DROWSINESS MONITORING AND STARMING SYSTEM USING LANDMARK DETUCTION AND KNN

A PROJECT REPORT

Submitted by

VENKATESH PRASAD.S [REGISTER NO: 211419104299]

VASANTHA KUMAR.R [REGISTER NO: 211419104297]

ROHITH.T [REGISTER NO: 211419104224]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2023

DROWSINESS MONITORING AND STARMING SYSTEM USING LANDMARK DETUCTION AND KNN

A PROJECT REPORT

Submitted by

VENKATESH PRASAD.S [REGISTER NO: 211419104299]

VASANTHA KUMAR.R [REGISTER NO: 211419104297]

ROHITH.T [REGISTER NO: 211419104224]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"DROWSINESS MONITORING AND STARMING SYSTEM USING LANDMARK DETUCTION AND KNN"** is the bonafide work of

"**VENKATESH PRASAD.S (REGISTER NO: 211419104299)**, **VASANTHA KUMAR.R (REGISTER NO: 211419104297), ROHITH.T (REGISTER NO: 211419104224)**" who carried out the project work under my supervision.

SIGNATURE

Dr.L.JABASHEELA, M.E., Ph.D,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,

POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mr.M.Krishnamoorthy, ME, MBA, (Ph.D)
SUPERVISOR
ASSOCIATE PROFESSOR

DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,

POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) were examined in the Anna University

End semester Examination held on 10.04.2023

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **VENKATESH PRASAD.S (REGISTER NO: 211419104299), VASANTHA KUMAR.R (REGISTER NO: 211419104297), ROHITH.T (REGISTER NO: 211419104224)** hereby declare that this project report titled **"DROWSINESS MONITORING AND STARMING SYSTEM USING LANDMARK DETUCTION AND KNN"** under the guidance of Mr.M.Krishnamoorthy is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Dr.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR,M.E.,** and **Dr. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the Head of the Department, **Dr.L.JABASHEELA, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mr.M.Krishnamoorthy,ME,MBA,(Ph.D) Associate Professor** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

VENKATESH PRASAD.S

VASANTHA KUMAR.R

ROHITH.T

ABSTRACT

On an average 1200 road accidents record daily in India out of which 400 leads to direct death and rest gets effected badly. The major reason of these accidents is drowsiness caused by both sleep and alcohol. Due to driving for long time or intoxication, drivers might feel sleepy which the biggest distraction for them while driving. This distraction might cost death of driver and other passengers in the vehicle and at the same time it also causes death of people in the other vehicles and pedestrians too. This mistake of one person on road would take their own life and also takes lives of other and put respective families in sorrow and tough situations

To prevent such accidents we, team 5A propose a system which alerts the driver if he/she feels drowsy. To accomplish this, we implement the solution using computer-vision based machine learning model. The driver's face is detected by face recognition algorithm continuously using a camera and the face of the driver is captured. The face of the driver is given as input to a classification algorithm which is trained with a data set of images of drowsy and non-drowsy faces. The algorithm uses landmark detection to classify the face as drowsy or not drowsy. If the driver's face is drowsy, a voice alert is generated by the system. This alert can make the driver aware that he/she is feeling drowsy and the necessary actions can then be taken by the driver. This system can be used in any vehicle on the road to ensure safety of the people who are travelling and prevent accidents which are caused due to the drowsiness of the driver

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

INTRODUCTION

Many road accidents which lead to death are because of drowsiness while driving. Drivers who drive long hours like truck drivers, bus drivers are likely to experience this problem. It is highly risky to drive with lack of sleep and driving for long hours will be more tiresome. Due to the drowsiness of the driver causes very dangerous consequences, it is estimated that 70,000 to 80,000 injures & crashes happen worldwide in a year. Even deaths have reached 1000-2000 every year. There are many unofficial deaths which are not confirmed by drivers that it was due to their drowsiness. This takes lives of many innocent people. It is a nightmare for a lot of people who travel across world. It is very important to identify the driver drowsiness and alert the driver to prevent crash.

The goal of this research is the detection of the indication of this fatigue of the driver. The acquisition system, processing system and warning system are the three blocks that are present in the detection system. The video of the driver's front face is captured by the acquisition system and it is transferred to the next stage i.e., processing block. The detection is processed if drowsiness of driver is detected, then the warning system gives a warning or alarm.

The methods to detect the drowsiness of the drive may be done using intrusive or nonintrusive method i.e., with and without the use of sensors connected to the driver. The cost of the system depends on the sensors used in the system. Addition of more parameters can increase the accuracy of the system to some extent. The motivation for the development of cost effective, and real-time driver drowsiness system with acceptable accuracy are the motivational factors of this work. Hence, the proposed system detects the fatigue of the driver from the facial images, and image processing technology and machine learning method are used to achieve a cost effective and portable system.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

2.1. Paper 1

Title: Drowsiness Detection System Utilizing Physiological Signals.

Author: Trupti K. Dange, T. S. Yengatiwar.

Year of publication: 2013.

Keywords: EOG, ECG, EEG, HRV, SVM, driver drowsiness detection.

The Physiological parameters-based techniques detect drowsiness based on drivers' physical conditions such as heart rate, pulse rate, breathing rate, respiratory rate and body temperature, etc. These biological parameters are more reliable and accurate in drowsiness detection as they are concerned with what is happening with driver physically. Fatigue or drowsiness, change the physiological parameters such as a decrease in blood pressure, heart rate and body temperature, etc. Physiological parameters-based drowsiness detection systems detect these changes and alert the driver when he is in the state, near to sleep. A list of physiological condition-based drowsiness detection system. These measures are invasive, so require electrodes to be directly placed on the driver's body.

1) EEG-BASED DRIVER FATIGUE DETECTION

The drivers' fatigue detection system using Electroencephalogram (EEG) signals is proposed to avoid the road accidents usually caused due to drivers' fatigue. The proposed method firstly finds the index related to different drowsiness levels. The system takes EEG signal as input which is calculated by a cheap single electrode neuro signal acquisition device. To evaluate the proposed method, data set for simulated car driver under the different levels of drowsiness is collected locally. And result shows that the proposed system can detect all subjects of tiredness.

2) WAVELET ANALYSIS OF HEART RATE VARIABILITY & SVM CLASSIFIER

Li and Chung [21] proposed the driver drowsiness detection that uses wavelet analysis of Heart Rate Variability (HRV) and Support Vector Machine (SVM) classifier. The basic purpose is to categorize the alert and drowsy drivers using the wavelet transform of HRV signals over short durations. The system firstly takes Photo Plethysmo Graphy (PPG) signal as input and divide it into 1-minute intervals and then verify two driving events using average percentage of eyelid closure over pupil over time (PERCLOS) measurement over the interval. Secondly, the system performs the feature extraction of HRV time series based on Fast Fourier Transform (FFT) and wavelet. A Receiver Operation Curve (ROC) and SVM classifier is used for feature extraction and 4 classification respectively. The analysis of ROC shows that the wavelet-based method gives improved results than the FFT-based method. Finally, the real time requirements for drowsiness detection, FFT and wavelet features are used to train the SVM classifier extracted from the HRV signals.

3) PULSE SENSOR METHOD

Mostly, previous studies focus on the physical conditions of drivers to detect drowsiness. That's why Rahim detects the drowsy drivers using infrared heart-rate sensors or pulse sensors. The pulse sensor measures the heart pulse rate from drivers' finger or hand. The sensor is attached with the finger or hand, detects the amount of blood flowing through the finger. Then amount of the blood's oxygen is shown in the finger, which causes the infrared light to reflect off and to the transmitter. The sensor picks up the fluctuation of oxygen that are connected to the Arduino as microcontroller. Then, the heart pulse rate is visualizing by the software processing of HRV frequency domain. Experimental results show that LF/HF (Low to high frequency) ratio decreases as drivers go from the state of being awake to the drowsy and many road accidents can be avoided if an alert is sent on time.

4) WEARABLE DRIVER DROWSINESS DETECTION SYSTEM

Mobile based applications have been developed to detect the drowsiness of drivers. But mobile phones distract the drivers' attention and may cause accident. To address the issue, Lenget proposed the wearable-type drowsiness detection system. The system uses self-designed wrist band consists of PPG signal and galvanic skin response sensor. The data collected from the sensors are delivered to the mobile device which acts as the main evaluating unit. The collected data are examined with the motion sensors that are built-in in the mobiles. Then five features are extracted from the data: heart rate, respiratory rate, stress level, pulse rate variability, and adjustment counter. The features are moreover used as the computation parameters to the SVM classifier to determine the drowsiness state. The experimental results show that the accuracy of the proposed system reaches up to 98.02 %. Mobile phone generates graphical and vibrational alarm to alert the driver.

5) WIRELESS WEARABLES METHOD

To avoid the disastrous road accidents, Warwick proposed the idea for drowsiness detection system using wearable Bio sensor called Bio-harness. The system has two phases. In the first phase, the physiological data of driver is collected using bio-harness and then analyzes the data to find the key parameters like ECG, heart rate, posture and others related to the drowsiness. In the second phase, drowsiness detection algorithm will be designed and develop a mobile app to alert the drowsy drivers.

6) DRIVER FATIGUE DETECTION SYSTEM

Chellappa presents the Driver fatigue detection system. The basic of the system is to detect the drowsiness when the vehicle is in the motion. The system has three components: external hardware (sensors and camera), data processing module and alert 5 unit. Hardware unit communicates over the USB port with the rest of the system. Physiological and physical factors like pulse rate, yawning, closed eyes, blink

duration and others are continuously monitored using somatic sensor. The processing module uses the combination of the factors to detect drowsiness. In the end, alert unit alerts the driver at multiple stages according to the severity of the symptoms.

7) HYBRID APPROACH UTILIZING PHYSIOLOGICAL FEATURES

To improve the performance of detection, A waist proposed the hybrid method which integrates the features of ECG and EEG. The method firstly extracts the time and frequency domain features like time domain statistical descriptors, complexity measures and power spectral measures from EEG. Then using ECG, features like heart rate, HRV, low frequency, high frequency and LF/HF ratio. After that, subjective sleepiness is measured to study its relationship with drowsiness. To select only statistically significant features, t-tests is used that can differentiate between the drowsy and alert. The features extracted from ECG and EEG are integrated to study the improvements in the performance using SVM. The other main contribution is to study the channel reduction and its impact on the performance of detection. The method measures the differences between the drowsy and alert state from physiological data collected from the driving simulated-based study. Monotonous driving environment is used to induce the drowsiness in the participants. The proposed method demonstrated that combining ECG and EEG improves the performance of system in differentiating the drowsy and alert states, instead of using them alone. The analysis of channel reduction confirms that the accuracy level reaches to 80% by just combining the ECG and EEG. The performance of the system indicates that the proposed system is feasible for practical drowsiness detection system.

2.2. Paper 2

Title: Drowsiness Detection with OpenCV (Using Eye Aspect Ratio)

Author: Adrian Rosebrock.

Year of publication: 2017.

Keywords: EAR, SVM, eye blink detection.

A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed. Recent landmark detectors, trained on in-the wild datasets exhibit excellent robustness against a head orientation with respect to a camera, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye opening. The proposed algorithm therefore estimates the landmark positions, extracts a single scalar quantity – eye aspect ratio (EAR) – characterizing the eye opening in each frame. 6 Many methods have been proposed to automatically detect eye blinks in a video sequence. Several methods are based on motion estimation in the eye region. Typically, the face and eyes are detected by a Viola-Jones type detector. Next, motion in the eye area is estimated from optical flow, by sparse tracking, or by frame-to-frame intensity differencing and adaptive thres holding. Finally, a decision is made whether the eyes are or are not covered by eyelids. Nowadays, robust real-time facial landmark detectors that capture most of the characteristic points on a human face image, including eye corners and eyelids, are available, Most of the state-of-the-art landmark detectors formulate a regression problem, where a mapping from an image into landmark positions or into other landmark parameterization is learned. These modern landmark detectors are trained on “in-the-wild datasets” and they are thus robust to varying illumination, various facial expressions, and moderate non-frontal head rotations proposed method. The eye blink is a fast closing and reopening of a human eye. Each individual has a little bit different pattern of blinks. The pattern differs in the speed of closing and opening, a degree of squeezing the eye and in a blink duration. The eye blink lasts approximately 100-400ms.We propose to exploit state-

of-the-art facial landmark detectors to localize the eyes and eyelid contours. From the landmarks detected in the image, we derive the eye aspect ratio (EAR) that is used as an estimate of the eye opening state. Since. The per frame EAR may not necessarily recognize the eye blinks correctly, a classifier that takes a larger temporal window of a frame into account is trained. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. A real-time eye blink detection algorithm was presented. We quantitatively demonstrated that regression-based facial landmark detectors are precise enough to reliably estimate a level of eye openness. While they are robust to low image quality (low image resolution in a large extent) and in-the-wild phenomena as non-frontality, bad illumination, facial expressions, etc. The proposed SVM method that uses a temporal window of the eye aspect ratio (EAR), outperforms the EAR thresholding.

2.3. Paper 3

Title: Real Time Driver Fatigue Detection Based on SVM Algorithm.
Authors: Burcu Kir Savas, Yasar Becerkli.
Year of publication: 2018.
Keywords: fatigue detection, SVM, driving safety, video processing

PROPOSED SYSTEM

In this study, SVM based driver fatigue prediction system is proposed to increase driver safety. The proposed system has five stages: PERCLOS, count of yawn, internal zone of the mouth opening, count of eye blinking and head detection to extract attributes from video recordings. The classification stage is done with Support Vector Machine (SVM). While the YawDD dataset is used during the training phase of the classification, real-time video recordings are used during the test phase. SVM is a classification algorithm separating data items. This algorithm proposed by Vladimir N. Vapnik based on statistical learning theory. SVM, one of the machine learning methods, is widely used in the field of pattern recognition. The main purpose of the

SVM is to find the best hyper plane to distinguish the data given as two-class or multiclass. The study was carried out in two classes. Whereas label 0 means that the driver is tired, label 1 means the driver is non-tired. Thus, it is aimed to distinguish tired drivers (driver fatigue) from non-tired drivers. In training driving simulation experiments, we had 10 volunteers (5 men and 5 women) whose ages were between 18-30. All the volunteers drove training simulation during the experiments. Attributions obtained in real time during the driving when fatigue detection system was working are indicated. The results of driver fatigue detection were classified using SVM classification algorithm. The total 713 data's line each of which has five stage features were sampled from 10 volunteers. All the data in the study are divided into two groups: 80% training data set (568 data set) and 20% test data set (145 data set). In the study, cross validation was selected as 10.

Measurements are as follows:

- TP means that a real fatigue is detected;
- TN means that a non-fatigue situation is correctly detected as non-fatigue by the system;
- FP means that a non-fatigue situation is incorrectly detected as real fatigue;
- FN means that a real fatigue situation is incorrectly detected as non-fatigue.

The accuracy of driver fatigue calculated as Accuracy= (TP+TN)/(TP+TN+FP+FN) As a result of the study, a system was designed to investigate the effects of fatigue and 8 insomnia on drivers physical transient behaviors, and to simulate drowsy drivers as a result of research. In this system, behavioral detection model is used for detecting fatigue drivers. The proposed system has five stages: PERCLOS, count of yawn, internal zone of the mouth opening, count of eye blinking and head detection to extract attributes from video recordings. As a result, the system is classified as SVM in two classes (not fatigue / fatigue). In this study a Real Time Driver Fatigue Detection

SVM Based on SVM Algorithm is presented whose test] results showed that the accuracy rate of fatigue detection is up to 97.93%

2.4. Paper 4

Title: Driver drowsiness detection using ANN image processing.
Authors: T. Vesselenyi1, S. Moca1, A. Rus1, T. Mitran1, B. Tătaru1.
Keywords: EEG, EOG, ANN, and Deep Learning
Year of publication: 2017.

