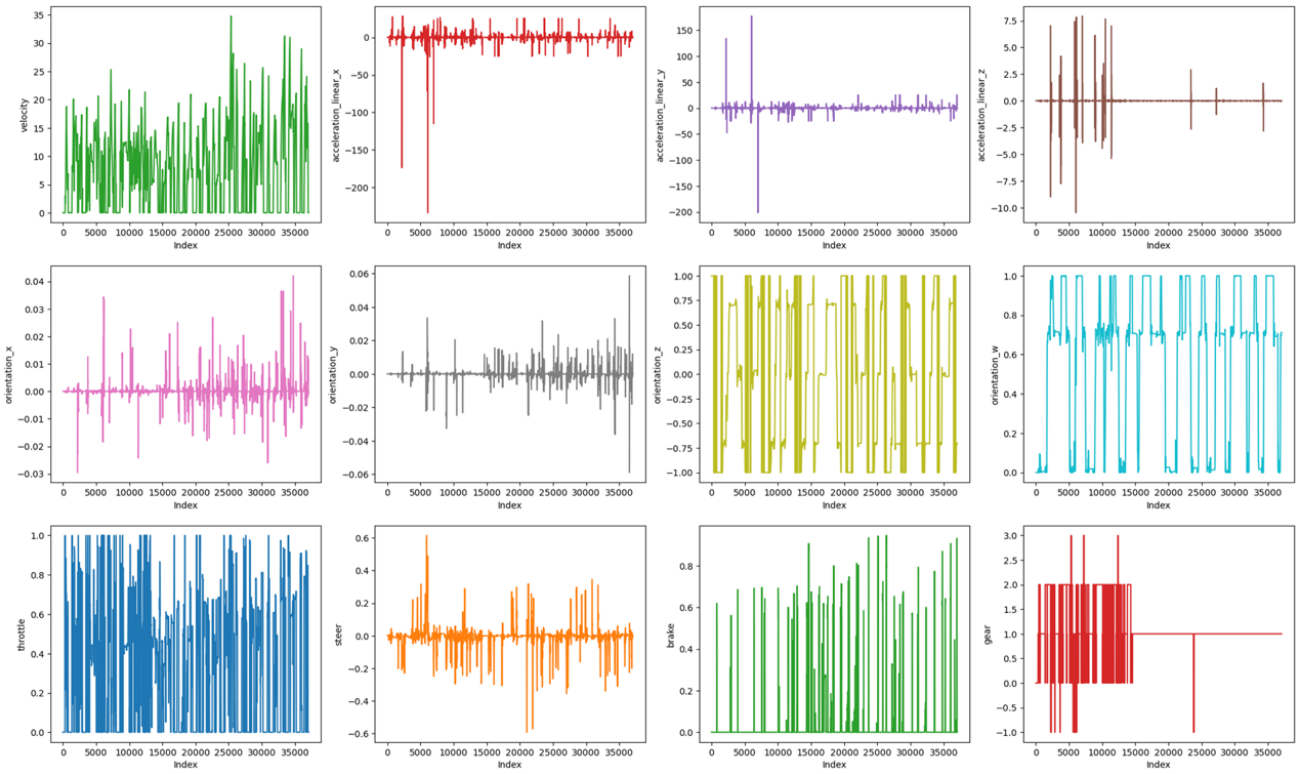


Figure 5.5: Sample of the 12 signals obtained from CARLA

These features are sensitive to even minute adjustments in driving style or vehicle behaviour. This sensitivity can be helpful for spotting minute behavioural changes or spotting potentially unsafe circumstances [24]. These characteristics are closely tied to how the vehicle moves and is oriented, as well as how the driver controls it. They are therefore extremely pertinent to any modelling work involving driver or vehicle behaviour. The sensors that detect these characteristics are already present in many vehicles, making it extremely simple to gather the data required for modelling. As they are based on actual data and driver inputs, the features are also rather easy to interpret. This can make it easier to understand the results of a modeling task and make decisions based on those results. The various signals recorded in the dataset are illustrated in Figure 5.5. and the overall distribution of the classes is illustrated in Figure 5.6.

Four classes are considered for implementation:

- Straight
- Left
- Right
- Stop

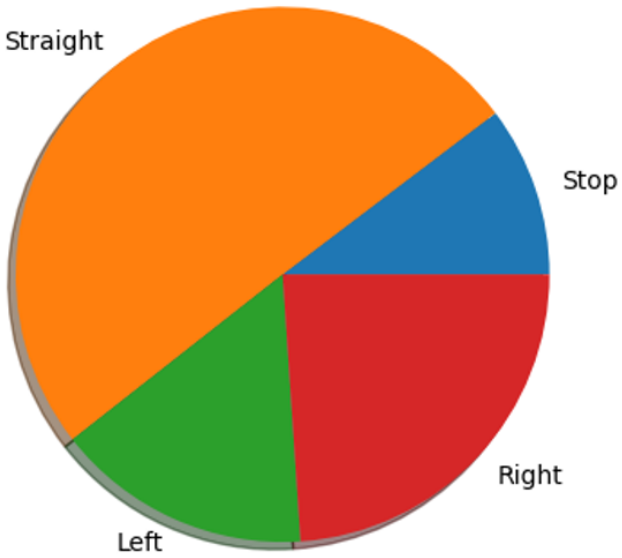


Figure 5.6: Class proportions in the entire dataset

Figure 5.7 shows the correlation heatmap of the dataset.

