

Driver Monitoring System

*Project phase-I report submitted
in
partial fulfillment of requirement for the award of degree of*

Bachelor of Technology in Information Technology

By

Mr. Haidar Ahmad

Mr. Haidar Hasan

Guide

Prof. Swati Shamkuwar



**Department of Information Technology
G H Raisoni College of Engineering, Nagpur**

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

Accredited by NAAC with “A+” Grade

Ranked 130th by NIRF, MHRD in the Engineering Category for India Ranking 2021,
Ranked 2nd by ARIIA 2020, MHRD in Private or Self Finance Institutions, 5 Star Rating
by MIC, MHRD 2021

Declaration

We, hereby declare that the project phase-I report titled “**Driver Monitoring System**” submitted herein has been carried out by us towards partial fulfillment of requirement for the award of Degree of Bachelor of Technology in **Information Technology**. The work is original and has not been submitted earlier as a whole or in part for the award of any degree / diploma at this or any other Institution / University.

We also hereby assign to G H Raison College of Engineering, Nagpur all rights under copyright that may exist in and to the above work and any revised or expanded derivatives works based on the work as mentioned. Other work copied from references, manuals etc. are disclaimed.

Place: Nagpur, Maharashtra
Date

Haidar Ahmad
Haidar Hasan

Certificate

The project phase-I report entitled as “**Driver Monitoring System**” submitted by **Haidar Ahmad, Haidar Hasan** for the award of Degree of Bachelor of Technology in Information Technology has been carried out under our supervision. The work is comprehensive, complete and fit for evaluation.



Institute Guide
Prof. Swati Shamkuwar
Assistant Professor

Department of Information Technology
G H R C E, Nagpur

RAISONI GROUP

— a vision beyond —

Dr. Mahendra Gaikwad
Head
Department of Information Technology
G H R C E, Nagpur

Dr. Sachin Untawale
Director
GHRCE, Nagpur

ACKNOWLEDGEMENT

We would like to take this opportunity to express our deep sense of gratitude to all who helped me directly or indirectly during this work.

We are thankful to our project guide, Prof. Swati Shamkuwar, Department of Information Technology, GHRCE, Nagpur for guiding us through this entire project and giving us good lessons and we wish to gain valuable knowledge and guidance from her in future also.

We are very grateful to Dr. Mahendra Gaikwad, Head of Department of Information Technology GHRCE, Nagpur and Dr. Sachin Untawle Director of GHRCE, Nagpur for his constant support and for providing necessary facilities for conducting out the project. we had a lot of fun completing this project and also learned a lot and we are sure that in the upcoming projects, we will actively participate and complete them with integrity.

Finally, we would like to express our gratitude to our family members who supported and encourages us all the time.

ABSTRACT

Drowsiness of the drivers is the principal cause of injuries in the world. Because of loss of sleep and tiredness, drowsiness can occur even as riding. The first-rate manner to keep away from accidents because of drivers' drowsiness is to come across drowsiness of the driving force and warn him before fall into sleep. To discover drowsiness many techniques like eye retina detection, facial function recognition has been used. Here on this project, we suggest a way of detecting eyelid movement, the usage of eyelid movement detection. On this record, we endorse an extra accurate drowsiness detection approach which is eyelid movement detection.

Several studies have reported indicating that driver drowsiness has caused multiple number of accidents over the past years. So, in an effort to prevent such accidents, measures are to be implemented. One such measure is a Driver Drowsiness Detection software; the software will set off an alarm once the driver keeps his/her eyes closed for a certain amount of time. This project proposes to detect whether the driver is sleepy or not by using OpenCV with eyelid related parameters. The data consists of around 4000 images of eyes under different optical conditions. The used dataset is divided into two parts for the sake of convenience i.e., open eye and close eye. With all the features, an OpenCV and keras a driver's fatigue or drowsiness model is built. The validation results indicate the precision and accuracy of the proposed model.

CONTENTS

Abstract	I
Contents	II
List of Figures	IV
Introduction.....	1
Problem Summary and Introduction	1
Aim and Objectives	1
Problem Specifications	2
Literature Review and Prior Art Search (PAS)	3
Materials / Tools required	4
Analysis, Design Methodology and Implementation Strategy	5
Observation Matrix	5
Ideation Canvas	6
Product Development Canvas	7
Dataset	8
System Design	9
Implementation	14
Implemented Functionality	14
pre-Requested	14
Tensorflow	14
Keras	15
Pillow	19
Numpy.....	20
Tqdm.....	22
Jupyter Notebook	23
VS code.....	25
Project File Structure	26
Getting and Performing Data Cleaning	27
Extracting the Feature from MobileNet () model:	30
Building the Model	32
Training the model	33
Results and Reports	36
Snapshots	38
Testing and Verification	40
Conclusion	43
Summary of the results	43

Advantages of Work / Results / Methodologies.....	43
Scope of future work.....	43
References	45

LIST OF FIGURES

Figure 1 Observation Matrix	6
Figure 2 Ideation Canvas	7
Figure 3 Product Development Canvas	8
Figure 4 Dataset open-eye folder	9
Figure 5 Dataset Closed-eye folder	9
Figure 6 System Block Diagram	10
Figure 7 Activity Diagram	11
Figure 8 Sequence Diagram	12
Figure 9 Class Diagram	12
Figure 10 Data flow Diagram	13
Figure 11 TensorFlow	15
Figure 12 Keras	19
Figure 13 NumPy.....	22
Figure 14 Jupyter Notebook	25
Figure 15 VS code.....	26
Figure 16 Project File Structure.....	27
Figure 17 performing Data cleaning.....	29
Figure 18 performing Data cleaning.....	30
Figure 19 Extracting the Feature	31
Figure 20 Loading the model.....	33
Figure 21 Batch example	35
Figure 22 Training the model	36
Figure 23 building the GUI	37
Figure 24 building the GUI	37
Figure 25 Importing the Data-set	38
Figure 26 Performing Data pre-processing	38
Figure 27 Loading Mobile-Net () model.....	39
Figure 28 Building our model.....	39
Figure 29 Training and saving our model.....	40
Figure 30 building the GUI	40
Figure 31 Testing with Closed eyes	41
Figure 32 Testing with Open eyes	42

INTRODUCTION

Problem Summary and Introduction

Driver drowsiness detection is a car safety technology which prevents accidents when the driver is getting drowsy. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads. Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident-avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. Driver inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person's attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and distraction, however, might have the same effects, i.e., decreased driving performance, longer reaction time, and an increased risk of crash involvement.

Our overall system is based on Acquisition of video from the camera that is in front of driver perform real-time processing of an incoming video stream in order to infer the driver's level of fatigue if the drowsiness is Estimated then the output is send to the alarm system and alarm is activated

Aim and Objectives

This project was first thought of as a problem in our modern times that needed to be addressed and worked upon, because it's believed the focus of the resources and research capabilities should be placed on our society's problems and dire need for development, enhancing the abilities of humans to create a sustainable environment for future generations and achieving common goals between countries, it's believed that without having these values and principles, a better world cannot be reached.

This project reflects our aim and efforts to contribute even by a small amount, to the further improvement and development of the community, hoping that we will extend this project for future generation to work upon it.

The main functionality of Driver Monitoring System is to perform real-time processing of an incoming video stream. It should be accessible and available to whoever requires such technology, we hope to add as much documentation and blue print in order to make it easier for future researchers to add their contribution to this project and enhance its features. Seeing as currently projects are limited by the technologies available, it's becoming clear that the scope and potential of this project is tremendous.

The objective is to learn the concepts of a CNN model, Open-CV and to build a working model of Driver Monitoring System by implementing CNN, Transfer Learning, and develop a Python based Windows application for user. Further options are possible but currently for the purpose of introducing a demo, the previous requirements are used in order to illustrate a clear and working functionality of the project, and shed some light into mechanisms of this project. This model can be implemented in many commerce and business cooperate as it easily fits a revenue model which can help introduce a new way of funding further research and invites more people to join in on the development journey of this project. Working on the efficiency of this project by focusing efforts on the output per input ratio, it will produce better results and contribute to the overall impact of this project on a bigger scale.

Problem Specifications

Current drowsiness detection systems monitoring the driver's condition requires complex computation and expensive equipment, not comfortable to wear during driving and is not suitable for driving conditions; for example, Electroencephalography (EEG) and Electrocardiography (ECG), i. e. detecting the brain frequency and measuring the rhythm of heart, respectively. A drowsiness detection system which uses a camera placed in front of the driver is more suitable to be use but the physical signs that will indicate drowsiness need to be located first in order to come up with a drowsiness detection algorithm that is reliable and accurate. Lighting intensity and while the driver tilts their face left or right are the problems occur during detection of eyes and mouth region. Therefore, this project aims to analyze all the previous research and method, hence propose a method to detect

drowsiness by using video or webcam. It analyzes the video images that have been recorded and come up with a system that can analyze each frame of the video.

Literature Review and Prior Art Search (PAS)

In Background work the various reference of the existing projects are taken into consideration which are similar to this current project.

Use of the Hough Transformation to Detect Lines and Curves in Pictures [1]

Less attention leads the driver to being distracted and the chance of road twist of fate is going high. Drowsiness related injuries have all the earmarks of being extra serious, due to the higher speeds concerning distraction and the driving force being no longer able to take any keeping off activity, or maybe brake, before the coincidence. The development of improvements for recognizing or stopping tiredness of the motive force is a tremendous take a look at inside the field of coincidence stopping systems. Due to the danger that that drowsiness presents on the street, techniques need to be created for checking its effects. Loss of the attention due to the tiredness reasons a few modifications inside the human's frame and sports. Those aspect outcomes and parameters empower us to efficiently degree the drowsiness stage.

Understanding User Profiles on social media [2]

The national motorway site visitor's protection administration estimates that from 1989 via 1993, driving force drowsiness/fatigue became a contributing element in crashes yearly on U.S. highways. A current study has tested the consequences of innovative sleep deprivation on using overall performance to assess the fee of crashes and the modifications in riding overall performance attributable to sleepiness. Because it might be dangerous to take a look at this below real driving conditions, the high-constancy highway using a simulator was used. a spread of measures, along with non-stop electroencephalogram (EEG) tracking, videotaping, and analyses of using overall performance data and questionnaire facts have been used to determine the consequences of sleep deprivation at the using performance of six men and six women aged 26-35. Motorway protection variables, which include range of crashes and number of lane tours, were unacceptably excessive on day 3 after 36 hours of no sleep and on day 4 after 60 hours without sleep. More subtle measures of motorway safety, together with velocity and lateral placement variance, had been also related to sleep deprivation

Materials / Tools required

Software requirements:

- ❖ OS: Windows 7 or above, Recommended: Windows 10.
- ❖ For development: Python 3.9 or above, Jupyter Notebook, Anaconda 3 or above, reliable internet bandwidth.
- ❖ For Execution: Python 3.9 and above, Command Prompt.

