Face Recognition

Installations Instructions:

- Create an empty Python 3.7 anaconda virtual environment using the command: `$ conda create --name face_rec2 python=3.7.6`
- Install all the required Python libraries using the command: `$ pip install -r requirements.txt`
- Run the Pytho script or notebook.

Comments:

- The code takes a video file as input and will generate the results directory where all required results will be saved.
- I have implemented 1,2,3,5 and 6 modules. But I do not clearly understand 4th module. I belive 2nd module is doing the same thing. 2nd module also initially detects faces in images and then recognizes them.
- Faces in faces directory are just simply the faces obtained from dataset directory. These can be generated by running 1st cell and 5th module cell.
- In order to use model trained on “Rafa Nadal”, run the modules 1, 2 and 3. This will use the faces saved in faces directory. Now, let us say that you want to train the system for a new person for example “Christiano Ronaldo”, following scenarios arise:
    - Let us say you want train the sytem for 6 different persons (Rafa, Ronaldo, Messi .... etc), you create dataset for 6 persons. But you want to recognize only 2 (Rafa and Ronaldo) of them, then this is also possible. This can be done by using **names_to_recognize = ["Rafa Nadal", “Christiano Ronaldo”]** variable in first cell. Here you need to write the name of people you want to recognize. Right now it is just Rafa Nadal.
    - If you want to train the system that should be able to recognize both Ronaldo and Rafa, then just simply create the Ronaldo images in dataset and run, 4th module cell. This will overwrite the previous faces. Then put the name of Christiano Ronaldo in names_to_recognize variable in 1st cell of notebook. Now module-2 can be used to recognize Ronaldo and Rafa.

- I have used names_to_recognize variable in 1st cell because I think it is good to have full control over names to be recognized.

Instructions for Dataset Preparation

- In order to train the system for a new person, create a folder in dataset dirctory and the name of the folder must be the name of the person which you want to recognize.
- Make sure that the each train image must have only one face/person because if there will be multiple people in train image, the system will consider all people in that image as desired person like “Rafa Nadal”.
- The face in train image should be clear and easy to recognize.
- You provided me with few resources for training but those videos were of no use because of multiple people in the image and low resolution. But images given were of good quality and are present in dataset.

Some Observations

- Only 10 images are enough for the person whose face you want to recognize. You can just even use a single image for a person and the code will still work but it is recommended to have atleast 5 images.
- Don’t just extract frames from a video and use all those frames because in that case, system will not obtain a variety of data. You have to use several different images from different resources.
- Don’t increase the number of images too much like more than 20. Keep it around 5, 10, 15 etc.. Increasing images too much will overfit the data and other people would also be recognized as Rafa Nadal.

Algorithm working:

The algorithm calculates distance between testing face embedding and all training face embeddings. If this distance is less than tolerance_threshold, match is considered True otherwise False. The testing face is considered to be the face of the person with which-maximum number of matches are found. If this name is present in names_to_recognize list then we make a note of all of its data to prepare CSV file.

Hence increasing this value means embeddings with larger distances will also be True and hence, other males or women may also match (Flexible classifier). Decreasing this value means embeddings with lower distances will be True which means even Rafa would may also not find a match (Strict classifier). Play with it but 0.6 is recommended from official documentation and works fine here as well.

Calculation of score:

face_recognition.face_distance() returns a 1 dimensional array whose size is the same as number of training face embeddings and each element of this array is the distance between testing face embedding and corresponding training face embedding. It's value is around 0.6 with most of the values in range 0.5-0.7. So we perform following operations on it to obtain a single desired score value out of this array.

- First we apply a Min-max scaling to normalize its values in the range 0-1.
- Thehen calculate the mean of these normalized distances.
- At this stage, lower mean value corresponds to a face which has larger number of True matches (Should have high score) and higher mean value corresponds to a face which has larger number of True matches (Should have low score). So in order to reverse this trend, we subtract the mean value from 1 and resulting value is the ultimate score.

So now score neither does not depend on number of training images nor it will give same value for two faces. It will be always different because all operations performed are linear and ultimately, score is representing the mean distance. However, mostly the score found to be less than 0.6 unlike conventional score values of 0.998, 0.98 etc....