
DRIVER DROWSINESS
DETECTION SYSTEM

A countless number of people drive on the highway day and night. Taxi drivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep. Due to which it becomes very dangerous to drive when feeling sleepy.

The majority of accidents happen due to the drowsiness of the driver. So, to prevent these accidents we will build a drowsiness detection system using Python, OpenCV, and Keras which will alert the driver when he feels sleepy.

Problem Statement

- *In this system the driver assistance System is presented in order to reduce the number of accidents caused by driver fatigue and thus improve road safety.*
- *This system treats the automatic detection of driver drowsiness based on visual information and transfer learning .We locate ,track and analyze the driver face and eyes and check whether the driver is drowsy .*

REQUIREMENTS

- Python - jupyter notebook.
- OpenCV - Face and eye detection.
- Keras and transfer learning- To build our classification model.
- TensorFlow - Keras uses TensorFlow as backend.
- Pygame - To play alarm sound.

LITERATURE REVIEW

- In computer science, image processing is the use of computer algorithms to perform image processing on images.
- As a subcategory or field of digital signal processing, image processing has many advantages over analog image processing.

NOVELTY OR UNIQUENESS

- We have used the transfer learning model than CNN nueral network as it is more accurate than CNN

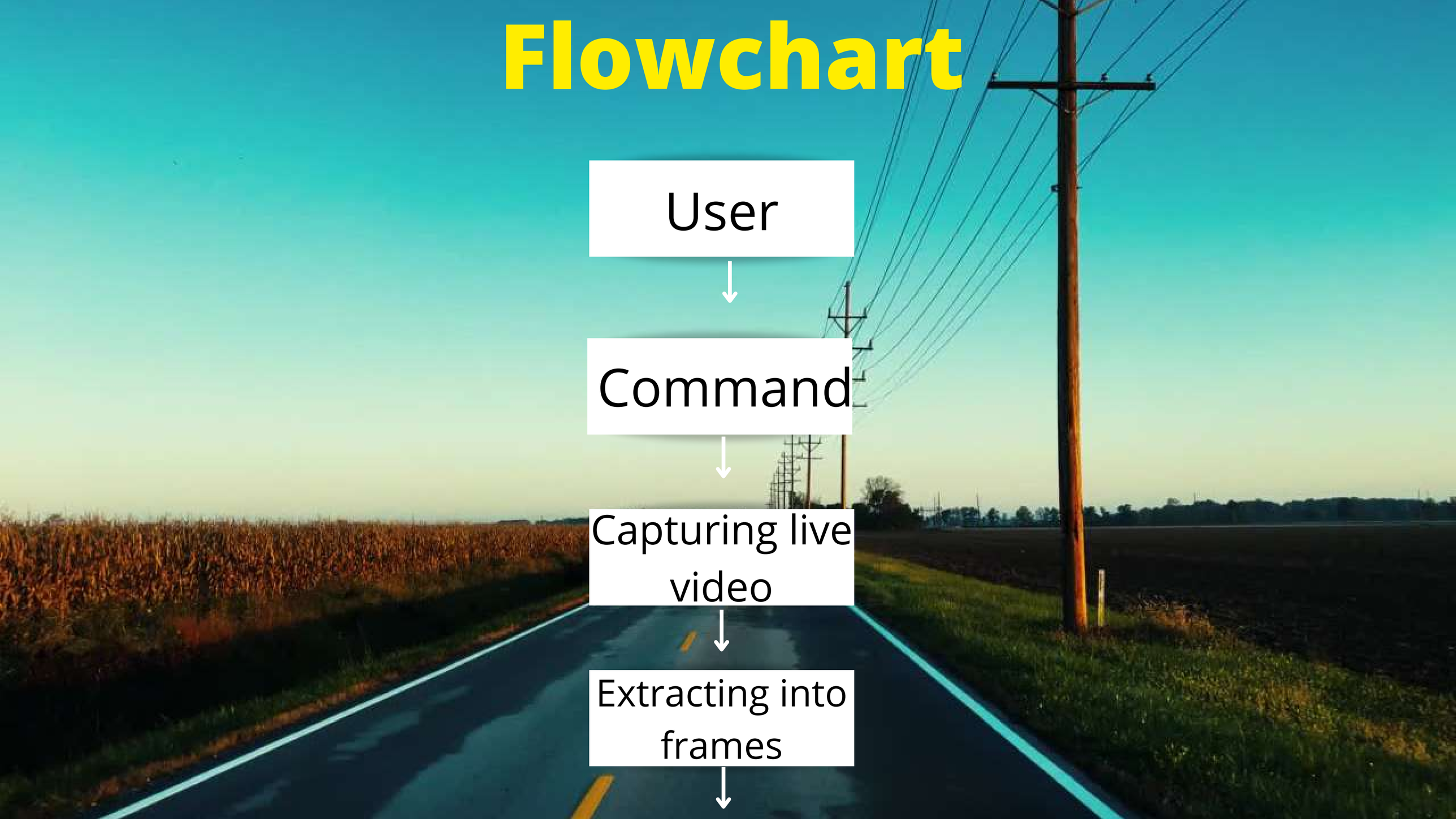
LIMITATIONS OF OUR PROJECT

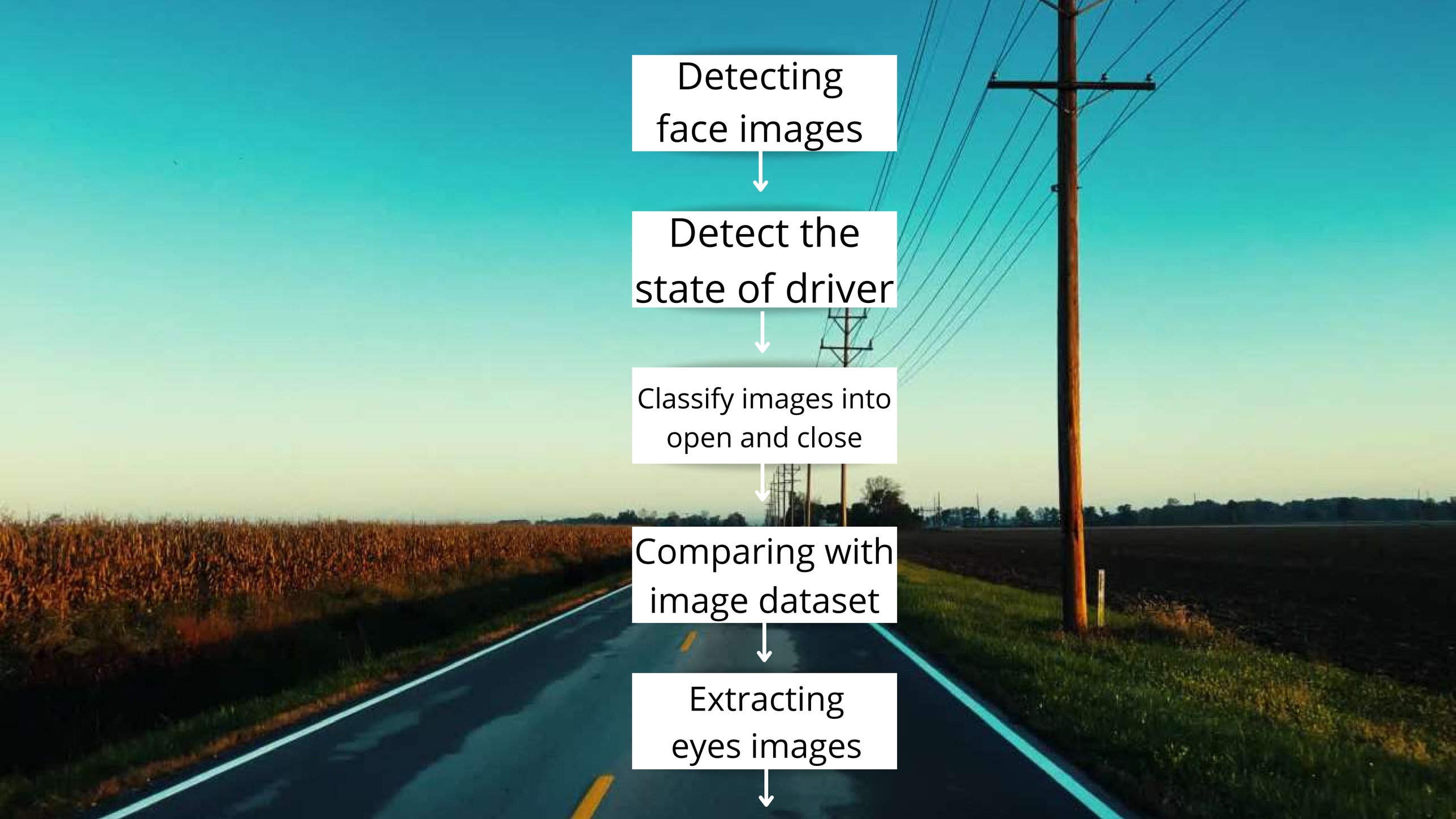
- Our project is not working with the people wearing spectacles as we require more dataset and more processing.
- Our project requires plain background.
- Our project is a bit lagging.

