GALGOTIAS UNIVERSITY

(Under the Uttar Pradesh Private Universities Act No. 12 of 2019)

Hand Emoji Recognition System

A Report for the Final Evaluation of Project 2

Submitted by

Mohd Ashraf

(1613101408/16SCSE101260)

in partial fulfillment for the award of the degree of

B.tech

In

Computer Science and Engineering

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Dr Varun Tiwari, M.Tech., Ph.D.,

(Professor)

May-2020

GALGOTIAS UNIVERSITY

(Under the Uttar Pradesh Private Universities Act No. 12 of 2019)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **“HAND EMOJI RECOGNITION SYSTEM”** is the bonafide work of **“MOHD ASHRAF (1613101408)”** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SABHARWAL,

PhD (Management), PhD (CS)

Professor & Dean,

SIGNATURE OF SUPERVISOR

Dr.VARUN TIWARI

M.Tech , PhD

Professor

Table of Contents

Abstract

Hand gestures and Emoji identification are the emerging areas of research. Emoji identification plays an imperative factor in our day-to-day life. This system contributes in an innovative, user friendly way of interaction with the computer, making it more flexible for the human interaction. Emoji identification exhibits ample application areas inclusive of human machine interaction, sign language, immersive game technology, social networking applications etc., we have introduced an approach which entirely rely upon the shape of hand gesture. It neglects the other hand gesture mechanisms such as skin colour, textures due to their dependence on variable light conditions and other influential factors. The images are takes as input with the help of web cam, further eliminating the background noise through pre-processing. The process of detection ,training and recognition of hand-gestures.Developed a 3 layer fully-connected neural network to and use back-propagation for each. There is no special hardware used anywhere in the process, comparing each label with the prediction probability to test model and observing how much iterations of training is required for the model to reach desired accuracy.

Introduction

In the application, machine learning algorithms and convolutional neural network layers are used in order to train the dataset to make a model which can then be used to detect the object we trained it for.

In the application we are detecting different hand emojis. Emojis are used to describe emotions or mood in electronic messages and web pages. Emoticons are of various types based on your required orientation of stimulating any thought in the form of miniature visuals. The first ASCII emoticons were : -) and : - (, were written by Scott Fahlman in 1982, but emoticons were actually given birth on the PLATO IV computer system in 1972.The emojis help us make our conversation over text better as we can express the right attitude and tone, which can’t be reflected over simple text messages. Different Hand Emojis are included in a folder which are utilised by the application.

In the project Iam using Gaussian mixture based segmentation algorithm which is used to detect the hand and the gesture.

Linear regression is supervised machine learning algorithm which is used for find out effecting relationship between variables. In this case the gestures we capture in real-time and the model generated after training are the different variables.

Input: We start the camera and the user can show the gestures to the system.

Output: The emoji which is detected according to the gesture provided by the user is displayed on the screen simultaneously .The Emojis shown will be of PNG format which are already stored in a folder.

Study of two very foremost algorithms in the Machine Learning Legacy and train their respective modals in order to classify handwritten digits ranging from 0-9 from the most common dataset around the MNIST dataset. With the presence of the most prestigious dataset we already have the data classified into their respective labels. A current approach on Computer vision techniques has a vital role of machine language algorithm especially when it comes to detection of any object or material or digits or even facts and figure in the near future that we are certainly proceeding to. We develop the two processes separately and then merge it into a single module for real time analysis. Weights and biases are randomly dispensed to each cell initially and analyzed further for a output value after each epoch the process is repeated until we get the output same as the desired output. Although before we jump to the conclusion we will be calculating the efficiency in the loss function or well said the cost function.

Logistic Regression is a type of Classification Algorithm where the output is just a probability of the input data hence its name of regression doesn’t go with the flow hence the main objective of this algorithm is to find the predictive probability of the input data and predict the best outcome out of the given set of data.

As compared to Deep neural network the phenomena of detection is pure ethical and, weight and biases are allocated to each input. Our computer analyses images as a dataset of numbers where each pixel represents a value. The values are different for different formats. As our dataset is captured in HSV format each cell value represents pixel intensity ranging from 0-255.0 being absolute black, 255 being white and grey values range in-between. The convolution layer works on a small matrix of dataset from whole picture and the values are evaluated and brought down to single value which is fed to a neuron, this is done through filter.The Modified National Institute of Standards and technology has been the greatest database of handwritten digits to train various image analysis systems. Thus taking the MNIST dataset we try to detect the correct handwritten digit using two approaches of Machine Learning. Comparing Logistic regression and Deep Neural Network and concluded its result to find the optimum and effective method to classify hand written digits with the help of MNIST dataset. These methods undergo multinomial logistic regression or softmax regression i.e. both the models can predict different possible outcome or in layman terms two discrete outcomes. Hence concluding which approach is better to classify handwritten digit of any given dataset.

Problem Statement

Emoji are a better for the purpose of communication as they are short and can deliver the information. These are basically miniature visuals which can easily be recognized and understood by anyone. Moreover if in a rush or hurry these emojis can make conversations easier and more meaningful. In our texting applications we have a huge range of emojis, searching them can be tedious task. Incorporating a method which can able the applications to recognise emojis through primary camera of the device can make conversations easier.. Especially for old aged people who are not so tech-savvy. The application to recognise emojis through hand gestures can provide great assistance to them. Whenever there is any computer vision task or the ability to recognize stuffs by the machine itself we run into machine learning algorithms and beauty of every ML algorithms lies in the ability to correct itself after every hit and run of its code generator. Thus first thing first we have a variety of options to pop on to but most importantly it is the ability to recognize which option will suit the given problem in an effective manner. And to do so we compare these options or algorithms on the basis of reduction of cost function with over a set of iterations over a given set of same dataset. Ideally we are recognizing handwritten digits from the world's most friendly Dataset i.e. the MNIST dataset to effectively recognize and compare these set of options to classify the result in the best way possible. So whenever we are comparing these algorithms it is known that each function in one way or the other and in their own terms hence we need to pull out the main deterministic factor i.e. the loss function in these certain algorithms.

