TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KALANKI, KATHMANDU

[CT 755]

A MAJOR PROJECT REPORT ON

"DRIVER STATE DETECTION SYSTEM USING DEEP LEARNING"

Submitted By

Anupam Gautam	(41218)
Ashish Dhakal	(41220)
Bibek Chand	(41223)
Khagendra Singh Jora	(41238)

Project Supervisor:

Asst. Prof. Dr. Raksha Dangol

A Major Project Final report submitted to the department of Electronics and Computer Engineering in the partial fulfillment of the requirements for degree of Bachelor of Engineering in Computer Engineering

Kathmandu, Nepal

9th March, 2024

DRIVER STATE DETECTION SYSTEM USING DEEP LEARNING

Submitted By

Anupam Gautam	(41218)
Ashish Dhakal	(41220)
Bibek Chand	(41223)
Khagendra Singh Jora	(41238)

Supervised by:

Dr. Raksha Dangol

Assistant Professor

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

"DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING"

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

Kalanki, Kathmandu

9th March, 2024

LETTER OF APPROVAL

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled "**DRIVER STATE DETECTION SYSTEM USING DEEP LEARNING**" submitted by:

Anupam Gautam	(41218)
Ashish Dhakal	(41220)
Bibek Chand	(41223)
Khagendra Singh Jora	(41238)

In the partial fulfillment of the requirements for the degree of Bachelor's Degree in Computer Engineering.

……..

Project Supervisor

Asst. Prof. Dr. Raksha Dangol

Department of Electronics and Computer Engineering

……..

Head of Department

Senior Asst. Prof. Prem Chandra Roy

Department of Electronics and Computer Engineering

……..

External Examiner

9th March, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We take this opportunity to express our deepest and sincere gratitude to our Project Supervisor **Dr. Raksha Dangol** for her insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project and also for her constant encouragement and advice throughout our Bachelor's programme.

We express our deep gratitude to **Er. Prem Chandra Roy**, Head of the Department of Electronics and Computer Engineering, **Er. Bikash Acharya, Er. Dhiraj Pyakurel** Deputy Head, Department of Electronics and Computer Engineering, **Er. Laxmi Prasad Bhatt,** Academic Project Coordinator, Department of Electronics and Computer Engineering, for their support, co-operation, and coordination.

The in-time facilities provided by the department throughout the Bachelor's program are also equally acknowledgeable.

We would like to convey our thanks to the teaching and non-teaching staff of the Department of Electronics & Communication and Computer Engineering, ACEM for their invaluable help and support throughout Bachelor's Degree. We are also grateful to all our classmates for their help, encouragement, and invaluable suggestions.

Finally, yet more importantly, we would like to express our deep appreciation to our grandparents, parents, and siblings for their perpetual support and encouragement throughout the Bachelor's degree period.

Project Members:

Anupam Gautam	(41218)
Ashish Dhakal	(41220)
Bibek Chand	(41223)
Khagendra Singh Jora	(41238)

ABSTRACT

This project introduces a driver state detection system aimed at tackling the critical issues of driver distraction and drowsiness, which are major contributors to road accidents. The system comprises two key components: distraction detection and drowsiness detection. The distraction detection module is capable of identifying various distractions including texting, talking on the phone, operating the radio, drinking, reaching behind, applying makeup, and talking to passengers on the given input. On the other hand, the drowsiness detection component is designed to detect signs of drowsiness in real-time using live video input. Through the use of deep neural networks, this system can accurately monitor driver behavior and promptly alert drivers to unsafe actions or indications of drowsiness. By effectively addressing both distraction and drowsiness through intelligent technologies, this project aims to significantly mitigate accidents and promote safer roads.

Keywords: *Computer Vision, Deep learning, Distraction, Driver State Detection, Drowsiness, Real-time Alert System*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence.
CDCNN	Custom Deep Convolutional Neural Network.
CNN	Convolutional Neural Network.
DarNet	Deep Active Ray Network.
GMM	Gaussian Mixture Model.
GPU	Graphics Processing Unit.
HDNN	Hybrid Deep Neural Network.
IMU	Inertial Measurement Unit.
LSTM	Long Short-Term Memory.
OS	Operating System.
ResNet	Residual Neural Network.
SDLC	Software Development Life Cycle Model.
VGG	Visual Geometry Group.
WHO	World Health Organization.
Xception	Extreme Inception.

CHAPTER 1: INTRODUCTION

1.1 Background

Road traffic accidents are a grave concern worldwide, resulting in a significant number of fatalities and injuries each year. According to the World Health Organization (WHO), approximately 1.3 million people lose their lives annually due to road accidents, and this number has been steadily increasing in recent years. The impact of these accidents is not limited to loss of life but also encompasses severe injuries and extensive economic damage to individuals and their families. Shockingly, road traffic injuries have become the leading cause of death among individuals aged 5 to 29. [1]

Various factors contribute to automobile accidents, but distracted driving and drowsiness have emerged as significant causes that contribute to fatal crashes and injuries. Distracted driving refers to any activity that diverts a driver's attention and impairs their ability to focus on the road. This can include engaging in activities like using a mobile phone, texting, reading, eating, or interacting with in-vehicle infotainment systems. Distractions can arise from the driver's own actions or from external stimuli that divert their attention.

There are several types of distractions that affect a driver's performance and safety on the road. Manual distractions occur when a driver takes their hands off the steering wheel, while visual distractions occur when a driver takes their eyes off the road. Cognitive distractions occur when a driver's mind wanders away from driving-related tasks. Other forms of distractions include olfactory distractions (related to smells), gustatory distractions (related to taste), and auditory distractions (related to sound). [2] Distracted driving, especially the use of cell phones, has been identified as a major cause of car accidents, accounting for 26% of all accidents. [1]

Sleepiness is another critical factor that compromises driver safety. Sleep deprivation or abnormalities can lead to reduced reaction times, impaired decision-making, and disrupted brain functioning. Sleepiness contributes significantly to road traffic accidents and is often experienced by fatigued drivers, who have sleep disorders, consume alcohol or medications, or drive for extended periods without adequate rest. Studies have consistently shown that sleepiness is one of the primary factors contributing to road accidents, accounting for 3% to over 30% of accidents globally. [3]

To address the safety risks associated with distracted driving and drowsiness, there is a need for systems that can monitor, detect, and predict driver inattentive behavior. Early detection and timely warnings can help prevent accidents by accurately identifying distracted driving states and alerting drivers accordingly.

1.2 Motivation

The alarming statistics and the devastating consequences of road traffic accidents caused by distracted driving and drowsiness provide strong motivation for the project. The need to address these issues is urgent, as they pose a significant threat to public safety and result in loss of life, severe injuries, and economic damages. By developing a comprehensive driver-state detection system, the aim is to contribute to the reduction of accidents and improve road safety.

1.3 Problem Statement

The problem at hand is the prevalence of road traffic accidents caused by distracted driving and drowsiness. These accidents result in significant loss of life, injuries, and economic damages. The existing approaches to address these issues are insufficient, and there is a need for an advanced driver-state detection system that can accurately identify and classify distracted and non-distracted driving states. The system should integrate distraction and drowsiness detection capabilities and provide timely alerts to drivers, helping them maintain focus and prevent accidents.

1.4 Project Objective

To develop the driver state detection system using deep learning.

1.5 Significance of Study

The significance of the project lies in its potential to improve road safety and reduce the number of accidents caused by distracted driving and drowsiness. By developing an advanced driver state detection system, the aim is to contribute to the reduction of fatalities, injuries, and economic damages associated with road traffic accidents. The integration of AI techniques and innovative technologies in the system allows for real-time monitoring and accurate detection of distracted and non-distracted driving states. This project addresses not only the immediate safety concerns but also aligns with the growing focus on driver assistance systems and the adoption of AI in the automotive industry. The findings and insights from this project can potentially inform future research and development efforts in the field of driver state detection and contribute to advancements in road safety technologies.

CHAPTER 2: LITERATURE REVIEW

This section provides a summary of relevant and significant literature related to the project, focusing on the use of various techniques for the detection of distracted driving behaviors and drowsiness.

C. Streiffer et al. proposed an analysis framework called DarNet for distracted driving behavior detection and classification. They employed a unified data collection approach and utilized a combination of convolutional and recurrent neural networks to analyze images of distracted driving. By leveraging IMU sensor data, the DarNet framework achieved improved accuracy in detecting six classes of distracted driving behaviors. [4]

H. M. Eraqi et al. introduced a CNN-based technique for recognizing distracted drivers. They achieved high precision in detecting a wide range of objects and ten driving distraction postures, including postures not covered by ImageNet classes [5].

A. Koesdwiady et al. proposed an end-to-end CNN-based solution for identifying distracted drivers. They utilized the VGG19 model for feature extraction, resulting in a high accuracy of 95% in testing.[6]

M. Leekha et al. presented a robust CNN-based architecture for distracted driver detection. Their design incorporated foreground extraction and progressively increasing 3 × 3 filters in successive layers, leading to improved accuracy and a lower false positive rate compared to existing models.[7]

J. M. Celaya-Padilla et al. developed a method for detecting drivers who are distracted by using their cell phones. They integrated a deep learning CNN with a wide-angle camera mounted on a ceiling and trained an Inception V3 network to detect drivers who are driving and texting on their cell phones.[8]

Behera et al. used both AUC and State Farm's distracted driver detection dataset with sequence information of the images, i.e., they used the video version of the dataset. They proposed a novel deep neural network approach called Multi-stream LSTM (M-LSTM). They used appearance features and contextual information (e.g., pose and objects) obtained from another pre-trained CNN with distinct combinations as multistream inputs for their M-LSTM. They achieved 52.22% and 91.25% accuracy on AUC and State Farm's distracted driver datasets, respectively. [9]

Xing et al. classified seven driving actions using a combination of body cropping, Gaussian Mixture Model (GMM) feature extraction, and CNN classification. Their approach achieved a binary distracted driver prediction accuracy of 91% using the AlexNet model. [10]

Omerustaoglu et al found that fusing sensor data with vision data increases the accuracy of distracted driver detection tasks successfully. Specifically, both hybrid and prediction level fusion increased the overall accuracy by 9%, from 76% to 85%, when compared to using only image data. It is found that using sensor data increased the normal driving detection accuracy from 74% to 85%. The statistical tests showed that both of these differences are significant. [11]

J. Li et al. developed a drowsiness detection system using a hybrid deep neural network (HDNN) that combined a CNN with an LSTM. Their model incorporated features from multiple modalities, including facial landmarks, eye features, and head pose estimation, achieving an accuracy of 98.84% in drowsiness detection. [12]

A. Gumaei et al. proposed a camera-based deep learning framework for the real-time identification of driver distractions, leveraging cloud computing and edge computing technologies. Their framework consisted of three modules: a distraction detection module, a training module, and an analysis module. By combining custom deep convolutional neural networks (CDCNN) and fine-tuned VGG16 models, they achieved high accuracy rates of 99.64% and 99.73% in detecting distracted driving behaviors. [13]

CHAPTER 3: REQUIREMENT ANALYSIS

3.1 Software Implementation

The software requirements of the project are:

- Python
- Pandas
- OpenCV
- NumPy
- TensorFlow
- Django
- Operating System (OS)(Windows)

3.1.1 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

3.1.2 Pandas

Pandas is a powerful and widely used open-source data manipulation and analysis library for Python. It provides easy-to-use data structures, such as data frames and series, along with a variety of functions to manipulate and analyze structured data efficiently

3.1.3 OpenCV

OpenCV is a popular open-source computer vision and image processing library. It provides a wide range of tools and functions for working with images and videos

3.1.4 NumPy

NumPy, short for Numerical Python, is a powerful open-source library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays.

