B. TECH. PROJECT REPORT

On

Development of Intelligent Drowsiness Detection System using Artificial Intelligence

BY

Kaustubh Mishra (1816055)

Yuvraj Kumar(1816052)

Sudhanshu Vidyarthi(1816057)

DEPARTMENT OF ELECTRONICS & INSTRUMENTATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

May 2022

Development of Intelligent Drowsiness Detection System using Artificial Intelligence

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of
BACHELOR OF TECHNOLOGY
in

ELECTRONICS & INSTRUMENTATION ENGINEERING

Submitted by:
Kaustubh Mishra (1816055)
Yuvraj Kumar (1816052)
Sudhanshu Vidyarthi(1816057)

Guided by:
Dr. Sudarsan Sahoo
Assistant Professor

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
May 2022

DECLARATION

We hereby declare that the project entitled "Development of Intelligent Drowsiness Detection System using Artificial Intelligence" submitted in partial fulfillment for the award of the degree of Bachelor of Technology under the supervision of Dr. Sudarsan Sahoo, Assistant Professor in the Department of Electronics & Instrumentation Engineering, NIT Silchar is an authentic work.

Further, I/we declare that I/we have not submitted this work for the award of any other degree elsewhere.

Kaustubh Mishra(1816055)

Yuvraj Kumar(1816052)

Sudhanshu Vidyarthi(1816057)

Date:-18th May,2022

CERTIFICATE

It is certified that the work contained in this thesis entitled 'Development of Intelligent Drowsiness Detection System using Artificial Intelligence' submitted by Kaustubh Mishra, Yuvraj Kumar and Sudhanshu Vidyarthi, with Scholar IDs : 1816055, 1816052, 1816057 respectively, for the award of B.Tech. degree is based on their own work carried out under my supervision.

Dr. Sudarsan Sahoo,
Assistant Professor

Acknowledgement

An endeavor over a long period requires advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We wish to express our sincere thanks and gratitude to our project guide Dr. Sudarsan Sahoo, Assistant Professor, Electronics and Instrumentation Department, NITS, for the simulating discussions, in analyzing problems associated with our project work and for guiding us throughout the project. Project meeting were highly informative. We express our sincere thanks for the encouragement, untiring guidance and the confidence he has shown in us.

We would also like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us in completing this project successfully.

Kaustubh Mishra(Scholar ID : 1816055)
Yuvraj Kumar (Scholar ID : 1816052)
Sudhanshu Vidyarthi (Scholar ID : 1816057)
B.Tech. IV Year
Department of Electronics & Instrumentation Engineering
NIT Silchar

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. This document is a semester end review report that details the technical aspects as well as our motivation behind the development of this project.

Table Of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Necessity

1.1.1 Technology and Human Psychology

Human beings have always looked for ways to increase the efficiency and output of their work. In order to do so, humans have always come up with new and better forms of technology that can aid and assist in their work. As technology advanced, so did the society and civilizations, and slowly but steadily technology infused with society such that today, technology can be seen in every aspect of human life. One such aspect is traveling and with the advancement in technology many different modes of transportation are available and our dependency on these modes is increasing exponentially. These methods of transportation have deeply affected our lives as we can travel between places in a very short span of time. Today, almost everyone uses one or other forms of transportation. Some use their own vehicles while others depend on public transport. However, there are some rules and codes of conduct that one needs to follow while driving these vehicles. One of the most basic codes of conduct is to stay vigilant and alert while driving.

Neglecting those rules for safer traveling has caused hundreds and thousands of accidents to occur on roads every year. The statistics show that such accidents and tragedies are increasing with each passing year. Therefore, It is evident that these rules and codes of conduct have been neglected by the society in general. It may seem trivial to many folks but these rules and regulations are of utmost importance While on the road, an automobile wields the most power and in irresponsible hands, it can be destructive and in some cases, that carelessness can cost the life of people on the road. One such carelessness is not admitting when we feel too sleepy and drowsy to drive. Thus, in order to monitor and prevent accidents or tragedies from occurring due to such negligence, many scientists and researchers from all over the world are working on devising various techniques and/or systems to detect drowsiness in a person while driving. But at times, some of the points and observations made are only limited to a specific scenario or not accurate enough. Thus, the aim of this project is to utilize the power of transfer learning and train a model to detect drowsiness in a person.

1.1.2 Facts and Statistics

According to the statistics from the WHO, distracted and fatigue driving is one of the main reasons behind road traffic accidents. Since road accidents are a major threat to human life, several reports put forwards by various national institutes like Central Road Research Institute(CRRI), Ministry of Road Transport and autonomous organizations like the SaveLife Foundation and Mahindra have revealed many interesting statistics regarding the road accidents due to drowsiness. Some of these stats are listed below.

- The study by CRRI revealed that 40% of road accidents on the Agra-Lucknow Expressway occur due to exhausted drivers feeling sleepy while driving.
- A study by Mahindra and Savelife Foundation showed **that truck** drivers in India drive 12 hours a day covering around 417 km, with almost 50% of them admitting that they feel sleepy or tired while driving.
- According to the study by Lancet, about 28 million Indians suffer from sleep apnoea, which clearly leads to many drowsy drovers on Indian roads.
- According to the Ministry of Road Transport and Highways (MoRTH) findings, the socio-economic costs of road crashes in India is at ₹147,114 crore, which is equivalent to about 1% of the country's GDP.

Several countries are actively making laws to reduce accidents due to drowsiness . For example, In India , the drivers are usually paired or sometimes, even in groups of 3-4 , for long travel journeys and each driver takes turns driving the vehicle . In Europe, the law requires that "Drivers should stop and rest for every 4.5 hours of continuous driving, and the rest period should be no less than 20 minutes". It is subjective to determine whether the driver is in fatigue state or not without sufficient quantified indexes and reliable data analysis. Thus, our project relies on a trained model to analyze the images and/or frames and determine whether the person is drowsy or not in real-time.

The figure below shows the fatality vs year plot and can be seen clearly that no. of deaths are increasing with each passing year. These deaths are primarily due to accidents involving fatigue or exhausted drivers.

Fig 1.1: No. of Deaths Vs Year (courtesy : TOI)

1.2 Background Study

Upon reading the research papers listed in the reference section, the study and development of driver drowsiness systems can be broadly divided into categories :

1. *Sensor based techniques* : These techniques as the name suggest employ the use of various sensors and equipments and can further be divided into two sub-categories :
   a. Techniques that record Mechanical motions like vibrations, wheel movement and Lane positions
   b. The techniques that record and analyze heart rate, muscle contraction, temperature, blood-flow, etc of the driver/subject/participant.

2. *Vision Based Techniques* : These techniques primarily utilize the camera to **analyze** the person and perform operations in real-time. These techniques try to understand the changes in the facial region of the person as frames change. In our project, we are developing a vision based technique using deep learning architecture.

The gaps mentioned in the table below signify the difference between our approach and the approach used by the researchers in those papers. The following Table 1.0 **summarizes** the research papers mentioned in the reference section :

Table 1.1 : Research paper Summary

S.No	Paper title	Outcome	Gaps
1.	Detecting Driver Drowsiness Based on Sensors. Arun Sahayadhas, Kenneth Sundaraj, Murugappan Murugappan, Sensors (Basel) 2012 Dec; 12(12): 16937– 16953. Published online 2012 Dec 7. doi: 10.3390/s121216937	Three measures for monitoring Drowsiness(Sensor Based)	Described only about the sensor techniques, no description about vision techniques.
2.	A Context-Aware EEG Headset System for Early Detection of Driver Drowsiness. Gang Li, Wan-Young Chung, Sensors (Basel) 2015 Aug; 15(8): 20873–20893. Published online 2015 Aug 21. doi: 10.3390/s150820873.	1. Successful Implementation of Drowsiness Detection System using EEG, gyroscope and mobile application. 2. EEG results of drowsiness, defined slightly drowsy events.	Only employed supervised machine learning model: SVM classifier, during classification(final) stage. The model was pretrained.
3.	Feng You, Yunbo Gong, Haiqing Tu, Jianzhong Liang, Haiwei Wang, "A Fatigue Driving Detection Algorithm Based on Facial Motion Information Entropy", Journal of Advanced Transportation, vol. 2020, Article ID 8851485, 17 pages, 2020. https://doi.org/10.1155/2020/8851485.	1. Developed Face Feature Vector and Face Feature Triangle 2. Developed a prototype of the final model.	Used tiny YOLO-v3 deep learning model. This model utilises convolutional layers but is different from CNN as whole image is read at once unlike in strides as in CNNs.

