
# VELLORE INSTITUTE OF TECHNOLOGY

# CHENNAI CAMPUS

# DRIVER DROWSINESS DETECTION SYSTEM

**CSE3009: INTERNET OF THINGS**

**PROJECT REPORT**

**SLOT: E2**

**FACULTY: DR. MANISH KUMAR**


| | |
|---|---|
| **T. DAMIAN LOURDES** | **19BCE1540** |
| **M. SHARATH SRIVATSAN** | **19BCE1688** |
| **NAVIN THOMSY** | **19BCE1695** |

# ACKNOWLEDGEMENT

We have taken a lot of effort into this project. However, completing this project would not have been possible without the support and guidance of a lot of individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to *Prof. Dr. Manish Kumar* for his guidance and supervision. We would like to thank him for providing the necessary information and resources for this project.

We would like to express our gratitude towards our parents for their kind co-operation and encouragement which help us a lot in completing this project.

Last but not the least, we would like to thank everyone who is involved in the project directly or indirectly.

# INDEX

# SYNOPSIS

This document is a review report on research conducted and a project undertaken in the field of computer engineering to develop a drowsiness program for drivers to prevent accidents due to driver fatigue and insomnia. The report suggested results and solutions to the limited use of various strategies submitted to the project. While the implementation of the project gives a real idea of how the system works and what changes can be made to improve the implementation of the whole system.

In addition, this paper is intended to review all authors' efforts to help them continue their work in the field to achieve their optimal use of safe roads.

# 1. INTRODUCTION

## 1.1 PURPOSE

### 1.1.1 HUMAN PSYCHOLOGY WITH CURRENT TECHNOLOGY

People often invent machines and innovative techniques to simplify and protect their lives, to perform routine tasks such as going to work, or for interesting purposes such as flying. With the advancement of technology, the means of transportation continued and our advances have begun to increase exponentially. Now, we can travel to places faster even with our grandparents who didn't think it was possible. In modern times, almost everyone in this world uses some form of transportation. Some people are rich enough to own their own cars while others use public transportation. However, there are certain rules and codes of conduct for those who drive regardless of their social status. One of them is to stay awake and active while driving.

Neglecting our duties in terms of safe travel has given impetus to the thousands of disasters to be associated with this wonderful invention every year. It may seem like a small thing to most people but following the rules and regulations on the road is very important. While on the road, the car uses a lot of energy and is in careless hands, it can be dangerous and sometimes, that negligence can damage the lives of people and even people on the road. Another form of negligence is not acknowledgment when we are too tired to drive. To monitor and prevent the destructive effects of such negligence, many researchers have written research papers on drowsiness detection system. But sometimes, some points and assumptions made by the system are not enough. Therefore, to provide information and an alternative way of

looking at the existing problem, in order to improve their use and continue to develop a solution, this project has been implemented.

### 1.1.2 FACTS & STATISTICS

Our current statistics show that in 2019 in India alone, 151,113 people died as a result of car-related accidents. Of these, at least 40 percent were caused by fatigue that causes drivers to make mistakes. This can be a very small number still, as among the many causes that can lead to an accident, the involvement of fatigue as a cause is often underestimated. Fatigue combined with poor infrastructure in developing countries like India is a catastrophic process. Fatigue, in general, is more difficult to measure or maintain than alcohol and drugs, with clear key indicators and readily available tests. Perhaps, the best solutions to this problem are to raise awareness of the dangers associated with fatigue and to encourage drivers to acknowledge fatigue when needed. The first one is harder and more expensive to achieve, and the last one is not possible without the past as driving for many hours has great benefits. When there is a growing demand for work, the pay gap goes up leading to more people accepting it. Such is the case with night transport vehicles. Money encourages drivers to make unwise decisions such as driving late at night and getting tired. This is because drivers are not aware of the serious dangers associated with driving while you are tired. Some countries have limits on the number of hours a driver can drive outside, but it is still not enough to solve this problem as its implementation is very difficult and expensive.