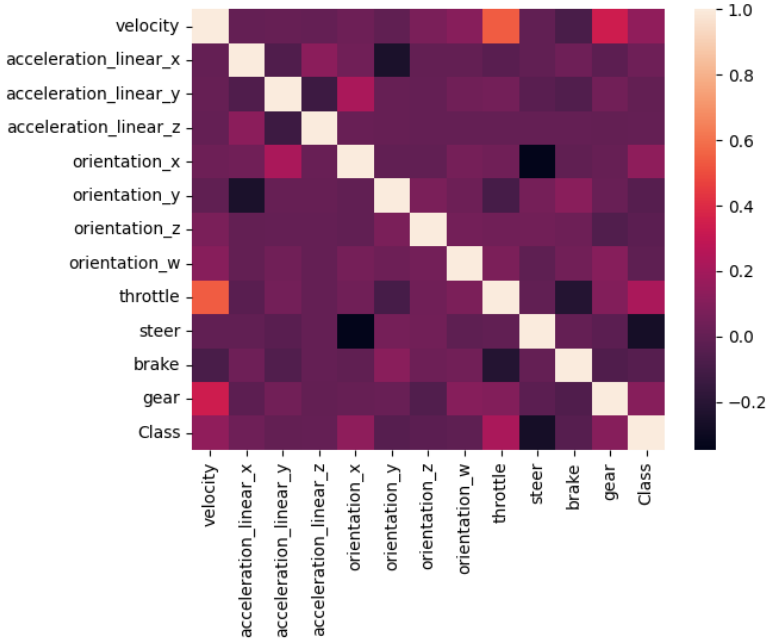


Figure 5.7: Correlation heatmap of the dataset

5.2 Model Pipeline

The prediction of driver's maneuver is done five seconds priori, therefore there is a 100 frame difference between current frame and frame for which the maneuver class is predicted. Figure 5.8 illustrates the sliding window and look ahead technique.

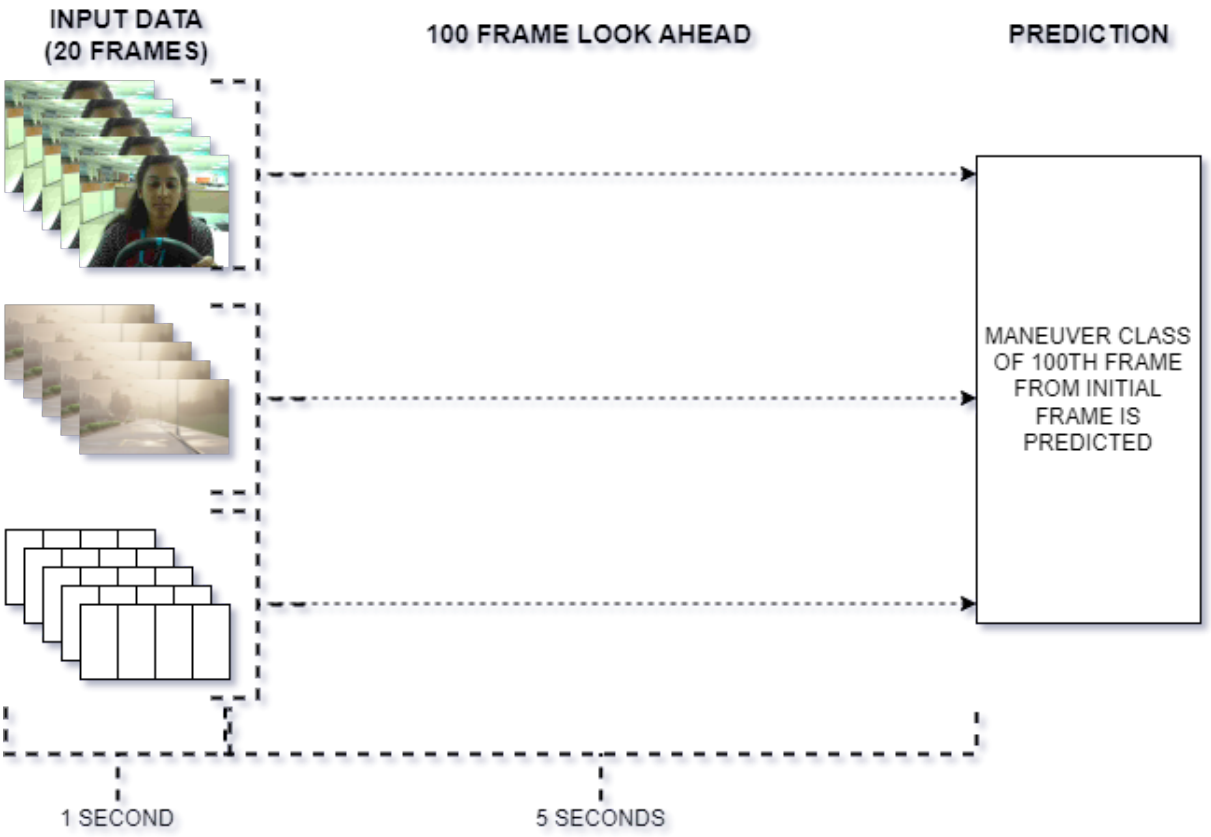


Figure 5.8: Sliding window and look ahead technique

Sliding window technique involves dividing a longer sequence of data into shorter overlapping segments (in this case, 20-frame segments), and applying a machine learning model to each segment to make a prediction [25]. The model is fed the data of a single second (20 frames) and class of the 100th frame from the current frame is predicted, which results in a five second look ahead. The 20th frame of the 20 frame sequence is the current frame. The ROS bridge is used to synchronize the 20 frames of driver face video data, road video data, and timesteps of sensor value data to be input to the model.

