SOFT COMPUTING

TOPIC : Driver Drowsiness Detection Based on Eye State

FACULTY : Neelu Khare

GROUP MEMBERS :

➢**Saksham Sharma**	**19BIT0056**
➢**Krittika Chaturvedi**	**19BIT0071**
➢**Ujjawal Srivastava**	**19BIT0072**

OBJECTIVE

Most of the accidents that happen today are the results of the fatigue and the drowsiness encountered by the driver driving the vehicle

During long trips people tend to fall asleep during the driving that results in some fatal accidents. Some researchers have proposed The driver Drowsiness detection system helps us to prevent these accidents and provide us with a reliable system that studies the following pattern of the driver.

1-Vehicle movements measurements like sudden steering movements,deviation of lanes,pressure on the acceleration pedal.

2-Behaviour of the driver while driving for example yawning ,eye closure,head pose.

3-Duration of the journey and the fatigue endured by an average person in that time. This data completely make the basis of the system working and giving the desired output

THE PROPOSED SYSTEM USING DEEP CNN BASED DROWSINESS DETECTION SYSTEM

Facial expressions- Drivers don’t fall asleep suddenly they tend to show some physical behavior leading up them to fall asleep. The yawning,rapid eye blinking,unusual head pose are the critical points that can establish a basis to the conclusion that the driver is getting sleepy.Computerized, non-intrusive, behavioral approaches are widely used for determining the drowsiness level of drivers by measuring their abnormal behaviors. PERCLOS (which is the percentage of eyelid closure over the pupil over time, reflecting slow eyelid closures, or “droops”, rather than blinks) has been analyzed in many studies.This measurement has been found to be a reliable measure to predict drowsiness and has been used in commercial products such as Seeing Machines and Lexus.The main limitation of using a vision-based approach is lighting. Normal cameras do not perform well at night. In order to overcome this limitation, some researchers have used active illumination utilizing an infrared Light Emitting Diode (LED).Mostly, image is acquired using simple CCD or web camera during day and IR camera during night at around 30 fps.After localizing the specific region of interest within the image, features such as PERCLOS, yawning frequency and head angle, are extracted using an efficient feature extraction technique, such as Wavelet Decomposition, Gabor Wavelets, Discrete Wavelet Transform or Condensation Algorithm.. The determination of drowsiness using PERCLOS and Eye Blink has a success rate of close to 100% and 98% , respectively. However it has to be noted that the high positive detection rate achieved was when the subjects didn’t wear glasses.

RELATED WORK

Physiological measures- As the driver becomes drowsy,their eyes start to blink rapidly,head sways, they change lanes quickly and often, but the above vehicle based and vision based data become apparent only when the driver is sleepy,which is sometimes too late for the prevention of the accident.However, physiological signals start to change in earlier stages of drowsiness.Many researchers have considered the following physiological signals to detect drowsiness: electrocardiogram (ECG), electromyogram (EMG), electroencephalogram (EEG) and electro-oculogram (EoG).Researchers have investigated horizontal eye movement by placing a disposable Ag-Cl electrode on the outer corner of each eye and a third electrode at the center of the forehead for reference. The electrodes were placed as specified so that the parameters - Rapid eye movements (REM) and Slow Eye Movements (SEM) which occur when a subject is awake and drowsy respectively, can be detected easily.The heart rate (HR) also varies significantly between the different stages of drowsiness, such as alertness and fatigue.

Therefore, heart rate, which can be easily determined by the ECG signal, can also be used to detect drowsiness. Others have measured drowsiness using Heart Rate Variability (HRV), in which the low (LF) and high (HF) frequencies fall in the range of 0.04–0.15 Hz and 0.14–0.4 Hz, respectively. HRV is a measure of the beat-to-beat (R-R Intervals) changes in the heart rate. The ratio of LF to HF in the ECG decreases progressively as the driver progresses from an awake to a drowsy state.

Soft Computing Technique to be used for the project

The proposed system aims to classify each frame in the videos of the dataset based on feature extraction and learning with the help of different **Convolutional Neural Networks (CNN)** models. CNN, is a deep learning algorithm, widely used in the image processing scenarios. The algorithm learns different weights and biases corresponding to various objects present in an input image, thereby differentiating images from one another. The algorithm takes an input image in the form of a matrix of pixel values, more precisely using its height, width, number of channels. The network comprises a number of layers, the first of which is convolutional layer, where input image is fed. The convolutional layer comprises a set of convolutional kernels, covering a small region of the image, which are convolved over the entire image to learn various features present in it.This layer produces the feature maps as a result of calculating the scalar product between the kernels and local regions of image.

CNN comprises a number of such convolutional layers along with activation and pooling layers. The activation layer contains an activation function, also known as decision function, which is responsible for analyzing nonlinear properties of an image. The pooling layers are responsible for image downsampling (i.e., reducing the volume of the image).In this layer, the 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. The nonlinear layer follows each convolutional layer and the pooling layer follows each activation layer.CNN also contains a fully connected (FC) Layer, which follows a series of convolutional, nonlinear and pooling layers. The FC layer is attached at the end of the network and is responsible for producing an

N-dimensional vector corresponding to N number of output classes.This layer used to produce class scores for detection of driver drowsiness using deep learning based on Eye State.

Dataset for this Project

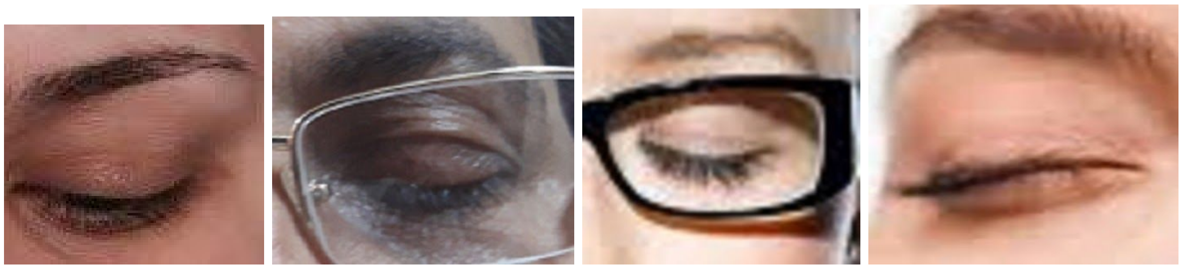
The dataset consists of around 1512 eye images of people taken in different scenarios and trained through the CNN model designed below. Many different types of images were obtained and for testing and training in the applied CNN model of the project. Images with different lighting and daytime are taken to improve the recognition capacity of the eye through the CNN model in. People in glasses were also asked to participate to make the CNN model familiar with the different accessories that could possibly be used in daily life by a person. Eye makeup and other variety of images are also included in the dataset to get the best training for the CNN model in real-life working situations.

Images of people with different skin colours are also included to create variety in the images of people that might be using the project in future. This creates a diversity in the dataset that might be encountered by the CNN model in the future using a drowsiness detection system for drivers of different races and skin tones.

