**Face Recognition with OpenCV Library**

Over last ten years, face recognition has become a hot topic in computer vision, which has been widely applied in unlock phone(iPhoneX), security camera, some digital make-up application(Meitu) and can be compared to other biometrics such as fingerprint or eye iris recognition system.

A general statement of the face recognition problem can be formulated as follows: Given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces.

Originally, I was intended to implement machine learning algorithm on detection of clinical prostate tissue, then I realized it would be difficult for a starter. So I started with the face recognition project which has been well-studied, and I would like to get familiar with OpenCv library, which will benefit the further study.

I will use the CASIA-WebFace database — MS-Celeb-1M dataset for training, which is significantly larger but also contains significantly more label noise, and therefore it is crucial to apply dataset filtering on this dataset.

I will use multi-task neural network to implement the task after trying EigenFace Recognizer, FisherFace Recognizer or Local Binary Patterns Histogram which OpenCv equipped with,

Ideally, when I test a new image, the output space should be circled the faces if there exists some, or return the original image if there’s no faces. Furthermore, I may want to get the locations and outlines of each person’s nose, eyes, mouth on the face, which will be super useful for lots of important stuff, like digital make-up App. Or I can apply on a video not just a still image.

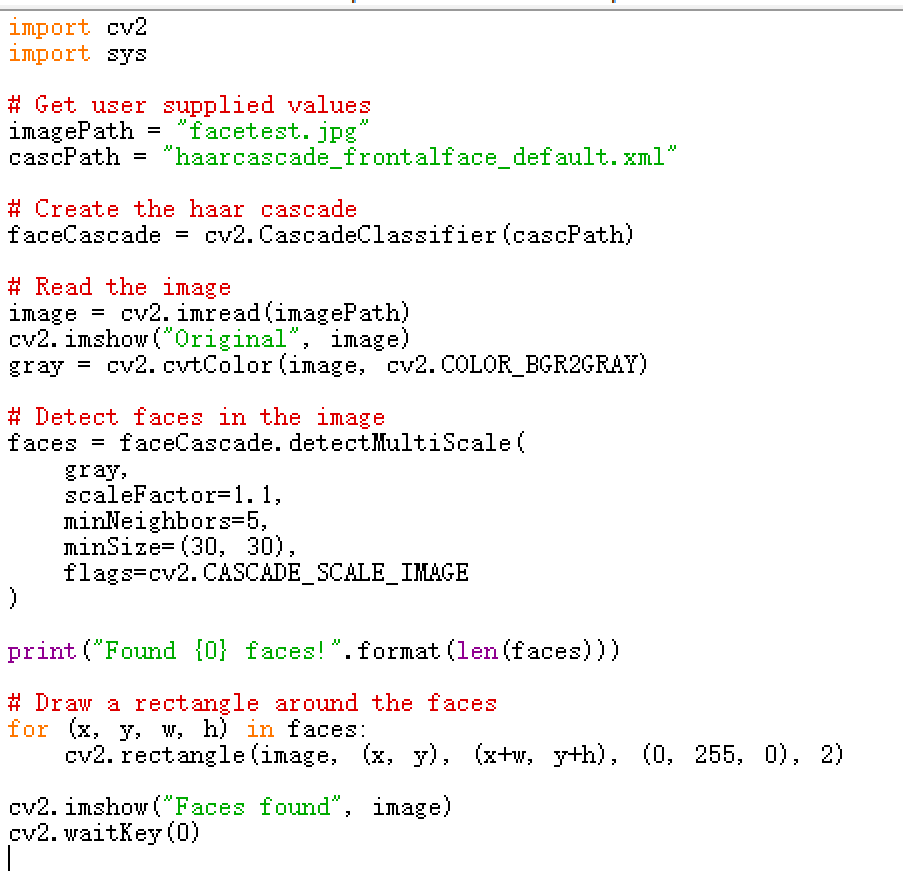
I found OpenCV is pretty mature for face recognition, just 30 lines python code can complete a test with Haar Cascade Classifier and detectMultiScale function. I tested by myself using the following codes and corresponding result. The green rectangle circled the recognized face.