SOFTWARE REQUIREMENT SPECIFICATION

Purpose

Hand Emoji Recognition system is an application which is used to recognize hand gestures and display the identified hand emoji used these days in various Texting Application and web based interactive platforms.

Document Conventions

The Documents subsequently explains the needs and traits of the project produced, with sufficient re-modeling of data. The important data has expressed in proper fonts and using techniques

Intended Audience and Reading Suggestions

The application developed is intended to be used by all the people globally who use any sort of texting application. The application is in development stage and for now it only recognizes hand gestures as hand emoticons. This application can be used beneficial for people who are not that tech-savvy

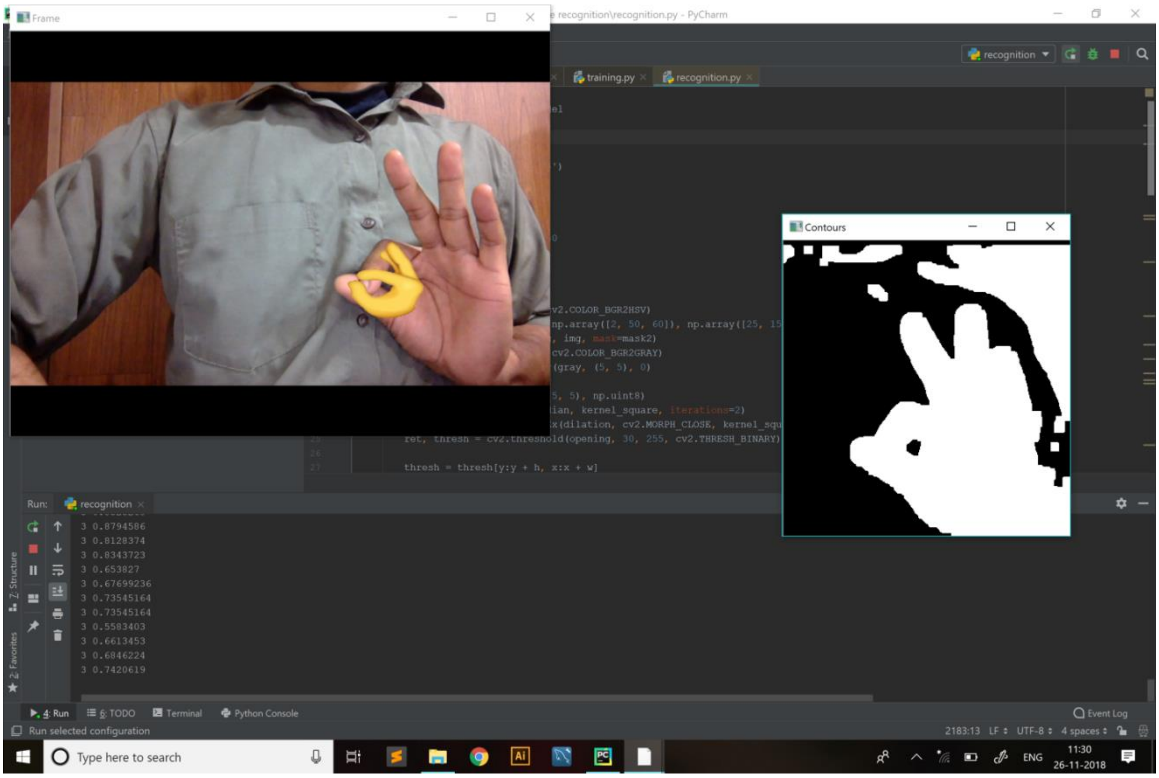
Product Scope

- It is a subset of Machine learning and Artificial Intelligence.
- It’s a Real Time Recognition System which is easy to use.
- It’s a four layer Hierarchical architecture for real time recognition of hand Emoticons
- Moreover it is User-Friendly
- It is very Cost-Effective, Faster, And Reliable, and towards the idea of typing less conversation.
- This method can be incorporated in any device which has a front camera and enables communication over text

Overall Description

Product Perspective

Product:



```
el
')
0
v2.COLOR_BGR2HSV)
np.array([2, 50, 60]), np.array([25, 15
, img, mask=mask2)
cv2.COLOR_BGR2GRAY)
(gray, (5, 5), 0)

5, 5), np.uint8)
ian, kernel_square, iterations=2)
x(dilation, cv2.MORPH_CLOSE, kernel_squ
ret, thresh = cv2.threshold(opening, 30, 255, cv2.THRESH_BINARY)

thresh = thresh[y:y + h, x:x + w]
```

```
3 0.8794586
3 0.8128374
3 0.8343723
3 0.653827
3 0.67699236
3 0.73545164
3 0.73545164
3 0.5583403
3 0.6613453
3 0.6846224
3 0.7420619
```

Operating Environment

The software requires Python 3.6 and an IDE that can run Python in it ex. PyCharm. The Software will coexist with mostly all the system based application without creating a problem to other tools. This can work on both UBUNTU and WINDOWS based systems

Design and Implementation Constraints

The software does not hold any constraints as the developer can always edit and remap the software according the fresh needs and ideas. The lightning in the room may affect the user input and difficulty in identification of gesture, better the processor better the training time and better the results Though the developers needs a specific tool set to work upon the described software. The client cannot make and code based changes.

User Documentation

The software comes with a manual and software based Readme File to help the client out

Assumptions and Dependencies

The Software uses Keras, Tensor flow and OvenCV dependencies from the Libraries of Python 3.6

Other Nonfunctional Requirements

Performance Requirements

It will require a minimum of 4GB of RAM and there is no memory constraint for the application.

Safety Requirements

There is no as such safety requirement for this application to run however for better performance of this application it is preferred to use this

Security Requirements

Once the gesture database is created user must not alter the given data in it as it may conflict at the time of training.

