TABLE OF CONTENTS

How Does It Work?

This code has two important parts to it.
First is face detection. This is the process of determining the precise location of the face(s) in the entire frame.

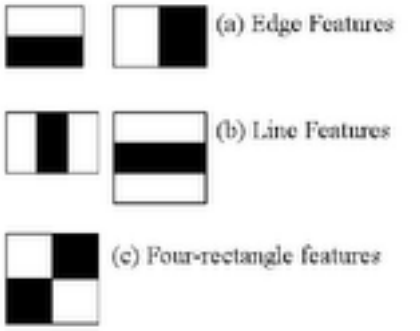
The Second is the actual face recognition. Once the presence of a face has been detected, the code has to determine if the face detected is of someone it has been instructed to identify.

We shall look at how each part is done in detail below .

Face detection:

The face is detected by a tried and tested method called Haar Cascades. This is a method that uses many different structures to identify 'Haar Features' ; eg: lines and edges in an image. Now, these lines, edges and other features can be used to locate a face as the eyebrows and nose bridge can be detected as lines and edges etc.

Now it's important to know that these features will be detected in multitudes across the image. This is because the Haar feature detector and classifier (the part that concludes if this feature is one likely to belong to a face) is a weak one. The accuracy of this relies on each of the many features we are looking for to add to the confidence of detecting if a frame contains a face or not.



Now, A window is passed across the enttire image (convolved) , snd at each step, the presence of each Haar feature is checked and kept count off. The more features a single frame contains, more likely it is to contain a face.

To the left is an image of a face frame with the different Haar features that match with it. Once this is done, we can

obtain the location of the face in the image. Using this we can crop the face out to the default dimensions (standard size) and then use that extracted face to perform recognition.

Now, there is a drawback using this, which is that this cannot detect faces that are turning away or obstructed. More over it doesn't account for the tilt or skew of the faces which often makes recognition a difficult task.

To overcome this we can do some image transformation on the detected face to align it and match some standard tilt and inclination. This is called an Affine Transformation. We use the angle of the line connecting the eyes and the bridge of the nose as references and ensure they are made straight. Importantly, affine transformation retains the relative distances and ratios of the original image despite the transformation this ensuring minimal data loss.

This however has not been added to our project due to time and computational complexity limitations.

Face Recognition :

This is the second part of our project where the extracted face has to be processed and identified.

We have chosen to use 'One shot learning' as it's popular in todays market use and requires no dynamic training for new datasets. i.e This doesn't change and using just one sample reference image, we can perform facial recognition with an impressive accuracy.

The idea is to use a pertained data set - one that effectively captures the unique features of the face and gives it to us as some unique embedding. For this we have used the Inception Inspired NN2 model as done in the FaceNet model. (Proposed by researches as Google).

(check cover page)

This well trained model takes in an input image (of a Face) and generates a 1x128 output vector that relates to the input face given. This is done for each of the user images and the embedding is stored in program memory. Thus when the faces are detected and passed on to the model live, a 1x128 output is obtained for each face image detected and the euclidian distance between each of the embeddings in memory and the face image is computed.

If the distance is below a minimum threshold, then the face is recognized as one of the people from the saved embedding and the job is done.

Now, occasional mis-recognition often happens, thus to improve this, we could make a histogram of the number of positive detections for each person and set a minimum frequency, only above which the face is considered to be recognized.

As to how the FaceNet model was trained etc, one can refer to the original paper published by them which implements the above model in PyTorch.
https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf

After recognition the Identity is saved to a file or list where it can be referred to for some sort of inference such as Attendance.

Code:

The code is attached below :

Note that the code has been partially borrowed and built upon from preexisting implementations of FaceNet by different people and combines snippets from many areas put together and assembled by us. It has been modified to suit our environment and requirements and the hyper parameters added or tuned to suit our test conditions.

Possible Enhancements

Scalability and Enhancements :

This model uses just one or a few images of the test subject to identify them and moreover can identify any number of faces simultaneously in one frame. This can be scaled to any extent, particularly for attendance as the processing can be done by a powerful remote serve while the each class just needs to send in a series of images in via the intranet to the server.

With this possibility of using HPCs to process, we can make a more robust model which minimizes false detection and ensures a seamless and ergonomic attendance taking process.

The ideal model we propose is that a camera is mounted somewhere in the classroom such that it's field of vision covers the whole class and attendance is taking autonomously while the class is being conducted.

This can be updated immediately onto the SLCM as part of an IOT network and students will be given a present receipt through SMS or other means.

Another thing that can be looked into is the post processing of detected face where alignment, illumination, etc can be standardized and made even.

Limitations of Current Model :

1. We need a better processing station to handle high traffic of image inflow without lag
2. We need a better camera that is mounted in a fixed place and gives us steady and quality images to process better
3. Dependent on connection to electricity and the intranet
4. Possible errors for very similar faces or twins
5. Highly dependent on specific facial landmarks that in rare cases could change
6. Dependent on lighting conditions