MINI PROJECT REPORT

On

Hand Gesture Recognition

Submitted by

Harshit Rai(171500127)
Keshavi Aggarwal(171500160)
Vanshika Shivani(171500373)

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA

Summary of the Project Work

The project entitled Hand Gesture Recognition was completed successfully. The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to study the Convolution Neural Network and with the help of that identify different fashion items .

This project helped us in gaining valuable information and practical knowledge on several topics like creating CNN models and Deep Learning using Python, Tensorflow & Keras, usage of functions of Keras module . The entire code is error free. Also the project helped us understanding about the development phases of a project and software development life cycle. We learned how to test different features of a project.

This project has given us great satisfaction in having designed an model which can be implemented to anyone by simple modifications. There is a scope for further development in our project to a great extent. A number of features can be added to this model in future like providing face detection and other functionality.

ACKNOWLEDGEMENT

The project work in this report is an outcome of continuous work over a period and drew intellectual support from various sources. I would like to articulate our profound gratitude and to all those people who extended their wholehearted co-operation and have helped me in completing this project successfully.

I am thankful to Mr. Sharad Gupta for teaching and assisting me in making the project successful. I would also like to thank our parents & other fellow mates for guiding and encouraging me throughout the duration of the project.

Harshit Rai(171500127)
Keshavi Aggarwal(171500160)
Vanshika Shivani(171500373)

Department of Computer Engineering and Applications
GLA University, Mathura
17 km Stone NH#2, Mathura-Delhi Road, P.O. Chaumuha
Mathura, 281406

DECLARATION

I hereby declare that the project work entitled "Hand Gesture Recognition" submitted to the GLA University Mathura, is a record of an original work done by me under the guidance of Mr.Mandeep Singh .

Signature of Candidate:

Name of candidate: Harshit Rai(171500127)
Keshavi Aggarwal(171500160)
Vanshika Shivani(171500373)

Course: Computer Science and Engineering

Year: III

Semester: V

ABSTRACT

Mini Project is the requirement for all engineering students in order to complete their Bachelor of Engineering degree at the GLA University, Mathura. Mini Project is a very important programme, since it complements both the academic and professional aspects of the engineering education . Exposing the students to the practical experience and actual working environment shall open the avenues for developing their skills and capabilities, as well as enhancing their intellectual and emotional persona. The Mini Project also can provide strong linkages between university-industries that shall pave opportunities for "smart partnerships" and industrially driven research. The outcomes of the EIT that are mainly based on the assessment covering the company's and university's evaluation will provide the feedback for student's performance after 75% completion of their engineering study. The remarks from the companies on the students will very much helpful for the university to have a continuous quality improvement especially on curriculum practiced.

Table of Content

Introduction To Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

A formal definition of deep learning is- neurons ,In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours.

The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

Installation of Jupyter Notebook on window

Instructions tested with Windows 10 64-bit and Continuum's Anaconda 5.20

Install GNU on Windows

1.Download and install GNU on Windows (Gow) from

https://github.com/bmatzelle/gow/releases/download/v0.8.0/Gow-0.8.0.exe. select The default options when prompted during the installation of Gow.

Install Anaconda and Jupyter Notebook

1.Downloads and install Anaconda from

https://repo.anaconda.com/archive/Anaconda3-2019.07-Windows-x86_64.exe.

Select the default options when prompted during the installation of Anaconda

1. Open "Anaconda Propt" by finding it in the window (start) menu.

2. Type the command in red to verified Anaconda was installed.

> python --version

Python 3.7.3

3. Type the command in red to update Anaconda

> conda update --all --yes

Start Jupyter Notebook

1. Type the command in red to start jupyter

Notebook > jupyter notebook

System and tools

Operating System

The programming work was carried out on one computer which ran the window 10 system.
The model was tested on two computers which ran window 10 and Ubuntu.

Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
Python is a programming language that lets you work quickly and integrate systems more efficiently.
There are two major Python versions- Python 2 and Python 3. Both are quite different.
Beginning with Python programming:

1) Finding an Interpreter:
Before we start Python programming, we need to have an interpreter to interpret and run our programs. There are certain online interpreters like https://ide.geeksforgeeks.org/, http://ideone.com/ or http://codepad.org/ that can be used to start Python without installing an interpreter.
Windows:There are many interpreters available freely to run Python scripts like IDLE (Integrated Development Environment) which is installed when you install the python software from.

