****CONTENTS ****

TOPICS **PAGE NO**

ABSTRACT

Driver drowsiness is one of the reasons for large number of road accidents these days. With the advancement in Computer Vision technologies, smart/intelligent cameras are developed to identify drowsiness in drivers, thereby alerting drivers which in turn reduce accidents when they are in fatigue. A new framework is proposed using deep learning to detect driver drowsiness based on Eye state and Mouth state while driving the vehicle. In this proposed project, Media pipe Face Mesh is used to detect the face and extract the eye and mouth landmarks from the face images from camera sequences and calculate facial features such as eye closure rate, pupil circularity, mouth aspect ratio. This is fed to Long Short-Term Memory model which has been trained on temporal sequences of data to detect whether a driver is drowsy or not. This proposed system alerts driver with an alarm when the driver is in sleepy mood.

List of Figures

Output screens

1.INTRODUCTION

1.1 MOTIVATION

Drowsiness is position of on the brink of rest, a strong urge for rest. Drowsiness alludes to be unable to stay your eyes open, or feeling tired or tired. Sleepiness, additionally called abundance languor. It may prompt carelessness or nodding off at unseemly occasions. It tends to be joined by shortcoming, laziness, and absence of mental sharpness. Discouragement, distress and stress are additionally connected with traded off rest. Presently drowsiness of individual driving vehicle is critical. It's not going to be due to some clinical issue yet long.
Driver exhaustion once in a while brings about street mishaps consistently. It is difficult to gauge the specific measure of rest related mishaps however research presents that driver exhaustion might be a contributing explanation in up to 20% in street mishaps. These sorts of mishaps are about half increasingly expected to bring about death or genuine hurt. They happen mostly at higher speed impacts. What's more, the driver who has nodded off can't slow down. Drowsiness decreases reaction time which is a genuine component of secure driving. It likewise lessens readiness, cautiousness, and focus with the goal that the ability to perform consideration-based exercises for example driving is hindered. The speed at which data is handled is additionally diminished by drowsiness. The nature of dynamic may likewise be influenced. Unmistakably drivers know when they are feeling lethargic, thus settle on a cognizant choice about whether to keep driving or to stop for a rest. It might be that the individuals who continue driving belittle the danger of really nodding off while driving. Or on the other hand it might be that a few drivers decide to disregard the dangers in the manner drivers' drink. Accidents brought about by tired drivers are destined to occur on long excursions on repetitive streets, for example, motorways, somewhere in the range of 2pm and 4pm, particularly in the wake of eating or taking a mixed beverage, somewhere in the range of 2am and 6am, in the wake of having less rest than typical, in the wake of drinking liquor, its driver takes meds that cause drowsiness and after long working hours or on ventures home after long moves, particularly night shifts.

Driver drowsiness is one of the major causes Sluggishness and exhaustion can frequently influence an individual's driving capacity some time before he/she even notification that he/she is getting drained. Weariness related accidents are regularly more extreme than others since driver's response times are postponed or the drivers have neglected to make any man oeuvres to maintain a strategic distance from an accident. The quantity of hours spent driving has a solid relationship to the quantity of weakness related mishaps.

Driver drowsiness is a genuine risk in transportation frameworks. It has been recognized as an immediate or contributing reason for street mishap. Drowsiness can truly slow response time, decline mindfulness and weaken a driver's judgment. It is inferred that driving while drowsy is like driving affected by liquor or medications. In industrialized nations, drowsiness has been assessed to be associated with 2% to 23% everything being equal.

Frameworks that distinguish when drivers are turning out to be drowsy and sound an admonition guarantee to be a significant guide in forestalling mishaps. Conceivable procedure for distinguishing drowsiness in drivers can be commonly separated into the accompanying classifications: detecting of physiological qualities, detecting of driver activity, detecting of vehicle reaction, observing the reaction of driver. Driver weariness is a noteworthy factor in an enormous number of vehicle mishaps. Ongoing insights gauge that yearly 1,200 passings and 76,000 wounds can be credited to exhaustion related accidents. The advancement of advances for identifying or forestalling drowsiness in the driver's seat is a significant challenge in the field of mishap shirking frameworks. On account of the risk that drowsiness presents out and about, strategies should be created for balancing its effects.

1.2 PROBLEM DEFINITION

Drowsiness is one of the factors for road accidents. Motor vehicle collisions lead to significant death and disability as well as significant financial cost to both security and individual due to the driver impairments. A system is developed to identify driver’s drowsiness by monitoring the eye closure and the movement of the driver while driving. In order to detect drowsiness in a very effective way we have proposed a system for drowsiness detection during driving by calculating various facial features in temporal sequences of frames and using LSTM Model on raspberry pi.

1.3 OBJECTIVE OF THE PROJECT

The objective of the driver drowsiness detection is to detect the drowsiness of a driver. It is important that this step is performed accurately because many features used to assess the level of drowsiness in a person. Our approach to extracting the landmarks of the face from the frames of the video stream and develop suitable features for our classification model such as the four core features that are concluded were eye aspect ratio, mouth aspect ratio, pupil circularity, and finally, mouth aspect ratio over eye aspect ratio.Measuring the Driver Drowsiness requires various factors to measure based on vehicle-based measures, subjective measures, driver behavioural measures and physiological measures,The main contribution in the system is to use the driver behavioral measures in order to detect whether the driver is drowsy or not which is mainly based on eye state ,mouth state, pupil state, taking in consideration these various factors, a much more accurate system is developed instead of relying on only one factor. To classify whether the driver is drowsy or not. There are many computerized methods for driver drowsiness using convolutional neural networks (CNNs). But to classify temporal sequences of data we proposed another method, The main contribution of this study is proposing a approach for driver drowsiness classification based on Long short term memory(LSTM) model. The proposed deep learning model LSTM will be used to classify and evaluate on the csv dataset which has been extracted from the UTA-RLDD video dataset for achieving very good classification performance i.e., whether it is drowsy or alert state of the driver.

MODULE-1

Data Collection and processing module on the requirement basis.

MODULE-2

Applying Long Short-Term Memory (LSTM) Model to the dataset available.

MODULE-3

Detect the face of the driver through live video stream by using the Raspberry Pi 3b+ and pi Camera.

1.4 LIMITATIONS OF THE PROJECT

- Applicable only when the driver is at a safe distance the camera.
- High cost of the Hardware.
- Not so effective in low light conditions.

1.5 ORGANIZATION OF THE DOCUMENT

This documentation is divided into seven chapters namely

Chapter 1:

-It contains Introduction, Motivation about the project and the main objective of the project.

Chapter 2:

-It contains Proposed system and its applications.

Chapter 3:

-It contains analysis and software requirement specifications and content diagram.

Chapter 4:

-It contains designs of the project. In this, UML designs are used.

Chapter 5:

-It contains implementation of the projects and outputs.

Chapter 6:

-It contains everything about testing.

Chapter 7:

-It states the conclusion drawn from the observations we have made.

1.6 CONCLUSION

As mentioned, we will implement all the above-mentioned modules and will develop a Long Short Term Memory model.

2. LITERATURE SURVEY

2.1 INTRODUCTION

The drowsiness is an intermediate state between awake and sleep, in which the observation and analysis of a conductor is very small. The lack of concentration due to the driver fatigue is a major cause that leads to the high number of accidents. In this work, an effort has been put to detect the state of drowsiness using facial parameters obtained using facial points. Moreover, the parameters related to eye and mouth organs have also been extracted. Deep neural networks are out performing when compared to many state-of-the art algorithms. Hence, long short-term memory (LSTM) units are considered to estimate the drowsiness level of a driver. It is found that they are very appropriate in processing of sequential multimedia data. An accuracy of 92% is obtained with the proposed approach. LSTM is more accurate than the Convolutional Neural Networks (CNNs). Our approach to extracting the landmarks of the face from the frames of the video stream and develop suitable features for our classification model such as the four core features that are concluded were eye aspect ratio, mouth aspect ratio, pupil circularity, and finally, mouth aspect ratio over eye aspect ratio and the driver behavioral measures in order to detect whether the driver is drowsy or not which is mainly based on eye state, mouth state, pupil state, taking in consideration these various factors, a much more accurate system is developed instead of relying on only one factor. To classify whether the driver is drowsy or not, An increase in the number of blinks has been considered as a state of drowsiness. Yawning has been considered as another feature for driver's drowsiness detection. LSTM will be used to classify and evaluate on the csv dataset which has been extracted from the UTA-RLDD video dataset for achieving very good classification to alert the driver.

2.2 BACKGROUND

OpenCV: OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including A powerful N-dimensional array object, Sophisticated (broadcasting) functions, Useful linear algebra, Fourier transform, and random number capabilities.

Keras: Keras is a open-source software library that provides a python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

TensorFlow: TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use. The tensor flow data API enables you to build complex input pipelines from simple, reusable pieces.

Matplotlib: Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Sklearn: Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

MediaPipe: MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works in Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano.

Torch: Torch is an open-source machine learning library, a scientific computing framework, and a script language based on the Lua programming language. It provides a wide range of algorithms for deep learning, and uses the scripting language LuaJIT, and an underlying C implementation.

Mlxtend: Mlxtend (machine learning extensions) is a Python library of useful tools for the day-to-day data science tasks.

2.3 DATASETS

This dataset consists of 60 GB videos. Each video is around 10 minutes long, and is labelled as belonging to one of three classes: alert (labelled as 0), low vigilant (labelled as 5) and drowsy (labelled as 10). The labels were provided by the participants themselves, based on their predominant state while recording each video. This type of labelling takes into account and emphasizes the transition from alertness to drowsiness. Each set of videos was recorded by a personal cell phone or web camera resulting in various video resolutions and qualities. The 60 subjects were randomly divided into five folds of 12 participants, for the purpose of cross validation.

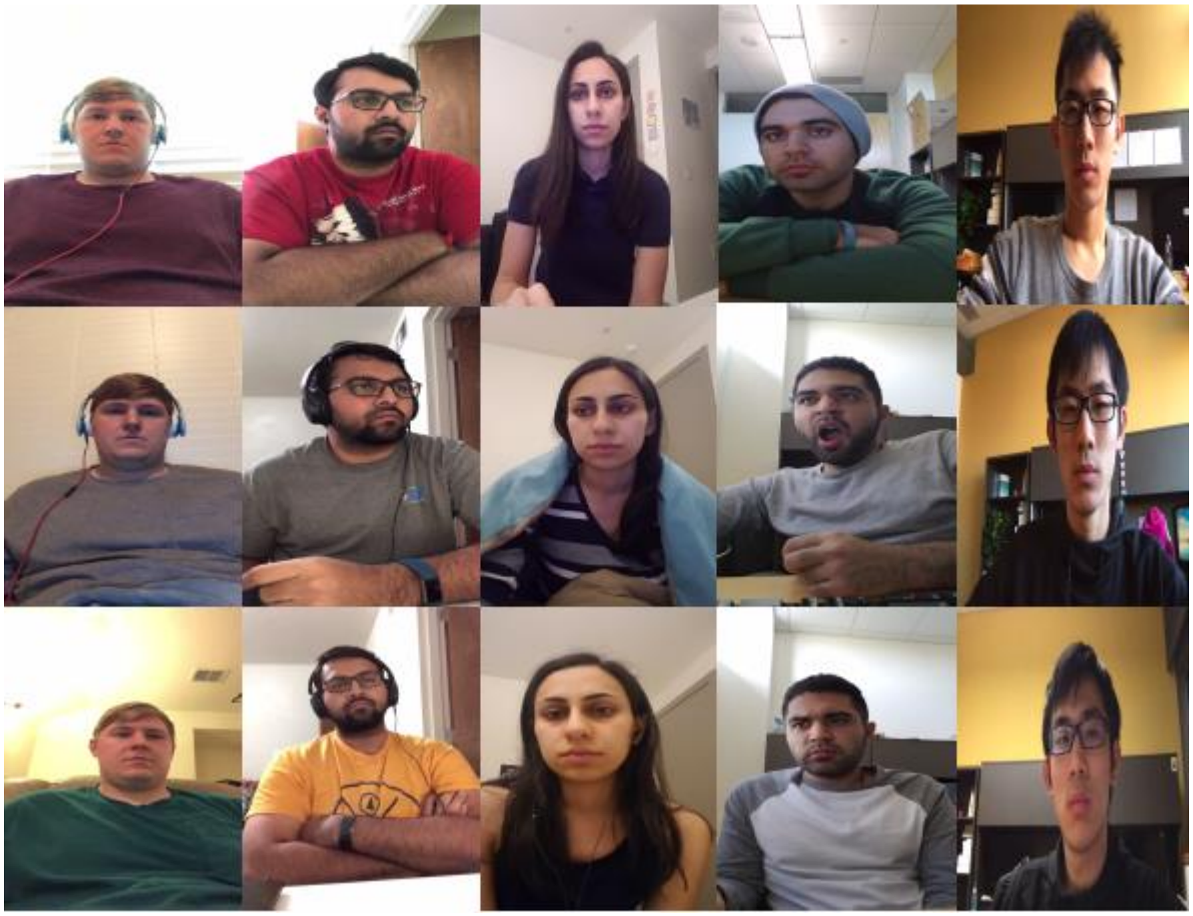


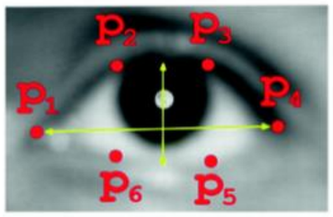
Figure 2.3.1:Drowsy Dataset(1st row:Alert, 2nd row:Neutral, 3rd row: Drowsy)

Data Preparation:

The Data Preparation phase includes all activities to create the final data set or data selection that will be loaded into the machine learning model for our classification. The Data Preparation phase includes all activities to create the final data set or data selection that will be loaded into the deep learning model for our classification. Based on the facial landmarks that we extracted from the frames of the videos, we ventured into developing suitable features for our classification model. While we hypothesized and tested several features, the four core features that we concluded on for our final models were eye aspect ratio, mouth aspect ratio, pupil circularity, and finally, mouth aspect ratio over eye aspect ratio.

Eye Aspect Ratio (EAR):

EAR, as the name suggests, is the ratio of the length of the eyes to the width of the eyes. The length of the eyes is calculated by averaging over two distinct vertical lines across the eyes as illustrated in the figure below.



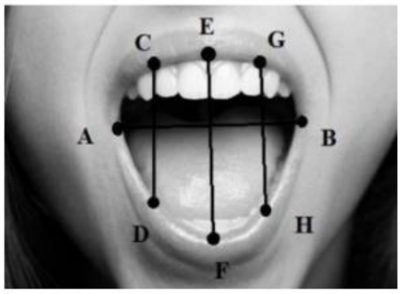
$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figure 2.3.2: Eye Aspect Ratio Formula

Our hypothesis was that when an individual is drowsy, their eyes are likely to get smaller and they are likely to blink more. Based on this hypothesis, we expected our model to predict the class as drowsy if the eye aspect ratio for an individual over successive frames started to decline i.e. their eyes started to be more closed or they were blinking faster.

Mouth Aspect Ratio (MAR):

Computationally similar to the EAR, the MAR, as you would expect, measures the ratio of the length of the mouth to the width of the mouth. Our hypothesis was that as an individual becomes drowsy, they are likely to yawn and lose control over their mouth, making their MAR to be higher than usual in this state.



$$\text{MAR} = \frac{|EF|}{|AB|}$$

Figure 2.3.3: Mouth Aspect Ratio Formula

Pupil Circularity (PUC):

PUC is a measure complementary to EAR, but it places a greater emphasis on the pupil instead of the entire eye.

$$Circularity = \frac{4 * \pi * Area}{perimeter^2} \qquad Area = \left(\frac{\text{Distance(p2, p5)}}{2}\right)^2 * \pi$$

$$\begin{aligned} Perimeter = {} & Distance(p1, p2) + Distance(p2, p3) + Distance(p3, p4) + \\ & Distance(p4, p5) + Distance(p5, p6) + Distance(p6, p1) \end{aligned}$$

Figure 2.3.4: Pupil Circularity

Mouth over Eye Ratio (MOE):

The benefit of using this feature is that EAR and MAR are expected to move in opposite directions if the state of the individual changes. As opposed to both EAR and MAR, MOE as a measure will be more responsive to these changes as it will capture the subtle changes in both EAR and MAR and will exaggerate the changes as the denominator and numerator move in opposite directions. Because the MOE takes MAR as the numerator and EAR as the denominator, our theory was that as the individual gets drowsy, the MOE will increase.

$$MOE = \frac{MAR}{EAR}$$

Mouth Over Eye Ratio (MOE)

Figure 2.3.5: Mouth Aspect Ratio over Eye Aspect Ratio

While all these features made intuitive sense, when tested with our classification models, they yielded poor results in the range of 55% to 60% accuracy which is only a minor improvement over the baseline accuracy of 50% for a binary balanced classification problem. Nonetheless, this disappointment led us to our most important discovery: the features weren’t wrong, we just weren’t looking at them correctly.

Feature Normalization:

When we were testing our models with the four core features discussed above, we witnessed an alarming pattern. Whenever we randomly split the frames in our training and test, our model would yield results with accuracy as high 70%, however, whenever we split the frames by individuals (i.e. an individual that is in the test set will not be in the training set), our model performance would be poor as alluded to earlier.

This led us to the realization that our model was struggling with new faces and the primary reason for this struggle was the fact that each individual has different core features in their default alert state. That is, person A may naturally have much smaller eyes than person B. If a model is trained on person B, the model, when tested on person A, will always predict the state as drowsy because it will detect a fall in EAR and PUC and a rise in MOE even though person A was alert. Based on this discovery, we hypothesized that normalizing the features for each individual is likely to yield better results and as it turned out, we were correct.

To normalize the features of each individual, we took the first three frames for each individual's alert video and used them as the baseline for normalization. The mean and standard deviation of each feature for these three frames were calculated and used to normalize each feature individually for each participant. Mathematically, this is what the normalization equation looked like:

$$z = \frac{x_i - \mu}{\sigma}$$

Now that we had normalized each of the four core features, our feature set had eight features, each core feature complemented by its normalized version. We tested all eight features in our models and our results improved significantly.

2.4 EXISTING SYSTEM

2.4.1 Convolution Neural Network (CNN):

Convolutional Neural Networks (CNN) are typically used to analyse image data and map images to output variables. However, we build a 1-D CNN and send in numerical features as sequential input data to try and understand the spatial relationship between each feature for the two states. CNN model has 5 layers including 1 convolutional layer, 1 flatten later, 2 fully connected dense layers, and 1 dropout layer before the output layer. The flatten layer flattens the output from the convolutional layer and makes it linear before passing it into the first dense layer. The dropout layer randomly drops 20% of the output nodes from the second dense layer in order to prevent model from overfitting to the training data. The final dense layer has a single output node that outputs 0 for alert and 1 for drowsy. An average of 83% accuracy was achieved by light weight models.

Disadvantages of CNN

- A Convolutional neural network is significantly slower due to an operation such as max pool.
- If the CNN has several layers then the training process takes a lot of time if the computer doesn't consist of a good GPU.
- A ConvNet requires a large Dataset to process and train the neural network.
- A time series represents a temporal sequence of data - and generally for sequential data LSTM is the preferred DNN algorithm as it handles sequences much better. CNN generally becomes useful when you want to capture neighbourhood information like in an image.

2.4.2 HAAR Cascade Algorithm:

Viola-Jones Face Detection Technique, popularly known as Haar Cascades, it is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The first contribution to the research was the introduction of the Haar features. These features on the image makes it easy to find out the edges or the lines in the image, or to pick areas where there is a sudden change in the intensities of the pixels. Consider Haar value from a rectangular image section. The darker areas in the Haar feature are pixels with values 1, and the lighter areas are pixels with values 0. Each of these is responsible for finding out one particular feature in the image. In the Viola — Jones research, they had a total of 38 stages for something around 6000 features. The number of features in the first five stages are 1, 10, 25, 25, and 50, and this increased in the subsequent stages. The initial stages with simpler and lesser number of features removed most of the windows not having any facial features, thereby reducing the false negative ratio, whereas the later stages with complex and more number of features can focus on reducing the error detection rate, hence achieving low false positive ratio.

Disadvantages of HAAR Cascade Algorithm:

- The downside to Haar cascades is that they tend to be prone to false-positive detections, require parameter tuning when being applied for inference/detection, and just, in general, are not as accurate as the more “modern” algorithms.

2.4.3 Recurrent Neural Network(RNN) :

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

RNN have a "memory" which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

Disadvantages of Recurrent Neural Network :

- Gradient vanishing and exploding problems.
- Training an RNN is a very difficult task.
- It cannot process very long sequences if using tanh or relu as an activation function.

2.4.4 k-Nearest Neighbor (k-NN) algorithm

The k-Nearest Neighbor (k-NN) algorithm is used to classify the driver's state of drowsiness based on the eye closure and head movement characteristic. In Bamidele et al. presented a nonintrusive DDD system, based on face and eye state tracking. The research utilized the NTHUDDD Computer Vision Lab's video Sensors 2022, 22, 2069 8 of 41 dataset. The proposed system starts by acquiring and pre-processing the required data. Then, it extracts the targeted features, including the PERCLOS, maximum closure duration of the eyes, and blink frequency. The extracted features are then fed to various classifiers to decide whether they belong to a drowsy or awake person. These classifiers include KNN, SVM, logistic regression, and artificial neural networks (ANN). The final results revealed that the best models were the KNN and ANN, with accuracies of 72.25% and 71.61%, respectively.

Disadvantages of KNN algorithm

- Doesn't work well with a large dataset.
- Classification time is long.

2.5 Proposed System

2.5.1 Long Short-Term Memory (LSTM) Network

LSTM is a novel recurrent network architecture in conjunction with an appropriate gradient-based learning algorithm introduced by Hochreiter and Schmidhuber in 1997. LSTM networks are a special kind of Recurrent Neural Networks (RNN), capable of learning long-term dependencies in the data. Recurrent Neural Networks are feedback neural networks that have internal memory that allows information to persist. LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time. Traditional neural networks have the limitation of not being able to add or modify the existing information. This limitation is addressed in RNN. Moreover, LSTMs are explicitly designed to avoid the long-term dependency problem. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module has a very simple structure, such as a single Hyperbolic tangent (tanh) layer. Hyperbolic tangent (tanh) is used to determine candidate values to get added to the internal state. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four layers that interact in an efficient having a single neural network layer, there are four layers that interact in an efficient manner to improve the performance of conventional RNN. manner to improve the performance of conventional RNN. **In LSTM, the inputs are related to each other.** The formula is as below:

$$h_t = f(h_{t-1}, x_t)$$

Figure 2.5.1.1: LSTM Hidden state formula

We chose to use an LSTM network because it allows us to study long sequences without having to worry about the gradient vanishing problems faced by traditional RNNs. Within the LSTM network, there are three gates for each time step: Forget Gate, Input gate, and Output Gate.

