A PROJECT REPORT

on

REAL TIME DROWSINESS DETECTION AND ALERT SYSTEM

submitted in fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

19WH1A05C5	**Ms. R. AKANKSHA**
19WH1A05E6	**Ms. N. SINDHUJA**
19WH1A05F7	**Ms. M. SUPRATHIKA**

Under the esteemed guidance of
Ms. Vidyullatha Sukhavasi
Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTU, Hyderabad)

Bachupally, Hyderabad – 500090

June, 2023

DECLARATION

We hereby declare that the work presented in this project entitled **"Real time drowsiness detection and Alert System Securing "** submitted towards completion of Project Work in IV year of B.Tech., CSE at **BVRIT HYDERABAD College of Engineering for Women**, Hyderabad is an authentic record of our original work carried out under the guidance of **Ms. Vidyullatha Sukhavasi, Assistant Professor, Department of CSE.**

Sign. with date:

Ms. REDDY AKANKSHA

(19WH1A05C5)

Sign. with date:

Ms. NANDIKATTI SINDHUJA

(19WH1A05E6)

Sign. with date:

Ms. MOTHKUR SUPRATHIKA

(19WH1A05F7)

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTU, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Work report on " **Real time Drowsiness detection and Alert system** " is a bonafide work carried out by **Ms. REDDY AKANKSHA (19WH1A05C5)**; **Ms. NANDIKATTI SINDHUJA (19WH1A05E6)**; **Ms. MOTHKUR SUPRATHIKA (19WH1A05F7)** in the fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. E. Venkateswara Reddy
Professor, CSE

Guide
Ms. Vidyullatha Sukhavasi
Assistant Professor

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. E. Venkateswara Reddy, Professor & HOD**, **Department of CSE**, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms Vidyullatha Sukhavasi, Assistant Professor, Department of CSE, BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **Computer Science and Engineering** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. REDDY AKANKSHA

(19WH1A05C5)

Ms. NANDIKATTI SINDHUJA

(19WH1A05E6)

Ms. MOTHKUR SUPRATHIKA

(19WH1A05F7)

ABSTRACT

Driver fatigue has become one of the key reasons for road accidents in modern days. Various surveys prove that if a driver is correctly identified as fatigued, and he, or she is timely alarmed regarding the same, the cases of accidents can be remarkably reduced. Through this project, an in-depth study of various existing techniques of fatigue in a driver is studied, followed by developing a deep learning-based model to accurately identify a driver's state using a novel technique of using spatiotemporal features of the face. It can be determined that the accuracy will remarkably increase when this technique is used.

This will directly give an indication of drowsiness/fatigue which can be further used as record of driver performance.

The face, an important part of the body, conveys a lot of information. When a driver is in a state of fatigue, the facial expressions, e.g., the frequency of blinking and yawning, are different from those in the normal state. In this paper, we propose a system called DriCare, which detects the drivers' fatigue status, such as yawning, blinking, and duration of eye closure, using video images, without equipping their bodies with devices. Owing to the shortcomings of previous algorithms, we introduce a new face-tracking algorithm to improve the tracking accuracy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER - 1
INTRODUCTION

1. INTRODUCTION

Real Time Drowsiness behaviours 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 eye blinking to monitor drowsiness or consider physical changes such as sagging posture, leaning of driver's head and open/closed state of eyes.

Driver drowsiness detection is car safety technology which helps to save the life of a driver by preventing accidents when the driver is getting drowsy. Real Time Drowsiness behaviors which are related to fatigue are in the form of eye closing, head nodding or brain activity. Hence, we can either measure change in physiological signals, such as brain waves, heart rate and eye blinking to monitor drowsiness or consider physical changes such as sagging posture, 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, microsleeps that are short periods of sleep 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 the driver and a timely warning is issued.

Fig 1 : Driver Drowsiness

1.1 Objectives of Project

The Project focuses on these objectives, which are:

- To suggest ways to detect fatigue and drowsiness while driving.
- To study on eyes and mouth from the video images of participants in the experiment of driving simulation conducted by MIROS that can be used as an indicator of fatigue and drowsiness.
- To investigate the physical changes of fatigue and drowsiness.
- To develop a system that use eyes closure and yawning as a way to detect fatigue and drowsiness.

1.2 Motivation

Driver fatigue, or drowsiness, contributes to many thousands of deaths and injuries on the roads every year. It has a role in up to 30% of fatal crashes and up to 15% of serious injuries . Driver fatigue occurs when a driver does not get enough sleep or after long-distance driving. The influence of fatigue on driving safety is similar to that of alcohol impairment. A survey of the U.S population found that 37% of workers got less than the recommended minimum of 7 h of sleep. Driver fatigue is responsible for 22.7% of fatal accidents and 20.5% of accidents with injuries on Canadian roads. In 40% of fatigue-related accidents, the driver was awake for over 17 h. The main problem with fatigue is that it is very difficult to be aware of the tiredness of drivers before they become too tired to drive safely. Driver fatigue is a major cause of car accidents, since sleepy drivers are unable to make rapid decisions, and they may have slower reaction times . Therefore, recent research also suggests that driver fatigue detection methodologies can be used to prevent such accidents . The research on driver fatigue detection can be classified into two categories: passive driver monitoring system and active driver fatigue control system. Among them, the first is the use of preventive measures, by which a driver is monitored, and an alarm is displayed to advise the driver to stop the vehicle during the fatigue stage. The second is the abnormal fatigue detection in which a driver is deemed to be asleep and the system can actively control the vehicle to avoid accidents.

In recent years, researchers have developed methodologies to detect or indicate the driver fatigue state prior to a collision . Generally, data from three different sources have been used to classify driver fatigue or drowsiness. Firstly, vehicle-based measures are used to detect a drowsy or unsafe state. For example, vehicle-based measuring may be obtained by steering wheel angle, acceleration pedal data, and lane departure information by an external sensor . Secondly, observation of the driver behavior method may be determined by continuous recording through a camera installed in the vehicle to monitor eye-closure time, eye blinking frequency, movement and pose of the head, and yawing, etc.. The most recognized measure of these models is the percentage of eye closure time (PERCLOS) , although others have used measures such as eye aspect ratio (EAR) . Thirdly, driver-based measures include physiological measurements such as electroencephalography (ECG), as well as heart rate, respiration rate, pulse pattern, and popular cortical signals, etc. .

In summary, owing to easy installation and low cost, observation of the driver behavior method has been widely used for fatigue detection. For example, attention technologies and smart eye technology employ the movement of the driver's eyes and position of the driver's head to determine the level of their fatigue.

In this, we propose a real-time observation of the driver behavior method to detect driver fatigue. Our method only uses the inside vehicle camera, which is handy for the driver to carry on, or convenient to install in the vehicle. The novelty of this study is to develop a novel real-time driver fatigue detection system that has two major advantages. First, we introduce a new face-tracking algorithm based on facial landmarks to improve detection accuracy.

1.3 Organization of Documentation

Project Documentation is an important part of project management. It is substantiated by the essential two functions of documentation: to make sure that project requirements are fulfilled and to establish traceability with regard to what has been done, who has done it, and when it has been done. Thus, documentation must lay the foundation for quality, traceability, and history for both the individual document and for the entire project documentation. It is thus extremely important that the documentation is well arranged, easy to read, and adequate.

Experienced project managers excel at making and following standard templates for their project documents. They reuse successful project plans, business cases, requirement sheets, and project status reports. This helps them focus on their core competency of managing the project rather than balancing the unmanageable paperwork.

1.4 Methodology

Some machine learning approaches include support vector machines, convolutional neural networks, in the context of drowsiness detection.

Accuracy was increased by utilizing facial landmarks which are detected by the camera and that is passed to a Convolutional Neural Network (CNN) to classify drowsiness. The achievement with this work is the capability to provide a lightweight alternative to heavier classification models with more than 88% for the category without glasses, more than 85% for the category night without glasses. On average, more than 83% of accuracy was achieved in all categories.

In the field of sleepiness detection, hidden Markov models and neural networks are used. Additionally, a meta-analysis of 25 studies using machine learning algorithms for sleepiness detection is done.

A Dynamic fatigue detection model based on the Hidden Markov Model (HMM) is proposed . Driver fatigue can be estimated by this model in a probabilistic way using various physiological and contextual information. Electroencephalogram (EEG), Electromyogram (EMG), and respiration signals were simultaneously recorded by wearable sensors and sent to the computer by Bluetooth during the real driving. From this physiological information, fatigue likelihood can be achieved using kernel distribution estimates at different time sections. Contextual information offered by specific environmental factors were used prior to fatigue.

As time proceeds, the posterior of fatigue can be gotten dynamically by this HMM recognition method.