## 1.2 DOCUMENT CONVENTIONS

Main Heading Font size: 22 (bold fonts)

Sub-headings Font size: 16 (bold fonts)

Sub-headings Content Font size: 16 (normal fonts)

## 1.3 INTENDED AUDIENCE

The intended audience for this document is the user and the project evaluation jury.

## 1.4 PRODUCT SCOPE

There are many products out there that offer a measurement of fatigue to drivers that is used in many cars. Driver drowsiness system offers the same performance but with better results and additional benefits. Also, notify the user of access to a specific sleep limit.

## 1.5 PROBLEM DEFINITION

Fatigue is a security issue that has not been seriously considered by any country in the world, largely because of its nature. Fatigue, in general, is more difficult to measure or maintain than alcohol and drugs, with clear key indicators and readily available tests. Perhaps, the best solutions to this problem are to raise awareness of the dangers associated with fatigue and to encourage drivers to acknowledge fatigue when needed. The first one is harder and more expensive to achieve, and the last one is not possible without the past as driving for many hours has great benefits.

# 2. LITERATURE REVIEW

## 2.1 LITERATURE SURVEY

**In 2008**, Hong Su et. al. [15] described '**A Partial Least Squares Regression-Based Fusion Model for Predicting the Trend in Drowsiness**'. They proposed a new technique of modelling driver drowsiness with multiple eyelid movement features based on an information fusion technique—partial least squares regression (PLSR), with which to cope with the problem of strong collinear relations among eyelid movement features and, thus, predicting the tendency of the drowsiness. The predictive precision and robustness of the model thus established are validated, which show that it provides a novel way of fusing multi-features together for enhancing our capability of detecting and predicting the state of drowsiness.

**In June, 2010**, Bin Yang et. al. [16] described '**Camera-based Drowsiness Reference for Driver State Classification under Real Driving Conditions**'. They proposed that measures of the driver's eyes are capable to detect drowsiness under simulator or experiment conditions. The performance of the latest eye tracking based in-vehicle fatigue prediction measures are evaluated. These measures are assessed statistically and by a classification method based on a large dataset of 90 hours of real road drives. The results show that eye-tracking drowsiness detection works well for some drivers as long as the blinks detection works properly. Even with some proposed improvements, however, there are still problems with bad light conditions and for persons wearing glasses. As a summary, the camera-based sleepiness measures provide a valuable contribution

for a drowsiness reference, but are not reliable enough to be the only reference.

**In 2011**, M.J. Flores et. al. [17] described '**Driver drowsiness detection system under infrared illumination for an intelligent vehicle'**. They proposed that to reduce the amount of such fatalities, a module for an advanced driver assistance system, which caters for automatic driver drowsiness detection and also driver distraction, is presented. Artificial intelligence algorithms are used to process the visual information in order to locate, track and analyse both the driver's face and eyes to compute the drowsiness and distraction indexes. This real-time system works during nocturnal conditions as a result of a near-infrared lighting system. Finally, examples of different driver images taken in a real vehicle at night-time are shown to validate the proposed algorithms.

**In June, 2012**, A. Cheng et. al. [18] described **'Driver Drowsiness Recognition Based on Computer Vision Technology'**. They presented a nonintrusive drowsiness recognition method using eye-tracking and image processing. A robust eye detection algorithm is introduced to address the problems caused by changes in illumination and driver posture. Six measures are calculated with percentage of eyelid closure, maximum closure duration, blink frequency, average opening level of the eyes, opening velocity of the eyes, and closing velocity of the eyes. These measures are combined using Fisher's linear discriminated functions using a stepwise method to reduce the correlations and extract an independent index. Results with six participants in driving simulator experiments demonstrate the feasibility of this video-based drowsiness recognition method that provided 86% accuracy.

## 2.2 SYSTEM REVIEW

This survey was designed to understand the need and the need for the general public, and in order to do so, we went through various sites and services to look at the basics. Based on this information, we conducted research that helped us discover new ideas and make different arrangements for our work. We have come to the conclusion that there is a need for the implementation of this program and we have seen that there is good progress in this field as well.