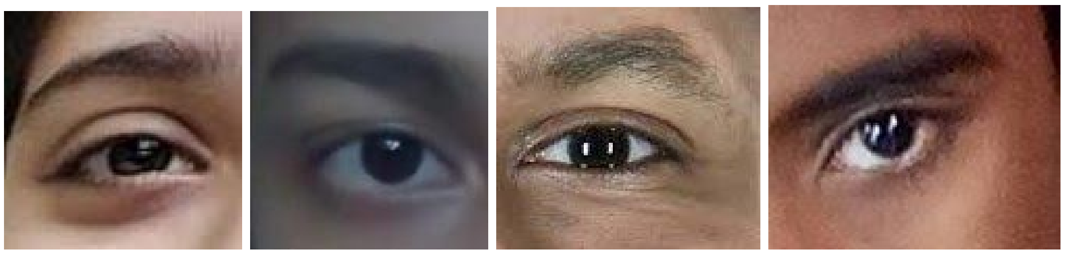
All the different images that are available are trained through the CNN model for maximum variation in the training and testing dataset of the project

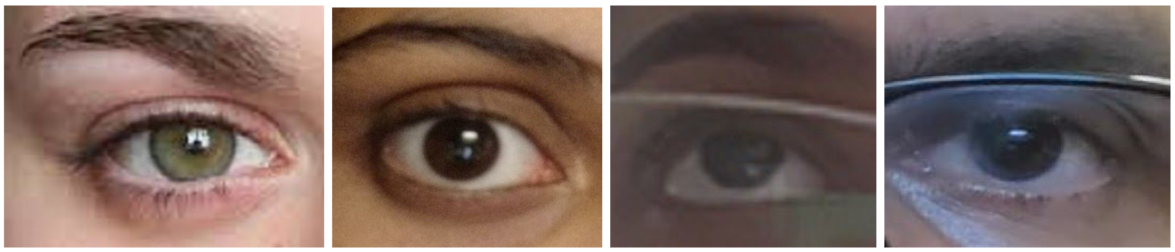
SAMPLE IMAGES

DROWSY-



NOT DROWSY-





The three classes were explained to the participants as follows:

1) **Alert :** One of the first three states highlighted in the KSS table in Table 1. Subjects were told that being alert meant they were completely conscious so they could easily drive for long hours.

2) **Low Vigilant :** As stated in level 6 and 7 of Table 1, this state corresponds to subtle cases when some signs of sleepiness appear, or sleepiness is present but no effort to keep alert is required. While subjects could possibly drive in this state, driving would be discouraged.

3) **Drowsy :** This state means that the subject needs to actively try to not fall asleep (level 8 and 9 in Table 1).

Link to Dataset :

https://drive.google.com/drive/folders/1RTnE9R5OKVWeA_EglRkkE1Eag1H6AX7Q?usp=sharing

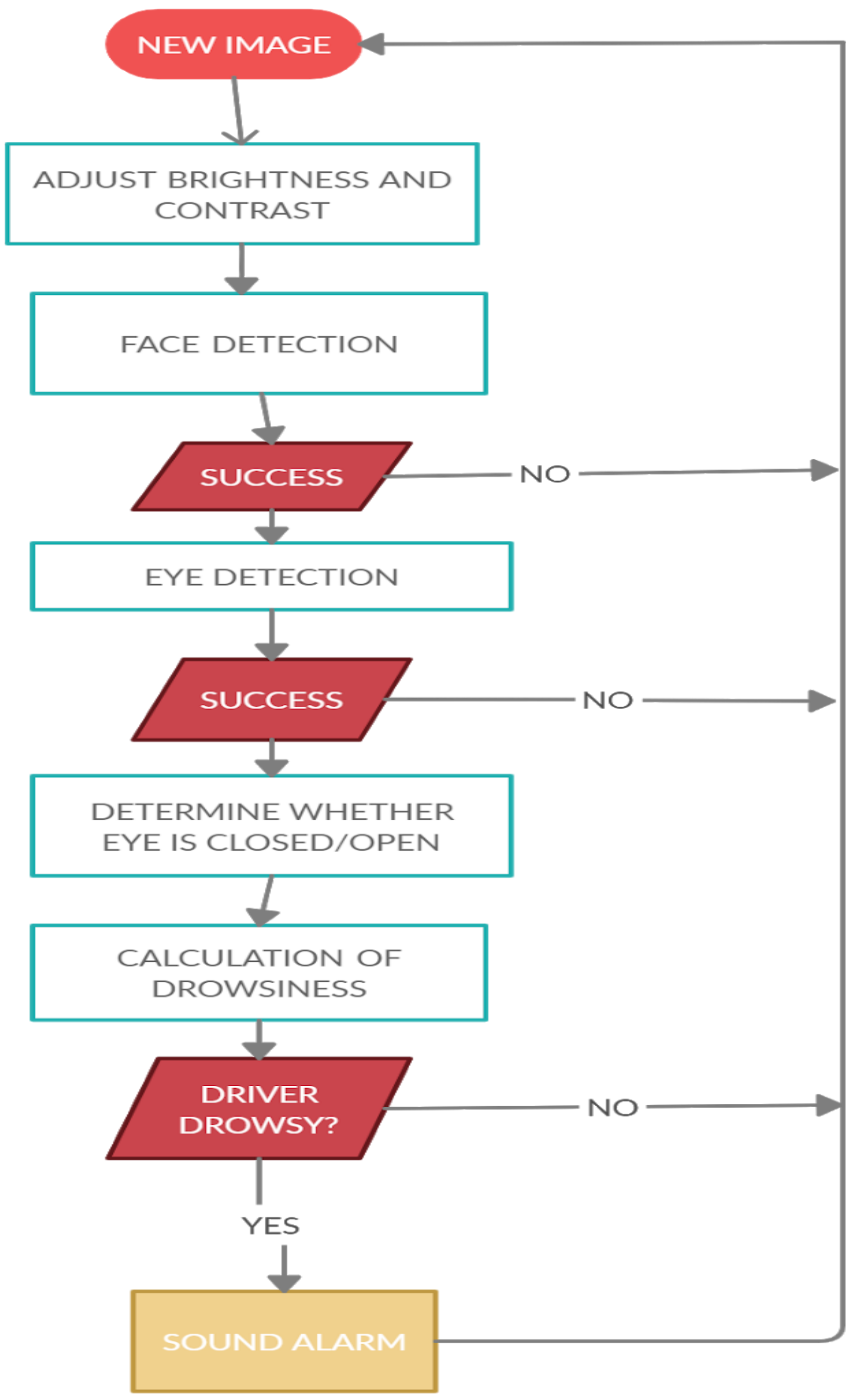
Language used

- ❏ Python
- ❏ OpenCV
- ❏ Keras

Research Paper

- ❏ Pdf file link : **Reseach_paper.pdf**

Design/Architecture


NEW IMAGE
ADJUST BRIGHTNESS AND CONTRAST
FACE DETECTION
SUCCESS
NO
EYE DETECTION
SUCCESS
NO
DETERMINE WHETHER EYE IS CLOSED/OPEN
CALCULATION OF DROWSINESS
DRIVER DROWSY?
NO
YES
SOUND ALARM

Methodology-

Dataset Preparation :

The very first struggle that we faced is to prepare the dataset for our project Driver Drowsiness Detection . For this Project we have taken an image Dataset . There were a total 1512 both closed and open eye images . First we divided our dataset into two parts training dataset and validation dataset. In which we have 1270 images in the training dataset and 242 images in validation dataset . Then we divided these images in two folders Drowsy and Not_Drowsy respectively.

These images were of different shape and sizes and were colored . Hence, first with the help of ImageDataGenerator function we rescaled our images in scale of 1/255 as its rgb value range was from 1 to 255, then we set color_mode to grayscale and target size of 24 X 24 respectively .

With the help of class_indices function we gave labels to these images as 0 and 1 on the bases of their folder name . Hence,

Drowsy	--------------------> 0
Not_Drowsy	--------------------> 1

Training The dataset => CNN model

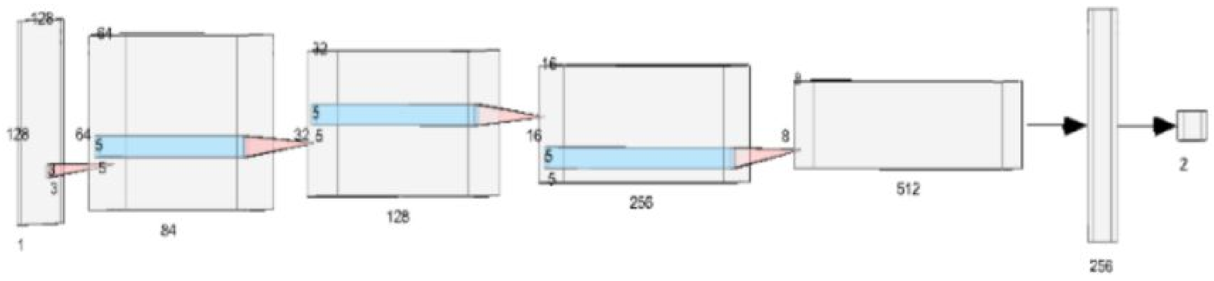
We have made our CNN model in python with the help of libraries tensorflow, keras and numpy.A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden 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 : 16 nodes, kernel size = 3
- ❖ Convolutional Layer : 32 nodes, kernel size = 3
- ❖ Convolutional Layer : 64 nodes, kernel size = 3
- ❖ Fully Connected Layer : 128 nodes
- ❖ Fully Connected Layer : 512 nodes

The final layer is also a fully connected layer with 1 node. In all the layers, a Relu activation function is used except the output layer in which we used Softmax.

After making our model we need to compile the model with model.compile taking loss as ‘binary_crossentropy’ and then we use model.fit with a number of epochs 10 to calculate the accuracy of our model.



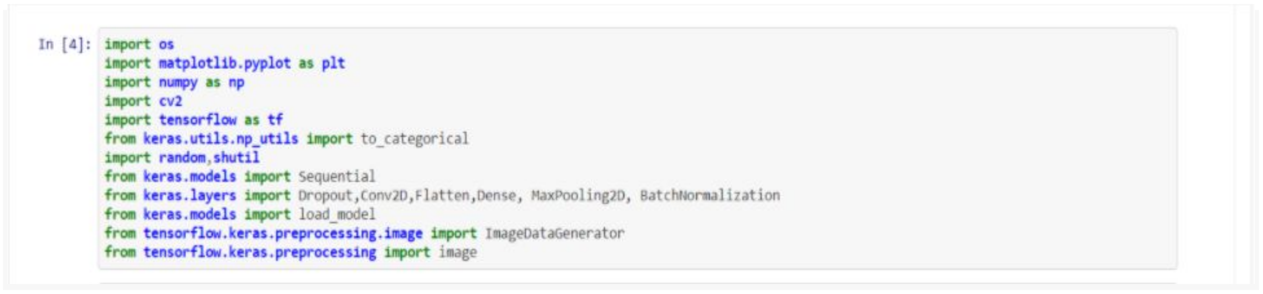
OpenCV Implementation :-

For the real time detection of the features we need to use OpenCV . For detecting face, left eye, right eye we have used haarcascade_frontalface, haarcascade_lefteye_2splits, haarcascade_righteye_2splits respectively . We set up a while loop to capture frames from the real time webcam input and converted the image in grayscale. Then we predict the left eye and right eye is opened or closed by passing the extracted image into our model . If both eyes are open we set up the score as 0 and predicted eyes are open otherwise if both eyes are closed the score will increase and play the alarm if score > 15 .

Programme Code

Importing Libraries

```
import os
import matplotlib.pyplot as plt
import numpy as np
import cv2
import tensorflow as tf
from keras.utils.np_utils import to_categorical
import random,shutil
from keras.models import Sequential
from keras.layers import Dropout,Conv2D,Flatten,Dense, MaxPooling2D, BatchNormalization
from keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
```



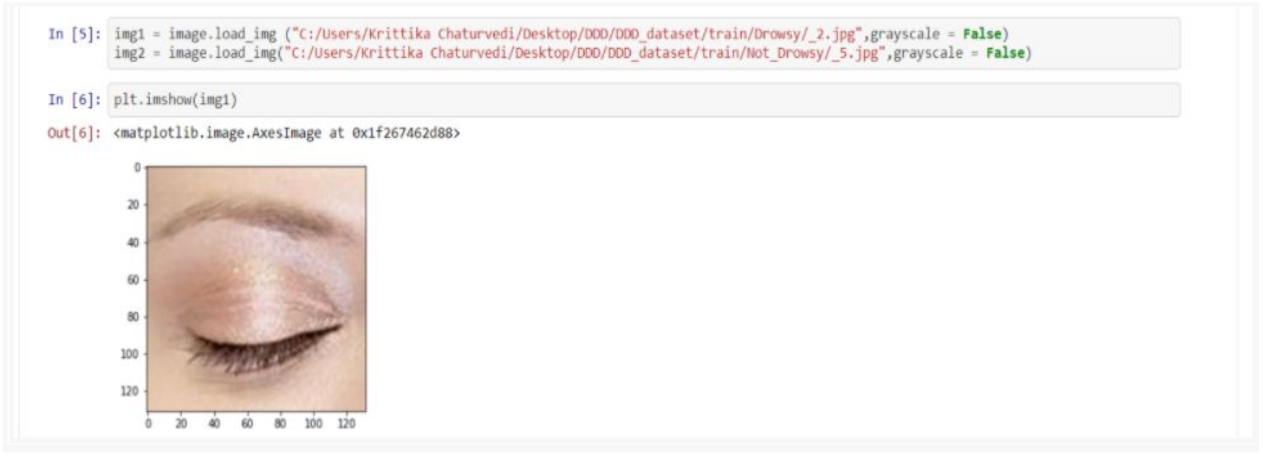
```
In [4]: import os
        import matplotlib.pyplot as plt
        import numpy as np
        import cv2
        import tensorflow as tf
        from keras.utils.np_utils import to_categorical
        import random,shutil
        from keras.models import Sequential
        from keras.layers import Dropout,Conv2D,Flatten,Dense, MaxPooling2D, BatchNormalization
        from keras.models import load_model
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        from tensorflow.keras.preprocessing import image
```

Loading an Image :

img1=image.load_img("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Drowsy/_2.jpg" ,grayscale = **False**)

img2=image.load_img("C:/Users/KrittikaChaturvedi/Desktop/DDD/DDD_dataset/train/Not_Drowsy/_5.jpg",grayscale = **False**)

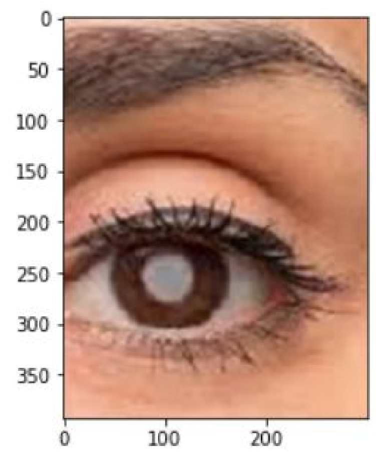
plt.imshow(img1)



```
In [5]: img1 = image.load_img ("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Drowsy/_2.jpg",grayscale = False)
        img2 = image.load_img("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Not_Drowsy/_5.jpg",grayscale = False)

In [6]: plt.imshow(img1)
Out[6]: <matplotlib.image.AxesImage at 0x1f267462d88>
```