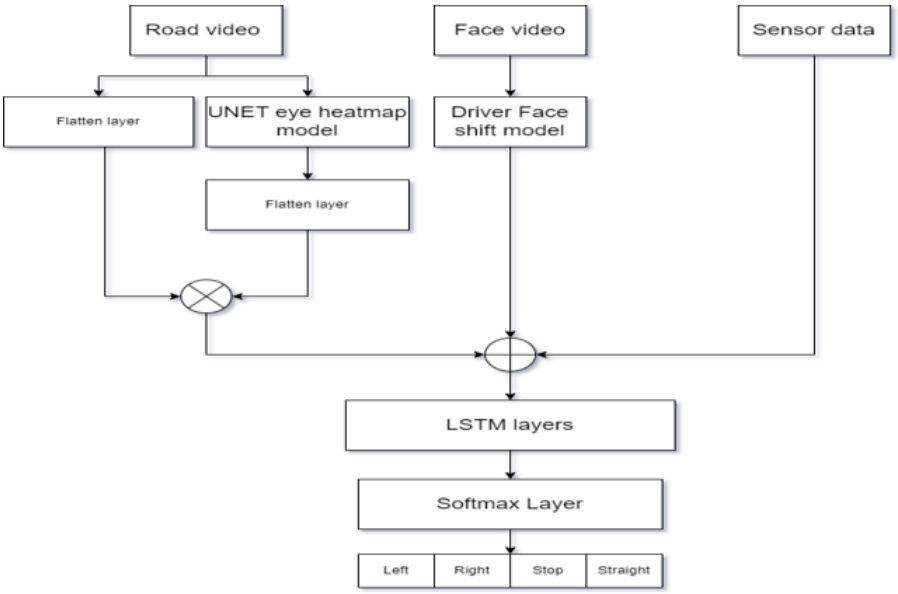


Figure 5.9: Model pipeline

The Figure 5.9 illustrates the flow of data in the real-time scenario. The data flows through this architecture as follows:

- The input data is a set of three types of features: image frames, time-series sensor data, and facial embeddings. These are provided as three separate inputs to the model.
- The twenty image frames are passed through a U-Net model to generate a set of segmentation masks. The U-Net model is applied at each time step separately as the input frames are in a sequence.
- The segmentation masks are multiplied element-wise with the original input frames to generate a set of masked frames.
- The masked frames are then passed through a convolutional layer to reduce the channel dimension from 3 to 1.
- The resulting feature maps are reshaped into a 3D tensor, where the first dimension corresponds to the time steps, the second dimension is a flattened representation of the feature maps, and the third dimension corresponds to the number of channels.
- The twenty time step time-series sensor data is passed through a 1D convolutional layer to reduce the number of features.
- The twenty face image frames are passed through a TimeDistributed layer that applies a pre-trained FaceNet model to each frame of the input sequence.
- The output of the U-Net model, the time-series sensor data, and the facial embeddings are concatenated along the channel dimension.

- The concatenated tensor is passed through a series of LSTM layers to generate a sequence of feature vectors that capture the temporal dynamics of the input data.
- Finally, a fully connected layer is applied to the last feature vector to generate the output predictions. The output layer has a softmax activation function to generate class probabilities of the frame five seconds ahead of time.

5.3 Overall Architecture

The input consists of three components: a sequence of frames with the shape (timestep, height, width, channels), a sequence of features with the shape (timestep, features), and a sequence of face images with the shape (timestep, 160, 160, channels). The model then applies a series of convolutional layers, followed by a multiplication operation between the U-Net model and the first input, and a reshape operation.

Next, the model applies a 1D convolution to the sequence of features, and concatenates the resulting tensor with the reshaped tensor from the previous step and the output of the FaceNet model applied to the sequence of face images.

Finally, the model applies a series of LSTM layers, followed by a Dense layer, and outputs a softmax classification over the four of classes.

Some potential advantages of this model architecture are:

- Incorporates information from multiple modalities: The model employs data from three separate sources, including facial images, feature vectors, and image frames, to determine a classification. Compared to models that just employ one modality, this might increase classification accuracy [26].
- Utilizes two established models: The architecture combines components of the U-Net and FaceNet models, both of which are widely used and have shown strong performance in their respective domains. By building on these established models, this architecture may be more likely to achieve strong performance in its target task.
- Flexibility: This model can be adapted to a variety of video classification tasks involving faces. By modifying the number of LSTM layers or changing the number of classes in the output layer, the model can be tailored to different use cases.
- Generalization: The model architecture can potentially generalize well to new video data with different characteristics, as the use of multiple modalities and the incorporation of pre-trained models may help the model to learn more robust and generalizable features.