The study regarding the possibility to develop a drowsiness detection system for car drivers based on three types of methods: EEG and EOG signal processing and driver image analysis
EEG (Electroencephalography) and EOG (Electrooculography) signals measurement and on the eye state (closed or opened) image classification. The EEG method monitors the brain activity through a sensor placed on a specific part of the scalp, The EOG method tracks the eye movements by measuring the signals from the muscles which are acting on the eye. The eye image analysis can monitor the opened or the closed state of the eye The EEG and EOG sensors, electrodes which have to be fixed with a conductive gel and in most devices must transmit the signal by wire, present a major discomfort. Research in the field of advanced materials and MEMS technology may solve these problems, as for example the use of dry electrodes for EEG. EEG Developments field are supported by efforts to create brain – computer interfaces for different applications, including devices that help disabled people. In this research the central point is to distinguish between low and high alpha rhythm peaks, which can make the difference between alert and drowsy states. To use EOG signals acquisitioned from 3 sensors (EOG1, EOG2, EOG3). After preprocessing, four types of different signals were identified. The combination of these four types of signals give the information to distinguish between left, right, up and down movements of the eye. Drowsiness detection using the processing of the driver's eye images for the

classification of the driver's drowsy or alert state, artificial neural networks were used. Artificial neural networks are extensively used for the image classification. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Deep Belief Networks, Restricted Boltzmann Machines and Deep Autoencoders are all methods belonging to the Deep Learning. These methods are used in a wide range of applications, the image classification being one of the fields in which these are employed with success. Matlab Neural Network Toolbox with 1 layer ANN and the autoencoder module of the Deep Learning Toolbox were used in order to study if these methods can be applied for image classification of the driver's drowsiness. In order to analyze the drowsiness state of the driver 200 images of a driver during a regular driving process were acquisitioned. One hundred of these images contain opened eyes or half opened eyes images and another hundreds of the images contain closed eyes images. It was assumed that the drowsiness is linked to the images in which the driver has closed eyes, and the alert state is linked to the images in which the driver has opened eyes.

2.5. Paper 5

Title: Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State.

Authors: Venkata Rami Reddy Chirra, Srinivasulu Reddy Uyyala and Venkata Krishna Kishore Kolli.

Year of publication: 2019.

Keywords: Deep CNN, Feature extraction

PROPOSED DEEP CNN BASED DROWSINESS DETECTION SYSTEM

Proposed system algorithm

(1) Viola-jones face detection algorithm is used to detect the face the images and given as input to Viola-jones eye detection algorithm

(2) Once the face is detected, Viola-jones eye detection algorithm is used to extract the eye region from the facial images and given as input to CNN.
(3) CNN with four convolutional layers is used to extract the deep features and those features are passed to fully connected layer.
(4) Soft max layer in CNN classify the images in to sleepy or non-sleepy images.

Face detection and eye region extraction

Whole face region may not be required to detect the drowsiness but only eyes region is enough for detecting drowsiness. At first step by using the Viola-jones face detection algorithm face is detected from the images. Once the face is detected, Viola-jones eye detection algorithm is used to extract the eye region from the facial images. it is the first algorithm used for face detection. For the face detection the Viola-Jones algorithm having three techniques those are Haar-like features, Ada boost and Cascade classifier. In this work, Viola-Jones object detection algorithm with Haar cascade classifier was used and implemented using OPEN CV with python. Haar cascade classifier uses Haar features for detecting the face from images.

Feature extraction and classification

Feature extraction is one type of dimensionality reduction where useful parts of an image represented as a feature vector.

Convolutional neural network

Convolutional neural network (CNN) is used in the proposed system for detection of driver drowsiness. Since a feature vector is needed for each drowsy image to compare with existing features in a database to detect either drowsy or not. Usually CNNs requires fixed size images as input so preprocessing is required. The preprocessing includes extracting the key frames from video based on temporal changes and store in database. From these stored images, feature vectors are generated in convolution layers of CNN. These feature vectors are then used for the detecting the driver

drowsiness. CNN have layers like convolutional layers, pooling (max, min and average) layers, ReLU layer and fully connected layer. Convolution layer is having kernels (filters) and each kernel having width, depth and height. This layer produces the feature maps as a result of calculating the scalar product between the kernels and local regions of image. CNN uses pooling layers (Max or Average) to minimize the size of the feature maps to speed up calculations. In this layer, input image is divided into different regions then operations are performed on each region. In Max Pooling, a maximum value is selected for each region and places it in the corresponding place in the output. ReLU (Rectified Linear Units) is a nonlinear layer. The ReLU layer applies the max function on all the values in the input data and changes all the negative values to zero. The following equation shows the ReLU activation function.In this proposed work a new method is proposed for driver drowsiness detection based on eye state. This determines the state of the eye that is drowsy or non- drowsy and alert with an alarm when state of the eye is drowsy. Face and eye region are detected using ViolaJones detection algorithm. Stacked deep convolution neural network is developed to extract features and used for learning phase. A Soft Max layer in CNN classifier is used to classify the driver as sleep or non-sleep. Proposed system achieved 96.42% accuracy. Proposed system 465 effectively identifies the state of driver and alert with an alarm when the model predicts drowsy output state continuously. In future we will use transfer learning to improve the performance of the system.

CHAPTER 3
SYSTEM ANALYSIS

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

There are three types of general methods to detect driver drowsiness; they are, vehicle-based, behavior based and physiological-based methods. The steering wheel movement, the accelerator of vehicle or pattern of vehicle brakes, vehicle's speed, and deviation in position of lane are monitored continuously in the method which is based on vehicle. If there is any deviation in the values detected, it is considered as driver drowsiness.

The sensors are not connected to the driver and this measurement is nonintrusive. Visual behavior like blinking of eye, closing of eye, yawning, bending of head etc. are examined for drowsiness detection in behavioral based method. A simple camera is used to take images to be sent as input to SVM algorithm to identify the above features and are called nonintrusive measurement.

Monitoring the physiological signals like EOG, ECG, the heartbeat, EEG, pulse rate etc. helps in detecting driver drowsiness based on physiological method and they are intrusive measurement due to the direct connection of sensors to the driver. The current detection of the drowsiness methods are mainly based on the machine learning algorithms.

DISADVANTAGE

- Accuracy is low
- Requires high resolution sensors
- Complex algorithm

3.2 PROPOSED SYSTEM

Our approach uses low resolution cameras to detect the face of a person using the Deep learning model trained on the neural network , then a key point ectraction algorithm is applied so that key points in the face are extracted and the key points is provided to the custom KNN algorithm implemented on Tensor flow

ADVANTAGE

- High accuracy
- Works with all kind of basic sensors
- More secure

3.3 HARDWARE ENVIRONMENT

- Basic spec sensor (camera)
- Works on any windows, Linux, and other operating software.

3.4 SOFTWARE ENVIRONMENT

- **Python**

  Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

  Python is dynamically typed and garbage-collected. It supports multiple programming paradigms.

  Including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

  Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular programming languages.

- **C**

  C is a general-purpose computer programming language. It was created in the 1970s by Dennis Ritchie, and remains very widely used and influential. By design, C's features cleanly reflect the capabilities of the targeted CPUs. It has found lasting use in operating systems, device drivers, protocol stacks, though decreasingly for application software. C is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.

  A successor to the programming language B, C was originally developed at Bell Labs by Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages with C compilers available for practically all modern computer architectures and operating systems. C has been standardized by ANSI since 1989 (ANSI C) and by the International Organization for Standardization (ISO).

  C is an imperative procedural language, supporting structured programming, lexical variable scope and recursion, with a static type system. It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

  Since 2000, C has consistently ranked among the top two languages in the TIOBE index, a measure of the popularity of programming languages

- **Open CV**

  OpenCV (**Open Source Computer Vision Library**) is a library of programming functions mainly for real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). The library is cross-platform and licensed as free and open-source software under Apache License 2. Starting in 2011, OpenCV features GPU acceleration for real-time operations. OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

- **Tensor Flow**

  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 was developed by the Google Brain team for internal Google use in research and production.The initial version was released under the Apache License 2.0 in 2015.

  Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java.

  This flexibility lends itself to a range of applications in many different sectors.

  TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google’s Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the

system is general enough to be applicable in a wide variety of other domains as well. Google open-sourced TensorFlow in November 2015. 33 TensorFlow's popularity is due to many things, but primarily because of the computational graph concept, automatic differentiation, and the adaptability of the Tensorflow python API structure. This makes solving real problems with TensorFlow accessible to most programmers. Google's Tensorflow engine has a unique way of solving problems. This unique way allows for solving machine learning problems very efficiently

- **D-lib**

  D-Lib was an online magazine dedicated to digital library research and development. Past issues are available free of charge. The publication was financially supported by contributions from the D-Lib Alliance. Prior to April 2006, the magazine was sponsored by the Defense Advanced Research Project Agency (DARPA) on behalf of the Digital Libraries Initiative and by the National Science Foundation (NSF). Despite its important role in shaping the then emerging digital library community, D-Lib eventually folded due in part to an unsustainable funding model, the maturity of the field, and the rise of social media and blogs. After 22 years, 265 issues and 1062 articles D-Lib ceased publication in July 2017.

  D-Lib was innovative in many ways, including: HTML-only publishing (no PDF versions), open access, and use of persistent identifiers for articles (the Handle System and later DOIs (which are implemented using the Handle System)).

