VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report
on

Driver Drowsiness detection

Submitted in partial fulfillment of the requirements for the VII Semester of degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

Submitted By

Apoorva Kashi
1RN18IS023

Leena Chandra
1RN18IS063

Under the Guidance of

Ms. Sudha V
Associate Professor
Department of ISE

ESTD:2001
An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098

2021-2022

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Internship work entitled ***Driver Drowsiness Detection*** has been successfully completed by **Apoorva Kashi (1RN18IS023) and Leena Chandra (1RN18IS063)** Bonafede students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements of 8th semester for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2021-2022.** The internship report has been approved as it satisfies the academic requirements in respect of internship work for the said degree.

Mr. R Rajkumar Internship Guide Associate Professor Department of ISE	**Dr. Suresh L** Professor and HoD Department of ISE RNSIT	**Dr. M K Venkatesha** Principal RNSIT

External Viva

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

DECLARATION

I, **Apoorva Kashi [USN: 1RN18IS023] and Leena Chandra [USN: 1RN18IS023]** student of VII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***Driver Drowsiness Detection*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2021-2022.

Place: Bengaluru
Date: 10.01.2021

Apoorva Kashi (1RN18IS023)

Leena Chandra (1RN18IS063)

ABSTRACT

Nowadays, more and more professions require long-term concentration. Drivers must keep a close eye on the road, so they can react to sudden events immediately. Driver fatigue often becomes a direct cause of many traffic accidents. Therefore, there is a need to develop the systems that will detect and notify a driver of her/him bad psychophysical condition, which could significantly reduce the number of fatigue-related car accidents. However, the development of such systems encounters many difficulties related to fast and proper recognition of a driver's fatigue symptoms. One of the technical possibilities to implement driver drowsiness detection systems is to use the vision-based approach. The technical aspects of using the vision system to detect a driver drowsiness are also discussed. Drowsiness and Fatigue of drivers are amongst the significant causes of road accidents. Every year, they increase the amounts of deaths and fatalities injuries globally.

When a driver doesn't get proper rest, they fall asleep while driving and this leads to fatal accidents. This particular issue demands a solution in the form of a system that is capable of detecting drowsiness and to take necessary actions to avoid accidents.

This report explains the final project, driver drowsiness detection system. The detection is achieved with three main steps, it begins with face detection and facial feature detection using the famous Viola Jones algorithm followed by eye tracking. By the use of correlation coefficient template matching, the eyes are tracked. Whether the driver is awake or asleep is identified by matching the extracted eye image with the externally fed template (open eyes and closed eyes) based on eyes opening and eyes closing, blinking is recognized as well as yawns.

ACKNOWLEDGMENT

At the very onset we would like to place our gratefulness to all those people who helped me in making the Internship a successful one.

Coming up, this internship to be a success was not easy. Apart from the sheer effort, the enlightenment of the very experienced teachers also plays a paramount role because it is, they who guided me in the right direction.

First of all, I would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of internship work.

In this regard, I express sincere gratitude to our beloved Principal **Dr. M K Venkatesha,** for providing us all the facilities.

We are extremely grateful to our own and beloved Professor and Head of Department of Information science and Engineering, **Dr. Suresh L**, for having accepted to patronize me in the right direction with all his wisdom.

We place our heartfelt thanks to **Ms. Sudha V**, Associate Professor, Department of Information Science and Engineering for having guided internship and all the staff members of the department of Information Science and Engineering for helping at all times.

I thank **Mr. Deepak Garg, CEO, NASTECH**, for providing the opportunity to be a part of the Internship program and having guided me to complete the same successfully.

I also thank our internship coordinator **Dr. R Rajkumar,** Associate Professor, Department of Information Science and Engineering. I would thank my friends for having supported me with all their strength and might. Last but not the least, I thank my parents for supporting and encouraging me throughout. I have made an honest effort in this assignment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES AND GRAPHS

ABBREVIATIONS

Acronym	Description
CNN	Convolution Neural Networks
RBFNN	Radial Basic Functional Network
IE	Internet Explorer
ISO	International Organization of Standardization
ANSI	American National Standard Institutes

CHAPTER 1

1.INTRODUCTION

1.1 BACKGROUND

Machine learning is an Associate in drowsiness detection of computer science (AI) that gives systems the flexibility to mechanically learn and improve from expertise while not being expressively programmed. The most probable difference between Python and programming is largely in programming we have a tendency to use conditional statements to expressively tell the program to figure victimization those conditions whereas, Python mechanically learns and improves from expertise primarily the user ought to train the dataset and also the machine language learns mechanically by means independently. Python focuses on the event of pc programs which will access knowledge and use it to learn for themselves. The process of learning begins with observations or knowledge, like examples, direct expertise, or instruction, so as to see patterns in knowledge and create higher choices in the future that support the examples that we offer. This type of process of learning is predominantly classified as supervised learning. The main objective of this type of learning is that the system already knows its output; it's just the system structures itself towards the output in this type of learning that also shows that the system learns even through feedback too. The essential point is to permit the PCs adapt naturally without human intercession or help and change activities likewise. Python algorithms are classified as:

1. Supervised ML Algorithms these algorithms can utilize what has been realized in the past to the new information being utilized by utilizing hailed models which will be useful in future scopes. Beginning with preparing the dataset while preparing the dataset the AI calculation sets up hindrances like fundamental capacities to make expectation of the yield esteems for future scopes by setting up these impediments the calculation consequently. takes in and develops itself from these obstructions. This sort of Learning needs an adequate measure of preparing to anticipate yields and to give yields to the particular new sources of info. This sort of learning calculation contrasts it's yield and the ideal yield and redresses itself and learns without anyone else by discover blunders and changing them.

2. Unsupervised Python Algorithms square measure utterly completely different from supervised Python algorithms; these algorithms square measure used once the dataset to coach is neither classified nor a flagged. So, the dataset ought to be cleansed before coaching it. Knowledge clean-up is largely classifying and labeling the dataset. During this sort the system doesn't turn out the proper output however investigation {the knowledge info the information} sets and attracts conclusions on the idea of the data clean up done before coaching the dataset and classifying and tired examples and describe the hidden structures from the dataset. Unsupervised learning is the coaching of machine mistreatment info that's neither classified nor labeled and permitting the rule to act on it info while not steerage. Here the task of the machine is to cluster unsorted info per similarities, patterns and variations with no previous coaching of knowledge. The biggest distinction between supervised and unsupervised algorithms is that supervised learning is well-versed steerage whereas unsupervised learning isn't.

3. Semi-administered Learning falls among regulated and solo learning. This type of learning uses both labelled and unlabelled data. Typically, larger proportionate of unlabelled data. Semi supervised learning requires both the types of data as when the labelled data requires more resources which are totally dependent on unlabelled data. 4. Reinforcement Learning is a type of dynamic learning that trains the algorithm using a system of reward and punishment that interacts with its environment to learn. This is inspired behavioural psychology. It is similar to supervised learning but as in supervised learning it is explicitly told how to perform a task and what the desired outcome will be. As in this type of learning it works through the problem on its own and finds a way to perform the task and get the desired result.

IMAGE PROCESSING: Computer vision, the field concerning machines being able to understand images and videos, is one of the hottest topics in the tech industry. Robotics, self-driving cars, and facial recognition all rely on computer vision to work. At the core of computer vision is *image recognition*, the task of recognizing what an image represents. Image processing is a way to convert an image to a digital aspect and perform certain functions on it, in order to get an enhanced image or extract other useful information from it. It is a type of signal time when the input is an image, such as a video frame or image and output can be an image or features associated with that image.

Usually, the Image Processing system includes treating images as two equal symbols while using the set methods used. Image processing basically involves the following three steps.

1. Importing an image with an optical scanner or digital photography.
2. Analysis and image management including data compression and image enhancement and visual detection patterns such as satellite imagery. It produces the final stage where the result can be changed to an image or report based on image analysis.
3. Image processing is a way by which an individual can enhance the quality of an image or gather alerting insights from an image and feed it to an algorithm to predict the later things.

The following libraries are involved in performing Image processing in python;

- Scikit-image
- OpenCV
- SciPy
- Pillow
- Matplotlib

1.2 EXISTING SYSTEM

Images or the real time video is captured from the camera installed in front of the driver's face. This video is converted into number of frames. OpenCV face OpenCV-classifier is loaded. Each frame is compared with the pre-defined features of the OpenCV-classifiers. When the features are matched the face is detected and a rectangle is drawn around the face. Using feature extraction we estimate the position of the eyes. By comparing with the OpenCV eye-OpenCV classifier, the eyes are detected, and rectangles are drawn around left and right eye. Manu B.N in 2016, has proposed a method that detect the face using OpenCV feature-based cascade classifiers.

Amna Rahman in 2015, has proposed a method to detect the drowsiness by using Eye state detection with Eye blinking strategy. In this method first, the image is converted to gray scale and the corners are detected using Harris corner detection algorithm which

will detect the corner at both side and at down curve of eye lid. After tracing the points then it will make a straight line between the upper two points and locates the mid-point by calculation of the line, and it connects the mid-point with the lower point. Now for each image it will perform the same procedure and it calculates the distance ‘d’ from the mid-point to the lower point to determine the eye state.

Finally, the decision for the eye state is made based on distance ’d’ calculated. If the distance is zero or is close to zero, the eye state is classified as “closed” otherwise the eye state is identified as “open”. They have also invoked intervals or time to know that the person is feeling drowsy or not. This is done by the average blink duration of a person is 100-400 milliseconds (i.e. 0.1-0.4 of a second). In-vehicle camera is commonly installed to realize the possible reasons of car accidents. Such a camera can also be used to detect the fatigue of the driver.

1.3 PROPOSED SYSTEM

Real Time Drowsiness behaviors which are related to fatigue are in the form of eye closing. head nodding or the brain activity. Hence, we can either measure change in physiological signals, such as brain waves, heart rate and blinking to monitor drowsiness or consider physical changes such as sagging leaning of driver's head and open/closed state of eyes The former technique, while more accurate, is not realistic since highly sensitive electrodes would have to be attached directly on the driver body and hence which can be annoying and distracting to the driver.

In addition, long time working would result in perspiration on the sensors, diminishing their ability to monitor accurately. The second technique is to measure physical changes i.e. open/closed eyes to detect fatigue is well suited for real world conditions since it is non-intrusive by using a video camera to detect changes in addition, micro sleeps that are short period of sleeps lasting 2 to 3 minutes are good indicators of fatigue. Thus, by continuously monitoring the eyes of the driver one can detect the sleepy state of over and a timely warning is issued.

The algorithm starts with the detection of heads on color pictures using deviations in color and structure of the human face and that of the background. By normalizing the

distance and position of the reference points, all faces should be transformed into the same size and position.

For normalization, eyes serve as point reference. Other OpenCV algorithm finds the eyes on any grayscale image by searching characteristic is features of the eyes and eye sockets. Tests made on a standard database show that the algorithm works very fast and it is reliable. In proposed method, first the image is acquired by the webcam for processing. The images of the driver are captured from the camera which is installed in front of the driver on the car dashboard.

It will be passed to pre-processing which prepares the image for further processing by the system. Then we search and detect the faces in each individual frame. If no face is detected, then another frame is acquired. If a face is detected, then a region of interest in marked within the face. This region of interest contains the eyes. Defining a region of interest significantly reduces the computational requirements of the system. After that the eyes are detected from the region of interest. Then the image is classified into one of four classes and a rectangle is drawn around the region with it's label to detect drowsiness.

CHAPTER 2

2.LITERATURE SURVEY

2.1. METHODOLOGIES

In computer science, image processing is the use of computer algorithms to perform image processing on images. As a subcategory or field of digital signal processing, image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the buildup of noise and signal distortion during processing. Since images are defined over two dimensions digital image processing may be modelled in the form of multidimensional system. There are different methodologies to identify drowsiness state of the driver. They can be categorized into the following three main categories:

1. Behavioral Parameters Based: Measuring the driver's fatigue without using non-invasive instruments comes under this category. Analyzing the behavior of the driver based on his/her eye closure ratio, blink frequency, yawning, position of the head and facial expressions. The current parameter used in this system is the eye-closure ratio of the driver.

2. Vehicular Parameters Based: Measuring the fatigue nature of the driver through vehicle driving patterns comes under this category. These parameters include lane changing patterns, steering wheel angle, steering wheel grip force, vehicle speed variability and many more.

3. Psychological Parameters Based: Measuring the drowsiness of the driver based on the physical conditions of the driver fall under this category. Such parameters may be respiration rate, heart-beat rate, body temperature and many more. Among other various approaches, these physiological parameters provide the most accurate results since they are based on the biological nature of the driver.

DROWSINESS DETECTION THROUGH REGION OF INTEREST: Region of interest (ROI) can detect a driver's face. As can be seen in the blue rectangle is the region of interest. The way to create an ROI area is to first obtain the green rectangle around the

the Haar Cascade Classifier in the first frame, which includes height, width. Then, the rectangle is scaled up to create region of interest. There are several steps to calculate the ROI area and we have to calculate ROI for each and every region of interest.
Disadvantages : It is uses extra frames or squares to detect face detection. It can't find in low light. There is no reason to use again region of interest while Haar cascade classifier can do the same process. It can't detect while using glasses in driving.

DETECTION OF DROWSINESS THROUGH LBPH: In this algorithm the faces are detected by using local binary patterns histograms (LBPH). The first computational step in lbph is to create an intermediate image that describes the original image in a binary format. The image is converted into matrix form, and we need to take a central value of the matrix to be used as and threshold value. This value is used to define neighbouring values which can be set to to either 0 or 1. The values which are 1 in the matrix form are to be considered and the remaining values are discarded. The values represent each pixel. Through this the region of face can be detected.
Disadvantages: It produces less accurate results. The computational time is high. This will work only if the data samples are less.

MOUTH AND YAWNING ANALYSIS: Fatigue is the major reason for road accidents. To avoid the issue, Sarada Devi and Bajaj proposed the driver fatigue detection system based on mouth and yawning analysis. Firstly, the system locates and tracks the mouth of a driver using cascade of classifier training and mouth detection from the input images. Then, the images of mouth and yawning are trained using SVM. In the end, SVM is used to classify the regions of mouth to detects the yawning and alerts the fatigue. For experiment, authors collect some videos and select 20 yawning images and more than 100 normal videos as dataset. The results show that the proposed system gives better results as compared to the system using geometric features. The proposed system detects yawning, alerts the fatigue earlier and facilitates to make the driver safe.

REAL TIME ANALYSIS USING EYE AND YAWNING Kumar proposed the real time analysis of Driver Fatigue Detection using behavioral measures and gestures like eye blink, head movement and yawning to identify the drivers' state. The basic purpose of the proposed method is to detect the close eye and open mouth simultaneously and

generates an alarm on positive detection. The system firstly captures the real time video using the camera mounted in front of the driver. Then the frames of captured video are used to detect the face and eyes by applying the viola-jones method, with the training set of face and eyes provided in OpenCV. Small rectangle is drawn around the center of eye and matrix is created that shows that the Region of Interest (ROI) that is eyes used in the next step. Since the both eyes blink at the same time that's why only the right eye is examined to detect the close eye state. If the eye is closed for certain amount of time it will be considered as closed eye. To determine the eye state, firstly the eye ball color is acquired by sampling the RGB components on the center of eye pixel.

Then the absolute thresholding is done on the eye ROI based on eye ball color and intensity map is obtained on Y-axis that show the distribution of pixels on y-axis which gives the height of eye ball and compared that value with threshold value which is 4 to distinguish the open and close eye. After that, if the eye blink is detected in each frame it will be considered as 1 and stored in the buffer and after the 100 frames, eye blinking rate is calculated. Then to detect the yawning motion of the mouth, contour finding algorithm is used to measure the size of mouth. If the height is greater than the certain threshold. It means person is taking yawning. To evaluate the performance of the proposed system, system has been measured under different conditions like persons with glasses, without glasses, with moustache and without moustache for 20 days in different timings. The system performs best when the drivers are without glasses.

