Computer Vision- Face recognition

OpenCV

Loading the dataset

- Yale faces database: http://vision.ucsd.edu/content/yale-face-database

In [ ]:
```
from PIL import Image
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1. Our data is in zip format
2. We use this code to unzip our data.

In [ ]:
```
import zipfile
path = '/content/drive/MyDrive/Computer Vision Masterclass/Datasets/yalefaces.zip'
zip_object = zipfile.ZipFile(file=path, mode = 'r')
zip_object.extractall('./')
zip_object.close()
```

Pre-processing the images

To list all the names in our folder

In [ ]:
```
import os
print(os.listdir('/content/yalefaces/train'))
```

```
['subject06.normal.gif', 'subject11.centerlight.gif', 'subject12.leftlight.gif', 'subject03.centerlig
ht.gif', 'subject06.sleepy.gif', 'subject03.normal.gif', 'subject10.rightlight.gif', 'subject08.surpr
ised.gif', 'subject05.noglasses.gif', 'subject03.noglasses.gif', 'subject02.surprised.gif', 'subject1
1.sleepy.gif', 'subject13.glasses.gif', 'subject04.centerlight.gif', 'subject07.sad.gif', 'subject12.
glasses.gif', 'subject11.surprised.gif', 'subject06.rightlight.gif', 'subject06.glasses.gif', 'subjec
t03.happy.gif', 'subject03.rightlight.gif', 'subject13.noglasses.gif', 'subject11.wink.gif', 'subject
14.noglasses.gif', 'subject02.sad.gif', 'subject09.happy.gif', 'subject11.rightlight.gif', 'subject1
5.normal.gif', 'subject12.wink.gif', 'subject04.sad.gif', 'subject06.wink.gif', 'subject05.normal.gi
f', 'subject12.happy.gif', 'subject08.noglasses.gif', 'subject10.normal.gif', 'subject05.wink.gif',
'subject08.wink.gif', 'subject04.rightlight.gif', 'subject10.noglasses.gif', 'subject02.sleepy.gif',
'subject14.wink.gif', 'subject07.surprised.gif', 'subject11.normal.gif', 'subject02.rightlight.gif',
'subject08.sad.gif', 'subject14.happy.gif', 'subject01.normal.gif', 'subject12.centerlight.gif', 'sub
ject05.centerlight.gif', 'subject06.noglasses.gif', 'subject10.sleepy.gif', 'subject02.noglasses.gi
f', 'subject12.sad.gif', 'subject15.glasses.gif', 'subject09.normal.gif', 'subject10.happy.gif', 'sub
ject06.sad.gif', 'subject07.wink.gif', 'subject09.glasses.gif', 'subject02.glasses.gif', 'subject07.n
oglasses.gif', 'subject02.normal.gif', 'subject13.happy.gif', 'subject10.glasses.gif', 'subject15.sur
prised.gif', 'subject13.normal.gif', 'subject01.noglasses.gif', 'subject11.leftlight.gif', 'subject0
6.surprised.gif', 'subject13.surprised.gif', 'subject03.wink.gif', 'subject13.centerlight.gif', 'subj
ect09.noglasses.gif', 'subject14.rightlight.gif', 'subject07.sleepy.gif', 'subject15.centerlight.gi
f', 'subject01.sleepy.gif', 'subject04.glasses.gif', 'subject14.surprised.gif', 'subject04.noglasses.
gif', 'subject01.sad.gif', 'subject07.centerlight.gif', 'subject05.sad.gif', 'subject05.glasses.gif',
'subject15.leftlight.gif', 'subject04.happy.gif', 'subject14.leftlight.gif', 'subject02.happy.gif',
'subject08.leftlight.gif', 'subject08.glasses.gif', 'subject09.leftlight.gif', 'subject09.wink.gif',
'subject05.happy.gif', 'subject13.wink.gif', 'subject03.sad.gif', 'subject08.happy.gif', 'subject11.n
oglasses.gif', 'subject03.sleepy.gif', 'subject12.surprised.gif', 'subject01.surprised.gif', 'subject
13.rightlight.gif', 'subject02.wink.gif', 'subject01.rightlight.gif', 'subject12.noglasses.gif', 'sub
ject05.leftlight.gif', 'subject14.centerlight.gif', 'subject13.leftlight.gif', 'subject14.glasses.gi
f', 'subject07.glasses.gif', 'subject15.happy.gif', 'subject12.sleepy.gif', 'subject15.sleepy.gif',
'subject10.wink.gif', 'subject15.noglasses.gif', 'subject04.sleepy.gif', 'subject10.leftlight.gif',
'subject07.rightlight.gif', 'subject08.centerlight.gif', 'subject10.surprised.gif', 'subject06.center
light.gif', 'subject01.wink.gif', 'subject05.rightlight.gif', 'subject08.sleepy.gif', 'subject01.left
light.gif', 'subject09.surprised.gif', 'subject09.centerlight.gif', 'subject04.wink.gif', 'subject15.
wink.gif', 'subject04.normal.gif', 'subject03.surprised.gif', 'subject14.sleepy.gif', 'subject11.sad.
gif', 'subject09.sleepy.gif', 'subject01.glasses.gif', 'subject07.normal.gif']
```

In [ ]:
```python
def get_image_data():
  paths = [os.path.join('/content/yalefaces/train', f) for f in os.listdir('/content/yalefaces/train')
  #print(paths)
  faces = []
  ids = []
  for path in paths:
    #print(path)
    image = Image.open(path).convert('L') # covert the image to grayscale image using PIL (PILLOW)
    #print(type(image))
    image_np = np.array(image, 'uint8') # convert to numpy array
    #print(type(image_np))
    id = int(os.path.split(path)[1].split('.')[0].replace('subject', '')) # Do selection just to the r
    #print(id)
    ids.append(id) #is a liste we need to convert to numpy array
    faces.append(image_np)

  return np.array(ids), faces
```

In [ ]:
```python
ids, faces = get_image_data() #use the function get_image_data()
```

In [ ]:
```python
ids #print ids
```

Out[ ]:
```
array([14,  5, 14, 14,  5,  7,  9,  5,  8,  8, 13,  6, 10,  1,  3, 11, 11,
        6, 12,  3,  3,  3, 13,  5,  1,  9, 13,  4, 11,  6, 14,  3, 14, 13,
       13, 12,  4,  8,  8, 15, 12,  2,  5, 11,  3,  7, 14,  1,  8, 12,  2,
        1,  6,  4, 10, 10, 14, 11, 10, 13, 11, 13, 12,  9,  2,  7, 11, 15,
       15,  2, 10,  9, 10, 14,  7,  2,  8,  4,  1, 12, 15, 11, 10, 12,  9,
       14, 12,  1, 12, 13,  2,  8, 15,  4, 10,  2,  7,  3,  9,  1,  1,  9,
       13,  7, 15, 11,  1, 15,  4,  3,  6,  7,  5,  7,  5,  7,  4, 15,  9,
        6,  5,  2,  6,  4,  5,  3,  2,  8,  9,  6, 10,  6,  4,  8, 15])
```

In [ ]:
```python
len(ids)
```

Out[ ]:
```
135
```

In [ ]:
```python
faces
```

In [ ]:
```python
len(faces)
```

Out[ ]:
```
135
```

In [ ]:
```
faces[0], faces[0].shape
```

Out[ ]:
```
(array([[130, 130, 130, ..., 255, 255, 255],
        [255, 255, 255, ..., 255, 255, 255],
        [255, 255, 255, ..., 255, 243, 245],
        ...,
        [255, 255, 255, ..., 255, 255, 255],
        [255, 255, 255, ..., 255, 255, 255],
        [ 68,  68,  68, ...,  68,  68,  68]], dtype=uint8),
 (243, 320))
```

In [ ]:
```
243 * 320 #nomber of pixel in image
```

Out[ ]:
```
77760
```

Training the LBPH classifier

In [ ]:
```
8 * 8
```

Out[ ]:
```
64
```

In [ ]:
```
# threshold: 1.7976931348623157e+308
# radius: 1
# neighbors: 8
# grid_x: 8
# grid_y: 8

lbph_classifier = cv2.face.LBPHFaceRecognizer_create(radius =3 , neighbors =12, grid_x =8, grid_y= 8)
lbph_classifier.train(faces, ids) # give teh classifier inputs faces and ids
lbph_classifier.write('lbph_classifier.yml') # save the classifier in  .yml
```

Recognizing faces

In [ ]:
```
lbph_face_classifier = cv2.face.LBPHFaceRecognizer_create()
lbph_face_classifier.read('/content/lbph_classifier.yml') #load the face classifier
```

In [ ]:
```
test_image = '/content/yalefaces/test/subject10.sad.gif'
```

pre-process the image

In [ ]:
```
image = Image.open(test_image).convert('L') # convert the image to grayscale
image_np = np.array(image, 'uint8') # convert to the numpay array
image_np # see the image
```

Out[ ]:
```
ndarray (243, 320) show data
```

In [ ]:
```
image_np.shape # the image shape
```

Out[ ]:
```
(243, 320)
```

In [ ]:
```
prediction = lbph_face_classifier.predict(image_np) #predicte the image classe
prediction
```

Out[ ]:
```
(10, 6.384336446373091)
```

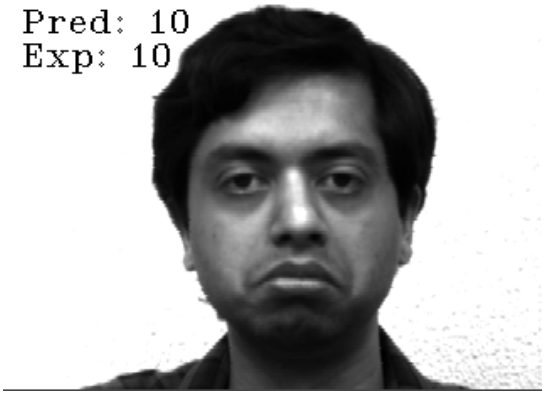
In [ ]:
```
prediction[0]
```

Out[ ]:
```
10
```

In [ ]:
```
expected_output = int(os.path.split(test_image)[1].split('.')[0].replace('subject', '')) # extracte th
expected_output
```

Out[ ]:
```
10
```

In [ ]:
```
cv2.putText(image_np, 'Pred: ' + str(prediction[0]), (10, 30), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,2
cv2.putText(image_np, 'Exp: ' + str(expected_output), (10, 50), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,
cv2_imshow(image_np)
```



Evaluating the face classifier

In [ ]:
```
paths = [os.path.join('/content/yalefaces/test', f) for f in os.listdir('/content/yalefaces/test')]
predictions = []
expected_outputs = []
for path in paths:
  #print(path)
  image = Image.open(path).convert('L') # load the image and covert to grayscale
  image_np = np.array(image, 'uint8') # convert the image the numpy array
  prediction, _ = lbph_face_classifier.predict(image_np) # do the prediction
  expected_output = int(os.path.split(path)[1].split('.')[0].replace('subject', ''))  # extracte the e

  predictions.append(prediction) # append the resulte of the production in predictions
  expected_outputs.append(expected_output) # append the expected output in expected_outputs
```

In [ ]:
```
type(predictions) # type of predictions
```

Out[ ]:
```
list
```

In [ ]:
```
predictions = np.array(predictions) # covert predictions to a numpy array
expected_outputs = np.array(expected_outputs) # convert expected_outputs to a numpy array
```

In [ ]:
```
type(predictions)
```

Out[ ]:
```
numpy.ndarray
```

In [ ]:
```
predictions
```

Out[ ]:
```
array([ 7,  6, 11, 11,  9,  4, 13,  9, 10,  9,  4,  5, 13, 14, 15, 12, 14,
        5,  5, 14,  7,  1,  1,  4,  3,  9,  7,  8, 12,  4])
```

In [ ]:
```
expected_outputs
```

Out[ ]:
```
array([ 9,  6, 11, 11,  9,  8, 13,  3, 10,  4,  2,  5, 13, 14, 15, 12, 14,
        2,  5,  6, 15,  1,  1, 10,  3,  7,  7,  8, 12,  4])
```

use the sklearn to calculate accutacy

In [ ]:
```
from sklearn.metrics import accuracy_score
accuracy_score(expected_outputs, predictions)
```

Out[ ]:
```
0.6333333333333333
```

In [ ]:
```
len(predictions)
```

Out[ ]:
```
30
```

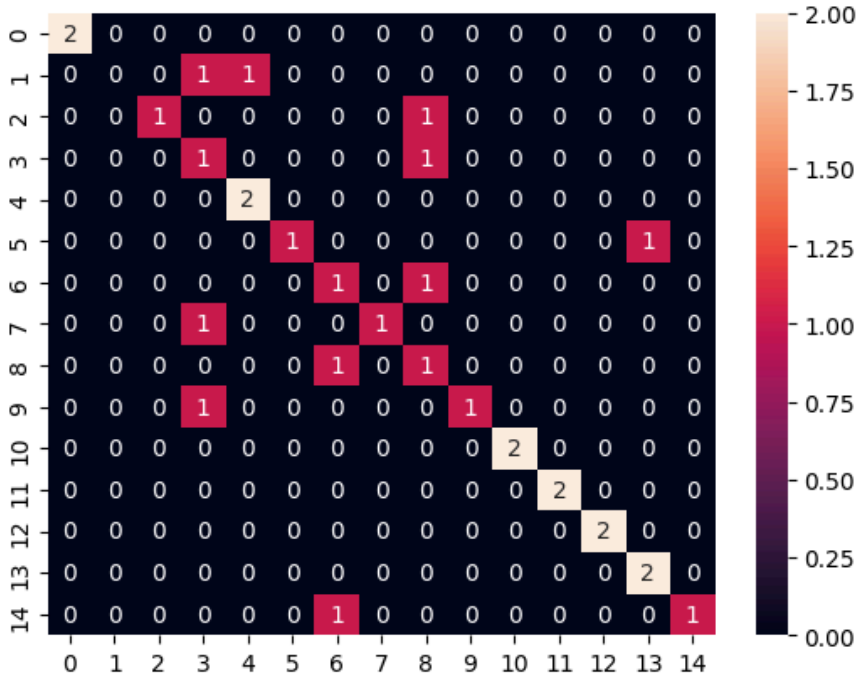
use sklearn to have confusion_matrix

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(expected_outputs, predictions)
cm
```

Out[ ]:
```
array([[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]])
```

the confusion matrix :it's interestion to see how the algorithm is porforming in each one of the classes

In [ ]:
```
import seaborn # a library in order to work with graphs in python
seaborn.heatmap(cm, annot=True);
```



Dlib

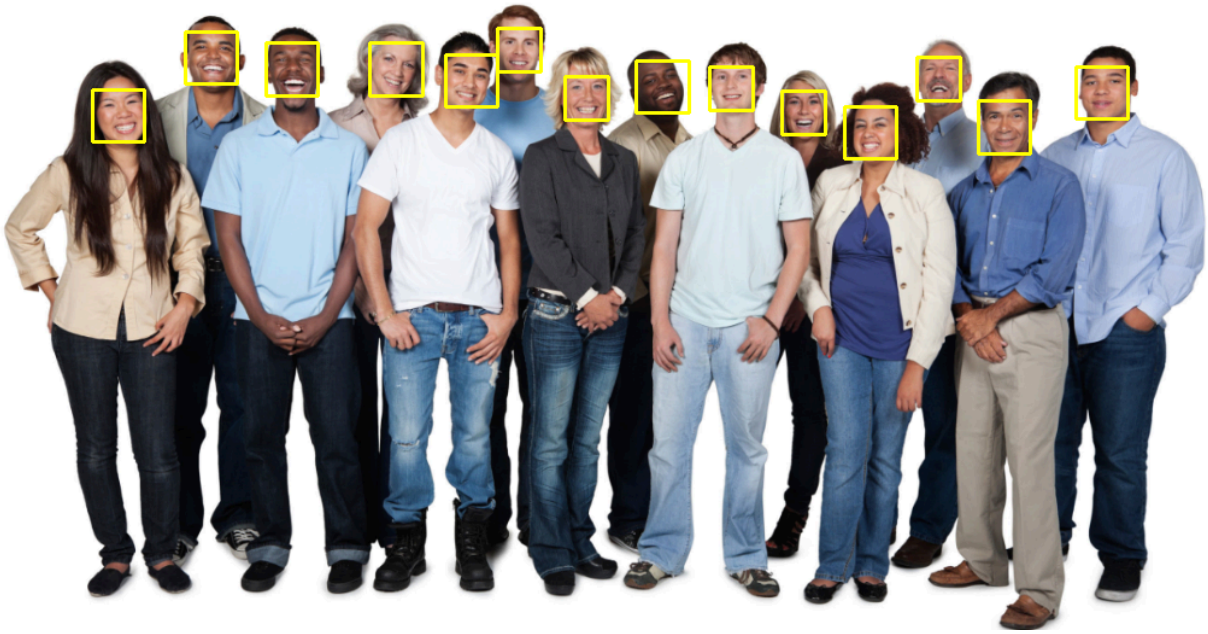
In [ ]:
```
import dlib
import cv2
from google.colab.patches import cv2_imshow
```

Detecting facial points

In [ ]:
```
face_detector = dlib.get_frontal_face_detector()
points_detector = dlib.shape_predictor('/content/drive/MyDrive/Computer Vision Masterclass/Weights/sha
```

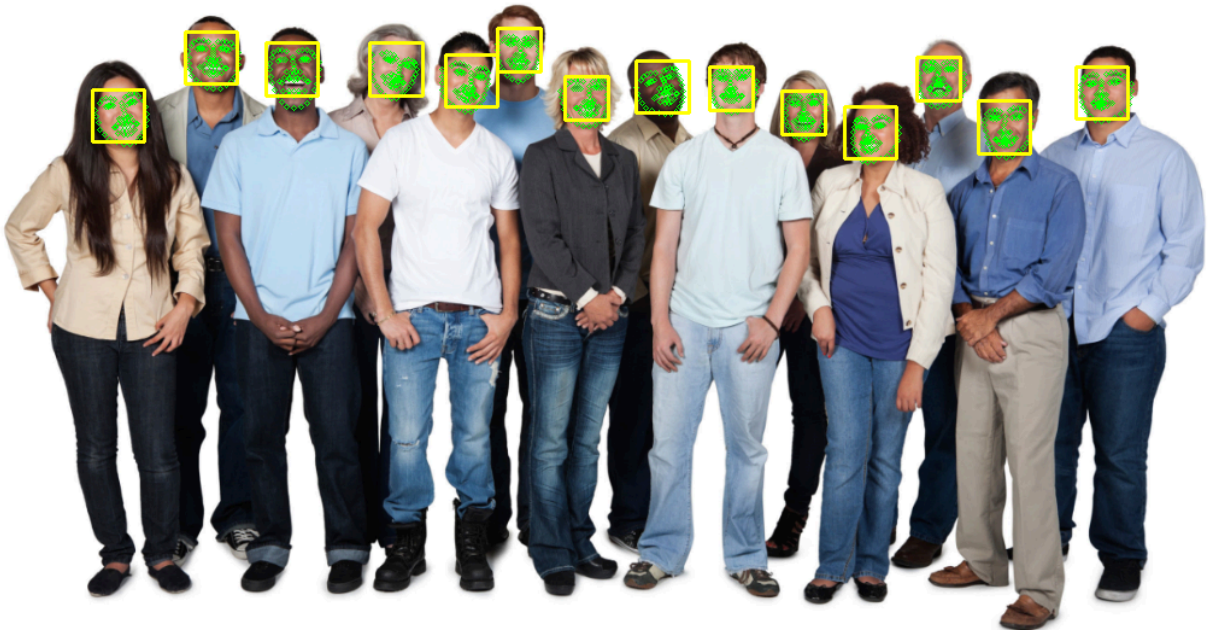
detecte faces

In [ ]:
```
image = cv2.imread('/content/drive/MyDrive/Computer Vision Masterclass/Images/people2.jpg')
face_detection = face_detector(image, 1) # 1 scale the image
for face in face_detection:
  l, t, r, b = face.left(), face.top(), face.right(), face.bottom()
  cv2.rectangle(image, (l, t), (r, b), (0,255,255), 2)
cv2_imshow(image)
```



Detect the 68 points on the face

In [ ]:
```
image = cv2.imread('/content/drive/MyDrive/Computer Vision Masterclass/Images/people2.jpg')
face_detection = face_detector(image, 1) # 1 scale the image
for face in face_detection:
  points = points_detector(image, face)
  for point in points.parts():
    cv2.circle(image, (point.x, point.y), 2, (0,255,0), 1)

  #print(points.parts())
  #print(len(points.parts()))

  l, t, r, b = face.left(), face.top(), face.right(), face.bottom()
  cv2.rectangle(image, (l, t), (r, b), (0,255,255), 2)
cv2_imshow(image)
```



Detecting facial descriptors

In [ ]:
```
import os
```

resnet :it means an architecture of a convolutional neural network in order to work with image calassification and also face recognize

In [ ]:
```
# Resnet: https://arxiv.org/abs/1512.03385
face_detector = dlib.get_frontal_face_detector()
points_detector = dlib.shape_predictor('/content/drive/MyDrive/Computer Vision Masterclass/Weights/sha
face_descriptor_extractor = dlib.face_recognition_model_v1('/content/drive/MyDrive/Computer Vision Mas
```

Detect the 68 points on the face to predicte shape in our training data

In [ ]:
```
index = {} # frrmat of dictionary
idx = 0
face_descriptors = None

paths = [os.path.join('/content/yalefaces/train', f) for f in os.listdir('/content/yalefaces/train')]
for path in paths:
  #print(path)
  image = Image.open(path).convert('RGB')
  image_np = np.array(image, 'uint8')
  face_detection = face_detector(image_np, 1)
  for face in face_detection:
    #detected faces
    l, t, r, b = face.left(), face.top(), face.right(), face.bottom()
    cv2.rectangle(image_np, (l, t), (r, b), (0, 0, 255), 2)
    #detected points
    points = points_detector(image_np, face)
    for point in points.parts():
      cv2.circle(image_np, (point.x, point.y), 2, (0, 255, 0), 1)
    cv2_imshow(image_np)
```

Detect the 68 points on the face

In [ ]:
```
index = {} # frrmat of dictionary
idx = 0
face_descriptors = None

paths = [os.path.join('/content/yalefaces/train', f) for f in os.listdir('/content/yalefaces/train')]
for path in paths:
  #print(path)
  image = Image.open(path).convert('RGB')
  image_np = np.array(image, 'uint8')
  face_detection = face_detector(image_np, 1)
  for face in face_detection:
    l, t, r, b = face.left(), face.top(), face.right(), face.bottom()
    cv2.rectangle(image_np, (l, t), (r, b), (0, 0, 255), 2)

    points = points_detector(image_np, face)
    for point in points.parts():
      cv2.circle(image_np, (point.x, point.y), 2, (0, 255, 0), 1)

    face_descriptor = face_descriptor_extractor.compute_face_descriptor(image_np, points) #the CNN inp
    #print(type(face_descriptor))  # the face descriptor extractor is a delib vector
    #print(len(face_descriptor)) # the len of the output of the CNN
    #print(face_descriptor) # the value of ache face descripter (infromation about faces)
    face_descriptor = [f for f in face_descriptor] # convert frome delib vector to a vector
    #print(face_descriptor)
    face_descriptor = np.asarray(face_descriptor, dtype=np.float64) # convert to numpy
    #print(face_descriptor)
    #print(face_descriptor.shape)
    face_descriptor = face_descriptor[np.newaxis, :] # convert a vector to matrix [1,128]
    #print(face_descriptor.shape)
    #print(face_descriptor)

    if face_descriptors is None:
      face_descriptors = face_descriptor
    else:
      face_descriptors = np.concatenate((face_descriptors, face_descriptor), axis = 0)
```

```
    index[idx] = path
    idx += 1
  #cv2_imshow(image_np)
```

In [ ]:
```
face_descriptors.shape
```

Out[ ]:
```
(132, 128)
```

In [ ]:
```
face_descriptors
```

Out[ ]:
```
array([[-0.14677812,  0.07504971,  0.02076296, ..., -0.01445033,
         0.12270239,  0.01553179],
       [-0.01982844,  0.06148595, -0.05128621, ..., -0.01377742,
         0.03838211,  0.03345009],
       [-0.15156889,  0.03644923,  0.02427856, ...,  0.00066456,
         0.10565172,  0.0100017 ],
       ...,
       [-0.10740127,  0.06253605,  0.06953968, ..., -0.02038986,
         0.11134732, -0.01935074],
       [-0.11481769,  0.08830089,  0.04191965, ...,  0.00937126,
         0.14437041,  0.07062955],
       [-0.16237783,  0.06176241,  0.01140093, ..., -0.01002233,
         0.08126575,  0.07473017]])
```

In [ ]:
```
len(index)
```

Out[ ]:
```
132
```

In [ ]:
```
index
```

Out[ ]:
```
{0: '/content/yalefaces/train/subject14.noglasses.gif',
 1: '/content/yalefaces/train/subject05.glasses.gif',
 2: '/content/yalefaces/train/subject14.happy.gif',
 3: '/content/yalefaces/train/subject14.rightlight.gif',
 4: '/content/yalefaces/train/subject05.leftlight.gif',
 5: '/content/yalefaces/train/subject07.surprised.gif',
 6: '/content/yalefaces/train/subject09.surprised.gif',
 7: '/content/yalefaces/train/subject05.centerlight.gif',
 8: '/content/yalefaces/train/subject08.leftlight.gif',
 9: '/content/yalefaces/train/subject08.wink.gif',
 10: '/content/yalefaces/train/subject13.normal.gif',
 11: '/content/yalefaces/train/subject06.rightlight.gif',
 12: '/content/yalefaces/train/subject10.normal.gif',
 13: '/content/yalefaces/train/subject01.noglasses.gif',
 14: '/content/yalefaces/train/subject03.normal.gif',
 15: '/content/yalefaces/train/subject11.sleepy.gif',
 16: '/content/yalefaces/train/subject11.wink.gif',
 17: '/content/yalefaces/train/subject06.normal.gif',
 18: '/content/yalefaces/train/subject12.centerlight.gif',
 19: '/content/yalefaces/train/subject03.wink.gif',
 20: '/content/yalefaces/train/subject03.surprised.gif',
 21: '/content/yalefaces/train/subject03.rightlight.gif',
 22: '/content/yalefaces/train/subject13.rightlight.gif',
 23: '/content/yalefaces/train/subject05.noglasses.gif',
 24: '/content/yalefaces/train/subject01.surprised.gif',
 25: '/content/yalefaces/train/subject09.leftlight.gif',
 26: '/content/yalefaces/train/subject13.glasses.gif',
 27: '/content/yalefaces/train/subject04.happy.gif',
 28: '/content/yalefaces/train/subject11.sad.gif',
 29: '/content/yalefaces/train/subject06.wink.gif',
 30: '/content/yalefaces/train/subject14.glasses.gif',
 31: '/content/yalefaces/train/subject03.centerlight.gif',
 32: '/content/yalefaces/train/subject14.surprised.gif',
 33: '/content/yalefaces/train/subject13.happy.gif',
 34: '/content/yalefaces/train/subject13.noglasses.gif',
 35: '/content/yalefaces/train/subject12.happy.gif',
 36: '/content/yalefaces/train/subject04.centerlight.gif',
 37: '/content/yalefaces/train/subject08.surprised.gif',
 38: '/content/yalefaces/train/subject08.happy.gif',
 39: '/content/yalefaces/train/subject15.happy.gif',
 40: '/content/yalefaces/train/subject12.surprised.gif',
 41: '/content/yalefaces/train/subject02.noglasses.gif',
 42: '/content/yalefaces/train/subject05.sad.gif',
 43: '/content/yalefaces/train/subject11.centerlight.gif',
 44: '/content/yalefaces/train/subject03.noglasses.gif',
 45: '/content/yalefaces/train/subject07.sad.gif',
 46: '/content/yalefaces/train/subject14.wink.gif',
 47: '/content/yalefaces/train/subject08.noglasses.gif',
 48: '/content/yalefaces/train/subject12.sad.gif',
 49: '/content/yalefaces/train/subject02.happy.gif',
 50: '/content/yalefaces/train/subject01.leftlight.gif',
 51: '/content/yalefaces/train/subject06.glasses.gif',
 52: '/content/yalefaces/train/subject04.rightlight.gif',
 53: '/content/yalefaces/train/subject10.surprised.gif',
 54: '/content/yalefaces/train/subject10.wink.gif',
 55: '/content/yalefaces/train/subject14.sleepy.gif',
 56: '/content/yalefaces/train/subject11.leftlight.gif',
 57: '/content/yalefaces/train/subject10.glasses.gif',
 58: '/content/yalefaces/train/subject13.wink.gif',
 59: '/content/yalefaces/train/subject13.surprised.gif',
 60: '/content/yalefaces/train/subject12.wink.gif',
 61: '/content/yalefaces/train/subject09.centerlight.gif',
 62: '/content/yalefaces/train/subject02.normal.gif',
 63: '/content/yalefaces/train/subject07.centerlight.gif',
 64: '/content/yalefaces/train/subject11.noglasses.gif',
 65: '/content/yalefaces/train/subject15.wink.gif',
 66: '/content/yalefaces/train/subject15.glasses.gif',
 67: '/content/yalefaces/train/subject02.glasses.gif',
 68: '/content/yalefaces/train/subject10.leftlight.gif',
 69: '/content/yalefaces/train/subject09.normal.gif',
 70: '/content/yalefaces/train/subject10.happy.gif',
 71: '/content/yalefaces/train/subject14.centerlight.gif',
 72: '/content/yalefaces/train/subject07.rightlight.gif',
 73: '/content/yalefaces/train/subject02.sleepy.gif',
 74: '/content/yalefaces/train/subject08.sleepy.gif',
 75: '/content/yalefaces/train/subject04.normal.gif',
```

```
 76: '/content/yalefaces/train/subject01.sad.gif',
 77: '/content/yalefaces/train/subject12.noglasses.gif',
 78: '/content/yalefaces/train/subject15.noglasses.gif',
 79: '/content/yalefaces/train/subject11.normal.gif',
 80: '/content/yalefaces/train/subject10.noglasses.gif',
 81: '/content/yalefaces/train/subject12.glasses.gif',
 82: '/content/yalefaces/train/subject09.wink.gif',
 83: '/content/yalefaces/train/subject14.leftlight.gif',
 84: '/content/yalefaces/train/subject12.leftlight.gif',
 85: '/content/yalefaces/train/subject01.glasses.gif',
 86: '/content/yalefaces/train/subject12.sleepy.gif',
 87: '/content/yalefaces/train/subject13.centerlight.gif',
 88: '/content/yalefaces/train/subject02.surprised.gif',
 89: '/content/yalefaces/train/subject08.glasses.gif',
 90: '/content/yalefaces/train/subject15.sleepy.gif',
 91: '/content/yalefaces/train/subject04.wink.gif',
 92: '/content/yalefaces/train/subject10.rightlight.gif',
 93: '/content/yalefaces/train/subject02.sad.gif',
 94: '/content/yalefaces/train/subject07.wink.gif',
 95: '/content/yalefaces/train/subject03.sleepy.gif',
 96: '/content/yalefaces/train/subject09.noglasses.gif',
 97: '/content/yalefaces/train/subject01.sleepy.gif',
 98: '/content/yalefaces/train/subject01.wink.gif',
 99: '/content/yalefaces/train/subject09.happy.gif',
100: '/content/yalefaces/train/subject13.leftlight.gif',
101: '/content/yalefaces/train/subject07.sleepy.gif',
102: '/content/yalefaces/train/subject15.surprised.gif',
103: '/content/yalefaces/train/subject11.surprised.gif',
104: '/content/yalefaces/train/subject01.normal.gif',
105: '/content/yalefaces/train/subject15.centerlight.gif',
106: '/content/yalefaces/train/subject04.noglasses.gif',
107: '/content/yalefaces/train/subject03.sad.gif',
108: '/content/yalefaces/train/subject06.surprised.gif',
109: '/content/yalefaces/train/subject07.glasses.gif',
110: '/content/yalefaces/train/subject07.noglasses.gif',
111: '/content/yalefaces/train/subject05.happy.gif',
112: '/content/yalefaces/train/subject07.normal.gif',
113: '/content/yalefaces/train/subject04.sleepy.gif',
114: '/content/yalefaces/train/subject15.leftlight.gif',
115: '/content/yalefaces/train/subject09.sleepy.gif',
116: '/content/yalefaces/train/subject06.sleepy.gif',
117: '/content/yalefaces/train/subject05.wink.gif',
118: '/content/yalefaces/train/subject02.rightlight.gif',
119: '/content/yalefaces/train/subject06.sad.gif',
120: '/content/yalefaces/train/subject04.sad.gif',
121: '/content/yalefaces/train/subject05.normal.gif',
122: '/content/yalefaces/train/subject03.happy.gif',
123: '/content/yalefaces/train/subject02.wink.gif',
124: '/content/yalefaces/train/subject08.sad.gif',
125: '/content/yalefaces/train/subject09.glasses.gif',
126: '/content/yalefaces/train/subject06.noglasses.gif',
127: '/content/yalefaces/train/subject10.sleepy.gif',
128: '/content/yalefaces/train/subject06.centerlight.gif',
129: '/content/yalefaces/train/subject04.glasses.gif',
130: '/content/yalefaces/train/subject08.centerlight.gif',
131: '/content/yalefaces/train/subject15.normal.gif'}
```

Calculating the distance between faces

In [ ]:
```
face_descriptors[131]
```

Out[ ]:
```
array([-0.16237783,  0.06176241,  0.01140093, -0.0432231 , -0.11218053,
        0.00119162, -0.03840479, -0.03233848,  0.11025289, -0.10359   ,
        0.14507101, -0.09910502, -0.2426108 , -0.02112983, -0.0268052 ,
        0.12784988, -0.12906173, -0.14507979, -0.11684252, -0.0882516 ,
        0.03342528,  0.01613965, -0.06863495,  0.05895548, -0.15508649,
       -0.29823059, -0.04945969, -0.10732467, -0.01147079, -0.11853084,
       -0.05007018,  0.04275068, -0.17954612, -0.09388153,  0.01433266,
        0.01096971,  0.02190497,  0.00592806,  0.17834187,  0.01202874,
       -0.16591868,  0.01946391,  0.12345148,  0.30224845,  0.15886973,
        0.13874418, -0.03960529, -0.0009208 ,  0.13613831, -0.29982734,
        0.10869649,  0.10674702,  0.02577681,  0.03023069,  0.10506182,
       -0.137419  , -0.03116703,  0.10201724, -0.17223378,  0.12476938,
        0.08538186, -0.03987823, -0.07481945, -0.08062833,  0.20180722,
        0.11836953, -0.09401987, -0.1113123 ,  0.17372882, -0.15655202,
       -0.05795558,  0.01554254, -0.06369782, -0.11933833, -0.29278922,
        0.0777179 ,  0.43370736,  0.16105327, -0.12379459,  0.01545268,
       -0.00660456, -0.06566939,  0.09722649,  0.01590143, -0.09149671,
       -0.03910387, -0.03890089,  0.05709735,  0.20708144,  0.08050783,
       -0.01336345,  0.19445851,  0.01642721, -0.04621297,  0.03844336,
        0.07108656, -0.13698944, -0.07743665, -0.15849647, -0.01519291,
        0.07813419, -0.06193293,  0.00936343,  0.15918523, -0.15426534,
        0.11208795, -0.04403237, -0.02483392, -0.05481757,  0.09453647,
       -0.12102751, -0.01341188,  0.0899919 , -0.17941429,  0.10717747,
        0.12459922,  0.05219061,  0.07166664,  0.08065946,  0.0068429 ,
        0.04788181,  0.04141653, -0.19275913, -0.16178647,  0.08907652,
       -0.01002233,  0.08126575,  0.07473017])
```

In [ ]:
```
# https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html#numpy.linalg.norm
np.linalg.norm(face_descriptors[131] - face_descriptors[131]) # calculate the distance between the sam
```

Out[ ]:
```
0.0
```

In [ ]:
```
np.linalg.norm(face_descriptors[131] - face_descriptors[130])
```

Out[ ]:
```
0.6341091365862859
```

In [ ]:
```
np.linalg.norm(face_descriptors[131] - face_descriptors[129])
```

Out[ ]:
```
0.7928864086073453
```

In [ ]:
```
np.linalg.norm(face_descriptors[131] - face_descriptors[105])
```

Out[ ]:
```
0.3695495151139848
```

In [ ]:
```
np.linalg.norm(face_descriptors[0] - face_descriptors, axis = 1) # Compare the face descriptor 0 with
```

Out[ ]:
```
array([0.        , 0.77643283, 0.21440541, 0.45481372, 0.80615682,
       0.77384493, 0.71850292, 0.79160933, 0.58790961, 0.53366132,
       0.70217471, 0.52115798, 0.74650959, 0.80551549, 0.64788999,
       0.64372402, 0.63707472, 0.46351493, 0.85089299, 0.68867067,
       0.68359063, 0.62819526, 0.63099083, 0.78262134, 0.8370851 ,
       0.74434851, 0.68018162, 0.42829375, 0.59774649, 0.51978241,
       0.27277952, 0.67923645, 0.18567814, 0.63975971, 0.72332326,
       0.82737141, 0.45845793, 0.60074197, 0.56432678, 0.63126626,
       0.85083721, 0.74124343, 0.77614517, 0.63309365, 0.64788999,
       0.71502252, 0.20617219, 0.55553763, 0.87032023, 0.76791532,
       0.78872844, 0.60560253, 0.496633  , 0.77509799, 0.77103148,
       0.20047163, 0.61222711, 0.74017204, 0.67409884, 0.63918182,
       0.8625751 , 0.73324235, 0.74124343, 0.76145282, 0.59328017,
       0.71430097, 0.69448671, 0.71548524, 0.74723199, 0.71269223,
       0.69891857, 0.4555046 , 0.75689097, 0.71721315, 0.51255916,
       0.38012254, 0.78985849, 0.85386396, 0.66041376, 0.57912238,
       0.74873565, 0.8561172 , 0.68302567, 0.51197218, 0.83709368,
       0.83382393, 0.82018174, 0.67414679, 0.79905429, 0.5481878 ,
       0.62639445, 0.4454934 , 0.79048986, 0.76448832, 0.70878215,
       0.68144667, 0.71269223, 0.78470115, 0.84312814, 0.68853218,
       0.7094274 , 0.70829242, 0.67737309, 0.60839703, 0.76256922,
       0.64825956, 0.38828703, 0.70504348, 0.45588601, 0.66551919,
       0.72746338, 0.77714885, 0.72746338, 0.40289212, 0.6262027 ,
       0.66224314, 0.42960924, 0.75718905, 0.76373573, 0.47675343,
       0.38012254, 0.78060236, 0.73199657, 0.73776384, 0.5413686 ,
       0.73029643, 0.46351493, 0.70503514, 0.48036969, 0.47115152,
       0.52795738, 0.65170279])
```

In [ ]:
```
np.argmin(np.linalg.norm(face_descriptors[0] - face_descriptors[1:], axis = 1))
```

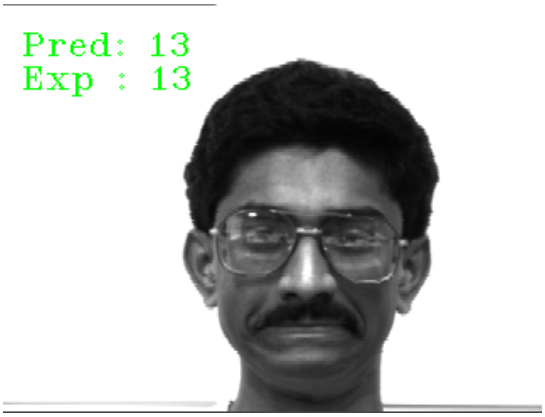
Out[ ]:
```
31
```

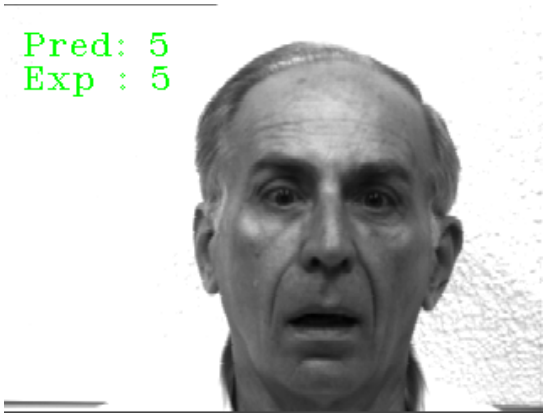
In [ ]:
```
np.linalg.norm(face_descriptors[0] - face_descriptors[1:], axis = 1)[91]
```

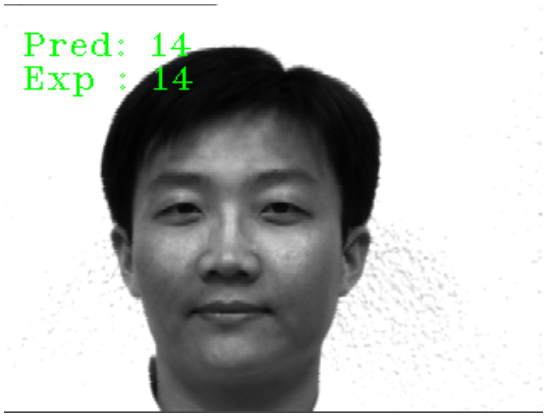
Out[ ]:
```
0.7904898580961541
```

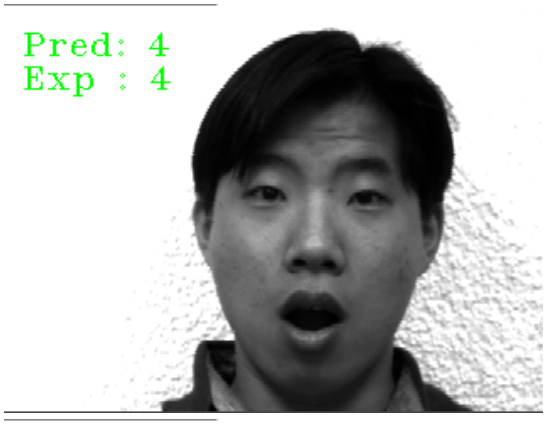
Detecting faces with Dlib

In [ ]:
```
threshold = 0.5 # The minimum distance accepted for the prediction.
# we add that to calculate the accuracy
predictions = []
expected_outputs = []

paths = [os.path.join('/content/yalefaces/test', f) for f in os.listdir('/content/yalefaces/test')]
for path in paths:
  image = Image.open(path).convert('RGB')
  image_np = np.array(image, 'uint8')
  face_detection = face_detector(image_np, 1)
  for face in face_detection:
    points = points_detector(image_np, face)
    face_descriptor = face_descriptor_extractor.compute_face_descriptor(image_np, points)
    face_descriptor = [f for f in face_descriptor]
    face_descriptor = np.asarray(face_descriptor, dtype=np.float64)
    face_descriptor = face_descriptor[np.newaxis, :]

    distances = np.linalg.norm(face_descriptor - face_descriptors, axis = 1)
    min_index = np.argmin(distances)
    min_distance = distances[min_index]
    if min_distance <= threshold:
      name_pred = int(os.path.split(index[min_index])[1].split('.')[0].replace('subject', ''))
    else:
      name_pred = 'Not identified'

    name_real = int(os.path.split(path)[1].split('.')[0].replace('subject', ''))

    predictions.append(name_pred)
    expected_outputs.append(name_real)

    cv2.putText(image_np, 'Pred: ' + str(name_pred), (10, 30), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,2
    cv2.putText(image_np, 'Exp : ' + str(name_real), (10, 50), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,2


  cv2_imshow(image_np)

predictions = np.array(predictions)
expected_outputs = np.array(expected_outputs)
```



