Door Opening using Facial Recognition

Project Description

Our project aims to provide a hi-tech security system using real-time facial recognition. This project uses Deep Learning to identify the person and recognize and give them access to open door automatically without any human interference.

Approach

- Creating a database of authorized persons.
- Processing these images to create their respective Embedding matrix.
- These embedding matrix contains unique features for the persons face.

- Take new input images from live camera feed.
- Using dlib frontal face detector which uses hog, to identify facial position in the input image.
- Creating embedding matrix for the cropped face from the input image.
- Comparing this embedding matrix with the saved embedding matrices to identify the person.
- This can be done by calculating the euclidean distance between the normalized embedding matrices.

- If the person is not in the database, give unknown or null as output.
- Opening the door on the basis, whether the person is in the database or not.

Tech Stack

- Jupyter Notebook
- Anaconda
- Google Colab
- Github

- VS-Code

- Python
- Keras
- Tensorflow

- Numpy
- Open CV
- Dlib
- Kaggle pins dataset
- Labeled Faces In The Wild(LFW) Dataset

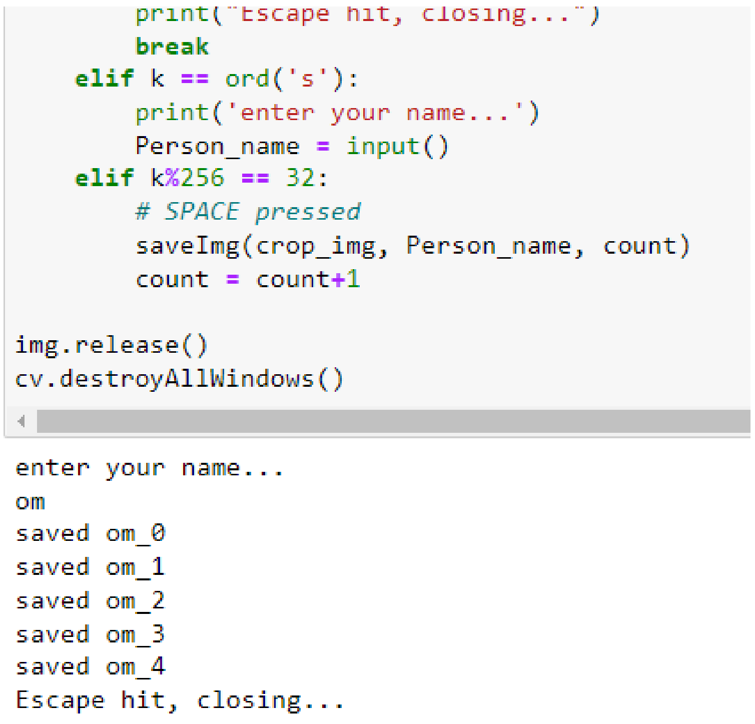
And discord…

Working/Implementation

- Create virtual environment in jupyter notebook
- Installing all required libraries in the virtual environment.
- Importing required libraries in jupyter file.

Database creation

- Using opencv to access the camera.
- Converting every frame from BGR by default to RGB
- Using dlib's frontal face detector which implements HOG to detect facial position in the input image for each frame
- This provides the coordinates of the boundaries of the face
- Drawing a rectangle around the face using these coordinates.
- At the same time using this coordinate to crop the image and save it in the folder named dataset. This task initiates when the person presses 's'.
- Then we enter the name to create a folder in the directory dataset of the same name as entered.
- If the folder named "dataset" does not exist then we create it using the code at the same time.
- The image is saved every time the person presses 'Space'.



```
        print("Escape hit, closing...")
        break
    elif k == ord('s'):
        print('enter your name...')
        Person_name = input()
    elif k%256 == 32:
        # SPACE pressed
        saveImg(crop_img, Person_name, count)
        count = count+1

img.release()
cv.destroyAllWindows()
```

```
enter your name...
om
saved om_0
saved om_1
saved om_2
saved om_3
saved om_4
Escape hit, closing...
```

