Facial recognition

- Tim. E. Doughery: Undergraduate Senior majoring in Comp. Sci.
- Chenhao Lu: Undergraduate Senior in Comp. Sci.
- Seho Chung: Undergraduate Senior in Comp. Sci.

Introduction: For this project, our objective is to develop a model for facial recognition and identification. This subject can be found in many modern-day devices, especially most flagship phones. It is a controversial topic, and yet sees much popularity in research. Google, for instance, has managed to develop perhaps the most popular facial recognition model known as facenet, upon which many open-source projects hope to emulate. Over the years, improvements to the study of facial recognition have come to light, or may yet come. However, our motivation in choosing this topic was not one of improvement, but rather understanding how one manages to develop a reliable model.

Method: For data, we decided to use the Labelled Faces in the Wild (LFW) dataset from the University of Massachusetts Amherst. Secondly, we will also use the LFW Simulated masked face dataset to test the model's ability to recognize masked faces. To preprocess these datasets, we will attempt to crop the faces and then divide the datasets into training and testing sets. Lastly, the model we will be using is Google's facenet, a deep convolutional neural network. We will first train the model on LFW, and once we attain a high accuracy, we will test it on the masked dataset, and retrain the model with a mixed dataset of both masked and unmasked faces.

Results: As of right now, we've successfully managed to preprocess the data and develop a model from a pre-trained inception_resnet_v1 model found here [1]. The model can take input images of dimension (160, 160), and output a 128-dimensional embedding that can be used to compute distances between images. The model is compiled using the triplet loss [2] and adam optimizer. The next step is to train the model using triplet loss function to improve the quality of the output embeddings, and develop a second classifier to classify the embeddings into the label categories.

Discussions:

We have encountered a few problems with the implementation and managed to resolve them with alternate approaches. The first problem arises from the OpenCV haar cascade where it failed to detect faces in some of the data samples in the LFW dataset. The solution we came up with was to switch to a deep-funneled LFW dataset that pre-aligns all images so that the eyes and noses are in the same position, and simply use the center part of the image as the face when the haar cascade fails. We also ran into problems with how to read and store the data. Our first attempt was to read all images into a dictionary with the names as the keys, but it did not work well since there were no efficient ways of splitting a dictionary into training and testing sets. We then tried using separate lists for images and labels, and despite solving the issue of splitting data, it took a very long time to read all images in at the same time. In our third and final attempt, we wrote a function that automatically creates the testing directory according to the suggested split [3] on the official LFW page. We then used the keras ImageDataGenerator module to read in images using batch size 32, and managed to significantly reduce the running time from the previous approaches.

References:

[1] https://github.com/nyoki-mtl/keras-facenet

[2] https://www.freecodecamp.org/news/making-your-own-face-recognition-system-29a8e728107c/

[3] http://vis-www.cs.umass.edu/lfw/peopleDevTest.txt

Example images in a batch: