1. INTRODUCTION

Driving requires the execution of a certain set of tasks while maintaining situational awareness and making swift, informed decisions. In order to process the obvious indications while driving, you need to maintain situational awareness. Therefore, keeping one's attention state intact when driving is really important.

Human answer period is restricted by fatigue, making it intolerable to drive safely. According to a study transported by “Union minister for artery transport and highways Our reputable Minister , in his meaning in the 2021 report, has noticed, “During the period 2021, a total number of 4,12,432 boulevard accidents have been stated in the country, demanding 1,53,972 lives and producing harms to 3,84,448 persons.”

These factors have greatly accelerated progress in the research of driver state monitoring, especially in the calculation of driver workload, the identification of driver activity, the identification of secondary tasks, and the detection of driving style. Many methods have been employed in the past. Several global corporations have used some of these techniques for driver assistance.

Yawns, sluggish reflexes, eyelid closures, a loosened steering grip, etc. are all signs of fatigue. One sign may not be used alone and accurately for tiredness identification since humans can display various symptoms and levels of exhaustion.

The suggested technology aims to increase overall transportation security by lowering collisions caused by tired and exhausted drivers. This leads to more accidents now than ever before. Driver fatigue and drowsiness are thought to be indicated by a variety of physical and facial signs, including sleepy eyes and yawning. These traits demonstrate the driver's unfit status. To detect exhaustion, the Eye Aspect Ratio (EAR) compares the horizontal and vertical lengths of familiar eye landmarks.

2. LITREATURE REVIEW

2.1 EXISTING METHODS OF DROWSINESS DETECTION

Mathematical models, shallow models, and deep learning models are used in driver tiredness detection systems.

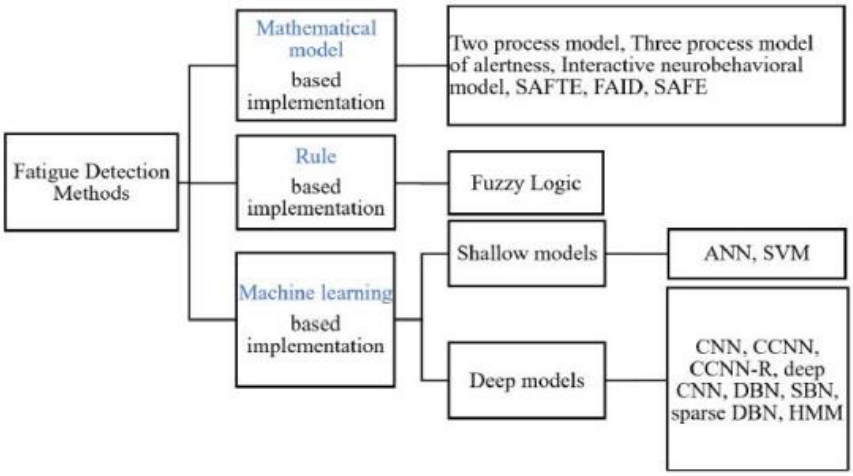


Figure. 2.1 Fatigue Detection Methods

2.1.1 MATHEMATICAL MODELS

One of the less difficult techniques to implementing an expert system is to do it using a rule-based framework. Fuzzy Inference Systems (FIS) are superior than rule-based systems when dealing with complex systems. Decisions in FIS are made using a fuzzy rule base and fuzzy membership functions, both of which are extensively used techniques in many fields. FIS provides in-built expert knowledge and uses an IF-THEN rule to translate inputs to outputs. The driver's condition (fit, fatigued, or hazardous) was inferred via a fuzzy algorithm that used the driver's mouth and eye states as input. There were three states for the eyes (blink, drowsy, and asleep) and two for the lips (normal, yawning). In a more recent study, researchers used features including the driver's eye and mouth states to determine their weariness using a two-layered FIS. Various other models are developed in continuation of this which incorporate various combinations of possible inputs to predict fatigue.

2.1.2 RULE BASED MODEL

Among the less difficult methods for implementing expert systems, rule-based implementation is thought to be one. Simple rule base systems should not be used for complex systems; instead, fuzzy inference systems (FIS) should be used. A fuzzy rule base and fuzzy membership functions are used in the widely used FIS approach to make judgements across many different domains. FIS employs the IF-THEN base rule and offers in-built expert knowledge when mapping inputs to outputs. One such technology for detecting driving inattention was a fuzzy system that utilised the driver's mouth and eye states as input to evaluate if the motorist was attentive, tired, or dangerous. The lips was categorised as regular and yawning, and the eyes as blinking, tired, and asleep. In a more recent study, researchers used features including the driver's eye and mouth states to determine their weariness using a two-layered FIS.

Advantages of using a FIS are:

- offers a lot of leeway and may be put to use in a wide variety of vision-based contexts.
- Provides data-free training as well as parallel processing thanks to all rules being implemented concurrently.
- have the capacity to learn by assimilating new data and rules.

2.1.3 MACHINE LEARNING BASED MODELS

Data-driven machine learning solutions are developed using substantial driving data from the lab and the road. Based on the depths of representation and the method of feature extraction, the category may be roughly split into shallow and deep models.

- **SHALLOW MODELS**

  Shallow models have a modest level of complexity and a fair capacity to forecast. Little layers and little training data are needed for shallow models, although preset discriminative features are necessary. Popular techniques in this area include support vector machines and ANNs with one hidden layer (SVM). To categorise a driver as alert, sleepy, or hazardous, many strategies use ANNs and SVMs that have been trained on parameters like PERCLOS (% of eyelid closure), the driver's EEG and ECG signals, and other features. But, as was already indicated, these methods employ precomputed characteristics. Deep Learning models are utilised since it is impossible to correctly predict these aspects.

- **CNN (DEEP LEARNING) MODELS**

Instead of focusing on one single job at a time, deep learning models take a more general approach to machine learning. In contrast to shallow models, deep models may learn to extract the characteristics from the training data. The main application of convolutional neural networks (CNN)-based models is the identification of driver tiredness. Our approach is based on this technology, which employs deep learning and 3D convolutional neural networks to analyse a video input of a driver's face and extract features that indicate whether the driver is awake or asleep.

- **OpenCV (DEEP LEARNING) MODELS**

Machine learning methods known as "deep learning models" use learning representations of the data rather than task-specific approaches. Deep models can extract the features from the training data, as opposed to shallow models, which cannot. It is feasible for a computer vision system to detect drowsy driving in real-time video and sound an alert. The "shape predictor face landmarks" that identify the critical facial cues make this recognition rapid and easy. Thus, the Open CV module also offers the freedom for identifying such grave and important detection.

FEATURES USED FOR DROWSINESS DETECTION

Various features incorporated in drowsiness detection are given in the following figure.

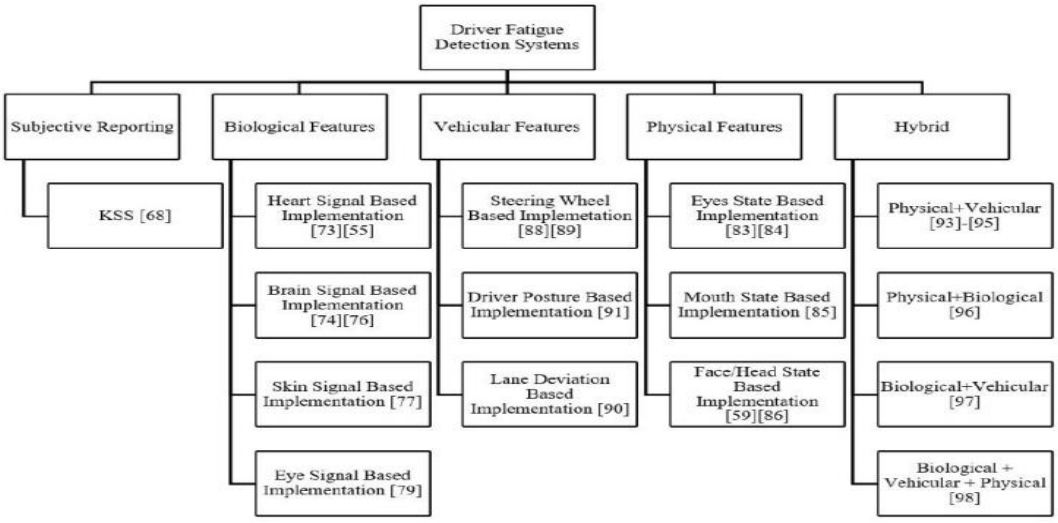


Figure. 2.1.3 Fatigue Detection Features

These features are used based on various requirements, and possess certain traits, which are, in brief, given in Table I.

By examining numerous criteria related to the features that will be employed, we can observe that the biological traits are invasive in nature, which will negatively affect the driver's ability to drive and other aspects of their life. Also, it cannot be used in real time, and the installation fees are exorbitant.

Vehicle characteristics, on the other hand, are not very precise because they greatly depend on the environment, for example. The steering wheel will move around a lot on a twisting road, yet a tired driver can still be identified by this. Moreover, some of the approaches' high installation costs make them less practical for real-time use.

When it comes to appearance, however, these drawbacks do not exist. Even while this method relies to some extent on the ambient light, it has additional benefits that outweigh this drawback. To that end, this research relies on external factors to ascertain the driver's condition.

TABLE I: COMPARISON OF VARIOUS FEATURES

Category	Signal	Parameter	Contact	Cost	Real-time Applicability	Limitations
Biological Features	Brain	EEG	Yes	Low	No	Extremely Intrusive Prone to human movement
	Heart	ECG	Yes	High	No	
	Skins	EMG	Yes	High	No	
Vehicular Features	Steering	SWA	Yes	High	Yes	Driver and Environment Dependency
	Lane	Lane Deviation	No	Low	Yes	
	Posture	Pressure	Yes	High	Yes	
Physical Features	Eyes	PERCLOS, Blink	No	Low	Yes	Illumination and Background Dependency
	Mouth	Yawn	No	Low	Yes	
	Face	Nod	No	Low	Yes	
	Nose	Structure	No	Low	Yes	

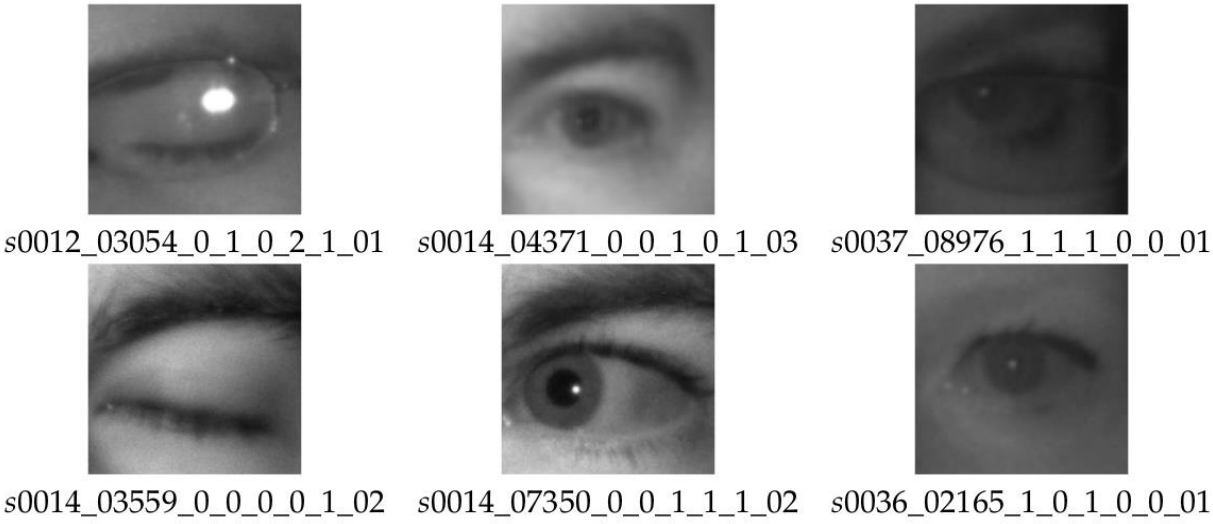
Data

We use the Driver Drowsiness Detection (DDD) dataset, which was presented for the first time at the 2016 Asian Conference on Computer Vision for academic research. Video was shot at a resolution of 480 x 640 pixels, with a frame rate of 30 fps during the day and 15 fps during the night. Videos of each person were captured under five different experimental circumstances.

1. without glasses
2. with glasses
3. with sunglasses
4. without glasses at night
5. with glasses at night

Drowsy and non-drowsy driving states have been assigned to video clips depicting simulated yawning, nodding, glancing away, conversing, laughing, and shutting eyelids, as well as usual driving. There are 34 people in the training set, 12 in the evaluation set, and 22 in the test set.

Figure 3.1. An examples of image annotations of the proposed dataset:



The example images from this dataset of open and closed eyes are shown below.

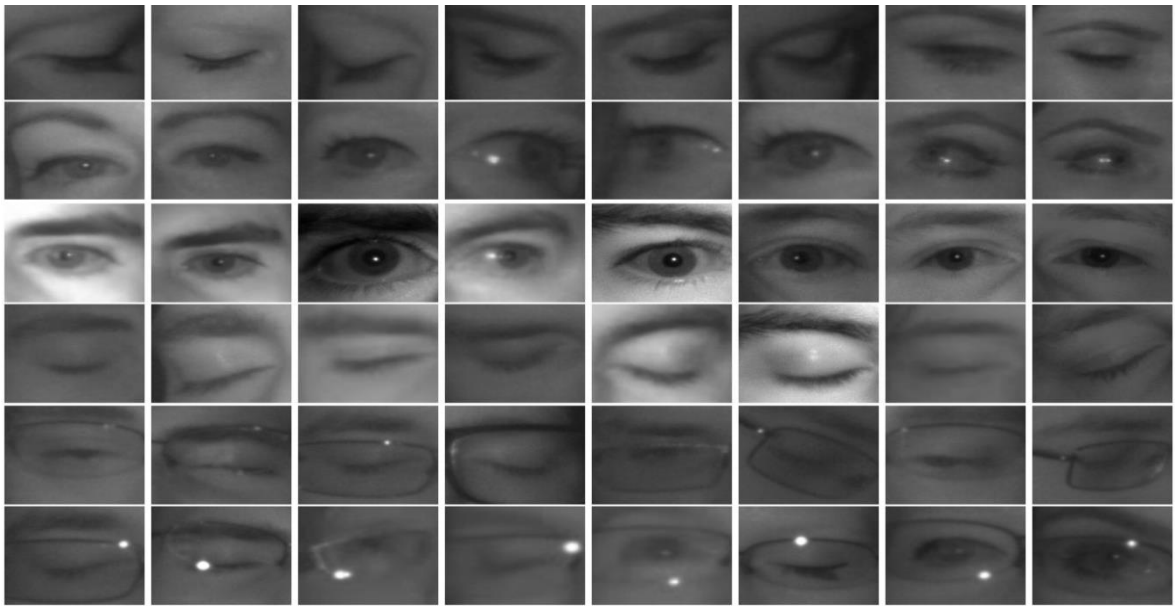


Figure 3.2 What our Data looks like (Drowsy and Non-drowsy case, respectively

As part of this study, the assessment dataset (1.5 hours of video) was utilised for model validation whereas the testing dataset (a total of 8.5 hours of video) was not. Nighttime films had their frame rate increased from 15 fps to 30 fps so that they would be consistent with the rest of the movies in the collection. Videos were scaled down from 480 640 to 240 320 to save training pre-processing time and disc space requirements. For both validation and training purposes, the video files were cut into 10-frame and 100-frame sequences. This resulted in 9094 training records with 100 frames and 17318 validation records with 10 frames. As a trade-off for better read performance, just a small portion of the 10-frame sequences from the source films are utilised for training.

PRE-PROCESSING & DATA AUGMENTATION

Due to the small amount of training data available in the DDD dataset, numerous pre-processing processes were applied to diversify the inputs to the neural network during training. These pre-processing stages were designed specifically for the problem of sleepiness detection and boost the model's resilience when used in practise. As a result, training precision is improved and overfitting is reduced.

Following pre-processing/ data augmentation steps were taken –

- At 30 frames per second, every movie is rendered into still pictures.
- Each image is tagged with its current level of drowsiness. (i.e. drowsy or not drowsy)
- The average luminance of each frame is adjusted. (by dividing from the largest value: 255).
- Some of the pictures have been turned 90 degrees at random.
- There is a horizontal flip in several of the images.
- Images are distorted by a factor of 0.2 in zoom, shear, and shift (height and width).
- A resampled size of 224 by 224 by 1 was used. This method makes the model robust against changes in viewpoint (horizontal and vertical), magnification, and facial structure.

Table 3.1 **Data after pre-processing**:

Data after pre-processing:	**No. of Frames**	**No. of Categories**
Training Set	2,73,266	2
Test Set	93,299	2

4. MODELS

In this project, we have adopted the approach of using the model having the following specifications:

- Method: Deep Learning Model
- Type: CNN, Open CV
- Pretrained on: ImageNet VGG16
- Category: Physical Feature detection

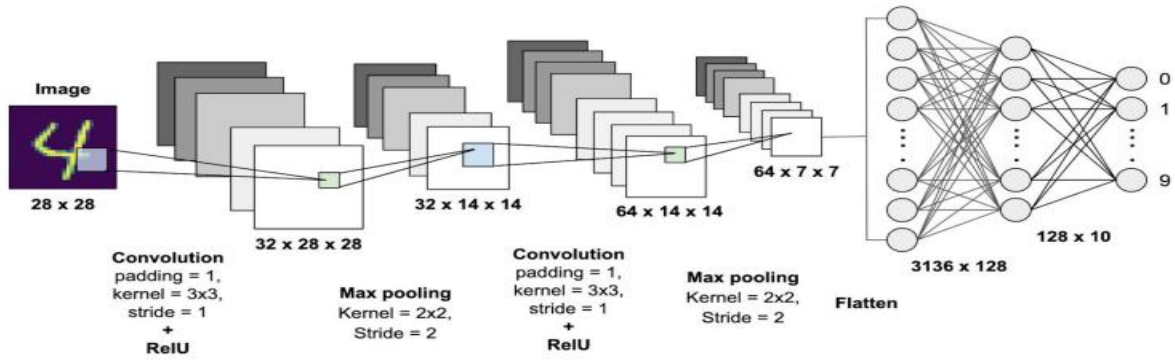


Figure 4. Model Architecture

In the following few sections, we shall explore the meaning and working of all these specifications.

4.1. CNN

Convolutional neural networks (ConvNets) are a kind of Deep Learning algorithm that can take in an input image, give significance (learnable weights and biases), and then classify the objects inside the image. ConvNet requires far less preparation than competing classification methods. ConvNets can learn filters/characteristics on their own with enough training, while in primitive approaches they must be hand-engineered.

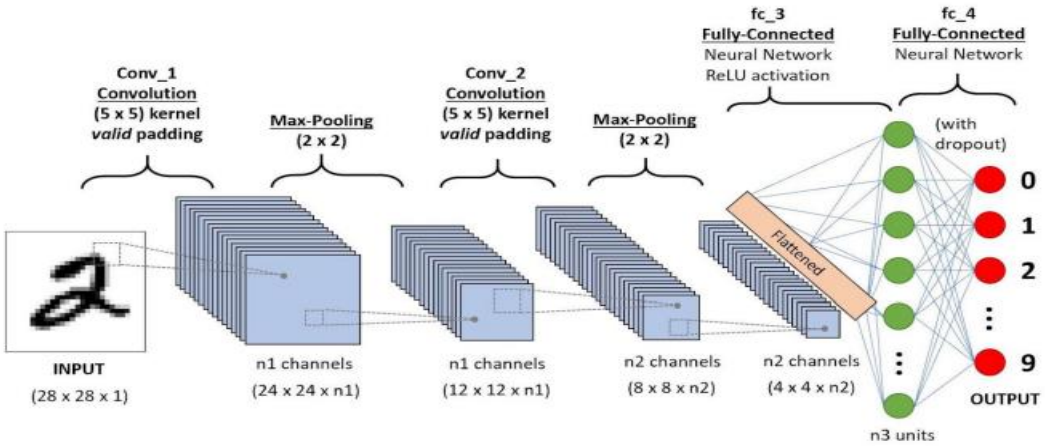


Figure 4.1. CNN sequence to classify handwritten digits

ConvNets' architecture mimics the human brain's neuronal connection pattern and takes design cues from the Visual Cortex's structure. The Receptive Field is the portion of the visual field to which each individual neuron is sensitive. To cover the whole visual field, a number of these fields overlap one another.

We have prepared two models for our purpose:

- **A Baseline Model Trained from Scratch**

- **A Fine-tuned VGG16 Model pretrained on ImageNet dataset.**

4.1.1. BASELINE MODEL :

A CNN model with three convolutional layers and the" Relu" activation function has been created with the intention of constructing a birth model in order to acquire a model that has been trained entirely from scrape(i.e. without any pretraining). After each Conv subcaste, these layers are followed by Max- pooling layers. Two thick, fully linked layers are coupled at the top, and they use a" Sigmoid" activation function to determine if a frame is sleepy or awake. The structure of the Baseline Model has been shown below:

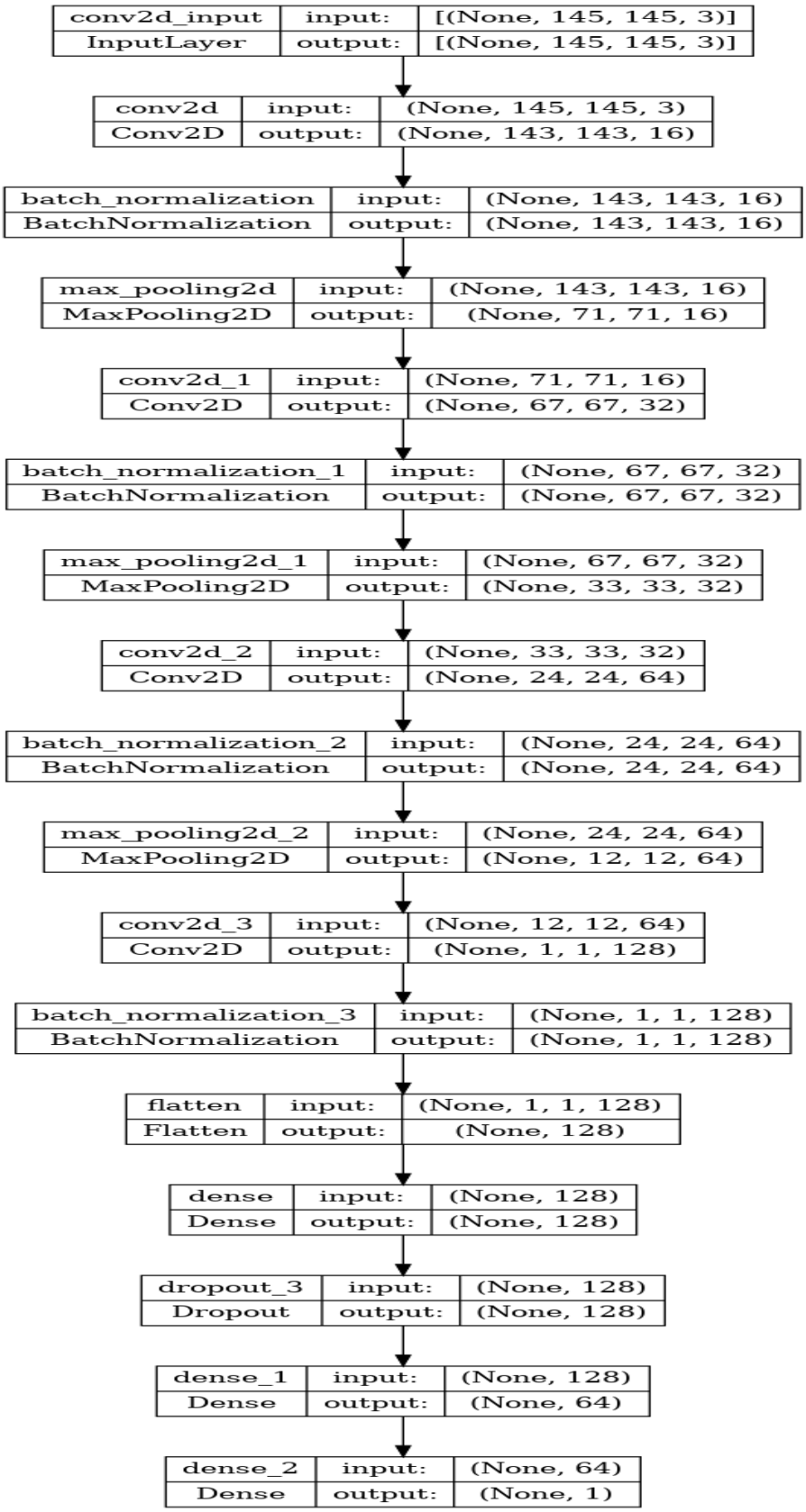


Figure 4.1.1.i. Baseline Model Structure (CNN)

The following are some key points that summarise the data flow in this model:

- The model's input is: (3 x 224 x 224)
- ConvLayers output: (1 x 56 x 64)
- Layers that are fully connected input: (1 x 1792)
- Final Results: 0 or 1 (Not Drowsy or Drowsy)

The total number of (142,371,3) batches that are trained every time(replication) is batches because we trained our birth model in 16- batch supplements. Overfitting is our main worry while training these networks. Overfitting occurs when a model learns patterns from too few examples that do not transfer to new data, or when the model starts relying on unrelated information when generating predictions. We have relied heavily on data augmentation to prevent this, which consists in forcing model weights to take smaller values thus mitigating the risk of overfitting up to a great extent.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 143, 143, 16)      448

 batch_normalization (BatchN  (None, 143, 143, 16)     64
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 71, 71, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 67, 67, 32)        12832

 batch_normalization_1 (Batc  (None, 67, 67, 32)       128
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 33, 33, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 24, 24, 64)        204864

 batch_normalization_2 (Batc  (None, 24, 24, 64)       256
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 1, 1, 128)         1179776

 batch_normalization_3 (Batc  (None, 1, 1, 128)        512
 hNormalization)

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 128)               16512

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 1,423,713
Trainable params: 1,423,233
Non-trainable params: 480
_________________________________________________________________
```

Table 4.1.1. Baseline Model Architecture Output

4.1.2. FINAL MODEL (FINE-TUNED WITH VGG16 IMAGENET):

We use a VGG16 model that has already been trained on the ImageNet dataset and then refine its weights to create our final model. The architecture of this model is as follows:

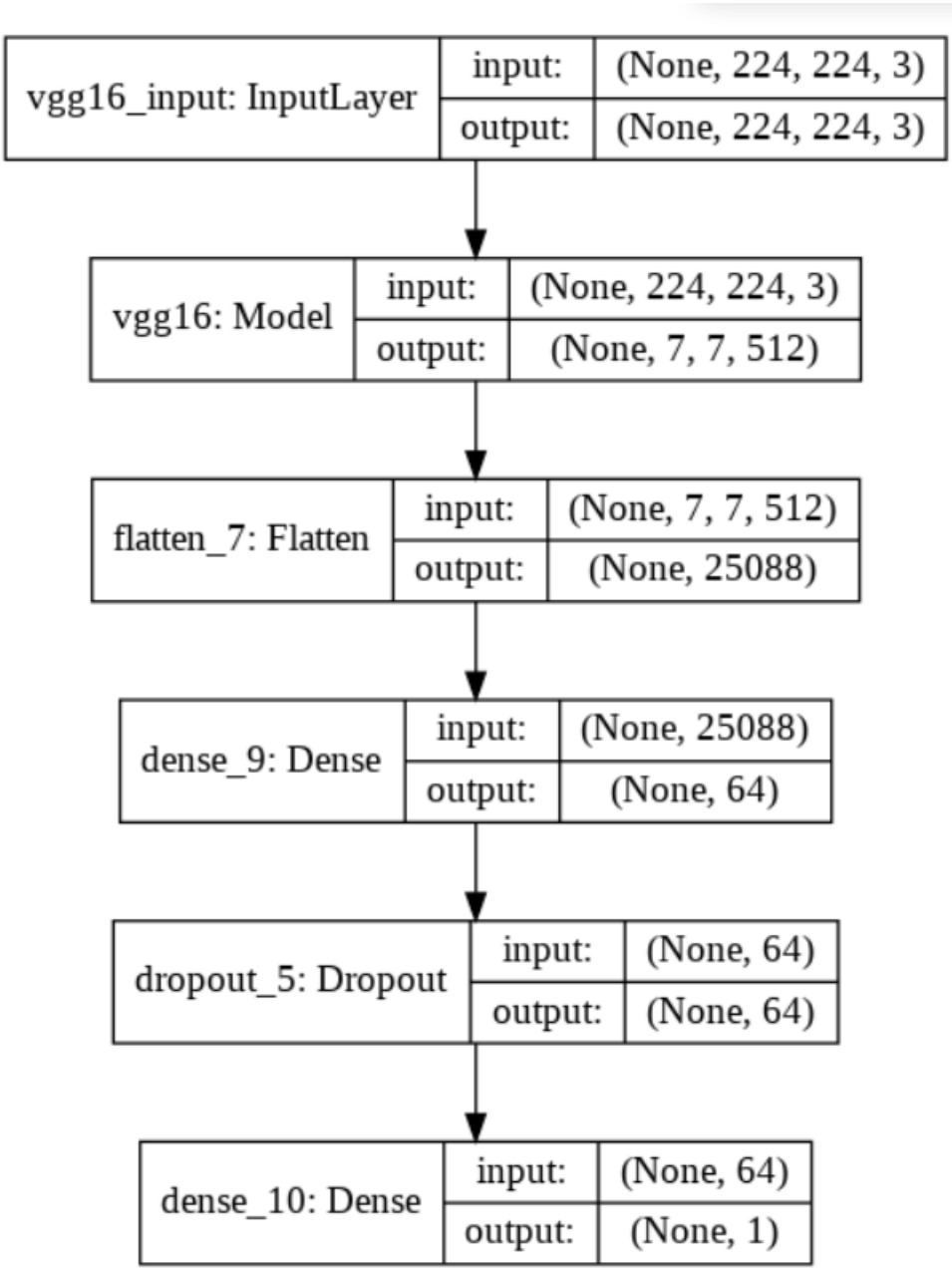


Figure 4.1.2. Final Model Structure

4.1.2.1 PRE-TRAINING / TRANSFER LEARNING

The goal of transfer learning (TL), a machine learning (ML) research issue, is to apply information discovered while resolving one problem to another that is unrelated but nevertheless poses similar challenges. To identify trucks, for instance, one can use the skills they had when learning to identify automobiles

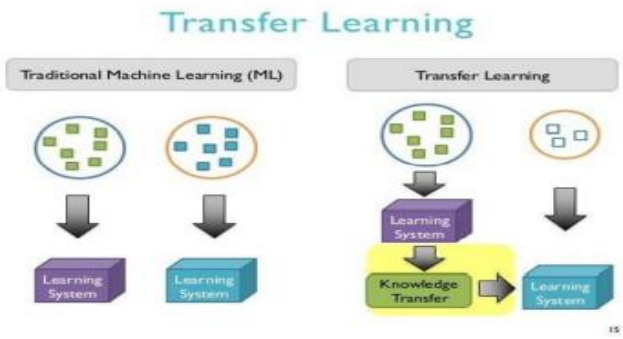


Figure 4.1.2.1.a. Transfer Learning in Practice

Core convolutional layers are often learned on the bigger dataset and their weights are present in pre-trained models. After that, we use our target dataset (smaller) to fine-tune the network's last layers (dense fully connected ones)

Due to scarcity of readily available data for our project, we are also using the application of Transfer Learning by pre-training our model on ImageNet Dataset

ImageNet Dataset: Each node in the WordNet hierarchy (currently just the nouns) is represented by hundreds of thousands of pictures in the ImageNet database.

The total number of images in the dataset are over 14 million distributed over more than 20,000 categories (or labels). The dataset looks as follows:

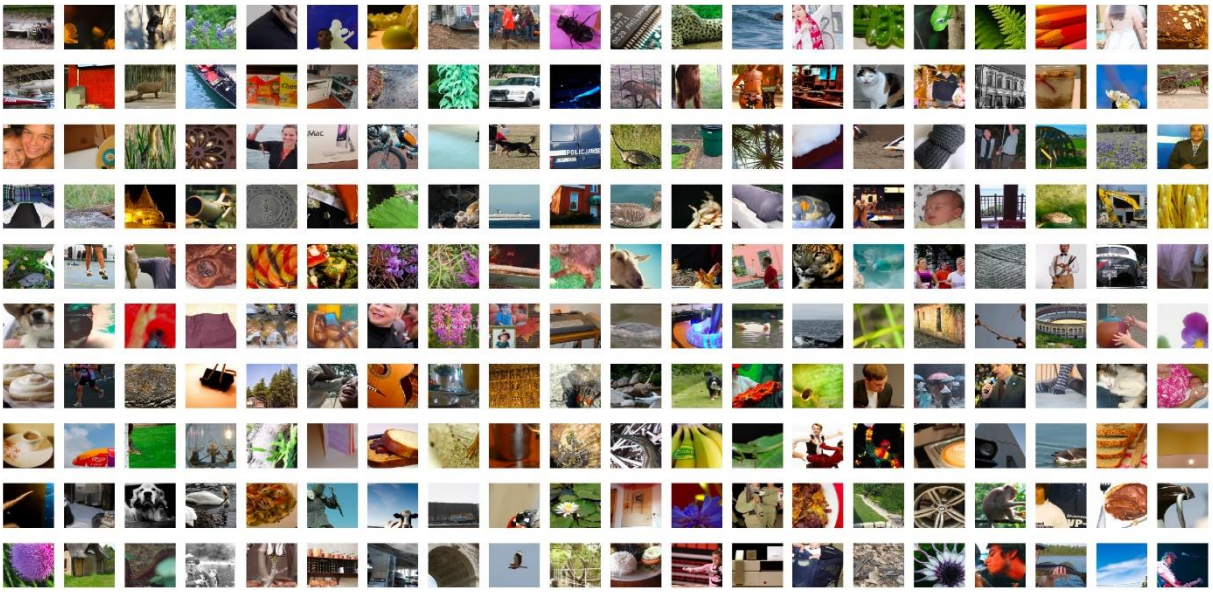


Figure 4.1.2.1.b. ImageNet Samples

4.1.1.2. VGG16 NETWORK

The convolution neural network (CNN) design VGG16 was used to triumph in the 2014 ILSVR (Imagenet) contest. It has stood the test of time as one of the most effective frameworks for vision models. VGG16's defining characteristic is that it makes due with fewer hyper-parameters than other popular models, opting instead for 3x3 convolution layers with stride 1 and the same 2x2 padding and maxpool layers with stride 2 in all cases. Layer configurations of convolution and maximum pool are preserved throughout. After two completely linked layers, a softmax is used as the output. (FC). The 16 in VGG16 indicates the number of weighted layers it has. This network contains a huge number of parameters—around 138 million. The diagram below depicts VGG16's underlying architecture.

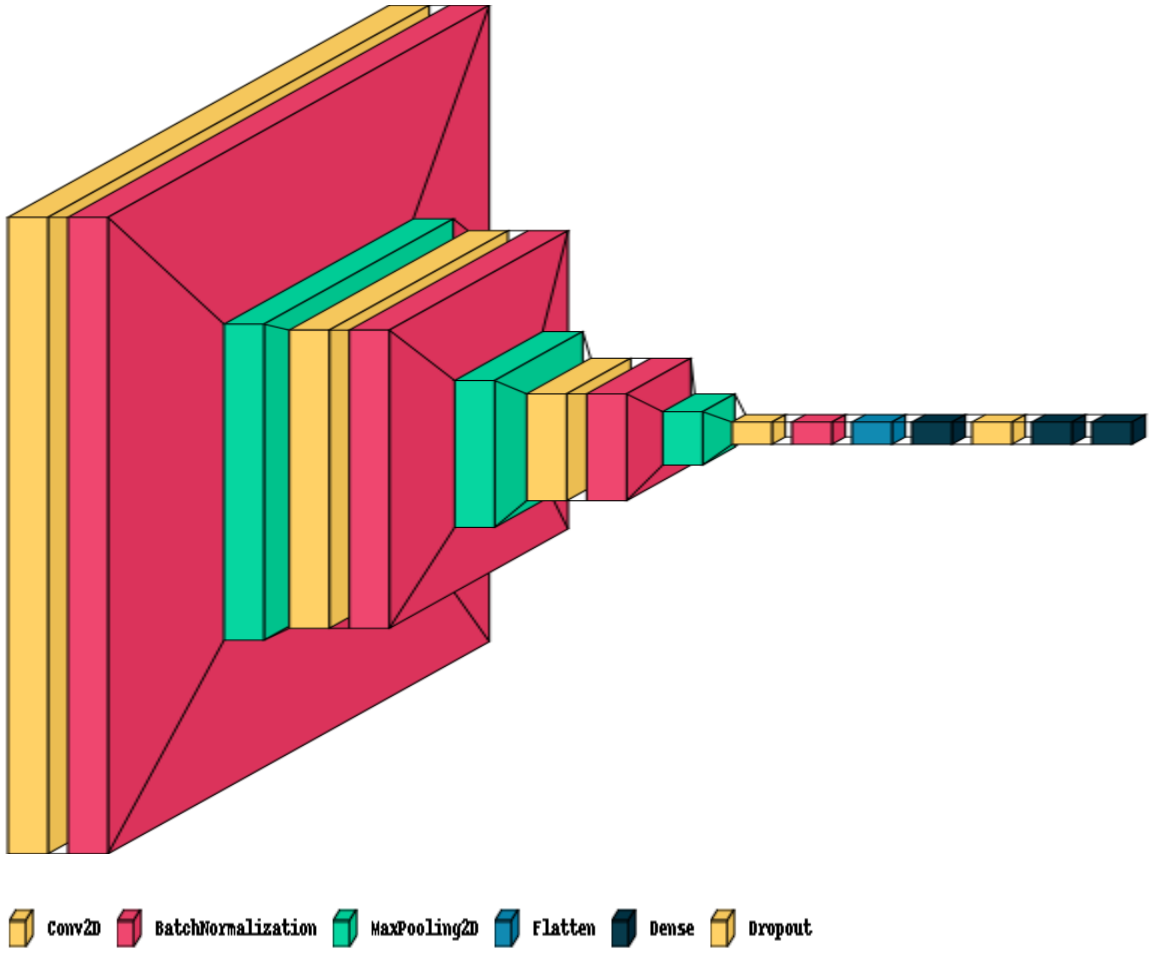


Figure 4.1.2.2. VGG16 Architecture

The ConvLayers are clearly pre-trained using the weights from the VGG16 networks used in the ImageNet dataset. The network's last layers are maintained dynamic so that the model may benefit from transfer learning while still being fine-tuned for the intended aim (drowsiness detection) by training the fully connected dense layers.

These are some key points that summarise how the data flowed in this model:

- Input to the model: (3 x 224 x 224)
- Output of ConvLayers: (4 x 4 x 512)
- Input to Fully Connected Layers: (1 x 8192)
- Final Output: 0 or 1 (Not Drowsy or Drowsy)

Our baseline model was trained in 16-batch increments, resulting in a total of 45,203 batches trained each iteration. We have mostly employed extensive Data Augmentation in order to prevent the overfitting issue. We also use a Regularization method called L2 Regularization, though, to properly address the issue. It entails putting pressure on model weights to accept lower values, thus reducing the chance of overfitting.

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 4, 4, 512)	14714688
flatten_4 (Flatten)	(None, 8192)	0
dense_5 (Dense)	(None, 64)	524352
dropout_3 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 1)	65

Total params: 15,239,105
Trainable params: 524,417
Non-trainable params: 14,714,688

Table 4.1.2.2.a.. Final Model Architecture Output

4.2. Computer Vision (Open CV)

Computer vision requires a massive quantity of information. It evaluates the data repeatedly until it can tell the difference between objects and recognise images. For instance, in order for a computer to learn the nuances and recognise a tyre, particularly one without any faults, it has to be given a massive volume of tyre pictures and tyre-related papers.

OpenCV allows for activities in computer vision and image processing to be carried out. In the realms of image data and computer vision, it is among the most widely used open-source Python modules. It has a wide range of applications, from image denoising and thresholding to edge detection and corner detection to contours and pyramids and segmentation and even face recognition.

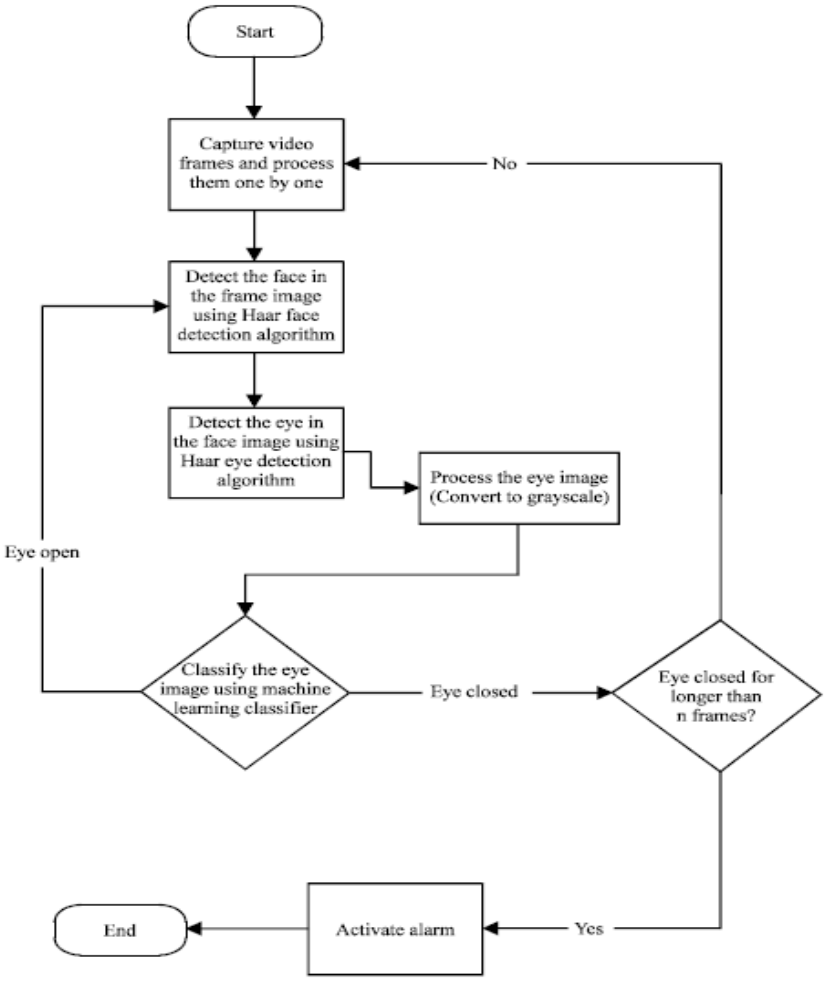


Figure 4.2. Flow chart for drowsiness detection

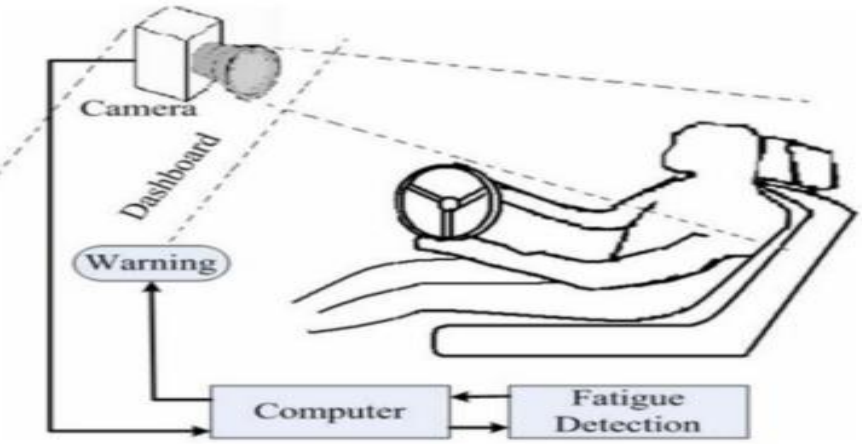


Figure 4.2.b. System Design

Sequential Model

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. Now, calling its summary () method to display its contents:

Table. 4.2.1. Sequential Data Table

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 143, 143, 256)     7168
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 71, 71, 256)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 69, 69, 128)       295040
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 34, 34, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        73792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 32)        18464
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dropout (Dropout)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 64)                100416
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 495,140
Trainable params: 495,140
Non-trainable params: 0
_________________________________________________________________
```

Now, since we have seen the working and architecture of both our models, let's see how they perform on our problem dataset.

Methodology

The Methodology of this system can be divided into two parts:

1. Detecting or localizing the face.
2. Predicting the landmarks of important regions in the detected face.
3. Once the landmarks are predicted, we use only the eye landmarks and the mouth landmarks to determine the Eye Aspect Ratio(EAR) and Mouth Aspect Ratio(MAR) to check if a person is drowsy.

Calculations for EAR and MAR are shown as:

Calculation of EAR and MAR

EAR_calulator.py

Raw

```python
from scipy.spatial import distance as dist
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

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
```

EAR and MAR Calculation.

Now, on the basis of code, let’s begin with how it works:

The dlib package's pre-trained facial landmark detector is used to locate the 68-(x,y) coordinates corresponding to the facial structure of the face[2]. These 68 coordinates stand for the most important parts of the face: the lips, the left and right brows, the left and right eyes, the nose, and the jaw. Out of these, we need just the (x,y) coordinates of the mouth, left eye, and right eye. This is accomplished as follows:

EAR and MAR Calculation

indexes.py Raw

```
# Grab the indexes of the facial landamarks for the left and right eye respectively
(lstart, lend) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rstart, rend) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(mstart, mend) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
```

Extracting the indexes for the left eye, right eye, and mouth.

Now, each eye is depicted by a set of 6-(x,y) matches offset at the abandoned corner of analysis (as if you were look at the human), and therefore active circling about the balance of the region[3]:

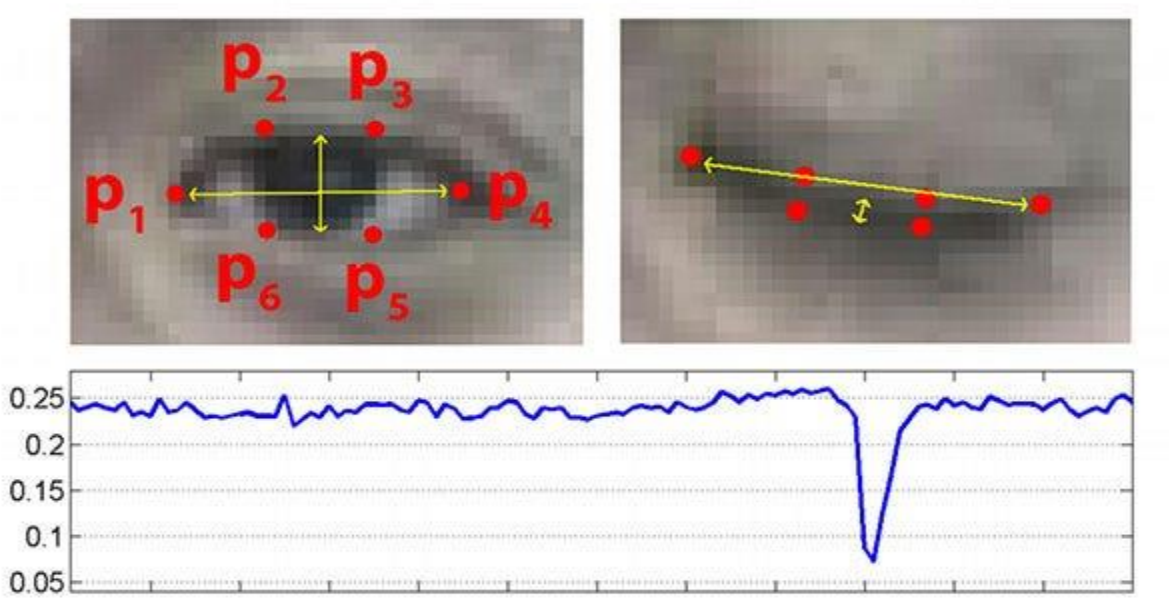


Figure 4.2.2.a. EAR Ratio and Graphs

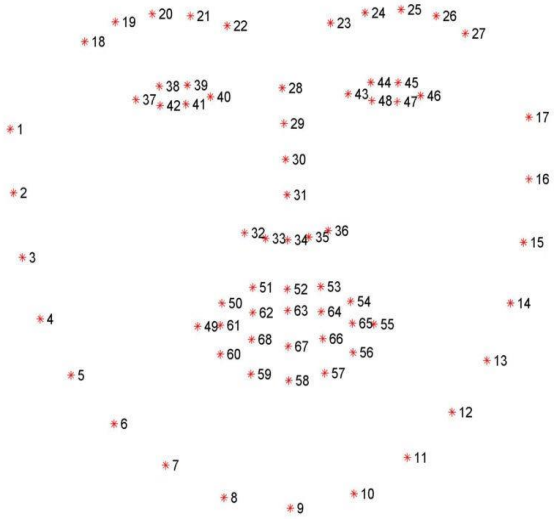
Top-left:- Open-eye representation of familiar visual cues. Closing the right eye reveals familiar landmarks in the top left. Aspect ratio of the eye throughout time, shown at bottom. Figure 1 by Soukupová and ech depicts a decrease in ocular aspect ratio after a blink [3].

Based on the paper, ***Real-Time Eye Blink Detection using Facial Landmarks***, we can then derive an equation that reflects this relationship called eye aspect ratio (EAR):

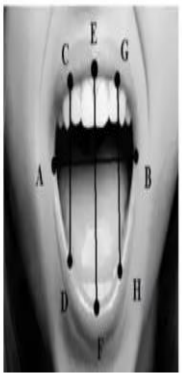
Eye Aspect Ratio(EAR) Equation

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Using this concept, we have calculated the Mouth Aspect Ratio:



Representing the face with 68-(x,y) coordinates.



$$\mathrm{MAR} = \frac{|EF|}{|AB|}$$

Mouth Aspect Ratio (MAR) Equation

As we see that the mouth is represented by a set of 20-(x,y) coordinates. So, we have used coordinates 62, 64, 66, and 68 to calculate the distance between then in the same way as EAR Calculation.