CHAPTER 4
SYSTEM DESIGN

SYSTEM DESIGN

4.1. ER diagram

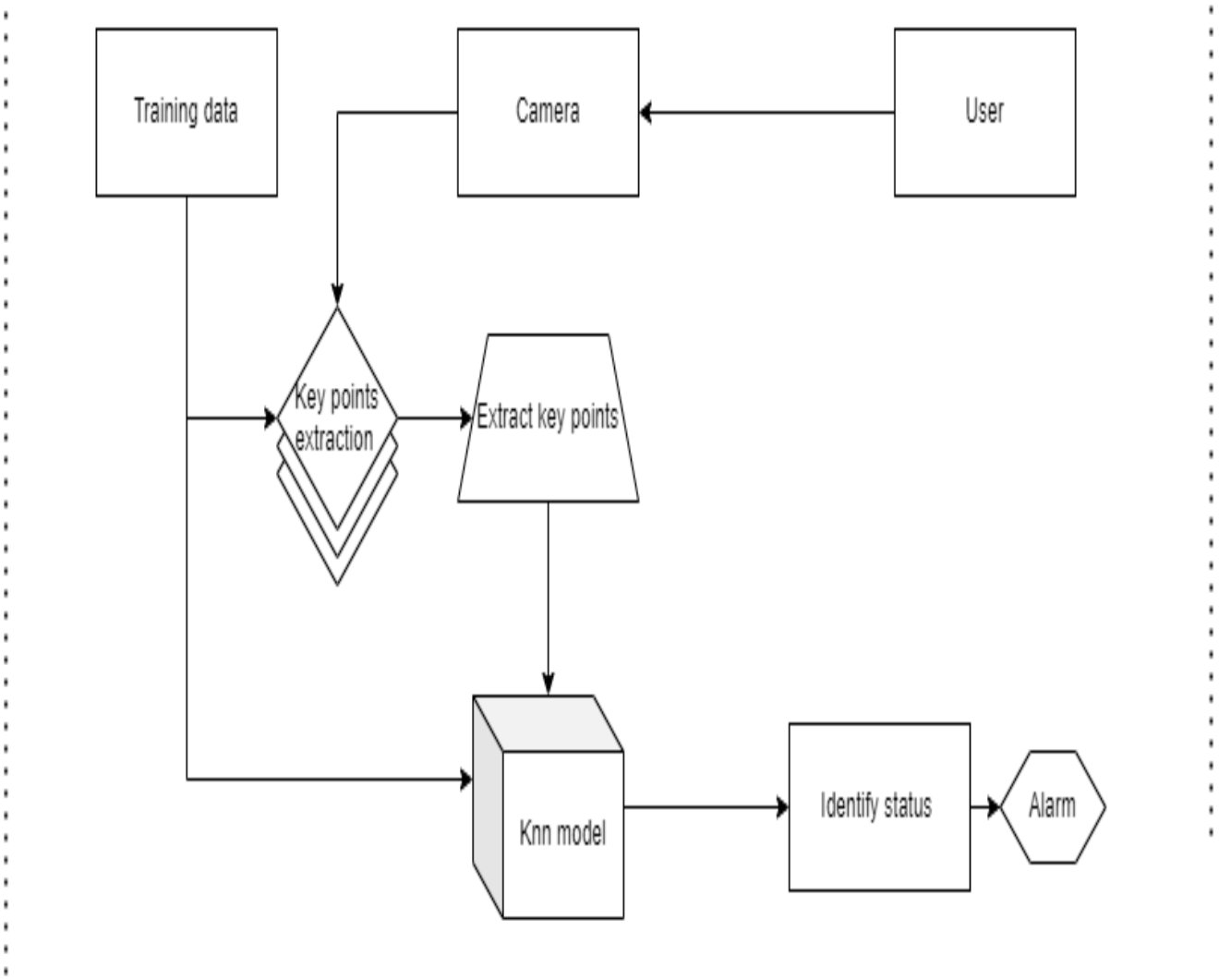


Fig 4.1.1

Er diagram represents the structure and the class structure

4.2. Data dictionary

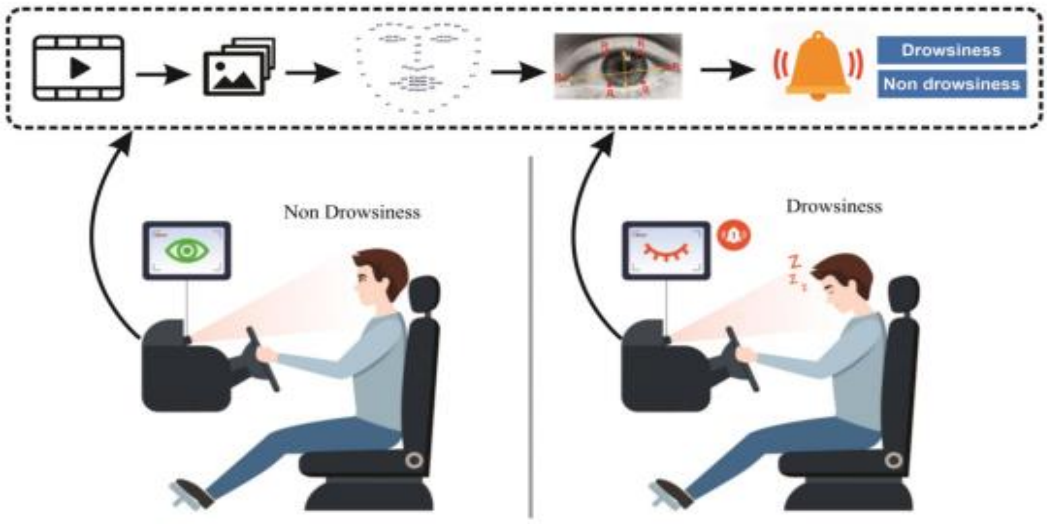


Fig 4.2.1

Data dictionary this represents the process of data convertion

4.3. Data Flow Diagram

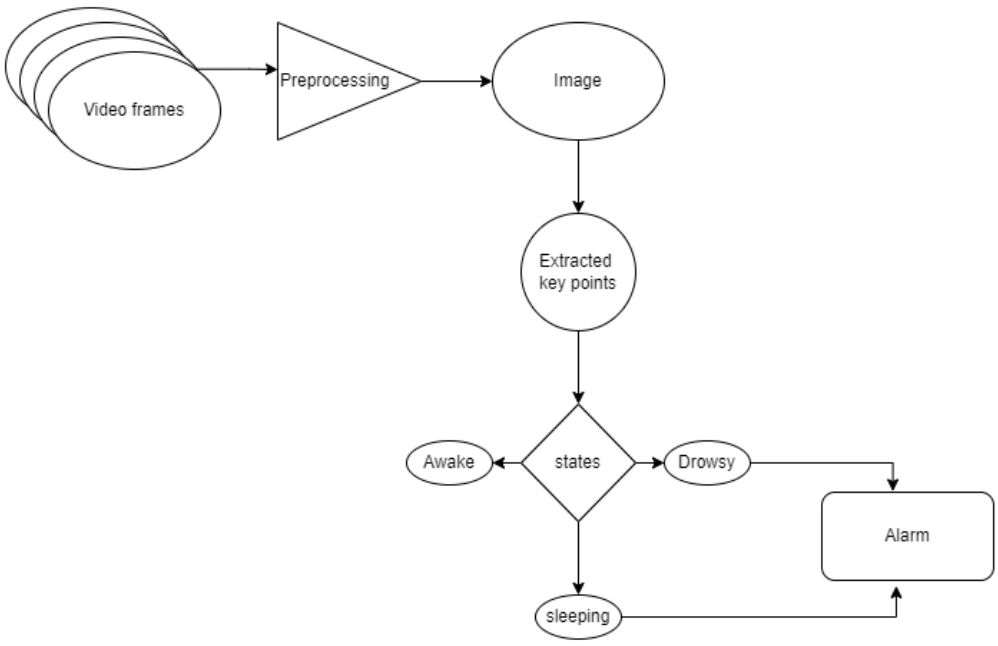


Fig 4.3.1

Data flow and the way in which the data is acquired

4.4. UML Diagrams

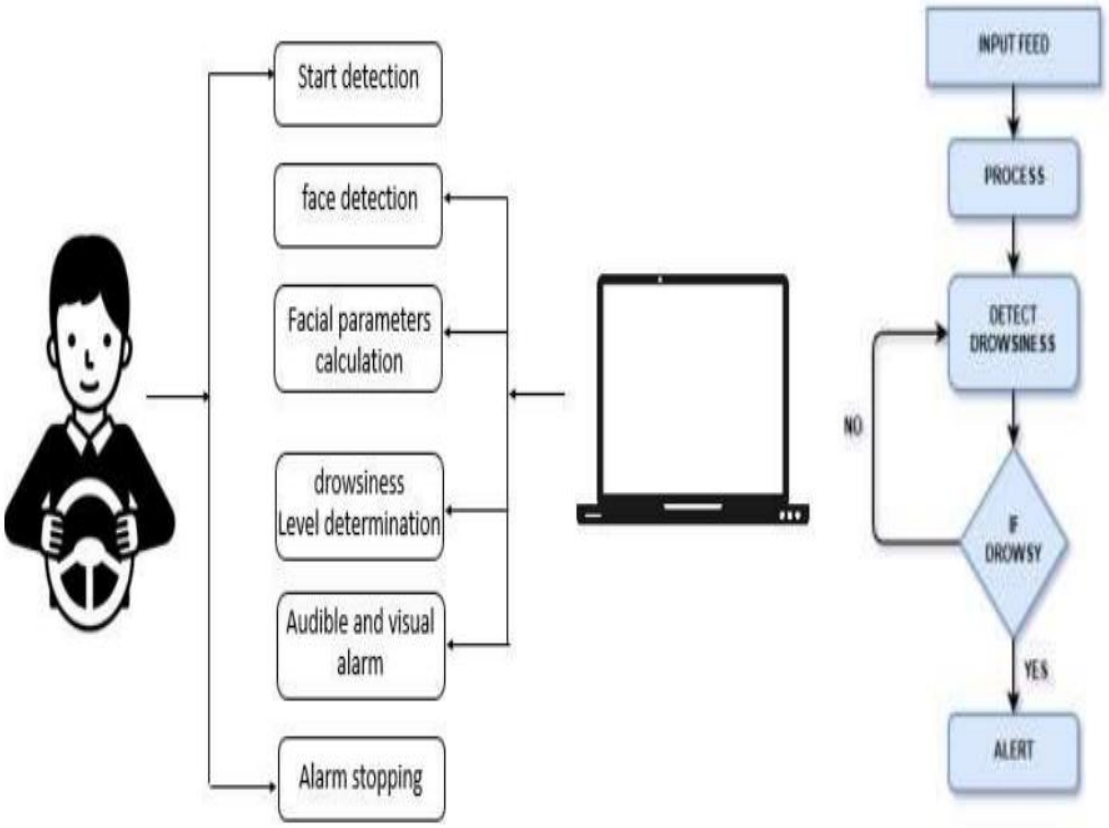


Fig 4.4.1

This represents the visual blue prints of the software and hardware connection

CHAPTER 5
SYSTEM ARCHITECTURE

SYSTEM ARCHITECTURE

5.1. Module

- **Video Capture**

  Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

  Steps to capture a video:

  - Use cv2. Video Capture () to get a video capture object for the camera.
  - Set up an infinite while loop and use the read () method to read the frames using the above created object.
  - Use cv2. Imshow () method to show the frames in the video.
  - Breaks the loop when the user clicks a specific key.

Fig 5.1.1 Video Capture

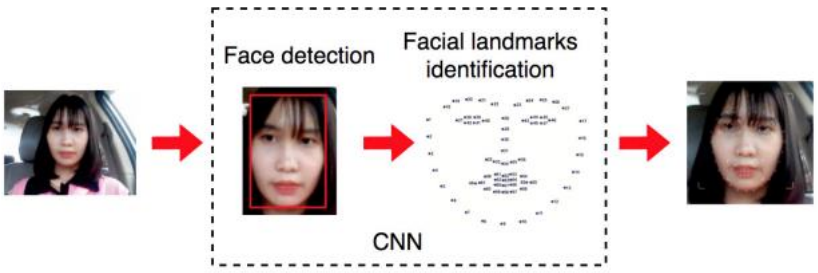


Fig 5.1.2 Detecting faces and facial landmarks.

- **Deduction**

  1. Face Detection
  2. Feature Detection
  3. Drowsy State Identification
  4. Alert System

    - **Face Detection**: The working of our application starts with the recording of the driver. Detecting the face of the subject is our priority as this greatly affects the later process. Face detection is considered to be the first module that is implemented using the HOG algorithm. HOG is the short form for Histogram of Oriented Gradients, it is a feature descriptor that is used to identify the object as it is explained in the above chapter. So, HOG is used as an image descriptor in our application

      In this module, all the images in the dataset are preprocessed. We have the images for 2 classes (alert and drowsy) in two directories. We use the "os" module to list the directory names and list all the image files in each directory. Then, we can access all the image files in each directory. We use OpenCV to read and resize the images. Instead of giving the entire image to classification model, only the face region is extracted and given to the model since the background and other portion of the image is unnecessary. For this, we use Face detection which is a computer vision technology that locates human faces in digital images.

      Feature Detection: As we already mentioned that our application is based on behavioral characteristics, so features status are used to state whether the driver is drowsy or not. Feature detection is the module where we use our second algorithm, SVM. SVM stands for Support Vector Machine which is a classification algorithm used to detect features on the subjects face accurately using facial landmarks. This module is implemented on every frame which is outputted from the previous module.

- **Drowsy state identification**: This is the module where the calculations and comparisons are done on the results of the previous module. As we said the eyes and mouth status are taken into consideration in concluding whether the subject is fatigue. Two operations are carried out in this module which are described below.
- **Calculation**: In detecting a drowsy state we calculate the aspect ratio of the eyes and mouth. Since the aspect ratio is calculated separately for each eye, we take an average of the values of both eyes.
- **Comparison**: The calculated aspect ratio values are compared with the threshold to decide the status of the driver. The threshold values are given to the system or we can say they are taken by default. The calculated aspect ratio of the eye has to be greater than the threshold if not the application considers that the driver is drowsy. It is reverse in the case of mouth, the calculated aspect ratio of the mouth should be less than the threshold if not the application considers that the driver is yawning.
- **Alert system:** This is the module of various options. Alerting the driver is our intention in this module. Alerting the subject can be achieved in many ways from sending a message to taking control of the car. Our system alters the driver by an alarm sound.

- **Key point Extraction**

  This is considered the first step towards data splitting and processing. The reference images in Figure 3 and Figure 4 are used to obtain the coordinates of the land markings. These features are then localised and extracted to minimise the maximum background noise. Basic features extracted are nose, eyes, mouth, eyebrows, and jawline. These facial landmarks extracted can be used in

many applications like an alignment of the face, swapping of faces, e.g., Snapchat, and other applications like blinking detection.

These facial landmarks extracted can be used in many applications like the alignment of the face, swapping of faces like in applications like Snapchat, and other applications like blinking detection.

The coordinates of the eyes are extracted from the numbers 37 to 42 for the right eye and 43 to 46 for the left eye. The positions can be seen in Figure 4 as a reference. Similarly, the mouth positions are noted from 49 to 68, as shown in Figure 5. However, the formula to calculate the Eye Aspect Ratio (EAR) is calculated using the coordinate points from Figure 3. EAR is defined as the ratio of width and height of the human eye and twice the width of an eye. The average values for EAR are 0.33 and 0.141 for opening and closing of eyes.

Fig 5.1.4

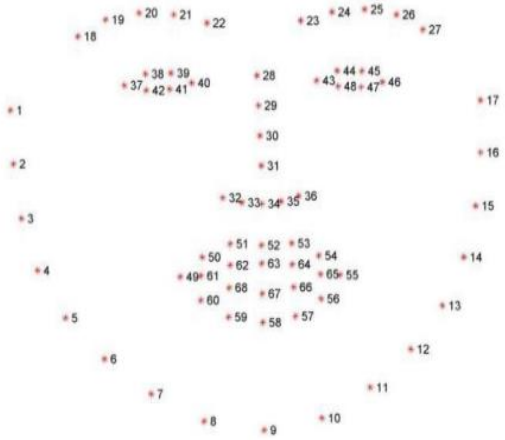


Fig 5.1.5

- **Blinking and Sleeping Classification**

The eye-blinking frequency is an indicator that allows a driver's drowsiness (fatigue) level to be measured. As in the works of Horng et al. and Dong and Wu, if five consecutive frames or during 0.25 s are identified as eye-closed the system is able to issue an alarm cue; PERCLOS also is implemented in this

System. Presents an instantaneous result of this system over a driver's image, whereas pictures the evolution drowsiness index graph for a driver's drowsiness sequence.

To identify drowsiness through eye analysis it is necessary to know its state: open or closed, through the time and develop an analysis over time, i.e., to measure the time that has been spent in each state. Classification of the open and closed state is complex due to the changing shape of the eye, among other factors, the changing position and the rotating of the face, and variations of twinkling and illumination. All this makes it difficult to analyze eye in a reliable manner. For the problems that have been exposed a supervised classification method has been used for this challenging task, in this case, a support vector machine (SVM).

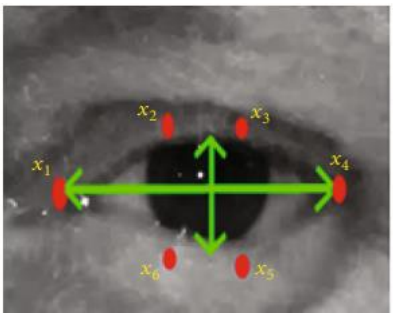


Fig 5.1.6

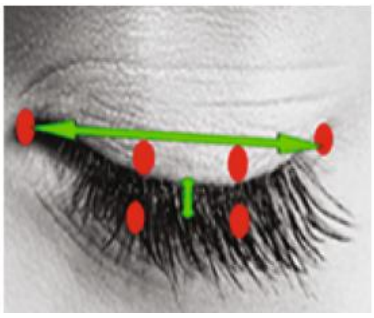


Fig 5.1.7

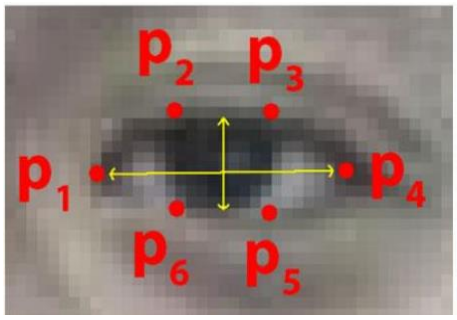


Fig 5.1.8

- **Output Displaying**

  The output will be displayed as sleeping when the eyes are in closed state for certain period of time, it will be displayed drowsy when the eyes are partially closed state and it will displayed active when the eyes are in open.

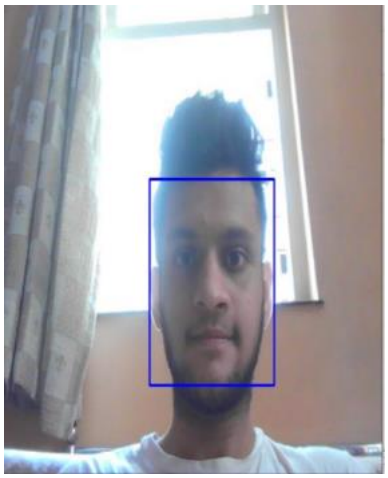


Fig 5.1.9 Active state

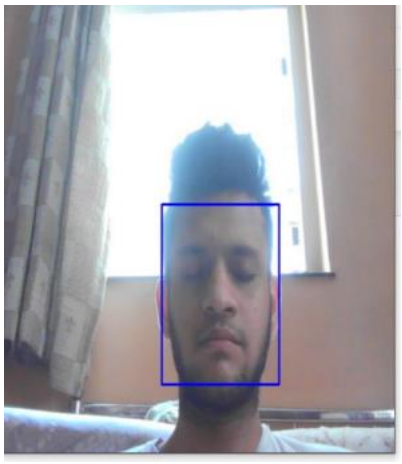


Fig5.1.10 Sleeping

5.2. Algorithms

Face detection :

This is a custom trained model build using tensorflow which is devide into 2 parts namely

1)bak bone

2)detection head

1)back bone :

Is the feature extraction part and the model tend to work on extract the key features of a human face

2) detection head:

this provides a location cordinates of the face

key point extracted :

this moedl gets the face as teh input and outputs the key points this key point is then pre processed and then used as input for predection model

