Big Data Programming

Spring 2019

Yang Li

Fangsheng Wu

Zhenghong He

Real-Time Face Recognition

1. **Introduction**

We are experiencing the fast moving world. Now a day the usage of manpower has been reduced by the digitalized systems. Thus, day to day needs of the human is changing in to machine based digital system. These changes make easier the activities for us and reduce our mistakes. Firstly, we need to know the definition of Face recognition. Recognition is the identifying a known face with a known name in digital images, regardless of the source. The resource can be a scanned copy of an image to a live video stream. Face recognition is a section of Machine learning with a good number of research topics focused on improving the existing algorithms. This application has many features which have many advantages of recognition in real time. Using a programmable system reduces the time and cost. It is an automated or semi-automated process of matching face images. The task of Face recognition is identifying an already detected object as a known or unknown face, and telling exactly who face it is and using for a database of faces in order to validate this input face.

1. **Source Code**

There are two parts of our code:

* Webcam\_cv3.py
* BDP\_Model.py

**This is the webcam\_cv3.py:**

1. import cv2  
   import sys  
   import logging as log  
   import datetime as dt  
   from time import sleep  
   import numpy as numP  
   from decimal import Decimal  
   from PIL import Image  
   import glob  
   import xlrd  
   import pandas as pd  
   from pandas import ExcelWriter  
   from pandas import ExcelFile  
   from numpy.linalg import inv  
   import gc  
   import random  
   import math  
   from decimal import Decimal  
   cascPath = "haarcascade\_frontalface\_default.xml"  
   faceCascade = cv2.CascadeClassifier(cascPath)  
   log.basicConfig(filename='webcam.log',level=log.INFO)  
   numP.seterr(all='ignore')  
   flat\_arr\_images\_X\_matrix=[]#all images in a folder for trainning X  
   flat\_arr\_images\_Y1\_matrix=[]#all images in a folder for trainning predict Y  
   n=0#iteration of scans  
   tData = pd.read\_excel(r"train.xlsx", sheet\_name='Sheet1')  
   print("Column headings:", tData.columns)  
   for filename in glob.glob(r"train\\*.PNG"):  
    im=Image.open(filename).resize((64,64)).convert('RGBA')#resize to smallest due memroy and matrix issue bestfit 64，64  
    print(filename)  
    flat\_arr\_images\_X\_matrix.append(numP.array(im).ravel())# here we can add bias,   
    flat\_arr\_images\_Y1\_row = []#read row in array struc  
    flat\_arr\_images\_Y1\_row.append(tData[tData.columns[0]][n])  
    flat\_arr\_images\_Y1\_matrix.append(flat\_arr\_images\_Y1\_row)  
    n=n+1  
   flat\_arr\_images\_X\_matrix = numP.matrix(flat\_arr\_images\_X\_matrix,dtype='float64')  
   flat\_arr\_images\_Y1\_matrix = numP.matrix(flat\_arr\_images\_Y1\_matrix,dtype='float64')  
   Transform\_flat\_arr\_images\_X\_matrix = flat\_arr\_images\_X\_matrix.T  
   dot\_flat\_arr\_images\_XandY\_matrix= Transform\_flat\_arr\_images\_X\_matrix \* flat\_arr\_images\_X\_matrix  
   for i in range(dot\_flat\_arr\_images\_XandY\_matrix[0].size):  
    for j in range(dot\_flat\_arr\_images\_XandY\_matrix[0].size):  
    dot\_flat\_arr\_images\_XandY\_matrix[i,j] =dot\_flat\_arr\_images\_XandY\_matrix[i,j]+random.uniform(0, 1)#add w  
   inverse\_dot\_flat\_arr\_images\_XandY\_matrix = dot\_flat\_arr\_images\_XandY\_matrix .I   
   beta\_Training\_NN\_Xt = inverse\_dot\_flat\_arr\_images\_XandY\_matrix \* Transform\_flat\_arr\_images\_X\_matrix  
   beta\_Training1 = beta\_Training\_NN\_Xt \* flat\_arr\_images\_Y1\_matrix   
   print("Beta1:",beta\_Training1,beta\_Training1.shape)  
     
   video\_capture = cv2.VideoCapture(0)  
   anterior = 0  
   while True:  
    if not video\_capture.isOpened():  
    print('Unable to load camera.')  
    sleep(5)  
    pass  
    img\_counter = 0  
    # Capture frame-by-frame  
    ret, frame = video\_capture.read()  
     
