Don Bosco Institute of Technology

Kumbalgodu, Mysuru Road, Bengaluru-560 074

Report

On

"Distracted Driver Detection"

Internship carried out in the

Department of Computer Science and Engineering

Under DBIT-SIC

Submitted by

Shashank Gowda R (1DB20CS098)

Usha N (1DB20CS121)

Under the guidance of

External Guide	**Internal Guide**
Mr. Prasad K	Dr. Venugeetha Y
Trainer	Professor, Coordinator – SIC
SIC	Dept. of CSE DBIT, Bengaluru

Held at

DBIT-SIC Laboratory

2022 – 2023

Don Bosco Institute of Technology

Kumbalagodu, Mysore Road, Bengaluru - 560074

www.dbit.co.in ph: +91-80-28437028/29/30 Fax: +91-80-28437031

Department of Computer Science and Engineering

Under DBIT-SIC

Ref # DBIT-SIC-AI / B1 / 2023 Date:

INTERNSHIP CERTIFICATE

TO WHOM IT MAY CONCERN

*This is to certify that **Mr. Shashank Gowda R** of the Department of Computer Science and Engineering bearing USN **1DB20CS098** has successfully completed an internship program in the **Artificial Intelligence** domain from 14th August 2023 to 9th September 2023 with a capstone Project of 90 hours followed by a presentation.*

Co-ordinator SIC **HOD CSE** **Principal**

Don Bosco Institute of Technology

Kumbalagodu, Mysore Road, Bengaluru - 560074

www.dbit.co.in ph: +91-80-28437028/29/30 Fax: +91-80-28437031

Department of Computer Science and Engineering

Under DBIT-SIC

Ref # DBIT-SIC-AI / B1 / 2023 / 001 Date:

INTERNSHIP CERTIFICATE

TO WHOM IT MAY CONCERN

*This is to certify that **Ms. Usha N** of the Department of Computer Science and Engineering bearing USN **1DB20CS121** has successfully completed an internship program in the **Artificial Intelligence** domain from 14th August 2023 to 9th September 2023 with a capstone Project of 90 hours followed by a presentation.*

Co-ordinator SIC **HOD CSE** **Principal**

ABSTRACT

This research leverages computer vision and deep learning techniques to achieve real-time identification of distractions and tiredness in the context of driving safety improvement. A multifaceted system has been developed, encompassing drowsiness detection, hands-on-the-wheel detection, and drive distraction detection. Each of these elements plays a pivotal role in sustaining the driver's focus and attentiveness throughout the journey.

To identify **drive distractions**, a deep learning model has been trained. Visual data from both the vehicle's surroundings and the driver are collected and prepared for analysis. The model, constructed using TensorFlow and Keras, is designed to recognize potential distractions in the video feed, such as phone usage and other inattentive behaviors. This technology is crucial for classifying distractions and providing real-time feedback to the driver.

Hand detection on the steering wheel is another essential feature of this system. It monitors the driver's hand position and whether they are on the steering wheel using OpenCV and the 'cvzone' module. The system actively engages with the driver by issuing alerts and directives based on the movements and gestures of the driver's hands. For example, it suggests appropriate speeds based on the detected hand positions and provides guidance on when to start the engine.

The system also enhances safety through **drowsiness identification** using facial landmark identification from the Dlib package. It continuously tracks the driver's eyes and analyzes blink patterns to assess their level of alertness. If drowsiness is detected, the system issues real-time warnings to ensure the driver remains awake and attentive to the road.

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1

INTRODUCTION

1.1 Existing System

Drive distraction detection mainly relies on manual observation by the driver and law enforcement in the absence of automated technology. Drivers are required to identify and reduce distractions on their own. Reports and visual cues are used by law enforcement to spot distracted driving.

Smartphone users can access a variety of mobile applications that promise to identify and discourage distracted driving. These programs employ the phone's sensors to find motion and other potential distractions. Their precision and potency can change, though.

Limitations of the existing system

- **Subjectivity:** Manual observation relies on the driver's awareness of his or her own actions and the officer's perception of the situation. It could be inconsistent and subjective.
- **Limited Accuracy:** It's possible that mobile applications and vehicle-based systems don't have the intelligence and accuracy necessary to detect tiny diversions and sleepiness patterns.
- **False Positives and Negatives:** Wearable technology may produce false alerts that cause unneeded diversions or fail to effectively detect tiredness.
- **Scalability:** Many drivers may not have access to these safety measures if these technologies are not broadly adopted and integrated into all vehicles.

1.2 Problem Statement

Road accidents are significantly influenced by distracted driving and driver fatigue, which is a serious danger to road safety. Distractions like using a phone, eating, or not paying attention still put lives in danger in spite of public awareness campaigns and legal prohibitions. Similar to this, driving when fatigued can result in accidents, especially on lengthy or late-night trips. The current approaches to these problems rely on subjective judgment, manual observation, and occasionally faulty technology.

There are three main components to the issue:

- The first step is to identify and reduce distractions while driving, such as using a phone or engaging in other inattentive activities. Distractions can take a driver's focus off the road, greatly raising the chance of an accident. There is a need for an automated system that can precisely identify distractions and give the driver feedback in real-time because current approaches are frequently manual and reactive.

- The second concern is driver fatigue detection, which can result in attention lapses and potentially catastrophic accidents, particularly on lengthy rides or at night. The accuracy and range of current sleepiness detection technologies are constrained, leaving a gap in the safety net that should keep drivers awake and concentrated while driving.

- The third component of the issue is the detection of the driver's hands on the steering wheel and the provision of guidance for their placement. The right hand placement is essential for keeping the car under control. The current systems for this, especially for young or inexperienced drivers, are frequently simplistic and lack real-time advice.

1.3 Objectives

- **Distraction Detection:** Create a deep learning model to automatically identify and categorize distractions from real-time video data, such as using a phone, eating, or being inattentive. Accurately distinguish between paying attention and being distracted when driving. When distractions are found, give the driver immediate input, such as visual or aural alarms.
- **Drowsiness Detection:** Implement a system for detecting drowsiness that continuously tracks the driver's eye blink patterns to spot the first signs of fatigue. Create a trustworthy algorithm that can tell the difference between the alert, tired, and asleep phases. When drowsiness is discovered, provide the driver with prompt warnings to help them stay awake and concentrated.
- **Detection of Hands on the Wheel:** Create a hands-detection system that uses computer vision to track the position of the driver's hand on the wheel. Make sure the system warns and guides the driver in real-time so they can keep their hands where they need to be for safe vehicle control. Provide detailed instructions for operations like igniting the engine and advising a suitable speed depending on hand placements.

- **Real-Time Reaction:** Integrate the technologies for sleepiness, hands, and attention detection to give the driver real-time feedback. Make sure the feedback is simple to utilize and doesn't cause too much distraction so the driver can keep their attention on the road.
- **Safety Improvement:** By proactively addressing distractions and fatigue, you can help to significantly reduce traffic accidents and injuries. Encourage the driver to adopt safe hand positioning techniques, especially for novice or inexperienced drivers, to increase road safety.
- **Integrity and Scalability:** Create a system that is flexible enough to be integrated into many kinds of vehicles, allowing it to be used by a variety of drivers. Assuring that the technology can be scaled for use in consumer or commercial cars would help to improve road safety for everyone.
- **Testing and Validation:** To ensure accuracy and dependability, thoroughly test the distraction, drowsiness, and hands detection modules. To assure the system's effectiveness, carry out rigorous validation and testing under diverse driving circumstances.
- **Future Planning:** Take into account the potential for upcoming improvements, including integration with advanced driver-assistance systems (ADAS) or self-driving cars. Investigate data analytics and reporting options to further increase driving safety.

Chapter 2

LITERATURE SURVEY

[1]. An Efficient Deep Learning Framework for Distracted Driver Detection - FAIQA SAJID

Developing a precise distracted driver detection framework is crucial to combat rising road accidents due to driver inattention. Leveraging the State Farm dataset with eight distinct distraction classes, we employ EfficientNet's transfer learning for robust feature extraction and utilize EfficientDet for accurate object detection and prediction. Our approach outperforms existing methods, with EfficientDetD3 achieving a remarkable 99.16% Mean Average Precision. This model, with specific parameter settings, can significantly enhance driver safety by identifying and mitigating distracted driving behaviors.

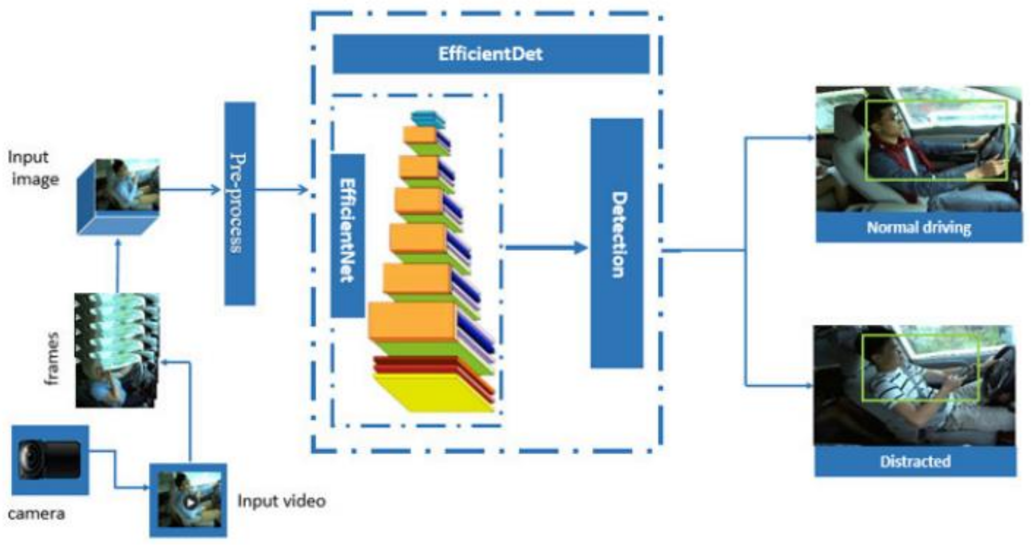


Fig 2.1 Proposed Model

[2]. Driver Distraction Detection Using Advanced Deep Learning Technologies Based on Images - Guofa Li

"This paper introduces an effective driver distraction detection method using transfer learning with deep learning algorithms. Utilizing three diverse datasets (State Farm, Origin, and 3MDAD), we evaluated the performance of AlexNet, VGG16, and ResNet18. Results demonstrate that ResNet18 on the Origin dataset achieved the best accuracy (99.99%), while AlexNet, VGG16, and ResNet18 scored 99.92%, 100%, and 99.99%, respectively, on the original dataset. The ResNet18-based method achieved accuracies of 84.82% and 97.7% on the State Farm and 3MDAD datasets. This research lays a strong foundation for implementing distraction detection applications in transportation systems.

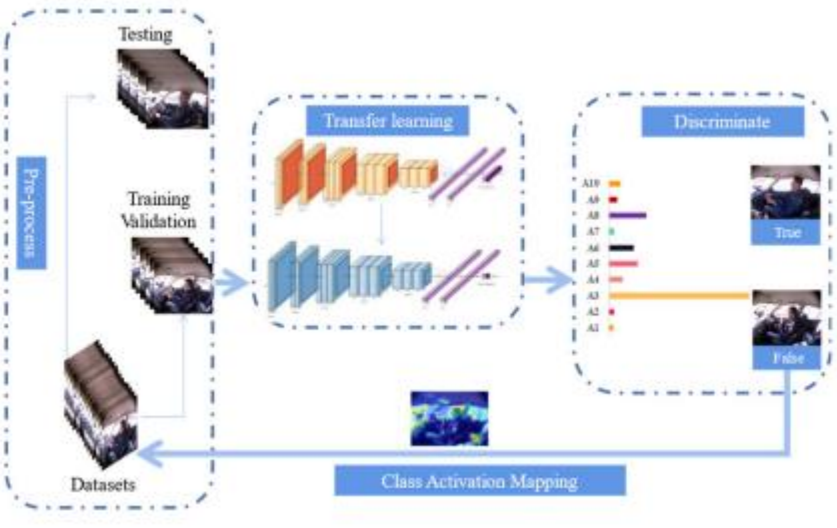


Fig 2.2 Proposed Model

[3]. Driver Distraction Detection Methods: A Literature Review and Framework - ALEXEY KASHEVNIK

This literature review highlights the critical issue of driver inattention and distraction as a major cause of road accidents. It emphasizes the need for information systems to detect these distractions. The paper provides a comprehensive framework by integrating various distraction detection methods, including manual, visual, and cognitive distractions. This framework visualizes the entire detection process, from data collection to behavior inference. It offers valuable insights for both researchers and developers working on distraction detection systems, particularly in the context of evolving automated driving.