M.A. Assari & M. Rahmati [1] proposed a system in which the drowsiness of the driver is detected by detecting the face through horizontal projection on the image and tracking the face components via template matching technique which comprised of eyebrows and eyes along with mouth. The proposed method has been implemented in simulation environment of MATLAB (Simulink). Addition of the IR lighting as sources of light helped in better detection of faces in this system.

According to (Machaca Et al, 2018) [12] the image acquired from the camera is filtered to reduce noise and then adjust contrast following this, the face is detected and then using ROI eyes are isolated and the eyes are tracked and by comparing the eyes with two externally fed templates eye opening and closing pattern is recognized which is used to calculate the rate of eye blinking, if the eye blinking rate is higher than the threshold

frequency then the driver is classified as not fit to drive. The entire process is carried out with MATLAB.

Tianyi Hong et al [2] presented a system which used face detection method basing on the cascade of classifiers trained through Adaboost technique. Optimization in this system is performed by applying the integral image of the original image to develop a canny filter for cascade processing and improve the performance. Integrated performance primitives(IPP) have been used for better and faster computational results. This system is validated in GENE-8310 embedded platform.

B. Warwick et al [3] proposed a system that is based on physiological approach in which the driver wears a wireless biosensor called BioHarness, a wearable device capable of collecting the physiological data and then transmitting to a smartphone. This data is then analysed through Fast Fourier Transform(FFT) and Power Spectral Density(PSD) which provide the desired vectored inputs that can be fed into a Neural Network. This system is run on a drowsiness detection mobile app by the researchers.

K. Dwivedi et al [4] developed a system which identifies drowsiness of the driver using representational learning. A Haar-like face detector feeds the images to a 2-layer convolutional neural network for extracted features which are then used to train a softmax layer classifier for detecting whether a driver is drowsy or not drowsy. This system was able to yield a satisfactory result of 78% accuracy in detecting the drowsiness and alerting the driver.

J.J. Yan et al [5] proposed a system in which the images captured are converted into grayscale using the Sobel operator for edge detection. The position of the eyes are calculated using template matching. To determine the states of the eyes, the binarization and quick sort techniques are used which also confirm the distribution of the black pixels in the grayscale image. In this study, P80 is taken as the important criterion of the driver's physical state. There is a threshold value to compare with pixel value If the amount of black pixels is lower than this threshold value then it is considered as the driver in drowsy state.

3. REQUIREMENT ANALYSIS, TOOLS &TECHNOLOGIES

3.1 FUNCTIONAL REQUIREMENTS

The functional requirement define the system or the components of the system. A function is basically inputs, behaviors and outputs. Stuff that can be called functional requirements are: calculations, technical details, data manipulation and processing. It tells us what a system is supposed to do. A Functional prerequisite is described as one portion or an element of a product , in the entire methodology of programming building. A capacity or part is also depicted as the lead of a section and its yields, given a great deal of data sources. A useful prerequisite may be the figuring identified with specialized and subtleties or data control and getting ready or whatever other express usefulness that describes the target of a particular structure uses the useful necessities are found being utilized cases.

a. Recording the driver's video through live capture.

b. Then recognizes changes and detects Eyes and Face to determine drowsiness

3.2 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a requirement that gives the criteria that can be used to judge how well a system can function. It comes under system/requirements engineering. It gives a judgement on the overall unlike functional requirements which define specific behavior or functions. Functional requirements are implemented by using the system design whereas system architecture is what is used for implementing the non-functional requirements. Non-functional requirements are also called constraints.

Some of the quality attributes are as follows:

Accessibility:

- Accessibility is a term that is used to describe if a product or software is accessible to the public and how easily it can be accessed.
- It is easy to access as the dataset is open source.

Maintainability:

- Maintainability tells us how easily a software or tool or system can be modified in order to correct defects and meet new requirements.
- Different programming languages can be used to make the predictive model based on the programmer's wishes. The datasets can also be modified and new data can be added. Different ML algorithms can also be used to check which algorithm will give the best result.
- As python and R are both programming languages that can adapt to new changes easily, it is easy to maintain this type of system.

Scalability:

- The system can work normally under situations such as low bandwidth and huge datasets.
- The R studio as well as Excel can take care of these data and can perform the algorithms with ease.

Portability:

- Portability is a feature which tells us about the ease at which we can reuse an existing piece of code when we move from one location or environment to some other.
- This system uses python and R programming languages and they can be executed under different operation conditions provided it meets its minimum configurations. Only system files and dependent assemblies would have to be configured in such a case.

a. Image processing is done using the captured video.

b. Image is stored in a library called OpenCV.

c. Stored image undergo various algorithm and detects if the driver's eyes are closed or open and detects the presence of a yawn.

3.3 HARDWARE REQUIREMENTS

Processor: Intel i5 or higher

RAM : 4 GB

Space on disk: Min 2 GB

Execution space Min 100m

3.4 TOOLS AND TECHNOLOGIES

3.4.1 TensorFlow.

It is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production Tensorflow computations are expressed as tensor.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

3.4.2 Machine learning

Machine learning is the kind of programming which gives computers the capability to automatically learn from data without being explicitly programmed. This means in other words that these programs change their behavior by learning hom data. Python is clearly one of the best languages for machine learning. Python does contain special libraries for machine learning namely scipy, pandas and numpy which are great for linear algebra and getting to know the kernel. methods of machine learning. The language is great to use when working with machine learning algorithms and has relatively easy syntax.

3.4.3 Python

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping in complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language including NASA, Google, YouTube, BitTorrent, etc. Python is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had a deep focus on code readability. Python language is used by the author due to his cross

platform compatibility as the main coding language for algorithms. Open CV and Dlib library is integrated in the python interpreter for using readymade optimized functions.
Python is a significant level programming language used on a very basic level for generally helpful programming. It was made in the year 1991 by Guido van Rossum. It has a structure sanity that underlines generally on points of view, for instance, code intelligibility and strikingly uses significant whitespace. It is a viable programming language and is commonly used over the globe for both little and tremendous scope programming.

Python has motorized memory on the board. It gives different programming perfect models some of which are object-arranged, fundamental, utilitarian and procedural. It has a broad and sweeping standard library.

3.4.4 OpenCV

OpenCV stands for Open Source Computer Vision. It's an Open Source 850 sensed library that includes hundreds of advanced Computer Vision algorithms that are optimized to use hardware acceleration OpenCV is commonly used for machine learning, 4 image processing, image proc, angulation, and much more. OpenCV has a modular structure. There are shared and static libraries and a CV Namespace. In short, OpenCV it ined in our application to easily load bitmap files that contain landscaping pictures and perform a blend operation between two pictures so that one picture can be seen in the background of another picture. This image manipulation is easily performed in a few lines of code using OpenCV versus other methods.

3.4.5 Sklearn

Scikit-learn is an open source software machine learning library for the Python programming language. It encompasses numerous classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to collaborate with the Python numerical and scientific libraries Numpy and Scipy. The scikit-learn which was initially called as scikits.learn was a "Google Summer of Code" project by David Cournapeau.

The name originates from the notion that it is a "Scikit" (Scipy Toolkit), a separately-developed and distributed third-party extension to Scipy. The original codebase was later rewritten by other developers.

3.4.6 Jupyter Notebook

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. Jupyter notebooks basically provides an interactive computational environment for developing Python based Data Science applications. They are formerly known as ipython notebooks. Jupyter notebooks can illustrate the analysis process step by step by arranging the stuff like code, images, text, output etc. in a step by step manner.

The Jupyter Notebook aims to support the latest versions of these browsers:

- Chrome
- Safari
- Firefox

CHAPTER 4

4. SYSTEM DESIGN

The Driver's face is continuously monitored using a video camera. In order to detect drowsiness the first step is to detect the face using a series of frame shots taken by the camera. Then the location of the eyes is detected and the retina of the eye is continuously monitored. The captured image is converted to digital signal using Open CV. To detect the yawn, only the face is considered. For open and closed eyes, face and eyes are detected. The image is then resized and converted to a numpy array with features and labels. The labels are ‘yawn’, ‘no yawn’, ‘closed’, and ‘open’. The system works well even in case of drivers wearing spectacles and under low light conditions also. During the monitoring, the system is able to decide if the eyes are opened or closed. By doing this many accidents will be reduced and it provides safe life to the driver and vehicle safety. A system for driver safety and car security is presented only in the luxurious costly cars. Using drowsiness detection system, driver safety can be implemented in normal cars also.