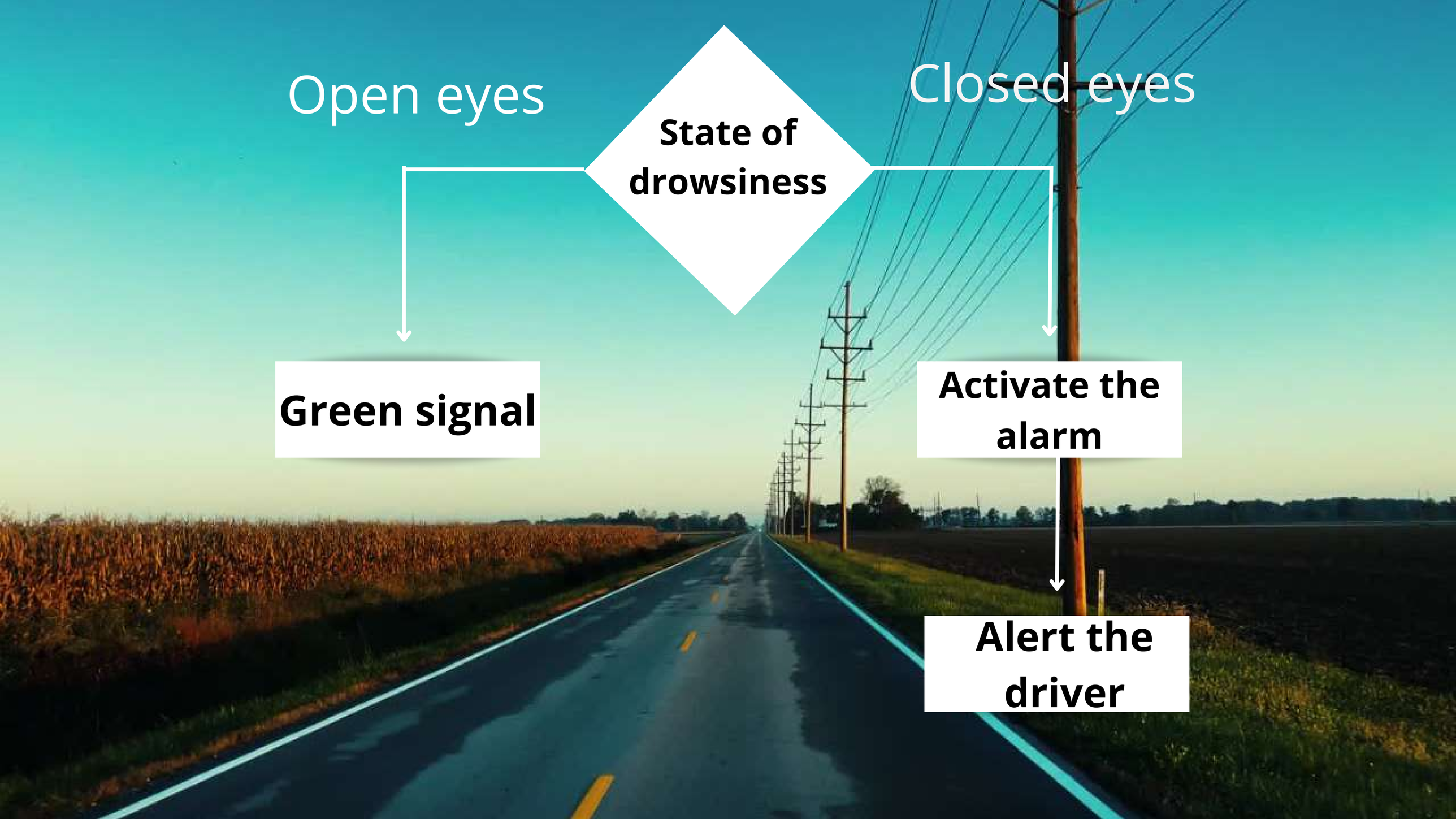
THE MODEL ARCHITECTURE

- *The model we used is built with Keras, tensor flow and transfer learning. Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.*

Flowchart




Detecting face images
Detect the state of driver
Classify images into open and close
Comparing with image dataset
Extracting eyes images


Open eyes
State of drowsiness
Closed eyes
Green signal
Activate the alarm
Alert the driver

In this Python project, we will be using OpenCV for gathering the images from webcam and feed them into a Deep Learning model which will classify whether the person's eyes are 'Open' or 'Closed'. The approach we will be using for this Python project is as follows :

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.

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 will use the method provided by OpenCV.

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.**

STEP 3

Detect the eyes from ROI and feed it to the classifier

- First, we set the cascade classifier for eyes in eye and reye respectively then detect the eyes.
- Now we need to extract only the eyes data from the full image.
- This can be achieved by extracting the eye's boundary box and then pulling out the eye image from the frame with this code.

STEP 4

Classifier will Categorize whether Eyes are Open or Closed

