CSE4020 - Machine Learning

Facial Detection and Recognition

Final Report

Video Review Link -
https://drive.google.com/open?id=1_gSJ1DVkjFwhSPzJ9lfLuS92WoUhMQdN

Submitted by

17BCE0191 Allen Biju Thomas
17BCE0211 Nachimuthu N
17BCE0279 Abhishek Thomas
17BCE0012 Sai Saathvik

Under the Guidance of

Prof. S.M. Jaisakthi

B. Tech in,
Computer Science and Engineering
School of Computing Science & Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

1. Table Of Contents

2.Abstract

We present an approach to the detection and identification of human faces and describe a working, near-real-time face recognition system which tracks a subject's head and then recognizes the person by comparing characteristics of the face to those of known individuals. Our approach treats face recognition as a two-dimensional recognition problem, taking advantage of the fact that faces are are normally upright and thus may be described by a small set of 2-D characteristic views. Face images are projected onto a feature space ("face space") that best encodes the variation among known face images. The resultant of the projection provides a 128d vector which describes the face. The models use this model to detect the unknown faces. The framework provides the ability to learn to recognize new faces in an unsupervised manner.

3.Motivation

This project is based on a image processing package in python called dlib and a face recognition package called as face_recognition. Face recognition is a growing field under AI and convoluted neural networks. The application of face recognition is limitless and advancements are being made almost every day. The global facial recognition market was valued at USD 4.4 billion in 2019, and it is projected to be valued at USD 10.9 billion by 2025, registering a CAGR of 17.6% over the forecast period, 2020-2025. Facial recognition has been gaining prominence in recent times, owing to its benefits over traditional surveillance techniques, like biometrics. Governments across the world have been investing significant resources in the facial recognition technology, among which, the United States and China are leading adopters.

Owning to increased cyberattacks, terrorism and identity thefts, the need for face recognition is now more than ever. Multi-factor authentication is an upcoming adaptation in the security aspects of any software and service and face recognition plays a huge role in this. A lot of potential is present in this field with all the major cloud providers adding face recognition as a service in their platform.

4.Objectives

The objective of this project is to develop a face recognition model in python language. This project is completely focused on getting an insight into face recognition technology and find out the future advancements and possibilities available. This project also focuses on introducing the various methods available for face recognition.

5.Introduction

In this project the learning method used by our model is called as deep metric learning. Deep metric learning is different from other learning methods as it instead of trying to output a single label or even the bounding boxes of objects in the image, it given out a real value featured vector. The dlib facial recognition model which is the base model used by face_recognition package used this 128d vector to identify the face. The training of the model is done using triplets. The way this triplet method work is that, during training the model first generates a 128d vector for the given face image, later when another face of the same class(person) is provided then model then generates a 128d vector for the new image and it compares the new vector with the old vector of the face, based on the new vector the old vector is tweaked such that both the vectors are as close as possible. Now, if a picture which belongs to a different class is given, then the already present vector relating to that image is tweaked so that they are as close as possible and vectors of other classes are as far as possible from the current vector. In this was all the vectors of different classes present in the trained model will be as far as possible from each other.

When using this model for prediction, the model again generates a 128d vector for the given input image and that generated vector is then compared with all the available vectors from during the training, the class of the vector which the closest to the vector of the input image is given as the output for the image. The underlying convoluted neural network used in this deep metric learning is ResNet-34 and this network in dlib is a pre-trained model with approx. 3 million images on the labelled faces in the wild dataset. This model was then used by Adam Geitgey to build the face_recognition package which detects the faces and generates the 128d vectors.

6.Literature Survey

A general statement of the face recognition problem (in computer vision) 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.

Facial recognition generally involves two stages:

Face Detection where a photo is searched to find a face, then the image

is processed to crop and extract the person's face for easier recognition.

Face Recognition where that detected and processed face is compared to a database of known faces, to decide who that person is.

Since 2002, face detection can be performed fairly easily and reliably with Intel's open source framework called OpenCV [1]. This framework has an in built Face Detector that works in roughly 90-95% of clear photos of a person looking forward at the camera. However, detecting a person's face when that person is viewed from an angle is usually harder, sometimes requiring 3D Head Pose Estimation. Also, lack of proper brightness of an image can greatly increase the difficulty of detecting a face, or increased contrast in shadows on the face, or maybe the picture is blurry, or the person is wearing glasses, etc.