The overall architecture of the model is illustrated in Figure 5.10

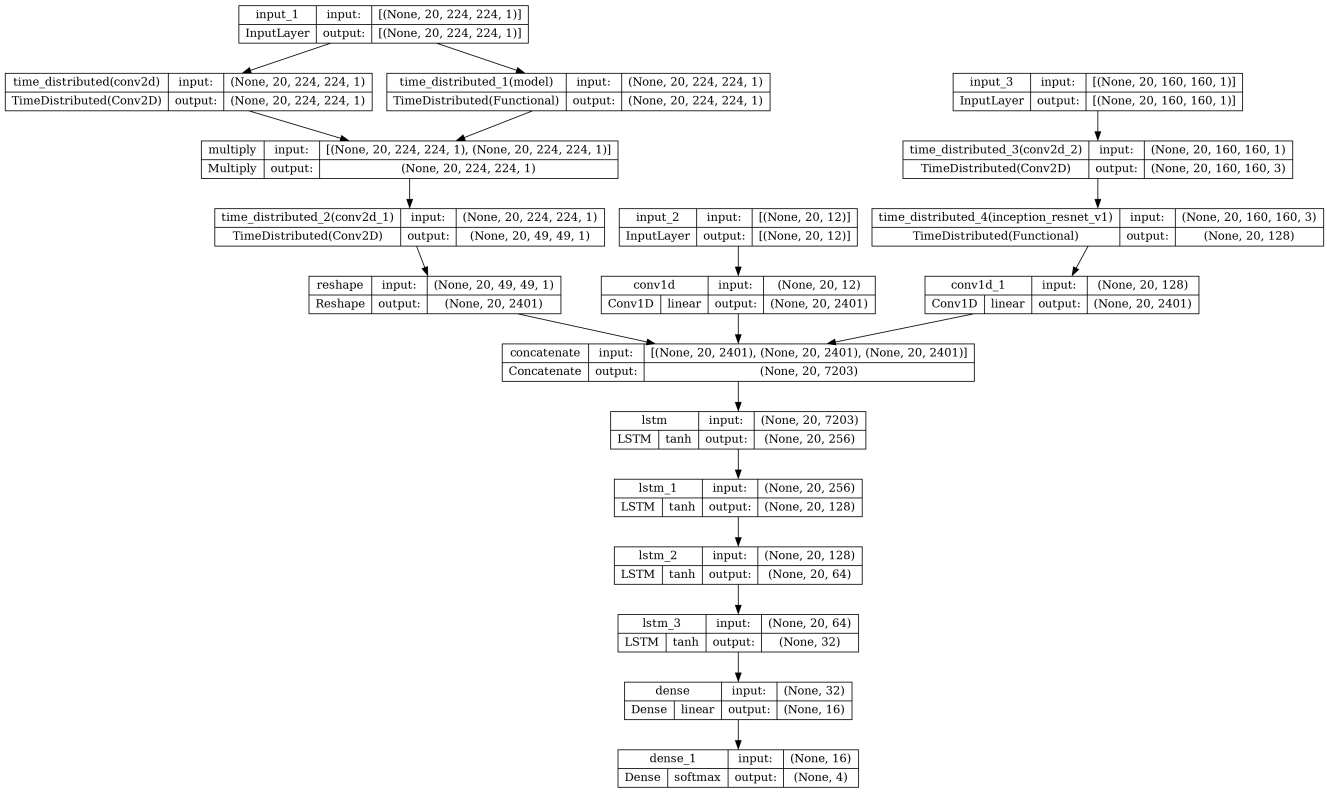


Figure 5.10: Overall Architecture

Chapter 6

Results and Discussion

6.1 Results of driver eye attention model

The driver eye attention model (UNET architecture) outputs a tensor of shape (224,224,1) which is the heatmap of the driver's eye attention for the corresponding road video frame. This tensor helps in providing the overall model only the necessary context from the road video, since there may exist various unwanted elements in the frame that may throw off the accuracy of the overall model. A sample input and its corresponding output of the UNET model is shown in Figure 6.1.

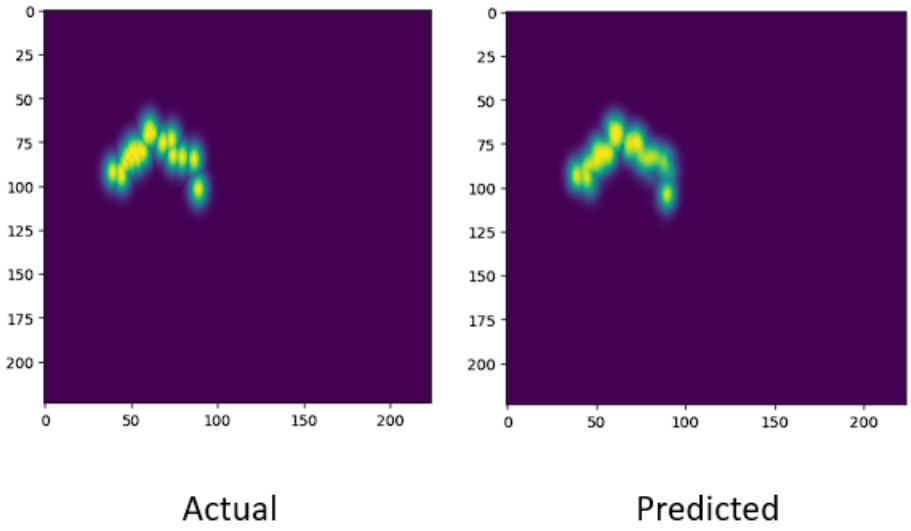


Figure 6.1: Example of driver eye attention model prediction

The UNET model is evaluated on four evaluation metrics usually used for evaluating the performance of segmentation models: IOU score, precision, recall, and mean squared error. Due to the large quantity of data in the DR(eye)VE dataset, the UNET model was trained in two phases, each phase utilizing half of the dataset for training the model. Splitting the dataset can also help prevent overfitting, which occurs when a model becomes too specialized

to the training data and does not generalize well to new, unseen data. By training the model on smaller batches, the model is forced to learn more generalizable patterns in the data, as opposed to memorizing specific examples in the training set. Additionally, training in smaller groups makes monitoring and troubleshooting the training process simpler [27]. Working with smaller batches can make it simpler to identify the problem's root cause and implement the necessary corrections if a problem occurs during training. Figure 6.2 shows the various training graphs corresponding to loss, recall, iou score, and precision.

