MACQUARIE University
SYDNEY·AUSTRALIA

MACQUARIE UNIVERSITY

Faculty of Science and Engineering

Department of Computing

COMP4060 - ADVANCED SOFTWARE ENGINEERING

CONTINUOUS PROFESSIONAL DEVELOPMENT PROJECT

PROJECT TITLE: “DROWSY DRIVER DETECTION SYSTEM”

Student Name : KODALI SAHITHI

Student ID : 45712050

Contact Details : 0451116411

Date : 26-05-2020

ACKNOWLEDGEMENT

I would like to articulate my deep gratitude to Prof. Michael Johnson for his expert guidance and continuous encouragement throughout the project work.

I also would like to thank him for providing feedback regularly helping us reflect on our improvement throughout the unit to develop the skills a Software Engineer needs to acquire before indulging into the workforce.

- Sahithi Kodali

ABSTRACT

Drowsy Driving has been one of the major causes behind the dangerous accidents occurring these days in the world. Drowsy Driving refers to the combination of driving and lack of sleep. In this busy world, it has become difficult to get sufficient sleep due to hectic schedules. Lack of Sleep leads to Sleep deprived driving which is prone to more fatal accidents compared to drunk driving. According to the Statistics, around 2,50,000 number of drivers fall asleep daily while driving. About quarter of the accidents in the world are due to sleep deprived sleeping.

One such solution which can curb the number of accidents due to drowsiness is by detecting if the driver is sleepy and alarming. This project is developed to implement one such Drowsy Driver Detection System. This system detects the eyes of the driver and alarms if the driver is sleepy. A driver is termed sleepy, if he/she closes the eyes for more than a certain period of time. Through this system, we can reduce the number of accidents in a great number thus helping the society in a significant way.

This project is implemented in two ways namely; Webcam based and using Convolutional Neural Network. Both the implementations tend to show good results in detecting the drowsiness of the driver.

CONTENTS

1. INTRODUCTION

Everyday, we see a countless number of vehicles on the road where every person has a different reason to drive in different situations. In this busy world, it has become really essential to handle a lot of things where getting essential sleep has been out of reach. Insufficient sleep leads to dangerous accidents while driving. Statistics prove that two out of every five drivers tend to sleep while driving due to lack of sleep.

One solution to curb the number of accidents due to drowsiness can be to detect if the driver is sleepy and issue an alert signal. This solution is implemented under the name “Drowsy Driver Detection System” in this project.

Designing this “Drowsy Driver Alert System” model, helps in reducing the accidents by a great number. This system is designed to detect the drowsiness of the driver and alerts the driver through an alarm. It focuses on the eyes and mouth of the driver and determines if the person is drowsy or yawning. If the driver is Drowsy and close eyes for more than a certain period of time the alarm activates thus alerting the driver.

2. IMPORTANCE AND MOTIVATION

Developing a Drowsy Driver Detection System is important because it can help the society by reducing the number of accidents due to lack of sleep to a great extent. This car safety technology helps in saving the lives of people including the driver, passengers etc. Though we cannot completely rely on this technology it can still be efficient in controlling the number of accidents due to fatigue.

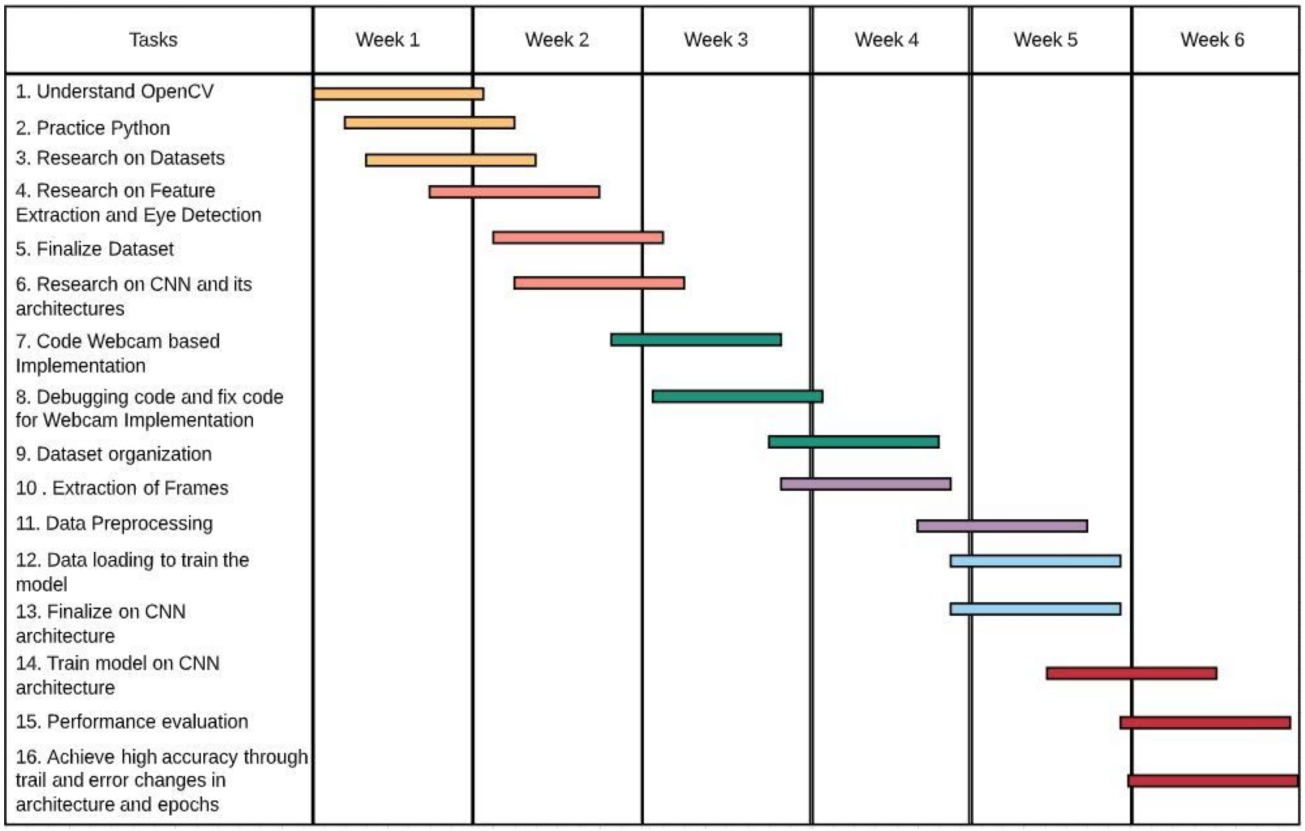
The implementation of this project requires knowledge about concepts related to the field of Data Science like Computer Vision and Deep Learning. Concepts related to Image and Video processing, Feature Extraction, Keras and Convolutional Neural Networks play a key role in the implementation of this system.