-

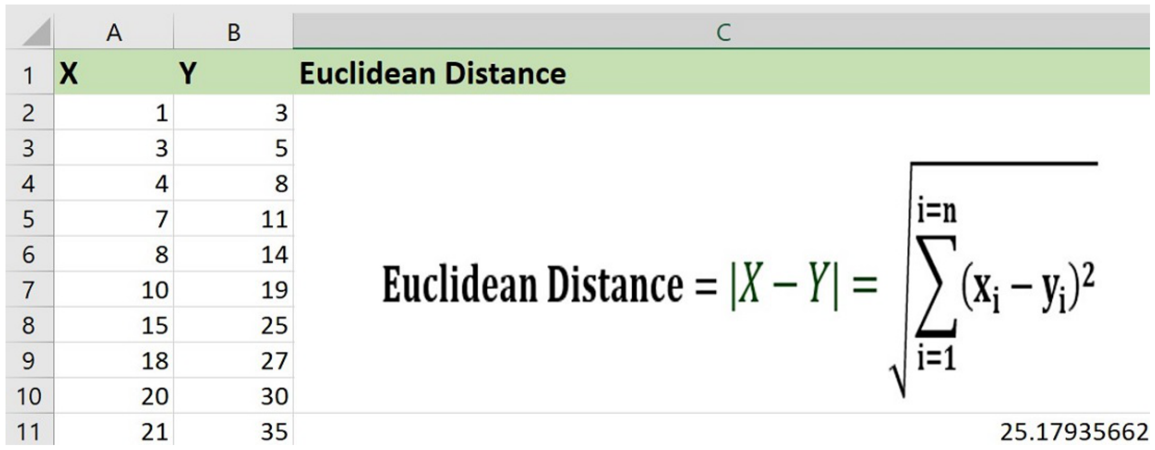
- This way we can create the dataset.
- When we are done saving cropped images, press ‘ESC’ to end the operation.

Model Creation

- Building our CNN model using Keras to create an Embedding Matrix.
- Dropping the last few layers of the model since they are trained on other faces and not on our dataset.
- Instead of training our model from scratch, loading pre-trained weights and updating them to suit our needs.
- We will be using weights trained on vggface2 dataset, which is trained on 3.3 million faces divided into 9131 classes.
- The weights used from the pretrained model is based on DeepFace Architecture.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 C1 (Conv2D)                 (None, 142, 142, 32)      11648

 M2 (MaxPooling2D)           (None, 71, 71, 32)        0

 C3 (Conv2D)                 (None, 63, 63, 16)        41488

 L4 (LocallyConnected2D)     (None, 55, 55, 16)        62774800

 L5 (LocallyConnected2D)     (None, 25, 25, 16)        7850000

 L6 (LocallyConnected2D)     (None, 21, 21, 16)        2829456

 F0 (Flatten)                (None, 7056)              0

 F7 (Dense)                  (None, 4096)              28905472

 D0 (Dropout)                (None, 4096)              0

 F8 (Dense)                  (None, 8631)              35361207

=================================================================
Total params: 137,774,071
Trainable params: 137,774,071
Non-trainable params: 0
_________________________________________________________________
```

- Now we can create an Embedding Matrix of dimension 4096 of a face,
- Creating a function for finding Euclidean Distance between two Embedding Matrices after normalizing it.

- 

	A	B	C
1	X	Y	Euclidean Distance
2	1	3	
3	3	5	
4	4	8	
5	7	11	
6	8	14	
7	10	19	
8	15	25	
9	18	27	
10	20	30	
11	21	35	25.17935662

$$\text{Euclidean Distance} = |X - Y| = \sqrt{\sum_{i=1}^{i=n} (x_i - y_i)^2}$$

- Before creating the embedding matrix, we will have to pre-process the image to convert it into an array.
- Creating an embedding matrix list and corresponding name list for the images in the dataset
- Saving this data for future use.
- Loading the data for prediction

Prediction Function

- Creating a function to predict a new image whether the person is in the dataset or not.
- We will create the embedding matrix of the input face and compare it with all the embedding matrices in the embedding matrices list which we created and loaded earlier.
- Comparison is done by calculating the euclidean distance between the normalized embedding matrices.
- If the euclidean distance is less than a threshold value then the new face is in the dataset and we can recognize him otherwise we consider the new face as unknown.
- This threshold value is a hyperparameter which we decide on the basis of several observations.

Real-Time Face Recognition

- Firstly we set the initial cropping boundaries to 0 to update it later.
- We also set open_door as False ie the door is currently locked.
- Using Open CV, we take live camera feed from the webcam and get input image in realtime.
- Then by using dlib's frontal face detector we find the facial coordinates and then crop the image so as to get only the face of the person from the entire image.
- The face detector takes input image and gives the coordinates of the boundaries of the face.
- Using these boundaries, we draw a rectangle around it.

- We set condition to avoid crashing of the video feed when the value of the boundary becomes negative and when the detector does not detect any face.
- If both these above conditioned are surpassed then we get the cropped face from the input image.
- This face is saved in Test_Face Folder and is constantly updated in realtime.
- This cropped face is then passed to the predict function which either identifies the person or says unknown.
- The name of the person is shown above the rectangle box drawn earlier,
- To unlock the door the user must press ‘space’.
- If the user is in the database then the door will be unlocked and we get “OPEN DOOR” output. Also open_door is now set to True and the gets unlocked and then we come out of the loop.
- After this, the camera is released and all output windows are closed.
- If the person is not in the dataset, we get “ACCESS DENIED” as output and open_door remains closed and the loop continues.
- To manually break the operation and come out of the loop press “ESC”.

References/resources

- https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/ (for HOG)
- https://github.com/swghosh/DeepFace/releases (for pretrained weights)
- Deepface architecture