NEURAL NETWORKS AND FUZZY CONTROL (ECE3009)

PROJECT REPORT

Facial Recognition and Verification Using Neural Networks

Submitted by,

Shree Hari M Nair - 16BEC0157
Adheik Dominic – 16BEC0620
Anuvind Ashok – 16BEC0790

PROJECT SUPERVISOR

Prof. Shankar Ganesh

Fall Semester 2019-20

VIT UNIVERSITY

This is to certify that the project work entitled “Facial Recognition” that is being submitted by Shree Hari M Nair, Adheik Dominic, Anuvind Ashok for NEURAL NETWORK AND FUZZY CONTROL is a record of bonafide work done under my supervision.

Date:

Signature of Faculty:

Prof. Shankar Ganesh.

ACKNOWLEDGEMENT

We would like to acknowledge thanks to our faculty, **Prof. Shankar Ganesh.**, for providing us with the opportunity of preparing this project report for the subject of Neural Network and Fuzzy Control (ECE3009). The project gave us a combined knowledge of working with neural networks and the programming for image processing.

We would be grateful to do any such similar projects in the future too.

ABSTRACT

There are two main kinds of face recognition tasks: face identification (who is who in a probe face set, given a gallery face set) and face verification (same or not, given two faces).The goal in this project is to investigate various CNN architectures and focus on the verification part. We propose to train the system using an initial dataset that obviously distinguishes on the basis of the various descriptors we're to use here. The outcome is a much simpler and yet effective network architecture using CNN for facial verification.

We aim to study various architectures of CNN and modify and compile themto create a suitable network for our objective and database

INTRODUCTION

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analysing patterns based on the person's facial textures and shape. There are two main kinds of face recognition tasks: face identification (who is who in a probe face set, given a gallery face set) and face verification (same or not, given two faces). As mentioned above, we focus on the verification part in this project.

Tools Required: Python IDLE 3.0 and a working Web-Cam.

Python IDLE 3.0:

IDLE (short for integrated development environment[1][2] or integrated development and learning environment[3]) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1.[4][5] It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit.

IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter. In this project, we've used some particular python libraries for the purpose of image processing and data analysis.

What is a library?

Libraries in python are predefined functions that can be directly used by the user to perform certain tasks. The user can import the libraries from the memory as objects and use those objects to access the functionalities of the libraries.

Python libraries used:

OpenCV-

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel[2]). The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

NumPy-

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers.

PIL-

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

What is unsupervised learning?

Unsupervised learning is a branch of machine learning that learns from test data that has not been labelled, classified or categorised. Instead of responding to feedback, unsupervised learning identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. Alternatives include supervised learning and reinforcement learning.

OBJECTIVE

The objective of this project is to create a Face recognition system has the benefit of being a passive, non-intrusive system for verifying personal identity.

The project will use two types of neural network techniques:

1. Self-organising map neural network
2. Learning Based Descriptor

ALGORITHM

- A large number of data (in this case, images) is fed into the system at the first stage of input.
- The data is then characterized and grouped under various categories.
- The categories in this case are of similar looking images.
- A sample input is then given into the system.
- The system runs the analysis and then calculates the nearest set of data that represents the sample.

CODE and Working:

Creating Dataset-

```
import cv2
vid_cam = cv2.VideoCapture(0)                                                    #activate webcam
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')     #create object
face_id = 1                                                                      #id for first sample
count = 0
while(True):                                                                     #while loop starts
   image_frame = vid_cam.read()                                                  #input from the webcam
   gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)                          #converting to grey
   faces = face_detector.detectMultiScale(gray, 1.3, 5)                          #frame for saved image
   for (x,y,w,h) in faces:                                                       #for the frame around face
      cv2.rectangle(image_frame, (x,y), (x+w,y+h), (255,0,0), 2)
      count += 1
      cv2.imwrite("C:\pyth\python37\dataset\." + str(face_id) + '.' + str(count) + ".jpg",
gray[y:y+h,x:x+w])                                                               #save the image
      cv2.imshow('frame', image_frame)                                           #string name frame
   if cv2.waitKey(100) & 0xFF == ord('q'):                                       #to exit while running
      break
   elif count>10:                                                                #number of samples
      break                                                                      # Stop video
vid_cam.release()                                                                # Close all started windows
cv2.destroyAllWindows()
```

Learning from Dataset-

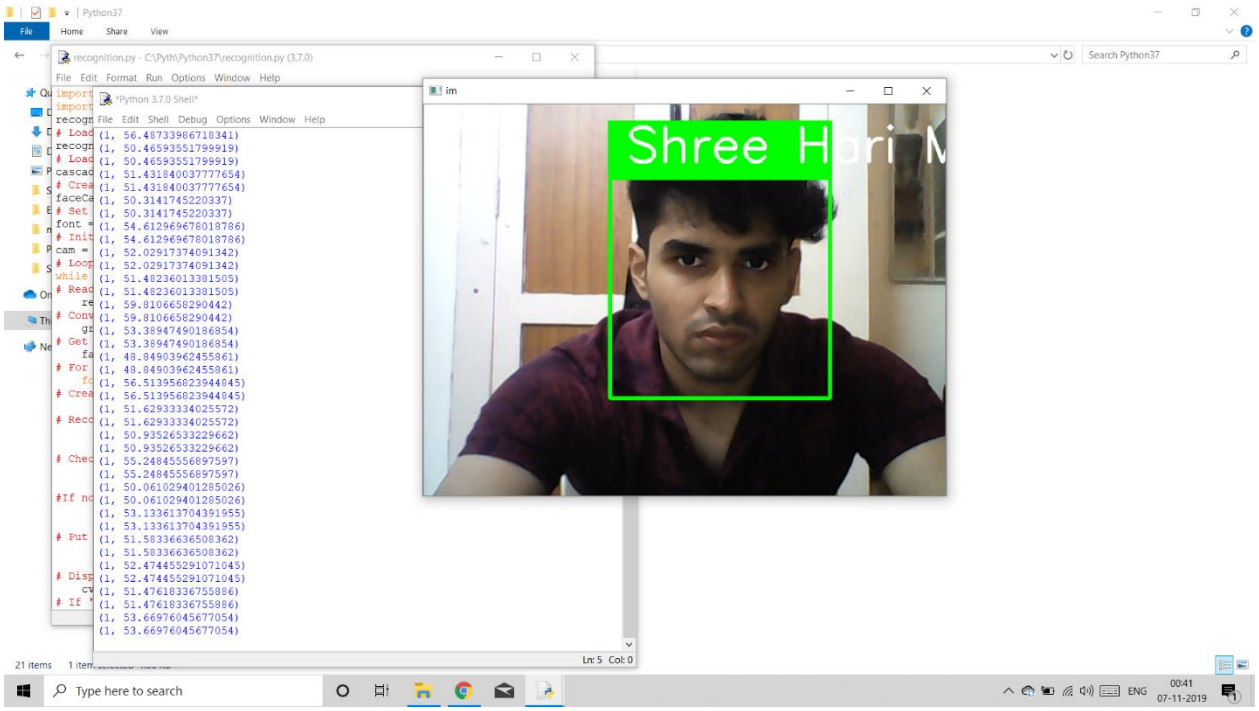
```
   import cv2,os                                                                 #import libraries
   import numpy as np                                                            #import NumPy
   from PIL import Image #import PIL
   recognizer = cv2.face.LBPHFaceRecognizer_create()  #creating a histogram and creating a function
   detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");      #detector for a face
   def getImagesAndLabels(path):                                                 #add face id and samples
      imagePaths = [os.path.join(path,f)
      for f in os.listdir(path)]
      faceSamples=[]
      ids = []
      for imagePath in imagePaths:
         PIL_img = Image.open(imagePath).convert('L')
         img_numpy = np.array(PIL_img,'uint8')
         id = int(os.path.split(imagePath)[-1].split(".")[1])
         print(id)
         faces = detector.detectMultiScale(img_numpy)
         for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
      return faceSamples,ids
   faces,ids = getImagesAndLabels("C:\pyth\python37\dataset\.")
   recognizer.train(faces, np.array(ids))
   recognizer.write('trainer/trainer.yml')recognizer.train(faces, np.array(ids))

   recognizer.write('trainer/trainer.yml')
```

Final Recognition-

```
import cv2
import numpy as np
recognizer = cv2.face.LBPHFaceRecognizer_create()                    # Load the trained mode
recognizer.read('trainer/trainer.yml')                               # Load prebuilt model for Frontal Face
cascadePath = "haarcascade_frontalface_default.xml"                  # Create classifier from prebuilt model
faceCascade = cv2.CascadeClassifier(cascadePath);                    # Set the font style
font = cv2.FONT_HERSHEY_SIMPLEX                                      # Initialize and start the video frame capture
cam = cv2.VideoCapture(0)                                            # Loop
while True:                                                          # Read the video frame
    ret, im =cam.read()                                          # Convert the captured frame into grayscale
    gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)                       # Get all face from the video frame
    faces = faceCascade.detectMultiScale(gray, 1.2,5)                # For each face in faces
    for(x,y,w,h) in faces:                                           # Create rectangle around the face
        cv2.rectangle(im, (x-20,y-20), (x+w+20,y+h+20), (0,255,0), 4)# Recognize the face belongs to which ID
        Id = recognizer.predict(gray[y:y+h,x:x+w])
        print(Id)                                                    # Check the ID if exist
        if(Id[0] == 1):
            Id = "Shree Hari M Nair"                                 #If not exist, then it is Unknown
        else:
            Id= "not recognized"                                     # Put text describe who is in the picture
        cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)
        cv2.putText(im, str(Id), (x,y-40), font, 2, (255,255,255), 3)# Display the video frame with the bounded rectangle
    cv2.imshow('im',im)                                              # If 'q' is pressed, close program
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break                                                        # Stop the camera
cam.release()                                                        # Close all windows
cv2.destroyAllWindows()
```

OUTPUT and RESULTS:



DATASETS

Number of Epochs and Accuracy Matrix:

Number of epochs	Number of trials	Successful verification	Accuracy
10	25	16	65
25	25	18	72
50	25	21	84
100	25	23	92

INFERENCE:

The number of images being taken in the dataset in the current code= 10.

This is also the number of epochs while the program is trained.

The accuracy of the program increases with respect to the number of epochs (i.i. the number of images in each sample)

REFERENCES

1) Zhimin Cao, Qi Yin, Xiaoou Tang, Jian Sun; “Face Recognition with Learning-based Descriptor”,978-1-4244-6985-7/10/$26.00 ©2010 IEEE.

2) Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, “ Deep Face Recognition”, Visual Geometry Group, Department of Engineering Science University of Oxford

3) Steve Lawrence, Member, IEEE, C. Lee Giles, Senior Member, IEEE, Ah Chung Tsoi, Senior Member, IEEE, Andrew D. Back, Member, IEEE; “Face Recognition: A Convolutional Neural-Network Approach”; IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 8, NO. 1, JANUARY 1997

Source for the Dataset:

Vadim Pisarevsky;“haarcascade_frontalface_default.xml”;https://github.com/vpisarev