READ ME FILE

PRITHISH KUMAR RATH (2K18/CO/262)

NILESH NISHANT (2K18/CO/235)

Drowsiness Detection in Drivers using Deep Learning Training

This is available in the **training folder**

About the project

This repository contains the files related to the project on frame-by-frame Drowsiness Detection in Drivers in videos using facial features. In this project, the aim is to extract frames from a video-based data and develop a CNN model that aims to detect fatigue in a driver using facial features. A CNN-based architecture is developed for the same.

Complete details regarding the project can be viewed from the detailed **Project Report**

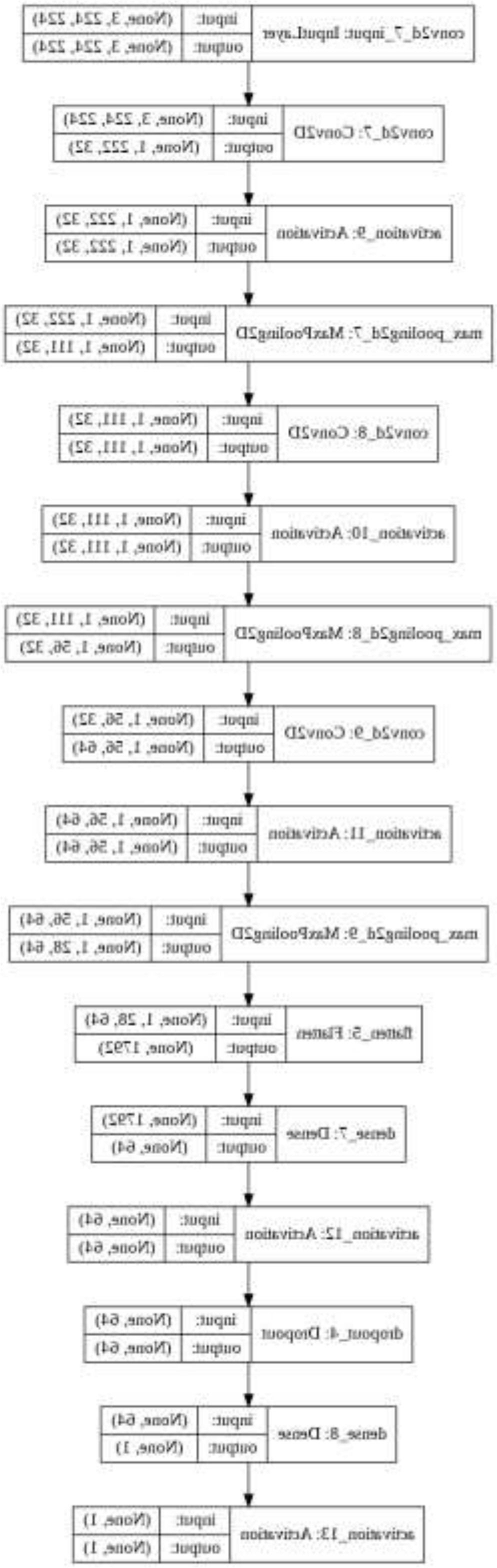
Dataset link: - http://cv.cs.nthu.edu.tw/php/callforpaper/datasets/DDD/

Models -

Two models are developed in this project, they are described as follows -

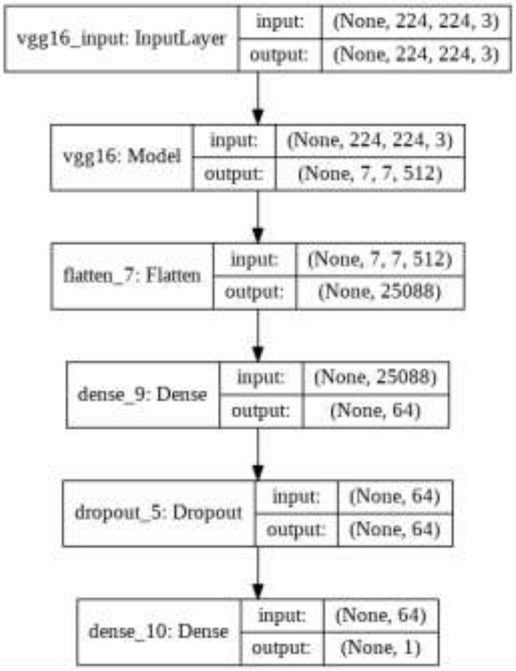
- **Baseline Model** - This is a standard model built and trained from scratch on the mentioned dataset to detect drowsiness. There is no concept of transfer learning involved.
- **Final Model** - This model is built in such a way that initial layers are that of the VGG-16 Model, pre-trained on imagenet weights. The initial layers are frozen for training, allowing us to fine-tune the later layers for effective feature recognition. The recognized features are then fed to a dense network for classification.