3.1.5 Django

Django is a high-level, open-source web framework for building web applications using the Python programming language. It follows the Model-View-Controller (MVC) architectural pattern, but in Django, it's often referred to as the Model-View-Template (MVT) pattern.

3.1.6 TensorFlow

TensorFlow is an open-source machine learning framework developed by the Google Brain team. It is designed to facilitate the development and deployment of machine learning models, particularly for deep learning tasks.

3.2 Hardware Implementation

The hardware requirements of this project are:

- Laptop
- GPU

3.3 Functional Requirements

The functional requirements of this project are:

- Correctness: The system should be able to recognize the state of the driver correctly.
- Flexibility: The system shall not be demanding to organize and should be understandable. It must also be easy to implement in most systems.
- Accuracy: The system must provide accurate and real time information.

3.4 Non-functional Requirements

The non-functional requirements of this project are:

- Speed: The system must provide real time results.
- Reliability: The system must be able to perform well even in suboptimal working conditions
- The application should be fault tolerant
- Scalability: The system is expected to be scalable so that it will be suitable for a very large number of users. The performance and accuracy of the system must remain consistent with the increase in size of the user.

3.5 Feasibility Study

Feasibility is a crucial aspect that determines the success of any system, software, or application. The primary concern of application users revolves around its feasibility, as it directly impacts the viability and effectiveness of the solution.

3.5.1 Technical Feasibility

Technical feasibility is done to make sure whether the available resources would be able to work in the existing infrastructure or not. It also compares the level of technology available for the users and the level of technology required for the system development. In the case of the project, basic hardware components as well as software components were enough.

3.5.2 Economic Feasibility

The designed system demonstrates economic feasibility. The total expenditure of the project primarily comprises computational power. The software is cost-effective since it does not necessitate any additional expensive hardware. Consequently, the project is deemed economically feasible.

3.5.3 Operational Feasibility

The system working is quite easy to use and learn due to its simple but attractive interface. Users require no special training for operating the system. Any user who has basic knowledge can operate the system.

CHAPTER 4: SYSTEM DESIGN AND ARCHITECTURE

4.1 Block Diagram

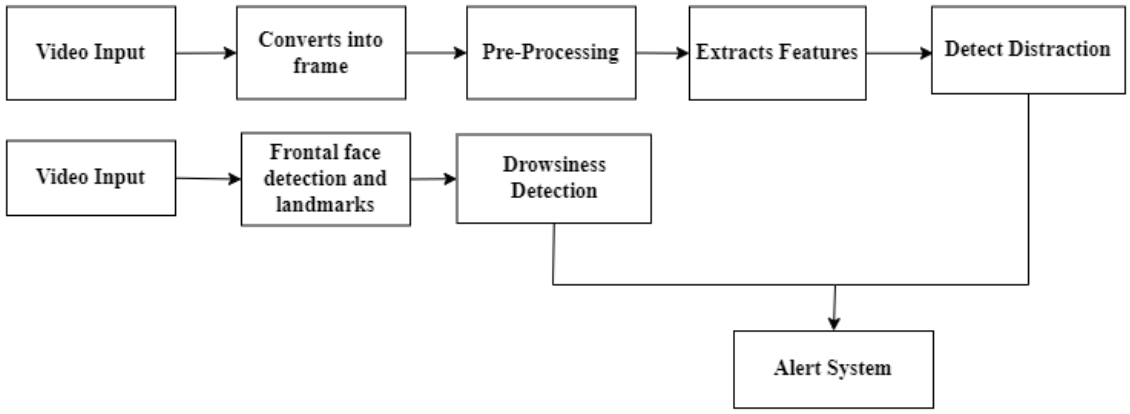

Figure 4.1: Block Diagram

The driver state detection system operates through two main pathways: distraction detection and drowsiness detection, both geared towards enhancing road safety. In the distraction detection pathway, the system meticulously processes video input, breaking it down into frames and analyzing posture features. By pinpointing the driver's position within the frame, it can identify any distractions present. When distractions are detected, the system promptly issues alerts.

Simultaneously, the drowsiness detection pathway utilizes live video from the system's camera. It employs Dlib pre-trained model to identify facial landmarks, focusing on the frontal face region. Subsequently, features such as Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) are computed based on these landmarks. These calculated values are then utilized as inputs for the drowsiness detection model, allowing it to analyze signs of drowsiness accurately. By promptly recognizing these indicators, the system helps ensure driver alertness, thereby reducing the risk of accidents on the road.

4.2 Flowchart Diagram

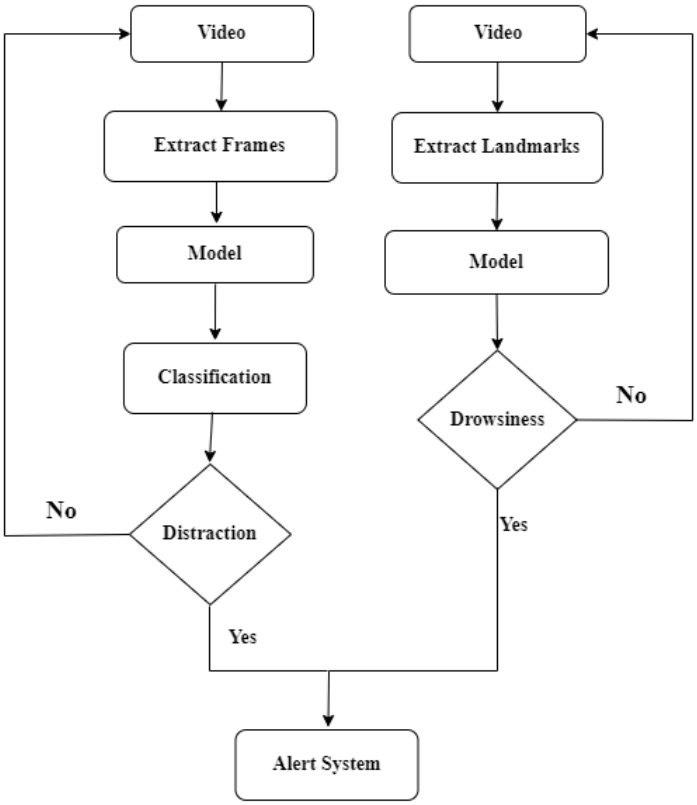

Figure 4.2: Flowchart Diagram

The driver state detection system is designed to assess the condition of the driver through two distinct pathways: distraction detection and drowsiness detection. In the distraction detection pathway, the system processes video input, extract frame and uses the model to identify any signs of distraction. Upon detection, alerts are issued to notify the driver. Meanwhile, the drowsiness detection pathway utilizes Dlib to identify facial landmarks and uses model to evaluate drowsiness. Both pathways converge at the alert system.

4.3 Use Case Diagram

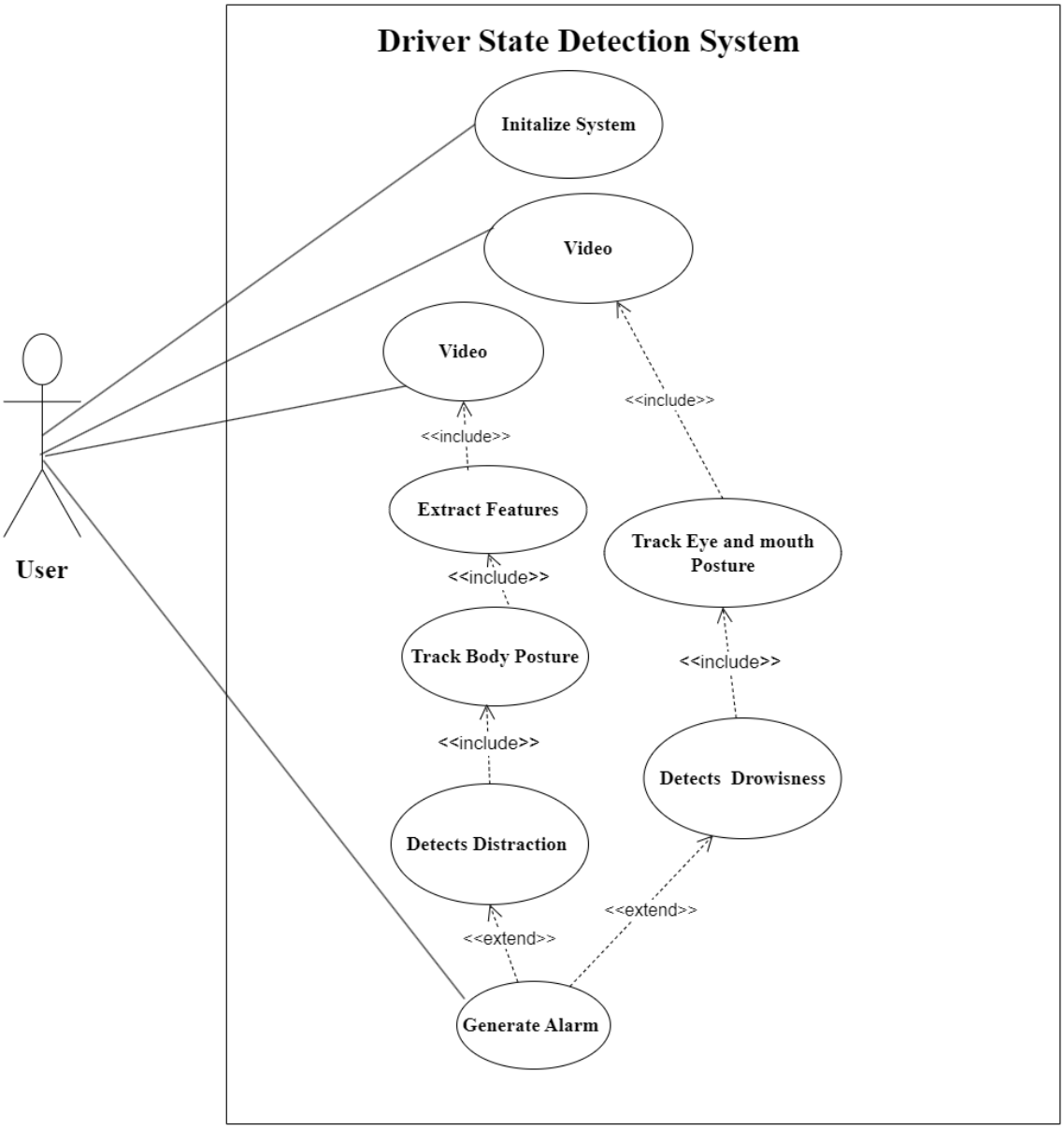

Figure 4.3: Use Case Diagram

The driver state detection system's use case involves the user initializing the system, which includes setting up a camera for video input. The system then extracts features from the video feed and tracks body posture to detect potential distractions, issuing alarms when necessary for distraction. Additionally, the system tracks eye and mouth landmarks in the video feed to accurately detect signs of drowsiness.