The term digital image processing generally refers to processing of a two dimensional picture by a digital computer. In a broader context, it implies digital processing of any two- dimensional data. A digital image is an array of real numbers represented by a finite number of bits. The principle advantage of Digital Image Processing methods is its versatility, repeatability and the preservation of original data precision.

Pixel: Pixel is the smallest element of an image. Each pixel corresponds to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255. The values of a pixel at any point correspond to the intensity of the light photons striking at that point. Each pixel stores a value proportional to the light intensity at that particular location.

This projected is designed to achieve certain goals which are:

- Driver drowsiness detection is a car safety technology which helps to save the life of the driver by preventing accidents when the driver is getting drowsy.

The main objective is to first design a system to detect driver's drowsiness by continuously detecting the state of the eyes (open or closed) and the mouth (yawn or no yawn).

- The system works in various lighting conditions.
- Traffic management can be maintained by reducing the accidents.

Once the eye aspect ratio calculated, algorithm can threshold it to determine if a person is blinking the eye aspect ratio will remain approximately constant when the eyes are open and then will rapidly approach zero during a blink, then increase again as the eye opens. The duration of blink further provide estimation of microsleep. The results are discussed in Result section and found satisfactory. The proposed algorithm focused solely on using the eye aspect ratio as a quantitative metric to determine if a person has blinked in a video.

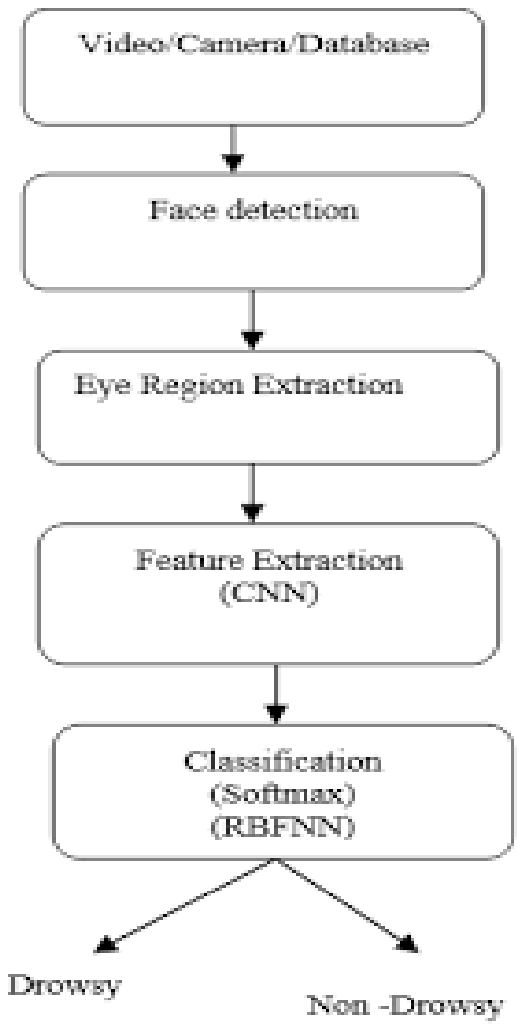


Figure 4.1 System Design Model

4.1 Model Architecture:

The model we used is built with Keras using **Convolutional Neural Networks (CNN)**. A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter.

The CNN model architecture consists of the following layers:

- Convolutional layer; 256 output channels, kernel size 3
- Convolutional layer; 128 output channels, kernel size 3
- Convolutional layer; 64 output channels, kernel size 3
- Convolutional layer; 32 output channels, kernel size 3
- Fully connected layer; 64 output nodes

The final layer is also a fully connected layer with 4 nodes. A Relu activation function is used in all the layers except the output layer in which we used Softmax.

CHAPTER 5

5. IMPLEMENTATION DETAILS

5.1 FUNCTIONAL MODULES

5.1.1 Face Detection

This module takes input from the camera and tries to detect a face in the video input. The detection of the face is achieved through the Haar classifiers mainly, the Frontal face cascade classifier. The face is detected in a rectangle format and converted to grayscale image and stored in the memory which can be used for training the model.

Face detection is accomplished by the OpenCV algorithm proposed by Paul Viola and Michael Jones in 2001. Due to the complex background, it is not a good choice to locate or detect both the eyes in the original image, for this we will take much more time on searching the whole window with poor results. So firstly, we will locate the face, and reduce the range in which we will detect both the eyes. After doing this we can improve the tracking speed and correct rate, reducing the effect of the complex background. Besides, we propose a very simple but powerful method to reduce the computing complexity.

Based on the face detection by the classifier model, each image array is appended into a numpy array along with the label to form the new data.

5.1.2 Cascade Classifier

The cascade classifier consists of number of stages, where each stage is a collection of weak learners. The weak learners are simple classifiers known as decision stumps. Boosting is used to train the classifiers. It provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

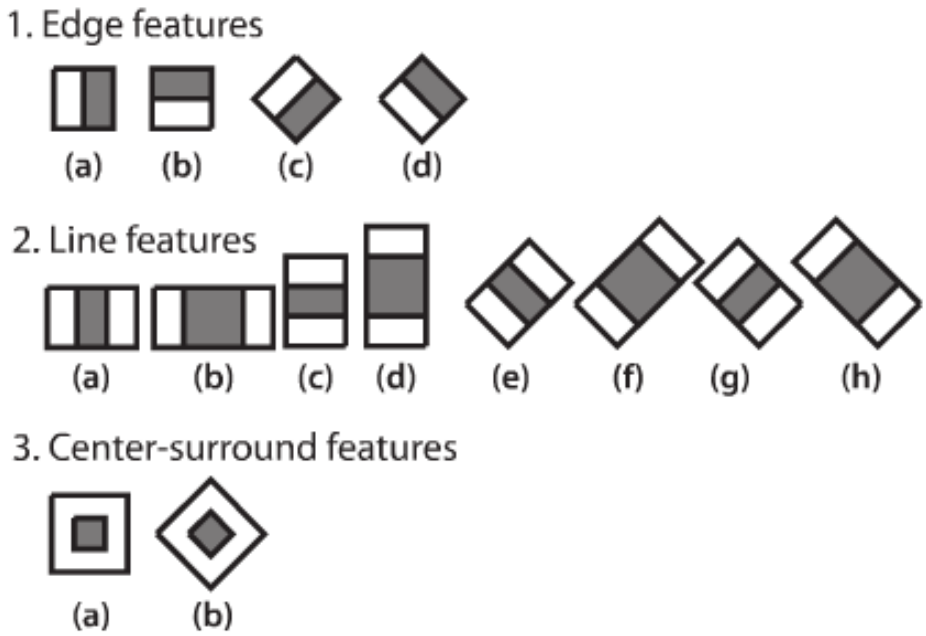


Figure5.1 Haar features

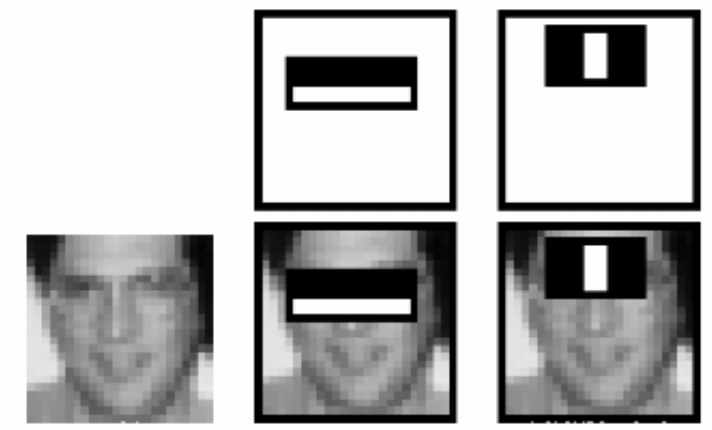


Figure 5.2 Haar Features applied on the image

Each stage of the classifier shows the region defined by the current location of the sliding window as either positive or negative. Positive indicates an object was found and negative indicates no object. If the label is negative, the detector shifts the the classification of this region is complete, and window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive. It is used to eliminate less likely regions quickly so that no more processing is needed. Hence, the speed of overall algorithm is increased.