Involving a high research edge, implementation of this project helps in improving the data handling skills driving me towards the end goal of becoming a Data Science related engineer. Learning about various concepts in Computer Vision and Deep Learning in the implementation of this project can provide me with a real time project experience thus developing my knowledge in Neural Networks, Image and Video processing.

The high end research and learning scope in this project along with its benefits for the society motivated me to choose “Drowsy Driver Detection system” as a Continuous Professional Development Project and implement it successfully.

3. PROJECT TIMELINE

This CPD Project has been implemented in a 6-week duration which is organized on a weekly basis as shown below using a Ghantt Chart



4. TOOLS AND TECHNOLOGIES

4.1. Tools

The tools used to implement this project are Anaconda and Google drive and Collab support. The reason for choosing these tools is further discussed in this section.

4.1.1. Anaconda - Jupyter

Anaconda framework provides various Data Science tools including Python through Jupyter, Deep Learning libraries, Keras and Tensorflow, OpenCV etc on a single platform. These packages and libraries can be imported and used as required after installation.

4.1.2. Google drive and Collab

Google Drive and Collab are used to manage the Dataset in a much easier way. Google Collab Trains the model in the project quickly compared to the local computers due to its 12GB GPU capacity. The ability to mount the drive to access files in the Google drive makes this duo stand out while working with huge datasets as in this project.

4.2. Technologies

The Implementation of this project involves various technologies like OpenCV, Tensorflow(Keras), Python Language for coding and knowledge on working with Neural Networks. A brief on these technologies is further discussed in this section and discussed in detail in the Implementations

4.2.1. OpenCV

OpenCV is a library which enables different functions related to Computer Vision. It includes various modules and methods related to Image Processing, Video Processing etc. Installing this library helps in working with images and videos easily (see Appendix 1)

4.2.2. Tensorflow

Tensorflow is the Artificial Intelligence Math library which uses data flow graphs to build models. It is used extensively in Deep Learning to build Neural Networks with multiple layers. It supports different Machine Learning and Deep Learning applications.(see Appendix 2)

4.2.3. Python

Python is one of the most used programming languages in the field of Data Science. It is a general purpose high level programming language which can handle the complexities in Machine Learning, Deep Learning models by providing vast library support.

4.2.4. Neural Networks

Neural Networks are specific algorithms built similar to the human brain. These networks help in recognising patterns thus playing a key role in Deep Learning. These Neural Networks continuously learn from raw data and patterns improving its results and performance over time. There are different types of Neural Networks such as Artificial Neural Networks(ANN), Convolutional Neural Networks(CNN), Recurrent Neural Network (RNN) etc. This project is implemented using Convolutional Neural Networks.(see Appendix 3)

5. TYPES OF IMPLEMENTATION

This project is implemented in two different ways namely; Webcam based Implementation and the other Implementation using Convolutional Neural Networks(CNN). These Implementations are further discussed in detail.

5.1. Implementation 1 - Webcam based

The Webcam based Implementation involves the concepts of Computer Vision. These Computer Vision operations can be performed using OpenCV library. The methods provided by OpenCV library are extensively used in the Image and Video Processing. This implementation utilizes these OpenCV libraries and Feature Extraction concepts to detect drowsiness in the driver.

The technology behind this Implementation, how feature extraction occurs, how does this help in real time drowsiness detection is further analyzed and discussed in detail in the Implementation 1 details.

5.2. Implementation 2 - CNN based

The CNN based Implementation as named involves the concepts of Convolutional Neural Networks, a Deep Learning set of Algorithms.

It involves a Dataset through which the model that has been designed learns and produces results. The learnt model is then tested on a similar test dataset to evaluate the model Performance. CNN comes into place to develop one such model which can learn through the dataset given.

Convolutional Neural Networks is a class of Deep Neural Networks. These can be built using various architectures such as VGGNet, LeNet, AlexNet, ResNET, Inception etc. By building the architecture the model learns by itself and produces the final results on a test dataset in the form of accuracy.

Accuracy determines the performance of the model. How the model can be evaluated, how we can build such architectures and how to increase the performance by improving the accuracy is further analyzed and discussed in detail in th eImplementation 2 details.

6. IMPLEMENTATION 1

The Webcam based Implementation includes various concepts like Feature Extraction, Detection of Region of Interest, Eye blink and Yawn normalization to detect the drowsy driver and built the detection system. OpenCV library is extensively used to detect each image or frame in a video. Each functionality used is discussed in detail further in this section.

6.1.1. Role of OpenCV

OpenCV library provides all the Computer Vision functions required for Image and Video processing in a single library. All the methods can be used by Importing the package in the coding environment. It provides methods to read,write and display images.

Each image is converted into numbers which are known as Pixel values with each number representing the pixel intensity. Reading and writing the images is crucial in implementing this project and this functionality is made easy by the OpenCV library.

These methods to read, write and display images and how they are used in the coding part of this implementation is described in depth in Appendix for reference. (see Appendix 1) [1]

6.1.2. Feature Extraction

After the detection of images and performing image processing i.e change color of the images as required, perform rotation and resizing if required we need to extract the features.

Here, we need to obtain the facial landmark points. This can be termed as a subset of shape prediction problems. Our main objective is to determine the important facial features using the shape predictor methods provided and then focus on the region of Interest according to the project goals. The Region of Interest in this project is Eyes and Mouth.

The facial landmark points can be determined in two steps:

i.) Localize the face on the images

To localize the face on the images, we utilize the inbuilt Deep Learning based algorithms. The algorithm uses some method(dlib facial landmark detection in this project) to detect the face in the image and provide the (x,y) coordinates as landmarks of the face. This localizes the face with (x,y) coordinates i.e a bounding box of the face is determined in the image. (see Appendix 4) *ii.) Detect the Region Of Interest(ROI)*

After localizing the face, we need to locate the Region of Interest (locating specific features required) which in this project are Eyes and Mouth. We need to locate the Eyes to determine if the person is Drowsy and Mouth to determine if the person is Yawning to to alert the driver. This step labels the required parts of the face using their coordinates which can be used as needed according to the requirements of the project.