4.4 DFD Diagram

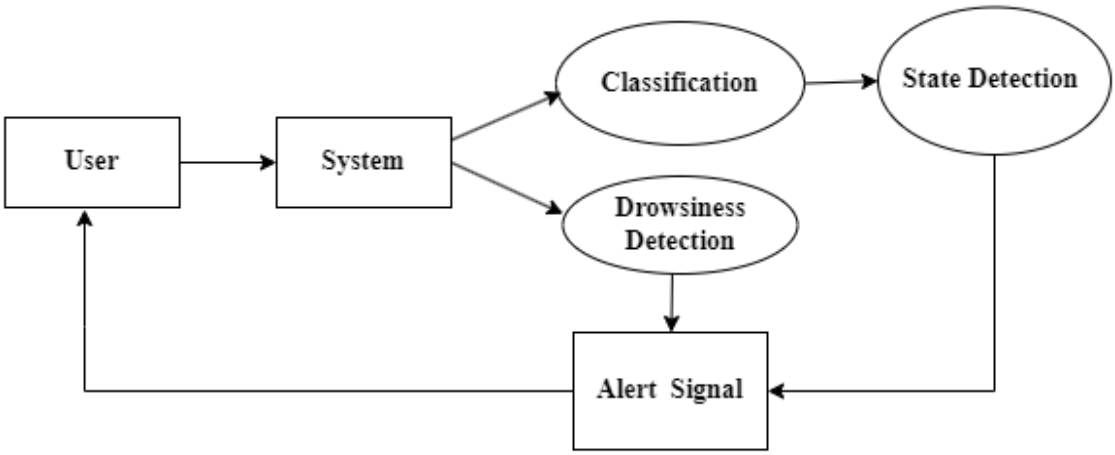

Figure 4.4: DFD Level 0

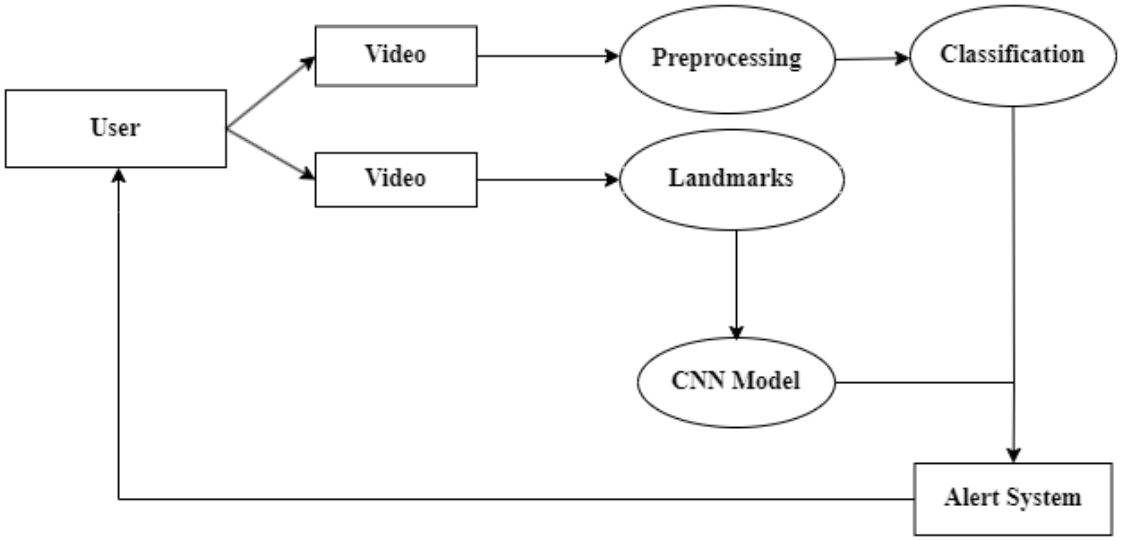

Figure 4.5: DFD level 1

At DFD level, the driver state detection system is segmented into subsystems: distraction detection, drowsiness detection, and alert signaling. It interfaces with external entities such as the user, video input source, and alert mechanism.

DFD1 elaborates further on the processing of video input for distraction detection and drowsiness detection separately. Distraction detection involves video processing and classification, while drowsiness detection utilizes a CNN model to analyze facial landmarks. Alert signals are subsequently generated according to the identified driver state.

CHAPTER 5: METHODOLOGY

5.1 Software Development Life Cycle Model (Iterative Model)

The chosen software development life cycle (SDLC) model for project implementation is the Iterative Model. This approach divides software development into smaller segments to facilitate easier management of work. Initially, a simplified implementation is prioritized, with complexity increasing after each iteration as additional features are incorporated until the final product is deployed. The Iterative Model is closely associated with incremental models, where an early prototype is delivered to the client for feedback, allowing for potential adjustments. Consequently, with each iteration, an enhanced version of the product is delivered. This model also enables the generation of working software swiftly, reducing the time allocated to documentation and allocating more time for development. By adopting this model, errors can be identified and rectified in the early stages of development leading to improved project outcome overall.

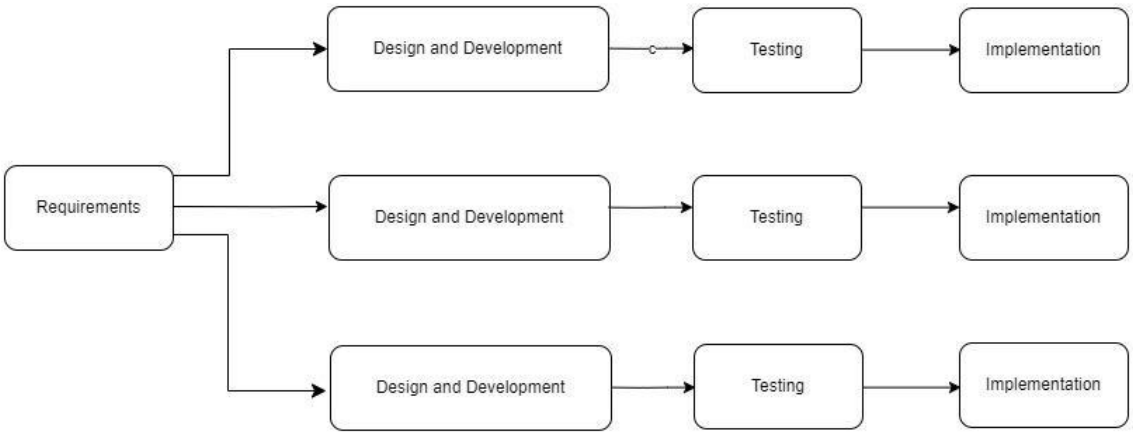

Figure 5.1: Iterative Model

[Source: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm/]

5.2 Dataset Collection

The dataset used in the project for distraction detection is the StateFarm distraction-detection dataset, which was obtained from Kaggle as part of the competition. It contains 22424 training images. The dataset defines 10 states here. They are:

c0: safe driving

c1: texting - right

c2: talking on the phone - right

c3: texting - left

c4: talking on the phone - left

c5: operating the radio

c6: drinking

c7: reaching behind

c8: hair and makeup

c9: talking to passenger

For drowsiness detection, the project utilized the Uta-reallife-drowsiness-dataset, from which 3,609 images were selected. The project then computed the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) from these images to determine drowsiness.

Regarding the CNN models for both distraction and drowsiness detection, the dataset was split into 80% for training and 20% for testing. This partitioning allowed for robust model training and evaluation.

For distraction detection using ensemble averaging, 18,732 images were utilized for training, while 3,692 images were reserved for testing purposes. This approach ensured the model's effectiveness in identifying distractions in various scenarios.

5.3 Algorithms Used

5.3.1 CNN (Convolution Neural Network)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. CNN are mostly used for images and basically the are two types of images, Grayscale and RGB. [14]

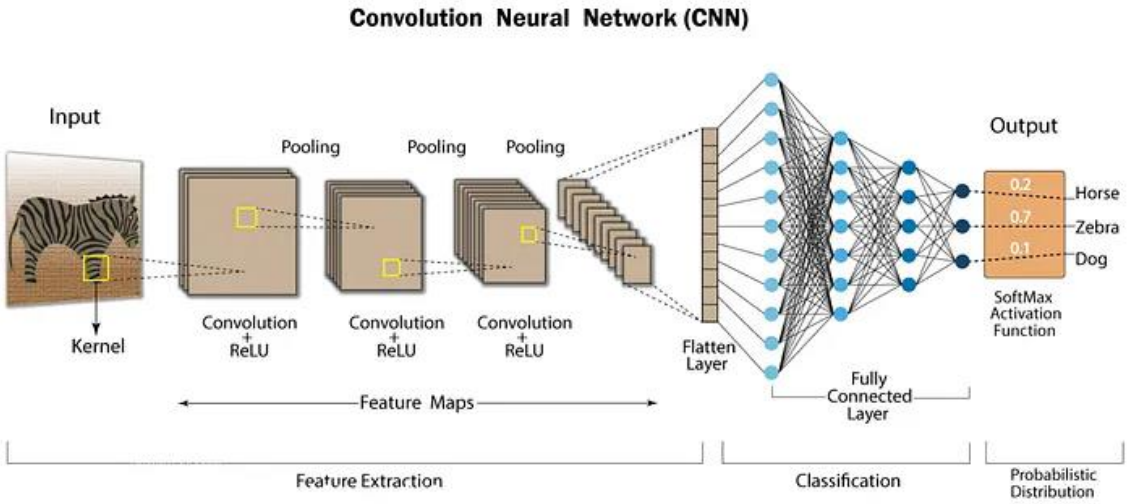

Figure 5.2: CNN Architecture

[Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148]

An RGB image can be regarded as a matrix of pixel values, consisting of three planes that represent the red, green, and blue color channels respectively. On the other hand, a grayscale image is similar, but it comprises only a single plane representing the intensity values of the pixels. These planes serve as essential components that provide distinct information about the image.