KNN model : this uses the principal of knn to produce the state of the person

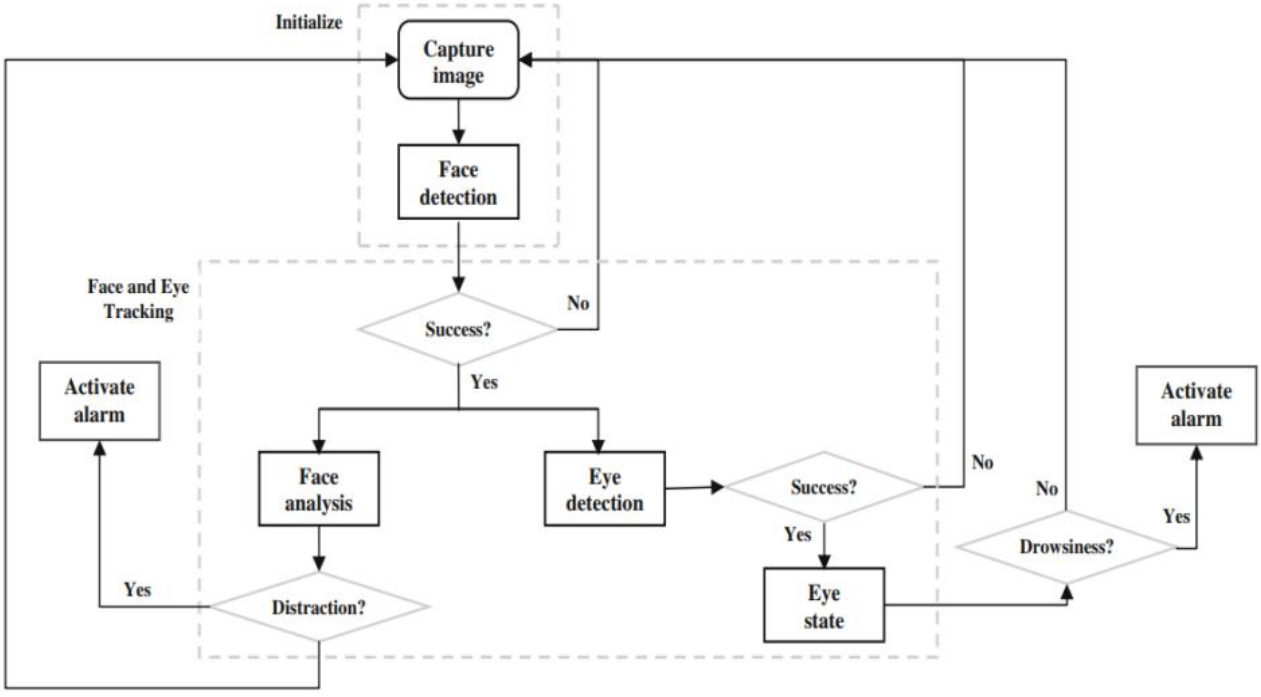


Fig 5.2.1

This explains the flow of the architecture

5.3. Algorithm summery

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 200)       5600
_________________________________________________________________
activation (Activation)      (None, 98, 98, 200)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 200)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 100)       180100
_________________________________________________________________
activation_1 (Activation)    (None, 47, 47, 100)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 100)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 100)       90100
_________________________________________________________________
activation_2 (Activation)    (None, 21, 21, 100)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 100)       0
_________________________________________________________________
flatten (Flatten)            (None, 10000)             0
_________________________________________________________________
dropout (Dropout)            (None, 10000)             0
_________________________________________________________________
dense (Dense)                (None, 64)                640064
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 915,994
Trainable params: 915,994
Non-trainable params: 0
_________________________________________________________________
```

This summery describes about the structure and the algorithm of the classification algorithm which uses the concept of KNN

5.4. Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-

engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The term “Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

There are two main parts to CNN architecture:

• A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction

• A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

5.5 Convolutional Layer

Convolutional layers are the major building blocks used in convolutional neural networks. This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM). If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image.

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

5.6 The Convolution Operation

We analyze the influence of nearby pixels by using something called a filter. We move this across the image from top left to bottom right. When building the network, we randomly specify values for the filters, which then continuously update themselves as the network is trained. After the filters have passed over the image, a feature map is generated for each filter. These are then taken through an activation function, which decides whether a certain feature is present at a given location in the image.

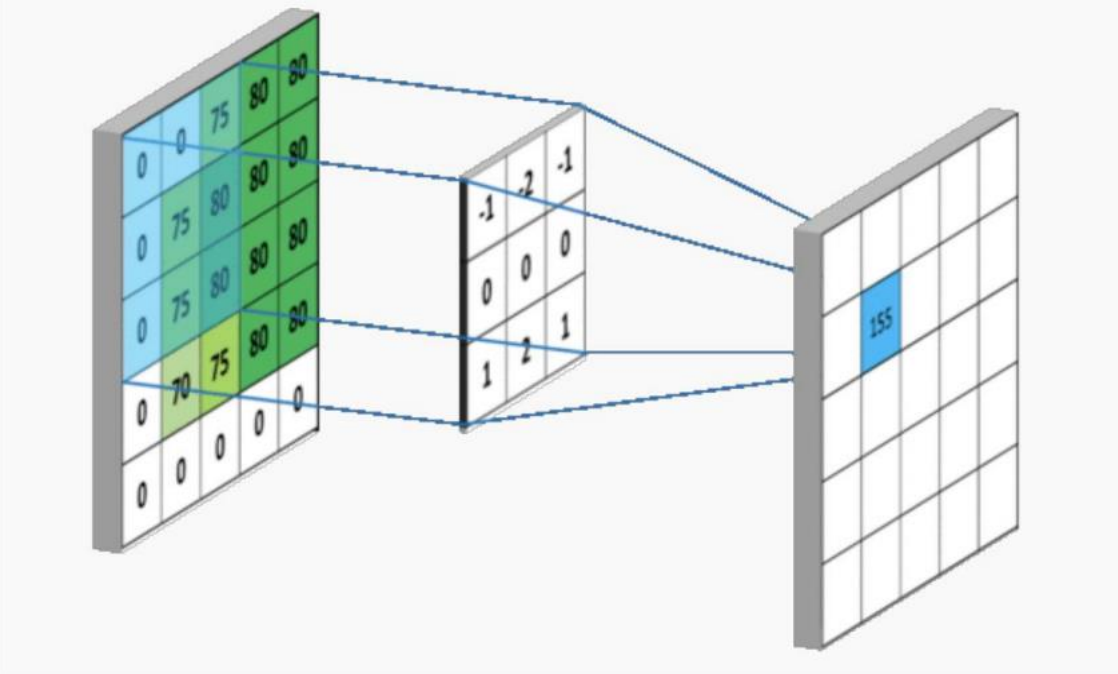


Fig 5.6.1

We have colour images which have 3 channels i.e., Red, Green and Blue. Each filter is convolved with the entirety of the 3D input cube but generates a 2D feature map. Because we have multiple filters, we end up with a 3D output: one 2D feature map per filter. Convolving the image with a filter produces a feature map that highlights the presence of a given feature in the image.

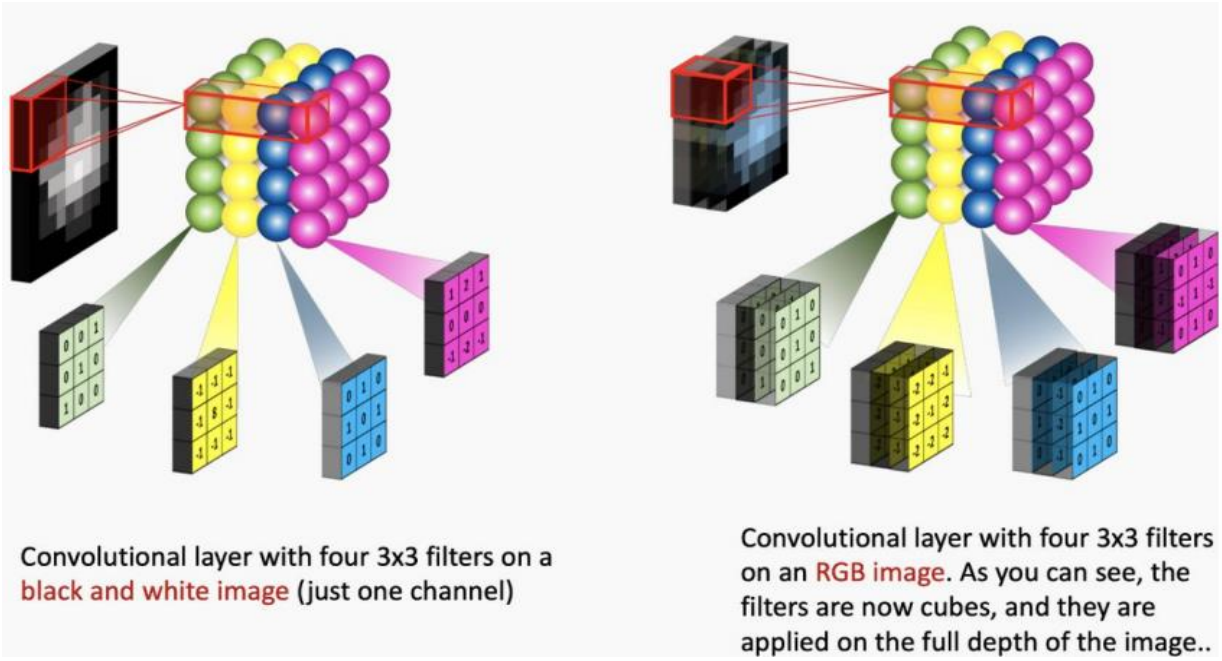


Fig 5.6.2

5.7 Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

5.8 Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

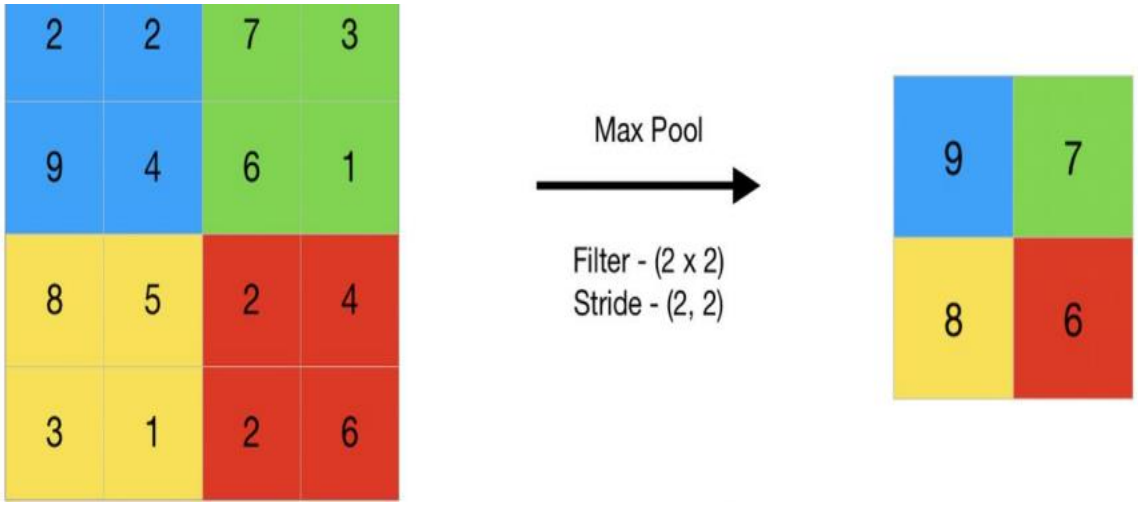


Fig5.8.1

5.9 Fully Connected layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place

5.10 Activation Functions

One of the most important parameters of the Neural Networks is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions has a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred for a multi-class classification, generally softmax us used.

5.11 Sigmoid

It is one of the most widely used non-linear activation function. Sigmoid transforms the values between the range 0 and 1.

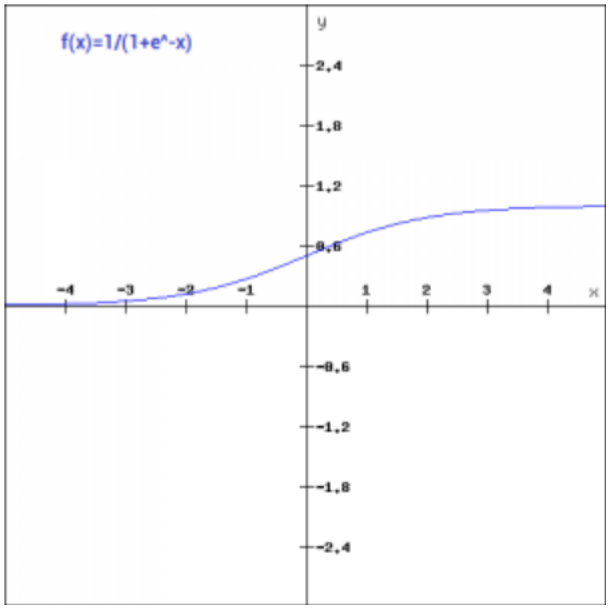


Fig 5.11.1

5.12 ReLU

The ReLU function is another non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The neurons will only be deactivated if the output of the linear transformation is less than 0.

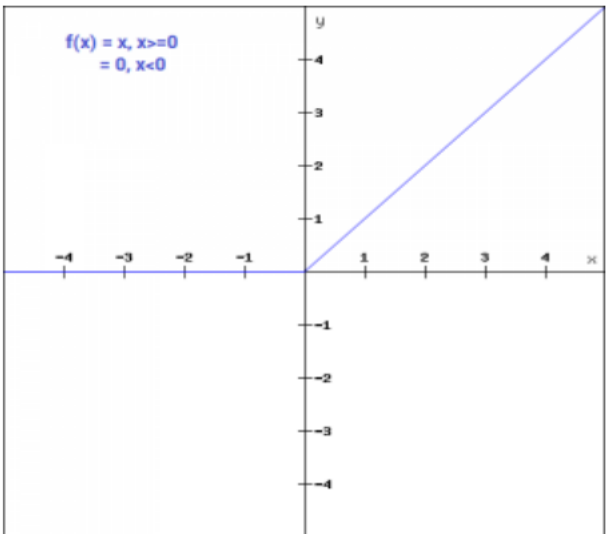


Fig5.12.1

5.13 Softmax

Softmax function is often described as a combination of multiple sigmoids. We know that sigmoid returns values between 0 and 1, which can be treated as probabilities of a data point belonging to a particular class. The softmax function can be used for multiclass classification problems. This function returns the probability for a datapoint belonging to each individual class

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

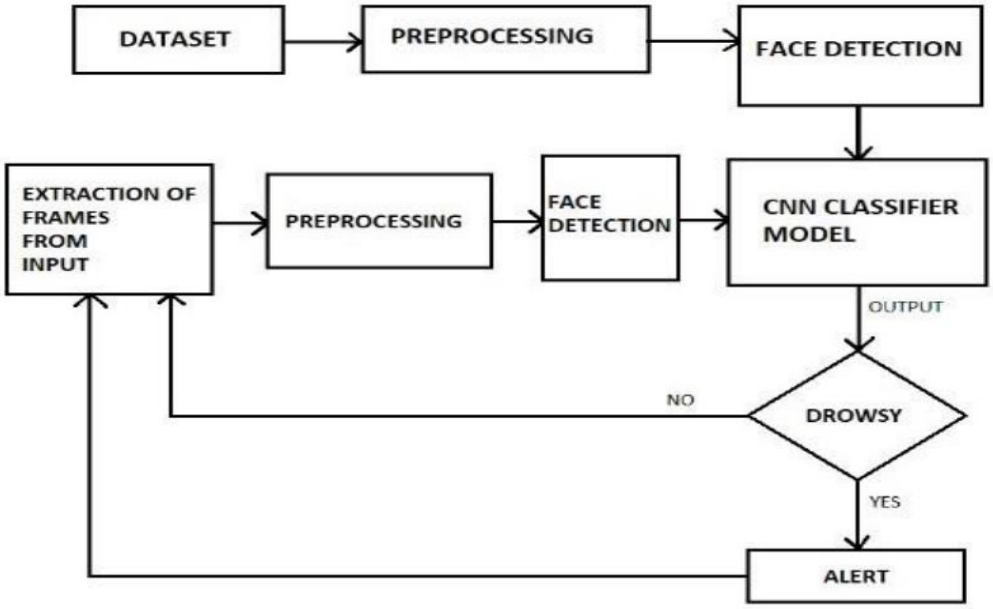


Fig 5.13.1

5.14. Training

Deep learning neural networks learn a mapping function from inputs to outputs. This is achieved by updating the weights of the network in response to the errors the model makes on the training dataset. Updates are made to continually reduce this error until either a good enough model is found or the learning process gets stuck and stops.

The stochastic gradient descent algorithm is used to solve the optimization problem where model parameters are updated each iteration using the backpropagation algorithm.

A neural network model uses the examples to learn how to map specific sets of input variables to the output variable. It must do this in such a way that this mapping works well for the training dataset, but also works well on new examples not seen by the model during training. This ability to work well on specific examples and new examples is called the ability of the model to generalize.

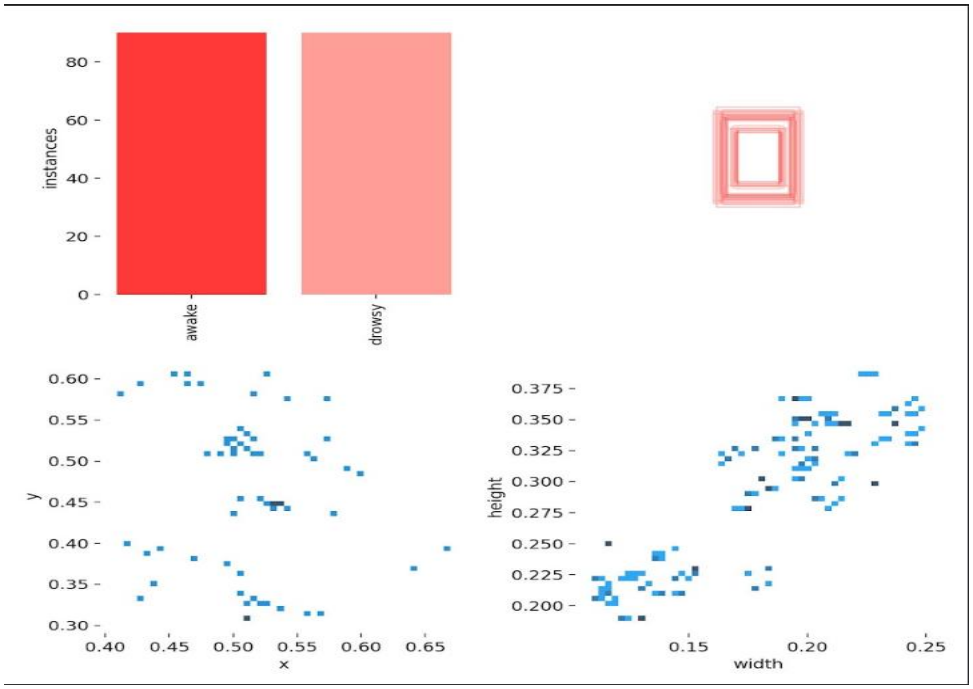


Fig 5.14.1

5.15 Testing

The usage of the word “testing” in relation to Machine Learning models is primarily used for testing the model performance in terms of accuracy/precision of the model. It can be noted that the word, "testing" means different for conventional software development and Machine Learning models development.

The goal of DL testing:

Quality assurance is required to make sure that the software system works according to the requirements. Were all the features implemented as agreed? Does the program behave as expected? All the parameters that you test the program against should be stated in the technical specification document.

Moreover, testing has the power to point out all the defects and flaws during development. We don't want your clients to encounter bugs after it is released and come to you waving their fists. Different kinds of testing allow us to catch bugs that are visible only during runtime.

However, in machine learning, a programmer usually inputs the data and the desired behavior, and the logic is elaborated by the machine. This is especially true for deep learning. Therefore, the purpose of machine learning testing is, first of all, to ensure that this learned logic will remain consistent, no matter how many times we call the program.

CHAPTER 6

Sample Code

Sample Code

Main.py