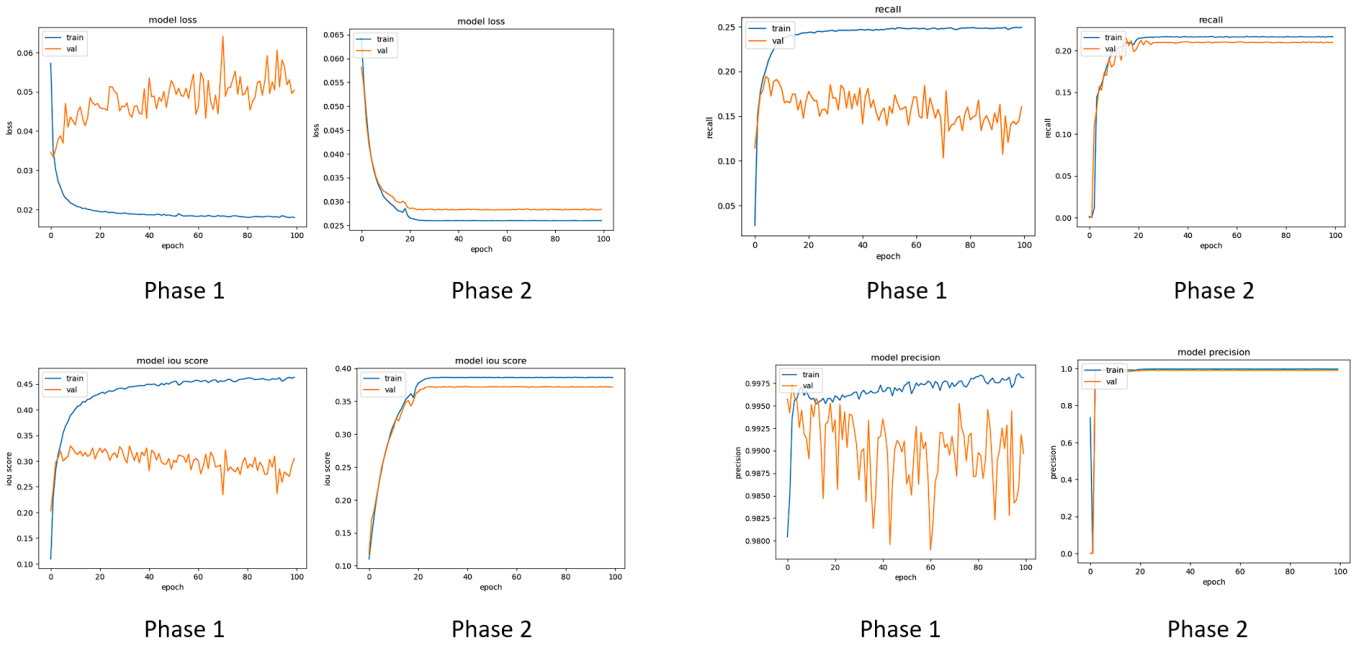


Figure 6.2: Driver eye attention Model evaluation metrics graphs

During the testing phase of the UNET model, a total of 20,000 data is taken for testing, out of which each classes are given equal weightage. The most important parameter to be monitered here is precision, as higher the precision, closer the predicted mask is to the vicinity of the actual eye attention of the driver [28]. The performance parameters are shown in Table 6.1.

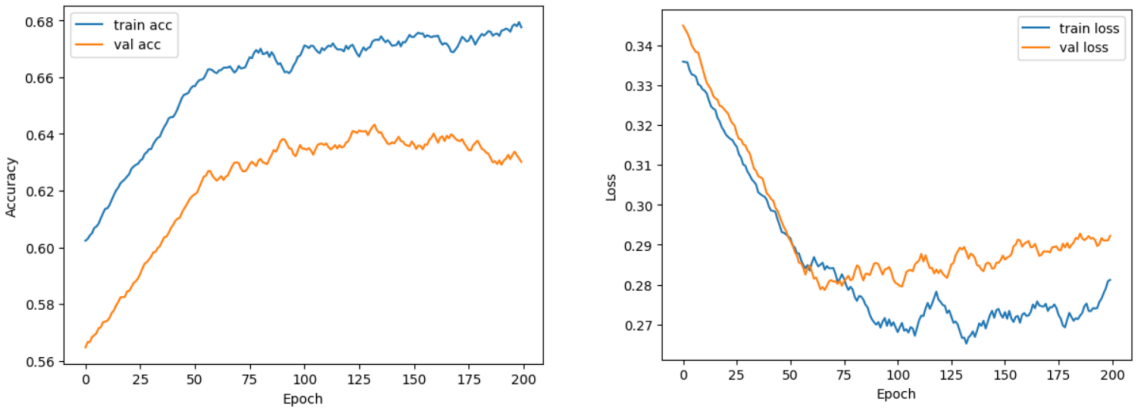
Table 6.1: Performance parameters of driver eye attention model

Performance Parameter	Train	Validation
Precision	99.10%	99.03%
Recall	22.00%	21.16%
IOU Score	37.60 %	36.50%
Loss	0.0260	0.0280

6.2 Results of driver maneuver prediction models

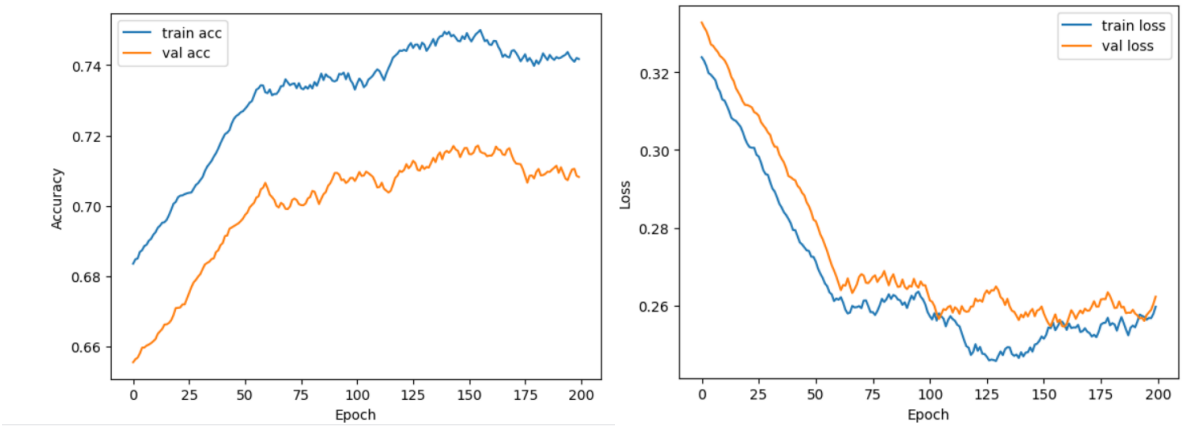
The individual models are constructed by freezing parts of the overall model that deals with the calculation of the other two data sets. The concatenation layer is only present in the feature fusion model.