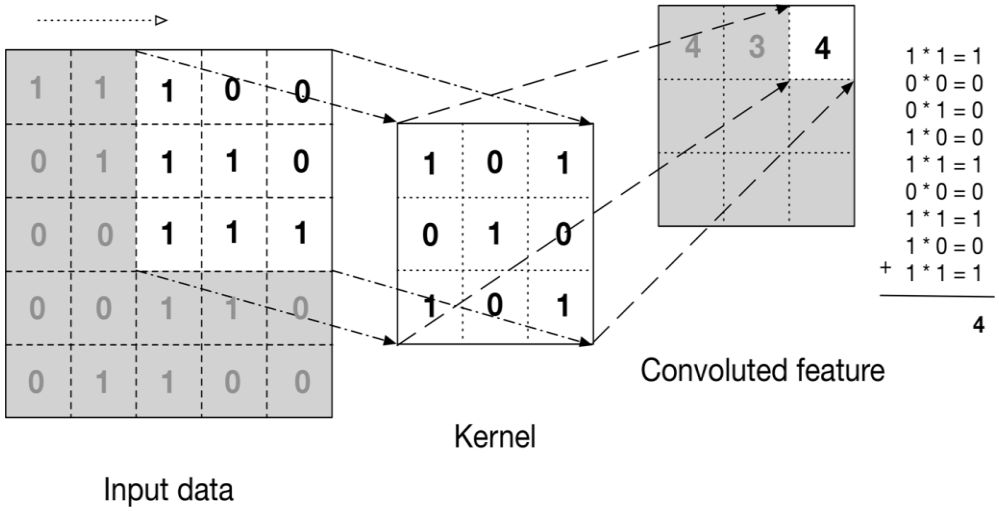

Figure 5.3: Grayscale Image

[Source: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/]

The kernel is the matrix that is applied to the input images that extracts the features of the images. The kernel moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products. The kernel moves on the input data by the stride value. Lastly, the kernel is used to extract high-level features like edges from the image.

Pooling is the process of siding a two-dimensional filter over each channel of a feature map and summarizing the features lying within the region covered by the filter. It reduces the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

A fully Connected Layer is like ANN where every input layer is connected to each output node so every input of the input vector influences every output of the output vector. In, CNN these layers are used for classification purposes. [15]

5.3.2 ResNet (Residual Neural Network)

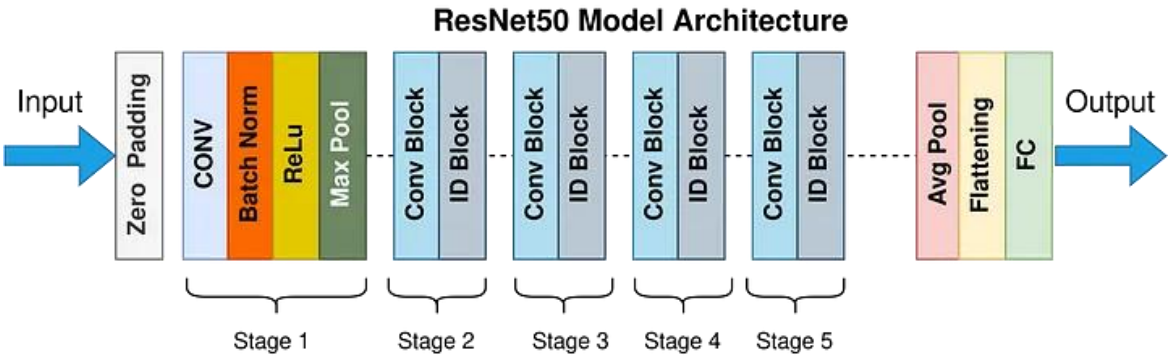

Figure 5.34: ResNet Architecture

*[*Source: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758*]*

The ResNet architecture is considered to be among the most popular Convolutional Neural Network architectures around. Introduced by Microsoft Research in 2015, [16]

The ResNet-50 architecture can be broken down into 6 parts.

- Input Pre-processing
- Cfg[0] blocks
- Cfg[1] blocks
- Cfg[2] blocks
- Cfg[3] blocks
- Fully-connected layer

5.3.3 Xception (Extreme Inception)

The Xception architecture is a deep neural network architecture designed for image classification tasks. Introduced by François Chollet in 2017, Xception stands for "Extreme Inception," and it is an extension of the ideas presented in the Inception architecture.

Xception Architecture

The key innovation in Xception is the use of depthwise separable convolutions, which decouple spatial and cross-channel information processing. The architecture can be summarized as follows:

Depthwise Separable Convolution:

- Depthwise Convolution: Applies a separate convolutional filter to each input channel independently, performing spatial filtering.
- Pointwise Convolution: Applies a 1x1 convolution to combine the spatial information, capturing cross-channel interactions. [17]

Entry Flow:

- Begins with a standard convolutional layer.
- Followed by a sequence of depthwise separable convolution blocks.
- Utilizes skip connections (residual connections) for improved information flow.

Middle Flow:

- Repeats a series of depthwise separable convolution blocks with skip connections.

Exit Flow:

- Concludes with a global average pooling layer to aggregate spatial information.
- Followed by a fully connected layer (SoftMax activation) for classification

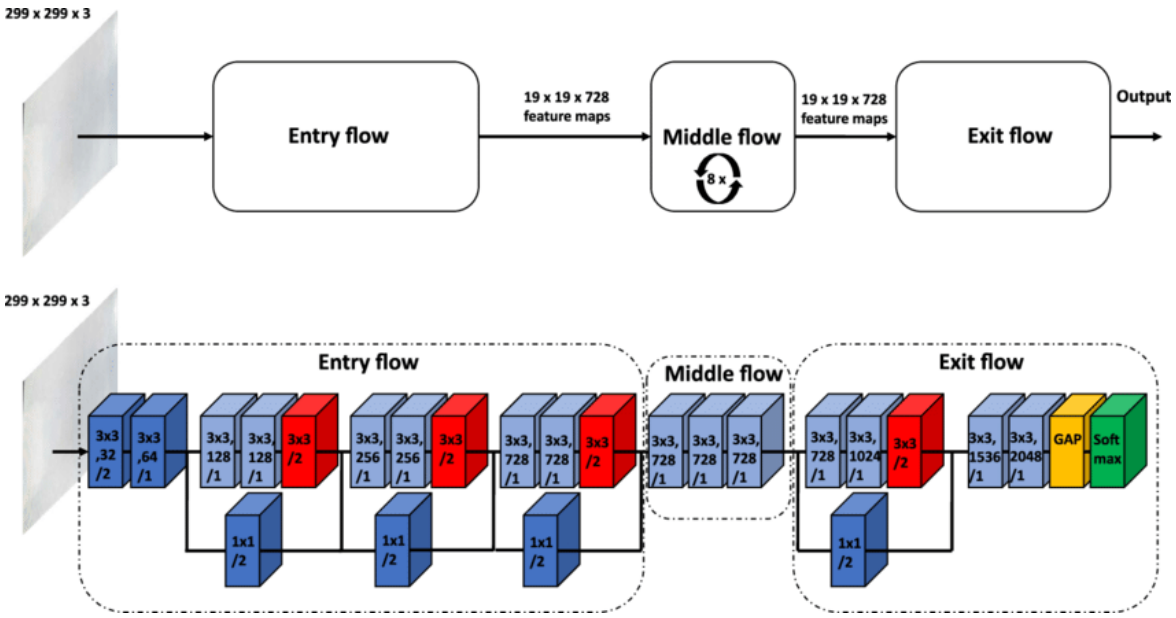

Figure 5.35: Xception Architecture

[Source: https://maelfabien.github.io/deeplearning/xception/]

5.3.4 MobileNet

MobileNet is a family of lightweight deep neural network architectures tailored for mobile and edge devices with constrained computational resources. Introduced by Google in 2017, MobileNet employs depthwise separable convolutions, breaking down standard convolutions into depthwise and pointwise convolutions to significantly reduce computational cost. The concept of inverted residuals with linear bottlenecks, as seen in MobileNetV2, enhances information flow. It also features hyperparameters like width and resolution multipliers, offering flexibility to balance model size and accuracy based on device capabilities. Known for its efficiency, MobileNet is commonly used in image classification, object detection, and semantic segmentation tasks, enabling real-time inference on devices with limited resources. [18]

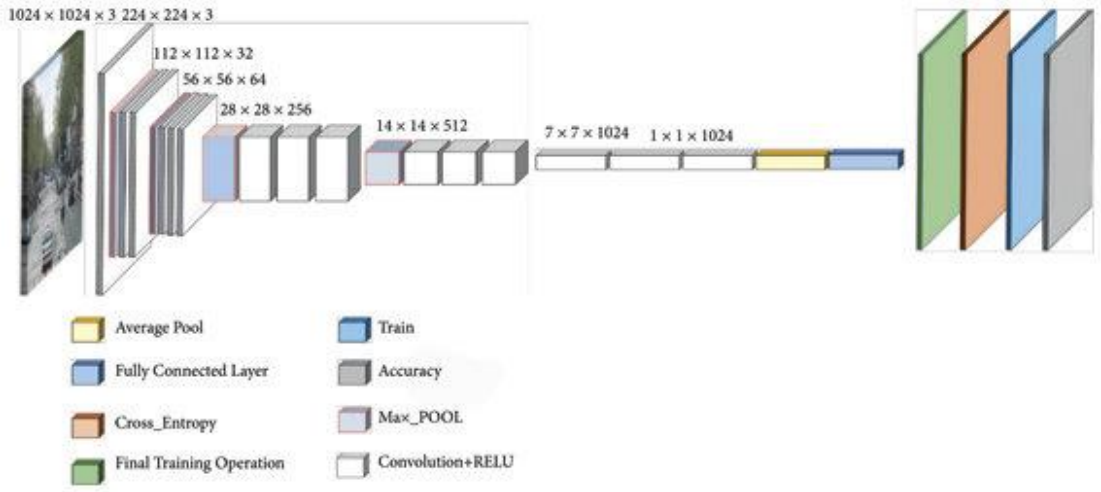

Figure 5.36: MobileNet

[Source: https://medium.com/@godeep48/an-overview-on-mobilenet-an-efficient-mobile-vision-cnn-f301141db94d]

5.3.5 Average Ensembling

After employing the three models, the project computed the average of their predictions. This average served as the final output for distraction detection. By aggregating the predictions from multiple models, the project aimed to enhance the accuracy and reliability of the distraction detection system.

5.3.6 Loss:

To determine the loss for State Detection, the project employed Log loss. This loss function measures the disparity between the predicted probabilities and the actual target values. It was calculated using the following formula:

$$\textbf{Log Loss} = -\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{i=1}^{N}\left[y_i log\left(p(y_i)\right) + (1-y_i).log\left(1-p(y_i)\right)\right]$$

Where:

- N is the number of data points
- y_i is the actual label (0 or 1) for the i-th data point
- $p(y_i)$ is the predicted probability that the i-th data point belongs to class 1

5.3.7 Dlib

For Drowsiness Detection, the project utilized Dilib, a pre-trained library specifically designed for detecting 68 facial landmarks.

