

DRIVER DROWSINESS DETECTION

REPORT

ABSTRACT

The main idea behind this project is to develop a nonintrusive system which can detect fatigue of any human and can issue a timely warning. Drivers who do not take regular breaks when driving long distances run a high risk of becoming drowsy (a state which they often fail to recognize early enough).

According to the expert's studies show that around one quarter of all serious motorway accidents are attributable to sleepy drivers in need of a rest, meaning that drowsiness causes more road accidents than drink-driving. This system will monitor the driver eyes using a camera and by developing an algorithm we can detect symptoms of driver fatigue early enough to avoid the person from sleeping. So, this project will be helpful in detecting driver fatigue in advance and will give warning output in form of alarm and signals. Moreover, the warning will be deactivated if the score predicted by the algorithm falls below a certain cut- off.

Moreover, if driver feels drowsy, then his eyes get completely or partially closed. We use a data set of various persons whose eyes are open and closed. Based on that data, we apply an algorithm to provide a certain score to differentiate between open, closed and partially closed eyes. If the score crosses the cut-off level, then the driver is sleepy. An alarm and a red screen pops up to warn the driver.

Therefore, the driver gets warning to rest and the chances of accident reduces.

TABLE OF CONTENTS

1. Introduction.....	1
2. Literature and Review	2
3. Methodology	3
3.1 Implementation.....	4
3.1.1 User Interface	4
3.1.2 Hardware Interface	4
Hardware Specifiactions	4
3.1.3 Software Interface	4
Software Specifications	4
3.1.4 Decision Stages	4
3.1.4.1 Face Detection	5
3.1.4.2 Eyes Detection	6
3.1.4.3 Recognition of eyes state	7
3.1.4.4 Eye state determination	8
3.1.4.5 Drowsiness Detection	8
4. Modelling diagram(Project Workflow)	9
5. Coding and Testing... ..	12
6. Output and Results.....	15
7. Conclusion	16
8. References	17

1) INTRODUCTION.

1.1 PURPOSE

Real Time Sleepiness behaviours which are related to fatigue are in the form of eye closing, head nodding or the brain activity. Hence, we can either measure change in physiological signals, such as brain waves, heart rate and eye blinking to monitor drowsiness or consider physical changes such as sagging posture, leaning of driver's head and open/closed state of eyes.

The former technique, while more accurate, is not realistic since highly sensitive electrodes would have to be attached directly on the driver's body and hence which can be annoying and distracting to the driver. In addition, long time working would result in perspiration on the sensors, diminishing their ability to monitor accurately.

The second technique is to measure physical changes (i.e. open/closed eyes to detect fatigue) is well suited for real world conditions since it is non-intrusive by using a video camera to detect changes. In addition, micro sleeps that are short period of sleeps lasting 2 to 3 minutes are good indicators of fatigue. Thus, by continuously monitoring the eyes of the driver one can detect the sleepy state of driver and a timely warning is issued.

1.2 OBJECTIVE

The main objective of our project is to build a model using python, which helps to monitor driver's sleepiness by observing the shape of his eyes. If found sleepy, our system warns the driver to sleep by playing an alarm that wakes him up.

With this Python project, we will be making a sleepiness detecting device. A countless number of people drive on the highway day and night. Taxi drivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep. Due to which it becomes very dangerous to drive when feeling sleepy.

The majority of accidents happen due to the drowsiness of the driver. So, to prevent these accidents we will build a system using Python, OpenCV, and Keras which will alert the driver when he feels sleepy.

1.3 MOTIVATION

In today's world, majority of accident happens due to driver's lack of sleep. A 2014 AAA Traffic Safety Foundation study found that 37 percent of drivers report having fallen asleep behind the wheel at some point in their lives. An estimated 21 percent of fatal crashes, 13 percent of crashes resulting in severe injury and 6 percent of all crashes, involve a drowsy driver.

So, to overcome this issue, we decided to make a system that may help to reduce this problem in a friendly way.

2) LITERATURE AND REVIEW

In this section, we have discussed various methodologies that have been proposed by researchers for sleepiness detection and blink detection during the recent years.

Manu B.N in 2016, has proposed a method that detect the face using Haar feature-based cascade classifiers. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier that will detect the object. So along with the Haar feature-based classifiers, cascaded Adaboost classifier is exploited to recognize the face region then the compensated image is segmented into numbers of rectangle areas, at any position and scale within the original image. Due to the difference of facial feature, Haarlike feature is efficient for real-time face detection. These can be calculated according to the difference of sum of pixel values within rectangle area and during the process the Adaboost algorithm will allow all the face samples and it will discard the non-face samples of images.

Amna Rahman in 2015, has proposed a method to detect the sleepiness by using Eye state detection with Eye blinking strategy. In this method first, the image is converted to gray scale and the corners are detected using Harris corner detection algorithm which will detect the corner at both side and at down curve of eye lid. After tracing the points then it will make a straight line between the upper two points and locates the mid-point by calculation of the line, and it connects the mid-point with the lower point. Now for each image it will perform the same procedure and it calculates the distance 'd' from the mid-point to the lower point to determine the eye state. Finally, the decision for the eye state is made based on distance 'd' calculated. If the distance is zero or is close to zero, the eye state is classified as "closed" otherwise the eye state is identified as "open". They have also invoked intervals or time to know that the person is feeling drowsy or not. This is done by the average blink duration of a person is 100-400 milliseconds (i.e. 0.1-0.4 of a second)

3) **METHODOLOGY**

There are different types of methodologies that have been developed to find out sleepiness:

1) **Physiological level approach:** This technique is an intrusive method wherein electrodes are used to obtain pulse rate, heart rate and brain activity information. ECG is used to calculate the variations in heart rate and detect different conditions for drowsiness. The correlation between different signals such as ecg (electrocardiogram), EEG (electroencephalogram), and EMG (electromyogram) are made and then the output is generated whether the person is drowsy or not.

2) **Behavioural based approach:** In this technique eye blinking frequency, head pose, etc. of a person is monitored through a camera and the person is alerted if any of these drowsiness symptoms are detected.

The various technologies that can be used are discussed as follows:

1) **TensorFlow:** It is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production. TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors".

2) **Machine learning:** Machine learning is the kind of programming which gives computers the capability to automatically learn from data without being explicitly programmed. This means in other words that these programs change their behavior by learning from data. Python is clearly one of the best languages for machine learning. Python does contain special libraries for machine learning namely **scipy**, **pandas** and **numpy** which are great for linear algebra and getting to know kernel methods of machine learning. The language is great to use when working with machine learning algorithms and has easy syntax relatively.