```
# Importing OpenCV Library for basic image processing functions
import cv2
# Numpy for array related functions
import numpy as np
# Dlib for deep learning based Modules and face landmark detection
import dlib
# face_utils for basic operations of conversion
from imutils import face_utils

# Initializing the camera and taking the instance
cap = cv2.VideoCapture(0)

# Initializing the face detector and landmark detector
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

# status marking for current state
sleep = 0
drowsy = 0
active = 0
status = ""
color = (0, 0, 0)


def compute(ptA, ptB):
    dist = np.linalg.norm(ptA - ptB)
    return dist


def blinked(a, b, c, d, e, f):
    up = compute(b, d) + compute(c, e)
    down = compute(a, f)
    ratio = up / (2.0 * down)

    # Checking if it is blinked
    if (ratio > 0.25):
        return 2
```

```
    elif (ratio > 0.21 and ratio <= 0.25):
        return 1
    else:
        return 0


while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = detector(gray)
    # detected face in faces array
    for face in faces:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()

        face_frame = frame.copy()
        face_frame=cv2.rectangle(face_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        landmarks = predictor(gray, face)
        landmarks = face_utils.shape_to_np(landmarks)

        # The numbers are actually the landmarks which will show eye
        left_blink = blinked(landmarks[36], landmarks[37],
                             landmarks[38], landmarks[41], landmarks[40], landmarks[39])
        right_blink = blinked(landmarks[42], landmarks[43],
                              landmarks[44], landmarks[47], landmarks[46],
landmarks[45])

        # Now judge what to do for the eye blinks
        if (left_blink == 0 or right_blink == 0):
            sleep += 1
            drowsy = 0
            active = 0
            if (sleep > 6):
                status = "SLEEPING !!!"
                color = (255, 0, 0)

        elif (left_blink == 1 or right_blink == 1):
            sleep = 0
            active = 0
```

```
            drowsy += 1
            if (drowsy > 6):
                status = "Drowsy !"
                color = (0, 0, 255)

        else:
            drowsy = 0
            sleep = 0
            active += 1
            if (active > 6):
                status = "Active :)"
                color = (0, 255, 0)

        cv2.putText(face_frame, status, (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1.2, color, 3)

        for n in range(0, 68):
            (x, y) = landmarks[n]
            face_frame=cv2.circle(face_frame, (x, y), 1, (255, 255, 255), -1)

            cv2.imshow("Frame", face_frame)
    #cv2.imshow("Result of detector", face_frame)
    key = cv2.waitKey(1)
    if key == 27:
        break
```

Drowsiness_train.ipynb

1. Install and Import Dependencies

```
import torch
from matplotlib import pyplot as plt
import numpy as np
import cv2
```

2. Load Model

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
AutoShape(
(model): DetectMultiBackend( (
```

model): DetectionModel(
model): Sequential(
(0): Conv(
(conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2),padding=(2, 2))
(act): SiLU(inplace=True)
)
(1): Conv(
(conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2),padding=(1,1))
(act): SiLU(inplace=True)
)
(2): C3(
(cv1): Conv(
(conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1)) (act):
SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1)) (act):
SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1)) (act):
SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(32, 32, kernel_size=(1, 1),stride=(1,1))(act):
SiLU(inplace=True))
(cv2): Conv(
(conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),padding=(1,1))
(act): SiLU(inplace=True)
)
)
)
)
(3): Conv(
(conv): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(act): SiLU(inplace=True)
)
(4): C3(
(cv1): Conv(
(conv): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)

)
(cv2): Conv(
(conv): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
(1): Bottleneck(
(cv1): Conv(
(conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),padding=(1,1))
(act): SiLU(inplace=True)
)
)
)
)
(5): Conv(
(conv): Conv2d(128, 256, kernel_size=(3, 3),stride=(2,2),padding=(1,1))
(act): SiLU(inplace=True)
)
(6): C3(
(cv1): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))

(act): SiLU(inplace=True)
)
(cv3): Conv((conv): Conv2d(256, 256, kernel_size=(1, 1),stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(128, 128, kernel_size=(3, 3),stride=(1,1),padding=(1,1))
(act): SiLU(inplace=True)
)
)
(6): C3(
(cv1): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
(2): Bottleneck(
(cv1): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))

(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
)
)
(7): Conv(
(conv): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(act): SiLU(inplace=True)
)
(8): C3(
(cv1): Conv(
(conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
)
)
(9): SPPF(
(cv1): Conv(
(conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)

)
(cv2): Conv(
(conv): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): MaxPool2d(kernel_size=5, stride=1, padding=2, dilation=1,
ceil_mode=False)
)
(10): Conv(
(conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(11): Upsample(scale_factor=2.0, mode=nearest)
(12): Concat()
(13): C3(
(cv1): Conv(
(conv): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
)
)
(14): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)

)
(15): Upsample(scale_factor=2.0, mode=nearest)
(16): Concat()
(17): C3(
(cv1): Conv(
(conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
)
)
(18): Conv(
(conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(act): SiLU(inplace=True)
)
(19): Concat()
(20): C3(
(cv1): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)

(cv3): Conv(
(conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(act): SiLU(inplace=True)
)
)
)
)
(21): Conv(
(conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(act): SiLU(inplace=True)
)
(22): Concat()
(23): C3(
(cv1): Conv(
(conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv3): Conv(
(conv): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(m): Sequential(
(0): Bottleneck(
(cv1): Conv(
(conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
(act): SiLU(inplace=True)
)
(cv2): Conv(
(conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))

```
(act): SiLU(inplace=True)
)
)
)
)
(24): Detect(
(m): ModuleList(
(0): Conv2d(128, 255, kernel_size=(1, 1), stride=(1, 1))
(1): Conv2d(256, 255, kernel_size=(1, 1), stride=(1, 1))
(2): Conv2d(512, 255, kernel_size=(1, 1), stride=(1, 1))
)
)
)
)
)
)
)
```

3. Images Detections

```
img = 'I-80_Eastshore_Fwy.jpg'
results = model(img)
results.print()
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
results.render()
[array([[[185, 202, 220],
         [185, 202, 220],
         [185, 202, 220],
         ...,
         [191, 207, 230],
         [191, 207, 230],
         [191, 207, 230]],
        [[185, 202, 220],
         [185, 202, 220],
         [185, 202, 220],
         ...,
         [191, 207, 230],
         [191, 207, 230],
         [191, 207, 230]],
        [[185, 202, 220],
         [185, 202, 220],
         [185, 202, 220],
```

```
       ...,
       [191, 207, 230],
       [191, 207, 230],
       [191, 207, 230]],
       ...,
       [[ 88, 88, 90],
       [ 89, 89, 91],
       [ 91, 91, 93],
       ...,
       [129, 131, 130],
       [123, 124, 126],
       [169, 170, 172]],
       [[ 88, 88, 90],
       [ 88, 88, 90],
       [ 89, 89, 91],
       ...,
       [125, 127, 126],
       [115, 116, 118],
       [139, 140, 142]],
       [[ 88, 88, 90],
       [ 88, 88, 90],
       [ 88, 88, 90],
       ...,
       [128, 130, 129],
       [116, 117, 119],
       [120, 121, 123]]], dtype=uint8)]
```

4. Real Time Detections

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
ret, frame = cap.read()

# Make detections
results = model(frame)

cv2.imshow('YOLO', np.squeeze(results.render()))

if cv2.waitKey(10) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

5. Train from scratch

```
 import uuid   # Unique identifier
import os
import time
IMAGES_PATH = os.path.join('data', 'image') #/data/images
labels = ['awake', 'drowsy']
number_imgs = 10
cap = cv2.VideoCapture(0)
# Loop through labels
for label in labels:
  print('Collecting images for {}'.format(label))
  time.sleep(10)

  # Loop through image range
  for img_num in range(number_imgs):
  print('Collecting images for{},image number{}'.format(label,      img_num))

    # Webcam feed
    ret, frame = cap.read()

    # Naming out image path
    imgname = os.path.join(IMAGES_PATH, label+'.'+str(uuid.uuid1())+'.jpg'
)

    # Writes out image to file
    cv2.imwrite(imgname, frame)

    # Render to the screen
    cv2.imshow('Image Collection', frame)

    # 2 second delay between captures
    time.sleep(2)

    if cv2.waitKey(10) & 0xFF == ord('q'):
      break
  cap.release()
  cv2.destroyAllWindows()
print(os.path.join(IMAGES_PATH,labels[0]+'.'+str(uuid.uuid1())+'.jpg'))
for label in labels:
  print('Collecting images for {}'.format(label))
  for img_num in range(30):
```

```
        print('Collecting images for {}, image number {}'.format(label, img_num))
        imgname = os.path.join(IMAGES_PATH, label+'.'+str(uuid.uuid1())+'.jpg'
)
        print(imgname)
!git clone https://github.com/tzutalin/labelImg
!pip install pyqt5 lxml --upgrade
!cd labelImg && pyrcc5 -o libs/resources.py resources.qrc
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="HgH3WWbZ6TUbl2Y4LTu4")
project = rf.workspace("panimalar").project("drowsiness-empbr")
dataset = project.version(2).download("yolov5")
!cd yolov5 && python train.py --img 320 --batch 16 --epochs 1500 --
data dataset.yaml --weights yolov5s.pt --workers 2
```

6. Load Custom Mode

```
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/runs/train/ex
p9/weights/last.pt', force_reload=True)
img = os.path.join('data', 'images', 'awake.005de685-9a33-11ed-bead-
089798794a79.jpg')
results = model(img)
results.print()
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    # Make detections
    results = model(frame)

    cv2.imshow('YOLO', np.squeeze(results.render()))
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
        cap.release()
  cv2.destroyAllWindows()
```

