

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

1. Introduction of Deep Learning	06
2. Installation of Jupyter Notebook on Windows.....	07
3. System and tools.....	08
3.1 Operating system	
3.2 Python	
3.3 Jupyter Notebook	
4. Keras Module.....	10
4.1 What is Keras?	
4.2 What is Backend?	
4.3 Advantages	
5. Introduction of CNN.....	12
5.1 Overview	
5.2 Architecture	
6. Introduction to the Hand Gesture Dataset.....	14
7. Project (Hand Gesture Recognition).....	16
8. Conclusion.....	26
9. References.....	27

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 logo, featuring the word "theano" in a blue, lowercase, sans-serif font.

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.

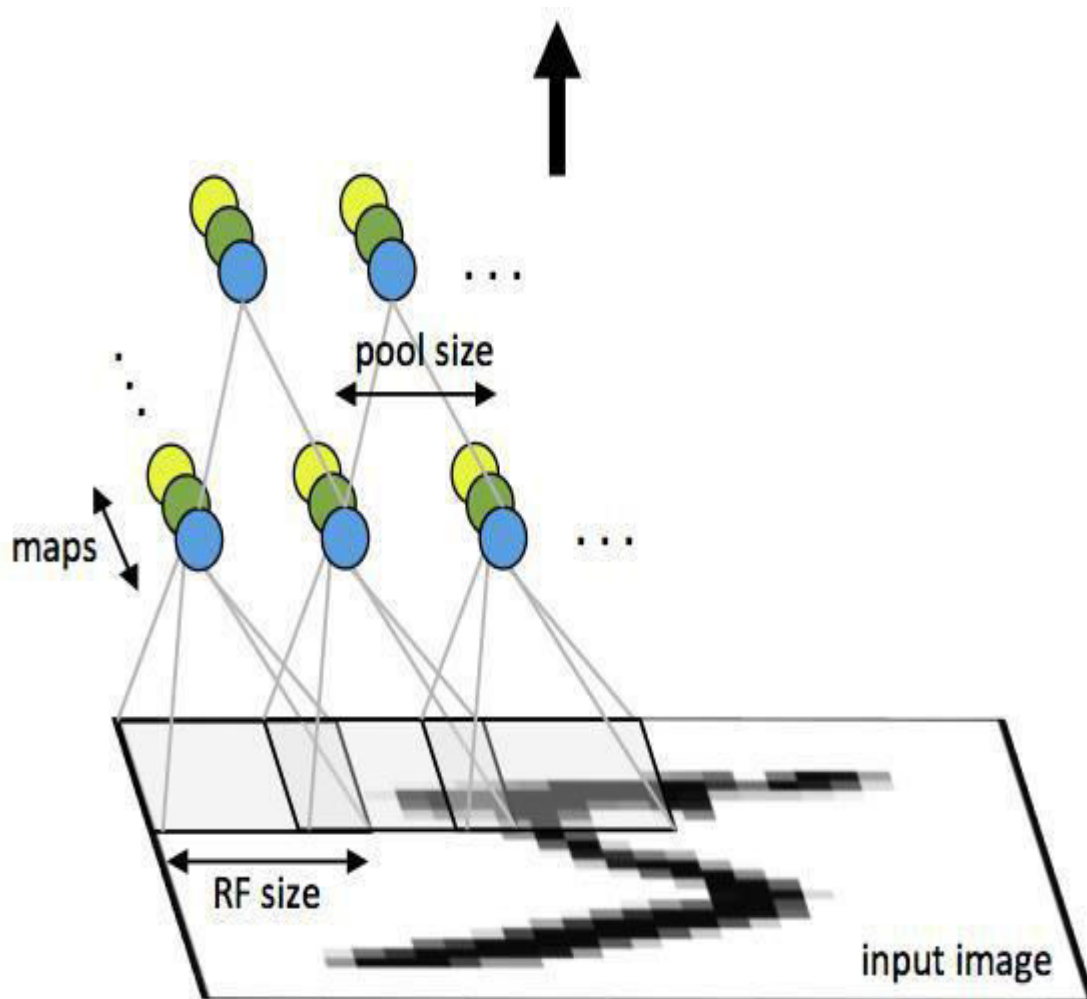
Introduction to Convolutional neural network

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 $m \times m \times r$ 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 $n \times n \times q$ 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 $p \times p$ 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.



