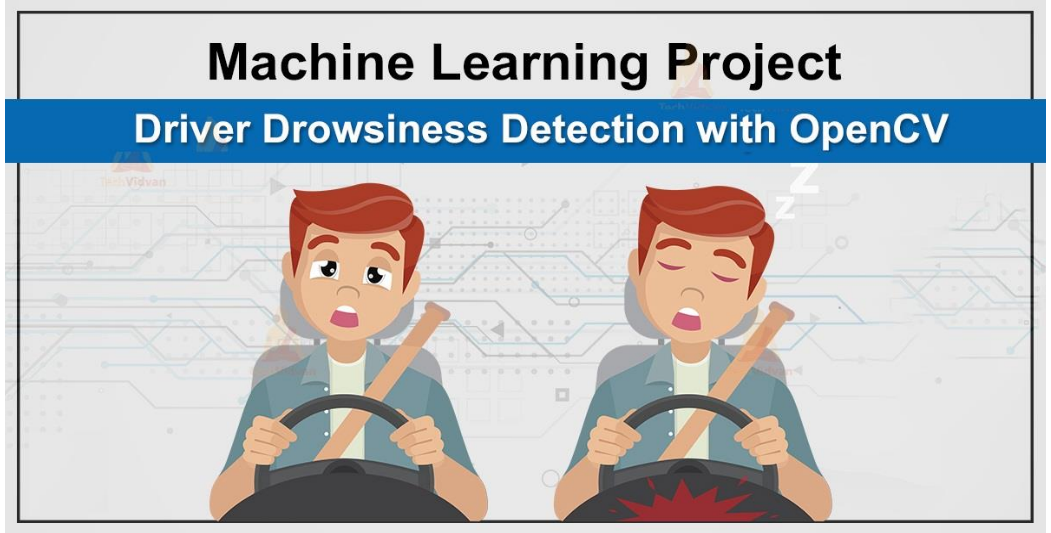
Oriental Institute of Science and Technology, Bhopal
Department of Computer Science and Engineering

Progress Seminar - I

on

DRIVER ALERT
[Driver Drowsiness Detection System]
(Minor Project)
Session 2021-2022



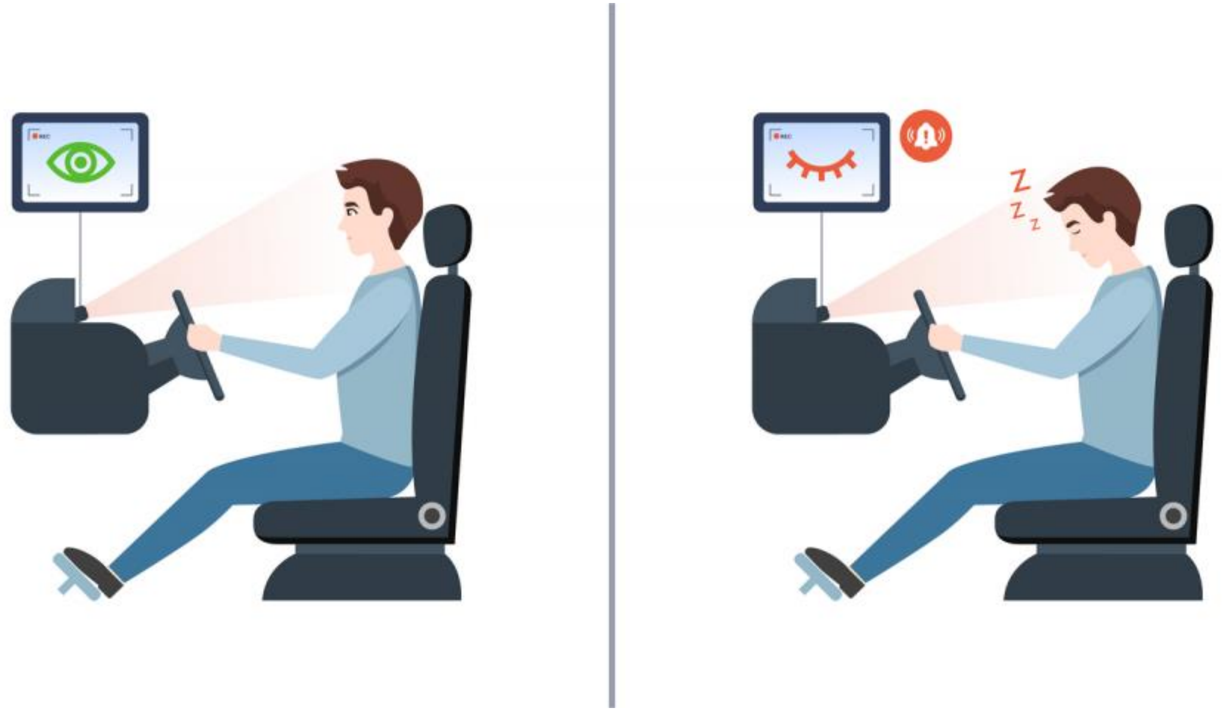
Guided By : *Prof. Goldi Jarbais*

Presented by:

1. Anshul Verma (0105CS191023) [Team Leader]
2. Harshit Shrivastava (0105CS191048)
3. Manish Nathrani (0105CS191062)

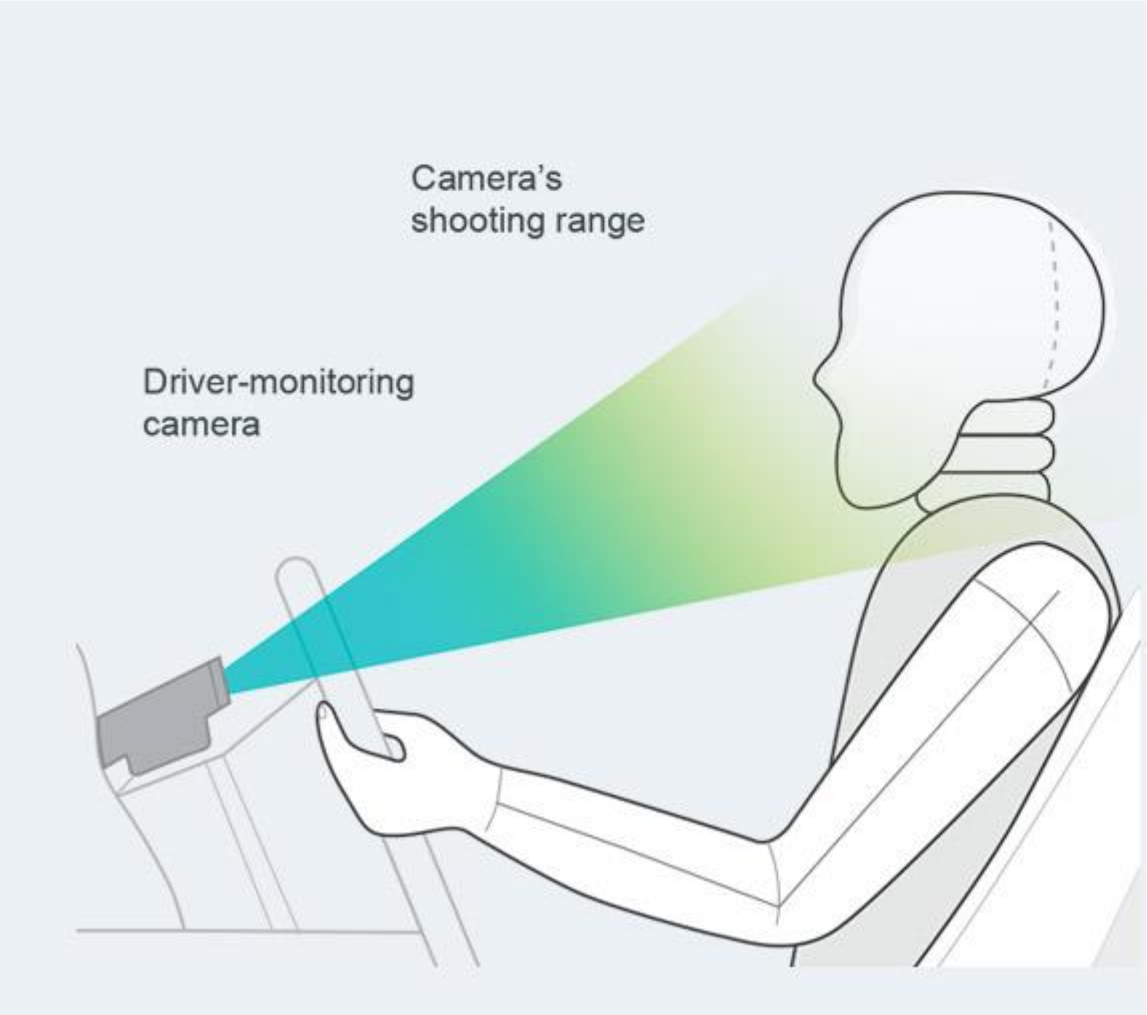
INTRODUCTION

- Nowadays Driver Drowsiness is a major factor in a large number of vehicle accidents.
- Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes.
- The development of technologies for detecting and avoiding drowsiness at the wheel is a major challenge in the field of accident avoidance systems.