3) **OpenCV:** OpenCV stands for Open Source Computer Vision. It's an Open Source BSD licensed library that includes hundreds of advanced Computer Vision algorithms that are optimized to use hardware acceleration. OpenCV is commonly used for machine learning, 4 image processing, image manipulation, and much more. OpenCV has a modular structure. There are shared and static libraries and a CV Namespace. In short, OpenCV is used in our application to easily load bitmap files that contain landscaping pictures and perform a blend operation between two pictures so that one picture can be seen in the background of another picture. This image manipulation is easily performed in a few lines of code using OpenCV versus other methods. OpenCV.org is a must if you want to explore and dive deeper into image processing and machine learning in general.

3.1 **IMPLEMENTATION**

3.1.1 **USER INTERFACES**

User interface requires a computer system with following tools and software installed on it :

- Python 3.6
- Tensorflow
- Keras
- OpenCV
- Pygame
- Sublime text editor

3.1.2 **HARDWARE INTERFACES**

Hardware Interface plays an important role in co-design of the embedded computer system. It links the software part and the hardware part of the system. The design process supports hardware interface synthesis. This report shows how the hardware and software interfaces can be implemented in embedded computer system.

Hardware Specifications

- RAM:4 GB or higher (+ 2GB for GPU)
- DiskSpace:2 GB or higher,4GB Recommended
- Screen Resolution : 1280 x 800 or higher

3.1.3 **SOFTWARE INTERFACES**

Software interface also plays an important role in coding. If the software of the computer is not latest it will not have new libraries and will not be able to provide full functioning.

Software specifications

- OS :Windows10
- PythonVersion:3.6.X.
- Compatible tools: Sublime text editor, Pycharm, Tensorflow, Keras, Pygame and OpenCV
- Other S/W:GUI Too

3.1.4 **The various detection stages are discussed as:**

3.14.1) **Face Detection:** For the face Detection, it uses Haar feature-based cascade classifiers. It is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle. Fig. 3.1 represents five haar like features & example is shown in Fig.3.2

Fig. 3.1

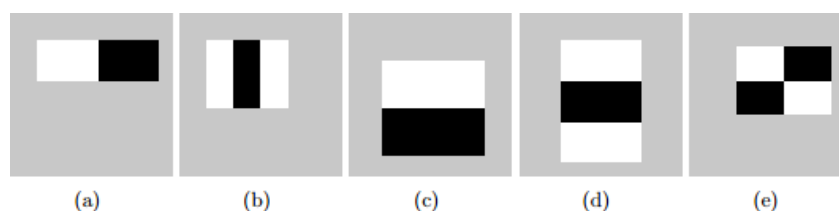
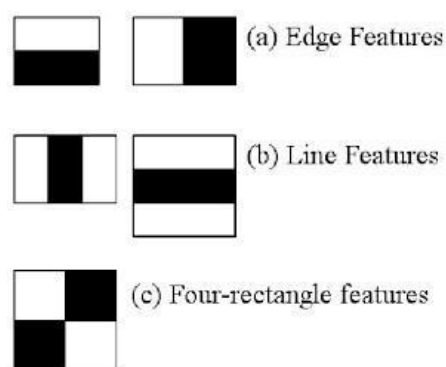


Fig. 3.2



A cascaded Adaboost classifier with the Haar-like features is exploited to find out the face region. First, the compensated image is segmented into numbers of rectangle areas, at any position and scale within the original image. Due to the difference of facial feature, Haar-like feature is efficient for real-time face detection. These can be calculated according to the difference of sum of pixel values within rectangle areas. The features can be represented by the different composition of the black region and white region. A cascaded Adaboost classifier is a strong classifier which is a combination of several weak classifiers. Each weak classifier is trained by Adaboost algorithm. If a candidate sample passes through the cascaded Adaboost classifier, the face region can be found. Almost all of face samples can pass through and nonface samples can be rejected.

3.1.42) **Eye detection**: In the system we have used facial landmark prediction for eye detection. Facial landmarks are used to localize and represent salient regions of the face, such as:

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. In the context of facial landmarks, our goal is detecting important facial structures on the face using shape prediction methods. Detecting facial landmarks is therefore a twostep process:

- Localize the face in the image.
- Detect the key facial structures on the face ROI.

Localize the face in the image: The face image is localized by Haar feature-based cascade classifiers which was discussed in the first step of our algorithm i.e. face detection. Detect the key facial structures on the face ROI: There are a variety of facial landmark detectors, but all methods essentially try to localize and label the following facial regions:

- Mouth
- Right eyebrow
- Left eyebrow
- Right eye
- Left eye
- Nose

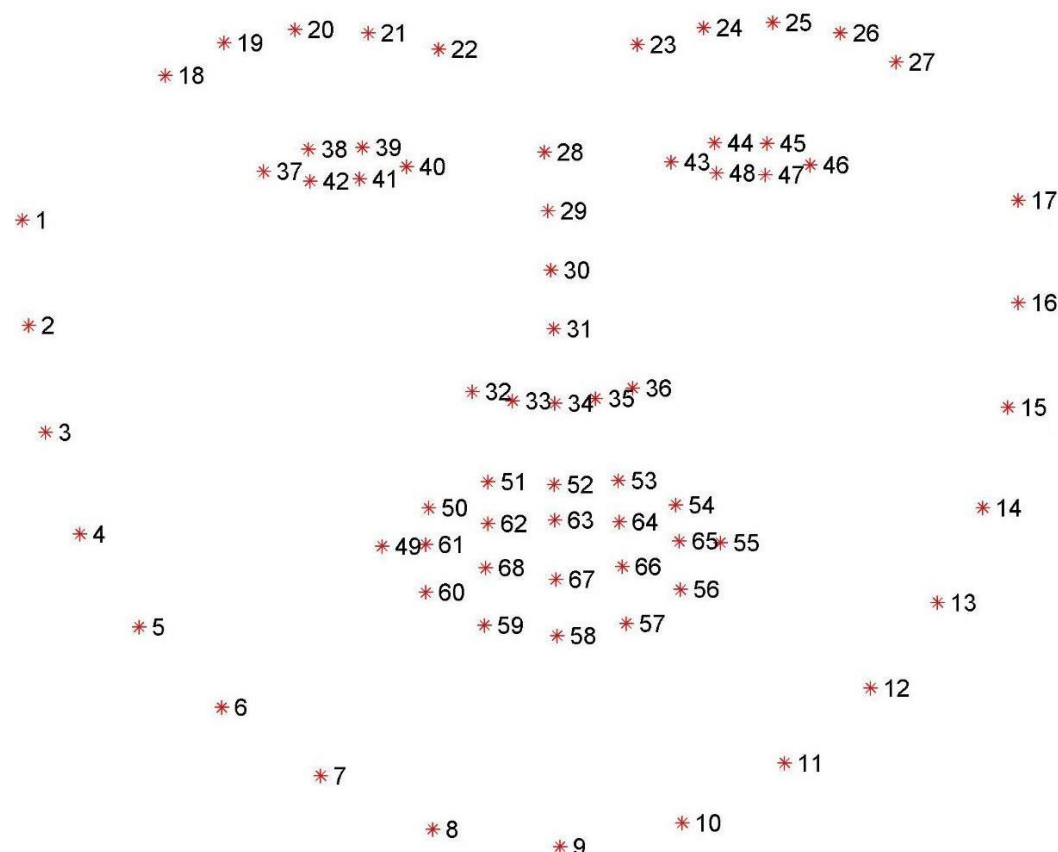
The facial landmark detector included in the dlib library is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).