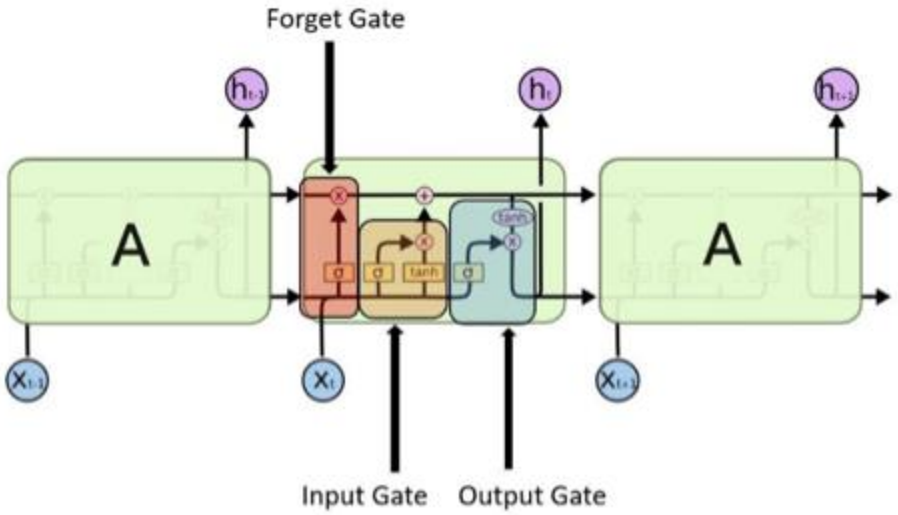


Figure 2.5.1.2: LSTM Network Visualized

Forget Gate: as its name suggests, the gate tries to “forget” part of the memory from the previous output.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figure 2.5.1.3: Forget Gate Formula

Input Gate: the gate decides what should be kept from the input in order to modify the memory.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figure 2.5.1.4: Input Gate Formula

Output Gate: the gate decides what the output is by combining the input and memory.

$$o_t = \sigma\left(W_o \ [\, h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.5.1.5: Output Gate Formula

First, the videos are converted into batches of data. Then, each batch was sent through a fully connected layer with 1024 hidden units using the sigmoid activation function. The next layer is our LSTM layer with 512 hidden units followed by 3 more FC layers until the final output layer as displayed below.

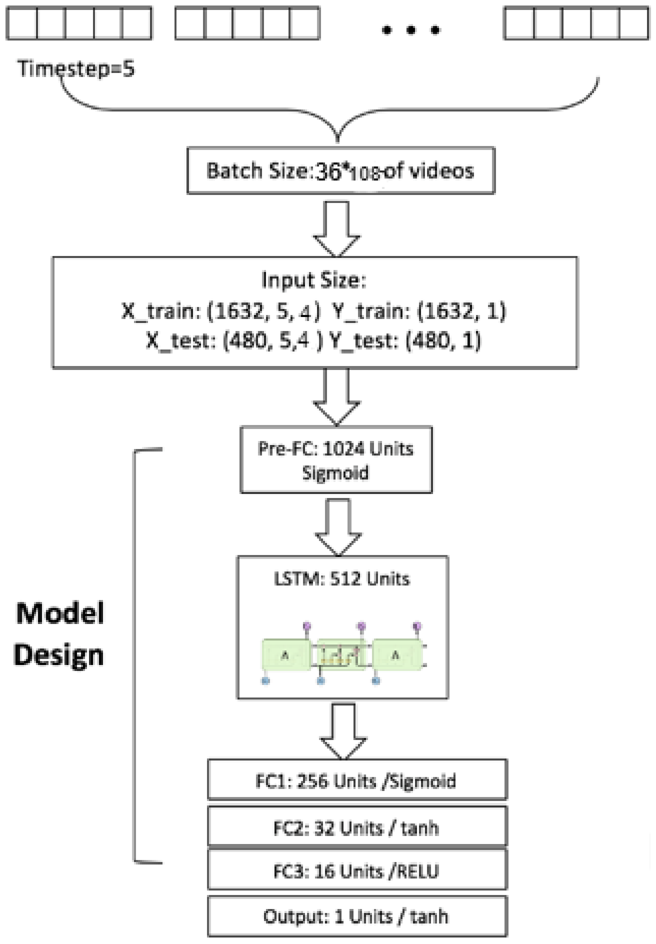


Figure 2.5.1.6: LSTM Network Design

Number of Epochs	50
Learning Rate	0.00005
Timestep	5

Figure 2.5.1.7: LSTM Parameters

After hyperparameter tuning, the optimized LSTM model achieved an overall accuracy of 92% .

2.5.2 MediaPipe FaceMesh:

MediaPipe Face Mesh is a solution that estimates 468 3D face landmarks in real-time even on mobile devices. It employs machine learning (ML) to infer the 3D facial surface, requiring only a single camera input without the need for a dedicated depth sensor. Utilizing lightweight model architectures together with GPU acceleration throughout the pipeline, the solution delivers real-time performance critical for live experiences. Additionally, the solution is bundled with the Face Transform module that bridges the gap between the face landmark estimation and useful real-time augmented reality (AR) applications. It establishes a metric 3D space and uses the face landmark screen positions to estimate a face transform within that space. The face transform data consists of common 3D primitives, including a face pose transformation matrix and a triangular face mesh. Under the hood, a lightweight statistical analysis method called Procrustes Analysis is employed to drive a robust, performant and portable logic. The analysis runs on CPU and has a minimal speed/memory footprint on top of the ML model inference.

Models:

FACE DETECTION MODEL

The face detector is the same Blaze Face model used in MediaPipe Face Detection. Please refer to MediaPipe Face Detection for details.

FACE LANDMARK MODEL

For 3D face landmarks we employed transfer learning and trained a network with several objectives: the network simultaneously predicts 3D landmark coordinates on synthetic rendered data and 2D semantic contours on annotated real-world data. The resulting network provided us with reasonable 3D landmark predictions not just on synthetic but also on real-world data.

ATTENTION MESH MODEL

In addition to the Face Landmark Model we provide another model that applies attention to semantically meaningful face regions, and therefore predicting landmarks more accurately around lips, eyes and irises, at the expense of more compute. It enables applications like AR makeup and AR puppeteering.

2.6 CONCLUSION

This chapter gives a detailed explanation of the project both existing and proposed along with the modules used in the system.

3. ANALYSIS

3.1 INTRODUCTION

Requirement Analysis is the first phase in the software development process. The main objective of the phase is to identify the problem and the system to be developed. The later phases are strictly dependent on this phase and hence the requirements for the system analyst to be clearer, precise about this phase. Any inconsistency in this phase will lead a lot of problems in other phases to be followed. Hence there will be several reviews before the final copy of the analysis is completed the system analyst will submit the details of the system to be developed in the form of a document called requirement specification.

The requirement analysis task is a process of discovery, refinement, modelling and specifications.The software scope, initially established by a system engineer and refined during software project planning,is refined in detail. Models of required data, information and control flow and operational behaviour are created. Alternative solutions are analysed and allocated to various software elements.

Requirement analysis provides the software designer with models that can be translated into data, architectural, interface and procedural design. Finally, the requirement specification provides the developer and the customer with the means to access quality once software builds.

Software requirements analysis may be divided into five are of effort

- Problem recognition
- Evaluation
- Modeling
- Review

Once the above tasks are accomplished, the specification document could be developed which now forms the basis for their main software engineering tasks. Keeping this in view the specification document of the current system has been generated.

The purpose of this document is to present a detailed description of DRIVER’S DROWSINESS DETECTION. It will explain the features of the system, that the system will do and the response of it for various inputs.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION

A Software Requirements Specification (SRS) is a complete description of the system to be developed. It includes a set of usecases that describes all the interactions the users will have with the software. Usecases are also known as functional requirements. In addition to the usecases, the SRS also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation requirements.

The Software Requirements Specification document lists all necessary requirements that the required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detail communications with the project team and the customer. It consists of

1. Introduction
2. Overall Description
3. Productive perspective
4. Specific requirements
5. Other requirements

The characteristics of a good Software Requirement Specifications are

1. Complete
2. Unambiguous
3. Verifiable

3.2.1 User Requirements

- The user requires a raspberry pi toolkit
- The user needs to connect the device to a power supply
- The driver should look ahead and drive

3.2.2 Software Requirements

Software Requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough 9 understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

The software requirements for this project are:

- Python 3.10
- Opencv
- Mediapipe
- Numpy
- Pygame
- Tensorflow
- Keras

OpenCV

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on.

The images are recognized as human eyes provide lots of information based on what they see. Machines are facilitated with seeing everything, convert the vision into numbers and store in the memory. Here the question arises how computer convert images into numbers. So the answer is that the pixel value is used to convert images into numbers. A pixel is the smallest unit of a digital image or graphics that can be displayed and represented on a digital display device. The picture intensity at the particular location is represented by the numbers. In the above image, we have shown the pixel values for a grayscale image consist of only one value, the intensity of the black color at that location.

There are two common ways to identify the images:

- Grayscale
- RGB

Mediapipe

MediaPipe Face Detection is an ultrafast face detection solution that comes with 6 landmarks and multi-face support. It is based on BlazeFace, a lightweight and well-performing face detector tailored for mobile GPU inference. The detector's super-realtime performance enables it to be applied to any live viewfinder experience that requires an accurate facial region of interest as an input for other task-specific models, such as 3D facial keypoint estimation (e.g., MediaPipe Face Mesh), facial features or expression classification, and face region segmentation. BlazeFace uses a lightweight feature extraction network inspired by, but distinct from MobileNetV1/V2, a GPU-friendly anchor scheme modified from Single Shot MultiBox Detector (SSD), and an improved tie resolution strategy alternative to non-maximum suppression. For more information about BlazeFace, please see the Resources section.

MediaPipe Face Mesh is a solution that estimates 468 3D face landmarks in real-time even on mobile devices. It employs machine learning (ML) to infer the 3D facial surface, requiring only a single camera input without the need for a dedicated depth sensor. Utilizing lightweight model architectures together with GPU acceleration throughout the pipeline, the solution delivers real-time performance critical for live experiences.

Additionally, the solution is bundled with the Face Transform module that bridges the gap between the face landmark estimation and useful real-time augmented reality (AR) applications. It establishes a metric 3D space and uses the face landmark screen positions to estimate a face transform within that space. The face transform data consists of common 3D primitives, including a face pose transformation matrix and a triangular face mesh. Under the hood, a lightweight statistical analysis method called Procrustes Analysis is employed to drive a robust, performant and portable logic. The analysis runs on CPU and has a minimal speed/memory footprint on top of the ML model inference.

Numpy

NumPy is the fundamental package for scientific computing in Python which provides a multidimensional array object other mathematical operations can be performed using this but simply speaking we just need it to convert our images into some form of an array so that we can store the model that has been trained.
To install the library you can type a simple line of code in your command shell:
pip install numpy.

Pygame

The pygame library is an open-source module for the Python programming language specifically intended to help you make games and other multimedia applications. Built on top of the highly portable SDL (Simple DirectMedia Layer) development library, pygame can run across many platforms and operating systems.

By using the pygame module, you can control the logic and graphics of your games without worrying about the backend complexities required for working with video and audio.

Tensorflow

TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use. The tf data API enables you to build complex input pipelines from simple, reusable pieces.

Keras

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

3.2.3 Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. This section will explain in detail about all the main hardware components used in the project.

The hardware components used are:

- Raspberry pi 3b+
- Buzzer 1.2v
- Raspberry pi Camera (ver. 1.3)

Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries.

It features a Broadcom system on a chip (SoC) with an integrated ARM- compatible central processing unit and on chip graphics processing unit. Raspberry pi is a complete environment on hardware and software for IOT prototyping. The following are it's features:

- 512 MB SD RAM Memory
- Broadcom BCM2835 SoC full high definition multi coprocessor
- Dual Core Video Core IV Multimedia coprocessor
- Single 2.0 USB connector
- HDMI Video Out
- 3.5 MM Jack, HDMI Audio Out
- MMC, SD, SDIO Card slot on board storage
- Linux Operating system
- Dimensions are 8.6cm*5.4cm*1.7cm
- On board 10/100 Ethernet RJ45 jack

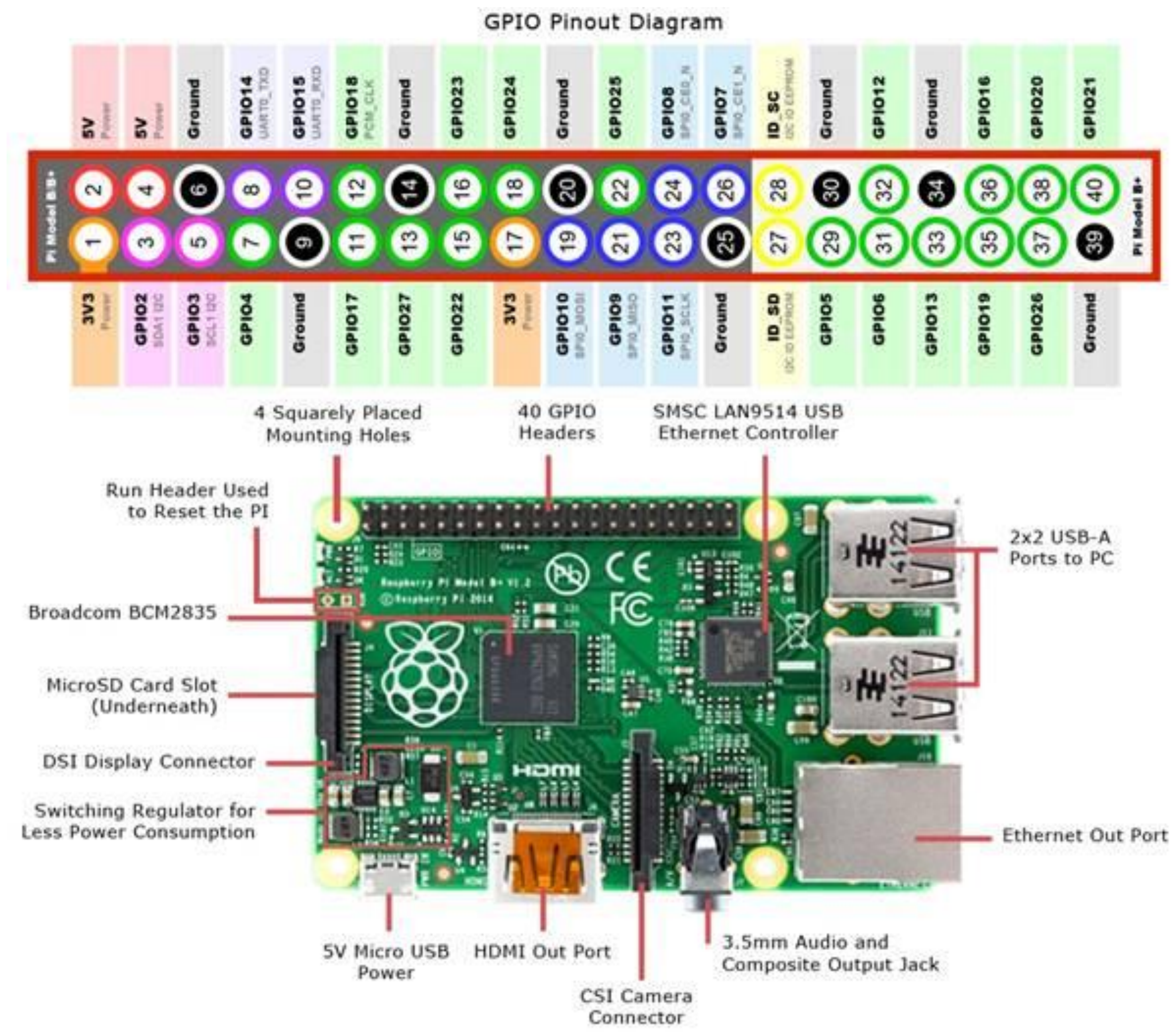


Figure 3.2.3.1: Raspberry pi

Memory

The raspberry pi model Aboard is designed with 256MB of SDRAM and model B is designed with 51MB.Raspberry pi is a small size PC compare with other PCs. The normal PCs RAM memory is available in gigabytes. But in raspberry pi board, the RAM memory is available more than 256MB or 512MB.

CPU (Central Processing Unit)

The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations. The raspberry pi uses ARM11 series processor.

GPU (Graphics Processing Unit)

The GPU is a specialized chip in the raspberry pi board and that is designed to speed up the operation of image calculations. This board designed with a Broadcom video core IV and it supports OpenGL

Ethernet Port

The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The raspberry pi Ethernet port is used to plug your home router to access the internet.

GPIO Pins

The general purpose input & output pins are used in the raspberry pi to associate with the other electronic boards. These pins can accept input & output commands based on programming raspberry pi. The raspberry pi affords digital GPIO pins. These pins are used to connect other electronic components. For example, you can connect it to the temperature sensor to transmit digital data.

XBee Socket

The XBee socket is used in raspberry pi board for the wireless communication purpose.

Power Source Connector

The power source cable is a small switch, which is placed on side of the shield. The main purpose of the power source connector is to enable an external power source.

UART

The Universal Asynchronous Receiver/ Transmitter is a serial input & output port. That can be used to transfer the serial data in the form of text and it is useful for converting the debugging code.

Raspberry Pi

The Raspberry Pi Camera Board v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60, and 640x480p90 video.

The camera used in the project has a sensor with a native resolution of 5 megapixel, and has a fixed focus lens onboard. In terms of still images, the camera is capable of 2592 x 1944 pixel static images, and also supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 video recording.

Connecting the camera

The flex cable inserts into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port. The flex cable connector should be opened by pulling the tabs on the top of the connector upwards then towards the Ethernet port. The flex cable should be inserted firmly into the connector, with care taken not to bend the flex at too acute an angle. The top part of the connector should then be pushed towards the HDMI connector and down, while the flex cable is held in place.

Enabling the Camera

Open the raspi-config tool from the terminal:

```
$sudo raspi-config
```

Select Enable camera and hit Enter ,then got to Finish and you'll be prompted to reboot.

Figure 3.2.3.2:Raspberry Pi Camera

Buzzer

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

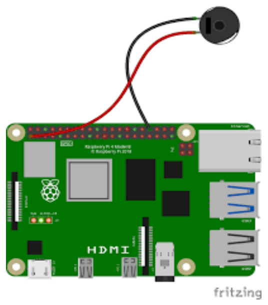


Figure 3.2.3.3:Buzzer

3.3 ALGORITHMS AND FLOWCHARTS

CONTROL FLOW OF THE PROJECT

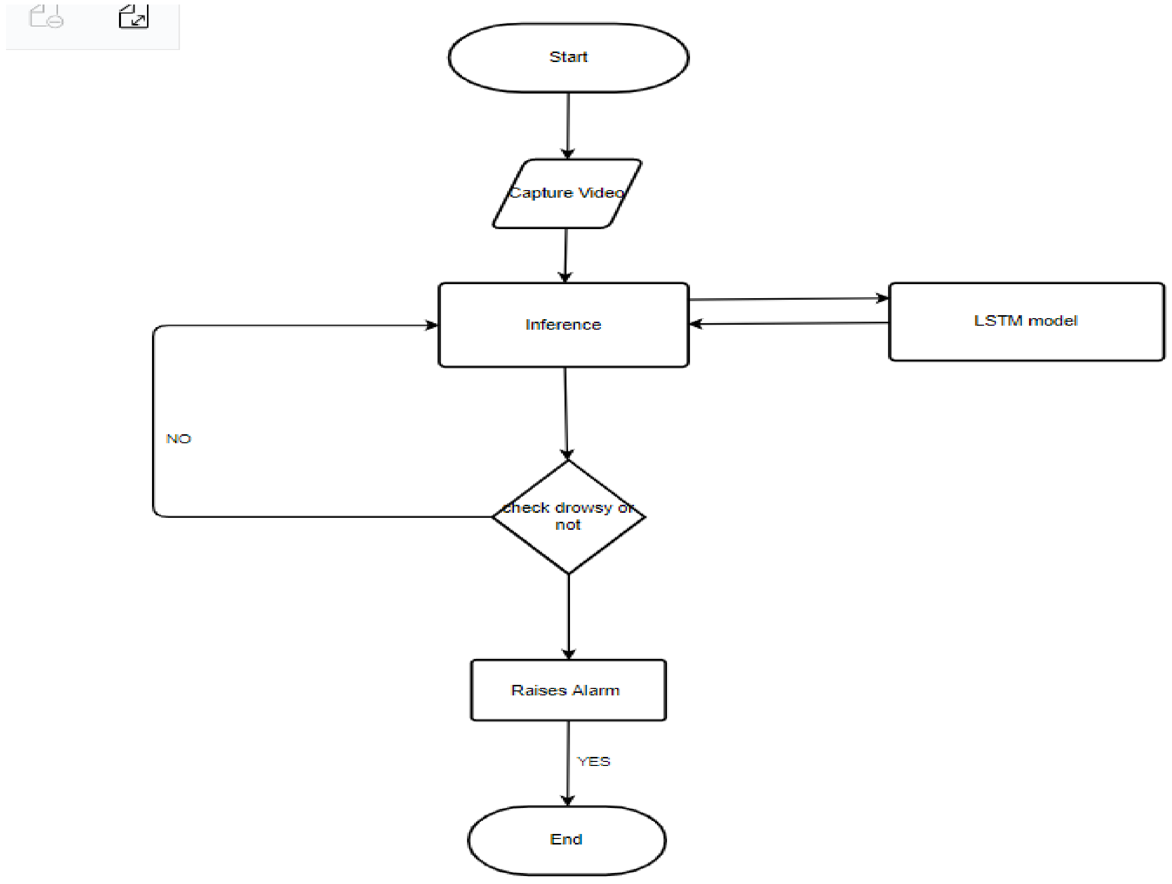


Figure 3.3: Flowchart Diagram

3.4 CONCLUSION

This chapter starts with the introduction of analysis in general followed by software requirement specification which contains user requirements i.e., what a user is expecting from the system. Software requirements i.e., what are the software's required to implement and run the system. Hardware requirements required for establishing and running the project. Content diagram explaining the basic contents of project and their functionality briefly. Algorithms and flow charts with specifies different algorithms used in development of project, their implementation details and flowcharts explaining their working.