DETAIL IDEA OF PROJECT

- The aim of this project is to develop a Driver Drowsiness Detection System.
- The focus is on designing a system that will accurately monitor the open or closed state of the drivers eyes in real-time.
- It is also found that the symptoms of driver fatigue can be detected early enough to avoid a car accident.
- Detection of drowsy involves a pattern of images of a face, and the observation of eye movements and blink rate.



HOW IT WORKS...

Step 1 – Take image as input from a camera.
Step 2 – Detect the face in the image and create a Region of Interest (ROI).
Step 3 – Detect the eyes from ROI and feed it to the classifier.
Step 4 – Classifier will categorize whether eyes are open or closed.
Step 5 – Calculate score to check whether the person is drowsy.

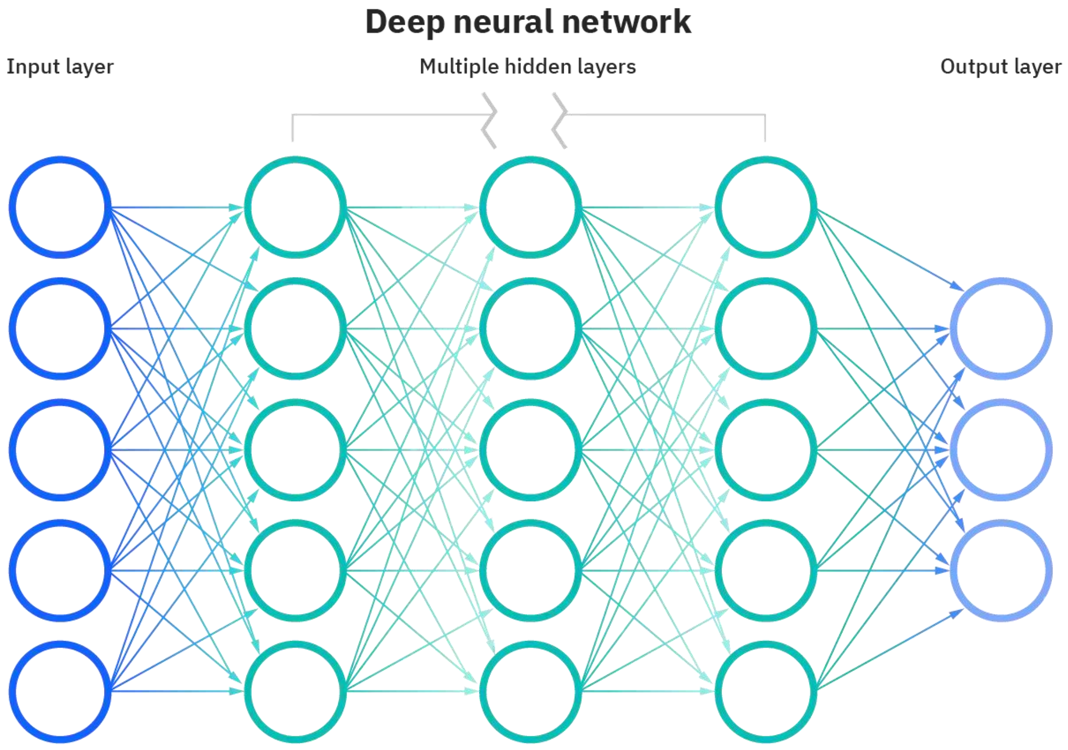
TIME FOR WAKING UP THE DRIVER TO AVOID ACCIDENTS :-

CARELESS DRIVERS

ALARMING SOUND

PRE-REQUISITE

- Use of Python
- Machine Learning
- Deep Learning Models
- Neural Network
  (specifically Convolutional Neural Network {CNN})
- Dataset from Kaggle



TECHNOLOGIES

- The necessary Python Libraries which will be required are : -

    1. OpenCV – face and eye detection

    2. TensorFlow – keras uses TensorFlow as backend

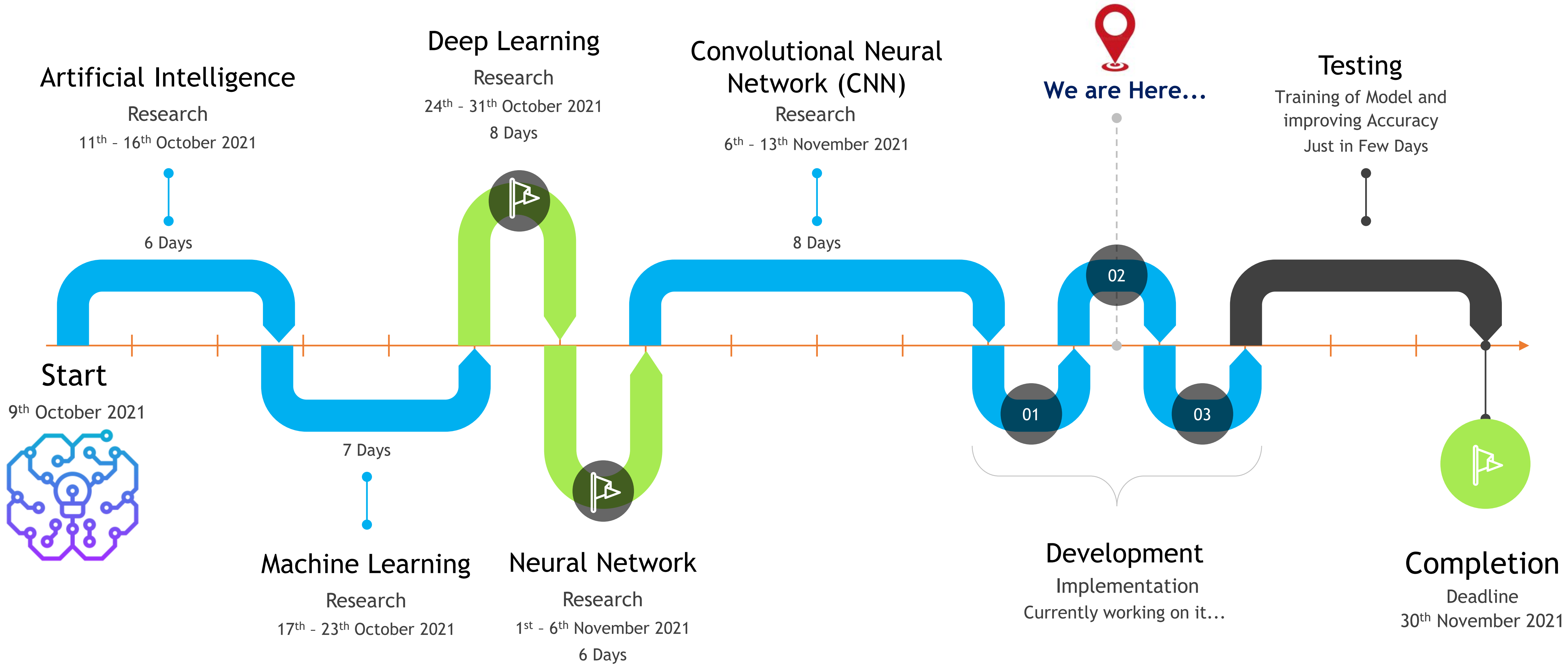
    3. Keras – to build our classification model

    4. Pygame – to play alarm sound

    5. Numpy – for mathematical operations

- The requirement for this Python project is a **Webcam** through which we will *Capture Images* and a **Speaker** to play the *Alarming Sound.*

TIMELINE OF THE PROJECT



METHODOLOGY

We found out Two Approaches to make this Project -

1. **Approach A :**

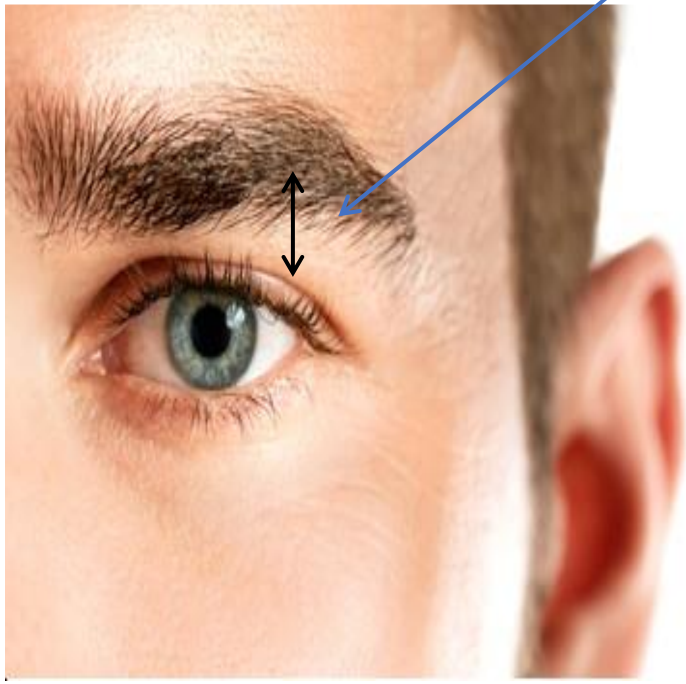
   Using 68 Facial Landmarks

2. **Approach B :**

   Using the Concept of Model Training and CNN

APPROACH A

COMPARING DISTANCES

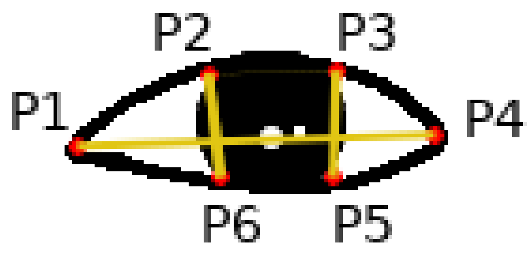


Below you can see a representation of the numbered facial landmarks

Facial Landmarks

EYE DETECTION

Eye Landmarks



Vertical and Horizontal distances

Vertical Distance

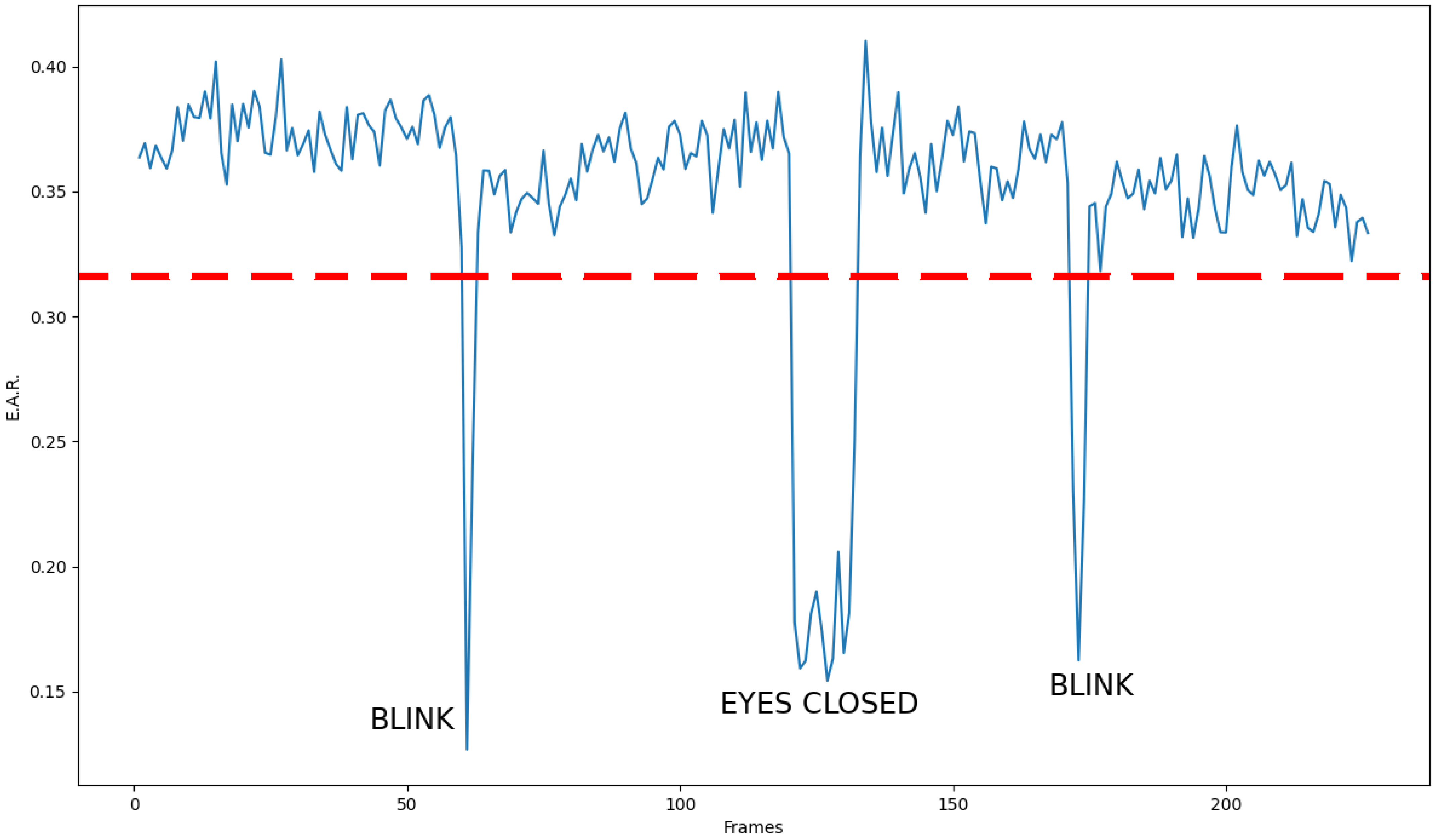
dist1 = P2 − P6
dist2 = P3 − P5

Horizontal Distance

dist3 = *P1 − P4*

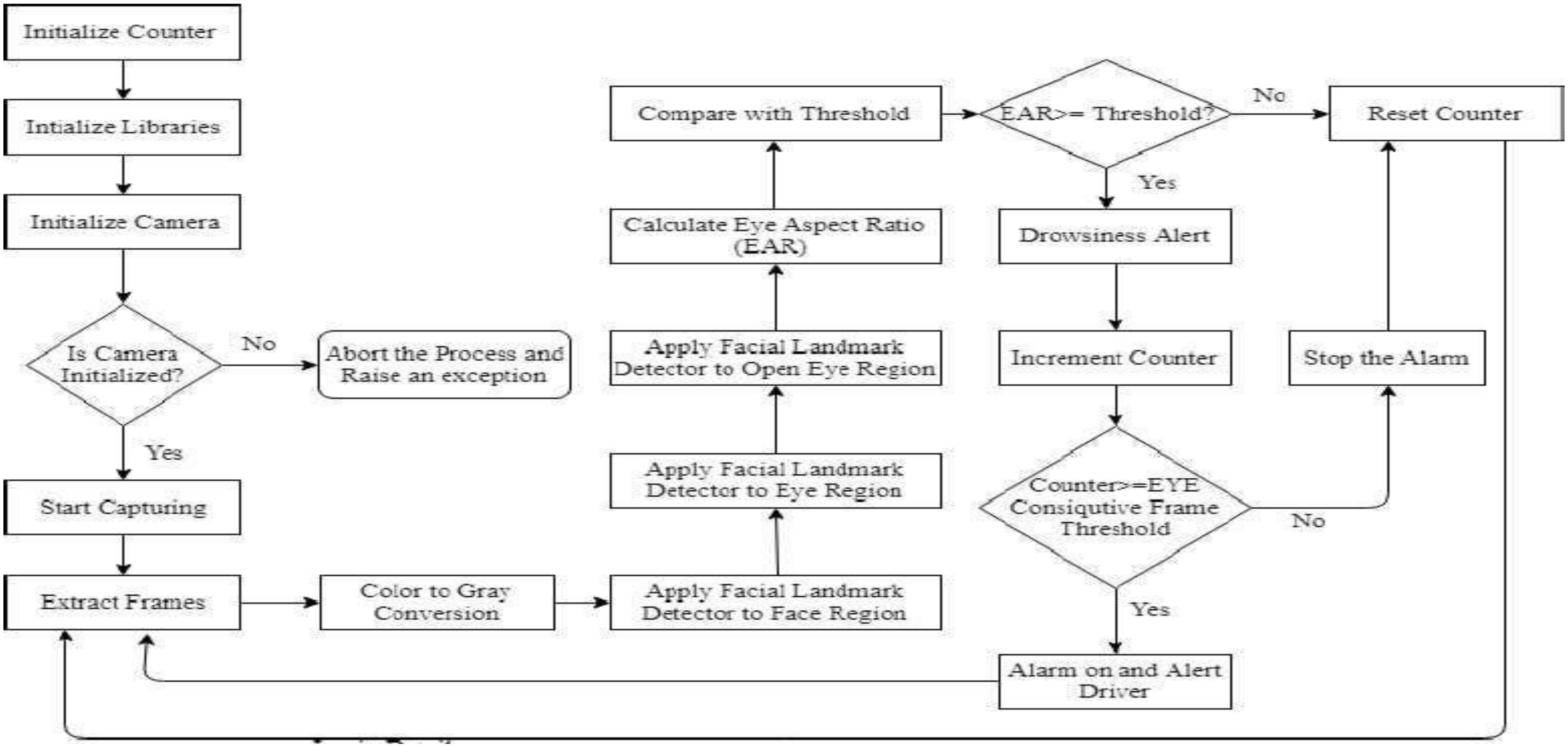
Formula :

$$\text{EAR(Eye Aspect Ratio)} = \frac{(dist1+dist2)}{2\,.dist3}$$

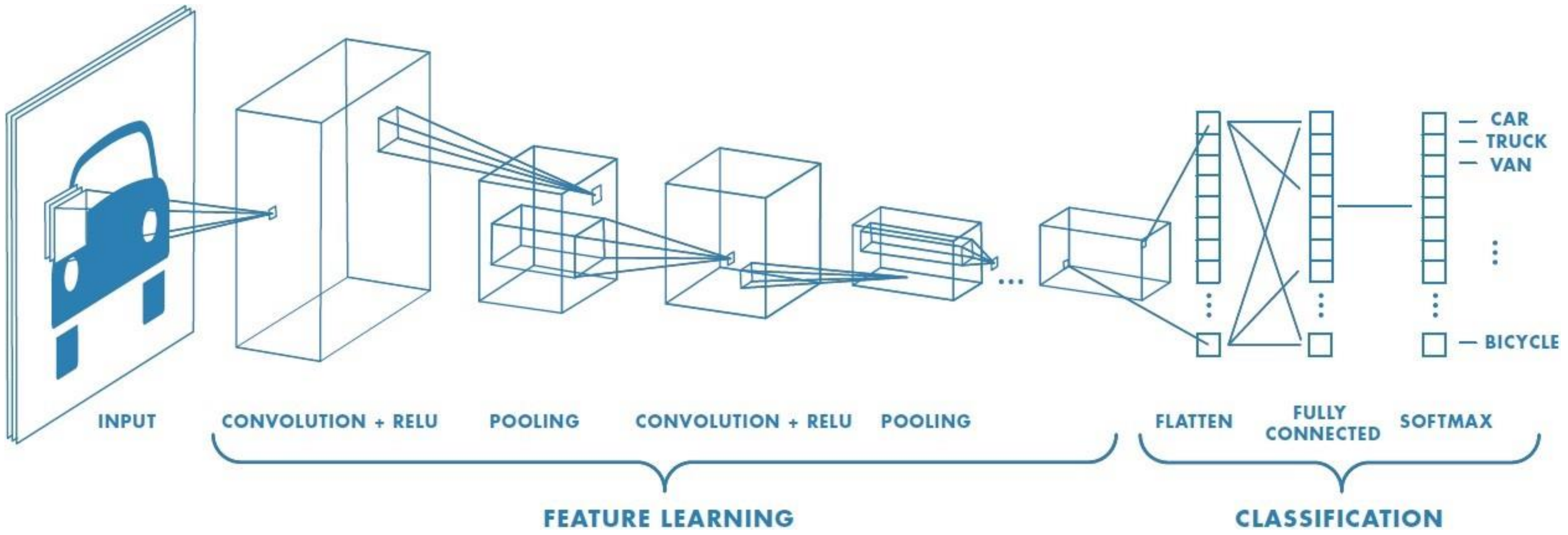


Blinks and eyes closed intervals

FLOWCHART


Initialize Counter
Intialize Libraries
Initialize Camera
Is Camera Initialized?
No
Abort the Process and Raise an exception
Yes
Start Capturing
Extract Frames
Color to Gray Conversion
Apply Facial Landmark Detector to Face Region
Apply Facial Landmark Detector to Eye Region
Apply Facial Landmark Detector to Open Eye Region
Calculate Eye Aspect Ratio (EAR)
Compare with Threshold
EAR>= Threshold?
No
Reset Counter
Yes
Drowsiness Alert
Increment Counter
Counter>=EYE Consiqutive Frame Threshold
No
Stop the Alarm
Yes
Alarm on and Alert Driver

APPROACH B

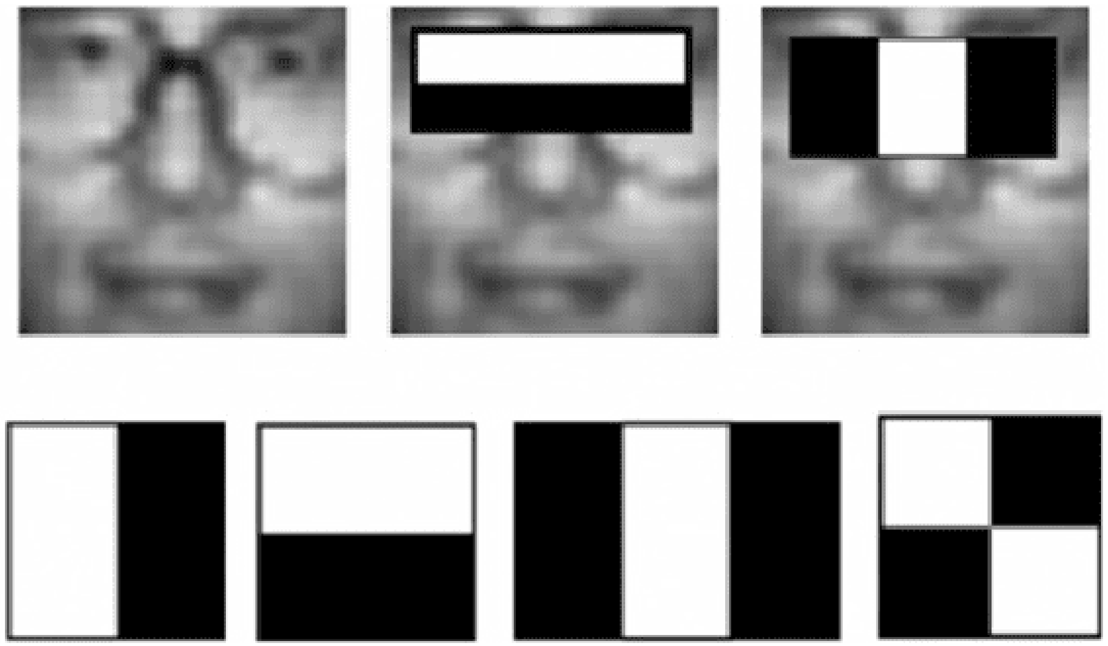

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

STEP 1 – Take Image as Input from a Camera

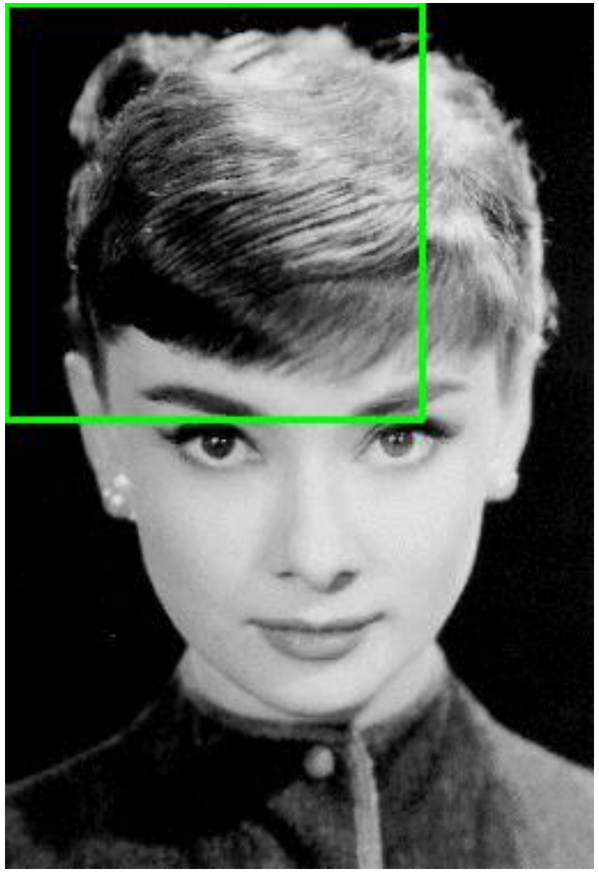
Camera

Image

STEP 2 – Detection using haar-cascade-classifier

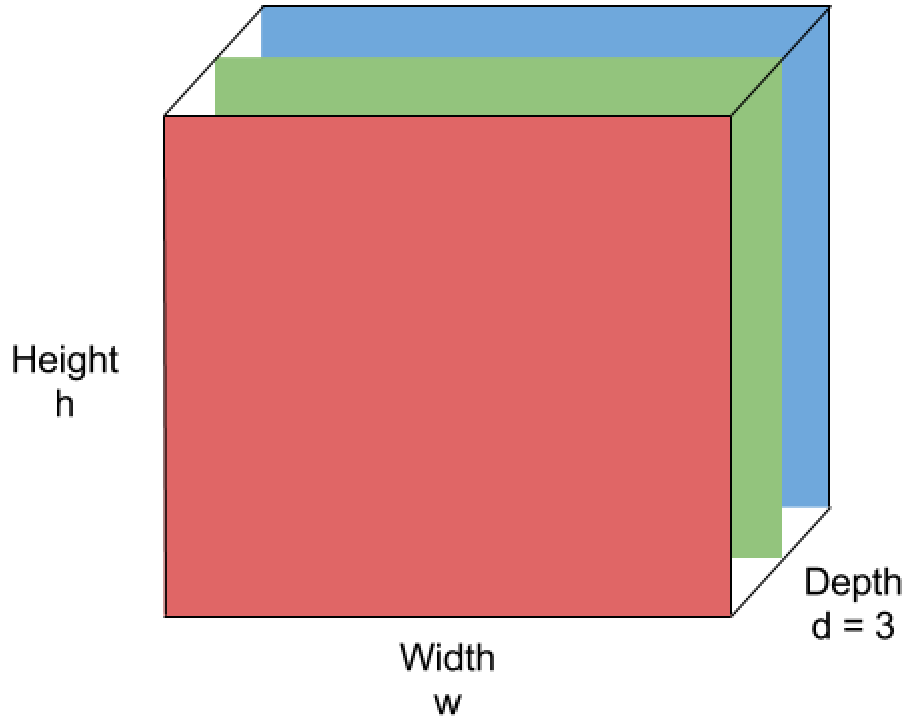


Haar-cascade

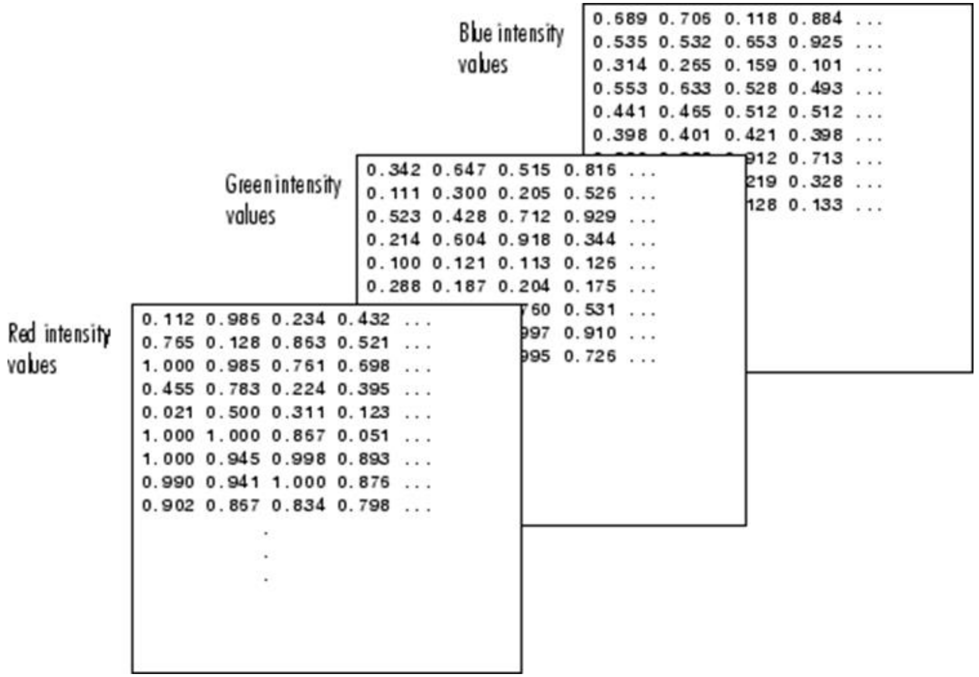


An example of a sliding window, moving from left-to-right and top-to-bottom, to locate the face in the image.

STEP 3 – Conversion of Image into Numpy Array



Image



Numpy Array

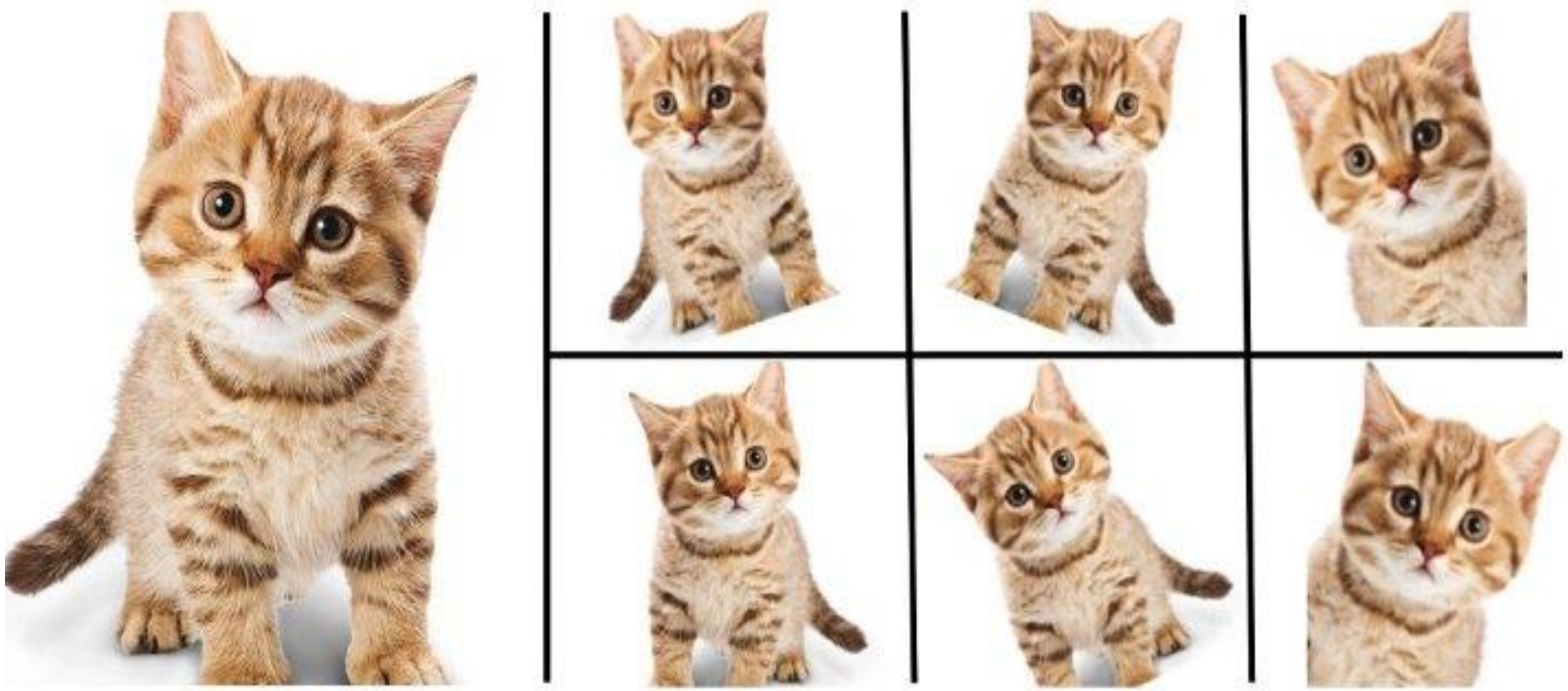
REASON FOR DOING THIS

What We See

08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

What Computers See

STEP 4 – Keras ImageDataGenerator for Image Augmentation



By Applying –

1. Random Rotations
2. Random Shifts
3. Random Flips
4. Random Brightness
5. Random Zoom

The ImageDataGenerator class in Keras is used for **implementing image augmentation**. The major advantage of the Keras ImageDataGenerator class is its ability to produce real-time image augmentation. This simply means it can generate augmented images dynamically during the training of the model making the overall mode more robust and accurate.

STEP 5 – Training the Model

THE CONVOLUTION STEP

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 Image

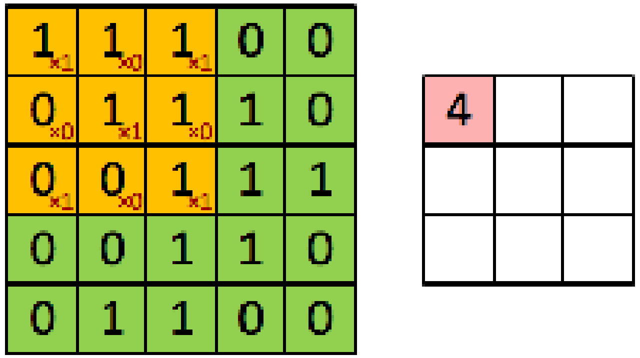
+

1	0	1
0	1	0
1	0	1

3 x 3 Filter

=

Stride = 1

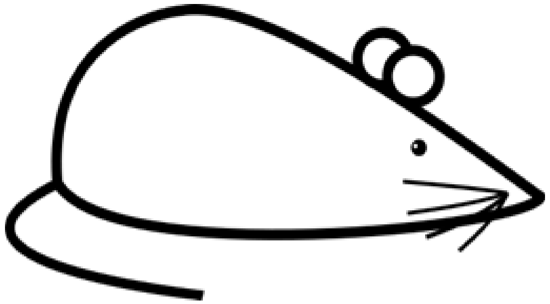


Image

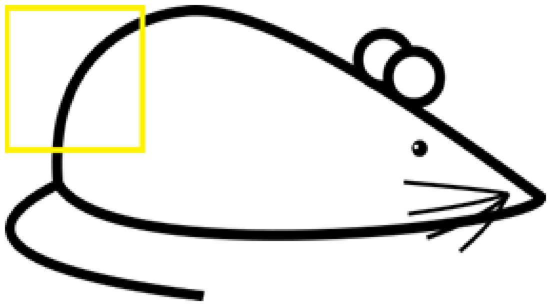
Convolved Feature

Stride : Stride is the number of pixels shifts over the input matrix.

THE CONVOLUTION STEP



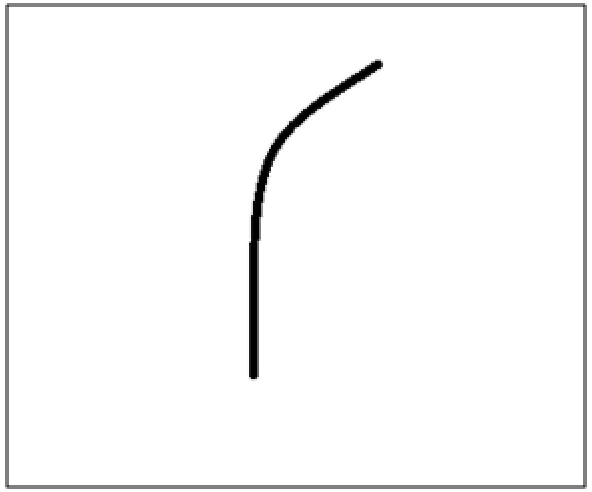
Original image



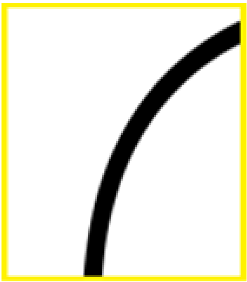
Visualization of the filter on the image

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

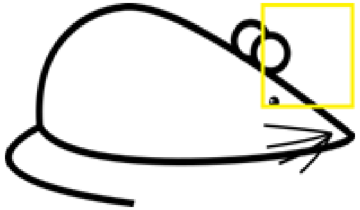
Pixel representation of the receptive field

$*$

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

$*$

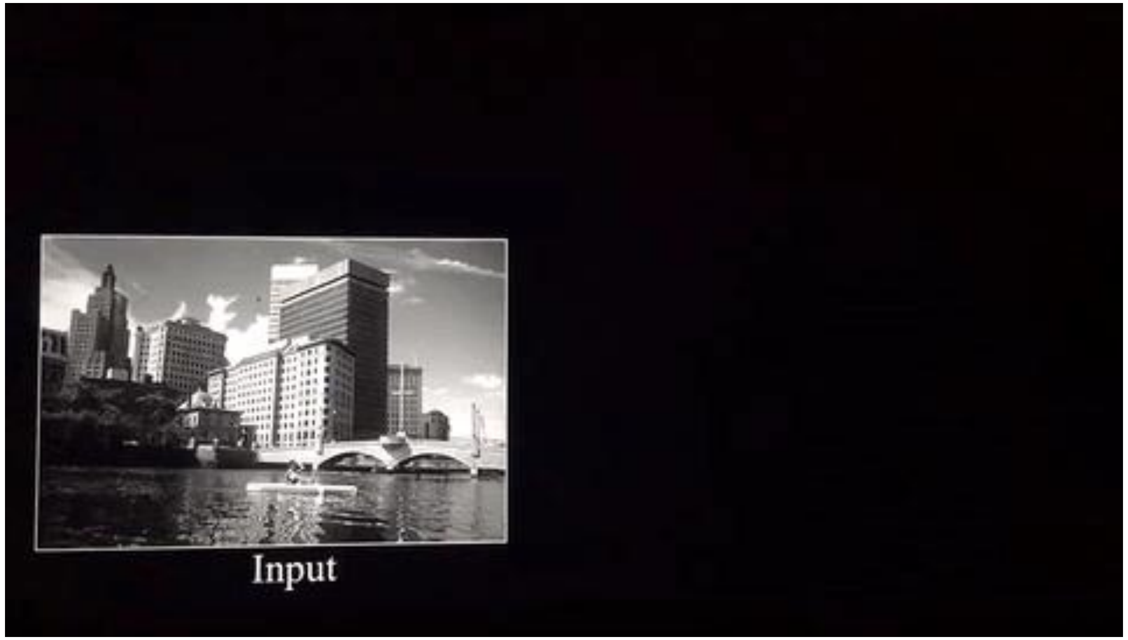
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

THE CONVOLUTION STEP

- This layer is the first layer that is used to extract the various features from the input images.
- In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM.
- By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).
- The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

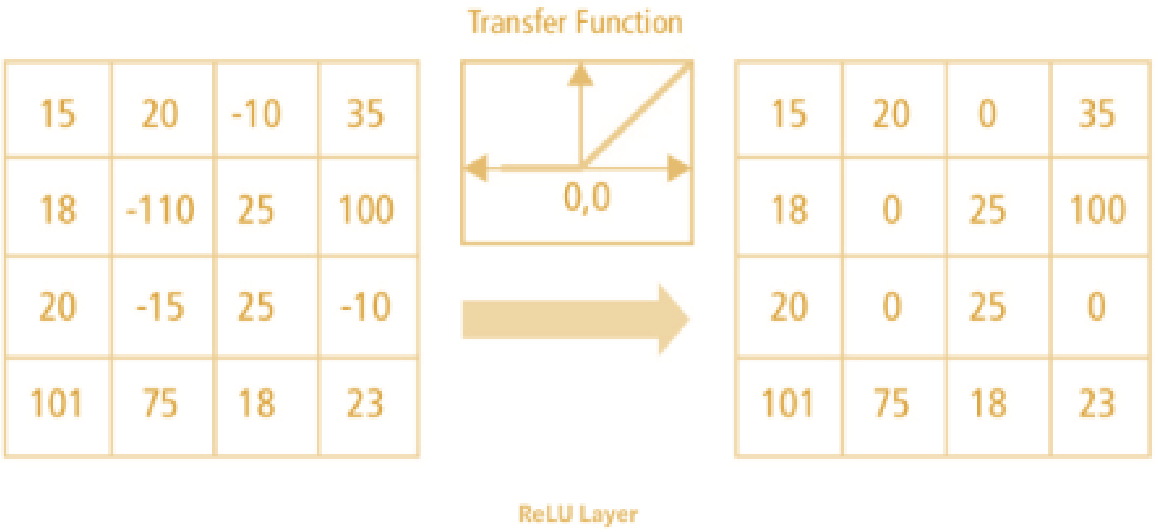


Padding :
Sometimes filter does not fit perfectly fit the input image. We have two options:
•Pad the picture with zeros (zero-padding) so that it fits
•Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

RECTIFIED LINEAR UNIT(RELU) STEP

ReLU stands for Rectified Linear Unit
for a non-linear operation.

The output is ***f(x) = max(0,x).***



15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

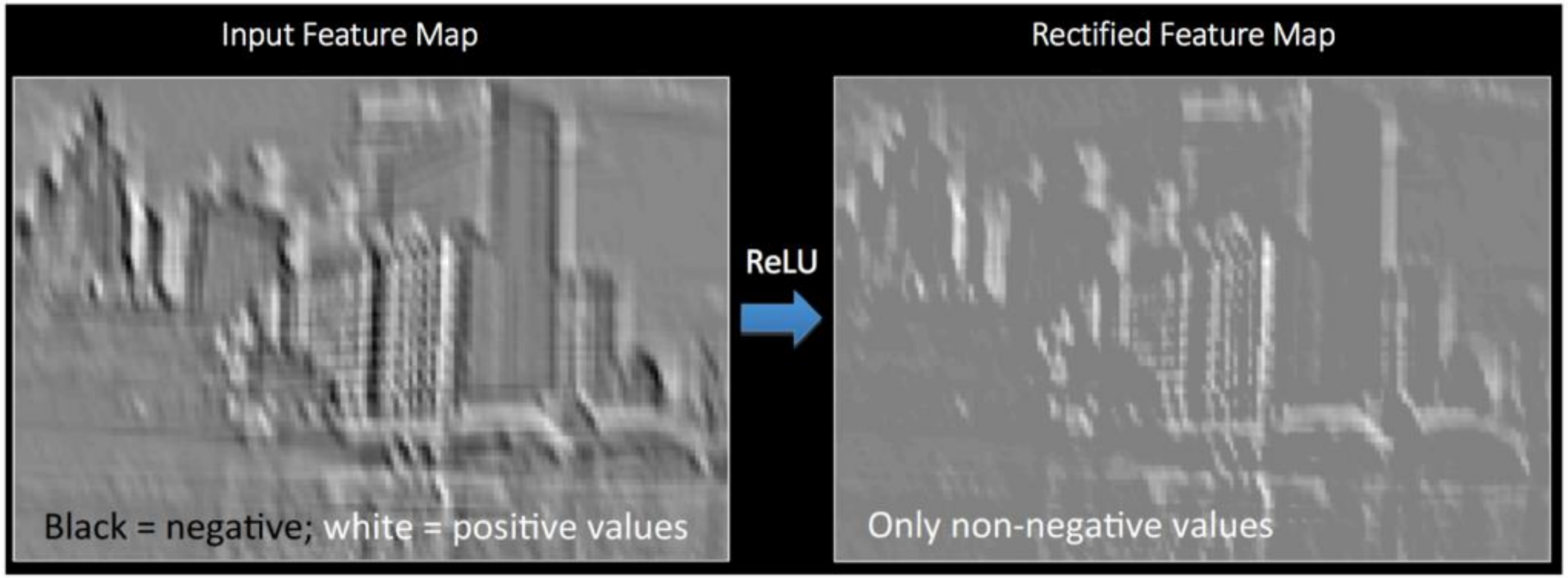
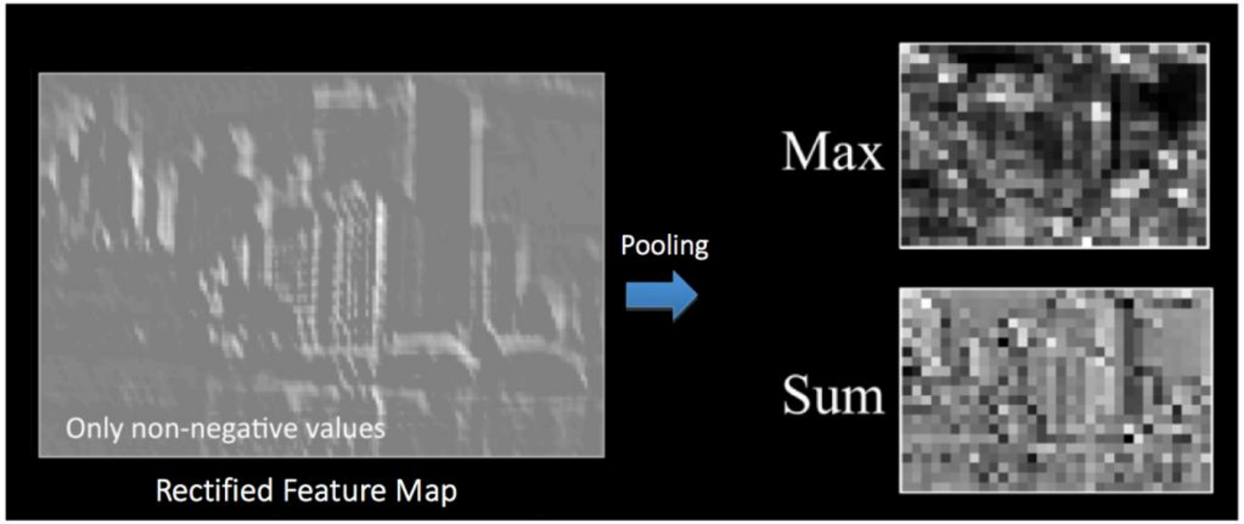


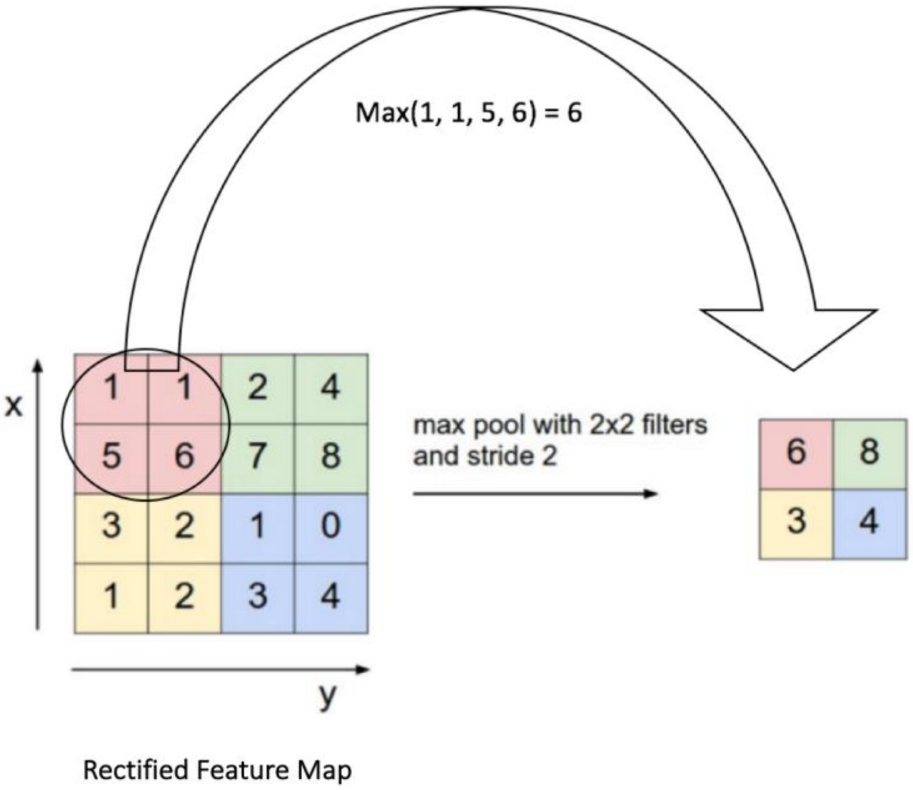
Image Conversion after Relu
Step

THE POOLING STEP

- Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

- In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window





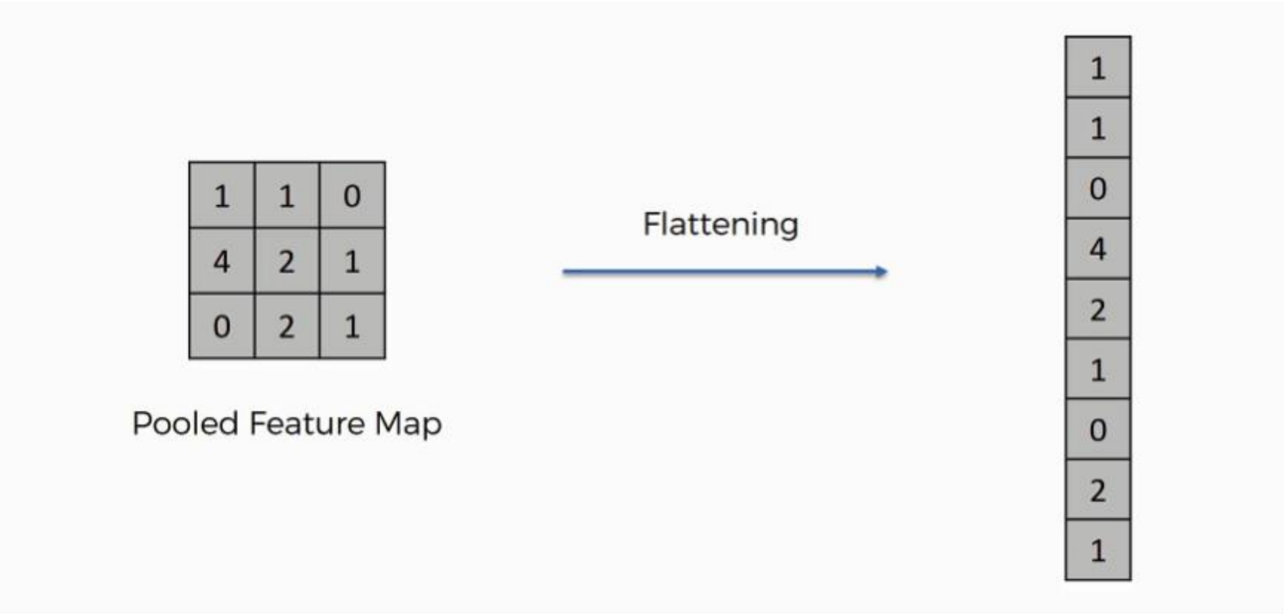
Max Pooling

THE FLATTENING STEP

- After a series of convolution and pooling operations on the feature representation of the image, we then flatten the output of the final pooling layers into a single long continuous linear array or a vector.

- The process of converting all the resultant 2-d arrays into a vector is called **Flattening**.

- Flatten output is fed as input to the fully connected neural network having varying numbers of hidden layers to learn the non-linear complexities present with the feature representation.

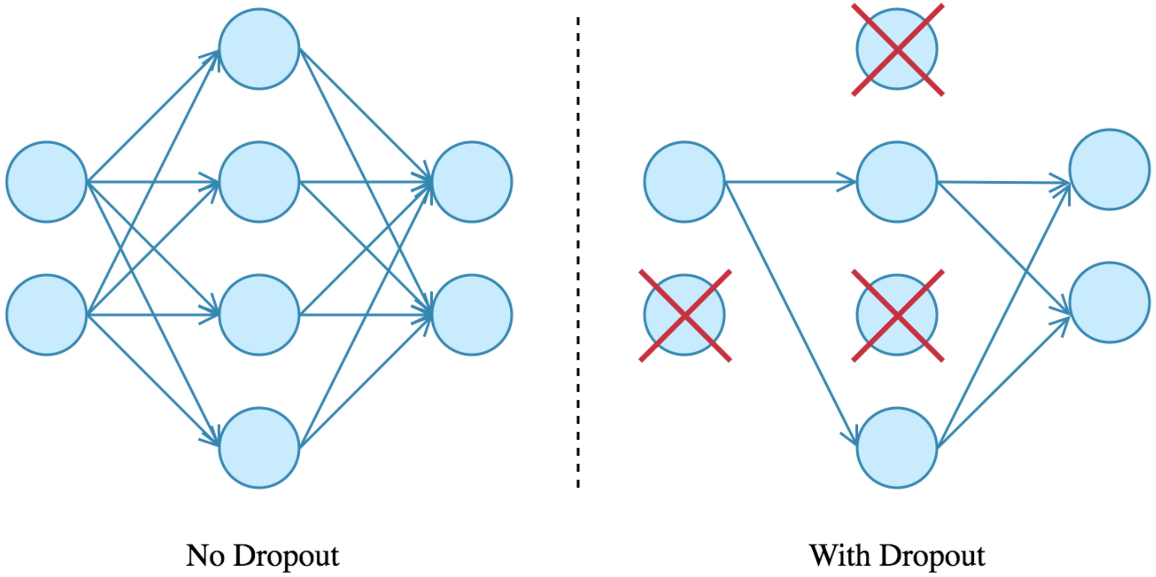


THE DROPOUT STEP

- When all the features are connected to the FC layer, it can cause **overfitting** in the training dataset.