Software Quality Attributes

The user can use a better primary camera to detect hand gesture in a more precise and fundamental way

Business Rules

No business rule as such required. As the application is still under development and is not incorporated by means of any business

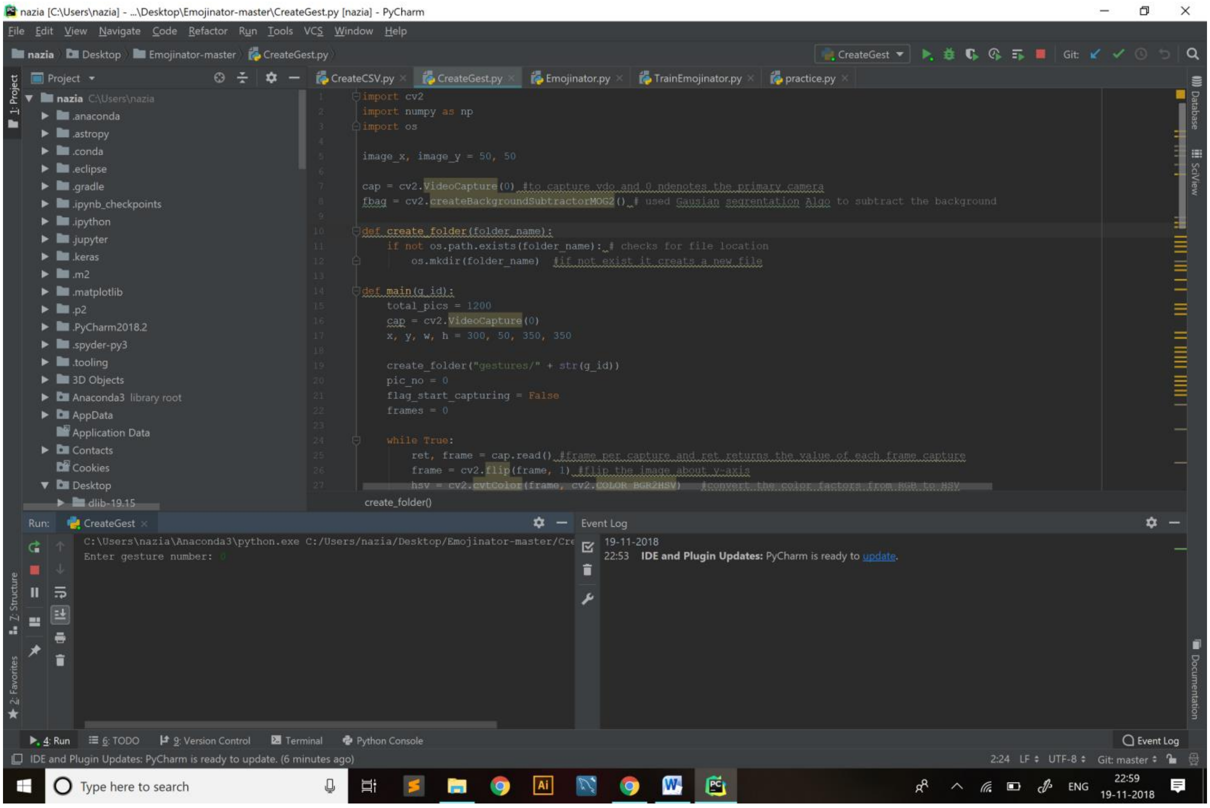
RELATED WORK

Machine learning enables our system to make decisions by operating in a self-learning manner. We feed the data and analyses the trends, so when presented with new data, it provides us with explanations according to experience i.e. training of model from the given set of training dataset. We talk about the two major approaches of the machine learning algorithms the first being the Logistic Regression and the other being the Deep neural Networks, we could have also gone for the shallow networks but we aim to implement the same in our future reports. The neurons must be arranged on to different layers, and then the connections between them are made to successfully make a network like structure. The input layer is fed with data; the output is what gives us a value which is interpreted between them can be any number of hidden layers containing neurons in conjugated layout.

The data comprises of noise and errors too, when all the values from the neural net are appended for evaluation of output value the model complies with errors. To avoid this a method called Dropout discussed in, which is introduced to reduce over fitting by randomly removing or not considering values of some neurons from the network and the rest are trained by back-propagation. Aftermath of which we calculate the cost of each iteration. All comparison have always been on the basis of certain variable or parameters, hence we consider the cost of each iteration for this comparison that is to be useful for also cost is the only determining factor in all these algorithms such as Gradient Decent or Gaussian mixture based segmentation algorithm.

Implementation

Step 1:- Capturing Data Set



```
import cv2
import numpy as np
import os

image_x, image_y = 50, 50

cap = cv2.VideoCapture(0) #to capture vdo and 0 ndenotes the primary camera
fbag = cv2.createBackgroundSubtractorMOG2() # used Gausian segrentation Algo to subtract the background

def create_folder(folder_name):
    if not os.path.exists(folder_name): # checks for file location
        os.mkdir(folder_name)  #if not exist it creats a new file

def main(g_id):
    total_pics = 1200
    cap = cv2.VideoCapture(0)
    x, y, w, h = 300, 50, 350, 350

    create_folder("gestures/" + str(g_id))
    pic_no = 0
    flag_start_capturing = False
    frames = 0

    while True:
        ret, frame = cap.read() #frame per capture and ret returns the value of each frame capture
        frame = cv2.flip(frame, 1) #flip the image about y-axis
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)   #convert the color factors from RGB to HSV
```