The performance of the feature fusion model in 5 second look ahead scenario is evaluated by comparing it with the performance of the model using only each individual type of data at a time. This establishes the superiority of fusing the features in the problem of driver maneuver prediction as compared to implementing each data individually. Each model is run for 200 epochs and the best model is considered with respect to the best validation accuracy so as to avoid overfitting of the model. Figures 6.3, 6.4, 6.5, and 6.6 show the training accuracy and loss graphs of the models.



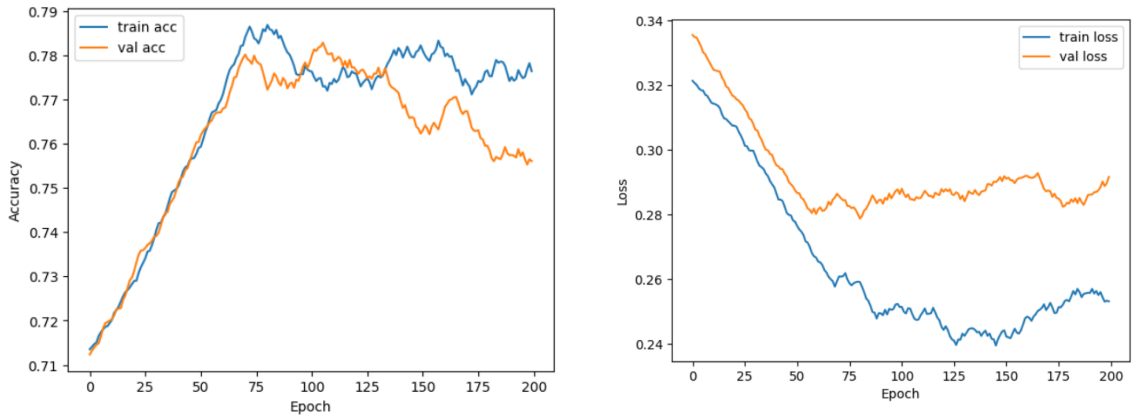
(a) Accuracy (b) Loss

Figure 6.3: Accuracy and loss graphs of driver face only model



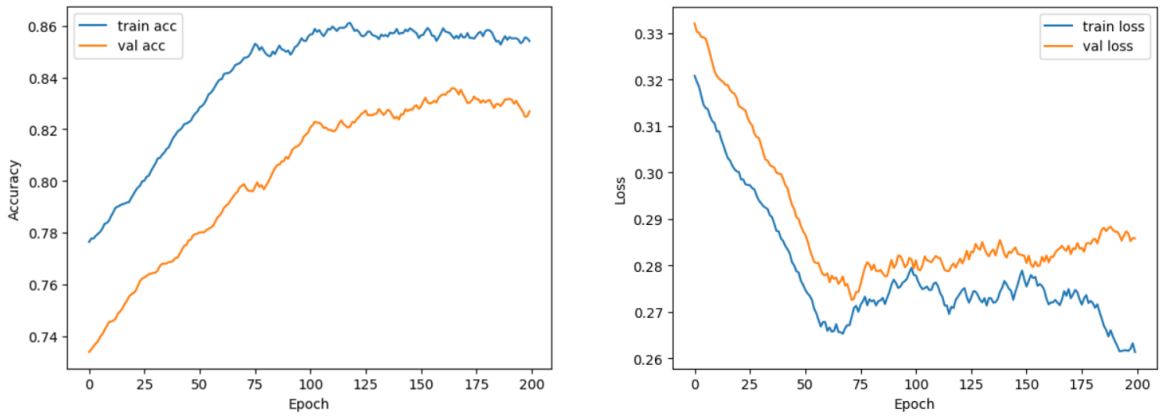
(a) Accuracy (b) Loss

Figure 6.4: Accuracy and loss graphs of road scenes only model



(a) Accuracy (b) Loss

Figure 6.5: Accuracy and loss graphs of signal data only model



(a) Accuracy (b) Loss

Figure 6.6: Accuracy and loss graphs of feature fusion model

Inferencing from the above graphs it is clear that the feature fusion model gives the highest accuracy in a 20 second look ahead scenario. The bar graph comparing the training accuracies are shown in Figure 6.7.