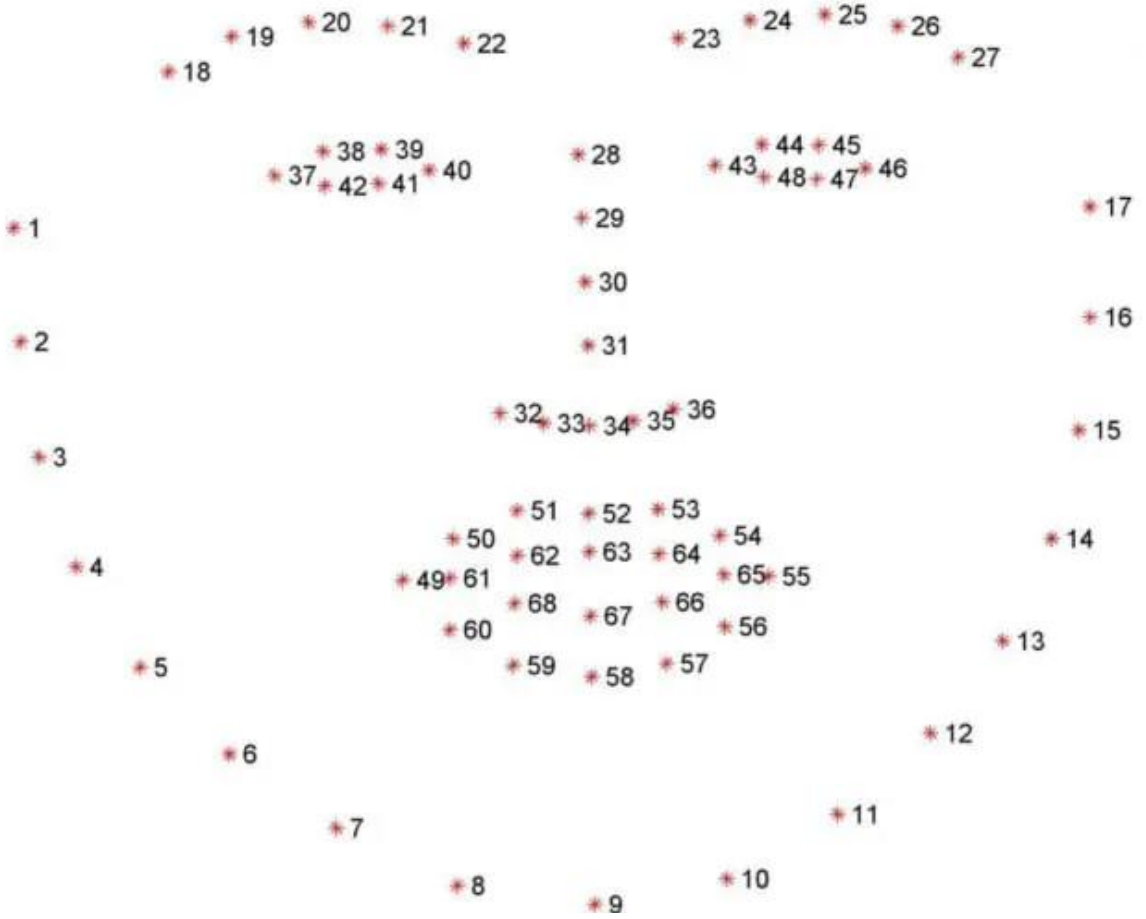

Figure 5.7: Dlib facepoints

[Source: https://www.studytonight.com/post/dlib-68-points-face-landmark-detection-with-opencv-and-python]

5.3.8 Facial Feature Detection

In the drowsiness detection phase, following the acquisition of facial landmark data from the Dlib library, the project focused on utilizing the Eye Aspect Ratio (EAR) and Mouth Opening Ratio (MAR) as the primary parameters for identifying signs of drowsiness.

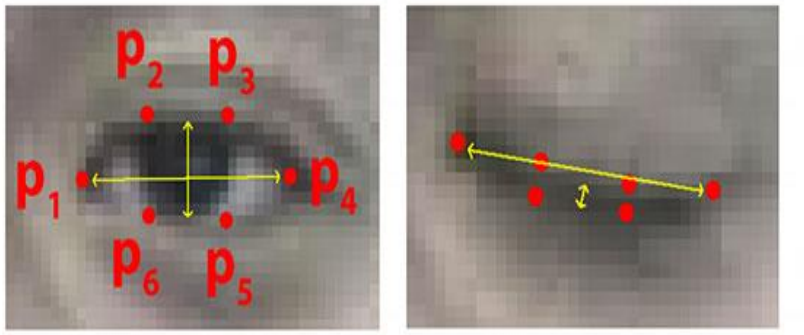

Figure 5.8: Points used for EAR Calculation

[Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Flink.springer.com%2Fchapter%2F10.1007%2F978-981-16-5987]

$$\boldsymbol{EAR = \frac{|| P2 - P6 || + || P3 - P5 ||}{2 \times || P1 - P4 ||}}$$

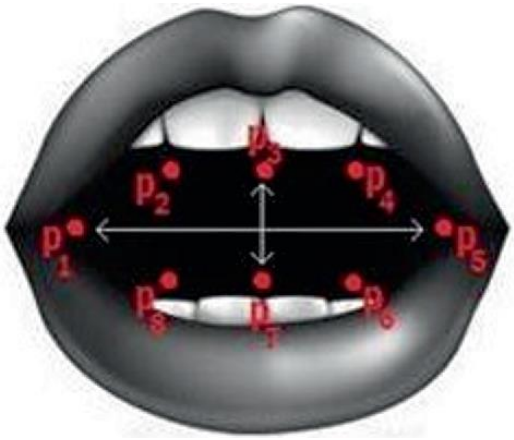

Figure 5.9: Points used for MAR Calculation

[Source: https://www.mdpi.com/2076-3417/12/4/2224]

$$\boldsymbol{MAR = \frac{|| P2 - P8 || + || P3 - P7 || + || P4 - P6 ||}{2 \times || P1 - P5 ||}}$$

5.3.9 Hyperparameters Used for Model Training

Table 5.1: Hyperparameters used for Model Training

S. N	Task	Model	Activation Function	Optimizer	Learning Rate
1	Distraction Detection	CNN	Relu,Softmax	Rmsprop	0.001
2	Distraction Detection (Ensemble Averaging)	Xception	Relu,Softmax	SGD	0.001
		Resnet	Relu,Softmax	SGD	0.001
		Mobilenet	Relu,Softmax	SGD	0.005
3	Drowsiness Detection	CNN	Sigmoid,Relu	Adam	0.001

CHAPTER 6: RESULTS AND ANALYSIS

6.1 Distraction Detection

Initially, a distraction detection system was developed using a CNN model. While it demonstrated excellent performance on the State Farm Distraction Detection training and testing datasets, it proved unsuitable for real-life datasets. Subsequently, an ensemble averaging approach was adopted, integrating ResNet, Xception, and MobileNet models. This strategy leveraged the unique characteristics of each model to get more accurate result.