1.4.1 Dataset

An academic Driver Drowsiness Detection (DDD) dataset is used, which was first introduced during the 2016 Asian Conference on Computer Vision. Videos were recorded at a 480 X640 resolution with a frame rate of 30 and 15 fps for day and night videos, respectively. For each subject, videos were recorded in a controlled setting in five conditions:

- without glasses
- with glasses
- with sunglasses
- without glasses at night
- with glasses at night

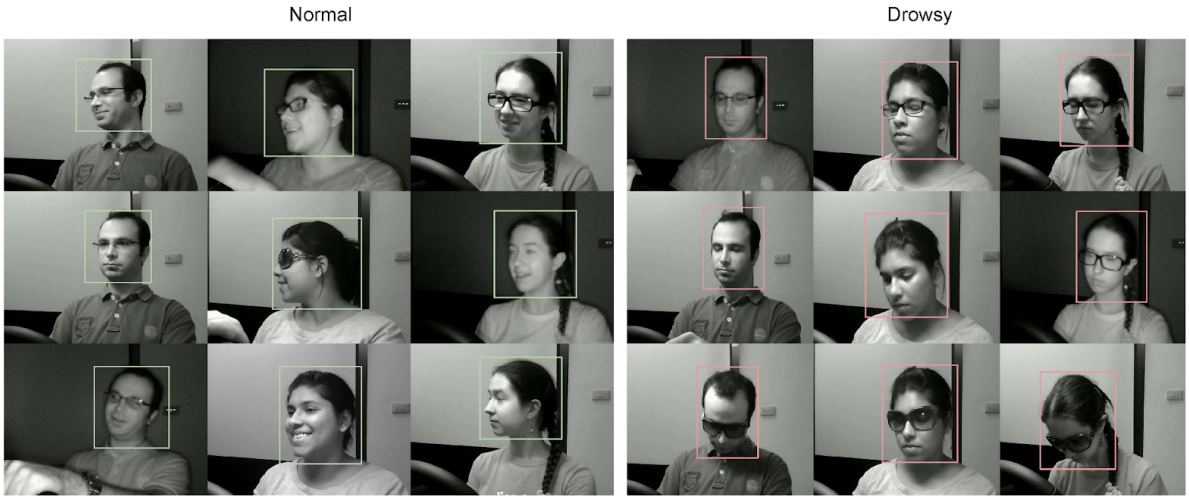


Fig 02 : Classification of Drowsiness

Simulated behaviors include yawning, nodding, looking aside, talking, laughing, closing eyes and regular driving, and video segments have been labelled as drowsy or non-drowsy. The dataset consists of training (18 persons), evaluation (4 persons) and testing (14 persons) sets. For this, the training dataset was used for model calibration (a total of 8.5 h of video), the evaluation dataset for validation purposes (1.5 h of video), while the testing dataset was not used. First, night videos were converted from 15 to 30 fps to match the frame rate of the other videos in the dataset. Videos were then resized from 480 640 to 240 320 to reduce pre-processing time during training and disc space. The video files were split into 100-frame sequences for training and 10-frame sequences for validation. This resulted in 9094 100-frame

training records and 17,318 10-frame validation records. Note that a small fraction of 10-frame sequences from the original videos is not used for training (i.e., 10-frame sequences spanning two records), which was a trade-off for faster read performance.

Fig 03 : Dataset

1.4.2 Proposed Workflow

For the purpose of implementing a baseline model so as to obtain a model which has been trained from scratch (i.e., without any pre-training), we have prepared a CNN model with 3 Convolutional Layers and a 'Relu' activation function.

These layers are followed by Max-pooling layers after each Conv layer. At the top, two dense fully connected layers are attached which finally classifies a frame as drowsy or non-drowsy using a 'Sigmoid' activation function.

The flow of data in this model, the following are some highlighting points:

- Input to the model: (3 x 224 x 224)
- Output of ConvLayers: (1 x 56 x 64)
- Input to Fully Connected Layers: (1 x 1792)
- Final Output: 0 or 1 (Not Drowsy or Drowsy)

We have trained our Baseline Model in batches of 16 which means the total number of batches that are trained per epoch (iteration) = 45,203 batches Our major concern while training such networks is Over fitting. To avoid this, we have predominantly used heavy Data Augmentation. But to really tackle the issue, we also employ a Regularization technique called L2 Regularization. Which consists in forcing model weights to take smaller values thus mitigating the risk of overfitting up to a great extent.

Final Model (with VGG16 ImageNet)

We prepare our final model after fine tuning weights from a VGG16 model pre-trained on Image Net dataset.

Driver Drowsiness Detection and alert system Workflow:

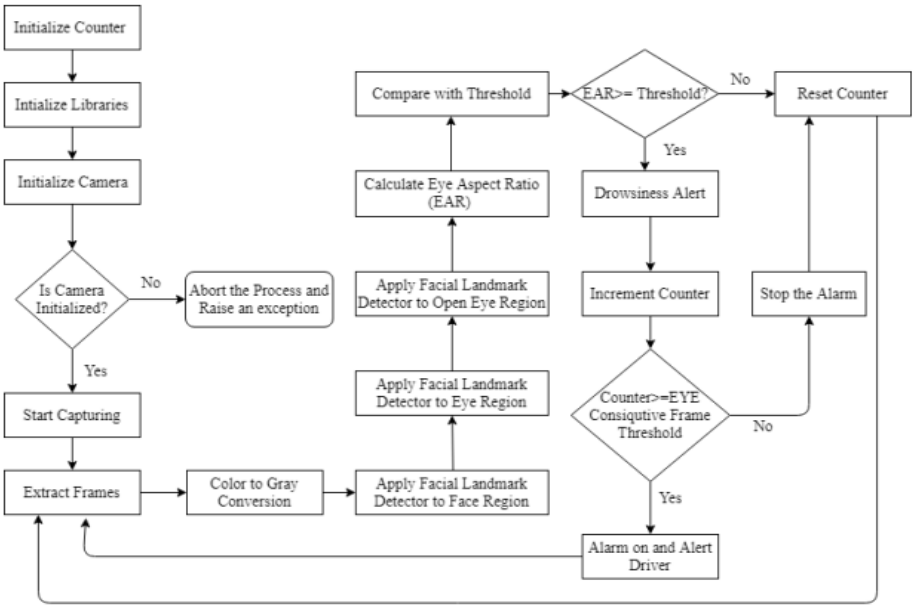


Fig 04 : Driver Drowsiness Detection and alert system Workflow

The above flow chart, explains the detail flow of drowsiness detection systems. It firsts captures the face from the video stream that is fixed in the automobiles then starts capturing the video. Based on the EAR ratio and MAR ratio that was calculated by the eucledian distance between the landmarks that were marked using dlib or opencv. If the EAR or MAR ratio is greater than the threshold then we can conclude, that the person is feeling drowsy whereas coming to MAR ration we get the counter so that if it crosses the threshold the counter value increments. If it reaches the threshold the the person gets the warning that he/she feeling drowsy and yawning.

Block diagram:

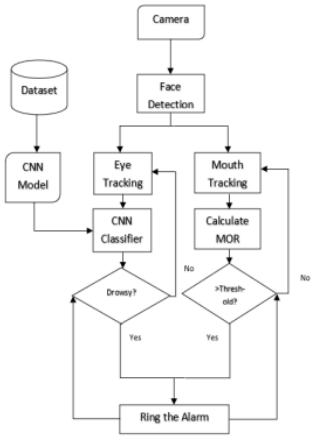


Fig 5 : Block Diagram of proposed drowsiness detection system

1.5.1 Pre-processing & Data Augmentation

Since the DDD dataset contains a limited amount of training data, several pre-processing steps were implemented to increase the variety of samples supplied to the neural network during training. These pre-processing steps were tailored to the issue of drowsiness detection and increase robustness of the model when applied in a real-world setting. This not only enhances training accuracy but also lowers overfitting. Following preprocessing/ data augmentation steps were taken –

- All the videos are converted into image frames at 30 frames per second .
- All the frames are labelled according to the drowsiness state (i.e. drowsy or not drowsy).
- Brightness of all the frames is normalized (by dividing from the largest value: 255).
- Some of the frames are randomly rotated by 40 degrees.
- Some of the frames are horizontally flipped.
- Images are given zoom, shear, shift (height & width) of magnitude 0.2
- The sample was rescaled to 224 x 224 x 1.

This approach achieves model invariance to translations (horizontal, vertical shifts), zooming, and face shapes.

Division in	No. of frames	No. of categories
Training set	7,23,248	2
Test Set	1,73,299	2

Table 1: Data after preprocessing

Splitting the data into train and test: It's usual practice in machine learning to divide a dataset into training and testing sets in order to assess a model's performance. The testing set is used to assess the model's performance on untested data, while the training set is used to fit the model's parameters.

Building the models: As from the problem statement, we are going to build two different models. the first model is the Baseline model developed from scratch(i.e., no transfer

learning) and the second is the fine tuned VGG16 model developed in order to reduce the testing time and improve the accuracy of the model.

Model Training: The model is trained iteratively across a number of epochs. Each epoch entails dividing the training data into smaller batches and making predictions on these batches. The optimization technique computes the loss and adjusts the model's parameters based on the gradients of the loss with respect to the parameters.
The validation set is used to evaluate the model's performance after each epoch or a defined number of iterations. Metrics such as accuracy are generated to assess the model's ability to generalize to previously unseen data. Monitoring these indicators aids in spotting overfitting and underfitting and directs model modifications.

Early stopping can be used to prevent overfitting. Tracking the model's performance on the validation set and halting the training process if the validation performance does not increase after a particular number of epochs.

CHAPTER - 2
RELATED WORK

2 RELATED WORK

In the following section, we will be looking at the literature survey which provides a comprehensive review of existing research, scholarly articles, books, and other relevant sources related to the topic at hand. It serves as the foundation for the project by exploring and analyzing the existing body of knowledge on the subject.

[1] “Driver Drowsiness Detection using Machine Learning Approach” - Priyanka Basavaraj Murdeshwar, Shruthi Tharanath, Surekha Reddy, proposed aa approach that has three stages: for detecting Face Haar Cascade has been used, for detecting Eyes they have used Adaboost (selecting important features) and detecting drowsiness based on EAR.

[2] “Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques” - Rateb Jabbar , Khalifa Al-Khalifa, Mohamed Kharbeche, Wael Alhajyaseen, Mohsen Jafari , Shan Jiang proposed an approach using Multi level perceptron.The landmark coordination extracted from images will act as the input to the Multilayer
Perceptron Classifier . The training will be processed till the accuracy is reached. 80.92% accuracy has been achieved by this method.

[3] Driver drowsiness detection using Behavioral measures and machine learning techniques: A review of state-of-art techniques - Mkhuseli Ngxande, Jules-Raymond Tapamo, Michael Burke proposed a way by using Machine learning techniques such as SVM, CNN, and HMM are reviewed on image yawn eye dataset. CNNs yielded more accurate results when compared to SVMs and HMMs.