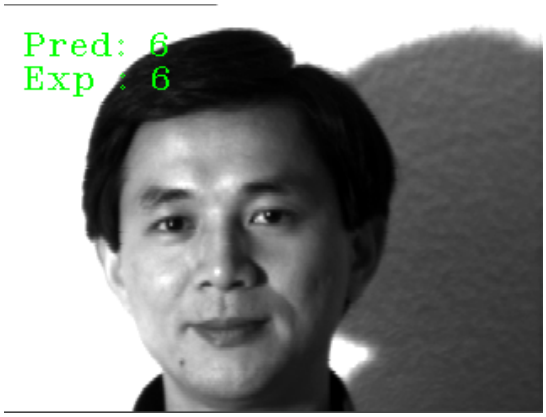

Pred: 5
Exp : 5


Pred: 14
Exp : 14

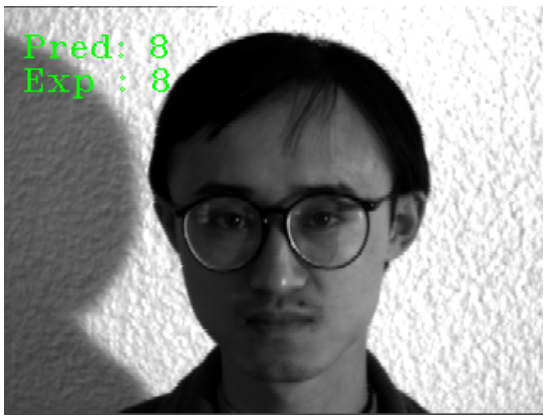

Pred: 4
Exp : 4


Pred: 7
Exp : 7

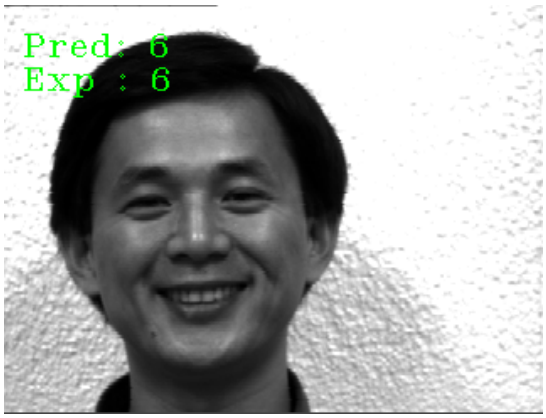

Pred: 11
Exp : 11

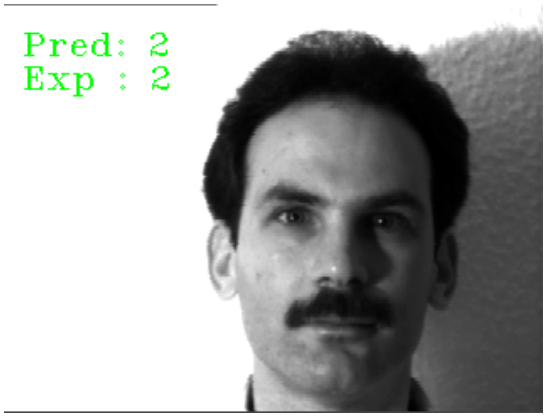

Pred: 6
Exp : 6

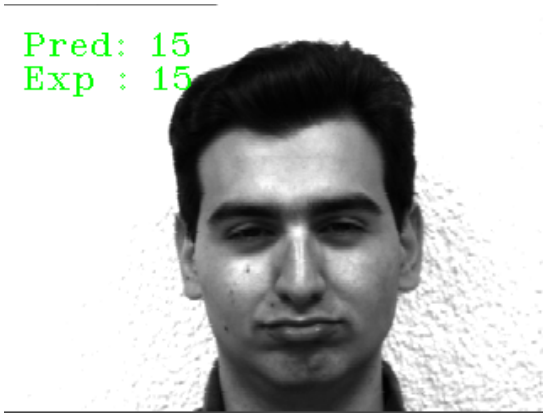

Pred: 10
Exp : 10


Pred: 8
Exp : 8

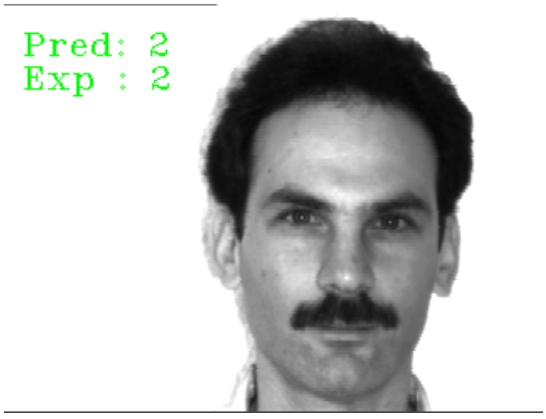

Pred: 3
Exp : 3


Pred: 6
Exp : 6


Pred: 2
Exp : 2

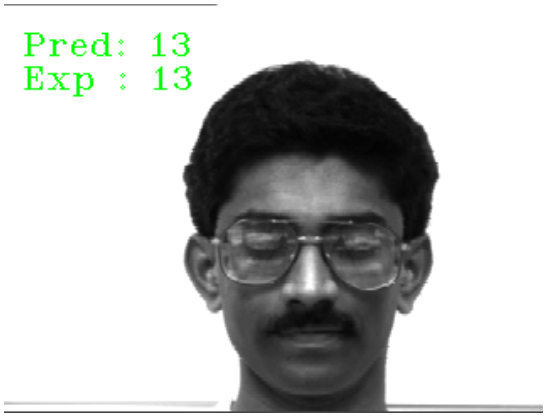

Pred: 15
Exp : 15

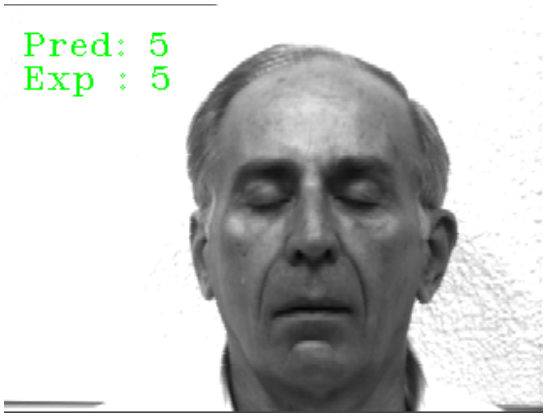

Pred: 9
Exp : 9


Pred: 2
Exp : 2

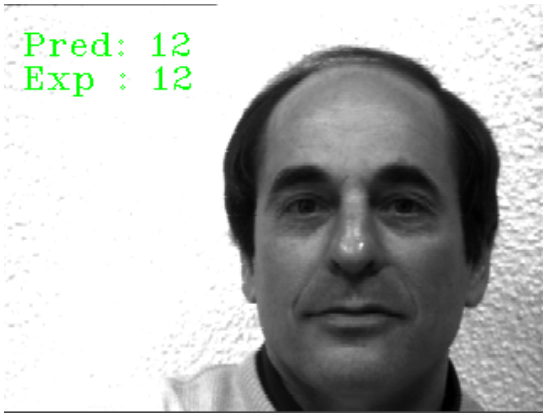

Pred: 3
Exp : 3

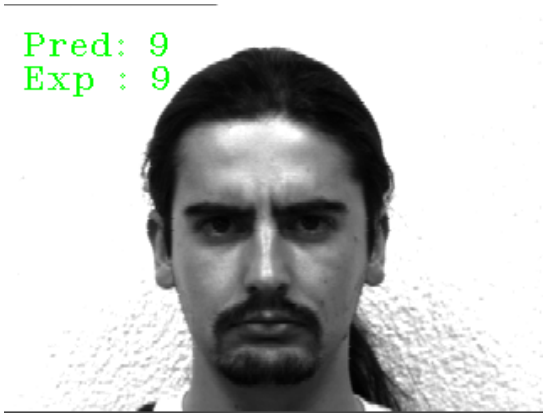

Pred: 7
Exp : 7

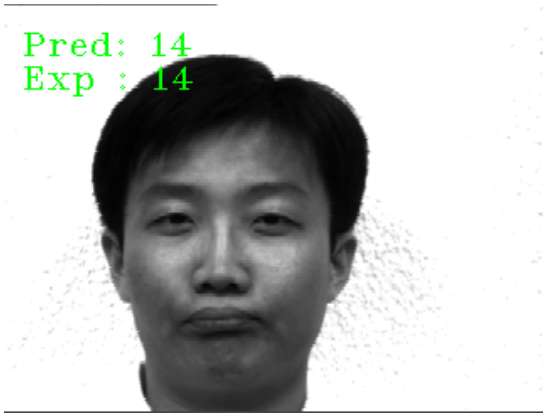

Pred: 13
Exp : 13


Pred: 5
Exp : 5


Pred: 1
Exp : 1


Pred: 12
Exp : 12

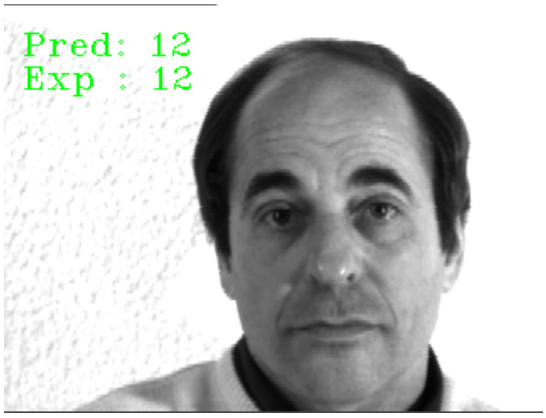

Pred: 9
Exp : 9

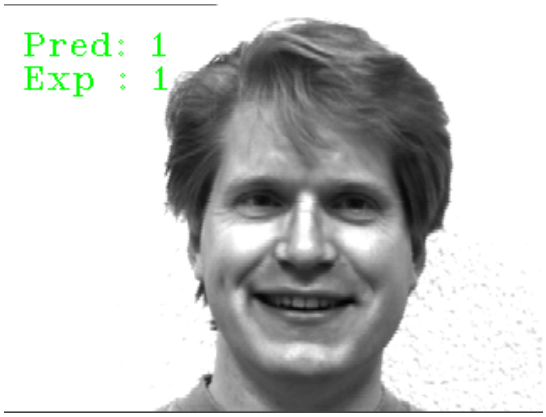

Pred: 14
Exp : 14

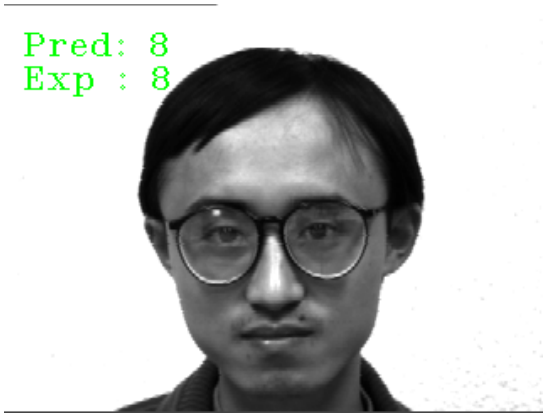

Pred: 10
Exp : 10


Pred: 15
Exp : 15


Pred: 11
Exp : 11







In [ ]:
```
predictions
```

Out[ ]:
```
array([13,  5, 14,  4,  7, 11,  6, 10,  8,  3,  6,  2, 15,  9,  2,  3,  7,
       13,  5,  1, 12,  9, 14, 10, 15, 11,  4, 12,  1,  8])
```

In [ ]:
```
expected_outputs
```

Out[ ]:
```
array([13,  5, 14,  4,  7, 11,  6, 10,  8,  3,  6,  2, 15,  9,  2,  3,  7,
       13,  5,  1, 12,  9, 14, 10, 15, 11,  4, 12,  1,  8])
```

In [ ]:
```
from sklearn.metrics import accuracy_score
accuracy_score(expected_outputs, predictions) # calculate the accuracy
```

Out[ ]:
```
1.0
```