This method starts by using:

1. A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure. 2. Priors, of more specifically, the probability on distance between pairs of input pixels.

The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.

The indexes of the 68 coordinates can be visualized on the image below:



We can detect and access both the eye region by the following facial landmark index show below:

- The right eye using [36, 42].
- The left eye with [42, 48].

These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on. It's important to note that other flavors of facial landmark detectors exist, including the 194 point model that can be trained on the HELEN dataset.

Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data.

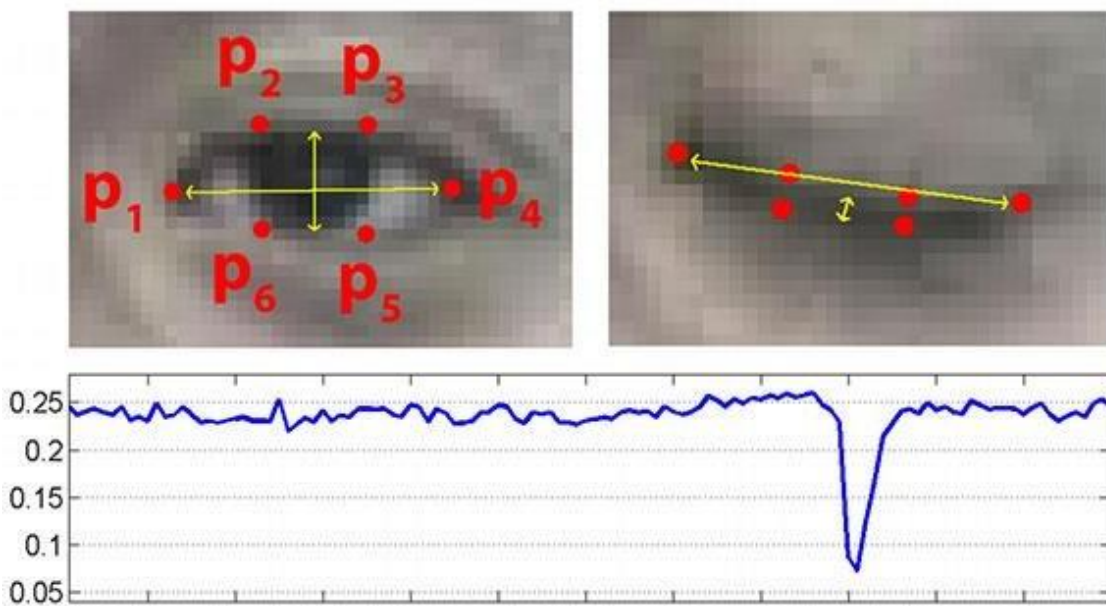
3.1.43) **Recognition of Eye's State:** The eye area can be estimated from optical flow, by sparse tracking or by frame-to-frame intensity differencing and adaptive thresholding. And Finally, a decision is made whether the eyes are or are not covered by eyelids. A different approach is to infer the state of the eye opening from a single image, as e.g. by correlation matching with open and closed eye templates, a heuristic horizontal or vertical image intensity projection over the eye region, a parametric model fitting to find the eyelids, or active shape models. A major drawback of the previous approaches is that they usually implicitly impose too strong requirements on the setup, in the sense of a relative face-camera pose (head orientation), image resolution, illumination, motion dynamics, etc. Especially the heuristic methods that use raw image intensity are likely to be very sensitive despite their real-time performance.

Therefore, we propose a simple but efficient algorithm to detect eye blinks by using a recent facial landmark detector. A single scalar quantity that reflects a level of the eye opening is derived from the landmarks. Finally, having a per-frame sequence of the eye-opening estimates, the eye blinks are found by an SVM classifier that is trained on examples of blinking and no blinking patterns.

Eye Aspected Ratio Calculation: For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

$$\text{EAR} = (\|p_2 - p_6\| + \|p_3 - p_5\|) / (2\|p_1 - p_4\|)$$

where p_1, \dots, p_6 are the 2D landmark locations. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals, and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.



3.144) **Eye State Determination:** Finally, the decision for the eye state is made based on EAR calculated in the previous step. If the distance is zero or is close to zero, the eye state is classified as “closed” otherwise the eye state is identified as “open”.

3.145) **Drowsiness Detection:**

The last step of the algorithm is to determine the person’s condition based on a pre-set condition for drowsiness. The average blink duration of a person is 100-400 milliseconds (i.e. 0.1-0.4 of a second). Hence if a person is drowsy his eye closure must be beyond this interval. We set a time frame of 5 seconds. If the eyes remain closed for five or more seconds, drowsiness is detected and alert pop regarding this is triggered.

4) Modelling Diagram (Project Workflow):