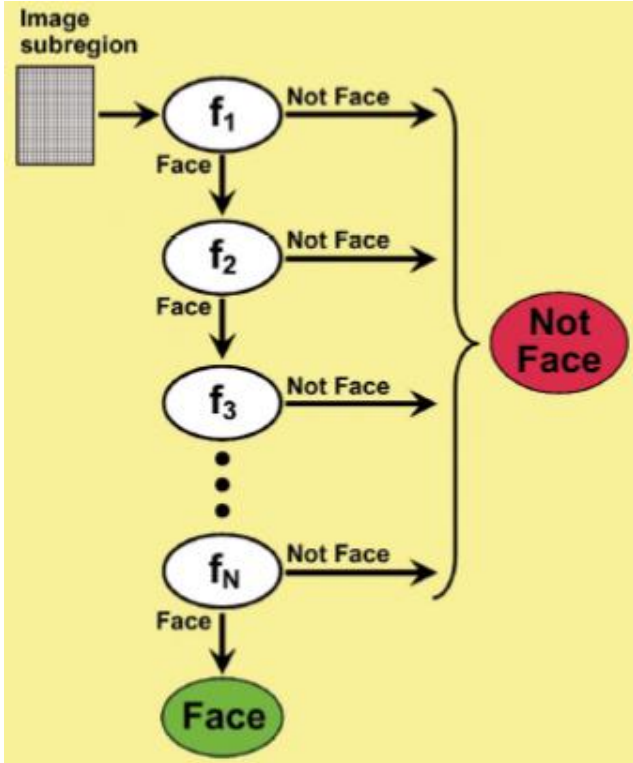


Figure 5.3 Face detection using Haar Cascade

5.1.3 Detect Multiscale

detectMultiScale function is used to detect the faces. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. It takes 3 common arguments — the input image, scaleFactor, and minNeighbours. scaleFactor specifies how much the image size is reduced with each scale. In a group photo, there may be some faces which are near the camera than others. Naturally, such faces would appear more prominent than the ones behind. This factor compensates for that. minNeighbours specifies how many neighbours each candidate rectangle should have to retain it. You can read about it in detail here. You may have to tweak these values to get the best results. This parameter specifies the number of neighbours a rectangle should have to be called a face. We obtain these values after trail and test over a specific range.

5.1.4 Eye Detection

Since the model works on building a detection system for drowsiness we need to focus on the eyes to detect drowsiness. The eyes are detected through the video input by implementing a haar classifier namely Haar Cascade Eye Classifier. The eyes are detected in rectangular formats.

Images or the real time video is captured from the camera installed in front of the driver's face. This video is converted into number of frames. OpenCV face OpenCV-classifier is loaded. Each frame is compared with the predefined features of the OpenCV-classifiers. When the features are matched the face is detected and a rectangle is drawn around the face. Using feature extraction we estimate the position of the eyes. By comparing with the OpenCV eye-OpenCV classifier, the eyes are detected and rectangles are drawn around left and right eye. In the system we have used facial landmark prediction for eye detection landmarks are used to localize and represent salient regions of the face. Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. In the context of facial landmarks, our goal is detecting important facial structures on the face using shape prediction methods. Detecting facial landmarks is therefore a two step process:

- Localize the face in the image
- Detect the key facial structures on the face ROI.

Localize the face in the image: The face image is localized by Haar feature-based cascade classifiers which was discussed in the first step of our algorithm i.e. face detection. Detect the key facial structures on the face ROI: There are a variety of facial landmark detectors, but all methods essentially try to localize and label the following facial regions:

- Mouth
- Left Eye
- Right eye

5.1.5 Face and eye tracking

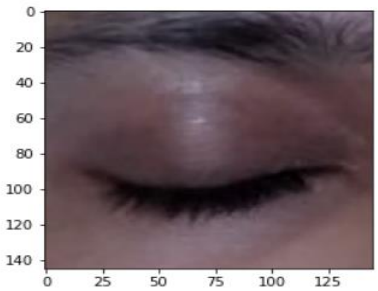
Due to the real-time nature of the project, we need to track the faces continuously for any form of distraction. Hence the faces are continuously detected during the entire time and

the state of the mouth is continuously monitored as yawn or no yawn. The eyes are also tracked continuously to check for open or closed eyes.

5.2 CODES SEGMENTS

```
import matplotlib.pyplot as plt
plt.imshow(plt.imread("train/Closed/_0.jpg"))
```

```
<matplotlib.image.AxesImage at 0x23924f9d970>
```




```
plt.imshow(plt.imread("train/yawn/1.jpg"))
```

```
<matplotlib.image.AxesImage at 0x239256c7c10>
```

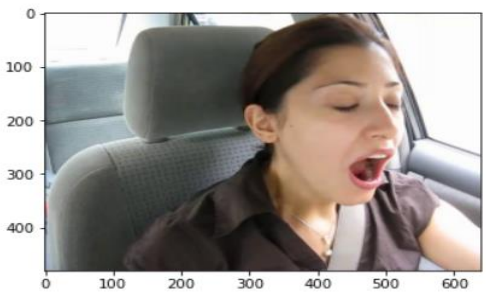


Figure 1.4 Visualization of random data

Extract and convert Array

```
def append_data():
    yaw_no = face_for_yawn()
    data = get_data()
    yaw_no.extend(data)
    return np.array(yaw_no, dtype =object)
```

Figure 5.5 Creating processed data

For yawn and no_yawn, take only face.

```
def face_for_yawn(direc="train", face_cas_path="prediction_images/haarcascade_frontalface_default.xml"):
    yaw_no = []
    IMG_SIZE = 145
    categories = ["yawn", "no_yawn"]
    for category in categories:
        path_link = os.path.join(direc, category)
        class_num1 = categories.index(category)
        print(class_num1)
        for image in os.listdir(path_link):
            image_array = cv2.imread(os.path.join(path_link, image), cv2.IMREAD_COLOR)
            face_cascade = cv2.CascadeClassifier(face_cas_path)
            faces = face_cascade.detectMultiScale(image_array, 1.3, 5)
            for (x, y, w, h) in faces:
                img = cv2.rectangle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
                roi_color = img[y:y+h, x:x+w]
                resized_array = cv2.resize(roi_color, (IMG_SIZE, IMG_SIZE))
                yaw_no.append([resized_array, class_num1])
    return yaw_no
```

```
yawn_no_yawn = face_for_yawn()
```

```
0
1
```

Figure 5.6 Processing images for yawn and no yawn

To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don't need color information to detect the objects.

We will be using haar cascade classifier to detect faces. This line is used to set our classifier **face = cv2.CascadeClassifier('path to our haar cascade xml file')**. Then we perform the detection using **faces = face.detectMultiScale(gray)**. It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

For open and closed eyes

```
def get_data(dir_path="train", face_cas="prediction-images/haarcascade_frontalface_default.xml", eye_cas="prediction-images/haarcascade.xml"):
    labels = ['Closed', 'Open']
    IMG_SIZE = 145
    data = []
    for label in labels:
        path = os.path.join(dir_path, label)
        class_num = labels.index(label)
        class_num +=2
        print(class_num)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)
                resized_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                data.append([resized_array, class_num])
            except Exception as e:
                print(e)
    return data
```

```
data_train = get_data()
```

```
2
3
```

Figure 5.7 Processing images for open and closed eyes

The same procedure to detect faces is used to detect eyes. First, we set the cascade classifier for eyes in **leye** and **reye** respectively then detect the eyes using **left_eye = leye.detectMultiScale(gray)**. Now we need to extract only the eyes data from the full image. This can be achieved by extracting the boundary box of the eye and then we can pull out the eye image from the frame

```
model = Sequential()

model.add(Conv2D(256, (3, 3), activation="relu", input_shape=X_train.shape[1:]))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Flatten())
model.add(Dropout(0.5))

model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="softmax"))

model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 143, 143, 256)     7168

 max_pooling2d (MaxPooling2D  (None, 71, 71, 256)      0
 )

 conv2d_1 (Conv2D)           (None, 69, 69, 128)       295040

 max_pooling2d_1 (MaxPooling  (None, 34, 34, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 32, 32, 64)        73792

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 14, 14, 32)        18464

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 flatten (Flatten)           (None, 1568)              0

 dropout (Dropout)           (None, 1568)              0

 dense (Dense)               (None, 64)                100416

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 495,140
Trainable params: 495,140
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.8 Neural Network Model

We are using CNN classifier for predicting the eye status. To feed our image into the model, we need to perform certain operations because the model needs the correct

dimensions to start with. First, we convert the color image into grayscale then resize the image, normalize our data for better. Now we predict eye and yawn with our model.