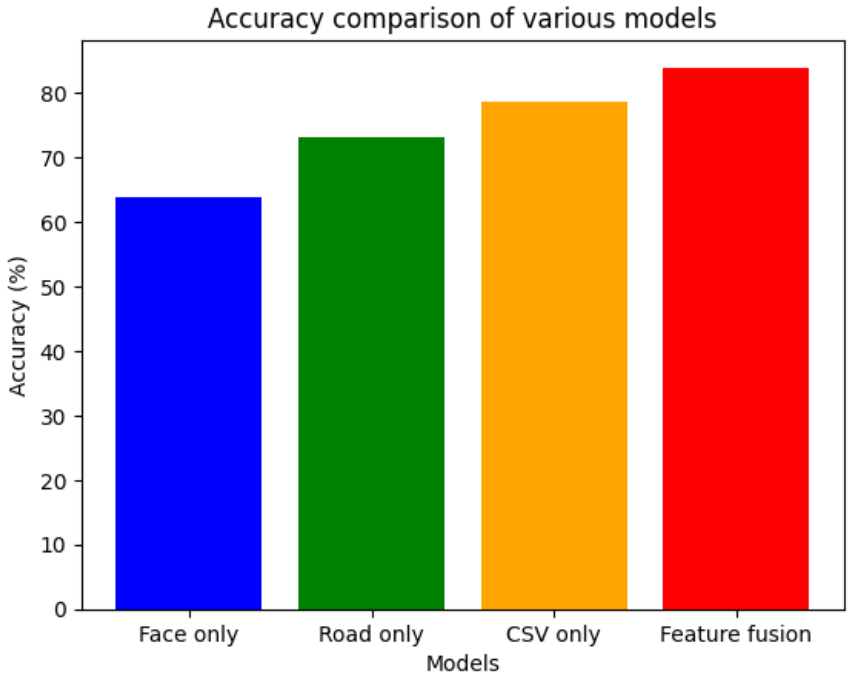


Figure 6.7: Accuracies of the various models

Testing is done on all four models with testing data of size 20000 where each class is represented equally (5000 datapoints each). The models are evaluated using the four parameters of accuracy, precision, recall, and F1 score. The inference time for each model (the time taken by the model to predict a single sequence of input data) is also evaluated. This metric helps in determining the efficiency of using the model in real time. The lower the inference time, the faster the model predicts. The corresponding performance parameter values are shown in Table 6.2.

Table 6.2: Performance parameters of the various models

Performance Parameter	Driver face video only	Road video only	Signals only	Feature fusion model
Accuracy	66.52%	75.37%	82.36%	**88.09%**
Overall Precision	65.28%	77.41%	79.13%	**87.93%**
Overall Recall	61.20 %	77.23%	83.66%	**87.95%**
F1 Score	63.17%	77.32%	81.33%	**87.80%**
Inference Time	95.83 ms	97.56 ms	**63.15 ms**	112.32 ms

The four class confusion matrix of the feature fusion model is shown in Figure 6.8. As evident in the confusion matrix, the model shows high variation in predicting the left class. This may be attributed to the fact that the data set used for training the model is from a right hand drive vehicle and so the drivers focus would more likely be concentrated on the right side [29].

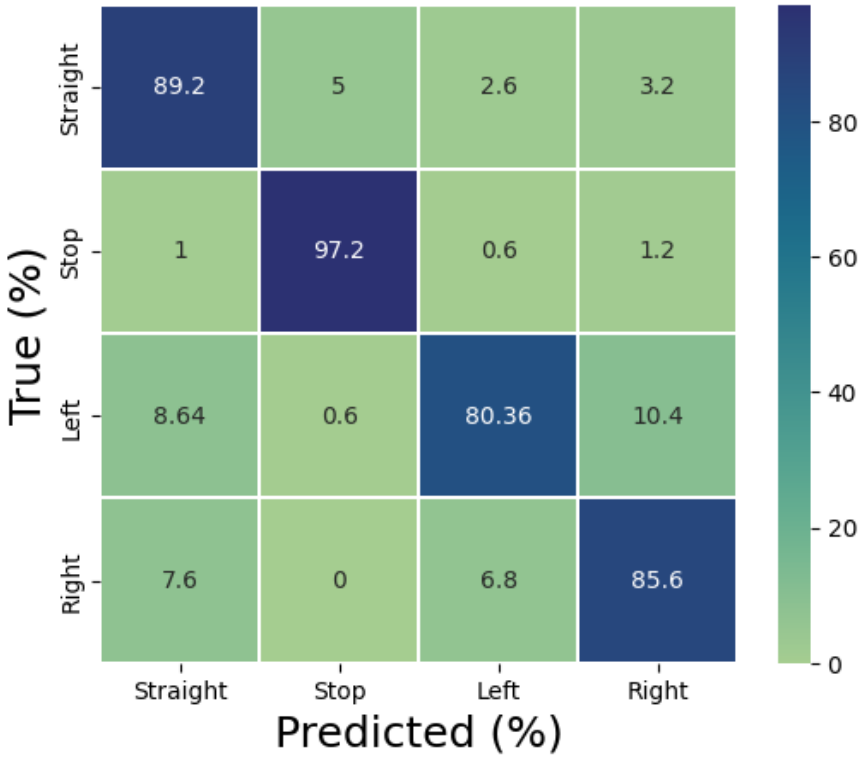


Figure 6.8: Confusion matrix of feature fusion model

The results of the driver eye attention model shows that the UNET model is really capable of localizing to the area at which the drivers eye is attented, as seen by the high precision.

The results of the driver maneuver prediction models show that the feature fusion model achieved the highest performance in terms of accuracy, overall precision, overall recall, and F1 score. These metrics indicate how well the model is able to correctly identify and classify different features in the video data. However, it is also important to consider the inference time, which represents how long the model takes to process the input data and produce an output. In this case, the signal only model achieved the lowest inference time, which is an important factor to consider when the model is deployed in real time [30].

Overall, the feature fusion model can be considered as the best model among the four models as it shows better results in the evaluation parameters among the four models, and its inference time, even though the lowest among the four models, gives a 4.8 second look ahead which is more than sufficient when implementing the model in real time scenarios.

Chapter 7

Conclusion

This work proposes a novel framework for predicting driver maneuvers in real time using deep learning techniques and feature fusion. By leveraging the driver's eye heatmap information, facial landmarking, and attention-based feature fusion, the proposed multi-input deep learning algorithm achieves an accuracy of 88.09% and an overall precision, recall, and F1 score of 87.925%, 87.95%, and 87.80%, respectively. The utilization of the UNet and pre-trained Facenet models for predicting driver eye attention and accurately mapping the driver's face, respectively, has proven to be highly beneficial in removing bias produced by the surroundings during training. These results suggest that the proposed framework has the potential to significantly improve the accuracy of driver maneuver prediction, which can ultimately contribute to enhancing driver safety and reducing accidents on the road.

