Dissertation on

"Driver Safety System using Internet Of Things "

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE19CS390B – Capstone Project Phase - 2

Submitted by:

SAHIL BAVARIYA	**PES1UG19CS414**
JOSHUA D'SOUZA	**PES1UG20CS811**
SHRADHYA RAKSHIT	**PES1UG20CS804**
P VIKRANTH REDDY	**PES1UG19CS319**

Under the guidance of

Prof. Priya Badarinath
Assistant Professor

January - May 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

"Driver Safety System using Internet of Things"

is a bonafide work carried out by

SAHIL BAVARIYA	**PES1UG19CS414**
JOSHUA D'SOUZA	**PES1UG20CS811**
SHRADHYA RAKSHIT	**PES1UG20CS804**
P VIKRANTH REDDY	**PES1UG19CS319**

in partial fulfillment for the completion of eighth semester Capstone Project Phase - 2 (UE17CS490B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Aug 2022 – Dec. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature **PRIYA BADARINATH** Asst Professor,	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **“Driver Safety System using Internet Of Things”** has been carried out by us under the guidance of

Prof. Priya Badarinath,is submitted in partial fulfillment of the course requirements for the award of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester August – December 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1UG19CS414 SAHIL BAVARIYA

PES1UG20CS811 JOSHUA D’SOUZA

PES1UG20CS804 SHRADHYA RAKSHIT

PES1UG19CS319 P VIKRANTH REDDY

ACKNOWLEDGEMENT

We would like to express my gratitude to Prof. Priya Badarinath, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE19CS390B - Capstone Project Phase – 2.

We are grateful to the project coordinators, Prof. Mahesh Basavanna, for organizing, managing, and helping with the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way.

Finally, this project could not have been completed without the continual support and encouragement we have received from my family and friends.

ABSTRACT

Road accidents are prevalent all around the world and a prime reason for that is driver drowsiness and/or driving under the influence. Our aim with this project is to curb such mishaps and to save more lives from being lost. We implement a real-time detection system which will track the movement of the eye and determine whether the person is in a drowsy state or not. Our system also takes into account the yawn frequency, if the driver in question is yawning. An added functionality of our system is to detect whether the driver is distracted while driving, upon which he/she will be alerted accordingly. We are utilizing cameras for taking in a live video input and a sensor for testing for the alcohol concentration in the driver's breath and in the air around him. In all cases, the driver will be alerted so as to prevent any possible mishaps.. Using various image processing techniques and Machine Learning algorithms, our inputs will be processed and analyzed. Combining these three parts we aim to achieve a greater accuracy in our system.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Every year approximately 1.3 million people lose their lives due to road accidents. An estimated 20 to 50 million people suffer from non-fatal injuries due to such mishaps. Statistics show that the primary reasons for the occurrence of such accidents are Driving Under Influence, Speeding, Fatigue and Distracted Driving. While the United Nations General Assembly hopes to cut down the global road accident rate by half by the year 2030, which is rather ambitious than realistic.

With such massive date rates all around the globe, tackling this issue is highly critical. Our proposed system aims at handling these problems and thereby helping in cutting down the accidents and thereby ensuring a safe driving experience for both the driver and the pedestrians.

Our system focuses on three major aspects: namely, Drowsiness Detection or determining whether the driver is in a drowsy state or not, Drunk Detection or ensuring that the driver is not driving under influence by testing for the alcohol concentration and Distraction Detection or an analysis of the emotional state of the driver i.e. if a driver is stressed or is being easily distracted by an object or his own intrusive thoughts, it can lead to him not keeping an eye out on the road and could inevitably lead to an accident.

The main input for our detection system would be a live video feed using a camera (Raspberry Pi Camera Module V2) which will subsequently be divided into frames and each individual frame will be analyzed. Using image processing techniques, facial recognition will be implemented on these frames wherein 68 facial landmarks will be identified on the person's face and will further be used in eye and mouth tracking. Tracking the state of the eye and the mouth, we can determine whether the person is drowsy or not. As far as the distraction detection is concerned, we shall have certain thresholds which will be compared with the input to see if the driver is focused on driving i.e. a time constraint for how long the eyes are focused

on the road and if the driver's eyes are averted for longer than the threshold then he/she shall be alerted to take a break.

Testing for the alcohol concentration in the air is also another aspect of our project wherein we will determine whether the driver is driving under the influence. We are utilizing MQ3 sensors for checking the concentration of alcohol in the driver's breath and/or in the air around him. If alcohol is detected then the driver would be alerted using an alarm.

The combined implementation of these three aspects of our project will help us bring forth a system that will ensure a better and safer driving experience for both the driver and the pedestrians involved.

CHAPTER 2

PROBLEM STATEMENT

Our project focuses on implementing a driver safety system wherein the three major reasons for road accidents are tackled. The three main aspects being Drowsiness Detection, Distraction Detection and Alcohol Detection. Facial recognition combined with image processing techniques will be applied to both detect whether the driver is drowsy and/or if he/she is distracted. In both situations the driver will be alerted. As far as alcohol detection is concerned, sensors will be used to check for the ethanol concentration in the air around him and determine whether the driver is drunk or not.

Our proposed system aims at combining these three modules to ensure a safe driving experience and will also help at reducing the rate of road accidents to a great extent.

CHAPTER 3

LITERATURE SURVEY

In this chapter, we present the current knowledge of the area and review substantial findings that help shape, inform and reform our study.

3.1 Detection of Driver Drowsiness by Calculating the Speed of Eye Blinking-2021

Muhammad Fawwaz Yusri, Patrick Mangat, and Oliver Wasenmiller

Abstract:

This system considers a simple real-time detection system for drowsiness merely based on the eye blinking rate derived from the eye aspect ratio. For the eye detection we use HOG and a linear SVM. If the speed of the eye blinking drops below some empirically determined threshold, the system triggers an alarm, hence preventing the driver from falling into microsleep. In this paper, we extensively evaluate the minimal requirements for the proposed system. We find that this system works well if the face is directed to the camera,but it becomes less reliable once the head is tilted significantly. The results of our evaluations provide the foundation for further developments of our drowsiness detection system.

Methodology:

-> Drowsiness is detected by analyzing the movement of the eyelids ie using different points as landmarks on the eyes to monitor movement

-> Landmarks are tracked using dlib library by using an ensemble of regression trees

-> Using EAR, when eyes are open it is 0.35 and that value rapidly drops to below 0.15 as the eyes start to close.

-> Max and MIn thresholds are calculated

->Normal Blinking Rates and Sleepy Blinking Rates are compared(between 3 participants) to determine the definitive thresholds.

Advantages:

Despite the changes in the distance between the face and the camera, the EAR(Eye Aspect Ratio) remains relatively the same. To attest to this, two still images with varying eye sizes are used instead of a live video capture. Image with a larger eye indicates a greater distance between the face and the camera as opposed to the image with a smaller eye. The deviation ITO EAR between both images is negligible.

Disadvantages:

-> Sometimes the eye blinking speed can drop below the threshold

-> If the head moves from right to left, it is rather difficult to keep track of the EAR and blinking speed since the movement is too rapid.

-> If the eyes are in sync with the head movement, then if the head faces upwards and downwards then detecting eyes is possible only to a certain degree since the eyes tend to look smaller.The speed is also inaccurate. When the head is moving from left to right, as long as the eyes are facing the camera, only for that duration the eyes can be detected but the speed is inaccurate.

-> Facial expressions such as smiling or frowning can greatly affect the detection of eyes and EAR accuracy.

3.2 Development of an intelligent drowsiness detection system for drivers using image processing technique

Suhaiman, A.A., May, Z., Rahman, N.A.A.

Abstract:

This intelligent system is developed to detect driver drowsiness and to trigger an alarm to alert said drivers. Due to the availability of several surrounding parameters current techniques have several limitations. Poor lighting affects the camera's ability to properly detect the face and the eye. This greatly affects the image processing technique due to late detection or no detection at all thereby decreasing the accuracy by a huge margin. This system proposes a real-time detection system which utilizes a camera and image processing libraries which captures and studies the eye and the entire face to determine the state of the person driving.

Methodology:

->Feature extraction from facial landmarks is the primary approach in detection of any image. 68 individual coordinates are identified on a face and are tracked.

->The eye region specifically is tracked and monitored and is measured with respect to a relation called Eye Aspect Ratio (EAR).

$$EAR = \frac{||p2-p6|| + ||p3-p5||}{2||p1-p4||}$$

wherein, the parameters are the facial landmarks detected earlier.

-> A total of 20 trials are conducted to verify and settle on the final EAR value

-> Using the accuracy formula, they have calculated the accuracy of the whole project which came down to about 80%.

3.3 Driver Drowsiness Detection using Eye Closeness Detection-2016

Aryan Khan Pisuth, Taweechai Chotchinasri,Varakorn Koschakosai

Abstract:

The purpose of this paper was to devise a way to alert drowsy drivers in the act of driving. Therefore, this study attempted to address the issue by creating an experiment in order to calculate the level of drowsiness. A requirement for this paper was the utilization of a Raspberry Pi Camera and Raspberry Pi 3 module, which were able to calculate the level of drowsiness in drivers. The frequency of head tilting and blinking of the eyes was used to determine whether or not a driver felt drowsy.

Methodology:

->A real time detection of the face using Raspberry Pi as the hardware component and image processing libraries to accurately detect the face and its features.

->The major parameters are the position of the head and eye blinking rate. The ROI(Region of Interest) is continuously tracked and monitored for any kind of movements.

-> A Haar Cascade Classifier has been used for face detection in the ROI.

3.4 Driver drowsiness detection with ANN image processing

T. Vesselenyi et al 2017 IOP Conf. Ser.: Mater. Sci. Eng. 252 01209

Abstract:

The paper introduces research on the potential for improving sleep motor vehicle acquisition system based on three types of modes: EEG and EOG signal processor image processing and analysis. In previous works the authors explained this research using the first two methods. In this paper the authors have read that it is possible to find the driver's sleepy or alert state based on photos taken while driving again to analyze the condition of the driver's eyes: open, open and closed. For these two purposes types of artificial neural networks used: 1 hidden network network and autoencoder network.

Concept for a drowsiness detection system:

Three of these methods are based on EEG (Electroencephalography) and EOG (Electrooculography) which shows the proportions of the optical (closed or open) image separation. The EEG system monitors brain activity through a sensor located in a specific area of the skin, the EOG system tracks eye movements by measuring signals from the muscles where eye function and visual acuity can monitor the open or closed eye.

Each method used in the system has its advantages and disadvantages. EEG and EOG sensors, electrodes will be adjusted to the gel running and most devices must transmit signals by telephone, causing severe discomfort. Important research in the field of advanced materials and MEMS technology can solve these problems. Advances in this field are supported by brain efforts - electronic communication for a variety of applications, including equipment for people with disabilities.

The use of EEG signal acquisition and processing can make the difference between awareness and drowsiness. The authors analyzed the feasibility of using EOG signals found in 3 sensors. The combination of these four types of signals provides information on distinguishing between left, right, up and down eye movements.

The possibilities for using EEG and EOG signals are evaluated to determine the driver. drowsiness.

Finding drowsy is done using image processing for the driver's eyes:

To differentiate the driver from drowsiness or alertness, artificial neural networks are used. Synthetic networks have been used extensively in image classification for decades. The paradigm called Deep Learning , Deep Faith Networks, Boltzmann Limited Borders and Deep Autoencoders are all modes which are part of the Deep Learning paradigm.

These methods are used in a variety of applications, image classification is one of the fields where these are used effectively. Lessons presented on this page, Matlab Neural Network Toolbox with 1 layer ANN layer and default code scanner Advanced Toolbox module used to learn if these methods can be used for separation of the driver's sleep image. As a basis, sleep was thought to be a thing connected to photos when the driver closed his eyes and alert mode connected to photos opened the driver's eyes. This module has been used for analysis .

To analyze driver sleep mode 200 photos of drivers at normal time the driving process was detected. Hundreds of these images contain eyes open or partially open eye images and a hundred other images contain closed eye images. To avoid overload of memory and large processing times, images have been resized as a low sample for a small amount of input data for neural networks

3.5 Distracted Driver Detection Published in 2020

Deep Ruparel,Abhay Rajde,Sahil Shah,Prof..Manya Gidwani

Abstract:

Accidents have been occurring at an alarming rate all around the world in recent years. When investigating the source of this rising rate, it was discovered that the majority of drivers were distracted while driving, resulting in accidents. According to the National Highway Traffic Safety Administration, distracted drivers cause approximately one in every five car accidents. They created a model that can accurately determine if a driver is distracted while driving or is driving safely. In this paper, they listed the numerous deep learning algorithms that they used for the detection like vgg16 model, mobilenet, sequential model, pretrained weights of the vgg16

model, and many others. which assisted them in developing a model capable of providing them with accurate and clear outcomes. Experiments reveal that their approach has a 93 percent accuracy rate.

3.6 Driver Drowsiness Detection Published in 2020

V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N, Dr.Nagamani N P

Abstract:

Machine learning methods are used to predict the state and emotions of drivers. This is an artificial intelligence application**.** Artificial intelligence is a way for systems to automatically learn better without explicit planning and improve without programming. Biometric indicators, driving behavior, and the driver's facial expressions can tell you about the driver's condition**.** This article provides a full overview of recent work related to systems that detect and warn of drowsy drivers. MATLAB is used for image processing, and various machine learning techniques such as the PERCLOS algorithm, HAAR-based cascade classifier, and OpenCV are presented to determine the driver's condition. Finally, they identified the challenges facing the current system and presented relevant research opportunities

3.7 Driver Drowsiness Detection System

Belal Alshaqaqi; Abdullah Salem Baquhaizel; Mohamed El Amine OUIS; Meriem Boumehed; Abdelaziz Ouamri; Mokhtar Keche

Abstract:

Drowsiness and Fatigue of drivers are the main causes of road accidents globally. Every year, Drowsiness and Fatigue increase the amounts of deaths and fatalities globally. In this paper, they showed a module called Advanced Driver Assistance System (ADAS) is presented to reduce the number of accidents due to drivers fatigue and also increase transportation safety. And this ADAS system deals with automatic driver drowsiness detection based on visual information through a cam and Artificial Intelligence. They proposed an algorithm to locate, track, and analyze both the drivers

face and eyes to measure PERCLOS, a scientifically supported measure of drowsiness associated with slow eye closure.

3.8 Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images

Elena Magán, M. Paz Sesmero , Juan Manuel Alonso-Weber and Araceli Sanchis

Abstract:

In this paper they present the development of an ADAS (advanced driving assistance system) focused on driver drowsiness detection, whose objective is to alert drivers of their drowsy state to avoid road traffic accidents. In driving, it is important that fatigue detection is performed in a nonintrusive way, and that the driver is not worried about the alarms when he is not drowsy. Their approach to open problems uses sequences of images upto 60s long and are recorded in such a way that the driver face is visible. To detect whether the driver shows symptoms of drowsiness or not, two alternative solutions have been developed, focusing mainly on minimization of false positives. The first alternative is deep learning techniques to extract numeric features from images, while the second one uses recurrent and convolutional neural networks, which they introduced into a system called fuzzy logic based system afterwards. Accuracy of up to 65% was achieved on both systems. Also, the accuracy of the training data is 65% and the accuracy of the test data is 60%. And the fuzzy logic-based system stood out in that it achieved a specificity of 93% without generating false alarms. Although the results obtained do not give very satisfactory results, they can be regarded as a solid basis for future work.

3.9 Real-Time Driver Drowsiness Detection using Computer Vision

Mahek Jain, Bhavya Bhagerathi, Sowmyarani C N

Abstract:

In addition, it is a system proposed to improve traffic safety by reducing accidents caused by driver's drowsiness and fatigue. This is the most common cause of recent accidents. Certain faces, eyes and gestures, including fatigue and yawning, are considered signs of driver drowsiness and fatigue. This is an important sign that the driver is not feeling very well. Eye Aspect Ratio (EAR) detects drowsiness by measuring the distance between horizontal and vertical landmarks. For yawn detection, the yawn value is measured using the distance between the lower lip and the upper lip, and this distance is compared with a threshold value. They introduced an eSpeak module (text-to-speech synthesizer) that provides an audible alert when the driver feels sleepy or yawning. The proposed program is designed to contribute to technologies that reduce the risk of accidents and prevent deaths from road traffic collisions.

3.10 DRIVER DROWSINESS DETECTION SYSTEM

Prof.Ankita V. Karale, 2Nikita P. Patil, 3Prajakta M. Sarvar, 4Pritam M. Sangle, 5Shila A. Shinde

METHODOLOGY:

Input Image:

This block contains sample images of the driver working with our system. Images will work as embedded in our system as well as it will help to identify the facial expressions and eye pattern of the driver in question. The photos taken are as important as they are and play a major role in output production. These images are then subject to image processing, Eigen algorithm etc. to determine if the driver is drowsiness or not.

Pre-processing:

In this block the main focus is on the image. In some cases, the image may be distorted or misinterpreted or it may even be an image that needs to be upgraded as needed. The process of fixing parts that are not in the picture or making the necessary enhancements takes place in this block. Includes similar functions (e.g., rotation, measurement, translation).

ROI (Interest Region):

When a face is detected its ROI is detected and processed. In the second stage the driver's drowsiness is detected by closing the foreskin. The driver's eye is monitored continuously and if it is found to be closed for a period of time the alarm goes off.

Eye Acquisition:

The total number of frames where the eyes are closed is visible. If this private number exceeds a certain limit, the diver will get a visual warning on the navigation mirror indicating that you are drowsy. Then an alarm will be generated.

Creating a Tie Box:

As we all know that a binding box is an imaginary rectangle that acts as a reference point for the acquisition of an object and creates a collision box that. Annotations of the data place these rectangles over the images, defining the object of interest between each image by defining the X and Y links as used in our system the brain box is created for visualization and the conflict box is created with its reference.

Tracking recognition:

Key face detection, detection, and tracking features and functions include: It gets a face Recording a photo file and seeing a face in it can be done here.

Face recognition:

By using a facial model you can see the face in the picture.

Face tracking:

Face tracking using camera preview images, from somewhere in the photo can be done here.

Sleep Warning:

This episode comes in or has a role to play especially if a certain driver is found to be drowsy. If the driver drowsy process to generate a warning alarm is performed here.

3.11 Driver Drowsiness Detection System Based on Visual Features

S. Jhansi Rani1 , Anand Rajasekaran 2 , Chandrasekhar A R 3 1Assistant Professor, Department of Information Technology, Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India

Initial Camera Setup:

The first step in the process would be to set up a driver-facing camera so we can successfully photograph the driver's face for further processing. The camera should be set in such a way that it does not interfere, that is, it does not enter the driver's seat while on the road again it should be placed properly so that the face of the shot is clear so it gives accurate results. A Raspberry Pi can be used to assemble parts in such a system, but raspberryPi comes with a low amount of RAM, all load to use all system functions, GUI display,and the process of handling the integration of performance can be too much. If dlib is compiled in aRaspberry Pi, an error message will be displayed. Butthis can be fixed if we update our Raspberry Pi system while seeking high memory and refreshment resize the file. The camera can also be connected to a standard laptop. After this hardware suspension once done, we can move on to the sleep detector, an algorithm for identifying drowsy drivers with caution video streaming from camera.

Face Discovery:

We use the Historical Feature of Oriented Gradients (HOG). descriptor that uses the distribution of directives for gradients as features and tend to be more accurate and faster in performance than other algorithms. Histogram of Oriented Gradients for Human Detection has shown that Image Histogram of Oriented Gradients (HOG). In this case, we will be tuning it to find faces. Principles followed by the HOG dictionary that the appearance and shape of a local object can often be seen very well in the area

distribution of gradients or directions for edges, even without the direct knowledge of the corresponding gradient or edge areas. In practice, this can be done by dividing the image window into smaller local regions (— Cells‖), in all cells that accumulate 1-D space histogram of gradient directions or edge shape over cell pixels. Combined entries for the histogram will eventually make a presentation. For the sake of consistency in the light, shadow, etc., is also helpful for comparison-standard local responses before using them. This could be done successfully by dividing the area scale of a histogram —power over a large area (—blocks) and applying the results from this to align all cells within the block. I standard descriptive blocks will be called Oriented Gradient Histogram (HOG) commentators. By closing the viewing window by dense (actually, overflowing) HOG definition definition grid and using the vector of the feature included in a Linear SVM based window standard

Face Discovery and Discharge:

Face detection local symbols can be made in several ways, such as right eyebrow, left eyebrow, righteye, left eye, nose, mouth and jaw. We use faces Global signal signal algorithm the launch of the One Millisecond Face Alignment with the Ensemble of Regression Trees. This detector algorithm is part of the dlib library. This method works by hand labeling,specific (x, y) links to surrounding regions, each facial structure and use of this set of professional facial features in the image. This detector is available in the dlib library and measures 68 (x, y).

Drowsiness Evaluation and countermeasures:

After successfully calculating E.A.R, we can use that number to test the driver's drowsiness. The value of E.A.R remains the same when the eye of the driver is turned on, but begins to decrease in value near zero when the eye begins to close. E.A.R is consistent in head and body posture. So, by using these findings we can classify the condition of the eyes as closed when E.A.R is zero or close to zero, otherwise the state is identified as open. The last part makes the decision sound alarm or not. The

average eye length blink is 100-400 milliseconds, so if the driver is right in a state of sleep, their time to close their eyes is beyond this period. In our system, the limit is set for 5 seconds, and if this exceeds the alarm sounded and a notification will appear in this regard.

3.12 Automatic driver distraction detection using deep convolutional neural networks

Md. Uzzal Hossain Md. Ataur Rahman Md. ManowarulIslamaArnishaAkhteraMd. Ashraf Uddin Bikash KumarPaul

Methodology:

In the proposal, first, the image is processed in advance to train the model in different driver positions including, food, texting, talking to others. Second, the pre-trained Convolutional model contains CNN-based in-depth learning formats including ResNet50, MobileNetV2 adopted to detect disturbed driver behavior. Finally, the model is tested using images of test data and we analyze the results of the proposal.

SL	Properties	VGG-16	ResNet50	MobileNetV2
1.	image	224×224×3	224×224×3	224×224×3
2.	weight	Imagenet	Imagenet	imagenet
3.	size	528	98	14
4.	Total layers	16	50	53
5.	Convolution layer	13	48	53
6.	Max pool	5	1	1
7.	Activation function	Softmax	Softmax	Softmax
8.	Total parameters	138.3 million	25.6million	3.5 million
Advantages/Limitations				
VGG16	• It is painfully slow to train. • Weights are quite large.			
ResNet50	• Lower complexity than VGG-16. • Deeper than VGG-16.			
MobileNetV2	• Faster than others. • Low-power models parameterized to meet the resource constraints. • Preferable in vision-based mobile and embedded applications.			