4.	Charlotte Jacobé de Naurois, Christophe Bourdin, Anca Stratulat, Emmanuelle Diaz, Jean-Louis Vercher, Detection and prediction of driver drowsiness using artificial neural network models, Accident Analysis & Prevention,Volume 126, 2019,Pages 95-104,ISSN 0001-4575, https://doi.org/10.1016/j.aap.2017.11.038.	Employed different ANN models for the prediction/detection of drowsiness and summarized about the features like eyelid closure, gaze, head movement and driving time, most useful and used by the best prediction models.	Didn't study the models on continuum of images and/or real-time video input.

1.3 Intended Audience

The intended audience for this project are the development team, the project evaluation panel, and the tech-savvy enthusiasts who wish to learn from the project and/or provide feedback on the project.

1.4 Problem Definition

Fatigue as a safety problem is yet to be deeply tackled by any country in the world mainly because of its nature. Though several countries have started addressing this issue and some have even established certain laws and codes of conduct to be followed, but the success rate is limited as often people are either unaware of such laws or at times are able to circumvent through these rules. Also fatigue, in general, is very difficult to measure or observe unlike other issues which have clear tests and key indicators available easily. This is mainly due to the subjective nature of Fatigue. Probably, the best solutions available now are the awareness about the fatigue-related accidents and promoting amongst drivers the notion to admit fatigue when needed. The latter solution is dependent on the success of the former whereas the success of the former is quite limited. Thus, this necessitates the development of a detection system that can alert the user in-case the user is found drowsy or sleepy while driving in real-time.

CHAPTER 2

Literature Survey

This survey gives a brief information of the technology used so that any interested individual can understand the prerequisites required to understand the project as certain terms are specific to the technology used.

2.1 Technology Used

a. PYTHON : Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of indentation. The language constructs and object oriented approach aids the programmers in writing clear, clean and meaningful codes for small or large-scale projects. Python is dynamically typed and supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

b. JUPYTER LAB: Project Jupyter is a non-profit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming language.

c. IMAGE PROCESSING : In computing, digital image processing is the use of computer algorithm to perform image processing on digital images.

d. MACHINE LEARNING : Machine Learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine Learning algorithms build a mathematical model based on sample data, known as "training data" in order to make predictions or decisions without being explicitly told.

e. TENSORFLOW : TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java.

f. Kaggle : Kaggle is an online community of machine learning enthusiasts and provides its users with free Tensor Processing Units(TPU) and GPU capabilities.

g. Streamlit : A Streamlit is an open source app framework in Python language. It helps to create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

CHAPTER 3

CONCEPT DESIGN

3.1 Objective

The objective for this project was to determine whether a person in real-time is drowsy or not depending upon whether his eyes are open or not and to play an alarm if the EAR(eye aspect ratio) falls below a threshold and MAR(mouth aspect ratio) above a threshold after a specific period of time.

3.2 Flowchart :

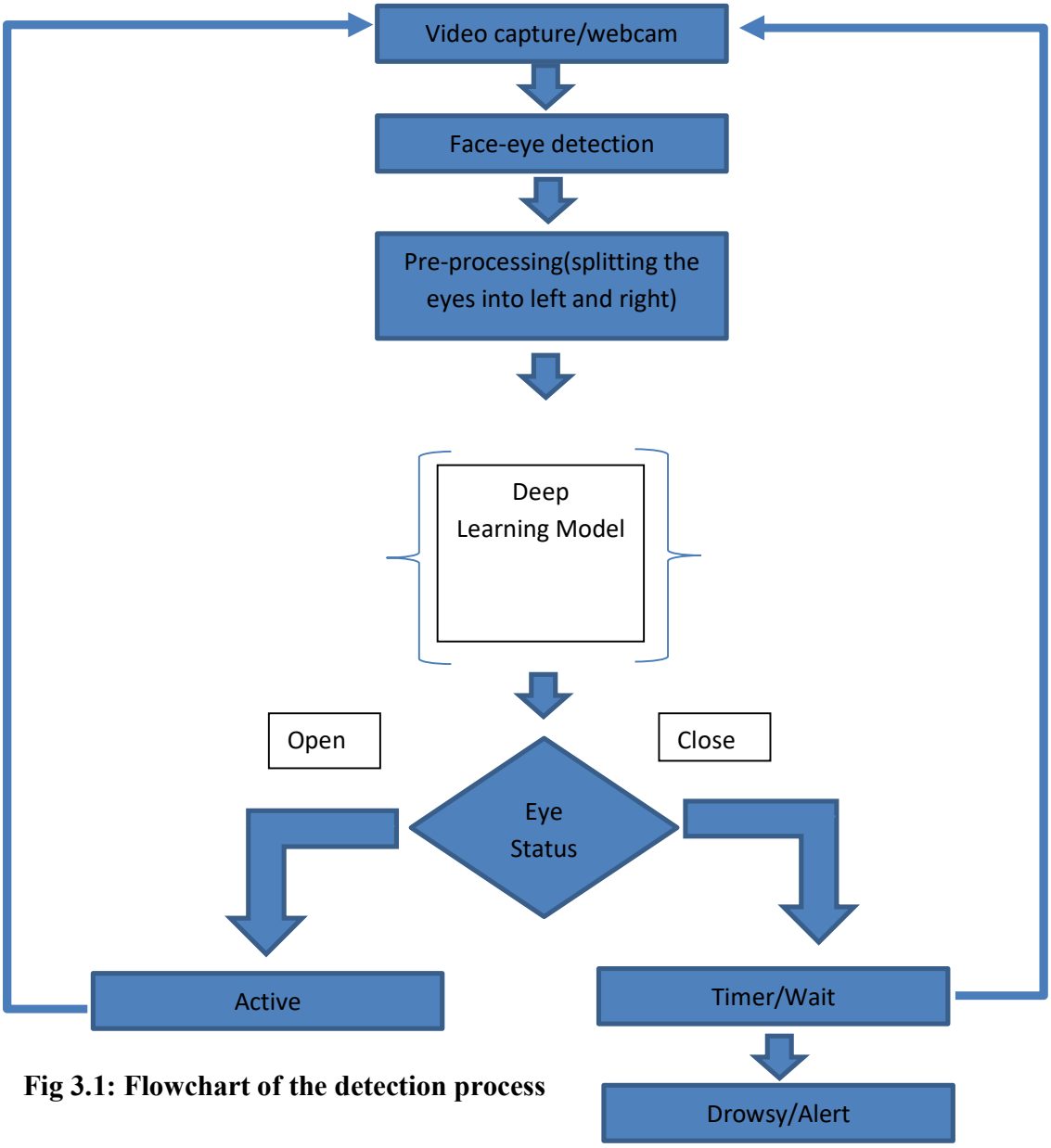

Fig 3.1: Flowchart of the detection process

This project utilizes opencv Library(in Python) which is aimed at providing real time computer vision. The project will also utilize dlib toolkit to allow for face and eye detection as well as Imutils package to allow for simple image manipulation functions to be available in our code. We will also create notebooks on kaggle as they provide online GPU & TPU capabilities.

Thus, the overall process is as follows :

Using opencv library to either capture faces in real-time or working on images/frames (during training). Then utlising dlib library or mediapipe library and Imutils package for image manipulation and preprocessing.

After preprocessing the data, customize and train the MobileNet model on the data to detect whether the eyes were open .

Then using the input to our customized CNN model to calculate EAR (eye-aspect ratio).

Then develop a set of conditions based on EAR, and classification results to determine whether the person is drowsy or not!

3.3 EAR CALCULATION

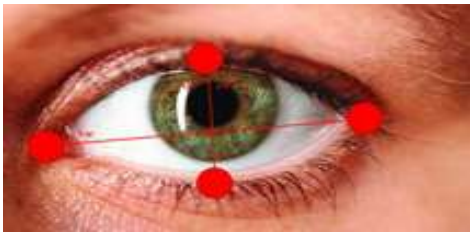

Fig 3.2 : EAR Calculation

EAR is a simple ratio between the distance from one end of the eye to other horizontally and vertically : See, Figure :

EAR = distance(eh)/distance(ew).....(1)

Where,

eh = vertical line

ew = horizontal line , Thus eyes closed have low EAR while open eyes have high EAR .