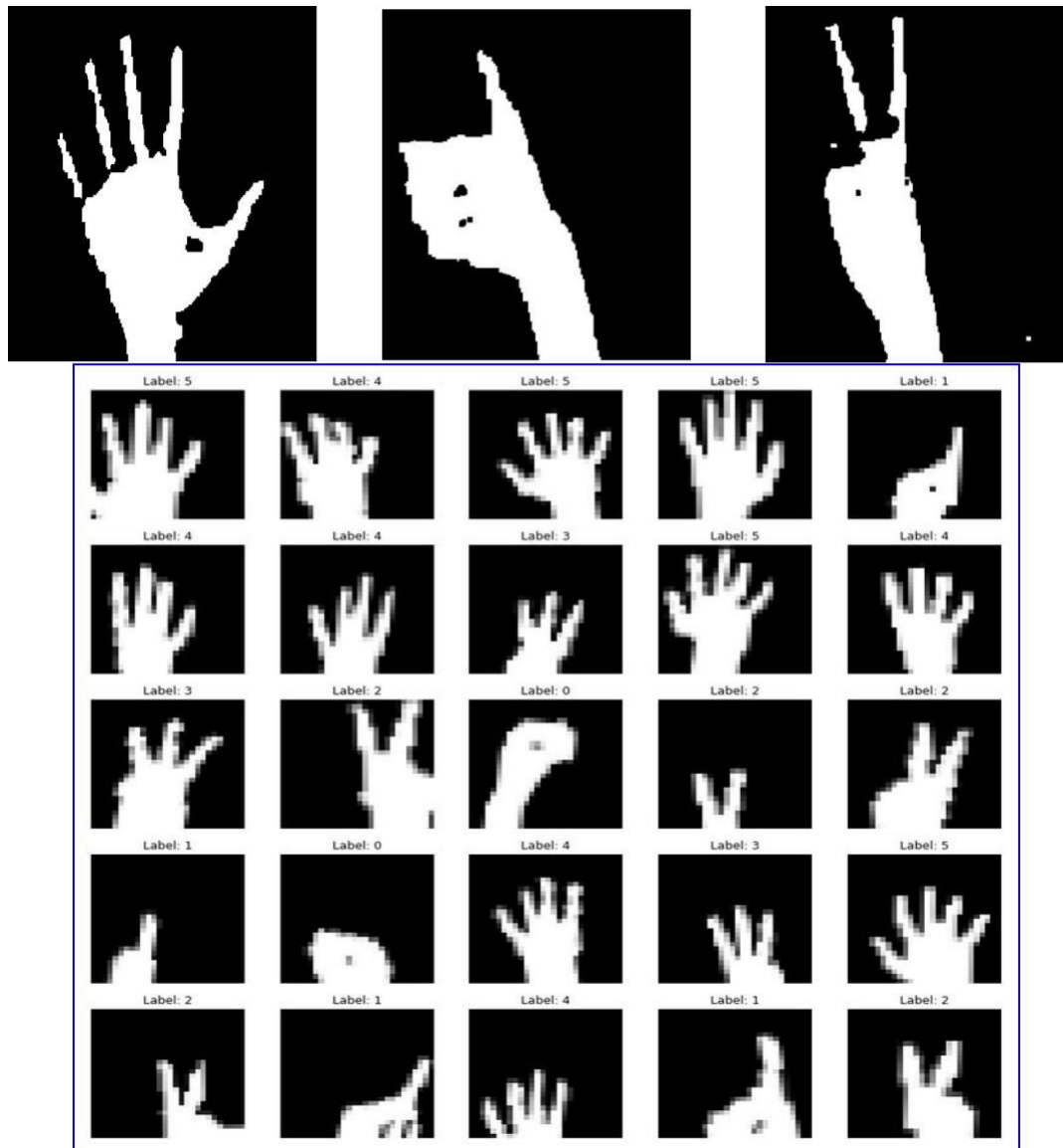
First layer of a convolutional neural network with pooling. Units of the same color have tied weights and units of different color represent different filter maps.

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.

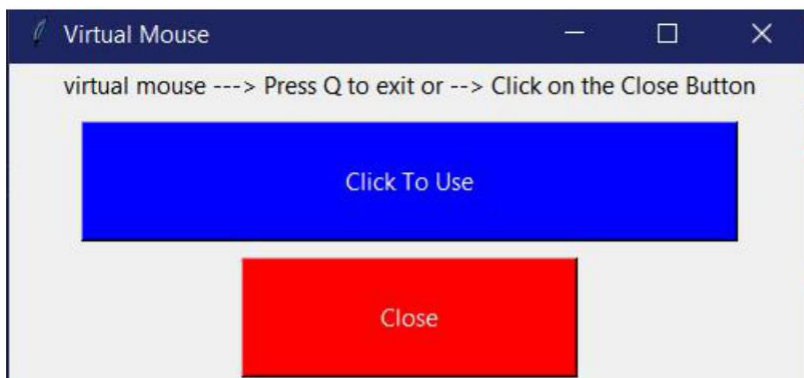
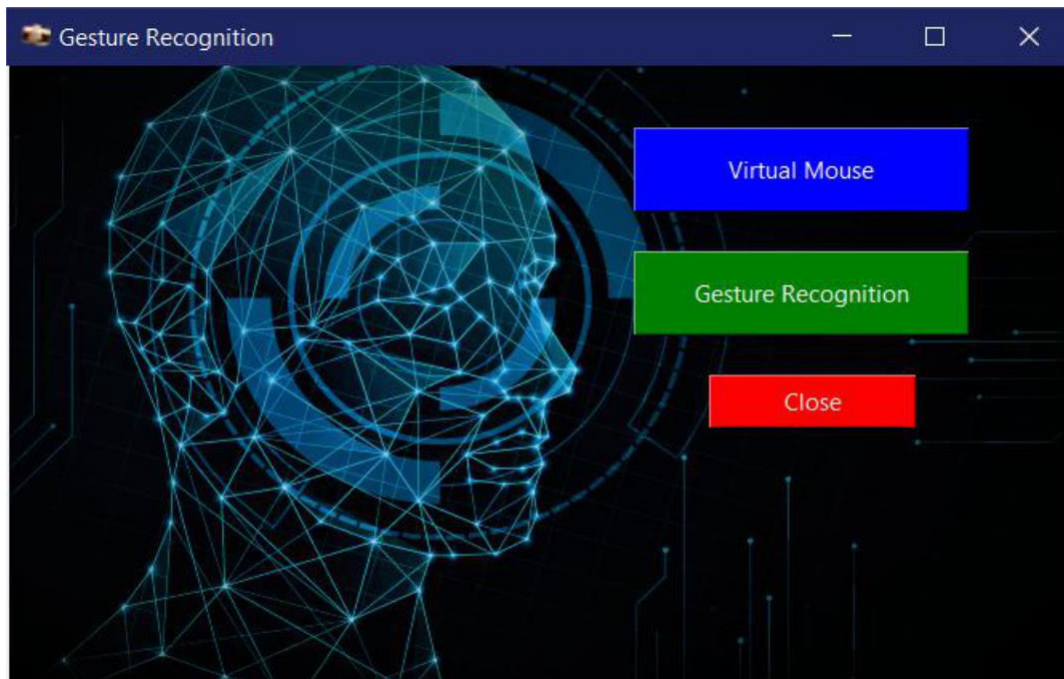
Introduction To Hand Gesture Dataset

Hand gesture Dataset which 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 > ...

```
1  from tkinter import Button,Canvas
2  from PIL import ImageTk,Image
3  from vir_mouse import main_virtual_mouse
4  import gesture
5  import tkinter
6
7  main_window = tkinter.Tk()
8  main_window.geometry("670x400")
9
10 my_img = ImageTk.PhotoImage(Image.open("D:\\Projects_Code\\Hand_gesture_recognition\\background1.png"))
11
12
13 my_label = tkinter.Label(image=my_img)
14 my_label.pack()
15
16
17 main_window.iconbitmap("D:\\Projects_Code\\Hand_gesture_recognition\\icon.ico")
18 main_window.title("Gesture Recognition")
19
20
21 def virtual_mouse(event):
22     def open(event):
23         main_virtual_mouse()
24
25     vir_mouse_window = tkinter.Tk()
26     vir_mouse_window.geometry("500x200")
27     vir_mouse_window.title("Virtual Mouse")
28     tkinter.Label(vir_mouse_window,text="virtual mouse ---> Press Q to exit or --> Click on the Close Button").pack()
29
30
31
32     b11 = tkinter.Button(vir_mouse_window,text="Click To Use",bg='blue',fg='white')
33     b11.bind("<Button-1>",open)
34     b11.config( height = 3, width = 50 )
35     b11.pack(pady=10)
36
```

```

32     b11 = tkinter.Button(vir_mouse_window,text="Click To Use",bg='blue',fg='white')
33     b11.bind("<Button-1>",open)
34     b11.config( height = 3, width = 50 )
35     b11.pack(pady=10)
36
37
38     bclose = tkinter.Button(vir_mouse_window,text="Close",command=vir_mouse_window.destroy,bg='red',fg='white')
39     bclose.config(height=3,width=25)
40     bclose.pack()
41
42
43     def gesture_recog_a(event):
44         def open(event):
45             gesture.gesture_recog()
46
47
48
49     gesture_window = tkinter.Tk()
50     gesture_window.geometry("500x200")
51     gesture_window.title("Gesture Recognition")
52     tkinter.Label(gesture_window,text="Gesture recognition ---> Press Q to exit or ---> Click on the Close Button").pack()
53     b22 = tkinter.Button(gesture_window,text="Click To Use",bg='green',fg='white')
54     b22.bind("<Button-1>",open)
55     b22.config( height = 3, width = 50 )
56
57     b22.pack(pady=10)
58
59     bclose = tkinter.Button(gesture_window,text="Close",command=gesture_window.destroy,bg='red',fg='white')
60     bclose.config(height=3,width=25)
61     bclose.pack()
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82

```

```

63     b1 = tkinter.Button(main_window,text="Virtual Mouse",bg='blue',fg='white')
64     b1.bind("<Button-1>",virtual_mouse)
65     b1.config( height = 2, width = 25 )
66     b1.pack(pady=10)
67     b1.place(relx = .9, rely = .1, anchor = 'ne')
68
69     b2 = tkinter.Button(main_window,text="Gesture Recognition",bg='green',fg='white')
70     b2.bind("<Button-1>",gesture_recog_a)
71     b2.config( height = 2, width = 25 )
72     b2.pack(pady=10)
73     b2.place(relx = .9, rely = .3, anchor = 'ne')
74
75
76     bclose_main = tkinter.Button(main_window,text="Close",command=main_window.quit,bg='red',fg='white')
77     bclose_main.config(height=1,width=15)
78     bclose_main.pack()
79     bclose_main.place(relx = .85, rely = .5, anchor = 'ne')
80
81     main_window.minsize(400,200)
82     main_window.mainloop()

```

```

D: > Projects_Code > Hand_gesture_recognition > gesture.py > ...
1  import cv2
2  import numpy as np
3  import math
4
5
6  def gesture_recog():
7      cap = cv2.VideoCapture(0)
8
9      while(1):
10
11         try: #an error comes if it does not find anything in window as it cannot find contour of max area
12             #therefore this try error statement
13
14             ret, frame = cap.read()
15             frame=cv2.flip(frame,1)
16             kernel = np.ones((3,3),np.uint8)
17
18             #define region of interest
19             roi=frame[100:300, 100:300]
20
21
22             cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)
23             hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
24
25
26
27         # define range of skin color in HSV
28         lower_skin = np.array([0,20,70], dtype=np.uint8)
29         upper_skin = np.array([20,255,255], dtype=np.uint8)
30
31         #extract skin color imagw
32         mask = cv2.inRange(hsv, lower_skin, upper_skin)
33
34
35
36
37
38
39
40
41
42
43
44         #find contours
45         contours,hierarchy= cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
46
47         #find contour of max area(hand)
48         cnt = max(contours, key = lambda x: cv2.contourArea(x))
49
50         #approx the contour a little
51         epsilon = 0.0005*cv2.arcLength(cnt,True)
52         approx= cv2.approxPolyDP(cnt,epsilon,True)
53
54
55         #make convex hull around hand
56         hull = cv2.convexHull(cnt)
57
58         #define area of hull and area of hand
59         areahull = cv2.contourArea(hull)
60         areacnt = cv2.contourArea(cnt)
61
62         #find the percentage of area not covered by hand in convex hull
63         arearatio=((areahull-areacnt)/areacnt)*100
64
65         #find the defects in convex hull with respect to hand
66         hull = cv2.convexHull(approx, returnPoints=False)
67         defects = cv2.convexityDefects(approx, hull)
68
69         # l = no. of defects
70         l=0
71
72         #code for finding no. of defects due to fingers
73         for i in range(defects.shape[0]):
74             s,e,f,d = defects[i,0]
75             start = tuple(approx[s][0])
76             end = tuple(approx[e][0])
77             far = tuple(approx[f][0])
78             pt= (100,180)
79

```

```

72     #code for finding no. of defects due to fingers
73     for i in range(defects.shape[0]):
74         s,e,f,d = defects[i,0]
75         start = tuple(approx[s][0])
76         end = tuple(approx[e][0])
77         far = tuple(approx[f][0])
78         pt= (100,180)
79
80
81     # find length of all sides of triangle
82     a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
83     b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
84     c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
85     s = (a+b+c)/2
86     ar = math.sqrt(s*(s-a)*(s-b)*(s-c))
87
88     #distance between point and convex hull
89     d=(2*ar)/a
90
91     # apply cosine rule here
92     angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
93
94
95     # ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)
96     if angle <= 90 and d>30:
97         l += 1
98         cv2.circle(roi, far, 3, [255,0,0], -1)
99
100    #draw lines around hand
101    cv2.line(roi,start, end, [0,255,0], 2)
102
103
104    l+=1
105
106    #print corresponding gestures which are in their ranges
107    font = cv2.FONT_HERSHEY_SIMPLEX
108    if l==1:
109        if areacnt<2000:
110            cv2.putText(frame,'Put hand in the box',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
111        else:
112            if arearatio<12:
113                cv2.putText(frame,'0 or Close Hand',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
114            elif arearatio<17.5:
115                cv2.putText(frame,'Best of luck',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
116            else:
117                cv2.putText(frame,'1 Finger',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
118
119    elif l==2:
120        cv2.putText(frame,'2 Fingers or Peace ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
121
122    elif l==3:
123
124        if arearatio<27:
125            cv2.putText(frame,'3 Fingers',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
126        else:
127            cv2.putText(frame,'ok',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
128
129    elif l==4:
130        cv2.putText(frame,'4 Fingers ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
131
132    elif l==5:
133        cv2.putText(frame,'5 Finger or Palm',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
134
135    elif l==6:
136        cv2.putText(frame,'reposition',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
137
138    else :
139        cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
140
141

```



```

106     #print corresponding gestures which are in their ranges
107     font = cv2.FONT_HERSHEY_SIMPLEX
108     if l==1:
109         if areacnt<2000:
110             cv2.putText(frame,'Put hand in the box',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
111         else:
112             if arearatio<12:
113                 cv2.putText(frame,'0 or Close Hand',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
114             elif arearatio<17.5:
115                 cv2.putText(frame,'Best of luck',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
116             else:
117                 cv2.putText(frame,'1 Finger',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
118
119     elif l==2:
120         cv2.putText(frame,'2 Fingers or Peace ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
121
122     elif l==3:
123         if arearatio<27:
124             cv2.putText(frame,'3 Fingers',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
125         else:
126             cv2.putText(frame,'ok',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
127
128     elif l==4:
129         cv2.putText(frame,'4 Fingers ',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
130
131     elif l==5:
132         cv2.putText(frame,'5 Finger or Palm',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
133
134     elif l==6:
135         cv2.putText(frame,'reposition',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
136
137     else :
138         cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
139
140
141
142     #show the windows
143     cv2.imshow('mask',mask)
144
145     cv2.imshow('frame',frame)
146 except:
147     pass
148
149
150 if cv2.waitKey(1) == ord('q'):
151     break
152
153 cv2.destroyAllWindows()
154 cap.release()
155

```

```

D: > Projects_Code > Hand_gesture_recognition > vir_mouse.py > ...
1  import cv2
2  import numpy as np
3  from pynput.mouse import Button, Controller
4
5  import mouse_press
6  import keyboard_press
7  import wx
8
9
10 def main_virtual_mouse():
11
12     mouse=Controller()
13     #keybord = Controller()
14
15     prey = 0
16     prex = 0
17     app=wx.App(False)
18     (sx,sy)=wx.GetDisplaySize()
19     (camx,camy)=(600,800)
20
21     lowerBound=np.array([33,80,40])
22     upperBound=np.array([102,255,255])
23
24     cam= cv2.VideoCapture(0)
25
26     kernelOpen=np.ones((5,5))
27     kernelClose=np.ones((10,10))
28     pinchFlag=0
29
30     while True:
31         ret, img=cam.read()
32         img=cv2.resize(img,(300,400))
33
34         #convert BGR to HSV
35         imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
36         # create the Mask
37         mask=cv2.inRange(imgHSV,lowerBound,upperBound)
38         #morphology
39
40         #convert BGR to HSV
41         imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
42         # create the Mask
43         mask=cv2.inRange(imgHSV,lowerBound,upperBound)
44         #morphology
45         maskOpen=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
46         maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)
47
48         maskFinal=maskClose
49         conts,h=cv2.findContours(maskFinal.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
50         if (len(conts)==3):
51             x1,y1,w1,h1=cv2.boundingRect(conts[0])
52             x2,y2,w2,h2=cv2.boundingRect(conts[1])
53             x3,y3,w3,h3=cv2.boundingRect(conts[2])
54
55             cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
56             cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)
57             cv2.rectangle(img,(x3,y3),(x3+w3,y3+h3),(255,0,0),2)
58
59             mouse_press.mousepress(1)
60             for i in range(100):
61                 pass
62             mouse_press.mouserelease(1)
63
64         elif(len(conts)==2):
65             x1,y1,w1,h1=cv2.boundingRect(conts[0])
66             x2,y2,w2,h2=cv2.boundingRect(conts[1])
67             cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
68             cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)
69
70             if abs(y1-y2)>20:
71                 if(pinchFlag==1):
72                     pinchFlag=0
73                     mouse_press.mouserelease(0)
74

