Name: Steven Phung

ID: 028023383

Professor: Ju Cheol Moon

Due: 4/7/22

Assignment #8 Report

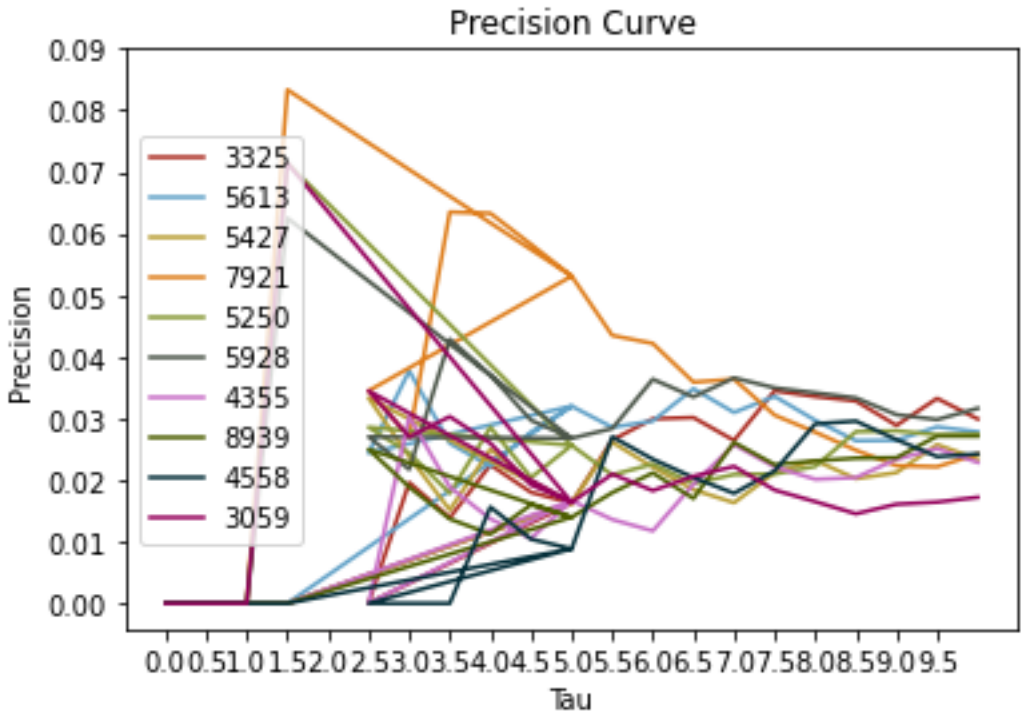
The open source code of YOLO V3 that I referenced and used for this assignment is called YOLOFace. I used the https://github.com/nbishdev/yoloface fork because it solved an index error that the main branch had. The pre-trained weights I used were the ones linked in the repo: yolov3-wider_16000.weights at http://shuoyang1213.me/WIDERFACE/ . I modified some of YOLOFace's files to have smoother output. For yoloface.py I commented out cv2 windows that showed the images since I just wanted it to run in the background and it not have to interrupt the system every few seconds. I also modified utils.py to write the coordinates of the bounding box to a file. Then I used these coordinates for image2vect.py to crop each image to only contain the face. Finally, it returned an embedding vector for the cropped image.

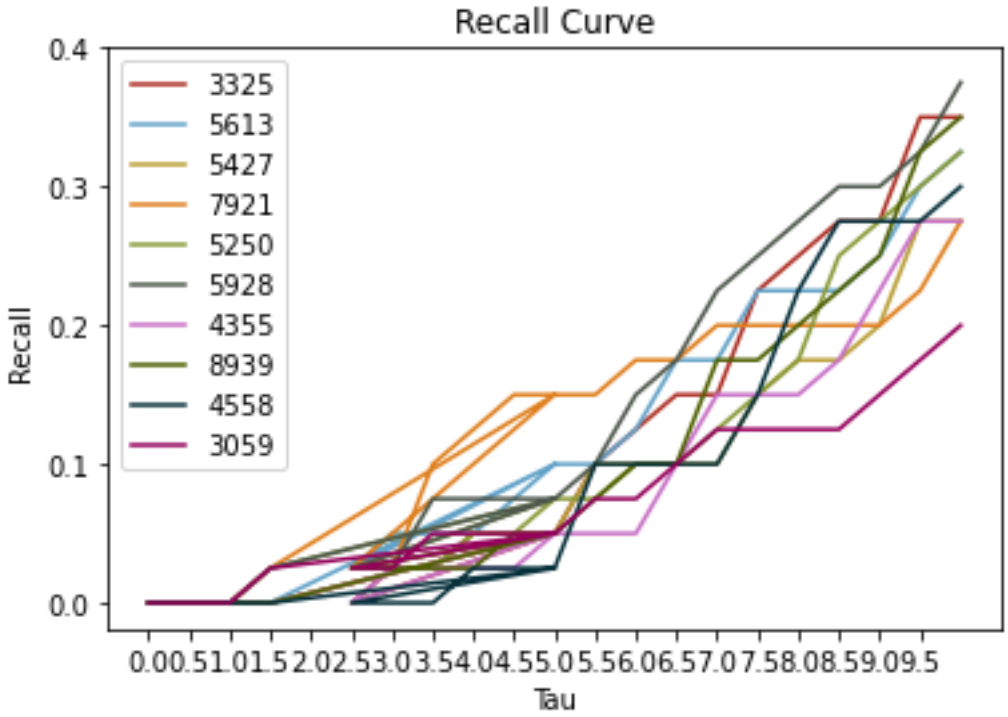
The program imageFinder.py takes a single command line input as the initial file to calculate distances for. The input is simply *file_name* such as "000109.jpg" or "000308.jpg." The embedding vector is received from image2vect.py and then done similarly for the other 1,199 images. Then I implemented euclidean_distance(vectorA, vectorB) function manually, it takes 2 vectors, for each vector's index it subtracts, then squares this summation, and adds it to a total. At the very end I square root the entire total and that is my calculated Euclidean distance. After this I had a list of hyper-parameter Tau's from 0-2 incrementing by 0.1. Based on the Tau values it would output which image files were recognized as the same celebrity of the input image. This was repeated for 9 other randomly selected unique celebrities. I am assuming 10 total unique so I

simply generated 9 others from a random sample which excludes the initial input celebrity. Finally, precision and recall were calculated for each celebrity and each tau in the interval.

I am unsure about my results and I think it may have to do with my Euclidean distance calculations. I think results were off because some images were much larger or some images were much smaller than others, resulting in large differences even if they were similar celebrities. If I had time to go back and repeat this project I would consider implementing some function that would resize the cropped images to all be a specific dimension, which would in turn result in equal embedding vectors. Lastly, I expected this program to take more time to finish, but even in the end it took more time than I expected.

Output for precision and recall curves:




Recall Curve
Recall
Tau
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5
3325
5613
5427
7921
5250
5928
4355
8939
4558
3059