# Minor Project Report

# On

# "Driver Drowsiness Detection System"

*In partial fulfillment of the requirement for the award of the degree of*



## SUBMITTED BY

Satyendra Singh Rathore (0901IT191054)

## SUBMITTED TO

Prof.Vikas Sejwar

Asst.Prof. Yogeshwar Singh

## Department of Information Technology

## Madhav Institute of Technology and Science, Gwalior

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)


**Session: 2021-22(July-Dec)**

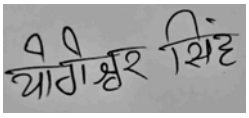
**Madhav Institute of Technology and Science, Gwalior (M.P.)**

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

# CERTIFICATE

This is to certify that Satyendra Singh Rathore (0901IT191054) minor project, " Driver Drowsiness Detection System " is a genuine record of a project completed under our supervision and guidance in partial fulfilment of the requirements for the award of a Bachelor of Technology in Information Technology in the Department of Information Technology, Madhav Institute of Technology and Science, Gwalior.

**(Prof.Vikas Sejwar)**
Mentor



**(Prof. YOGESHWAR SINGH)**
Mentor

# CANDIDATE'S DECLARATION

I hereby declare that the Project entitled "**Driver Drowsiness Detection System"** which is being submitted in the partial fulfilment of the requirement for the award of **Bachelor of Technology** in **Information Technology**.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results that are not original to this work.

To the best of my knowledge the material presented in this Project has not been submitted elsewhere for the award of any other degree/diploma.

**Date: 02/12/2021**

**Place: Gwalior** **Satyendra Singh Rathore (0901IT191054)**

# ACKNOWLEDGEMENT

We would like to articulate our deep gratitude to our project guide Prof. Vikas Sejwar, Asst. Prof. Saumil Maheshwari for his guidance, advice and constant support in the project work. We would like to thank him specially for being our advisor here at Madhav Institute Of Technology and Science, Rourkela. We would like to thank all faculty members and staff of the Department of Electronics and communication Engineering, MITS, Gwalior for their generous help in various ways for this project.

Last but not the least; we give our sincere thanks to all of our friends who have patiently extended all sorts of help in this project.

Satyendra Singh Rathore

(0901IT191054)

# ABSTRACT

Driver fatigue is one of the major causes of accidents in the world. Detecting the drowsiness of the driver is one of the surest ways of measuring driver fatigue. In this project we aim to develop a prototype drowsiness detection system. This system works by monitoring the eyes of the driver and sounding an alarm when he/she is drowsy.

The system so designed is a non-intrusive real-time monitoring system. The priority is on improving the safety of the driver without being obtrusive. In this project the eye blink of the driver is detected. If the drivers eyes remain closed for more than a certain period of time, the driver is said to be drowsy and an alarm is sounded. The programming for this is done in OpenCV using the Haarcascade library for the detection of facial features.

# CONTENTS

# Chapter 1:

## INTRODUCTION

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes.

The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time.

First we input the facial image using a webcam. Preprocessing was first performed by binarizing the image. The top and sides of the face were detected to narrow down the area where the eyes exist. Using the sides of the face, the center of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the top of the face, horizontal averages of the face area were calculated. Large changes in the averages were used to define the eye area.

*****

# Chapter 2:

## Literature Survey

### SYSTEM REVIEW

This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field too.

### TECHNOLOGY USED

a. PYTHON - Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

b. IMAGE PROCESSING - In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images.

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

.

# Chapter 3:

## Software Requirement Specification

For software perquisites, one must have python (3.6 or higher) installed on their system with for the following program to run effectively:

- **Keras** : pip install opencv-python (face and eye detection).
- **Tensorflow :** pip install tensorflow (keras uses TensorFlow as backend).
- **Pygame :** pip install keras (to build our classification model).
- **OpenCV :** pip install pygame (to play alarm sound).
- **Numpy :** pip install Numpy.

### Operating System

- Windows

## Hardware Requirements Specification

I. Laptop with basic hardware.
II. Webcam

*****

# Chapter 4:

# Why OpenCV?

## What Is OpenCV?

OpenCV [OpenCV] is an open source (see *http://opensource.org*) computer vision library available from *http://SourceForge.net/projects/opencvlibrary*.

OpenCV was designed for computational efficiency and having a high focus on real-time image detection. OpenCV is coded with optimized C and can take work with multicore processors. If we desire more automatic optimization using Intel architectures.