4.DESIGN

4.1 INTRODUCTION

It is a process of planning the new or modified system. Analysis specifies what a new or modified system does. Design specifies how to accomplish the same. Design is essentially a bridge between requirement specification and the final solution satisfying the requirements. Design of a system is essentially a blue print or a solution for the system. The design process for a software system has two levels. At first level the focus is on depending in which modules are needed for the system, the specification of these modules and how the modules should be interconnected. This is what is called system designing of top level design. In the second level, the internal design of the modules, or how the specification of the module can be satisfied is described upon. This design level is often called detailed design or logic design. The first level produces system design, which defines the components needed for the system, and how the components interact with each other. It focus is on depending on in which that modules are needed for the system, the specification of these modules and how the module should be interconnected.

4.2 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML is different from the other common programming languages like C++, Java, and COBOL etc. It is designed to enable users to develop an expressive and ready to use visual modelling language. In addition, it supports high level development concepts such as frameworks, patterns and collaborations. UML can be described as a general-purpose visual modelling language to visualize, specify, construct and document software system. But it is not limited within this boundary. To be more precise, UML is a pictorial language used to make software blue prints UML has a direct relation with object-oriented analysis and design. After some standardization UML is become an OMG (Object Management Group) standard.

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

Goals of UML:

1. Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.

2. Provide extensibility and specialization mechanisms to extend the core concepts.

3. Be independent of particular programming languages and development processes.

4. Provide a formal basis for understanding the modelling language.

5. Encourage the growth of the OO tools market.

6. Support higher-level development concepts such as collaborations, frameworks and components.

7. The system can be a software or non-software. So, it must be clear that UML is not just a development method.

Basic Building Blocks of UML:

The basic building blocks in UML are **things** and **relationships**. These are combined in different ways following different rules to create different types of diagrams. In UML there are Eight types of diagrams, below is a list and brief description of them. The more in-depth descriptions in the document, will focus on the first five diagrams in the list, which can be seen as the most general, sometimes also referred to as the UML core diagrams.

Components of the UML:

The UML consists of a number of graphical elements that combine to form diagrams.

Because it's a language, the UML has rules for combining these elements. The purpose of the diagrams to present multiple views of the system, and this set of multiple views is called a Model. A UML Model of a system is something like a scale model of a building. UML model describes what a system is supposed to do. It doesn't tell how to implement the system.

These are the artefacts of a software-intensive system. The abbreviation for UML is Unified

Modelling Language and is being brought of a designed to make sure that the existing ER Diagrams which do not serve the purpose will be replaced by this UML Diagrams where in these language as its own set of Diagrams.

Some of the Diagrams that help for the Diagrammatic Approach for the Object-Oriented Software Engineering are:

- Class Diagrams
- Use Case Diagrams
- Sequence Diagrams
- State chart Diagrams
- Activity Diagrams

Using the above mentioned diagrams we can show the entire system regarding the working of the system or the flow of control and sequence of flow the state of the system and the activities involved in the system.

CLASS DIAGRAM

A Class is a category or group of things that has similar attributes and common behaviour. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle area contains the attributes and thelowest areas show the operations. Class diagrams provides the representation that developers work from.

The classes in a Class diagram represent both the main objects, interactions in the application and the classes to be programmed.
In the diagram, classes are represented with boxes which contain three parts:

1. The top part contains the name of the class. It is printed in bold and cantered, and the first letter is capitalized.

2. The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.

3. The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lower case. In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those described objects. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

Visibility:

To specify the visibility of a class member (i.e., any attribute or method), the sanitation’s must be placed before the member's name:

+ Public

/ Derived

# Protected

- Private

~ Package

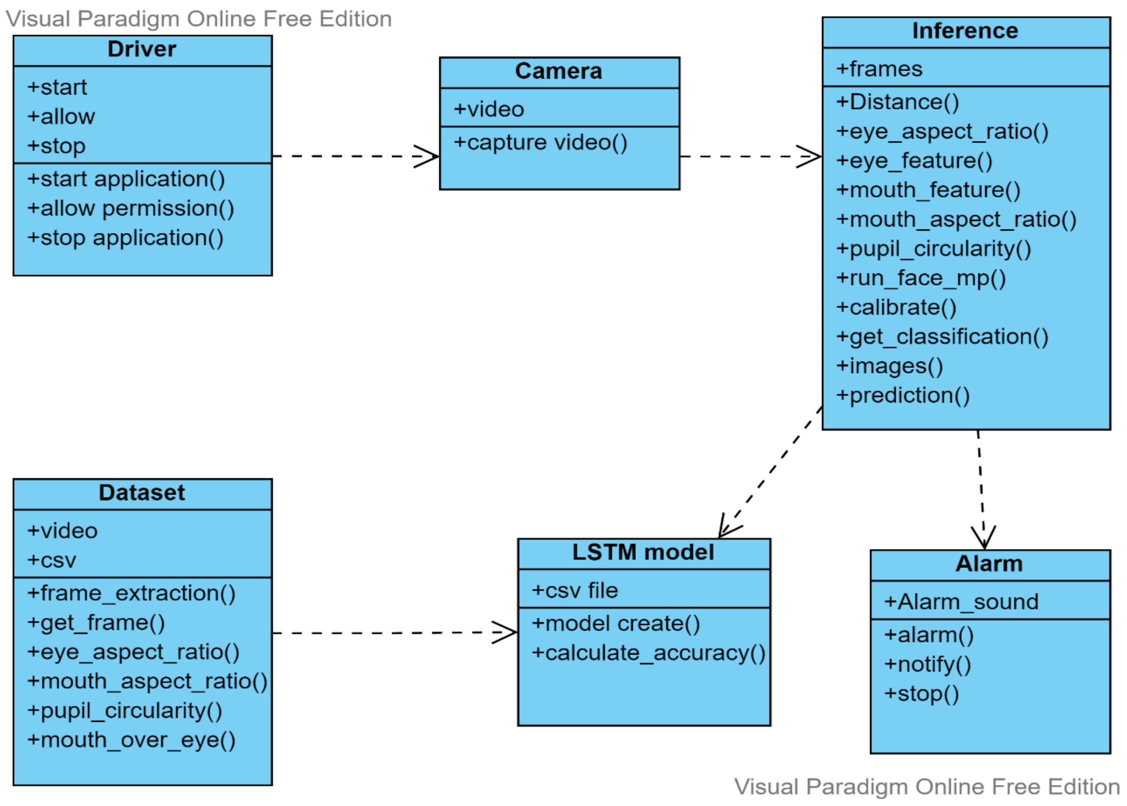


figure 4.2.1: Class Diagram

Relationships:

A relationship is a general term covering the specific types of logical connections found on class and object diagrams. UML shows the following relationships:

1.Links: A Link is the basic relationship among objects.

2.Association: An association represents a family of links. An association can be named and the ends of an association can be adorned with role names, ownership indicators, Multiplicity, visibility, and other properties.

3.Aggregation: Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association can. In UML, it is graphically represented as a hollow diamond shape on the containing class with a single line that connects it to the contained class. The aggregate is semantically an extended object that is treated as a unit in many operations, although physically it is made of several lesser objects.

4.Composition: Composition is a stronger variant of the "has a" association relationship; composition is more specific than aggregation. Composition usually has a strong lifecycle dependency between instances of the container class and instances of the contained class(es). The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained class(es) to the containing class.

5.Generalization: The Generalization relationship ("is a") indicates that one of the two related classes (the subclass) is considered to be a specialized form of the other (the super type) and the super class is considered a 'Generalization' of the subclass. The UML graphical representation of a Generalization is a hollow triangle shape on the super class end of the line (or tree of lines) that connects it to one or more subtypes. The generalization relationship is also known as the inheritance or "is *a"* relationship. Generalization can only be shown on class diagrams and on Use case diagrams.

6.Realization: In UML modelling, a realization relationship is a relationship between those two model elements, in which one model element realizes (implements or executes) the behaviour that the other model element specifies. The UML graphical representation of a Realization is a hollow triangle shape on the interface end of the *dashed* line (or tree of lines) that connects it to one or more implementers. A plain arrow head is used on the interface end of the dashed line that connects it to its users. Realizations can only be shown on class or component.

7.Dependency: Dependency is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class.

8.Multiplicity: This association relationship indicates that (at least) one of the two related classes makes reference to the other. The UML representation of an association is a line with an optional arrow head indicating the *role* of the object(s) in the relationship and an optional notation at each end indicating the *multiplicity* of instances of that entity (the number of objects that participate in the association).

USE-CASE DIAGRAM

A Use-Case is a description of systems behaviour from a understand point. For system developer this is a valuable tool: it's a tried-and-true technique for gathering system requirements from a user's point of view. That is important if the goal is to build a system that real people can use. A little stick figure is used to identify an actor the ellipse represents use-case.
Here, we have two actors namely developer and user. The developer is involved in all the steps whereas the user is only interested in visualizing the result, testing the model and displaying the result.

Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse

Actors: An actor is a person, Organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

Associations: Associations between actors and use case are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction
described by a use case. Associations are modeled as line connecting use
cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

Use Case Relationships:

Three relationships among use cases are used often in practice.

1.Include

In one form of interaction, a given use case may include another. Include is a directed relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case.

2. Extend

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behaviour of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label ‖<<extend>>‖.

3.Generalization

In the third form of relationship among use cases, generalization/specialization relationship exists. A given use case may have common behaviours, constraints and assumptions to the general use case. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case.

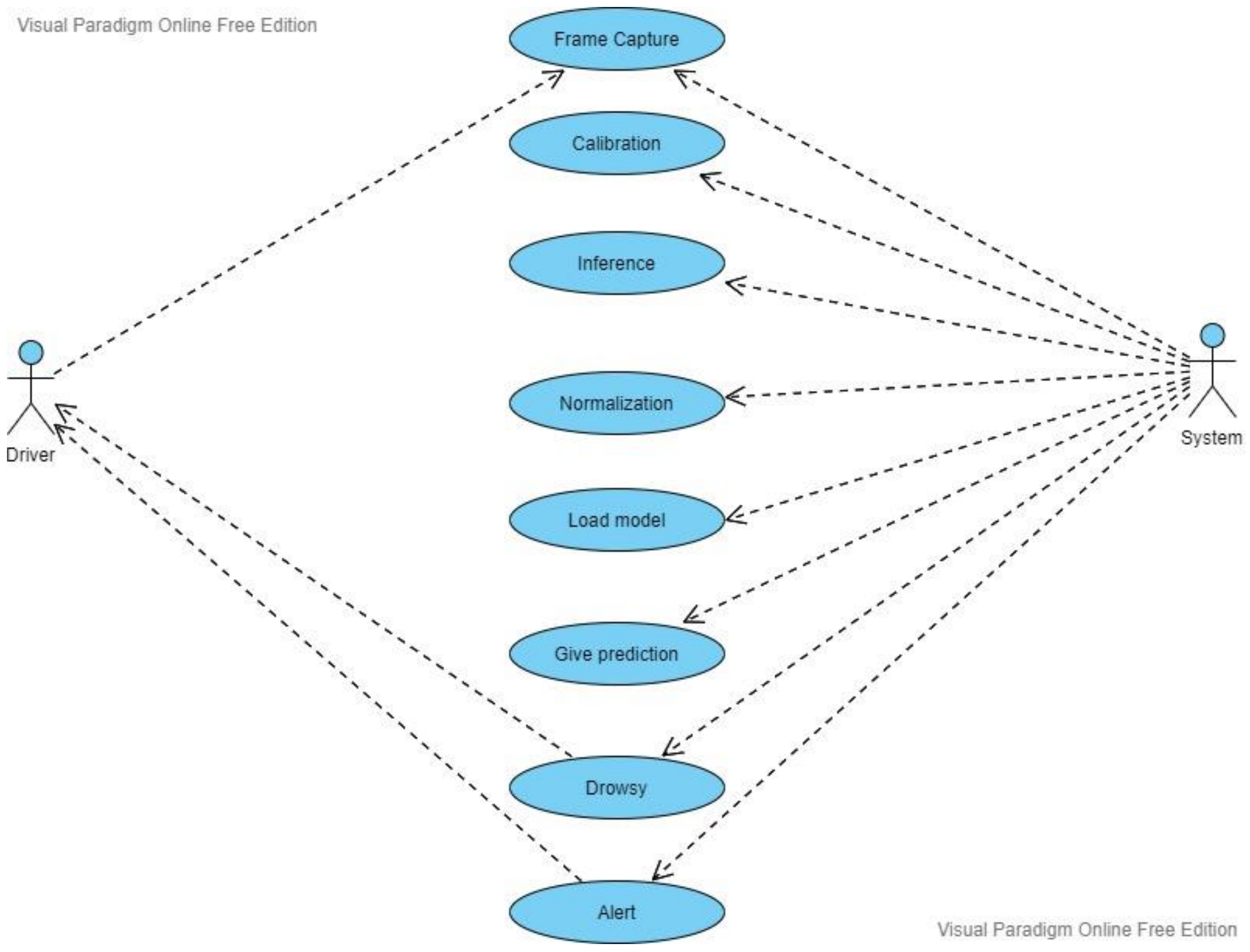


figure 4.2.2: Use Case Diagram

SEQUENCE DIAGRAMS

In a functioning system object interacts with one another and these interactions occur over time. The UML Sequence Diagrams shows the time-based dynamics of the interaction. The sequence diagrams consist of objects represented in the use always named rectangles (If the name underlined), messages represented as solid line arrows and time represented as a vertical progression.

Sequence diagrams describe interactions among classes in terms of an exchange of messages overtime. It is an interaction diagram that emphasizes the time ordering of messages. A narrow rectangle on an objects lifetime represents activation- an exchange of one of that objects operation. Messages are arrows that connect one.

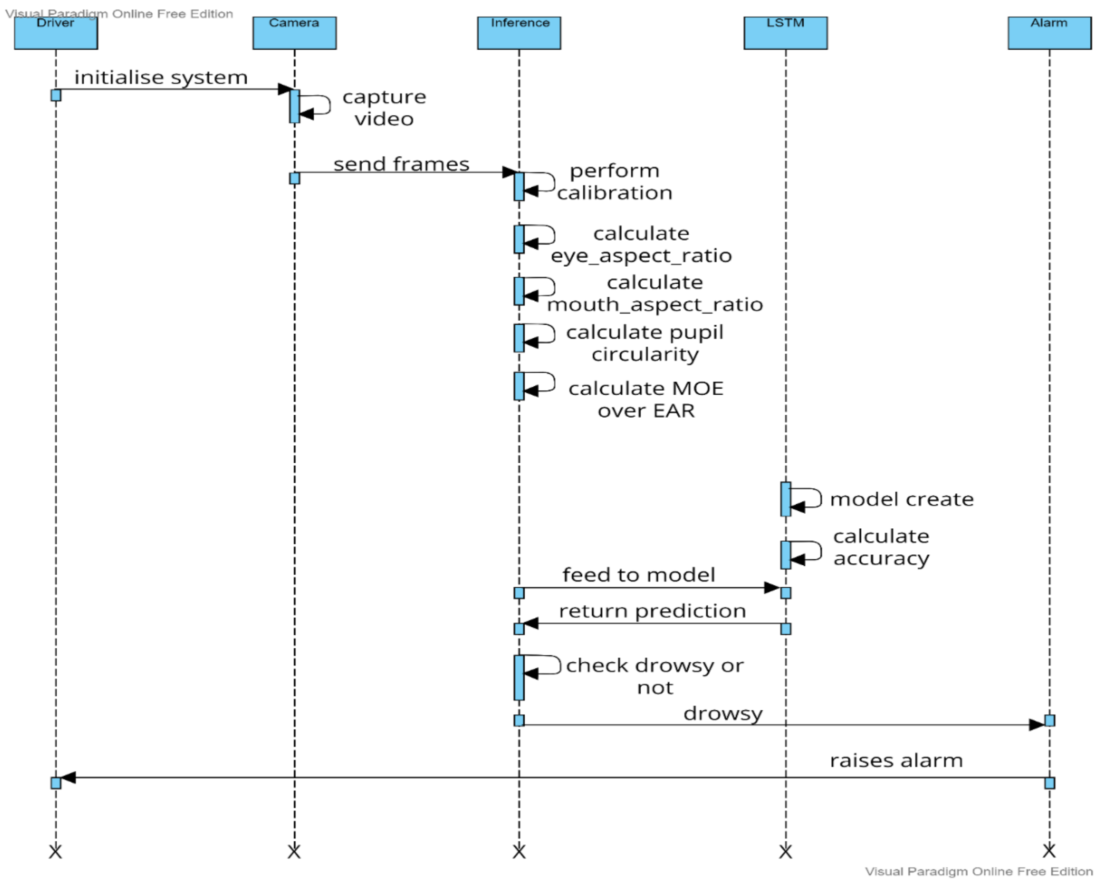


figure 4.2.3: Sequence Diagram

ACTIVITY DIAGRAM:

Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behavior diagrams.
An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as −

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

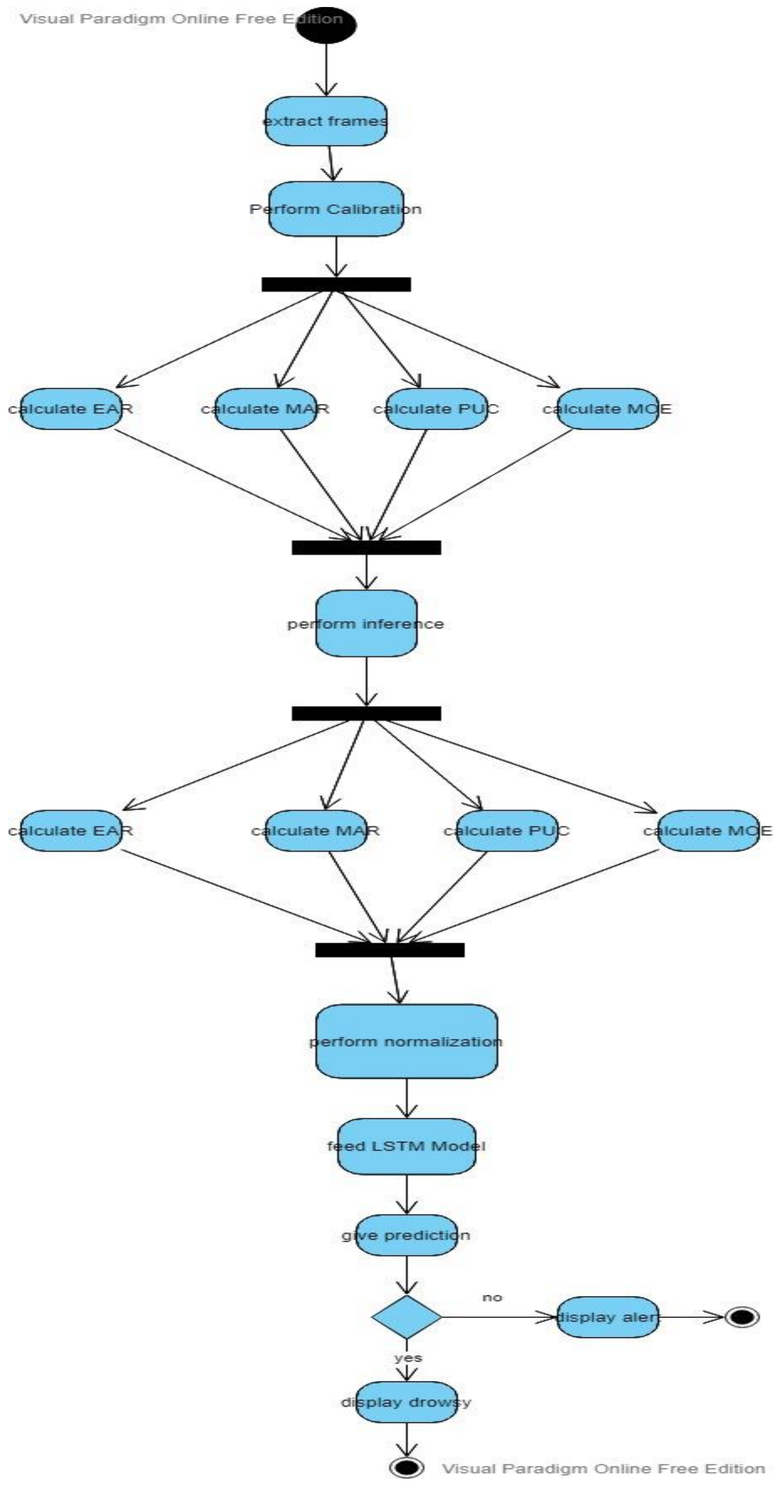


figure 4.2.4: Activity Diagram

5. IMPLEMENTATION AND RESULTS

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

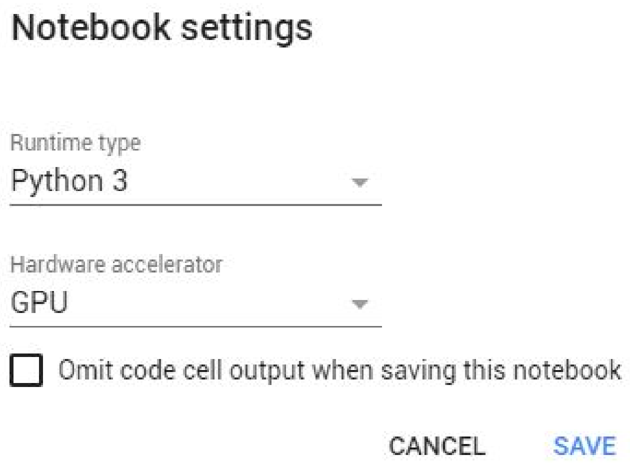
5.1 PROJECT IMPLEMENTATION STEPS

- Perform feature calculation and performing pre-processing which includes data normalization.
- Implementation of LSTM model
- Hardware Implementation.

5.2 IMPLEMENTATION

STEP-1 : Data Preparation

Setting up Google Colab: Google Colab is an online managed Jupyter Notebook environment where you can train deep learning models on GPU. The free plan of Google Colab allows you to train the deep learning model for up to 12 hrs before the runtime disconnects. You have to select runtime as GPU before launching the Jupyter notebook as shown below –


Notebook settings

Runtime type
Python 3

Hardware accelerator
GPU

☐ Omit code cell output when saving this notebook

Fig 5.2.1 – Notebook settings

We have uploaded the dataset on our google drive but before we can use it in Colab we have to mount our google drive directory onto our runtime environment as shown below. This command will generate a URL on which you need to click, authenticate your Google drive account and copy the authorization key over here and press enter.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

For each video, we used OpenCV to extract 1 frame per second starting at the 3-minute mark until the end of the video.