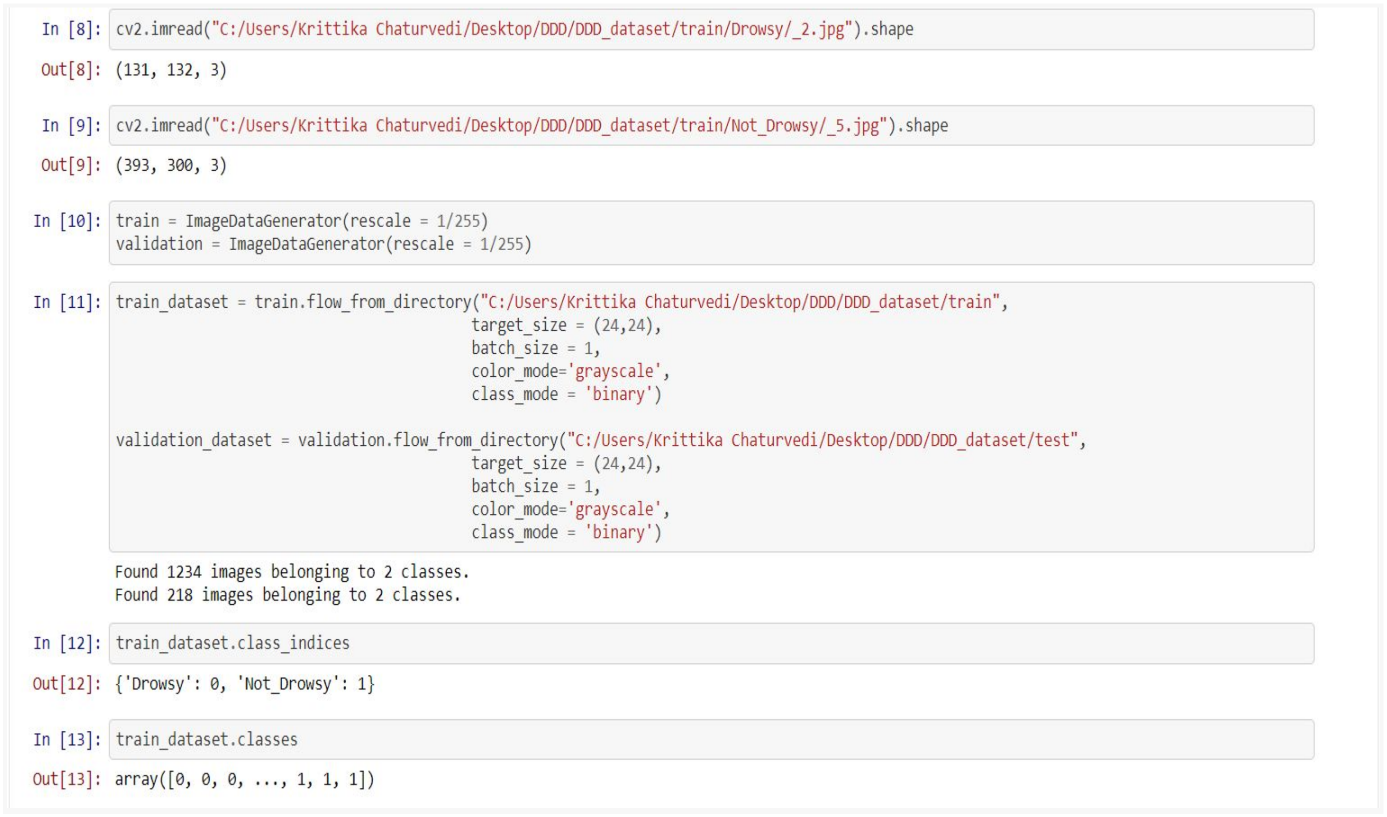
plt.imshow(img2)

```
Out[7]: <matplotlib.image.AxesImage at 0x1f2677a9808>
```



```
cv2.imread("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Drowsy/_2.jpg").shape
Out[8]: (131, 132, 3)
cv2.imread("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Not_Drowsy/_5.jpg").shape
Out[9]: (393, 300, 3)


train = ImageDataGenerator(rescale = 1/255)
validation = ImageDataGenerator(rescale = 1/255)

train_dataset=train.flow_from_directory("C:/Users/KrittikaChaturvedi/Desktop/DDD/DDD_dataset/train"
,target_size = (24,24), batch_size = 1,color_mode='grayscale',class_mode = 'binary')


validation_dataset =validation.flow_from_directory("C:/Users/Krittika
Chaturvedi/Desktop/DDD/DDD_dataset/test", target_size = (24,24),batch_size = 1,
                                        color_mode='grayscale',
                                         class_mode = 'binary')




train_dataset.class_indices
Out[12]: {'Drowsy': 0, 'Not_Drowsy': 1}

train_dataset.classes
```

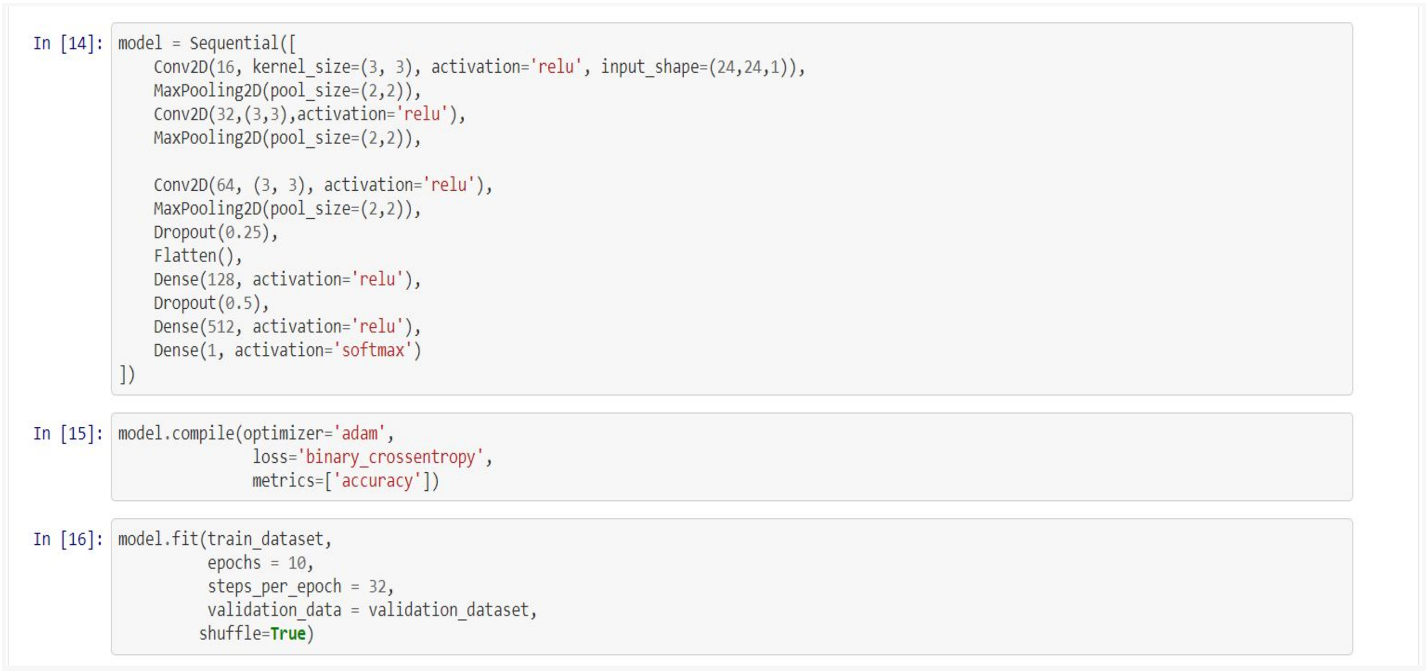
Out[13]: array([0, 0, 0, ..., 1, 1, 1])