Train.ipynb

```
   # -*- coding: utf-8 -*-
"""train.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1GDBOIRIaPhvQLMyxsDJDB2DdLJOk5kSx
"""
#importing the playsound module to play audio from playsound import playsound
#Creating an object for capturing video
vid = cv2.VideoCapture(0)
img_size=100
i=0
data1=[]
color=(225,225,225)
while(i<=10):
# Capture the video frame
# by frame
ret, frame = vid.read()
cv2.imshow('frame', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
#Coverting the image into gray scale
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
for (x, y, w, h) in faces:
cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
cv2.imshow("Frame", frame)
roi = frame[y:y + h, x:x + w]
resized=cv2.resize(roi,(img_size,img_size))
#resizing the gray scale into 100x100, since we need a fixed common size for all the images in the dataset
data1.append(resized)
 i+=1
if(i==5):
data1=np.array(data1)/255.0
```

```
#data1=np.reshape(data1,(data1.shape[0],img_size,img_size,1))
input_shape = (100, 100, 3)
#data=np.reshape(data,(img_size,img_size,1))
prediction = (model.predict(data1))
print("Not drowsy")
cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
cv2.imshow("Frame", frame)
data1=[]
i=0
# After the loop
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

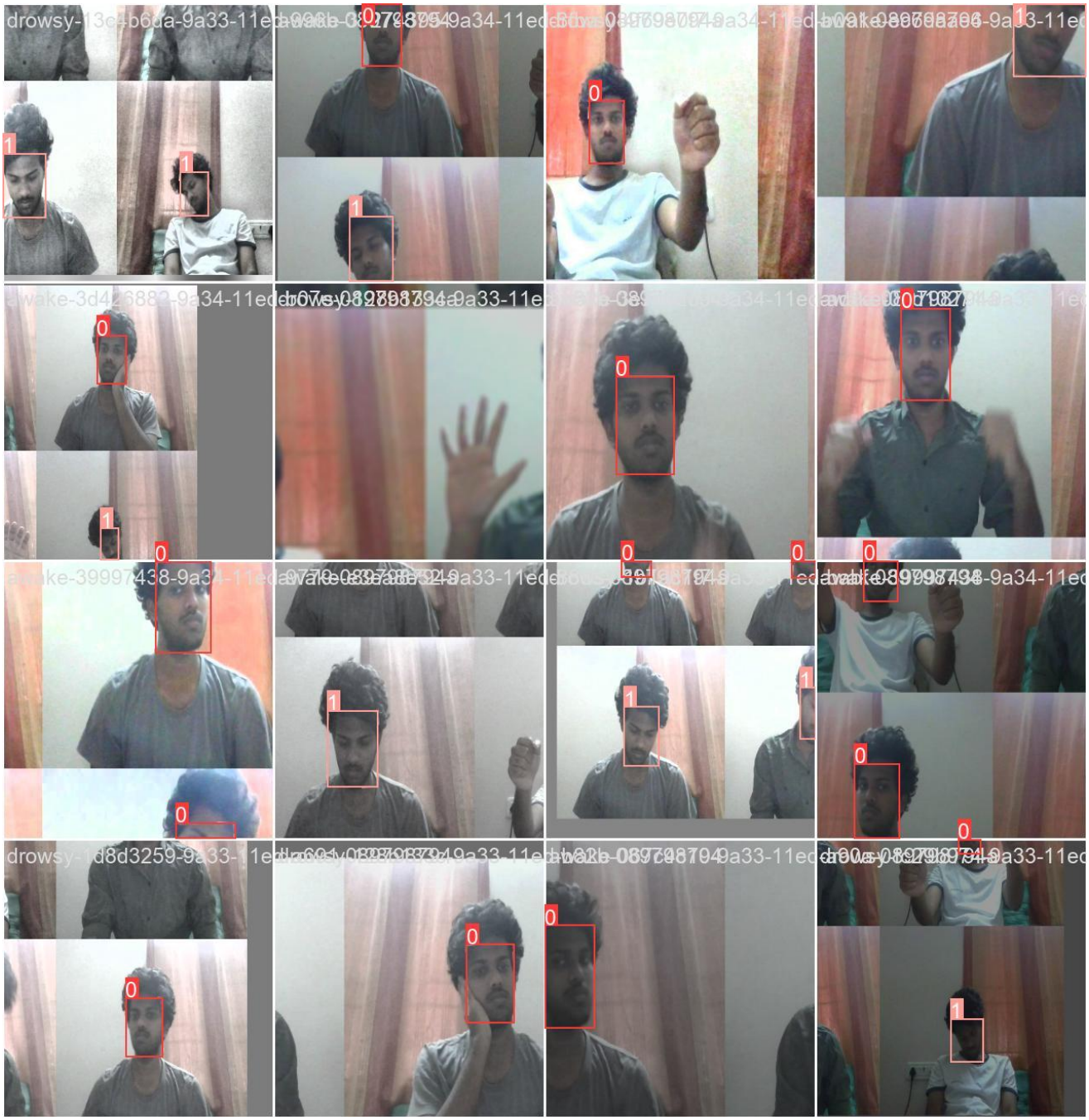
Code for reading images after Face Detection

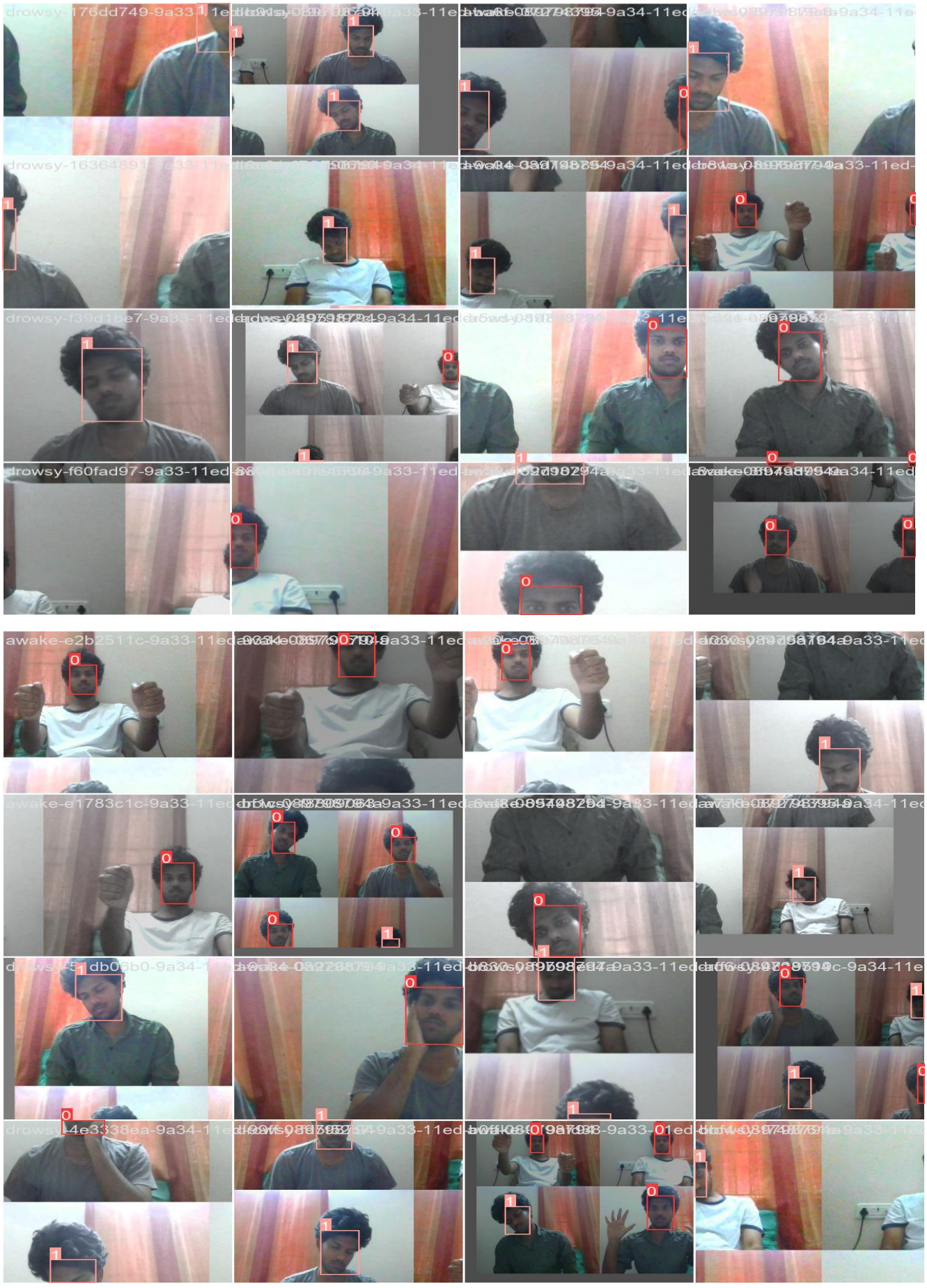
```
data=[]
target=[]
a=0
d=0
for category in categories:
 #Path to access each directory
folder_path=os.path.join(data_path,category)
img_names=os.listdir(folder_path)
for img_name in img_names:
#Path to access each image in the particular directory
img_path=os.path.join(folder_path,img_name)
#Read the image
img=cv2.imread(img_path)
if(category=="alert"): a+=1
 elif(category=="drowsy"):
d+=1
try:
#Appending the image to data list and its label to target list data.append(img)
target.append(label_dict[category])
except Exception as e:
print('Exception:',e)
```

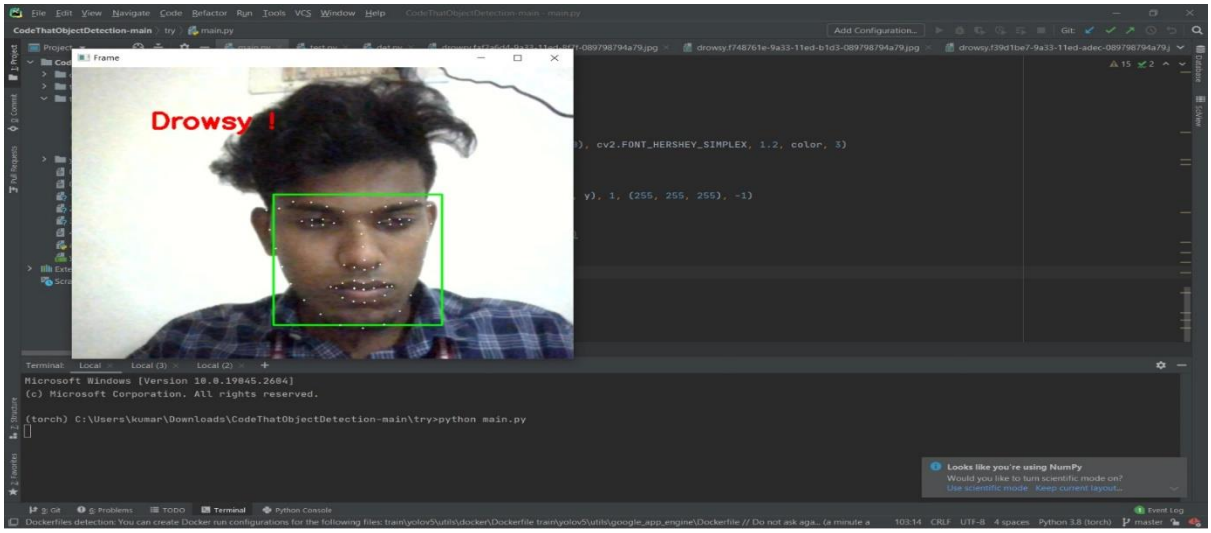
CHAPTER 7
SCREEN SHORTS

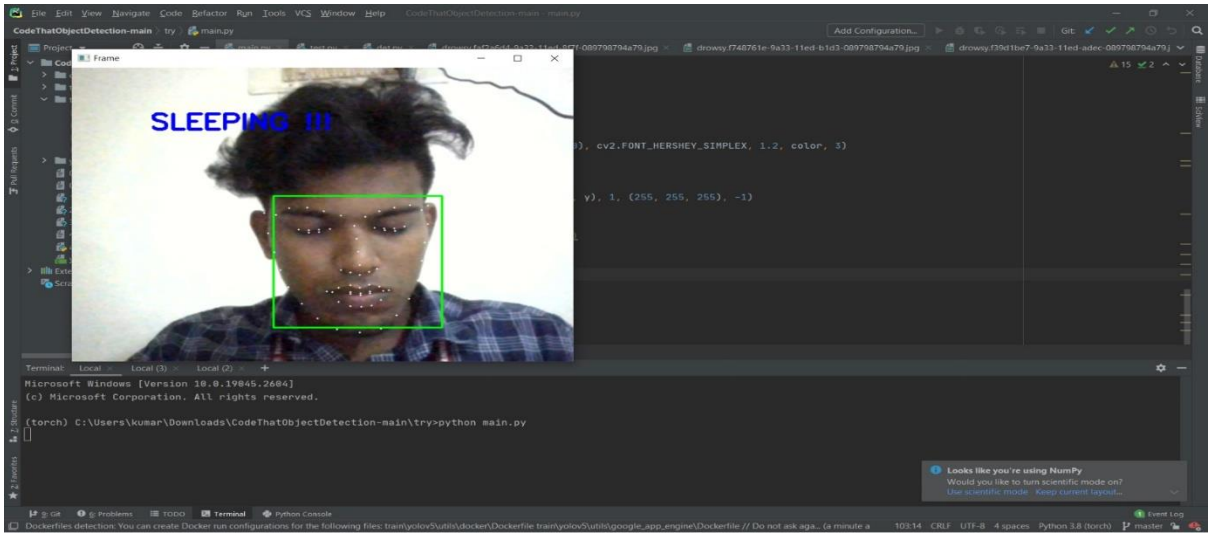
SCREEN SHORTS

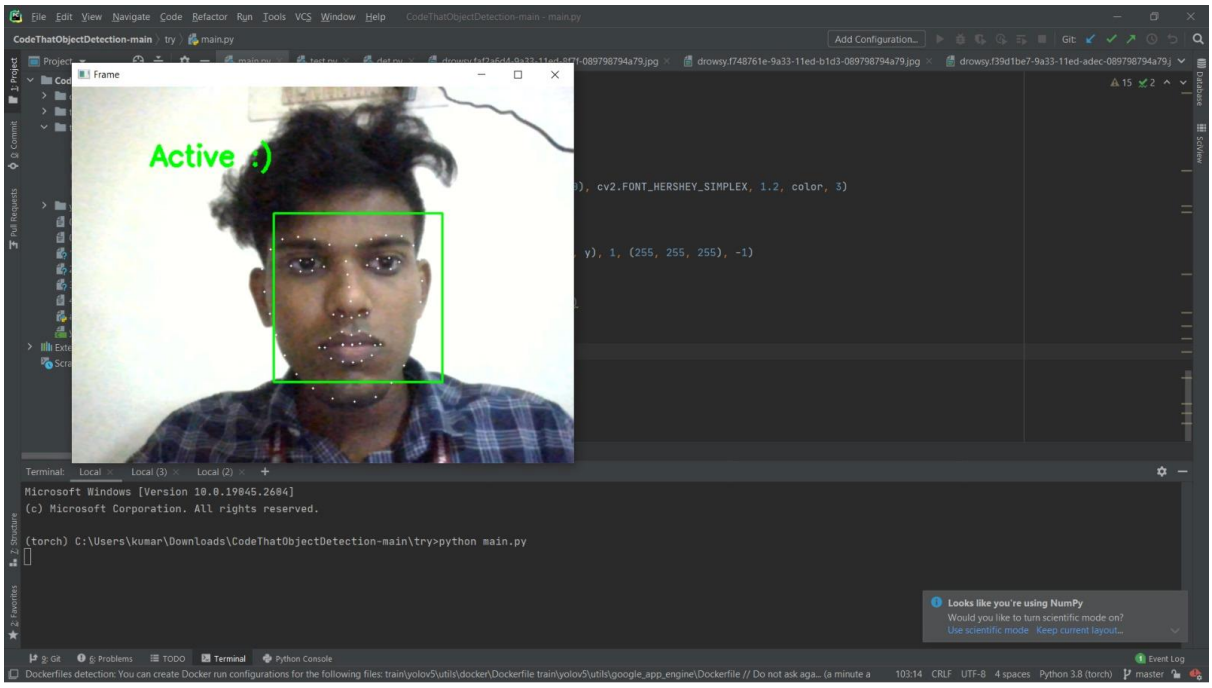
7.1 Training data and preprocessed data

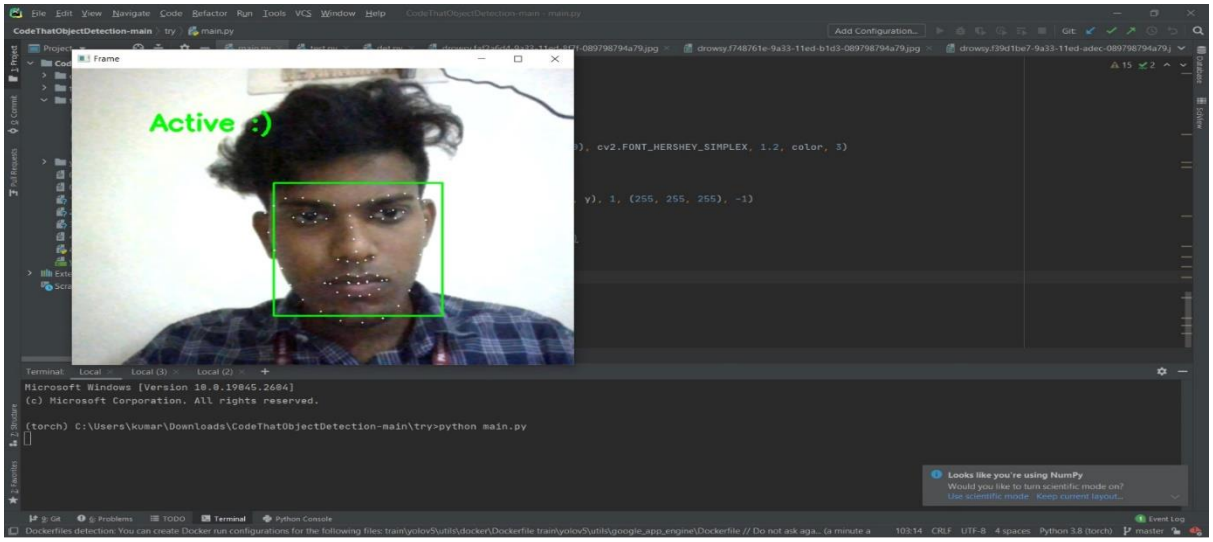


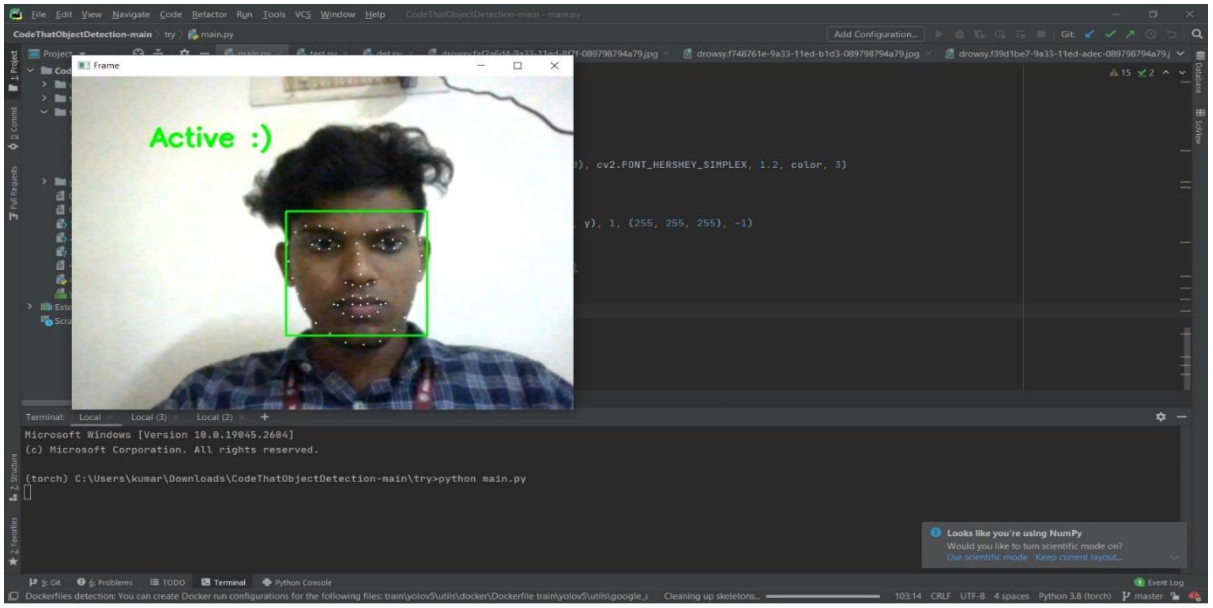


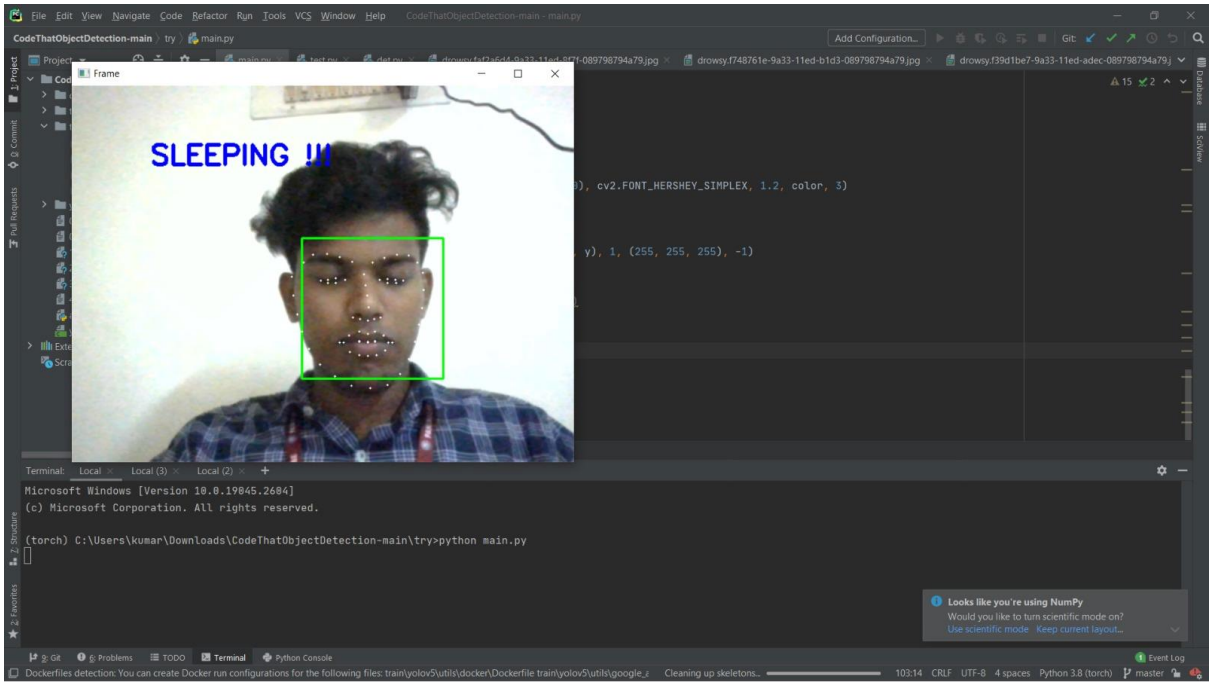

Frame
Drowsy !
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py
Looks like you're using NumPy
Would you like to turn scientific mode on?

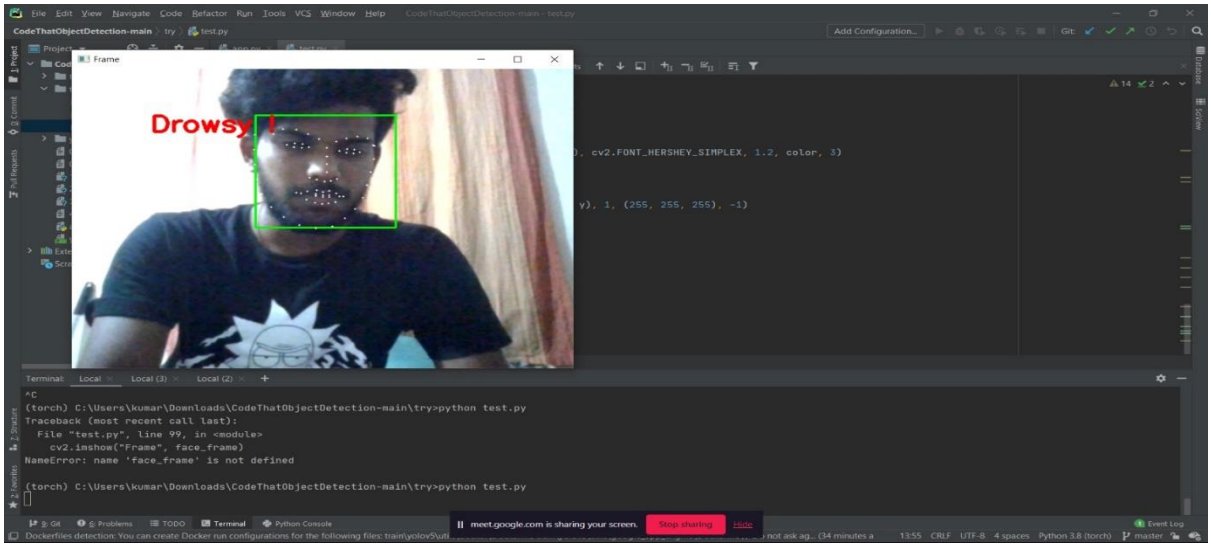

Frame
SLEEPING !!!
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py
Looks like you're using NumPy
Would you like to turn scientific mode on?

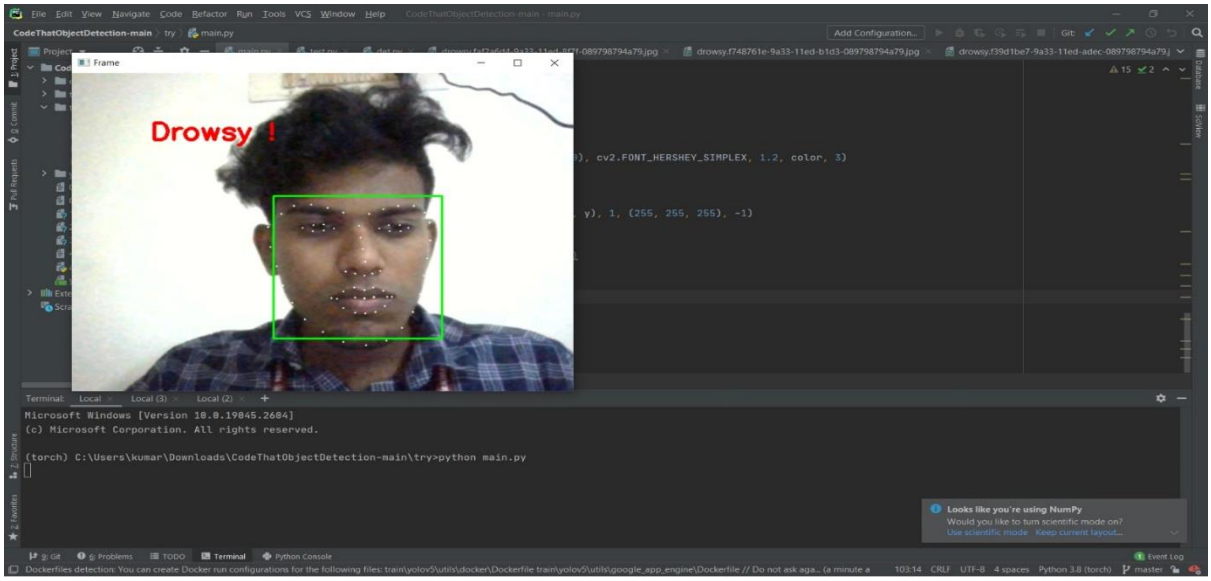

Frame
Active :)
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py
Looks like you're using NumPy
Would you like to turn scientific mode on?

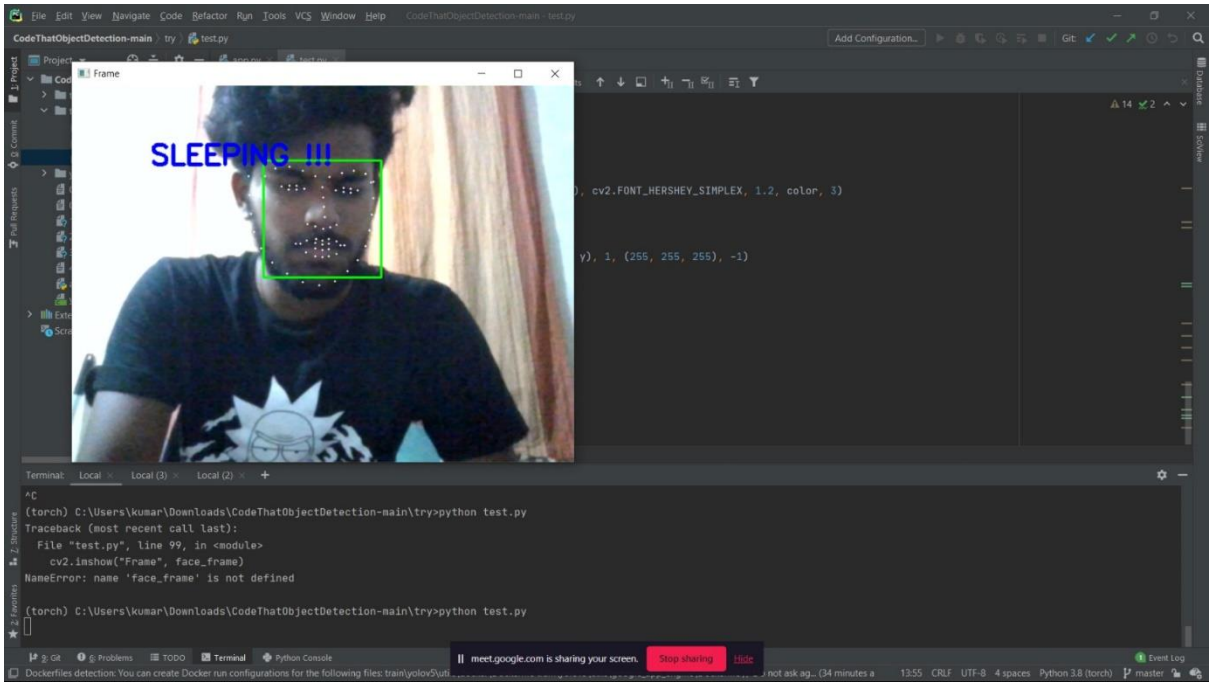

Frame
Active :)
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py

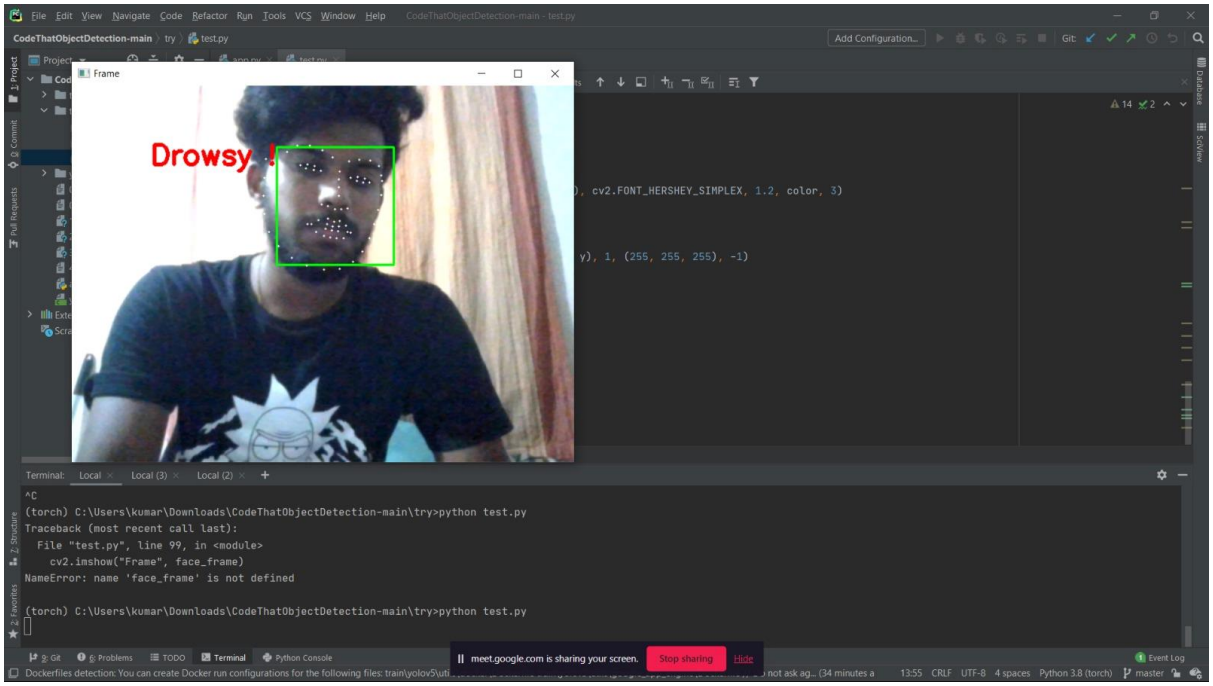

Frame
Active :)
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py

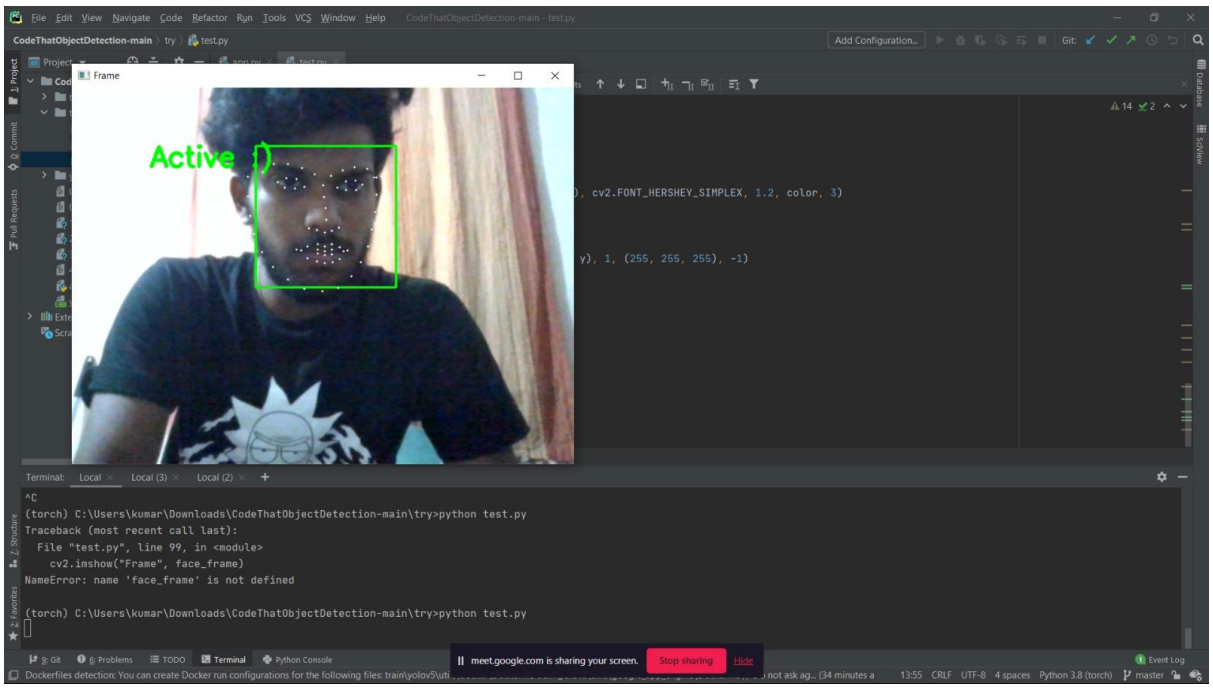

Frame
SLEEPING !!!
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py

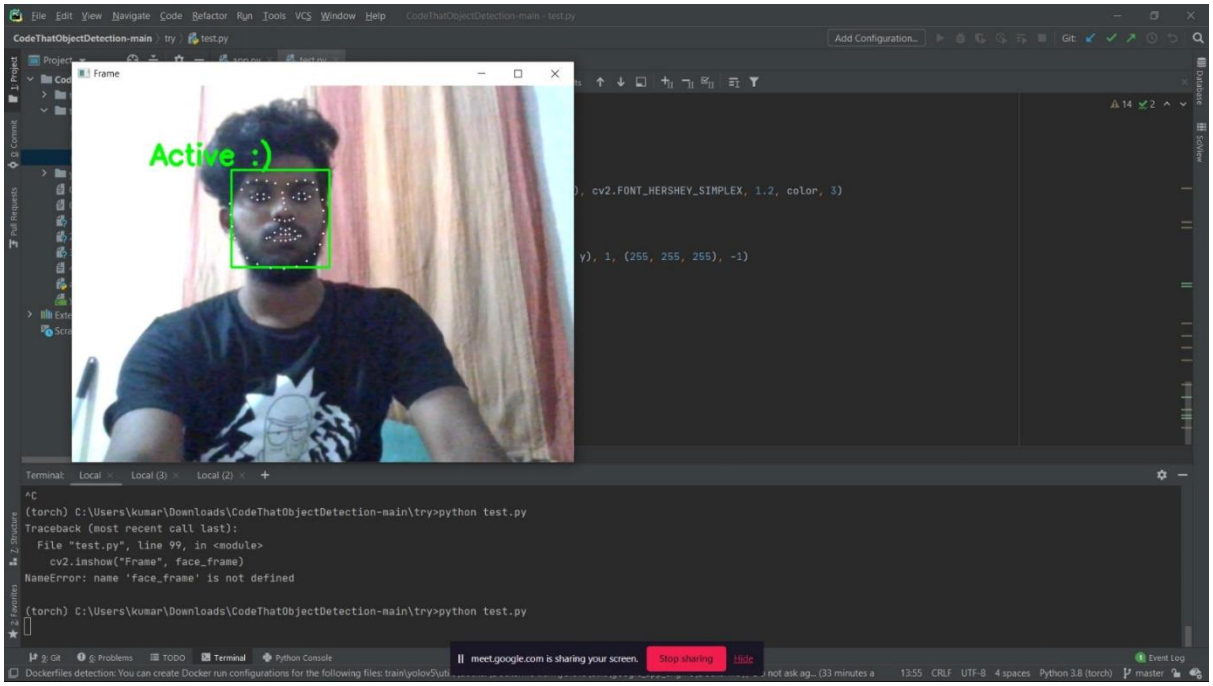

Frame
Drowsy !
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
Traceback (most recent call last):
File "test.py", line 99, in <module>
cv2.imshow("Frame", face_frame)
NameError: name 'face_frame' is not defined
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py

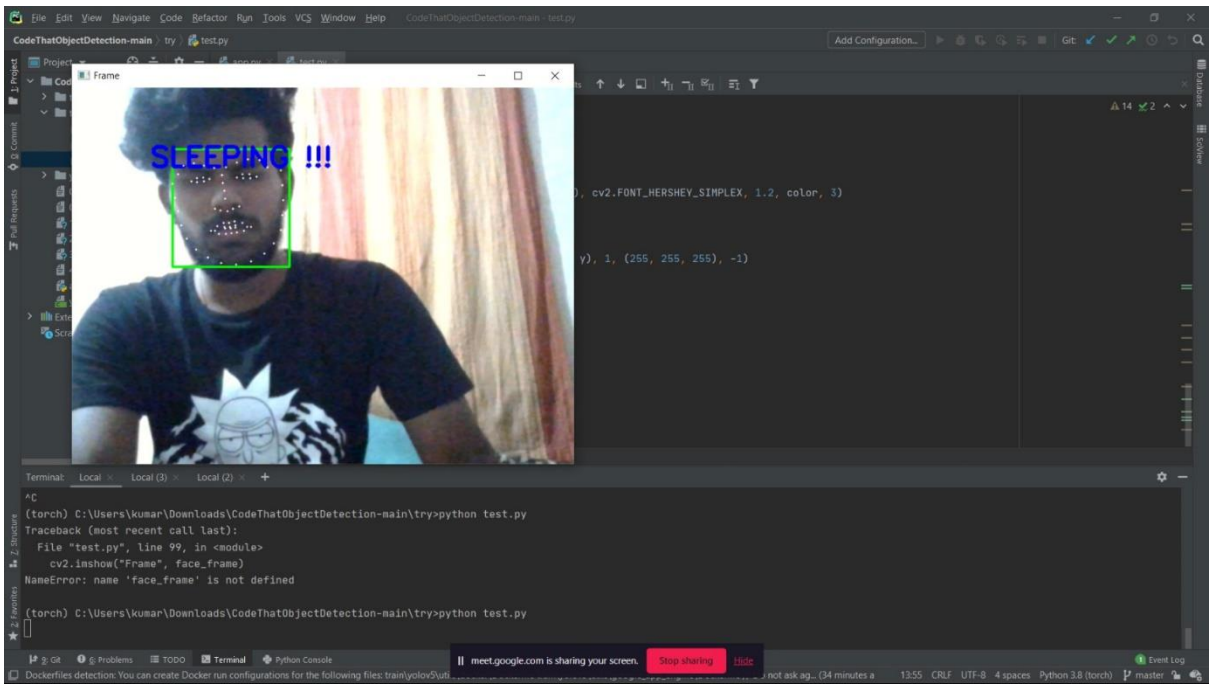

Frame
Drowsy !
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python main.py


Frame
SLEEPING !!!
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
Traceback (most recent call last):
File "test.py", line 99, in <module>
cv2.imshow("Frame", face_frame)
NameError: name 'face_frame' is not defined
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py


