UROP

SRM University AP

Department of CSE

UROP project report on

Drowsiness Detection System

by

Manish Guduri -AP19110010521

HemaSundar Eddala – AP19110010481

Aadarsh Reddy E – AP19110010471

Ruthik Venkat P – AP19110010549

Under the guidance of

Dr Radha Guha Mam

Table of contents

Abstract

An accident is an incident that occurs suddenly, unexpectedly, and inadvertently under unforeseen circumstances. Accidents are increasing day to day, many people are getting injured seriously and some people are even dying. Almost one-quarter of serious accidents are happening due to the drivers handling the vehicle in a state of drowsiness. Sometimes people have long journeys without halts. In this case, there are high chances of becoming sleepy. A solution that can be developed to detect and notify drivers is done by making use of technologies of computer vision and machine learning that make computers more intelligent by developing techniques that help them to gather high-level understanding from digital images or videos. The main idea behind this project is to develop a computer vision system, using OpenCV, which is no-intrusive that can automatically detect driver drowsiness in a real-time video stream and will alert the driver by playing an alarm if they appear to be drowsy. This system would make use of an algorithm that detects the eye landmark points of the driver and based on this can determine if the driver is in a drowsy state or not, and appropriately sound an alarm.

Introduction

In today's world, almost everyone is using a vehicle and with an increase in the number of vehicles, road accidents are increasing. The use of transportation requires lots of safety systems and maintenance. Currently, we are using vehicles with a safety system that includes the use of brakes, airbags, seatbelts, etc. But these are only helpful post-accident and lack to alert the driver before the accident occurs. Many accidents are occurring due to driver drowsiness. Every year accidents caused due to human errors result in increasing amounts of deaths and injuries on a global scale. Driver drowsiness is acknowledged as an important factor in road accidents. So, to overcome this there is a constant demand for a system to detect and alert the driver in case of drowsiness. Drowsiness occurring due to driving for a long time is one of the most common and dangerous issues related to road safety. According to a study by the Central Road Research Institute, about 40% of road accidents occur due to drivers who are exhausted and falling asleep behind the wheel. Driver drowsiness detection system can be used for preventing the above problems without endangering human life and hence is one of the effective ways to stop accidents.

In this project, we are going to implement code to determine drowsiness with the help of the eyes of the driver and the time interval between eye blinking. The driver's eyes show signs of tiredness. In this project to detect the face and eyes of a person, we use some existing code and prepare a model to detect whether the eyes are closed or open with the help of the MRL data set. The data set consists of closed eyes and open eyes in different situations, like in heavy light, with glasses, etc. When a person's eyelids remain closed for a while, it signals that the motorist is in a sleepy state.

- **Transfer learning**

  The process of using already a trained model on our problem is known as transfer learning. In transfer learning, a machine uses previous assignment knowledge to improve prediction on a new task.

  During transfer learning, the knowledge of an already trained machine learning model is transferred to a different but closely related problem. Like if you have a model for determining whether an image contains a dog or not, you can use that model to determine whether the image contains a person by changing a few layers.

- **MobileNet**

  MobileNet is a model for image categorisation, it is based on CNN architecture model. There are more models available, but Mobilenet is unique in that it uses very minimal computation resources to run so that it is light weight model and also it applies transfer learning.

Literature survey

- Here we aim to determine whether the driver is drowsy or active.
- The main factors to consider whether is a person is drowsy or not are:
  - Eyes
  - Yawning

Taking advantage of these visual characteristics, computer vision is the feasible and appropriate technology to treat this problem. The eyes are the main factor to determine drowsiness. If the person's eyes are closed it means the person is sleeping but when we blink our eyes at that time also it'll be considered as eyes are closed. So, to overcome this issue we have to consider a time limit. If the eyes are closed till that time limit then it sounds an alarm or shows the person is sleeping.

First step is to detect the face of the person as we are focussing on the drowsiness detection we don’t write code for face recognition.

Next step is to detect eyes in the face.

Software requirements

- Programming language
    - Python3
- IDE Used
    - Jupiter notebook
- Libraries
    - Numpy
    - matplotlib
    - Opencv
    - winsound
    - tensorflow
- Operating system
    - Windows or ubuntu

Hardware Requirements

- A system with basic hardware
- Webcam

Methodology

The main concept of Driver Drowsiness Detection is to capture a driver’s eyes from a camera and be able to accurately calculate the level of drowsiness in drivers with real-time processing. OpenCV Version is applied for various features of computer vision. The language used is python.