Baseline Model



conv2d_7_input: InputLayer | input: (None, 3, 224, 224) | output: (None, 3, 224, 224)
conv2d_7: Conv2D | input: (None, 3, 224, 224) | output: (None, 1, 222, 32)
activation_9: Activation | input: (None, 1, 222, 32) | output: (None, 1, 222, 32)
max_pooling2d_7: MaxPooling2D | input: (None, 1, 222, 32) | output: (None, 1, 111, 32)
conv2d_8: Conv2D | input: (None, 1, 111, 32) | output: (None, 1, 111, 32)
activation_10: Activation | input: (None, 1, 111, 32) | output: (None, 1, 111, 32)
max_pooling2d_8: MaxPooling2D | input: (None, 1, 111, 32) | output: (None, 1, 56, 32)
conv2d_9: Conv2D | input: (None, 1, 56, 32) | output: (None, 1, 56, 64)
activation_11: Activation | input: (None, 1, 56, 64) | output: (None, 1, 56, 64)
max_pooling2d_9: MaxPooling2D | input: (None, 1, 56, 64) | output: (None, 1, 28, 64)
flatten_5: Flatten | input: (None, 1, 28, 64) | output: (None, 1792)
dense_7: Dense | input: (None, 1792) | output: (None, 64)
activation_12: Activation | input: (None, 64) | output: (None, 64)
dropout_4: Dropout | input: (None, 64) | output: (None, 64)
dense_8: Dense | input: (None, 64) | output: (None, 1)
activation_13: Activation | input: (None, 1) | output: (None, 1)

Final Model



Training -

The results of training accuracies vs. epochs are shown below. Complete details of architecture and results can be viewed from the **Model & Training Results** folder.

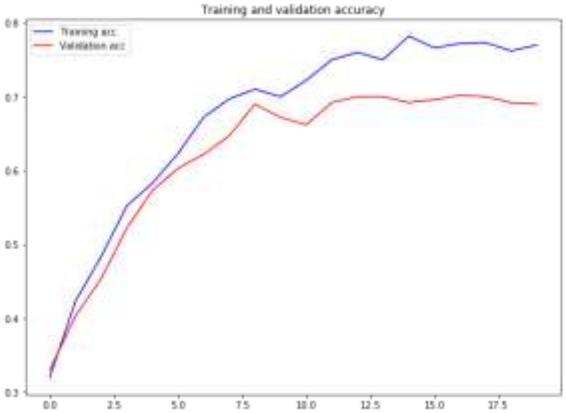


Figure 1 is of Baseline Model

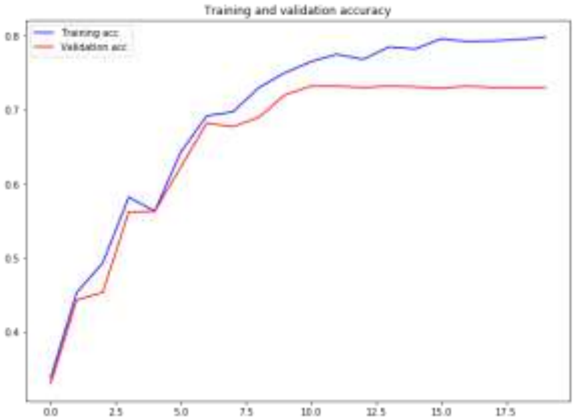


Figure 2 is of Final Model.

Results & Conclusions –

S. No.	Model	Accuracy
1	Baseline Model	68.6%
2	Final Model	73.2 %

- VGG16 model identifies features more effectively than the engineered baseline model because of more depth in the architecture, which allows it to identify both lower level and higher level features
- Transfer learning helps us to make the training faster as the initial layers are used to identify local features only, which are same for almost all models. So, freezing the initial layers allows us to speed-up the training remarkably
- Features like dropout, regularization, etc. allows us to make the model more robust
- Ultimately, due to many such improvements done in the final model, it significantly outperforms the baseline model in terms of detection

Steps to execute the code in this repository:

1. Install Anaconda Distribution and all relevant libraries
2. Get the licensed NTHU DDD Dataset
3. Run the code on jupyter notebook

Drowsiness Detection System (LIVE WEB CAM)

Installing and Configuring dlib:

We need to create an environment in order to install dlib, as it cannot be directly installed using pip. So, follow this commands in order to install dlib into your system if you haven't installed it previously. Make sure you have Anaconda installed, as we will be doing everything in Anaconda Prompt.

Step 1: Update conda

```
conda update conda
```

Step 2: Update anaconda

```
conda update anaconda
```

Step 3: Create a virtual environment

```
conda create -n env_dlib
```

Step 4: Activate the virtual environment

```
conda activate env_dlib
```

Step 5: Install dlib

```
conda install -c conda-forge dlib
```

If all these steps are completed successfully, then dlib will be installed in the virtual environment **env_dlib**. Make sure to use this environment to run the entire project.

Step to deactivate the virtual environment

```
conda deactivate
```

Running the system:

Step 1:

Download the repository into your system

Step 2:

Download the file **shape_predictor_68_face_landmarks.dat**(if not available) Make sure you download it in the same folder.

https://drive.google.com/file/d/14weZIclFncz8BMOmrkLp9PadLIccbSBa/view

Step 3:

Install all the system requirments by:

```
pip install -r requirements.txt
```

Step 4:

After the system has been setup. Run the command:

```
python app1.py
```

Step 5:

Open your browser and in the search bar type: **localhost:8000** as this port is mostly used by flask. In case, this port is not available in your system, flask will try to use another port. The port number will be displayed in the command prompt. So, type in the same port number in that case as: **localhost:<port_number>**.

After all these steps have been completed successfully, you will see a web page opening up in the browser. Now you are free to explore the system.

Working Details:

The basic thing about drowsiness detection is pretty simple. We first detect a face using dlib's frontal face detector. Once the face is detected , we try to detect the facial landmarks in the face using the dlib's landmark predictor. The landmark predictor returns 68 (x, y) coordinates representing different regions of the face, namely - mouth, left eyebrow, right eyebrow, right eye, left eye, nose and jaw. Ofcourse, we don't need all the landmarks, here we need to extract only the eye and the mouth region.

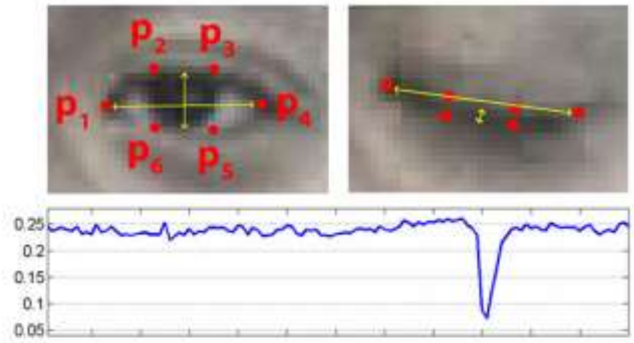
Now, after extraxting the landmarks we calculate the **Eye Aspect Ratio (EAR)** as:

```
def eye_aspect_ratio(eye):
        # Vertical eye landmarks
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])
        # Horizontal eye landmarks
        C = dist.euclidean(eye[0], eye[3])

        # The EAR Equation
        EAR = (A + B) / (2.0 * C)
        return EAR
```

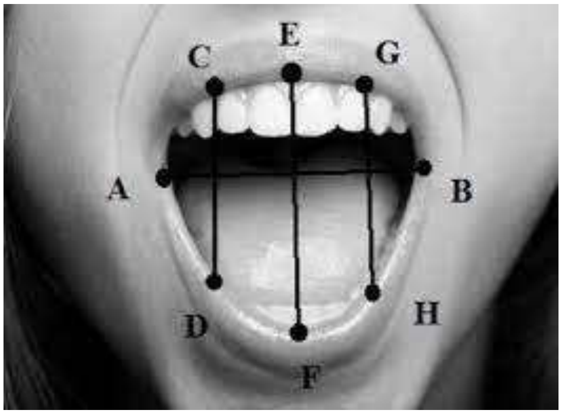
The eye region is marked by 6 coordinates. These coordinates can be used to find whether the eye is open or closed if the value of EAR is checked with a certain threshold value.

In the same way I have calculated **the aspect ratio for the mouth** to detect if a person is yawning. Although, there is no specific metric for calculating this, so I have taken for points, 2 each from the upper and lower lip and calculated the mean distance between them as:

```
def mouth_aspect_ratio(mouth):
        A = dist.euclidean(mouth[13], mouth[19])
        B = dist.euclidean(mouth[14], mouth[18])
        C = dist.euclidean(mouth[15], mouth[17])

        MAR = (A + B + C) / 3.0
        return MAR
```



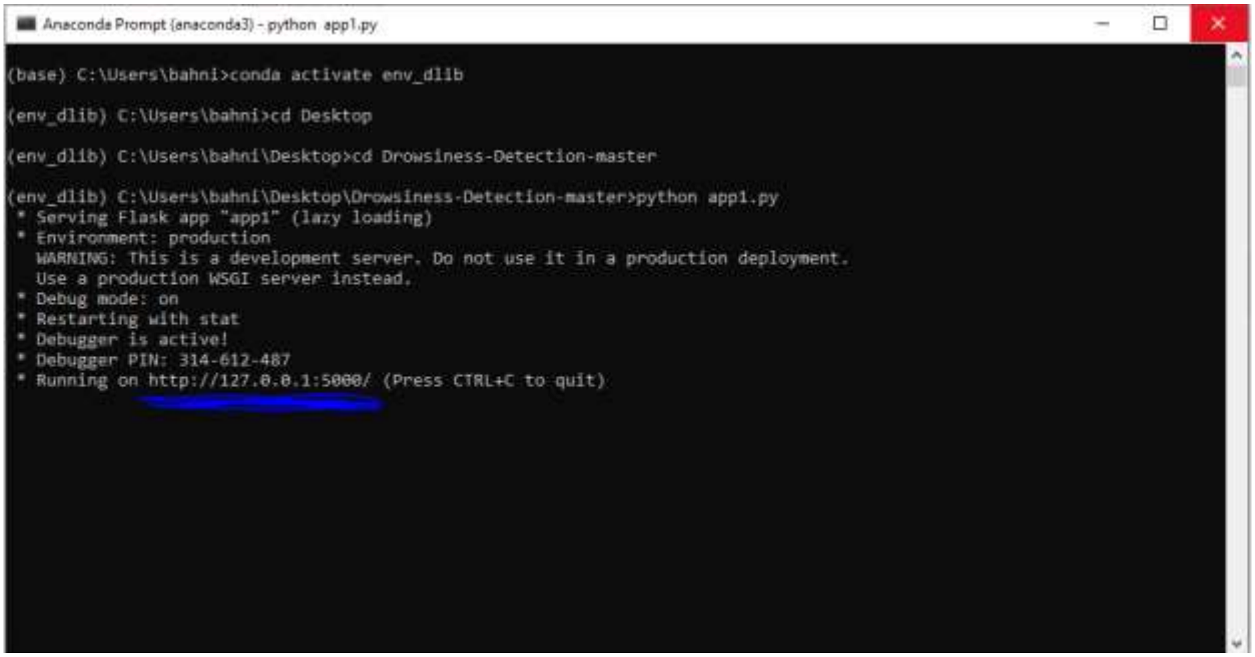
Eye Aspect Ratio (EAR)



Mouth Aspect ratio (MAR)

Results:

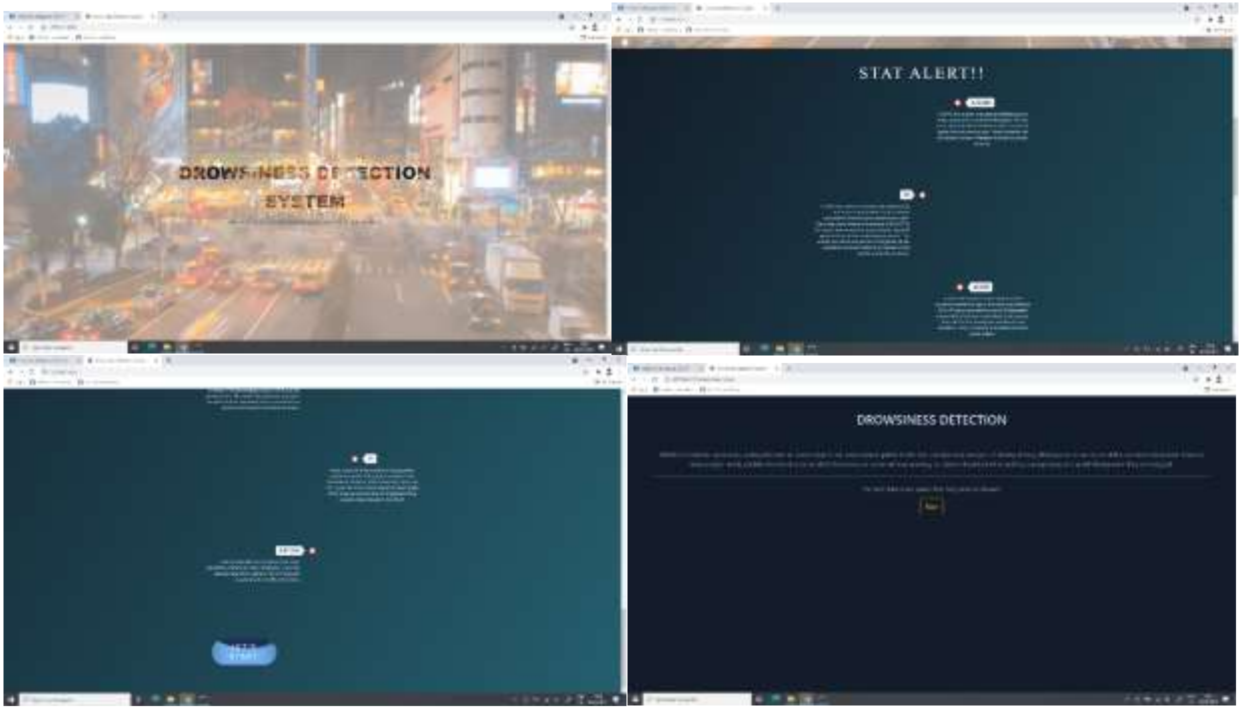
After running the program, the link is displayed coy the link on the browser