Frame
Drowsy !
), cv2.FONT_HERSHEY_SIMPLEX, 1.2, color, 3)
y), 1, (255, 255, 255), -1)
Terminal: Local Local (3) Local (2)
^C
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
Traceback (most recent call last):
File "test.py", line 99, in <module>
cv2.imshow("Frame", face_frame)
NameError: name 'face_frame' is not defined
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
meet.google.com is sharing your screen.
Stop sharing
Hide


Frame
Active :)
), cv2.FONT_HERSHEY_SIMPLEX, 1.2, color, 3)
y), 1, (255, 255, 255), -1)
Terminal: Local Local (3) Local (2)
^C
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
Traceback (most recent call last):
File "test.py", line 99, in <module>
cv2.imshow("Frame", face_frame)
NameError: name 'face_frame' is not defined
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
meet.google.com is sharing your screen.
Stop sharing
Hide


Frame
Active :)
), cv2.FONT_HERSHEY_SIMPLEX, 1.2, color, 3)
y), 1, (255, 255, 255), -1)
Terminal: Local Local (3) Local (2)
^C
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
Traceback (most recent call last):
File "test.py", line 99, in <module>
cv2.imshow("Frame", face_frame)
NameError: name 'face_frame' is not defined
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
meet.google.com is sharing your screen.
Stop sharing
Hide


Frame
SLEEPING !!!
), cv2.FONT_HERSHEY_SIMPLEX, 1.2, color, 3)
y), 1, (255, 255, 255), -1)
Terminal: Local Local (3) Local (2)
^C
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
Traceback (most recent call last):
File "test.py", line 99, in <module>
cv2.imshow("Frame", face_frame)
NameError: name 'face_frame' is not defined
(torch) C:\Users\kumar\Downloads\CodeThatObjectDetection-main\try>python test.py
meet.google.com is sharing your screen.
Stop sharing
Hide

CHAPTER 8

CONCLUSIONS

8.1 CONCLUSIONS

In this paper, a research project to develop a non-intrusive driver's drowsiness system based on Computer Vision and Artificial Intelligence has been presented. This system uses advanced technologies for analyzing and monitoring drivers eye state in real-time and in real driving conditions.

The proposed algorithm for face tracking, eye detection and eye tracking and accurate under varying light, external illuminations interference, vibrations, changing background and facial orientations

8.2 Future Enhancement

For future work, the objective will be to reduce the percentage error, i.e., to reduce the false alarms; for this, extra experiments will be developed, using additional drivers and incorporating new modules.

CHAPTER 9
REFERENCES

REFERENCES

1. Armingol, J.M., de la Escalera, A., Hilario, C., Collado, J., Carrasco, J., Flores, M., Pastor, J., Rodriguez, F.: IVVI: intelligent vehicle based on visual information. Robot. Auton. Syst. 55, 904–916 (2007). doi:10.1016/j.robot.2007.09.004
2. Bergasa, L., Nuevo, J., Sotelo, M., Vazquez, M.: Real time system for monitoring driver vigilance. In: IEEE Intelligent Vehicles Symposium, Parma, 14–17 June 2004
3. Brookshear, J.G.: Theory of Computation: Formal Languages, Automata and Complexity. Addison Wesley Iberoamericana, Reading (1993)
4. Brandt, T., Stemmer, R., Mertsching, B., Rakotomirainy, A.: Affordable visual driver monitoring system for fatigue and monotony. IEEE Int. Conf. Syst. Man Cybern. 7, 6451–6456 (2004)
5. Branzan, A., Widsten, B., Wang, T., Lan, J., Mah, J.: A computer vision-based system for realtime detection of sleep onset in fatigued drivers. In: IEEE Intelligent Vehicles Symposium, pp. 25–30 (2008)
6. Chang, C., Lin, C.: LIBSVM: a library for support vector machine (2001). www.csie.ntu. edu.tw/∼cjlin/libsvm
7. Chen, Y.W., Kubo, K.: A robust eye detection and tracking technique using Gabor filters. In: Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE, vol. 1, pp. 109–112 (2007)
8. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other KernelBased Learning Methods. Cambridge University Press, Cambridge (2000)