As part of future work, experimenting with different deep learning modelsmay be taken into consideration. While the UNet and Facenet models used in this work have proven to be effective, there may be other deep learning models that could yield better results. Experimenting with different models could lead to better accuracy and more robust predictions. Additionally, the proposed model can be extended to incorporate more complex maneuvers such as lane change and merging, which require a more sophisticated analysis of the driver's behavior.

References

[1] C. Ou and F. Karray, "Deep learning-based driving maneuver prediction system," *IEEE transactions on vehicular technology*, vol. 69, no. 2, pp. 1328–1340, 2019.

[2] X. Peng, Y. L. Murphey, R. Liu, and Y. Li, "Driving maneuver early detection via sequence learning from vehicle signals and video images," *Pattern Recognition*, vol. 103, p. 107276, 2020.

[3] V. Mahajan, C. Katrakazas, and C. Antoniou, "Prediction of lane-changing maneuvers with automatic labeling and deep learning," *Transportation research record*, vol. 2674, no. 7, pp. 336–347, 2020.

[4] S. Martin, S. Vora, K. Yuen, and M. M. Trivedi, "Dynamics of driver's gaze: Explorations in behavior modeling and maneuver prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 141–150, 2018.

[5] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Smartphone-based adaptive driving maneuver detection: A large-scale evaluation study," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2330–2339, 2017.

[6] N. Khairdoost, M. Shirpour, M. A. Bauer, and S. S. Beauchemin, "Real-time driver maneuver prediction using lstm," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 714–724, 2020.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] C. Olah and S. Carter, "Attention and augmented recurrent neural networks," *Distill*, vol. 1, no. 9, p. e1, 2016.

[9] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[10] Q. Miao, Y. Li, W. Ouyang, Z. Ma, X. Xu, W. Shi, and X. Cao, "Multimodal gesture recognition based on the resc3d network," in *Proceedings of the IEEE international conference on computer vision workshops*, 2017, pp. 3047–3055.

[11] V. Radu, C. Tong, S. Bhattacharya, N. D. Lane, C. Mascolo, M. K. Marina, and F. Kawsar, "Multimodal deep learning for activity and context recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–27, 2018.

[12] P. Marin-Plaza, A. Hussein, D. Martin, and A. de la Escalera, "icab use case for ros-based architecture," *Robotics and Autonomous Systems*, vol. 118, pp. 251–262, 2019.

[13] S. Stević, M. Krunić, M. Dragojević, and N. Kaprocki, "Development and validation of adas perception application in ros environment integrated with carla simulator," in *2019 27th Telecommunications Forum (TELFOR)*. IEEE, 2019, pp. 1–4.

[14] A. Ledezma, V. Zamora, Ó. Sipele, M. P. Sesmero, and A. Sanchis, "Implementing a gaze tracking algorithm for improving advanced driver assistance systems," *Electronics*, vol. 10, no. 12, p. 1480, 2021.

[15] H. Pulver, F. Eiras, L. Carozza, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "Pilot: Efficient planning by imitation learning and optimisation for safe autonomous driving," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1442–1449.

[16] P. Pirri, C. Pahl, N. El Ioini, and H. R. Barzegar, "Towards cooperative maneuvering simulation: Tools and architecture," 01 2021, pp. 1–6.

[17] Y. Hou, Z. Liu, T. Zhang, and Y. Li, "C-unet: Complement unet for remote sensing road extraction," *Sensors*, vol. 21, no. 6, p. 2153, 2021.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[19] C. Wu and Y. Zhang, "Mtcnn and facenet based access control system for face detection and recognition," *Automatic Control and Computer Sciences*, vol. 55, pp. 102–112, 2021.

[20] H. Abbasimehr and R. Paki, "Improving time series forecasting using lstm and attention models," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2022.

[21] S. Alletto, A. Palazzi, F. Solera, S. Calderara, and R. Cucchiara, "Dr (eye) ve: a dataset for attention-based tasks with applications to autonomous and assisted driving," in *Proceedings of the ieee conference on computer vision and pattern recognition workshops*, 2016, pp. 54–60.

[22] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[23] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 909–916.

[24] G. Underwood, D. Crundall, and P. Chapman, "Driving simulator validation with hazard perception," *Transportation research part F: traffic psychology and behaviour*, vol. 14, no. 6, pp. 435–446, 2011.

[25] Ş. Işık, K. Özkan, S. Günal, and Ö. N. Gerek, "Swcd: a sliding window and self-regulated learning-based background updating method for change detection in videos," *Journal of Electronic Imaging*, vol. 27, no. 2, pp. 023 002–023 002, 2018.

[26] Q.-S. Sun, S.-G. Zeng, Y. Liu, P.-A. Heng, and D.-S. Xia, "A new method of feature fusion and its application in image recognition," *Pattern Recognition*, vol. 38, no. 12, pp. 2437–2448, 2005.

[27] K. M. Kahloot and P. Ekler, "Algorithmic splitting: A method for dataset preparation," *IEEE Access*, vol. 9, pp. 125 229–125 237, 2021.

[28] F. Solera and R. Cucchiara, "Predicting the driver's focus of attention: The dr (eye) ve project," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, 2019.

[29] A. Koesdwiady, S. M. Bedawi, C. Ou, and F. Karray, "End-to-end deep learning for driver distraction recognition," in *Image Analysis and Recognition: 14th International Conference, ICIAR 2017, Montreal, QC, Canada, July 5–7, 2017, Proceedings 14*. Springer, 2017, pp. 11–18.

[30] L.-A. Tran, T.-D. Do, D.-C. Park, and M.-H. Le, "Robustness enhancement of object detection in advanced driver assistance systems (adas)," *arXiv preprint arXiv:2105.01580*, 2021.