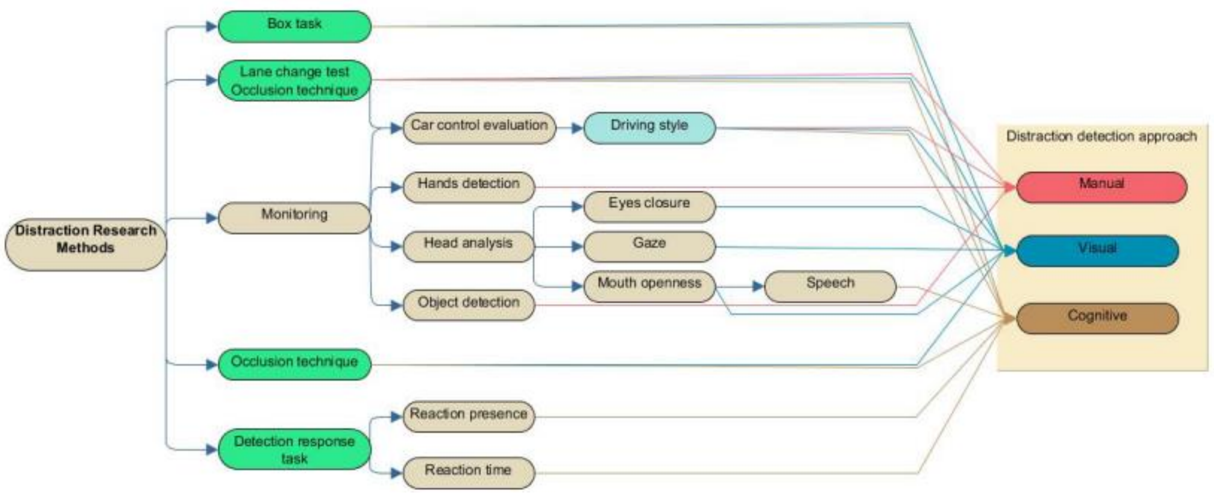


Fig 2.3 Proposed Model

[4]. Multimodal driver distraction detection using a dual-channel network of CNN and Transformer – Luntian Mou

To address the rising concern of distracted driving and enhance safety, we present a dual-channel feature extraction model combining CNN and Transformer networks. This model effectively leverages both local and global features from multi-source perception data. We optimize the model's fitting ability to time series data through a bilevel optimization approach. Extensive experiments on a multimodal driver distraction dataset demonstrate the superior performance of our method compared to existing approaches.

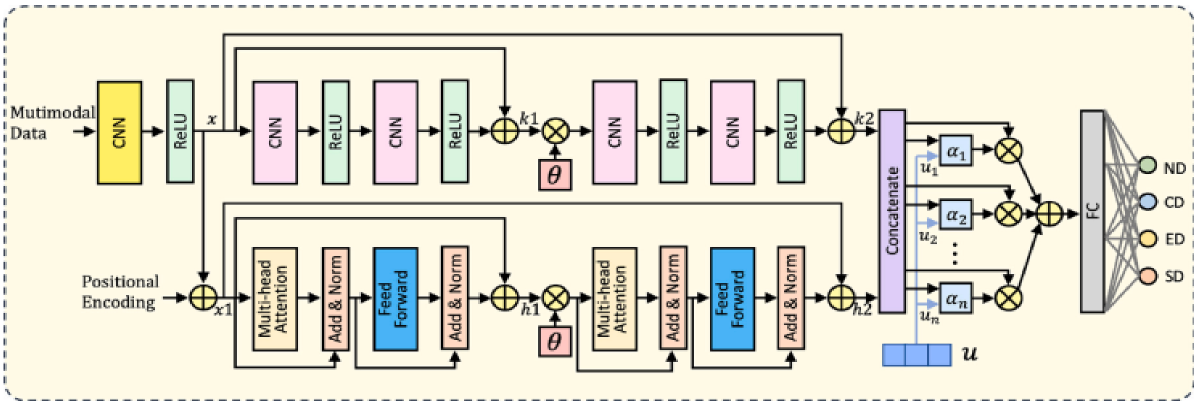


Fig 2.4 Proposed Model

[5]. Automatic driver distraction detection using deep convolutional neural networks - Md. Uzzol Hossain

"To address the global issue of road accidents caused by driver distraction, we present a CNN-based method capable of detecting diverse distracted driving behaviors. Leveraging transfer learning with CNN, VGG-16, ResNet50, and MobileNetV2 architectures, we trained the model on a substantial dataset. Results revealed that the pre-trained MobileNetV2 model exhibited the highest classification efficiency, underlining its potential in mitigating road accidents through distraction detection."

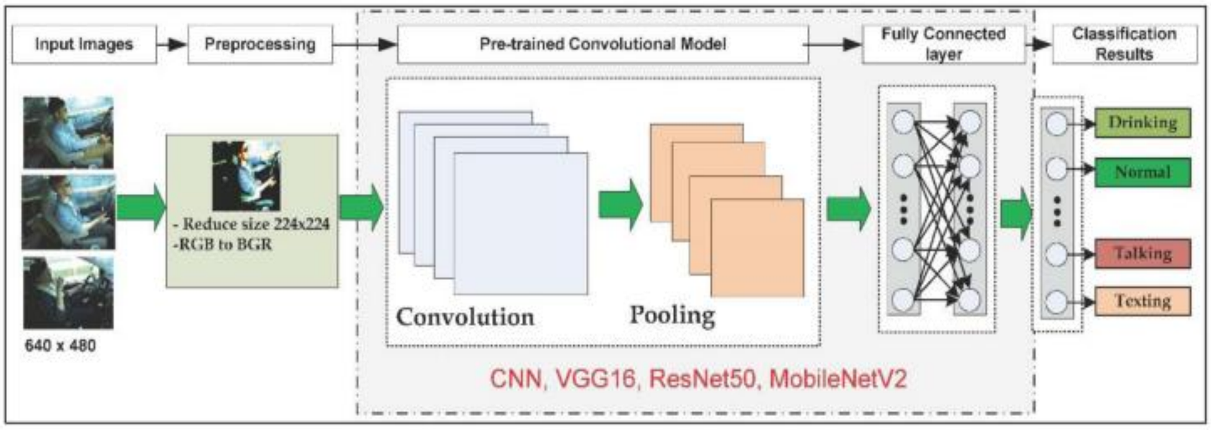


Fig 2.5 Proposed Model

[6]. Driver distraction detection via multi-scale domain adaptation network – Jing Wang

This paper addresses the pressing issue of distracted driving, a leading cause of road accidents. To enhance the generalization ability of distraction detection models affected by various factors, a multi-scale domain adaptation network (MSDAN) is introduced. MSDAN incorporates multi-scale convolution for feature extraction, domain adaptation for adaptability through adversarial training, and dropout for improved generalization. Extensive experiments on a large-scale driver distraction dataset validate the method's effectiveness in accurately detecting distractions and improving generalization performance, especially in cross-driver and cross-dataset scenarios.

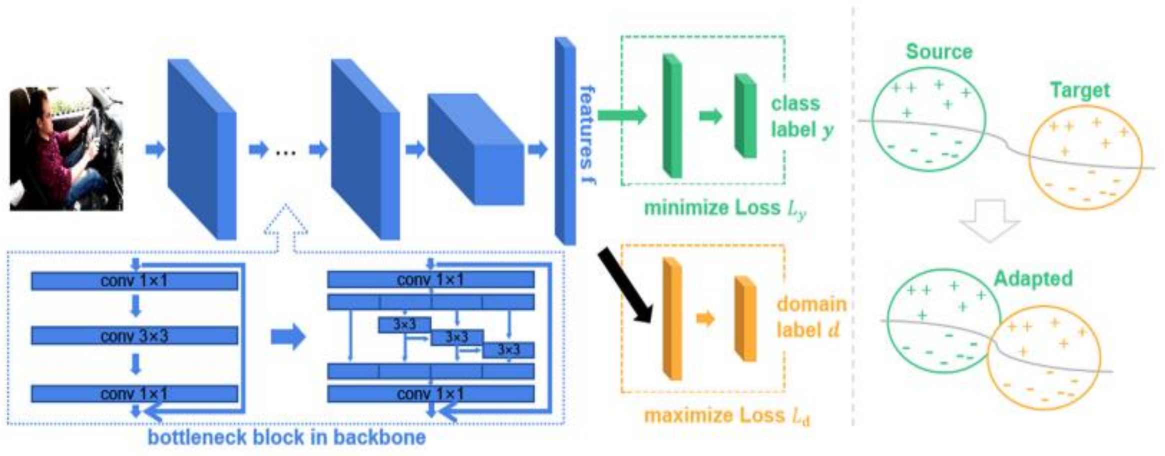


Fig 2.5 Proposed Model

Chapter 3

PROPOSED SYSTEM

3.1 Requirements

Hardware Requirements

Hard Disk	Minimum 512 GB SSD
RAM	Minimum 16 GB (To Support model Training)
GPU	NVIDIA GPU with CUDA

Software Requirements

Operating System	Windows, Linux, MacOS, or any other OS which supports python
Programming Language	Python – 3.10 (To support TensorFlow)
IDE	VS Code(1.81.1) or Google Collab

3.2 Libraries

NumPy

- NumPy is a fundamental library for numerical and array operations in Python. It provides support for multi-dimensional arrays and various mathematical functions.
- NumPy is used for the efficient handling and processing of numerical data and arrays in various aspects of your project.

Pandas

- Pandas is a Python library for data manipulation and analysis. It offers data structures like DataFrames and Series, making it easier to work with structured data.
- Pandas are used for managing and processing data, especially for data preprocessing and analytics.

os

- The 'os' module is part of Python's standard library and provides functions to interact with the operating system. It allows us to perform various file and directory operations.
- 'os' is used for tasks such as file manipulation, directory traversal, and handling file paths in our project.

cv2 (OpenCV)

- OpenCV (Open Source Computer Vision Library) is a comprehensive library for computer vision tasks, including image and video processing, object detection, and feature extraction.
- OpenCV is essential for face and eye detection, hand tracking, and image preprocessing.

TensorFlow

- TensorFlow is a popular deep-learning framework developed by Google. It is widely used for creating and training machine learning and deep learning models.
- TensorFlow is used to build and train the distraction detection model in your project.

Adam

- Adam is an optimization algorithm used in training deep learning models. It is an extension of stochastic gradient descent (SGD) and is effective for minimizing loss functions during model training.
- Adam is utilized as the optimization algorithm when compiling the deep learning model in your project.

image_dataset_from_directory

- 'image_dataset_from_directory' is a function provided by TensorFlow that simplifies the process of creating image datasets from directories. It's used for data preparation in deep learning tasks.
- This function is used to load and preprocess image data for training and validation.

Conv2D, MaxPooling2D, Dense, Flatten, Rescaling, Dropout

- These are layers and operations commonly used in deep neural networks. 'Conv2D' and 'MaxPooling2D' are for convolution and pooling, 'Dense' is for fully connected layers, 'Flatten' reshapes data, 'Rescaling' normalizes pixel values, and 'Dropout' prevents overfitting.
- These layers and operations are integral to the design and architecture of your deep learning model.

EarlyStopping

- 'EarlyStopping' is a callback in Keras (a part of TensorFlow) that allows early stopping of model training when a monitored metric stops improving, preventing overfitting.
- It is employed to halt training when validation accuracy stops improving in your project.

joblib

- Joblib is a library for lightweight pipelining in Python. It's used for saving and loading machine learning models, especially for non-Python data types.
- 'joblib' may be used to save and load trained models or data preprocessing pipelines.

dlib

- Dlib is a C++ library for various computer vision tasks, including facial landmark detection, image processing, and machine learning.
- Dlib is critical for detecting facial landmarks and monitoring eye blink patterns for drowsiness detection.

face_utils

- 'imutils' is a library for simplifying computer vision and image processing tasks in OpenCV. 'face_utils' includes utility functions for working with facial features.
- Usage in Project: 'face_utils' is used to facilitate face landmark detection and analysis.

GlobalAveragePooling2D, ImageDataGenerator, MobileNet

- These are specific components and pre-trained models available in Keras/TensorFlow. 'GlobalAveragePooling2D' computes the average output, 'ImageDataGenerator' augments image data, and 'MobileNet' is a pre-trained deep learning model.
- They can be part of the deep learning model architecture, enhancing its capabilities.

cvzone

- 'cvzone' is a Python library designed to simplify computer vision tasks, making hand tracking and other computer vision tasks more accessible and user-friendly.
- 'cvzone' is used for hands detection on the steering wheel, streamlining the process of hand tracking and position monitoring.

