Drivers Drowsiness Detection

Models

Four models we will talk about

The first model will determine if the eye is closed or open using CNN

- The idea of this model :

is to identify the person’s face, then determine the eyes from the face, and then determine whether it is open or closed.

- Note: the input stream will be converted into grayscale images.

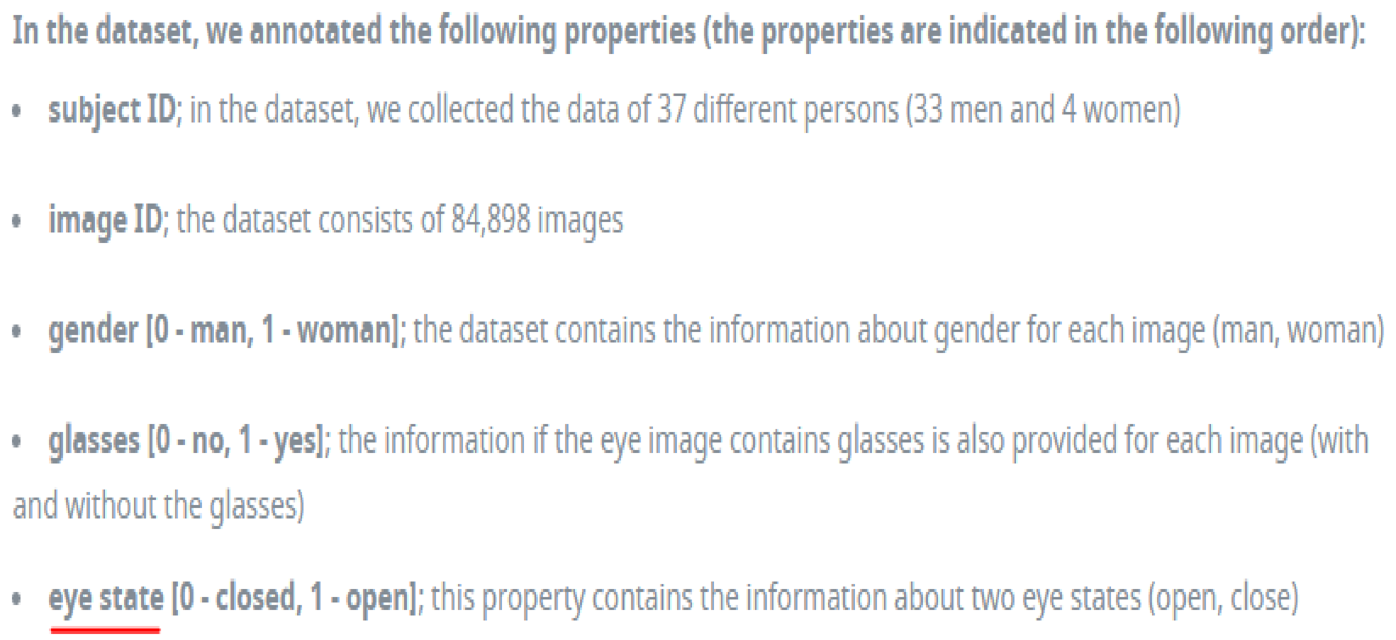
- The implementation using:

  - Tensorflow / keras
  - Opencv for computer vision

  And some other libraries that will be discussed

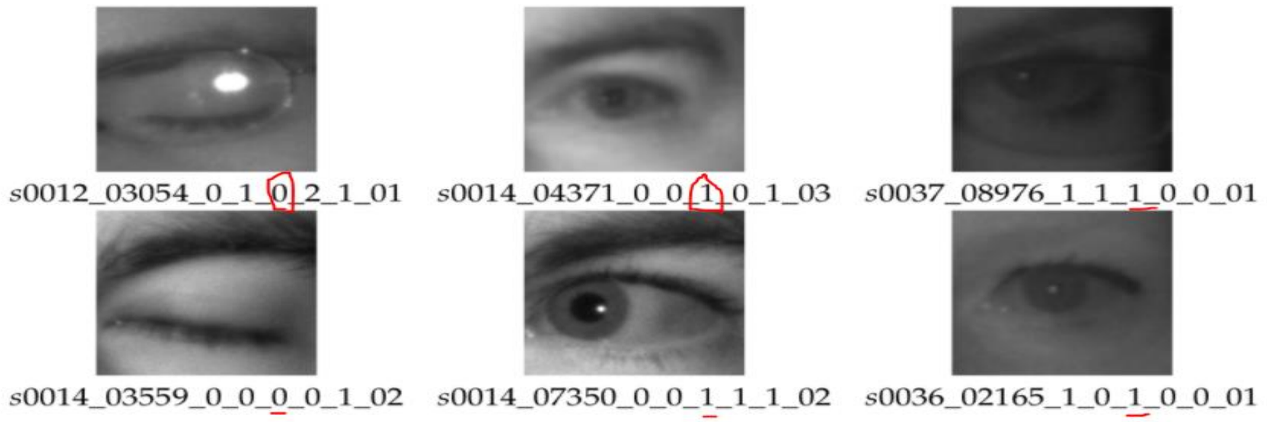
Method:

- I apply customized deep learning model based on eyes data set which is (MRL eye dataset) (link)
- This data was organized by using a simple automated python (organize_mrl_data.py) file
- **Eye state [0-closed, 1- open] = index 16**



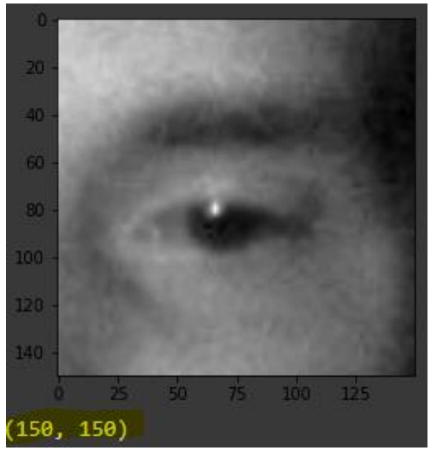
In the dataset, we annotated the following properties (the properties are indicated in the following order):

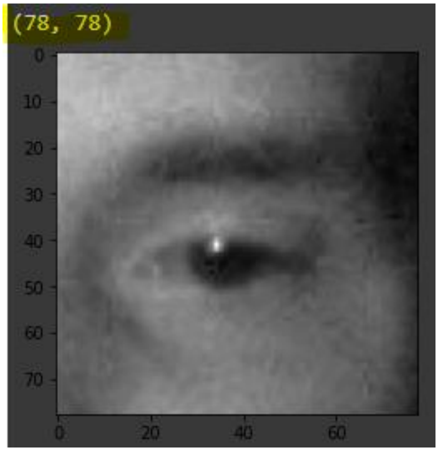
- **subject ID**; in the dataset, we collected the data of 37 different persons (33 men and 4 women)
- **image ID**; the dataset consists of 84,898 images
- **gender [0 - man, 1 - woman]**; the dataset contains the information about gender for each image (man, woman)
- **glasses [0 - no, 1 - yes]**; the information if the eye image contains glasses is also provided for each image (with and without the glasses)
- **eye state [0 - closed, 1 - open]**; this property contains the information about two eye states (open, close)

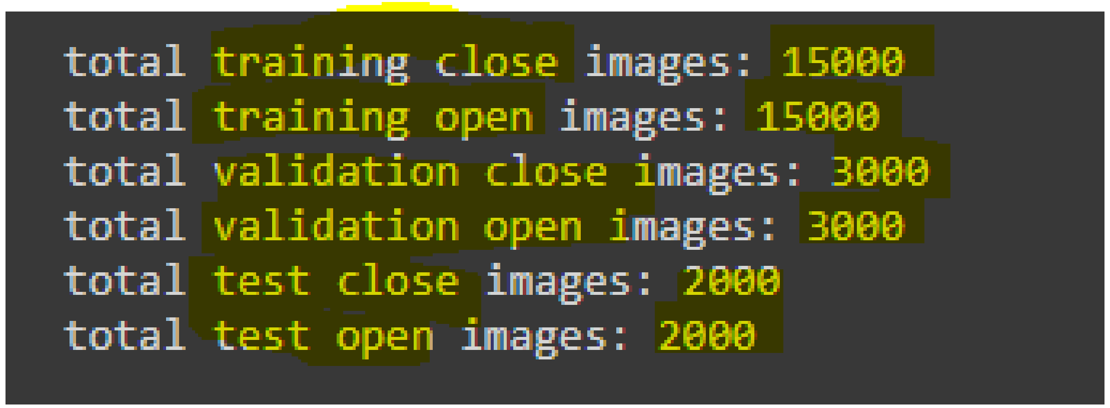


Data processing :

- Before we build the model, we must first prepare the data. I noticed that the size of the images in the training was 78 * 78 pixels, which is small, so I increased it to 150 * 150 pixels.
- After that, I created an automated code to organize training , validation, and test data,which are as follows:







```
total training close images: 15000
total training open images: 15000
total validation close images: 3000
total validation open images: 3000
total test close images: 2000
total test open images: 2000
```