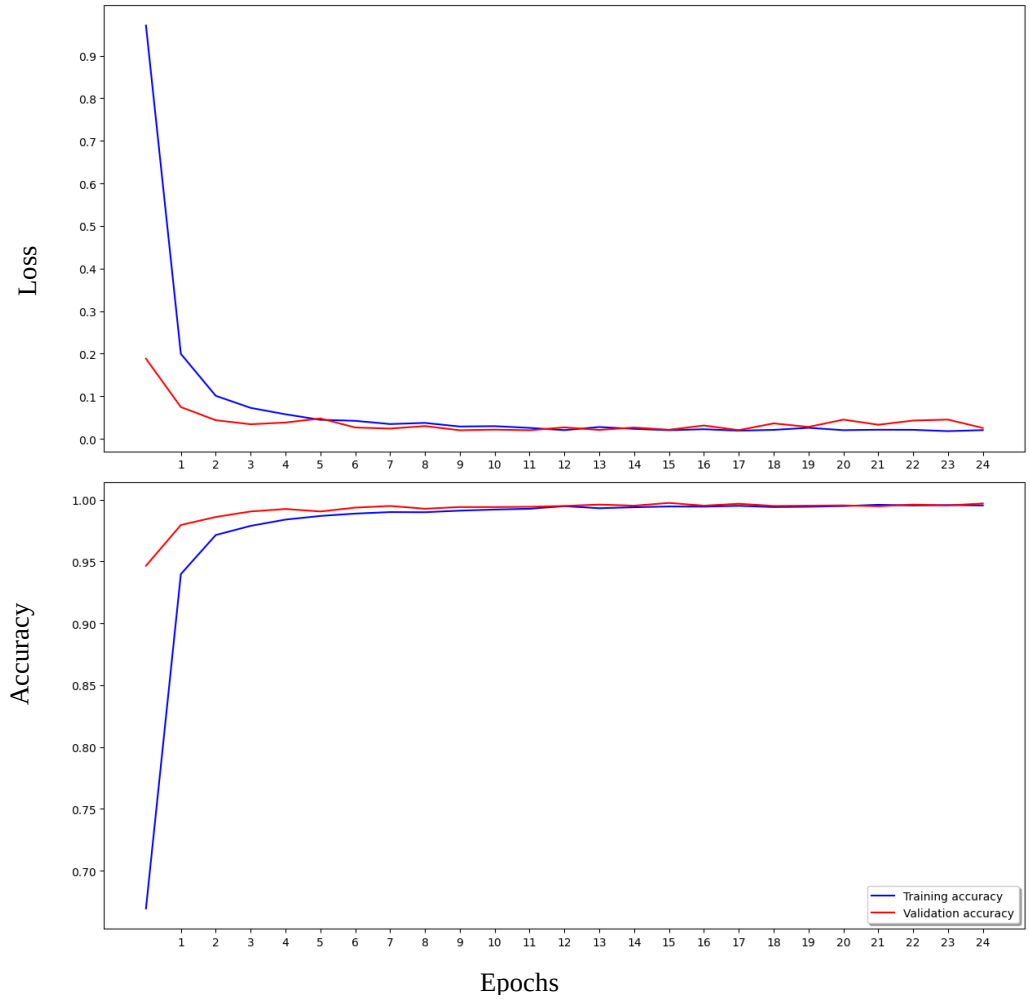

Figure 6.1: CNN training and validation graph

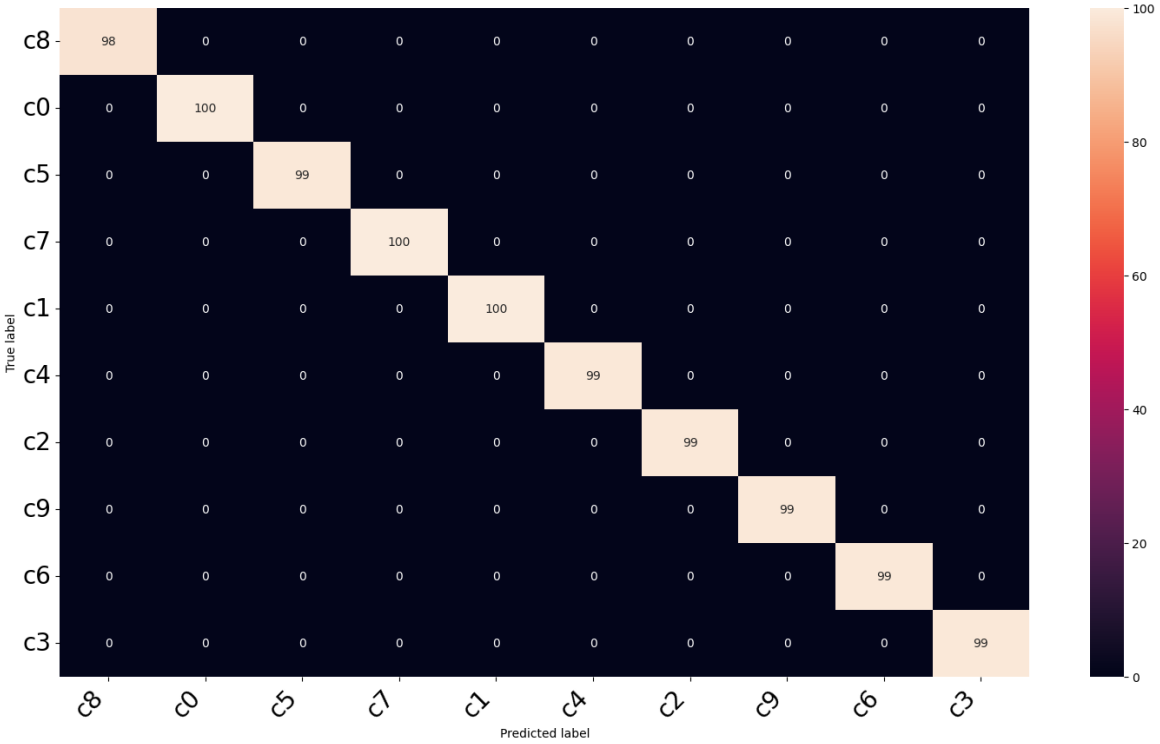

Figure 6.11: Confusion Matrix for CNN

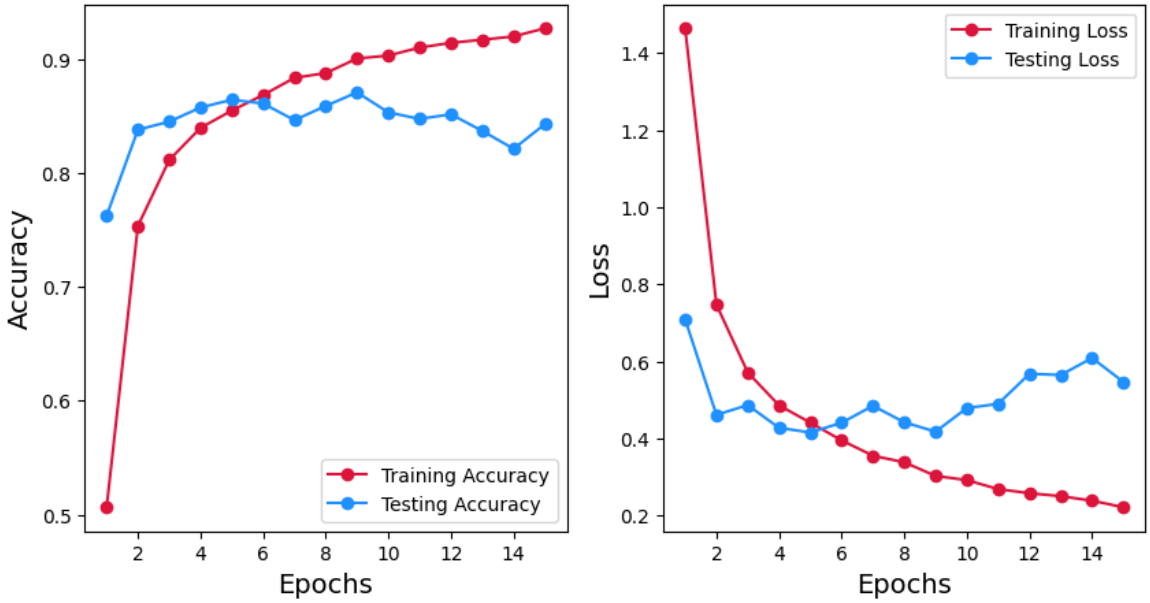

Figure 6.12: MobileNet

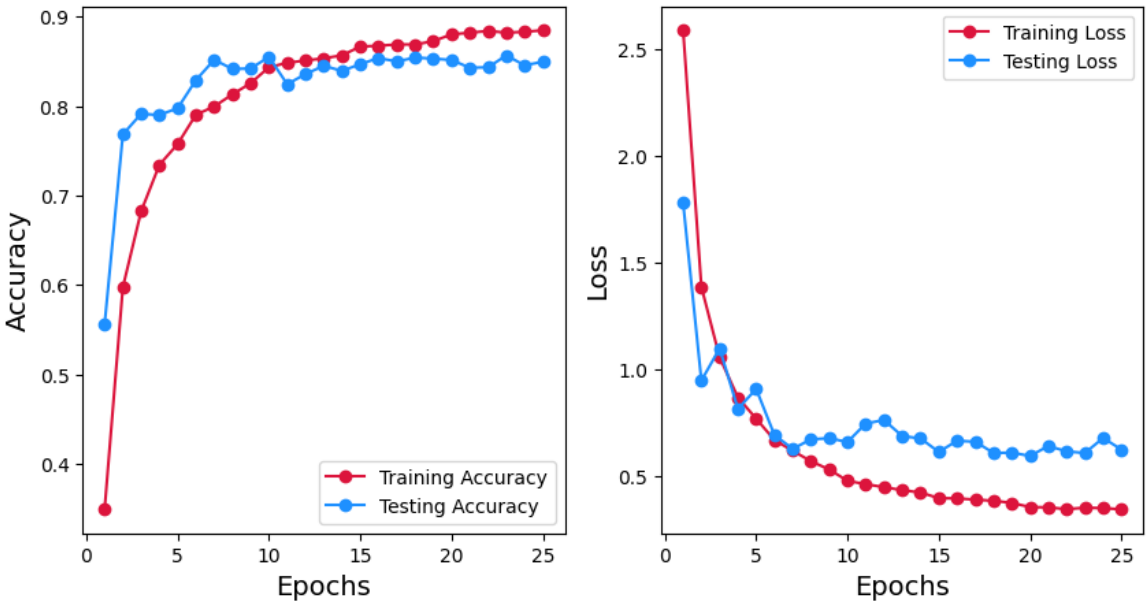

Figure 6.13: Resnet

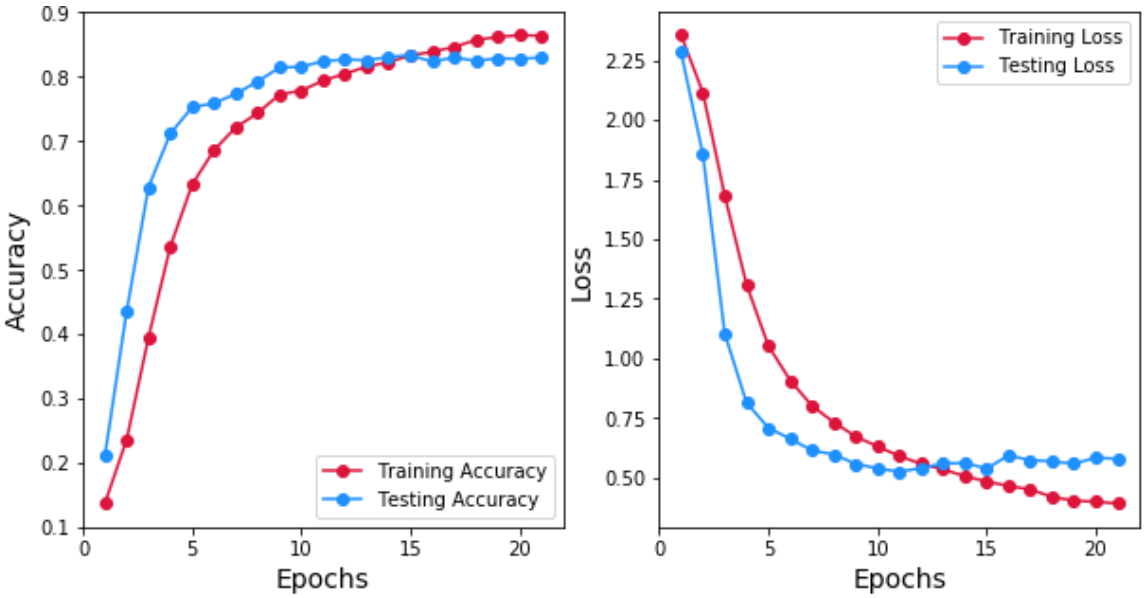

Figure 6.14: Exception

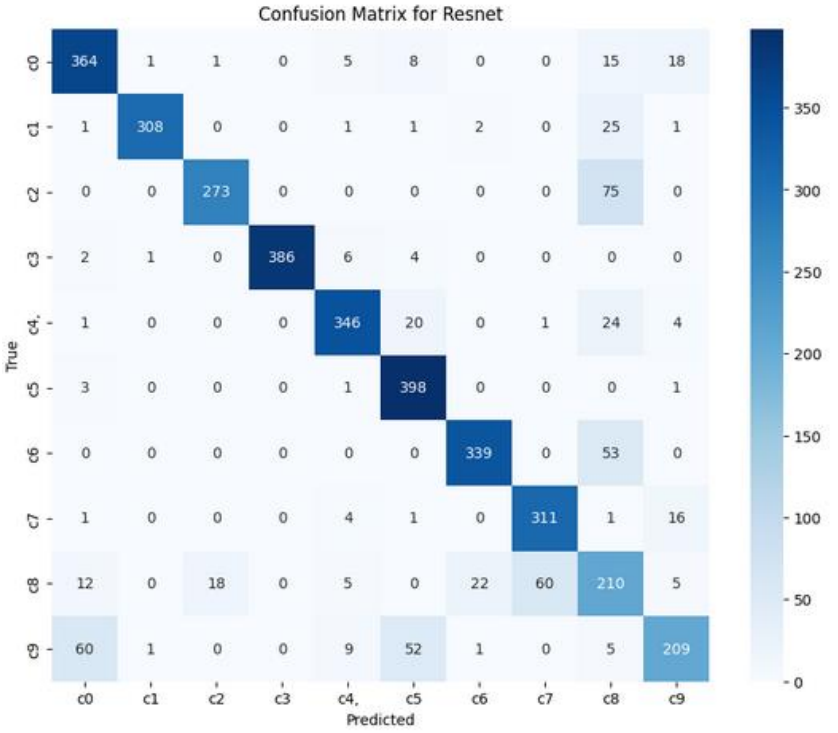

Figure 6.6: Confusion Matrix for ResNet

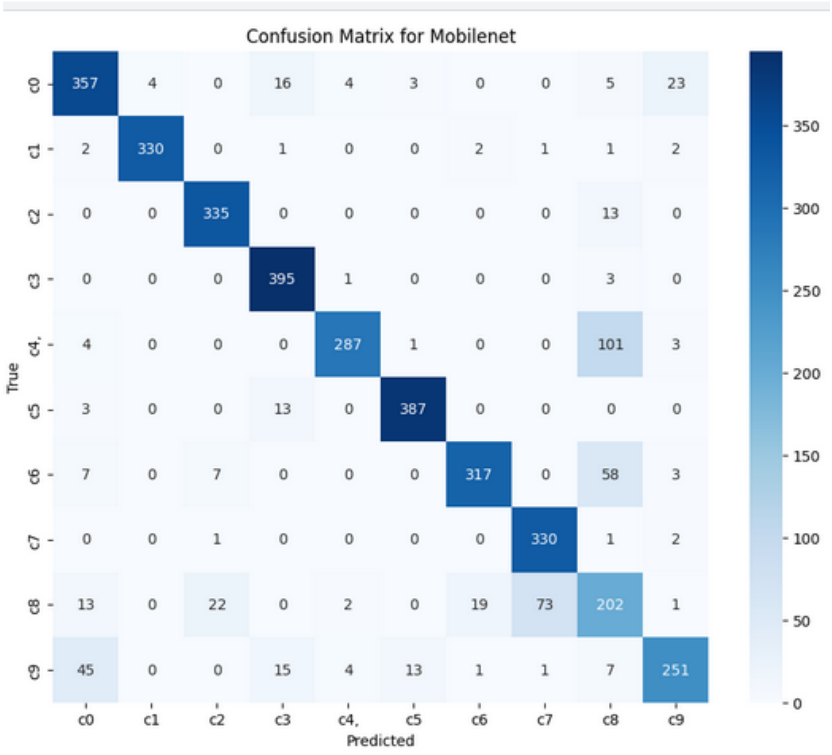

Figure 6.7: Confusion Matrix for MobileNet

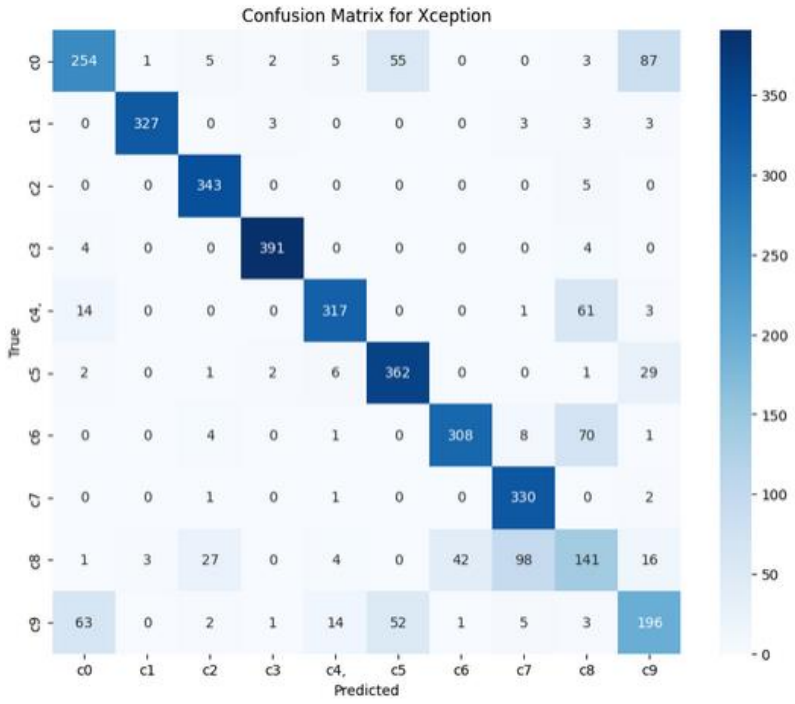

Figure 6.8: Confusion Matrix for Xception

Table 6.11: Comparison of Different Models

Model	Val Accuracy	Log loss
Exception Model	83.02	0.53
ResNet 50 Model	84.97	0.55
MobileNet Model	84.29	0.4
Ensembling	88	0.35

Figure 6.9: Safe Driving Detected

Figure 6.10: Drinking Detected

Figure 6.11: Texting Right Detected

6.2 Drowsiness Detection

A drowsiness detector was developed with the assistance of Dlib, a pre-trained library capable of detecting 68 facial points. Following this, the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) were computed. Subsequently, a CNN model was employed to discern the drowsiness state of person.

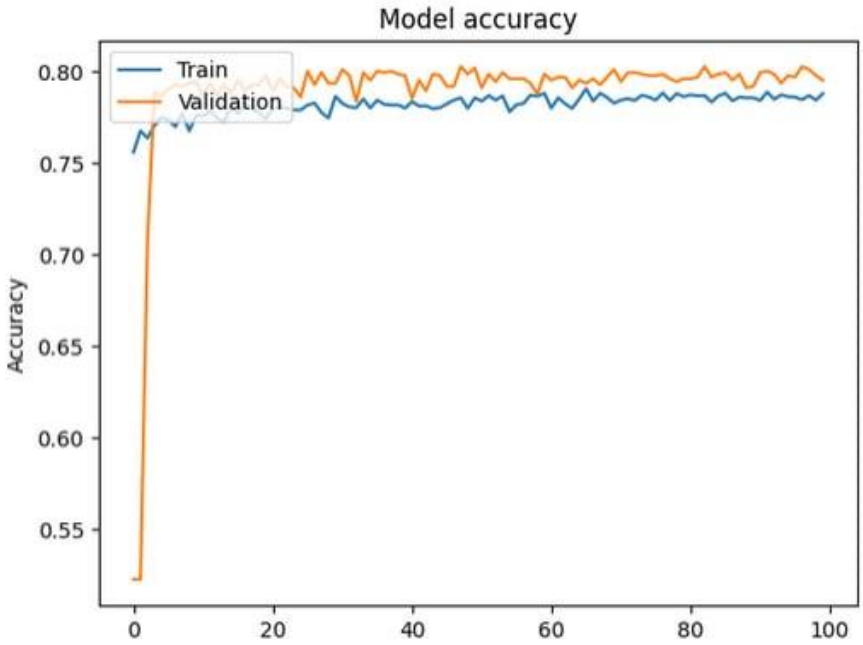

Figure 6.12: CNN training accuracy and validation accuracy

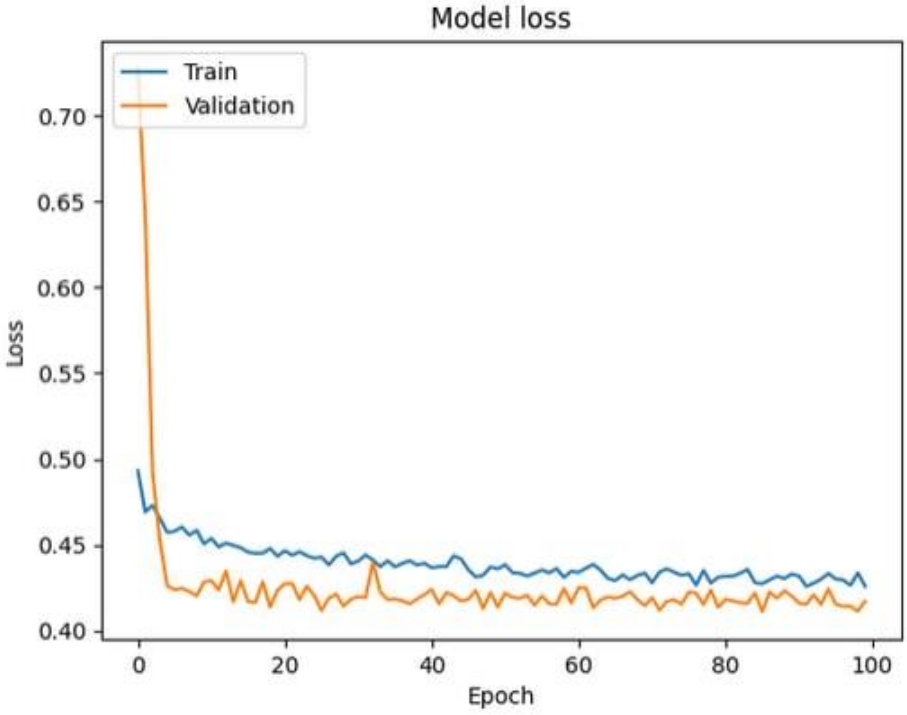

Figure 6.13: CNN training loss and validation loss

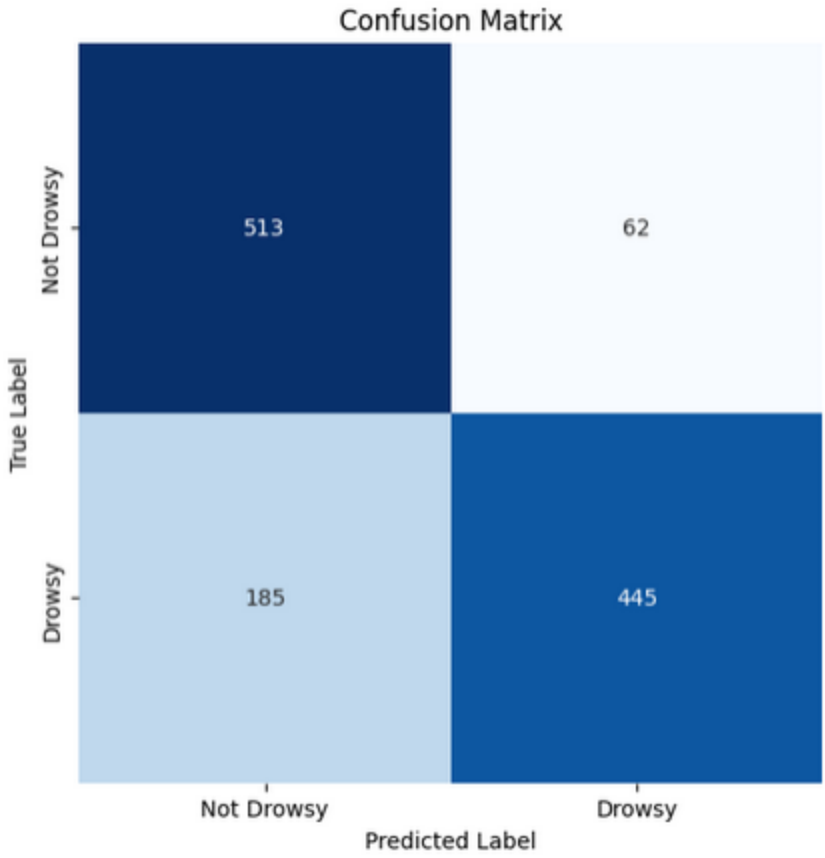

Figure 6.14: Confusion Matrix for CNN

Figure 6.215: Drowsiness Detected

CHAPTER 7: CONCLUSION LIMITATIONS AND FUTURE ENHANCEMENT

7.1 Conclusion

The driver state detection system presented in this project signifies a significant breakthrough in addressing the critical issues of driver distraction and drowsiness, both prominent factors contributing to road accidents. By incorporating sophisticated distraction and drowsiness detection modules driven by deep neural networks, the system showcases remarkable accuracy in real-time identification of distractions and signs of drowsiness. This proactive system facilitates prompt alerts to drivers, thereby enhancing their vigilance and promoting safer driving practices. Through the amalgamation of intelligent technologies, this project not only strives to reduce accidents but also aims to cultivate a culture of road safety, ultimately paving the way for safer roads and communities.