[4] “Driver Drowsiness Detection System for Vehicle Safety” - A. Subbarao, K.Sahithya proposed that the eye blink of the driver is detected by using an eye blink sensor. The disparity across the eye will vary as per eye blink.The output is high, if the eye is closed or else output is low.

Features are used based on various requirements, and possess certain traits, which are, in brief, given in Table. On looking at various parameters associated with features to be used, we

see that the biological features are intrusive in nature, due to which the driving experience and other factors of the driver will get hampered. Further, it does not have real time applicability, and the installation costs are high. Simultaneously, vehicular features are not that accurate, because it will depend a lot on the surrounding, e.g. A turning road will have a lot of movement of the steering wheel, and it can still detect this as a drowsy driver. Simultaneously, installation cost is also high in some of the techniques, and hampers real-time applicability. Whereas, in physical features, these disadvantages are not present. There is some dependency on the brightness of the surrounding, but still, other advantages make this way better as compared to using other features. So, in this project, physical features are used to determine the state of the driver.

Category	Signal	Parameter	Contact	Cost	Realtime Applicability	Limitations
Biological Features	Brain	EEG	Yes	Low	No	Extremely Intrusive Prone to human movement
	Heart	ECG	Yes	High	No	
	Skin	EMG	Yes	High	No	
Vehicle Features	Steering	WA	Yes	High	Yes	Driver and environment dependency
	Lane	Lane Deviation	No	Low	Yes	
	Posture	Pressure	Yes	High	Yes	
Physical Features	Eyes	Wink	No	Low	Yes	Illumination and Background Dependency
	Mouth	Yawn	No	Low	Yes	
	Face	Nod	No	Low	Yes	
	Nose	Structure	No	Low	Yes	

Table 2: Comparison of various Features

CHAPTER 3
THEORETICAL ANALYSIS OF PROPOSED PROJECT

3. THEORETICAL ANALYSIS OF PROPOSED PROJECT

3.1 Requirements Gathering

In the following section, we shall discuss the software and hardware requirements of the project.

3.1.1 Software Requirements

Programming Language	: Python 3.11
Dataset	: Drowsiness video dataset
Packages	: Argparse, Numpy, Pandas, Matplotlib, CV2, sklearn
Tool	: Jupyter Notebook
Web interface	: HTML, Tkinter
Language	: Python

3.1.2 Hardware Requirements

Operating System	: Windows
Processor	: Intel Core i7-2348M
CPU Speed	: 2.30 GHz
Memory	: 4GB and above
Hard Disk	: 50GB

3.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

● Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

● Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse is part of this, and so is access to powerful constructs that avoid tedious repetition. Code says about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the case with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched.

HTML

HTML stands for Hypertext Markup Language, and it is the most widely used language

1o write Web Pages.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply “mark-up” a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

An HTML element is set off from other text in a document by “tags", which consist of the element name surrounded by "<" and ">". The name of an element inside a tag is case-insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the <title> tag can be written as <Title>, <TITLE>yr in any other way. However, the convention and recommended practice is 1o write tags in lowercase.

The main parts of HTML clements are as follows:

- The opening tag: This consists of the name of the element (in this case, p), wrapped in opening and closing angle brackets. This states where the element begins or starts to take effect — in this case where the paragraph begins.
- The closing tag: This is the same as the opening tag. except that it includes a forward slash before the element name. This states where the element ends - in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
- The content: This is the content of the element. which in this case, is just text.
- The element: The opening tag, the closing tag, and the content together comprise the element.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

Elements can also have attributes. Attributes contain extra information about the element that you don't want to appear in the actual content. Tlere, class is the attribute name and editor-note is the attribute value. The class attribute allows you to give the element a non-unique identifier that can be used to target it (and any other elements with the same class value) with style information and other things. Some attributes have to value, such as required

Attributes that set a value always have:

- A space between it and the clement name (or the previous attribute, if the element already has one or more attributes).
- The attribute name is followed by an equal sign.
- The attribute value is wrapped by opening and closing quotation marks.

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 contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Panda solved this problem.

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. It has features which are used for exploring, cleaning, transforming and visualizing from data.

Pandas are well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational statistical data sets. The data need not be labeled at all to be placed into a pandas data structure

Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as NaN) in floating point as well
- Size mutability: columns can be inserted and deleted from Data Frame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, Data Frame, etc. automatically align the data for you in computations.
- Powerful, flexible group by functionality to perform split-apply-combine operations

on data sets, for both aggregating and transforming data.

- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into Data Frame object.
- Intelligent label-based slicing, fancy indexing, and sub setting of large data SCIS.
- Intuitive merging and joining data sets

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

1. Matplotlib provides a simple way to access large amounts of data

With Matplotlib, developers can create accurate plots based on huge amounts of data. When analyzing these data plots, good developers will make it easier to see patterns and trends in the data sets. Thus, Matplotlib simplifies data, making it more accessible.

2. It is flexible and supports various forms of data representation

As noted above Matplotlib supports data representation in bar charts, graphs, scattered plots, and other forms of visualization. This flexibility means that it can adapt effectively to your company's needs.

3. It is easy to navigate

The Matplotlib platform isn't too complex. Hence, both beginners and advanced developers can apply their programming skills to the platform, producing professional results. Matplotlib also has subplots that further facilitate the creation and comparison of data sets.

4. It ensures accessibility by providing high-quality images

Since the main goal of Matplotlib is to provide a way to access and display data, its plots and

images must be of high quality. To meet this requirement, Matplotlib provides high-quality images in various formats, such as PDF, PGF, and PNG.

5. It is a powerful tool with numerous applications

Matplotlib's data-visualization qualities can be used in various forms, such as Python scripts, shells, web application servers, and Jupyter notebooks. As such, its operations are versatile.

6. It is useful in creating advanced visualizations

Matplotlib is primarily a 2D plotting library. However, it includes extensions that developers can apply to create advanced 3D plots for data visualization. In this way, the platform ensures that working with data is easier and more productive.

7. It is open-source, saving you cash

An open-source platform requires no paid license. Because Matplotlib is free to use, you save the extra cost you usually incur when producing data visualizations.

Unlike other data-visualization platforms, Matplotlib in Python only requires a few lines of code to generate a plot for data sets.

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 stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors".

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 from 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 kernel methods of machine learning. The language is great to use when working with machine learning algorithms and has relatively easy syntax.

Machine Learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own. Different algorithms can be used in machine learning for different tasks, such as simple linear regression that can be used **for prediction problem**s like **stock market prediction,** and **the KNN algorithm can be used for classification problems**.

Machine Learning Algorithm can be broadly classified into three types:

1. Supervised Learning Algorithms
2. Unsupervised Learning Algorithms
3. Reinforcement Learning Algorithm

1) Supervised Learning Algorithm

Supervised learning is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing sample test data to check whether it predicts the correct output.

2) Unsupervised Learning Algorithm

It is a type of machine learning in which the machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be trained using the unlabelled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision.

3) Reinforcement Learning

In Reinforcement learning, an agent interacts with its environment by producing actions, and learn with the help of feedback. The feedback is given to the agent in the form of rewards, such as for each good action, he gets a positive reward, and for each bad action, he gets a negative reward. There is no supervision provided to the agent.

OpenCV

OpenCV stands for Open Source Computer Vision. It's an Open Source BSD licensed library that includes hundreds of advanced Computer Vision algorithms that are optimized to use hardware acceleration. OpenCV is commonly used for machine image processing, image manipulation, and much more. OpenCV has a modular structure. There are shared and static libraries and a CV Namespace. In short, OpenCV is used 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. OpenCV.org is a must if you want to explore and dive deeper into image processing and machine learning in general.

Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

Although Tkinter is considered the de facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to quickly build something that's functional and cross-platform.

Keras

Keras is a deep learning API written in Python, running on top of machine learning. Being able to go from idea to result as fast as possible is key to doing good research. Keras means horn in Greek. It is a reference to a literary image from ancient Greek and Latin literature, first found in the Odyssey, where dream spirits (Oneiroi, singular Oneiros\re divided between those who deceive dreamers with false visions, who arrive to Earth through a gate of ivory, and those who announce a future that will come to pass, who arrive through a gate of horn. Up until version 2.3. Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.As of version a OnLY TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

CHAPTER 4
DESIGN

4. DESIGN

4.1 Introduction

Software design is an essential component of the software engineering process that is applicable to any development paradigm and application arca. It is the first step in the development of any engineered product or system. The designer’s primary purpose is to build a model or representation of an entity that will be built later.

Following the formulation and analysis of system requirements, system design is the first of three technical processes in software development. It's a fundamental process required for the development and validation of software. Its significance can be summarized in a single word: "Quality". Design fosters quality in software development by providing representations of software that can be evaluated for quality.

Design is critical because it is the only way to accurately transform a customer's perspective into a finished software product or system. It forms the basis for all subsequent software engineering steps. Without a strong design, there is a risk of creating an unstable system that will be difficult to test and whose quality will not be assessed until the very end.

During the design phase, progressive refining of data structure, software structure, and procedural aspects takes place. These aspects are created, examined, and documented to ensure that the system is well-designed. System design can be approached from both a technical and a project management standpoint.

Architectural design, data structure design, interface design, and procedural design are the four main activities of design from technical The goal of architectural designs to define the overall structure and organization of the software system. It entails determining the system's key components, their relationships, and the overall flow of data, and control.

Design, in addition to technical concerns, has a project management perspective. It entails planning and estimating the design phase’s resources, timeframes, and hazards.It guarantees that the design process is well-organized, efficient, and in line with the project's overall goals and restrictions.

As a result, software design is critical in software engineering. It is essential for fostering quality in software development by providing accurate representations and models that can be evaluated for quality. A solid design is critical for constructing a stable system, facilitating effective testing, and ensuring the overall success of the software development process.