9. Daugman, J.G.: Uncertainty relation for resolution in space, spatial frequency and orientation optimized by two-dimensional cortial filters. J. Opt. Soc. Am. 2(7), 1160–1169 (1985)

10. D'Orazio, T., Leo, M., Distante, A.: Eye detection in faces images for a driver vigilante system. IEEE Intelligent Vehicles Symposium University of Parma, Italy, 14–17 June (2004)
11. Dong, W., Wu, X.: Driver fatigue detection based on the distance of eyelid. In: IEEE Int. Workshop VLSI Design & Video Tech., Suzhou, China (2005)
12. Fletcher, L., Petersson, L., Zelinsky, A.: Driver assistance systems based on vision in and out of vehicles. In: IEEE Proceedings of Intelligent Vehicles Symposium, pp. 322–327 (2003)
13. Gejgus, P., Sparka, M.: Face Tracking in Color Video Sequences. The Association for Computing Machinery Inc., New York (2003)
14. Grace, R.: Drowsy driver monitor and warning system. International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design (2001)
15. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature Extraction: Foundations and Applications. Springer, Berlin (2006)
16. Hagenmeyer, L.: Development of a multimodal, universal human–machine-interface for hypovigilance-management-systems. Ph.D. thesis, University of Stuttgart (2007)
17. Horng, W., Chen, C., Chang, Y.: Driver fatigue detection based on eye tracking and dynamic template matching. In: Proceedings of the IEEE International Conference on Networking, Sensing & Control (2004)
18. Isard, M., Blake, A.: Condensation: conditional density propagation for visual tracking. Int. J. Comput. Vis. 29(1), 5–28 (1998). doi:10.1023/A:1008078328650
19. Isard, M.A.: Visual motion analysis by probabilistic propagation of conditional density. Ph.D. thesis, Oxford University (1998)
20. Jafar, I., Ying, H.: A new method for image contrast enhancement based on automatic specification of local histograms. IJCSNS Int. J. Computer Sci. Netw. Secur. 7(7), 1–10 (2007)

21. Ji, Q., Zhu, Z., Lan, P.: Real time nonintrusive monitoring and prediction of driver fatigue. IEEE Trans. Veh. Technol. 53(4), 1052–1068 (2004). doi:10.1109/TVT.2004.830974
22. Ji, Q., Yang, X.: Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. Real-Time Imaging 8, 357–377 (2002)
23. Liu, C.: Gabor-based kernel PCA with fractional power polynomial models for face recognition. IEEE Trans. Pattern Anal. Mach. Intell. 26(5), 572–581 (2004)
24. Longhurst G.: Understanding Driver Visual Behaviour. Seeing Machine Pty Limited, Acton (2002)
25. Looney, C.G.: Pattern Recognition Using Neural Networks, Theory and Algorithms for Engineers and Scientists. Oxford University Press, Oxford (1997)
26. McLachlan, G.J.: The EM Algorithm and Extensions. Wiley, New York (1997)
27. Mujtaba I.M.: Application of Neural Networks and Other Learning Technologies in Process Engineering. Imperial College Press, London (2001)
28. NHTSA: evaluation of techniques for ocular measurement as an index of fatigue and the basis for alertness management. Final report DOT HS 808762, National Highway Traffic Safety Administration, Virginia 22161, USA (1998)
29. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. 9, 62–66 (1979). doi:10.1109/TSMC.1979.4310076
30. Parker, J.R.: Practical Computer Vision Using C. Wiley, New York (1994)
31. Tian, Z., Qin, H.: Real-time driver's eye state detection. In: IEEE International Conference on Vehicular Electronics and Safety, pp. 285–289 (2005)
32. Satake, J., Shakunaga, T.: Multiple target tracking by appearance-based condensation tracker using structure information. In: Proceedings of the 17th International Conference on Patter Recognition (ICPR'04), vol. 3, pp. 294–297 (2004)
33. Swingler, K.: Applying Neural Networks: A Practical Guide. Academic, New York (1996)

34. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Conference on Computer Vision and Pattern Recognition (2001)