Fig 1

CHAPTER 4

DATA

This chapter serves to describe the data under consideration. Understanding the way all the data work is vital in the process of creating a good solution to the problem at hand.

Overview for Distraction Model Dataset

The data set has driver images, taken in a car showing the driver engaging in activities other than driving (texting, drinking, talking on the phone, makeup, reaching behind, etc). The dataset is State Farm Distracted Driver Dataset from the American University of Cairo(AUC) and it has been obtained from Kaggle.

Total images : 102150 Training image: 22424(Labeled) Testing images : 79726

Images are coloured and Dimension: 640 x 480 pixels The 10 classes to predict are:

C0: Safe driving C1: texting - left

Fig 2

C2: talking on the phone - left

C3: texting - right

Fig 3

C4 : Talking on the phone - right

C5 - Operating the Radio

Fig 4

C6 - Drinking

C7 - Reaching Behind

Fig 5

C8 - Hair and Makeup

C9 - Talking to passenger

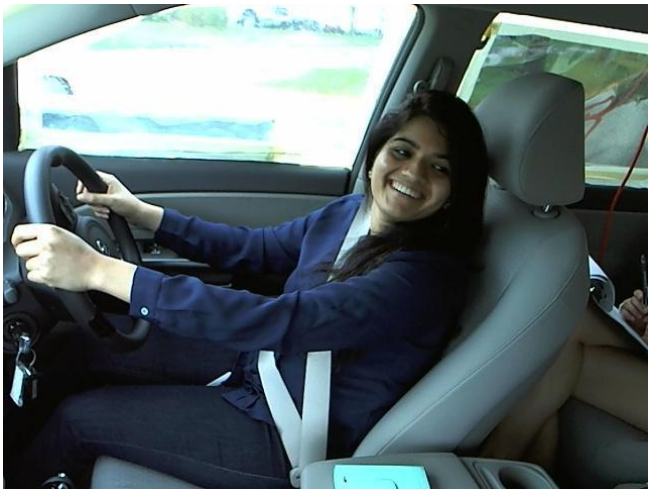


Fig 6

Overview for Drowsiness Model Dataset:

The data set has driver images, taken in a car with a driver drowsy during driving (Eyes closed, eyes open, yawning, mouth closed). This dataset is obtained from Kaggle.

Total images : 2763 Training images : 751 (Labeled) Testing images : 2012

Images are coloured and Dimension: 640 x 480 pixels

There are 4 classes:

Eyes closed: Eyes open:

Fig 7

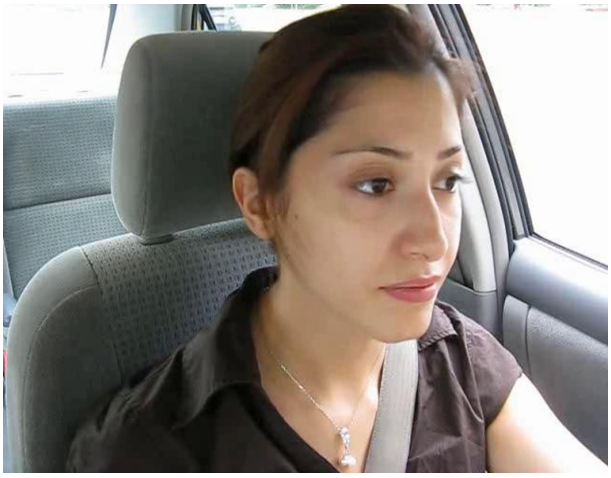


Fig 8

CHAPTER 5

METHODOLOGY

This chapter comprehensively details the methodology used to solve the problem at hand.

5.1 Overview

We briefly discuss the methods employed for our two main modules : Drowsiness Detection and Distraction Detection of the driver. The overall gist of the methodology involves detection of whether the driver is drowsy by tracking key facial features of the driver viz eyes and mouth. The blinking rate of the eye alongside the duration of the yawn is considered. On the other hand, for distraction detection we consider 10 classes of distractions which our model will classify in accordance to the drivers actions as captured by the live video feed. In both cases, a camera captures a live video and we extract frames from said video which are further input into their respective models for classification.

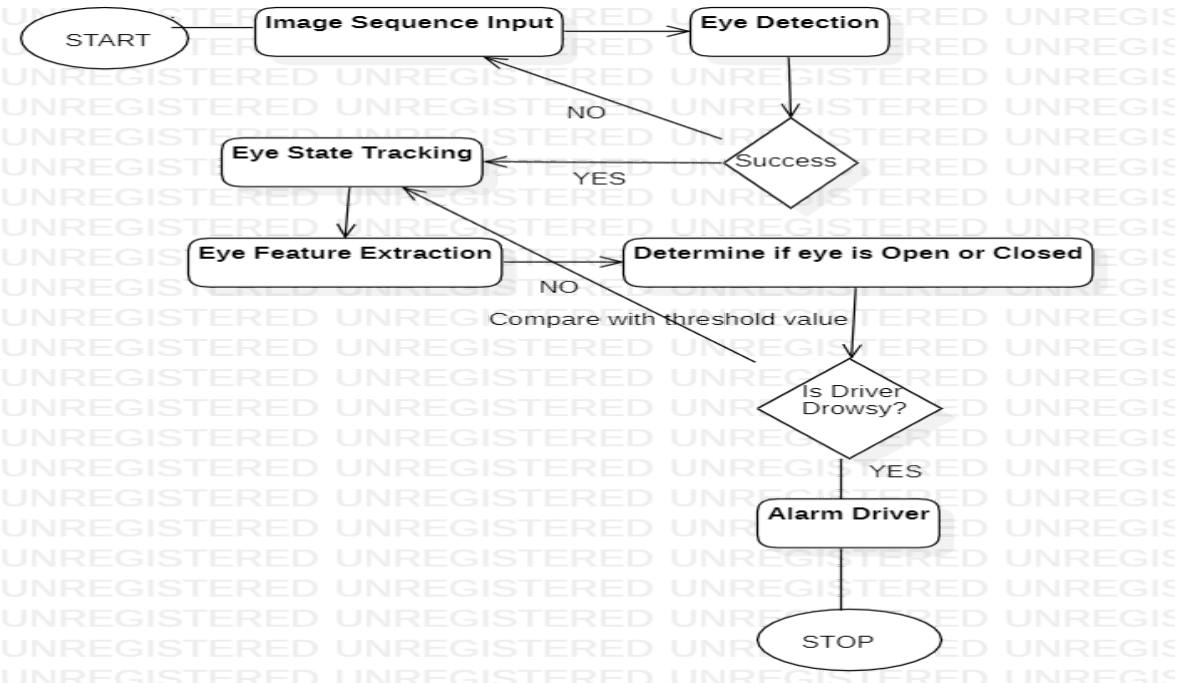


Fig. 9. Sequence Diagram for Drowsiness Detection

An alarm is rung in both cases in order to alert the driver and various pre-recorded messages are also played telling the driver to halt or to take some rest for a while.

5.2 Drowsiness Detection

Drowsiness Detection of the driver involves tracking the driver's eyes and mouth and identifying whether the eyes are closed or open, identifying whether the mouth is closed or open.
If eyes are closed for more than the threshold time, the driver is drowsy.
If the driver yawns more than the threshold value, the driver is drowsy.

To achieve this goal, we built an inception model with an accuracy of 95%.

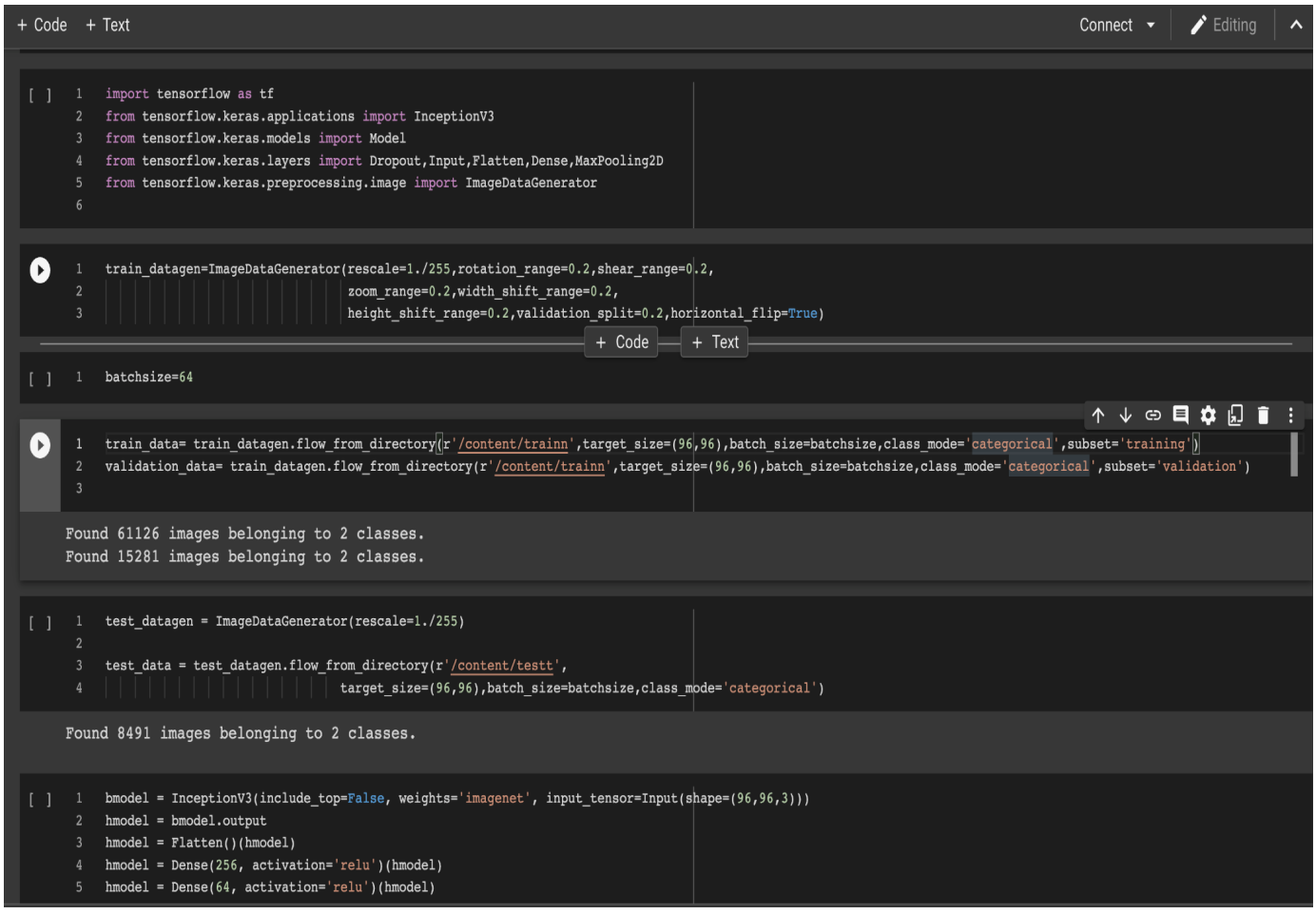
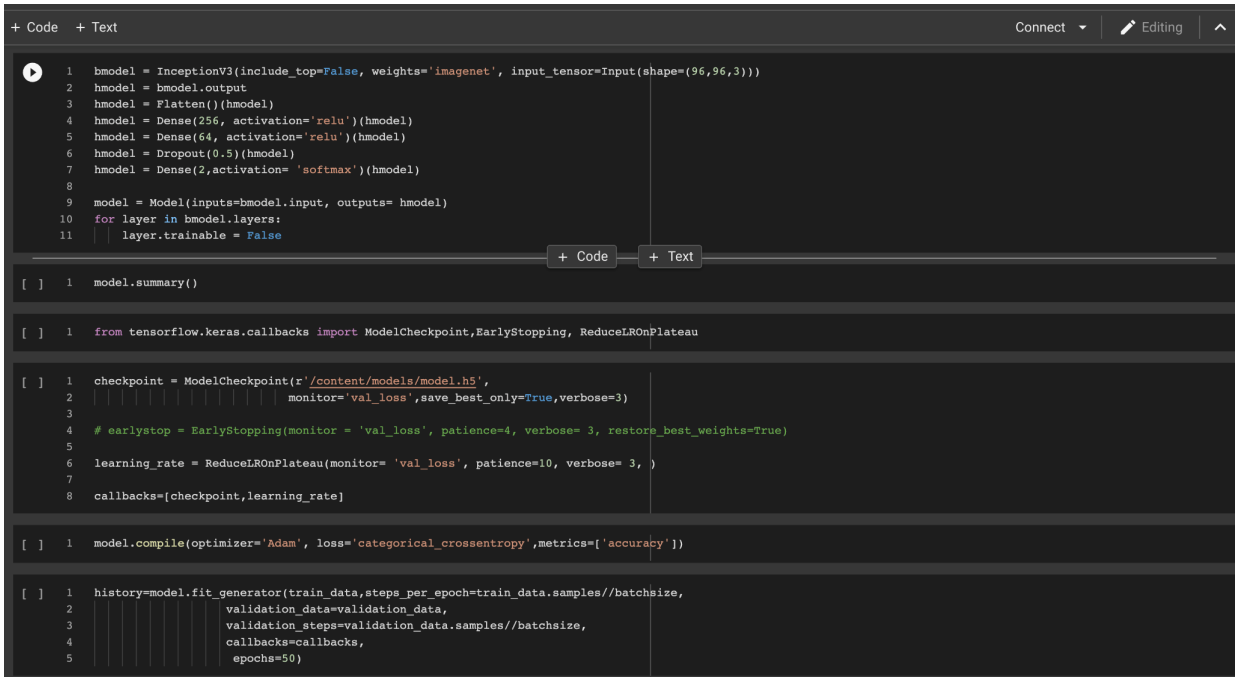