4.2 System Architecture

4.2.1 Pre-training//Transfer Learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Usually in pre-trained models, core convolutional layers are trained on the larger dataset and their weights are fixed. After this, the final layers of the network (dense fully connected layers) are fine-tuned using our target dataset (smaller) Due to scarcity of readily available data for our project, we are also using the application of Transfer Learning by pre-training our model on Image Net Dataset Image Net Dataset : Image Net is an image database organized according to the Word Net hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The total number of images in the dataset are over 14 million distributed over more than 20,000 categories (or labels).

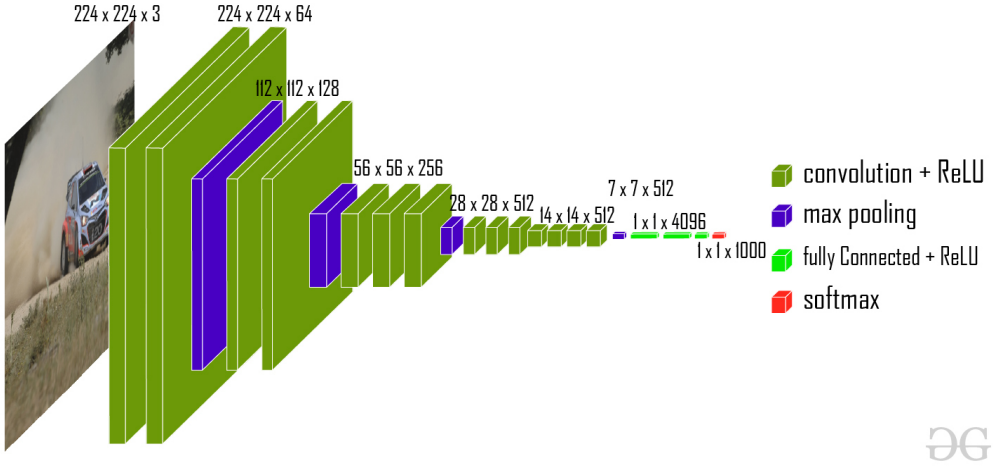


Fig 6: VGG16 Architecture

VGG16 Network VGG16 is a convolution neural net (CNN) architecture which was used to win SVR (Image Net) competition in 2014. Most unique characteristic about VGG16 is that instead of having a large number of hyper-parameters it focuses on having convolution layers of 3x3 filter with stride 1 and always using the same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC (fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it having 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx.) parameters. The final layers of the network are kept dynamic so that the model can take advantage of transfer

learning while at the same time get fine-tuned for the actual target (drowsiness detection) by training the fully connected dense layers.

- Input to the model: (3 x 224 x 224)
- Output of ConvLayers: (4 x 4 x 512)
- Input to Fully Connected Layers: (1 x 8192)

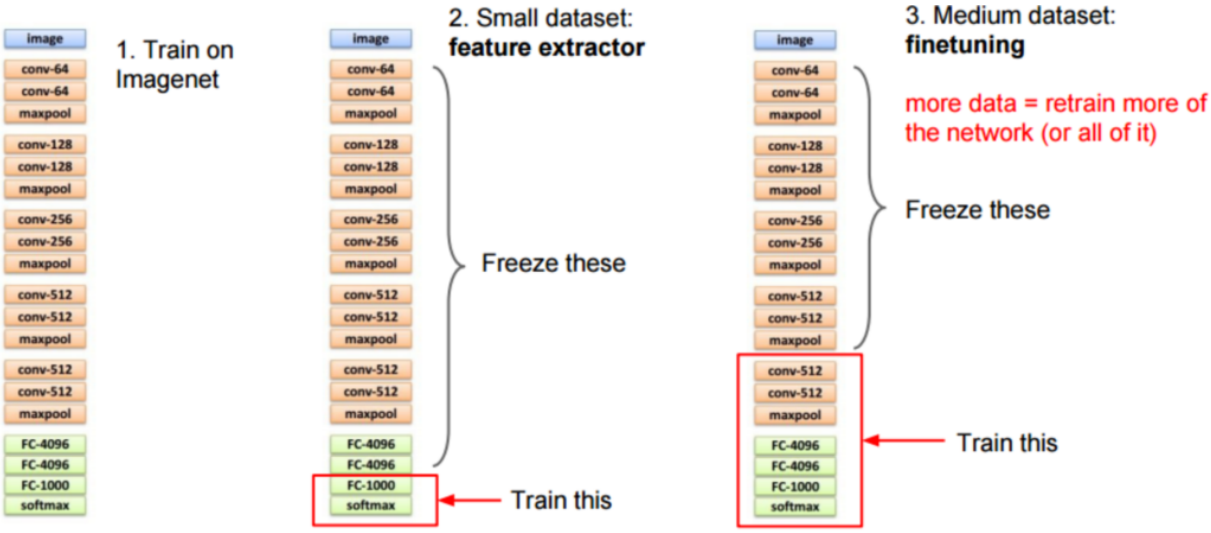


Fig 7: Fine tuning approach

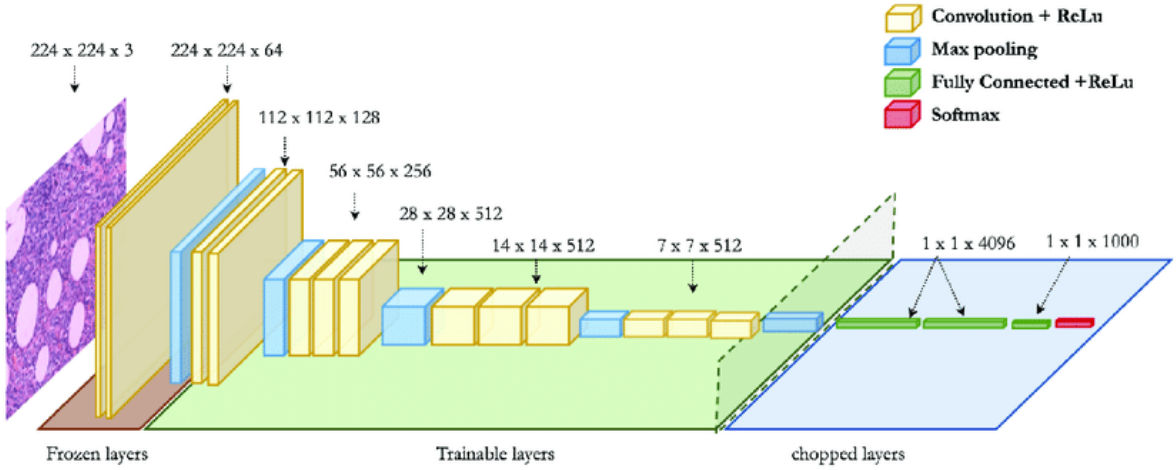


Fig 8: Representation of fine-tuned VGG16 architecture

4.3 UML Diagrams

4.3.1 Use Case Diagram

Use case diagrams are used to represent the dynamic behavior of a system.It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. The actor here is the driver and the camera used for real time capturing of images of eyes and mouth. If drowsiness is detected it gives alarm and if not again starts with capturings of images.

An illustration of the interactions between users or external systems and a system under investigation is called a use case diagram. Here is a use case diagram for drowsiness detection System and alert System.

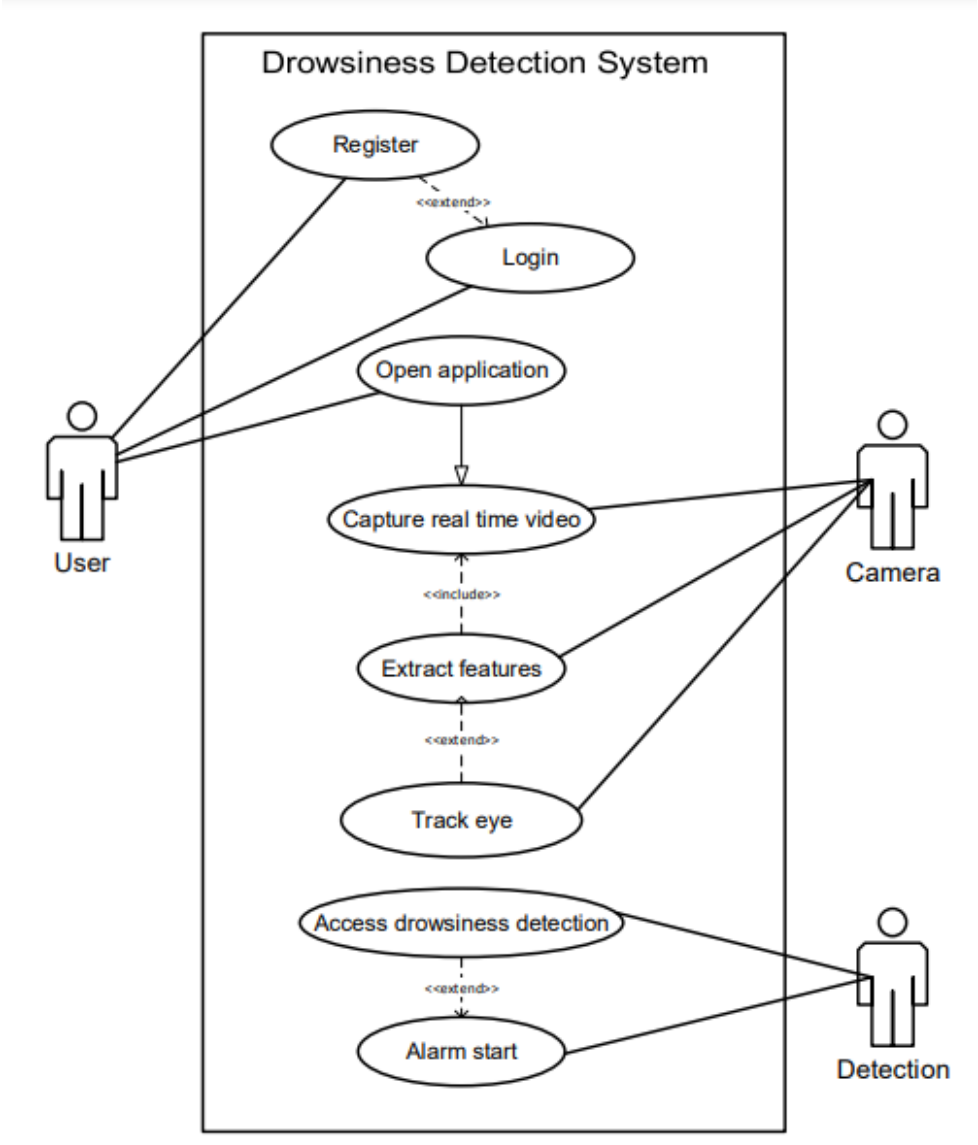


Fig 9: Use Case Diagram

4.3.2 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

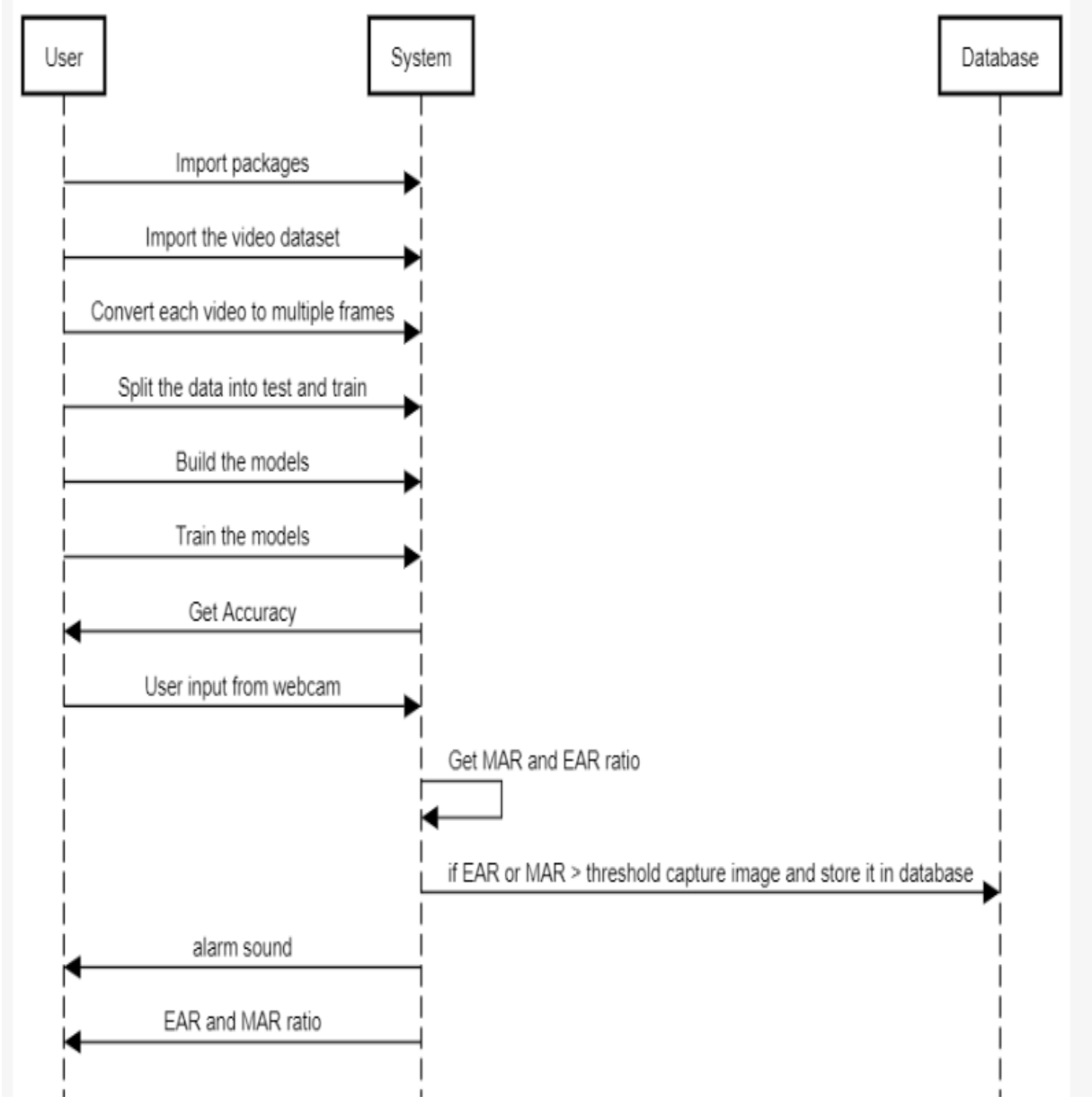


Fig 10: Sequence diagram

CHAPTER 5
ALGORITHMS

5. ALGORITHMS

5.1 VGG-16

VGG Architecture: The input to the network is an image of dimensions *(224, 224, 3)*. The first two layers have *64* channels of a *3*3* filter size and the same padding. Then after a max pool layer of stride *(2, 2)*, two layers have convolution layers of 128 filter size and filter size *(3, 3)*. This is followed by a max-pooling layer of stride *(2, 2)* which is the same as the previous layer. Then there are *2* convolution layers of filter size *(3, 3)* and *256* filters. After that, there are *2* sets of *3* convolution layers and a max pool layer. Each has *512* filters of *(3, 3)* size with the same padding. This image is then passed to the stack of two convolution layers. In these convolution and max-pooling layers, the filters we use are of the size *3*3* instead of *11*11* in AlexNet and *7*7* in ZF-Net. In some of the layers, it also uses *1*1* pixel which is used to manipulate the number of input channels. There is a padding of *1-pixel* (same padding) done after each convolution layer to prevent the spatial feature of the image.

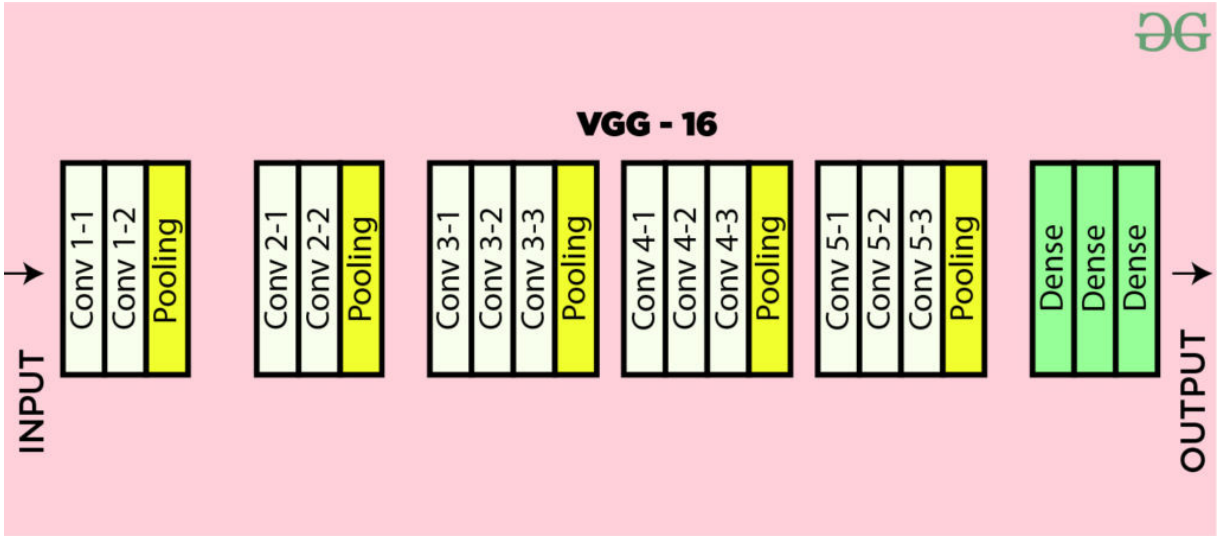


Fig 11: VGG-16 architecture Map

After the stack of convolution and max-pooling layer, we got a *(7, 7, 512)* feature map. We flatten this output to make it a *(1, 25088)* feature vector. After this there is *3 fully* connected layer, the first layer takes input from the last feature vector and outputs a *(1, 4096)* vector, the second layer also outputs a vector of size *(1, 4096)* but the third layer output a *1000* channels for *1000* classes of ILSVRC challenge i.e. 3rd fully connected layer is used to implement softmax function to classify 1000 classes. All the hidden layers use ReLU as its activation function. ReLU is more computationally efficient because it results in faster learning and it also decreases the likelihood of vanishing gradient problems.

Configuration: The table below listed different VGG architectures. We can see that there are 2 versions of VGG-16 (C and D). There is not much difference between them except for one that except for some convolution layers, *(3, 3)* filter size convolution is used instead of *(1, 1)*. These two contain *134* million and *138* million parameters respectively.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 12: Different VGG Configuration

Object Localization In Image: To perform localization, we need to replace the class score by bounding box location coordinates. A bounding box location is represented by the 4-D vector (center coordinates(x,y), height, width). There are two versions of localization architecture, one is a bounding box shared among different candidates (the output is *4* parameter vector) and the other is a bounding box is class-specific (the output is *4000* parameter vector). The paper experimented with both approaches on VGG -16 (D) architecture. Here we also need to change loss from classification loss to regression loss functions (such as MSE) that penalize the deviation of predicted loss from the ground truth.