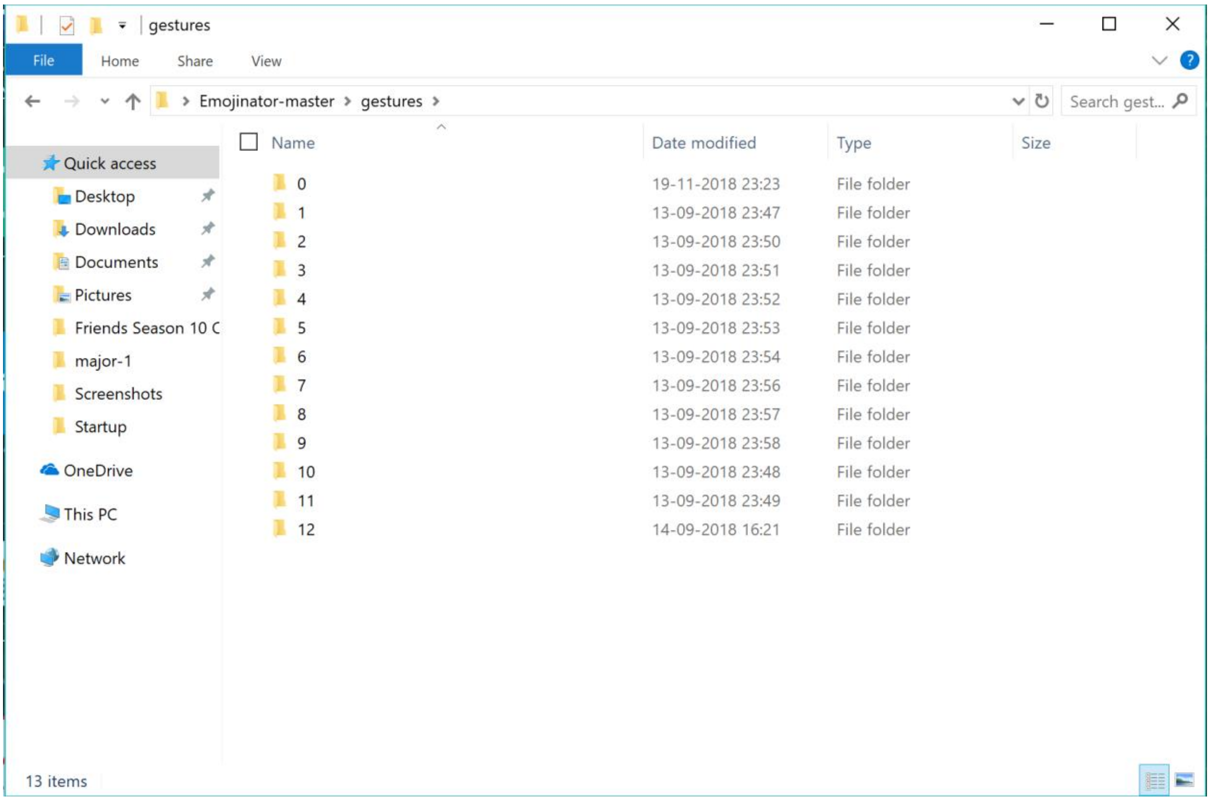
create_folder()

```
C:\Users\nazia\Anaconda3\python.exe C:/Users/nazia/Desktop/Emojinator-master/Cre
Enter gesture number: 0
```

```
19-11-2018
22:53   IDE and Plugin Updates: PyCharm is ready to update.
```

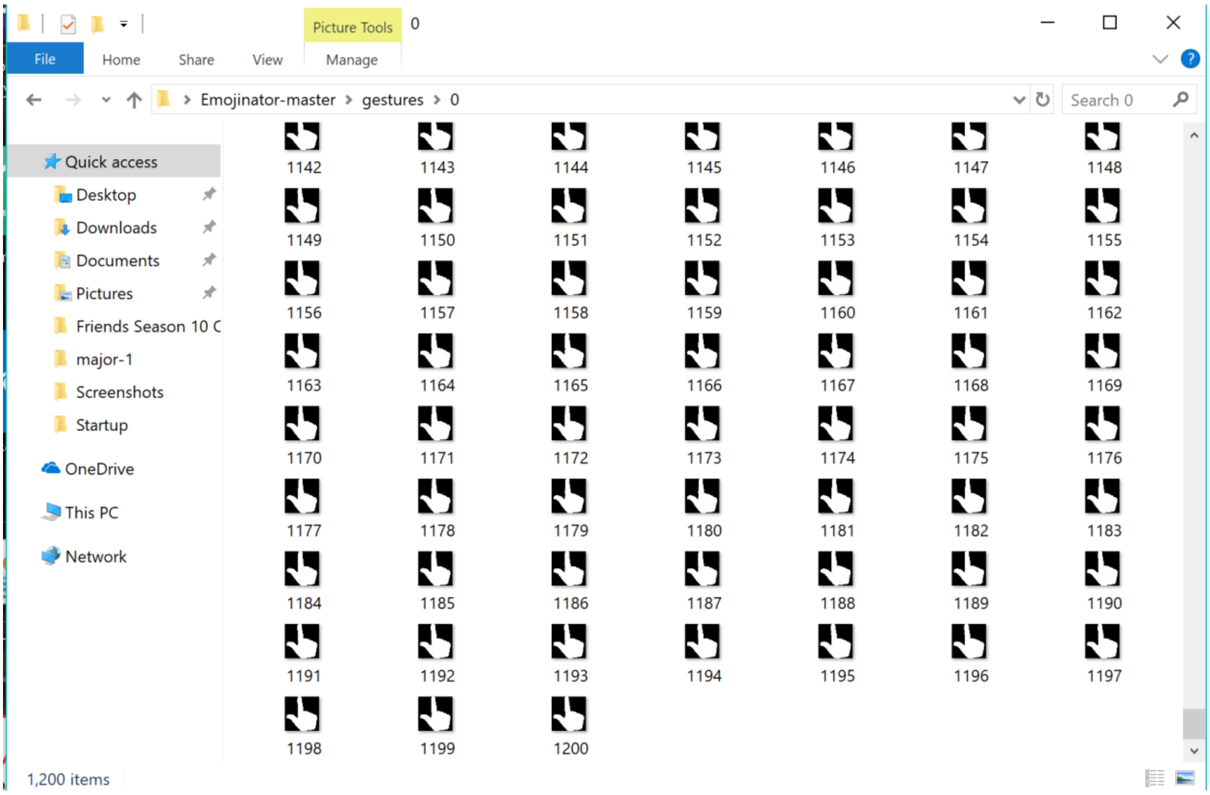
IDE and Plugin Updates: PyCharm is ready to update. (6 minutes ago)

We have a separate file for all the hand emojis we are going to recognize in this application. In order to train our application, the application needs to know which gesture represents which hand emoji. We capture 1200 images of each gesture which are later used for training. In above image you can see, we enter the gesture number. Suppose it's 0. Then a new folder will be created and 1200 images of the captured gesture are stored in that file.Such folders for all the emojis are created. We are using 11 hand-emojis in our application.The possible hand-gestures for all of them are captured and stored. All of these 1200 images are not similar they should vary in position, orientation etc so that recognition can be more accurate even if the user input is diverse.