There are several user defined packages we need to import in order to create our own dataset. Those are drowsiness_functions.py and driver_standardization.py

But to import those packages, the system needs to be told about the path. Use the sys package and then type in this line of code.

```
import sys
sys.path.append('/content/gdrive/MyDrive/data')
```

All the required libraries and modules should be imported for data preparation, pre-processing, and model training.

```
import tensorflow
import drowsiness_standardisation
import drowsiness_functions
import os
from mlxtend.image import extract_face_landmarks
import cv2
import numpy as np
import pandas as pd
from drowsiness_standardisation import Standardization
import matplotlib.pyplot as plt
import torch
import glob
from IPython.display import clear_output
from tensorflow.keras.utils import to_categorical
from keras.layers import Dense, Flatten, Dropout, ZeroPadding3D
from keras.layers.recurrent import LSTM
from keras.models import Sequential, load_model
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.layers.wrappers import TimeDistributed
from keras.layers.convolutional import (Conv2D, MaxPooling3D, Conv3D,
   MaxPooling2D)
from collections import deque
```

Now we need to capture facial landmarks by extracting 1FPS in each video starting at 3 minute mark.

To Load the frames from videos

```
# loading frames from videos
# x = folder containing individual participant subfolders containing videos
x = os.listdir('/content/gdrive/MyDrive/data/videos/')
for i,j in zip(x, [os.listdir('/content/gdrive/MyDrive/data/videos/'+i) for i in x]):
   frame_extraction(folder=i, participants=j)
```

Now extract the facial landmarks from each of the .mov videos

These .mov videos are labelled as 0.mov, 5.mov, 10.mov

0 represents : Alert(vigilant), 5 represents : neither Alert or Drowsy, 10 represents : Drowsy

Extract each frame and then calculate their eye aspect ratio, mouth aspect ratio, pupil circularity, mouth aspect ratio over eye aspect ratio.

And then store the calculated values in csv file format along with the labels(state) csv file.

```
# Extracting 1 FPS in each video starting at 3 minute mark (skip beginning)
def frame_extraction(folder , participants, MAX_FRAMES=240):
   def getFrame(sec):
      """Capture frames from video at predefined starting point and calculates mea
sures"""
      start = 180000
      vidcap.set(cv2.CAP_PROP_POS_MSEC, start + sec*1000)
      hasFrames, image = vidcap.read()
      return hasFrames, image
   participant, state = 0, 0
   # Loop through all videos in a specific folder. Best results with .mov files.
   for j in participants:
      for i in np.arange(0, 11, 5):
         # using video capture class from OpenCV
         clear_output(wait=True) #flush
         data = []
         labels = []
         # drive where the UTA videos are located
```

```
        vidcap = cv2.VideoCapture(r'/content/gdrive/MyDrive/data/videos/' + str(f
older) + '/' + str(j) +'/' + str(i) + '.mov')

        sec = 0
        frameRate = 1
        success, image  = getFrame(sec)
        participant = j
        state = i
        count = 0
        # Extract frames (per video), the more frames the longer the extraction tak
es
        while success and count < MAX_FRAMES:
           landmarks = extract_face_landmarks(image)
           if landmarks is not None and sum(sum(landmarks)) != 0:
              data.append(landmarks)
              labels.append([i])
              sec = sec + frameRate
              sec = round(sec, 2)
              success, image = getFrame(sec)
              count = count+1
           else:
              sec = sec + frameRate
              sec = round(sec, 2)
              success, image = getFrame(sec)
              print("not detected")
        if success:
           data = np.array(data)
           labels = np.array(labels)
           features = []
           for d in data:
              # extract relevant facial landmarks from OpenCV
              landmarks = d[36:68]
              # here we create our features for our base classifier
              ear = eye_aspect_ratio(landmarkse)
              mar = mouth_aspect_ratio(landmarks)
              cir = circularity(landmarks)
              mouth_eye = mouth_over_eye(landmarks)
```

```
            features.append([int(participant), ear, mar, cir, mouth_eye])
        features = np.array(features)

        # save captured data as csv files with numpy for performance reasons
        np.savetxt('/content/gdrive/MyDrive/data/drowsiness_formula_files/' +
str(folder) +'_features_'+str(participant)+'_'+str(state)+'.csv', features, delimiter =
",")
        np.savetxt('/content/gdrive/MyDrive/data/drowsiness_formula_files/' +
str(folder) +'_labels_'+str(participant)+'_'+str(state)+'.csv', labels, delimiter = ",")
```

Make script for calculating facial features such as eye aspect ratio, mouth aspect ratio, pupil circularity, mouth aspect ratio over eye aspect ratio, So it could be used to calculate facial features extracted from faces detected in frames using dlib landmark shape predictor.
These functions are called when the facial features are to be calculated from the face detected in frame extracted from video.

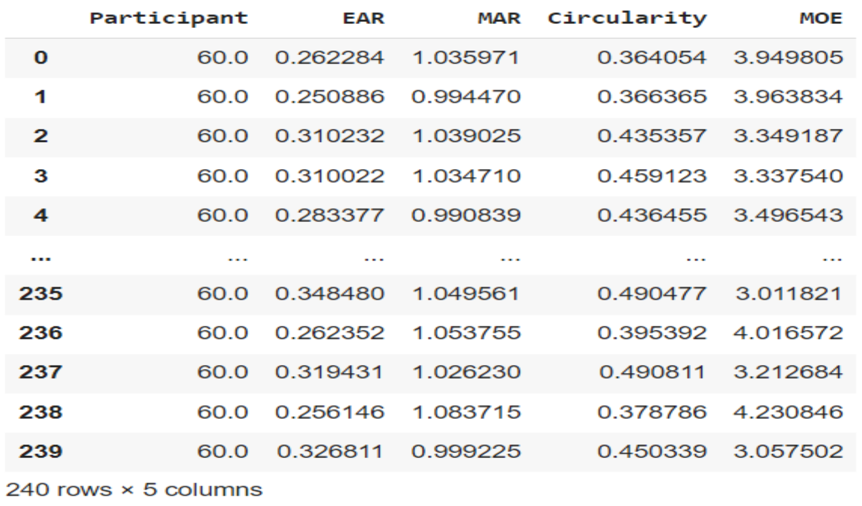
drowsiness_functions.py

```
from scipy.spatial import distance
def eye_aspect_ratio(eye):
    #calculating eye aspect ratio
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

def mouth_aspect_ratio(mouth):
    # calculating mouth aspect ratio
    A = distance.euclidean(mouth[14], mouth[18])
    C = distance.euclidean(mouth[12], mouth[16])
    mar = (A ) / (C)
    return mar

def circularity(eye):
    # calculating pupil cicularity
    A = distance.euclidean(eye[1], eye[4])
    radius  = A/2.0
    Area = math.pi * (radius ** 2)
    p = 0
    p += distance.euclidean(eye[0], eye[1])
    p += distance.euclidean(eye[1], eye[2])
    p += distance.euclidean(eye[2], eye[3])
    p += distance.euclidean(eye[3], eye[4])
    p += distance.euclidean(eye[4], eye[5])
    p += distance.euclidean(eye[5], eye[0])
    return 4 * math.pi * Area / (p**2)

def mouth_over_eye(eye):
    # calculating mouth over eye aspect ratio
    ear = eye_aspect_ratio(eye)
    mar = mouth_aspect_ratio(eye)
    mouth_eye = mar/ear
    return mouth_eye
```

Check the csv file if it is properly been saved with calculated features of each frame.

```
import pandas as pd
pd.read_csv(r"/content/gdrive/MyDrive/data/drowsiness_formula_files/Fold5_pa
rt2_features_60_5.csv", header=None, names=["Participant", "EAR", "MAR", "Circ
ularity", "MOE"])
```

Output:



```
     Participant       EAR       MAR  Circularity       MOE
0           60.0  0.262284  1.035971     0.364054  3.949805
1           60.0  0.250886  0.994470     0.366365  3.963834
2           60.0  0.310232  1.039025     0.435357  3.349187
3           60.0  0.310022  1.034710     0.459123  3.337540
4           60.0  0.283377  0.990839     0.436455  3.496543
...          ...       ...       ...          ...       ...
235         60.0  0.348480  1.049561     0.490477  3.011821
236         60.0  0.262352  1.053755     0.395392  4.016572
237         60.0  0.319431  1.026230     0.490811  3.212684
238         60.0  0.256146  1.083715     0.378786  4.230846
239         60.0  0.326811  0.999225     0.450339  3.057502

240 rows × 5 columns
```

Fig 5.2.2 – Table containing calculated facial features of each participant

Now read in all the feature files and merge them to one.

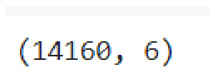
```
# Read in all feature files and merge them to one
df_features = [pd.read_csv(f, header=None, names=["Participant", "EAR", "MAR",
"Circularity", "MOE"]) for f in glob.glob("/content/gdrive/MyDrive/data/drowsine
ss_formula_files/Fold*_features_*.csv")]
df_labels = [pd.read_csv(f, header=None, names=["Y"]) for f in glob.glob("/conten
t/gdrive/MyDrive/data/drowsiness_formula_files/Fold*_labels_*.csv")]

df = pd.concat([pd.concat(df_features,ignore_index=True), pd.concat(df_labels,ig
nore_index=True)], axis=1)
print(df.shape)
# Save merged file
df.to_csv(r'/content/gdrive/MyDrive/data/totalwithrespondent.csv',index=False)
```

Save the file as totalwithrespondent.csv

Now import the totalwithrespondent.csv file

```
#Reading the CSV back into a dataframe
df_total = pd.read_csv('/content/gdrive/MyDrive/data/totalwithrespondent.csv')
# check the shape of the file
df_total.head()
df_total.shape
```

Output:



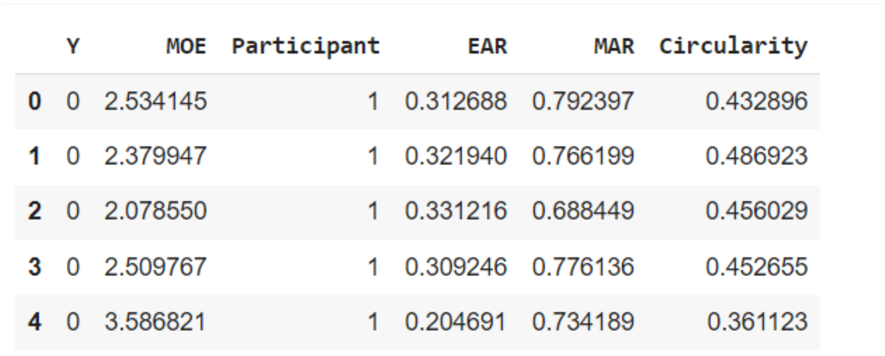
```
(14160, 6)
```

Fig 5.2.3 - The merged file now consists of 14160 data rows and 6 attributes.

Reorder the totalwithrespondent.csv file attributes

```
#Reordering the columns
cols=df_total.columns.tolist()
cols=cols[-1:] + cols[4:5] + cols[:4]
df_total = df_total[cols]
df_total.head()
```

Output:



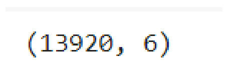
	Y	MOE	Participant	EAR	MAR	Circularity
0	0	2.534145	1	0.312688	0.792397	0.432896
1	0	2.379947	1	0.321940	0.766199	0.486923
2	0	2.078550	1	0.331216	0.688449	0.456029
3	0	2.509767	1	0.309246	0.776136	0.452655
4	0	3.586821	1	0.204691	0.734189	0.361123

Fig 5.2.4 – Reordering the columns

Check if the candidate only has 5.0 labeled data and drop them from dataset.

```
# Check if a candidate has only 5.0 labeled data
t = df_total.drop_duplicates("Participant")
df_total = df_total[~df_total.Participant.isin(t[t.Y > 0].Participant)]
df_total.head()
df_total.shape
```

Output:



```
(13920, 6)
```

Fig 5.2.5 - The dataset now consists of 13920 data rows and 6 attributes.

Now the reason we perform standardization is every person don’t have the same neutral measurements of their eyes and mouth. If we don’t perform standardization , we can face lot of issues when performing it on different persons. If one person is categorized as alert, another person with different facial measurements can be detected as drowsy. So Standardization must be performed.

Write the drowsiness_standardization script using z-score normalization.

drowsiness_standardization.py

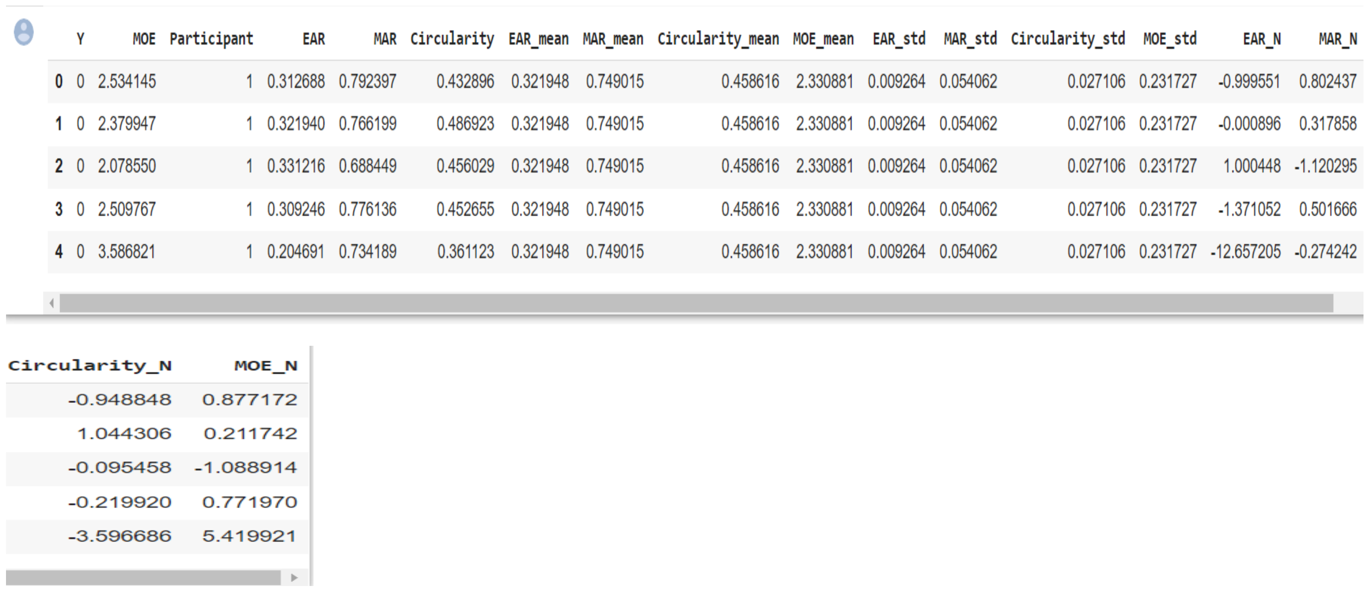
```
class Standardization:

    def __init__(self, df_total):
        self.df_total = df_total

    #Functions for getting mean and std of each feature
    def calculate_Standardization(self):
        def mean_EAR(respondent):
            return df_means.loc[respondent]["EAR"]

        def mean_MAR(respondent):
            return df_means.loc[respondent]["MAR"]

        def mean_Circularity(respondent):
            return df_means.loc[respondent]["Circularity"]

        def mean_MOE(respondent):
            return df_means.loc[respondent]["MOE"]

        def std_EAR(respondent):
            return df_std.loc[respondent]["EAR"]

        def std_MAR(respondent):
            return df_std.loc[respondent]["MAR"]

        def std_Circularity(respondent):
            return df_std.loc[respondent]["Circularity"]
        def std_MOE(respondent):
            return df_std.loc[respondent]["MOE"]
        #Separating the rows which are "Alert" only
        df_alert = self.df_total[self.df_total["Y"] == 0]

        #Creating separate dataframes for each participants's first, second and third
"Alert" frame
        df_alert_1 = df_alert.iloc[0::240, :]
        df_alert_2 = df_alert.iloc[1::240, :]
        df_alert_3 = df_alert.iloc[2::240, :]

        #Merging them into one dataframe
        alert_first3 = [df_alert_1,df_alert_2,df_alert_3]
        df_alert_first3 = pd.concat(alert_first3)
        df_alert_first3 = df_alert_first3.sort_index()

        #Based on the first 3 "Alert" frames, calculating per participant the mean and
std for each feature
        pd.options.mode.chained_assignment = None
        df_means = df_alert_first3.groupby("Participant")[["EAR", "MAR",
"Circularity", "MOE"]].mean()
        df_std = df_alert_first3.groupby("Participant")[["EAR", "MAR", "Circularity",
"MOE"]].std()

        #Adding respondent-wise mean and std for each feature to each row in the
original dataframe
        self.df_total["EAR_mean"] = self.df_total["Participant"].apply(mean_EAR)
        self.df_total["MAR_mean"] = self.df_total["Participant"].apply(mean_MAR)
        self.df_total["Circularity_mean"] =
self.df_total["Participant"].apply(mean_Circularity)
        self.df_total["MOE_mean"] = self.df_total["Participant"].apply(mean_MOE)

        self.df_total["EAR_std"] = self.df_total["Participant"].apply(std_EAR)
        self.df_total["MAR_std"] = self.df_total["Participant"].apply(std_MAR)
        self.df_total["Circularity_std"] =
self.df_total["Participant"].apply(std_Circularity)
        self.df_total["MOE_std"] = self.df_total["Participant"].apply(std_MOE)
```

```
        #Calculating now normalized features for each row in the original dataframe
        self.df_total["EAR_N"] = (self.df_total["EAR"] - self.df_total["EAR_mean"]) /
self.df_total["EAR_std"]
        self.df_total["MAR_N"] = (self.df_total["MAR"] - self.df_total["MAR_mean"])
/ self.df_total["MAR_std"]
        self.df_total["Circularity_N"] = (self.df_total["Circularity"] -
self.df_total["Circularity_mean"]) / self.df_total["Circularity_std"]
        self.df_total["MOE_N"] = (self.df_total["MOE"] - self.df_total["MOE_mean"])
/ self.df_total["MOE_std"]
        return self.df_total
```

Now calculate normalized values for each every feature in dataset in every row

```
pd.options.mode.chained_assignment = None

# Performing Standardization on the dataset
df_total = Standardization(df_total).calculate_Standardization()
df_total.head()
```

Output:



	Y	MOE	Participant	EAR	MAR	Circularity	EAR_mean	MAR_mean	Circularity_mean	MOE_mean	EAR_std	MAR_std	Circularity_std	MOE_std	EAR_N	MAR_N
0	0	2.534145	1	0.312688	0.792397	0.432896	0.321948	0.749015	0.458616	2.330881	0.009264	0.054062	0.027106	0.231727	-0.999551	0.802437
1	0	2.379947	1	0.321940	0.766199	0.486923	0.321948	0.749015	0.458616	2.330881	0.009264	0.054062	0.027106	0.231727	-0.000896	0.317858
2	0	2.078550	1	0.331216	0.688449	0.456029	0.321948	0.749015	0.458616	2.330881	0.009264	0.054062	0.027106	0.231727	1.000448	-1.120295
3	0	2.509767	1	0.309246	0.776136	0.452655	0.321948	0.749015	0.458616	2.330881	0.009264	0.054062	0.027106	0.231727	-1.371052	0.501666
4	0	3.586821	1	0.204691	0.734189	0.361123	0.321948	0.749015	0.458616	2.330881	0.009264	0.054062	0.027106	0.231727	-12.657205	-0.274242

Circularity_N	MOE_N
-0.948848	0.877172
1.044306	0.211742
-0.095458	-1.088914
-0.219920	0.771970
-3.596686	5.419921

Fig 5.2.6 – Performing Standardization on dataset

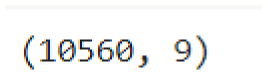
Save the file to a with the updated normalized values

```
#saving file to a csv with all information
df_total.to_csv(r'/content/gdrive/MyDrive/data/totalwithallinfo.csv',index=False)

# Saving the file to a CSV with all the information
df_main = df_total.drop(["EAR_mean","MAR_mean", "Circularity_mean", "MOE_
mean", "EAR_std", "MAR_std", "Circularity_std", "MOE_std"], axis=1)
df_main.to_csv(r'/content/gdrive/MyDrive/data/totalwithmaininfo.csv',index=Fal
se)
```

Remove all the rows which consists of "5" Y attribute value. Because for binary class classification , we only need of Alert state and Drowsy state but not for "Not Alert or Not Drowsy".

```
# read in prepared data
df = pd.read_csv(r'/content/gdrive/MyDrive/data/totalwithmaininfo.csv',sep=',')
participants = set(df.Participant)
df = df.drop(["Participant"], axis=1)
df = df[df.Y != 5.0]      # remove all the rows which consist of "5" Y attribute value(It represents n
either Alert or Drowsy).
df.loc[df.Y == 0.0, "Y"] = int(0)
df.loc[df.Y == 10.0, "Y"] = int(1)

df.shape
```

Output: It now only consists of 10560 data rows and 9 attributes.



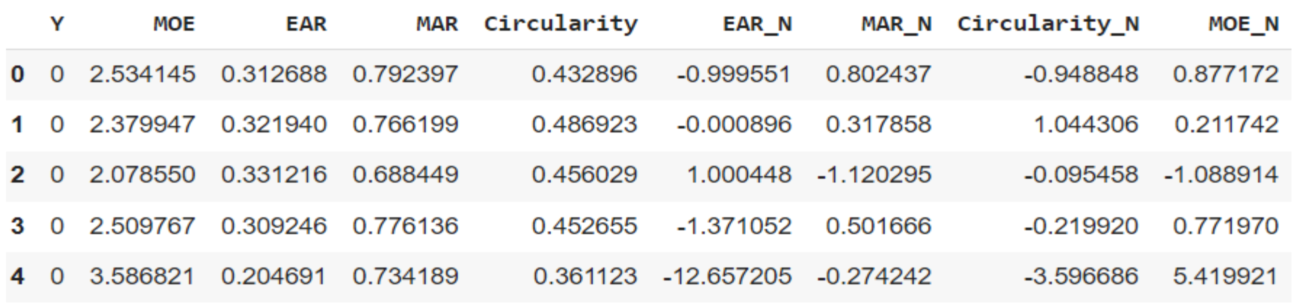
```
(10560, 9)
```

Fig 5.2.7 – Prepared data consists of 10560 data rows and 9 attributes.