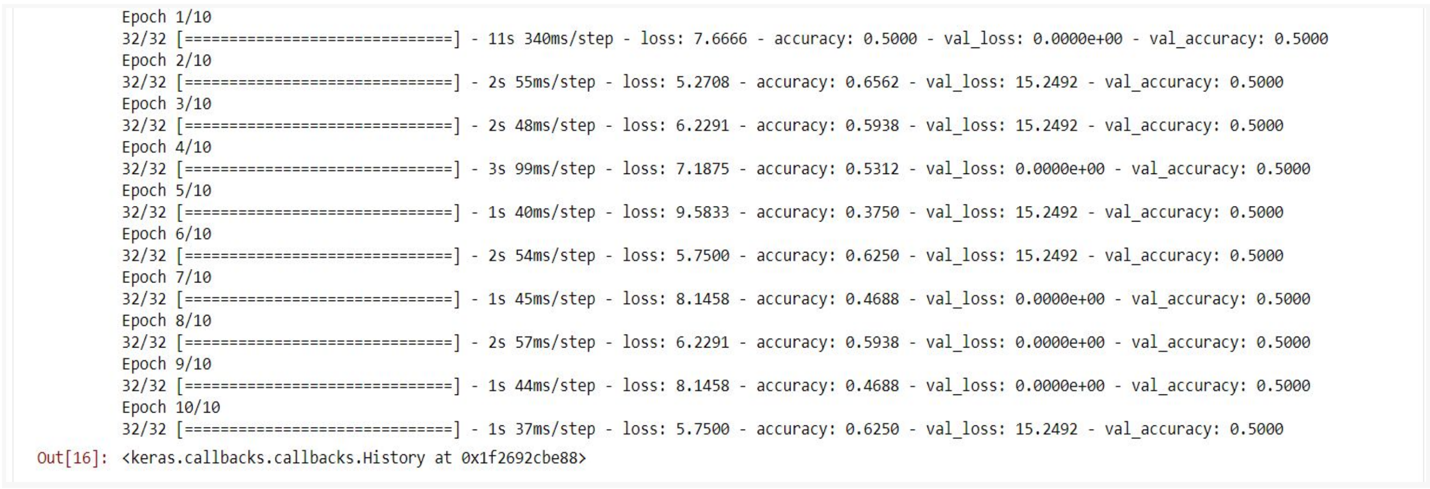
```
In [8]:  cv2.imread("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Drowsy/_2.jpg").shape
Out[8]:  (131, 132, 3)

In [9]:  cv2.imread("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train/Not_Drowsy/_5.jpg").shape
Out[9]:  (393, 300, 3)

In [10]: train = ImageDataGenerator(rescale = 1/255)
         validation = ImageDataGenerator(rescale = 1/255)

In [11]: train_dataset = train.flow_from_directory("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/train",
                                                   target_size = (24,24),
                                                   batch_size = 1,
                                                   color_mode='grayscale',
                                                   class_mode = 'binary')

         validation_dataset = validation.flow_from_directory("C:/Users/Krittika Chaturvedi/Desktop/DDD/DDD_dataset/test",
                                                   target_size = (24,24),
                                                   batch_size = 1,
                                                   color_mode='grayscale',
                                                   class_mode = 'binary')

         Found 1234 images belonging to 2 classes.
         Found 218 images belonging to 2 classes.

In [12]: train_dataset.class_indices
Out[12]: {'Drowsy': 0, 'Not_Drowsy': 1}

In [13]: train_dataset.classes
Out[13]: array([0, 0, 0, ..., 1, 1, 1])
```