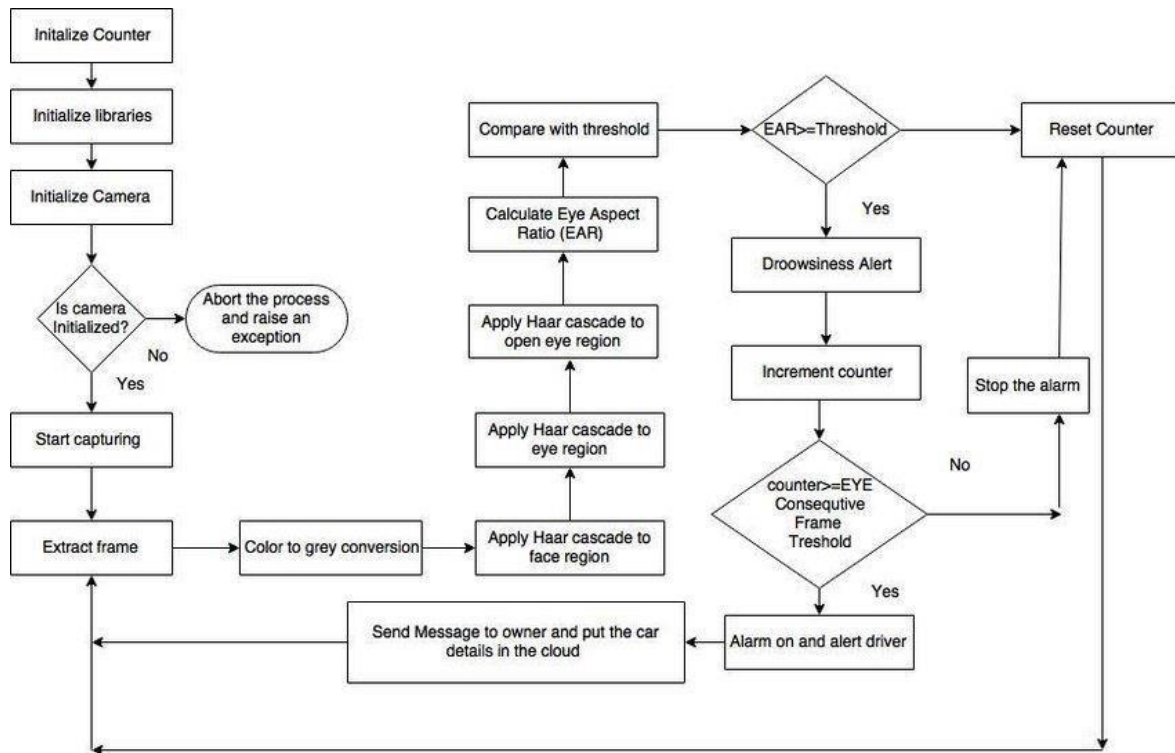


Fig. 4.1 Flow chart of System

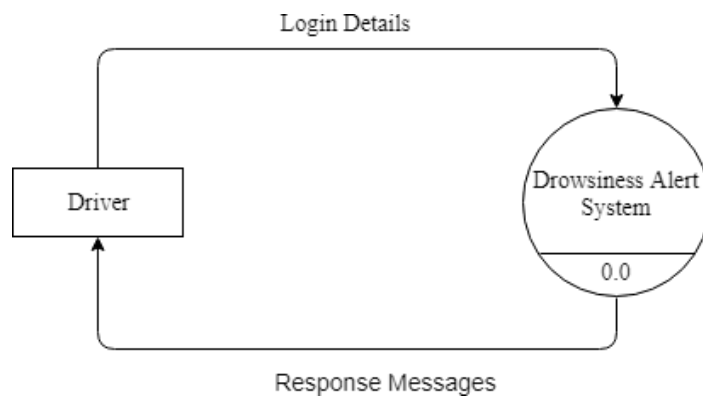


Fig.4.2 DFD level 0

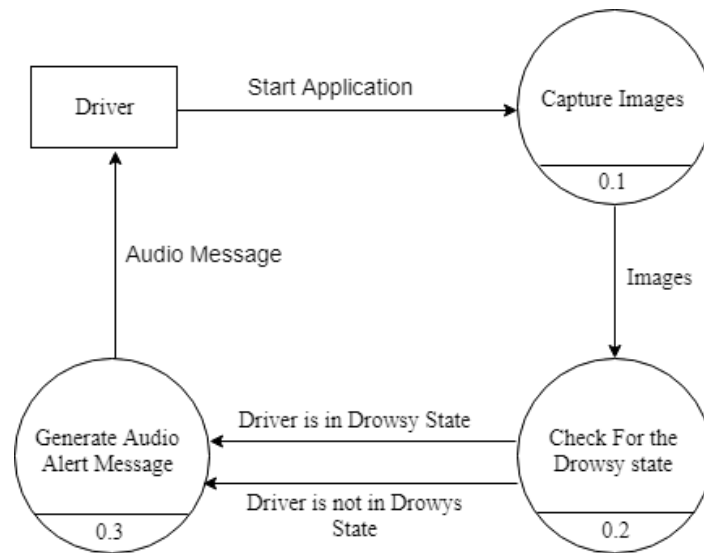


Fig 4.3 DFD level 1

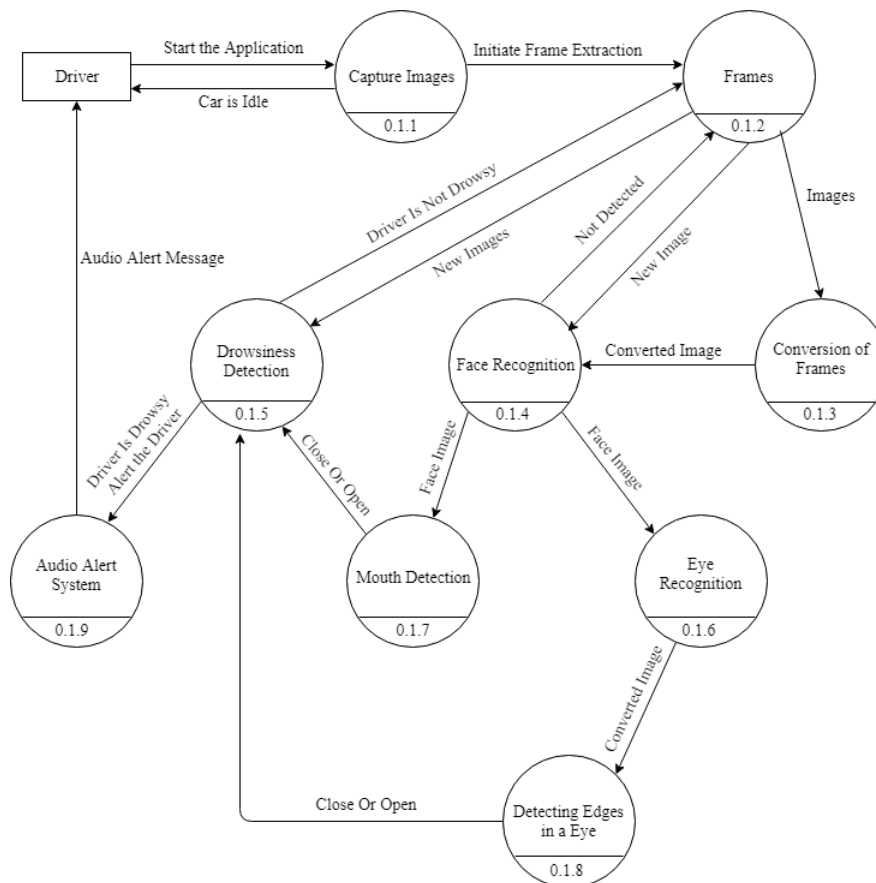