One of OpenCVs goals is to provide a simple-to-use computer vision infrastructure which helps people to build highly sophisticated vision applications fast. The OpenCV library, containing over 500 functions, spans many areas in vision. Because computer vision and machine learning oft en goes hand-in-hand,

OpenCV also has a complete, general-purpose, Machine Learning Library (MLL).

This sub library is focused on statistical pattern recognition and clustering. The MLL is very useful for the vision functions that are the basis of OpenCV's usefulness, but is general enough to be used for any machine learning problem. [4]

## What Is Computer Vision?

Computer vision is the transforming of data from a still, or video camera into either a representation or a new decision. All such transformations are performed to achieve a particular goal. A computer obtains a grid of numbers from a camera or from the disk, and that's that. Usually, there is no built in pattern recognition or automatic control of focus and aperture, no cross-associations with years of experience. For the most part, vision systems are still fairly naïve.

# Why OpenCV?

## Specific:

OpenCV was designed for image processing. Every function and data structure has been designed with an Image Processing application in mind.

## Speedy:

After doing some real time image processing with both Matlab and OpenCV, we usually got very low speeds, a maximum of about 4-5 frames being processed per second with Matlab. With OpenCV however, we get actual real time processing at around 30 frames being processed per second.

Sure we pay the price for speed – a more cryptic language to deal with, but it's definitely worth it. We can do a lot more, like perform some really complex mathematics on images using C and still get away with good enough speeds for your application.

## Efficient:

Matlab uses just way too much system resources. With OpenCV, we can get away with as little as 10mb RAM for a real-time application. Although with today's computers, the RAM factor isn't a big thing to be worried about. However, our drowsiness detection system is to be used inside a car in a way that is non-intrusive and small; so a low processing requirement is vital.

Thus we can see how OpenCV is a better choice than Matlab for a real-time drowsiness detection system.

*****

# Chapter 5:

# Machine Learning

## What Is Machine Learning

The goal of **machine learning** is to turn data into information.

Machine learning converts data into information by detecting rules or patterns from that data.

## Training and Test Set

Machine learning works on data like temperature values or stock prices or color intensities, and in our case face and eye detection. The data is usually preprocessed into features. To reach our goals, machine learning algorithms can analyze our obtained features and hence adjust the weights, the thresholds, and all the other parameters for maximizing performance set by those goals. This method of parameter adjustment for meeting a goal is what is called learning.

It is very important to understand how efficiently machine learning methods can work. This might be a delicate task. Usually, we break up the input data set into a very large training set and a relatively small test set. Then we can run our classifier on the training set in order to learn for a prediction model. Once done, we can then test our prediction classifier obtained on the remainder of the images left in the set.

The test set is not applied for training; also we don't allow the classifier to see the labels of the test set.

When the classifier performs well, we have a possibly lucrative model that can be used on data in the actual world.

After the classifier has been setup, it sees faces that it could not see before and it makes decisions with regards to what it had learned during training. Finally, when we develop a classification system, we usually use a validation data set.

At times, testing the entire system at the finish is a big step to take. We usually want to change parameters in the way before preparing our classifier for the final testing.While we're running across the training data, we can perform pretests on validation data in order to see our progress.

If we are happy with our output for the validation set, we can run the classifier up on the test set for the final result.

After having labeled the data that was obtained from various sources, we should decide which features we need to extract from these objects. Also, we must know what objects we are after. If the faces always appear upright, there is no reason for using rotation-invariant features and also no reason for trying to rotate the objects before processing.

# Using Machine Learning in Vision

Usually, most algorithms take a data vector having many features as input. Here, the number of features may number in the thousands. If our task is recognizing a certain kind of object—take for example, a person's face. The first problem that we encounter is obtaining and labeling the training data which falls into positive (i.e. there is a face in the window) and negative (i.e. no face) cases. We soon realize that faces can appear at various scales: i.e. their image might consist of only a few pixels, or we might be looking at an ear which is filling the whole screen. Worse still, faces are usually occluded. We have to define what we actually mean when we say that a face is in the window.

When we have requisite background window information, we can first remove it in order to help other objects stand out. Then we perform our image processing. This may consist of normalizing the image and then computing the various features. The resulting data vectors are all given the label that is associated with the object, action, or window.