This method will be able to calculate the mathematical midpoint through these line equations, while detection can be used to locate the actual midpoint and the difference between these two points can also serve as a method for the alert system.

3.4 CNN

CNNs are feed forward neural networks that are trained through backward propagation. In machine learning, CNNs have been successfully employed for analyzing visual imagery.

CNN uses a variation of multilayer perceptrons designed to require minimal preprocessing. They have been sometimes also called shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

As described by Prof. Fei Fei Li (ex-Vice President Google and Chief Scientist of AI/Ml Google Cloud and Current Professor at Stanford and Co-director Stanford AI Lab) in her CS231 Convolutional Neural Networks Course at Stanford(in 2017) : A neural network is a black box in the sense that while it can approximate any function, studying its structure won't give you any insights on the structure of the function being approximated. But a basic understanding is required to understand the huge impact it has had on the Image Classification process since its inception.

CNN building Blocks :
Convolutional Layer
Activation Layer/Function
Pooling Layer
Batch Normalization Layers
Fully Connected Layers
Loss layers

CNN's design consists of Convolutional, pooling, fully connected and weights.

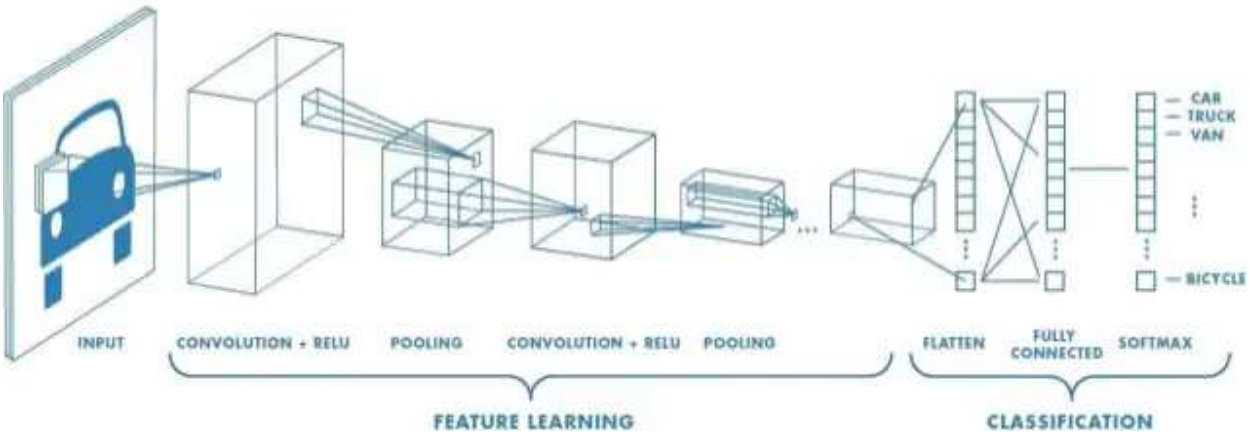

Fig 3.3 : CNN Architecture (courtesy : Mathworks.com)

3.5 MobileNet Architecture

MobileNet is a type of CNN architecture designed for mobile and embedded vision applications. These types of architectures are streamlined to utilize depth-wise separable convolutions to build lightweight neural networks that can have low latency for mobile and embedded devices. They are designed to maximize accuracy and efficiency while being mindful about the restricted resources or data for an on-device or embedded application. A standard mobile Net has 28 layers with approximately 4.2 million parameters which can be reduced by tuning width multiplier hyperparameters efficiently. It is also dependent upon the dimensions of the image. We customized the model to suit our needs and then trained the model on a training dataset.
The reason for working with a pretrained model is that the model will not start with some random association of weights but will utilize a set of standard weights.

We preferred MobileNet because of the following reasons :

1. It can be used for mobile and embedded vision applications.
2. It is the smallest model available, and provides good accuracy and predictions. Thus, helps us conserve space-time complexity.

Look at the Table showing the MobileNet Architecture :

Table 3.1 : Mobile Net Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

(courtesy : Research Team Cornell University)

3.6 Understanding the Likelihood Scenario

A bit of mathematics to understand the likelihood scenario. For simplicity, sigmoid function is used with a single input feature and single parameter(weight). Considering a simple binary classification(or Logistic Regression) process.

$$h_\theta(x) = Sigmoid(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1$$

$P(y=1|x;\theta) = h_\theta(x)$ *as at present we are performing simple binary classification between* 0 & 1

and $0 \le h_\theta(x) \le 1$ *just like probability*

$$So\ P(y=1|x;\theta) = 1 - P(y=1|x;\theta) = 1 - h_\theta(x) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 2$$

Therefore, Probability Function:

$$P(y|x;\theta) = \left(h_\theta(x)\right)^y \left(1-h_\theta(x)\right)^{1-y} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3$$

So over m examples or dataset values we have to maximize the probability on this dataset

So, we need to figure out the likelihood with the choosen θ.

$$L(\theta) = P(\vec{Y}|\vec{X};\vec{\theta}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 4$$

So this is the probability that needs to be maximized. Therefore, we can break this probability into products consisting of probability of each training example assuming each probability as independent.

$$L(\theta) = \prod_{i=1}^{m} P(y^i|x^i,\theta) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 5$$

$$L(\theta) = \prod_{i=1}^{m} \left(h_\theta(x^i)\right)^{y^i} \left(1-h_\theta(x^i)\right)^{1-y^i} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 6$$

$$\log(L(\theta)) = l(\theta) = \sum_{i=1}^{m} y^i \left(\log\left(h_\theta(x^i)\right)\right) + 1 - y^i \log\left(1-h_\theta(x^i)\right) \ldots\ldots\ldots\ldots 7$$

Continuing from the discussion of the livelihood, another important thing to notice is how will the likelihood reach maxima or minima, thus, it is important to learn about “learning-rate” and “optimiser”.

Learning-rate is simply by how much the model should take in new information.

Optimisers are basically added so that when the model is far from a global or most stable state, the stride or jump to gather new information is fast or high and when it is close to the stable state it becomes slow and the strides become small. The project is utilising “Adam” optimiser.

3.7 ADAM

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problems involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

Adam Optimizer inherits the strengths or the positive attributes of the above two methods and builds upon them to give a more optimized gradient descent.

Fig 3.4 showing global minimum and local minimum

Here, the rate of gradient descent is controlled in such a way that there is minimum oscillation when it reaches the global minimum while taking big enough steps (step-size) so as to pass the local minima hurdles along the way. Hence, combining the features of the above methods to reach the global minimum efficiently.

Mathematical Calculation :

```
Moment_1 = 0; moment_2 = 0;
For t in range(num_iterations):
        dx = compile_gradient(x)
        Moment_1 = beta1*moment_1 + (1-beta1)*dx
        Moment_2 = beta2*moment_2 + (1-beta2)*dx*dx
        Unbias_1 = Moment_1/(1-beta1**t)
        Unbias_2 = Moment_2/(1-beta2**t)
        X - = learning_rate*Unbias_1/(np.sqrt(Unbias_2)+ 1*e^(-7))
```

Finally, a good initial point as suggested after experimentation for calculation and fast culmination towards a stable state is beta1 = 0.9 and learning rate = 1e^(-3).

CHAPTER 4

EXPERIMENT AND ANALYSIS

4.1 Software Requirements Specification

- Anaconda (distribution manager of Python and data science related software packages as well as virtual environment manager with GUI and CLI navigator)
- Python (3.7.2)
- IDE : Jupyter Notebook

Libraries/Packages Used :

- tensorflow(2.1)
- Keras API (open-source library providing Python Interface for neural networking) (2.4.2)
- opencv(4.3)
- Dlib
- Imutils
- Numpy(1.20.2)
- Sklearn
- Playsound, etc
- **scipy**
- **streamlit**

Operating System :

- Windows or Ubuntu

Web Notebooks :

- Kaggle, Google Collaboratory

4.2 Data Collection

The project required a dataset that can offer close visualization of the human eye. Thus, upon research, we found an MRL data set compatible with the project. The MRL dataset was actually collected to work on training a model for eye disease classification. But we found this dataset to offer us close visualization of various human eyes, thus we decided to work on this data.

From the MRL dataset we differentiated the images into closed eyes and open eyes.

Since the data was very large about 358 mb with + 1,00,000 images, each with either 86x86 or 94x94 resolution so the model is trained using only some part of the data.

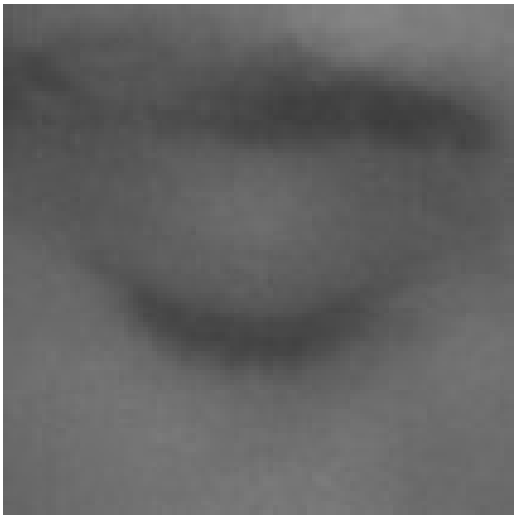

Fig 4.1 : Image in MRL Dataset

This is an image in the MRL dataset and as can be seen offers close visualization of the human eye.

4.3 Data Preprocessing

In this part, First the data is checked by looking at some of its images in Grayscale.

Then upon understanding the data, divided it into training and testing dataset. Next, added the Category tags i.e. Closed_Eye and Open_Eye tags to the images of the eyes depending on the folder, images belonging to as done in the data collection part. Followed by resizing the images from 86x86 or 94x94 to a common size of 224x224. Since converting the images to the standard pixel size of 256x256 for deep learning purposes introduced garbage-pixel values in the image as well as images turned "lighter" we settled down to 224x224 pixel-size for images. Then, converted the image from GrayScale to RGB(RED GREEN BLUE) format used for colored images so essentially transformed these images into color format!, followed by reading those images and transforming them into arrays and labels to be used as a dataset.

Then divide the dataset into two lists X and Y. X stores the features and Y stores the label(i.e. whether the image is Closed_eye or Open_eye) for training the model. Finally, created the pickle file of both the X and Y lists to store the data in the serialized binary format or as a byte stream.

4.4 Model Training

4.4.1 Baseline Model :

For the purpose of implementing a baseline model so as to obtain a model which has been trained from scratch (i.e. without any pretarining), 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 'tanh' activation function. The structure of the Baseline Model has been shown below:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 1, 222, 32)        64544

 activation (Activation)     (None, 1, 222, 32)        0

 max_pooling2d (MaxPooling2D  (None, 1, 111, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 1, 111, 32)        9248

 activation_1 (Activation)   (None, 1, 111, 32)        0

 max_pooling2d_1 (MaxPooling  (None, 1, 56, 32)        0
 2D)

 conv2d_2 (Conv2D)           (None, 1, 56, 32)         9248

 activation_2 (Activation)   (None, 1, 56, 32)         0

 max_pooling2d_2 (MaxPooling  (None, 1, 28, 32)        0
 2D)

 flatten (Flatten)           (None, 896)               0

 dense (Dense)               (None, 64)                57408

 activation_3 (Activation)   (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

 activation_4 (Activation)   (None, 1)                 0

=================================================================
Total params: 140,513
Trainable params: 140,513
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.2 : Baseline Model Architecture

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

- Input to the model: (1 x 222 x 32)
- Output of ConvLayers: (1 x 111 x 32)
- Input to Fully Connected Layers: (1 x 896)
- Final Output: 1 or 0 (Not Drowsy or Drowsy)

Our major concern while training such networks is Overfitting. Overfitting happens when a model exposed to too few examples learns patterns that do not generalize to new data, i.e. when the model starts using irrelevant features for making predictions. 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 upto a great extent.

Baseline Model Structure :

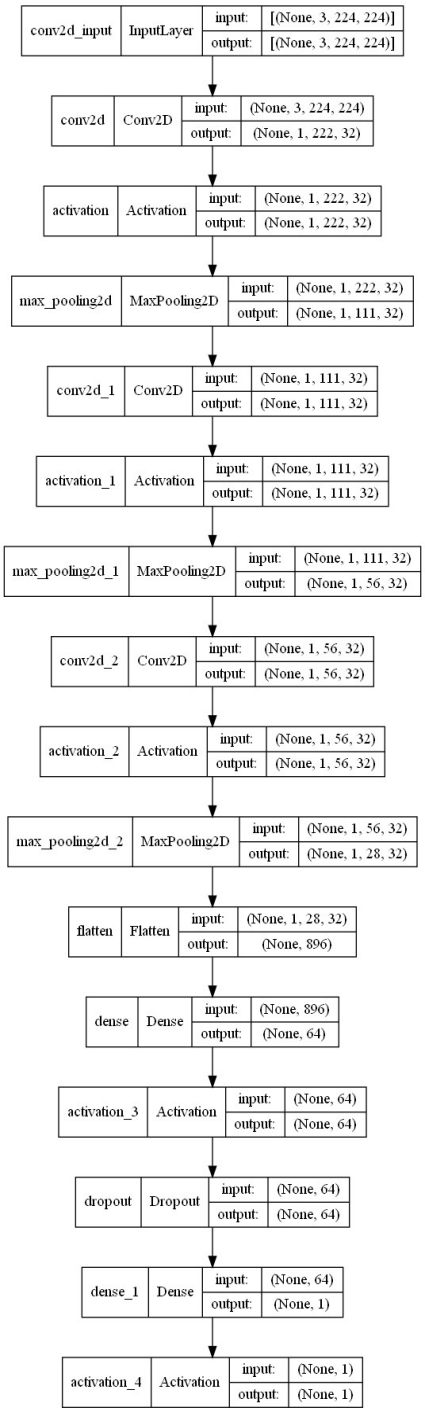

Fig 4.3 : Baseline model structure

4.4.2 Final (Main) Model :

First we imported the mobile net architecture through keras and tensorflow then customized the pre-trained model by removing its four layers and adding three layers of our own and finally adding the activation function (sigmoid) for binary classification (classifying as open eye or close eye).

Then we performed model training. We trained the model for a total of 10 epochs and at the end of each epoch calculated its validation loss and accuracy.

Then, converted the image from GrayScale to RGB(RED GREEN BLUE) format used for colored images so essentially transformed these images into color format !, followed by reading those images and transforming them into arrays and labels to be used as a dataset.

.

```
In [26]: new_model.compile(loss = 'binary_crossentropy',optimizer = 'adam', metrics = ['accuracy'])
```

Fig 4.4 : showing the model compilation configuration

The above image shows that the model is compiled and the loss will be calculated as binary cross entropy with optimizer as adam and the metric to check for is accuracy.

```
In [28]: history = new_model.fit(X,Y,epochs = 10, validation_split = 0.25)
```

```
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
165/165 [==============================] - 639s 4s/step - loss: 2.1119 - accuracy: 0.8297 - val_loss: 5.6710 - val_accuracy: 0.
5909
Epoch 2/10
165/165 [==============================] - 616s 4s/step - loss: 0.7211 - accuracy: 0.9417 - val_loss: 0.8134 - val_accuracy: 0.
9394
Epoch 3/10
165/165 [==============================] - 614s 4s/step - loss: 0.5829 - accuracy: 0.9549 - val_loss: 0.5206 - val_accuracy: 0.
9634
Epoch 4/10
165/165 [==============================] - 618s 4s/step - loss: 0.5875 - accuracy: 0.9451 - val_loss: 2.0064 - val_accuracy: 0.
8629
Epoch 5/10
165/165 [==============================] - 618s 4s/step - loss: 0.2771 - accuracy: 0.9724 - val_loss: 0.5005 - val_accuracy: 0.
9589
Epoch 6/10
165/165 [==============================] - 612s 4s/step - loss: 0.2188 - accuracy: 0.9794 - val_loss: 0.1958 - val_accuracy: 0.
9811
Epoch 7/10
165/165 [==============================] - 611s 4s/step - loss: 0.1510 - accuracy: 0.9840 - val_loss: 0.2434 - val_accuracy: 0.
9760
Epoch 8/10
165/165 [==============================] - 612s 4s/step - loss: 0.1812 - accuracy: 0.9830 - val_loss: 0.1560 - val_accuracy: 0.
9834
Epoch 9/10
165/165 [==============================] - 613s 4s/step - loss: 0.1143 - accuracy: 0.9890 - val_loss: 0.2415 - val_accuracy: 0.
9594
Epoch 10/10
165/165 [==============================] - 621s 4s/step - loss: 0.5316 - accuracy: 0.9491 - val_loss: 0.1953 - val_accuracy: 0.
9709
```

Fig 4.5 : model training

It can be observed that the model is getting trained and despite our best efforts to minimize the training time it still took approximately 10 minutes each epoch thus, the total training time of 10 epochs can be calculated as 10 * (~10) = 100 minutes or 1 hr and 40 minutes . If we increase the number of epochs the accuracy will increase.

4.5 Working with Images

Working on unknown images

```
In [31]: img = cv2.imread('sad_woman.jpg')

In [32]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[32]: <matplotlib.image.AxesImage at 0x1ee006db208>
```

Fig 4.6 : Working on Unknown Images

The model successfully detected the eye and classified it as open eye**.**

```
In [36]: eye = eyecascade.detectMultiScale(gray,1.1,4)

In [37]: for (x,y,w,h) in eye:
             cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0), 2)

In [38]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[38]: <matplotlib.image.AxesImage at 0x1ee00842710>
```

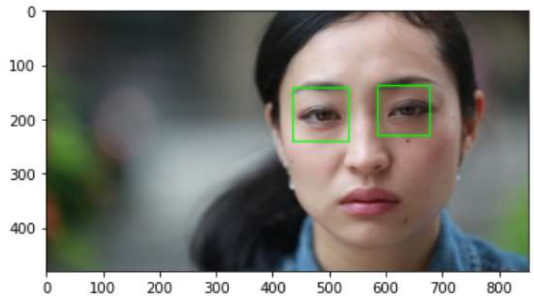

Fig 4.7 : Successful Detection

```
In [44]: prediction = new_model.predict(final_image)

In [45]: prediction

Out[45]: array([[0.9998747]], dtype=float32)
```

Fig 4.8 : prediction

The output is positive and has a value very close to ~1 (0.9998747) showing that the eyes detected are open eyes.

4.6 Working on Real - Time Video

Now that the model is able to accurately detect the eyes from the images, it can now be utilized to create an alert system capable of detecting whether the person is drowsy or not in real-time.
There are two major concerns for the alert system:

1. It should not be so fast that it cannot differentiate between blinking and drowsiness.
2. It should not be slow enough such that it is unable to detect drowsiness in emergency situations.

Finally, this project attempts to rectify the 1st major concern as the 2nd one is more related to the processor or cpu used rather than the actual efficiency of the code.
As explained in the methodology, the deep learning model determines the eye status. If the eyes are found open it calculates the EAR value to make sure that the eyes remain open otherwise it displays an alert on the screen and plays an alarm.
In case of a closed eye, the system waits for a certain interval and then starts playing the alarm sound at a different frequency.

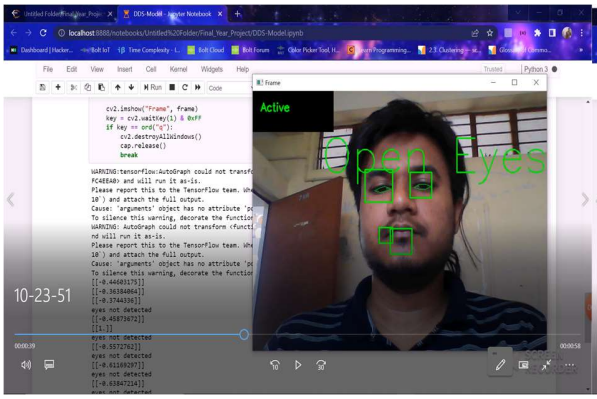

Fig 4.9 : Open eye(real-time)

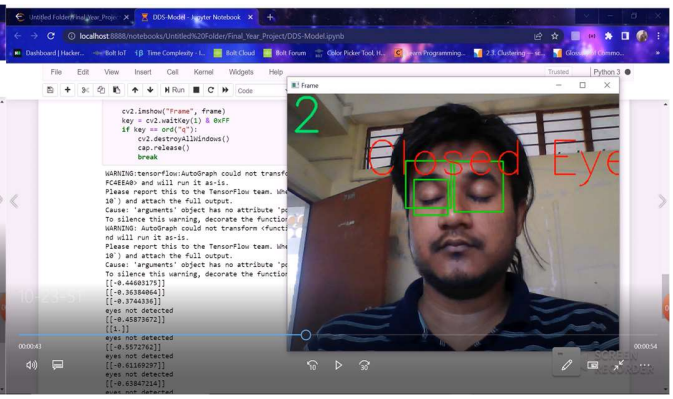

Fig 4.10: Closed eye(real-time)

4.7 Developing a user Interface

A User Interface is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website.
A User Interface is also developed as a part of this project. This is mainly done so as to show how this system can be used on any device. The app is developed as a web application so that it can be hosted on a server and thus, can be used on any system.

There are three main sections in the web application as can be seen on the SideBar.

1. **About Me :**
   This is a general purpose section that provides information about the application and the reason behind the development of this application. Furthermore, this page also includes the linkedin profile links for people who are interested to connect or want to provide valid feedback.

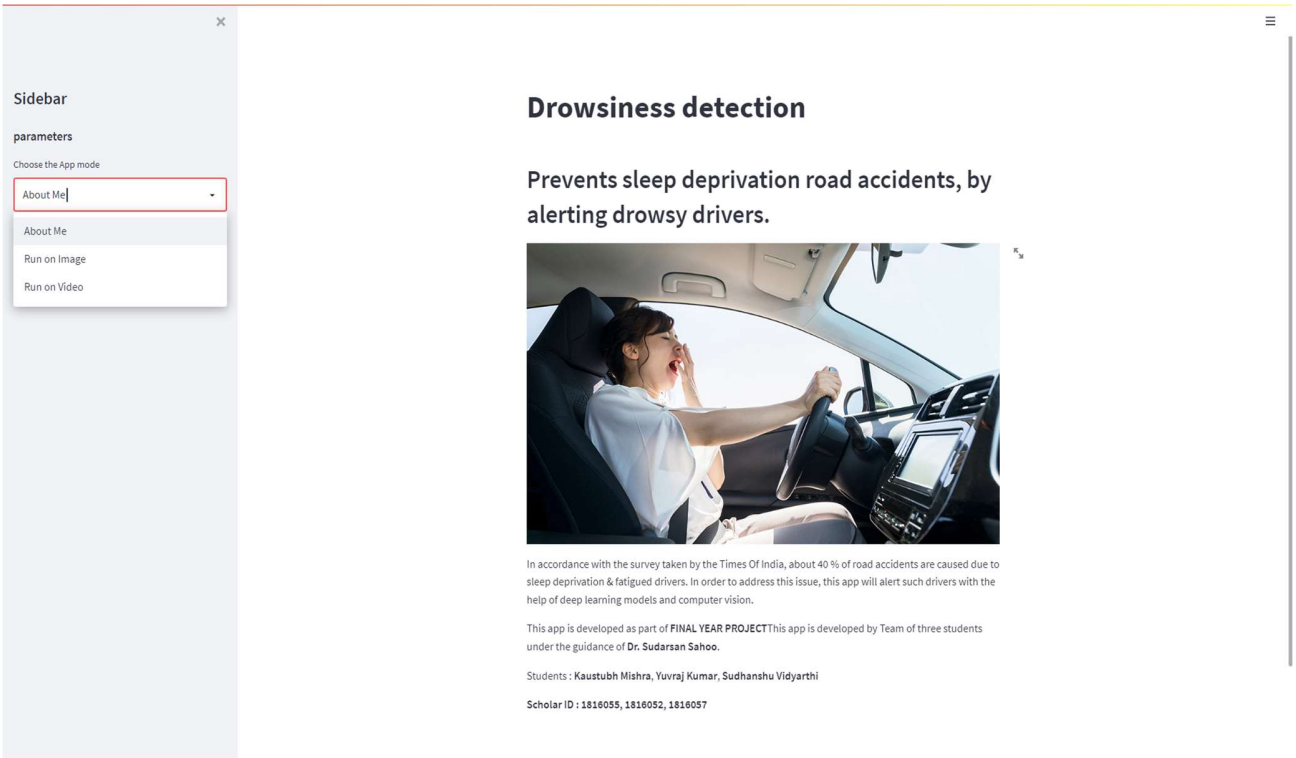

Fig 4.11 : About Me section

2. **Run on Image :**

This page or section of the application checks whether the eyes are open or not on the images fed to it by the user. A demo image is already present so that the user can understand how this section works. Furthermore, this section also displays the **given image, the extracted region from the image and the prediction of the model for the input image**. This section thus provides the user with an understanding on how the model will work on the real time videos wherein it will access each frame as an independent image and check whether the eyes are open or closed on that frame, followed by other frames.