Face recognition however is much less reliable than face detection, with an accuracy of 30-70% in general. Face recognition has been a strong field of research since the 1990s, but is still a far way away from a reliable method of user authentication. More and more techniques are being developed each year. The Eigenface technique is considered the simplest method of accurate face recognition, but many other (much more complicated) methods or combinations of multiple methods are slightly more accurate.

OpenCV was started at Intel in 1999 by Gary Bradski for the purposes of accelerating research in and commercial applications of computer vision in the world and, for Intel, creating a demand for ever more powerful computers by such applications. Vadim Pisarevsky joined Gary to manage Intel's Russian software OpenCV team. Over time the OpenCV team moved on to other companies and other Research. Several of the original team eventually ended working in robotics and found their way to Willow Garage. In 2008, Willow Garage saw the need to rapidly advance robotic perception capabilities in an open

way that leverages the entire research and commercial community and began actively supporting Opencv, with Gary and Vadim once again leading the effort [2]

Intel's open-source computer-Vision library can greatly simplify computer programming. It includes vision advanced capabilities face detection, face tracking, face recognition, Kalman filtering, and a variety of artificial intelligence (AI) methods - in ready-to use form. In addition, it provides many basic computer-vision algorithms via its lower-level APIS.

OpenCV has the advantage of being a multi-platform framework; it supports both Windows and Linux, and more recently, Mac OS X.

OpenCV has so many capabilities it can seem overwhelming at first. A good understanding of how these methods work is the key to getting good results when using OpenCV. Fortunately, only a select few need to be known beforehand to get started.

OpenCV's functionality that will be used for facial recognition is contained within several modules. Following is a short description of the key namespaces:

CXCORE namespace contains basic data type definitions, linear algebra and statistics methods, the persistence functions and the handlers.

Somewhat oddly, the graphics functions for drawing on images are located here as well.

CV namespace and contains camera image calibration processing methods. The computational geometry functions are also located here.

CVAUX namespace is described in OpenCV's documentation as containing obsolete and experimental code.

However, the simplest interfaces for face recognition are in this module. The code behind them is specialized for face recognition, and they're widely used for that purpose.

ML namespace contains machine- learning interfaces.

HighGUI namespace contains the basic I/O interfaces and multi-platform windowing capabilities.

CVCAM namespace contains interfaces for video access through DirectX on 32 bit Windows platforms.

Eigenfaces is considered the simplest method of accurate face recognition, but many other (much more complicated) methods or combinations of multiple methods are slightly more accurate.

Most resources on face recognition are for basic Neural Networks, which usually don't work as well as Eigenfaces does.

And unfortunately there are only some basic explanation for better type of face recognition than Eigenfaces, such as recognition from video and other techniques at the Face Recognition Homepage [4] or 3D Face Recognition Wikipedia [5] and Active page Appearance Models page [6]. But for other techniques, you should read some recent computer vision research papers from CVPR and other computer vision conferences. Most computer vision or machine vision conference include new advances in face detection and face recognition that give slightly better accuracy. So for example you can look for the CVPR10 and CVPRO9 conferences [7].

7.Proposed Methodology

a. Dataset description:

All the models used in this project are pre-trained using the labelled images in the wild dataset. To customize the model for our own purposes a custom-built dataset which contains the images of all the team members in this project was built. This dataset consists of images in which there is only one-person present. The images are structures in such a way that a folder is created for each class(person) and all the images of that person is put into that folder.

b. Architectural Diagram:

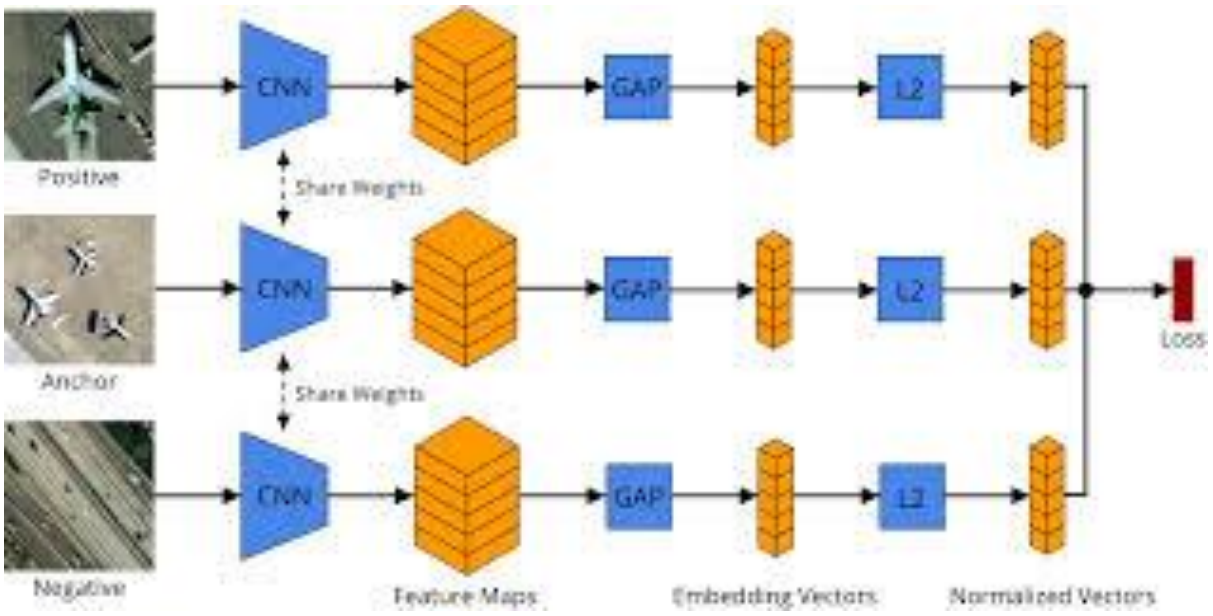

Fig 1: architecture of triplet network

c. Module Diagram

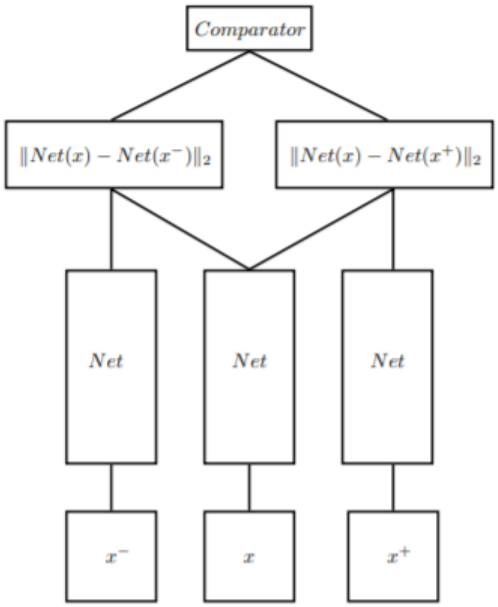

Fig 2: triplet network structure

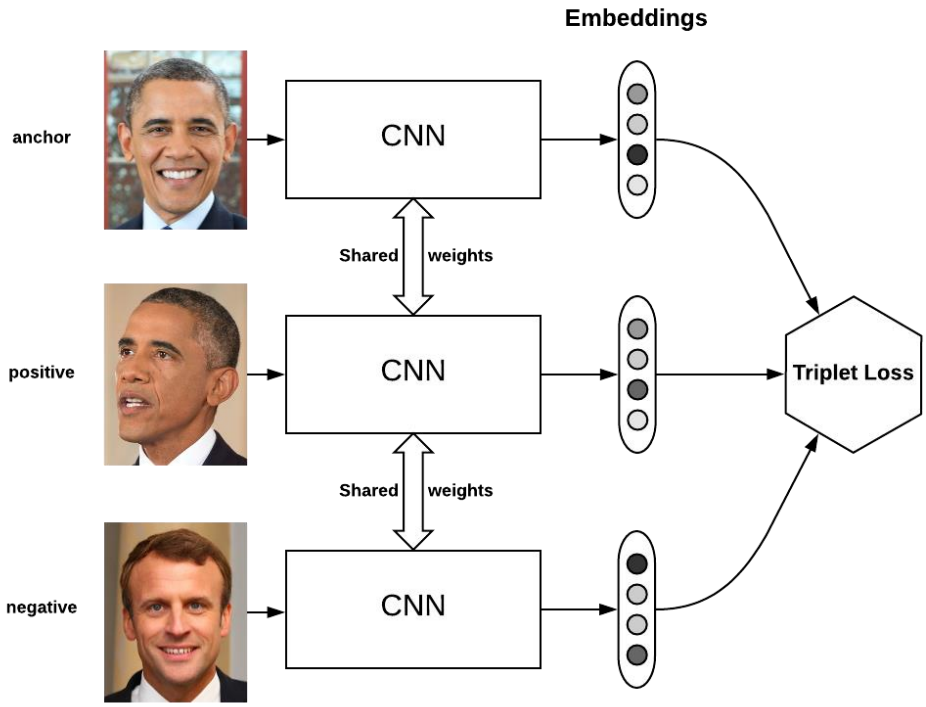

Fig 3: example of working of triplet network

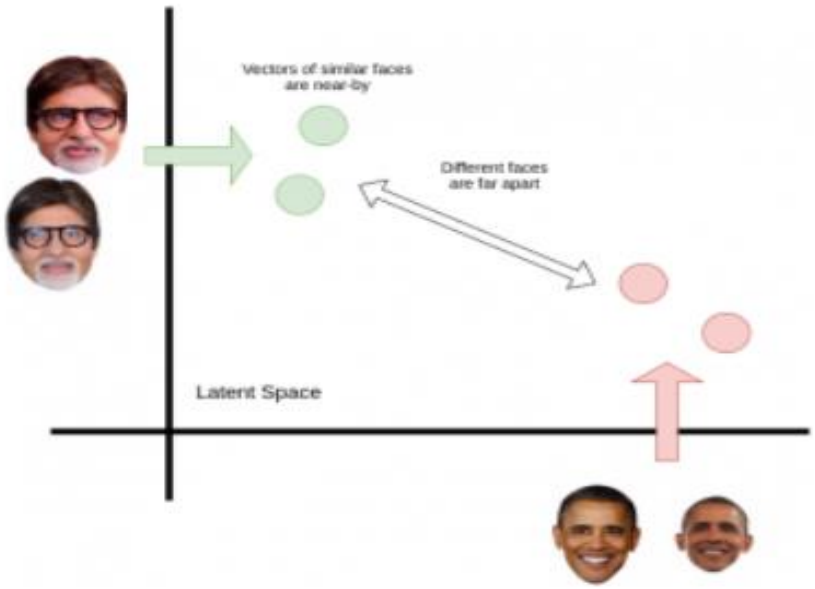

fig 4: image showing representation of image vector in 2D

The above figure shows the plotting of the images for a 2-dimensional images, this figure explains it in 2 dimension, but the same kind of representation occurs in 128 dimension.

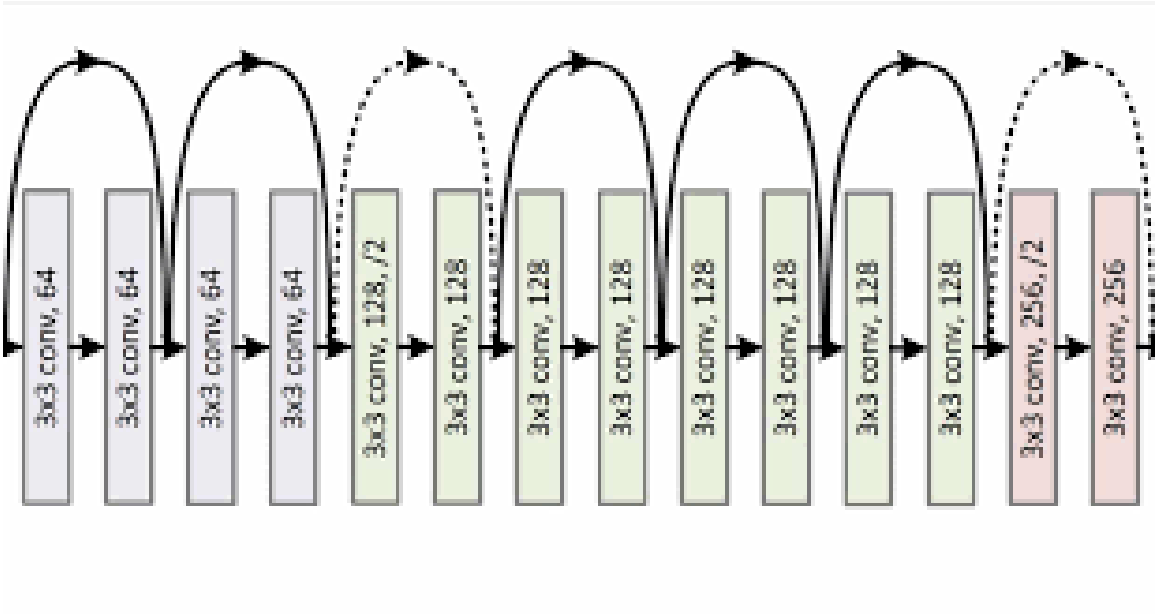

fig 5: image showing the architecture of ResNet34 model.

d. Experimental Setup:

Programming Language - Python

Packages Used :

Opencv - For Reading and manipulating Image and video data

Face-recognition - For encoding images and finding the similarity between different faces using Conventional neural network(CNN).

Pickle - For Saving Trained Values

DataSet :

A collection of pictures of each individual in the group were used to train the model

e. Results and discussion

Pictures of group members were used to train the model , and the tolerance was kept pretty high as we did not have enough data for precise face detection. But the model still performed well even with less training data and high tolerance, it managed to detect and distinguish each of out face.

Sometime the Names were mismatched but with more training data we are sure the model would perform better

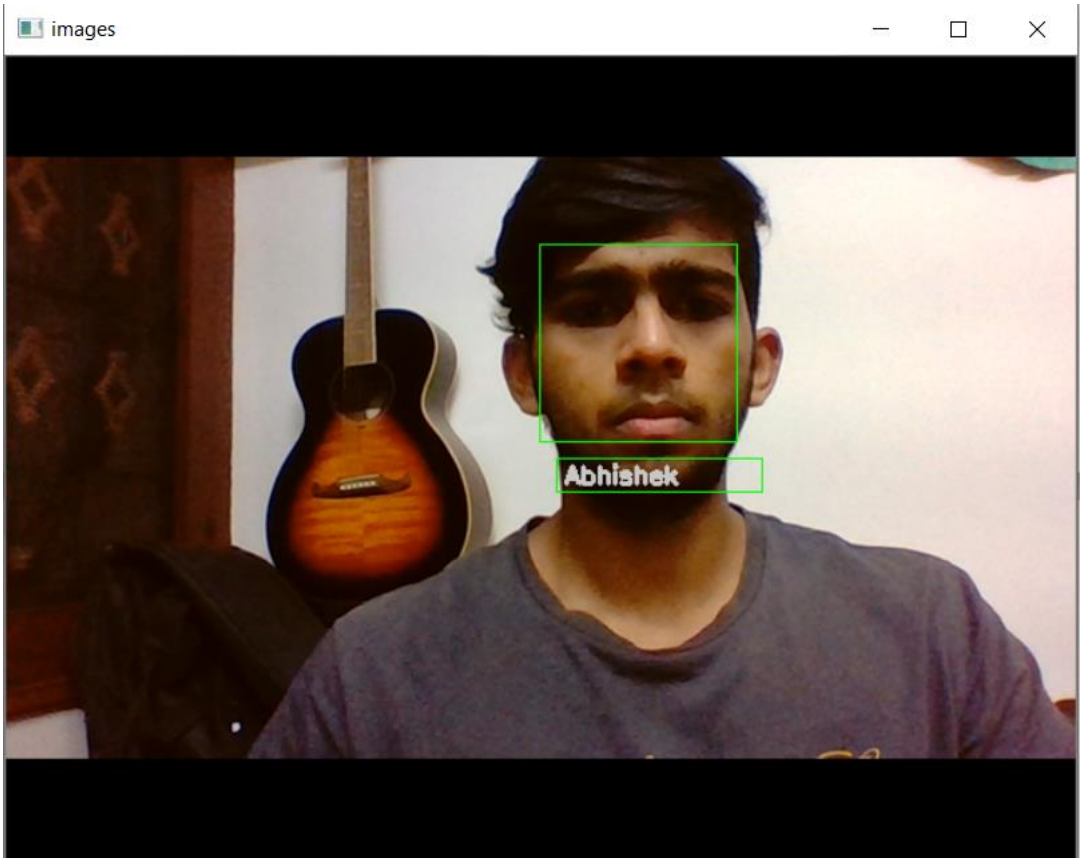

fig 6: output 1

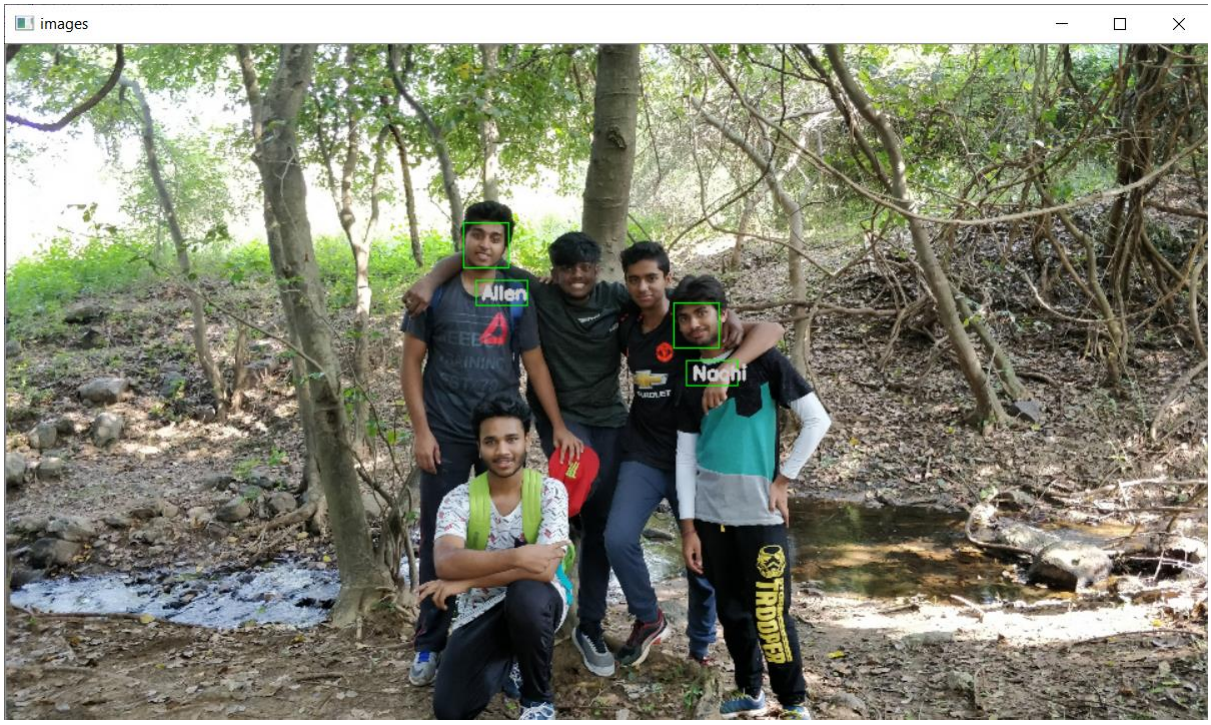

fig 7: output 2

fig 8: output 3

fig 9: output 4

fig 10: output 5

8.Conclusion

In this project we were able to achieve a fully function face recognition model. This model is capable of identifying multiple faces in one image. It is also capable of recognizing millions of faces given that the data for training was provided. With the exposure we received while working on this project, it came to our knowledge that there are better and more complex models that are used to detect faces with liveliness with ensures that a person is present and its not just the photo of the person kept in front of the camera. These advanced model are capable of being embedded into software and services as a part of multifactor authentication.

9.References

[1] Face Detection and Recognition using OpenCV, Article, http://shervinemami.info/faceRecognition.html, Published by Shervin Emami, 2010

[2] Seeing with OpenCV, Article, http://www.cognotics.com/opencv/servo_2007_series/part_1/index.html, Published by Robin Hewitt, 2010

[3] OpenCV Homepage, http://opencv.willowgarage.com

[4] Face Recognition Homepage, http://www.face-rec.org/algorithms/ [5] Wikipedia, Three-dimensional face recognition, http://en.wikipedia.org/wiki/Three-dimensional_face_recognition

[6] Wikipedia, Active appearance model, http://en.wikipedia.org/wiki/Active_appearance_model

[7] Computer Vision Papers, http://www.cvpapers.com/

[8] Wikipedia, Histogram equalization, http://en.wikipedia.org/wiki/Histogram_equalization

[9] Shervin Emami, Rotating or Resizing an Image in OpenCV, http://shervinemami.info/imageTransforms.html

10. Appendix - Code

a. training.py

```
import face_recognition
import os
import cv2
import pickle
TOLERANCE = 0.5
FRAME_THICK = 1
FONT_THICK = 2
KNOWN_DIR = "Known"
known_faces=[]
known_names=[]
print("Started Training")
for name in os.listdir(KNOWN_DIR):
    for f in os.listdir(f"{KNOWN_DIR}/{name}"):
        image=face_recognition.load_image_file(f"{KNOWN_DIR}/{name}/{f}")
        try:
            encoding=face_recognition.face_encodings(image,num_jitters=50)
            print("Name ",name," ",f)
            print("Encoding ",encoding)
            known_faces.append(encoding[0])
            known_names.append(name)
        except:
            pass
print("Stopped Training")
with open('faces.pickle','wb') as f:
    pickle.dump(known_faces,f,protocol=pickle.HIGHEST_PROTOCOL)
with open('names.pickle','wb') as f:
```

```
    pickle.dump(known_names,f,protocol=pickle.HIGHEST_PROTOCOL)
print("Saved")
```

b. image_recognise.py

```
import face_recognition
import os
import cv2
import pickle
KNOWN_DIR = "Known"
UNKNOWN_DIR = "unknown"
TOLERANCE = 0.45
FRAME_THICK = 1
FONT_THICK = 2
model = "hog"
with open('faces.pickle', 'rb') as handle:
    known_faces = pickle.load(handle)
with open('names.pickle', 'rb') as handle:
    known_names = pickle.load(handle)
for name in os.listdir(UNKNOWN_DIR):
    image = face_recognition.load_image_file(f"{UNKNOWN_DIR}/{name}")
    image = cv2.resize(image, (960, 540))
    locations = face_recognition.face_locations(image,model=model)
    encoding=face_recognition.face_encodings(image,locations)
    image=cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
    print(locations)
    for encod,loc in zip(encoding,locations):
        results = face_recognition.compare_faces(known_faces,encod,TOLERANCE)
        print(results)
```

```
        match = None
        if True in results:
            match = known_names[results.index(True)]
            print(match)
            top_left = (loc[3],loc[0])
            bottom_right = (loc[1],loc[2])
            color=[0,255,0]
            cv2.rectangle(image, top_left,bottom_right,color,FRAME_THICK)
            top_left = (loc[3]+10,loc[2]+10)
            bottom_right = (loc[1]+15,loc[2]+30)
            color=[0,255,0]
            cv2.rectangle(image, top_left,bottom_right,color)
            cv2.putText(image,match,(loc[3]+15,loc[2]+25),
                    cv2.FONT_HERSHEY_SIMPLEX,0.5,(200,200,200),FONT_THICK)
    cv2.imshow("images",image)
    cv2.waitKey(0)
```

c. video_recognise.py

```
import face_recognition
import os
import cv2
import pickle
TOLERANCE = 0.5
FRAME_THICK = 1
FONT_THICK = 2
```

```
with open('faces.pickle', 'rb') as handle:
    known_faces = pickle.load(handle)
with open('names.pickle', 'rb') as handle:
    known_names = pickle.load(handle)
video = cv2.VideoCapture(0)
model = "cnn"
b=True
while b:
    ret,image = video.read()
    locations = face_recognition.face_locations(image,model=model)
    encoding=face_recognition.face_encodings(image,locations)
    for encod,loc in zip(encoding,locations):
        results = face_recognition.compare_faces(known_faces,encod,TOLERANCE)
        print(results)
        match = None
        if True in results:
            match = known_names[results.index(True)]
            top_left = (loc[3],loc[0])
            bottom_right = (loc[1],loc[2])
            color=[0,255,0]

            cv2.rectangle(image, top_left,bottom_right,color,FRAME_THICK)

            top_left = (loc[3]+10,loc[2]+10)
            bottom_right = (loc[1]+15,loc[2]+30)
            color=[0,255,0]

            cv2.rectangle(image, top_left,bottom_right,color)
            cv2.putText(image,match,(loc[3]+15,loc[2]+25),
```

```
                cv2.FONT_HERSHEY_SIMPLEX,0.5,(200,200,200),FONT_THICK)

        cv2.imshow("images",image)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            b=False
            break
##        cv2.waitKey(0)
```