In this, there are only two classes based on the eyelid closure

i. Sleepy (Eyes closed constantly)

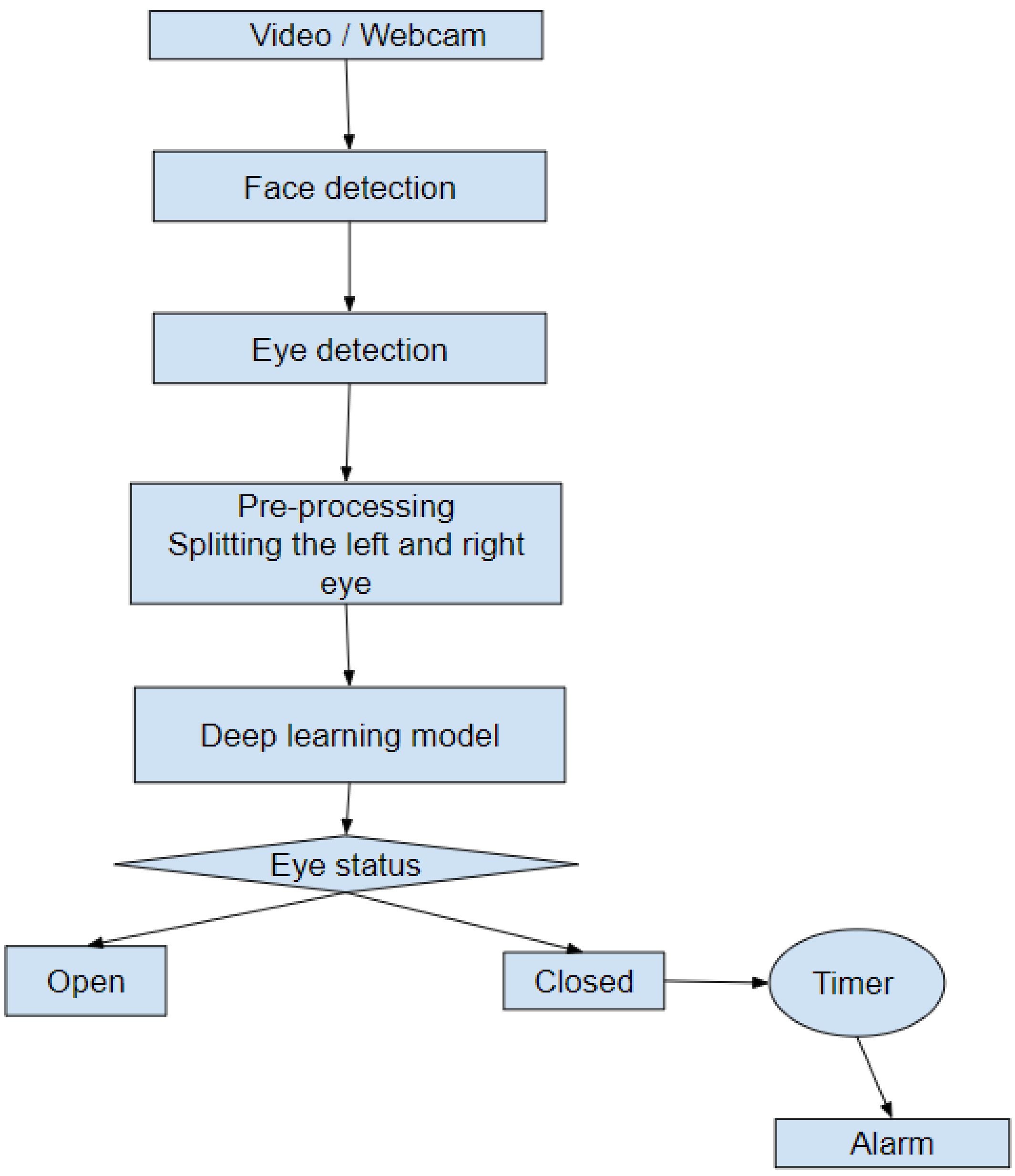
ii. Active (Eyes are open)

Here we train the model using the eye images. But when comes to implementation we use a webcam to capture the face and use some already available codes to find the face and eyes.

We pre-process the model to split the eyes into left or right.