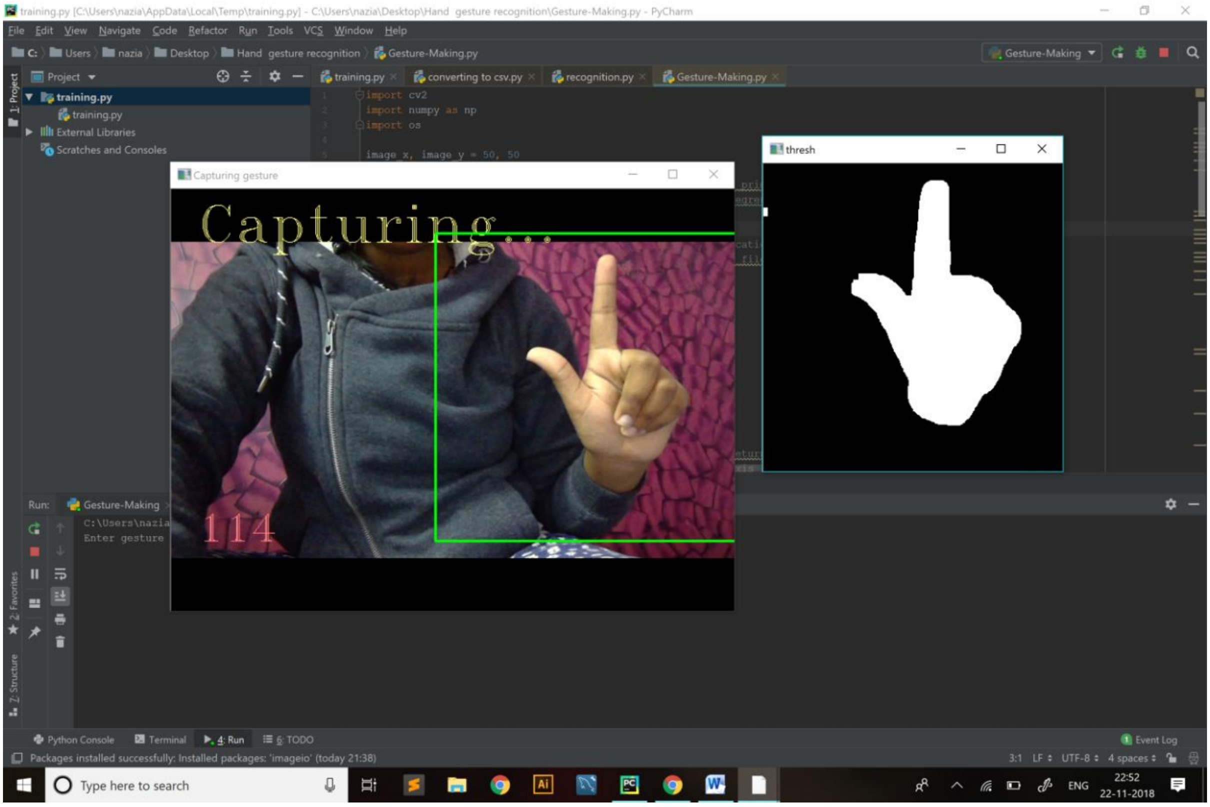
Name	Date modified	Type	Size
0	19-11-2018 23:23	File folder	
1	13-09-2018 23:47	File folder	
2	13-09-2018 23:50	File folder	
3	13-09-2018 23:51	File folder	
4	13-09-2018 23:52	File folder	
5	13-09-2018 23:53	File folder	
6	13-09-2018 23:54	File folder	
7	13-09-2018 23:56	File folder	
8	13-09-2018 23:57	File folder	
9	13-09-2018 23:58	File folder	
10	13-09-2018 23:48	File folder	
11	13-09-2018 23:49	File folder	
12	14-09-2018 16:21	File folder	

13 items



1142 1143 1144 1145 1146 1147 1148
1149 1150 1151 1152 1153 1154 1155
1156 1157 1158 1159 1160 1161 1162
1163 1164 1165 1166 1167 1168 1169
1170 1171 1172 1173 1174 1175 1176
1177 1178 1179 1180 1181 1182 1183
1184 1185 1186 1187 1188 1189 1190
1191 1192 1193 1194 1195 1196 1197
1198 1199 1200

1,200 items



training.py [C:\Users\nazia\AppData\Local\Temp\training.py] - C:\Users\nazia\Desktop\Hand gesture recognition\Gesture-Making.py - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
C: 〉 Users 〉 nazia 〉 Desktop 〉 Hand gesture recognition 〉 Gesture-Making.py
Gesture-Making
Project
training.py
training.py
External Libraries
Scratches and Consoles
training.py
converting to csv.py
recognition.py
Gesture-Making.py
Capturing gesture
Capturing...
114
thresh
Run:
Gesture-Making
Python Console
Terminal
4: Run
6: TODO
Event Log
3:1 LF ÷ UTF-8 ÷ 4 spaces ÷
Type here to search
ENG
22:52
22-11-2018

```
import cv2
import numpy as np
import os

image_x, image_y = 50, 50
```

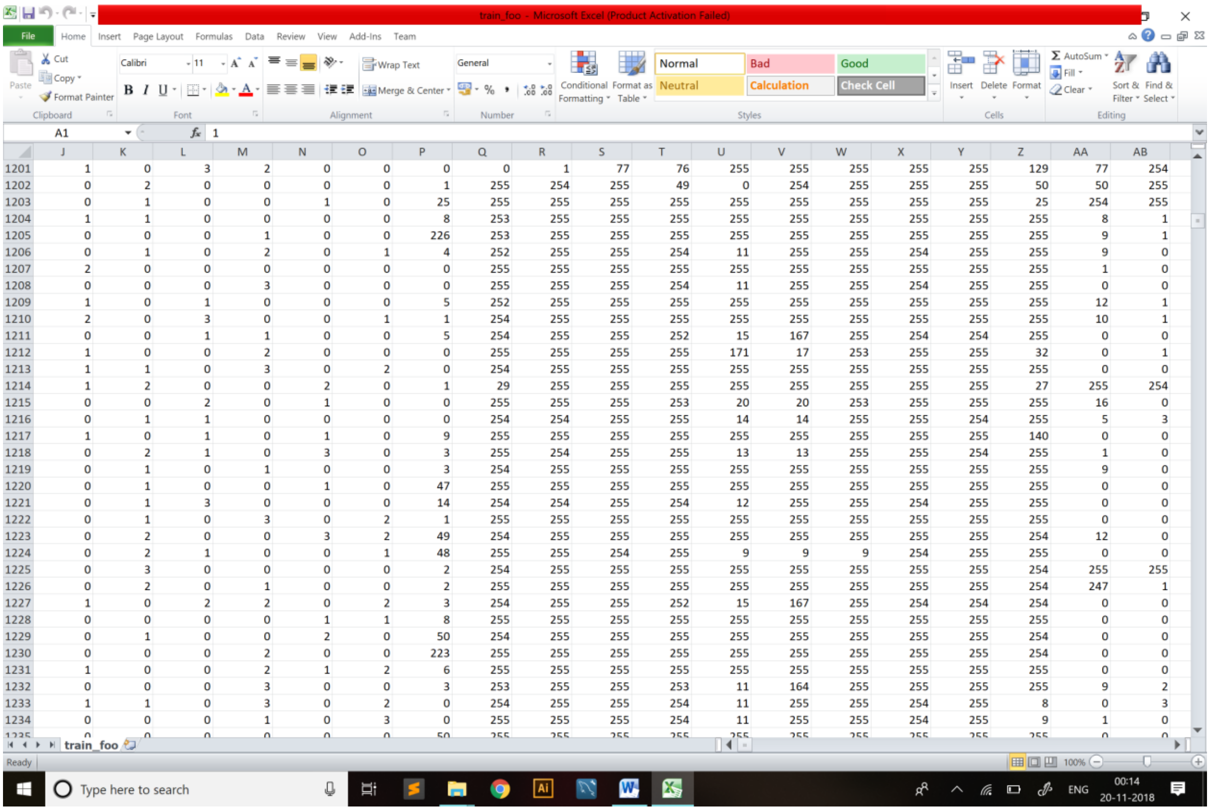
```
C:\Users\nazia
Enter gesture
```

Packages installed successfully: Installed packages: 'imageio' (today 21:38)

We use the Gaussian Mixture-based Background/Foreground Segmentation Algorithm which generates a foreground mask i.e. basically a binary image containing pixels which do belong to moving objects in the scene the technique is carried out by the primary camera.
The algorithm works like a background subtractor. We calculate the foreground mask performing a subtraction between the current frame and a background model.so given, rest it covers everything that is considered as a background ,we convert the image from RGB to HSV (hue, saturation, value) i.e. we calculate the hue, saturation and value factors in the segmented part of the image and uses the bitwise and function of CV2 which blackout the background where the hue and saturation is low as compared to the front where these factors are high sue to illumination factors and then an inverse mask is created for Gaussian blurring in which we mainly remove the noise and smoothen the image. Now the final thresh output is created by using all the above segmentation according to pixels i.e. the dark objects turn black and the light object turns white. This completes the process of Gaussian Mixture-based Background/Foreground Segmentation Algorithm.

The capturing process starts as we press “c”. We have captured 1200 images of each emoji. which are used for the training purpose of each Hand Emoji.

Step 2:- Conversion The captured gesture files which are 1200 jpeg files for each emoji are converted into csv format and stored in a file named train_foo.csv(comma separated values).



A1 fx 1

	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1201	1	0	3	2	0	0	0	0	1	77	76	255	255	255	255	255	129	77	254
1202	0	2	0	0	0	0	1	255	254	255	49	0	254	255	255	255	50	50	255
1203	0	1	0	0	1	0	25	255	255	255	255	255	255	255	255	255	25	254	255
1204	1	1	0	0	0	0	8	253	255	255	255	255	255	255	255	255	255	8	1
1205	0	0	0	1	0	0	226	253	255	255	255	255	255	255	255	255	255	9	1
1206	0	1	0	2	0	1	4	252	255	255	254	11	255	255	254	255	255	9	0
1207	2	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255	255	1	0
1208	0	0	0	3	0	0	0	255	255	255	254	11	255	255	254	255	255	0	0
1209	1	0	1	0	0	0	5	252	255	255	255	255	255	255	255	255	255	12	1
1210	2	0	3	0	0	1	1	254	255	255	255	255	255	255	255	255	255	10	1
1211	0	0	1	1	0	0	5	254	255	255	252	15	167	255	254	254	255	0	0
1212	1	0	0	2	0	0	0	255	255	255	255	171	17	253	255	255	32	0	1
1213	1	1	0	3	0	2	0	254	255	255	255	255	255	255	255	255	255	0	0
1214	1	2	0	0	2	0	1	29	255	255	255	255	255	255	255	255	27	255	254
1215	0	0	2	0	1	0	0	255	255	255	253	20	20	253	255	255	255	16	0
1216	0	1	1	0	0	0	0	254	254	255	255	14	14	255	255	254	255	5	3
1217	1	0	1	0	1	0	9	255	255	255	255	255	255	255	255	255	140	0	0
1218	0	2	1	0	3	0	3	255	254	255	255	13	13	255	255	254	255	1	0
1219	0	1	0	1	0	0	3	254	255	255	255	255	255	255	255	255	255	9	0
1220	0	1	0	0	1	0	47	255	255	255	255	255	255	255	255	255	255	0	0
1221	0	1	3	0	0	0	14	254	254	255	254	12	255	255	254	255	255	0	0
1222	0	1	0	3	0	2	1	255	255	255	255	255	255	255	255	255	255	0	0
1223	0	2	0	0	3	2	49	254	255	255	255	255	255	255	255	255	254	12	0
1224	0	2	1	0	0	1	48	255	255	254	255	9	9	9	254	255	255	0	0
1225	0	3	0	0	0	0	2	254	255	255	255	255	255	255	255	255	254	255	255
1226	0	2	0	1	0	0	2	255	255	255	255	255	255	255	255	255	254	247	1
1227	1	0	2	2	0	2	3	254	255	255	252	15	167	255	254	254	254	0	0
1228	0	0	0	0	1	1	8	255	255	255	255	255	255	255	255	255	255	0	0
1229	0	1	0	0	2	0	50	254	255	255	255	255	255	255	255	255	254	0	0
1230	0	0	0	2	0	0	223	255	255	255	255	255	255	255	255	255	254	0	0
1231	1	0	0	2	1	2	6	255	255	255	255	255	255	255	255	255	255	0	0
1232	0	0	0	3	0	0	3	253	255	255	253	11	164	255	255	255	255	9	2
1233	1	1	0	3	0	2	0	254	255	255	254	11	255	255	254	255	8	0	3
1234	0	0	0	1	0	3	0	255	255	255	254	11	255	255	254	255	9	1	0
1235	0	0	0	0	0	0	50	255	255	255	255	255	255	255	255	255	255	0	0