Hardware requirements:

- ❖ CPU: A minimum of 8th generation (Intel Core i5 processor) 2.30 GHz is recommended.
- ❖ RAM: A minimum of 8 GB RAM.
- ❖ Storage: A minimum of 20 GB.

ANALYSIS, DESIGN METHODOLOGY AND IMPLEMENTATION STRATEGY

Observation Matrix

Observation Matrix (figure – 1) is a way to find out how dependent the measurements are on the condition of the system.

The matrix consists of 4 parts:

Observations, which includes all the possible platform we can use to introduce the model, each one of them has its own pros and cons, although many of them are viable options to develop the mode, as a starting point we have to begin development on the most suitable platform, the observations are:

- ❖ The website
- ❖ Web Application
- ❖ Android Application
- ❖ Linux App
- ❖ Mac program
- ❖ Windows Software
- ❖ IOS APP

The scouted challenges, after careful consideration and research, the option to develop on an IOS app was not viable considering the limited user base so it was eliminated.

Based on the desirability, feasibility, Viability, it became clear that having web application is not suitable seeing as there are better options.

Lastly, seeing as the most popular method was the Windows Software, it was considered as the desired method to develop the model with, furthermore, the development process will take less running time and help the developer in being more efficient in his work.

Observation Matrix

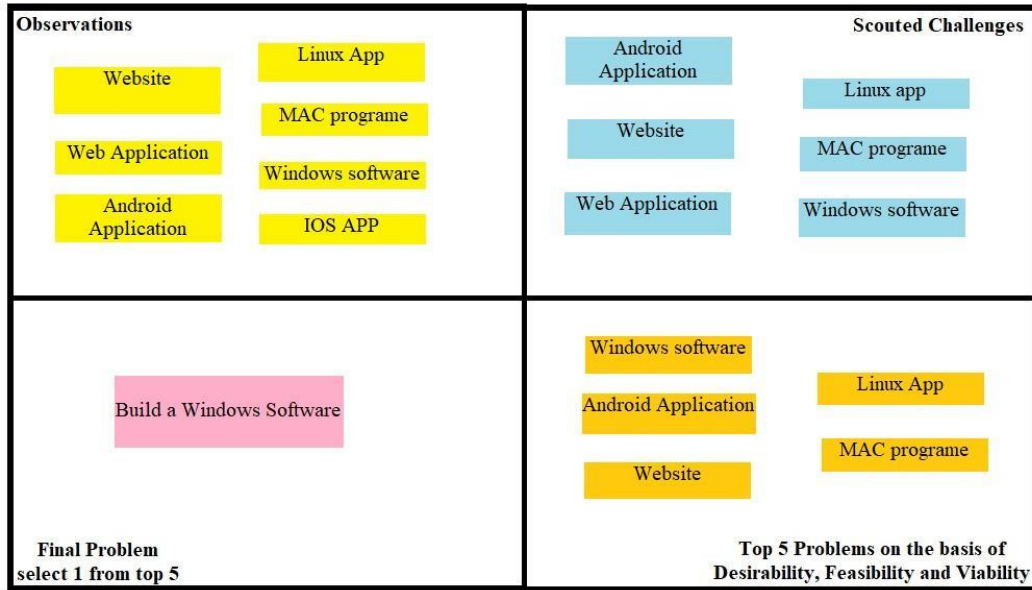


Figure 1 Observation Matrix

Ideation Canvas

The Ideation canvas will help to make the ideas clear in order to implement them into the product. First the people are defined, Engineers, Drivers. By defining the people, we are able to better understand the features that need to be implemented and the requirements.

The activities of the model only include, detecting driver eye status, start/stop the alarm.

Thirdly, the Situation/Context/Location where the product will be used, can be defined as follows: The product can be hosted by a website using cloud computing services which allows for better reach and less requirement needed to run it since websites use SAAS (Software as A Service) model. Also, the Product can be introduced as a mobile application, which will help make it available to a wider range of users and make it accessible from different environments.

Ideation Canvas

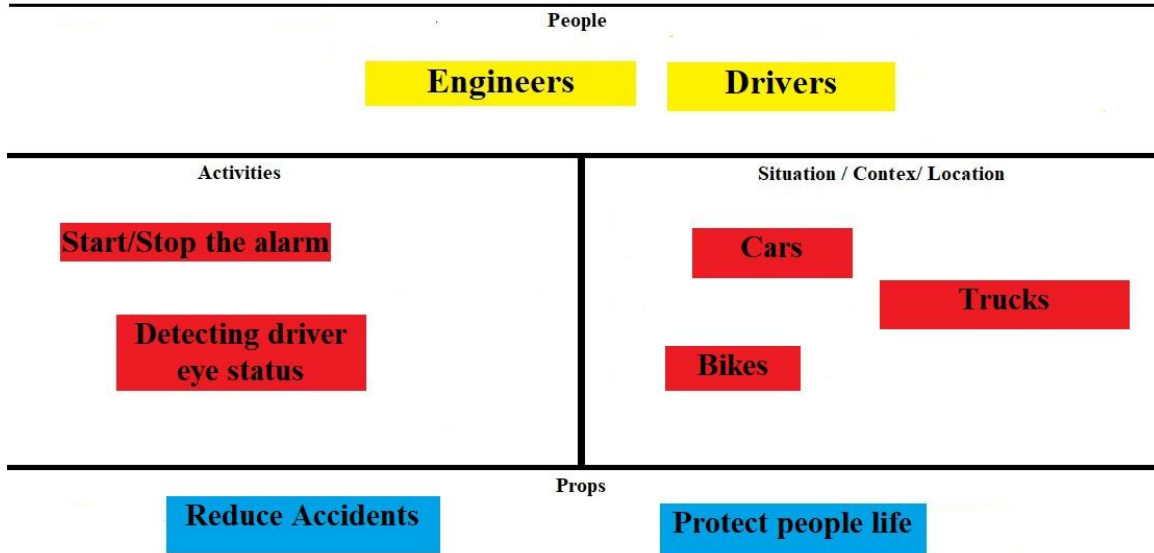


Figure 2 Ideation Canvas

Product Development Canvas

Product Development Canvas is a planning canvas intended to build products that have a good user experience through an aim to work on feature development. It includes both agile methodologies with UX principles to aid the validation of the product solutions. In this canvas, there are 8 components. The first one, Purpose, explains how the goal is to produce an accurate satisfying description which is accomplished by constantly optimizing the algorithms and feeding the model with the proper datasets. Secondly comes the people, referring to who can use the product, which are all kind of users who require such technology without any specific category of users. It can be a regular person, like Driver, or any software Developer. Next comes the product experience, which after a brief survey the experience was described as Easy, Simple, and doesn't require any technical knowledge or specific skill in any domain. Then comes a very important part, the product functions, which can be explained as follows, the product takes in an input from the user, it then performs analysis on the input by utilizing neural networks, which helps detect the eye status. The product features are what makes it stand out, its user-friendly GUI makes it easy for users to navigate its functionality and get a good user experience, secondly, its computer application and wide spectrum of possibilities makes it desirable and popular. The components for this product are easy to attain and simple. First, a pc is highly recommended to host to product and use it, although the specifications of the pc are high,

users can use the product without any performance issues. The customer revalidation, the revalidation by the customer was that the product was easy to use and didn't require much time to navigate through, secondly, it's multipurpose and wide range of use cases was highly applauded. The reject/reassign/retain, the product received some negative feedback, such as requiring high end device in order to run it on, the high amount of data needed to handle was also an issue that needed to be addressed which leads to the next point. Long training time is issue that needs to be worked on, less amount of data and faster algorithm needs to be introduced and improved.

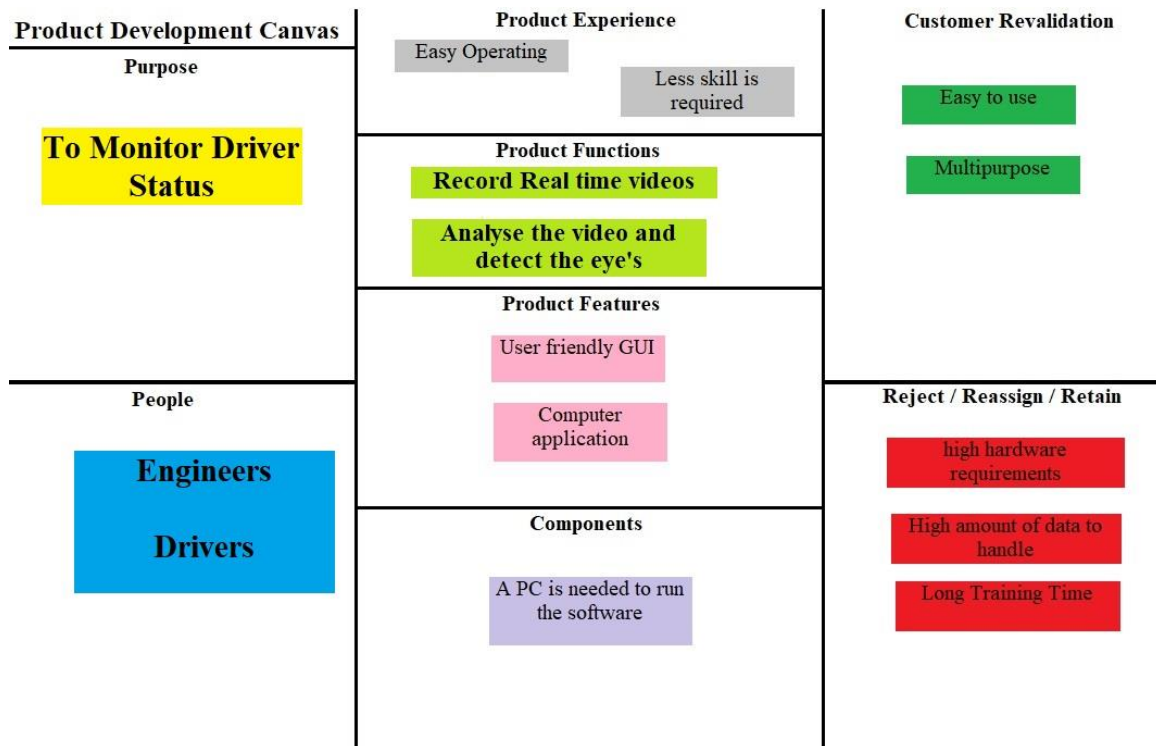


Figure 3 Product Development Canvas

Dataset

We will be using a data set consists of 4000 photos [13], divided into two classes "Open eye" and "Closed eye"

"Open eye" folder contains 2000 photo

"Closed eye" folder contain 2000 photo

The image should be converted to suitable features so that they can be trained into our model.