## 2.3 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.

**d. OpenCV** -OpenCV is an open-source computer vision library. It is designed for computational efficiency and with a strong focus on real time applications. It helps to build sophisticated vision applications quickly and easily. OpenCV satisfied the low processing power and high-speed requirements of our application

# 3. PROPOSED SYSTEM

## 3.1 THE DATASET

The database used for this model was created by us. To create the database, we wrote a script to detect eyes from camera and saved it to our local disk. We have categorized them as ‘Open’ or ‘Closed’. Data cleared manually by removing unwanted images that were not needed to create the model. The data contains around 7000 images of human eyes under a variety of lighting conditions. After training the model we saved it in "models / cnnCat2.h5".

Now, you can use this model to distinguish when the human eye is open or closed.

## 3.2 THE MODEL ARCHITECTURE

The model we used was built with Kera using Convolutional Neural Networks (CNN). A convolutional neural network is a special type of deep neural network that works very well for the purposes of image separation. CNN basically consists of an input layer, an output layer and a hidden layer that can have multiple layers of layers. The convolution function is performed on these layers using a filter that enables 2D matrix replication in the layer and filter.

## 3.3 PREREQUISITES

1. **OpenCV** (face and eye detection).
2. **TensorFlow –** (keras uses TensorFlow as backend).
3. **Keras –** pip install keras (to build our classification model).
4. **Pygame –** pip install pygame (to play alarm sound).
5. **Windows/MacOS/Linux –** any desktop OS.
6. **Webcam.**

## 3.4 ALGORITHM

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

Using a webcam, we will take pictures of driver as input. We made an infinite loop that will capture every frame. We use the function provided by OpenCV, **cv2.VideoCapture(0)** to use the camera and set the capture object (cap). **cap.read()** will take each frame and store in a variable.

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

First we detect the face and we convert it into grayscale because OpenCV algorithm only processes grayscale . Haar cascade classifier is used to identify face object. This line is used to set our classifier **face = cv2.CascadeClassifier('path of haar xml)**. Then detection using **faces=face.detectMultiScale(gray)**. It returns values 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 method to detect faces is used to detect eyes. First, we set the cascade classifier for eyes in left eye **leye** and right eye **reye** respectively then detect the eyes using **left_eye = leye.detectMultiScale(gray).** Now we need to pull only the eyes data from the face image. This is done by creating 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 data of the left eye. This will be inputted into our CNN classifier which will make a decision if eyes are open or closed. Similarly, we will be using the same process for the right eye.

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

Using CNN classifier, we determine the eye status. We input images with correct dimensions. Color image is converted to grayscale using **r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY)**. We resize the image to 24x24 pixels as our model was trained on those dimensions **cv2.resize(r_eye, (24,24))**. The data is normalized for better convergence **r_eye = r_eye/255** (All values will be between 0-1). The model is loaded using **model = load_model('models/cnnCat2.h5')**. Now we determine each eye status with our model

**lpred = model.predict_classes(l_eye)**. If **lpred[0] = 1**, it means that eyes are open, if **lpred[0] = 0** then, it means that eyes are closed.

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

The score is a value that's given to measure how long the driver's eyes have been closed. If both eyes are closed, we will keep incrementing the score and when eyes are open, we decrement the score. We are display the result on the screen using **cv2.putText()** function which will display the status "Open" or "Close".

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

# 4. EXPERIMENTAL SETUP

1) Raspberry Pi (running any OS that can run python compiler)

2) USB Webcam

3) Install the following commands to install necessary libraries in command prompt:

- OpenCV-> pip install opencv-python
- TensorFlow-> pip install tensorflow
- Keras-> pip install keras
- Pygame-> pip install pygame

Alternatively, you can use any IDE. In my case I have used PyCharm 2021 and enabled the necessary libraries without running commands manually.

4) Load the source files and run the main python file to start the capture.