Displaying the attributes in dataset.

df.head()

Output:



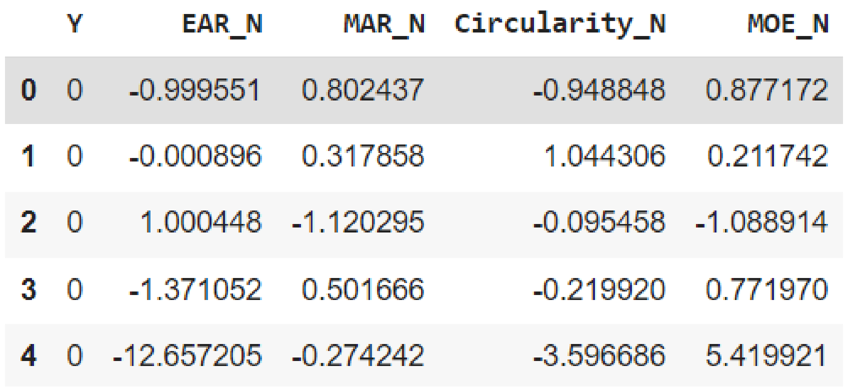
	Y	MOE	EAR	MAR	Circularity	EAR_N	MAR_N	Circularity_N	MOE_N
0	0	2.534145	0.312688	0.792397	0.432896	-0.999551	0.802437	-0.948848	0.877172
1	0	2.379947	0.321940	0.766199	0.486923	-0.000896	0.317858	1.044306	0.211742
2	0	2.078550	0.331216	0.688449	0.456029	1.000448	-1.120295	-0.095458	-1.088914
3	0	2.509767	0.309246	0.776136	0.452655	-1.371052	0.501666	-0.219920	0.771970
4	0	3.586821	0.204691	0.734189	0.361123	-12.657205	-0.274242	-3.596686	5.419921

Fig 5.2.8 – Displaying attributes in dataset

Only keep the normalized attributes and it's corresponding labels and drop the rest of the attributes for model training.

```
df = df.drop(["MOE","EAR","MAR","Circularity"], axis=1)
df.head()
```

Output:



	Y	EAR_N	MAR_N	Circularity_N	MOE_N
0	0	-0.999551	0.802437	-0.948848	0.877172
1	0	-0.000896	0.317858	1.044306	0.211742
2	0	1.000448	-1.120295	-0.095458	-1.088914
3	0	-1.371052	0.501666	-0.219920	0.771970
4	0	-12.657205	-0.274242	-3.596686	5.419921

Fig 5.2.9 - The dataset now only consists of 4 normalized attributes and it's labels.

STEP-2 : Model Training

Now the Dataset which has been preprocessed has been prepared in df dataframe.

Now the first step is to split the dataset into test dataset and train dataset respectively.

1). X_train - This includes your all independent variables,these will be used to train the model.

2). X_test - This is remaining 20% portion of the independent variables from the data which will not be used in the training phase and will be used to make predictions to test the accuracy of the model.

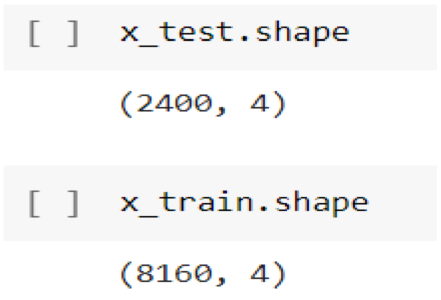
3). y_train - This is your dependent variable which needs to be predicted by this model, this includes category labels against your independent variables, we need to specify our dependent variable while training/fitting the model.

4). y_test - This data has category labels for your test data, these labels will be used to test the accuracy between actual and predicted categories.

```
#split it into train and test datasets
train_percentage = 17/22
train_index = int(len(df)*train_percentage)
test_index = len(df)-train_index
df_train = df[:train_index]
df_test = df[-test_index:]
x_test = df_test.drop(["Y"],axis=1)
y_test = df_test["Y"]
x_train = df_train.drop('Y',axis=1)
y_train = df_train['Y']
```

Check the shape of the x_train dataset and x_test dataset.

Output:



```
[ ]  x_test.shape

     (2400, 4)

[ ]  x_train.shape

     (8160, 4)
```

Fig 5.2.10 - Check the shape of the x_train dataset and x_test dataset.

The train dataset and the test dataset has now been split into 8:2 ratio.

Visualize the class distribution of drowsy and alert attribute value within the train dataset and test dataset.

```
# class distribution
label = 'Alert', 'Drowsy'
plt.figure(figsize = (8,8))
plt.subplot(121)
plt.title("Training Set")
plt.pie(df_train.groupby('Y').size(), labels = label, autopct='%1.1f%%', startangle=4
5, colors={"grey", "darkgrey"})
plt.subplot(122)
plt.title("Test Set")
plt.pie(df_test.groupby('Y').size(), labels = label, autopct='%1.1f%%', startangle=90
, colors={"grey", "darkgrey"})
plt.show()
```

Output:

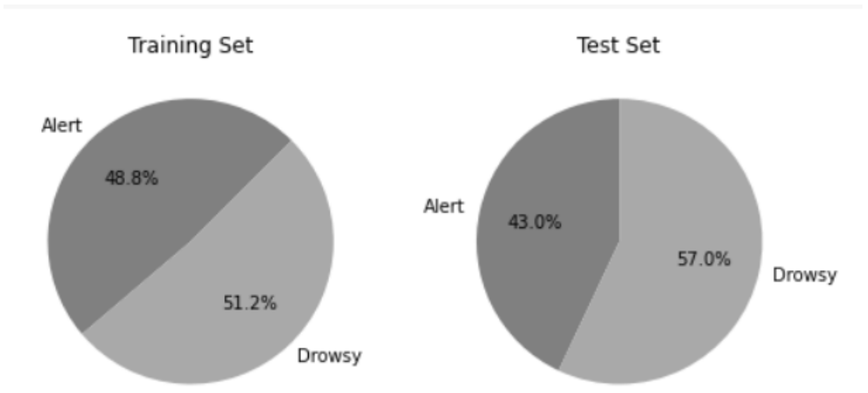
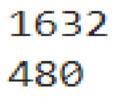


Fig 5.2.11 – Pie chart distribution of Training Dataset and Test Dataset

Now LSTM model only accepts 3D Representation of data but our current dataframe is of only 2 Dimensions, we need to convert x_train and x_test to 3D representation of data as "(1632,5,4)" and "(480,5,4)" to train our data, also reshape the y_train and y_test corresponding to x_train and x_test as(1632,1) and (480,1).

```
x_shaped_train  = np.array(x_train).reshape(1632,5,8)
x_shaped_test  = np.array(x_test).reshape(480,5,8)
y_train = np.array(y_train)
y_test = np.array(y_test)

y_shaped_train  = []
for i in range(0, len(y_train), 5):
  y_shaped_train.append([y_train[i]])
print(len(y_shaped_train))

y_shaped_test  = []
for i in range(0, len(y_test), 5):
  y_shaped_test.append([y_test[i]])
print(len(y_shaped_test))
```

<u>Output:</u>



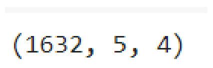
```
1632
480
```

Fig 5.2.12 – 3-Dimensional Sample size of training dataset and testing dataset

Check the shape of x_shaped_train, x_shaped_test, y_shaped_train, y_shaped_test.

```
x_shaped_train.shape  # shape of x_shaped_train dataframe
```

<u>Output:</u>

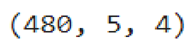


```
(1632, 5, 4)
```

Fig 5.2.13 – 3-Dimensional dataset of x_shaped_train

```
x_shaped_test.shape   # shape of x_shaped_test dataframe
```

<u>Output:</u>

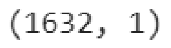


```
(480, 5, 4)
```

Fig 5.2.14 – 3-Dimensional dataset of x_shaped_test

```
y_shaped_train = np.array(y_shaped_train) # shape of y_shaped_train dataframe
y_shaped_train.shape
```

<u>Output:</u>

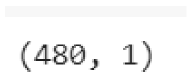


```
(1632, 1)
```

Fig 5.2.15 – 3-Dimensional dataset of y_shaped_train

```
y_shaped_test = np.array(y_shaped_test)   # shape of y_shaped_test dataframe
y_shaped_test.shape
```

<u>Output:</u>



```
(480, 1)
```

Fig 5.2.16 – 3-Dimensional dataset of y_shaped_test

In our proposed system, we are using Long Short-Term Memory(LSTM) Networks which helps in dealing with our sequential data. These are capable of learning long-term dependencies in the data. Compared to RNN which is also a feedback neural network, the reason LSTM neural networks are chosen because it allows to study long sequences of data without worrying about gradient vainishing problem.Vanishing gradient is as more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train. For shallow network with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively.

Gradients of neural networks are found using backpropagation. Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one. By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.

However, when *n* hidden layers use an activation like the sigmoid function, *n* small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.

A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

In our proposed system, the presence of the forget gate's activations allows the LSTM to decide, at each time step, that certain information should not be forgotten and to update the model's parameters accordingly.

In the model training code, we use accuracy metric and model = Sequential() , so we can define our own layers during model training.
Add a pre-fully dense connected layer into the network of 1024 units with an sigmoid activation function.

The next layer is our LSTM layer with 512 hidden units

It is possible to access the hidden state output for each input time step.

This can be done by setting the *return_sequences* attribute to *True* when defining the LSTM layer, as follows

return_sequences=True

Set the input_shape so that it expects 1 or more samples, 5 timesteps and 4 features and dropout value as 0.5.

After the relevant features have been obtained from the sequential data, Now we need to Flatten the result to 1 Dimension. Add the Flatten Layer to the network.

Then the flattened data is sent to 3 fully connected layers with sigmoid , tanh, and relu activation functions respectively, also add the Dropout layers which helps to reduce the overfitting of data.

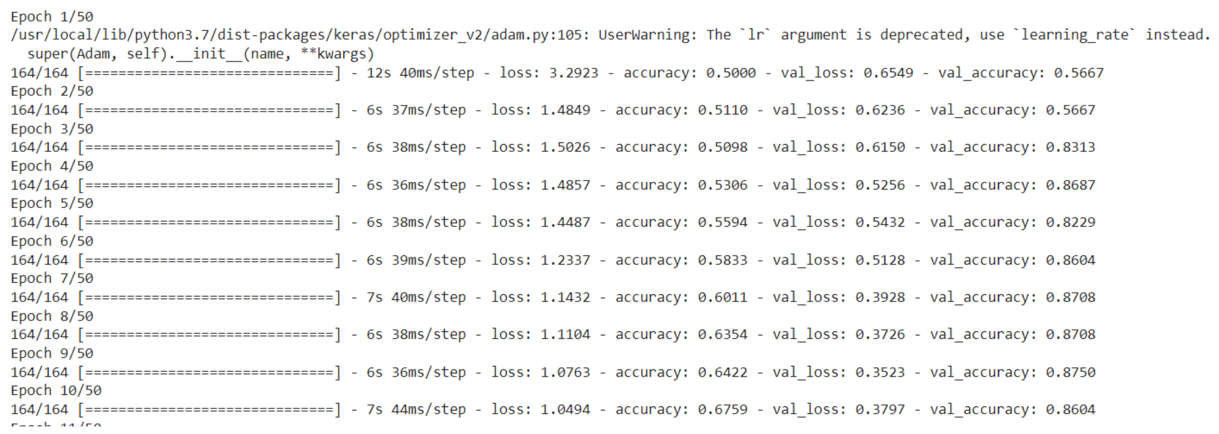
Finally add the output layer which gives final classification which predicts whether drowsy or not.

The network has finally been constructed, now compile the model with its loss function as loss='binary_crossentropy', optimizer as Adam which is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs and with accuracy metrics

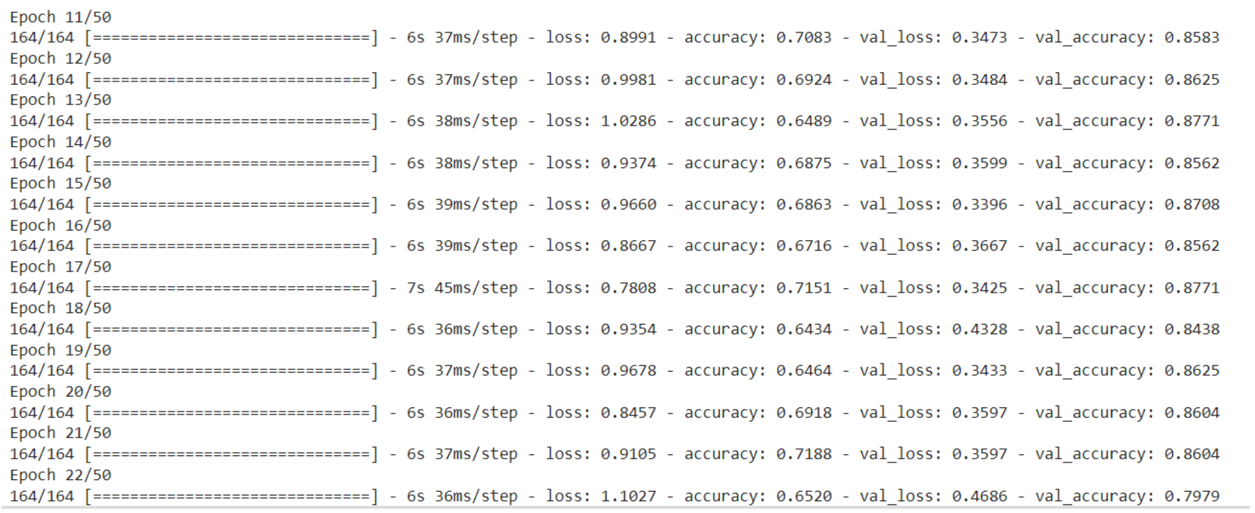
Now fit the model inorder to train the model with training data as x_shaped_train, y_shaped_train and validation data as x_shaped_test and y_shaped_test, with epochs as 50 and batch size set as 10.

```
metrics = ['accuracy']
model = Sequential()
model.add(Dense(1024, activation='sigmoid'))
model.add(LSTM(512, return_sequences=True,
               input_shape=(5, 4,),
               dropout=0.5))

model.add(Flatten())
# Dense is fully connected layer. 16 hidden units
# activation for lstm is basically sigmoid or tanh
model.add(Dense(216, activation='sigmoid')) #FC1
model.add(Dense(32, activation='tanh')) #FC2
model.add(Dropout(0.5))
model.add(Dense(16, activation='relu'))#FC3
model.add(Dropout(0.5))
model.add(Dense(1, activation='tanh'))#Output Layer
optimizer = Adam(lr=0.00005)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=metrics)
model.fit(x_shaped_train, y_shaped_train, validation_data = (x_shaped_test,y_sh
aped_test), epochs=50, batch_size=  10)
path = "/content/gdrive/MyDrive/data/lstmmodelgpu.pth"
torch.save(model, path)
```