Figure 4 Dataset open-eye folder

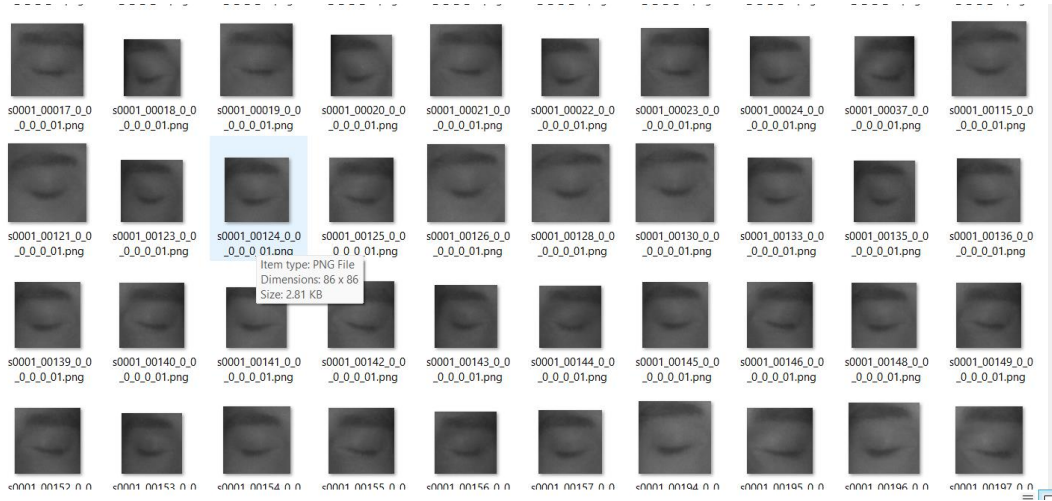


Figure 5 Dataset Closed-eye folder

System Design

In this Python project, we will be using OpenCV for gathering the images from webcam and feed them into a Deep Learning model which will classify whether the person's eyes are 'Open' or 'Closed'. The approach we will be using for this Python project is as follows:

- ❖ Take image as input from a camera.
- ❖ Detect the face in the image and create a Region of Interest (ROI).
- ❖ Detect the eyes from ROI and feed it to the classifier.
- ❖ Classifier will categorize whether eyes are open or closed.
- ❖ Calculate score to check whether the person is drowsy

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

We will implement transfer learning by use a pre-trained model MobileNet() and extract its features to use it in our model

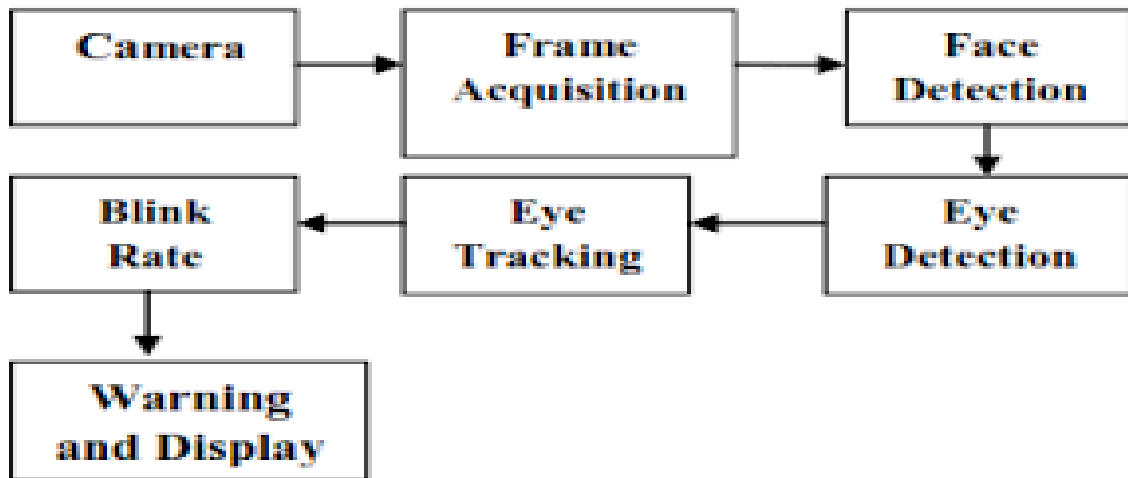


Figure 6 System Block Diagram

Activity diagram is one of the most important diagrams in UML to show the dynamic traits of the model, Activity diagram is simply a diagram to represent the flow from one activity to another activity. The activity can be shown as an operation of the model.

The main goal of activity diagrams is almost the same to the other four diagrams. It highlights the dynamic interactions of the system. Other four diagrams are used to illustrate the message flow from one object to another but this diagram is used to show message flow from one activity to another.

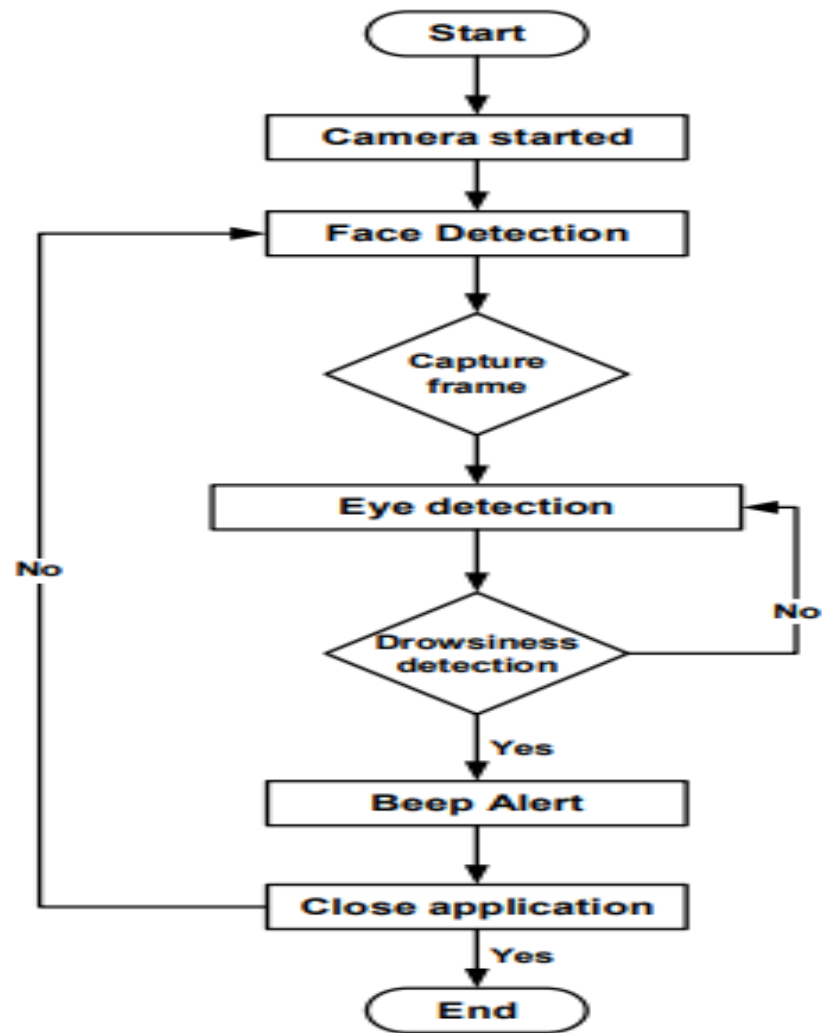


Figure 7 Activity Diagram

Sequence Diagram in explains how the sequence of the program is, to briefly summarize it.