## MAIN SOURCE FILE:

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time


mixer.init()
sound = mixer.Sound('alarm.wav')

face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')



lbl=['Close','Open']

model = load_model('models/cnncat2.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(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]

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

    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(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()
```

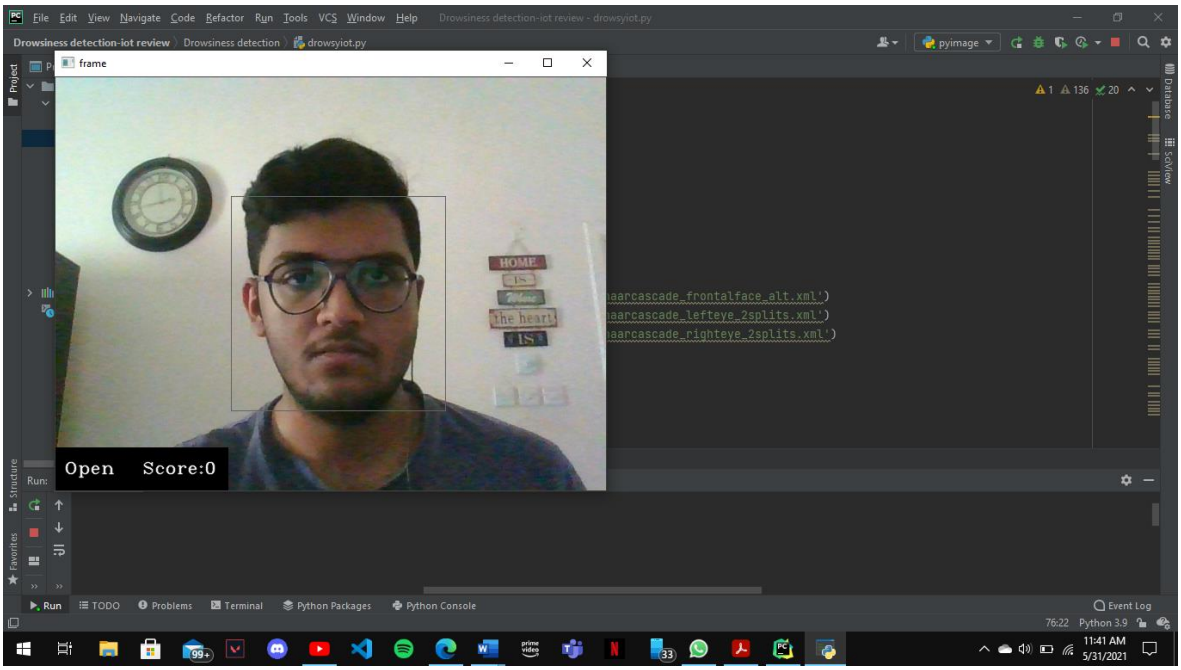
## MODEL CODE:

```
import os
from keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
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


def generator(dir, gen=image.ImageDataGenerator(rescale=1./255),
shuffle=True,batch_size=1,target_size=(24,24),class_mode='categorical' ):

    return
gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,color_mode='grayscal
e',class_mode=class_mode,target_size=target_size)

BS= 32
TS=(24,24)
train_batch= generator('data/train',shuffle=True, batch_size=BS,target_size=TS)
valid_batch= generator('data/valid',shuffle=True, batch_size=BS,target_size=TS)
SPE= len(train_batch.classes)//BS
VS = len(valid_batch.classes)//BS
print(SPE,VS)


# img,labels= next(train_batch)
# print(img.shape)

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
    MaxPooling2D(pool_size=(1,1)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#32 convolution filters used each of size 3x3
#again
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1,1)),

#64 convolution filters used each of size 3x3
#choose the best features via pooling