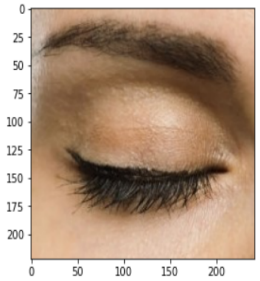
Reading Image Data Sets

Visualize random image from Open Data Sets

+ Code + Markdown

```
#closeEye
import matplotlib.pyplot as plt
plt.imshow(plt.imread("../input/drowsiness-dataset/train/Closed/_120.jpg"))
```

```
<matplotlib.image.AxesImage at 0x7f22106c0910>
```



Closed Eye

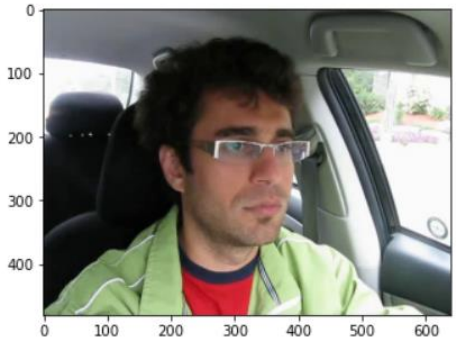
```
#openEye
import matplotlib.pyplot as plt
plt.imshow(plt.imread("../input/drowsiness-dataset/train/Open/_100.jpg"))
```

```
<matplotlib.image.AxesImage at 0x7ff310047a50>
```

Open Eye

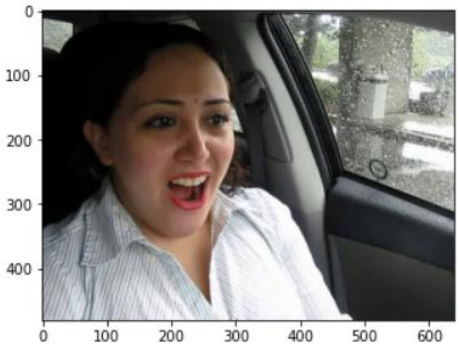
In [8]:
```
#no yawn
plt.imshow(plt.imread("../input/drowsiness-dataset/train/no_yawn/1028.jpg"))
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x7f2c59ba7c50>
```



In [9]:
```
#yawn
plt.imshow(plt.imread("../input/drowsiness-dataset/train/yawn/104.jpg"))
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x7f2c59b16710>
```



Sample Codes

For yawn and not_yawn. Take only face

For yawn and not_yawn Image. Capturing only face

```
def face_for_yawn(direc="../input/drowsiness-dataset/train", face_cas_path="../input/prediction-images/haarcascade_frontalface_default.xml"):
    yaw_no = []
    IMG_SIZE = 145
    categories = ["yawn", "no_yawn"]
    for category in categories:
        path_link = os.path.join(direc, category)
        class_num1 = categories.index(category)
        print(class_num1)
        for image in os.listdir(path_link):
            image_array = cv2.imread(os.path.join(path_link, image), cv2.IMREAD_COLOR)
            face_cascade = cv2.CascadeClassifier(face_cas_path)
            faces = face_cascade.detectMultiScale(image_array, 1.3, 5)
            for (x, y, w, h) in faces:
                img = cv2.rectangle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
                roi_color = img[y:y+h, x:x+w]
                resized_array = cv2.resize(roi_color, (IMG_SIZE, IMG_SIZE))
                yaw_no.append([resized_array, class_num1])
    return yaw_no


yawn_no_yawn = face_for_yawn()
```

Output

0
1

Training and Validation

```
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, "b", label="trainning accuracy")
plt.plot(epochs, val_accuracy, "r", label="validation accuracy")
plt.legend()
plt.show()

plt.plot(epochs, loss, "b", label="trainning loss")
plt.plot(epochs, val_loss, "r", label="validation loss")
plt.legend()
plt.show()
```

Prediction on Data Sets (EAR & YAWN)

```
import seaborn as sns

ax = sns.heatmap(cf_matrix, annot=True, cmap='Blues')
ax.xaxis.set_ticklabels(['yawn','no_yawn','Closed','Open'])
ax.yaxis.set_ticklabels(['yawn','no_yawn','Closed','Open'])
```

Prediction Function

```
labels_new = ["yawn", "no_yawn", "Closed", "Open"]
IMG_SIZE = 145
def prepare(filepath, face_cas="../input/prediction-images/haarcascade_frontalface_default.xml"):
    img_array = cv2.imread(filepath, cv2.IMREAD_COLOR)
    img_array = img_array / 255
    resized_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    return resized_array.reshape(-1, IMG_SIZE, IMG_SIZE, 3)

model = tf.keras.models.load_model("./drowiness_new6.h5")
```

Statistics of Road Accident Caused By Drowsiness -2019

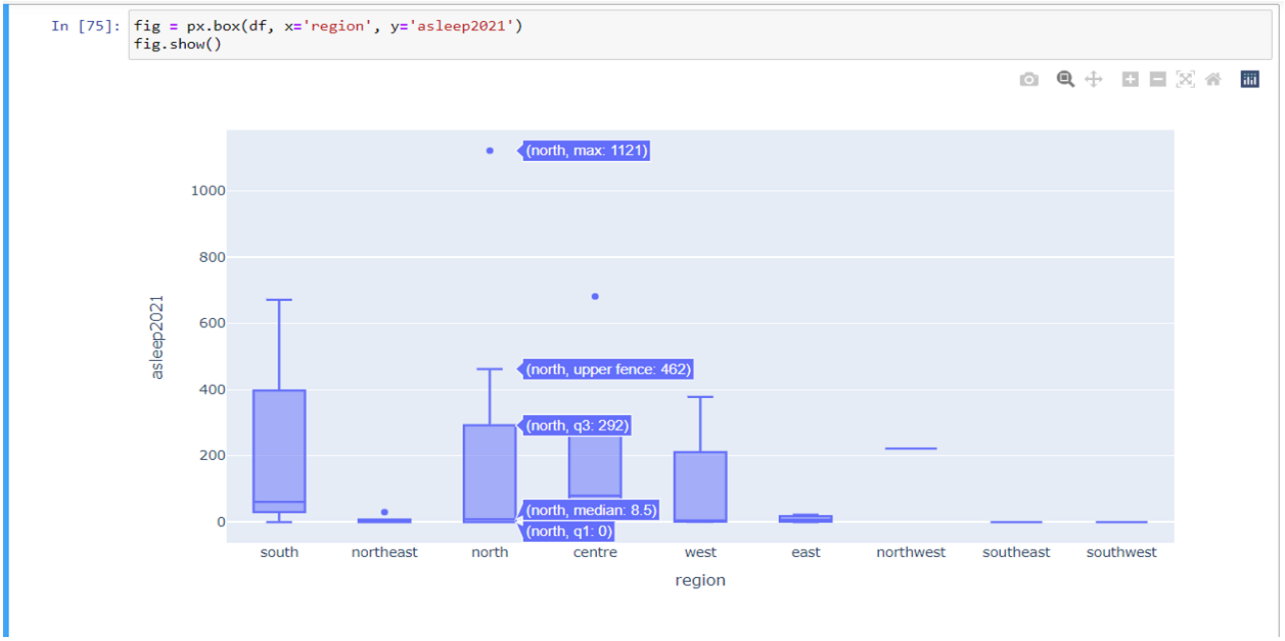
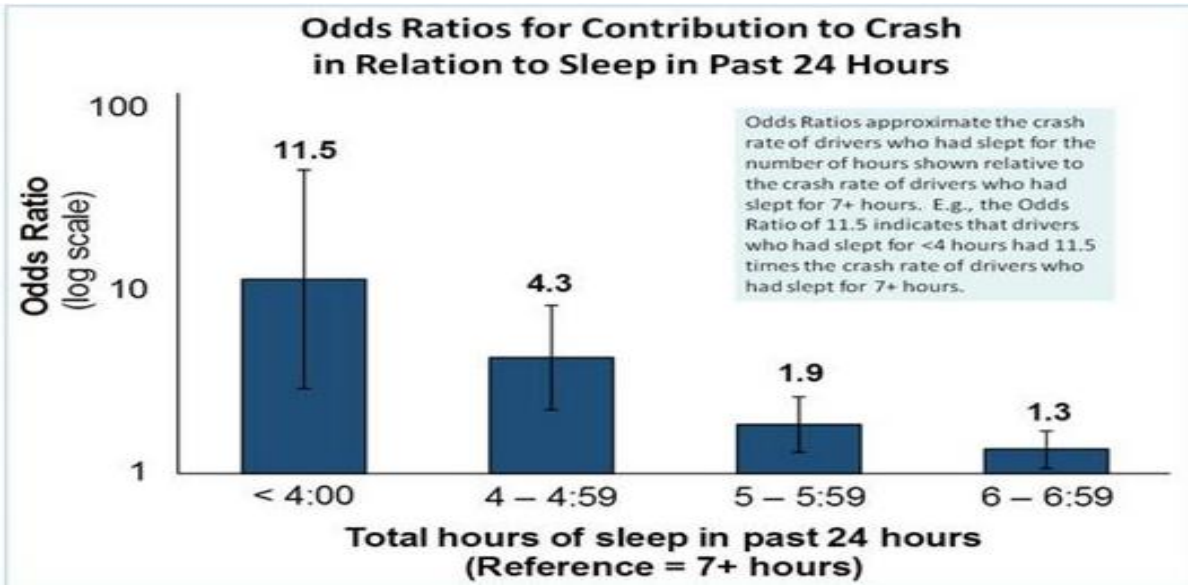


Figure 4.2.2.1. Region Plot

Report and Statistics from Ministry of Road Transport and Highways



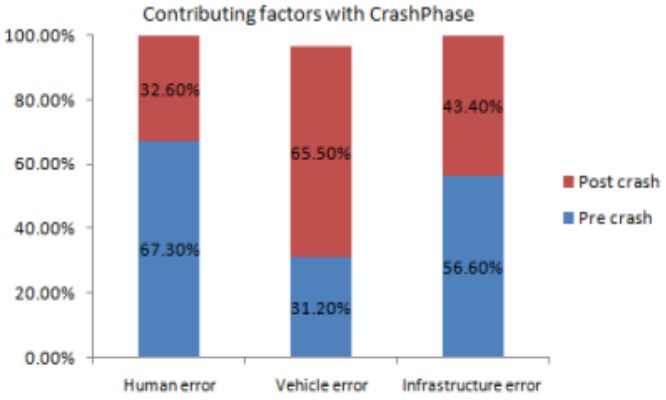


Figure1: Distribution of contributing factors into human, vehicle, and infrastructure.