Fig 10



```
+ Code + Text                                                    Connect   Editing

bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(96,96,3)))
hmodel = bmodel.output
hmodel = Flatten()(hmodel)
hmodel = Dense(256, activation='relu')(hmodel)
hmodel = Dense(64, activation='relu')(hmodel)
hmodel = Dropout(0.5)(hmodel)
hmodel = Dense(2,activation= 'softmax')(hmodel)

model = Model(inputs=bmodel.input, outputs= hmodel)
for layer in bmodel.layers:
    layer.trainable = False

+ Code + Text

[ ]  model.summary()

[ ]  from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPlateau

[ ]  checkpoint = ModelCheckpoint(r'/content/models/model.h5',
                                  monitor='val_loss',save_best_only=True,verbose=3)

     # earlystop = EarlyStopping(monitor = 'val_loss', patience=4, verbose= 3, restore_best_weights=True)

     learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=10, verbose= 3, )

     callbacks=[checkpoint,learning_rate]

[ ]  model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])

[ ]  history=model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
                                 validation_data=validation_data,
                                 validation_steps=validation_data.samples//batchsize,
                                 callbacks=callbacks,
                                  epochs=50)
```

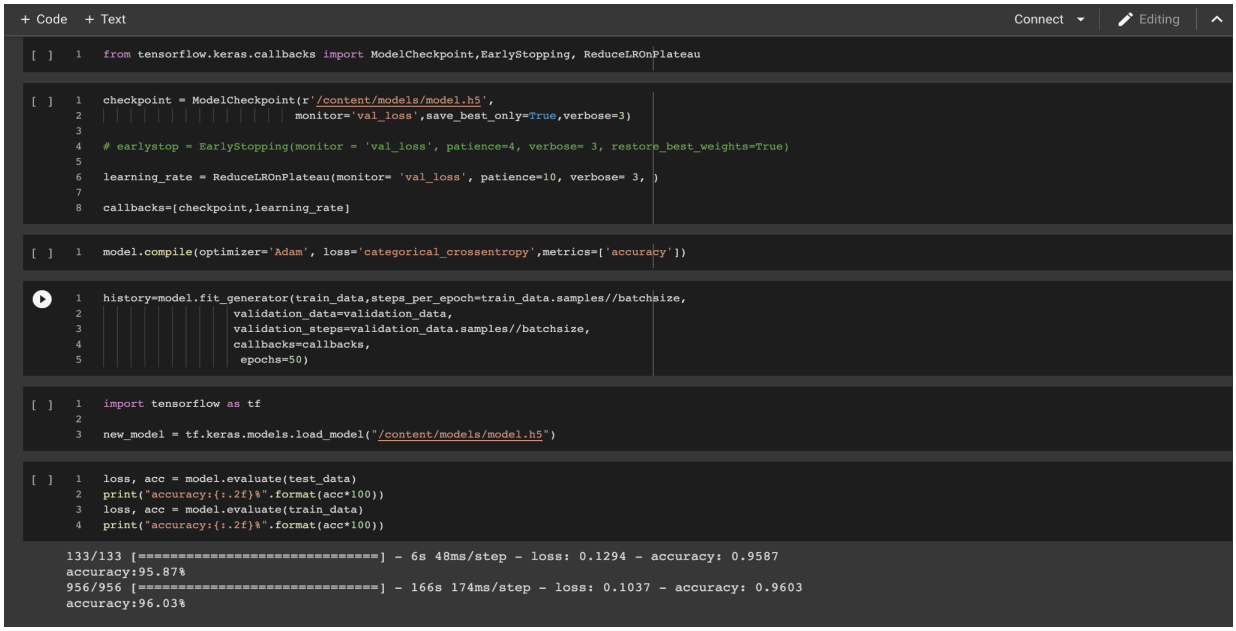
Fig 11

The first dense layer has the output space of 256 and “RELU” activation function is used.

The second dense layer has the output space of 64 and “RELU” activation function is used.

Then a dropout layer of 0.5 value is added to reduce the contribution of some neurons.

The third dense layer has the output space of 2 and “SOFTMAX” activation function is used



```
+ Code + Text                                                    Connect   Editing

[ ]  from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPlateau

[ ]  checkpoint = ModelCheckpoint(r'/content/models/model.h5',
                                  monitor='val_loss',save_best_only=True,verbose=3)

     # earlystop = EarlyStopping(monitor = 'val_loss', patience=4, verbose= 3, restore_best_weights=True)

     learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=10, verbose= 3, )

     callbacks=[checkpoint,learning_rate]

[ ]  model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])

     history=model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
                                 validation_data=validation_data,
                                 validation_steps=validation_data.samples//batchsize,
                                 callbacks=callbacks,
                                  epochs=50)

[ ]  import tensorflow as tf

     new_model = tf.keras.models.load_model("/content/models/model.h5")

[ ]  loss, acc = model.evaluate(test_data)
     print("accuracy:{:.2f}%".format(acc*100))
     loss, acc = model.evaluate(train_data)
     print("accuracy:{:.2f}%".format(acc*100))

133/133 [==============================] - 6s 48ms/step - loss: 0.1294 - accuracy: 0.9587
accuracy:95.87%
956/956 [==============================] - 166s 174ms/step - loss: 0.1037 - accuracy: 0.9603
accuracy:96.03%
```

Fig 12

But there was 1 big issue with this model, the size was too large for a 4GB Raspberry Pi to handle. Therefore, the Raspberry Pi started lagging while taking frames and that is not acceptable as the result should be coming in an instant else it is of no use.

So we decided to use an in-built library in python.

DLIB library uses a 68-Landmark points method to detect the face of the person in an image.

```
24  c=3
25
26  def alarm(msg):
27      global alarm_status
28      global alarm_status2
29      global saying
30
31      while alarm_status:
32          print('call')
33          s = 'espeak "'+msg+'"'
34          os.system(s)
35
36      while alarm_status2:
37          print('call')
38          saying = True
39          s = 'espeak "' + msg + '"'
40          os.system(s)
41          saying = False
42
43  def eye_aspect_ratio(eye):
44          A = dist.euclidean(eye[1], eye[5])
45          B = dist.euclidean(eye[2], eye[4])
46          C = dist.euclidean(eye[0], eye[3])
47          ear = (A + B) / (2.0 * C)
48          return ear
49
50  def final_ear(shape):
51      (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
52      (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
53
54      leftEye = shape[lStart:lEnd]
55      rightEye = shape[rStart:rEnd]
56
57      leftEAR = eye_aspect_ratio(leftEye)
58      rightEAR = eye_aspect_ratio(rightEye)
59
60      ear = (leftEAR + rightEAR) / 2.0
61      return (ear, leftEye, rightEye)
62
```

Fig 13

```
while True:

    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        #rects = detector(gray, 0)
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
            minNeighbors=5, minSize=(30, 30),
            flags=cv2.CASCADE_SCALE_IMAGE)

        #for rect in rects:
    for (x, y, w, h) in rects:
        rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))

        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

        eye = final_ear(shape)
        ear = eye[0]
        leftEye = eye [1]
        rightEye = eye[2]

        distance = lip_distance(shape)

        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
        lip = shape[48:60]
        cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

        if ear < EYE_AR_THRESH and distance < YAWN_THRESH:
            COUNTER += 1

            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                if alarm_status == False:
                    alarm_status = True
                    t = Thread(target=alarm, args=('wake up ',))
```