Jupyter Notebook

The Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include: - Live code - Interactive widgets - Plots - Narrative text - Equations - Images - Video

These documents provide a complete and self-contained record of a computation that can be converted to various formats and shared with others using email, Dropbox, version control systems (like git/GitHub) or nbviewer.jupyter.org.

Keras Module

What is Keras?

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer.

Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

What is Backend?

Backend is a term in Keras that performs all low-level computation such as tensor products, convolutions and many other things with the help of other libraries such as Tensorflow or Theano. So, the "backend engine" will perform the computation and development of the models. Tensorflow is the default "backend engine" but we can change it in the configuration.

Theano is an open source project that was developed by the MILA group at the University of Montreal, Quebec, Canada. It was the first widely used Framework. It is a Python library that helps in multi-dimensional arrays for mathematical operations using Numpy or Scipy. Theano can use GPUs for faster computation, it also can automatically build symbolic graphs for computing gradients. On its website, Theano claims that it can recognize numerically unstable expressions and compute them with more stable algorithms, this is very useful for our unstable expressions.

On the other hand, Tensorflow is the rising star in deep learning framework. Developed by Google's Brain team it is the most popular deep learning tool. With a lot of features, and researchers contribute to help develop this framework for deep learning purposes.

Advantages of Keras

Fast Deployment and Easy to understand

Keras is very quick to make a network model. If you want to make a simple network model with a few lines, Keras can help you with that. Look at the example below:

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(64, activation='relu', input_dim=50)) #input shape of 50
model.add(Dense(28, activation='relu')) #input shape of 50
model.add(Dense(10, activation='softmax'))
```

Because of friendly the API, we can easily understand the process. Writing the code with a simple function and no need to set multiple parameters.

Large Community Support:

There are lots of AI communities that use Keras for their Deep Learning framework. Many of them publish their codes as well tutorial to the general public.

Have multiple Backends

You can choose Tensorflow, CNTK, and Theano as your backend with Keras. You can choose a different backend for different projects depending on your needs. Each backend has its own unique advantage.

Cross-Platform and Easy Model Deployment

With a variety of supported devices and platforms, you can deploy Keras on any device

like iOS with CoreML

Android with Tensorflow Android,

Web browser with .js support

Cloud engine

Raspberry Pi

Multi GPUs Support

You can train Keras with on a single GPU or use multiple GPUs at once. Because Keras has a built-in support for data parallelism so it can process large volumes of data and speed up the time needed to train it.

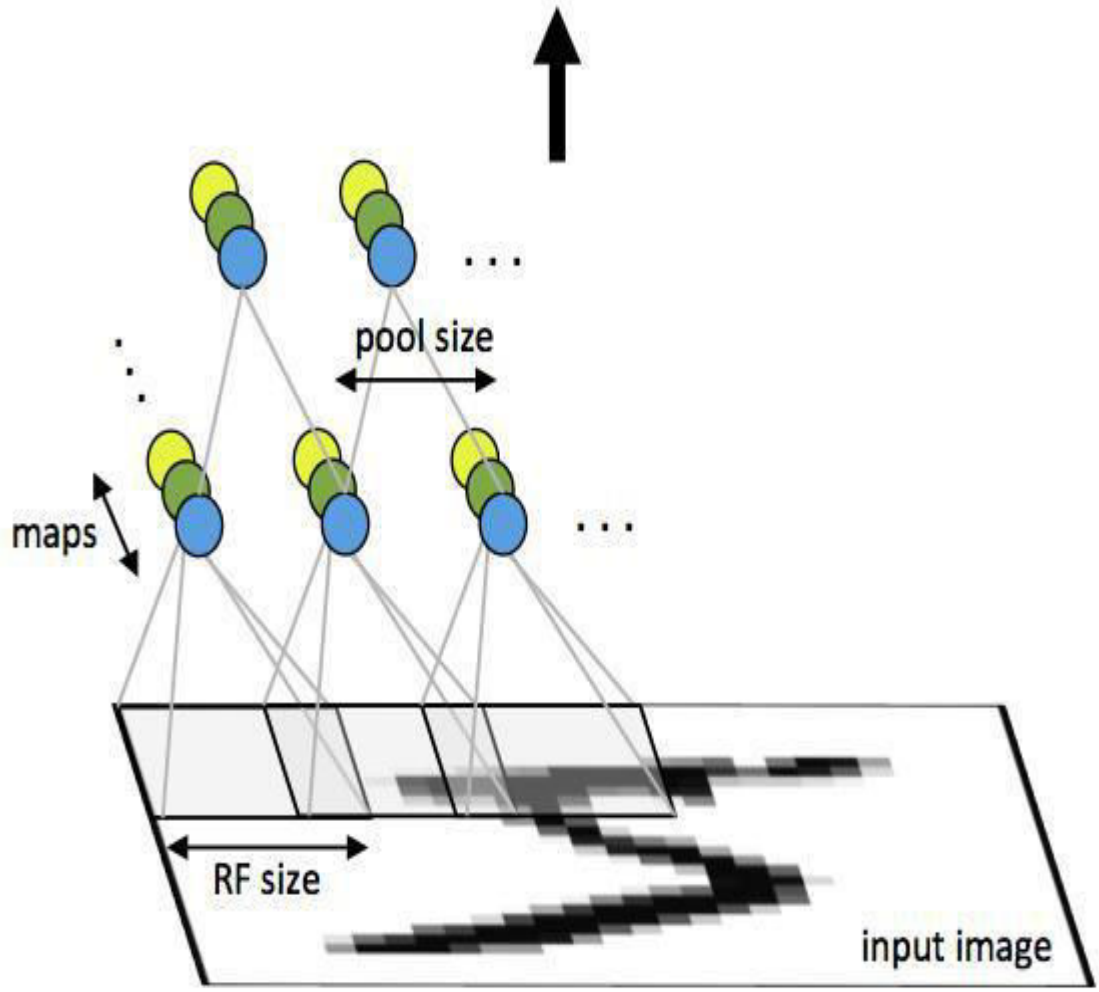
Introduction to Convolutional neural network

Overview

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article we will discuss the architecture of a CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based optimization. See the respective tutorials on convolution and pooling for more details on those specific operations.