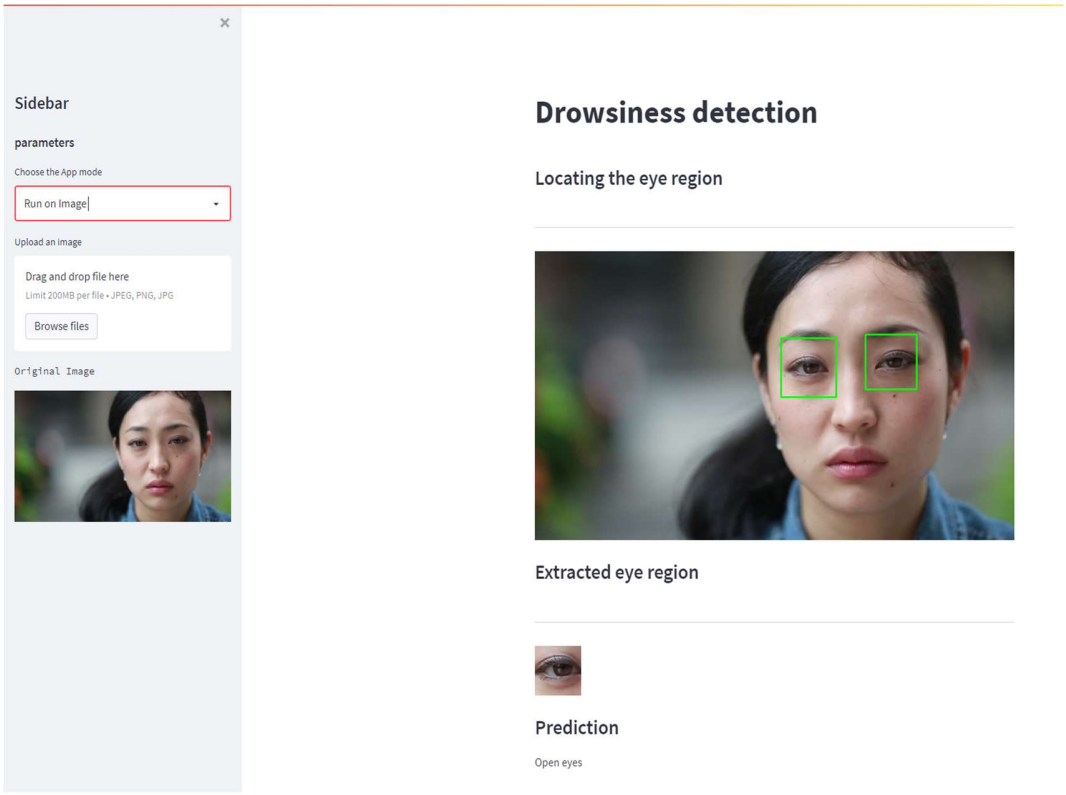

Fig 4.12: Run on Image section (showing demo image)

3. **Run on Video :**

This section or page works on videos. In this section, a demo video is already placed that shows how the system will run on videos. Salient features of this page are :

1. The sidebar in this section shows the button to use web-cam and get a live feed instead of using any video.
2. It also has a button to record the ongoing process of detection whether on the webcam or video.
3. It also displays the fps (frames per second) rate and the width of the frame being used by the webcam or video.
4. The sidebar also shows the actual playable video in-case a video is given to the system.
5. The detected mode of the eye and the detected region are displayed on the frame itself.

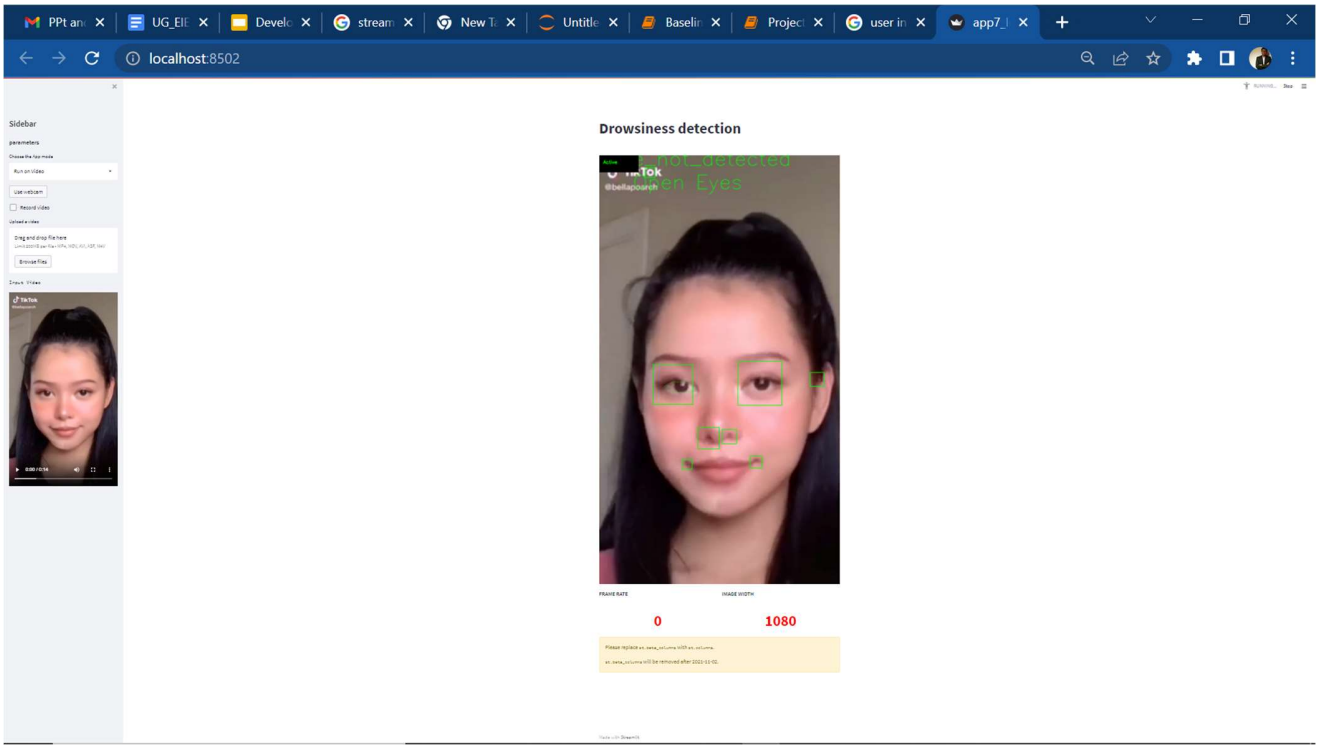

Fig 4.13 : Run on Video section (showing demo video)

Thus, as can be seen from above, the app works perfectly fine and performs all the intended tasks. The model is also working as expected.
This app will be hosted on a server (preferably heroku) and then it will be available to be used on any device as a web application.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 BaseLine Model Vs Final Model

The Final(Main) model was successfully trained and the following figures, figure 5.1 show loss vs epochs curve of training and validation data and figure 5.2 show accuracy vs epochs curve of training and validation data for the final model.