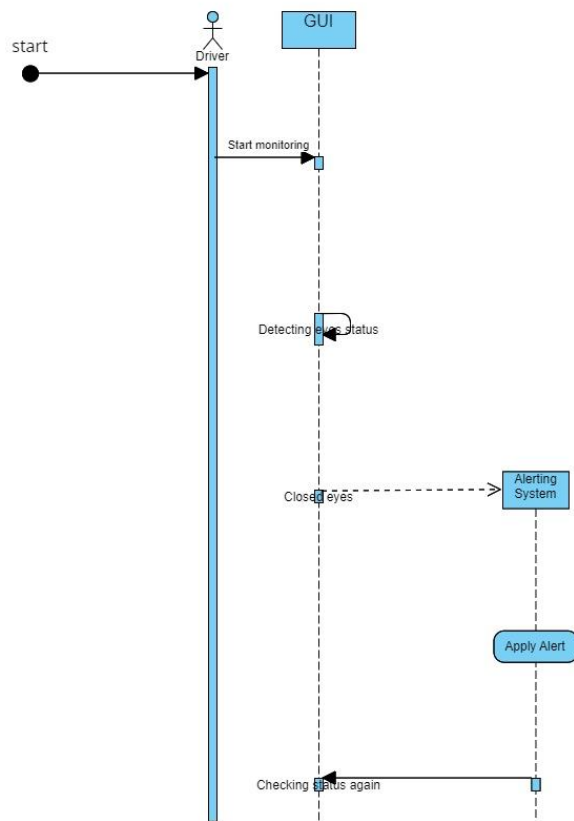


Figure 8 Sequence Diagram

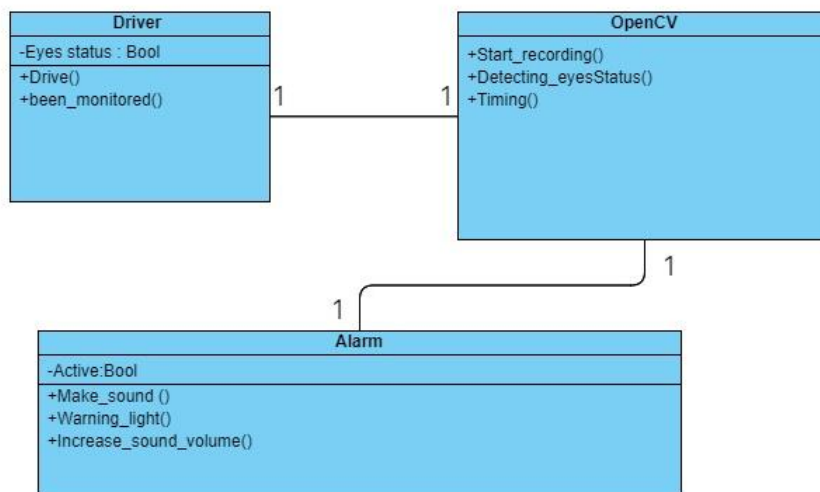


Figure 9 Class Diagram

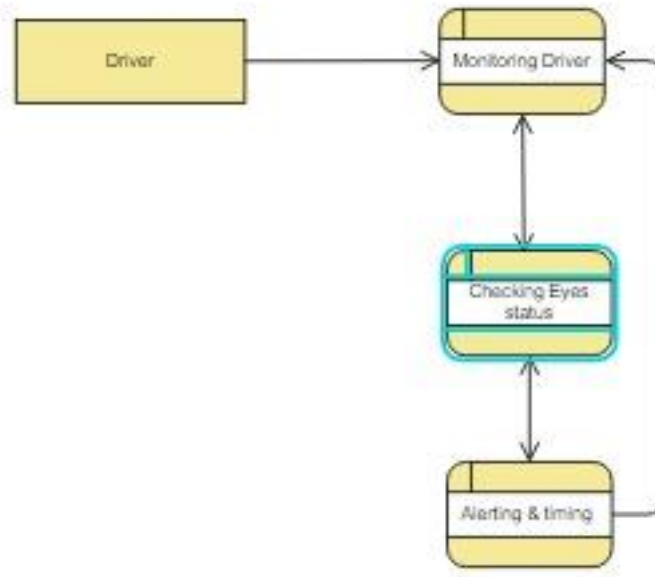


Figure 10 Data flow Diagram

IMPLEMENTATION

Implemented Functionality

pre-Requested

This project requires good knowledge of Deep learning, Python, working on Jupyter notebooks, Keras library, Numpy, and Natural language processing. We have installed all the following necessary libraries:

- ❖ Tensorflow
- ❖ Keras
- ❖ Pillow
- ❖ Numpy
- ❖ Tqdm
- ❖ Jupyter Notebook
- ❖ VS code

Tensorflow

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an open-source library for fast numerical computing.

It was created and is maintained by Google and was released under the Apache 2.0 open-source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least of which is RankBrain in Google search and the fun DeepDream project. It can run on single CPU systems and GPUs, as well as mobile devices and large-scale distributed systems of hundreds of machines.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities and methods for the same.

TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.



Figure 11 TensorFlow

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.

- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Keras & TensorFlow 2

TensorFlow 2 is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

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

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination. It cannot handle low-level computations, so it makes use of the Backend library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

Initially, it had over 4800 contributors during its launch, which now has gone up to 250,000 developers. It has a 2X growth ever since every year it has grown. Big companies like Microsoft, Google, NVIDIA, and Amazon have actively contributed to the development of Keras. It has an amazing industry interaction, and it is used in the development of popular firms likes Netflix, Uber, Google, Expedia, etc.

What makes Keras special?

- Focus on user experience has always been a major part of Keras.

- Large adoption in the industry.
- It is a multi backend and supports multi-platform, which helps all the encoders come together for coding.
- Research community present for Keras works amazingly with the production community.
- Easy to grasp all concepts.
- It supports fast prototyping.
- It seamlessly runs on CPU as well as GPU.
- It provides the freedom to design any architecture, which then later is utilized as an API for the project.
- It is really very simple to get started with.
- Easy production of models actually makes Keras special.

Keras user experience

- Keras is an API designed for humans
- Best practices are followed by Keras to decrease cognitive load, ensures that the models are consistent, and the corresponding APIs are simple.
- Not designed for machines
- Keras provides clear feedback upon the occurrence of any error that minimizes the number of user actions for the majority of the common use cases.
- Easy to learn and use.
- Highly Flexible
- Keras provide high flexibility to all of its developers by integrating low-level deep learning languages such as TensorFlow or Theano, which ensures that anything written in the base language can be implemented in Keras.

How Keras support the claim of being multi-backend and multi-platform?

Keras can be developed in R as well as Python, such that the code can be run with TensorFlow, Theano, CNTK, or MXNet as per the requirement. Keras can be run on CPU, NVIDIA GPU, AMD GPU, TPU, etc. It ensures that producing models with Keras is really simple as it totally supports to run with TensorFlow serving, GPU acceleration (WebKeras, Keras.js), Android (TF, TF Lite), iOS (Native CoreML) and Raspberry Pi.

Keras Backend

Keras being a model-level library helps in developing deep learning models by offering high-level building blocks. All the low-level computations such as products of Tensor,

convolutions, etc. are not handled by Keras itself, rather they depend on a specialized tensor manipulation library that is well optimized to serve as a backend engine. Keras has managed it so perfectly that instead of incorporating one single library of tensor and performing operations related to that particular library, it offers plugging of different backend engines into Keras.

Keras consist of three backend engines, which are as follows:

- TensorFlow

TensorFlow is a Google product, which is one of the most famous deep learning tools widely used in the research area of machine learning and deep neural network. It came into the market on 9th November 2015 under the Apache License 2.0. It is built in such a way that it can easily run on multiple CPUs and GPUs as well as on mobile operating systems. It consists of various wrappers in distinct languages such as Java, C++, or Python.

- Theano

Theano was developed at the University of Montreal, Quebec, Canada, by the MILA group. It is an open-source python library that is widely used for performing mathematical operations on multi-dimensional arrays by incorporating scipy and numpy. It utilizes GPUs for faster computation and efficiently computes the gradients by building symbolic graphs automatically. It has come out to be very suitable for unstable expressions, as it first observes them numerically and then computes them with more stable algorithms.

- CNTK

Microsoft Cognitive Toolkit is deep learning's open-source framework. It consists of all the basic building blocks, which are required to form a neural network. The models are trained using C++ or Python, but it incorporates C# or Java to load the model for making predictions.

Advantages of Keras

Keras encompasses the following advantages, which are as follows:

- It is very easy to understand and incorporate the faster deployment of network models.
- It has huge community support in the market as most of the AI companies are keen on using it.

- It supports multi backend, which means you can use any one of them among TensorFlow, CNTK, and Theano with Keras as a backend according to your requirement.
- Since it has an easy deployment, it also holds support for cross-platform. Following are the devices on which Keras can be deployed:
 - iOS with CoreML
 - Android with TensorFlow Android
 - Web browser with .js support
 - Cloud engine
 - Raspberry pi
- It supports Data parallelism, which means Keras can be trained on multiple GPU's at an instance for speeding up the training time and processing a huge amount of data.

Disadvantages of Keras

The only disadvantage is that Keras has its own pre-configured layers, and if you want to create an abstract layer, it won't let you because it cannot handle low-level APIs. It only supports high-level API running on the top of the backend engine (TensorFlow, Theano, and CNTK).



Figure 12 Keras

Pillow

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3.

Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as “jpeg”, “png”, “bmp”, “gif”, “ppm”, “tiff”. You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

Image Archives

The Python Imaging Library is best suited for image archival and batch processing applications. Python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

Image Display

You can display images using Tk PhotoImage, BitmapImage and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits.

For debugging purposes, there is a show () method to save the image to disk which calls the external display utility.

Image Processing

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

What is Image Processing?

Image processing is a method of analyzing and manipulating digital images. The main goal is to extracting some information that can be used somewhere else. Let's make it simple to understand with a real-life example.

For example - Nowadays, CCTV cameras are instated in the traffic light that capture digital images, identify the vehicle's number plate, and check whether the vehicle violated the traffic rules or not. All this process is related to image processing.

Numpy

What is NumPy?

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

NumPy stands for Numerical Python.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including 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.

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

Travis Oliphant created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed.

NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

The need of NumPy

With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

There are the following advantages of using NumPy for data analysis.

- NumPy performs array-oriented computing.
- It efficiently implements the multidimensional arrays.
- It performs scientific computations.
- It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
- NumPy provides the in-built functions for linear algebra and random number generation.

Nowadays, NumPy in combination with SciPy and Matplotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB.

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

Which Language is NumPy written in?

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.



Figure 13 NumPy

Tqdm

tqdm is a library in Python which is used for creating Progress Meters or Progress Bars. tqdm got its name from the Arabic name taqaddum which means ‘progress’.

Implementing tqdm can be done effortlessly in our loops, functions or even Pandas. Progress bars are pretty useful in Python because:

- One can see if the Kernel is still working.
- Progress Bars are visually appealing to the eyes.
- It gives Code Execution Time and Estimated Time for the code to complete which would help while working on huge datasets

Progress bars are filled up according to the percentage of progress made in accomplishing a task. The progress can be calculated by dividing `number_of_item_processed` by `total_input_item`. Various factors affect the progress bar, such as network speed, latency, and if persisting data into local storage to derive a more accurate ETA (Estimated Time of Arrival).

We can create simple and hassle-free progress bars using the Python external library named `tqdm`. We can add it to the code and make it look lovely.

The `tqdm` stands for *taqadum* in Arabic, which means progress. Python `tqdm` module works on various platform such Linux, Window, Mac, etc. and it is also compatible with the IPython/Jupyter notebooks.

Need for Progress Bar?

If we are working with the smaller data set, the progress will not bother in our workflow. However, the progress bar can be used for iterating over a dataset, training a model, or encoding a large information set.

- The progress bar provides us an estimation of the process that has been given the approximation of the time it might take more.
- It gives us information that the progress is still running and has not been terminated rudely.

Jupyter Notebook

Jupyter Notebook is an open-source, web-based interactive environment, which allows you to create and share documents that contain live code, mathematical equations, graphics, maps, plots, visualizations, and narrative text. It integrates with many programming languages like Python, PHP, R, C#, etc.

Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

Advantages of Jupyter Notebook

There are the following advantages of Jupyter Notebook

- All in one place: As you know, Jupyter Notebook is an open-source web-based interactive environment that combines code, text, images, videos, mathematical equations, plots, maps, graphical user interface and widgets to a single document.
- Easy to convert: Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows you to render a publicly available notebook in the browser directly.
- Easy to share: Jupyter Notebooks are saved in the structured text files (JSON format), which makes them easily shareable.
- Language independent: Jupyter Notebook is platform-independent because it is represented as JSON (JavaScript Object Notation) format, which is a language-independent, text-based file format. Another reason is that the notebook can be

processed by any programming language, and can be converted to any file formats such as Markdown, HTML, PDF, and others.

- Interactive code: Jupyter notebook uses ipywidgets packages, which provide many common user interfaces for exploring code and data interactivity.

Disadvantages of Jupyter Notebook

There are the following disadvantages of Jupyter Notebook:

- It is very hard to test long asynchronous tasks.
- Less Security
- It runs cell out of order
- In Jupyter notebook, there is no IDE integration, no linting, and no code-style correction.



Figure 14 Jupyter Notebook

VS code

Visual Studio Code (famously known as VS Code) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Dockerfile. Moreover, VS Code allows you to add on and even creating new extensions including code linters, debuggers, and cloud and web development support.

The VS Code user interface allows for a lot of interaction compared to other text editors.

To simplify user experience, VS Code is divided into five main regions:

- The activity bar
- The side bar
- Editor groups
- The panel
- The status bar

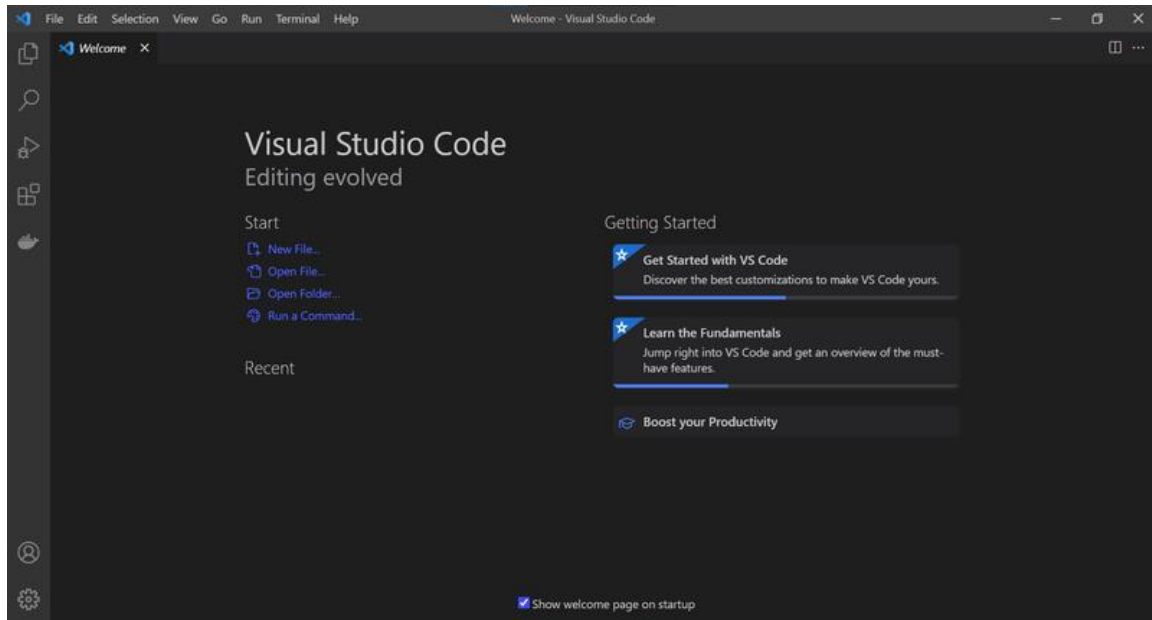


Figure 15 VS code

Project File Structure

- haar cascade files:
having 3 files:
 - ❖ haarcascade_frontalface_alt.xml
 - ❖ haarcascade_lefteye_2splits.xml
 - ❖ haarcascade_righteye_2splits.xml

What are Haar Cascades?

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location.

This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various objects like cars, bikes, buildings, fruits, etc.

Haar cascade uses the cascading window, and it tries to compute features in every window and classify whether it could be an object.

Sample haar features traverse in window-sized across the picture to compute and match features.

Haar cascade works as a classifier. It classifies positive data points → that are part of our detected object and negative data points → that don't contain our object.

- Haar cascades are fast and can work well in real-time.
- Haar cascade is not as accurate as modern object detection techniques are.
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

The below files will be created by us while making the project.

- Models – It will contain our trained models.
- Driver Monitoring System.ipynb – Jupyter notebook in which we train and build our Driver Monitoring System.
- main.py - Python file for Graphical User Interface
- Alarm.wav – sound file for the alarm

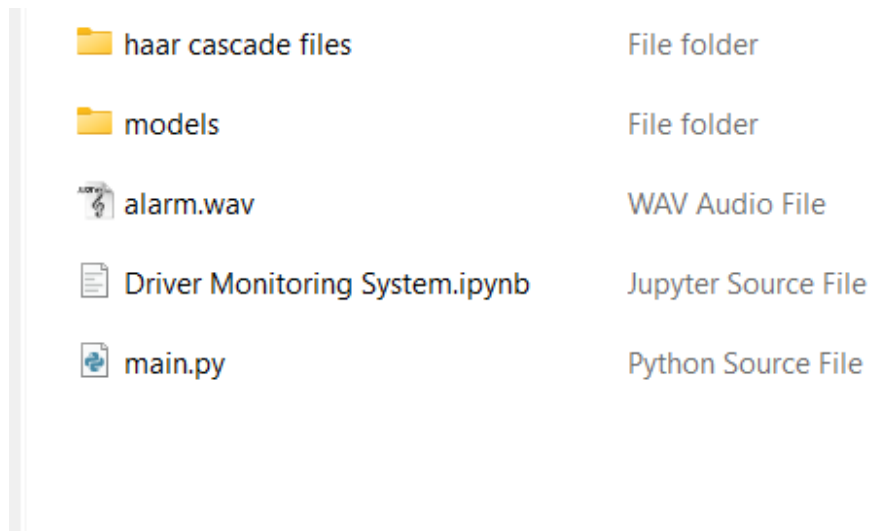


Figure 16 Project File Structure

Getting and Performing Data Cleaning

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning. Professional data scientists usually invest a very large portion of their time in this step because of the belief that "Better data beats fancier algorithms".

If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large.

Obviously, different types of data will require different types of cleaning.

Steps involved in Data Cleaning:

- Removal of unwanted observations

This includes deleting duplicate/ redundant or irrelevant values from your dataset. Duplicate observations most frequently arise during data collection and Irrelevant observations are those that don't actually fit the specific problem that you're trying to solve.

- Redundant observations alter the efficiency by a great extent as the data repeats and may add towards the correct side or towards the incorrect side, thereby producing unfaithful results.
- Irrelevant observations are any type of data that is of no use to us and can be removed directly.

- Fixing Structural errors

The errors that arise during measurement, transfer of data, or other similar situations are called structural errors. Structural errors include typos in the name of features, the same attribute with a different name, mislabeled classes, i.e. separate classes that should really be the same, or inconsistent capitalization.

For example, the model will treat America and America as different classes or values, though they represent the same value or red, yellow, and red-yellow as different classes or attributes, though one class can be included in the other two classes. So, these are some structural errors that make our model inefficient and give poor quality results.

- Managing Unwanted outliers

Outliers can cause problems with certain types of models. For example, linear regression models are less robust to outliers than decision tree models. Generally, we should not remove outliers until we have a legitimate reason to remove them. Sometimes, removing them improves performance, sometimes not. So, one must have a good reason to remove the outlier, such as suspicious measurements that are unlikely to be part of real data.

- Handling missing data

Missing data is a deceptively tricky issue in machine learning. We cannot just ignore or remove the missing observation. They must be handled carefully as they can be an indication of something important. The two most common ways to deal with missing data are:

- ❖ Dropping observations with missing values.

The fact that the value was missing may be informative in itself.

Plus, in the real world, you often need to make predictions on new data even if some of the features are missing!

- ❖ Imputing the missing values from past observations.

Again, “missingness” is almost always informative in itself, and you should tell your algorithm if a value was missing.

Even if you build a model to impute your values, you’re not adding any real information. You’re just reinforcing the patterns already provided by other features.

Missing data is like missing a puzzle piece. If you drop it, that’s like pretending the puzzle slot isn’t there. If you impute it, that’s like trying to squeeze in a piece from somewhere else in the puzzle.

So, missing data is always an informative and an indication of something important. And we must be aware of our algorithm of missing data by flagging it. By using this technique of flagging and filling, you are essentially allowing the algorithm to estimate the optimal constant for missingness, instead of just filling it in with the mean.

```
In [3]: plt.imshow(img_array, cmap="gray")
Out[3]: <matplotlib.image.AxesImage at 0x1fdb8b8370>
```



```
In [4]: img_array.shape
Out[4]: (86, 86)
```

```
In [5]: import os
Datadirectory = r'C:\Users\Eskander\Documents\Marwadi projects\Driver monitoring system\project\Data_Set'
Classes = ['Closed_Eyes', 'Open_Eyes']

for category in Classes:
    path = os.path.join(Datadirectory, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
        backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)
        plt.imshow(img_array, cmap="gray")
        plt.show()
        break
#break
```

Figure 17 performing Data cleaning

```

In [6]: img_size = 224
new_array = cv2.resize(backtorgb, (img_size, img_size))
plt.imshow(new_array, cmap="gray")
plt.show()

In [7]: training_data = []

def create_training_data():
    for category in Classes:
        path = os.path.join(Datadirectory, category)
        class_num = Classes.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)
                new_array = cv2.resize(backtorgb, (img_size, img_size))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass

In [8]: create_training_data()

```

Figure 18 performing Data cleaning

Extracting the Feature from MobileNet () model:

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

Feature extraction can be accomplished manually or automatically:

- ❖ Manual feature extraction requires identifying and describing the features that are relevant for a given problem and implementing a way to extract those features. In many situations, having a good understanding of the background or domain can help make informed decisions as to which features could be useful. Over decades of research, engineers and scientists have developed feature extraction methods for images, signals, and text. An example of a simple feature is the mean of a window in a signal.
- ❖ Automated feature extraction uses specialized algorithms or deep networks to extract features automatically from signals or images without the need for human intervention. This technique can be very useful when you want to move quickly from raw data to developing machine learning algorithms. Wavelet scattering is an example of automated feature extraction.