6.1.3. Dlib Facial Landmark Detector

To detect the facial landmarks we utilized the Facial Landmark Predictor provided by Dlib Library. Dlib provides the inbuilt algorithm to detect the facial landmarks points.

This predictor starts by using a training set of manually labelled facial landmarks on an image. It specifies the (x,y) coordinates of the facial structure with each specific facial feature denoted with their indexes. An ensemble of regression trees are trained on this training data to estimate the facial landmark positions. The predictor used in the project was trained on a 68 point iBUG 300-W dataset . Implementation of the dlib library can be understood in depth through the attached article in the Appendix section. (see Appendix 5)

This pre-trained detector is used to estimate the location of 68 (x,y) coordinates used to map the facial structure. These indexes of the coordinates can be visualized as below(fig1). The implementation details is described in detail in the Appendix section(see Appendix 5) [2]

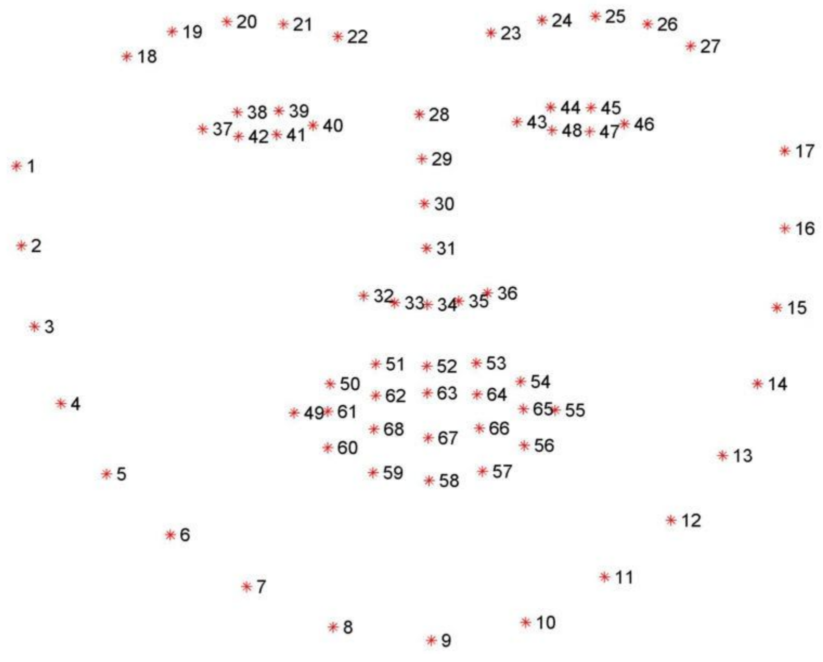


Fig 1: 68 face landmark points Visualization [5]

6.1.4. Eye blink and Yawn Detection

The Region Of Interest (ROI) in this Project is Eyes and Mouth(fig 2). So, we extract the landmark points of eyes and mouth to analyze their dimensions. By examining their dimensions we can determine if the person is closing eyes or not and yawning or not.

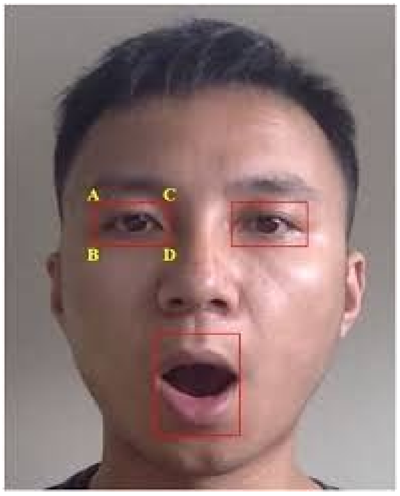


Fig 2: ROI as eyes and mouth [6]

To determine if the person is closing eyes or not, the ratio between the height and width of each eye is determined. According to fig1, the coordinates of eyes are [37, 42] . Consider these points in abstract form as below(fig 3) and visualize how ratio is determined.

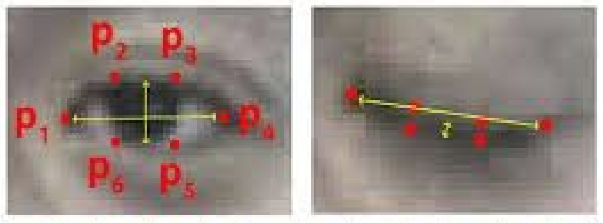


Fig 3: Eye coordinates when open and close [7]

Aspect Ratio = ||p2 - p6| + |p3-p5|| / |p1 - p4| [7]

where; p1, p2, p3…… are the coordinates of the eyes in the range [37, 42].

After determining the ratio of both Right and Left eye, the mean of both right and left eye is obtained. A threshold is set through a trial and Error approach. If the mean is less than the threshold value then the person is considered to be drowsy. A count is set which increases when the person is drowsy, but activates the alarm only after a fixed count is reached which in our program is 30 (~ 4-7 seconds). This is set to get rid of false alarms when the driver is just blinking or yawning.

Similarly, the ratio of mouth is calculated with the coordinates within range [43, 48]. A threshold value is set based on the ratio of Mouth. If the ratio increases the threshold value, then the person is said to be yawning. When the person is Yawning, the system delivers a message saying, “You are Yawning. Please drink Water”. The Details of the Implementation of the code is described in the Appendix section in detail(see Appendix 6)

6.1.5. Results

By Implementing the concepts above, the results can be visualized in the following figures.

All the images have few metrics, described briefly as follows:

1.) GOOD : Person is not Yawning
2.) ALERT : The person is Alert
3.) DROWSY: The person is slightly drowsy
4.) DROWSY ALARM : The person is highly drowsy, hence activate alarm
5.) Count_yawn : Count to measure the time of Yawning
6.) Count: Count to measure the total time the person is Drowsy

The Bounding Box indicates that the facial landmark points are determined and are marked as Red dots in the Image. The code for implementing this can be referred to in the Appendix (see Appendix 4)