The values in each cell represent the hue value at each pixel of the images. This file contains values for all the images we have captured.This is the dataset which will actually be used in the training process.

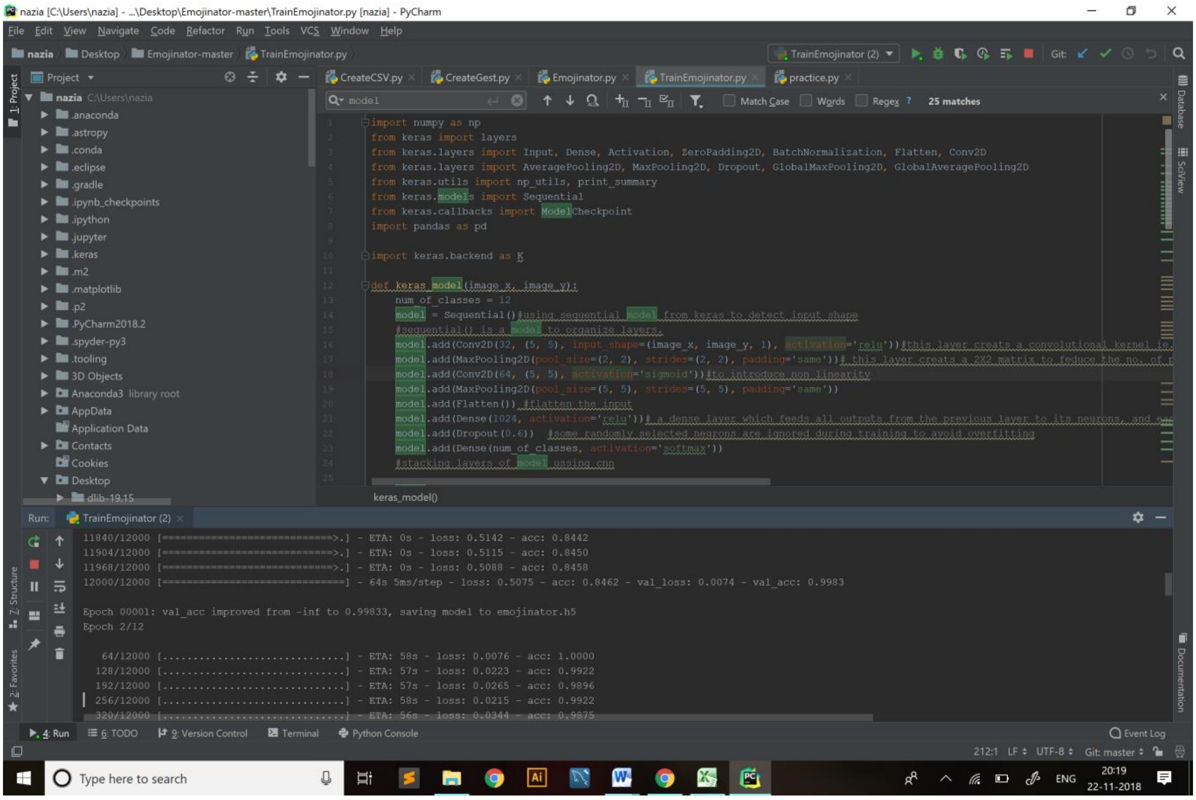
Step 3:-Training

The training of the dataset is done following the algorithms linear regression and gradient descent. The algorithm is applied on the layers made by using the concept of convolutional neural networks. They are made up of neurons that have weights and biases from each node to another node. Each neuron receives some inputs, performs a dot product and optionally follows it. Each neuron feeds its output data as input to the next neuron and this is followed till the end until we reach the labels and then the expected result and the output are compared. Through which we actually detect the difference and compute losses, so that we can adjust the weights at each neuron and perform dots product following with the passing on of values. This whole process is termed as 1 epoch.It depends on the type of dataset we are using that how many rounds of epoch will be required to get satisfactory output and we will receive more accurate output as much we train.

Each node of layer is connected to all the nodes of the second layer.We have made 2 convolutional layers and flattened them.

Linear regression is a type of supervised machine learning algorithm.In the application there are predefined labels which will be the output according to the input provided by the user.