With the ascent of deep learning, feature extraction has been largely replaced by the first layers of deep networks – but mostly for image data. For signal and time-series

applications, feature extraction remains the first challenge that requires significant expertise before one can build effective predictive models.

What is MobileNet?

MobileNet is a CNN architecture model for Image Classification and Mobile Vision. There are other models as well but what makes MobileNet special is that it has very less computation power to run or apply transfer learning. This makes it a perfect fit for Mobile devices, embedded systems and computers without GPU or low computational efficiency with compromising significantly with the accuracy of the results. It is also best suited for web browsers as browsers have limitation over computation, graphic processing and storage.

MobileNet Architecture

- ❖ MobileNets for mobile and embedded vision applications is proposed, which are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks.
- ❖ Two simple global hyper-parameters that efficiently trade off between latency and accuracy are introduced.
- ❖ The core layer of MobileNet is depthwise separable filters, named as Depthwise Separable Convolution. The network structure is another factor to boost the performance. Finally, the width and resolution can be tuned to trade off between latency and accuracy.

```
In [14]: #pip install keras==2.6

In [15]: import tensorflow as tf
         from tensorflow import keras
         from tensorflow.keras import layers

In [16]: model = tf.keras.applications.mobilenet.MobileNet()

In [17]: model.summary()
=====
input_1 (InputLayer)      [(None, 224, 224, 3)]      0
conv1 (Conv2D)            (None, 112, 112, 32)      864
conv1_bn (BatchNormaliza (None, 112, 112, 32)      128
conv1_relu (ReLU)         (None, 112, 112, 32)      0
conv_dw_1 (DepthwiseConv2D) (None, 112, 112, 32)      288
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
conv_dw_1_relu (ReLU)     (None, 112, 112, 32)      0
conv_pw_1 (Conv2D)        (None, 112, 112, 64)      2048
conv_pw_1_bn (BatchNormaliza (None, 112, 112, 64)      256
...
In [18]: base_input = model.layers[0].input

In [19]: base_output = model.layers[-4].output

In [20]: Flat_layer = layers.Flatten()(base_output)
         final_output = layers.Dense(1)(Flat_layer)
```

Figure 19 Extracting the Feature

This technique is also called transfer learning, we use the pre-trained model that have been already trained on large datasets and extract the features from these models and use them for our tasks. We are using the MobileNet () which has been trained on large dataset. We can directly import this model from the keras.applications library.

Building the Model

A machine learning model is defined as a mathematical representation of the output of the training process. Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

There are various types of machine learning models available based on different business goals and data sets.


```

In [18]: base_input = model.layers[0].input

In [19]: base_output = model.layers[-4].output

In [20]: flat_layer = layers.Flatten()(base_output)
         final_output = layers.Dense(1)(flat_layer)
         final_output = layers.Activation('sigmoid')(final_output)

In [21]: new_model = keras.Model(inputs = base_input, outputs = final_output)

In [22]: new_model.summary()

```

conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormalization)	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0

```

In [23]: new_model.compile(loss="binary_crossentropy", optimizer = "adam", metrics = ["accuracy"])

In [25]: new_model.fit(X,Y, validation_split = 0.1, epochs = 3 )

```

Figure 20 Loading the model

Training the model

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.

In Machine Learning, whenever you want to train a model with some data, then Epoch refers to one complete pass of the training dataset through the algorithm. Moreover, it takes a few epochs while training a machine learning model, but, in this scenario, you will face an issue while feeding a bunch of training data in the model. This issue happens due to limitations of computer storage. To overcome this issue, we have to break the training data into small batches according to the computer memory or storage capacity. Then only we can train a machine learning model by feeding these batches without any hassle. This process is called batch in machine learning, and further, when all batches are fed exactly once to train the model, then this entire procedure is known as Epoch in Machine Learning. Epochs are defined as the total number of iterations for training the machine learning model with all the training data in one cycle. In Epoch, all training data is used exactly once.

Further, in other words, Epoch can also be understood as the total number of passes an algorithm has completed around the training dataset. A forward and a backward pass together counted as one pass in training.

Usually, when a machine learning model is trained, then it requires a little number of Epochs. An Epoch is often mixed up with iteration.

What is Iteration?

Iteration is defined as a total number of batches required to complete one epoch, where a number of batches are equal to the total number of iterations for one epoch.

Let's understand the iteration and epoch with an example, where we have 3000 training examples that we are going to use to train a machine learning model.

In the above scenario, we can break up the training dataset into sizeable batches. So let's suppose we have considered the batches of 500 examples in each batch, then it will take 6 iterations to complete 1 Epoch.

Mathematically, we can understand it as follows:

- Total number of training examples = 3000;
- Assume each batch size = 500;
- Then the total number of Iterations = Total number of training examples/Individual batch size = $3000/500$
- Total number of iterations = 6
- And 1 Epoch = 6 Iterations

What is Batch in Machine Learning?

Batch size is defined as the total number of training examples that exist in a single batch.

You can understand batch with the above-mentioned example also, where we have divided the entire training dataset/examples into different batches or sets or parts.

Let's understand the concept of mixing up an Epoch and iteration with the below example where we have considered 1000 datasets as shown in the below image.

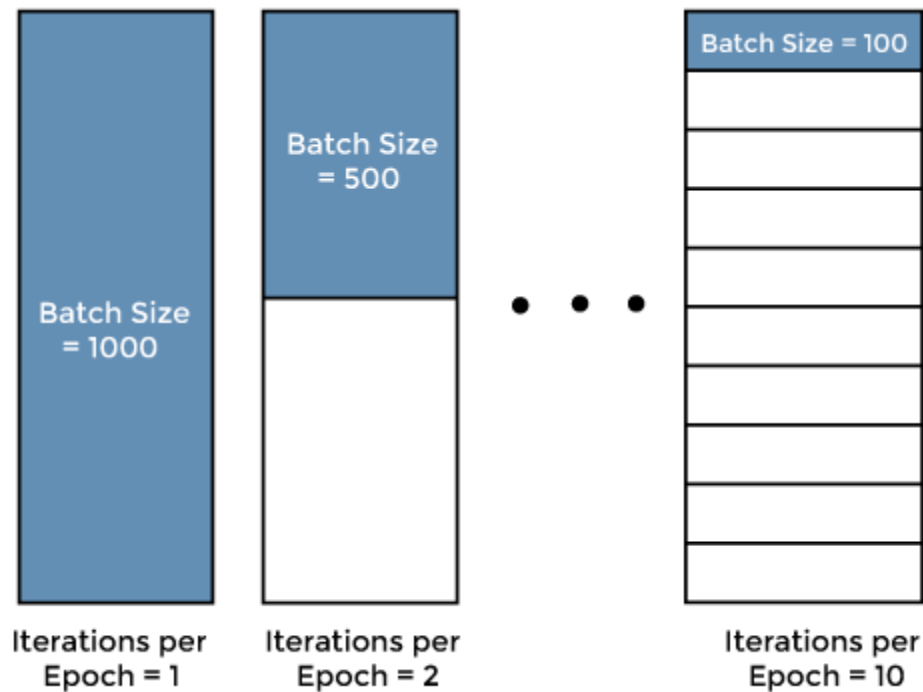


Figure 21 Batch example

In the above figure, we can understand this concept as follows:

- If the Batch size is 1000, then an epoch will complete in one iteration.
- If the Batch size is 500, then an epoch will complete in 2 iterations.

Similarly, if the batch size is too small or such as 100, then the epoch will be complete in 10 iterations. So, as a result, we can conclude that for each epoch, the required number of iterations times the batch size gives the number of data points. However, we can use multiple numbers epochs for training the machine learning model.

There are a few important points that everyone should keep in mind during training a machine learning model. These are as follows:

- Epoch is a machine learning terminology that refers to the number of passes the training data goes through machine learning algorithm during the entire data points.
- If there is a large amount of data available, then you can divide entire data sets into common groups or batches.
- The process of running one batch through the learning model is known as iteration. In Machine Learning, one cycle in entire training data sets is called an Epoch. However, in ideal conditions, one cycle in entire training data sets is called an Epoch but training a model typically requires multiple numbers of Epochs.
- Better generalization can be achieved with new inputs by using more epochs in the training of the machine learning model.

- Given the complexity and variety of data in real-world applications, hundreds to thousands of epochs may be required to achieve reasonable test data correctness.

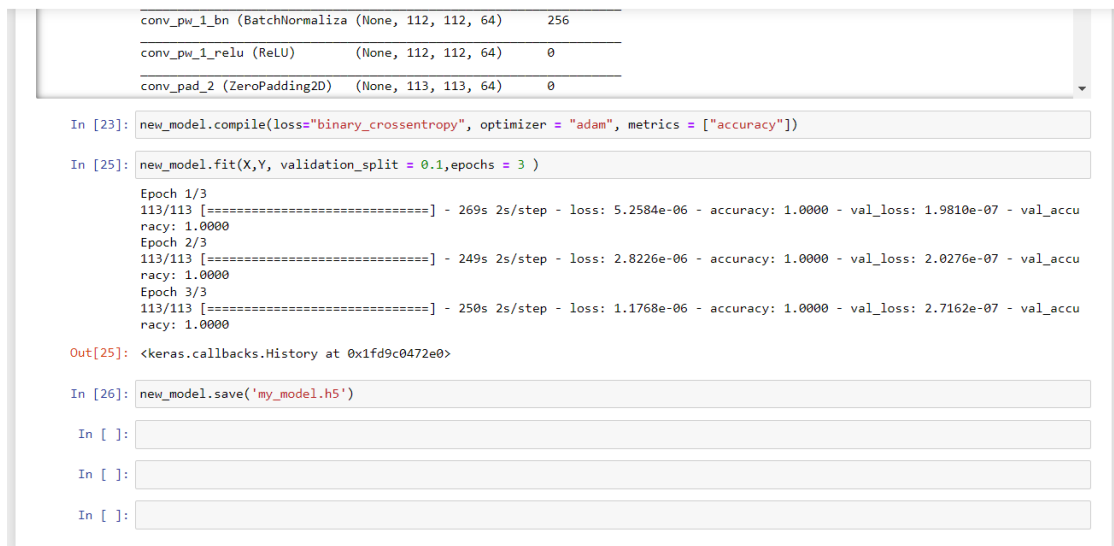
Furthermore, the term epoch has several definitions depending on the topic at hand.