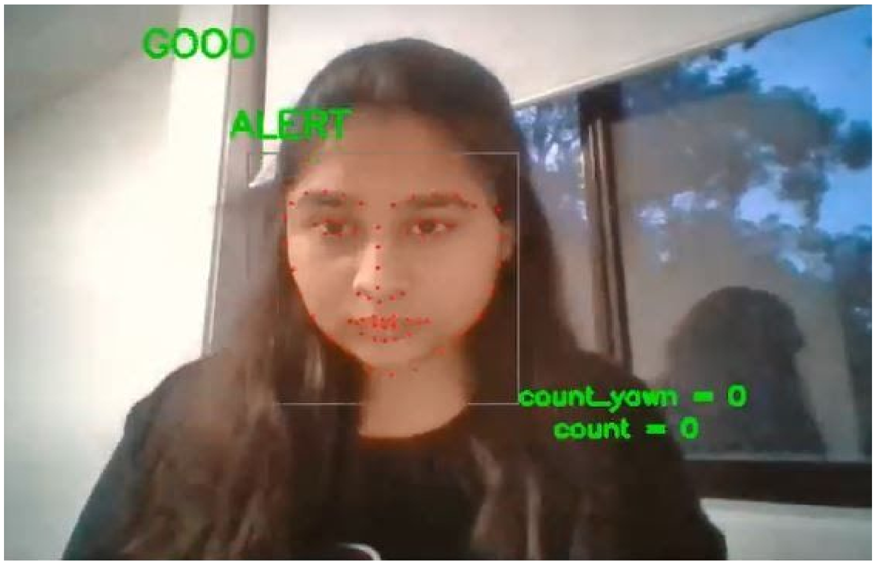


Fig 4 : Image when the person is Alert and not Yawning

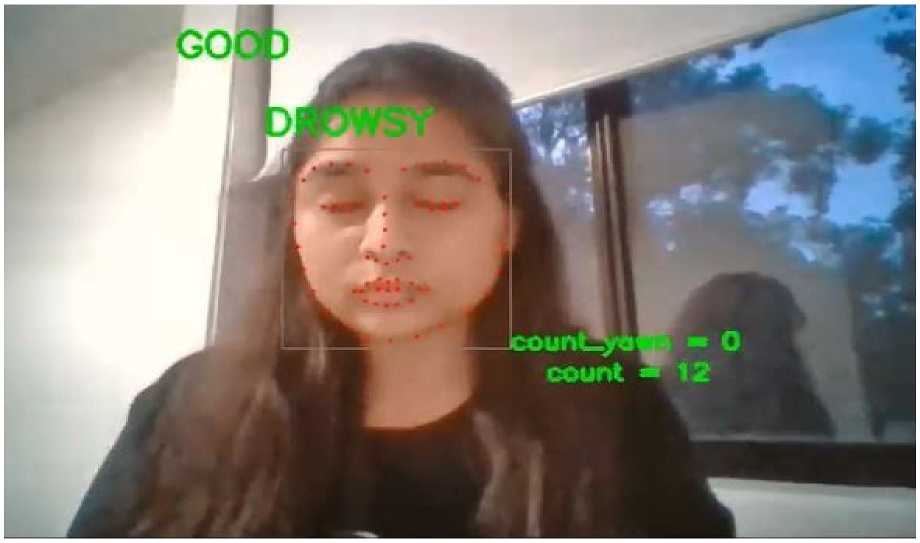


Fig 5 : Image when the person is Drowsy and not Yawning

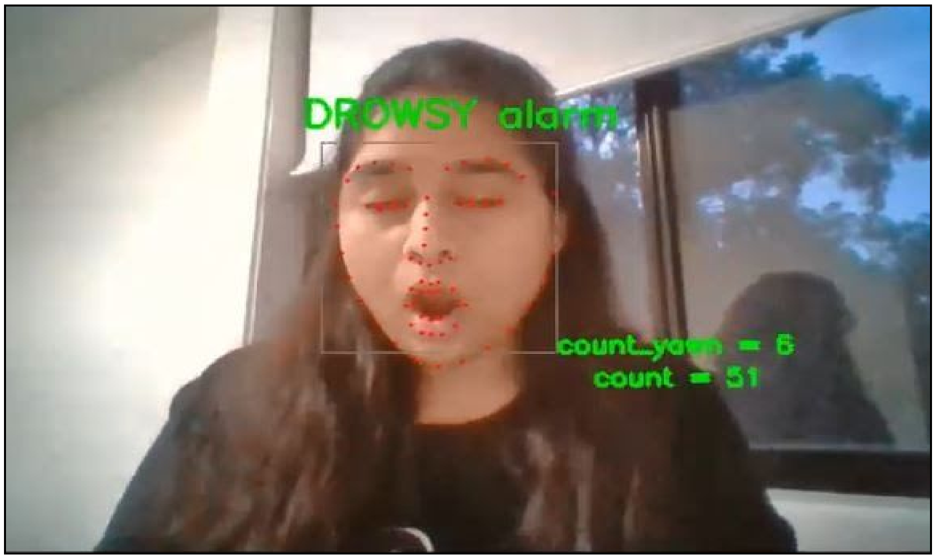


Fig 6 : Image when the person is Highly Drowsy (Alarm indication) and slightly Yawning