*****

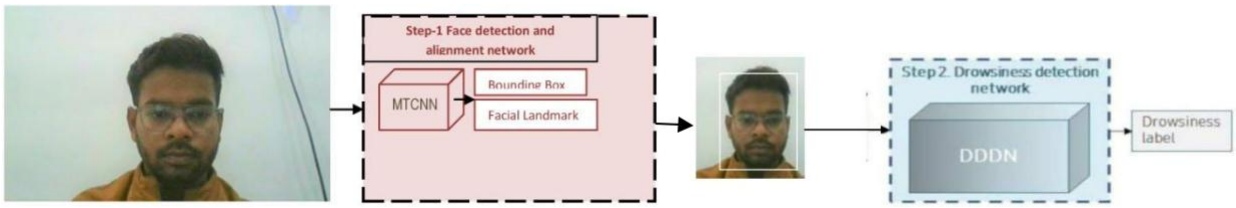
# Chapter 6:

## Drowsiness Detection using Deep Learning

In Recent times, as Machine learning and deep learning are growing earlier throughout the technology community, deep learning is wide wont to resolve tough issues that can't be handled properly exploitation standard strategies. Deep learning supported Convolutional Neural Networks (CNNs) makes a innovation particularly for pc Vision tasks like image classification, object detection, feeling recognition, etc.

## Compression Algorithms of Deep Learning Model

While deep learning is influential on numerous classification tasks, it's AN encumbrance to deploy these algorithms for sensible applications on embedded systems since model size of deep learning is usually large and nice machine complexness is needed. Therefore, within the fashionable years, algorithms to decrease model size and increase speed are planned by mistreatment varied ways in which. Normally, trained networks embrace terminated info, thus a number of weights are often thrown out by applying pruning while not accuracy drop.




Input

***Overall framework of drowsiness detection: Step 1 consists of face and landmark detector and step 2 consists of drowsiness detection network from the detected* face**

*****

# Chapter 7:

## System Design

USE CASE DIAGRAM

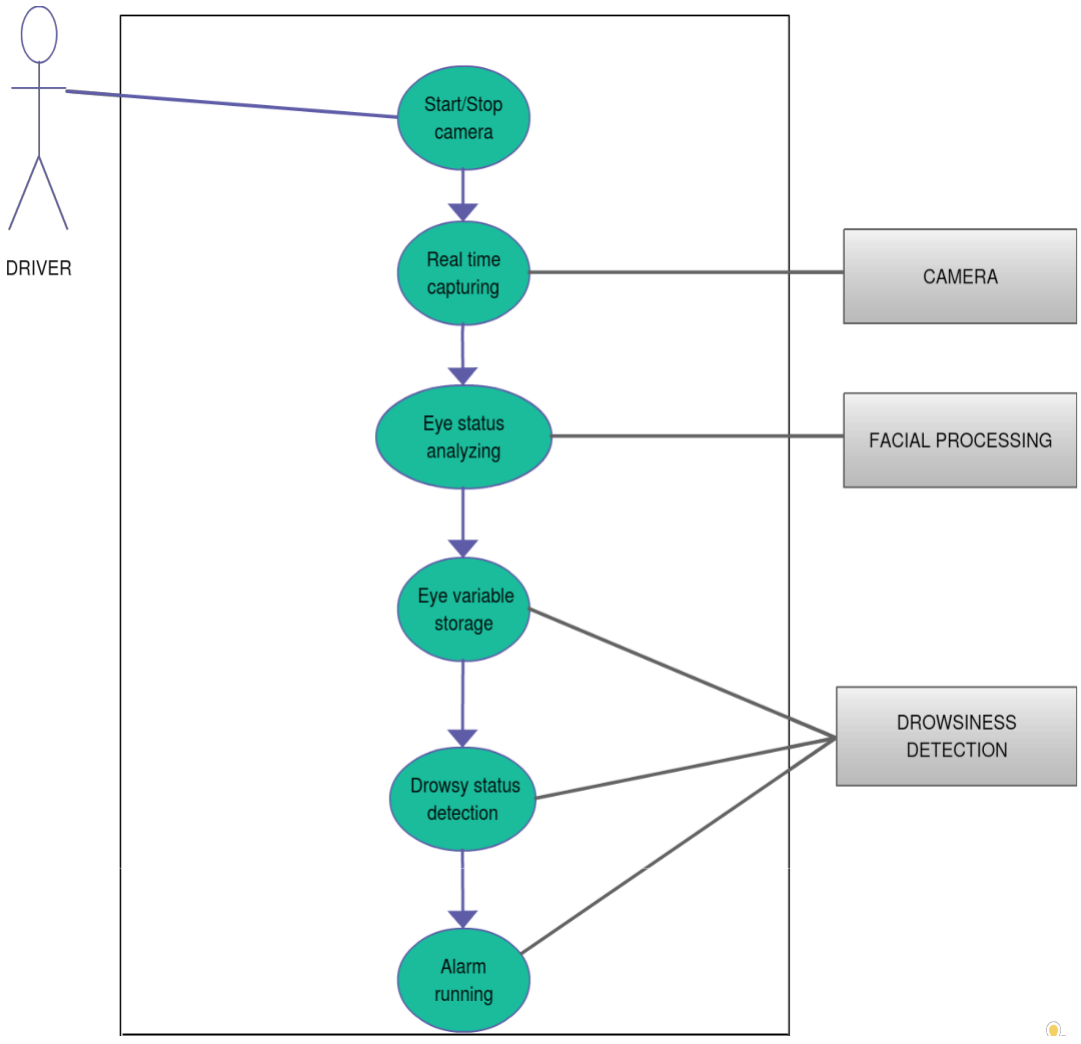
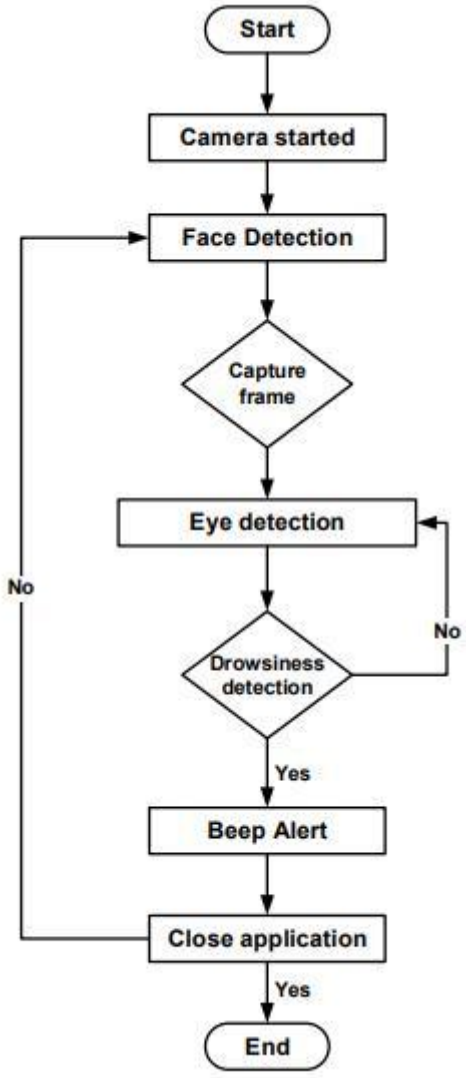



FIGURE 1

# *Flow Of System*




Block Diagram

# Chapter 8:

## 1. Methodology

This section presents the proposed architecture. The baseline architecture is described.

## Architecture

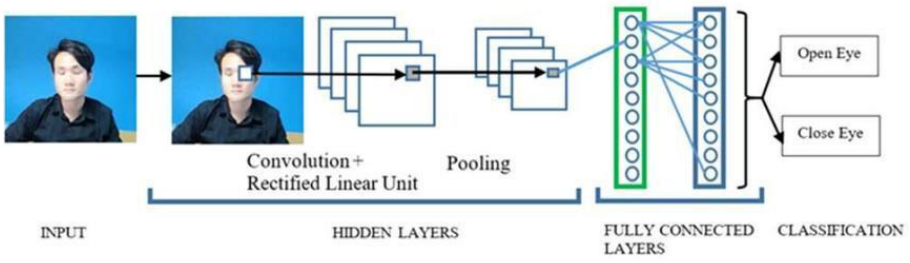
The complete design of the projected temporary state detection involves 2 phases as exemplified in Figure one. it's a ballroom dancing method within which the primary step is that the joint face detection and alignment and therefore the second is that the temporary state detection model. For the face detection and placement task, Multi-Task Cascaded Convolutional Networks (MTCNN) is employed since it's called one in all the quickest and correct face detector. Exploiting cascaded structure, it is able to do high speed in joint face detection and alignment. As a results of face detection and alignment, face boundary coordinates and 5 landmark points containing locations of left-eye, right-eye, nose, left-lip-end and right-lip-end are obtained.