Results: VGG-16 was one of the best performing architectures in the ILSVRC challenge 2014.It was the runner up in the classification task with a top-5 classification error of *7.32%* (only behind GoogLeNet with a classification error of *6.66%*). It was also the winner of the localization task with *25.32%* localization error.

Limitations Of VGG 16:

- It is very slow to train (the original VGG model was trained on the Nvidia Titan GPU for 2-3 weeks).
- The size of VGG-16 trained imageNet weights is *528* MB. So, it takes quite a lot of disk space and bandwidth which makes it inefficient.
- 138 million parameters lead to exploding gradients problem.

Fine-tuning is a simple method. It uses an already trained network and re-trains part of that by the new data set.

In the image below, each line means the weight. In this case, the neural network architecture which is made by red lines and blue nodes is fixed and the red line, meaning weight, is already trained by huge amounts of data. You can add one or some layers just after the architecture and train just those parts(sometimes including some layers before).

CHAPTER 6
IMPLEMENTATION

6. IMPLEMENTATION

6.1 Backend Approach :

Importing Libraries: The essential libraries, such as pandas, tensorflow matplotlib, numpy, os, sklearn, PIL, and opencv modules

Setting up dataset paths: The config is defined to set the path location for the extracted frames from the video dataset.

Splitting the data into train and test: It's usual practice in machine learning to divide a dataset into training and testing sets in order to assess a model's performance. The testing set is used to assess the model's performance on untested data, while the training set is used to fit the model's parameters.

Building the models: As from the problem statement, we are going to build two different models. the first model is the Baseline model developed from scratch(i.e., no transfer learning) and the second is the fine tuned VGG16 model developed in order to reduce the testing time and improve the accuracy of the model.

Model Training: The model is trained iteratively across a number of epochs. Each epoch entails dividing the training data into smaller batches and making predictions on these batches. The optimization technique computes the loss and adjusts the model's parameters based on the gradients of the loss with respect to the parameters.
The validation set is used to evaluate the model's performance after each epoch or a defined number of iterations. Metrics such as accuracy are generated to assess the model's ability to generalize to previously unseen data. Monitoring these indicators aids in spotting overfitting and underfitting and directs model modifications.

Get Accuracy : Compare the baseline model accuracy and the VGG16 fine tuning approach accuracy. Since VGG-16 is a pre-trained model, we get more accuracy to the VGG-16 than compared to baseline model. If there is any dispensary in accuracy then we need to train the model again. The accuracy for the baseline model is low due to overfitting problem.

6.2 Frontend Approach :

The essential modules and dependencies like tkinter, flask, cv2, keras, numpy, are to be imported successfully.

Flask : The Flask(__name__) command creates an instance of the Flask application. The folder in which uploaded files will be saved is defined by the Upload_folder variable, which is set.

Route Definition : The @app.route decorator is used to define the route. Each route links the "index.html" template is displayed when a user navigates to the homepage or starting page. The "conntact.html" template is used to contact the owner.

After all these steps have been completed successfully, you will see a web page opening up in the browser. Now you are free to explore the system.

6.2.1 Working Details:

The basic thing about drowsiness detection is pretty simple. We first detect a face using dlib's frontal face detector. Once the face is detected , we try to detect the facial landmarks in the face using the dlib's landmark predictor. The landmark predictor returns 68 (x, y) coordinates representing different regions of the face, namely - mouth, left eyebrow, right eyebrow, right eye, left eye, nose and jaw. Ofcourse, we don't need all the landmarks, here we need to extract only the eye and the mouth region.

Now, after extracting the landmarks we calculate the Eye Aspect Ratio (EAR) as:

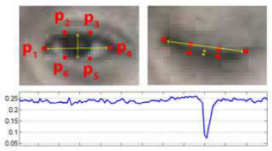
```
def eye_aspect_ratio(eye):
        # Vertical eye landmarks
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])
        # Horizontal eye landmarks
        C = dist.euclidean(eye[0], eye[3])

        # The EAR Equation
        EAR = (A + B) / (2.0 * C)
        return EAR
```

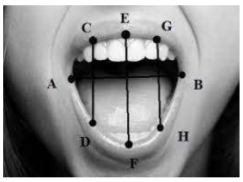
In the same way I have calculated the aspect ratio for the mouth to detect if a person is yawning, Although, there is no specific metric for calculating this, so | have taken for points, 2 each from the upper and lower lip and calculated the mean distance between them as:

```
def mouth_aspect_ratio(mouth):
        A = dist.euclidean(mouth[13], mouth[19])
        B = dist.euclidean(mouth[14], mouth[18])
        C = dist.euclidean(mouth[15], mouth[17])

        MAR = (A + B + C) / 3.0
        return MAR
```



Eye Aspect Ratio (EAR)



Mouth Aspect ratio (MAR)

6.3 Output Screenshots:

The result of the comparative model : base model vs VGG-16 Fine tuning approach model

Architecture of Baseline and VGG-16 fine tuning approach model:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 1, 148, 32)	43232
activation_1 (Activation)	(None, 1, 148, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 1, 74, 32)	0
conv2d_2 (Conv2D)	(None, 1, 74, 32)	9248
activation_2 (Activation)	(None, 1, 74, 32)	0
max_pooling2d_2 (MaxPooling2	(None, 1, 37, 32)	0
conv2d_3 (Conv2D)	(None, 1, 37, 64)	18496
activation_3 (Activation)	(None, 1, 37, 64)	0
max_pooling2d_3 (MaxPooling2	(None, 1, 19, 64)	0
flatten_1 (Flatten)	(None, 1216)	0
dense_1 (Dense)	(None, 64)	77888
activation_4 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65
activation_5 (Activation)	(None, 1)	0

Total params: 148,929
Trainable params: 148,929
Non-trainable params: 0

Fig 13: Architecture of Base Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_4 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                524352
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65
=================================================================
Total params: 15,239,105
Trainable params: 524,417
Non-trainable params: 14,714,688
_________________________________________________________________
```

Fig 14: Architecture of VGG-16 Fine tuning approach model

Accuracy:

The accuracy for the VGG-16 model has more accuracy than compared to the Base model since it is a predefined model.

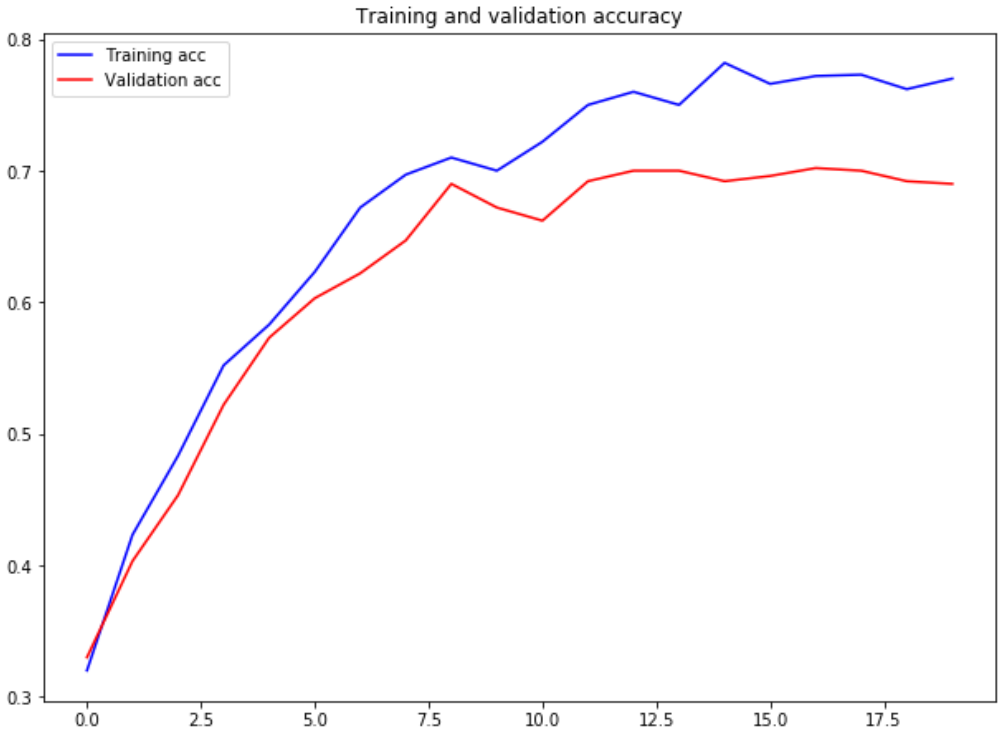


Fig 15: Accuracy for the Base model

Fig 16: Accuracy for VGG-16 fine tuning pre-trained model

When the user open the system the user navigates to the homepage of the website. From the homepage user gets the clear picture of the website. Inorder to get the clear idea of the website user can to the user manual where the clear description how this website works is given.

After running the program, the link is displayed coy the link on the browser.