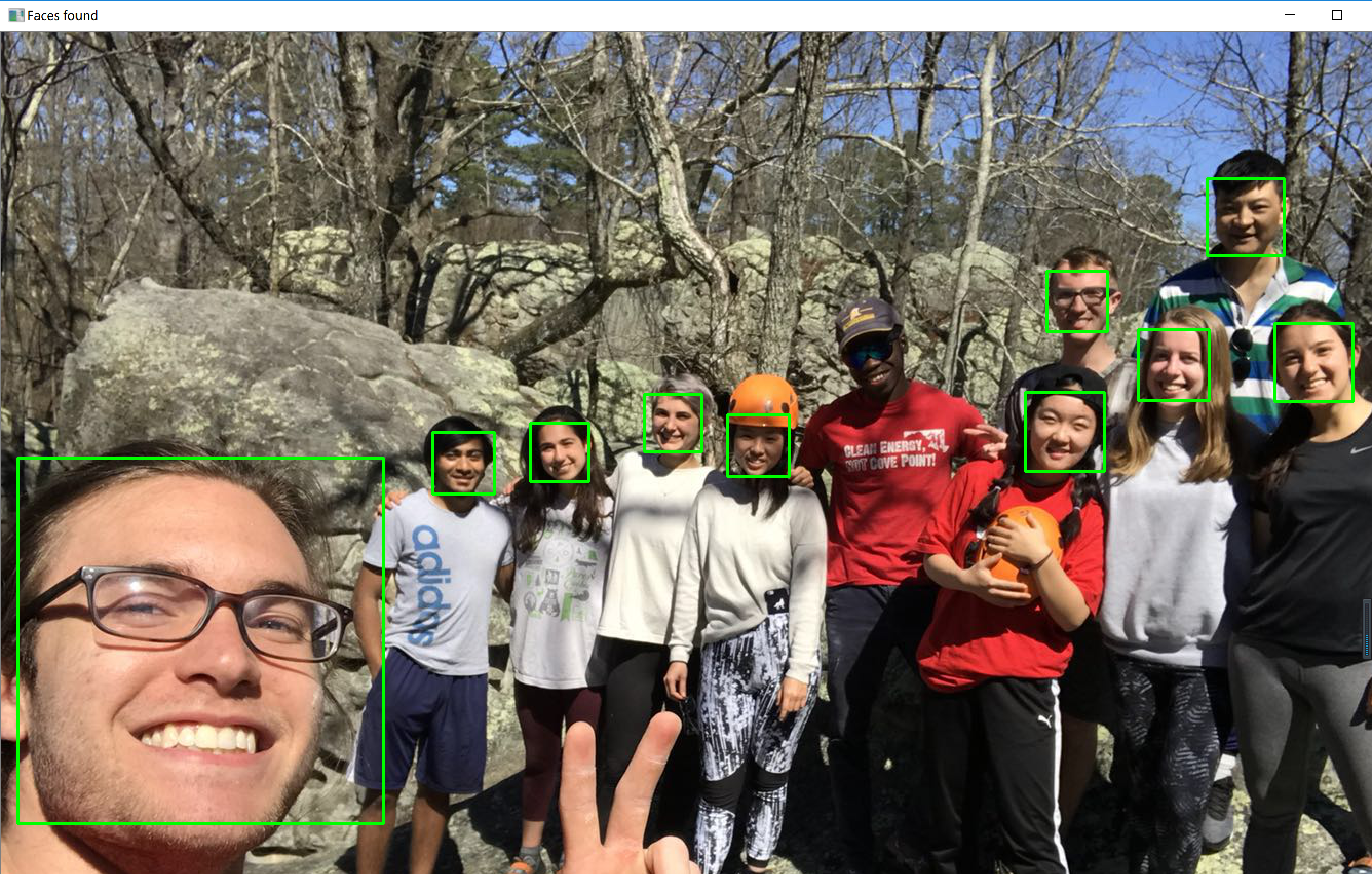
You can see file in my github

<https://github.com/zzzghm/CMPS-4720-6720-Sp18/tree/master/Course%20Project>

face1.py

facetest.jpg

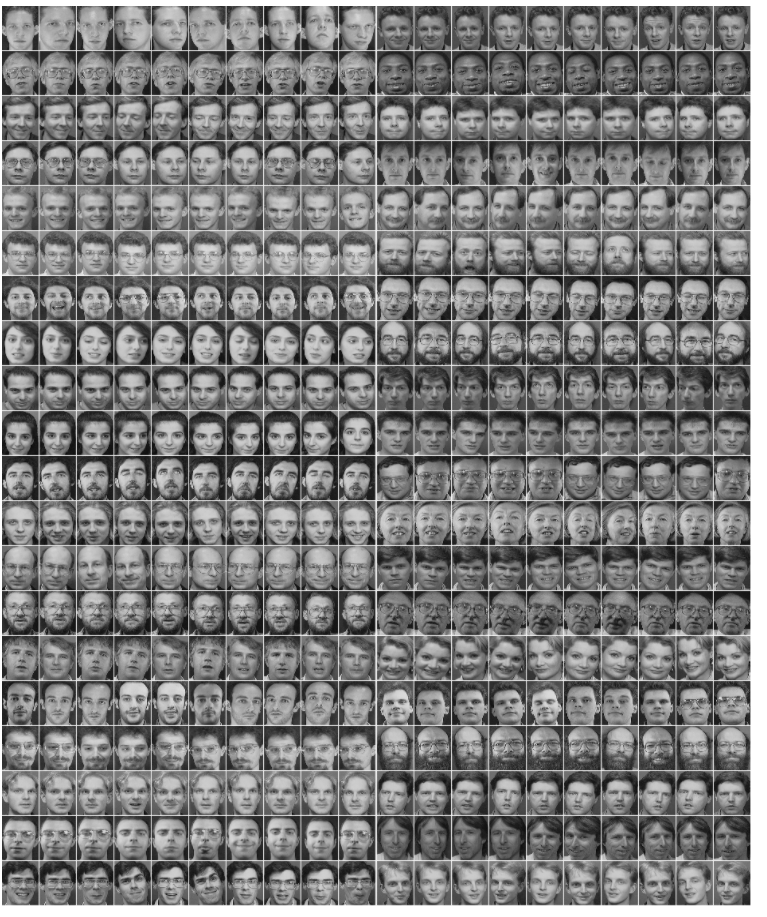




So I decided to write my own code based on the paper I presented ‘Deep learning in Metastatic Breast Cancer’ to implement the CNN algorithm.

First, I started with a pretty smaller database ‘the Olivetti faces dataset’. As described on the original website: There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

The image is quantized to 256 grey levels and stored as unsigned 8-bit integers; the loader will convert these to floating point values on the interval [0, 1], which are easier to work with for many algorithms. The “target” for this database is an integer from 0 to 39 indicating the identity of the person pictured; however, with only 10 examples per class, this relatively small dataset is more interesting from an unsupervised or semi-supervised perspective. The original dataset consisted of 92 x 112, while the version available here consists of 64x64 images.