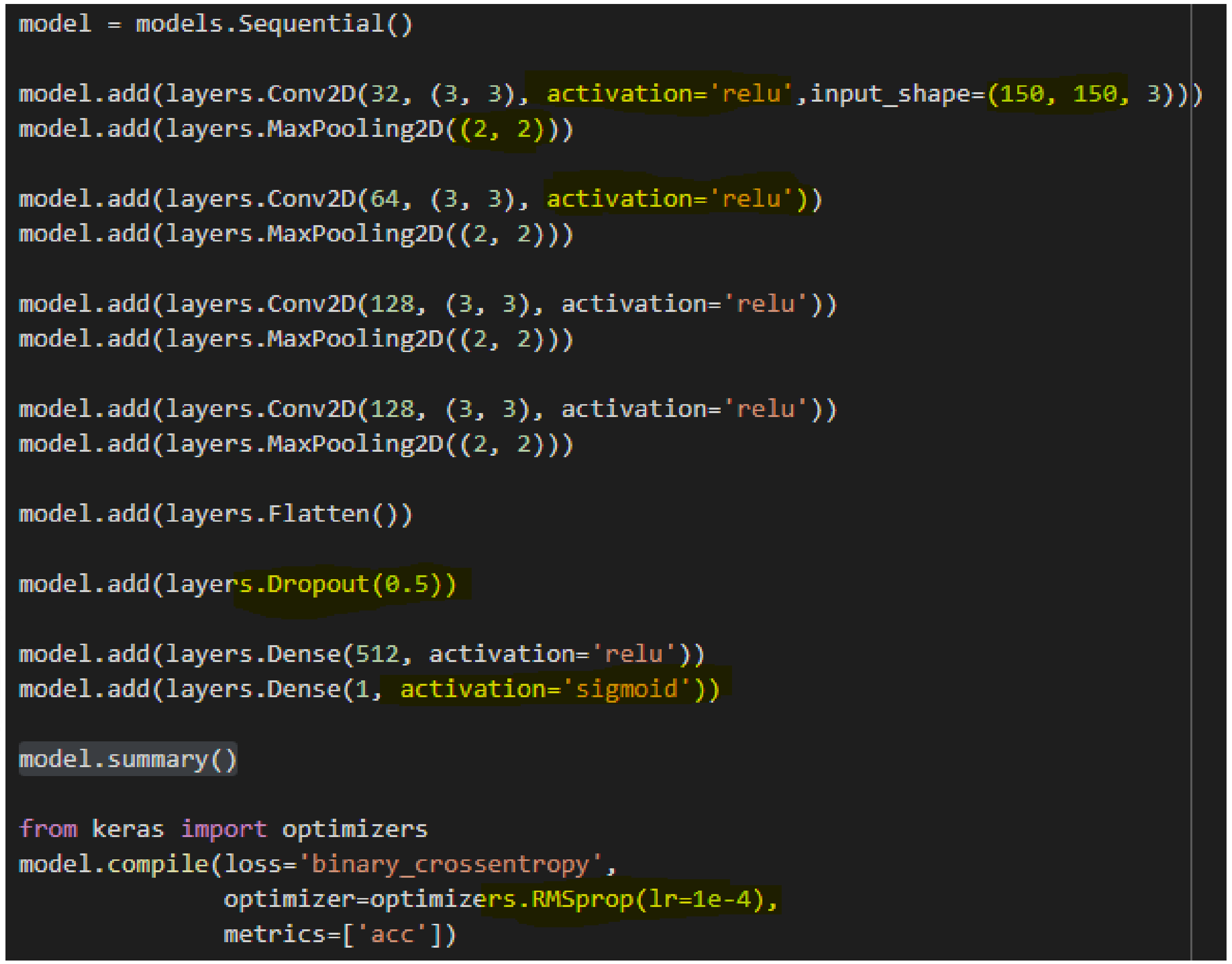
Model architecture :

- I am using (keras) library to build a CNN model.
- As we can see I started the

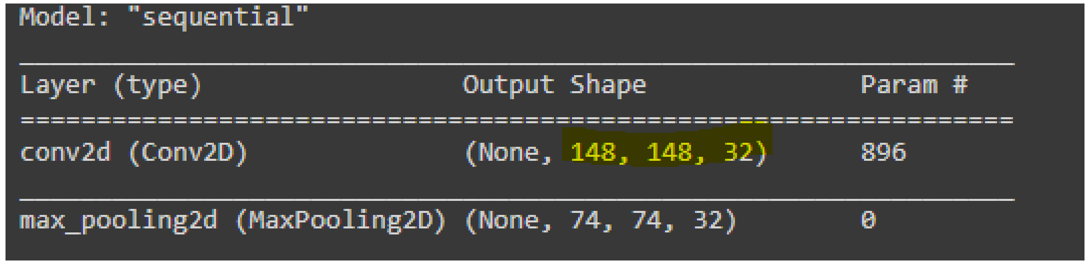
first layer with 32 kernels(filters) and use (relu) activation function.

And the second layer using (2,2)maxpooling

Which mean this layer will divide the output from the brevise layer into (4)



```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu',input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dropout(0.5))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

from keras import optimizers
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])
```