Architecture

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r=3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m−n+1. Each map is then subsampled typically with mean or max pooling over p x p contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map. The figure below illustrates a full layer in a CNN consisting of convolutional and subsampling sublayers. Units of the same color have tied weights.



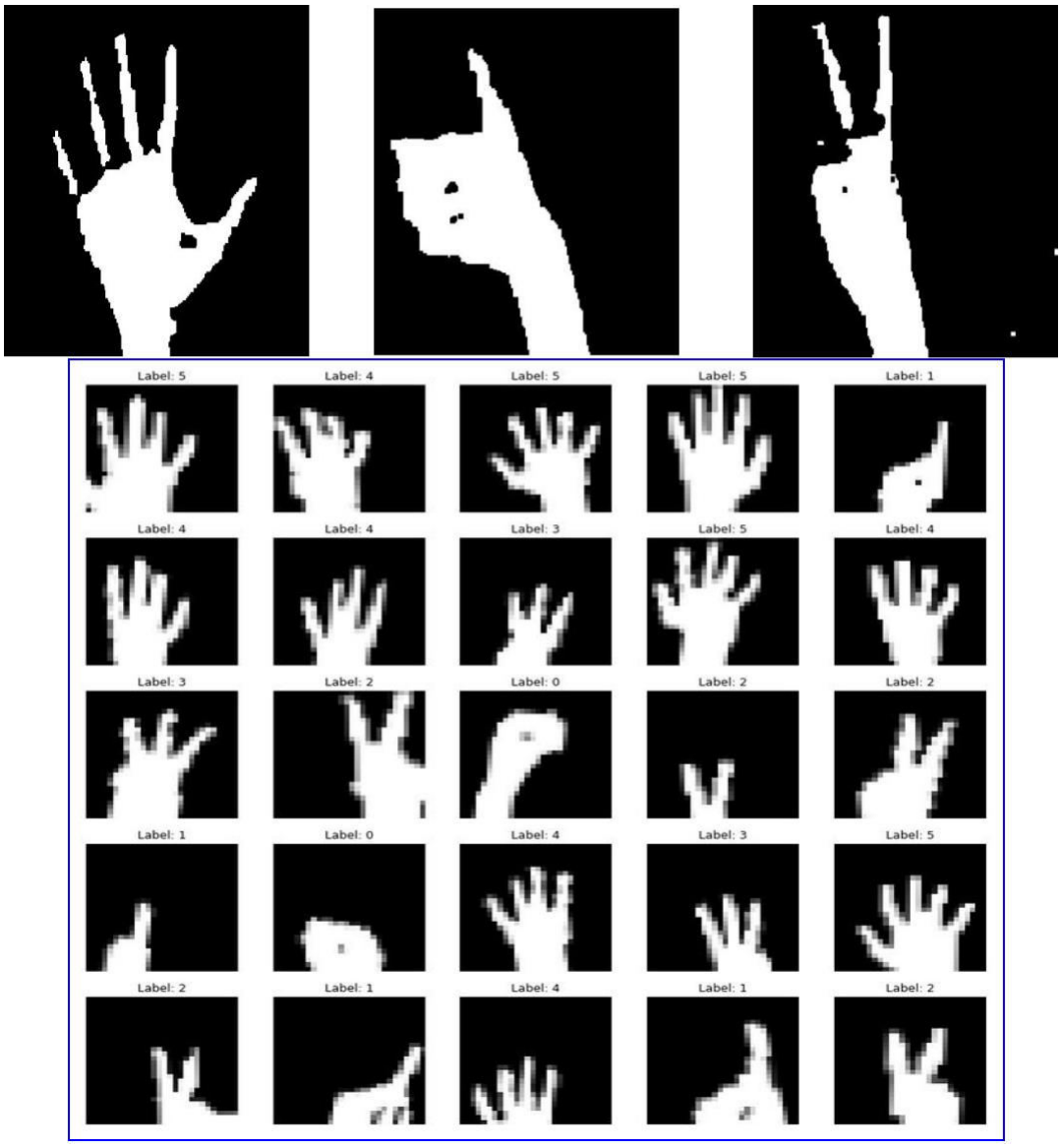
First layer of a convolutional neural network with pooling. Units of the same color have tied weights and units of different color represent different filter maps.

After the convolutional layers there may be any number of fully connected layers. The densely connected layers are identical to the layers in a standard multilayer neural network.

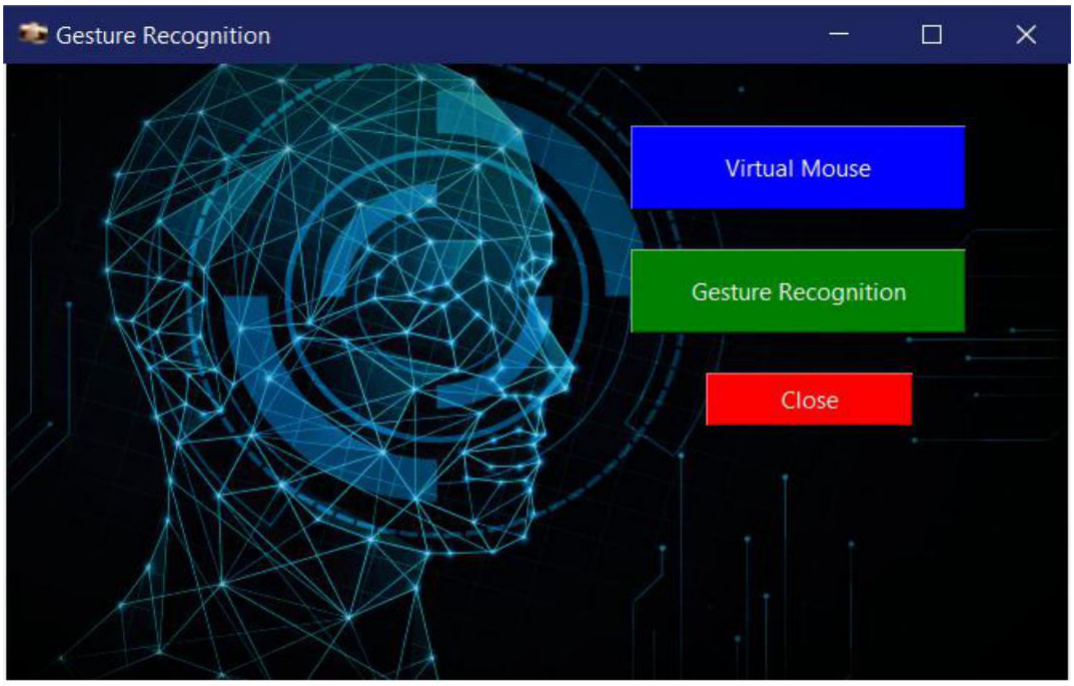
Introduction To Hand Gesture Dataset

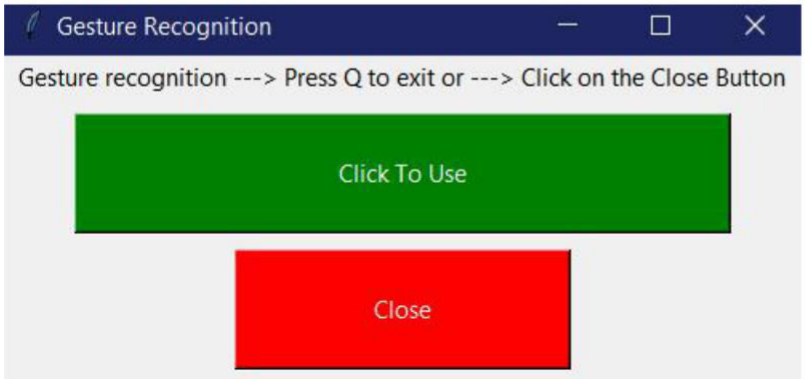
Hand gesture Dataset whic is used in this project is created by me and images—consisting of a training set of 1800 examples. Each example is a 50x50 grayscale image, associated with a label from 5 classes. I used this dataset to serve as a direct drop-in replacement for the original traditional dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

Here's an example how the data looks (each class takes three-rows):



Graphical User Interface of Project





Project Description of Hand Gesture Recognition

D: > Projects_Code > Hand_gesture_recognition > gui_main.py > ...

```
from tkinter import Button,Canvas
from PIL import ImageTk,Image
from vir_mouse import main_virtual_mouse
import gesture
import tkinter

main_window = tkinter.Tk()
main_window.geometry("670x400")

my_img = ImageTk.PhotoImage(Image.open("D:\\Projects_Code\\Hand_gesture_recognition\\background1.png"))


my_label = tkinter.Label(image=my_img)
my_label.pack()


main_window.iconbitmap("D:\\Projects_Code\\Hand_gesture_recognition\\icon.ico")
main_window.title("Gesture Recognition")


def virtual_mouse(event):
    def open(event):
        main_virtual_mouse()

    vir_mouse_window = tkinter.Tk()
    vir_mouse_window.geometry("500x200")
    vir_mouse_window.title("Virtual Mouse")
    tkinter.Label(vir_mouse_window,text="virtual mouse ---> Press Q to exit or --> Click on the Close Button").pack()



    b11 = tkinter.Button(vir_mouse_window,text="Click To Use",bg='blue',fg='white')
    b11.bind("<Button-1>",open)
    b11.config( height = 3, width = 50 )
    b11.pack(pady=10)
```

```
    b11 = tkinter.Button(vir_mouse_window,text="Click To Use",bg='blue',fg='white')
    b11.bind("<Button-1>",open)
    b11.config( height = 3, width = 50 )
    b11.pack(pady=10)


    bclose = tkinter.Button(vir_mouse_window,text="Close",command=vir_mouse_window.destroy,bg='red',fg='white')
    bclose.config(height=3,width=25)
    bclose.pack()


def gesture_recog_a(event):
    def open(event):
        gesture.gesture_recog()



    gesture_window = tkinter.Tk()
    gesture_window.geometry("500x200")
    gesture_window.title("Gesture Recognition")
    tkinter.Label(gesture_window,text="Gesture recognition ---> Press Q to exit or ---> Click on the Close Button").pack()
    b22 = tkinter.Button(gesture_window,text="Click To Use",bg='green',fg='white')
    b22.bind("<Button-1>",open)
    b22.config( height = 3, width = 50 )

    b22.pack(pady=10)

    bclose = tkinter.Button(gesture_window,text="Close",command=gesture_window.destroy,bg='red',fg='white')
    bclose.config(height=3,width=25)
    bclose.pack()


b1 = tkinter.Button(main_window,text="Virtual Mouse",bg='blue',fg='white')
b1.bind("<Button-1>",virtual_mouse)
b1.config( height = 2, width = 25 )
b1.pack(pady=10)
b1.place(relx = .9, rely = .1, anchor = 'ne')

b2 = tkinter.Button(main_window,text="Gesture Recognition",bg='green',fg='white')
b2.bind("<Button-1>",gesture_recog_a)
b2.config( height = 2, width = 25 )
b2.pack(pady=10)
b2.place(relx = .9, rely = .3, anchor = 'ne')


bclose_main = tkinter.Button(main_window,text="Close",command=main_window.quit,bg='red',fg='white')
bclose_main.config(height=1,width=15)
bclose_main.pack()
bclose_main.place(relx = .85, rely = .5, anchor = 'ne')

main_window.minsize(400,200)
main_window.mainloop()
```

D: > Projects_Code > Hand_gesture_recognition > gesture.py > ...

```python
import cv2
import numpy as np
import math


def gesture_recog():
    cap = cv2.VideoCapture(0)

    while(1):

        try:  #an error comes if it does not find anything in window as it cannot find contour of max area
          #therefore this try error statement

            ret, frame = cap.read()
            frame=cv2.flip(frame,1)
            kernel = np.ones((3,3),np.uint8)

            #define region of interest
            roi=frame[100:300, 100:300]


            cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)
            hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)



    # define range of skin color in HSV
            lower_skin = np.array([0,20,70], dtype=np.uint8)
            upper_skin = np.array([20,255,255], dtype=np.uint8)

     #extract skin colur imagw
            mask = cv2.inRange(hsv, lower_skin, upper_skin)



     #find contours
            contours,hierarchy= cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

    #find contour of max area(hand)
            cnt = max(contours, key = lambda x: cv2.contourArea(x))

     #approx the contour a little
            epsilon = 0.0005*cv2.arcLength(cnt,True)
            approx= cv2.approxPolyDP(cnt,epsilon,True)


     #make convex hull around hand
            hull = cv2.convexHull(cnt)

      #define area of hull and area of hand
            areahull = cv2.contourArea(hull)
            areacnt = cv2.contourArea(cnt)

     #find the percentage of area not covered by hand in convex hull
            arearatio=((areahull-areacnt)/areacnt)*100

      #find the defects in convex hull with respect to hand
            hull = cv2.convexHull(approx, returnPoints=False)
            defects = cv2.convexityDefects(approx, hull)

     # l = no. of defects
            l=0

     #code for finding no. of defects due to fingers
            for i in range(defects.shape[0]):
                s,e,f,d = defects[i,0]
                start = tuple(approx[s][0])
                end = tuple(approx[e][0])
                far = tuple(approx[f][0])
                pt= (100,180)
```

```
#code for finding no. of defects due to fingers
        for i in range(defects.shape[0]):
            s,e,f,d = defects[i,0]
            start = tuple(approx[s][0])
            end = tuple(approx[e][0])
            far = tuple(approx[f][0])
            pt= (100,180)


            # find length of all sides of triangle
            a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
            b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
            c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
            s = (a+b+c)/2
            ar = math.sqrt(s*(s-a)*(s-b)*(s-c))

            #distance between point and convex hull
            d=(2*ar)/a

            # apply cosine rule here
            angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57


            # ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)
            if angle <= 90 and d>30:
                l += 1
                cv2.circle(roi, far, 3, [255,0,0], -1)

            #draw lines around hand
            cv2.line(roi,start, end, [0,255,0], 2)


            l+=1
```

```
    #print corresponding gestures which are in their ranges
        font = cv2.FONT_HERSHEY_SIMPLEX
        if l==1:
            if areacnt<2000:
                cv2.putText(frame,'Put hand in the box',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
            else:
                if arearatio<12:
                    cv2.putText(frame,'0 or Close Hand',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                elif arearatio<17.5:
                    cv2.putText(frame,'Best of luck',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

                else:
                    cv2.putText(frame,'1 Finger',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        elif l==2:
            cv2.putText(frame,'2 Fingers or Peace ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        elif l==3:

            if arearatio<27:
                cv2.putText(frame,'3 Fingers',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
            else:
                cv2.putText(frame,'ok',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        elif l==4:
            cv2.putText(frame,'4 Fingers ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        elif l==5:
            cv2.putText(frame,'5 Finger or Palm',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        elif l==6:
            cv2.putText(frame,'reposition',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        else :
            cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
```

```
        #print corresponding gestures which are in their ranges
            font = cv2.FONT_HERSHEY_SIMPLEX
            if l==1:
                if areacnt<2000:
                    cv2.putText(frame,'Put hand in the box',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                else:
                    if arearatio<12:
                        cv2.putText(frame,'0 or Close Hand',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                    elif arearatio<17.5:
                        cv2.putText(frame,'Best of luck',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

                    else:
                        cv2.putText(frame,'1 Finger',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==2:
                cv2.putText(frame,'2 Fingers or Peace ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==3:

                if arearatio<27:
                    cv2.putText(frame,'3 Fingers',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                else:
                    cv2.putText(frame,'ok',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==4:
                cv2.putText(frame,'4 Fingers ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==5:
                cv2.putText(frame,'5 Finger or Palm',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==6:
                cv2.putText(frame,'reposition',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            else :
                cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            #show the windows
                cv2.imshow('mask',mask)

                cv2.imshow('frame',frame)
            except:
                pass


            if cv2.waitKey(1) == ord('q'):
                break

    cv2.destroyAllWindows()
    cap.release()
```

D: > Projects_Code > Hand_gesture_recognition > vir_mouse.py > ...

```python
import cv2
import numpy as np
from pynput.mouse import Button, Controller

import mouse_press
import keyboard_press
import wx


def main_virtual_mouse():

    mouse=Controller()
    #keybord = Controller()

    prey = 0
    prex = 0
    app=wx.App(False)
    (sx,sy)=wx.GetDisplaySize()
    (camx,camy)=(600,800)

    lowerBound=np.array([33,80,40])
    upperBound=np.array([102,255,255])

    cam= cv2.VideoCapture(0)

    kernelOpen=np.ones((5,5))
    kernelClose=np.ones((10,10))
    pinchFlag=0

    while True:
        ret, img=cam.read()
        img=cv2.resize(img,(300,400))

        #convert BGR to HSV
        imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
        # create the Mask
        mask=cv2.inRange(imgHSV,lowerBound,upperBound)
        #morphology
```

```python
        #convert BGR to HSV
        imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
        # create the Mask
        mask=cv2.inRange(imgHSV,lowerBound,upperBound)
        #morphology
        maskOpen=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
        maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)

        maskFinal=maskClose
        conts,h=cv2.findContours(maskFinal.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
        if (len(conts)==3):
            x1,y1,w1,h1=cv2.boundingRect(conts[0])
            x2,y2,w2,h2=cv2.boundingRect(conts[1])
            x3,y3,w3,h3=cv2.boundingRect(conts[2])

            cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
            cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)
            cv2.rectangle(img,(x3,y3),(x3+w3,y3+h3),(255,0,0),2)

            mouse_press.mousepress(1)
            for i in range(100):
                pass
            mouse_press.mouserelease(1)

        elif(len(conts)==2):
            x1,y1,w1,h1=cv2.boundingRect(conts[0])
            x2,y2,w2,h2=cv2.boundingRect(conts[1])
            cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
            cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)

            if abs(y1-y2)>20:
                if(pinchFlag==1):
                    pinchFlag=0
                    mouse_press.mouserelease(0)
```

```
                cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
                cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)
                cx1=x1+w1//2
                cy1=y1+h1//2
                cx2=x2+w2//2
                cy2=y2+h2//2
                cx=(cx1+cx2)//2
                cy=(cy1+cy2)//2
                cv2.line(img, (cx1,cy1),(cx2,cy2),(255,0,0),2)
                cv2.circle(img, (cx,cy),2,(0,0,255),2)
                mouseLoc=(sx-(cx*sx//camx), cy*sy//camy)
                mouse.position=mouseLoc
                while mouse.position!=mouseLoc:
                    pass
            else:
                if (y1 - prey)>0:
                    prey = y1
                    keyboard_press.keypress(2)
                    keyboard_press.keyrelease(2)
                elif (y1 - prey)<0:
                    prey = y1
                    keyboard_press.keypress(1)
                    keyboard_press.keyrelease(1)


        elif(len(conts)==1):
            x,y,w,h=cv2.boundingRect(conts[0])
            if(pinchFlag==0):
                pinchFlag=1
                mouse_press.mousepress(0)
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            cx=x+w//2
            cy=y+h//2
            cv2.circle(img,(cx,cy),(w+h)//4,(0,0,255),2)
            mouseLoc=(sx-(cx*sx//camx), cy*sy//camy)
            mouse.position=mouseLoc
            while mouse.position!=mouseLoc:
                pass
        cv2.imshow("cam",img)


        if cv2.waitKey(1) == ord('q'):
            break


    cam.release()
    cv2.destroyAllWindows()
```

```python
#from pynput.keyboard import Key, Controller
import pyautogui
import cv2
import numpy as np
import wx

#keybord = Controller()

def keypress(b):
    if b==1:
        pyautogui.press('up')
    elif b==2:
        pyautogui.press('down')


def keyrelease(b):
    if b==1:
        pyautogui.keyUp('up')
    elif b==2:
        pyautogui.keyUp('down')
```

```python
from pynput.mouse import Button, Controller

import cv2
import numpy as np
import wx
m=Controller()

def mousepress(b):
    if b==0:
        m.press(Button.left)
    elif b==1:
        m.press(Button.right)

def mouserelease(b):
    if b==0:
        m.release(Button.left)
    elif b==1:
        m.release(Button.right)
```

```
import tensorflow as tf
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout , Activation , Flatten
from keras.layers import Convolution2D , MaxPooling2D
from keras.utils import np_utils
import cv2
import os

import warnings
warnings.filterwarnings(action='ignore')


training_data = []
datadir = "D:/Projects_Code/Hand_gesture_recognition/gesture_data/"
cat = ['00','01','02','03','04']
for c in cat:
    path = os.path.join(datadir,c)
    class_num = cat.index(c)
    for img in os.listdir(path):

        img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
        new_array = cv2.resize(img_array,(100,100))
        training_data.append([new_array,class_num])

model = Sequential()

model.add(Convolution2D(32,(5,5), activation='relu',input_shape=(1,100,100),data_format='channels_first'))
model.add(MaxPooling2D(pool_size=(2,2)))

# number of gaps between each pool

model.add(Convolution2D(32,(5,5), activation='relu',input_shape=(1,100,100),data_format='channels_first'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.50))


model.add(Flatten())
```

```
model.add(Convolution2D(32,(5,5), activation='relu',input_shape=(1,100,100),data_format='channels_first'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.50))


model.add(Flatten())


model.add(Dense(1000,activation='tanh'))
model.add(Dropout(0.25))

model.add(Dense(500,activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(128,activation='tanh'))
model.add(Dropout(0.25))

model.add(Dropout(0.05))
model.add(Dense(5,activation='softmax'))



model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(x_train,y,batch_size=32,epochs=5,verbose=1)


lower = np.array([0, 48, 80], dtype = "uint8")
upper = np.array([20, 255, 255], dtype = "uint8")
kernelopen = np.ones((2,2))
kernelclose = np.ones((3,3))


test_img = cv2.imread("D:/Projects_Code/Hand_gesture_recognition/gesture_data/test.jpg")
```

Git Hub Project Link--

https://github.com/Vanshika29/Mini-project-II

Conclusion

Difference between results (say accuracy score) of test dataset and training dataset is 0.1%

As we see, the difference between scores of test dataset and training dataset is very low (0.1%), hence, the conclusion is that our final model (XGBoost) is good enough, and it doesn't have any overfitting or underfitting.

Although XGBoost (with n_estimators=20 and max_depth = 10) is good enough, there may be a chance to improve this model further, by say, increasing the number of estimators and trying out some more hyperparameters.

As we see above, Ensemble also has given good results, we can try Ensemble with some more models and with some more hyperparameters to improve the results further.

This experiment was limited to Machine Learning algorithms. You can try Deep Learning techniques (say CNN, etc.) to improve the results further.

References

https://github.com/zalandoresearch/fashion-mnist#why-we-made-fashion-mnist

https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/

https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/