Fig 4.4 DFD level 2

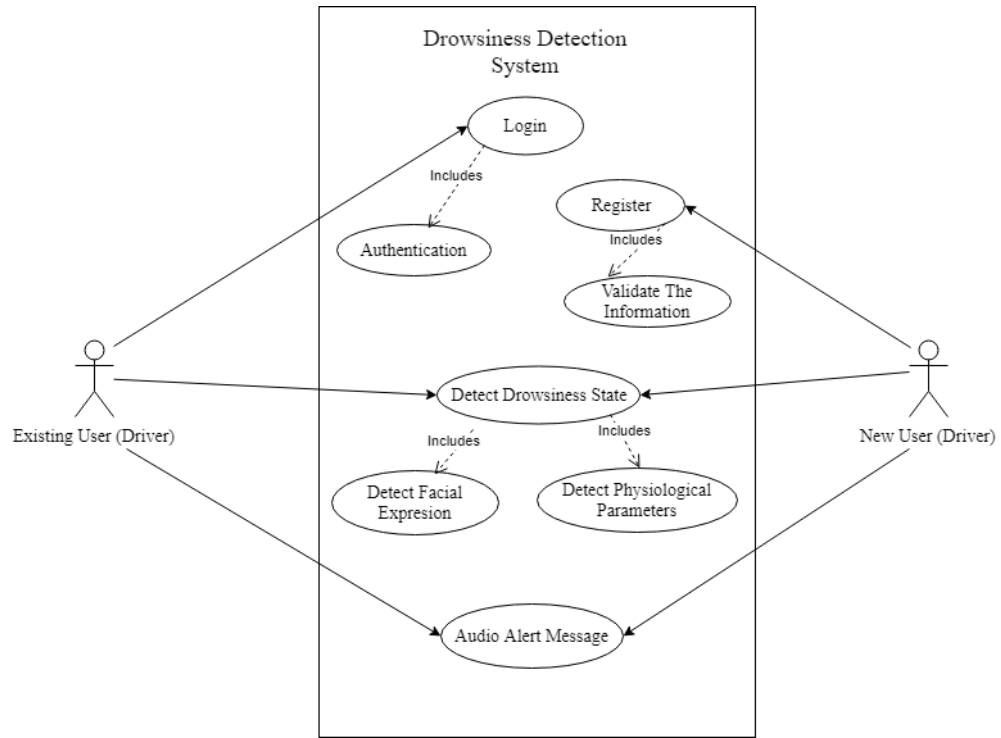


Fig 4.5 use-case Diagram

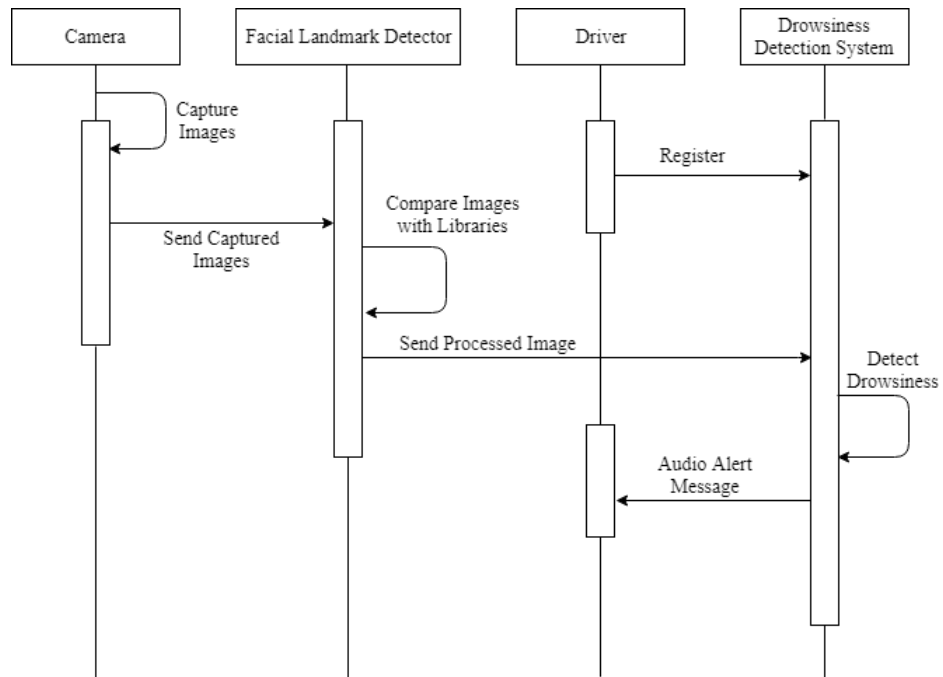
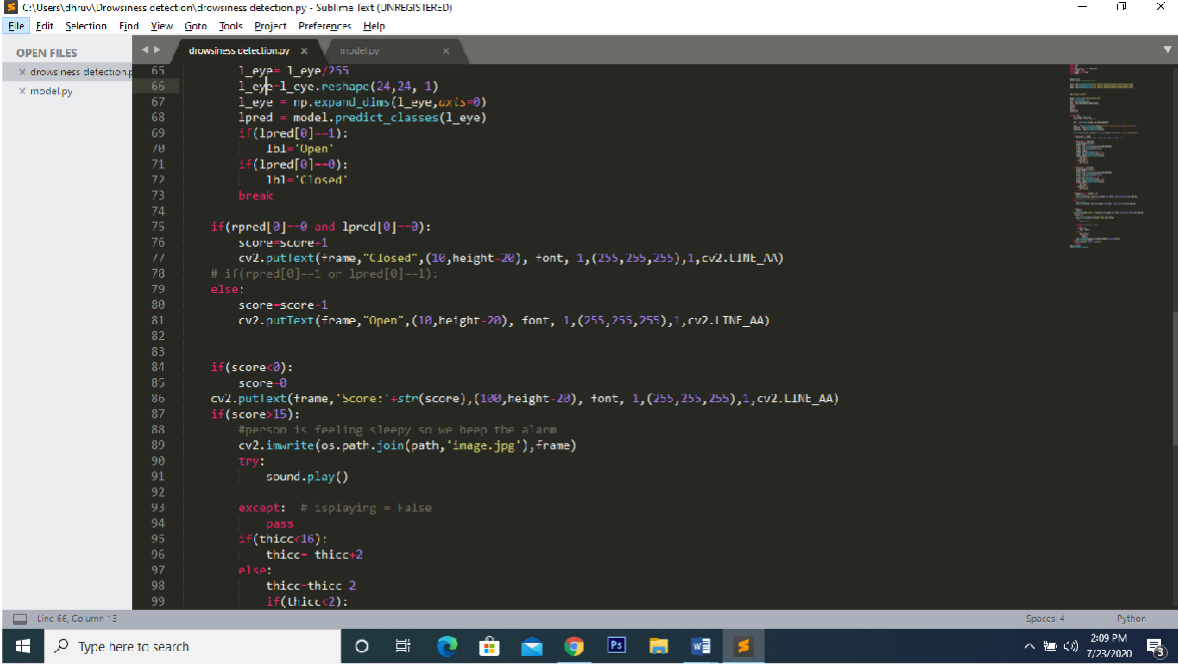
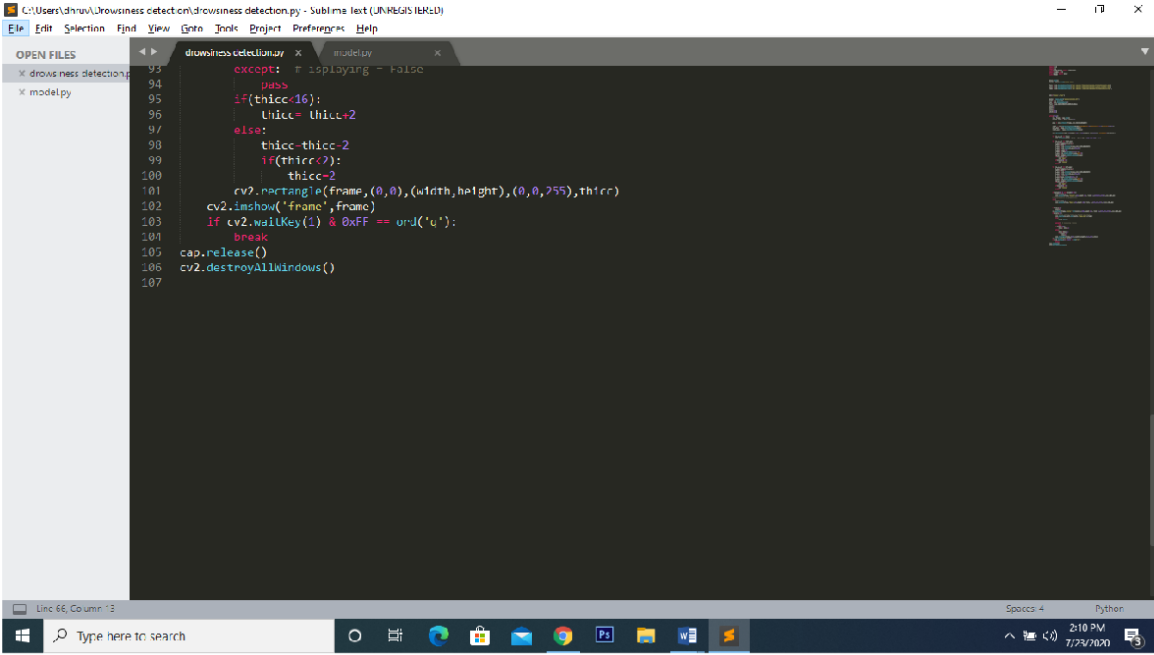