```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("./drowiness_new6.h5")

results={0:'yawn',1:'no_yawn', 2:'Closed', 3: 'Open'}
GR_dict={0:(0,0,255),1:(0,255,0), 2:(0,0,255), 3:(0,255,0)}

rect_size = 4
cap = cv2.VideoCapture(0)


haarcascade = cv2.CascadeClassifier('prediction_images/haarcascade.xml')
haarcascade2 =  cv2.CascadeClassifier('prediction_images/haarcascade_frontalface_default.xml')

while True:
    (rval, im) = cap.read()
    im=cv2.flip(im,1,1)


    rerect_size = cv2.resize(im, (im.shape[1] // rect_size, im.shape[0] // rect_size))
    faces = haarcascade.detectMultiScale(rerect_size)
    for f in faces:
        (x, y, w, h) = [v * rect_size for v in f]

        face_img = im[y:y+h, x:x+w]
        rerect_sized=cv2.resize(face_img,(145,145))
        normalized=rerect_sized/255.0
        reshaped=np.reshape(normalized,(1,145,145,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),GR_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),GR_dict[label],-1)
        cv2.rectangle(im,(x,y),(x+w,y+h),GR_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),GR_dict[label],-1)
        cv2.putText(im, results[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

    cv2.imshow('LIVE',   im)
    key = cv2.waitKey(10)

    if key == 27:
        break

cap.release()

cv2.destroyAllWindows()
```

Figure 5.9 Live Capture from web camera

With a webcam, we will take images as input. So to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by

OpenCV, **cv2.VideoCapture(0)** to access the camera and set the capture object (cap). **cap.read()** will read each frame and we store the image in a frame variable.

CHAPTER 6

6. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and or/a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations does not fail in unacceptable manner.

6.1 Types of Testing

6.1.1. Functional Testing:

Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black Box Testing tec logic of the system being tested is not known to the tester.Functional testing is normally performed during the levels of System Testing and Acceptance Testing. Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function's specifications.
- Determine the output based on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs

6.1.2. Non-Functional Testing

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. An excellent example of a non-functional test would be to check how many people can simultaneously login into a software. Non-functional testing is equally important as functional testing and affects client satisfaction.