Driver temporary state Detection (DDD) in second step indicates the projected models for detection driver's temporary state. DDD takes within the output of the primary step (face detection and alignment) as its input.

## Model Architecture

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

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



# Chapter 9:

## Steps for performing Driver Drowsiness Detection

- The "haar cascade files" folder consists of the xml files that are needed to detect objects from the image. In our case, we are detecting the face and eyes of the person.
- The models folder contains our model file "cnnCat2.h5" which was trained on convolutional neural networks.
- We have an audio clip "alarm.wav" which is played when the person is feeling drowsy.
- "Model.py" file contains the program through which we built our classification model by training on our dataset. You could see the implementation of convolutional neural network in this file.
- "Drowsiness detection.py" is the main file of our project. To start the detection procedure, we have to run this file.

Let's now understand how our algorithm works step by step.

**Step 1 – Take Image as Input from a Camera**

With a webcam, we will take images as input. So to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by OpenCV, **cv2.VideoCapture(0)** to access the camera and set the capture object (cap). **cap.read()** will read each frame and we store the image in a frame variable.

**Step 2 – Detect Face in the Image and Create a Region of Interest (ROI)**

To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don't need color information to detect the objects. We will be using haar cascade classifier to detect faces. This line is used to set our classifier **face = cv2.CascadeClassifier(' path to our haar cascade xml file')**. Then we perform the detection using **faces = face.detectMultiScale(gray)**. It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

```
for (x,y,w,h) in faces:

cv2.rectangle(frame, (x,y), (x+w, y+h), (100,100,100), 1 )
```

**Step 3 – Detect the eyes from ROI and feed it to the classifier**

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

```
l_eye = frame[ y : y+h, x : x+w ]
```

**l_eye** only contains the image data of the eye. This will be fed into our CNN classifier which will predict if eyes are open or closed. Similarly, we will be extracting the right eye into **r_eye**.

**Step 4 – Classifier will Categorize whether Eyes are Open or Closed**

We are using CNN classifier for predicting the eye status. To feed our image into the model, we need to perform certain operations because the model needs the correct dimensions to start with. First, we convert the color image into grayscale using **r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY)**. Then, we resize the image to 24*24 pixels as our model was trained on 24*24 pixel images **cv2.resize(r_eye, (24,24))**. We normalize our data for better convergence **r_eye = r_eye/255** (All values will be between 0-1). Expand the dimensions to feed into our classifier. We loaded our model using **model = load_model('models/cnnCat2.h5')** . Now we predict each eye with our model **lpred = model.predict_classes(l_eye)**. If the value of lpred[0] = 1, it states that eyes are open, if value of lpred[0] = 0 then, it states that eyes are closed.

**Step 5 – Calculate Score to Check whether Person is Drowsy**

The score is basically a value we will use to determine how long the person has closed his eyes. So if both eyes are closed, we will keep on increasing score and when eyes are open, we decrease the score. We are drawing the result on the screen using cv2.putText() function which will display real time status of the person.

```
cv2.putText(frame, "Open", (10, height-20), font, 1, (255,255,255), 1, cv2.LINE_AA )
```

A threshold is defined for example if score becomes greater than 15 that means the person's eyes are closed for a long period of time. This is when we beep the alarm using **sound.play()**.