HandTrackingModule

- 'HandTrackingModule' is a module from the 'cvzone' library that specializes in hand tracking and detecting hand gestures.

- Project: This module plays a central role in monitoring and guiding the driver's hand positioning on the steering wheel.

3.3 Dataset

It's a common scenario: the traffic signal switches to green, but the car ahead stays still. Sometimes, an ordinary-looking vehicle starts to veer unpredictably. As you drive past the distracted driver, the familiar sight emerges – someone fully engrossed in their smartphone, either scrolling through social media or deeply engaged in a conversation on their handheld device.

3.3.1 The Distraction Dilemma

According to the Centers for Disease Control and Prevention's (CDC) motor vehicle safety division, distracted driving is the silent menace on our roads. It's a stark reality that one in every five car accidents is caused by a distracted driver. This grim statistic translates to 425,000 people injured and 3,000 lives lost to the scourge of distracted driving every year.

3.3.2 A Vision for Safer Roads

Enter State Farm, a company committed not just to ensuring its customers but also to making the roads safer for everyone. With the aim of challenging the status quo and improving these alarming statistics, State Farm is taking a bold step. They're harnessing the power of technology to combat distracted driving.

3.3.3 The Challenge

The challenge is not just to classify drivers' behavior, but to do so with remarkable accuracy. The dataset provided contains a treasure trove of 2D dashboard camera images, capturing moments behind the wheel.

3.3.4 The Quest for Insight

What's the task at hand? We are tasked with identifying and classifying each driver's behavior. Are they driving attentively, ensuring their seatbelt is fastened, or perhaps taking a selfie with friends in the backseat? The goal is to unravel the stories behind these images, one snapshot at a time.

The Ten Classes

To navigate our journey, we will be working with ten distinct classes. Each class encapsulates a unique behavior, revealing the intricate tapestry of distracted driving. Let's take a glimpse at what these classes represent:

c0: Normal 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

These classes reflect the diverse range of behaviors exhibited by drivers behind the wheel. The challenge is to develop a solution that can not only distinguish between these classes but also contribute to safer roads by providing real-time insights into driver behavior.

Exploratory Data Analysis

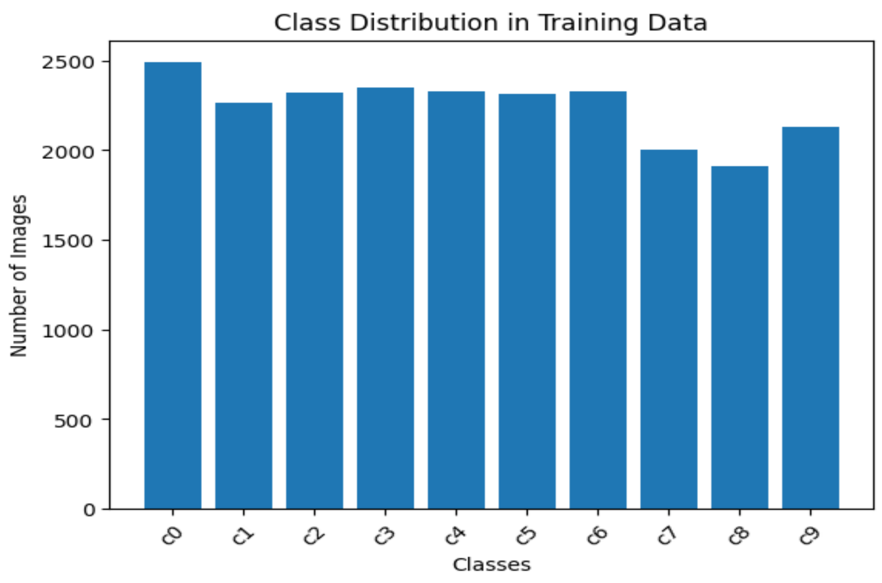
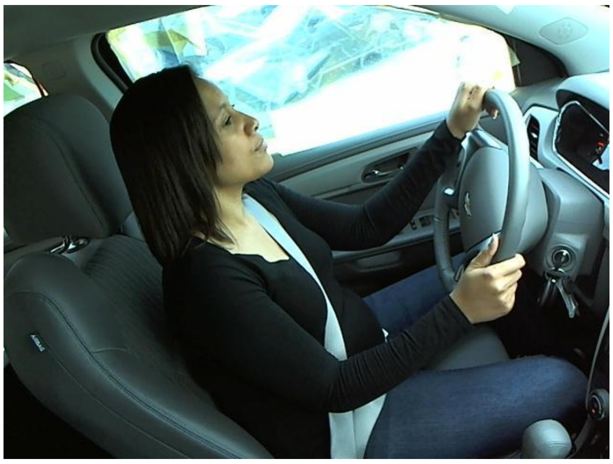
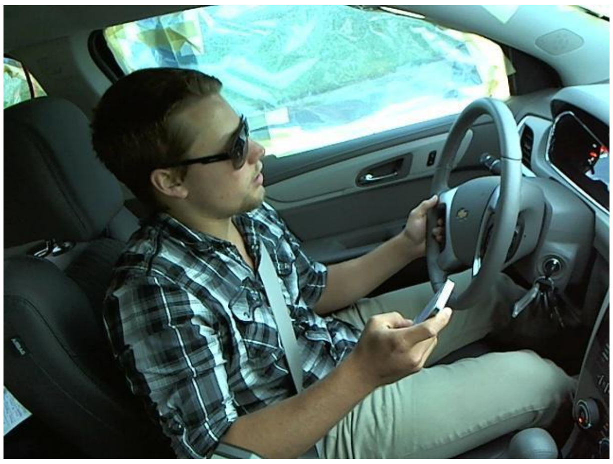
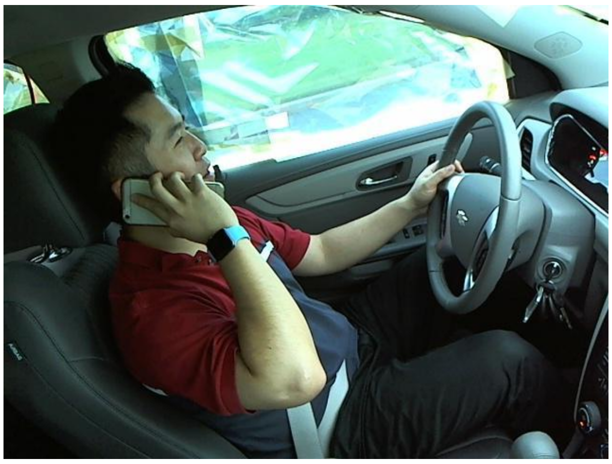


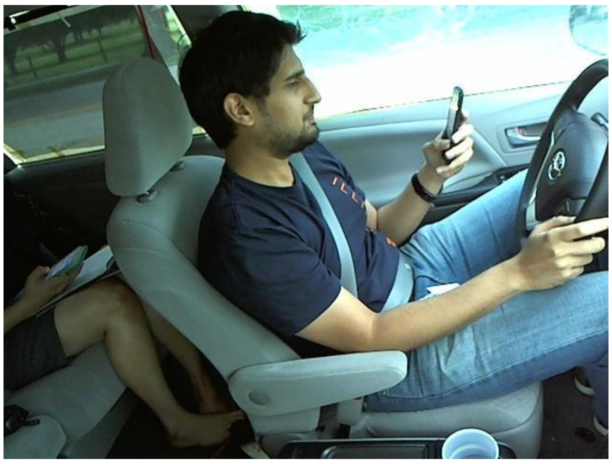
Fig 3.1

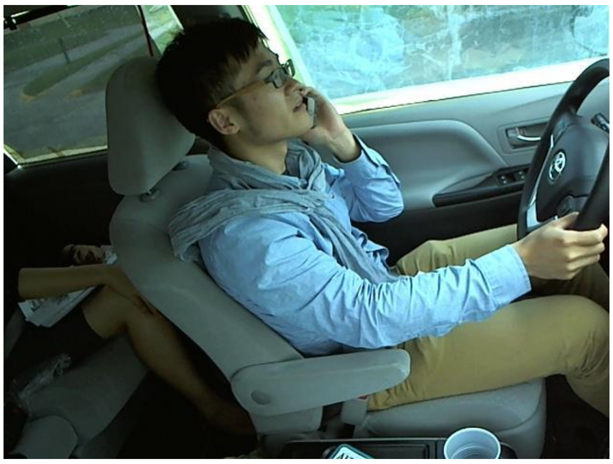
Overview of images in the dataset

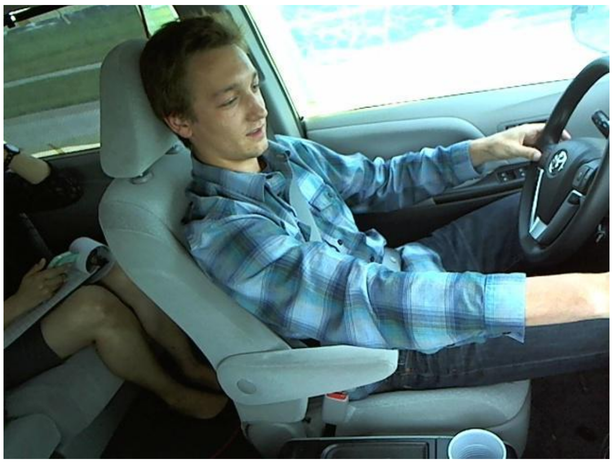

c0: Driver driving safely (Normal Driving)

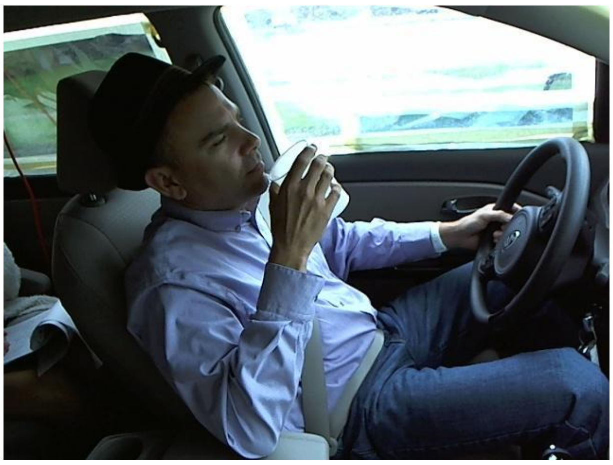

c1: Driver texting and driving (Texting – Right)

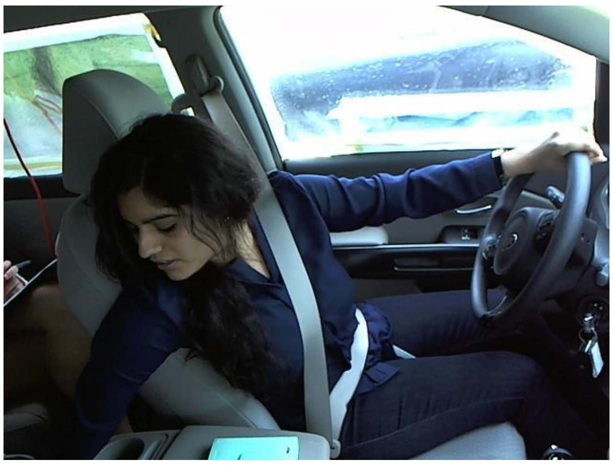

c2: Driver talking on the phone and driving (Talking on the Phone Right)

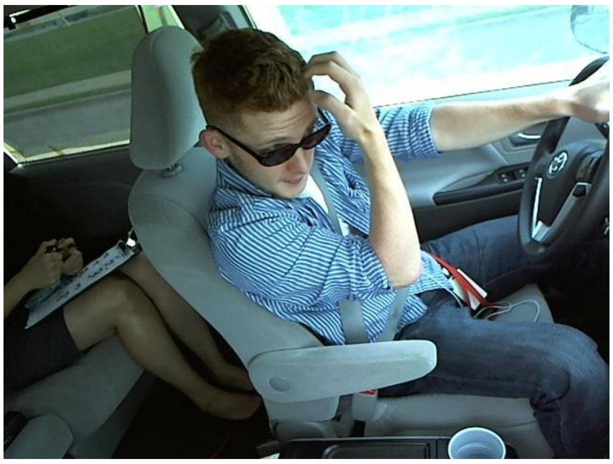

c3: Driver texting and driving (texting Left)

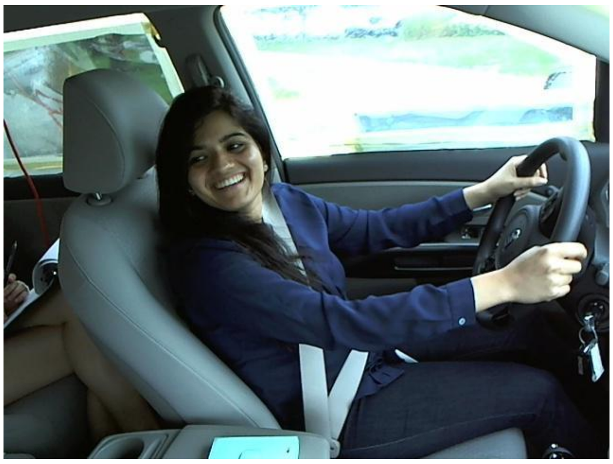

c4: Driver talking on the phone and driving (Talking on the Phone Left)


c5:Driver Operating radio and Driving (Operating the Radio)


c6: Driver drinking and driving (Drinking)


c7: Driver reaching behind and driving (Reaching Behind)


c8: Hair and Makeup


c9: Driver talking to passenger and driving (Talking to Passenger)

3.4 Algorithms

3.4.1 CNN

Convolutional Neural Networks, often abbreviated as CNNs, represent a pivotal breakthrough in the domain of deep learning and image analysis. Designed to mimic the human visual system, CNNs are a class of artificial neural networks tailored to excel in tasks like image classification, object detection, facial recognition, and more. This section elucidates the inner workings and significance of CNNs in the context of our project.

