Face Verification

Amirhesam Bolandi
Reyhaneh Kharazmi

Computer Vision
Dr. Ali Nazari
January 24, 2023

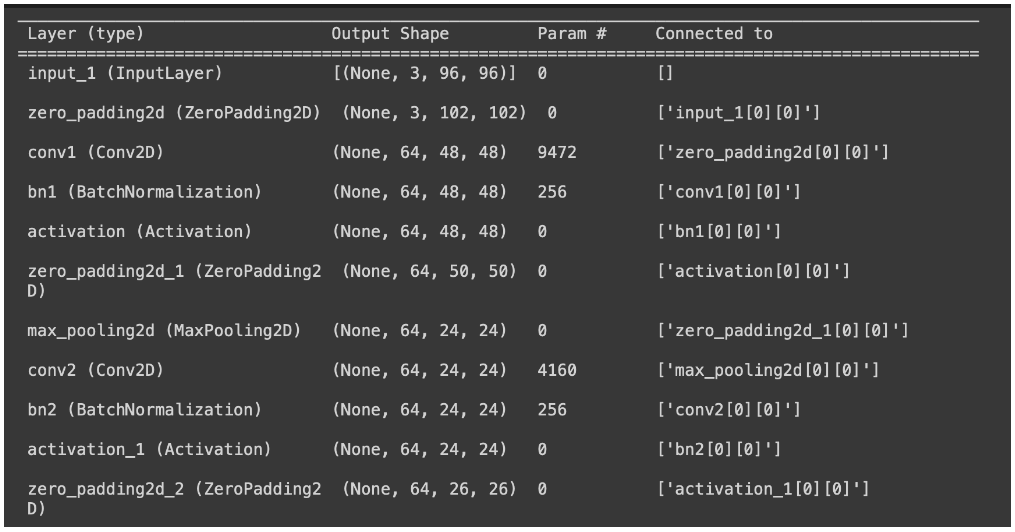
Solution

We use siamese networks in this project as they are the best option for face verification. In fact we take advantage of them in order to transfer images from visual space into vector space. The key point here is that we need an embedding model which is responsible for this transformation. The target is to embedded data in a way that similar pictures belonging to the same class (same person in this case) locate close to each other in the embedding space and dissimilar pictures belonging to different classes (different people in this case) locate far away from each other. To achieve this goal, we train a model including convolutional layers, maxpooling, and padding , and etc. .

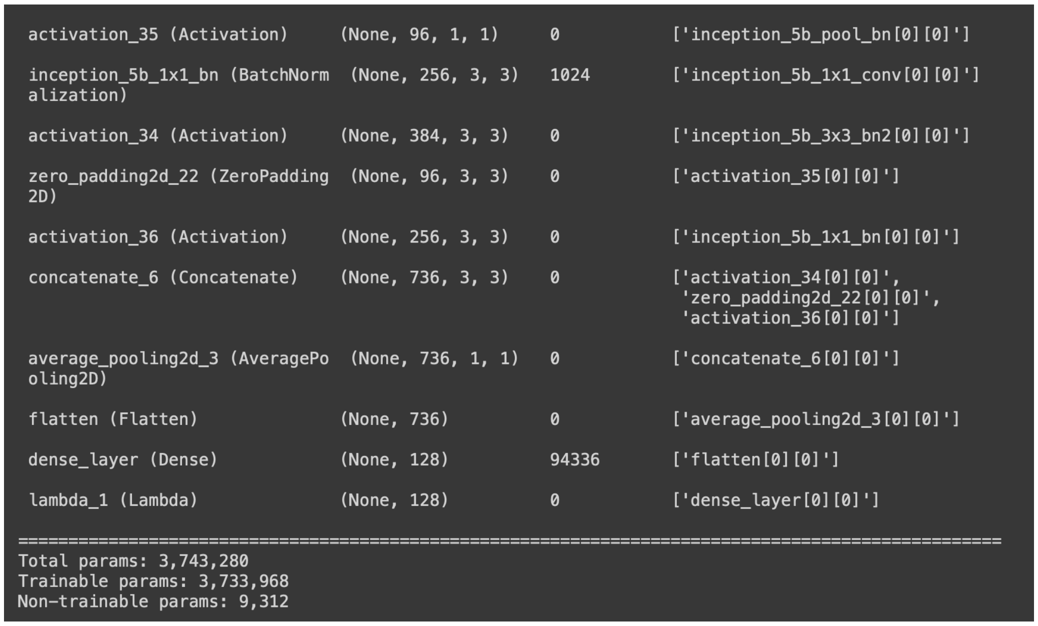
We use triplet loss function which is responsible for calculating the distance between similar and dissimilar samples of people and minimizing the distance between similar ones and maximizing the distance between dissimilar ones.