Fig 4.6 Sequence Diagram

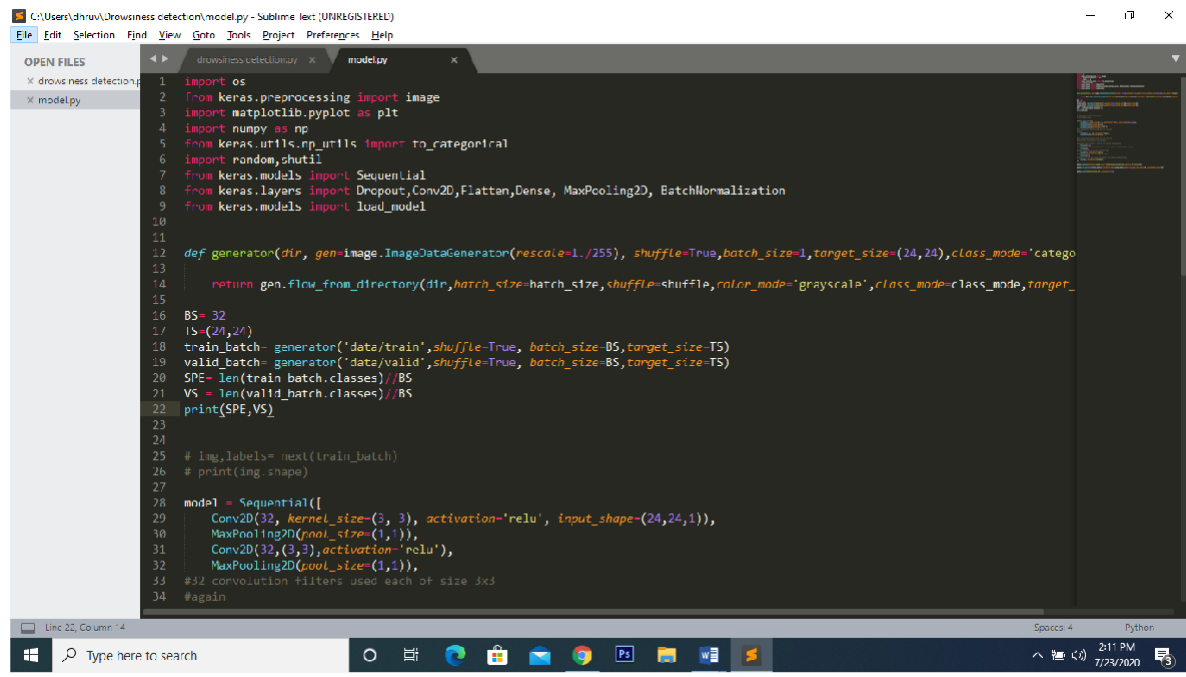


```
1_eye=1_eye/255
1_eye=1_eye.reshape(24,24,1)
1_eye=np.expand_dims(1_eye,axis=0)
lpred=model.predict_classes(1_eye)
if(lpred[0]==1):
    lbl='Open'
    if(lpred[0]==0):
        lbl='Closed'
    break
if(rpred[0]==0 and lpred[0]==0):
    score=score+1
    cv2.putText(frame,"Closed",(10,height-20),font,1,(255,255,255),1,cv2.LINE_AA)
    # if(rpred[0]==1 or lpred[0]==1):
else:
    score=score-1
    cv2.putText(frame,"Open",(10,height-20),font,1,(255,255,255),1,cv2.LINE_AA)

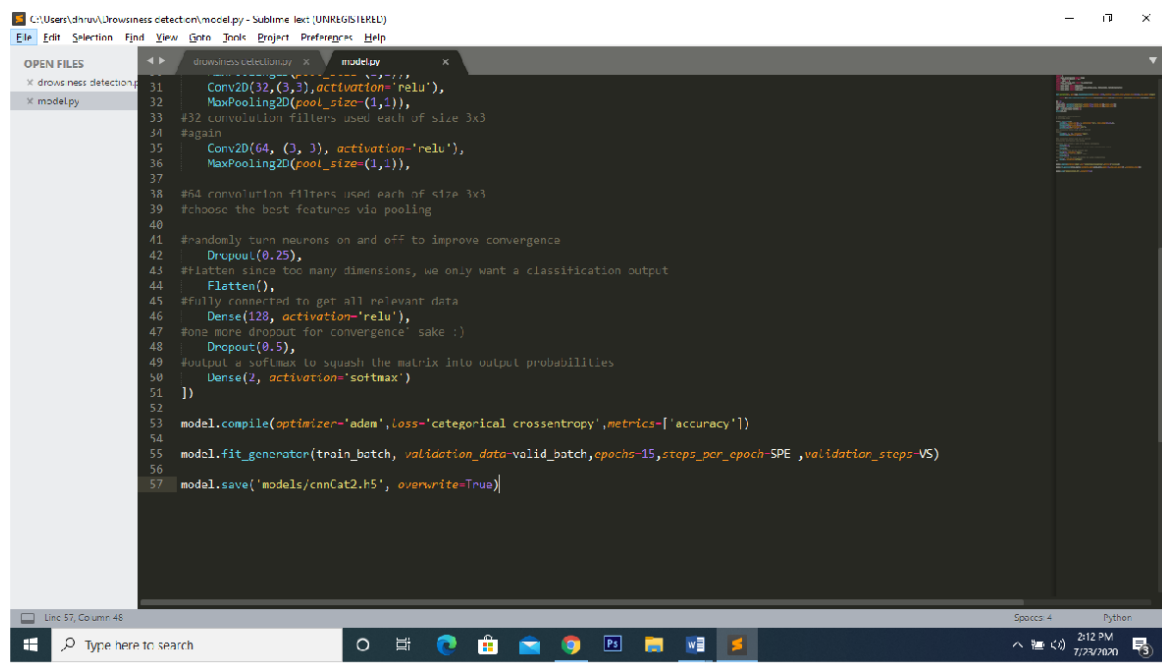
if(score<0):
    score=0
cv2.putText(frame,"score:"+str(score),(100,height-20),font,1,(255,255,255),1,cv2.LINE_AA)
if(score>15):
    #person is feeling sleepy so we beep the alarm
    cv2.imwrite(os.path.join(path,"image.jpg"),frame)
    try:
        sound.play()
    except: # isplaying = false
        pass
    if(thicc<16):
        thicc=thicc+2
    else:
        thicc=thicc-2
        if(thicc<2):
            thicc=2
cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
cv2.imshow('frame',frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```