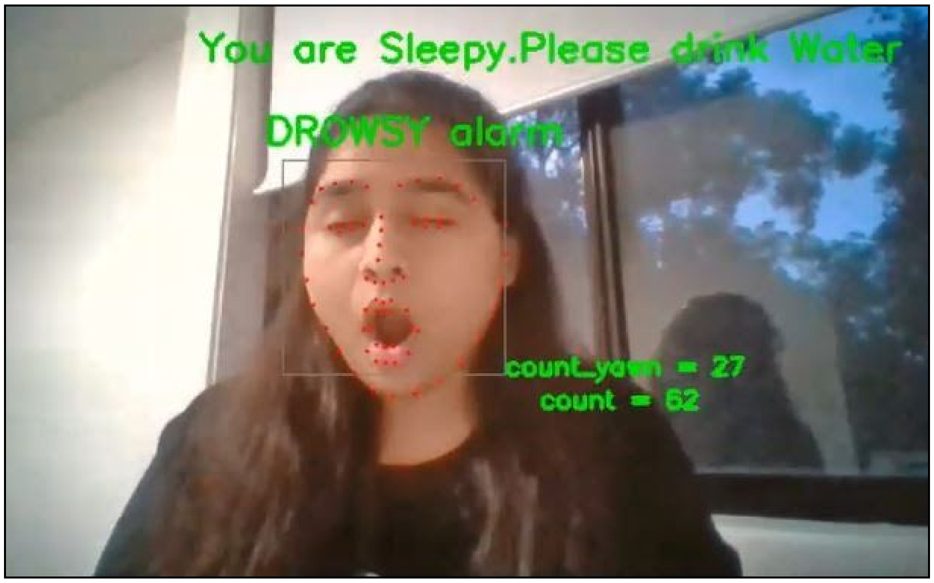


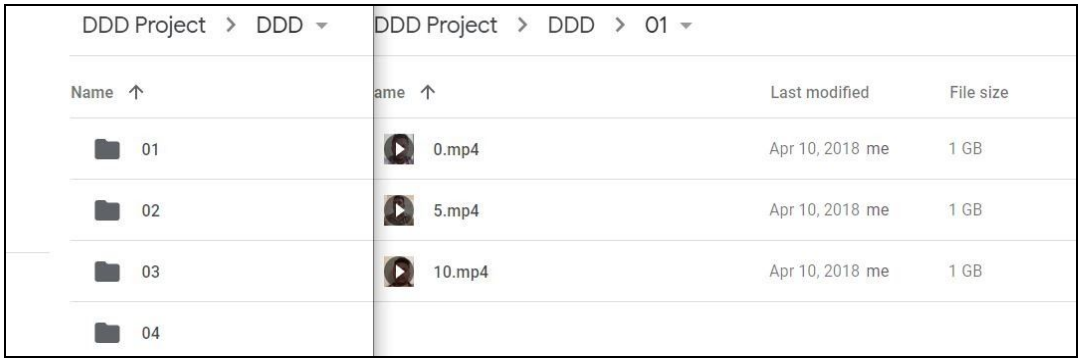
Fig 7 : Image when the person is Highly Drowsy (Alarm indication) and Yawning

7. IMPLEMENTATION 2

This Implementation involves the key concept of Convolutional Neural Network. The model is trained based on the dataset and then tested on unseen data to determine the performance.

7.1.1. Brief on Dataset

The dataset consists of 30 videos with each video having 3 sub-videos of a person in alert (0.mp4), slightly Drowsy (5.mp4) and completely Drowsy modes(10.mp4). (fig 8)



DDD Project > DDD

Name ↑
01
02
03
04

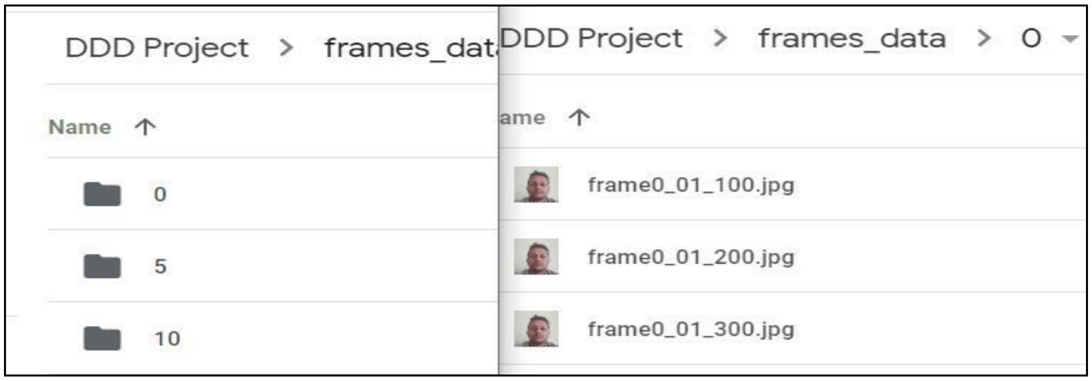
DDD Project > DDD > 01

ame ↑	Last modified	File size
0.mp4	Apr 10, 2018 me	1 GB
5.mp4	Apr 10, 2018 me	1 GB
10.mp4	Apr 10, 2018 me	1 GB

Fig 8: Image sample of Dataset

7.1.2. Extraction of Frames and Image processing

The next step after obtaining the dataset is to extract frames from each video. From each video we extract about 120 frames with a total of 10000 frames from all the videos.(fig 9)



DDD Project > frames_dat

Name ↑
0
5
10

DDD Project > frames_data > 0

ame ↑
frame0_01_100.jpg
frame0_01_200.jpg
frame0_01_300.jpg

Fig 9: Image sample of frames extracted

After extraction of Frames, the frames need to be processed into the same format, size and color. Preprocessing is required to obtain the best performance of the model.

We rotate all the frames into same angle, resize them into similar dimensions(240*135) and change all the frames into a similar color. This can be implemented using the OpenCV methods provided. The code for implementing the extraction of frames and the processing can be referred to in the Appendix(see Appendix 7)

7.1.3. Summary on CNN Architecture

The dataset is then trained using Convolutional Neural Networks. CNN performs image processing and pattern recognition efficiently. CNN consists of different architectures of which VGGnet Architecture, one of the best architecture for Multi Classification and hence is used in this implementation. [3]

VGGNet architecture is a sequential layer architecture with multiple Convolution layers with different dimensions, Max Pooling, activation functions to obtain one dimensional structure of an image. Convolution can be defined as sliding a filter over an input whose dimensions are reduced using Max Pooling. Implementation of the code is described in detail in the Appendix. (see appendix 8) [4]

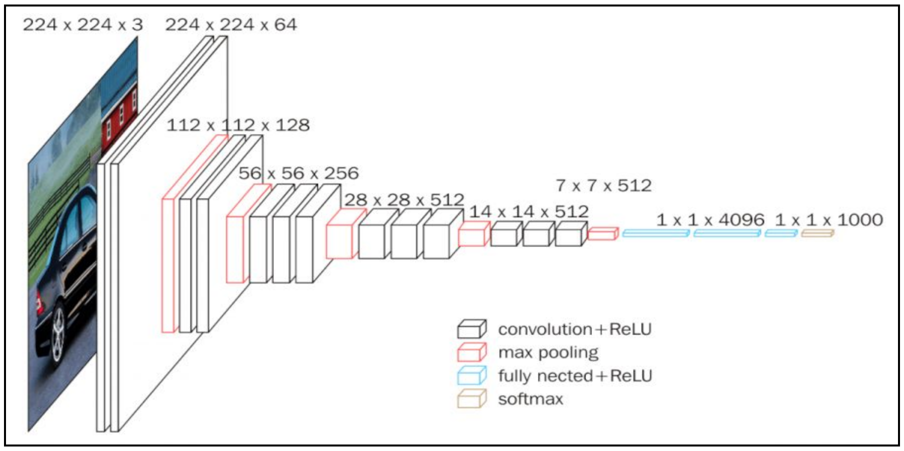
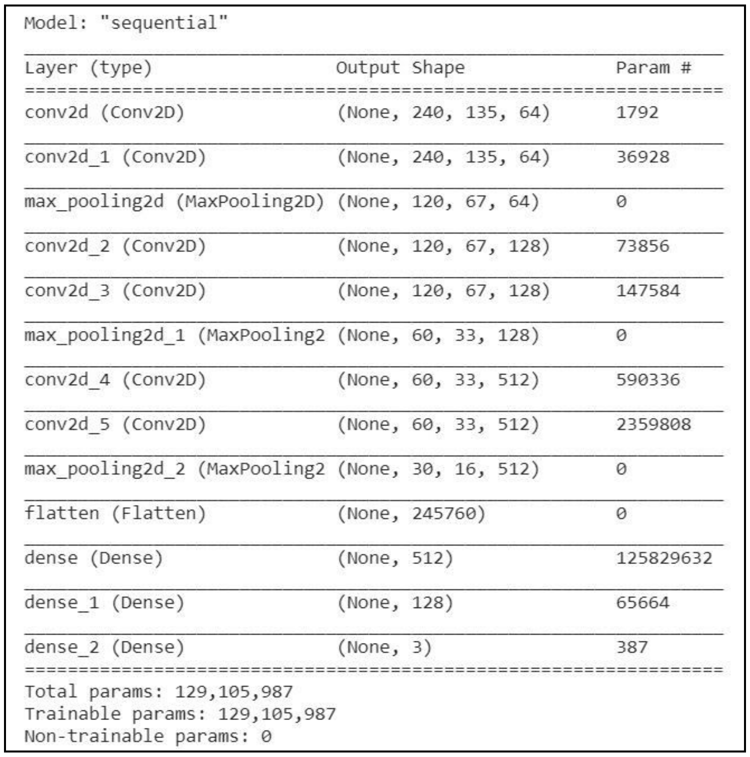


Fig 10 : Sample of CNN structure [8]



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 240, 135, 64)      1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 240, 135, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 120, 67, 64)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 120, 67, 128)      73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 120, 67, 128)      147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 60, 33, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 60, 33, 512)       590336
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 60, 33, 512)       2359808
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 16, 512)       0
_________________________________________________________________
flatten (Flatten)            (None, 245760)            0
_________________________________________________________________
dense (Dense)                (None, 512)               125829632
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
=================================================================
Total params: 129,105,987
Trainable params: 129,105,987
Non-trainable params: 0
```

Fig 11: Summary of CNN architecture used

7.1.4. Training and Testing of the model

The dataset is divided into a 70:30 ratio for Training and testing the model. 70% of the dataset is trained to the CNN model built and compared to the ground truth given i.e Alert, Slightly Drowsy and Drowsy.(See Appendix 9)

The model is trained on the training data by dividing this data into a given number of epochs with each epoch having a certain number of frames chosen randomly and trained repeatedly. On each epoch, the accuracy is checked through which an average accuracy can be determined.

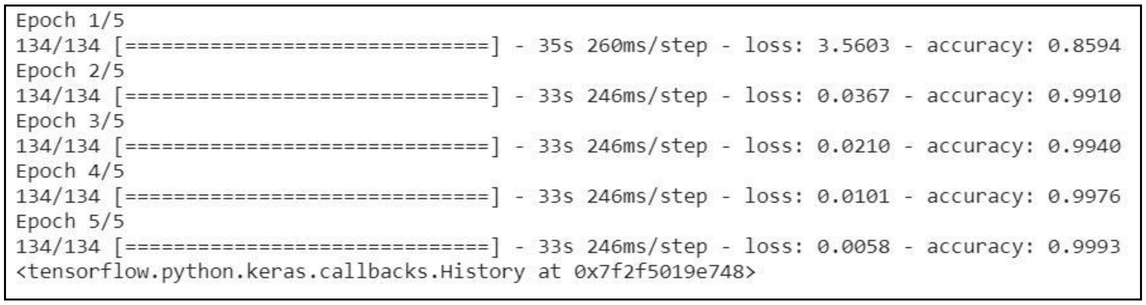
The trained model is then tested on the remaining 30% dataset. The Accuracy for the test dataset is determined. The weights and layers are adjusted through Trial and error methods to obtain the best accuracy possible.

The code for training the model can be referred to in the Appendix section (see Appendix 10)

7.1.5. Performance Evaluation

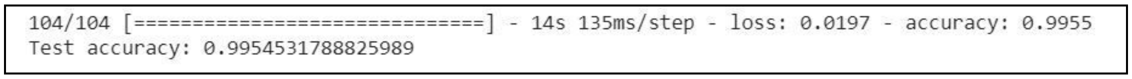
The model is then evaluated to check its Performance. The model is said to perform well if the accuracy is high and the training accuracy is nearly equal to the testing Accuracy. Overfitting(Good performance on Training data but low generalization on Testing Data) and Underfitting(Poor performance on training and Testing Data) of the model needs to be checked and required changes need to be made to obtain the best performance of the model. The higher the Accuracy, the better the performance of the model.

The Training and testing accuracy of the model implemented can be visualized below. Code of the implementation can be referred to in the Appendix (see Appendix 11)



```
Epoch 1/5
134/134 [==============================] - 35s 260ms/step - loss: 3.5603 - accuracy: 0.8594
Epoch 2/5
134/134 [==============================] - 33s 246ms/step - loss: 0.0367 - accuracy: 0.9910
Epoch 3/5
134/134 [==============================] - 33s 246ms/step - loss: 0.0210 - accuracy: 0.9940
Epoch 4/5
134/134 [==============================] - 33s 246ms/step - loss: 0.0101 - accuracy: 0.9976
Epoch 5/5
134/134 [==============================] - 33s 246ms/step - loss: 0.0058 - accuracy: 0.9993
<tensorflow.python.keras.callbacks.History at 0x7f2f5019e748>
```

Fig 12 : Accuracy of the model on the training data



```
104/104 [==============================] - 14s 135ms/step - loss: 0.0197 - accuracy: 0.9955
Test accuracy: 0.9954531788825989
```

Fig 13 : Accuracy of the model on the testing data

8. FUTURE IMPROVEMENTS

The future Improvement for this project can be Implementing the Alarm sound when the person is Highly Drowsy when the indication of Drowsy alarm is displayed.

We can also implement an algorithm such that the car slows down if the person continues to be drowsy even after the alarm stops.

9. LEARNINGS AND DRAWBACKS

Learnings:

1.) Importance of selecting a standard Dataset can be understood.
2.) Significance of Data Processing is realized
3.) The role of choosing a suitable Architecture.

Drawbacks:

1.) Real time testing for the CNN implementation is not executed.
2.) Excessive Trial and Error Approach.

10. HOW IT ASSISTED IN MY CPD

➢ Experience with Computer Vision and Deep Learning concepts
➢ Understand Feature Extraction
➢ Enhance knowledge on CNN architectures
➢ Exposure on Evaluation Techniques

The above listed learnings can show the progression towards reaching the end goal of Data Science Engineer. The concepts used are some of the crucial topics in working with Data, thus assisting in the Continuous Professional Development.

11. CONCLUSION

Thus, in this way the Drowsy Driver Detection System can be implemented in both ways. It also assists in a great way in reducing the number of accidents helping the society. The Implementation of this project not only helps the society but also helps in Continuous Professional Development thus helping in reaching the end goal of Data Science related engineer.

12. REFERENCES

1. Saurabh Pal. 2019,March 25. 16 OpenCV Functions to Start your Computer Vision journey (with Python code) Retrieved from: https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/

2. Adrian Rosebrock. 2017, April 13. Facial landmarks with dlib, OpenCV, and Python Retrieved from:https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

3. Sumit Saha. 2018, Dec2016. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Retrieved from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

4. Raimi Karim. 2019, July 30. Illustrated: 10 CNN Architectures. Retrieved from: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

5. Huber Patrick. 2018, Sept 6. Real-time 3D morphable shape model fitting to monocular in-the-wild videos. Retrieved from: https://www.researchgate.net/figure/The-ibug-68-facial-landmark-points-mark-up_fig9_327500528

6. Wanghua Deng and Ruoxue Wu. 2019, Sept 5. Real-Time Driver-Drowsiness Detection System Using Facial Features. Retrieved from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8808931&tag=1

7. Adrian Rosebrock. 2017, April 24. Eye blink detection with OpenCV, Python, and dlib. Retrieved from: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

8. Muneeb ul Hassan. .2018, Nov 20. VGG16 – Convolutional Network for Classification and Detection. Retrieved from: https://neurohive.io/en/popular-networks/vgg16/

13. APPENDIX

Appendix 1

More information on OpenCV can be accessed through this article.
https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/

Appendix 2

More information on Tensorflow can be accessed through this article.
https://www.tensorflow.org/

Appendix 3

More information on NeuralNetworks can be accessed through this article.
https://pathmind.com/wiki/neural-network

Appendix 4

Import required libraries and packages into the coding environment

```
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2
```

Take a bounding predicted by dlib and convert it to the format (x, y, w, h) as we would normally do with OpenCV. Return a tuple of (x, y, w, h)

```
def rect_to_bb(rect):
   x = rect.left()
   y = rect.top()
   w = rect.right() - x
   h = rect.bottom() - y
   return (x, y, w, h)
```

Initialize the list of (x, y)-coordinates. Loop over the 68 facial landmarks and convert them to a 2-tuple of (x, y)-coordinates. Return the list of (x, y)-coordinates

```
def shape_to_np(shape, dtype="int"):
    coords = np.zeros((68, 2), dtype=dtype)
    for i in range(0, 68):
       coords[i] = (shape.part(i).x, shape.part(i).y)
     return coords
```

Appendix 5

Dlib is an implementation of ensemble tress which can be understood in depth using the following paper https://www.semanticscholar.org/paper/One-millisecond-face-alignment-with-an-ensemble-of-Kazemi-Sullivan/d78b6a5b0dcaa81b1faea5fb0000045a62513567

Initialize dlib's face detector (HOG-based) and then create the facial landmark predictor.

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('D:/comp 4060/DDD/shape_predictor_68_face_landmarks.dat')
```

Appendix 6

Determine the facial landmarks for the face region, then convert the facial landmark (x, y)- coordinates to a NumPy array. Convert dlib's rectangle to a OpenCV-style bounding box [i.e., (x, y, w, h)], then draw the face bounding box. Loop over the (x, y)-coordinates for the facial landmarks and draw them on the image. Show the output image with the face detections + facial landmarks

```
vid = cv2.VideoCapture(0)
count = 0
count_yawn = 0
previous_output = "ALERT"
mouth_output = "GOOD"

while(True):
    ret,image = vid.read()
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    rects = detector(gray, 1)
    for (i, rect) in enumerate(rects):
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        left_eye_f =
(abs(shape[37,1]-shape[41,1])+abs(shape[38,1]-shape[40,1]))/abs(shape[36,0]-shape[39,0])
```

```
        right_eye_f =
 (abs(shape[43,1]-shape[47,1])+abs(shape[44,1]-shape[46,1]))/abs(shape[42,0]-shape[45,0])
      Yawn =
 (abs(shape[49,1]-shape[59,1])+abs(shape[61,1]-shape[67,1])+abs(shape[62,0]-shape[66,0])
+abs(shape[63,0]-shape[65,0])+abs(shape[53,0]-shape[55,0]))/abs(shape[48,0]-shape[54,0])

         (x, y, w, h) = face_utils.rect_to_bb(rect)
        cv2.rectangle(image, (x, y), (x + w, y + h), (127, 127, 127), 1)

        if ((left_eye_f+right_eye_f)/2)>0.56:
            count = 0
            previous_output = "ALERT"
            cv2.putText(image,"ALERT", (x-10,y-10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

        elif ((left_eye_f+right_eye_f)/2)<0.55:
            if previous_output == "DROWSY":
                count = count + 1
                cv2.putText(image,"DROWSY", (x-10,y-10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
            if count>=30:
                count = count +1
                previous_output = "DROWSY alarm"
                cv2.putText(image,"DROWSY alarm", (x-10,y-10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
            elif previous_output == "ALERT":
                previous_output = "DROWSY"
                count = 1
                cv2.putText(image,"DROWSY", (x-10,y-10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

        cv2.putText(image,"count = %d" % count, (x + w + 20, y + h + 20),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        if Yawn < 0.8:
            count_yawn = 0
            mouth_output = "GOOD"
            cv2.putText(image,"GOOD", (x-60,y-60),
        cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

        elif Yawn >= 0.8:
            mouth_output = "SLEEPY"
            count_yawn = count_yawn + 1

        if count_yawn >= 8:
            mouth_output = "SLEEPY"
            count_yawn = count_yawn+1
            cv2.putText(image," You are Sleepy.Please drink Water", (x-60,y-60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

        cv2.putText(image,"count_yawn = %d" % count_yawn, (x + w , y + h ),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

                for (x, y) in shape:
            cv2.circle(image, (x, y), 1, (0, 0, 255), -1)
    cv2.imshow("Output", image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
vid.release()
cv2.destroyAllWindows()
```