Model

We used Keras model from tensorflow library with 5 layers including zero padding, normalizer 2D convolution, max pooling. We trained it by using batches and in 5 epochs. Then it maps the visual data into 128-dimensional vector.



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 3, 96, 96)]  0           []

 zero_padding2d (ZeroPadding2D)  (None, 3, 102, 102)  0          ['input_1[0][0]']

 conv1 (Conv2D)                 (None, 64, 48, 48)   9472        ['zero_padding2d[0][0]']

 bn1 (BatchNormalization)       (None, 64, 48, 48)   256         ['conv1[0][0]']

 activation (Activation)        (None, 64, 48, 48)   0           ['bn1[0][0]']

 zero_padding2d_1 (ZeroPadding2  (None, 64, 50, 50)  0           ['activation[0][0]']
 D)

 max_pooling2d (MaxPooling2D)   (None, 64, 24, 24)   0           ['zero_padding2d_1[0][0]']

 conv2 (Conv2D)                 (None, 64, 24, 24)   4160        ['max_pooling2d[0][0]']

 bn2 (BatchNormalization)       (None, 64, 24, 24)   256         ['conv2[0][0]']

 activation_1 (Activation)      (None, 64, 24, 24)   0           ['bn2[0][0]']

 zero_padding2d_2 (ZeroPadding2  (None, 64, 26, 26)  0           ['activation_1[0][0]']
 D)
```



```
 activation_35 (Activation)     (None, 96, 1, 1)     0           ['inception_5b_pool_bn[0][0]']

 inception_5b_1x1_bn (BatchNorm  (None, 256, 3, 3)   1024        ['inception_5b_1x1_conv[0][0]']
 alization)

 activation_34 (Activation)     (None, 384, 3, 3)    0           ['inception_5b_3x3_bn2[0][0]']

 zero_padding2d_22 (ZeroPadding  (None, 96, 3, 3)    0           ['activation_35[0][0]']
 2D)

 activation_36 (Activation)     (None, 256, 3, 3)    0           ['inception_5b_1x1_bn[0][0]']

 concatenate_6 (Concatenate)    (None, 736, 3, 3)    0           ['activation_34[0][0]',
                                                                  'zero_padding2d_22[0][0]',
                                                                  'activation_36[0][0]']

 average_pooling2d_3 (AveragePo  (None, 736, 1, 1)   0           ['concatenate_6[0][0]']
 oling2D)

 flatten (Flatten)              (None, 736)          0           ['average_pooling2d_3[0][0]']

 dense_layer (Dense)            (None, 128)          94336       ['flatten[0][0]']

 lambda_1 (Lambda)              (None, 128)          0           ['dense_layer[0][0]']

==================================================================================================
Total params: 3,743,280
Trainable params: 3,733,968
Non-trainable params: 9,312
```

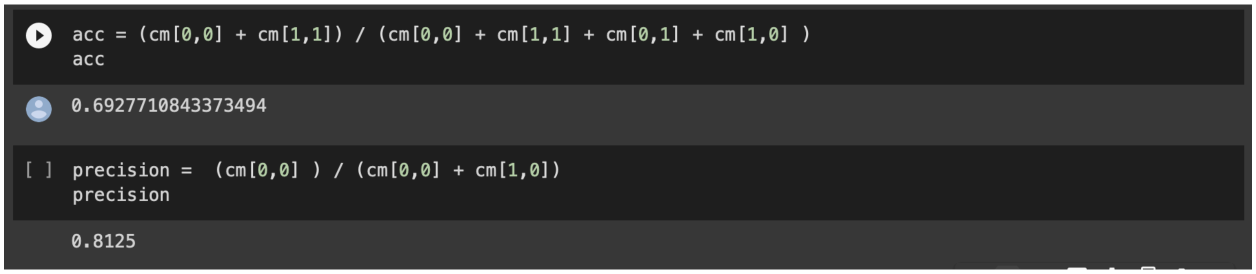
Dataset

We used the LFW dataset and chose 26 folders(people) randomly and used one of each folder's photos as a test image, so the model can not train it, and the else used for training.

In data generation function for training the model we chose 2 folders, one for the negative direction and one for positive direction and the base line photo.

Results

To evaluate the performance of our work, we made a confusion matrix and evaluated the approach based on accuracy and precision metrics (both are shown in the image below). To determine the confusion matrix, we made 2 arrays, one of them includes the images of test set and other one includes image of the train set. The critical point about our test set is that all image in that have not ever been fed to the model during the train phase, so we just use them in the test phase. To make the confusion matrix, we compare the corresponding data in both arrays and predict whether they belong to a same person or not. We collect this predictions and compare them to the reality. Accordingly, we can intelligently gain the accuracy and other metrics. The results of evaluation shows that our approach performs significantly well.



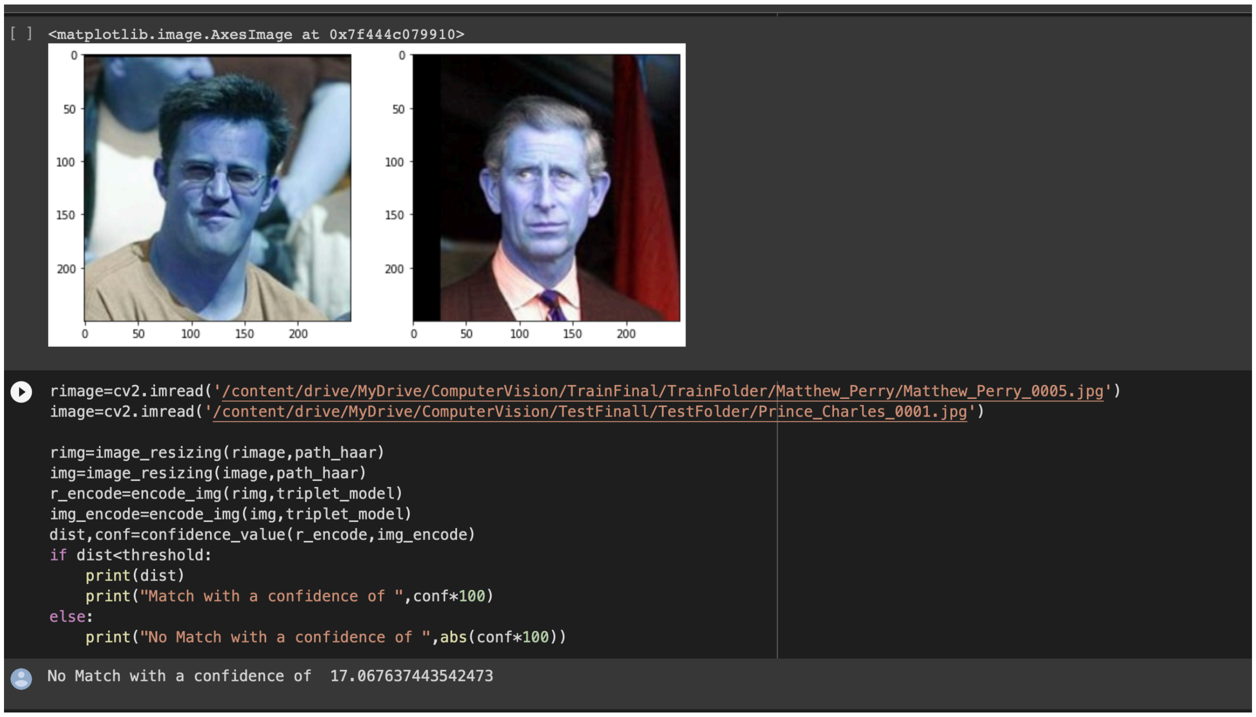
```
acc = (cm[0,0] + cm[1,1]) / (cm[0,0] + cm[1,1] + cm[0,1] + cm[1,0] )
acc
```

```
0.6927710843373494
```

```
precision =  (cm[0,0] ) / (cm[0,0] + cm[1,0])
precision
```

```
0.8125
```

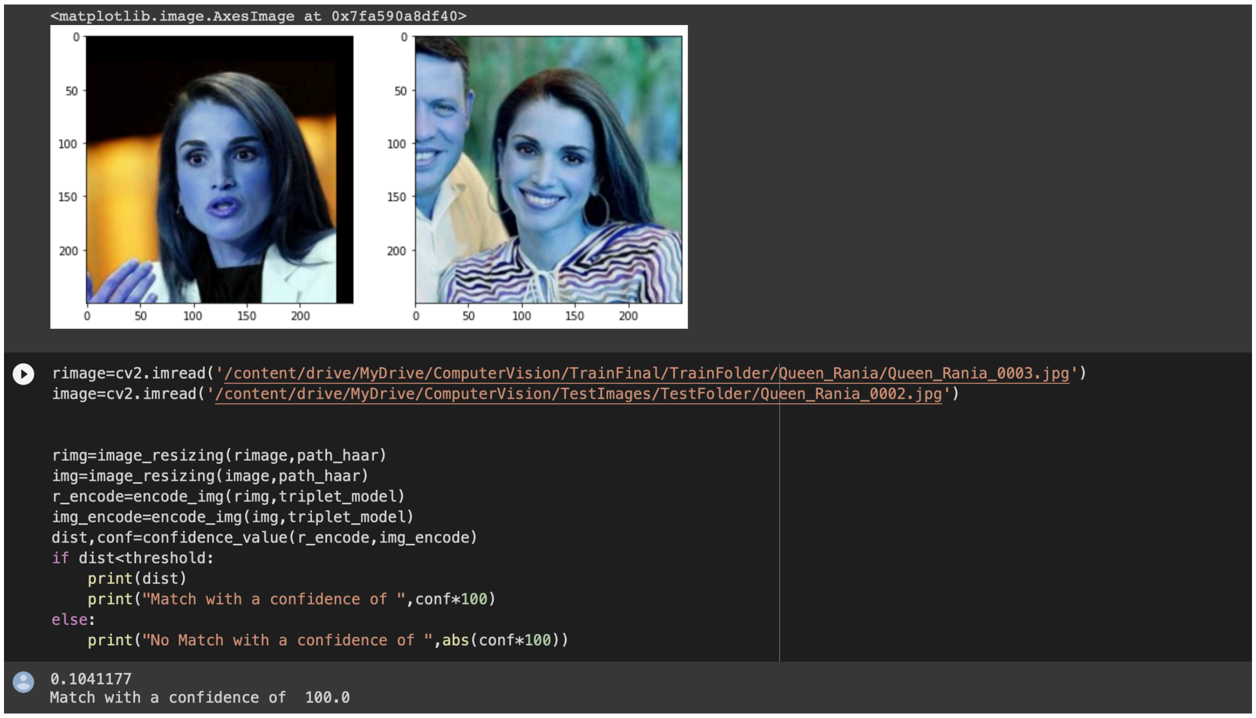
Some results:



```
<matplotlib.image.AxesImage at 0x7f444c079910>
```

```
rimage=cv2.imread('/content/drive/MyDrive/ComputerVision/TrainFinal/TrainFolder/Matthew_Perry/Matthew_Perry_0005.jpg')
image=cv2.imread('/content/drive/MyDrive/ComputerVision/TestFinall/TestFolder/Prince_Charles_0001.jpg')

rimg=image_resizing(rimage,path_haar)
img=image_resizing(image,path_haar)
r_encode=encode_img(rimg,triplet_model)
img_encode=encode_img(img,triplet_model)
dist,conf=confidence_value(r_encode,img_encode)
if dist<threshold:
    print(dist)
    print("Match with a confidence of ",conf*100)
else:
    print("No Match with a confidence of ",abs(conf*100))
```

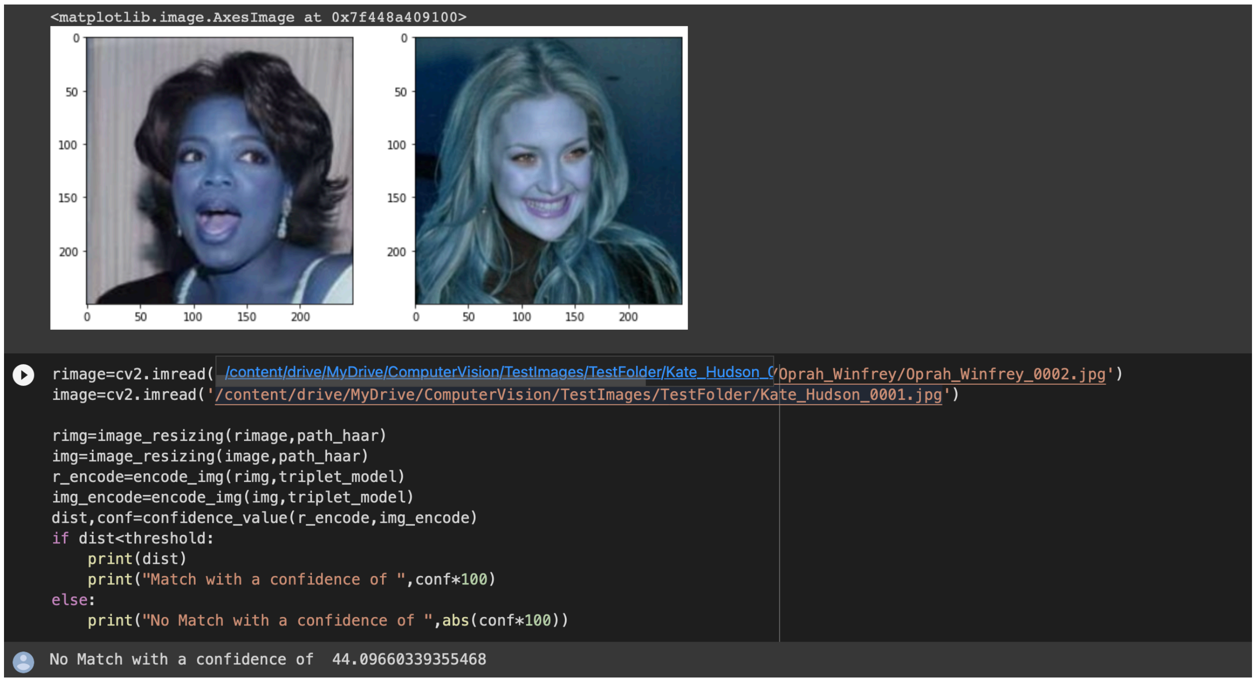
```
No Match with a confidence of  17.067637443542473
```



```
<matplotlib.image.AxesImage at 0x7fa590a8df40>
```

```
rimage=cv2.imread('/content/drive/MyDrive/ComputerVision/TrainFinal/TrainFolder/Queen_Rania/Queen_Rania_0003.jpg')
image=cv2.imread('/content/drive/MyDrive/ComputerVision/TestImages/TestFolder/Queen_Rania_0002.jpg')


rimg=image_resizing(rimage,path_haar)
img=image_resizing(image,path_haar)
r_encode=encode_img(rimg,triplet_model)
img_encode=encode_img(img,triplet_model)
dist,conf=confidence_value(r_encode,img_encode)
if dist<threshold:
    print(dist)
    print("Match with a confidence of ",conf*100)
else:
    print("No Match with a confidence of ",abs(conf*100))
```

```
0.1041177
Match with a confidence of  100.0
```



```
<matplotlib.image.AxesImage at 0x7f448a409100>
```

```
rimage=cv2.imread( /content/drive/MyDrive/ComputerVision/TestImages/TestFolder/Kate_Hudson_0Oprah_Winfrey/Oprah_Winfrey_0002.jpg')
image=cv2.imread('/content/drive/MyDrive/ComputerVision/TestImages/TestFolder/Kate_Hudson_0001.jpg')

rimg=image_resizing(rimage,path_haar)
img=image_resizing(image,path_haar)
r_encode=encode_img(rimg,triplet_model)
img_encode=encode_img(img,triplet_model)
dist,conf=confidence_value(r_encode,img_encode)
if dist<threshold:
    print(dist)
    print("Match with a confidence of ",conf*100)
else:
    print("No Match with a confidence of ",abs(conf*100))
```

```
No Match with a confidence of  44.09660339355468
```

Colab link: https://colab.research.google.com/drive/1Vj91AXIHimzPLpuDBwLcubHTmKv28rAQ?usp=sharing