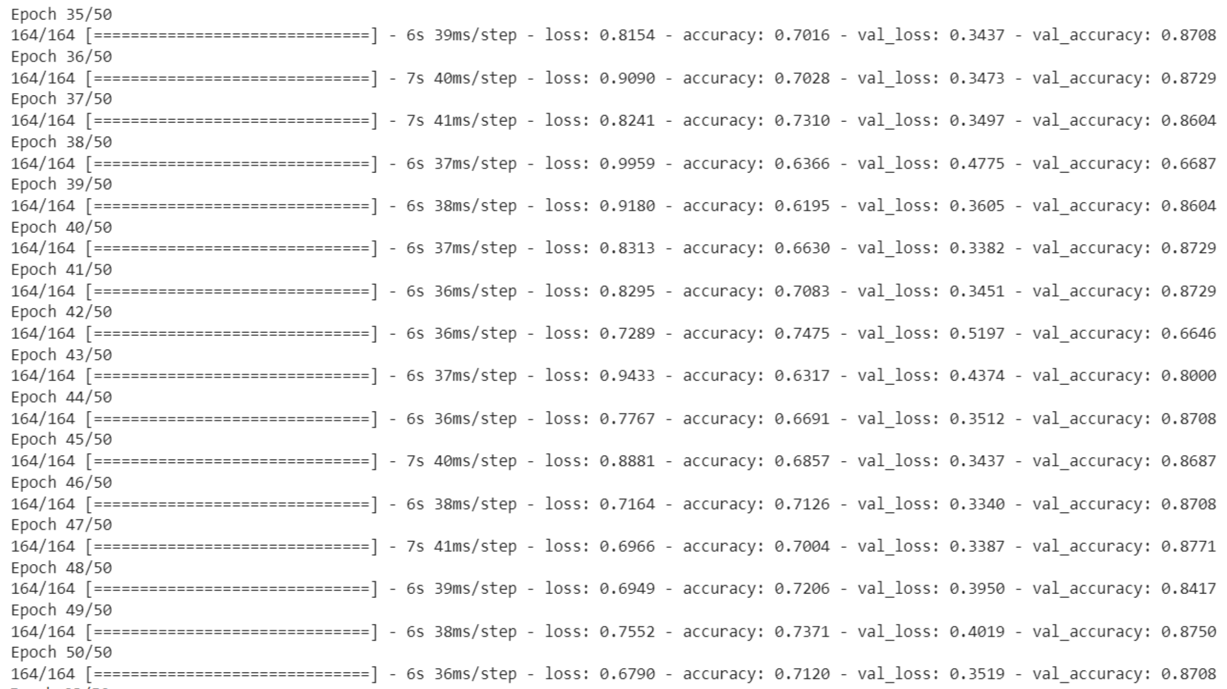
<u>Output:</u>



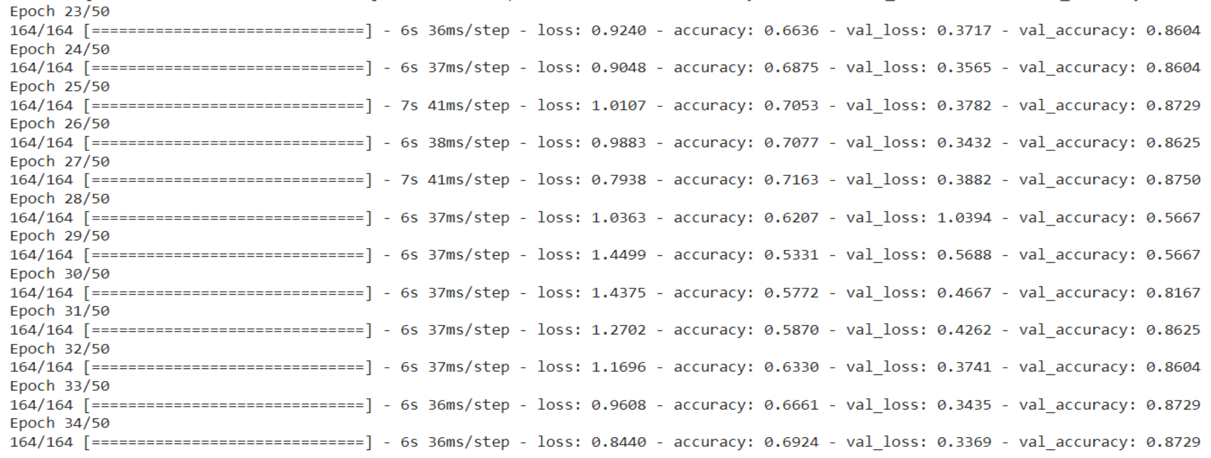
```
Epoch 1/50
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
164/164 [==============================] - 12s 40ms/step - loss: 3.2923 - accuracy: 0.5000 - val_loss: 0.6549 - val_accuracy: 0.5667
Epoch 2/50
164/164 [==============================] - 6s 37ms/step - loss: 1.4849 - accuracy: 0.5110 - val_loss: 0.6236 - val_accuracy: 0.5667
Epoch 3/50
164/164 [==============================] - 6s 38ms/step - loss: 1.5026 - accuracy: 0.5098 - val_loss: 0.6150 - val_accuracy: 0.8313
Epoch 4/50
164/164 [==============================] - 6s 36ms/step - loss: 1.4857 - accuracy: 0.5306 - val_loss: 0.5256 - val_accuracy: 0.8687
Epoch 5/50
164/164 [==============================] - 6s 38ms/step - loss: 1.4487 - accuracy: 0.5594 - val_loss: 0.5432 - val_accuracy: 0.8229
Epoch 6/50
164/164 [==============================] - 6s 39ms/step - loss: 1.2337 - accuracy: 0.5833 - val_loss: 0.5128 - val_accuracy: 0.8604
Epoch 7/50
164/164 [==============================] - 7s 40ms/step - loss: 1.1432 - accuracy: 0.6011 - val_loss: 0.3928 - val_accuracy: 0.8708
Epoch 8/50
164/164 [==============================] - 6s 38ms/step - loss: 1.1104 - accuracy: 0.6354 - val_loss: 0.3726 - val_accuracy: 0.8708
Epoch 9/50
164/164 [==============================] - 6s 36ms/step - loss: 1.0763 - accuracy: 0.6422 - val_loss: 0.3523 - val_accuracy: 0.8750
Epoch 10/50
164/164 [==============================] - 7s 44ms/step - loss: 1.0494 - accuracy: 0.6759 - val_loss: 0.3797 - val_accuracy: 0.8604
```



```
Epoch 11/50
164/164 [==============================] - 6s 37ms/step - loss: 0.8991 - accuracy: 0.7083 - val_loss: 0.3473 - val_accuracy: 0.8583
Epoch 12/50
164/164 [==============================] - 6s 37ms/step - loss: 0.9981 - accuracy: 0.6924 - val_loss: 0.3484 - val_accuracy: 0.8625
Epoch 13/50
164/164 [==============================] - 6s 38ms/step - loss: 1.0286 - accuracy: 0.6489 - val_loss: 0.3556 - val_accuracy: 0.8771
Epoch 14/50
164/164 [==============================] - 6s 38ms/step - loss: 0.9374 - accuracy: 0.6875 - val_loss: 0.3599 - val_accuracy: 0.8562
Epoch 15/50
164/164 [==============================] - 6s 39ms/step - loss: 0.9660 - accuracy: 0.6863 - val_loss: 0.3396 - val_accuracy: 0.8708
Epoch 16/50
164/164 [==============================] - 6s 39ms/step - loss: 0.8667 - accuracy: 0.6716 - val_loss: 0.3667 - val_accuracy: 0.8562
Epoch 17/50
164/164 [==============================] - 7s 45ms/step - loss: 0.7808 - accuracy: 0.7151 - val_loss: 0.3425 - val_accuracy: 0.8771
Epoch 18/50
164/164 [==============================] - 6s 36ms/step - loss: 0.9354 - accuracy: 0.6434 - val_loss: 0.4328 - val_accuracy: 0.8438
Epoch 19/50
164/164 [==============================] - 6s 37ms/step - loss: 0.9678 - accuracy: 0.6464 - val_loss: 0.3433 - val_accuracy: 0.8625
Epoch 20/50
164/164 [==============================] - 6s 36ms/step - loss: 0.8457 - accuracy: 0.6918 - val_loss: 0.3597 - val_accuracy: 0.8604
Epoch 21/50
164/164 [==============================] - 6s 37ms/step - loss: 0.9105 - accuracy: 0.7188 - val_loss: 0.3597 - val_accuracy: 0.8604
Epoch 22/50
164/164 [==============================] - 6s 36ms/step - loss: 1.1027 - accuracy: 0.6520 - val_loss: 0.4686 - val_accuracy: 0.7979
```



```
Epoch 35/50
164/164 [==============================] - 6s 39ms/step - loss: 0.8154 - accuracy: 0.7016 - val_loss: 0.3437 - val_accuracy: 0.8708
Epoch 36/50
164/164 [==============================] - 7s 40ms/step - loss: 0.9090 - accuracy: 0.7028 - val_loss: 0.3473 - val_accuracy: 0.8729
Epoch 37/50
164/164 [==============================] - 7s 41ms/step - loss: 0.8241 - accuracy: 0.7310 - val_loss: 0.3497 - val_accuracy: 0.8604
Epoch 38/50
164/164 [==============================] - 6s 37ms/step - loss: 0.9959 - accuracy: 0.6366 - val_loss: 0.4775 - val_accuracy: 0.6687
Epoch 39/50
164/164 [==============================] - 6s 38ms/step - loss: 0.9180 - accuracy: 0.6195 - val_loss: 0.3605 - val_accuracy: 0.8604
Epoch 40/50
164/164 [==============================] - 6s 37ms/step - loss: 0.8313 - accuracy: 0.6630 - val_loss: 0.3382 - val_accuracy: 0.8729
Epoch 41/50
164/164 [==============================] - 6s 36ms/step - loss: 0.8295 - accuracy: 0.7083 - val_loss: 0.3451 - val_accuracy: 0.8729
Epoch 42/50
164/164 [==============================] - 6s 36ms/step - loss: 0.7289 - accuracy: 0.7475 - val_loss: 0.5197 - val_accuracy: 0.6646
Epoch 43/50
164/164 [==============================] - 6s 37ms/step - loss: 0.9433 - accuracy: 0.6317 - val_loss: 0.4374 - val_accuracy: 0.8000
Epoch 44/50
164/164 [==============================] - 6s 36ms/step - loss: 0.7767 - accuracy: 0.6691 - val_loss: 0.3512 - val_accuracy: 0.8708
Epoch 45/50
164/164 [==============================] - 7s 40ms/step - loss: 0.8881 - accuracy: 0.6857 - val_loss: 0.3437 - val_accuracy: 0.8687
Epoch 46/50
164/164 [==============================] - 6s 38ms/step - loss: 0.7164 - accuracy: 0.7126 - val_loss: 0.3340 - val_accuracy: 0.8708
Epoch 47/50
164/164 [==============================] - 7s 41ms/step - loss: 0.6966 - accuracy: 0.7004 - val_loss: 0.3387 - val_accuracy: 0.8771
Epoch 48/50
164/164 [==============================] - 6s 39ms/step - loss: 0.6949 - accuracy: 0.7206 - val_loss: 0.3950 - val_accuracy: 0.8417
Epoch 49/50
164/164 [==============================] - 6s 38ms/step - loss: 0.7552 - accuracy: 0.7371 - val_loss: 0.4019 - val_accuracy: 0.8750
Epoch 50/50
164/164 [==============================] - 6s 36ms/step - loss: 0.6790 - accuracy: 0.7120 - val_loss: 0.3519 - val_accuracy: 0.8708
```



```
Epoch 23/50
164/164 [==============================] - 6s 36ms/step - loss: 0.9240 - accuracy: 0.6636 - val_loss: 0.3717 - val_accuracy: 0.8604
Epoch 24/50
164/164 [==============================] - 6s 37ms/step - loss: 0.9048 - accuracy: 0.6875 - val_loss: 0.3565 - val_accuracy: 0.8604
Epoch 25/50
164/164 [==============================] - 7s 41ms/step - loss: 1.0107 - accuracy: 0.7053 - val_loss: 0.3782 - val_accuracy: 0.8729
Epoch 26/50
164/164 [==============================] - 6s 38ms/step - loss: 0.9883 - accuracy: 0.7077 - val_loss: 0.3432 - val_accuracy: 0.8625
Epoch 27/50
164/164 [==============================] - 7s 41ms/step - loss: 0.7938 - accuracy: 0.7163 - val_loss: 0.3882 - val_accuracy: 0.8750
Epoch 28/50
164/164 [==============================] - 6s 37ms/step - loss: 1.0363 - accuracy: 0.6207 - val_loss: 1.0394 - val_accuracy: 0.5667
Epoch 29/50
164/164 [==============================] - 6s 37ms/step - loss: 1.4499 - accuracy: 0.5331 - val_loss: 0.5688 - val_accuracy: 0.5667
Epoch 30/50
164/164 [==============================] - 6s 37ms/step - loss: 1.4375 - accuracy: 0.5772 - val_loss: 0.4667 - val_accuracy: 0.8167
Epoch 31/50
164/164 [==============================] - 6s 37ms/step - loss: 1.2702 - accuracy: 0.5870 - val_loss: 0.4262 - val_accuracy: 0.8625
Epoch 32/50
164/164 [==============================] - 6s 37ms/step - loss: 1.1696 - accuracy: 0.6330 - val_loss: 0.3741 - val_accuracy: 0.8604
Epoch 33/50
164/164 [==============================] - 6s 36ms/step - loss: 0.9608 - accuracy: 0.6661 - val_loss: 0.3435 - val_accuracy: 0.8729
Epoch 34/50
164/164 [==============================] - 6s 36ms/step - loss: 0.8440 - accuracy: 0.6924 - val_loss: 0.3369 - val_accuracy: 0.8729
```

Fig 5.2.17 – Training of LSTM Model

STEP-3 : Calculate Accuracy

Use the model.predict(x_shaped_test) > 0.5 in order to check if it predicts each data row from x_shaped_test which consists of 480 data rows is being predicted as 0 or 1(Alert and Drowsy) .

```
predict=(model.predict(x_shaped_test) > 0.5).astype("int32")
# Checking how it predicts using the trained model
print(predict)
```

Output:

```
[[0][0][0][0][0][0][0][0][0][0][0][0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [1] [1] [1]
[1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [1] [1] [0] [0] [0]
[0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [0] [0] [0] [0] [1] [1] [1] [1] [1] [0] [0] [0]
[0] [0] [1] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0]
[0] [0] [0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1]
[1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1]
[1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [0] [0]
[0] [1] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [1] [0] [0] [0] [1] [0] [0] [0] [0] [0]
[0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [1] [1] [1] [1]
[1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [1] [1] [0] [0] [0] [0]
[0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [0] [0] [0] [1] [1] [1] [1] [1] [0] [0] [0] [0] [0]
[1] [1] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0]
[0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1]
[1] [1] [1] [1] [0] [0] [0] [0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1]
[1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0] [0] [0] [0] [0]
[0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0] [0]
[0] [0] [0] [0] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [1] [0] [0]
[0] [0]]
```

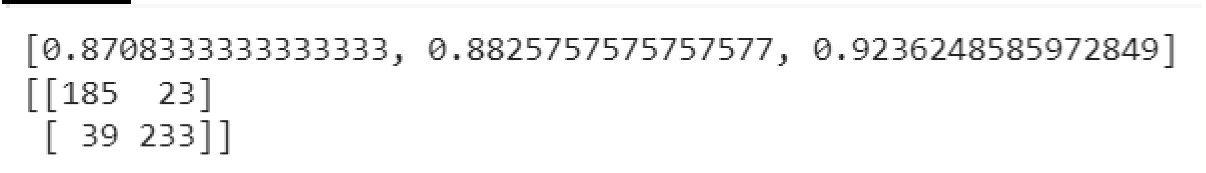
First import the required libraries for measuring accuracy of the model which was trained.

```
#import libraries for testing
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score, f1_score
from sklearn.metrics import accuracy_score
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.calibration import calibration_curve
```

For other measure of calculating accuracy , we can use the confusion matrix , roc auc score and f1 score.

```
pred_rnn=(model.predict(x_shaped_test) > 0.5).astype("int32")
y_score_10 = model.predict_on_batch(x_shaped_test)
acc10 = accuracy_score(y_shaped_test, pred_rnn)
f1_score_10 = metrics.f1_score(y_shaped_test, np.array(pred_rnn))
roc_10 = metrics.roc_auc_score(y_shaped_test, y_score_10)
print([acc10, f1_score_10, roc_10])
cf_matrix = confusion_matrix(y_shaped_test, pred_rnn)
print(cf_matrix)
```

Output:



```
[0.8708333333333333, 0.8825757575757577, 0.9236248585972849]
[[185  23]
 [ 39 233]]
```

Fig 5.2.18 – acc_score, f1-score, roc score of Trained LSTM Model

Accuracy score tries to calculate the fraction of predictions our model got right.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Fig 5.2.19 – Accuracy score formula

We got an accuracy score of approximatly by 87%.

Next is f1 score , The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. Precision quantifies the number of positive class predictions that actually belong to the positive class. Recall quantifies the number of positive class predictions made out of all positive examples in the dataset. F-Measure provides a single score that balances both the concerns of precision and recall in one number.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Fig 5.2.20 – F1-Score formula

We got an f1 score of about 88% for our classifier.
Next is Roc Auc score accuracy metric,
A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve, or ROC curve.
We got an roc accuracy score of 92% , generally the higher the score the better it is,
A confusion matrix can be used to obtain the rate of true positive results, true negative results, false positive results, false negative results from the classification model.
Output:

```
[[185  23]
 [ 39 233]]
```

Fig 5.2.21 – Confusion Matrix on test dataset

We can also plot a confusion matrix graph in order to understand the prediction of data in a much more clear manner.

```
import seaborn as sns

group_names = ['True Neg','False Pos','False Neg','True Pos']

group_counts = ["{0:0.0f}".format(value) for value in
            cf_matrix.flatten()]

group_percentages = ["{0:.2%}".format(value) for value in
                cf_matrix.flatten()/np.sum(cf_matrix)]

labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
        zip(group_names,group_counts,group_percentages)]

labels = np.asarray(labels).reshape(2,2)

ax = sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```

Output:

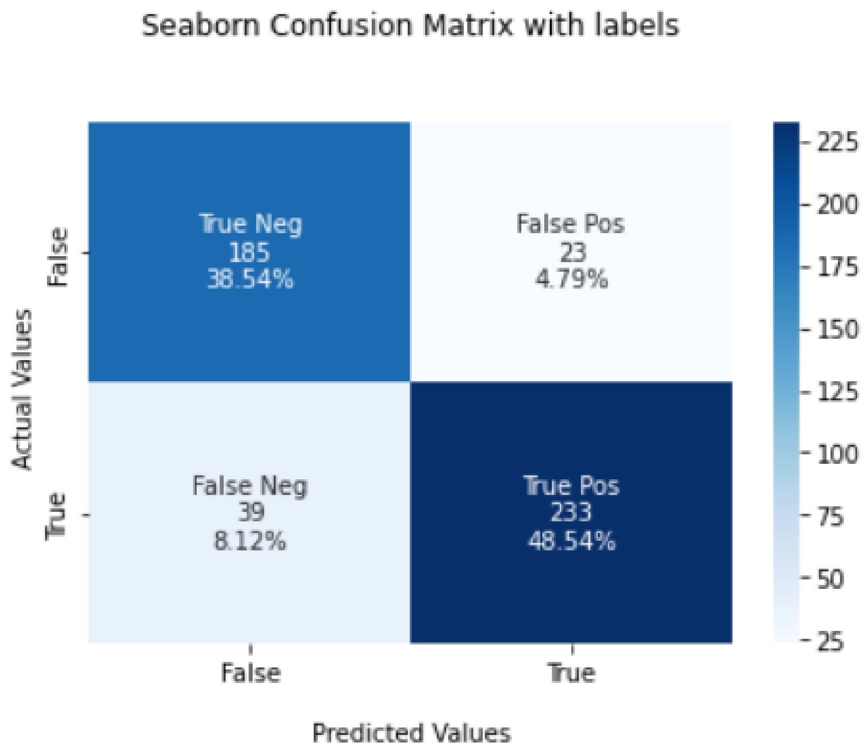


Fig 5.2.22– Confusion Matrix

Plot the roc curve, AUC - ROC curve is **a performance measurement for the classification problems at various threshold settings**. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.

```
plt.figure(figsize=(8,8))
plt.plot([0, 1], [0, 1],'r--')
fpr, tpr, thresholds = roc_curve(y_shaped_test, y_score_10)
plt.plot(fpr, tpr, label= 'ROC curve (area = %0.2f)' % roc_10)
plt.title('ROC Curve for LSTM')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
```

Output:

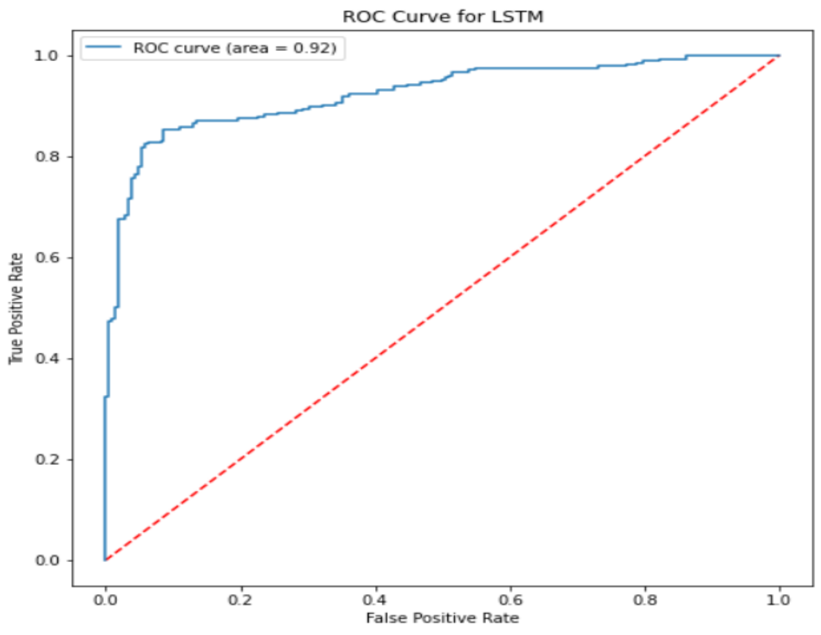


Fig 5.2.23 – True Postive rate VS False Positive Rate graph(ROC-Curve)

STEP-4 : Inference

Deep learning Inference is the process of using a trained LSTM model to make predictions against previously unseen data.

The code uses MediaPipe model for detecting a face in the video stream.

Features based on facial landmark positions are calculated and are used by a LSTM model to predict whether the use is alert or drowsy.

As a first step, calibration is performed, which calculates the facial features over a small number of frames. The user is reminded to be in neutral position in this process.

Calibration returns the mean and standard deviation values to normalize the facial features at inference step.

During inference, first the features are calculated and are normalized. These features are sent over to the LSTM model to make a prediction. The LSTM model computes its prediction over twenty five continuous frames and returns the predicted state.

Inference.py:

```
import cv2

import mediapipe as mp

import math

import numpy as np

import os

import time

import torch

import led_blink

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(26,GPIO.OUT)

cap = cv2.VideoCapture(0)#camera port 0
```

```
def distance(p1, p2):
    # Calculate distance between two points

    return (((p1[:2] - p2[:2])**2).sum())**0.5

def eye_aspect_ratio(landmarks, eye):
    # Calculate the ratio of the eye length to eye width.

    N1 = distance(landmarks[eye[1][0]], landmarks[eye[1][1]])
    N2 = distance(landmarks[eye[2][0]], landmarks[eye[2][1]])
    N3 = distance(landmarks[eye[3][0]], landmarks[eye[3][1]])
    D = distance(landmarks[eye[0][0]], landmarks[eye[0][1]])
    return (N1 + N2 + N3) / (3 * D)

def eye_feature(landmarks):
    # Calculate the eye feature as the average of the eye aspect ratio for the two
eyes

    return (eye_aspect_ratio(landmarks, left_eye) + \
    eye_aspect_ratio(landmarks, right_eye))/2

def mouth_feature(landmarks):
    # Calculate mouth feature as the ratio of the mouth length to mouth width

    N1 = distance(landmarks[mouth[1][0]], landmarks[mouth[1][1]])
    N2 = distance(landmarks[mouth[2][0]], landmarks[mouth[2][1]])
    N3 = distance(landmarks[mouth[3][0]], landmarks[mouth[3][1]])
    D = distance(landmarks[mouth[0][0]], landmarks[mouth[0][1]])
    return (N1 + N2 + N3)/(3*D)

def pupil_circularity(landmarks, eye):
    # Calculate pupil circularity feature.

    perimeter = distance(landmarks[eye[0][0]], landmarks[eye[1][0]]) + \
            distance(landmarks[eye[1][0]], landmarks[eye[2][0]]) + \
            distance(landmarks[eye[2][0]], landmarks[eye[3][0]]) + \
            distance(landmarks[eye[3][0]], landmarks[eye[0][1]]) + \
            distance(landmarks[eye[0][1]], landmarks[eye[3][1]]) + \
            distance(landmarks[eye[3][1]], landmarks[eye[2][1]]) + \
            distance(landmarks[eye[2][1]], landmarks[eye[1][1]]) + \
            distance(landmarks[eye[1][1]], landmarks[eye[0][0]])
    area = math.pi * ((distance(landmarks[eye[1][0]], landmarks[eye[3][1]]) * 0.5)
** 2)
    return (4*math.pi*area)/(perimeter**2)

def pupil_feature(landmarks):
    # Calculate the pupil feature as the average of the pupil circularity for the two
eyes

    return (pupil_circularity(landmarks, left_eye) + \
        pupil_circularity(landmarks, right_eye))/2

def run_face_mp(image):
    # Get face landmarks using the FaceMesh MediaPipe model.

    image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = face_mesh.process(image)

    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    if results.multi_face_landmarks:
        landmarks_positions = []
        # assume that only face is present in the image
        for _, data_point in enumerate(results.multi_face_landmarks[0].landmark):
            landmarks_positions.append([data_point.x, data_point.y, data_point.z]) #
saving normalized landmark positions
        landmarks_positions = np.array(landmarks_positions)
        landmarks_positions[:, 0] *= image.shape[1]
        landmarks_positions[:, 1] *= image.shape[0]

        # draw face mesh over image
        for face_landmarks in results.multi_face_landmarks:
                mp_drawing.draw_landmarks(
                image=image,
                landmark_list=face_landmarks,
                connections=mp_face_mesh.FACEMESH_CONTOURS,
                landmark_drawing_spec=drawing_spec,
                connection_drawing_spec=drawing_spec)

        ear = eye_feature(landmarks_positions)
        mar = mouth_feature(landmarks_positions)
        puc = pupil_feature(landmarks_positions)
        moe = mar/ear
    else:
        ear = -1000
        mar = -1000
        puc = -1000
        moe = -1000

    return ear, mar, puc, moe, image

def calibrate(calib_frame_count=25):
    # Perform calibration. Get features for the neutral position.

    ears = []
    mars = []
    pucs = []
    moes = []
    cap = cv2.VideoCapture(0)
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue

        ear, mar,puc, moe, image = run_face_mp(image)
        if ear != -1000:
            ears.append(ear)
            mars.append(mar)
            pucs.append(puc)
            moes.append(moe)

        cv2.putText(image, "Calibration", (int(0.02*image.shape[1]),
int(0.14*image.shape[0])),
                cv2.FONT_HERSHEY_SIMPLEX, 1.5, (255, 0, 0), 2)
        cv2.imshow('MediaPipe FaceMesh', image)
        if cv2.waitKey(5) & 0xFF == ord("q"):
            break
        if len(ears) >= calib_frame_count:
            break

    cv2.destroyAllWindows()
    cap.release()
    ears = np.array(ears)
    mars = np.array(mars)
    pucs = np.array(pucs)
    moes = np.array(moes)
    return [ears.mean(), ears.std()], [mars.mean(), mars.std()], \
        [pucs.mean(), pucs.std()], [moes.mean(), moes.std()]


def get_classification(input_data):
    # Perform classification over the facial  features.

    model_input = []
    model_input.append(input_data[:5])
    model_input.append(input_data[3:8])
    model_input.append(input_data[6:11])
    model_input.append(input_data[9:14])
    model_input.append(input_data[12:17])
    model_input.append(input_data[15:])
    model_input = torch.FloatTensor(np.array(model_input))

    preds = torch.sigmoid(model(model_input)).gt(0.5).int().data.numpy()
    print(preds)
    return int(preds.sum() >= 5)
```

```
def infer(ears_norm, mars_norm, pucs_norm, moes_norm):
    # Perform inference.
    ear_main = 0
    mar_main = 0
    puc_main = 0
    moe_main = 0
    decay = 0.9 # use decay to smoothen the noise in feature values

    label = None

    input_data = []
    frame_before_run = 0

    cap = cv2.VideoCapture(0)
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue

        ear, mar, puc, moe, image = run_face_mp(image)
        if ear != -1000:
            ear = (ear - ears_norm[0])/ears_norm[1]
            mar = (mar - mars_norm[0])/mars_norm[1]
            puc = (puc - pucs_norm[0])/pucs_norm[1]
            moe = (moe - moes_norm[0])/moes_norm[1]
            if ear_main == -1000:
                ear_main = ear
                mar_main = mar
                puc_main = puc
                moe_main = moe
            else:
                ear_main = ear_main*decay + (1-decay)*ear
                mar_main = mar_main*decay + (1-decay)*mar
                puc_main = puc_main*decay + (1-decay)*puc
                moe_main = moe_main*decay + (1-decay)*moe
        else:
            ear_main = -1000
            mar_main = -1000
            puc_main = -1000
            moe_main = -1000

        if len(input_data) == 20:
            input_data.pop(0)
        input_data.append([ear_main, mar_main, puc_main, moe_main])



        print("length")
        print(len(input_data))
        if len(input_data) == 20:
```