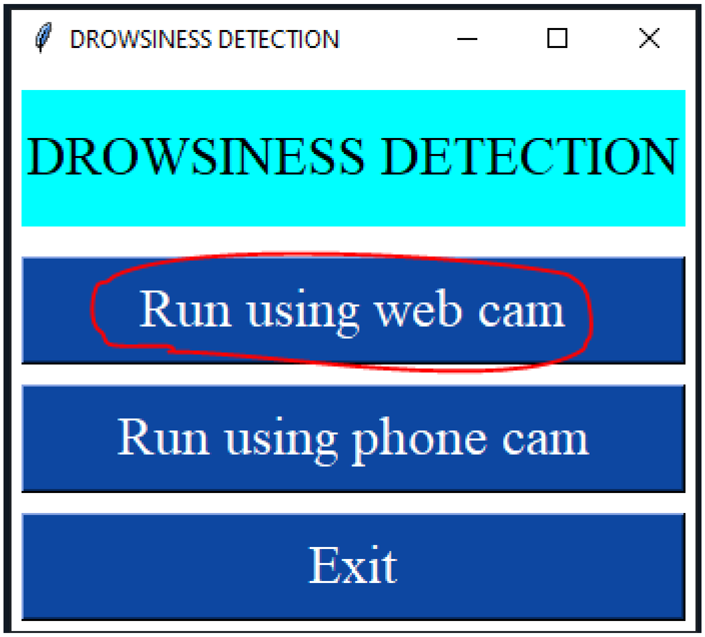
Anaconda Prompt (anaconda3) - python app1.py

```
(base) C:\Users\bahni>conda activate env_dlib

(env_dlib) C:\Users\bahni>cd Desktop

(env_dlib) C:\Users\bahni\Desktop>cd Drowsiness-Detection-master

(env_dlib) C:\Users\bahni\Desktop\Drowsiness-Detection-master>python app1.py
 * Serving Flask app "app1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 314-612-487
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

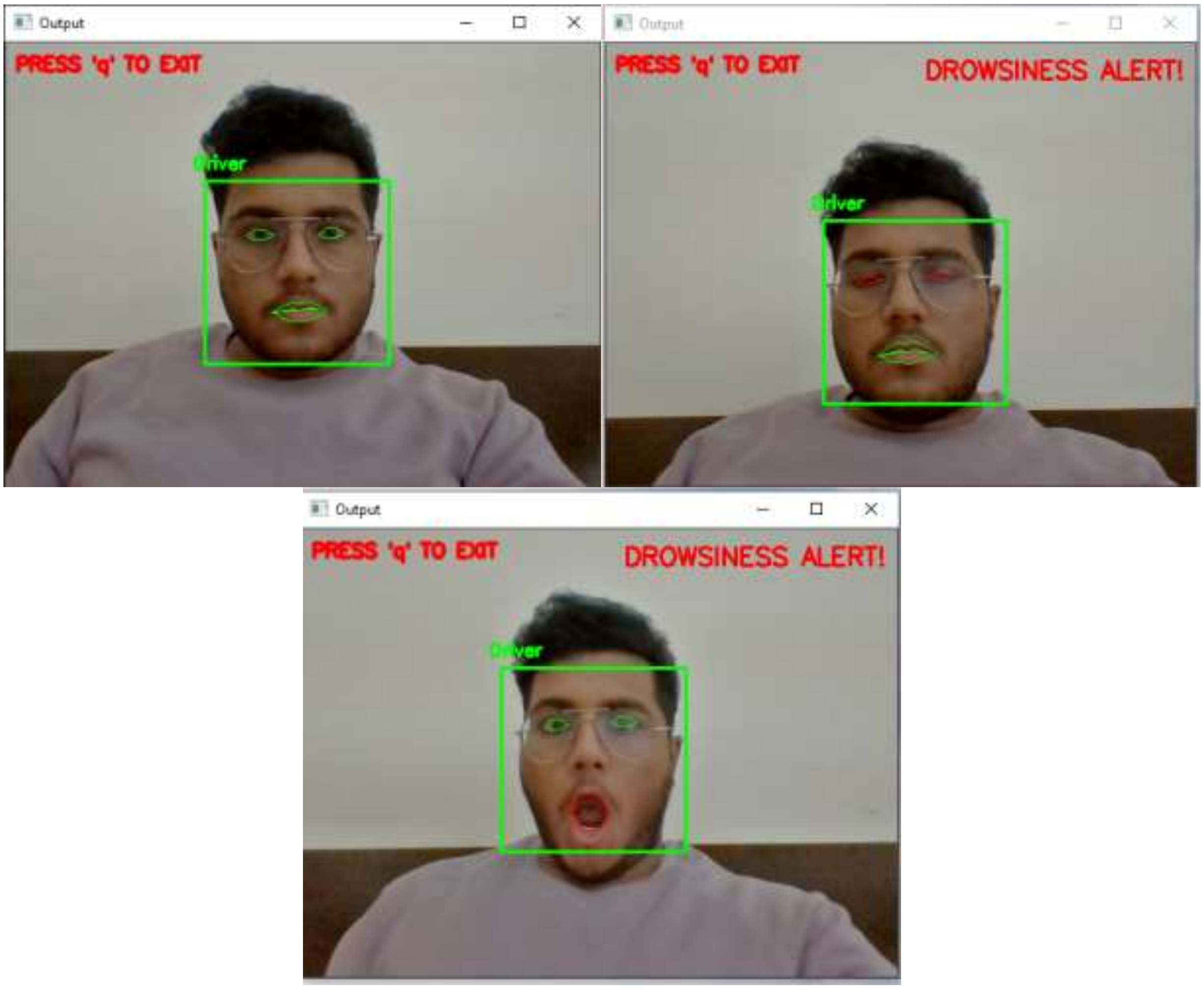
The GUI has been created using basic HTML, CSS and JavaScript and we have used Flask to render the python code into the website. Tkinter has also been used in order to make things simpler. It has 2 buttons: Run and Exit. The GUI looks like:



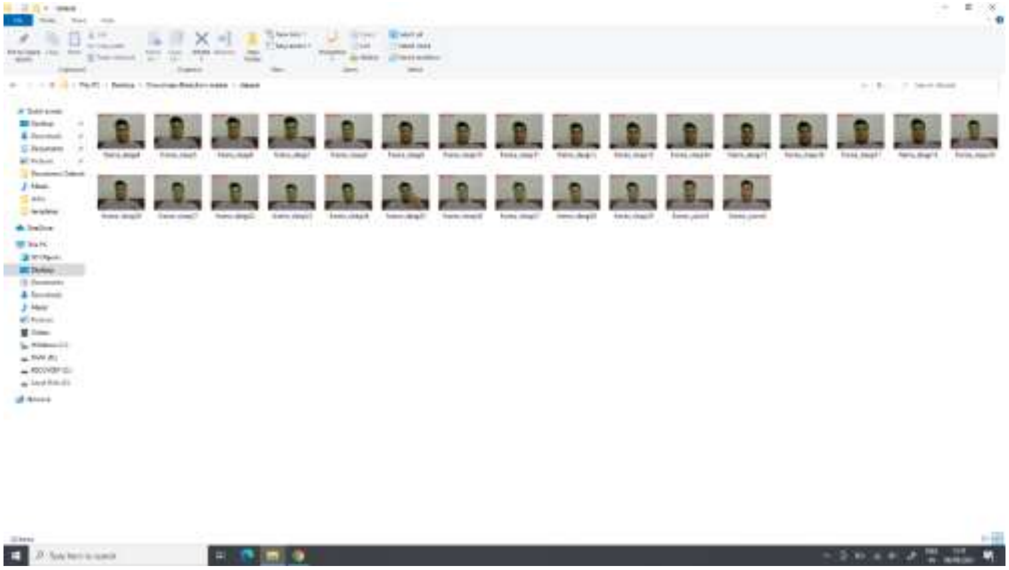
Click the web cam option



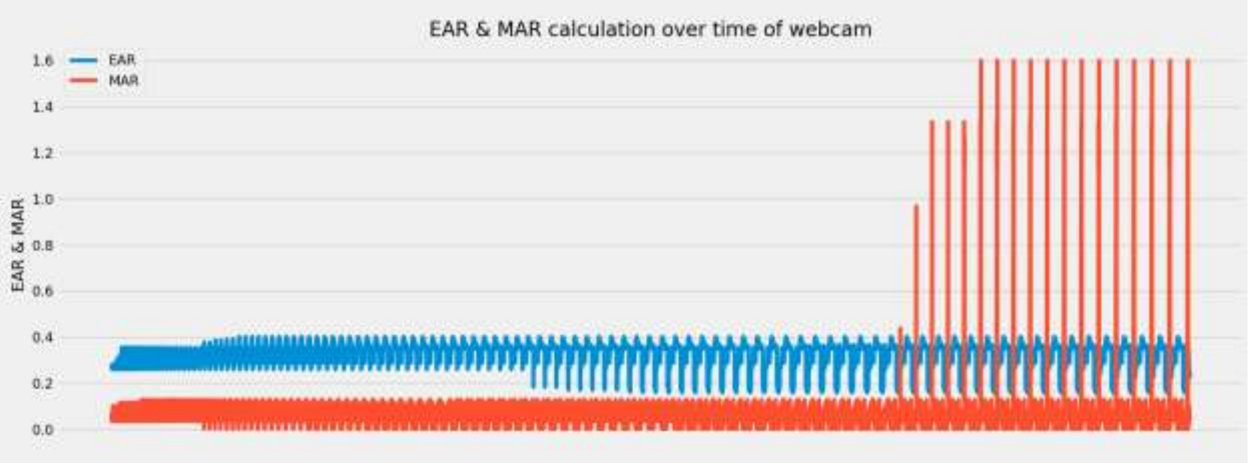
The outputs of the working system detecting drowsiness is shown as:



Also, in order to keep a proof of the moment when the person was sleeping or yawning, we kept a separate folder **dataset** where those frames are stored as:



Also, we have tried plotting the MAR and EAR graph Vs. Time in order to make the working clearer to the audience. The graph looks like:



Future Scope:

Any leads on hosting this Flask App will be useful.

References:

[1]Facial landmarks with dlib, OpenCV and Python: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/
[2]Eye blink detection with OpenCV, Python, and dlib: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
[3]Drowsiness Detection with OpenCV: https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/
[4]Real-Time Eye Blink Detection using Facial Landmarks: http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf