UNIVERSITÄT KLAGENFURT

FUNDAMENTALS OF IMAGE PROCESSING

USING IMAGE PROCESSING IN THE PROPOSED DROWSINESS DETECTION SYSTEM DESIGN

Driver Drowsiness Detection System

Author:
Ahmed Kafrana
Hazem Ibrahim

Date: February 28, 2022

1 Abstract

Background: Drowsiness is one of the underlying causes of driving accidents, which contribute, to many road fatalities annually. Although numerous methods have been developed to detect the level of drowsiness, techniques based on image processing are quicker and more accurate in comparison with the other methods. The aim of this study was to use image-processing techniques to detect the levels of drowsiness in a driving simulator.
Methods: The facial expressions, as well as location of the eyes, were detected by Viola-Jones algorithm. Criteria for detecting drivers' levels of drowsiness by eyes tracking included eye blink duration blink frequency and PERCLOS that was used to confirm the results.
Results: Eye closure duration and blink frequency have a direct ratio of drivers' levels of drowsiness. The mean of squares of errors for data trained by the network and data into the network for testing, were 0.0623 and 0.0700, respectively. Meanwhile, the percentage of accuracy of detecting system was 93.
Conclusion: The results showed several dynamic changes of the eyes during the periods of drowsiness. The present study proposes a fast and accurate method for detecting the levels of drivers' drowsiness by considering the dynamic changes of the eyes.

2 Introduction

Detecting the levels of drivers' drowsiness has a key role in reducing the number of fatal injuries in traffic accident. Recent statistics and reports show that 20 to 50 million people are killed or injured in car crashes all over the world. Assessments conducted by US NHTSA (National Highway Traffic Safety Administration) showed that 100000 car accidents occur every year for which the drivers' drowsiness is one of the principal contributor. National Sleep Foundation of USA declared that 54 % of adult drivers had driven during sleepiness and 28 % of these drivers had fallen asleep completely.
Having considered such statistics, devising the systems accurately to determine the levels of drivers' drowsiness is of prime importance in order to reduce the number of car accidents. Generally, there three kinds of systems exist to determine the levels of drivers' fatigue: techniques based on physiological signals, techniques based on driver's performance and techniques based on image processing.
In methods developed based on physiological signals, some electrodes are attached to the body to detect the signals from brain and heart. This method would be irritating and is considered as a nuisance to the drivers. In methods developed based on the driver's performance, much time is required to analyze driver's performance, which consequently leads to low accuracy. In some cases, while the driver falls asleep for a moment, the status of the vehicle does not change, so, the system is disturbed in detecting micro-sleeps. However, methods developed based on image processing are fast and precise to detect drivers' drowsiness. Fatigue and drowsiness lead to some apparent signs on driver's face. Interpreting such signals are the principals of

the methods developed based on the image processing. In many cases, the first step in image processing is to recognize the facial zone from the received images. Then, the parameters related to drowsiness such as eyes are recognized and their changes are investigated to detect the levels of sleepiness. By photographing the driver and image processing, the visual signals of sleepiness could be detected. Amongst various facial features, eyes are relatively of more importance and many studies have been conducted on the processing of the condition of the eyes. For instance, some factors like PERCLOS (Percentage of Eye Closure), duration of eyes closure and the number of blinks were utilized by IR Illuminator to determine the vigilance level. Drowsiness was detected merely based on PERCLOS. A small camera conducted the observing test detecting the levels of drowsiness by the number of eyes blinks and accuracy of 98 %. Regarding the importance of detecting the levels of drowsiness precisely without irritating the driver and considering the fact that almost no real time techniques for such purpose have been mentioned in the previous studies, we conducted an investigation of drowsiness by image processing of professional drivers and observed the movements of their eyes within the sequence of images.

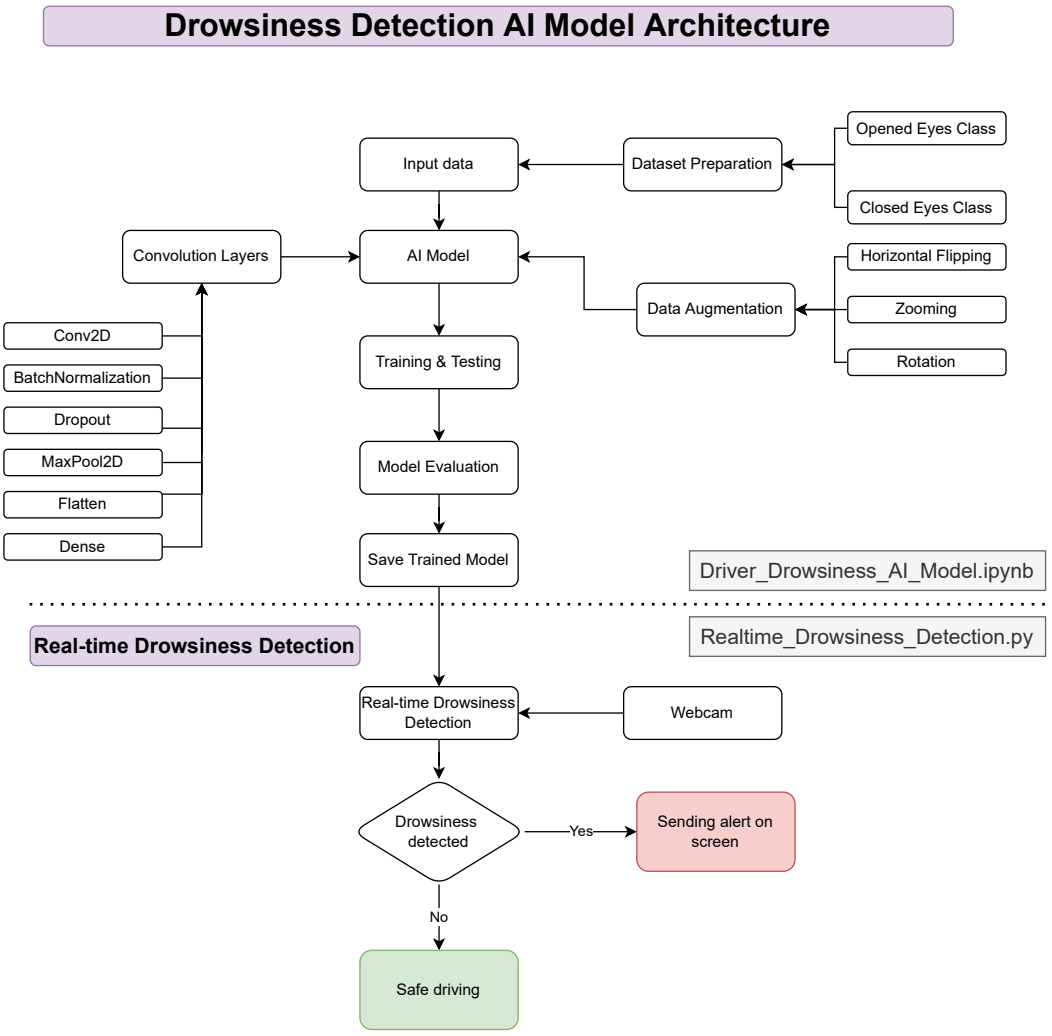
3 Data Description

The project uses the Drowsiness dataset present on the Kaggle platform. The dataset is present on this link. The original dataset contains four classes for classifying images into Open Eyes, Closed Eyes, Yawning, or No-Yawning. However, this project's scope is to classify drowsiness based on whether the eyes are closed or open. So, I will be using only two classes of the dataset. Characteristics of the dataset are as follows:

- The dataset contains a total of 1452 images in two categories.
- Each category has 726 images.
- The dataset is already balanced, so no need to balance the dataset.
- Class Labels — 'Open Eye' and 'Closed Eye'.
- Class Labels were encoded such that 0 represents Open Eye and 1 illustrates Closed Eye.

4 System architecture

4.1 Flow Diagram



4.2 Pseudo Code

```
# Dataset
# 2 Classes for Drowsiness Detection

Dataset = [[OpenedEyes], [ClosedEyes]]

# Input data
TrainDataset = Dataset.Split[0]
TestingDataset = Dataset.Split[1]

# Data Visualization
```

```
TrainDataset.show()
TestingDataset.show()

# Define the Model Architecture
# Adding the layers
Model = Sequential()
Model.Add(Conv2D)
Model.Add(BatchNormalization)
Model.Add(Dropout)
Model.Add(MaxPool2D)
Model.Add(Flatten)
Model.Add(Dense)

# Define the Optimizer
Optimizer = Adam(learning_rate)

# Data Augmentation
Dataset = Augmentation(Rotation, Zoooming, Flipping)

# Model Training
Model.fit(BatchSize, Epochs)

# Plotting Losses and Accuracy
Plot(Model.history(TrainAccuracy))
Plot(Model.history(TestAccuracy))

# Model Evaluation
Model.Evaluate(TestingDataset)
ConfusionMatrix(TestingDataset)

# Saving Trained Model
Drowsiness_AI_MODEL = Model.save('TrainedModel.h5')

# Real-time Drowsiness Detection
# Run Realtime_Drowsiness_Detectioin.py
Video = Camera.VideoCapture
Image = Video.ReadImage

# Prediction of Drowsiness for real-time driving
DrowsinessDetection = Drowsiness_AI_MODEL.predict(Image)
```

5 Performance and Model Evaluation

5.1 Performance

The following Performance Metrics are used:

- Loss vs Number of Epochs Plot
- Accuracy vs Number of Epochs Plot
- Classification Report
- Confusion Matrix

The results of the metrics mentioned above are as follows:

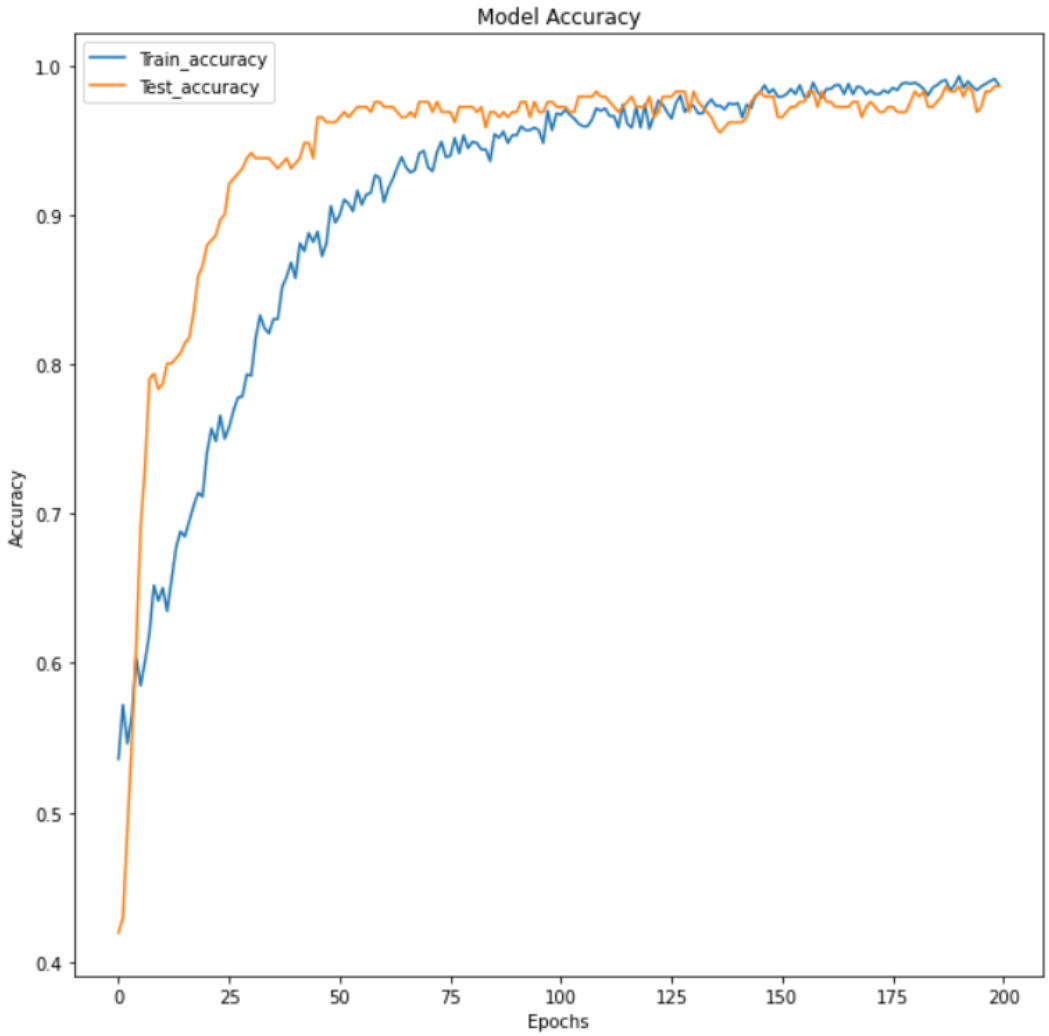


Figure 1: Model Accuracy

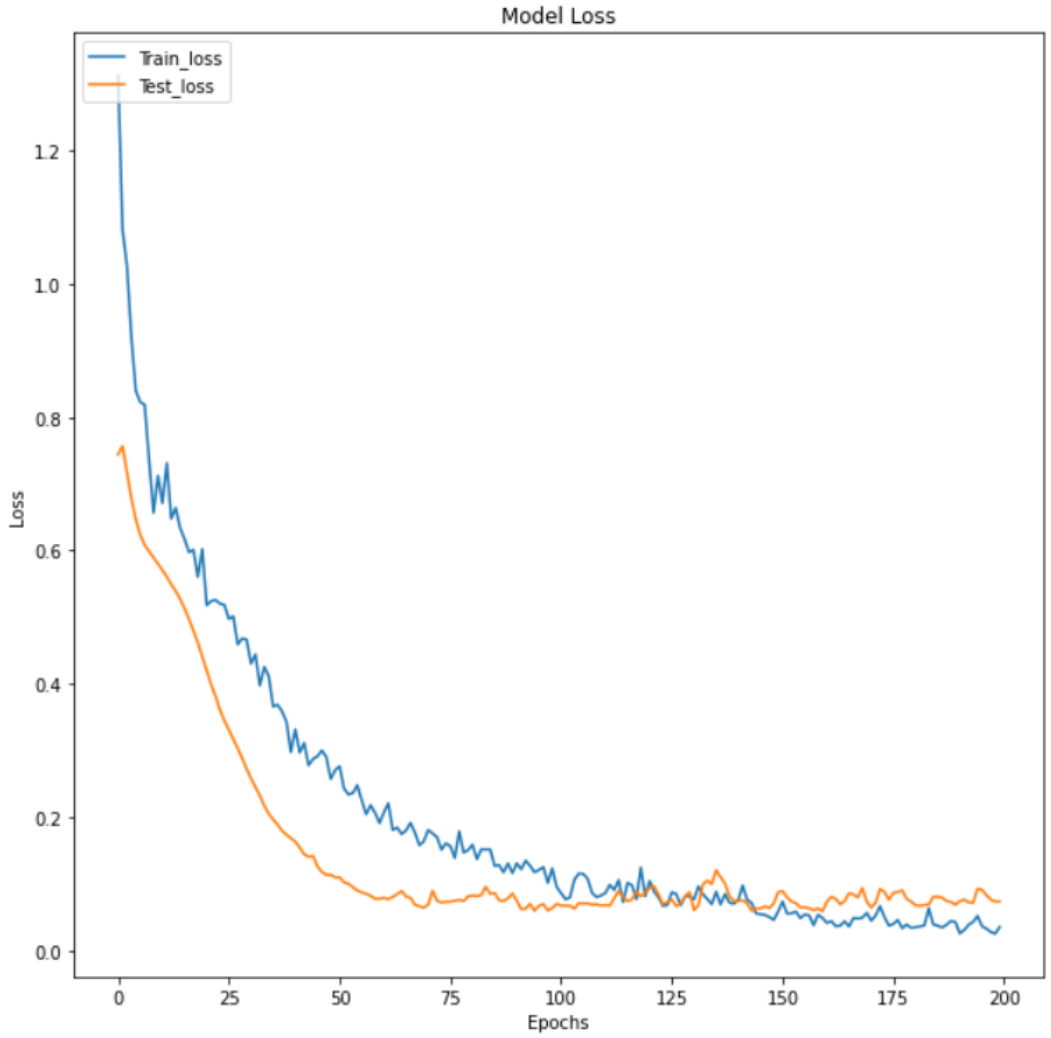


Figure 2: Model Loss

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	169
1	0.98	0.99	0.98	122
accuracy			0.99	291
macro avg	0.98	0.99	0.99	291
weighted avg	0.99	0.99	0.99	291

Figure 3: Classification Report

Confusion Matrix:

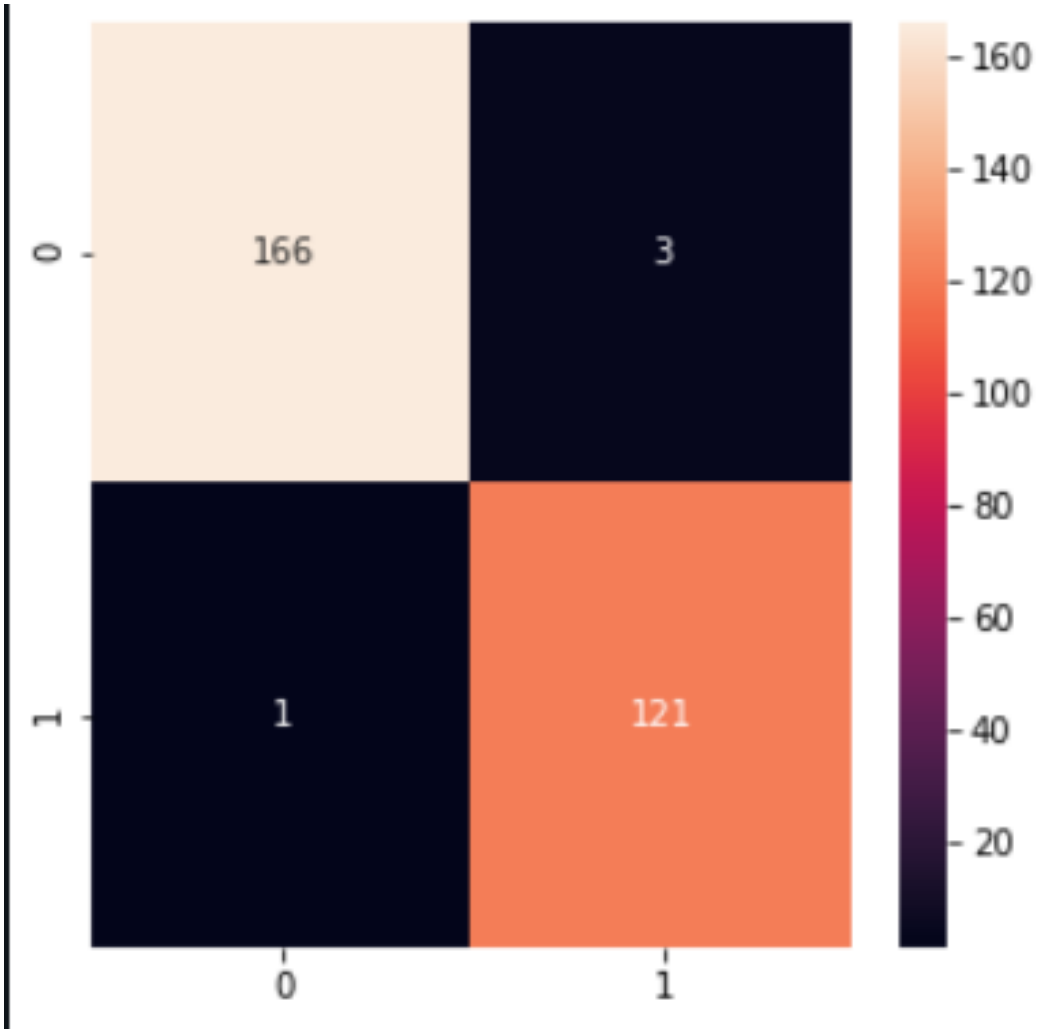


Figure 4: Confusion Matrix

5.2 Model Evaluation

The predictions of the model on images of eyes can be seen in the following pictures:

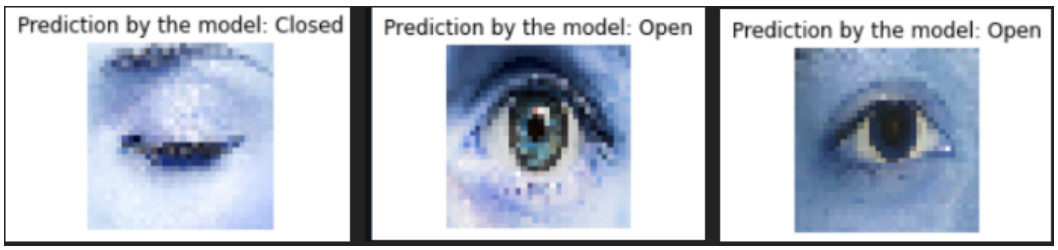


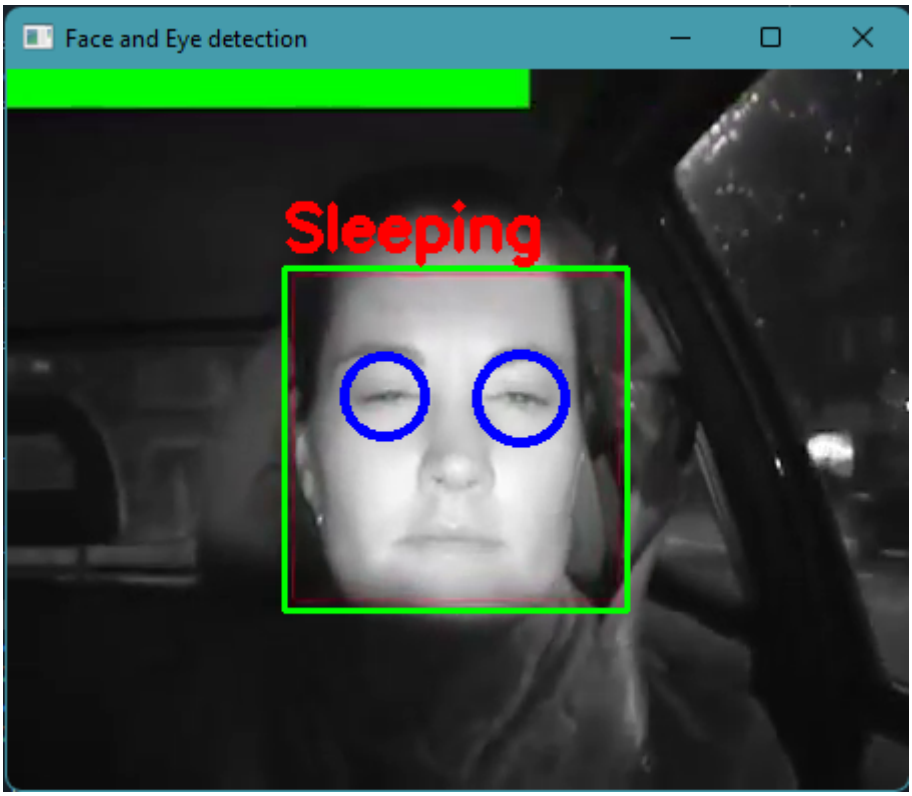
Figure 5: Testing the model

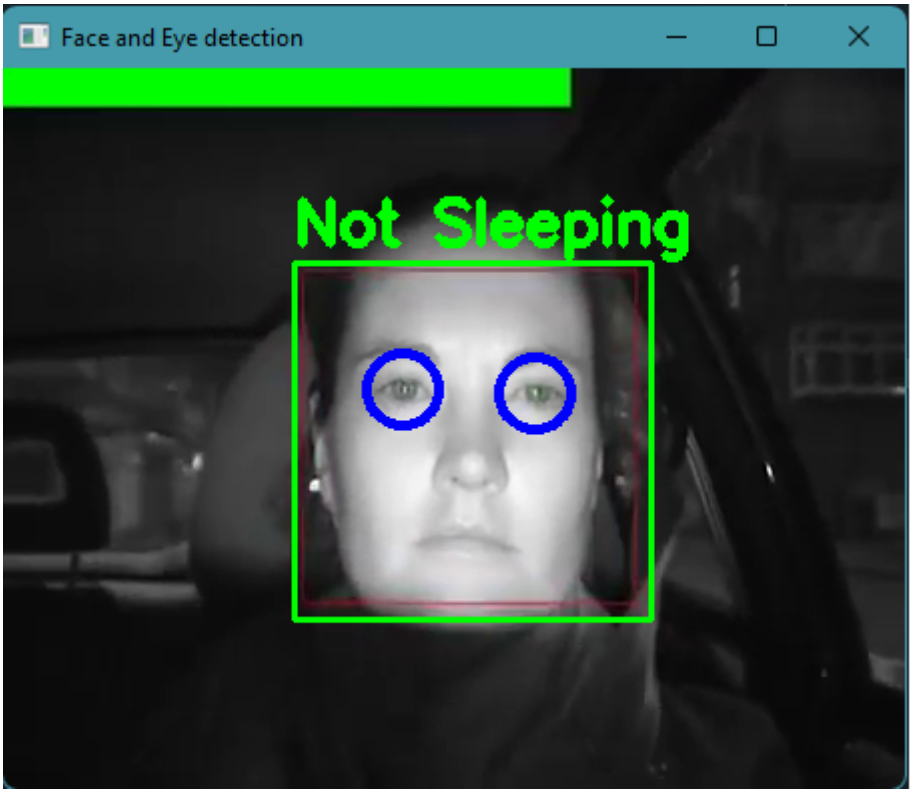
6 Results obtained and their interpretation

This project's last and final step is to build a pipeline for making predictions on full-face images. The channel includes Face detection, eyes detection, preprocessing the ROI of the image, passing to the model for prediction and displaying results on the image.
the results can be viewed as follows:

```
## Here were we get the faces in the frame
faces = faceCascade.detectMultiScale(gray,
                                     scaleFactor=1.1,
                                     minNeighbors=5,
                                     minSize=(60, 60),
                                     flags=cv2.CASCADE_SCALE_IMAGE)
for (x,y,w,h) in faces:
    cv2.rectangle(frame, (x, y), (x + w, y + h),(0,255,0), 2)
    faceROI = frame[y:y+h,x:x+w]
    eyes = eyeCascade.detectMultiScale(faceROI)
    eyes_ROI = []
    for (x2, y2, w2, h2) in eyes:
        ## calculating the center of the eye
        eye_center = (x + x2 + w2 // 2, y + y2 + h2 // 2)
        ## calculating the radius of the eye
        radius = int(round((w2 + h2) * 0.25))
        ## drawing the cricle on the frame
        frame = cv2.circle(frame, eye_center, radius, (255, 0, 0), 4)
        ## Cropping the eye from the whole frame
        eye = org_img[eye_center[1]- radius:eye_center[1] + radius,eye_center[0] - radius:eye_center[0] + radius]

        ## preparing the cropped eye to be provided to the model
        eye = cv2.resize(eye, (32, 32))
        eyes_ROI.append(eye)
        eye = np.array(eye)
        eye = np.expand_dims(eye, axis=0)
        ## The trained model is used to predict the drowsiness
        prediction = model.predict(eye)
        color = 0
        if(prediction[0][0] > prediction[0][1]):
            #print("Danger!")
            message = 'Sleeping'
            color = RED
        else:
            message = 'Not Sleeping'
            #print("Safe")
            color = GREEN
        ## Put the text on the window itself
        cv2.putText(frame, message, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1.0,color, 3)
    ## Show the cropped eyes together
    if(len(eyes_ROI) > 0):
        ## concatinating the images into one
        eyes_cropped = np.concatenate(eyes_ROI, axis=1)
        cv2.imshow('Croppped Eyes', eyes_cropped)
```


Face and Eye detection
Sleeping


Face and Eye detection
Not Sleeping

7 Conclusion

Driver Drowsiness is a significant reason for thousands of road accidents all over the world. Driver drowsiness detection is a car safety technology that helps prevent accidents caused by the driver getting drowsy. The project aims at providing a solution of Driver Drowsiness Detection using CNN and image processing. The project aimed at optimizing the model to limit the number of parameters under 250k for easy deployment on edge devices. Thus effectively bringing AI out on edge — in actual and physical real-world use cases.