```
          label = get_classification(input_data)



        cv2.putText(image, "EAR: %.2f" %(ear_main), (int(0.02*image.shape[1]),
int(0.07*image.shape[0])),
            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
        cv2.putText(image, "MAR: %.2f" %(mar_main), (int(0.27*image.shape[1]),
int(0.07*image.shape[0])),
            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
        cv2.putText(image, "PUC: %.2f" %(puc_main), (int(0.52*image.shape[1]),
int(0.07*image.shape[0])),
            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
        cv2.putText(image, "MOE: %.2f" %(moe_main), (int(0.77*image.shape[1]),
int(0.07*image.shape[0])),
            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
        if label is not None:
          if label == 0:
            color = (0, 255, 0)
          else:
            color = (0, 0, 255)
             led_blink.bb()
          cv2.putText(image, "%s" %(states[label]), (int(0.02*image.shape[1]),
int(0.2*image.shape[0])),
                  cv2.FONT_HERSHEY_SIMPLEX, 1.5, color, 2)
        cv2.imshow('MediaPipe FaceMesh', image)
        if cv2.waitKey(5) & 0xFF == ord("q"):
            break

    cv2.destroyAllWindows()
    cap.release()



right_eye = [[33, 133], [160, 144], [159, 145], [158, 153]] # right eye landmark
positions
left_eye = [[263, 362], [387, 373], [386, 374], [385, 380]] # left eye landmark
positions
mouth = [[61, 291], [39, 181], [0, 17], [269, 405]] # mouth landmark coordinates
states = ['alert', 'drowsy']

# Declaring FaceMesh model
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(
    min_detection_confidence=0.3, min_tracking_confidence=0.8)
mp_drawing = mp.solutions.drawing_utils
drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)

model_lstm_path = 'lstmmodelgpu.pth'
model = torch.jit.load(model_lstm_path)
```

```
model.eval()

print ('Starting calibration. Please be in neutral state')
time.sleep(1)
ears_norm, mars_norm, pucs_norm, moes_norm = calibrate()
print ('Starting main application')
time.sleep(1)
infer(ears_norm, mars_norm, pucs_norm, moes_norm)
face_mesh.close()
GPIO.cleanup()
```

GPIO Buzzer Code:

led_blink.py