Fig 14

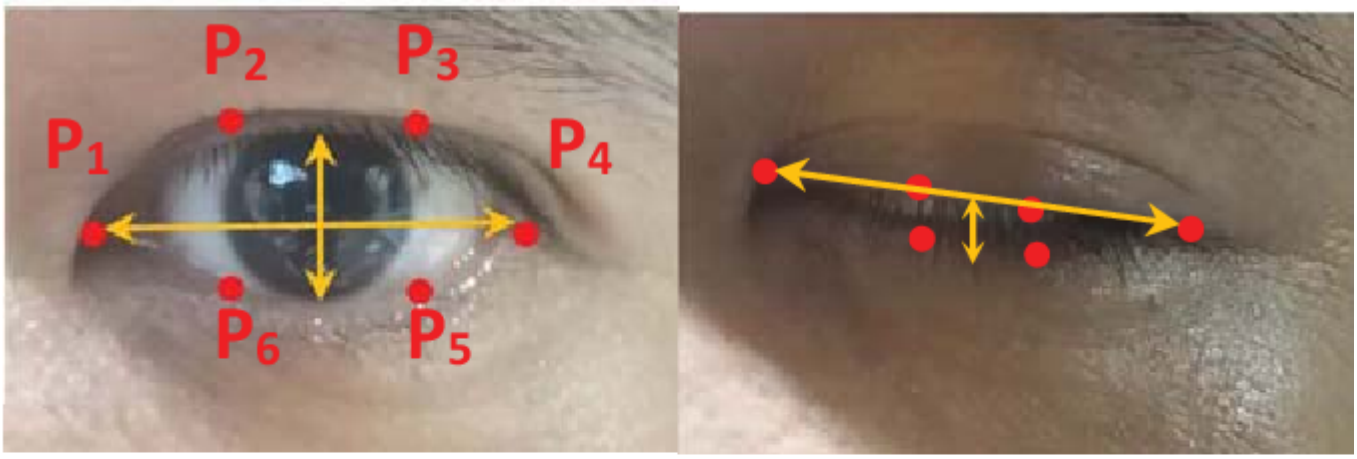
```
        else:
            COUNTER = 0
            alarm_status = False

        if (distance > YAWN_THRESH):
            COUNTER_YAWN += 1
            print(COUNTER_YAWN)
            if COUNTER_YAWN >= YAWN_CONSEC_FRAMES and COUNTER_YAWN <=YAWN_CONSEC_FRAMES :
                if alarm_status2 == False and saying == False:
                    alarm_status2 = True
                    t = Thread(target=alarm, args=('take some fresh air',))
                    t.deamon = True
                    t.start()
                cv2.putText(frame, "Take A Coffee Break!!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
            elif COUNTER_YAWN > YAWN_CONSEC_FRAMES+1:
                    COUNTER_YAWN=0
        else:
            alarm_status2 = False


        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
        cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)


    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

Fig 15

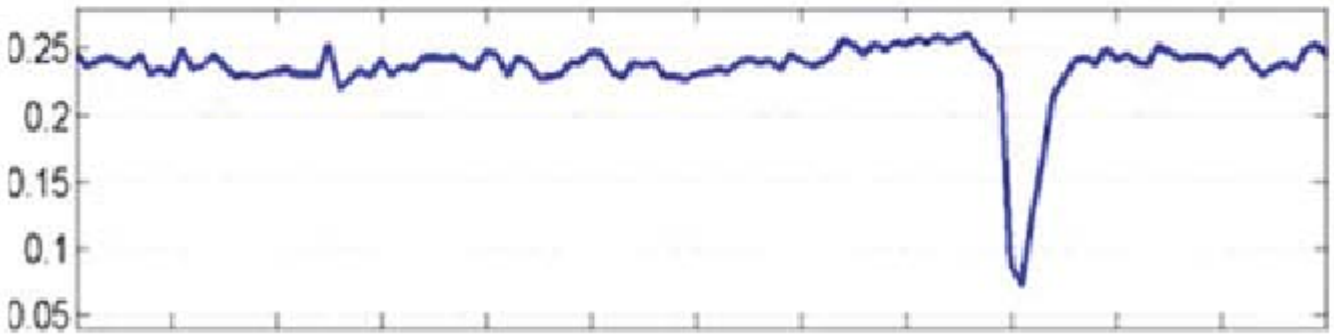
We have used the Eye Aspect Ratio (EAR) to check whether the eye and mouth are open or closed.

If eyes are closed for more than the threshold time, the driver is drowsy.

If the driver yawns more than the threshold value, the driver is drowsy.



(a)



(b)

Fig 16

5.3 Distraction Detection

Distraction Detection of the driver involves tracking the driver's action and the position in which they are sitting and identifying whether they fall under the 10 major classes of distraction. As per the State Farm Distracted Driver Dataset (22424 images) , which was created by the American University of Cairo (AUC) has named 10 classes of distractions that a driver can be found exhibiting while driving.

Following are the 10 major classes of distraction:

C0- Safe Driving	C5 - Operating the Radio
C1- Texting- Left	C6 - Drinking
C2 - Texting on the phone- Left	C7 - Reaching Behind
C3 - Texting- Right	C8 - Hair and Makeup
C4 - Talking on the phone- Right	C9 - Talking to passenger

Images of 26 drivers have been taken showing all kinds of distractions exhibited with each driver having an average of 862 images.

5.3.1 Data Preprocessing and Loading

We read images from the respective path using python library CV2. We read the images in their original RGB form and we resize the images to 150 x 150 (rows and cols) since deep learning and machine learning models train faster on smaller images. We load the images from the training dataset and we append the images and their respective labels to a list. This will act as our training images and training labels. The next most important step is to normalize our data. It is a process by which we make our data either dimensionless or make sure that they have similar distributions throughout. It is known to improve model performance greatly. Normalizing our data will ensure that every variable is given equal importance or weight and thereby preventing any singular variable from steering the performance of the model.
At this stage we reshape our data, turn them into numpy arrays and we conduct a train-test split of 80-20 which will be used later for testing our model performance.

We repeat the same steps for our test dataset and therefore create our validation data for testing of our model performance. We have a variable for mapping of our 10 labels. This constitutes the common data preprocessing that is done prior to training with our models. Initially we implemented a simple scratch

CNN followed by two pretrained models i.e VGG-16 and Inception-V3. Further, an average ensemble is conducted on the two pretrained models and the best performing of the two is taken as our final model for multiclass classification.

5.3.2 CNN Model

We implemented our CNN model using 6 convolutional layers along with the ReLu activation function which has an advantage over other activation functions in a way that it does not activate all neurons at the same time. There is no problem of vanishing gradient which makes the accuracy more efficient and faster. The batch size is 40 and it trains for 10 epochs. We add batch normalization after every convolutional layer and a dense layer after every second convolutional layer which helps in changing the dimensionality of the outputs of this layer and we add a standard dropout layer to reduce our number of inputs for the next set of convolutional layers. We end the model with a softmax activation function to create a multinomial probability distribution for multiclass classification. We use categorical loss entropy, which is a combination of a softmax activation function and categorical loss entropy and is mainly used for multiclass

classification.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        320

batch_normalization (BatchN  (None, 62, 62, 32)        128
ormalization)

conv2d_1 (Conv2D)            (None, 62, 62, 32)        9248

batch_normalization_1 (Batc  (None, 62, 62, 32)        128
hNormalization)

max_pooling2d (MaxPooling2D  (None, 31, 31, 32)        0
)

dropout (Dropout)            (None, 31, 31, 32)        0

conv2d_2 (Conv2D)            (None, 29, 29, 64)        18496

batch_normalization_2 (Batc  (None, 29, 29, 64)        256
hNormalization)

conv2d_3 (Conv2D)            (None, 29, 29, 64)        36928

batch_normalization_3 (Batc  (None, 29, 29, 64)        256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 15, 15, 64)        0
2D)

dropout_1 (Dropout)          (None, 15, 15, 64)        0

conv2d_4 (Conv2D)            (None, 13, 13, 128)       73856

batch_normalization_4 (Batc  (None, 13, 13, 128)       512
hNormalization)

conv2d_5 (Conv2D)            (None, 13, 13, 128)       147584

batch_normalization_5 (Batc  (None, 13, 13, 128)       512
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 7, 7, 128)         0
2D)

dropout_2 (Dropout)          (None, 7, 7, 128)         0
```

```
flatten (Flatten)            (None, 6272)              0

dense (Dense)                (None, 512)               3211776

batch_normalization_6 (Batc  (None, 512)               2048
hNormalization)

dropout_3 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 128)               65664

batch_normalization_7 (Batc  (None, 128)               512
hNormalization)

dropout_4 (Dropout)          (None, 128)               0

dense_2 (Dense)              (None, 10)                1290

=================================================================
Total params: 3,569,514
Trainable params: 3,567,338
Non-trainable params: 2,176
_________________________________________________________________
<keras.engine.sequential.Sequential object at 0x7f7505810110>
```

Fig 17. CNN Model Architecture

5.3.3 VGG-16 Model

We employ the same data preprocessing methods as stated earlier and then we use a pretrained model called VGG-16. This model consists of 16 layers with certain weights (imagenet). It has convolutional layers and max pooling layers. It consists of 14 million total parameters which are all trainable.

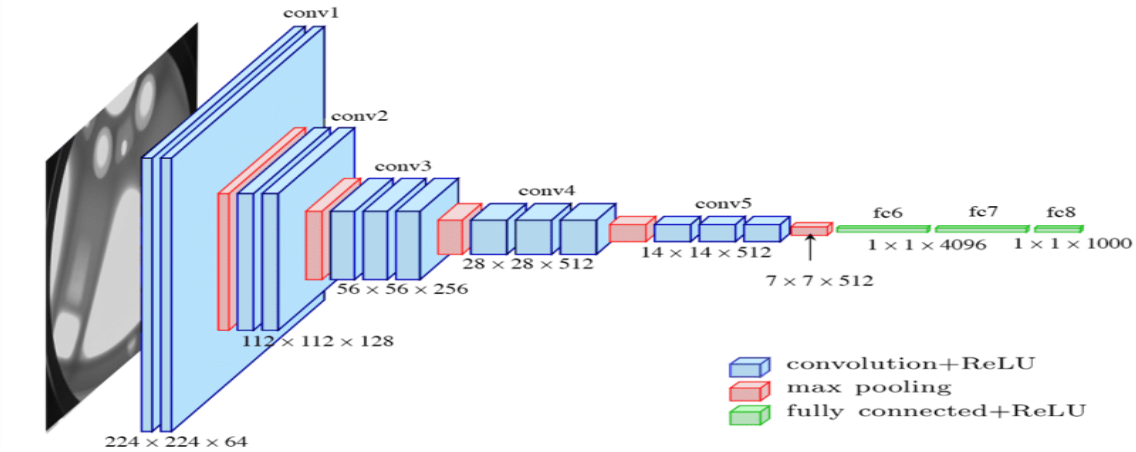


Fig.18. Standard VGG-16 Architecture

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Image_input (InputLayer)    [(None, 150, 150, 3)]     0

 block1_conv1 (Conv2D)       (None, 150, 150, 64)      1792

 block1_conv2 (Conv2D)       (None, 150, 150, 64)      36928

 block1_pool (MaxPooling2D)  (None, 75, 75, 64)        0

 block2_conv1 (Conv2D)       (None, 75, 75, 128)       73856

 block2_conv2 (Conv2D)       (None, 75, 75, 128)       147584

 block2_pool (MaxPooling2D)  (None, 37, 37, 128)       0

 block3_conv1 (Conv2D)       (None, 37, 37, 256)       295168

 block3_conv2 (Conv2D)       (None, 37, 37, 256)       590080

 block3_conv3 (Conv2D)       (None, 37, 37, 256)       590080

 block3_pool (MaxPooling2D)  (None, 18, 18, 256)       0

 block4_conv1 (Conv2D)       (None, 18, 18, 512)       1180160

 block4_conv2 (Conv2D)       (None, 18, 18, 512)       2359808

 block4_conv3 (Conv2D)       (None, 18, 18, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 9, 9, 512)         0

 block5_conv1 (Conv2D)       (None, 9, 9, 512)         2359808

 block5_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

 block5_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Fig 19. VGG-16 Architecture for Distraction Detection

```
output_vgg16_conv = model_vgg16_conv(vgg16_input)

#Add the fully-connected layers

x = Flatten(name='flatten')(output_vgg16_conv)
# x = Dense(4096, activation='relu', name='fc1')(x)
# x = Dense(4096, activation='relu', name='fc2')(x)
x = Dense(10, activation='softmax', name='predictions')(x)

vgg16_pretrained = Model(inputs = vgg16_input, outputs = x)
vgg16_pretrained.summary()

# Compile CNN model
sgd = tf.keras.optimizers.SGD(lr = 0.001)
vgg16_pretrained.compile(loss='categorical_crossentropy',optimizer = sgd,metrics=['accuracy'])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Image_input (InputLayer)    [(None, 150, 150, 3)]     0

 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 flatten (Flatten)           (None, 8192)              0

 predictions (Dense)         (None, 10)                81930

=================================================================
Total params: 14,796,618
Trainable params: 14,796,618
Non-trainable params: 0
_________________________________________________________________
```

Fig. 20 VGG-16

This architecture contains one input layer of shape 150 x 150 (rows and columns), followed by a functional layer. A flatten layer followed by a dense layer of shape 10 to reduce the output to the final 10 categories.

```
score2 = vgg16_pretrained.evaluate_generator(validation_generator, nb_validation_samples // batch_size)
print("Loss for model 1",score1[0])
print("Loss for model 2 (data augmentation):", score2[0])

print("Test accuracy for model 1",score1[1])
print("Test accuracy for model 2 (data augmentation):", score2[1])
```

```
<ipython-input-32-9c4272940d94>:1: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in
  score2 = vgg16_pretrained.evaluate_generator(validation_generator, nb_validation_samples // batch_size)
Loss for model 1 0.04361744225025177
Loss for model 2 (data augmentation): 0.07111914455890656
Test accuracy for model 1 0.9921395182609558
Test accuracy for model 2 (data augmentation): 0.9800000190734863
```

Fig.21 Accuracy and Loss

5.3.4 Inception-V3 Model

We employ the same data preprocessing methods as stated earlier and then we use a pretrained model called Inception-V3. This model consists of 21 million total parameters and 34K untrainable parameters. It consists of a total of 42 layers.

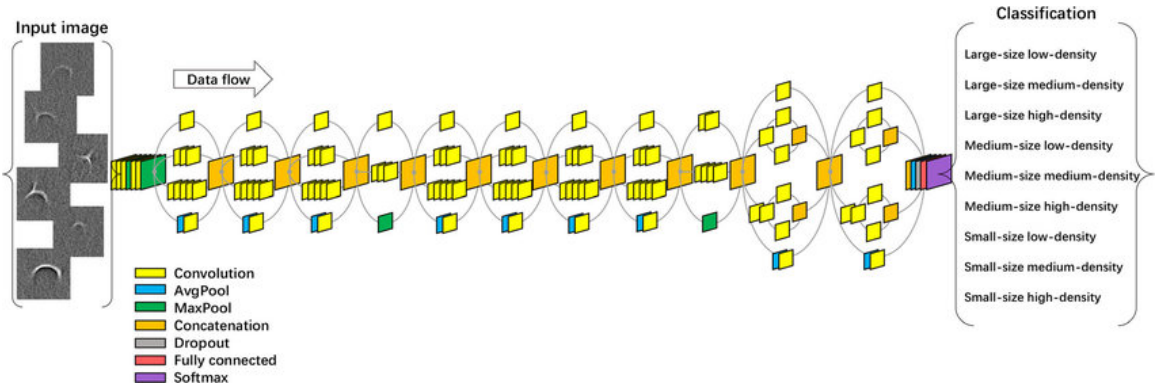
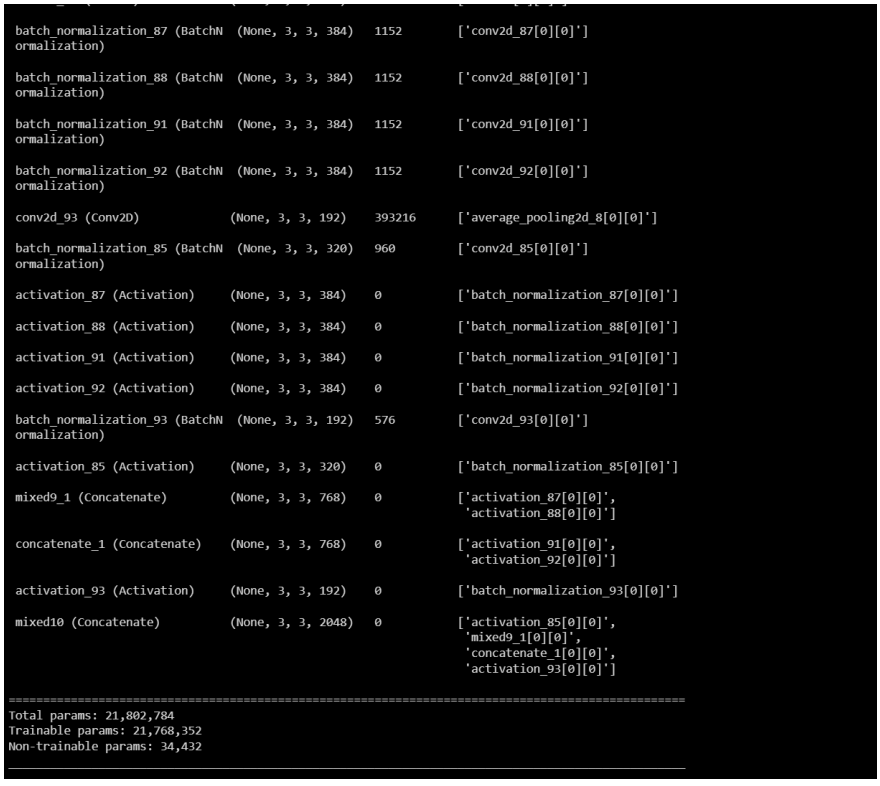
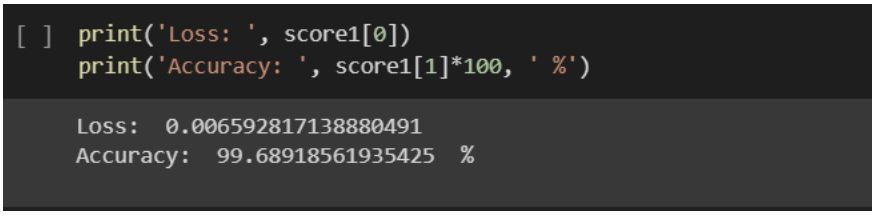


Fig 22. Standard Inception-V3 Architecture



```
batch_normalization_87 (BatchN  (None, 3, 3, 384)   1152        ['conv2d_87[0][0]']
ormalization)

batch_normalization_88 (BatchN  (None, 3, 3, 384)   1152        ['conv2d_88[0][0]']
ormalization)

batch_normalization_91 (BatchN  (None, 3, 3, 384)   1152        ['conv2d_91[0][0]']
ormalization)

batch_normalization_92 (BatchN  (None, 3, 3, 384)   1152        ['conv2d_92[0][0]']
ormalization)

conv2d_93 (Conv2D)              (None, 3, 3, 192)   393216      ['average_pooling2d_8[0][0]']

batch_normalization_85 (BatchN  (None, 3, 3, 320)   960         ['conv2d_85[0][0]']
ormalization)

activation_87 (Activation)      (None, 3, 3, 384)   0           ['batch_normalization_87[0][0]']

activation_88 (Activation)      (None, 3, 3, 384)   0           ['batch_normalization_88[0][0]']

activation_91 (Activation)      (None, 3, 3, 384)   0           ['batch_normalization_91[0][0]']

activation_92 (Activation)      (None, 3, 3, 384)   0           ['batch_normalization_92[0][0]']

batch_normalization_93 (BatchN  (None, 3, 3, 192)   576         ['conv2d_93[0][0]']
ormalization)

activation_85 (Activation)      (None, 3, 3, 320)   0           ['batch_normalization_85[0][0]']

mixed9_1 (Concatenate)          (None, 3, 3, 768)   0           ['activation_87[0][0]',
                                                                 'activation_88[0][0]']

concatenate_1 (Concatenate)     (None, 3, 3, 768)   0           ['activation_91[0][0]',
                                                                 'activation_92[0][0]']

activation_93 (Activation)      (None, 3, 3, 192)   0           ['batch_normalization_93[0][0]']

mixed10 (Concatenate)           (None, 3, 3, 2048)  0           ['activation_85[0][0]',
                                                                 'mixed9_1[0][0]',
                                                                 'concatenate_1[0][0]',
                                                                 'activation_93[0][0]']

==================================================================================================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
__________________________________________________________________________________________________
```

Fig. 23. Inception-V3 Architecture for Distraction Detection



```
[ ]  print('Loss: ', score1[0])
     print('Accuracy: ', score1[1]*100, ' %')

     Loss:  0.006592817138880491
     Accuracy:  99.68918561935425  %
```

Fig.24. Accuracy and Loss

5.3.5 Ensemble - Averaging

To overcome the overfitting issues, we implemented an ensemble model by using an averaging method. We take the mean of predictions made by both Inception-V3 and VGG-16. We achieve an accuracy of 97% and hence we choose Inception-V3 as the model to make predictions of live video capture. We chose inception v3 since it is the closest to the ensemble average.

```
from statistics import mean,median


ensemble_predictions = []
predictions = []

for i in range(len(x_test)):
#for i in range(1):
    mean_prediction = []

    for j in range(len(y_test[0])):
      predictions.append(vgg16_pred[i][j])
      predictions.append(inceptionv3_pred[i][j])

      # trimmed_value = (sum(predictions) - max(predictions) - min(predictions))/(len(predictions) - 2)

      mean_value = mean(predictions)

      predictions = []
      mean_prediction.append(mean_value)

    mean_prediction = mean_prediction/ sum(mean_prediction)
    ensemble_predictions.append(mean_prediction)
```

Fig. 25. Mean of predictions for both models

5.4 Alcohol Detection

We have used Arduino to support the MQ3 sensor.

The components are:

1. Arduino UNO
2. MQ3 sensor
3. Buzzer

The result is shown by Buzzer, when the sensor detects the presence of the ethanol in the air, it'll send a signal to the buzzer. The buzzer value will change from low to high.