Appendix 7

➔ *Import required libraries and packages into the coding environment*

```
import cv2
import os
from glob import glob
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Conv2D,Dense,MaxPooling2D,Flatten,Dropout
import sklearn
from sklearn.model_selection import train_test_split
```

➔ *Code to extract frames from each video*

```
folder_names = os.listdir('/content/drive/DDD Project/DDD')
path = '/content/drive/DDD Project/frames_data'
for i in [0,5,10]:
   print(i)
   for folder_name in folder_names:
      print(folder_name)
      vidcap = cv2.VideoCapture('/content/drive/DDD
Project/DDD'+'/'+folder_name+'/'+str(i)+'.mp4')
      success,image = vidcap.read()
      count = 0
      if success==False:
         print('check '+ folder_name)
      while success:
         if count%100==0:
            cv2.imwrite(path+'/'+str(i)+'/'+'frame'+str(i)+'_'+str(folder_name)+'_%d.jpg' % count,
image)
         success,image = vidcap.read()
         #print('Read a new frame: ', success)
         count += 1
```

➔ *Code for finding the shape of the images*

```
import cv2
import os
path = '/content/drive/DDD Project/frames_data/'
image_shapes = []
for i in [0,5,10]:
   print(i)
   folder_names = ['01','02','04','06','07','08','09','10','14','16','17','18','19','21','22','25','27','28','30']
   for folder in folder_names:
      print(folder)
      img = cv2.imread(path+str(i)+'/'+'frame'+str(i)+'_'+folder+'_100.jpg')
      shape = img.shape
      if shape not in image_shapes:
         image_shapes.append(shape)
         print(image_shapes)
print(image_shapes)
```

```
image_shapes = [(480, 720, 3), (720, 1280, 3), (2560, 1440, 3), (1080, 1920, 3), (1920, 1080, 3),
(1280, 720, 3), (352, 640, 3)]
img = cv2.imread('/content/drive/DDD Project/frames_data/5/frame5_30_1000.jpg')
sorted(glob('/content/drive/Shared drives/DDD
Project/frames_data/'+str(0)+'/frame'+str(0)+'_'+'04'+'_*.jpg'))
```

➔ *Code for rotating and removing non-essential frames in the images*

```
path = '/content/drive/DDD Project/frames_data/'
for i in [0]:
    folders = ['01']
    for f in folders:
        for im in sorted(glob(path+str(i)+'/frame'+str(i)+'_'+f+'_*.jpg')):
            img = cv2.imread(im)
            img_rotated = cv2.rotate(img,cv2.ROTATE_90_COUNTERCLOCKWISE)
            cv2.imwrite(im,img_rotated)

#resizing the image
folder_names = ['01','02','04','06','07','08','09','10','14','16','17','18','19','21','22','25','27','28','30']
path = '/content/drive/DDD Project/frames_data/'
for i in [0,5,10]:
    for folder in folder_names:
        images_path = sorted(glob(path+str(i)+'/frame'+str(i)+'_'+folder+'_*.jpg'))
        for image_path in images_path:
            img=cv2.imread(image_path)
            shape = img.shape
            if shape == (480,270,3):
                img =cv2.resize(img,(135,240),interpolation = cv2.INTER_AREA)
                cv2.imwrite(image_path,img)

folders = ['01','02','04','06','07','08','09','10','14','16','17','18','19','21','22','25','27','28','30']
total_folders = os.listdir('/content/drive/DDD Project/DDD')
for i in total_folders:
    if i not in folders:
        for k in [0,5,10]:
            rem_imgs = sorted(glob('/content/drive/DDD
Project/frames_data/'+str(k)+'/'+'frame'+str(k)+'_'+i+'_*.jpg'))
            if len(rem_imgs)!=0:
                for p in rem_imgs:
                    os.remove(p)
```

Appendix 8

More information on Convolutional Neural Networks architectures can be accessed through this link.
https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

Code for building the model of CNN architecture

```
model = tf.keras.Sequential()
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer =
'he_uniform',padding='same',input_shape = (240,135,3)))
model.add(Conv2D(64,(3,3),activation = 'relu',kernel_initializer = 'he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(128,(3,3),activation = 'relu',kernel_initializer = 'he_uniform',padding='same'))
model.add(Conv2D(128,(3,3),activation = 'relu',kernel_initializer = 'he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(512,(3,3),activation = 'relu',kernel_initializer = 'he_uniform',padding='same'))
model.add(Conv2D(512,(3,3),activation = 'relu',kernel_initializer = 'he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(512,activation = 'relu',kernel_initializer = 'he_uniform'))
model.add(Dense(128,activation = 'relu',kernel_initializer = 'he_uniform'))
model.add(Dense(3,activation = 'softmax'))
```

Appendix 9

Provide the ground truth for the model and save the image numpy arrays of the images

```
zero = sorted(glob('/content/drive/DDD Project/frames_data/0/*.jpg'))
five = sorted(glob('/content/drive/DDD Project/frames_data/5/*.jpg'))
ten = sorted(glob('/content/drive/DDD Project/frames_data/10/*.jpg'))
ground_truth = np.zeros((len(zero)+len(five)+len(ten),))
ground_truth[0:len(zero),] = 0
ground_truth[len(zero):+len(zero)+len(five),] = 1
ground_truth[len(zero)+len(five):len(zero)+len(five)+len(ten),] = 2


def data_reading(image_paths):
   data = np.zeros((len(image_paths),240,135,3))
   for i in range(len(image_paths)):
      print(i)
      data[i,:,:,:] = cv2.imread(image_paths[i])
   return data.astype('float32')/255

x_train = data_reading(x_train)
np.save('/content/drive/DDD Project/frames_data/x_train.npy',x_train)
x_test = data_reading(x_test)
np.save('/content/drive/DDD Project/frames_data/x_test.npy',x_test)
np.save('/content/drive/DDD Project/frames_data/y_test.npy',y_test)
np.save('/content/drive/DDD Project/frames_data/y_train.npy',y_train)
```

Appendix 10

Loading data from the numpy arrays and training the model

```
x_train = np.load('/content/drive/DDD Project/frames_data/x_train.npy')
x_test = np.load('/content/drive/DDD Project/frames_data/x_test.npy')
y_train = np.load('/content/drive/DDD Project/frames_data/y_train.npy')
y_test = np.load('/content/drive/DDD Project/frames_data/y_test.npy')

from keras.utils import np_utils
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Appendix 11

Test the model and evaluate the Accuracy

```
model.fit(x_train,y_train,batch_size=50,epochs=5,verbose=1)
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

****