- We are using transfer learning model to classify our images into open and closed eyes .

STEP 5

Calculate Score to Check whether Person is Drowsy

- The score is 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 the score and when eyes are open, we decrease the score.
- And if the score is more than 15 then it will beep the alarm to alert the driver of the vehicle.

SOURCE CODE

Dataset

We have taken a MRLEYES dataset from google which contains face images of 36 different persons in 36 folders , each folder containing about 1200-2000 face images. So we have created a code to classify these 75000 images into two subfolders i.e. closed and open eyes. Then we have created a test folder containing 2500 images of both subfolders and a train folder containing both the subfolder.

In [1]:

```
import os
import shutil
import glob
from tqdm import tqdm
```

-We have imported some libraries.

```
In [2]:

Raw_DIR= r'C:\Users\ARIHANT\Desktop\driver drowsiness\mrlEyes_2018_01\mrlEyes_2018_01'
for dirpath, dirname, filenames in os.walk(Raw_DIR):
    for i in tqdm([f for f in filenames if f.endswith('.png')]):
        if i.split('_')[4]=='0':
            shutil.copy(src=dirpath+'/'+i, dst=r'C:\Users\ARIHANT\Desktop\driver drowsiness

        elif i.split('_')[4]=='1':
            shutil.copy(src=dirpath+'/'+i, dst=r'C:\Users\ARIHANT\Desktop\driver drowsiness
```

We have divided our database into two categories:

1)open eyes **2)closed eyes**

OUTPUT

```
0it [00:00, ?it/s]
100%|██████████| 1114/1114 [00:13<00:00, 84.39it/s]
100%|██████████| 679/679 [00:08<00:00, 76.08it/s]
100%|██████████| 1069/1069 [00:15<00:00, 67.53it/s]
100%|██████████| 736/736 [00:10<00:00, 67.85it/s]
100%|██████████| 1012/1012 [00:14<00:00, 71.57it/s]
100%|██████████| 624/624 [00:08<00:00, 69.85it/s]
100%|██████████| 832/832 [00:12<00:00, 64.38it/s]
100%|██████████| 387/387 [00:05<00:00, 70.85it/s]
100%|██████████| 399/399 [00:05<00:00, 78.57it/s]
100%|██████████| 1648/1648 [00:18<00:00, 87.86it/s]
100%|██████████| 8728/8728 [01:36<00:00, 90.23it/s]
100%|██████████| 3605/3605 [00:37<00:00, 95.17it/s]
100%|██████████| 8884/8884 [01:36<00:00, 92.43it/s]
100%|██████████| 1132/1132 [00:12<00:00, 93.45it/s]
100%|██████████| 1889/1889 [00:22<00:00, 82.87it/s]
100%|██████████| 1648/1648 [00:22<00:00, 74.53it/s]
100%|██████████| 4410/4410 [00:54<00:00, 80.61it/s]
100%|██████████| 6175/6175 [01:18<00:00, 78.59it/s]
100%|██████████| 704/704 [00:09<00:00, 73.68it/s]
100%|██████████| 987/987 [00:11<00:00, 88.23it/s]
```

SOURCE CODE

(MODEL)

In [1]:

```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data
```

We have imported some libraries.

```
In [2]: tf.test.is_gpu_available()

WARNING:tensorflow:From C:\Users\ARIHANT\AppData\Local\Temp\ipykernel_20024\337460
670.py:1: is_gpu_available (from tensorflow.python.framework.test_util) is depreca
ted and will be removed in a future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
Out[2]: False
```

We have checked if gpu is available or not.

In [3]:
```
batchsize=8
```

In [4]:
```
train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0
    zoom_range=0.2,width_shift_range=0.2,
    height_shift_range=0.2, validation_split=0.2)

train_data= train_datagen.flow_from_directory(r'C:\Users\ARIHANT\Desktop\driver dr
                                              target_size=(80,80),batch_size=batchsize,class_mode

validation_data= train_datagen.flow_from_directory(r'C:\Users\ARIHANT\Desktop\drive
                                              target_size=(80,80),batch_size=batchsize,class_mode
```

```
Found 61178 images belonging to 2 classes.
Found 15294 images belonging to 2 classes.
```

We have set batchsize to 8,we are classifying images into 2 classes .

In [5]:
```
test_datagen = ImageDataGenerator(rescale=1./255)

test_data = test_datagen.flow_from_directory(r'C:\Users\ARIHANT\Desktop\driver dro
                                             target_size=(80,80),batch_size=batchsize,class_mod
```

```
Found 5184 images belonging to 2 classes.
```

Rescale

In [6]:

```
bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shap
hmodel = bmodel.output
hmodel = Flatten()(hmodel)
hmodel = Dense(64, activation='relu')(hmodel)
hmodel = Dropout(0.5)(hmodel)
hmodel = Dense(2,activation= 'softmax')(hmodel)

model = Model(inputs=bmodel.input, outputs= hmodel)
for layer in bmodel.layers:
    layer.trainable = False
```

We have used trasnfer learning to import the model using inception V3 which converted grey scale images into RGB.

model

```
In [7]: model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 80, 80, 3)]       0         []

 conv2d (Conv2D)             (None, 39, 39, 32)        864       ['input_1[0][0]']

 batch_normalization (BatchNorm  (None, 39, 39, 32)    96        ['conv2d[0][0]']
 alization)

 activation (Activation)     (None, 39, 39, 32)        0         ['batch_normaliza
tion[0][0]']

 conv2d_1 (Conv2D)           (None, 37, 37, 32)        9216      ['activation[0]
[0]']

 batch_normalization_1 (BatchNo  (None, 37, 37, 32)    96        ['conv2d_1[0]
[0]']
 rmalization)

 activation_1 (Activation)   (None, 37, 37, 32)        0         ['batch_normaliza
tion_1[0][0]']

 conv2d_2 (Conv2D)           (None, 37, 37, 64)        18432     ['activation_1[0]
[0]']

 batch_normalization_2 (BatchNo  (None, 37, 37, 64)    192       ['conv2d_2[0]
[0]']
```

9/21/22, 9:59 PM model

In [8]:
```
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPl
```

In [9]:
```
checkpoint = ModelCheckpoint(r'C:\Users\ARIHANT\Desktop\driver drowsiness\models\m
                             monitor='val_loss',save_best_only=True,verbose=3)

earlystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3, restore_be

learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

callbacks=[checkpoint,earlystop,learning_rate]
```

Now we are saving our model.