```
(base) C:\Users\Windows 10\Documents\project>python app1.py
 * Serving Flask app "app1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 102-739-413
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig 17: Running the application

The GUI has been created using basic HTML, CSS and JavaScript and we have used Flask to render the python code into the website. Tkinter has also been used in order to make things simpler. It has 2 buttons: Run and Exit. The GUI looks like:

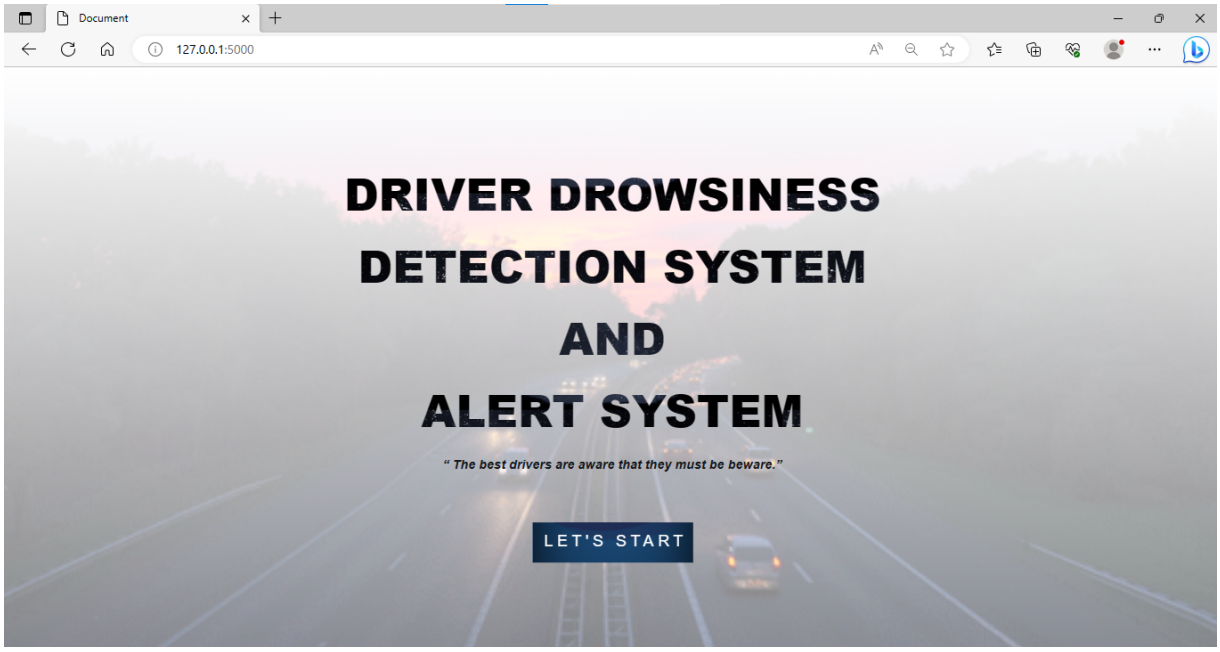
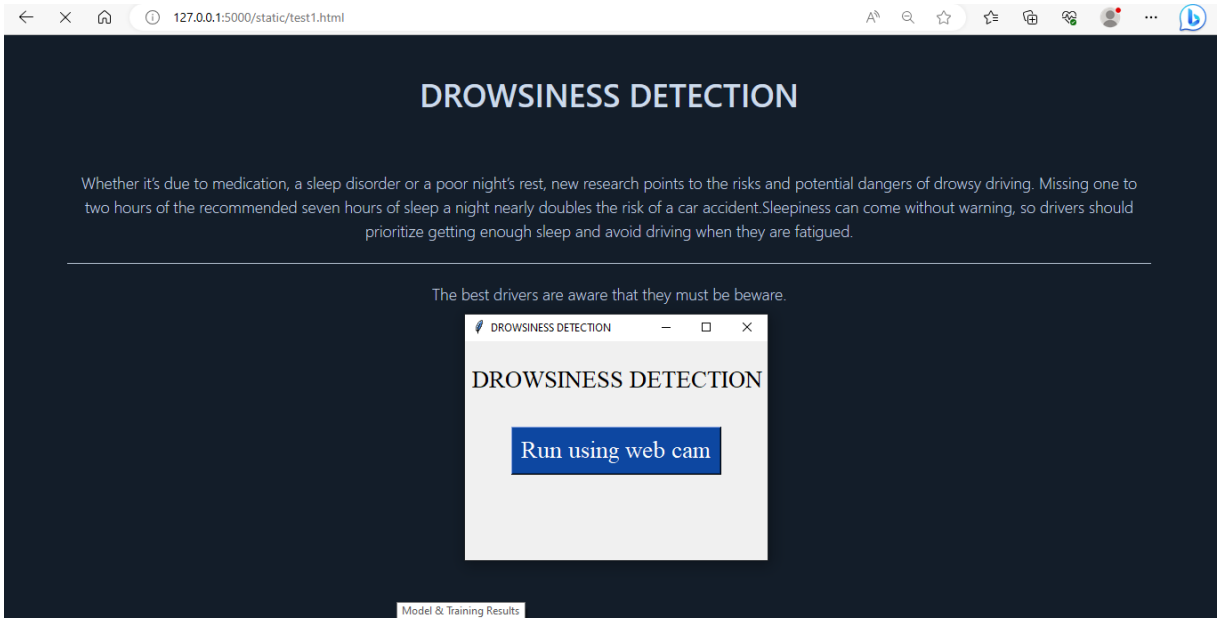


Fig 18: HomePage of the website



Click the webcam option inorder to open the webcam for live streaming.

The outputs of the working system detecting drowsiness is shown as:

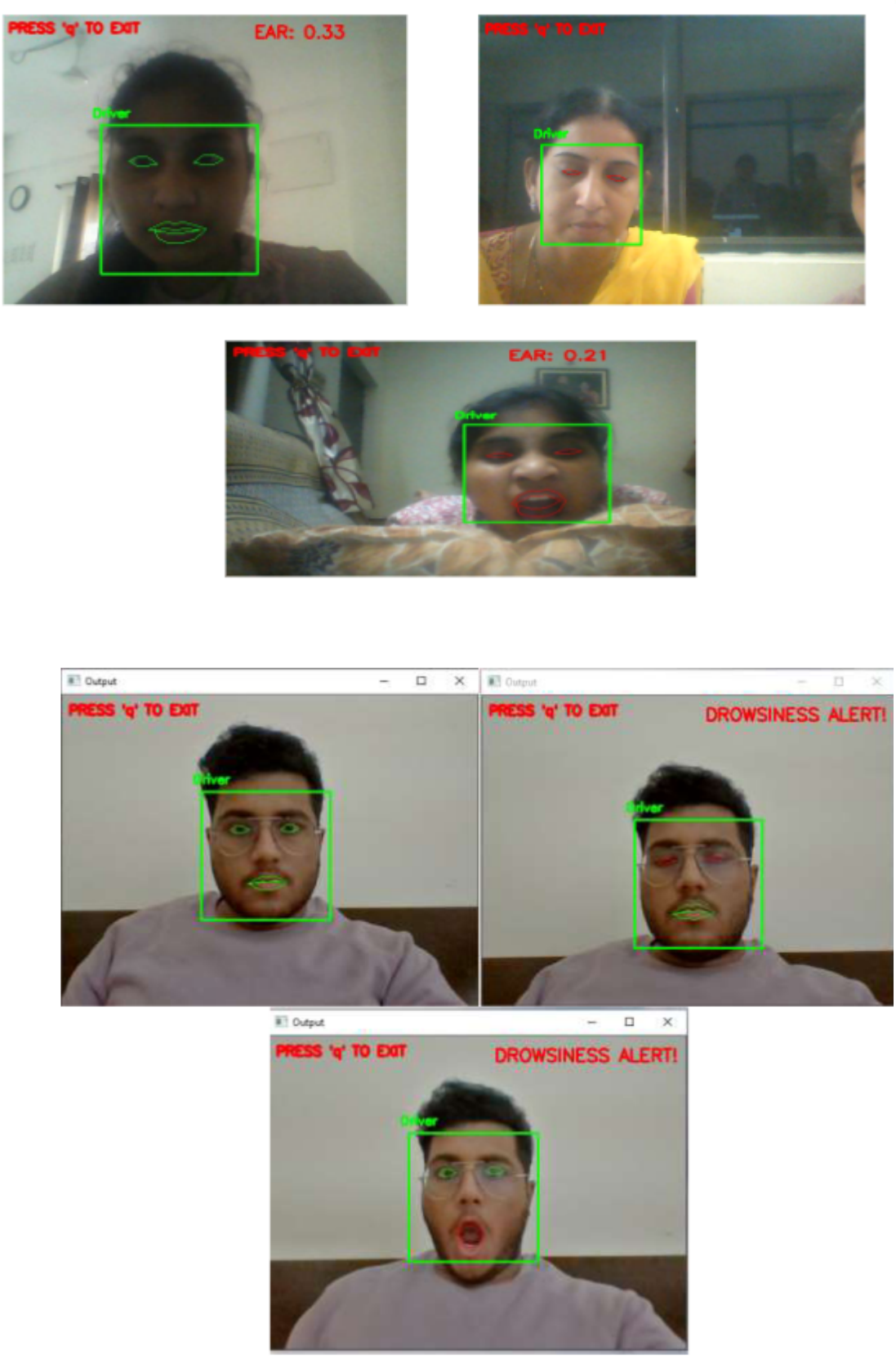


Fig 19:Outputs of the working system detecting drowsiness

Also, in order to keep a proof of the moment when the person was sleeping or yawning, we kept a separate folder dataset where those frames are stored as:

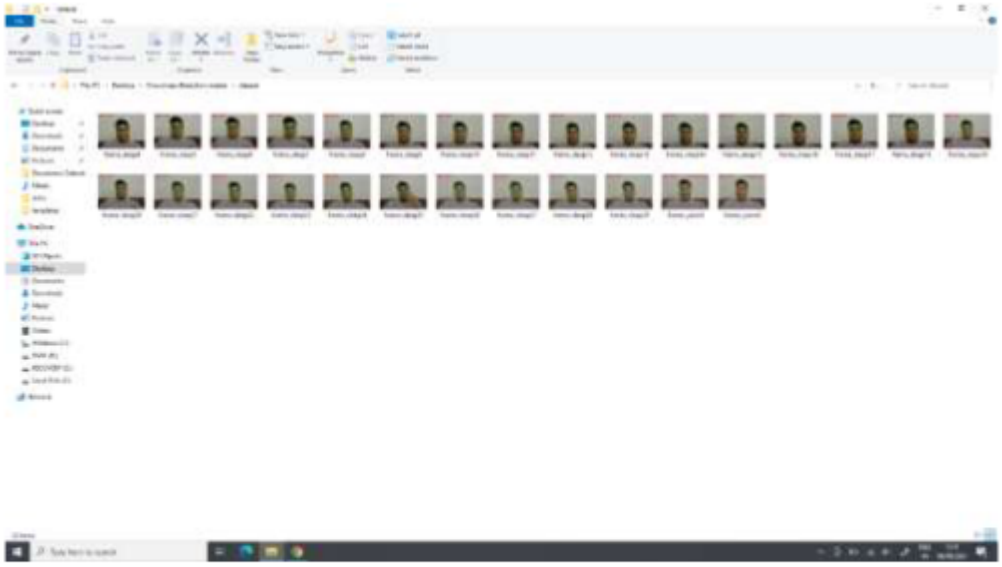


Fig 20: Folder for storing images

Also, we have tried plotting the MAR and EAR graph Vs. Time in order to make the working clearer to the audience. The graph looks like:

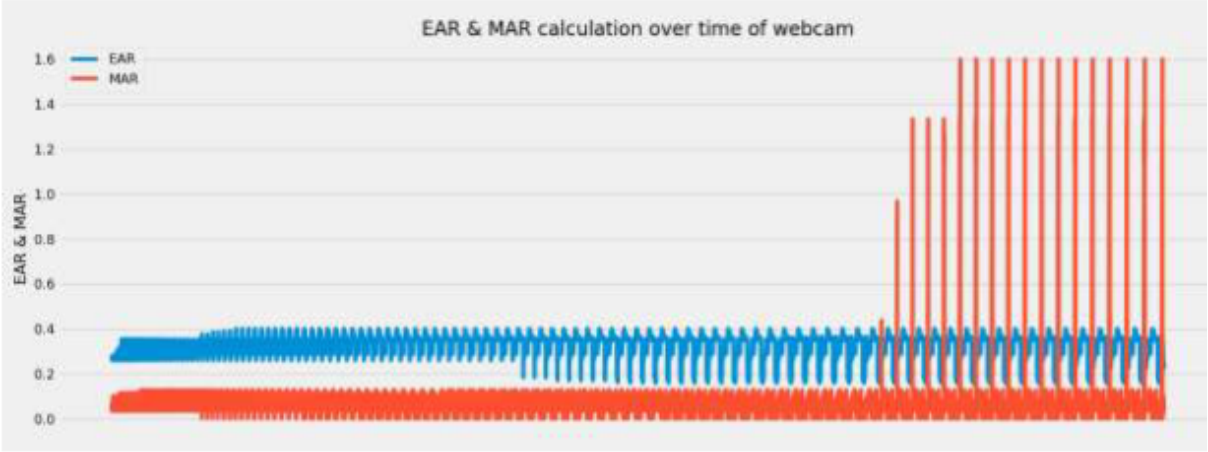


Fig 21: EAR & MAR Ratio

CHAPTER 7
CONCLUSION AND FUTURE

\

7. CONCLUSION AND FUTURE SCOPE

A non-invasive system to localize the eyes and monitor fatigue was developed. Information about the eye's position is obtained through self-developed image processing algorithms. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for too long, a warning signal is issued.

In the future,we will investigate the interaction of heart rate measurements,low mental workload during driving simulation and fatigue as well as perform numerical analysis of tractor accidents using a driving simulator

Moving forward, there are a few things we can do to further improve our results and fine-tune the models. First, we need to incorporate distance between the facial landmarks to account for any movement by the subject in the video. Realistically the participants will not be static on the screen and we believe sudden movements by the participant may signal drowsiness or waking up from micro-sleep. Second, we want to update parameters with our more complex models (NNs, ensembles, etc.) in order to achieve better results. Third and finally, we would like to collect our own training data from a larger sample of participants (more data!!!) while including new distinct signals of drowsiness like sudden head movement, hand movement, or even tracking eye movements.

Normalization was crucial to our performance. We recognized that everybody has a different baseline for eye and mouth aspect ratios and normalizing for each participant was necessary. Outside of runtime for our models, data pre-processing and feature extraction/normalization took up a bulk of our time. It will be interesting to update our project and look into how we can decrease the false-negative rate for kNN and other simpler models.

CHAPTER 8

REFERENCES

8. REFERENCES

[1] Rateb Jabbar, Khalifa Al-Khalifa, Mohamed Kharbeche, Wael Alhajyaseen, Mohsen Jafari, Shan Jiang "Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques" proceedings for 9th international conference on Ambient Systems, Networks, and Technologies.

[2] Mohit Dua, Shakshi, Ritu Singla, Saumya Raj, Arti Jangra "Deep CNN models-based ensemble approach to driver drowsiness detection " Neural Computing and Applications **33**, 3155–3168 (2021).

[3] Mkhuseli Ngxande, Jules-Raymond, Michael Burke "Driver drowsiness detection using Behavioral measures and machine learning techniques: A review of state-of-art techniques" Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)

[4] N. C. for Statistics and Analysis, "Crash Stats: Drowsy Driving 2015," October 2017. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/ Public/ViewPublication/812446.

[5] M. Walker, Why We Sleep: Unlocking the Power of Sleep and Dreams. Scribner, 2017. [Online]. Available: https://books.google.com.qa/books? id=8bSuDgAAQBAJ.

[6] W. H. Organization, "The top 10 causes of death." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/ the-top-10-causes-of-death.

[7] M. Benz, "Mercedes benz safety - s class." [Online]. Available: https://www.mercedes-benz.co.uk/passengercars/mercedes-benz-cars/ models/s-class/saloon-w222/safety/intelligent-drive.module.html.

[8] A. D. McDonald, J. D. Lee, C. Schwarz, and T. L. Brown, "A contextual and temporal algorithm for driver drowsiness detection," Accident Analysis & Prevention, vol. 113, pp. 25–37,Apr.2018.[Online].Available: https://www.sciencedirect.com/science/article/pii/S0001457518300058.

[9] U. S. F. M. C. S. A. T. Division, “PERCLOS: A Valid Psychophysiological Measure of Alertness As Assessed by Psychomotor Vigilance,” October 1998.

[10] C. S. Wei, Y. T. Wang, C. T. Lin, and T. P. Jung, “Toward Drowsiness Detection Using Non-hair-Bearing EEG-Based Brain-Computer Interfaces,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2018.

[11] V. J. Kartsch, S. Benatti, P. D. Schiavone, D. Rossi, and L. Benini, “A sensor fusion approach for drowsiness detection in wearable ultra-low-power systems,” Information Fusion, vol. 43, pp. 66–76, Sep. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/ pii/S1566253517306942.

[12] Dang, K., Sharma, S. (2017). Review and comparison of face detection algorithms. 7th International Conference on Cloud Computing, Data Science & Engineering, pp. 629-633. https://doi.org/10.1109/confluence.2017.7943228

[13] Park, S., Pan, F., Kang, S., Yoo, C.D. (2017). Driver drowsiness detection system based on feature representation leaming using various deep networks. In Chen CS., Lu J, Ma KK. (eds) Computer Vision - ACCV 2016 Workshops, Lecture Notes in Computer Science, pp. 154-164. https:/doi.org/10.1007/978-3- 319-54526-4 12

[14] Picot, A., Charbonnier, S., Caplier, A. (2012). On-Line Detection of drowsiness using brain and visual information. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 42(3): 764- 775. https://doi.org/10.1109/tsmca.2011.2164242

[15] Krajewski, J., Sommer, D., Trutschel, U., Edwards, D., Golz, M. (2009). Steering wheel behavior based estimation of fatigue. The Fifth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, pp. 118-124. https://doi.org/10.17077/drivingassessment.1311

[16] J. Baraniak, J. Hauer, N. Schuhmann and G. Leugering, "Implementation of Adaptive Filters for Biomedical Applications", *IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering; SIBIRCON 2008*, 2008.

[17] G. Borghini, L. Astolfi, G. Vecchiato, D. Mattia and F. Babiloni, "Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload fatigue and drowsiness", *Neurosci Biobehav Rev*, vol. 44, pp. 58-75, Ju1 2014.

[18] P. M. Tekade and S. Gawali, "Investigation and New Method of No intrusive Detection of Driver Drowsiness", *International Journal of Engineering and Innovative Technology*, vol. 1, pp. 210-216, May 2012.

[19] S. Tateno, X. Guan, R. Cao and Z. Qu, "Development of Drowsiness Detection System Based on Respiration Changes Using Heart Rate Monitoring", *2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan SICE 2018*, pp. 1664-1669, 2018.

[20] L. Celona, L. Mammana, S. Bianco and R. Schettini, "A multi-task CNN framework for driver face monitoring", *IEEE International Conference on Consumer Electronics - Berlin ICCE-Berlin*, vol. 2018-Septe, pp. 1-4, 2018.

[21] R. Jabbar, K. AI-Khalifa, M. Kharbeche, W. Alhajyasen, M. Jafari and S. Jiang, "Applied internet of things iot: Car monitoring system for modeling of road safety and traffic system in the state of qatar", *Qatar Foundation Annual Research Conference Proceedings*, vol. 3, pp. ICTPP1072, 2018.

[22] S. Ren, K. He, R. Girshick, J. Sun, C. Cortes, N. D. Lawrence, et al., "Faster r-cnn: Towards real-time object detection with region proposal networks" in Advances in Neural Information Processing Systems, Curran Associates, Inc., vol. 28, pp. 91-99, 2015.

[23] Y. Xie, K. Chen and Y. L. Murphey, "Real-time and Robust Driver Yawning Detection with Deep Neural Networks", *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence SSCI 2018*, pp. 532-538, 2019.

[24] Sathasivam, S., Mahamad, A.K., Saon, S., Sidek, A., Som, M.M., Ameen, H.A.: Drowsiness detection system using eye aspect ratio technique. In 2020 IEEE Student Conference on Research and Development (SCOReD) (2020).