7.2 Limitations

The system requires two cameras positioned strategically: one in front of the driver and another positioned at the side end above the door. This setup allows for the detection of both drowsiness and distraction states of the driver. It's crucial that the cameras maintain a fixed position for the model to function effectively, as the system is optimized to detect driver states from specific angles within the car.

It's important to note that the system may encounter challenges in accurately detecting drowsiness in drivers wearing glasses. Due to the potential obstruction of facial features by the glasses, the system's accuracy in drowsiness detection may be compromised in such cases.

7.3 Future Enhancement

For future enhancements, the project team plans to integrate the model into physical hardware, complete with a camera and Raspberry Pi. Additionally, efforts will be directed towards optimizing the model for efficient operation on lower-end hardware devices.

REFERENCES

[1] World Health Organization (WHO), "Road Traffic Injuries," 20 June 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. [Accessed 8 June 2023].

[2] World Health Organization (WHO), "Global Status Report on Road Safety," 17 June 2018. [Online]. Available: https://www.who.int/publications/i/item/9789241565684. [Accessed 8 June 2023].

[3] S. Saleem, "Risk assessment of road traffic accidents related to sleepiness during driving: A systematic review," Eastern Mediterranean Health Journal (EMHJ), vol. 28, no. 9, 2022.

[4] C. Streiffer, R. Raghavendra, T. Benson and a. M. Srivatsa, "Darnet: A deep learning solution for distracted driving detection," in 8th ACM/IFIP/USENIX Middleware Conference: Industrial Track, Las Vegas, NV, USA, December 11-15, 2017.

[5] H. M. Eraqi, Y. Abouelnaga, M. H. Saad and M. N. Moustafa, "Driver distraction identification with an ensemble of convolutional neural networks," Journal of Advanced Transportation, vol. 2019, p. Article ID 4125865, 2019.

[6] A. Koesdwiady, S. M. Bedawi, C. Ou and F. Karray, "End-to-end deep learning for driver distraction recognition," in Proceedings of the International Conference Image Analysis and Recognition, Montreal, QC, Canada, 5-7 July 2017.

[7] M. Leekha, M. Goswami, R. R. Shah, Y. Yin and R. Zimmermann, "Are you paying attention? Detecting distracted driving in real-time," in Proceedings of the 2019 IEEE 5th International Conference on Multimedia Big Data (BigMM), Singapore, 11-13 September 2019.

[8] J. M. Celaya-Padilla, C. E. Galván-Tejada, J. S. A. Lozano-Aguilar, L. A. Zanella-Calzada, H. Luna-García, J. I. Galván-Tejada, N. K. Gamboa-Rosales, A. Velez

Rodriguez and H. Gamboa-Rosales, "Texting & driving detection using deep convolutional neural networks," Applied Sciences, vol. 9, p. 2960, 2019.

[9] A. Behera, A. Keidel and B. Debnath, "Context-driven Multi-stream LSTM (M-LSTM) for Recognizing Fine-Grained Activity of Drivers," in 40th German Conference on Pattern Recognition (GCPR) 2018, Stuttgart, Germany, October 9-12, 2018.

[10] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis and F. Y. and Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pp. 5379-5390, 2019.

[11] F. Omerustao, C. O. glu and S. G. Kar, "Distracted driver detection by combining in-vehicle and image data using deep learning," Applied Soft Computing Journal, vol. 96, p. 106657, 2020.

[12] Y. Liu, Y. Zhang, J. Li, J. Sun, F. Fu and J. and Gui, "Towards early status warning for driver's fatigue based on cognitive behavior models," in International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management, 2013.

[13] A. Gumaei, M. Al-Rakhami, M. M. Hassan, A. Alamri, M. Alhussein, A. Razzaque and G. Fortino, "A deep learning-based driver distraction identification framework over edge cloud," Neural Computing and Applications, pp. 1-16, 2020.

[14] M. Mandal, "Introduction to Convolutional Neural Networks (CNN)," 01 May 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/. [Accessed 08 June 2023].

[15] P. Raghav, "Medium," 4 March 2018. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Accessed 1 3 2024].

[16] K. .. Jhe, "Deep Residual Learning for Image Recognition,," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[17] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[18] A. G. Howard, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017. [Online]. Available: arXiv:1704.04861. [Accessed 2024].

APPENDIX: SOURCE CODE

Xception:

```
## Defining the input
from keras.layers import Input
xception_input = Input(shape = (224, 224, 3), name = 'Image_input')
from keras.applications.xception import preprocess_input, decode_predictions
from keras.applications.xception import Xception
#Get the  weights and layers
model_xception_conv = Xception(weights= 'imagenet', include_top=False, input_shape= (224,224,3))
from keras.models import Model
output_xception_conv = model_xception_conv(xception_input)
#Add the fully-connected layers
x=GlobalAveragePooling2D()(output_xception_conv)
x=Dense(1024,activation='relu')(x)
x = Dropout(0.1)(x)
x=Dense(1024,activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(512,activation='relu')(x)
x = Dense(10, activation='softmax', name='predictions')(x)
xception_pretrained = Model(inputs=xception_input, outputs=x)
xception_pretrained.summary()
sgd = optimizers.SGD(learning_rate=0.001)
xception_pretrained.compile(loss='categorical_crossentropy',optimizer = sgd,metrics=['accuracy'])
```

```
from keras.preprocessing.image import ImageDataGenerator

from keras.callbacks import ModelCheckpoint,EarlyStopping

checkpointer = ModelCheckpoint('xception_weights_aug_extralayer_alltrained_sgd2_V2.hdf5', verbose=1, save_best_only=True)

earlystopper = EarlyStopping(monitor='val_loss', patience=10, verbose=1)

datagen = ImageDataGenerator(

  height_shift_range=0.5,

  width_shift_range = 0.5,

  zoom_range = 0.5,

  rotation_range=30

    )

data_generator = datagen.flow(X_train, y_train, batch_size = 64)

xception_model = xception_pretrained.fit_generator(data_generator,steps_per_epoch = len(X_train) / 64, callbacks=[checkpointer, earlystopper],epochs = 30, verbose = 1, validation_data = (X_test, y_test))
```

Resnet Model

```
# Defining the input

from keras.layers import Input

resnet50_input = Input(shape = (224, 224, 3), name = 'Image_input')

## The RESNET model

from keras.applications.resnet50 import preprocess_input, decode_predictions

from keras.applications.resnet50 import ResNet50

#Get the RESNET weights and layers

model_resnet50_conv = ResNet50(weights= 'imagenet', include_top=False, input_shape= (224,224,3))
```

```
model_resnet50_conv.summary()

from keras.models import Model

output_resnet50_conv = model_resnet50_conv(resnet50_input)

#Add the fully-connected layers

x = Flatten(name='flatten')(output_resnet50_conv)

x = Dense(10, activation='softmax', name='predictions')(x)

resnet50_pretrained = Model(inputs=resnet50_input, outputs=x)

resnet50_pretrained.summary()

# Compile CNN model

sgd = optimizers.SGD(lr = 0.001)

resnet50_pretrained.compile(loss='categorical_crossentropy',optimizer = sgd,metrics=['accuracy'])

from keras.preprocessing.image import ImageDataGenerator

from keras.callbacks import ModelCheckpoint,EarlyStopping

checkpointer = ModelCheckpoint('resnet_weights_aug_alltrained_sgd2_setval.hdf5', verbose=1, save_best_only=True)

earlystopper = EarlyStopping(monitor='accuracy', patience=7, verbose=1)

datagen = ImageDataGenerator(

  height_shift_range=0.5,

  width_shift_range = 0.5,

  zoom_range = 0.5,

  rotation_range=30

    )

data_generator = datagen.flow(X_train, y_train, batch_size = 64)

resnet50_model = resnet50_pretrained.fit_generator(data_generator,steps_per_epoch = len(X_train) / 64, callbacks=[checkpointer, earlystopper,lrate],epochs = 25, verbose = 1, validation_data = (X_test, y_test))
```

Mobilenet

```
base_model=MobileNet(weights='imagenet',include_top=False)
x=base_model.output
x=GlobalAveragePooling2D()(x)
preds=Dense(10,activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=preds)
model.summary()
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint,EarlyStopping
checkpointer = ModelCheckpoint('mobilenet_sgd_nolayers.hdf5', verbose=1, save_best_only=True)
earlystopper = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
datagen = ImageDataGenerator(
  height_shift_range=0.5,
  width_shift_range = 0.5,
  zoom_range = 0.5,
  rotation_range=30
    )
data_generator = datagen.flow(X_train, y_train, batch_size = 64)
mobilenet_model = model.fit_generator(data_generator,steps_per_epoch = len(X_train) / 64, callbacks=[checkpointer, earlystopper],epochs = 25, verbose = 1, validation_data = (X_test, y_test))
```

CNN model for distraction

```
model = Sequential()
```

```
model.add(Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(128,128,3), kernel_initializer='glorot_normal'))

model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(filters=128, kernel_size=2, padding='same', activation='relu', kernel_initializer='glorot_normal'))

model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(filters=256, kernel_size=2, padding='same', activation='relu', kernel_initializer='glorot_normal'))

model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(filters=512, kernel_size=2, padding='same', activation='relu', kernel_initializer='glorot_normal'))

model.add(MaxPooling2D(pool_size=2))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(500, activation='relu', kernel_initializer='glorot_normal'))

model.add(Dropout(0.5))

model.add(Dense(10, activation='softmax', kernel_initializer='glorot_normal'))

model.summary()
```

CNN model for drowsiness

```
import cv2

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, classification_report

from tensorflow.keras import Sequential
```

```
from tensorflow.keras.layers import Conv2D, Flatten, Dense, BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam
# Split the data into features (X) and labels (y)
X = df.iloc[:, :-1].values
y = df['drowsy'].values
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize the input data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Reshape the data for CNN input (assuming each sample has 2 features)
X_train = X_train.reshape(X_train.shape[0], 1, 2, 1)
X_test = X_test.reshape(X_test.shape[0], 1, 2, 1)
# Create the CNN model
model = Sequential()
model.add(Conv2D(512, kernel_size=(1, 2), activation='relu', input_shape=(1, 2, 1)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(256, activation='relu'))
Dropout(0.5)
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))
```

```
# Compile the model
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
# Display the model summary
model.summary()
# Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))
```