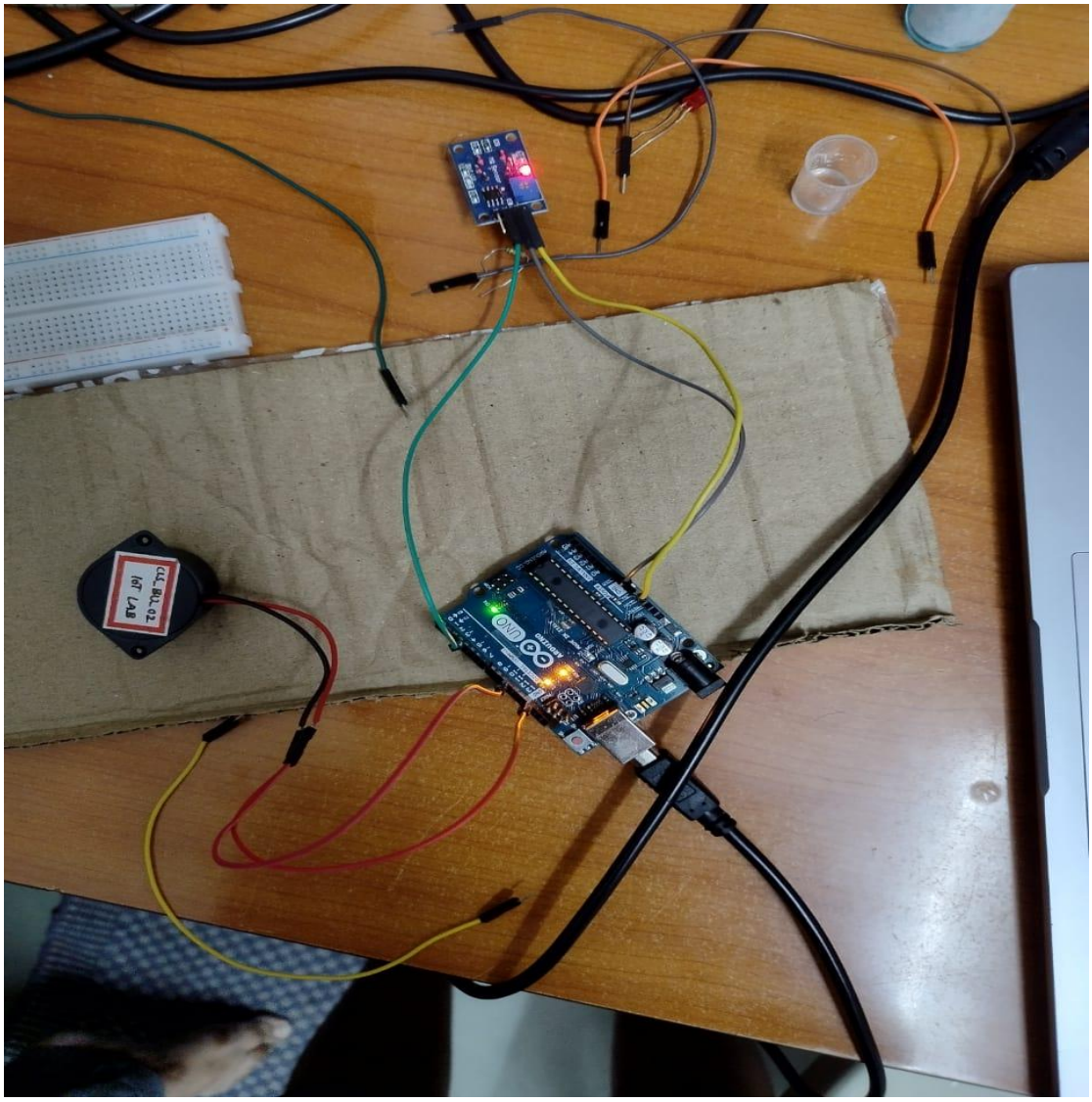


Fig. 26. Alcohol Detection System using Arduino

CHAPTER 6

RESULTS AND DISCUSSION

This chapter is a compilation of all the major results along with associated discussion.

6.1 Drowsiness Detection:

Based on the threshold values provided, if the eyes remain closed for that given duration the driver is detected to be drowsy and he is subsequently alarmed with a buzzer and recorded messages telling him to halt or take rest.

Similarly, based on the threshold values provided, if the driver yawns longer than that then he is detected as drowsy and is alarmed with a buzzer and recorded messages.

This works well for when the driver's face is visible or it's in well lit condition and he is not covering his face. Future scope for this part of the project would be to make this system work under low light conditions and for faces with masks.

Fig 27 Eyes open and Closed

6.2 Distraction Detection:

6.2.1 CNN:

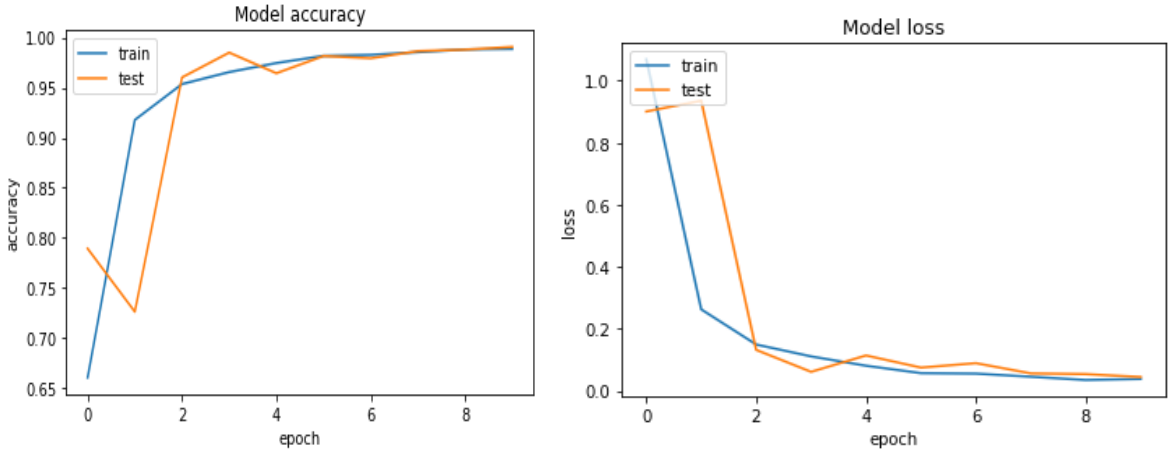


Fig 28

The accuracy achieved is 98% and a loss of 0.04, despite which the model fails to accurately classify the unlabeled images in the testing dataset. This is due to the fact that the training dataset has multiple images of the same person with very slight variations which introduces the problem of data leakage in our model.

With Augmentation:

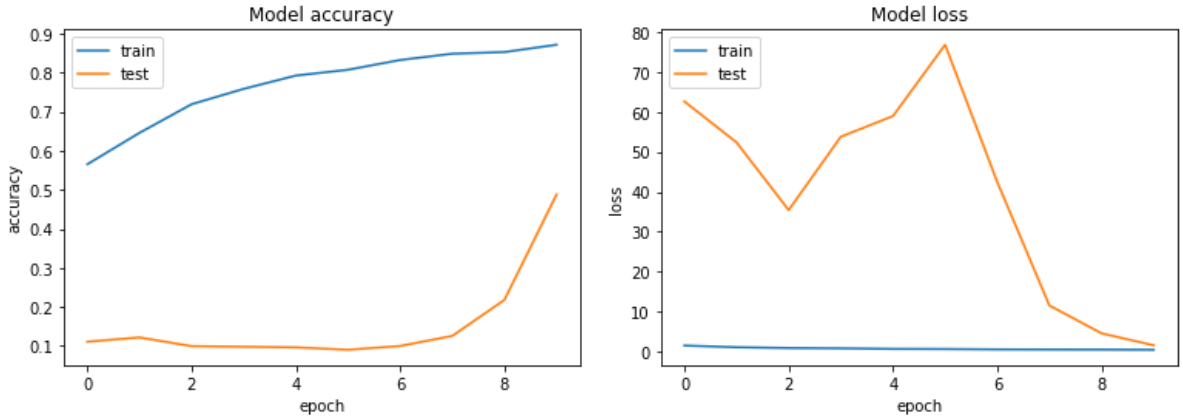


Fig 29

To overcome this we have to use image augmentation and alter the existing images. Image Augmentation is done by rescaling the images and tweaking various parameters of an image to bring about variations in it.

After augmentation, the accuracy on the model is 87% and the loss is 15% which means there is overfitting

in the model. For such reasons we decided to use pretrained models.

6.2.2 VGG-16:

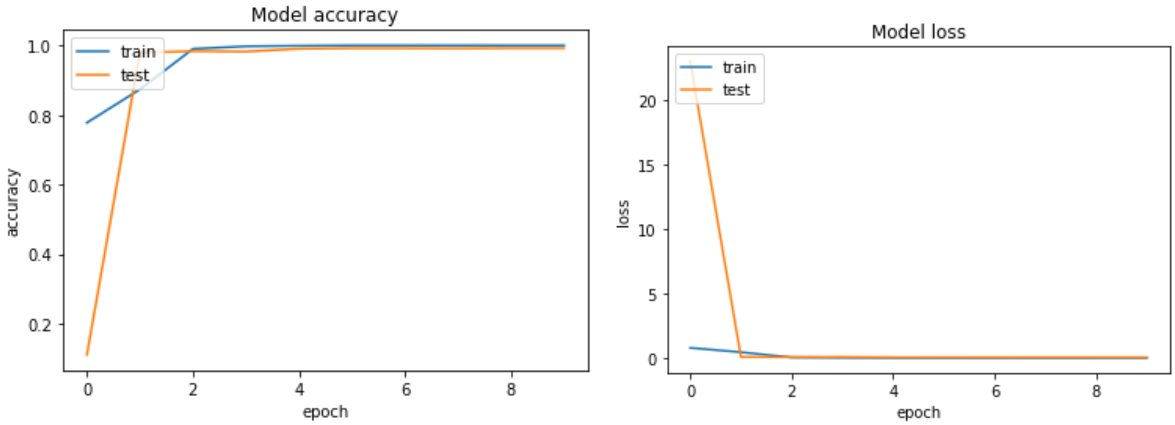


Fig 30

The accuracy achieved is 99% with a loss of 0.04. This model also suffers from the issue of data leakage therefore image augmentation must be applied to our data.

Post augmentation, we get an accuracy of 98% but the validation accuracy is much lower which indicates overfitting of the model.

6.2.3 Inception V3:

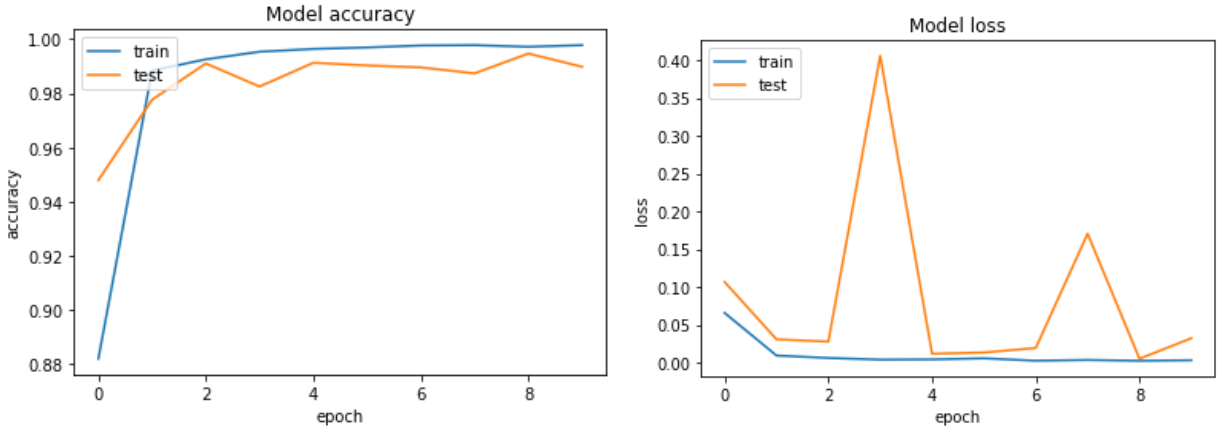


Fig 31

The accuracy achieved with this model is 98% and a loss of 0.04. This model also suffers with data leakage issues and therefore image augmentation is done.

Post augmentation we achieve an accuracy of 80% and a loss of 70% which indicates there is a huge overfitting in the model.