# Dataset:

Example-

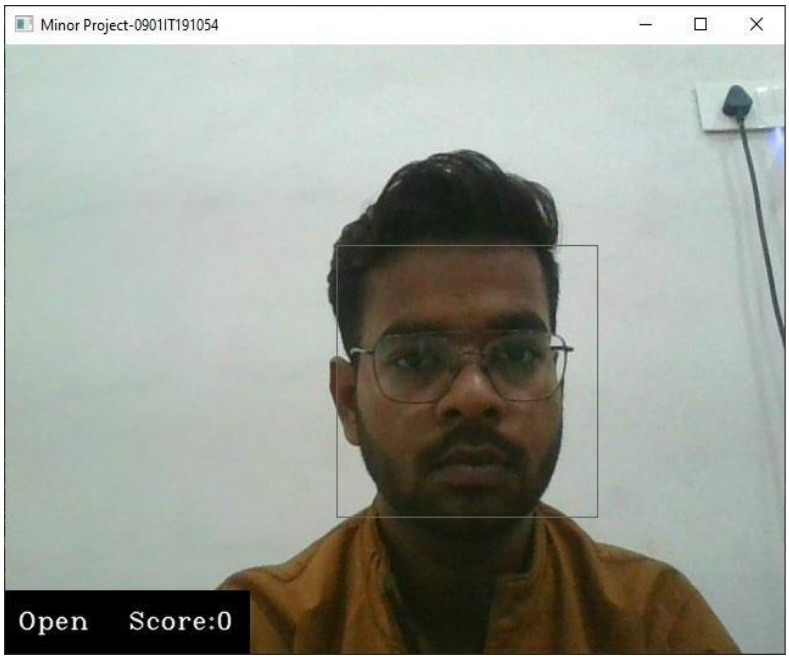
**Closed:**

**Open:**

*********

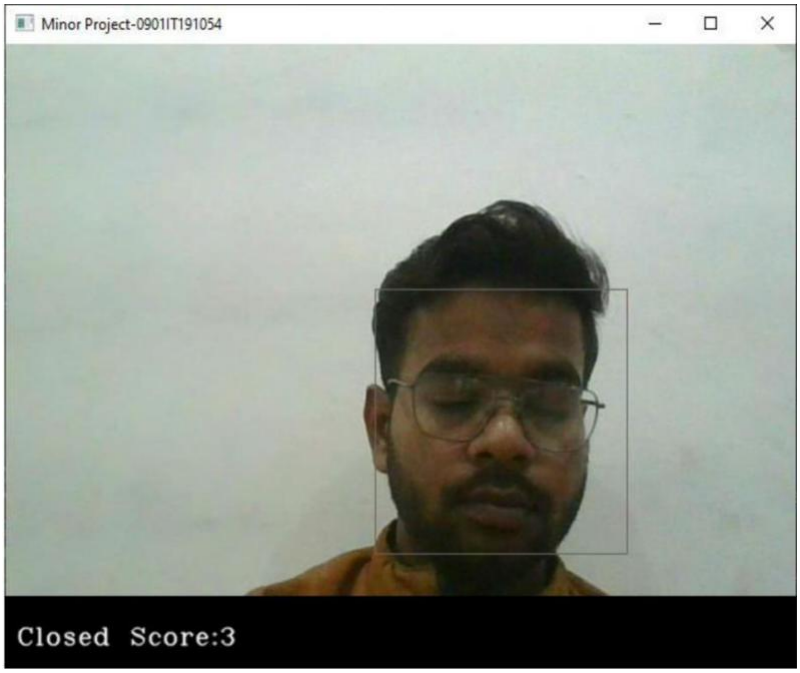
# Chapter 10:
# Results on proposed model