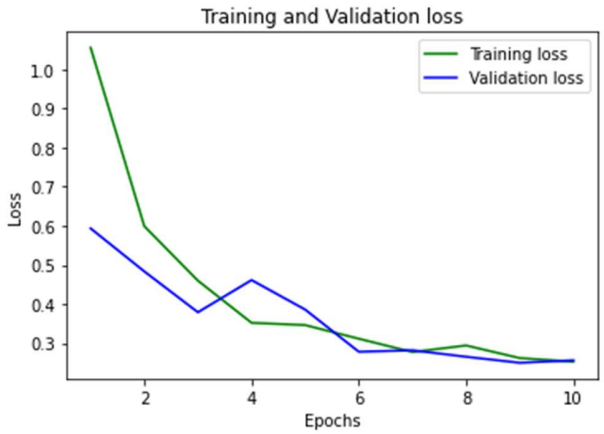

Fig 5.1 : Loss Curves

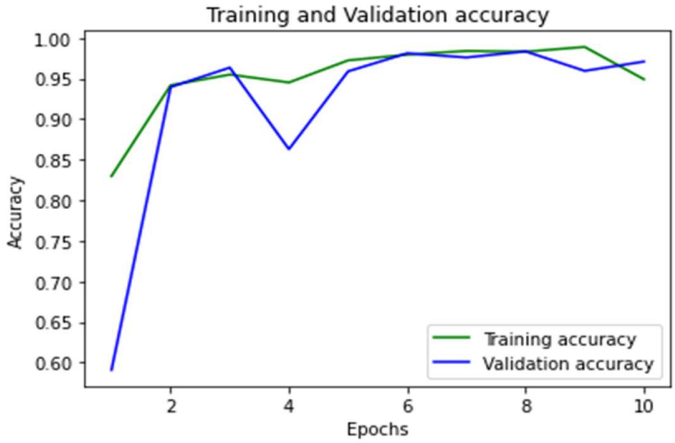

Fig 5.2 : Accuracy curves

The training metrics (loss and accuracy) can be seen in green and the validation metrics (loss and accuracy) are shown in blue. As can be seen, the model starts with quite low training loss due to standardization of filter weights by using the pretrained model and thus has high training accuracy from the start but the validation loss is quite high initially as the filter weights of the model are not configured for this particular dataset and therefore do not extract any meaningful information initially. For the same reason the initial validation accuracy is low(but still better than any model created from the scratch) and thus, slowly at the end of each epoch they take the losses into consideration (both the training and validation loss) and through backward propagation(using adam optimizer) modify the weights to better predict the type of eyes(close eye or open eye) and hence slowly the losses continue to decrease and accuracy improves.

Notice, before 6 epochs both the validation curves(accuracy and loss) decrease rapidly, signifying that the adam optimizer is moving fast or gaining momentum and taking huge steps to reach a stable state.

Also notice that after 6 epochs both the validation curves(accuracy and loss) start to become stable indicating that the model is reaching its intended stable state and as it continues to move forward the difference between losses of each epoch decreases signifying that the adam optimizer is slowing down.

Similarly, for the baseline model :

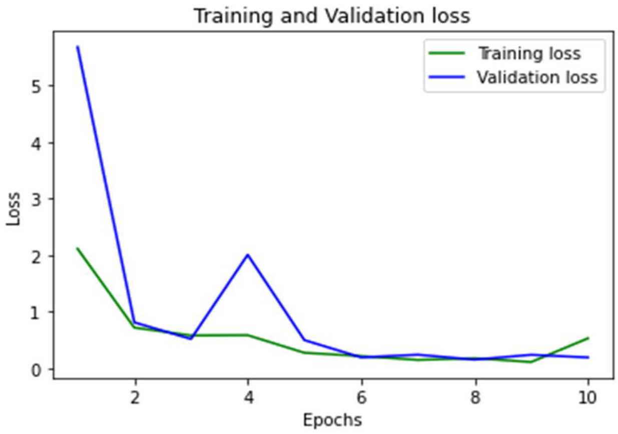

Fig 5.3 : Baseline Loss curve

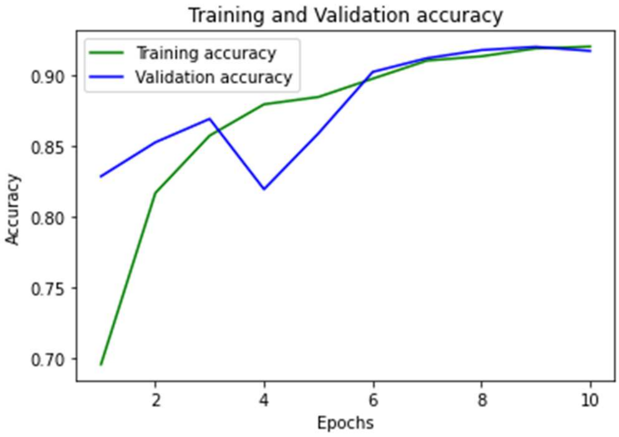

Fig 5.4 : Baseline Accuracy Curve

In order to compare our Final or Main model performance with the baseline model, we used some test images (not used before) from the MRL dataset itself and checked the accuracy of the predictions.

In order to test the accuracy, we took 384 test images from the MRL dataset and found out the predictions made by our models. If the prediction > 0 it meant open eyes otherwise close eyes.
The expected response(output) of the models is stored in the Y_test array as shown below.

```
In [63]: Y_test

Out[63]: array([1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
                0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
                0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
                1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
                1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
                0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
                0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
                0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
                0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1,
                0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
                1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
                0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
                1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1])
```

Fig 5.5 : Y_test (expected output)

The predicted response of the baseline and Main Model are shown below:

```
In [69]: predict

Out[69]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Fig 5.6: Baseline Model predictions