In [ ]:

In [ ]:
```
model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy

model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
                    validation_data=validation_data,
                    validation_steps=validation_data.samples//batchsize,
                    callbacks=callbacks,
                     epochs=20)
```

In [ ]:
```
acc_tr, loss_tr = model.evaluate_generator(train_data)
print(acc_tr)
print(loss_tr)
```

In [ ]:
```
acc_vr, loss_vr = model.evaluate_generator(validation_data)
print(acc_vr)
print(loss_vr)
```

In [ ]:
```
acc_test, loss_test = model.evaluate_generator(test_data)
print(acc_tr)
print(loss_tr)
```

In [ ]:

Now we are training our model:

1)Training train data 2)Training validation data

3)Training test data

SOURCE CODE

(MAIN)

In [1]:

```
import cv2
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
from pygame import mixer
```

```
pygame 2.1.2 (SDL 2.0.18, Python 3.9.12)
Hello from the pygame community. https://www.pygame.org/contribute.html (htt
ps://www.pygame.org/contribute.html)
```

-We have imported some libraries.

In [2]:

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_defau
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
model = load_model(r'C:\Users\ARIHANT\Desktop\driver drowsiness\models\model.h5')
```

-We have use haarcascade to detect the face quickly, and then loaded our model.

In [3]:

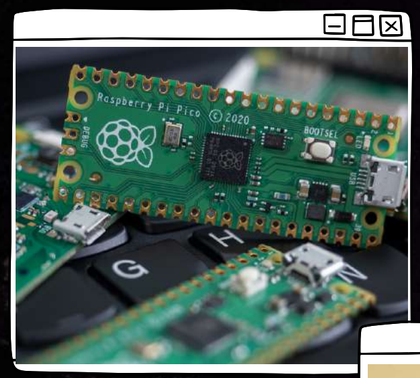
```
mixer.init()
sound= mixer.Sound(r'C:\Users\ARIHANT\Desktop\driver drowsiness\alarm.wav')
cap = cv2.VideoCapture(0)
Score = 0
while True:
    ret, frame = cap.read()
    height,width = frame.shape[0:2]
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces= face_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)
    eyes= eye_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)

    cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)
```

-Taking the file of alarm from our directory and then capturing our video.

```
for (x,y,w,h) in faces:
    cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3 )

for (ex,ey,ew,eh) in eyes:
    #cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3 )