Why use more than one Epoch?

It may not look correct that passing the entire dataset through an ML algorithm or neural network is not enough, and we need to pass it multiple times to the same algorithm.

So it needs to be kept in mind that to optimize the learning, we use gradient descent, an iterative process. Hence, it is not enough to update the weights with a single pass or one epoch.

Moreover, one epoch may lead to overfitting in the model.



conv_pw_1_bn	(BatchNormaliza (None, 112, 112, 64)	256
conv_pw_1_relu	(ReLU) (None, 112, 112, 64)	0
conv_pad_2	(ZeroPadding2D) (None, 113, 113, 64)	0

```

In [23]: new_model.compile(loss="binary_crossentropy", optimizer = "adam", metrics = ["accuracy"])

In [25]: new_model.fit(X,Y, validation_split = 0.1,epochs = 3 )

Epoch 1/3
113/113 [=====] - 269s 2s/step - loss: 5.2584e-06 - accuracy: 1.0000 - val_loss: 1.9810e-07 - val_accu
racy: 1.0000
Epoch 2/3
113/113 [=====] - 249s 2s/step - loss: 2.8226e-06 - accuracy: 1.0000 - val_loss: 2.0276e-07 - val_accu
racy: 1.0000
Epoch 3/3
113/113 [=====] - 250s 2s/step - loss: 1.1768e-06 - accuracy: 1.0000 - val_loss: 2.7162e-07 - val_accu
racy: 1.0000

Out[25]: <keras.callbacks.History at 0x1fd9c0472e0>

In [26]: new_model.save('my_model.h5')

In [ ]:

In [ ]:

In [ ]:

```

Figure 22 Training the model

Results and Reports

The model has been trained, now, we will make a separate file main.py which will load the model create the GUI.

```

1  import cv2
2  import os
3  from keras.models import load_model
4  import numpy as np
5  from pygame import mixer
6  import time
7
8
9  mixer.init()
10 sound = mixer.Sound("alarm.wav")
11
12 face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
13 leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
14 reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')
15
16
17
18 lbl=['Close','Open']
19
20 model = load_model('models/cnn_cat2.h5')
21 path = os.getcwd()
22 cap = cv2.VideoCapture(0)
23 font = cv2.FONT_HERSHEY_COMPLEX_SMALL
24 count=0
25 score=0
26 thicc=2
27 rpred=[99]
28 lpred=[99]
29
30 while(True):
31     ret, frame = cap.read()
32     height,width = frame.shape[:2]
33
34     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
35
36     faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))

```

Figure 23 building the GUI

```

rpred=[99]
lpred=[99]

while(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
    left_eye = leye.detectMultiScale(gray)
    right_eye = reye.detectMultiScale(gray)

    cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) , thickness=cv2.FILLED )

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

    for (x,y,w,h) in right_eye:
        r_eye=frame[y:y+h,x:x+w]
        count=count+1
        r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye,(24,24))
        r_eye= r_eye/255
        r_eye= r_eye.reshape(24,24,-1)
        r_eye = np.expand_dims(r_eye,axis=0)
        rpred = np.argmax(model.predict(r_eye), axis=-1)
        if(rpred[0]==1):
            lbl='open'
        if(rpred[0]==0):
            lbl='closed'
        break

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye= l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = np.argmax(model.predict(l_eye), axis=-1)
        if(lpred[0]==1):
            lbl='open'
        if(lpred[0]==0):
            lbl='closed'
        break

```

Figure 24 building the GUI

Snapshots

```
In [4]: img_array.shape
Out[4]: (86, 86)

In [5]: import os
Datadirectory = r'...'
Classes = ['Closed_Eyes', 'Open_Eyes']

for category in Classes:
    path = os.path.join(Datadirectory, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
        backtorgb = cv2.cvtColor(img_array,cv2.COLOR_GRAY2RGB)
        plt.imshow(img_array, cmap="gray")
        plt.show()
        break
#break
```



Figure 25 Importing the Data-set

```
In [6]: img_size = 224
new_array = cv2.resize(backtorgb, (img_size,img_size))
plt.imshow(new_array, cmap="gray")
plt.show()

In [7]: training_data = []

def create_training_data():
    for category in Classes:
        path = os.path.join(Datadirectory, category)
        class_num = Classes.index(category)
        for img in os.listdir(path):
            try :
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
                backtorgb = cv2.cvtColor(img_array,cv2.COLOR_GRAY2RGB)
                new_array = cv2.resize(backtorgb, (img_size,img_size))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass

In [8]: create_training_data()
```

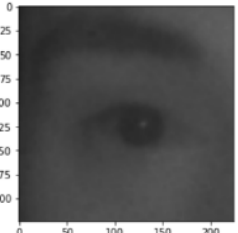


Figure 26 Performing Data pre-processing

```

In [11]: X = []
        y = []
        for features, label in training_data:
            X.append(features)
            y.append(label)

        X = np.array(X).reshape(-1, img_size, img_size, 3)

In [12]: X = X/255.0

In [13]: Y = np.array(y)

In [14]: #pip install keras==2.6

In [15]: import tensorflow as tf
        from tensorflow import keras
        from tensorflow.keras import layers

In [16]: model = tf.keras.applications.mobilenet.MobileNet()

In [17]: model.summary()

```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128

Figure 27 Loading Mobile-Net () model

```

conv_pw_1 (Conv2D) (None, 112, 112, 64) 2048

In [18]: base_input = model.layers[0].input

In [19]: base_output = model.layers[-4].output

In [20]: flat_layer = layers.Flatten()(base_output)
        final_output = layers.Dense(1)(flat_layer)
        final_output = layers.Activation('sigmoid')(final_output)

In [21]: new_model = keras.Model(inputs = base_input, outputs = final_output)

In [22]: new_model.summary()

```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048

```

In [23]: new_model.compile(loss="binary_crossentropy", optimizer = "adam", metrics = ["accuracy"])

```

Figure 28 Building our model

```

In [23]: new_model.compile(loss="binary_crossentropy", optimizer = "adam", metrics = ["accuracy"])

In [25]: new_model.fit(X,Y, validation_split = 0.1,epochs = 3 )

Epoch 1/3
113/113 [=====] - 269s 2s/step - loss: 5.2584e-06 - accuracy: 1.0000 - val_loss: 1.9810e-07 - val_accu
racy: 1.0000
Epoch 2/3
113/113 [=====] - 249s 2s/step - loss: 2.8226e-06 - accuracy: 1.0000 - val_loss: 2.0276e-07 - val_accu
racy: 1.0000
Epoch 3/3
113/113 [=====] - 250s 2s/step - loss: 1.1768e-06 - accuracy: 1.0000 - val_loss: 2.7162e-07 - val_accu
racy: 1.0000

Out[25]: <keras.callbacks.History at 0x1fd9c0472e0>

In [26]: new_model.save('my_model.h5')

In [ ]:

In [ ]:

In [ ]:

```

Figure 29 Training and saving our model

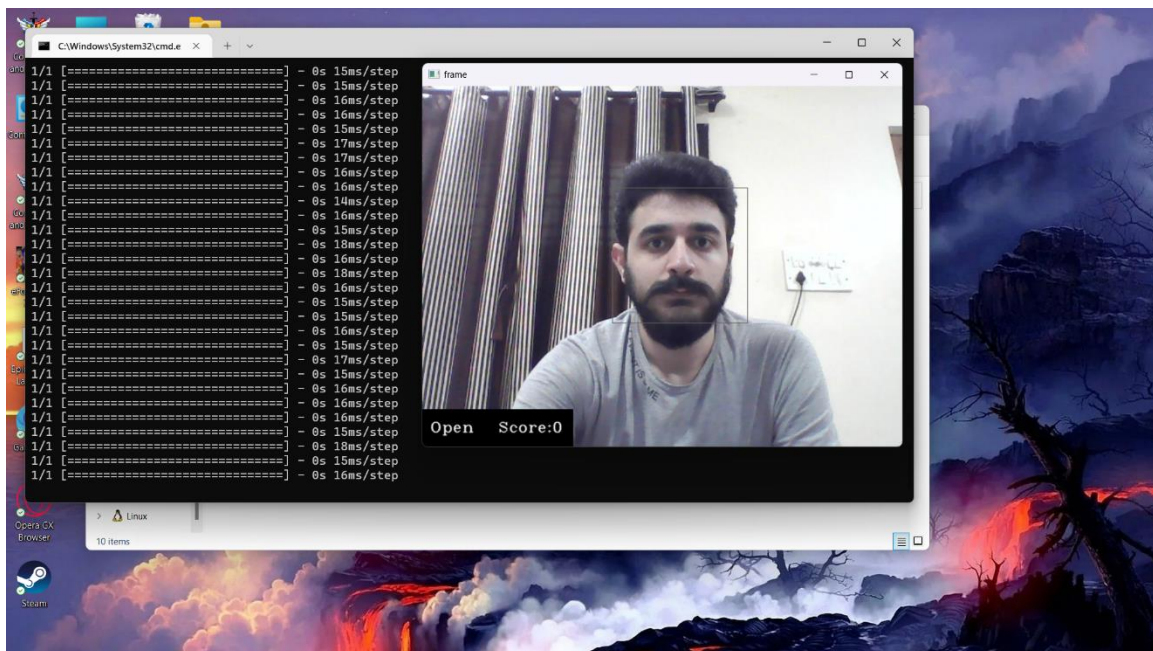


Figure 30 building the GUI

Testing and Verification

BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations. Although developed for translation, it can be used to evaluate text generated for a suite of natural language processing tasks.

The Bilingual Evaluation Understudy Score, or BLEU for short, is a metric for evaluating a generated sentence to a reference sentence. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0. The score was developed for

evaluating the predictions made by automatic machine translation systems. It is not perfect, but does offer 5 compelling benefits:

- ❖ It is quick and inexpensive to calculate.
- ❖ It is easy to understand.
- ❖ It is language independent.
- ❖ It correlates highly with human evaluation.
- ❖ It has been widely adopted.

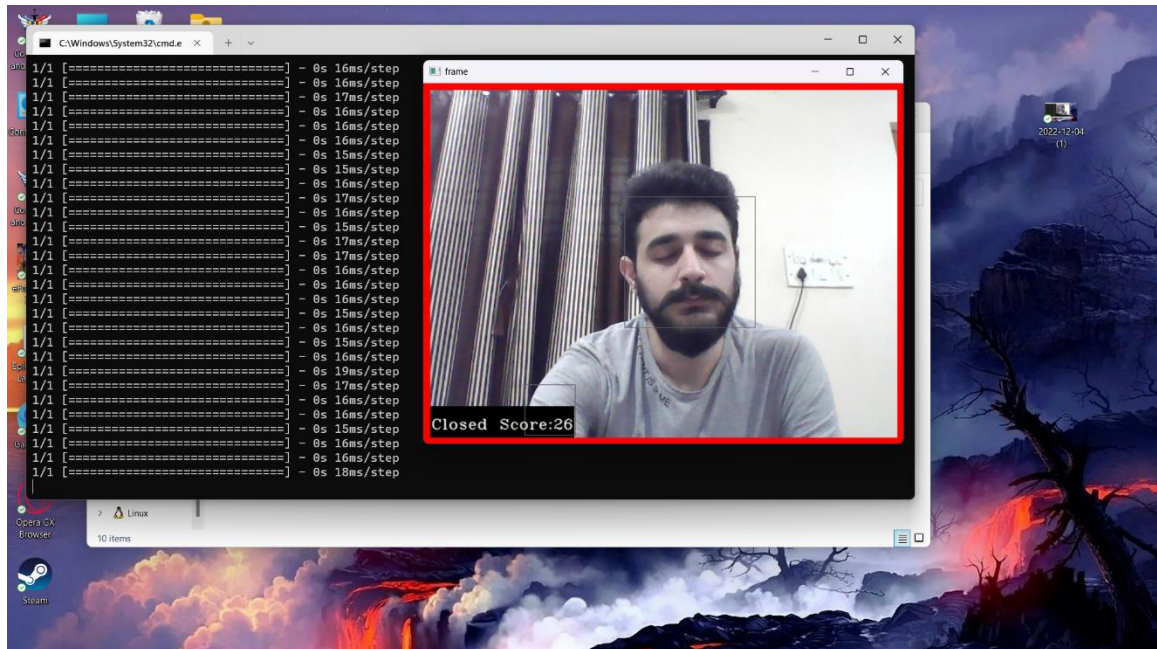


Figure 31 Testing with Closed eyes

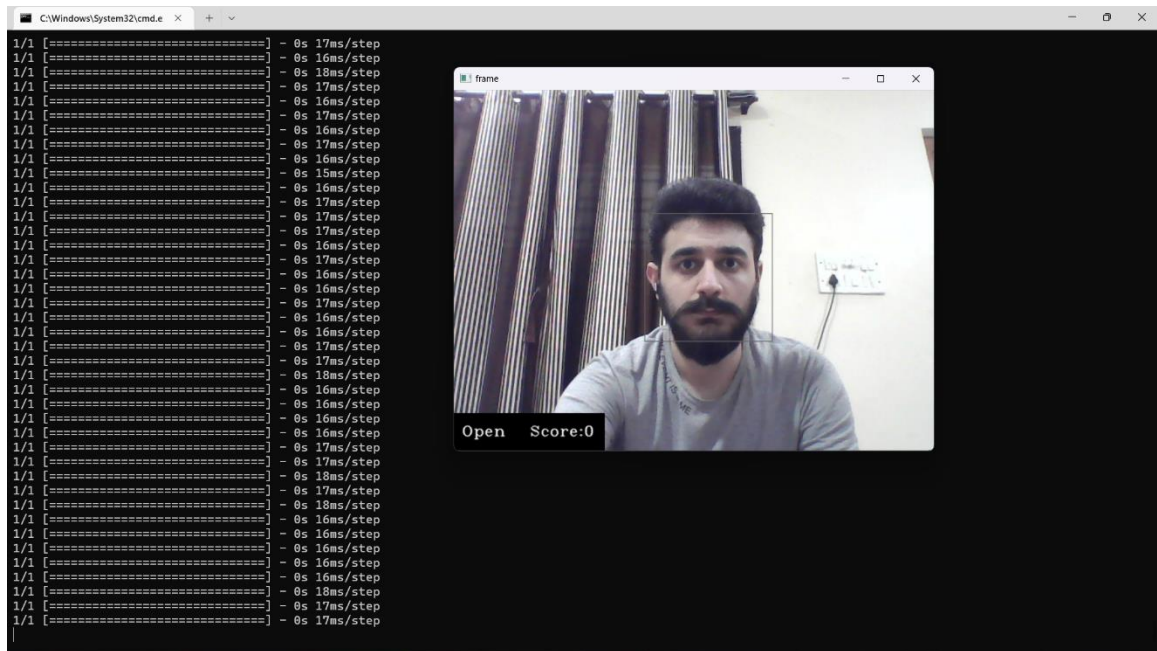


Figure 32 Testing with Open eyes

CONCLUSION

Summary of the results

After building the model and implementing the project we got the expected results, as we got a very high accuracy in detecting the drowsiness by following the eye status. And after testing the GUI we found it very reliable and user friendly

Advantages of Work / Results / Methodologies

In this project we propose a new task which aims to monitor drivers and protect them. Experiments on a new benchmark dataset collected show that our approach helped in detecting eye status of a driver to predict the drowsiness status and alert the driver to save him from an accident.

The drowsiness detection system is capable of detecting drowsiness in quickly. The system which can differentiate normal eye blink and drowsiness can prevent the driver from entering the state of sleepiness while driving. The system works well irrespective of driver wearing spectacles and under low light conditions also. During the monitoring, the system is able to decide if the eyes are closed or opened. When the eyes have been closed for too long a warning signal is issued. The ultimate goal of the system is to check the drowsiness condition of the driver. Based on the eye movements of the driver, the drowsiness is detected and according to eye blink, the alarm will be generated to alert the driver and to reduce the speed of the vehicle along with the indication of parking light. By doing this, many accidents will be reduced and provides safety to the driver and vehicle. A system that is driver safety and car security is presented only in luxurious costly cars. Using eye detection, driver security and safety can be implemented in normal car also.

Scope of future work

The main goal of this project is to develop a simulation system that can detect drowsiness using a webcam. The system must meet certain requirement which is detecting drowsiness throughout the video frames. Thus, the driver can avoid accident. Secondly, it has to detect only drowsiness signs so that the system will not misinterpret any random signs that it received from the driver. Improvement on the algorithms to detect eyes and mouth need to be done for future implementation. Luminance changes have to be encounter to ensure the detection of the gradient of eyes is sufficient to improve the detection results. The quality

of the video or images used in detecting drowsiness affects the result of the detection. Therefore, a good quality and high frame rate of images (number of pixel) is one of the factors to get better detection. Better techniques can be used to compare which technique is more reliable in detecting drowsiness. Implementing other method to detect drowsiness is also one of the improvements to the system so that it can ensure the system to be reliable to detect drowsiness. Besides that, a better internal specification of laptop or device can be used to run the system in order to obtain a smooth execution of the algorithm and a reliable system. Thus, by making this project successful, the numbers of road accident can be reduced when this project is implemented in the vehicle to detect the drowsiness of the driver.

The future works may focus on the utilization of outer factors such as vehicle states, sleeping hours, weather conditions, mechanical data, etc. for fatigue measurement. Driver drowsiness poses a major problem to highway safety. 24 hours operations, high annual mileage, exposure to the challenging environmental condition, and demanding work schedules all contribute to the serious safety issue. Monitoring the driver's state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measure to necessary to address this problem. Currently there is no adjustment in zoom or direction of the camera during operation. Future work may be automatically zoom in on eyes once they are localized. This would avoid trade-off between having wide field of view in order to locate the eyes, and narrow view in order to detect fatigue.

REFERENCES

- [1] Use of the hough transformation to detect lines and curves in picture; r. Duda and p. E.Hart.
- [2] K. Shu, S. Wang and H. Liu, "Understanding User Profiles on Social Media for Fake News Detection," 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2018, pp. 430-435, doi: 10.1109/MIPR.2018.00092.
- [3] 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," *Neuroscience & Biobehavioral Reviews*, 2012.
- [4] B. T. Jap, S. Lal, P. Fischer, and E. Bekiaris, "Using EEG spectral components to assess algorithms for detecting fatigue," *Expert Systems with Applications*, vol. 36, pp. 2352-2359, 2009.
- [5] D. Liu, P. Sun, Y. Xiao, and Y. Yin, "Drowsiness Detection Based on Eyelid Movement," in *Education Technology and Computer Science (ETCS)*, 2010 Second International Workshop on, 2010, pp. 49-52.
- [6] H. Seifoory, D. Taherkhani, B. Arzhang, Z. Eftekhari, and H. Memari, "An Accurate Morphological Drowsy Detection," ed: IEEE, 2011.
- [7] D. J. McKnight, "Method and apparatus for displaying grey-scale or color images from binary images," ed: Google Patents, 1998.
- [8] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Transactions on Graphics*, vol. 21, pp. 277-280, 2002.
- [9] I. Garcia, S. Bronte, L. Bergasa, N. Hernandez, B. Delgado, and M. Sevillano, "Visionbased drowsiness detector for a realistic driving simulator," in *Intelligent Transportation Systems (ITSC)*, 2010 13th International IEEE Conference on, 2010, pp. 887-894.
- [10] T. Danisman, I. M. Bilasco, C. Djeraba, and N. Ihaddadene, "Drowsy driver detection system using eye blink patterns," in *Machine and Web Intelligence (ICMWI)*, 2010 International Conference on, 2010, pp. 230-233.

- [11] D. F. Dinges and R. Grace, "PERCLOS: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance," Federal Highway Administration. Office of motor carriers, Tech. Rep. MCRT-98-006, 1998.37
- [12] S. T. Lin, Y. Y. Tan, P. Y. Chua, L. K. Tey, and C. H. Ang, "PERCLOS Threshold for Drowsiness Detection during Real Driving," *Journal of Vision*, vol. 12, pp. 546-546, 2012.
- [13] <https://www.kaggle.com/datasets/tauilabdelilah/mrl-eye-dataset>