The Non Functional requirements were also not given proper attention in the earlier test cycles. However, this has changed now. Non-functional tests are now most important as they consider all the application performance and security issues these days. This testing has a greater impact on applications when it comes to the performance of the application under high user traffic. This testing ensures that your application is stable and is able to handle load under extreme conditions.

6.1.3. Regression Testing

REGRESSION TESTING is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it. The likelihood of any code change impacting functionalities that are not directly associated with the code is always there and it is essential that regression testing is conducted to make sure that fixing one thing has not broken another thing. During regression testing, new test cases are not created but previously created test cases are re-executed. Regression [noun] literally means the act of going back to a previous place or state; return or reversion. In an ideal case, a full regression test is desirable but oftentimes there are time/resource constraints.

In such cases, it is essential to do an impact analysis of the changes to identify areas of the software that have the highest probability of being affected by the change and that have the highest impact to users in case of malfunction and focus testing around those areas. Due to the scale and importance of regression testing, more and more companies and projects are adopting regression test automation tools.

6.1.4. Performance Testing

Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload. Performance testing can involve quantitative tests done in a lab, or occur in the production environment in limited scenarios. Typical parameters include processing speed, data transfer rate, network bandwidth and throughput, workload efficiency and reliability. For example, an organization can measure the response time of a program when a user requests an action or the number of millions of instructions per second (MIPS) at which a mainframe functions

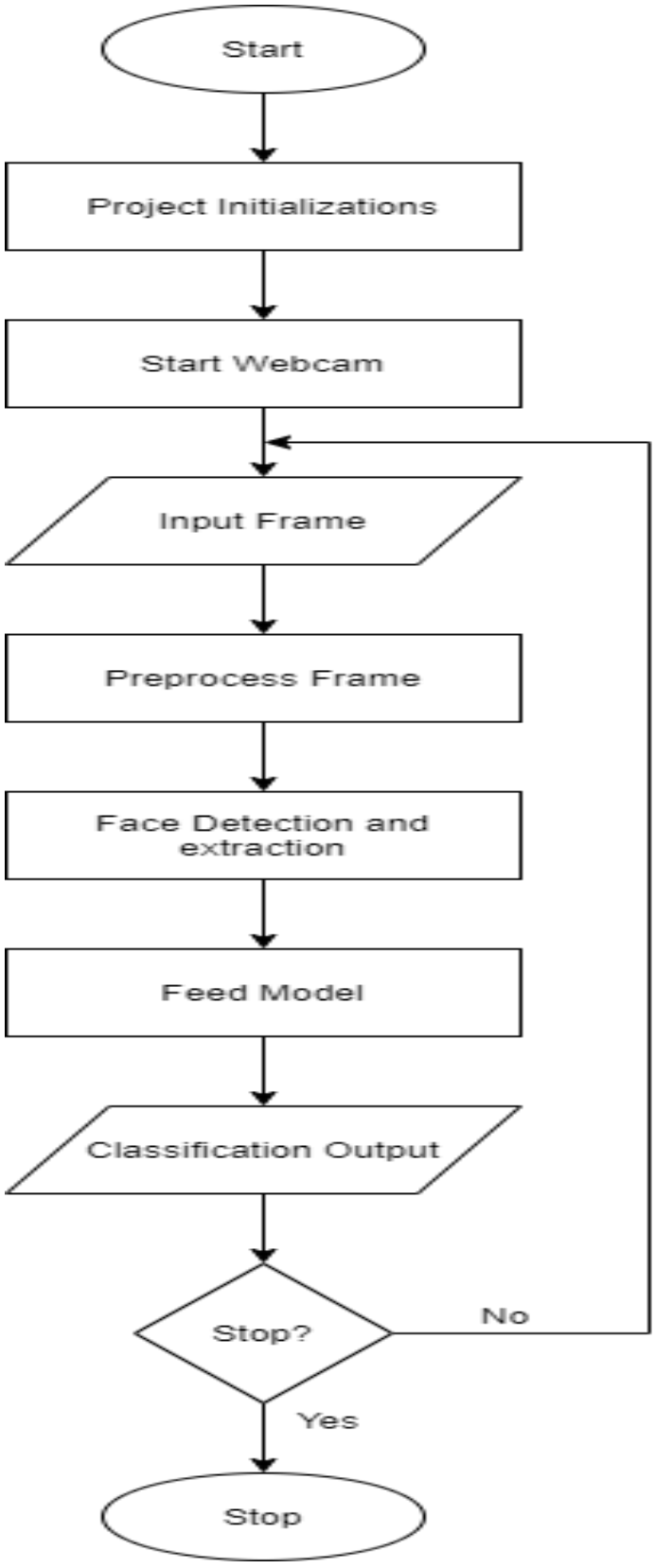


Figure 6.1 Flow chart for testing process

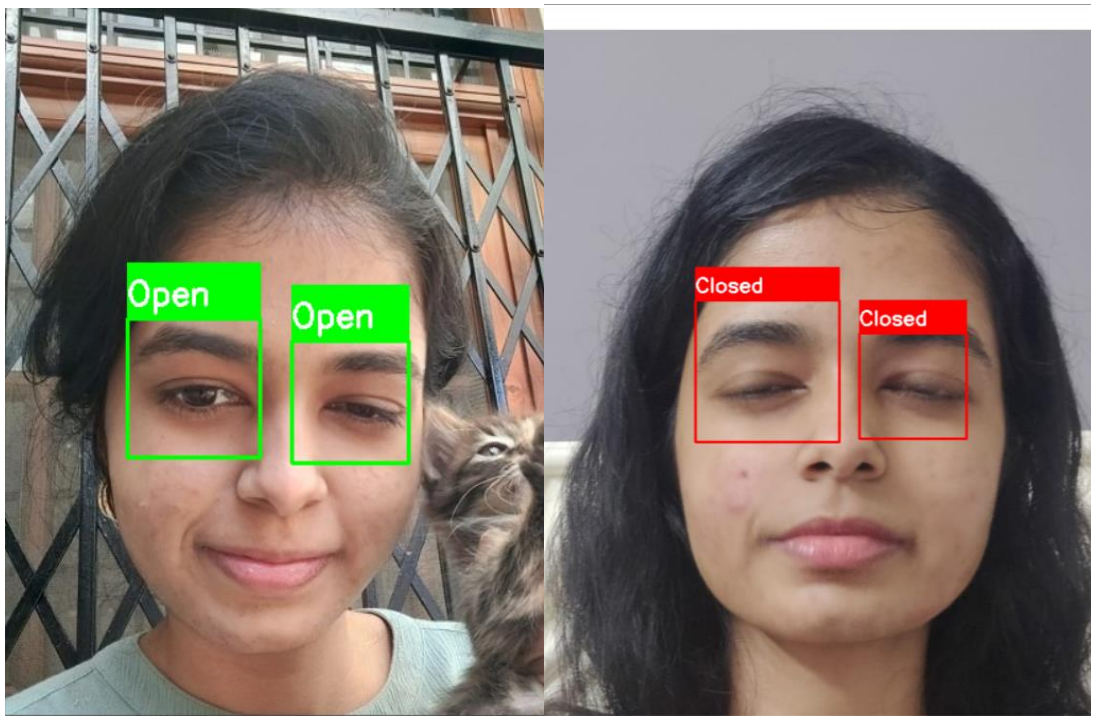


Figure 6.2 Testing on image -1

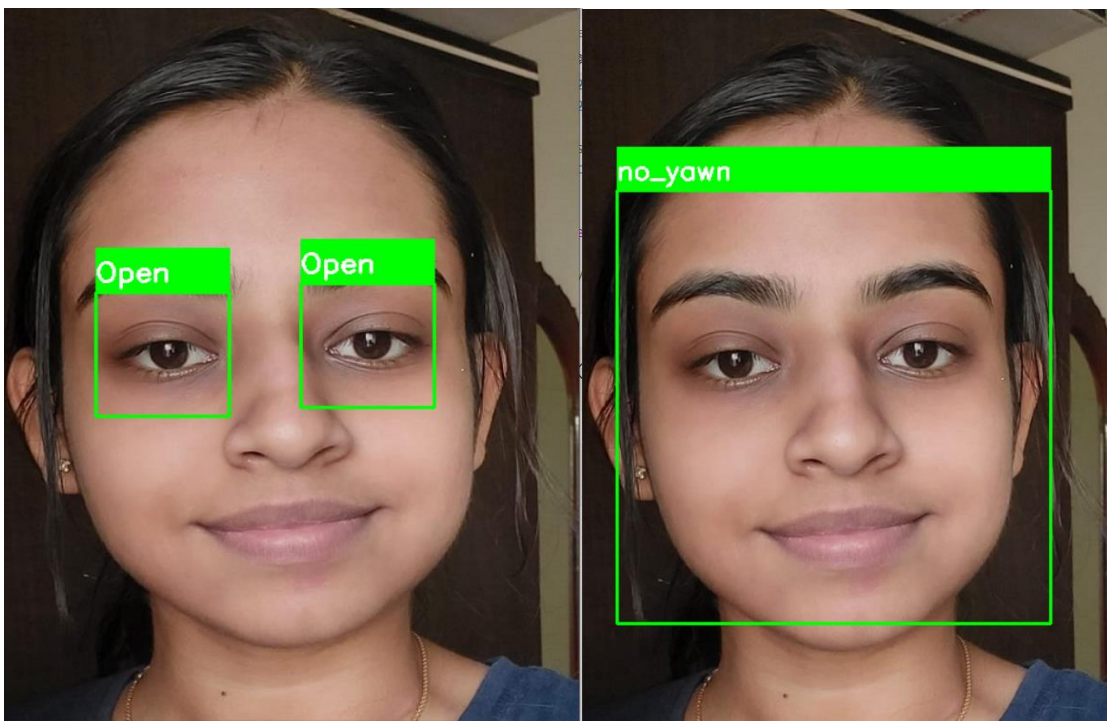


Figure 6.3 Testing on image -2