```
In [79]: predict_1

Out[79]: array([1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
                0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1,
                0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1,
                0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
                1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
                0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
                1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
                1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
                0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0,
                1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
                0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
                1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0,
                0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1,
                1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
                1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
                0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0])
```

Fig 5.7: Final(Main) Model predictions

Thus, the accuracy of the models as calculated is :

Table 5.1 : Model Accuracy

Model	**Accuracy**
Baseline	**50.2% - 55%**
Final	**95.1%-96%**

Hence, it can be shown that the accuracy of Models using pre-trained layers or architectures is better than the models created from scratch without any pre-training.
Higher accuracy of pre-trained models is due to the fact that the weights of perceptrons are not randomly allotted but carefully calculated through some other similar data.

5.2 User Interface

The interface developed is working as expected. It responds accurately to the inputs given from the user and produces relevant output. This interface is created using streamlit a python library dedicated to the development and deployment of machine learning and deep learning models on real-world applications.

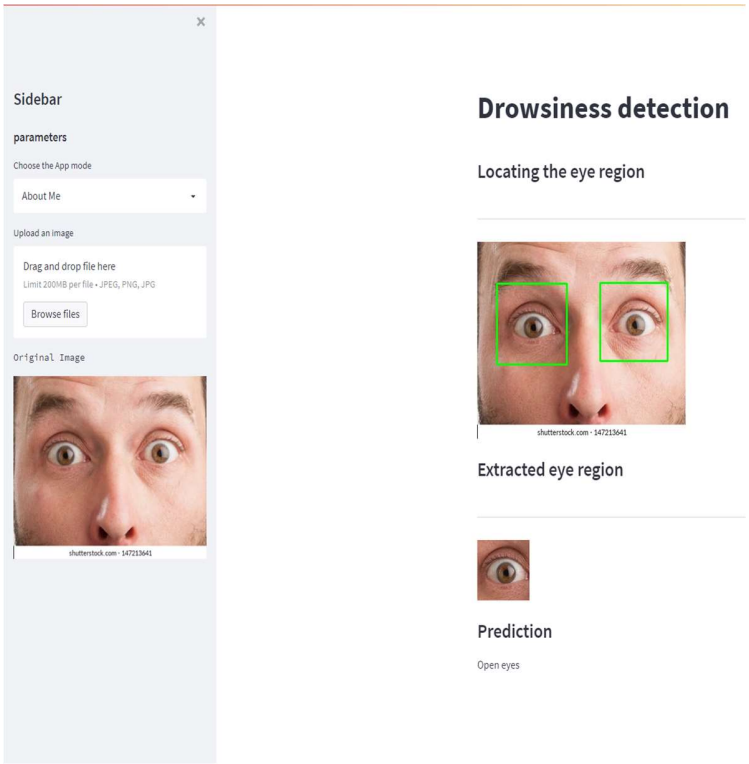

Fig 5.8 : Detecting open eyes and displaying extracted region

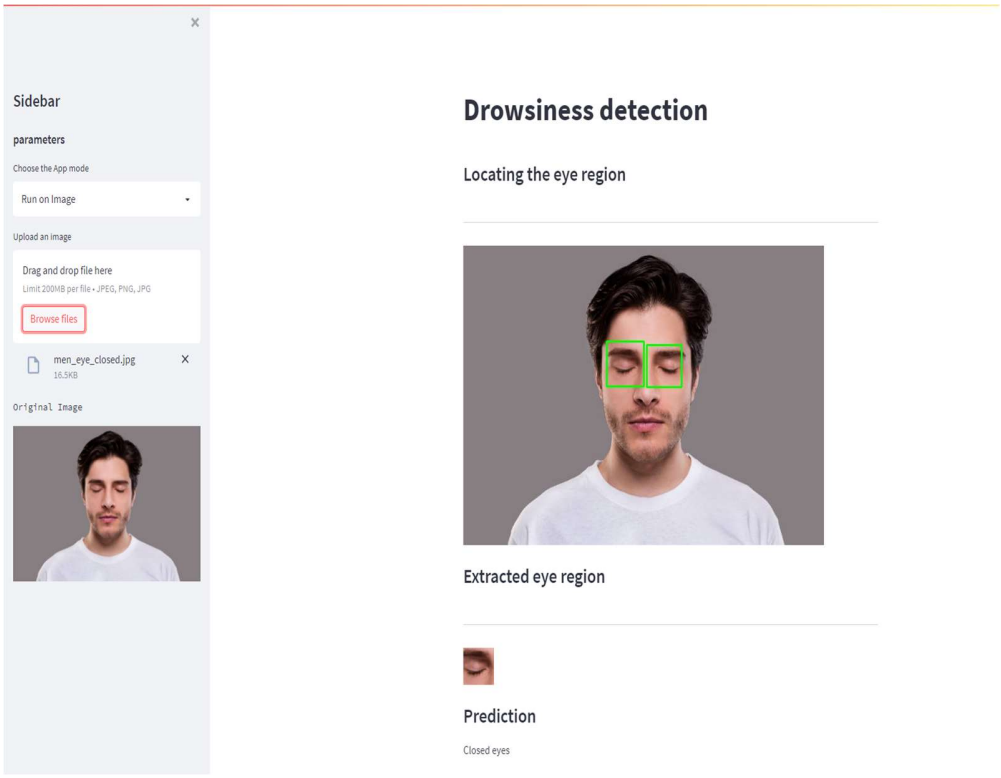

Fig 5.9 : Detecting closed eye and displaying extracted region

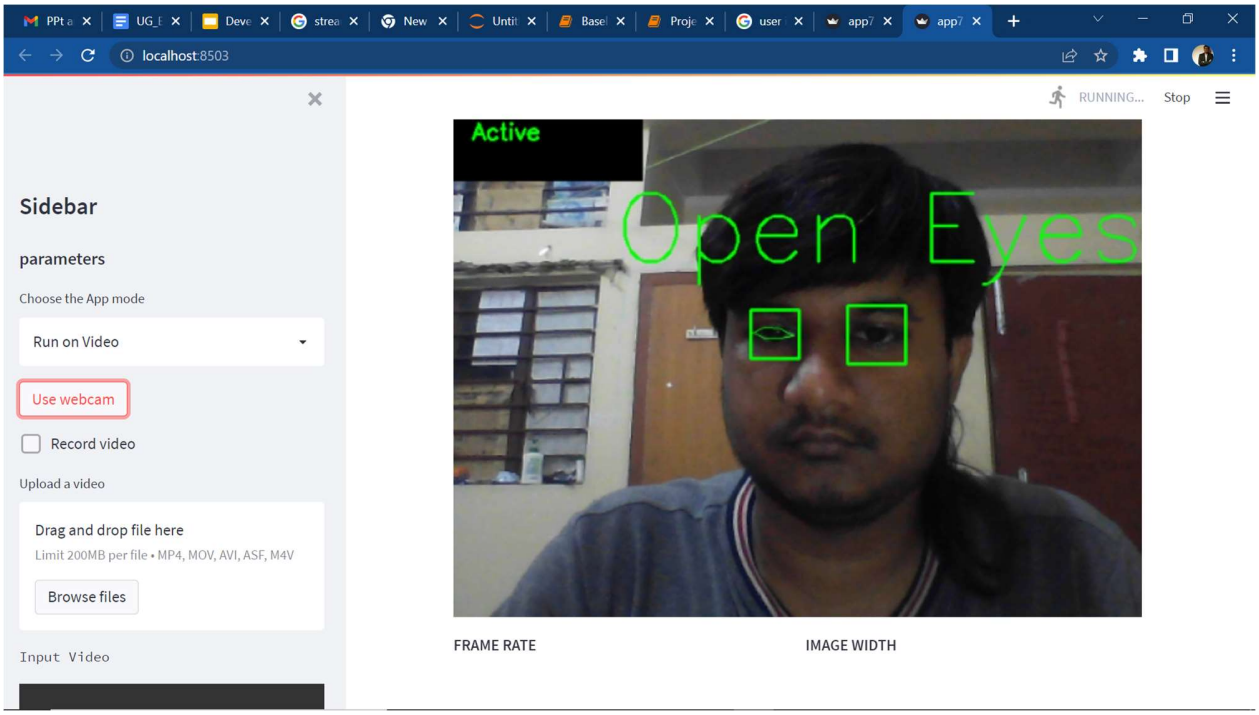

Fig 5.10 : Using webcam (live-feed)

CHAPTER 6

CONCLUSION AND SCOPE FOR FUTURE WORK

As seen in the previous chapter, the model successfully predicted whether eyes are open or closed and can now be used on unknown images for predictions. The accuracy is also good (around 95%), thus, the model is now ready to work on images whereas the accuracy of the baseline model created from scratch is low (around 53%). From here, It can be concluded that the Pre-trained model provided far better accuracy and predictions as compared to the model created from scratch. This is primarily due to the fact that the number of layers in the baseline model are less densely packed than in the Pre-trained Model. Also, the Pre-trained model judiciously uses its loss function to move towards the optimum point whereas the baseline model failed to do so. Furthermore, the Pre-trained weights of the Final model provided a better starting point than the randomized initial weights of the baseline model.

Finally, it can be concluded that

1. The process of drowsiness detection is very important for both individual and community safety.
2. The growing use of artificial intelligence can be immensely helpful in predicting fatigue of the driver.
3. Our model has various advantages as the installation is simple - of a camera, and it does not hinder the driver's experience and comfort like in other techniques such as biological features or vehicle-based features.
4. Not prone to external disturbances, considers only the driver's face.
5. The app works as designed and produces expected results.

Still, there are several areas of improvement in the app which can be regarded as scope for future works.

1. The frame rate processing of the app is low. This may be partially attributed to the system the app is working on but can also open a roadway for scope of future work.
2. Even though the model is trained on a large dataset(on 7000 images), still the accuracy for videos is not as high as for still images and can be trained on even bigger dataset and on a system with bigger memory so that the processor doesn't run out of space. This can be an area for further work.
3. Another area of improvement lies in developing a methodology to integrate this system as part of an embedded system in any microprocessor.
4. Yet another scope of future research is to modify the code to develop it as a mobile application.

REFERENCES

1. Feng You, Yunbo Gong, Haiqing Tu, Jianzhong Liang, Haiwei Wang, "A Fatigue Driving Detection Algorithm Based on Facial Motion Information Entropy", Journal of Advanced Transportation, vol. 2020, Article ID 8851485, 17 pages, 2020. https://doi.org/10.1155/2020/8851485.

2. Charlotte Jacobé de Naurois, Christophe Bourdin, Anca Stratulat, Emmanuelle Diaz, Jean-Louis Vercher, Detection and prediction of driver drowsiness using artificial neural network models, Accident Analysis & Prevention,Volume 126, 2019,Pages 95-104,ISSN 0001-4575, https://doi.org/10.1016/j.aap.2017.11.038.

3. Gang Li, Wan-Young Chung, A Context-Aware EEG Headset System for Early Detection of Driver Drowsiness ,2015 Aug; 15(8): 20873–20893. Published online 2015 Aug 21. doi: 10.3390/s150820873.

4. Arun Sahayadhas, Kenneth Sundaraj, Murugappan Murugappan, Detecting Driver Drowsiness Based on Sensors (Basel) ,2012 Dec; 12(12): 16937– 16953. Published online 2012 Dec 7. doi: 10.3390/s121216937