After pre-processing the data we use the method of transfer learning. MobileNet classifier is used to train our model by changing the last layers. It is trained on 1000 classes but here in our case, we have only two classes that is binary classification. Next, it determines whether the eyes are closed or open if the eyes are open then no issue but if the eyes are closed it starts a timer and if the timer exceeds a level it sounds like an alarm.

Flow chart



Code snippets

- To read image

img_array=cv2.imread("s0012_08162_0_0_1_1_0_02.png",cv2.IMREAD_GRAYSCAL
E)

- For resizing the image and converting to grayscale image

```
img_size=224

new_array=cv2.resize(backtorgb, (img_size,img_size))

plt.imshow(new_array, cmap='gray')

plt.show()
```

- **Reading all the images and converting them to an array of labels and data**

```
directory='training'

classes=["closed_eyes","open_eyes"]

training_data=[]

def create_training_Data():

    for c in classes:

        path=os.path.join(directory, c)

        class_num=classes.index(c)

        for img in os.listdir(path):

            try:

                img_array=cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)

                backtorgb=cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)

                new_array=cv2.resize(backtorgb, (img_size,img_size))

                training_data.append([new_array,class_num])

            except Exception as e:

                pass
```

- **Creating the model**

```
model=tf.keras.applications.mobilenet.MobileNet()
base_input=model.layers[0].input
base_output=model.layers[-4].output
Flat_layer=layers.Flatten()(base_output)
final_output=layers.Dense(1)(Flat_layer) ##one node (1/0)
```

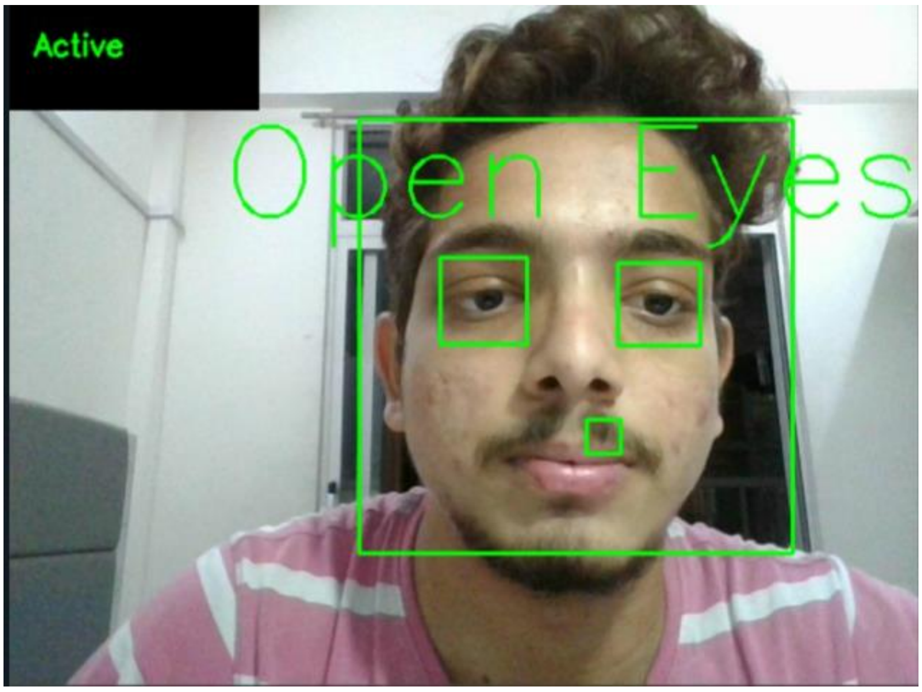
```
final_output=layers.Activation('sigmoid')(final_output)
new_model = keras.Model(inputs=base_input,outputs=final_output)
new_model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
new_model.fit(X,Y, epochs = 1 ,validation_split=0.1)
```