```
prediction = model.predict([prepare("train/no_yawn/1067.jpg")])
np.argmax(prediction)
```

✓ 0.1s

```
1
```

```
prediction = model.predict([prepare("train/Closed/_101.jpg")])
np.argmax(prediction)
```

✓ 0.1s

```
2
```

+ Code + Markdown

```
prediction = model.predict([prepare("train/Open/_101.jpg")])
np.argmax(prediction)
```

✓ 0.1s

```
3
```

```
prediction = model.predict([prepare("train/yawn/45.jpg")])
np.argmax(prediction)
```

✓ 0.1s

```
0
```

Figure 6.4 Testing performed random images

CHAPTER 7

7.RESULTS

7.1 OUTPUT SNAPSHOTS

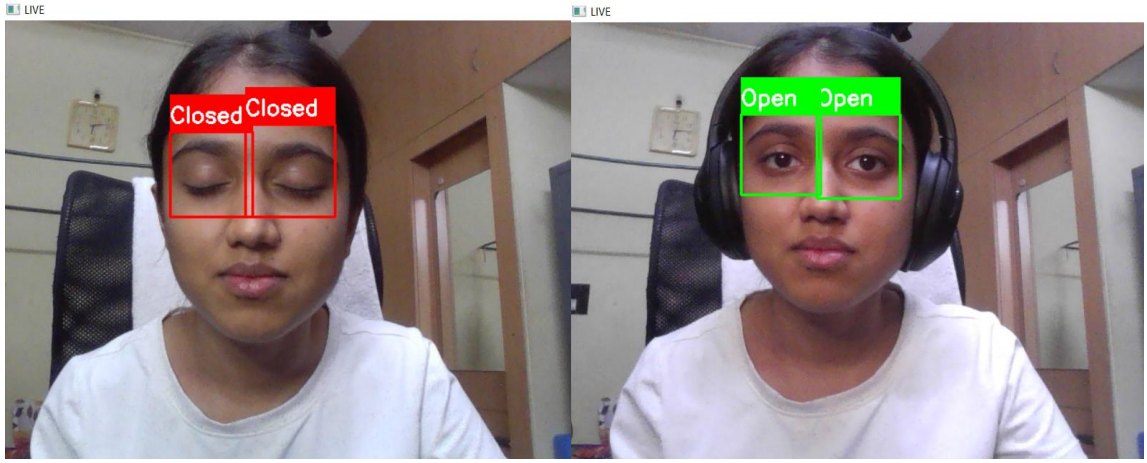


Figure 7.1 Live capture testing -1

The fig 7.2 is output for eye detection. The system detects eyes in the given particular frame in rectangular frames. The algorithm used for detecting the eyes is haar cascade. It uses haar features which are used for detecting the eyes in rectangular frames. Then the model classifies the image. draws rectangles around the region of interest along with the labels as ‘Open’ or ‘Closed’.

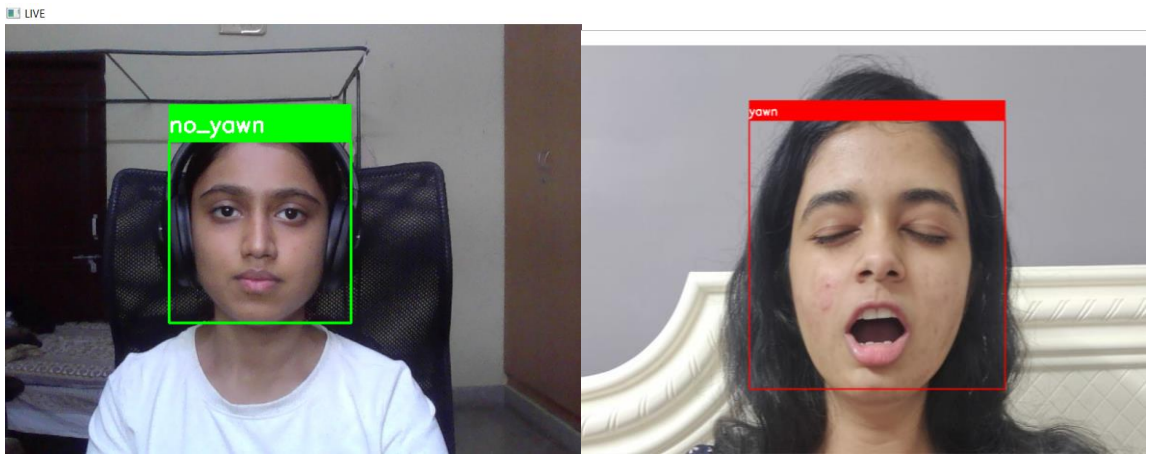


Figure 7.2 Live capture testing - 2

The fig 7.2 is output for face detection module. The input to this module is continuous stream of video and output will be face detection with in rectangular bounds. The face is detected by using haar cascade algorithm . It uses haar features through which the face is

detected in a rectangular frames. The detection of the face is achieved through the Haar classifiers mainly, the Frontal face cascade classifier. The face is detected in a rectangle format and converted to grayscale image and stored in the memory which can be used for training the model. Then it classifies the image as ‘yawn’ or ‘no yawn’ which is seen on the screen along with the labels.

7.2 GRAPHS AND TABLES

	precision	recall	f1-score	support
yawn	0.88	0.48	0.62	63
no_yawn	0.68	0.91	0.78	74
Closed	0.87	0.98	0.92	215
Open	0.98	0.88	0.93	226
accuracy			0.88	578
macro avg	0.85	0.81	0.81	578
weighted avg	0.89	0.88	0.87	578

Figure 7.3 Confusion matrix result

Recall: From all the positive classes, how many we predicted correctly.

Precision: From all the classes we have predicted as positive, how many are actually positive.

Accuracy: From all the classes (positive and negative), how many of them we have predicted correctly.

F-score: F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

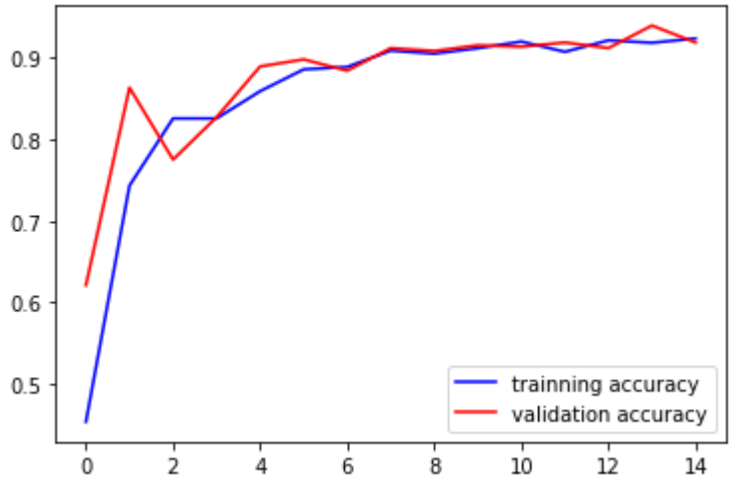


Figure 7.4 Training and Validation accuracy

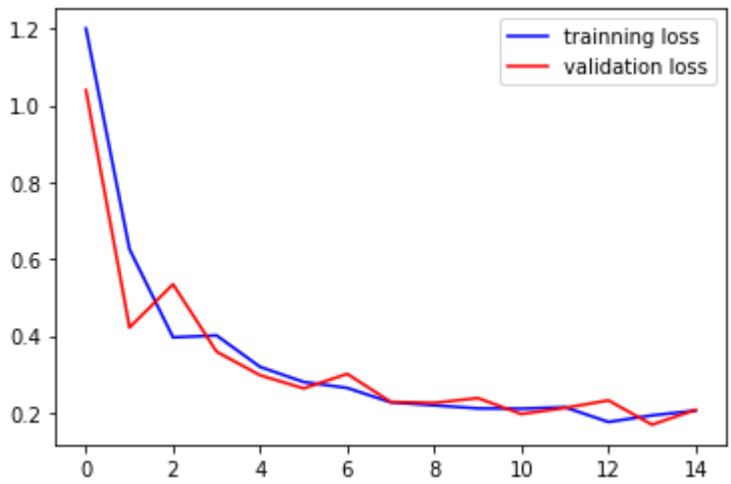


Figure 7.5 Training and validation loss

The accuracy of the model is 90%. It can also be observed in the accuracy loss graph obtained below.

- The loss value is below 0.1.
- It can be observed that with every iteration of the model training, the loss is getting gradually decreased and the accuracy is getting increased
- The plot of training loss decreases to a point of stability. The plot of validation loss decreases to a point of stability and has a small gap with the training loss. Thus it can be concluded that it does not overfit.

CHAPTER 8

8. CONCLUSION AND FUTURE ENHANCEMENT

The current study developed an automated system for detecting drowsiness of the driver. The continuous video stream is read from the system and is used for detecting the drowsiness. It is detected by using haar-cascade algorithm. The haar-cascade algorithm uses haar features to detect face and eyes. Haar features are predefined are used for detecting different things. It is used to detect face and eyes in the image which is then fed to the classifier to classify the images as ‘Open’, ‘Closed’, ‘yawn’, and ‘no_yawn’.

This work can also be extended by implementing in full night light using IR web cam. It is camera which uses infrared radiations to detect whether the person is drowsy or not. While this is a research project, there is scope when this completely turns out to be developed into an application which can be run by the end users on their own for their own purposes on their own systems.

The work can be extended by recording the time for which the eyes remain closed and then defining a threshold, for example if score becomes greater than 15 that means the person’s eyes are closed for a long period of time. This is when we beep the alarm and notify the driver.

CHAPTER 9

9. REFERNCES

[1] [1] Assari, M. A., & Rahmati, M. (2011). Driver drowsiness detection using face expression recognition. 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA).

[2] M. Yauri-Machaca, B. Meneses-Claudio, N. Vargas-Cuentas and A. Roman-Gonzalez, "Design of a Vehicle Driver Drowsiness Detection System Through Image Processing using Matlab," 2018 IEEE 38th Central America and Panama Convention (CONCAPAN XXXVIII), 2018, pp. 1-6, doi: 10.1109/CONCAPAN.2018.8596513.

[3] Tianyi Hong, Huabiao Qin, & Qianshu Sun. (2007). An Improved Real Time Eye State Identification . System in Driver Drowsiness Detection. 2007 IEEE International Conference on Control and Automation.

[4] Warwick, B., Symons, N., Chen, X., & Xiong, K. (2015). Detecting Driver Drowsiness Using Wireless Wearables. 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems.

[5] Dwivedi, K., Biswaranjan K & Sethi, A. (2014). Drowsy driver detection using representation learning. 2014 IEEE International Advance Computing Conference (IACC).

[6] Yan, J.-J., Kuo, H.-H., Lin, Y.-F., & Liao, T.-L. (2016). Real-Time Driver Drowsiness Detection System.Based on PERCLOS and Grayscale Image Processing. 2016 International Symposium on Computer, Consumer and Control (IS3C).

[7] Face detection using haar cascades tutorial https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0

[8] Hanojhan Rajahrajasingh (Author), 2019, Drowsiness Detection Using Image Processing, Munich, GRIN Verlag, https://www.grin.com/document/506703