Training Dataset through CNN Model -

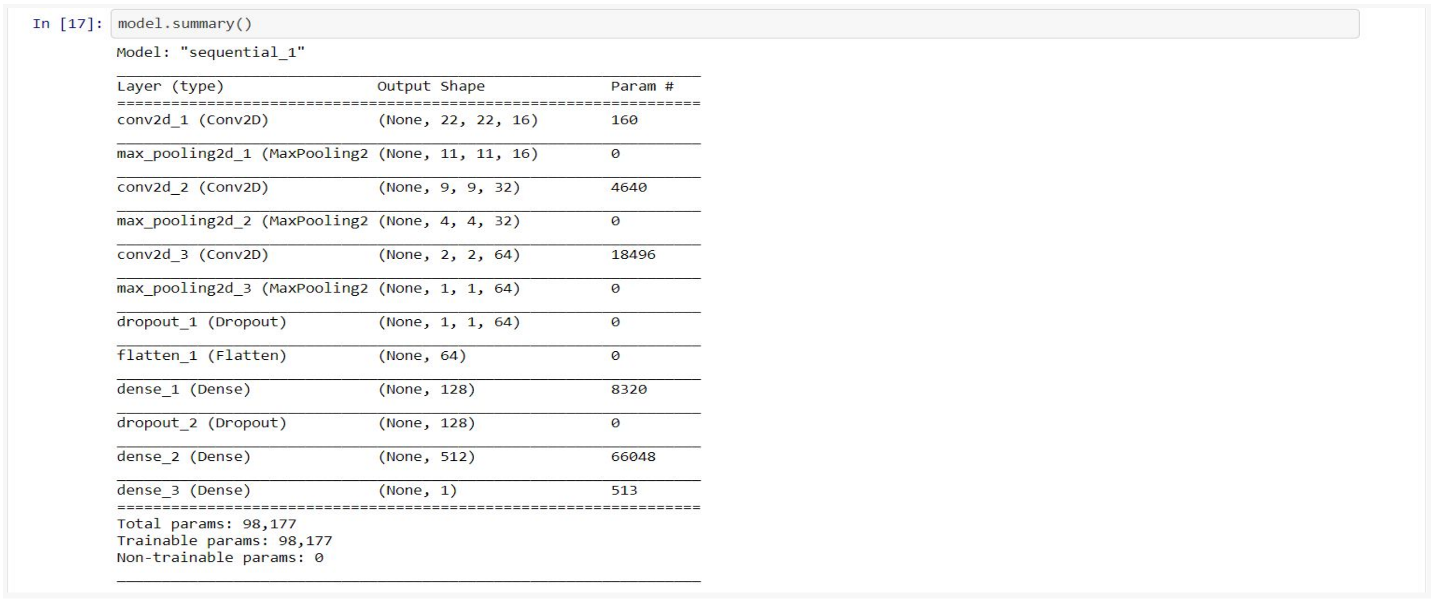
```
model = Sequential([
  Conv2D(16, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
  MaxPooling2D(pool_size=(2,2)),
  Conv2D(32,(3,3),activation='relu'),
  MaxPooling2D(pool_size=(2,2)),

  Conv2D(64, (3, 3), activation='relu'),
  MaxPooling2D(pool_size=(2,2)),
  Dropout(0.25),
  Flatten(),
  Dense(128, activation='relu'),
  Dropout(0.5),
  Dense(512, activation='relu'),
  Dense(1, activation='softmax')
])

model.compile(optimizer='adam',
            loss='binary_crossentropy',
            metrics=['accuracy'])
model.fit(train_dataset,
        epochs = 10,
        steps_per_epoch = 32,
        validation_data = validation_dataset,
       shuffle=True)
```



```
In [14]: model = Sequential([
             Conv2D(16, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
             MaxPooling2D(pool_size=(2,2)),
             Conv2D(32,(3,3),activation='relu'),
             MaxPooling2D(pool_size=(2,2)),

             Conv2D(64, (3, 3), activation='relu'),
             MaxPooling2D(pool_size=(2,2)),
             Dropout(0.25),
             Flatten(),
             Dense(128, activation='relu'),
             Dropout(0.5),
             Dense(512, activation='relu'),
             Dense(1, activation='softmax')
         ])

In [15]: model.compile(optimizer='adam',
                       loss='binary_crossentropy',
                       metrics=['accuracy'])

In [16]: model.fit(train_dataset,
                   epochs = 10,
                   steps_per_epoch = 32,
                   validation_data = validation_dataset,
                  shuffle=True)
```



```
         Epoch 1/10
         32/32 [==============================] - 11s 340ms/step - loss: 7.6666 - accuracy: 0.5000 - val_loss: 0.0000e+00 - val_accuracy: 0.5000
         Epoch 2/10
         32/32 [==============================] - 2s 55ms/step - loss: 5.2708 - accuracy: 0.6562 - val_loss: 15.2492 - val_accuracy: 0.5000
         Epoch 3/10
         32/32 [==============================] - 2s 48ms/step - loss: 6.2291 - accuracy: 0.5938 - val_loss: 15.2492 - val_accuracy: 0.5000
         Epoch 4/10
         32/32 [==============================] - 3s 99ms/step - loss: 7.1875 - accuracy: 0.5312 - val_loss: 0.0000e+00 - val_accuracy: 0.5000
         Epoch 5/10
         32/32 [==============================] - 1s 40ms/step - loss: 9.5833 - accuracy: 0.3750 - val_loss: 15.2492 - val_accuracy: 0.5000
         Epoch 6/10
         32/32 [==============================] - 2s 54ms/step - loss: 5.7500 - accuracy: 0.6250 - val_loss: 15.2492 - val_accuracy: 0.5000
         Epoch 7/10
         32/32 [==============================] - 1s 45ms/step - loss: 8.1458 - accuracy: 0.4688 - val_loss: 0.0000e+00 - val_accuracy: 0.5000
         Epoch 8/10
         32/32 [==============================] - 2s 57ms/step - loss: 6.2291 - accuracy: 0.5938 - val_loss: 0.0000e+00 - val_accuracy: 0.5000
         Epoch 9/10
         32/32 [==============================] - 1s 44ms/step - loss: 8.1458 - accuracy: 0.4688 - val_loss: 0.0000e+00 - val_accuracy: 0.5000
         Epoch 10/10
         32/32 [==============================] - 1s 37ms/step - loss: 5.7500 - accuracy: 0.6250 - val_loss: 15.2492 - val_accuracy: 0.5000
Out[16]: <keras.callbacks.callbacks.History at 0x1f2692cbe88>
```



```
In [17]: model.summary()
         Model: "sequential_1"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #   
         =================================================================
         conv2d_1 (Conv2D)            (None, 22, 22, 16)        160       
         _________________________________________________________________
         max_pooling2d_1 (MaxPooling2 (None, 11, 11, 16)        0         
         _________________________________________________________________
         conv2d_2 (Conv2D)            (None, 9, 9, 32)          4640      
         _________________________________________________________________
         max_pooling2d_2 (MaxPooling2 (None, 4, 4, 32)          0         
         _________________________________________________________________
         conv2d_3 (Conv2D)            (None, 2, 2, 64)          18496     
         _________________________________________________________________
         max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0         
         _________________________________________________________________
         dropout_1 (Dropout)          (None, 1, 1, 64)          0         
         _________________________________________________________________
         flatten_1 (Flatten)          (None, 64)                0         
         _________________________________________________________________
         dense_1 (Dense)              (None, 128)               8320      
         _________________________________________________________________
         dropout_2 (Dropout)          (None, 128)               0         
         _________________________________________________________________
         dense_2 (Dense)              (None, 512)               66048     
         _________________________________________________________________
         dense_3 (Dense)              (None, 1)                 513       
         =================================================================
         Total params: 98,177
         Trainable params: 98,177
         Non-trainable params: 0
         _________________________________________________________________
```

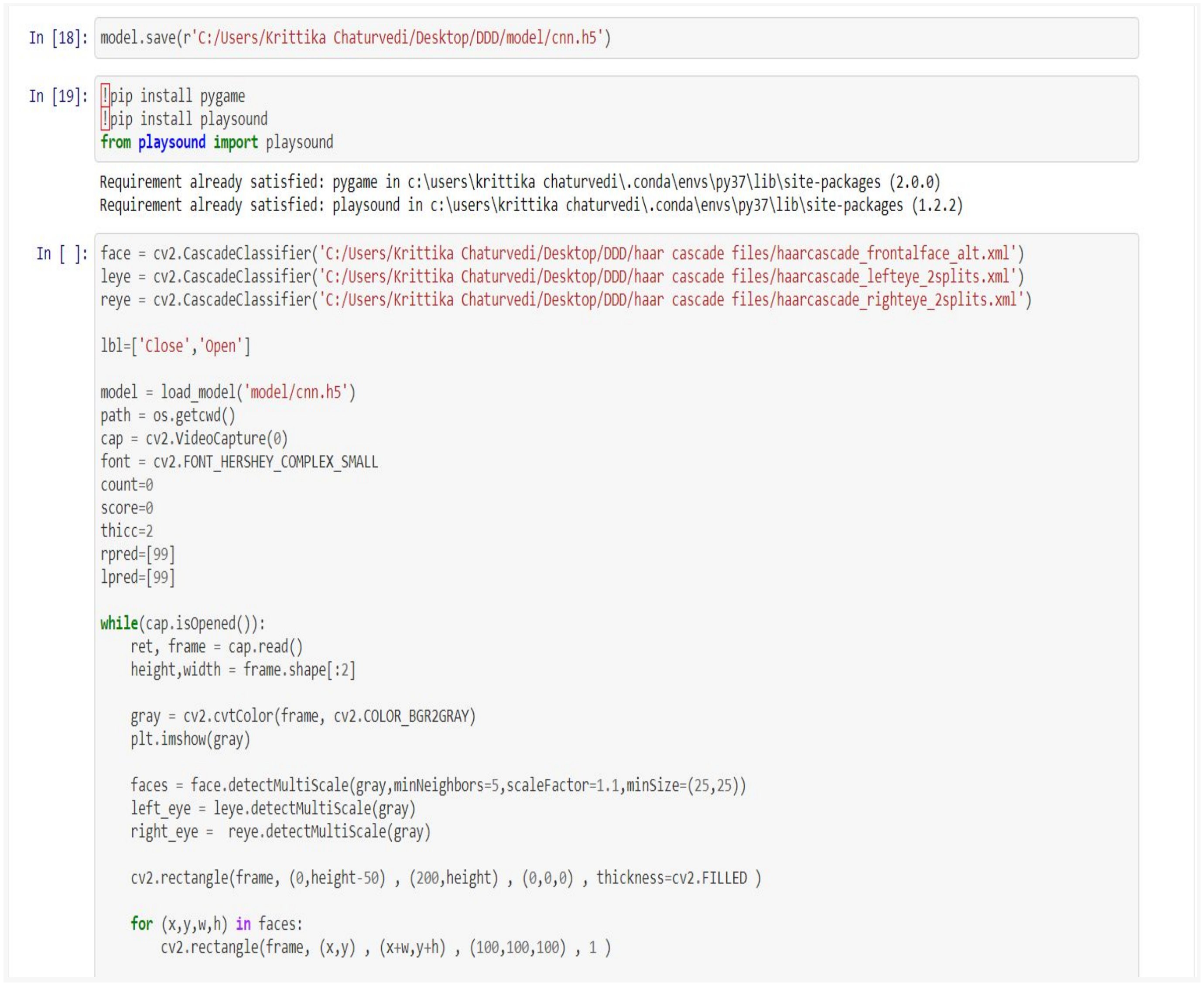
```
model.save(r'C:/Users/Krittika Chaturvedi/Desktop/DDD/model/cnn.h5')
```

OpenCV CODE-

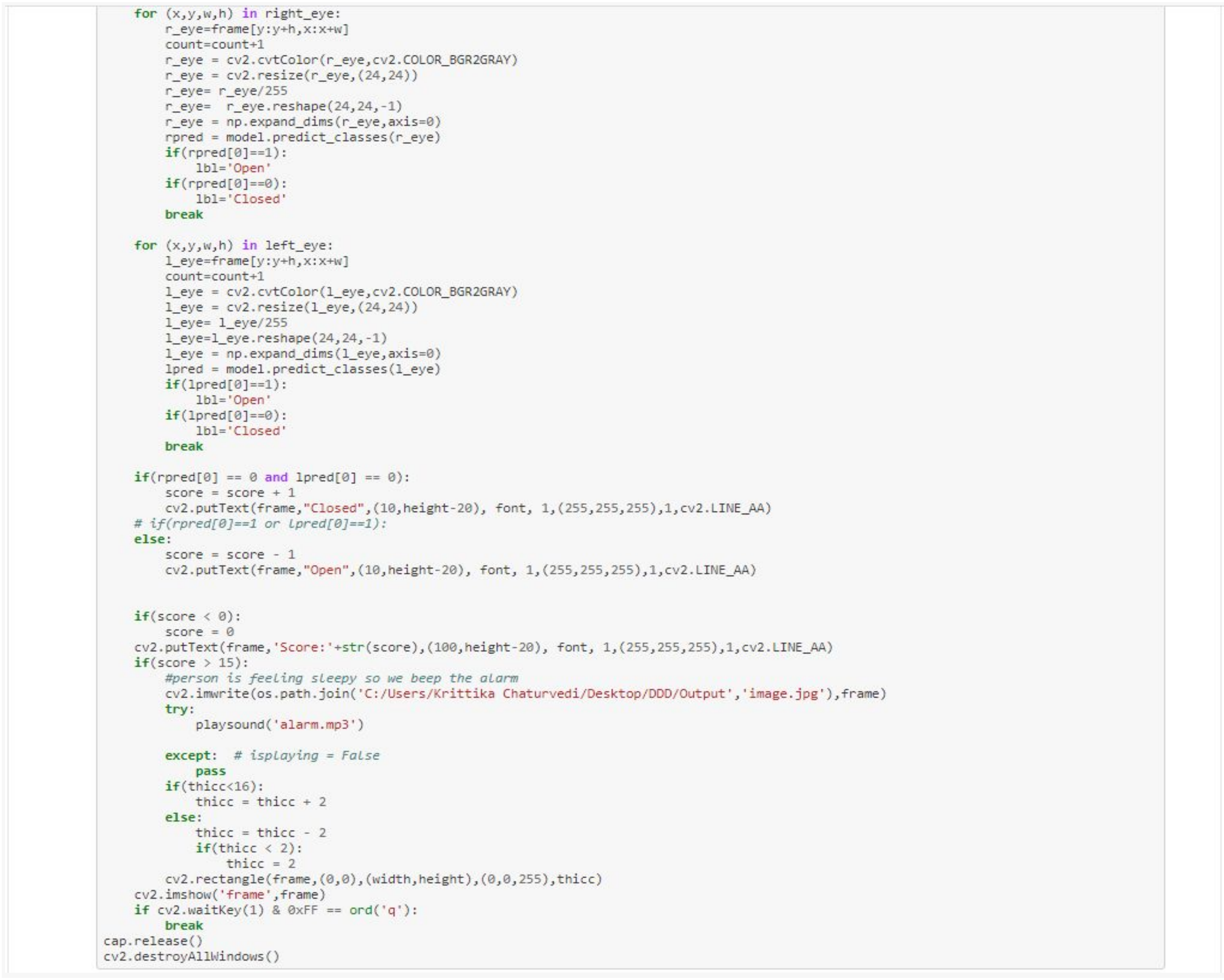
```
from playsound import playsound

face = cv2.CascadeClassifier('C:/Users/Krittika Chaturvedi/Desktop/DDD/haar cascade files/haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('C:/Users/Krittika Chaturvedi/Desktop/DDD/haar cascade files/haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('C:/Users/Krittika Chaturvedi/Desktop/DDD/haar cascade files/haarcascade_righteye_2splits.xml')

lbl=['Close','Open']

model = load_model('model/cnn.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
```

```
rpred=[99]
lpred=[99]

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

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

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

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

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

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

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = model.predict_classes(l_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('C:/Users/Krittika Chaturvedi/Desktop/DDD/Output','image.jpg'),frame)
        try:
            playsound('alarm.mp3')
                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()
```