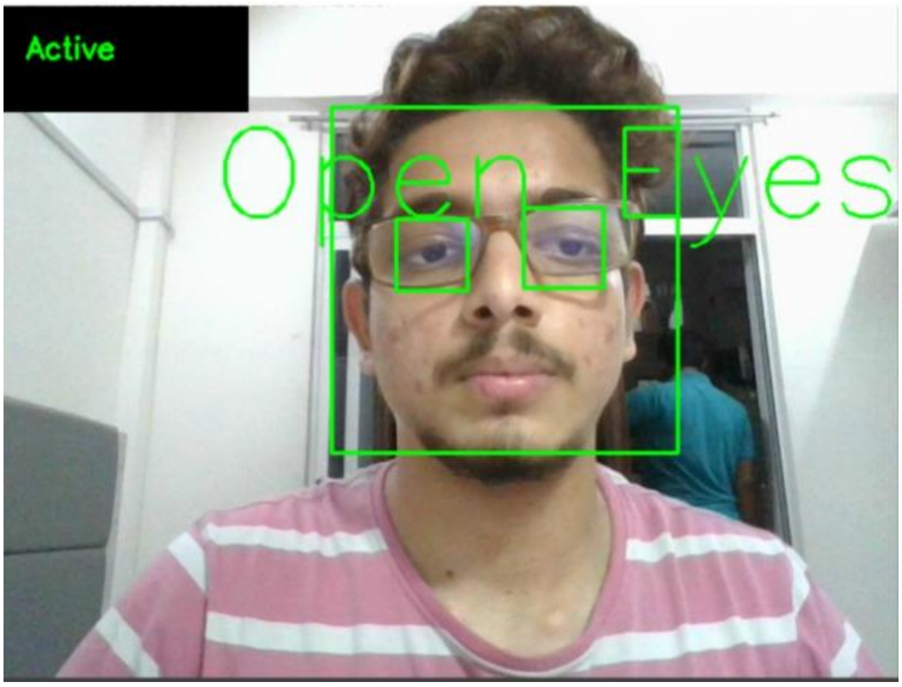
- **For real time detection of face**

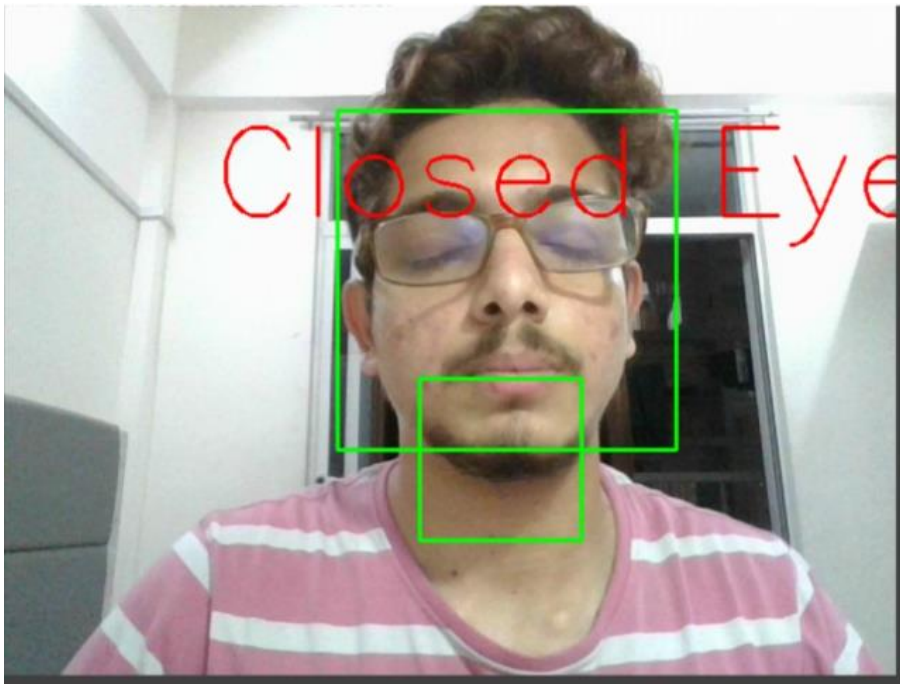
```
import winsound
frequency = 2500
duration = 1000
import numpy as np
import cv2

path = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(1)
#Check if the webcam is opened correctly
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Cannot open webcam")
counter = 0
while True:
    ret,frame = cap.read()
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_eye.xml')
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #print(faceCascade.empty)
    eyes = eye_cascade.detectMultiScale(gray,1.1,4)
    for x,y,w,h in eyes:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        eyess = eye_cascade.detectMultiScale(roi_gray)
        if len(eyess) == 0:
            print("eyes are not dectected")
        else:
            for (ex,ey,ew,eh) in eyess:
                eyes_roi = roi_color[ey: ey+eh, ex:ex + ew]

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    print(faceCascade.empty())
    faces = faceCascade.detectMultiScale(gray,1.1,4)

    # Draw a rectangle around the faces
    for(x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    font = cv2.FONT_HERSHEY_SIMPLEX

    # Use putText() methord for
    # inserting text on Video

    final_image =cv2.resize(eyes_roi, (224,224))
    final_image = np.expand_dims(final_image,axis = 0) ## need Fourth dimension
    final_image =final_image/255.0

    Prediction = new_model.predict(final_image)
    if (Prediction>0.09):
        status = "Open Eyes"
        cv2.putText(frame,
                status,
                (150, 150),
                font, 3,
                (0, 255, 0),
                2,
                    cv2.LINE_4)

            x1,y1,w1,h1 = 0,0,175,75
            # Draw black background rectangle

            cv2.rectangle(frame, (x1, x1), (x1 + w1, y1 + h1), (0,0,0), -1)
            # Add text
            cv2.putText(frame, 'Active', (x1 + int(w1/10),y1 + int(h1/2)),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,0), 2)



        else:
            counter = counter +1
            status = "Closed Eyes"
            cv2.putText(frame,
                    status,
                    (150, 150),
                    font, 3,
                    (0, 0, 255),
                    2,
                    cv2.LINE_4)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255,0), 2)
            if counter>5:

                x1,y1,w1,h1 = 0,0,175,75
                #Draw black background rectangle
                cv2.rectangle(frame, (x1, x1), (x1 + w1, y1 + h1), (0,0,0), -1)
                #Add text
                cv2.putText(frame, 'Sleep Alert !!', (x1 + int(w1/10),y1 + int(h1/2)),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255), 2)
                winsound.Beep(frequency, duration)
                counter = 0
        cv2.imshow('Drowsiness Detection Tutorial',frame)


        if cv2.waitKey(2) & 0xFF == ord('q'):
            break


    cap.release()
    cv2.destroyAllWindows()
```

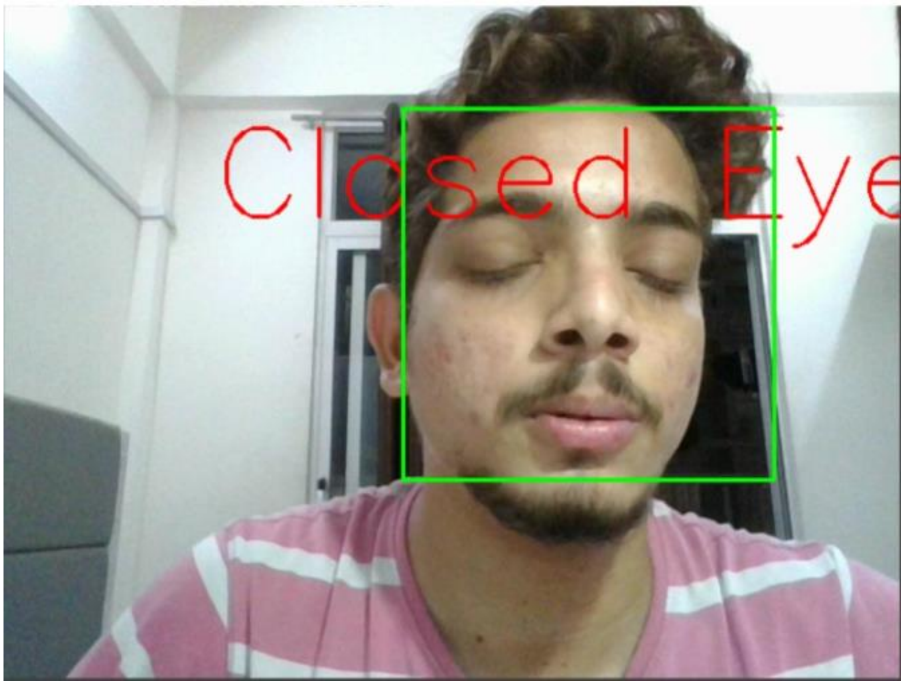
Result

-




Active
Open Eyes


Closed Eye



Conclusion

- For the eye values ~ >.009 it detects as open eye and for values <0.009 it determines as closed eyes.
- Transfer learning gave us good results.

References

- ML | Introduction to Transfer Learning - GeeksforGeeks
- Kaggle Datasets
- Driver drowsiness detection - Wikipedia
- (3) Transfer Learning | Deep Learning Tutorial 27 (Tensorflow, Keras & Python) - YouTube