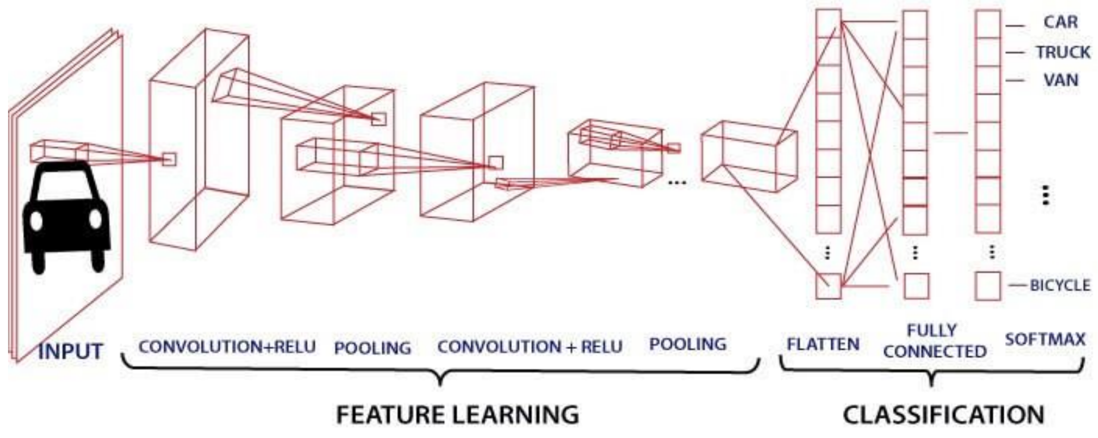


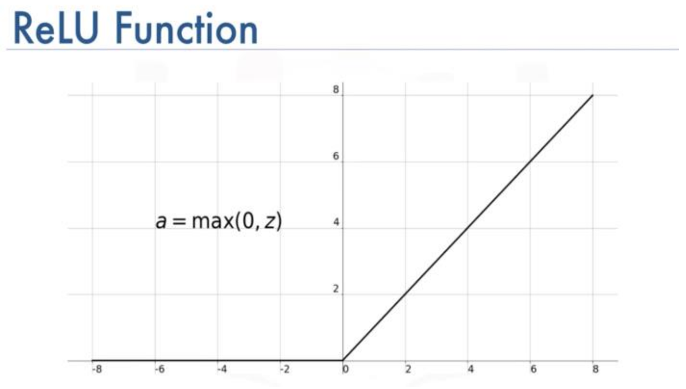
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
```

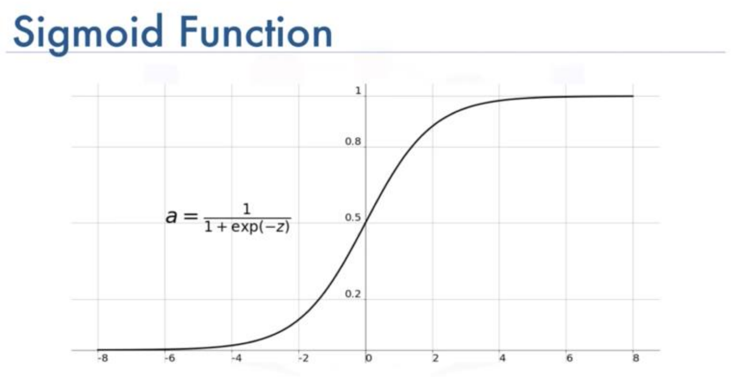
Model architecture (cont):

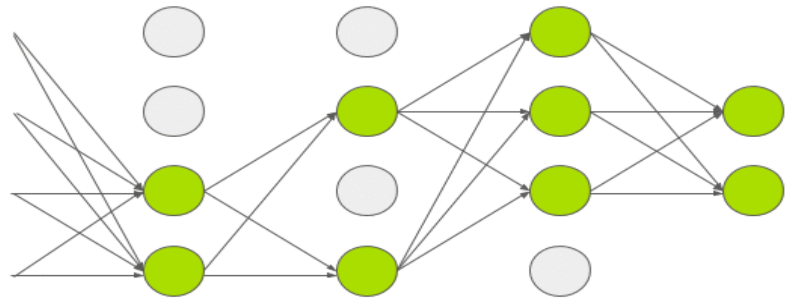
This is an approximation of building a CNN that shows how (conv , relu) and (polling) are done before (the fully connected layer)

As we can see in this photo in the last layer this model use (softmax) as an activation function ,but in our model we using (sigmoid) Why ? Because this is a binary classification









Link

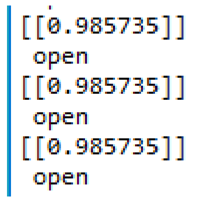
The run & some issues:

This is some outputs from the model .
As we can see when eyes open the result above
(0.8 which I Set it as a threshold or you can make it 0.5)
and when eyes is close the result become
(<0.8 or <0.5)

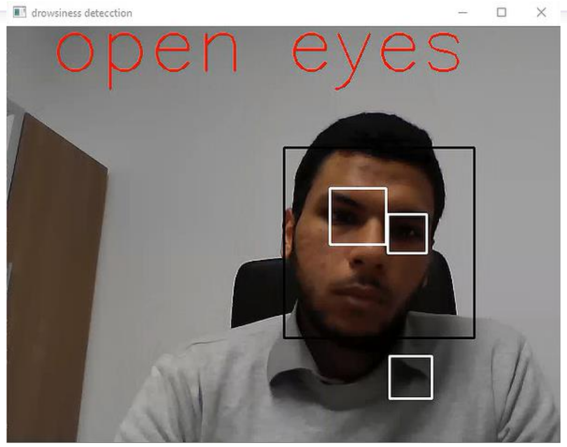
But I face some issues with opencv for detecting the eye , as when the eye close the model for detecting the eye does not work I searched for this problem on Stack OverFlow and I figured this . (link)

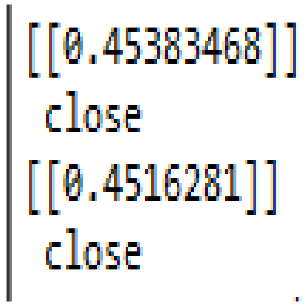
So, I decided to use dlib to see which one was better.

Some other sources : (1) (2) (3) (4)

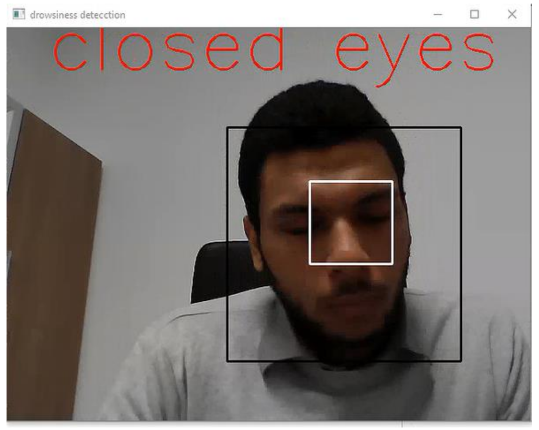