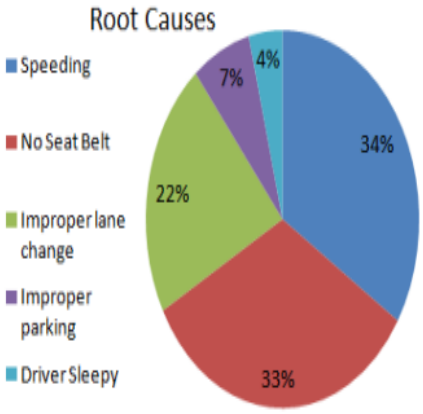


Figure2: Distribution of Root Causes

Figure 4.2.2.2. :- MRTH INDIA Report

RESULTS

The table below provides a performance comparison of some of the most common designs and methods from the literature. Please be aware that in all of our tests, the pre-processing step of Data Augmentation employs the same randomised distortions. The best accuracies achieved by our respective models are as follows:

Table 5.1. Data Augmentation Table (Base Line vs Fine-tuned Model)

Scenario	Baseline Model	Final Model (Fine-Tuned)
No Glasses	73.25%	77.17
Glasses	75.64%	80.43
Sunglasses	76.22%	81.09
Night_No glasses	73.50%	79.33
Night_Glasses	65.63%	68.84
Al.l	73.20%	77.84

When compared to other popular works in the literature, our approach showed true promise, which is evident from the following results:

Table 5.2. CNN Accuracy Table

Model/ Approach	Accuracy
3D Convolution	
130, Carreira and Zisserman (19]	75.4%
lmageNet & Kinetics pre-trained,Wijnands et al. (20]	73.9%
2D Convolution	
lnceptionVl,Szegedy et al. (18]	69.6%
MobileNetV2_ 1.4, Sandler et al. (16]	72.8%
Ours (Final Model)	**77.84%**

The accuracy of the model during testing is 77.84%. The model is compiled using CNN 2D

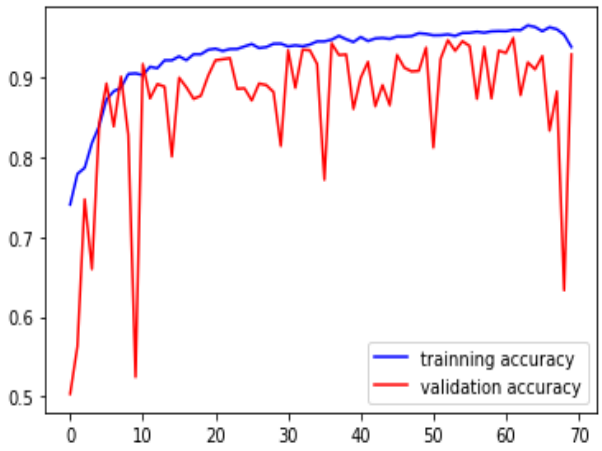
As per classification report, The best accuracies achieved by our respective models are as follows:

	Precision	Recall	F1-score	Support
Yawn	0.75	0.92	0.83	63
No_yawn	0.87	0.74	0.8	74
Closed	0.92	0.94	0.93	215
Open	0.96	0.93	0.94	226
Accuracy			0.91	578
Macro avg	0.88	0.88	0.88	578
Weighted avg	0.91	0.91	0.91	578

Table 5.3 Classification report for Computer Vison (Open CV) Table

The accuracy of the model during testing is **91%**. The model is compiled using Gray Wolf Optimizer as it has yielded the best results based on Face and Eye Tracking.

When compared to other popular works in the literature, our approach showed true promise, which is evident from the following results:



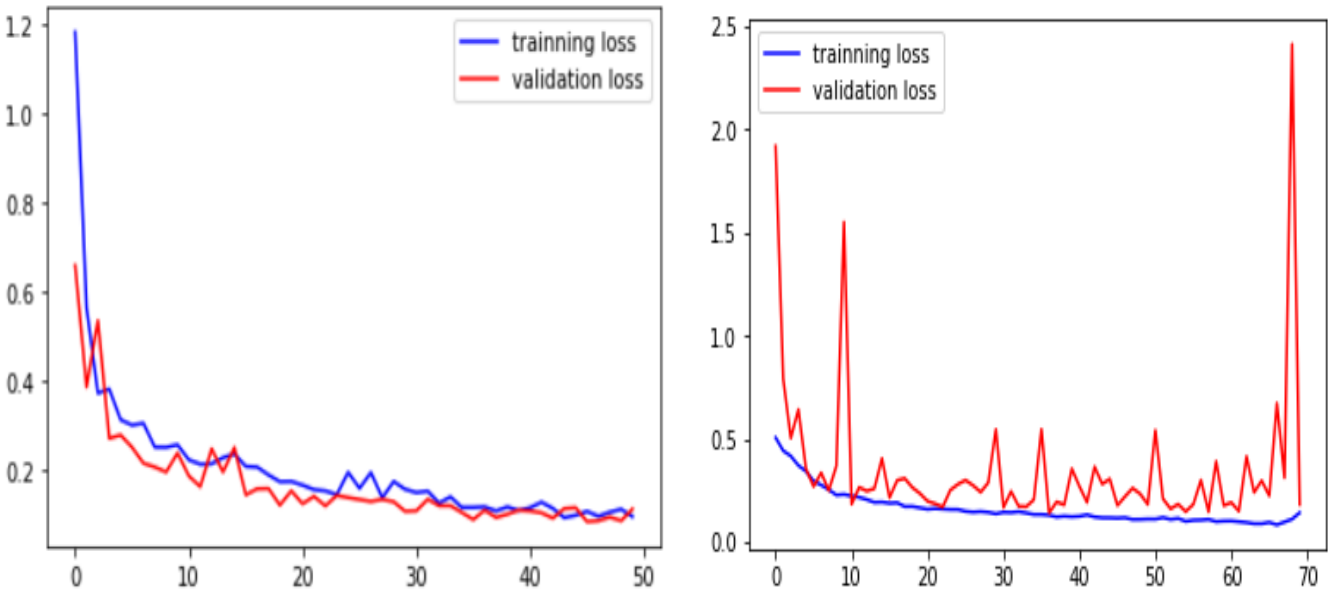


Figure 5.1. Training VS Validation graph

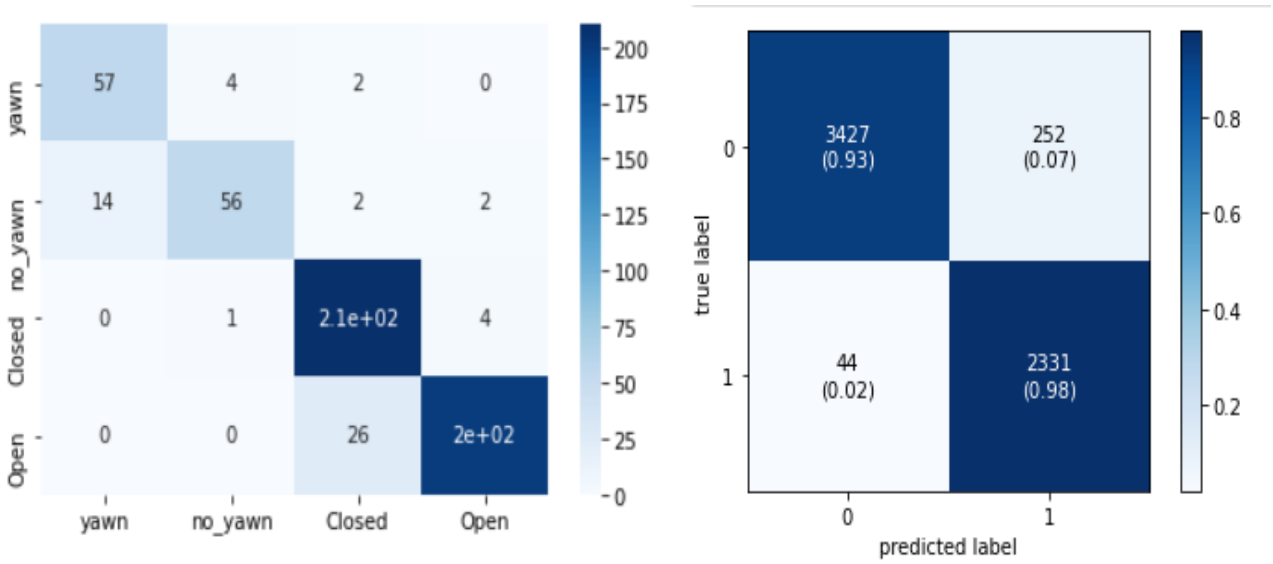


Figure 5.2. Confusion Matrix for Open CV/ CNN

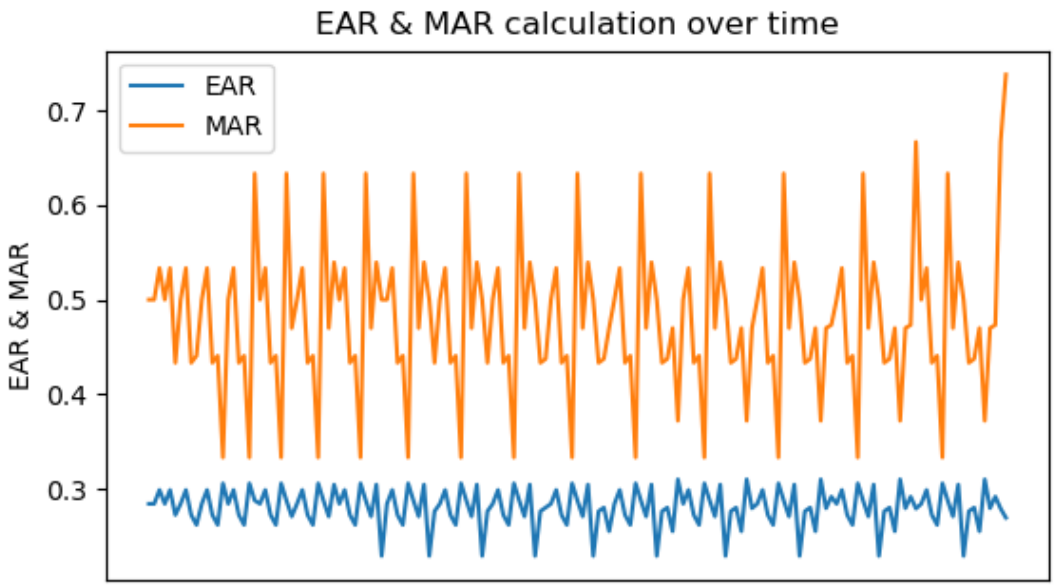
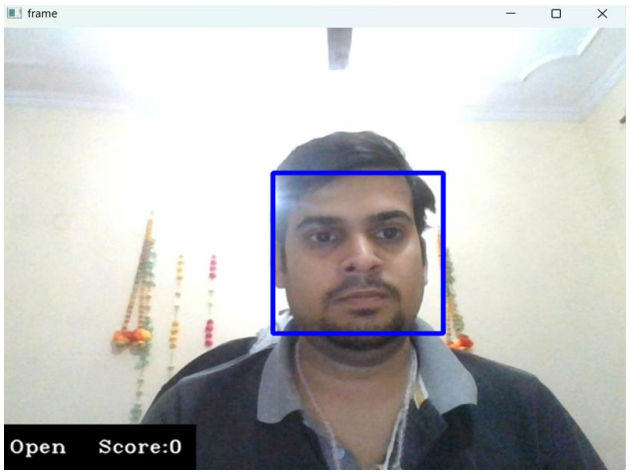
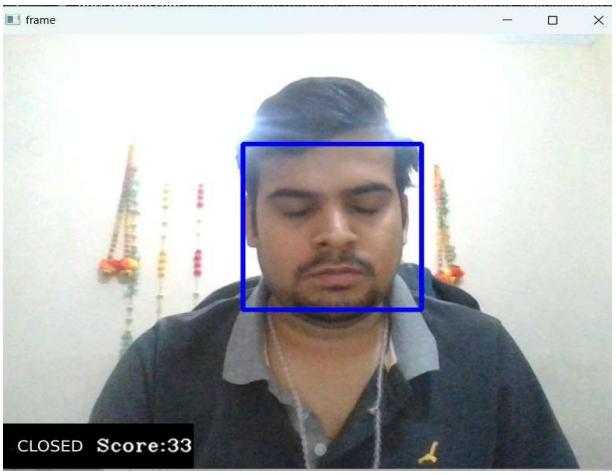


Figure 5.3. Graph showing the variation of EAR and MAR with time.

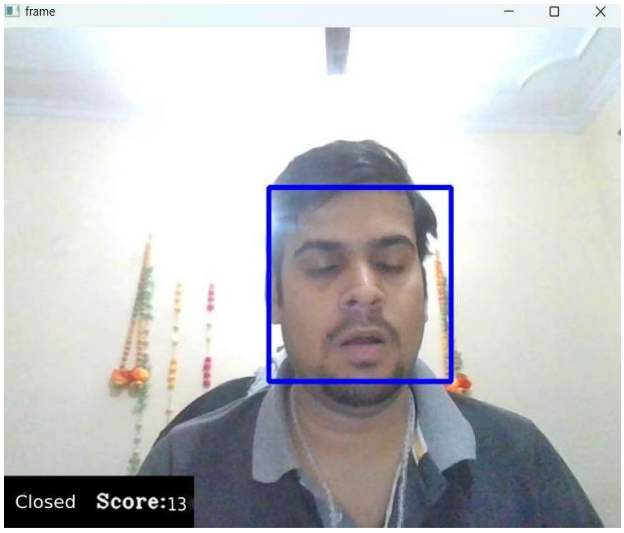
Open Eye



Closed Eye



Yawning



Eye Rubbing

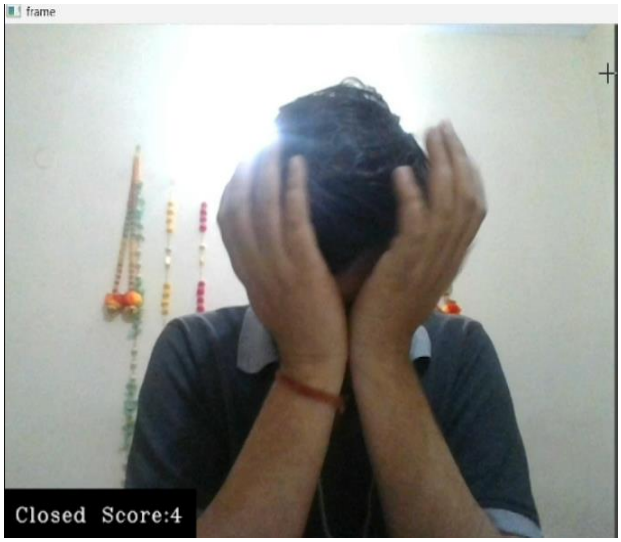


Figure 5.4. Drowsiness Detection using web cam and sending alerts

6. CONCLUSION

- The process of drivers' drowsiness detection is very important for both individual and community safety.
- The growing use of Artificial Intelligence can be immensely useful in predicting fatigue of a driver.
- With an accuracy of 73.2% using CNN, our model can predict in real time whether a driver is sleepy or alert based on its facial recognition and critical attributes.
- It determines if a motorist is sleepy or alert in real time by analysing its face using the Computer Vision (Open CV) Gray Wolf Optimizer model. The model has an accuracy of 91.2%.
- A camera can be easily installed, and unlike biological or vehicle-based features, it doesn't interfere with the driver's pleasure or comfort, among its many benefits.
- Not prone to external disturbances, considers only the driver's face.

6.1. Future Scope:

- Additional criteria, like as the vehicle's condition or the presence of foreign substances on the user's face, can be used to improve the system's accuracy.
- It could be done to create a programme that would either wake the user up or keep them from falling asleep.
- It may be implemented into an Internet-of-Things (IoT) device for usage in vehicles to monitor driver fatigue.
- Netflix, Hotstar, and other streaming service platforms may use similar models and methodologies to determine if a user is asleep and pause the video if necessary.

6.2. LIMITATIONS:

If the eye frames are not recorded clearly (for example, because of goggles or mirrored eyewear), the model's accuracy will suffer. Experiments often disregard camera functions like auto-adjustment of zoom and rotation. When the eyes have been pinpointed, an automated zoom-in will improve precision. When the driver's back is to the camera, facial features are less reliably detected.

Detailed Plan of Work

Sr. No.	Tasks	Expected date of completion	Names of Deliverables	Status
1.	Synopsis	03-Dec-2022	Synopsis.pdf	Completed
2.	Study Phase: Literature review, existing solutions, and comparisons of existing solutions if any	03-Dec-2022 08-Dec-2022	Added to report	Completed
3.	Solution Design	10-Dec2022	Added to report	Completed
4.	Acquisition of ILDC/Kaggle Dataset	Data Acquisition	Data Acquisition	Completed
5.	Learning and Understanding Shallow Model (SVM)	02-01-2023	Learning-1	Learning-1
6.	Learning and Understanding Deep Learning model (CNN)	10-01-2023	Learning-2	Completed
7.	Learning Open CV and Transformation Models	12-01-2023	Learning-2	Completed
8.	Development Part 1	20-01-2023	Baseline model development with pre-trained model	Completed
9.	Mid Sem Review	14-12-2021	MidSemppt.ppt	Completed
10.	Feedback Incorporate	08-02-2023	Added to report	Completed
11.	Development Part 2	27-02-2023	Added to report	Completed
12.	Feedback Incorporate	23 -02-2023	Added to report	Completed
13.	Model improvement if any	23-02-23	Added to report	Completed
14.	Draft Report preparation	08-03-2023	Completed	Completed
15.	Final Report preparation for submission	13-0.2023	Final report	Completed

6. REFERENCES

i. M. R. Endsley, “Toward a theory of situation awareness in dynamic systems,” Hum. Factors, J. Hum. Factors Ergonom. Soc., vol. 37, no. 1, pp. 32–64, 1995.

ii. Y. Xing, C. Lv, D. Cao, H. Wang, and Y. Zhao, “Driver workload estimation using a novel hybrid method of error reduction ratio causality and support vector machine,” Measurement, vol. 114, pp. 390–397, Jan. 2018.

iii. Y. Xing et al., “Identification and analysis of driver postures for in vehicle driving activities and secondary tasks recognition,” IEEE Trans. Comput. Social Syst., vol. 5, no. 1, pp. 95–108, Mar. 2018.

iv. S. Kaplan, M. A. Guvensan, A. G. Yavuz, and Y. Karalurt, “Driver behavior analysis for safe driving: A survey,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 6, pp. 3017–3032, Dec. 2015.

v. C. M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, “Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey,” IEEE Trans. Intell. Transp. Syst., vol. 19, no. 13, pp. 666–676, Mar. 2016

vi. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. https://doi.org/10.1016/j.neu net.2014.09.003

vii. Silberman N, Guadarrama S (2016) TensorFlow-Slim image classification model library. https://github.com/tensorflow/models/tree/master/research/slim

viii. Carreira J, Zisserman A (2017) Quo vadis, action recognition? A new model and the Kinetics dataset. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), IEEE, Honolulu, HI, pp 4724–4733. https://doi.org/10.1109/CVPR.2017.502

ix. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A Page | 24 large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

x. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

xi. http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

xii. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8930504

xiii. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7959292/pdf/sensors-21-01734.pdf

xiv. https://www.ijrte.org/wp-content/uploads/papers/v8i3/C4028098319.pdf

xv. https://arxiv.org/pdf/2112.10298.pdf

xvi. https://www.iieta.org/journals/ria/paper/10.18280/ria.330609

xvii. https://www.hindawi.com/journals/cin/2020/7251280/