- *Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.*

- To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

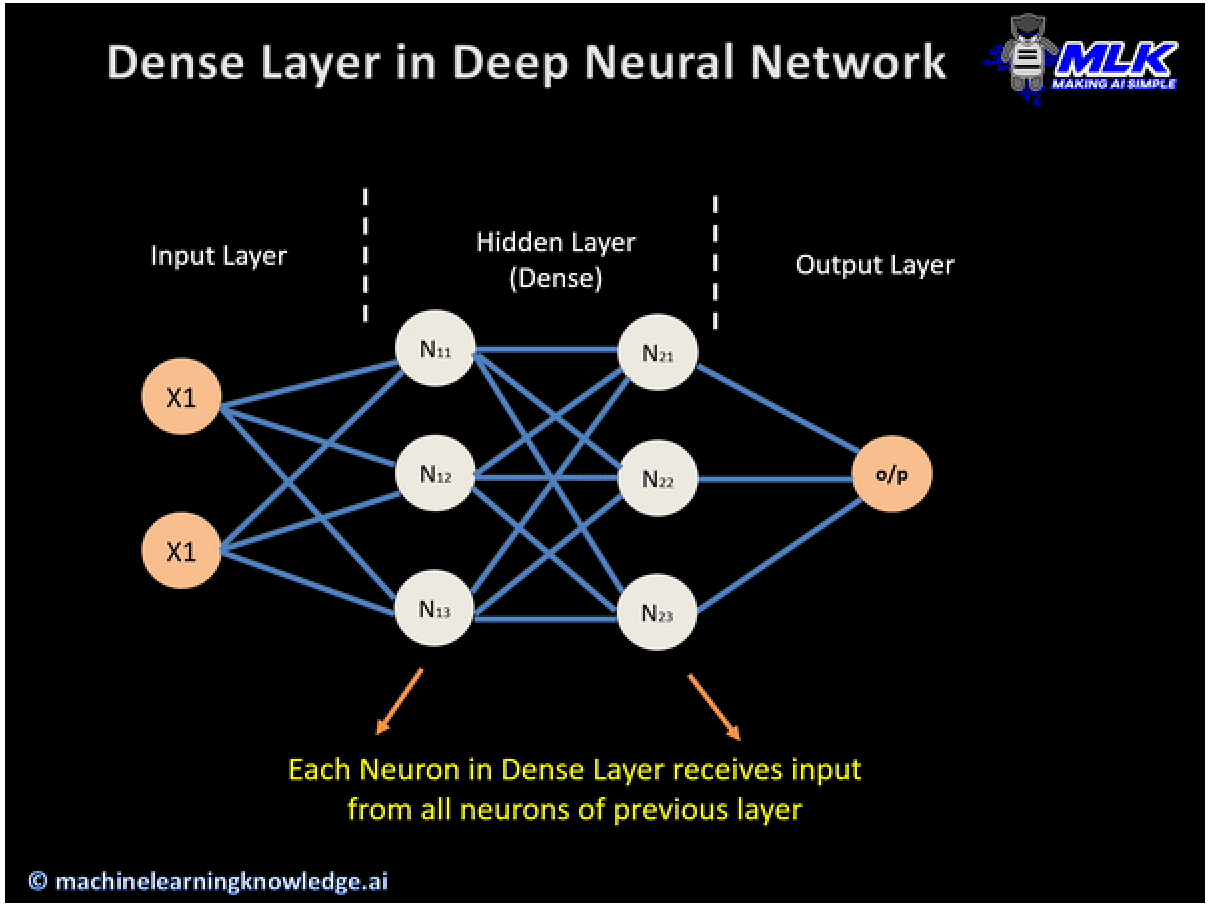


THE DENSE LAYER

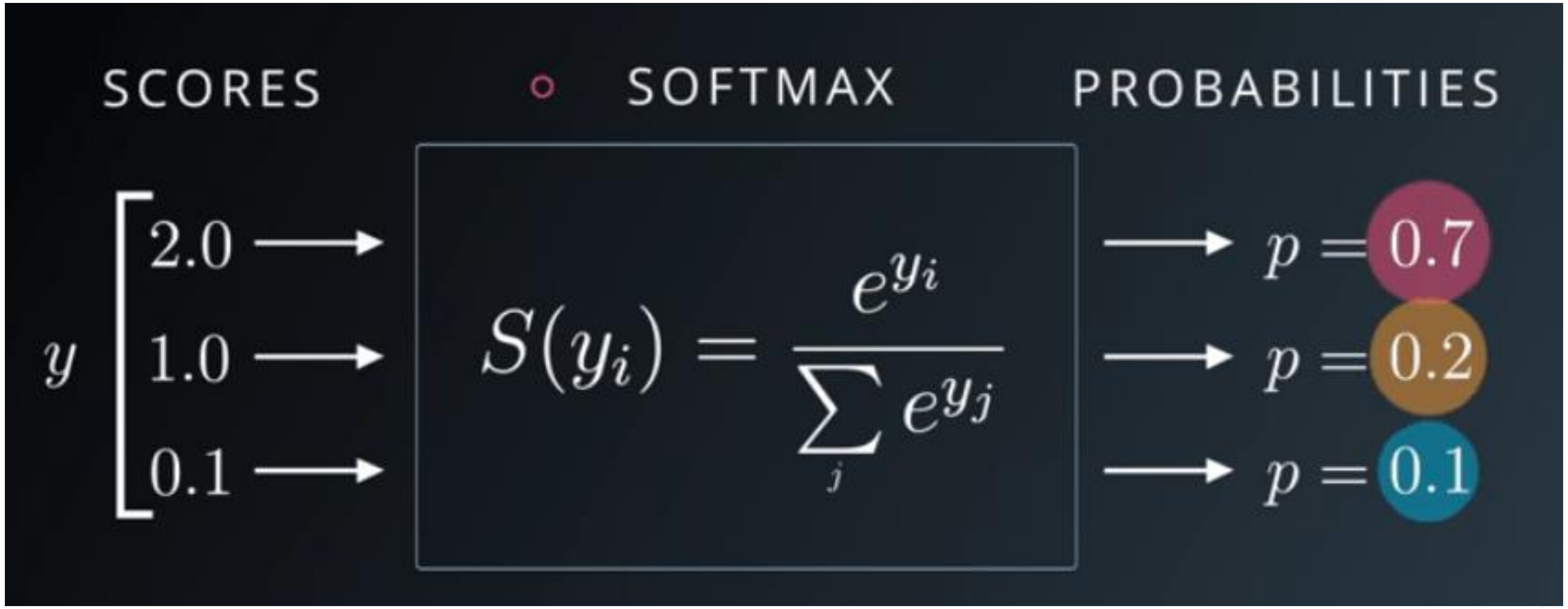
- The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer.

- In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of backpropagation.

- The output generated by the dense layer is an 'm' dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector.



SOFTMAX ACTIVATION FUNCTION



- **Softmax** is often used as ***the activation for the last layer of a classification network*** because the result could be interpreted as a probability distribution.

- **Softmax *assigns decimal probabilities to each class in a multi-class problem***.

Then we Train the Model for certain Epochs (Iterations)….

APPLICATIONS

- **Cars** - Drowsy driver detection methods can form the basis of a system to potentially reduce the number of crashes related to drowsy driving.

- **Anti Sleep Pilot** - Danish device that can be fitted to any vehicle, uses a combination of accelerometers and reaction tests.

- **Vigo** - Smart Bluetooth headset that detects signs of drowsiness through the eyes and head motion, and uses a combination of light, sound, and vibration to alert the user.

REFERENCES

- https://techvidvan.com/tutorials/driver-drowsiness-detection-system/
- https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/


Thank You!

ALGORITHM

- Let's now understand how our algorithm works step by step :-

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

With a webcam, we will take images as input. So to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by OpenCV, **cv2.VideoCapture(0)** to access the camera and set the capture object (cap). **cap.read()** will read each frame and we store the image in a frame variable.

ALGORITHM

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

  To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don’t need color information to detect the objects. We will be using haar cascade classifier to detect faces. This line is used to set our classifier **face = cv2.CascadeClassifier(‘ path to our haar cascade xml file’)**. Then we perform the detection using **faces = face.detectMultiScale(gray)**. It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

ALGORITHM

❖ ***Step 3 – Detect the eyes from ROI and feed it to the classifier***

The same procedure to detect faces is used to detect eyes. First, we set the cascade classifier for eyes in **leye** and **reye** respectively then detect the eyes using **left_eye = leye.detectMultiScale(gray)**. Now we need to extract only the eyes data from the full image. This can be achieved by extracting the boundary box of the eye and then we can pull out the eye image from the frame with this code.

ALGORITHM

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

  To feed our image into the model, we need to perform certain operations because the model needs the correct dimensions to start with. First, we convert the color image into grayscale using **r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY)**. Then, we resize the image to 24*24 pixels as our model was trained on 24*24 pixel images **cv2.resize(r_eye, (24,24))**. We normalize our data for better convergence **r_eye = r_eye/255** (All values will be between 0-1). Expand the dimensions to feed into our classifier. We loaded our model using **model = load_model('models/cnnCat2.h5')** . Now we predict each eye with our model
  lpred = model.predict_classes(l_eye). If the value of lpred[0] = 1, it states that eyes are open, if value of lpred[0] = 0 then, it states that eyes are closed.

ALGORITHM

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

  The score is basically a value we will use to determine how long the person has closed his eyes. So if both eyes are closed, we will keep on increasing score and when eyes are open, we decrease the score. We are drawing the result on the screen using cv2.putText() function which will display real time status of the person.

SOURCE CODE

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time
mixer.init()
sound = mixer.Sound('alarm.wav')
face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')
lbl=['Close','Open']
model = load_model('models/cnncat2.h5’)
..................
```