    # preprocessing steps
    eye= frame[ey:ey+eh,ex:ex+w]
    eye= cv2.resize(eye,(80,80))
    eye= eye/255
    eye= eye.reshape(80,80,3)
    eye= np.expand_dims(eye,axis=0)
    # preprocessing is done now model prediction
    prediction = model.predict(eye)
```

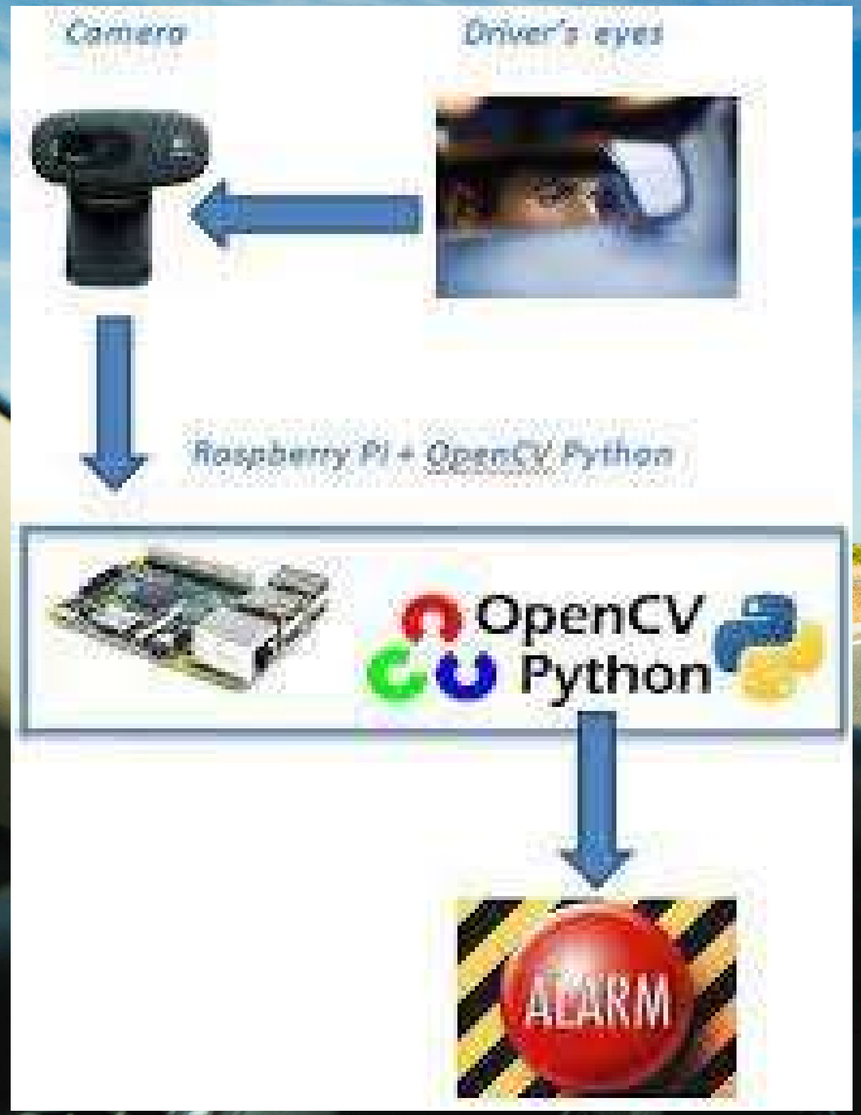
-Processing the steps to make region of interest i.e. in this case eyes.

```
        # if eyes are closed
        if prediction[0][0]>0.30:
            cv2.putText(frame,'closed',(10,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMA
                        thickness=1,lineType=cv2.LINE_AA)
            cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT_HERSHEY_
                        thickness=1,lineType=cv2.LINE_AA)
            Score=Score+1
            if(Score>15):
                try:
                    sound.play()
                except:
                    pass

        # if eyes are open
        elif prediction[0][1]>0.90:
            cv2.putText(frame,'open',(10,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL
                        thickness=1,lineType=cv2.LINE_AA)
            cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT_HERSHEY_
                        thickness=1,lineType=cv2.LINE_AA)
            Score = Score-1
            if (Score<0):
                Score=0


    cv2.imshow('frame',frame)
    if cv2.waitKey(33) & 0xFF==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



REAL WORLD APPLICATION:

- This System can be used in any automobiles with ease.
- This can be done by using Raspberry pi and with the help of a Web camera and an alarm for the alarm sound .



- The Driver eyes are constantly monitered by Webcam.
- This Data is sent to the image classifier model which is running on Raspberry Pi.
- The classifier checks every frame and classifies the frame as open or close.
- If the eyes are being closed for a few seconds straight then the alarm is Triggered.
- The alarm goes off when the driver is active and his eyes are open without any drowsiness.

CONCLUSION

- This system can be used to reduce the amount of roads accidents that happens to great extent.
- This can save a lot of lives, which is a main motive of this system.
- This system does not need any complex system to work effectively.
- Taking the facts into consideration driver drownsiness detection system is the future of road safety.


THANK YOU !!