```
[[0.985735]]
 open
[[0.985735]]
 open
[[0.985735]]
 open
```





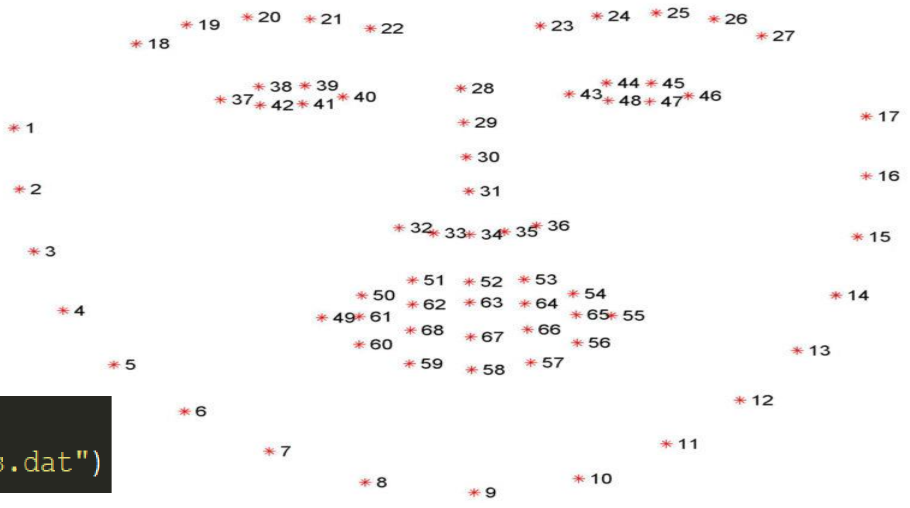
```
[[0.45383468]]
 close
[[0.4516281]]
 close
```



The second model for drowsy with dlib:

- We will try to remedy the defect of the previous model by using dlib library which is a modern C++ toolkit containing machine learning algorithms.
- As we can see this is face land marks.

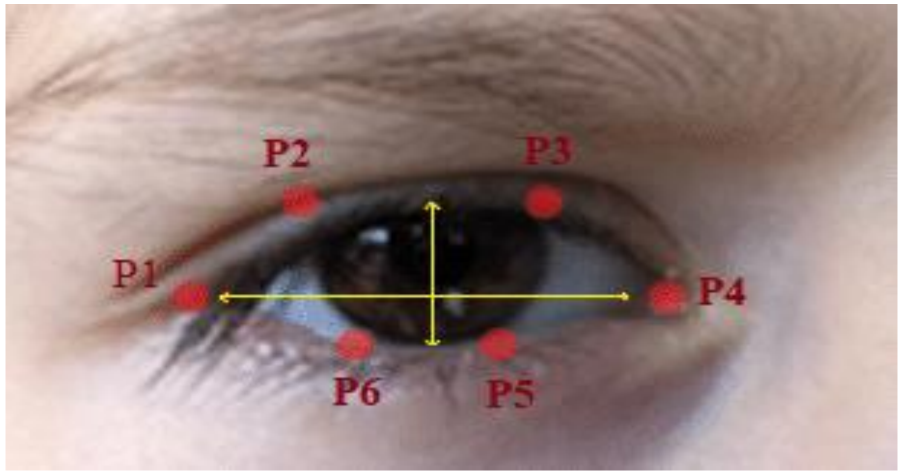
Which can detected with (face landmarks)



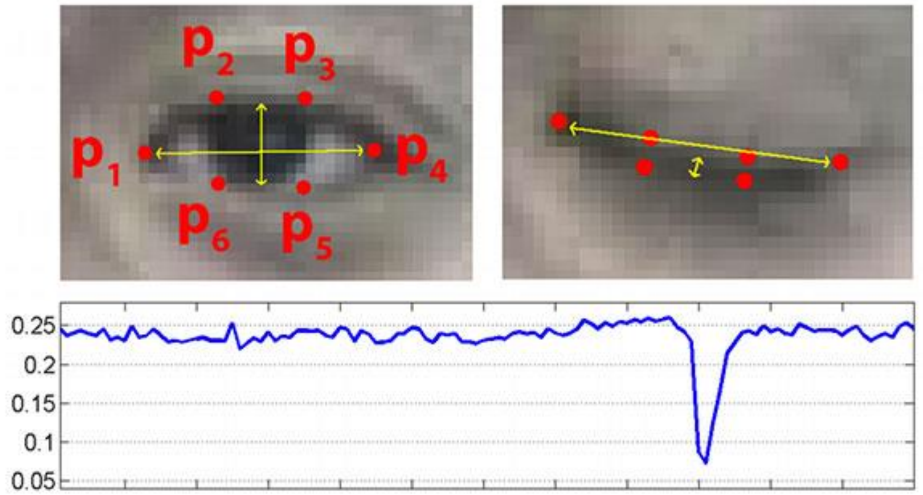
```
hog_face_detector = dlib.get_frontal_face_detector()
dlib_facelandmark = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