```

```

69         cv2.rectangle(img, (x1,y1), (x1+w1,y1+h1), (255,0,0), 2)
70         cv2.rectangle(img, (x2,y2), (x2+w2,y2+h2), (255,0,0), 2)
71         cx1=x1+w1//2
72         cy1=y1+h1//2
73         cx2=x2+w2//2
74         cy2=y2+h2//2
75         cx=(cx1+cx2)//2
76         cy=(cy1+cy2)//2
77         cv2.line(img, (cx1,cy1), (cx2,cy2), (255,0,0), 2)
78         cv2.circle(img, (cx,cy), 2, (0,0,255), 2)
79         mouseLoc=(sx-(cx*sx//camx), cy*sy//camy)
80         mouse.position=mouseLoc
81         while mouse.position!=mouseLoc:
82             pass
83         else:
84             if (y1 - prey)>0:
85                 prey = y1
86                 keyboard_press.keypress(2)
87                 keyboard_press.keyrelease(2)
88             elif (y1 - prey)<0:
89                 prey = y1
90                 keyboard_press.keypress(1)
91                 keyboard_press.keyrelease(1)
92
93
94         elif(len(conts)==1):
95             x,y,w,h=cv2.boundingRect(conts[0])
96             if(pinchFlag==0):
97                 pinchFlag=1
98                 mouse_press.mousepress(0)
99             cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
100             cx=x+w//2
101             cy=y+h//2
102             cv2.circle(img, (cx,cy), (w+h)//4, (0,0,255), 2)
103             mouseLoc=(sx-(cx*sx//camx), cy*sy//camy)
104             mouse.position=mouseLoc
105             while mouse.position!=mouseLoc:
106                 pass
107
108         cv2.imshow("cam",img)
109
110         if cv2.waitKey(1) == ord('q'):
111             break
112
113
114     cam.release()
115     cv2.destroyAllWindows()

```

```
1  #from pynput.keyboard import Key, Controller
2  import pyautogui
3  import cv2
4  import numpy as np
5  import wx
6
7  #keybord = Controller()
8
9  def keypress(b):
10     if b==1:
11         pyautogui.press('up')
12     elif b==2:
13         pyautogui.press('down')
14
15
16  def keyrelease(b):
17     if b==1:
18         pyautogui.keyUp('up')
19     elif b==2:
20         pyautogui.keyUp('down')
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```



```
1  import tensorflow as tf
2  import numpy as np
3  from keras.models import Sequential
4  from keras.layers import Dense, Dropout , Activation , Flatten
5  from keras.layers import Convolution2D , MaxPooling2D
6  from keras.utils import np_utils
7  import cv2
8  import os
9
10 import warnings
11 warnings.filterwarnings(action='ignore')
12
13
14 training_data = []
15 datadir = "D:/Projects_Code/Hand_gesture_recognition/gesture_data/"
16 cat = ['00', '01', '02', '03', '04']
17 for c in cat:
18     path = os.path.join(datadir,c)
19     class_num = cat.index(c)
20     for img in os.listdir(path):
21
22         img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
23         new_array = cv2.resize(img_array,(100,100))
24         training_data.append([new_array,class_num])
25
26 model = Sequential()
27
28 model.add(Convolution2D(32,(5,5), activation='relu',input_shape=(1,100,100),data_format='channels_first'))
29 model.add(MaxPooling2D(pool_size=(2,2)))
30
31 # number of gaps between each pool
32
33 model.add(Convolution2D(32,(5,5), activation='relu',input_shape=(1,100,100),data_format='channels_first'))
34 model.add(MaxPooling2D(pool_size=(2,2)))
35 model.add(Dropout(0.50))
36
37
38 model.add(Flatten())
--
```

```

33 model.add(Convolution2D(32,(5,5), activation='relu',input_shape=(1,100,100),data_format='channels_first'))
34 model.add(MaxPooling2D(pool_size=(2,2)))
35 model.add(Dropout(0.50))
36
37
38 model.add(Flatten())
39
40
41
42 model.add(Dense(1000,activation='tanh'))
43 model.add(Dropout(0.25))
44
45 model.add(Dense(500,activation='relu'))
46 model.add(Dropout(0.25))
47
48 model.add(Dense(128,activation='tanh'))
49 model.add(Dropout(0.25))
50
51 model.add(Dropout(0.05))
52 model.add(Dense(5,activation='softmax'))
53
54
55
56 model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
57 model.fit(x_train,y,batch_size=32,epochs=5,verbose=1)
58
59
60 lower = np.array([0, 48, 80], dtype = "uint8")
61 upper = np.array([20, 255, 255], dtype = "uint8")
62 kernelopen = np.ones((2,2))
63 kernelclose = np.ones((3,3))
64
65
66 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/>