- **Convolution Operation:** The convolution operation is a mathematical operation applied to an input image (usually represented as a 2D or 3D array) and a filter (also a 2D or 3D array).

- **Feature Map:** The result of the convolution operation is a feature map, which highlights certain patterns or features in the input image. Mathematically, the feature map can be represented as a 2D array.
- **Max-Pooling Operation:** Max-pooling is a down-sampling operation commonly used in CNNs. It reduces the size of the feature map while retaining important information.
- **Fully Connected Layer:** Fully connected layers involve matrix multiplications and activation functions. Let's denote a fully connected layer as FC. The input to the FC layer is a vector of flattened feature map values, and the output is another vector of values.
- **Backpropagation and Training**: CNNs are trained using techniques like stochastic gradient descent (SGD). The loss function, often denoted as L, measures the difference between the predicted output and the actual target. The goal is to minimize this loss by adjusting the model's parameters (weights and biases) through gradient descent.

3.4.2 Face detection algorithms

This is the computer vision technique designed to automatically locate and identify human faces within images or video streams. These algorithms play a fundamental role in a wide range of applications, including facial recognition, emotion analysis, image retrieval, video surveillance, and more.

Features of Face Detection Algorithms:

- **Robustness:** These algorithms are designed to work in various lighting conditions, with different skin tones, facial expressions, and head orientations.
- **Real-time Processing:** Many modern face detection algorithms are optimized for real-time applications, such as video surveillance and camera-based systems.
- **Efficiency:** Efficiency is crucial, especially for embedded systems and mobile applications, so the algorithms are often designed to be computationally efficient.
- **Scale and Pose Invariance:** These algorithms can detect faces at different scales (e.g., close-up or distant) and with variations in pose (e.g., frontal, profile, or tilted faces).

Haar Cascades:

One of the earliest and most popular face detection methods is the Haar Cascade classifier. It uses a set of features called Haar-like features and a trained classifier to detect faces. Haar Cascade classifiers work by sliding a window over an image and applying a series of simple filters to check for the presence of facial features like eyes, nose, and mouth. While it's relatively fast and efficient, Haar Cascade-based methods may have limitations in detecting faces in challenging conditions.

3.4.3 Hand Tracking Algorithms

Hand-tracking algorithms are computer vision techniques designed to automatically locate and track a person's hand or hands in images or video streams. These algorithms play a crucial role in a variety of applications, including gesture recognition, sign language interpretation, augmented reality, virtual reality, and human-computer interaction.

cvzone: 'cvzone' is a Python library that provides pre-built functionalities for computer vision tasks, including hand tracking. It simplifies the process of integrating hand tracking into applications and projects. It's particularly useful for those who want a convenient and quick way to add hand tracking capabilities.

HandTrackingModule: This module, presumably part of 'cvzone' or a similar library, specifically focuses on hand tracking. It might include pre-trained models or functions for detecting and tracking hands in video frames, making it easier for developers to work with hand tracking in their projects.

Features of Hand Tracking Algorithms:

- **Real-time Processing:** Many hand tracking algorithms are optimized for real-time applications, as they are often used in interactive systems that require immediate responses.
- **Efficiency:** Efficiency is critical, particularly for applications on resource-constrained devices like mobile phones or augmented reality glasses.
- **Multi-Hand Tracking:** Some algorithms are capable of tracking multiple hands in the same frame, allowing for more complex interactions.

3.5 Coding

3.5.1 Distraction Detection

Model building: The constructed deep learning model follows a sequential architecture, serving as an image classifier for identifying distracted driving behaviors. It commences with data preprocessing, specifically rescaling the input images to normalize pixel values within the [0, 1] range. The subsequent layers encompass Conv2D for feature extraction, MaxPooling2D for spatial dimension reduction, and Dropout layers to curb overfitting. The Flatten layer reshapes the data for fully connected layers, which include units of 1024, 512, and 256, each employing ReLU activation. The conclusive layer, activated by softmax, delivers the final classification, with 10 neurons representing diverse distracted driving behaviors. This model efficiently extracts image features and accurately categorizes these behaviors, significantly contributing to the enhancement of road safety.

```
# creating our model
model = tf.keras.models.Sequential([
   Rescaling(scale = 1/255,input_shape=(100,100,3)),
   Conv2D(32,(3,3),activation='relu'),
   MaxPooling2D((2,2)),
   Dropout(0.1),
   Conv2D(64,(3,3),activation='relu'),
   MaxPooling2D((2,2)),
   Conv2D(32,(3,3),activation='relu'),
   MaxPooling2D((2,2)),
   Dropout(0.1),
   Flatten(),
   Dense(1024,activation='relu'),
   Dropout(0.1),
   Dense(512,activation='relu'),

   Dense(256,activation='relu'),
   Dropout(0.1),
   Dense(10,activation='softmax'),
])

# compiling our model
model.compile(optimizer = Adam(lr=0.01),loss = 'categorical_crossentropy',metrics=['acc'])
model.summary()
```

Training the model

```
history = model.fit(train_data,epochs=10,validation_data=val_data)
```

Plotting Training and validation loss

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = history.epoch
plt.figure(figsize=(10,10))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
```

Drowsiness Detection

Detecting drowsiness and driver distraction through facial landmarks. It uses OpenCV, dlib, and imutils libraries to process a video stream. The program identifies faces in the video, tracks facial landmarks and measures eye blink ratios. Based on the blink ratio, it categorizes the driver's state into three categories: awake, drowsy, or asleep. It then displays this status on the video feed and visually marks facial landmarks. The code effectively tracks and monitors the driver's condition, making it a valuable component in a driver monitoring system for improving road safety and preventing accidents.

```
#Initializing the face detector and landmark detector
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

#status marking for current state
sleep = 0
drowsy = 0
active = 0
status=""
color=(0,0,0)

def compute(ptA,ptB):
        dist = np.linalg.norm(ptA - ptB)
        return dist

def blinked(a,b,c,d,e,f):
        up = compute(b,d) + compute(c,e)
        down = compute(a,f)
        ratio = up/(2.0*down)

        #Checking if it is blinked
        if(ratio>0.25):
                return 2
        elif(ratio>0.21 and ratio<=0.25):
                return 1
        else:
                return 0
while True:
  _, frame = cap.read()
  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
  faces = detector(gray)
  #detected face in faces array
  for face in faces:
    x1 = face.left()
    y1 = face.top()
    x2 = face.right()
    y2 = face.bottom()
    face_frame = frame.copy()
    cv2.rectangle(face_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
    landmarks = predictor(gray, face)
    landmarks = face_utils.shape_to_np(landmarks)
```

```
        #The numbers are actually the landmarks which will show eye
        left_blink = blinked(landmarks[36],landmarks[37],
                landmarks[38], landmarks[41], landmarks[40], landmarks[39])
        right_blink = blinked(landmarks[42],landmarks[43],
                landmarks[44], landmarks[47], landmarks[46], landmarks[45])
        #Now judge what to do for the eye blinks
        if(left_blink==0 or right_blink==0):
                sleep+=1
                drowsy=0
                active=0
                if(sleep>6):
                        status="SLEEPING !!!"
                        color = (255,0,0)
        elif(left_blink==1 or right_blink==1):
                sleep=0
                active=0
                drowsy+=1
                if(drowsy>6):
                        status="Drowsy !"
                        color = (0,0,255)
        else:
                drowsy=0
                sleep=0
                active+=1
                if(active>6):
                        status="Active :)"
                        color = (0,255,0)
```

Hands Detection

Hands Detection module demonstrates a driver assistance system utilizing hand tracking and computer vision. It captures video from a camera feed and uses the "cvzone" library for hand tracking, counting the number of hands detected. Initially, if two hands are detected, it prompts the message "Engine is started, you can drive. Happy Journey." If no hands are present, it displays a message advising the driver to place their hands on the steering wheel. Based on the count of detected hands, it dynamically updates the message, indicating whether the driver can accelerate at the maximum speed, drive at a reduced speed of 25 km/h with one hand, or if the brakes are being applied with no hands detected. This system assists drivers in adhering to safe driving practices by providing real-time feedback through the camera feed.

```
prev_num_hands = -1
count = 0

while True:
   isTrue, frame = video.read()
   hands, img = detector.findHands(frame)

   num_hands = len(hands)
```

```
  if count == 0 and num_hands == 2:
    cv.putText(frame, "Engine is started you can drive. Happy Journey", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)
    count += 1
  if count == 0:
    cv.putText(frame, "Place your hands on the steering to start the engine", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2, cv.LINE_AA)

  if count > 0:
    # if num_hands != prev_num_hands:
    #     prev_num_hands = num_hands

      if num_hands == 2:
        cv.putText(frame, "You can drive at the max speed", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)
      elif num_hands == 1:
        cv.putText(frame, "You can drive at max speed of 25Km/h", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)
      elif num_hands == 0:
        cv.putText(frame, "Brakes are being applied", (50, 50), cv.FONT_HERSHEY_SIMPLEX,
1, (0, 0, 255), 2, cv.LINE_AA)
```

3.6 Result

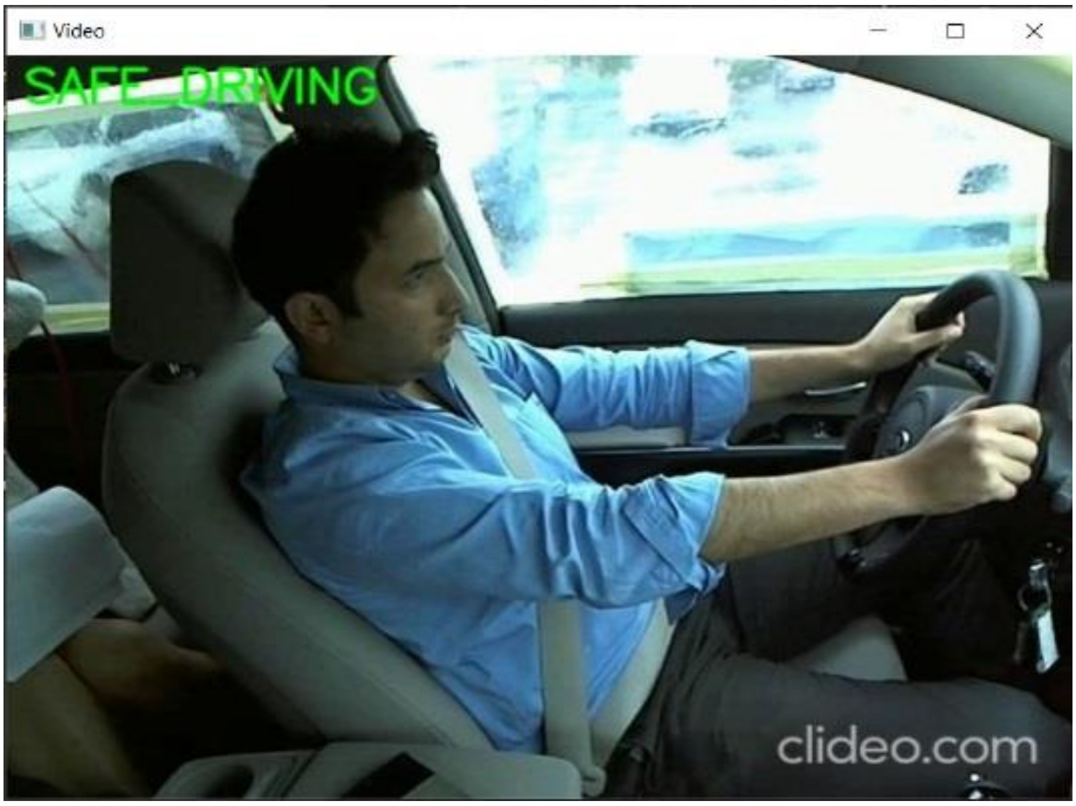


Fig 3.6.1 Driver driving the car the predicated output is safe driving

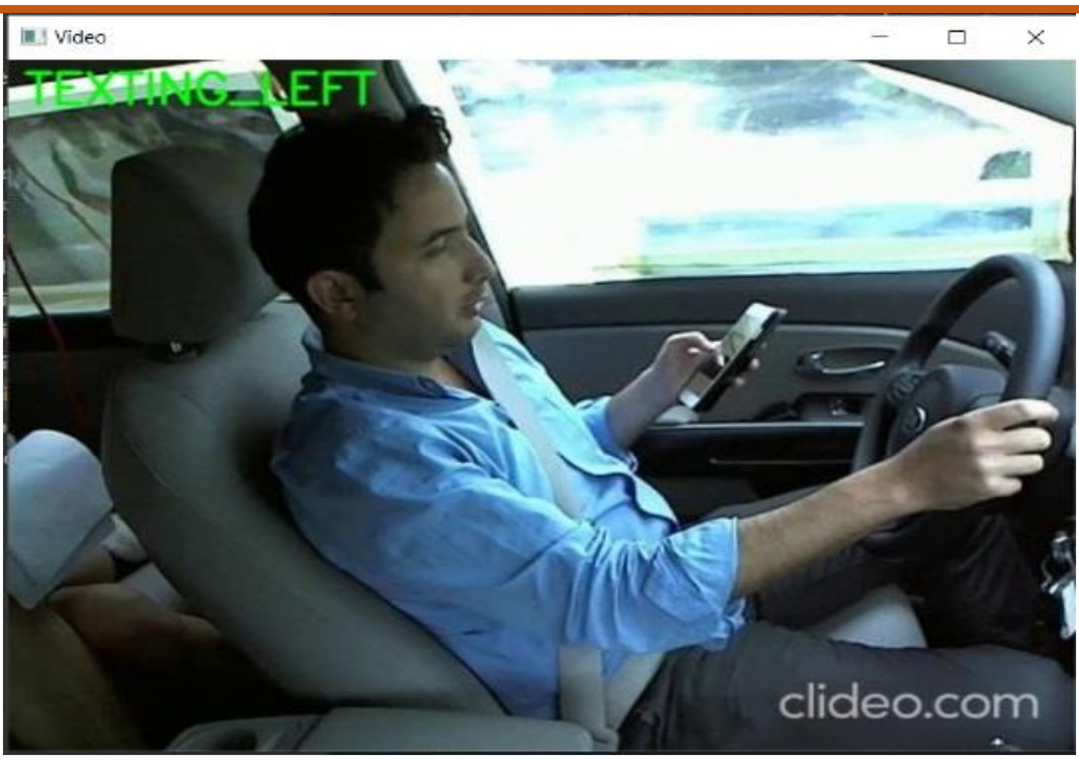


Fig 3.6.2 Driver texting and driving, the predicted output is Texting left

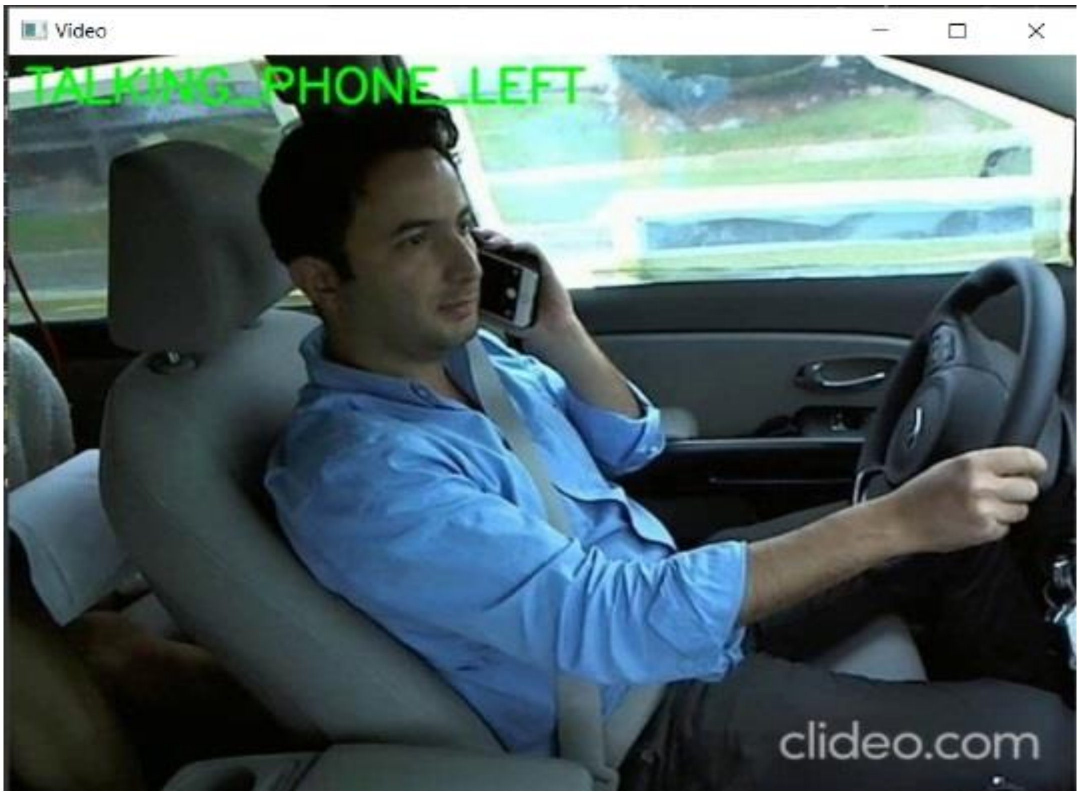


Fig 3.6.3 Driver talking in the phone and driving, the Predicated output is Talking phone left