Algorithm:

- Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the eye .
- **Condition:** It checks 20 consecutive frames and if the Eye Aspect ratio is less than 0.25,Alert is generated.
- Links (1) (2).



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

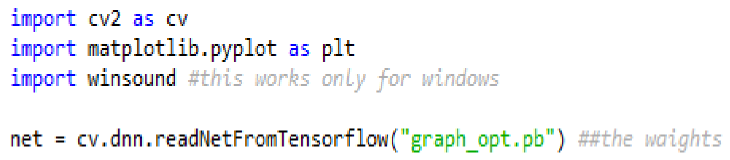


The third model is pose estimation using DNN:

- We also know that looking to the right or left while driving or directly using the phone is wrong while driving .
- So why ? if he needs to talk on the phone directly, it prevents his eyes from looking into the blind spot, causing some accidents.
- If you want to talk to phone you can hang the phone on the dashboard or the car windshield and speak from the speaker.
- I decided to use position estimation to determine whether the driver is looking left or right, and also to see if the driver's hand, whether the right hand or the left hand, is raised from shoulder level or not.

Method:

- To implement this model we will use Tensorflow frozen model Which is in an open source github paper(link) (1) (2).
- There is another good paper for more understanding (link)
- As we can see I use (winsound library for giving a sound when there is a mistake)



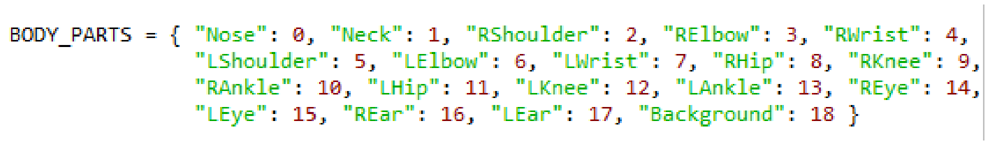
```
import cv2 as cv
import matplotlib.pyplot as plt
import winsound #this works only for windows

net = cv.dnn.readNetFromTensorflow("graph_opt.pb") ##the waights
```

And then we load the model using tensorflow module

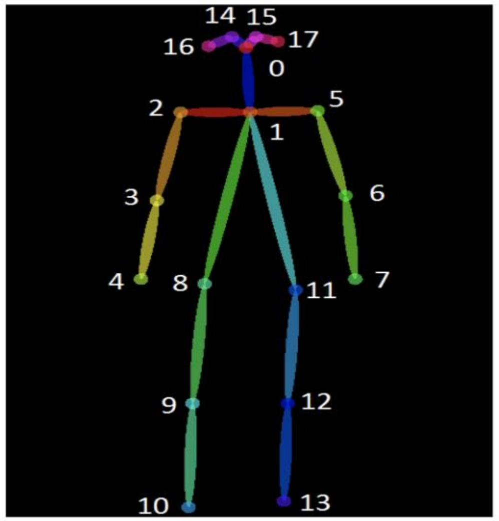
Method (cont):

-As we see there are numbers for each part of the body .
-From this we can know every part that falls on the screen.

-We can notice that if one of the ears (16 or 17) disappears from the screen,
the person is looking right or left
-And also, we located one of the wrists(الساعد),
(4 or 7)and found that it was above the shoulders, this is considered wrong, because that distracts the eyes.

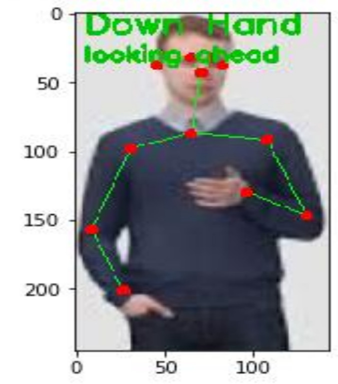


```
BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
               "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
               "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "REye": 14,
               "LEye": 15, "REar": 16, "LEar": 17, "Background": 18 }
```



Some outputs:

```
Out[21]: <matplotlib.image.AxesImage at 0x18f9f1f3940>
```

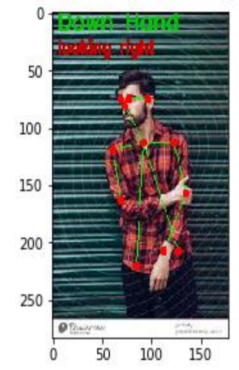


The cases :