6.2.4 Ensemble Averaging:

We load our model weights i.e. VGG-16 and Inception-V3 and we make predictions using them.
We calculate the mean of the predictions of both models and then we analyze which model performs the best in classifying all 10 categories and its value is closer to the average.The mean for the predictions was 97.5%. Inception-V3 performed better than VGG-16 in identifying all classes and hence we chose that as our final model.

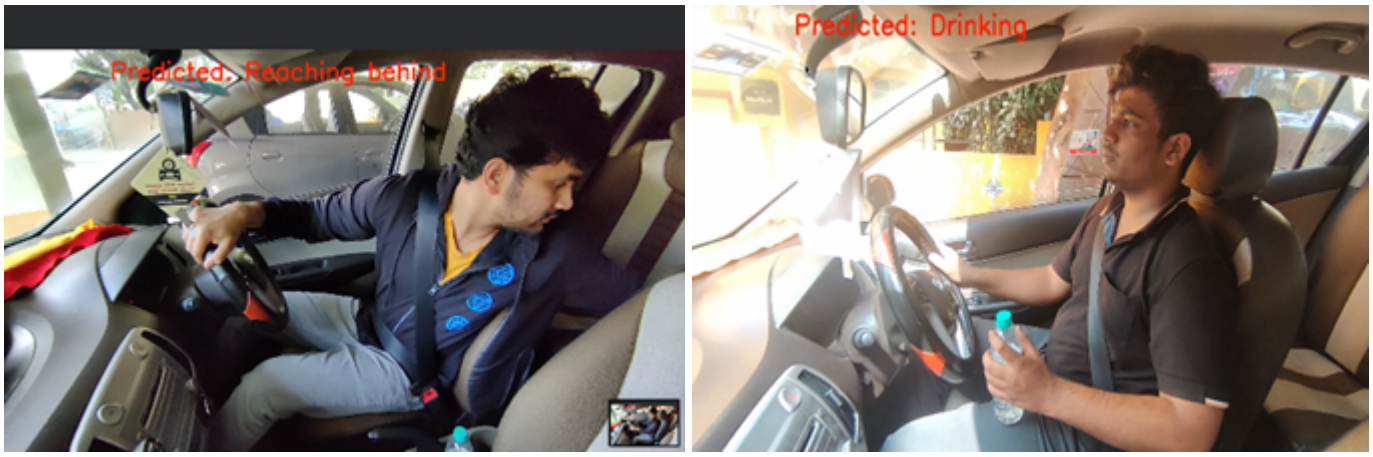


Fig. 32 Distraction Detection

6.3 Alcohol Detection

We have connected the buzzer and the MQ3 sensor to the Arduino. Whenever the sensor detects the presence of ethanol in the air, the value of the buzzer changes to high. It's working properly.

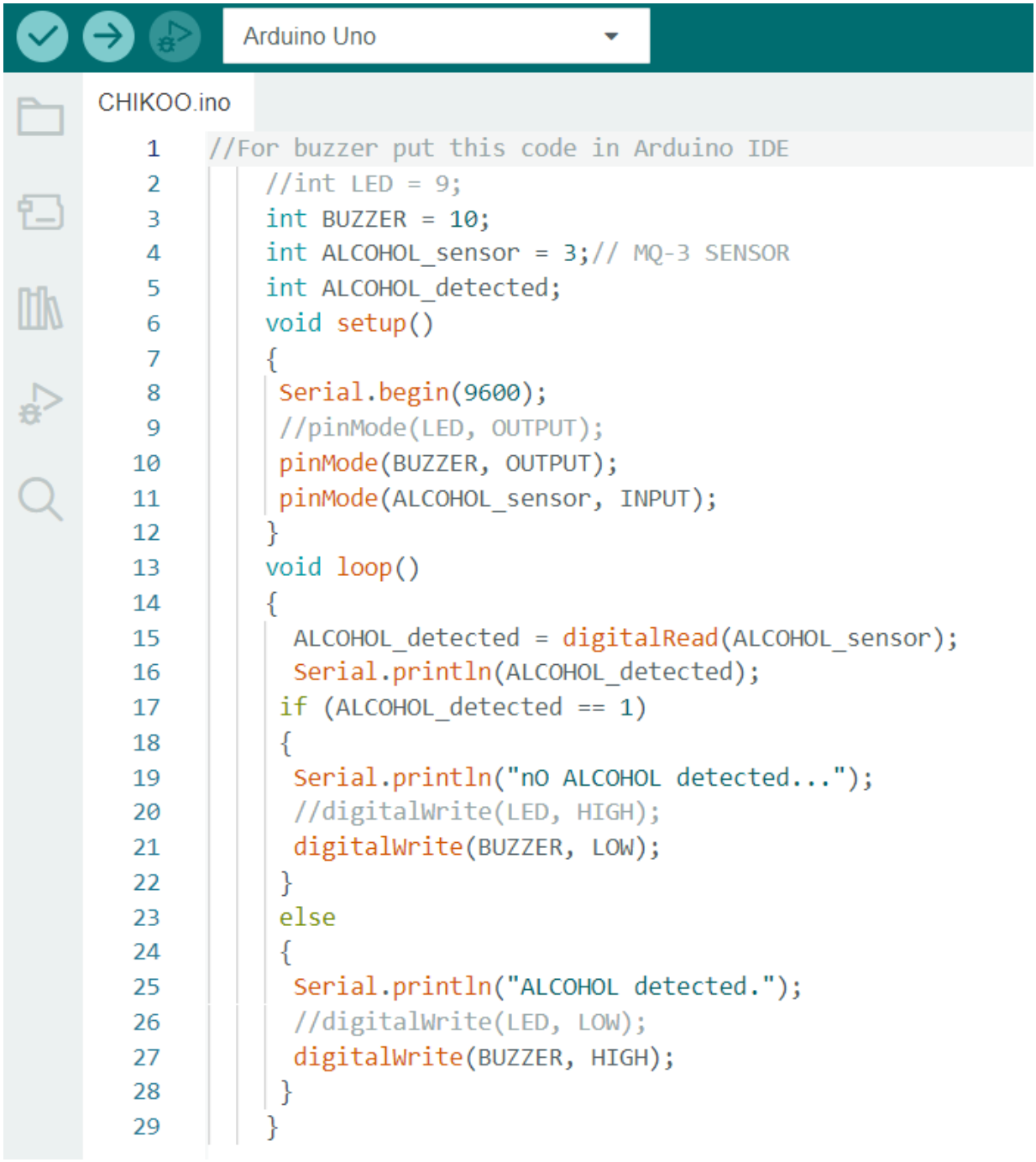

Arduino Uno
CHIKOO.ino
//For buzzer put this code in Arduino IDE
//int LED = 9;
int BUZZER = 10;
int ALCOHOL_sensor = 3;// MQ-3 SENSOR
int ALCOHOL_detected;
void setup()
{
Serial.begin(9600);
//pinMode(LED, OUTPUT);
pinMode(BUZZER, OUTPUT);
pinMode(ALCOHOL_sensor, INPUT);
}
void loop()
{
ALCOHOL_detected = digitalRead(ALCOHOL_sensor);
Serial.println(ALCOHOL_detected);
if (ALCOHOL_detected == 1)
{
Serial.println("nO ALCOHOL detected...");
//digitalWrite(LED, HIGH);
digitalWrite(BUZZER, LOW);
}
else
{
Serial.println("ALCOHOL detected.");
//digitalWrite(LED, LOW);
digitalWrite(BUZZER, HIGH);
}
}

Fig 33

CHAPTER 7

CONCLUSION AND FUTURE WORK

We have prepared a multi-functional project with functionalities as Drowsiness, Distraction and Drunk detection with Drowsiness and Drunk detection working nicely and Distraction Model showing 98% accuracy on the test data.Our project is novel because we have done drowsiness detection using yawning frequency and we are getting the best accuracy in distraction detection.The price of the systems in which future technologies for sensing and measuring arousal are placed will rely on the price of the component sensors.

However, the accuracy of the system will increase as machine learning (deep learning) usage advances with the addition of computers and GPUs, and it is anticipated that the number of installations in vehicles will rise. However, testing in a moving vehicle environment is necessary.

There won't be a need to employ this technology to identify and evaluate driver weariness for accident avoidance in the far future when autonomous driving levels exist and the human driver isn't the primary driver.

So, in the age of autonomous driving, devices for detecting distraction and exhaustion will make drivers feel more at ease inside the car. In these situations, the driver is put to sleep by a system that detects and estimates sleepiness in the driver. The device then enables the driver to get a good night's sleep and travel comfortably.

In other words, drowsiness detection and distraction detection systems in the future will serve as a system that realizes comfortable driving, lessens driver tiredness while operating a vehicle, and enables drivers to act without becoming fatigued when they reach their destination.

In the future, the level 4 and level 5 automated driving sleepiness detection/estimation system will be used not only in cars but also in other sleep-related professions and industries, detecting drowsiness and providing comfort to individuals

BIBLIOGRAPHY

[1] Detection of Driver Drowsiness by Calculating the Speed of Eye Blinking-2021, Muhammad Fawwaz Yusri, Patrick Mangat, and Oliver Wasenmiller

[2] Development of an intelligent drowsiness detection system for drivers using image processing techniques, Suhaiman, A.A., May, Z., Rahman, N.A.A.

[3] Driver drowsiness detection with ANN image processing , T. Vesselenyi et al 2017 IOP Conf. Ser.: Mater. Sci. Eng. 252 01209

[4] Driver drowsiness detection with ANN image processing TO. Vesselenyi et al 2017 IOP Conf. Ser.: Mater. Sci. Eng. 252 01209 https://iopscience.iop.org/article/10.1088/1757-899X/252/1/012097

[5] Distracted Driver Detection Published in 2020,Deep Ruparel,Abhay Rajde,Sahil Shah,Prof..Manya Gidwani

[6] Driver Drowsiness Detection Published in 2020, V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N, Dr.Nagamani N P

[7] [7]Driver Drowsiness Detection Published in 2013, Belal ALSHAQAQ, Abdullah Salem BAQUHAIZEL; Mohamed El Amine OUIS, Meriem BOUMEHED; Abdelaziz OUAMRI; Mokhtar KECHE

[8] Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images Published in 2022, Elena Magán, M. Paz Sesmero , Juan Manuel Alonso-Weber and Araceli Sanchis

[9] Real-Time Driver Drowsiness Detection using Computer Vision Published in 2021, Mahek Jain, Bhavya Bhagerathi, Sowmyarani C N

[10] Prof.Ankita V. Karale, 2Nikita P. Patil, 3Prajakta M. Sarvar, 4Pritam M. Sangle, 5Shila A. Shind https://www.jetir.org/papers/JETIR2105036.pdf

[11] S. Jhansi Rani1 , Anand Rajasekaran 2 , Chandrasekhar A R 3 1Assistant Professor, Department of Information Technology, Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India

[12] Rizwan Ali Naqvi 1 , Muhammad Arsalan 2 , Abdul Rehman 3 , Ateeq Ur Rehman 4 , Woong-Kee Loh 5 and Anand Paul 3 https://www.mdpi.com/2072-4292/12/3/587/htm

[13] Md. Uzzal Hossain Md. Ataur Rahman Md. ManowarulIslamaArnishaAkhteraMd. Ashraf Uddin Bikash KumarPaul, https://www.sciencedirect.com/science/article/pii/S2667305322000163?via%3Dihub#! Compucom Institute of Technology and Management, India