    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
     
    faces = faceCascade.detectMultiScale(  
    gray,  
    scaleFactor=1.1,  
    minNeighbors=5,  
    minSize=(128, 188)  
    )  
     
     
    # Draw a rectangle around the faces  
    for (x, y, w, h) in faces:  
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)  
     
     
    if anterior != len(faces):  
    anterior = len(faces)  
    log.info("faces: "+str(len(faces))+" at "+str(dt.datetime.now()))  
    img\_name = "opencv\_frame\_{}.png".format(img\_counter)  
    cv2.imwrite(img\_name, frame)  
   ## print("x:",x,"\ty:",y,"\tx+w:",x+w,"\ty+h:",y+h)  
    fp = Image.open(img\_name)  
    cropped = fp.crop((x,y,x+w,y+h))  
    cropped.save("cropped.png")  
   ## cropped.show()  
   ## print("{} written!".format(img\_name))  
    img\_counter += 1  
    numP.seterr(all='ignore')  
    real\_time\_image = [] # all images in a folder for trainning X  
    im = Image.open("cropped.png").resize((64, 64)).convert('RGBA') # resize to smallest due memroy and matrix issue bestfit 64,64  
    real\_time\_image.append(numP.array(im).ravel()) # here we can add bias,  
    real\_time\_image = numP.matrix(real\_time\_image, dtype='float64')   
    print(real\_time\_image)  
    predict\_y = real\_time\_image\* beta\_Training1  
    print("predict\_value:",predict\_y,"\n")  
    if predict\_y[0][0] > 0.3 and predict\_y[0][0] < 3.3 :  
    print("Aceess Granted, face matched.\n")  
    break  
     
      
    # Display the resulting frame  
    cv2.imshow('Video', frame)  
     
     
    if cv2.waitKey(1) & 0xFF == ord('q'):  
    break  
     
    # Display the resulting frame  
    cv2.imshow('Video', frame)  
     
   # When everything is done, release the capture  
   video\_capture.release()  
   cv2.destroyAllWindows()

**This is BDP\_Model.py:**

import numpy as numP  
from decimal import Decimal  
from PIL import Image  
import glob  
import xlrd  
import pandas as pd  
from pandas import ExcelWriter  
from pandas import ExcelFile  
from numpy.linalg import inv  
import gc  
import random  
import math  
def main():  
 numP.seterr(all='ignore')  
 flat\_arr\_images\_X\_matrix=[]#all images in a folder for trainning X  
 flat\_arr\_images\_Y1\_matrix=[]#all images in a folder for trainning predict Y  
 flat\_arr\_images\_Y2\_matrix=[]#all images in a folder for trainning predict Y  
 flat\_arr\_images\_XY\_matrix = []  
 n=0#iteration of scans  
 tData = pd.read\_excel(r"train.xlsx", sheet\_name='Sheet1')  
 print("Column headings:", tData.columns)  
 for filename in glob.glob(r"train\\*.PNG"):  
 im=Image.open(filename).resize((64,64)).convert('RGBA')#resize to smallest due memroy and matrix issue bestfit 64，64  
 flat\_arr\_images\_X\_matrix.append(numP.array(im).ravel())# here we can add bias,   
 flat\_arr\_images\_Y1\_row = []#read row in array struc  
 flat\_arr\_images\_Y2\_row = []#read row in array struc  
 flat\_arr\_images\_Y1\_row.append(tData[tData.columns[0]][n])  
 flat\_arr\_images\_Y1\_matrix.append(flat\_arr\_images\_Y1\_row)  
 n=n+1  
 flat\_arr\_images\_X\_matrix = numP.matrix(flat\_arr\_images\_X\_matrix,dtype='float64')  
 x\_rowSize = flat\_arr\_images\_X\_matrix[0].size  
 flat\_arr\_images\_Y1\_matrix = numP.matrix(flat\_arr\_images\_Y1\_matrix,dtype='float64')  
 Transform\_flat\_arr\_images\_X\_matrix = flat\_arr\_images\_X\_matrix.T  
 dot\_flat\_arr\_images\_XandY\_matrix= Transform\_flat\_arr\_images\_X\_matrix \* flat\_arr\_images\_X\_matrix  
 for i in range(dot\_flat\_arr\_images\_XandY\_matrix[0].size):  
 for j in range(dot\_flat\_arr\_images\_XandY\_matrix[0].size):  
 dot\_flat\_arr\_images\_XandY\_matrix[i,j] =dot\_flat\_arr\_images\_XandY\_matrix[i,j]+random.uniform(0, 1)#add w  
 inverse\_dot\_flat\_arr\_images\_XandY\_matrix = dot\_flat\_arr\_images\_XandY\_matrix .I   
 beta\_Training\_NN\_Xt = inverse\_dot\_flat\_arr\_images\_XandY\_matrix \* Transform\_flat\_arr\_images\_X\_matrix  
 beta\_Training1 = beta\_Training\_NN\_Xt \* flat\_arr\_images\_Y1\_matrix   
## beta\_Training2 = beta\_Training\_NN\_Xt \* flat\_arr\_images\_Y2\_matrix  
 print("Beta1:",beta\_Training1,beta\_Training1.shape)#,"\nBeta2:")#,beta\_Training2,beta\_Training2.shape,'\n')  
# getting trainning model then to validate   
 test\_image\_X\_matrix=[]  
 test\_image\_Y1\_matrix=[]  
 test\_image\_Y2\_matrix=[]  
 n=0  
 test\_Data = pd.read\_excel(r"\*\BDPFinal\train.xlsx", sheet\_name='test')  
 for filename in glob.glob(r"\*\BDPFinal\test\t2.\*"):  
 im=Image.open(filename).resize((64,64)).convert('RGBA') #resize to smallest due memroy and matrix issue  
 test\_image\_X\_matrix.append(numP.array(im).ravel())# here we can add bias,  
 test\_image\_Y1\_row = []#read row in array struc  
 test\_image\_Y1\_row.append(test\_Data[test\_Data.columns[0]][n])  
 test\_image\_Y1\_matrix.append(test\_image\_Y1\_row)  
 n=n+1  
 test\_image\_X\_matrix = numP.matrix(test\_image\_X\_matrix,dtype='float64')  
 test\_image\_Y1\_matrix = numP.matrix(test\_image\_Y1\_matrix,dtype='float64')  
 print("train\_image\_X for\_Sheng:",test\_image\_X\_matrix,test\_image\_X\_matrix.shape,'\n')  
 predict\_image\_Y1\_matrix = test\_image\_X\_matrix \* beta\_Training1  
 print("predict\_model\_Y for\_Sheng:",predict\_image\_Y1\_matrix )#, ",\tpredict\_value\_Y2 tobramycin resistance:",predict\_image\_Y2\_matrix ,'\n')  
 print("actual\_image\_value\_Sheng:",test\_image\_Y1\_matrix[0,0])#,",\tactual\_value\_Y2 tobramycin resistance:",test\_image\_Y2\_matrix[0,0],'\n')   
 print("accuracy :", ((predict\_image\_Y1\_matrix-test\_image\_Y1\_matrix[0,0])/test\_image\_Y1\_matrix[0,0])\*100,"%,")#,  
main()

1. **Any Output Files**

There is only one output file which is the screenshot image. However, it is not output. It is what we used for (face recognition)prediction.

1. **Write Up**

There are three main parts of this project:

1. Face Detection and Data Gathering

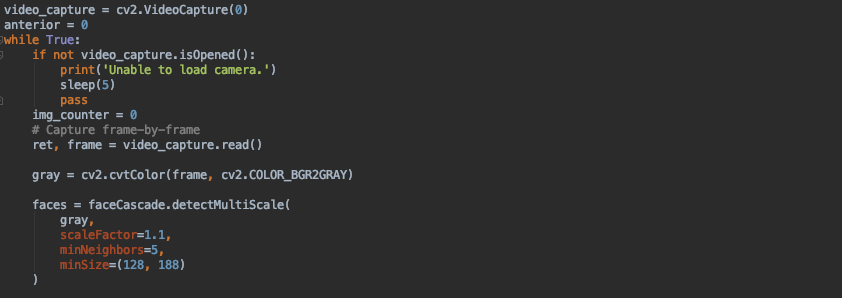
2. Train and Recognizer

3. Face Recognition

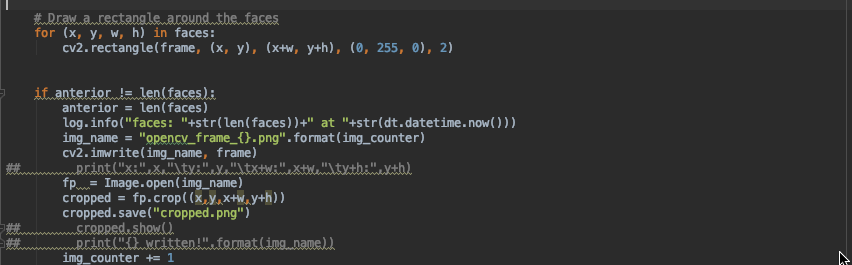
**What does the code do?**

1. Face Detection and Data Gathering

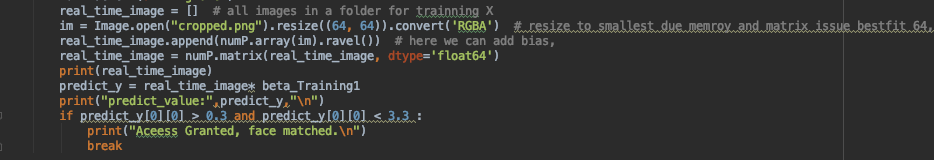
The first part of our mission is to do face detection.



This part of the code (Please see screenshot above) uses openCV2 to open the camera , and then it starts streaming. OpenCV is a python library that is focusing on real-time computer vision; Intel originally developed it. We used the "HaarCascade classifier" to detect a face; it is an effective and popular detection method. The minimum size of the scale is (128,188).



For face detection, we decided to draw a rectangle around the face. It is easier for us when we implement the face recognition part. We will use linear regression to compare the trained model with the streaming face; therefore, we will need to capture pictures. Since we have already drawn a rectangle around the face, thus we can crop the rectangle part of the image. And then, it will be saved in the local directory; later it will be used for face recognition (remark: since it is a real-time face recognition application; therefore, the program will keep taking pictures and replace the previous image). The below codes are crop images and run our model.

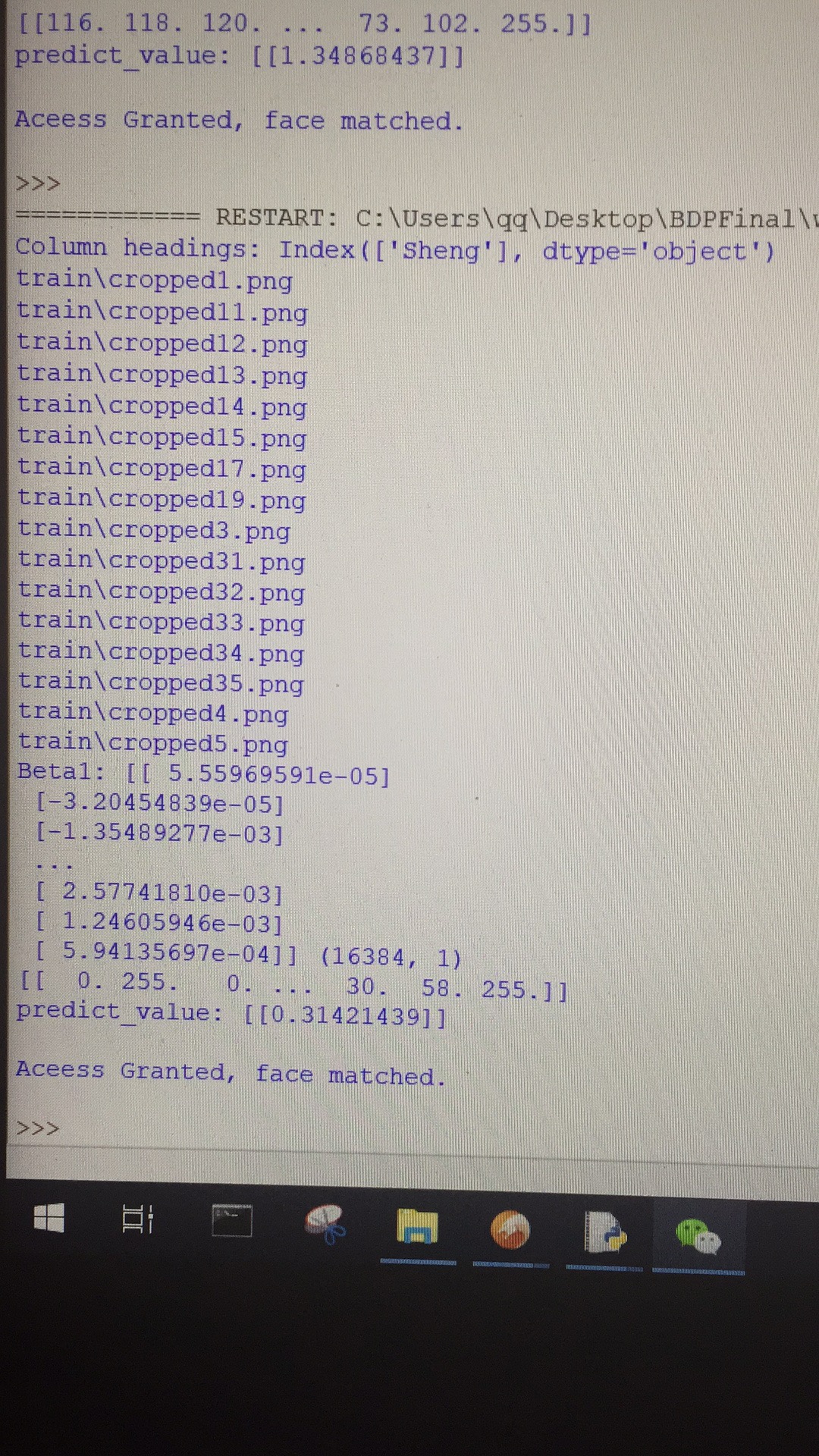


2. Train and Recognizer

So, initially I fed my own picture with various facial expression and its corresponding

label. By this, we can obtain a model simply by implement Linear regression, Logistic regression, and RBM. With the label and training facial images, a model Beta we end up with that is a most informative and representative feature variable this serves the purpose of real timing recognition step.

3. Face Recognition



For the process of previous, we were able to find the training model; here as, a step to validate my model with the real time image. This happens to the credit of webcam and FrontAIFace model where a predefined algorithm is used to detect human face in the camera. When is true, we take a frame shoot, store it local and then crop it to the size that only displays the actual face all the other environment in the background we can consider it as noise or trivial pixels. Then, a cropped facial image would be taken to validate our training model. Now we have a successful real time facial recognition model.

Analysis of the model you used:

Linear regression, logistic regression, and RBM three of we used, of all the attempts, only the linear regression has a decent accuracy and comparatively short training time.

How could this be improved:

I think RBM, Deep belief Network, ANN would be a very good breaking point to processing image related problem. Instead of training among all the images, extract feature from one after one would give better result.

There are many improvements we can make:

First, we only trained one person's pictures. Therefore it only can recognize one person's face. We can improve it by change linear regression to other machine learning or deep learning model (like Support Vector Machine), so face recognition function is more practical.

Second, we still can increase the prediction accuracy by training more pictures and take more picture to compare with the model. Last, there are many other better machine learning and deep learning models that can be used for face recognition; we can test all of them and find the best model that gives us the highest accuracy and shortest runtime.