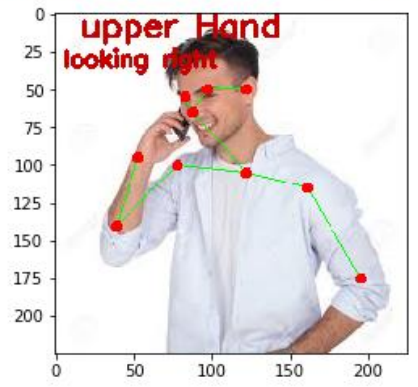
- If the human is looking ahead .
- If he looks left or right .
- If the wrist is down or up

looking right

```
Out[30]: <matplotlib.image.AxesImage at 0x18faaaba4e0>
```



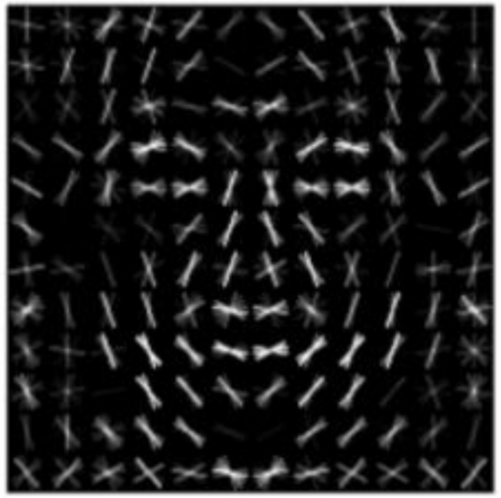
```
Out[28]: <matplotlib.image.AxesImage at 0x23c422c2710>
```



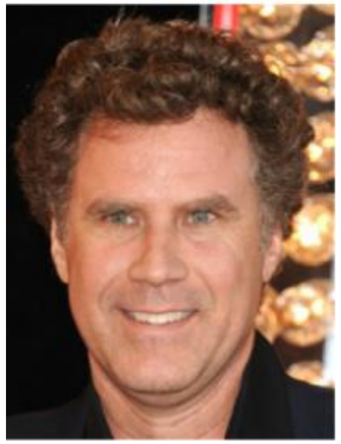
Fourth Model face recognition system:

- There is some steps to perform facial recognition .
- Step 1: locate and extract faces from each image.
- Step 2: identify facial features in each image.
- Step 3: align faces to match pose template.
- Step 4: encode faces using a trained neural network
- Step 5: check Euclidean distance between face encodings (Link) -> use vpn (link) (link)

HOG face pattern generated from lots of face images

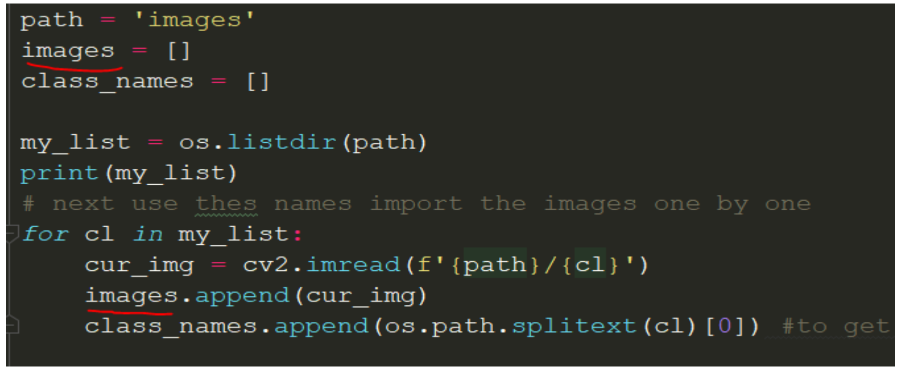


What is Hog ? It is histogram of oriented gradients.

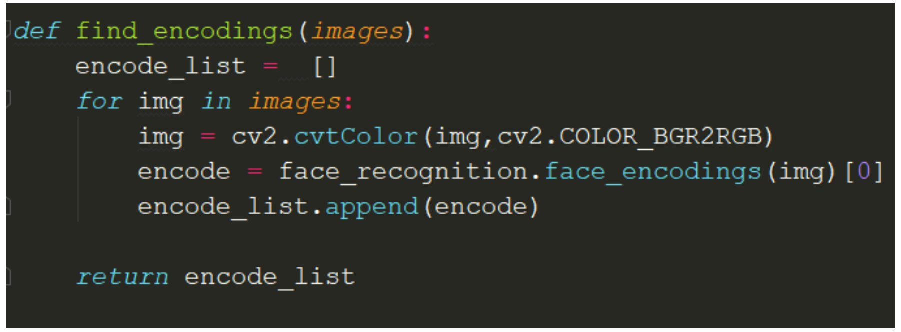


Implementation idea :

- I will put all our data which is persons images in a file (images)
- After that I will encode the faces using function (fing_encodings) which take the images(list) and convert every single image on it into (RGB) ,because of cv read image(BGR) ,so we need to convert it , after that encode the face with (face_recognition.face_encodings)
- Then append this encode into the list which we will return it.
- After that we will call this function to encode all images in (images list).