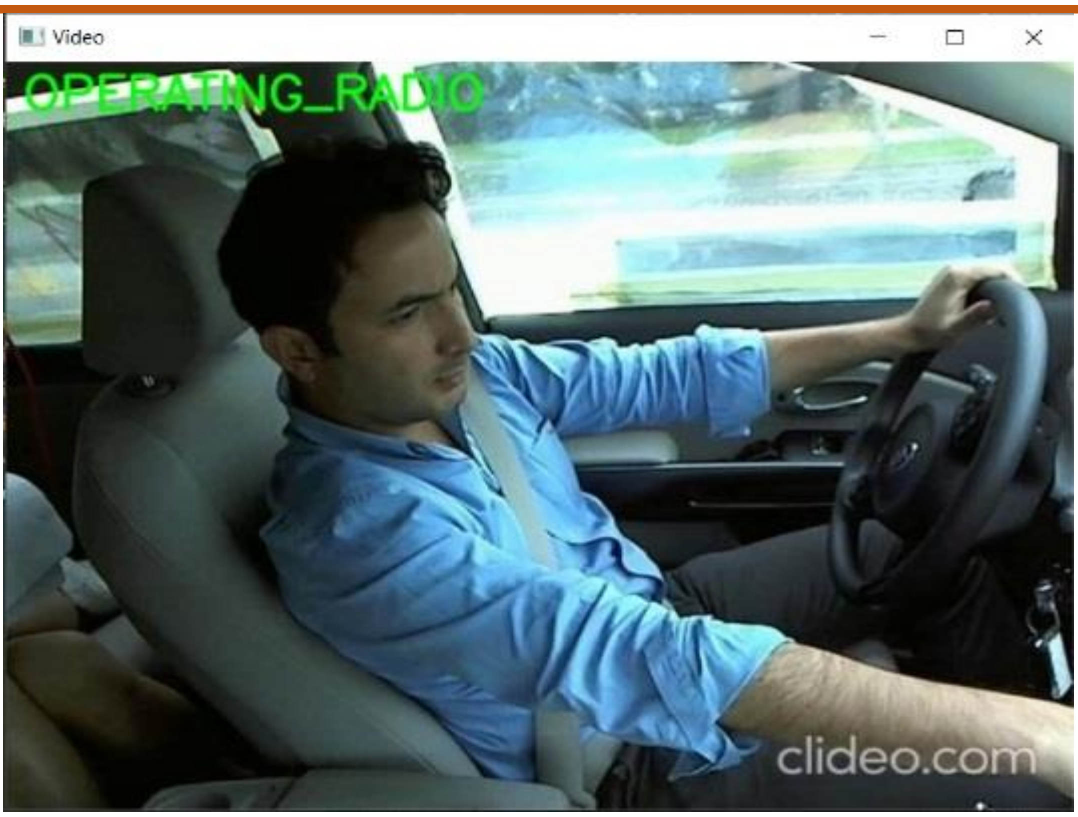


Fig 3.6.4 Driver operating the radio and driving car Predicated output Operating Radio

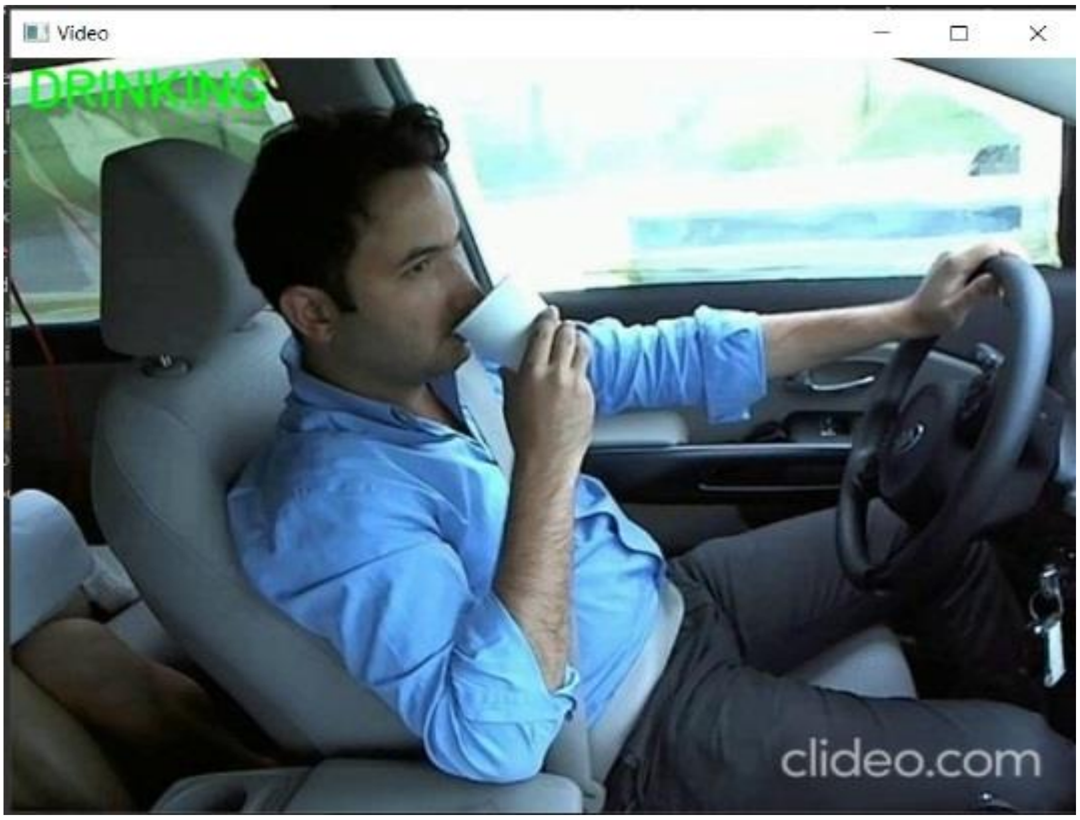


Fig 3.6.5 Driver drinking and driving car Predicated output Drinking

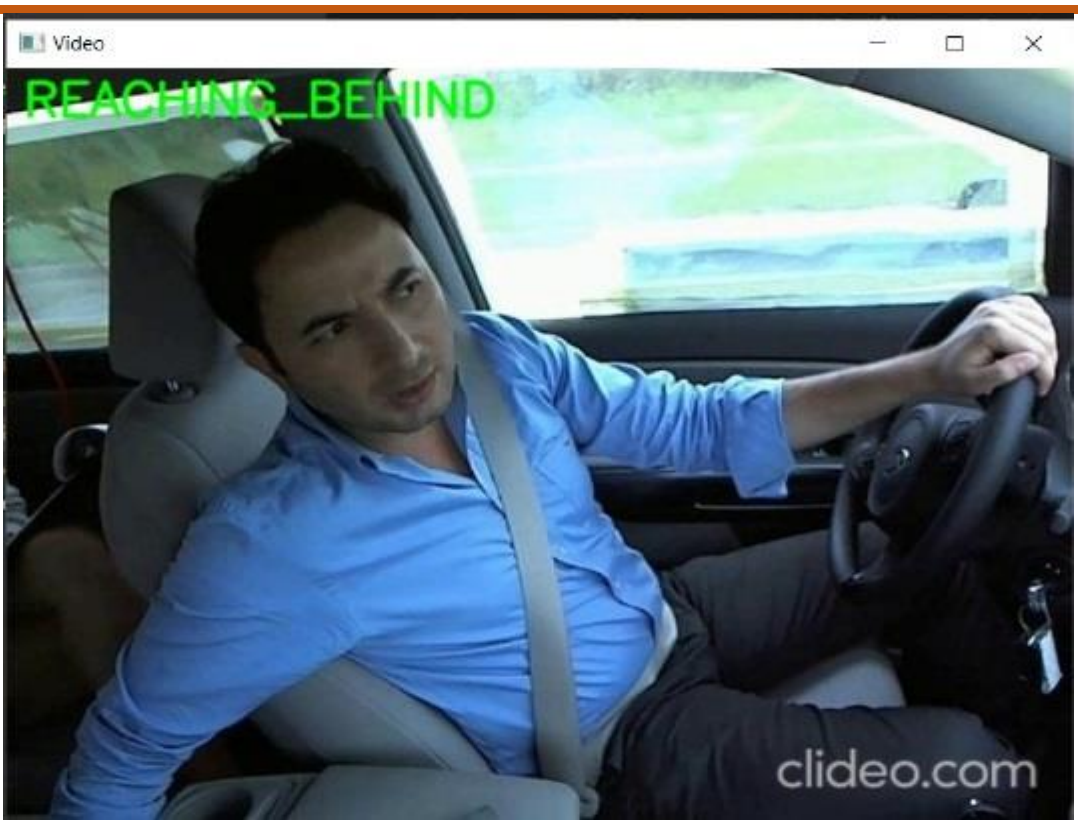


Fig 3.6.6 Driver reaching behind and driving car Predicated output Reaching Behind

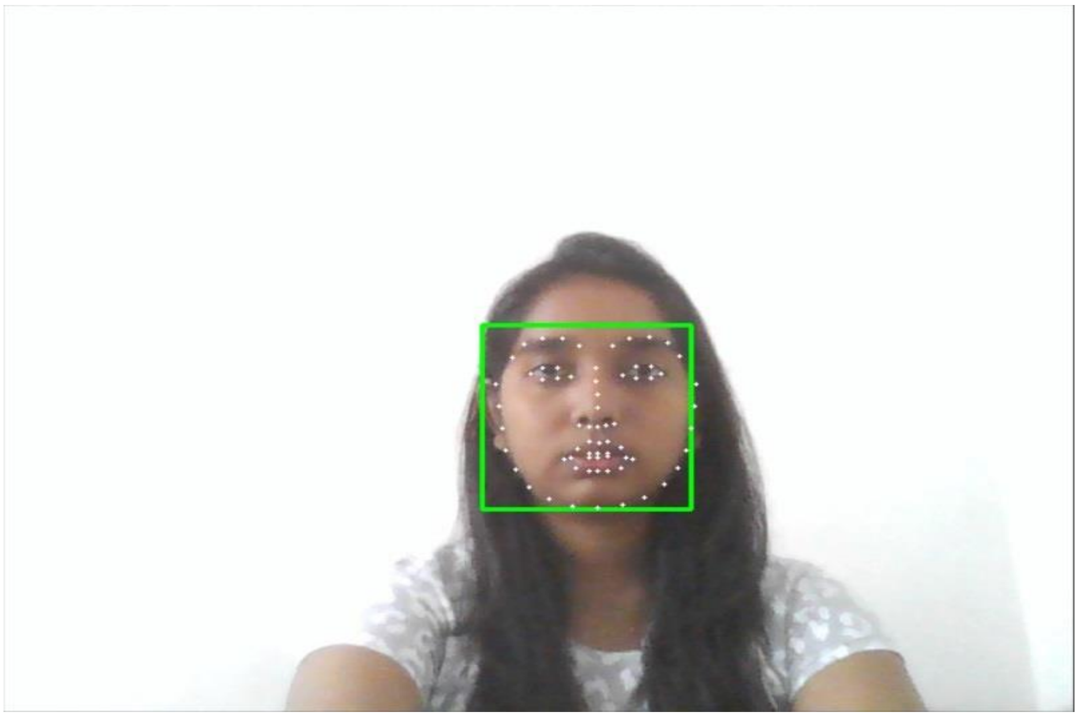


Fig 3.6.7 Face landmark recognition

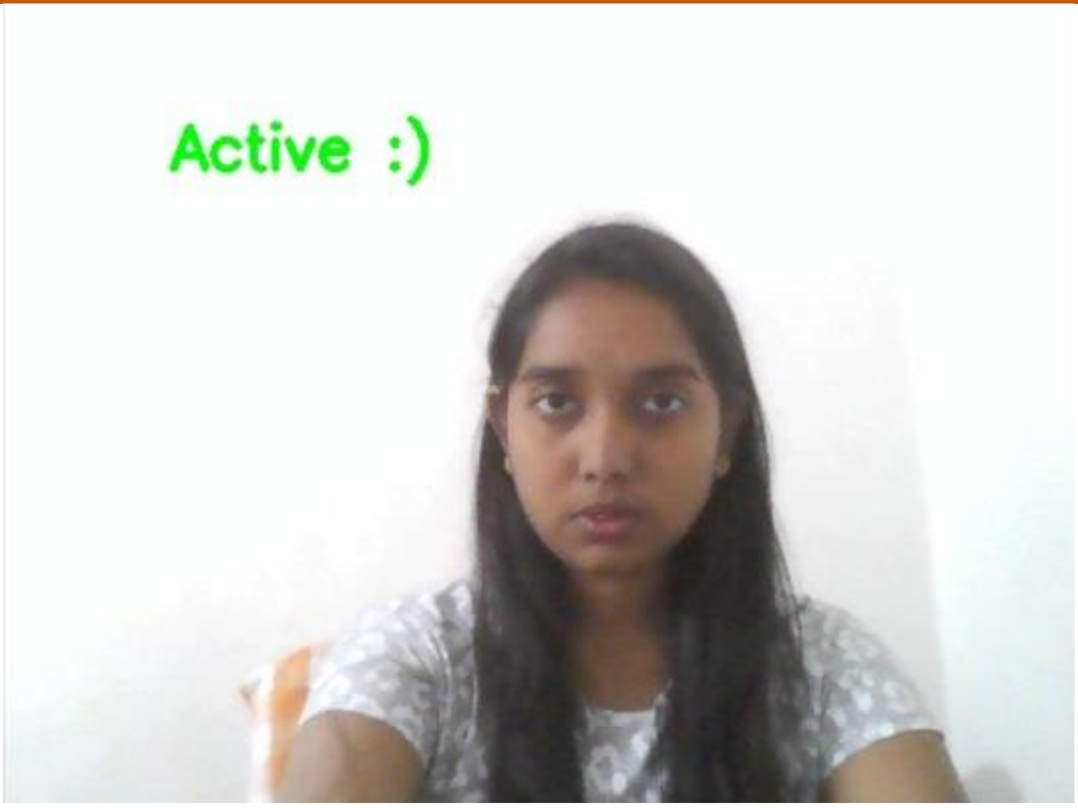


Fig 3.6.8 Active Driver face result

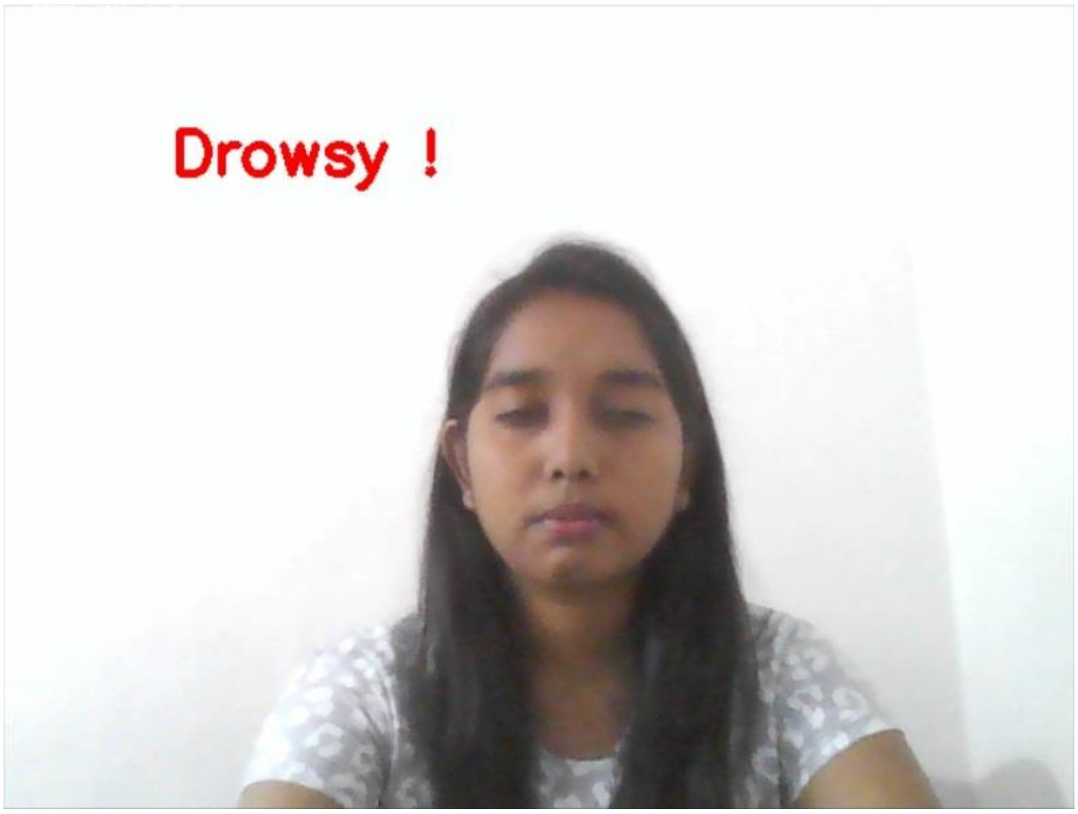


Fig 3.6.9 Drowsy Driver face result

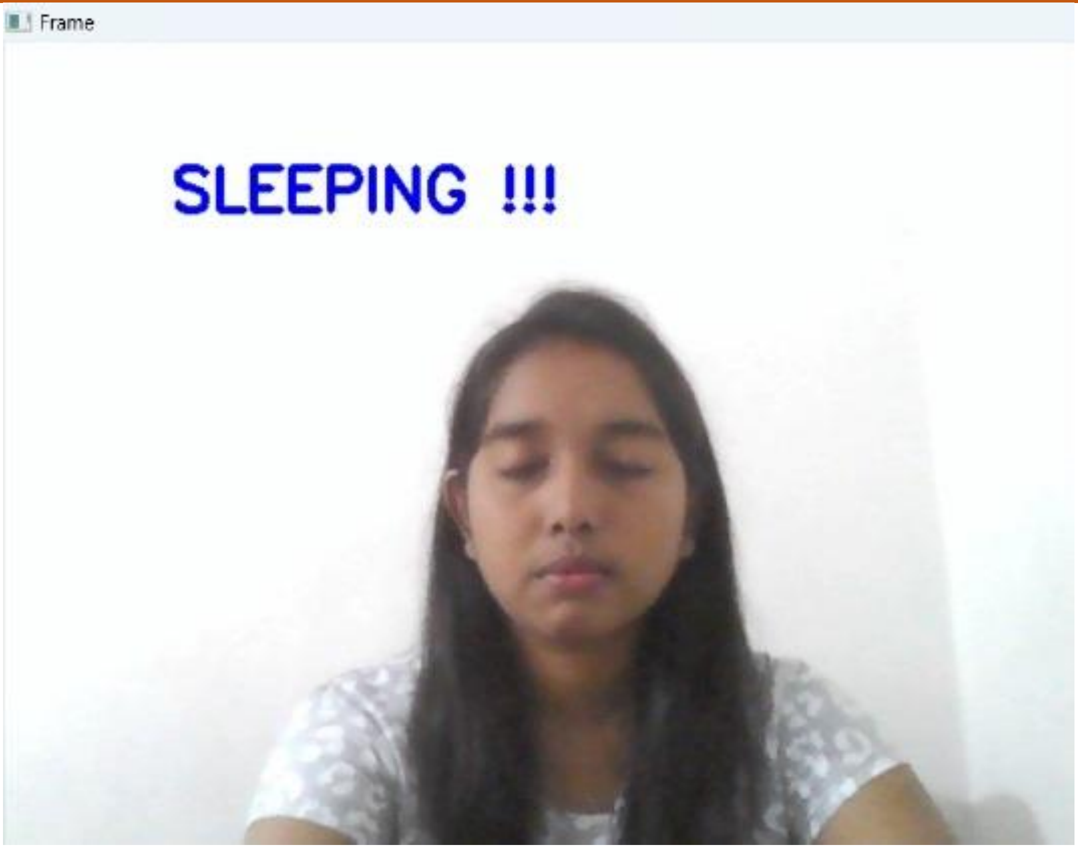


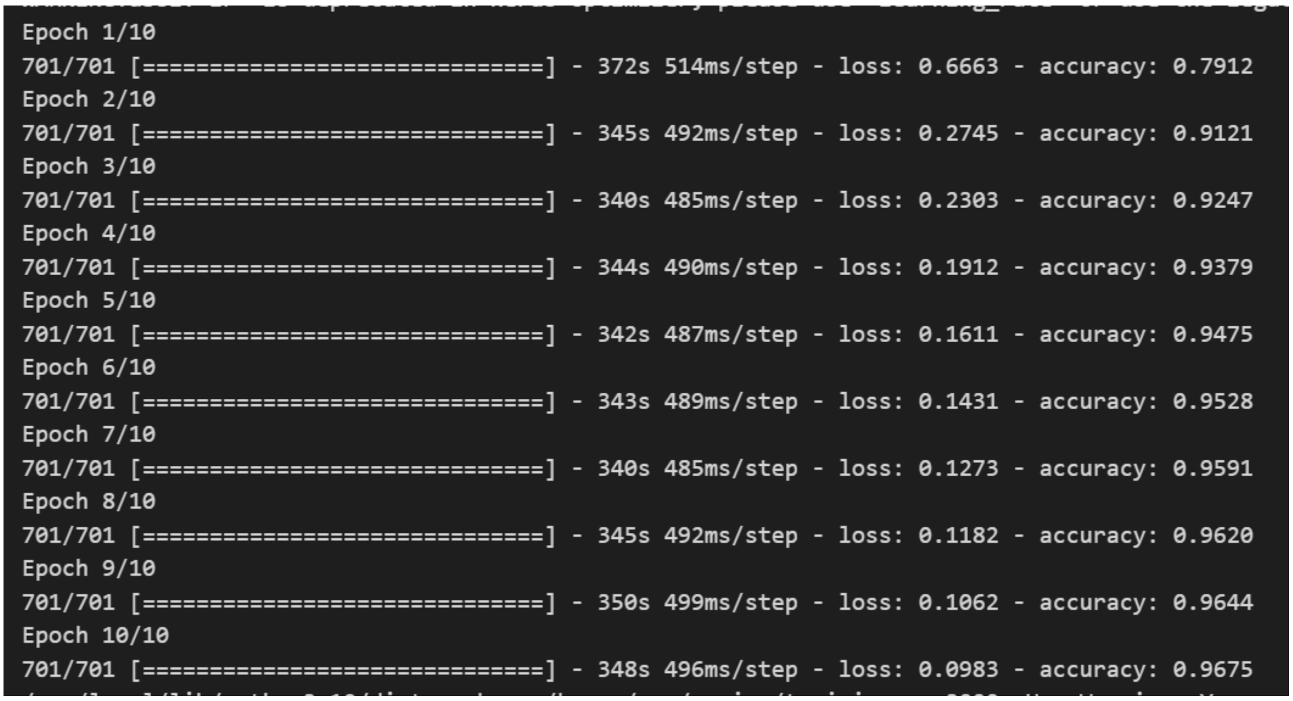
Fig 3.6.10 Sleepy Driver face result

Chapter 4

RESULT ANALYSIS

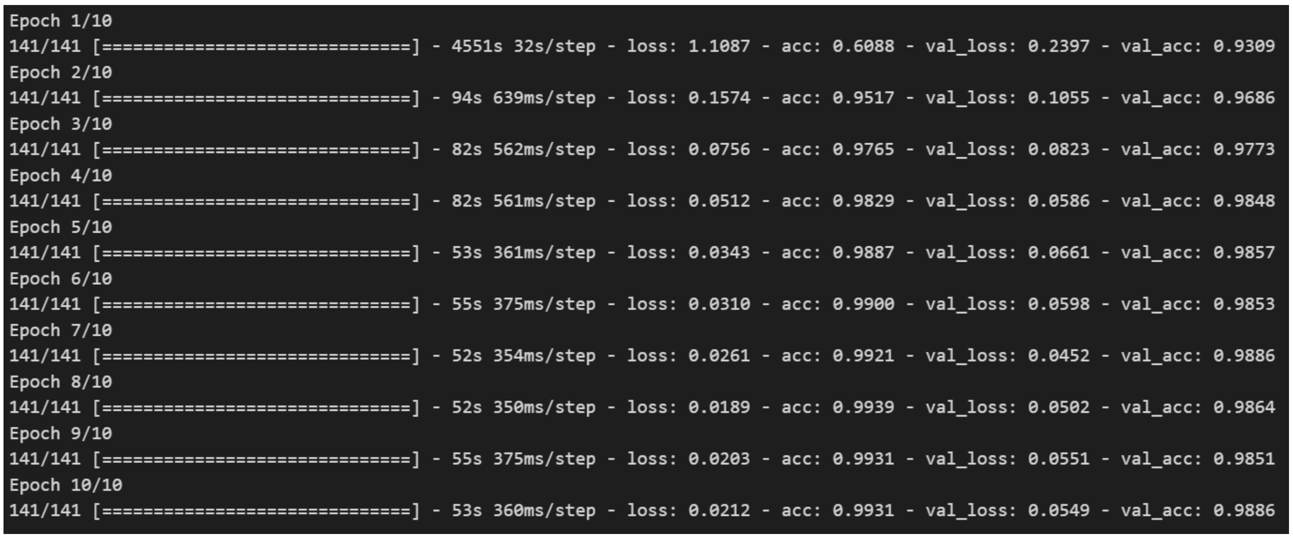
4.1 Comparison

MobileNet Vs Sequential CNN model



```
Epoch 1/10
701/701 [==============================] - 372s 514ms/step - loss: 0.6663 - accuracy: 0.7912
Epoch 2/10
701/701 [==============================] - 345s 492ms/step - loss: 0.2745 - accuracy: 0.9121
Epoch 3/10
701/701 [==============================] - 340s 485ms/step - loss: 0.2303 - accuracy: 0.9247
Epoch 4/10
701/701 [==============================] - 344s 490ms/step - loss: 0.1912 - accuracy: 0.9379
Epoch 5/10
701/701 [==============================] - 342s 487ms/step - loss: 0.1611 - accuracy: 0.9475
Epoch 6/10
701/701 [==============================] - 343s 489ms/step - loss: 0.1431 - accuracy: 0.9528
Epoch 7/10
701/701 [==============================] - 340s 485ms/step - loss: 0.1273 - accuracy: 0.9591
Epoch 8/10
701/701 [==============================] - 345s 492ms/step - loss: 0.1182 - accuracy: 0.9620
Epoch 9/10
701/701 [==============================] - 350s 499ms/step - loss: 0.1062 - accuracy: 0.9644
Epoch 10/10
701/701 [==============================] - 348s 496ms/step - loss: 0.0983 - accuracy: 0.9675
```

Fig 4.1 Accuracy of ModelNet Model



```
Epoch 1/10
141/141 [==============================] - 4551s 32s/step - loss: 1.1087 - acc: 0.6088 - val_loss: 0.2397 - val_acc: 0.9309
Epoch 2/10
141/141 [==============================] - 94s 639ms/step - loss: 0.1574 - acc: 0.9517 - val_loss: 0.1055 - val_acc: 0.9686
Epoch 3/10
141/141 [==============================] - 82s 562ms/step - loss: 0.0756 - acc: 0.9765 - val_loss: 0.0823 - val_acc: 0.9773
Epoch 4/10
141/141 [==============================] - 82s 561ms/step - loss: 0.0512 - acc: 0.9829 - val_loss: 0.0586 - val_acc: 0.9848
Epoch 5/10
141/141 [==============================] - 53s 361ms/step - loss: 0.0343 - acc: 0.9887 - val_loss: 0.0661 - val_acc: 0.9857
Epoch 6/10
141/141 [==============================] - 55s 375ms/step - loss: 0.0310 - acc: 0.9900 - val_loss: 0.0598 - val_acc: 0.9853
Epoch 7/10
141/141 [==============================] - 52s 354ms/step - loss: 0.0261 - acc: 0.9921 - val_loss: 0.0452 - val_acc: 0.9886
Epoch 8/10
141/141 [==============================] - 52s 350ms/step - loss: 0.0189 - acc: 0.9939 - val_loss: 0.0502 - val_acc: 0.9864
Epoch 9/10
141/141 [==============================] - 55s 375ms/step - loss: 0.0203 - acc: 0.9931 - val_loss: 0.0551 - val_acc: 0.9851
Epoch 10/10
141/141 [==============================] - 53s 360ms/step - loss: 0.0212 - acc: 0.9931 - val_loss: 0.0549 - val_acc: 0.9886
```

Fig 4.1 Accuracy of Sequential CNN Model

4.2 Graph

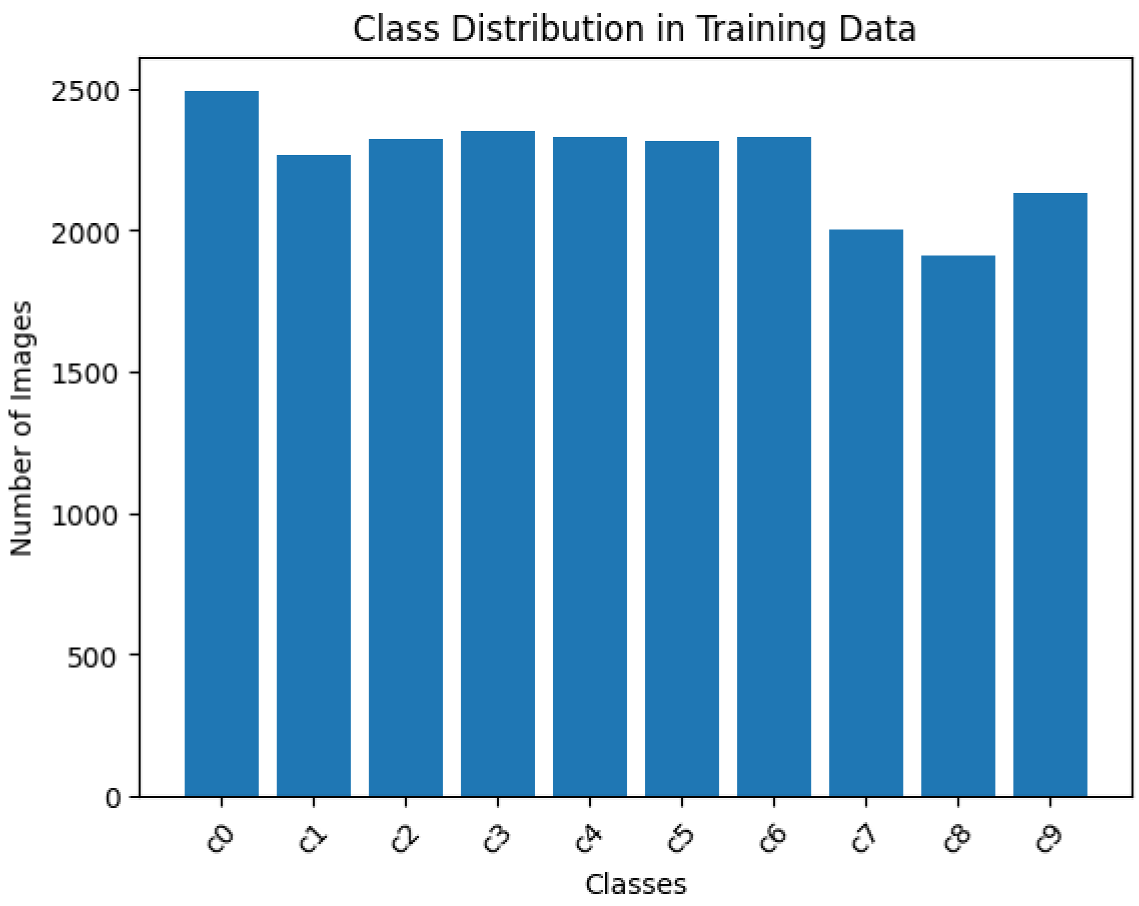


Fig 4.2.1 Bar Graph of Dataset

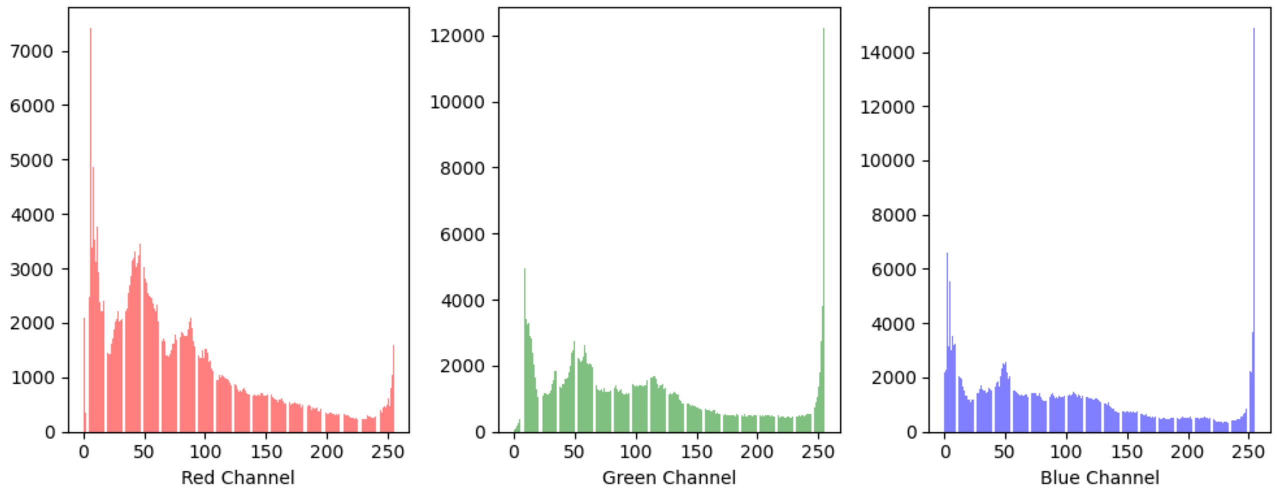


Fig 4.2.2 RGB Distribution of images in Dataset Images

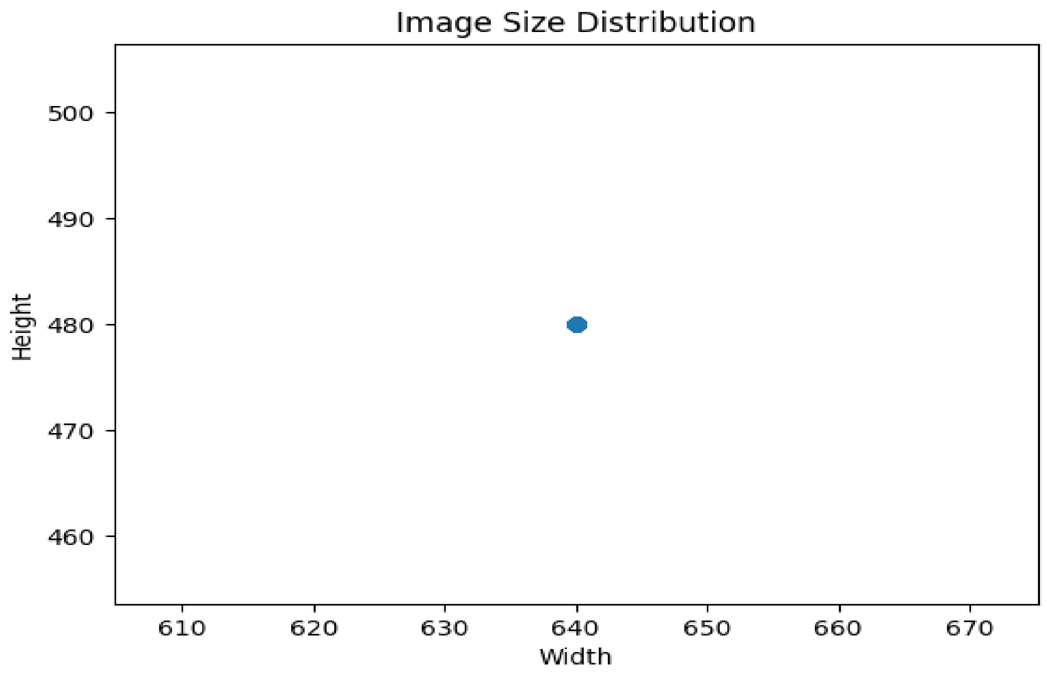


Fig 4.2.3 Scatter plot for image dimension in dataset

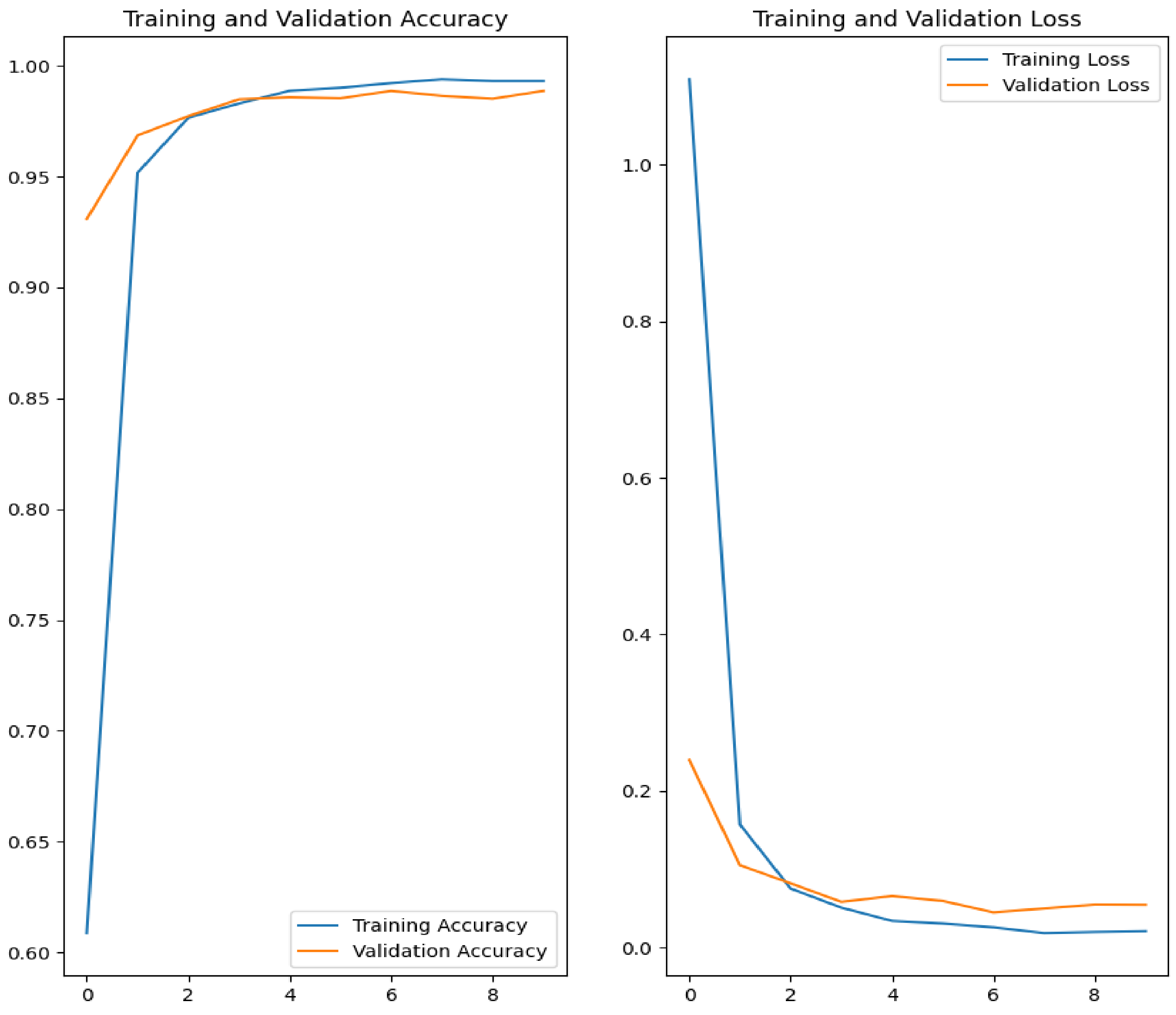


Fig 4.2.4 Training and Validation Accuracy and loss

Chapter 5

CONCLUSION

Transforming Road Safety with Advanced Driver Monitoring

In the pursuit of enhancing road safety, our project has ventured into the realm of advanced driver monitoring and distraction detection. The journey embarked upon was guided by a vision to reduce accidents, prevent injuries, and save lives. With a multi-faceted approach, our project addressed the critical issues of distracted driving, drowsiness detection, and hand positioning on the steering wheel. The collective efforts yielded a robust system that not only detected anomalies but also provided real-time feedback and insights to empower safer driving practices.

Key Features:

- Distracted Driver Detection: Leveraging the power of Convolutional Neural Networks (CNNs), our system proficiently analyzed dashboard camera images to classify driver behavior. It accurately discerned actions like texting, phone usage, and even radio operations. By doing so, it contributed to raising awareness and taking preventive actions against one of the leading causes of road accidents.

- Drowsiness Detection: Recurrent Neural Networks (RNNs) played a pivotal role in our endeavor to identify the subtle signs of driver drowsiness. By scrutinizing the sequential data of eye blinks, the system could differentiate between an alert and a drowsy driver, intervening when necessary to avert potential disasters.

- Hands Positioning on the Steering Wheel: The utilization of hand tracking algorithms, such as 'cvzone' and 'HandTrackingModule,' ensured that hands remained appropriately placed on the steering wheel. This proactive approach aimed to minimize the chances of accidents caused by improper hand positioning.

Future Enhancements

The successful implementation of driver monitoring and distraction detection in our project doesn't mark the peak but rather serves as a foundation for a future where road safety becomes a lifestyle rather than just a concept. The significance of this work extends beyond the domains of technology and data analysis; it encompasses the well-being of individuals, families, and entire communities. With this broader vision in mind, we look forward to the future possibilities of this initiative.

Integration into Hardware (e.g., Raspberry Pi):

A promising direction for the future involves incorporating our driver monitoring system into vehicle hardware. By utilizing platforms like the Raspberry Pi or similar embedded systems, an onboard assistant can be created, actively encouraging responsible driving.
This system has the potential to offer real-time feedback to the driver, gently reminding them to maintain focus, keep their hands on the wheel, and avoid distractions.
Advanced Behavioral Analysis:

Expanding the capabilities of the system, we can explore more sophisticated behavioral analysis. Beyond recognizing common distractions, the system can be trained to detect signs of aggressive driving, driver fatigue, and even health-related concerns, providing early alerts and assistance.

REFERENCES

5.1 Papers

- An Efficient Deep Learning Framework for Distracted Driver Detection.
- Driver Distraction Detection Using Advanced Deep Learning Technologies Based on Images.
- Driver Distraction Detection Methods: A Literature Review and Framework.
- Multimodal driver distraction detection using a dual-channel network of CNN and Transformer.
- Automatic driver distraction detection using deep convolutional neural networks.
- Driver distraction detection via multi-scale domain adaptation network.

5.2 Web Links

- Facial Landmark
- CNN Documentation
- MobileNet Documentation
- VGG16 Documentation
- RESNET Documentation
- OpenCV Documentation
- cvzone Documentation
- Dataset
- TensorFlow Documentation
- Keras Documentation