```
except: # isplaying = false
    pass
if(thicc<16):
    thicc=thicc+2
else:
    thicc=thicc-2
    if(thicc<2):
        thicc=2
cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
cv2.imshow('frame',frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

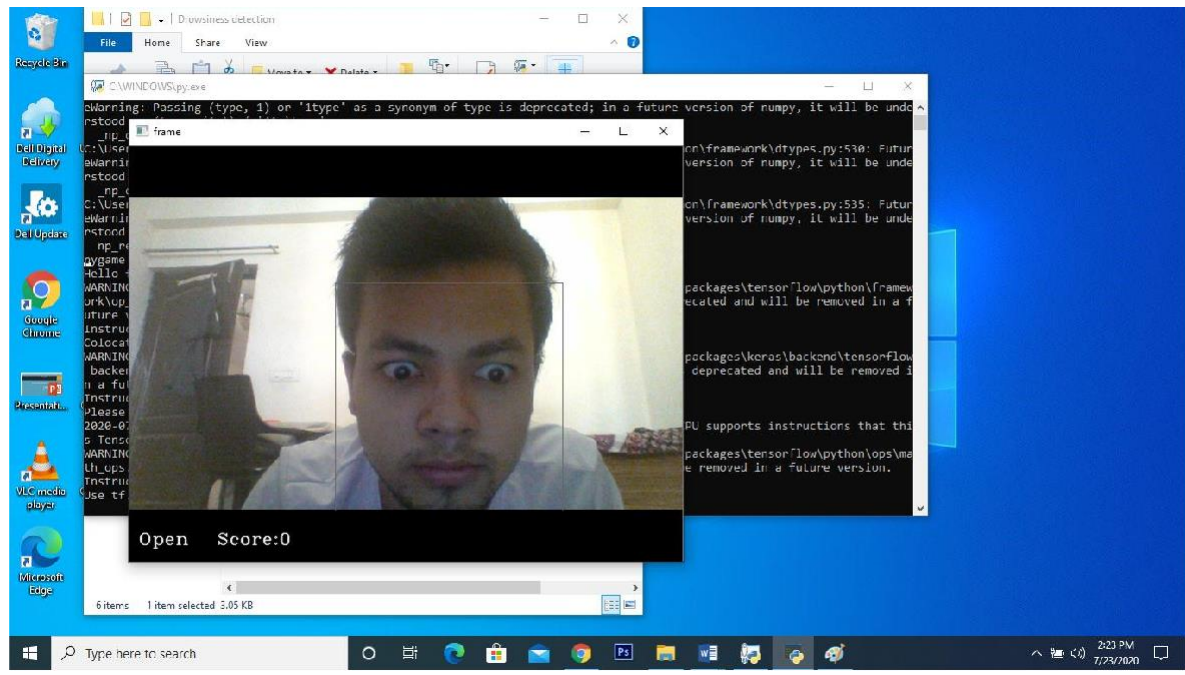



```
1 import os
2 from keras.preprocessing import image
3 import matplotlib.pyplot as plt
4 import numpy as np
5 from keras.utils.np_utils import to_categorical
6 import random,shutil
7 from keras.models import Sequential
8 from keras.layers import Dropout,Conv2D,Flatten,Dense, MaxPooling2D, BatchNormalization
9 from keras.models import load_model
10
11
12 def generator(dir, gen=image.ImageDataGenerator(rescale=1./255), shuffle=True,batch_size=1,target_size=(24,24),class_mode='catego
13
14     return gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,color_mode='grayscale',class_mode=class_mode,target_
15
16 BS= 32
17 IS=(24,24)
18 train_batch= generator('data/train',shuffle=True, batch_size=BS,target_size=IS)
19 valid_batch= generator('data/valid',shuffle=True, batch_size=BS,target_size=IS)
20 SPE= len(train_batch.classes)//BS
21 VS = len(valid_batch.classes)//BS
22 print(SPE,VS)
23
24
25 # img,labels= next(train_batch)
26 # print(img.shape)
27
28 model = Sequential([
29     Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
30     MaxPooling2D(pool_size=(1,1)),
31     Conv2D(32,(3,3),activation='relu'),
32     MaxPooling2D(pool_size=(1,1)),
33     #32 convolution filters used each of size 3x3
34     #again
35     Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
36     MaxPooling2D(pool_size=(1,1)),
37     #64 convolution filters used each of size 3x3
38     #choose the best features via pooling
39     #randomly turn neurons on and off to improve convergence
40     Dropout(0.25),
41     #flatten since too many dimensions, we only want a classification output
42     Flatten(),
43     #fully connected to get all relevant data
44     Dense(128, activation='relu'),
45     #one more dropout for convergence's sake :)
46     Dropout(0.5),
47     #output a softmax to squash the matrix into output probabilities
48     Dense(2, activation='softmax')
49 ])
50
51
52
53 model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
54
55 model.fit_generator(train_batch, validation_data=valid_batch,epochs=15,steps_per_epoch=SPE ,validation_steps=VS)
56
57 model.save('models/cnnCat2.H5', overwrite=True)
```