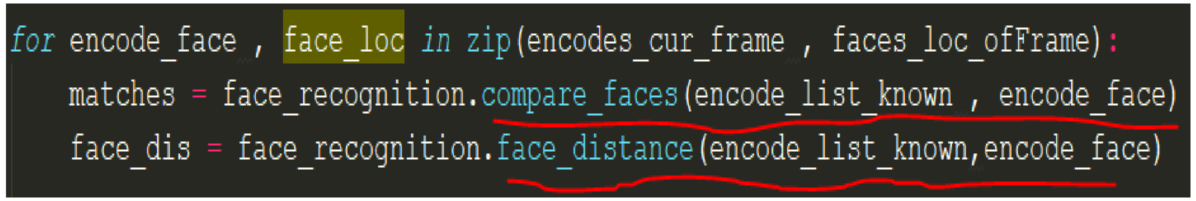
```
path = 'images'
images = []
class_names = []

my_list = os.listdir(path)
print(my_list)
# next use thes names import the images one by one
for cl in my_list:
    cur_img = cv2.imread(f'{path}/{cl}')
    images.append(cur_img)
    class_names.append(os.path.splitext(cl)[0]) #to get
```



```
def find_encodings(images):
    encode_list =  []
    for img in images:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encode_list.append(encode)

    return encode_list
```

Implementation idea (cont):

- After that we will make capture video with (opencv) to detect the faces using webcam .
- Using : (compare_faces fun) which



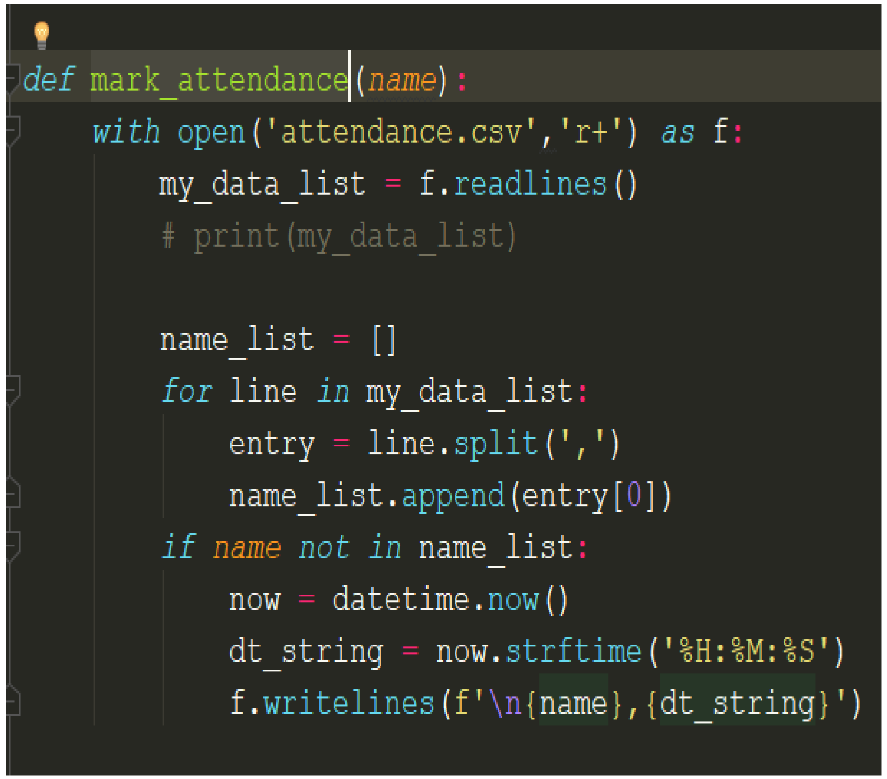
```
for encode_face , face_loc in zip(encodes_cur_frame , faces_loc_ofFrame):
    matches = face_recognition.compare_faces(encode_list_known , encode_face)
    face_dis = face_recognition.face_distance(encode_list_known,encode_face)
```

Take a list of encoding images and compare it with a single encoded image after that return a (Boolean list)contain only true , false values.

While , (face_distance fun) the same parameters ,but return a list with a float values between (0-1) which mean the num. close to 1 is more different between this two images.

Implementation idea (cont):

- There is another fun. Which is less important.
- (make_attendance fun): which takes the name of person and save it in an excel file with time he attended.
- So why ? It is act like a simple data base but we will use (postgresql)



```python
def mark_attendance(name):
    with open('attendance.csv','r+') as f:
        my_data_list = f.readlines()
        # print(my_data_list)

        name_list = []
        for line in my_data_list:
            entry = line.split(',')
            name_list.append(entry[0])
        if name not in name_list:
            now = datetime.now()
            dt_string = now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dt_string}')
```

Time map & the rest of work:

- Time map : I cannot specify the time it took to do this, but I can give an approximate time for each stage.

  -(4–5)weeks: to learn the fundamentals of ML with some

  libraries like (numpy – panda –scikit-learn-matplotlib)

  -(6-7)weeks: to learn the basics of DL with its various libraries

  (tensorflow – keras – opencv-face_recognition-dlib)

  -(2-3)weeks: continue working on these models and learn somefundamentals for Django framework to know how to connect these models to the back end.

- Rest of work: know how to connect these models together and work on tests to increase the performance.

Conclusion:

- Thanks to (Dr.azaa)

  Supervisor for this project.

  Student : kareem gamal Mahmoud Mohamed.

  Level : 4 cs