1. **Normal Face Detection**

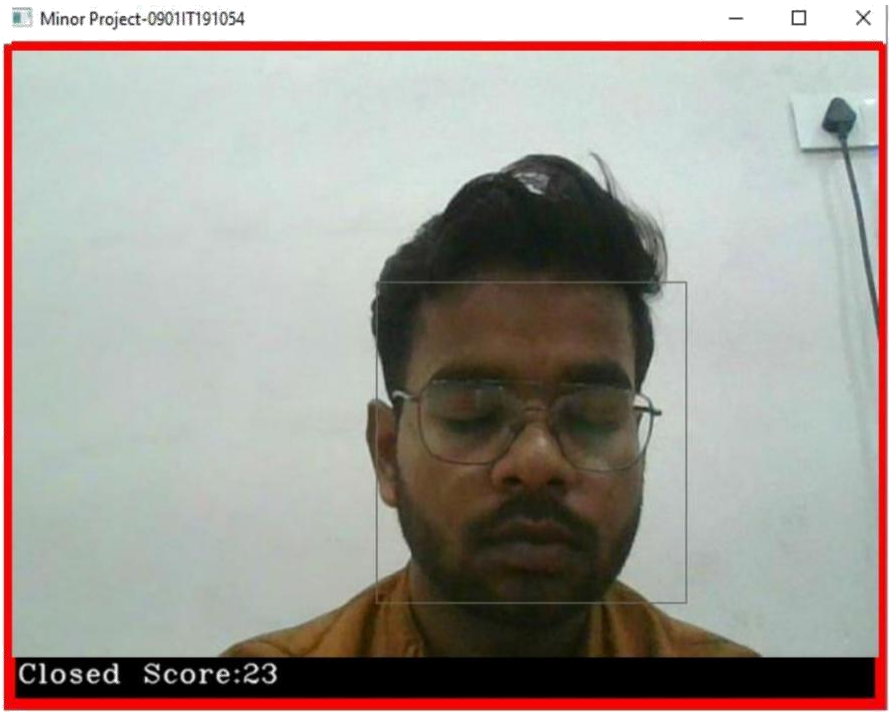



**Here, the score is 0 hence the model predicts that the person is not feeling drowsy.**

2. **Drowsy Eyes Detection**




**The score starts rising on the closure of eyes as detected by the system.**




**Here, the model predicts that the person has closed he's for more than 15 frames (threshold) determining that he is sleepy. The system will do the output frame red and with the beep alarm(alarm.wav) sound.**

*****

# Chapter 11:
# Summary and Colclusion

## Summary:-

To obtain the result a large number of videos were taken and their accuracy in determining eye and drowsiness was tested.

The system was tested for different people in different ambient lighting conditions( daytime and nighttime). When the webcam backlight was turned ON and the face is kept at an optimum distance, then the system is able to detect blinks as well as drowsiness with more than 95% accuracy. This is a good result and can be implemented in real-time systems as well.

## Conclusion:

It completely meets the objectives and requirements of the system. The framework has achieved an unfaltering state where all the bugs have been disposed of. The framework cognizant clients who are familiar withthe framework and comprehend it's focal points and the fact that it takescare of the issue of stressing out for individuals having fatigue-related issues to inform them about the drowsiness level while driving.

## Future Scope:

The model can be improved incrementally by using other parameters like blink rate, yawning, state of the car, etc. If all these parameters are used it can improve the accuracy by a lot.

We plan to further work on the project by adding a sensor to track the heart rate in order to prevent accidents caused due to sudden heart attacks to drivers and further adding the crash sensor which calls the emergency services and informing the family.

Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when the user is asleepand stop the video accordingly. It can also be used in application that prevents user from sleeping.

# Limitations:-

The limitations of the system are as follows.

1. **Dependence on ambient light:-** With poor lighting conditions even though face is easily detected, sometimes the system is unable to detect the eyes.
2. **Optimum range required:-** when the distance between face and webcam is not at optimum range then certain problems are arising.

   When face is too close to webcam(less than 30 cm) , then the system is unable to detect the face from the image. So it only shows the video as output as algorithm is designed so as to detect eyes from the face region.

   When face is away from the webcam(more than 70cm) then the backlight is insufficient to illuminate the face properly. So eyes are not detected with high accuracy which shows error in detection of drowsiness.
3. **Delay in sounding alarm:-** When drowsiness level exceeds a certain threshold, an alarm is produced by a system speaker. It requires a media player to run the audio file. There is a significant delay between when drowsiness is detected and when the media player starts and generates the alarm. But in real time, drowsiness is a continuous phenomenon rather than a one off occurrence. So the delay is not that problematic.
4. **Orientation of face:-** when the face is tilted to a certain extent it can be detected, but beyond this our system fails to detect the face. So when the face is not detected, eyes are also not detected.

   This problem is resolved by using tracking functions which track any movement and rotation of the objects in an image. A trained classifier for tilted face and tilted eyes can also be used to avoid this kind of problem.
5. **Poor detection with spectacles:-** When the driver wears glasses the system fails to detect eyes which is the most significant drawback of our system. This issue has not yet been resolved and is a challenge for almost all eye detection systems designed so far.
6. **Problem with multiple faces:-** If more than one face is detected by the webcam, then our system gives an erroneous result. This problem is not important as we want to detect the drowsiness of a single driver.

*****

Thus we have successfully designed a prototype drowsiness detection system using OpenCV software and Haar Classifiers. The system so developed was successfully tested, its limitations identified and a future plan of action developed.

# References


[1] http://www.ee.ryerson.ca/~phiscock/thesis/drowsy-detector/drowsy-detector.pdf

[2] https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[3] http://www.scribd.com/doc/15491045/Learning-OpenCV-Computer-Vision-with-the-OpenCV-Library

[4] Learning CNN and AN from:: https://towardsdatascience.com

[5] CNN by codebasics on youtube.

[6] Learning OpenCV by Murtaza's workshop- Robotics and AI

[7] Dataset for the model is taken from Kaggle.