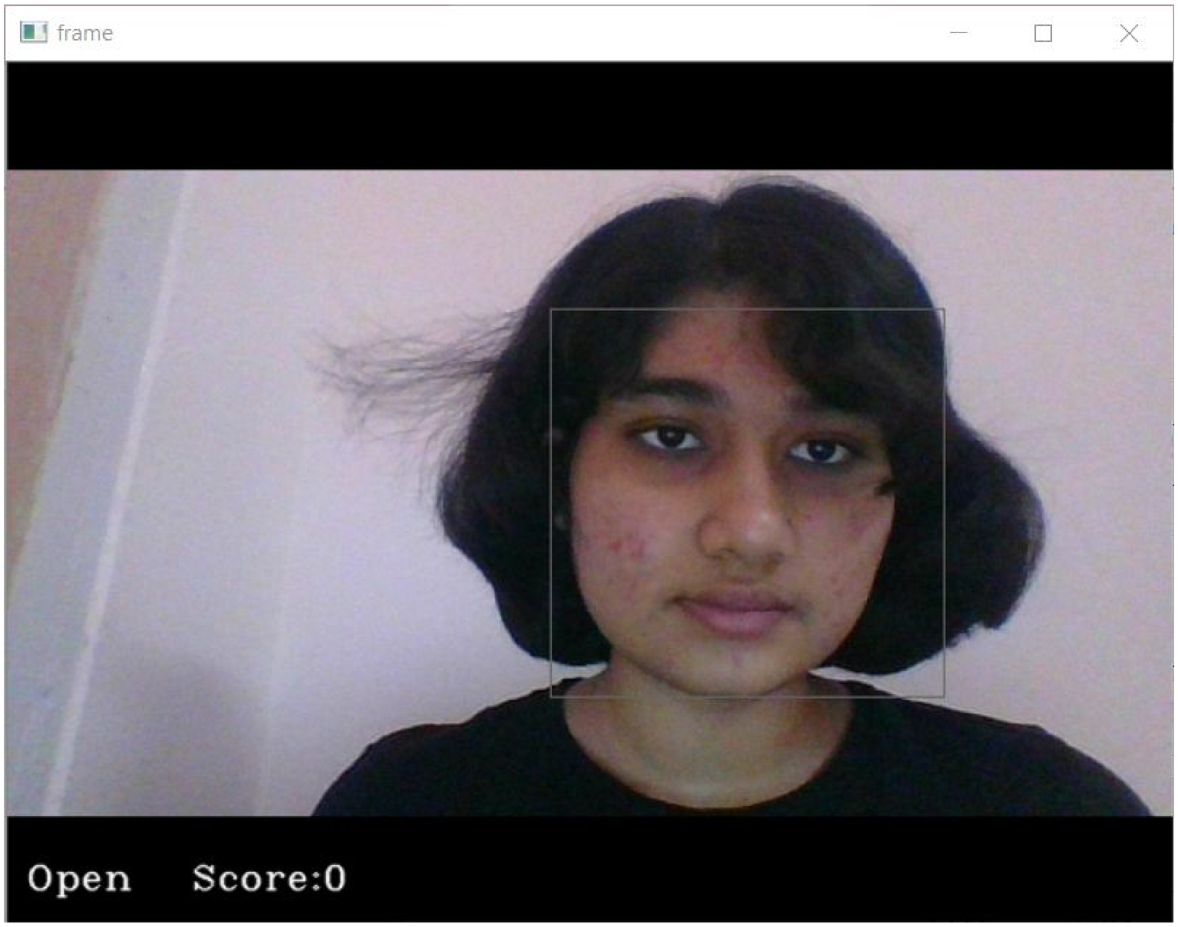
```
In [18]: model.save(r'C:/Users/Krittika Chaturvedi/Desktop/DDD/model/cnn.h5')

In [19]: !pip install pygame
         !pip install playsound
         from playsound import playsound

         Requirement already satisfied: pygame in c:\users\krittika chaturvedi\.conda\envs\py37\lib\site-packages (2.0.0)
         Requirement already satisfied: playsound in c:\users\krittika chaturvedi\.conda\envs\py37\lib\site-packages (1.2.2)

In [ ]: face = cv2.CascadeClassifier('C:/Users/Krittika Chaturvedi/Desktop/DDD/haar cascade files/haarcascade_frontalface_alt.xml')
        leye = cv2.CascadeClassifier('C:/Users/Krittika Chaturvedi/Desktop/DDD/haar cascade files/haarcascade_lefteye_2splits.xml')
        reye = cv2.CascadeClassifier('C:/Users/Krittika Chaturvedi/Desktop/DDD/haar cascade files/haarcascade_righteye_2splits.xml')

        lbl=['Close','Open']

        model = load_model('model/cnn.h5')
        path = os.getcwd()
        cap = cv2.VideoCapture(0)
        font = cv2.FONT_HERSHEY_COMPLEX_SMALL
        count=0
        score=0
        thicc=2
        rpred=[99]
        lpred=[99]

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

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

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

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

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



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

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = model.predict_classes(l_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('C:/Users/Krittika Chaturvedi/Desktop/DDD/Output','image.jpg'),frame)
        try:
            playsound('alarm.mp3')

        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()
```

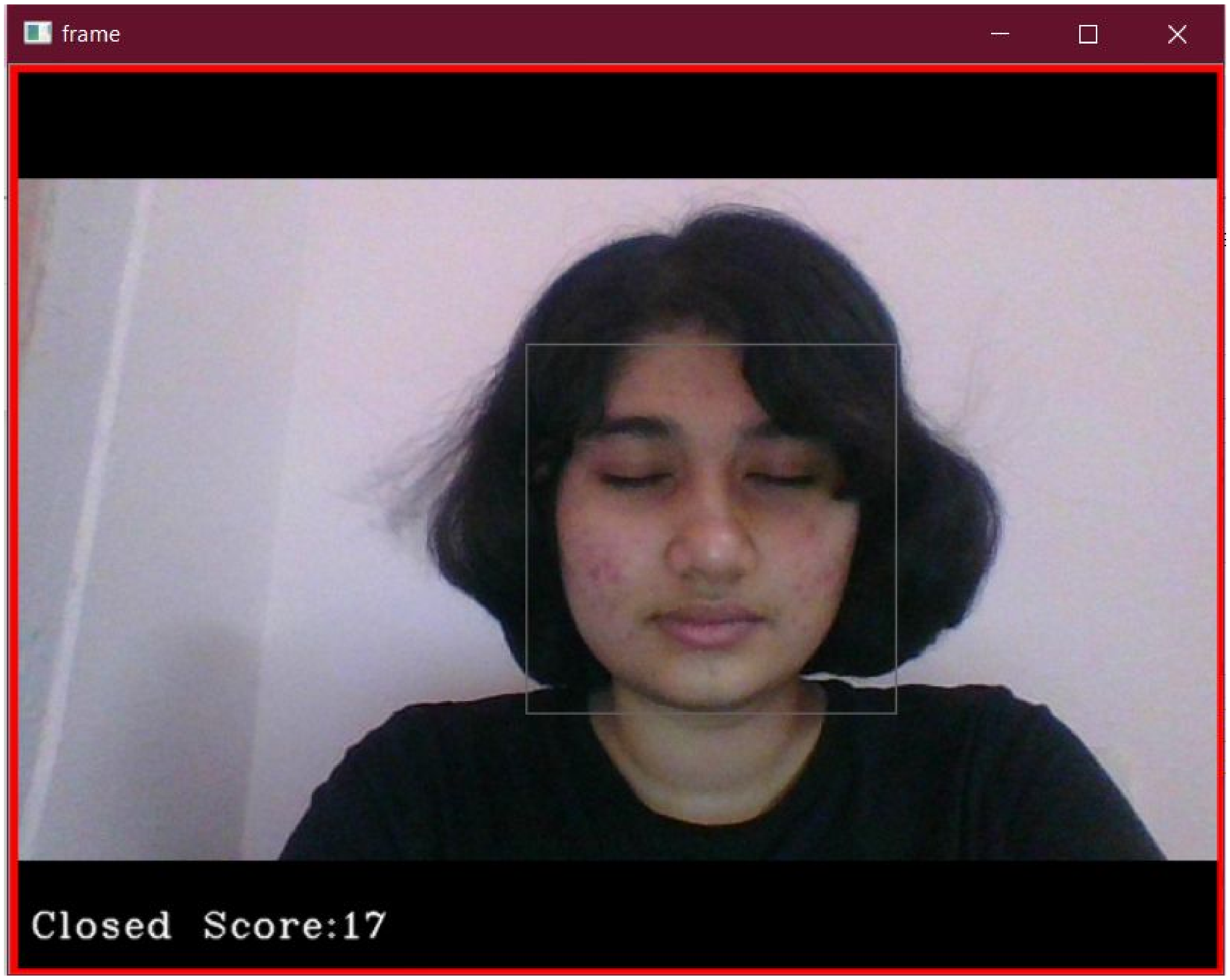
Results

Open Eye Detection



When the score is 0 the eyes are predicted as open . The score is zero when both left eye and right eye is predicted as open.We extracted the right eye and left eye with the help of openCV haar cascade classifiers .

Closed Eye Detection (Will play Alarm)



OpenCV is counting the frame rate , we set a score of 15 . When the score is greater than 15 for consecutive images(score increases if CNN models predict both eyes are closed) the system predicts that the person is drowsy and plays an alarm and the screen appears red .

Conclusion-

We studied various research papers on the detection system for the driver's drowsiness and found that the techniques/ models they were using like PECLOS ,LSTM had very good accuracy of 96 to 98% of detection of eye. We also found that some of these models, when encountered with the rapid eye blinking and people having a habit of blinking eye very frequently, the alarm was sounded. So in our model we tried to create a score system which does not sound the alarm until the score variable crosses a certain value mark,when the eyes are continuously closed. This helped us to create a system that was efficient in detecting the driver’s drowsiness detection system.