```
import time
import RPi.GPIO as GPIO
def bb():
GPIO.setmode(GPIO.BCM)
GPIO.setup(26,GPIO.OUT)
GPIO.output(26,True)
time.sleep(3)
GPIO.output(26,False)
time.sleep(3)
GPIO.cleanup()
```

5.3 **OUTPUT SCREENS**:

Calibration:

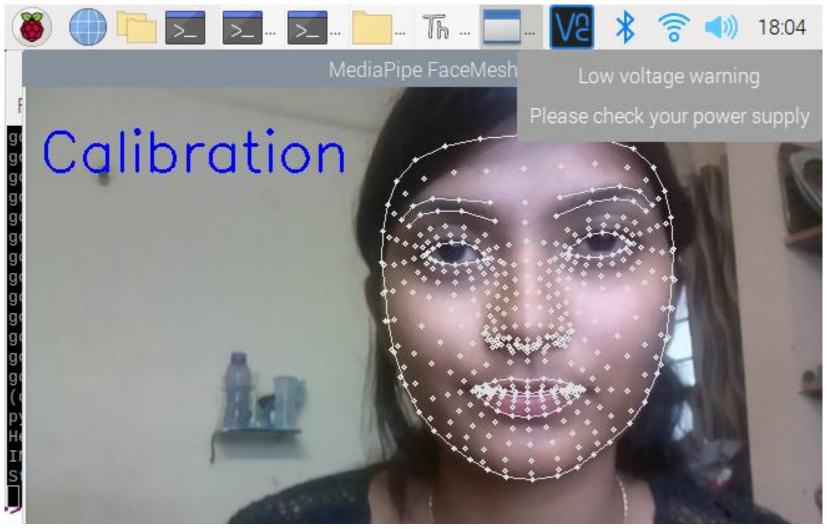


Fig 5.3.1 – Output displaying Calibration

Alert:

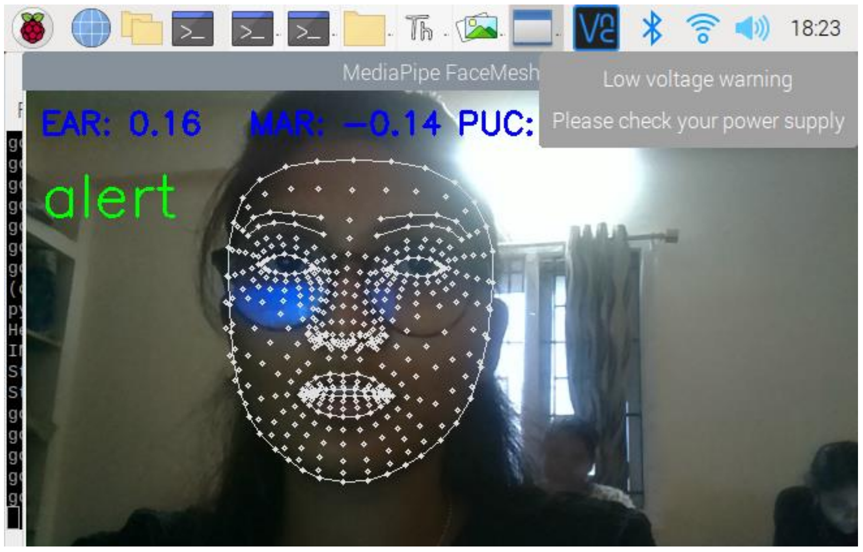


Fig 5.3.2 – Output displaying Alert

Drowsy:

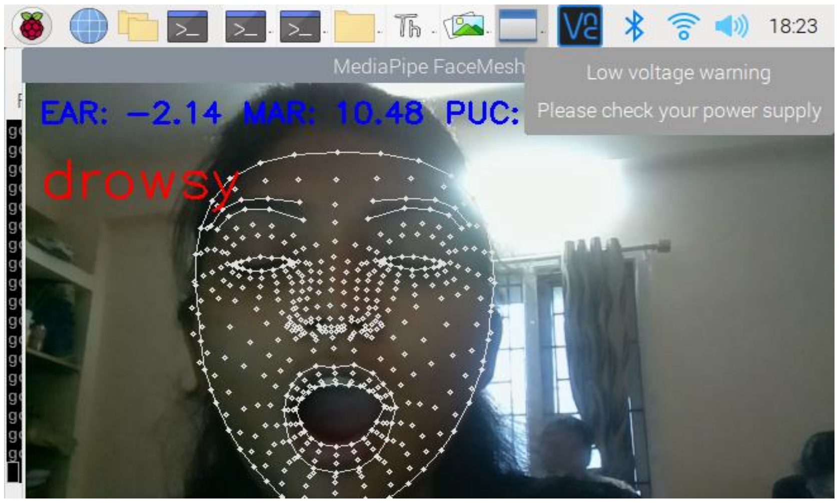


Fig 5.3.3 – Output displaying Drowsy

6.TESTING AND VALIDATION

6.1 INTRODUCTION

Software testing is critical element of software quality assurance and represents the ultimate review of specification design and coding. Software testing is one of the broader topics and often referred to as verification to all the activities to all the activities that endure the software built is traceable to use requirements. Software testing can be taken as one of the errors in the developed system. Testing is the process of executing a program with the intend of finding the errors.

GOALS AND OBJECTIVES:

The objective is to decide the tests that systematically uncover the different classes of errors and to do so with a minimum amount of time and effort. A good test is one that has high probability of finding the yet undiscovered errors. The product has been implemented by the software developer itself. Testing has been carried out according to the test plan and test procedures stated in this test specification. This document gives a general description of the test specification of the system.

SCOPE:

The test case is a document that describes an input, an action or event and an expected response to determine if a feature of an application is working correctly. So for example, the project entitled MEC Archive also requires various test cases like login test, upload test, mentor allocation test, ranking test, search test etc. Testing begins at the middle level and works toward the integration of web-based system. Testing and debugging are different activities. But any testing strategies include debugging strategies. In the test phase the testing of the developed system is done along with the system data.

TEST PLAN:

It describes the overall testing strategy and the project management issues that are required to properly execute effective tests. The plan typically contains a detailed understanding of the eventual workflow.

TEST STRATEGY: The overall strategy for software testing is described and how the testing process will take place. It is created to inform project managers, testers and developers about some key issues of the testing process.

6.2 DESIGN OF THE TESTCASES AND SCENARIOS

LEVELS OF TESTING

In order to uncover the errors, present in different phases we have the concept of levels of testing. The basic steps involved in testing are:

UNIT TESTING

Unit testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. This test is carried out during programming time itself.

INTEGRATION TESTING

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Integration Testing is the second level of testing performed after Unit Testing and before System Testing. Developers themselves or independent testers perform Integration Testing.

SYSTEM TESTING

System testing is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing. Normally, independent Testers perform System Testing.

6.3 VALIDATION:

It is a dynamic mechanism of testing and validating the actual product. It checks whether the software meets the requirements and expectations of a customer. It always involves executing the code. It can find bugs that the verification process cannot catch.

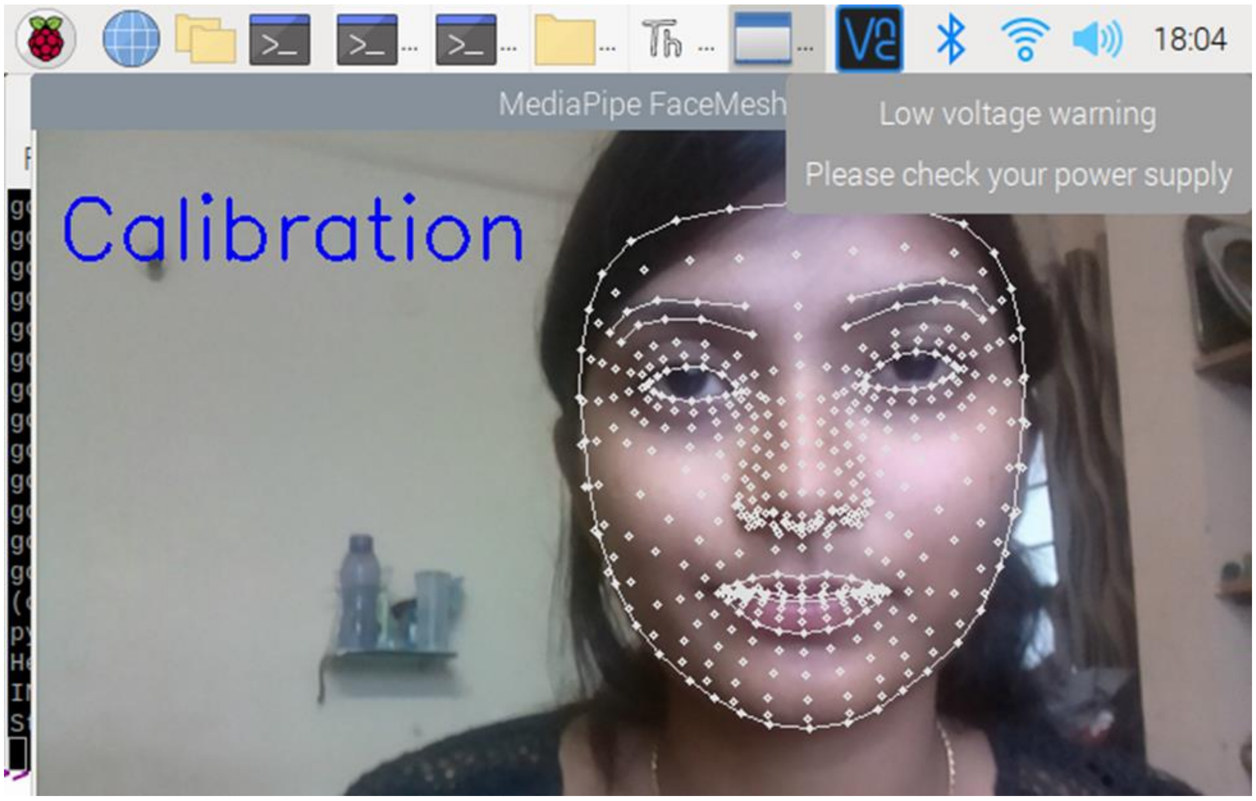
TEST CASES

Test No.	Test case	Expected output	Actual output	Result
1.	Performing calibration if a person is sitting in front of camera	Face should be detected and video captured properly over 25 frames.	Face should be detected and video captured properly over 25 frames.	Success
2.	Calibration is not performing if person is not in front of camera	Face should not be detected and calibration process is continued	Face should be detected and calibration process is continued	Success
3.	Detecting a drowsy person when the person is wearing glasses.	Drowsiness of the person is detected While wearing glasses	Drowsiness of the person is detected While wearing glasses	Success
4.	Detecting an Alert person when the	Alertness of the person is detected	Alertness of the person is detected	Success

	person is Alert.			
5.	When the face of drowsy driver is positioned center infront of camera	Drowsiness of the person is detected	Drowsiness of the person is detected	Success
6.	When drowsy driver positioned infront of camera in Ambient lightning.	Drowsiness of the person is detected in ambient lightning	Drowsiness of the person is detected in ambient lightning	Success
7.	When Alert driver positioned infront of camera in Ambient lightning.	Alertness of the person is detected in ambient lightning	Alertness of the person is detected in ambient lightning	Success

<u>Testcase 1:</u>

Performing calibration if a person is sitting in front of camera



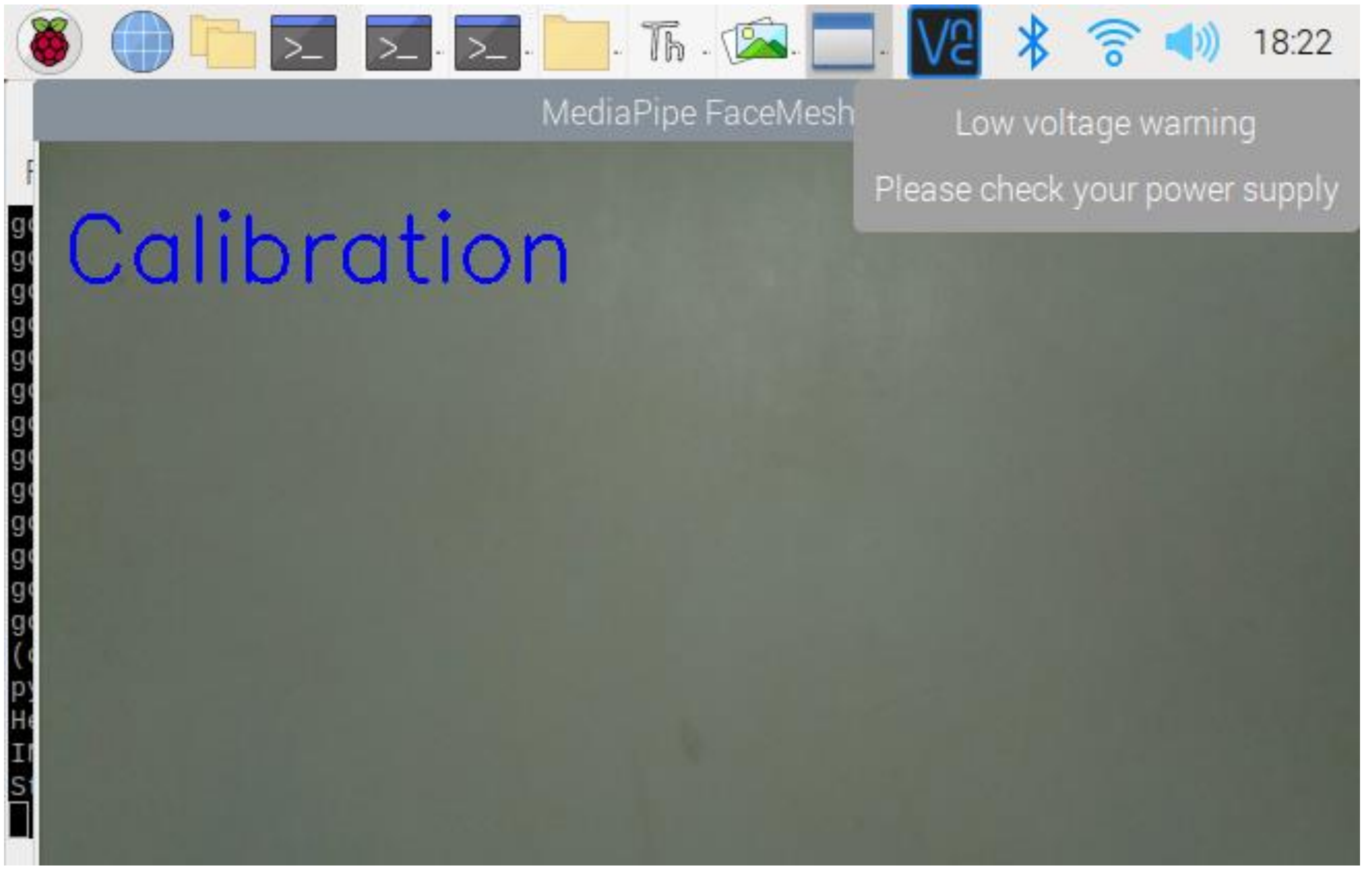
Low voltage warning

Please check your power supply

Figure 6.3.1: Performing Calibration

Test case 2:

Calibration is not performed if person is not in front of camera



Low voltage warning

Please check your power supply

Figure 6.3.2: Not Performing Calibration when person not detected

Test case 3:

Drowisness was sucuccesfully detected when the person is wearing spectacles.

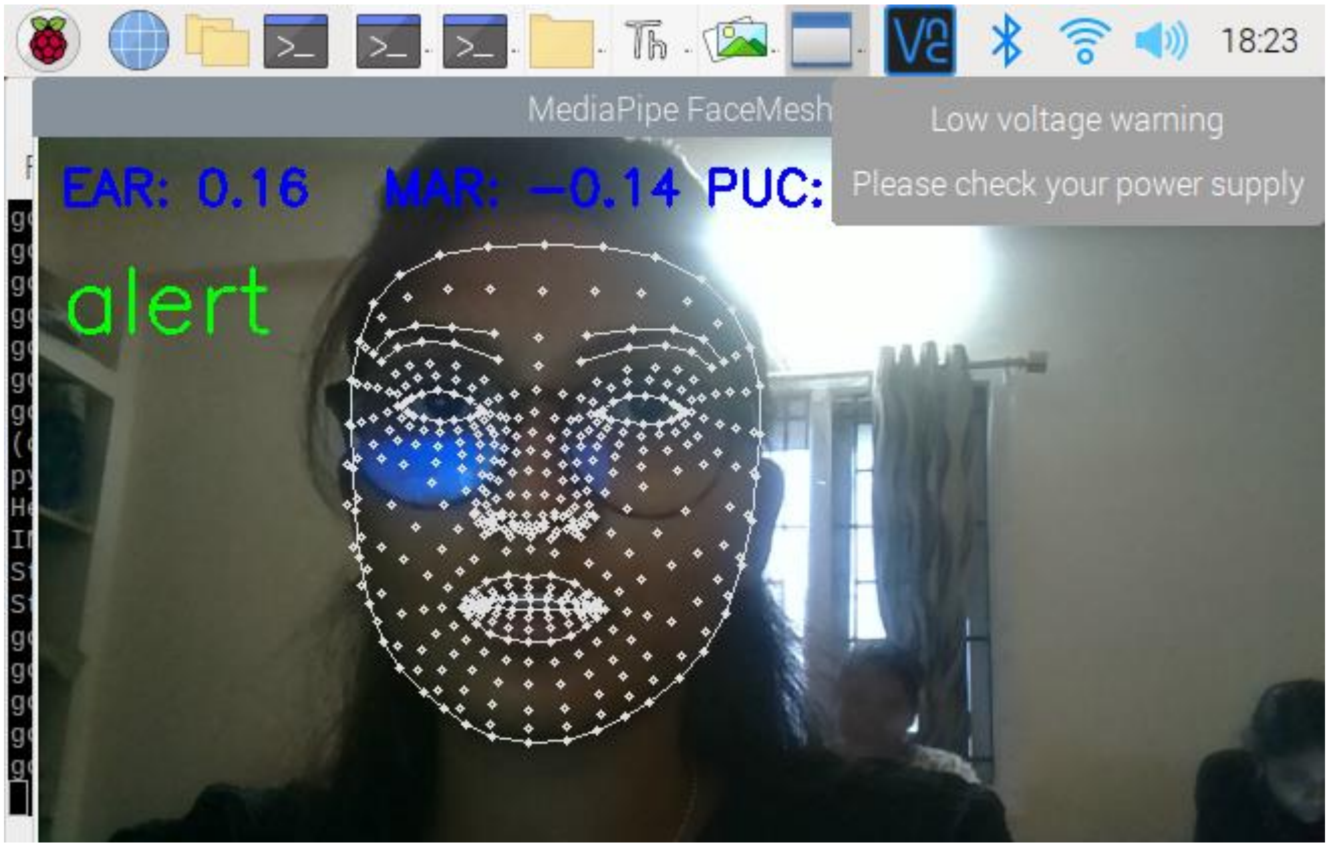


Figure 6.3.3: Not Performing Calibration when person not detected

Test case 4:

Alertness of the person is detected when person is Alert.

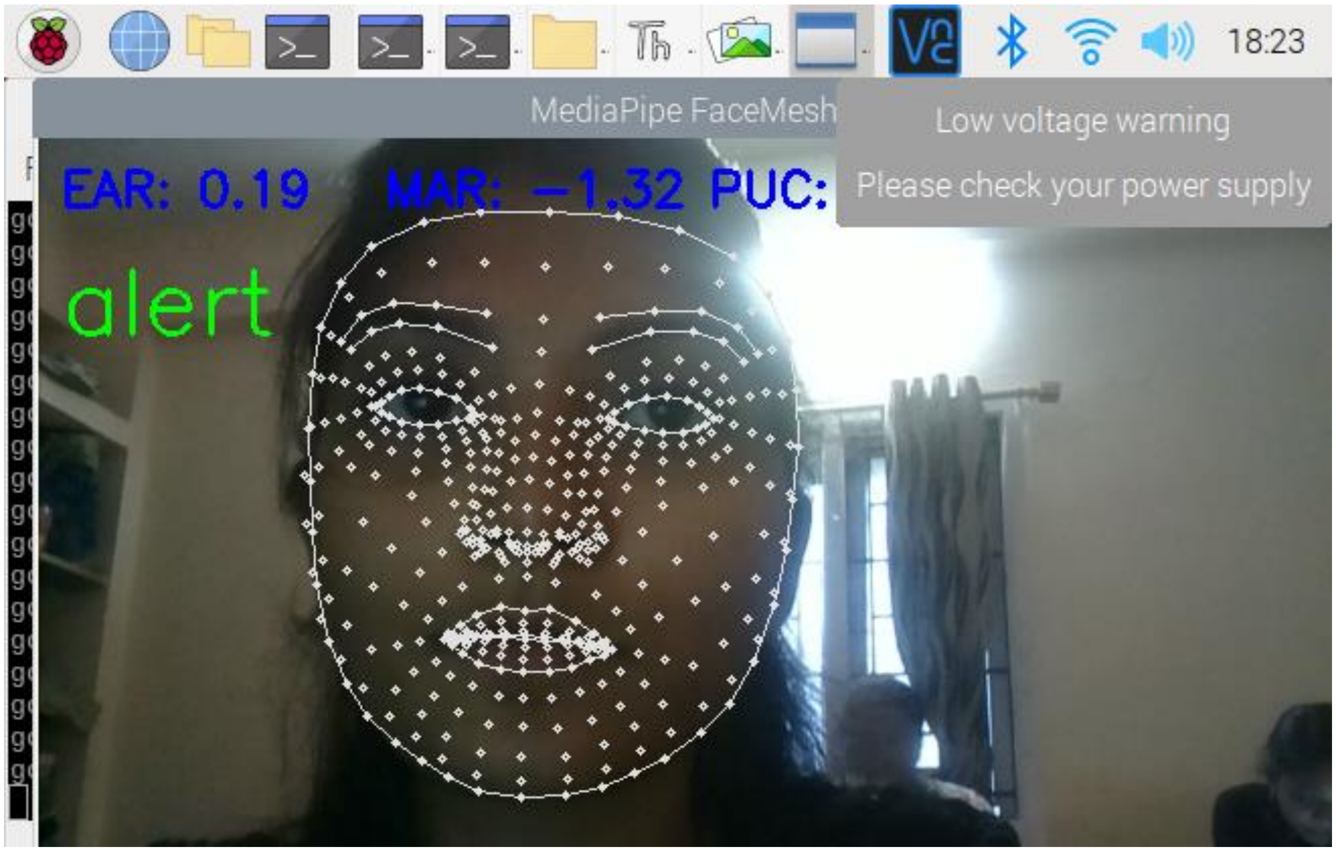


Figure 6.3.4: Displaying alert status when person is Alert

<u>Test case 5:</u>

The face position was centerd and drowisness was succuesfully detected.

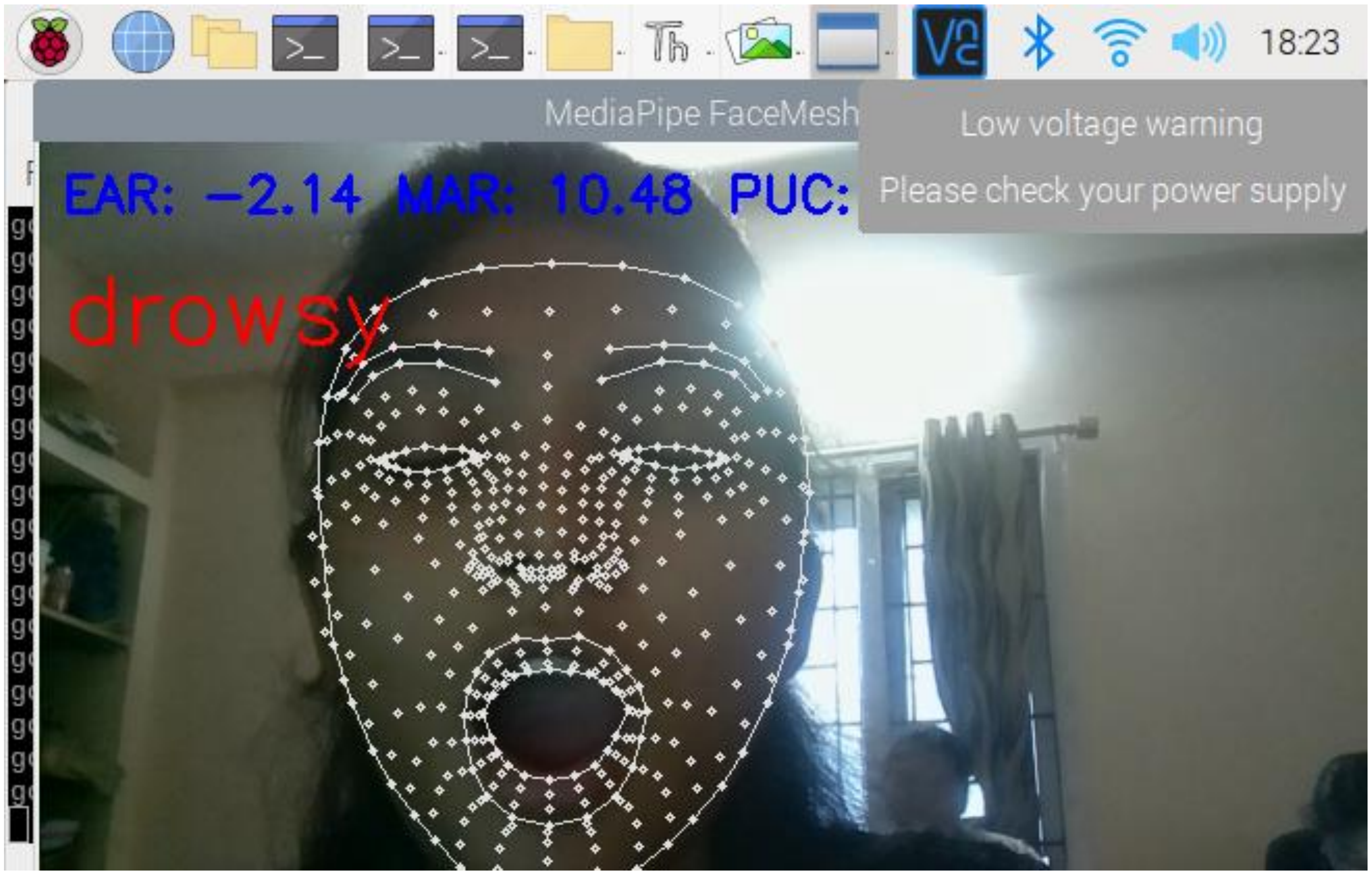
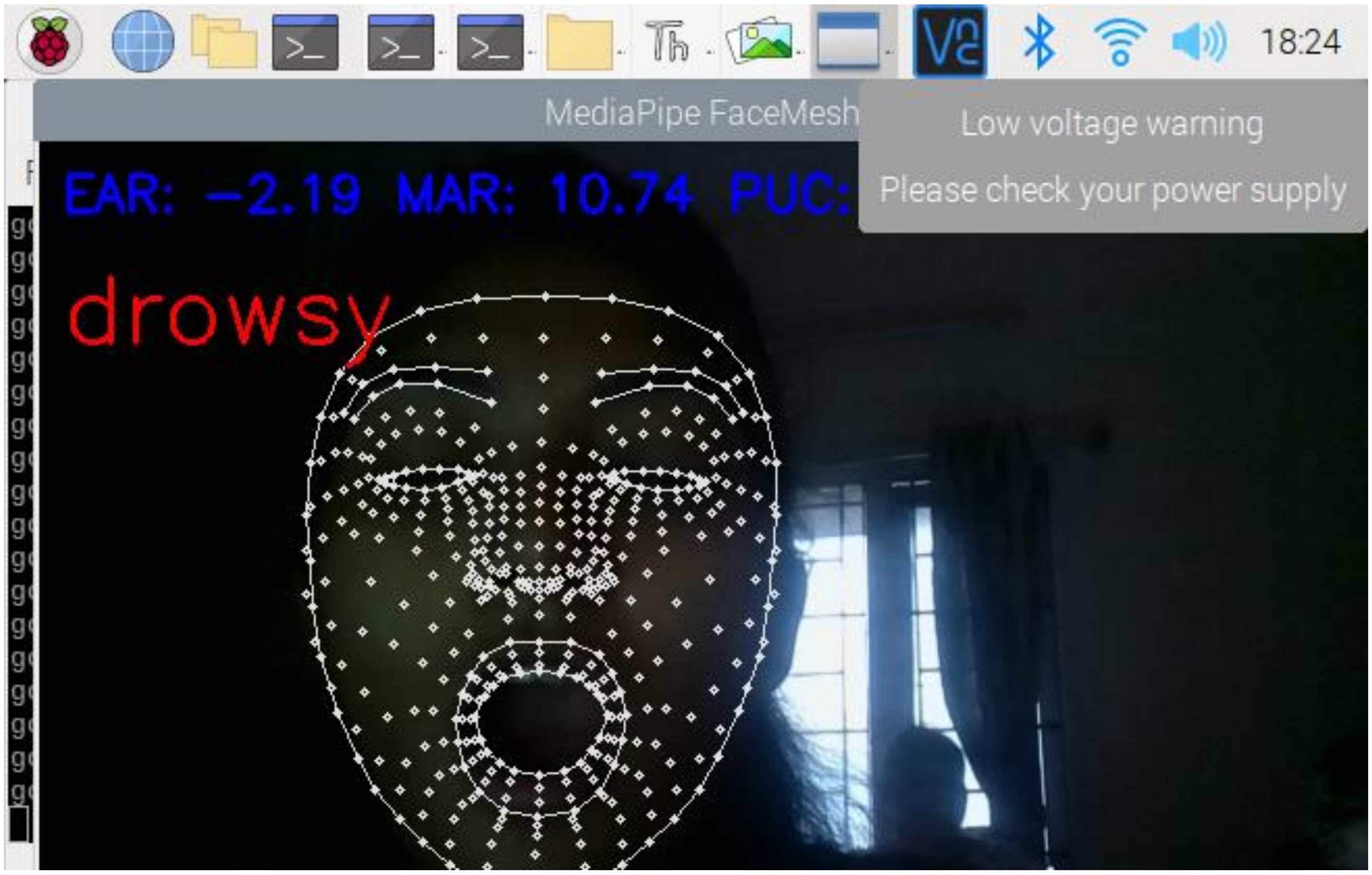


Figure 6.3.5: Displaying drowsy status when person is drowsy

Test case 6:

When there is a ambient amount of light the person face and state is succesfully detected**.** The person is drowsy



EAR: -2.19 MAR: 10.74 PUC:
drowsy

Figure 6.3.6: Displaying drowsy status when person is drowsy in Ambient Lightning

Test case 7:

When there is a ambient amount of light the person face and state is succesfully detected. Person is in alert state

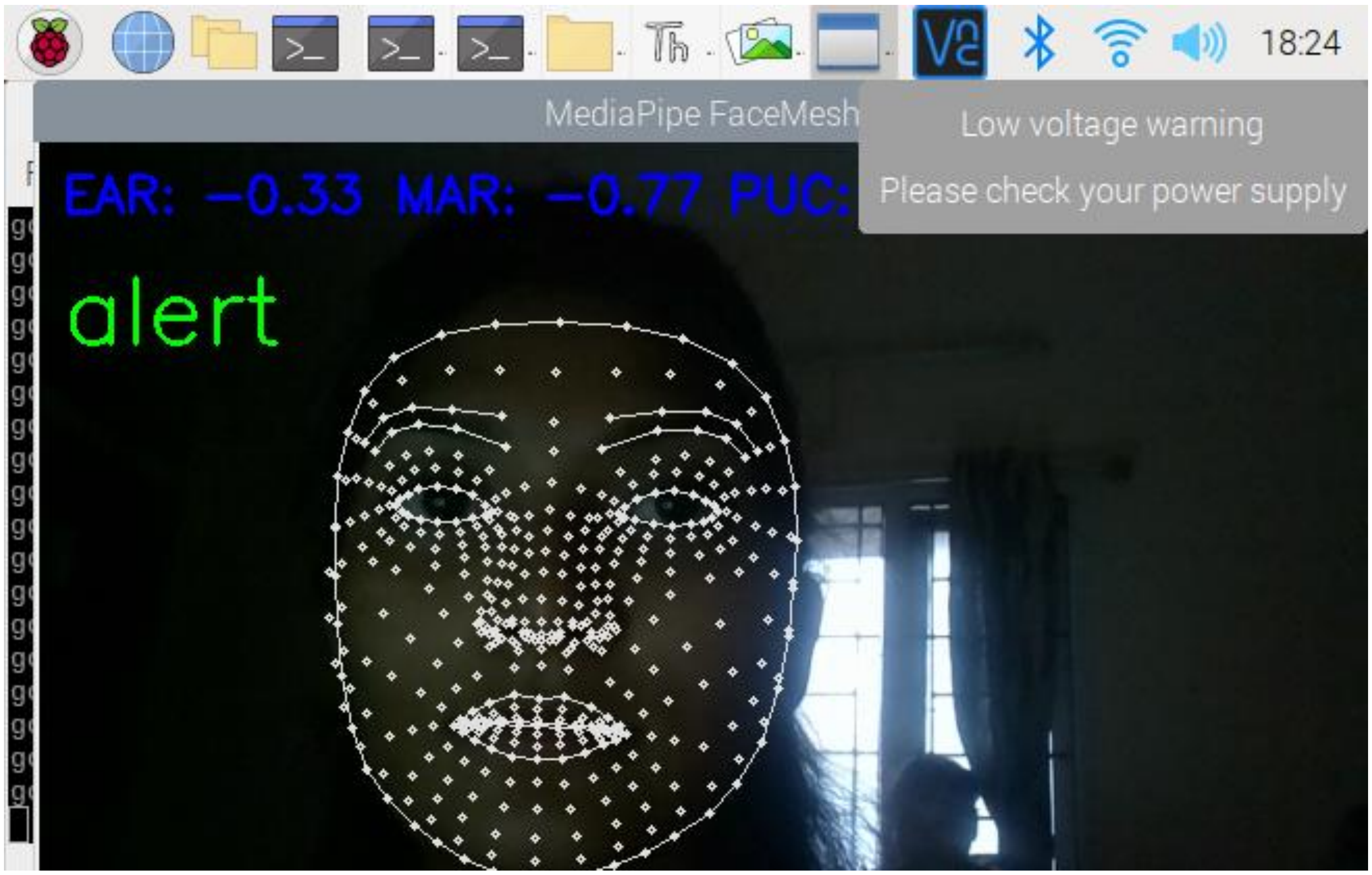


Figure 6.3.7: Displaying Alert status when person is Alert in Ambient lightning

6.4 CONCLUSION

This chapter is introduced with explanation of testing and validation in software development life cycle in general. Design of test cases and different scenarios is given in detail along with screen shorts, followed by validations of generated test cases and scenarios.

7.CONCLUSION AND FUTURE SCOPE

A prototype system is implemented using LSTM model. The purpose of our project is to help solving real life problem in very cost effective way. It alerts the driver Whenever the driver feels drowsy and closes his eyes for more than a few seconds, the buzzer is blown. As a result, it alerts the driver. It also warns the passengers to allow the driver some rest or halt the vehicle. As a result, the accident ratio decreases. Hence, our project if commercially developed will help in saving the precious life of driver & passengers.

SCOPE FOR FUTURE WORK

The future works may focus on the vehicle’s location ,if any serious collision occurs with a drowsy driver.It is the main component to detect the latitude and longitude of any area on the Earth with date and time from satellite .In this system, the mishap locatio is easily traced and the location is forwarded using GPS.

8.REFERENCES

1. Driver Fatigue Detection Based on Facial Key Points and LSTM, Long Chen,[1] Guojiang Xin,[2] Yuling Liu,[1] and Junwei Huang[3]

   https://www.hindawi.com/journals/scn/2021/5383573/

2. 2020 IEEE 17th India Council International Conference (INDICON) | 978-1-7281-6916-3/20/$31.00 ©2020 IEEE | DOI:10.1109/INDICON49873.2020.9342348

   https://sci-hub.hkvisa.net/10.1109/indicon49873.2020.9342348

3. Using long short term memory and convolutional neural networks for driver drowsiness detection, Azhar Quddas , Ali Shahadi Zhandi , Laura Prest , Felix J E Comeau.

   https://pubmed.ncbi.nlm.nih.gov/33848710/

4. Driver Sleepiness Detection Using LSTM Neural Network Yini Deng1, Yingying Jiao1, and Bao-Liang Lu1,2,3(B).

   https://bcmi.sjtu.edu.cn/home/blu/papers/2018/Deng2018DriverSleepiness.pdf

5. Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images Elena Magán , M. Paz Sesmero * , Juan Manuel Alonso-Weber and Araceli Sanci

   https://www.mdpi.com/2076-3417/12/3/1145/pdf

6. Long Short-term Memory Article in Neural Computation · December 1997 DOI: 10.1162/neco.1997.9.8.1735 · Source: PubMed

   https://www.researchgate.net/profile/Sepp-Hochreiter/publication/13853244_Long_Short-term_Memory/links/5700e75608aea6b7746a0624/Long-Short-term-Memory.pdf

7. Wikipedia of Long Short Term Memory

   https://en.wikipedia.org/wiki/Long_short-term_memory

8. Attention Mesh: High-fidelity Face Mesh Prediction in Real-time, Ivan Grishchenko, Artsiom Ablavatski, Yury Kartynnik, Karthik Raveendran, Matthias Grundmann

   https://arxiv.org/pdf/2006.10962.pdf

9. Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio, Sukrit Mehta, Sharad Dadhich, Sahil Gumber https://www.researchgate.net/publication/332052055_RealTime_Driver_Drowsiness_Detection_System_Using_Eye_Aspect_Ratio_and_Eye_Closure_Ratio

10. S. Tereza and C. Jan, "Real-Time Eye Blink Detection using Facial Landmarks," 21st Computer Vision Winter Workshop Luka ˇCehovin, Rok Mandeljc, Vitomir ˇStruc (eds.) Rimske Toplice, 2016.

https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

11. Real-time system for monitoring driver vigilance, L.M. Bergasa; J. Nuevo; M.A. Sotelo; M. Vazquez
https://ieeexplore.ieee.org/document/1336359
12. Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In Proceedings of 36th Internaional Conference and Exhibition on Computer Graphics and Interactive Techniques, pages 187–194, 1999.
https://www.face-rec.org/algorithms/3d_morph/morphmod2.pdf
13. Rasberry pi 3b+ Installation and working.
https://pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/
14. J. Cech, V. Franc, and J. Matas. A 3D approach to facial landmarks: Detection, refinement, and tracking. In Proc. International Conference on Pattern Recognition, 2014.
http://cmp.felk.cvut.cz/ftp/articles/cech/Cech-ICPR-2014.pdf
15. T. Danisman, I. Bilasco, C. Djeraba, and N. Ihaddadene. Drowsy driver detection system using eye blink patterns. In Machine and Web Intelligence (ICMWI), Oct 2010. 1, 6, 7
https://www.researchgate.net/publication/251970873_Drowsy_driver_detection_system_using_eye_blink_patterns
16. D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models," in Proc. BMVC, 2006, pp. 929–938.
http://www.bmva.org/bmvc/2006/papers/024.pdf
17. Automatic Assessment of Eye Blinking Patterns through Statistical Shape Model, Federico M. Sukno, Sri-Kaushik Pavani, Constantine Butakoff & Alejandro F. Frangi

https://link.springer.com/chapter/10.1007/978-3-642-04667-4_4
18. Raspberry Pi as Internet of Things hardware: Performances and Constraints
https://www.researchgate.net/publication/272175660_Raspberry_Pi_as_Internet_of_Things_hardware_Performances_and_Constraints.
19. Driver Drowsiness Alert System with Effective Feature Extraction Ashlesha Singh , Chandrakant Chandewar2 , and Pranav Pattarkine
https://ijrest.net/downloads/volume-5/issue-4/pid-ijrest-54201808.pdf
20. Karamjeet Singh,Rupinder Kaur,‖Physical and Physiological Drowsiness Detection Methods‖, IJIEASR, pp.35-43,vol.2,2013.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.4863&rep=rep1&type=pdf

21. Driver Drowsiness Detection Methods: A Comprehensive survey D. Venkata Subbaiah(Research Scholar) , Prof. Prasad Reddy P.V.G.D , Prof.K. Venkata Rao http://www.ijrat.org/downloads/Vol-7/march-2019/Paper%20ID-73201918.pdf
22. Nur Fatin Izzati y., M.M.Ibrahim, N.A.Manap, nur Shazwani A., ― Analysis of eye closure duration based on height of Iris ―, 2016 6th International Conference on control system, Malaysia. https://www.researchgate.net/publication/316451292_Analysis_of_Eye_Closure_Duration_Based_on_the_Height_of_Iris
23. B. Mandal, L. Li, G. S. Wang, and J. Lin, ―Towards Detection of Bus Driver Fatigue Based on Robust Visual Analysis of Eye State,‖ IEEE Trans. Intell. Transp. Syst., vol. PP, no. 99, pp. 1– 13, 2016. http://kresttechnology.com/krest-academic-projects/krest-mtech-projects/ECE/M%20Tech-ECE%20EMBEDDED%20201718/MTECH%20IEEE%20ECE%20EMB%20BASE%20PAPER/37.%20Towards%20Detection%20of%20Bus%20Driver%20Fatigue%20Based%20on%20Robust%20Visual%20Analysis%20of%20Eye%20State.pdf
24. J. May and C. Baldwin, "Driver fatigue: The importance of identifying causal factors of fatigue when considering detection and countermeasure technologies," Transp. Res. F, Traffic Psychol. https://www.researchgate.net/publication/222915010_Driver_fatigue_The_importance_of_identifying_causal_factors_of_fatigue_when_considering_detection_and_countermeasure_technologies
25. S. Lal and A. Craig, "A critical review of the psychophysiology of driver fatigue," Biol. Psychol., vol. 55, no. 3, pp. 173–194, 2001. https://pubmed.ncbi.nlm.nih.gov/11240213/