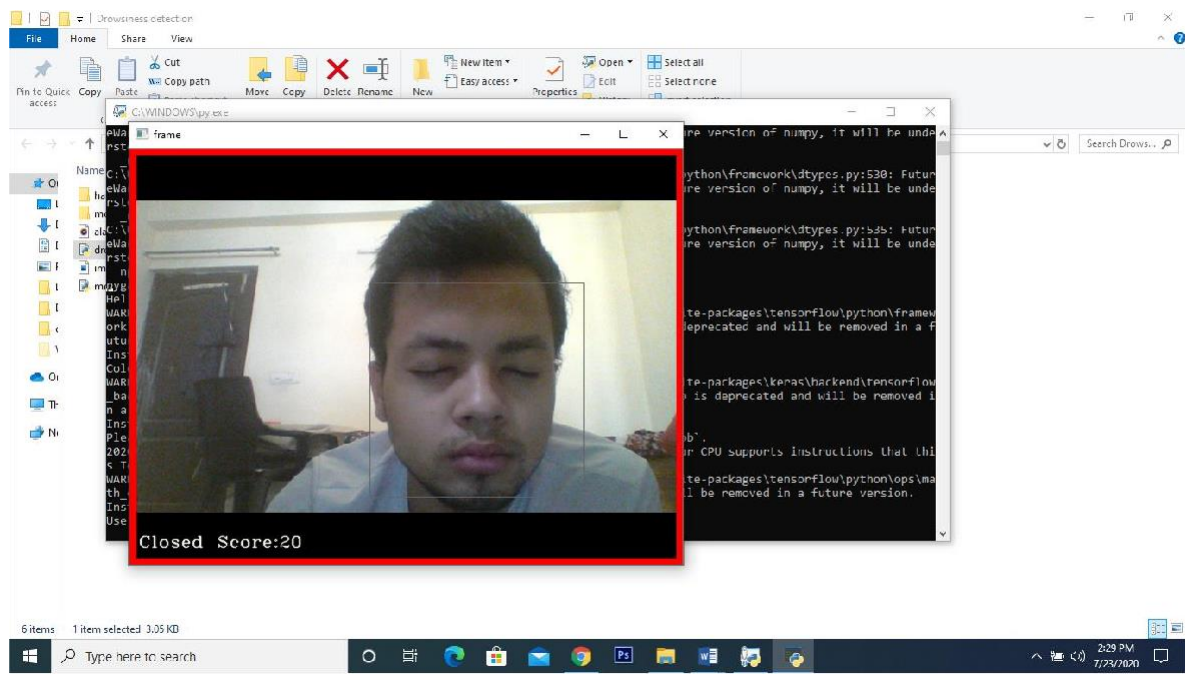


```
31     Conv2D(32,(3,3),activation='relu'),
32     MaxPooling2D(pool_size=(1,1)),
33     #32 convolution filters used each of size 3x3
34     #again
35     Conv2D(64, (3, 3), activation='relu'),
36     MaxPooling2D(pool_size=(1,1)),
37
38     #64 convolution filters used each of size 3x3
39     #choose the best features via pooling
40
41     #randomly turn neurons on and off to improve convergence
42     Dropout(0.25),
43     #flatten since too many dimensions, we only want a classification output
44     Flatten(),
45     #fully connected to get all relevant data
46     Dense(128, activation='relu'),
47     #one more dropout for convergence's sake :)
48     Dropout(0.5),
49     #output a softmax to squash the matrix into output probabilities
50     Dense(2, activation='softmax')
51 ])
52
53
54 model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
55
56 model.fit_generator(train_batch, validation_data=valid_batch,epochs=15,steps_per_epoch=SPE ,validation_steps=VS)
57
58 model.save('models/cnnCat2.H5', overwrite=True)
```

6) OUTPUT AND RESULTS:



EYES OPEN STATE: No alarm and no red screen



EYES CLOSED STATE: Alarm beeps and red screen as a warning to the driver

7) CONCLUSION

7.1 Conclusions

A real-time eye blink detection algorithm was presented. We quantitatively demonstrated that Haar feature-based cascade classifiers and regression-based facial landmark detectors are precise enough to reliably estimate the positive images of face and a level of eye openness. While they are robust to low image quality (low image resolution in a large extent) and in-the-wild.

Limitations:

Use of spectacles: In case the user uses spectacle then it is difficult to detect the state of the eye. As it hugely depends on light hence reflection of spectacles may give the output for a closed eye as opened eye. Hence for this purpose the closeness of eye to the camera is required to avoid light.

Multiple face problem: If multiple face arises in the window then the camera may detect more number of faces undesired output may appear. Because of different condition of different faces. So, we need to make sure that only the driver face come within the range of the camera. Also, the speed of detection reduces because of operation on multiple faces.

7.2 Future Work

We are very excited by the vast future possibilities that our project has to offer. Possible improvements include getting rid of spectacle and multiple face problem. We are also looking forward to apply this concept to other fields as well. Its other applications involve students, security forces etc. when they get drowsy. It will warn them to take rest and discontinue from their work for a certain amount of time.

The major objective is to reduce the number of road accidents due to lack of sleep.

This is future of our project we are looking at and looking forward to implementing all of the above successfully.

8) **REFERENCES:**

IEEE standard:

Journal Paper,

[1] Facial Features Monitoring for Real Time Drowsiness Detection by Manu B.N, 2016 12th International Conference on Innovations in Information Technology (IIT) [Pg. 78- 81]

<https://ieeexplore.ieee.org/document/7880030>

[2] Real Time Drowsiness Detection using Eye Blink Monitoring by Amna Rahman Department of Software Engineering Fatima Jinnah Women University 2015 National Software Engineering Conference (NSEC 2015)

<https://ieeexplore.ieee.org/document/7396336>

[3] Implementation of the Driver Drowsiness Detection System by K. Sriyathi International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 9, September 2013

Names of Websites referred:

<https://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-Of-Human-Eyes-Using-a>

<https://realpython.com/face-recognition-with-python/>

<https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>

<https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>

<https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/>

<https://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-Of-HumanEyesUsing-a>

https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html

<https://www.learnopencv.com/training-better-haar-lbp-cascade-eye-detector-opencv/>

THANK YOU