I would like to build the neural network based on LeNet5. Inputs are the images, C1 is convolution layer, S2 is pooling layer, C3 is another convolution layer, S4 is another pooling layer, then fully connected to C5, C5 to F6, which can be seen as two MLP hidden layers, logistic regression as classification in the end, the output of the model will be the 0-39 labels.

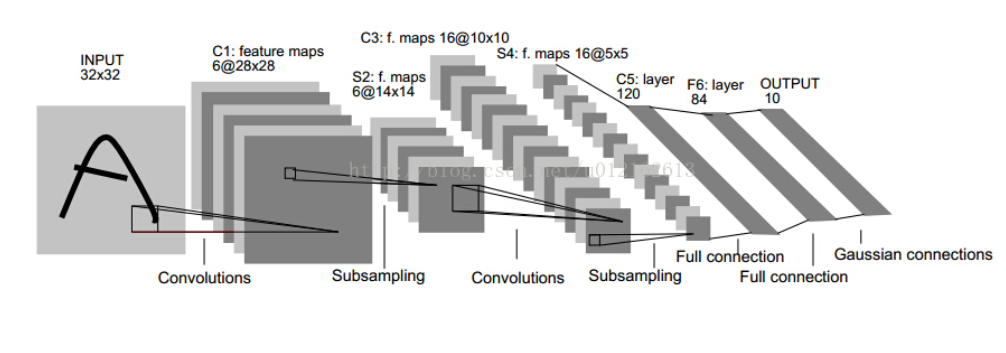
You can see the code on github

<https://github.com/zzzghm/CMPS-4720-6720-Sp18/tree/master/Course%20Project>

olivettifaces.gif

CNN\_face\_detection\_train.py

CNN\_face\_detection\_test.py



And the result is saved as result.py

For CNN\_face\_detection\_train.py

**Best validation score of 5.000000 % obtained at iteration 304, with test performance 7.500000 %**

**The code for file train\_CNN\_olivettifaces.py ran for 2.17m**

For CNN\_face\_detection\_test.py

**picture: 89 is person 8, but mis-predicted as person 34**

**picture: 178 is person 17, but mis-predicted as person 37**

**picture: 189 is person 18, but mis-predicted as person 6**

**picture: 299 is person 29, but mis-predicted as person 39**

**picture: 368 is person 36, but mis-predicted as person 20**