#randomly turn neurons on and off to improve convergence
    Dropout(0.25),
```

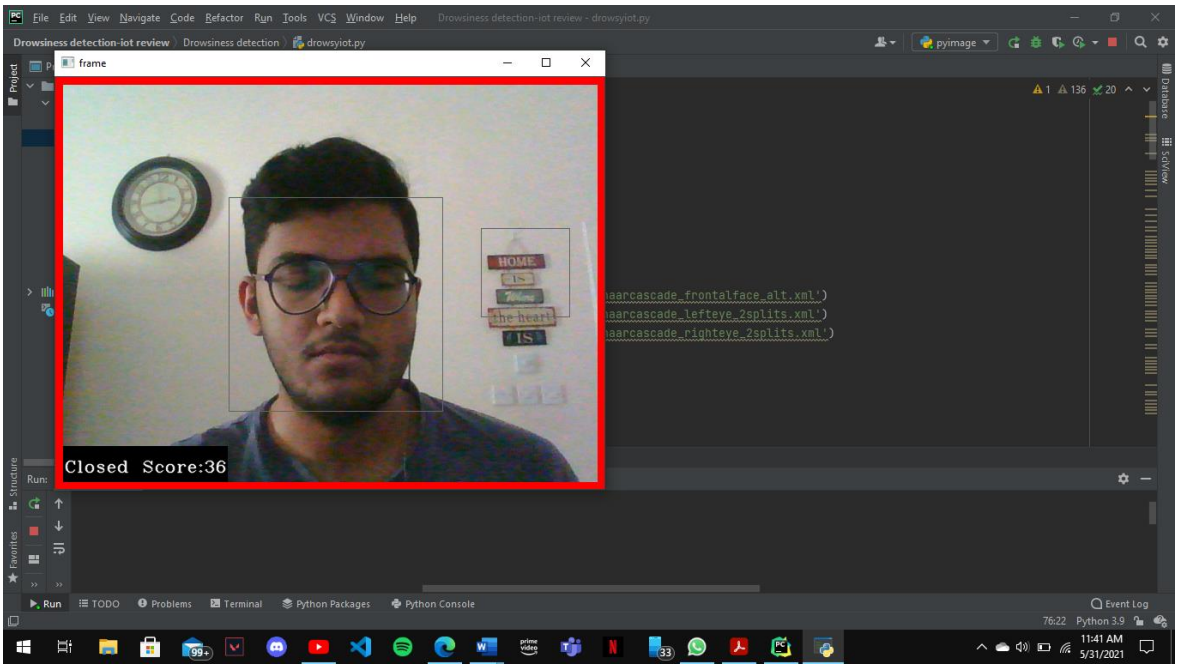
```
#flatten since too many dimensions, we only want a classification output
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
#one more dropout for convergence' sake :)
    Dropout(0.5),
#output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

model.fit_generator(train_batch,
validation_data=valid_batch,epochs=15,steps_per_epoch=SPE ,validation_steps=VS)

model.save('models/cnnCat2.h5', overwrite=True)
```

# 5. EXPERIMENTAL RESULTS

Open Eyes and Alert: Score<15




Closed Eyes and Drowsy: Score>15



# 6. CONCLUSION AND FUTURE SCOPE

## 6.1 CONCLUSION

The project meets the objectives and requirements of the system. The system works in majority of cases with code breaking bugs squashed. The system successfully serves the purpose by detecting driver status and alerting sleepy driver

## 6.2 FUTURE SCOPE

The model can be improved further by using other checks like blinking, yawning, whether the car is moving or not, etc. If all these checks are used it can improve the accuracy.

We can also include more cameras to detect other forms of distraction that takes the drivers attention from the road. For example-Texting and calling while driving, turning around to talk to passengers etc.

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

# 7. REFERENCES


[1] COMPUTATIONALLY EFFICIENT FACE DETECTION; B. SCHLKOPF-A. BLAKE, S. ROMDHANI, AND P. TORR.

[2] USE OF THE HOUGH TRANSFORMATION TO DETECT LINES AND CURVES IN PICTURE; R. DUDA AND P. E. HART.

[3] JAIN, "FACE DETECTION IN COLOR IMAGES; R. L. HSU, M. ABDEL-MOTTALEB, AND A. K. JAIN.

[4] OPEN/CLOSED EYE ANALYSIS FOR DROWSINESS DETECTION; P.R. TABRIZI AND R. A. ZOROOFI.

[5] A REVIEW: DRIVER DROWSINESS DETECTION SYSTEM; CHISTY, JASMEEN GILL.

[6] http://ncrb.gov.in/StatPublications/ADSI/ADSI2015/chapter1A%20traffic%20accidents.pdf

[7] http://www.jotr.in/text.asp?2013/6/1/1/118718

[8] http://dlib.net/face_landmark_detection_ex.cpp.html