Linear regression can be described as the line of best fit i.e. the line that can be line that relate the dependent and the independent variable so that we can focus on new observations just like predicting sales, value or figure over time. The correlation between these dependent and independent variable do not show linearity but do show a linear pattern over the linear line y=mx + c. thus the simple linear regression model also includes the E where E is the error term thus the equation bieng y = Bo + B1X + E. The dataset is divided into training and testing data. The dataset is not linear,the values fall across the graph. To reduce error between these datapoints and the best fit equation we use gradient descent.

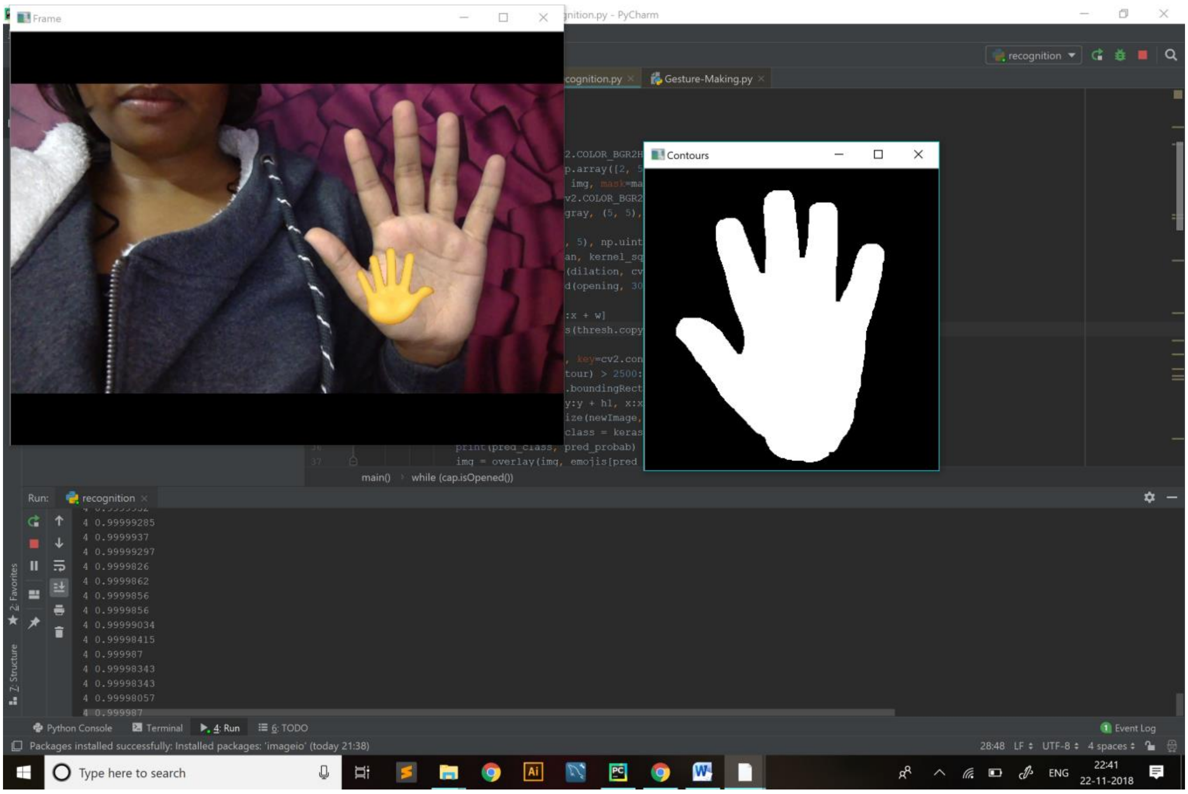


```
import numpy as np
from keras import layers
from keras.layers import Input, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D
from keras.layers import AveragePooling2D, MaxPooling2D, Dropout, GlobalMaxPooling2D, GlobalAveragePooling2D
from keras.utils import np_utils, print_summary
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint
import pandas as pd

import keras.backend as K

def keras_model(image_x, image_y):
    num_of_classes = 12
    model = Sequential()#using sequential model from keras to detect input shape
    #sequential() is a model to organize layers.
    model.add(Conv2D(32, (5, 5), input_shape=(image_x, image_y, 1), activation='relu'))#this layer creats a convolutional kernel ie.
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'))# this layer creats a 2X2 matrix to feduce the no. of p
    model.add(Conv2D(64, (5, 5), activation='sigmoid'))#to introduce non linearity
    model.add(MaxPooling2D(pool_size=(5, 5), strides=(5, 5), padding='same'))
    model.add(Flatten()) #flatten the input
    model.add(Dense(1024, activation='relu'))# a dense layer which feeds all outputs from the previous layer to its neurons, and eac
    model.add(Dropout(0.6))  #some randomly selected neurons are ignored during training to avoid overfitting
    model.add(Dense(num_of_classes, activation='softmax'))
    #stacking layers of model ussing cnn
```

```
11840/12000 [============================>.] - ETA: 0s - loss: 0.5142 - acc: 0.8442
11904/12000 [============================>.] - ETA: 0s - loss: 0.5115 - acc: 0.8450
11968/12000 [============================>.] - ETA: 0s - loss: 0.5088 - acc: 0.8458
12000/12000 [==============================] - 64s 5ms/step - loss: 0.5075 - acc: 0.8462 - val_loss: 0.0074 - val_acc: 0.9983

Epoch 00001: val_acc improved from -inf to 0.99833, saving model to emojinator.h5
Epoch 2/12

   64/12000 [..............................] - ETA: 58s - loss: 0.0076 - acc: 1.0000
  128/12000 [..............................] - ETA: 57s - loss: 0.0223 - acc: 0.9922
  192/12000 [..............................] - ETA: 57s - loss: 0.0265 - acc: 0.9896
  256/12000 [..............................] - ETA: 58s - loss: 0.0215 - acc: 0.9922
  320/12000 [..............................] - ETA: 56s - loss: 0.0344 - acc: 0.9875
```

Step 4:- Recognition

We take gesture input from user, and use the model derived after training of the dataset,the label or the hand emoji according to the input gesture will be shown. The emoji will be determined with the help of model created.

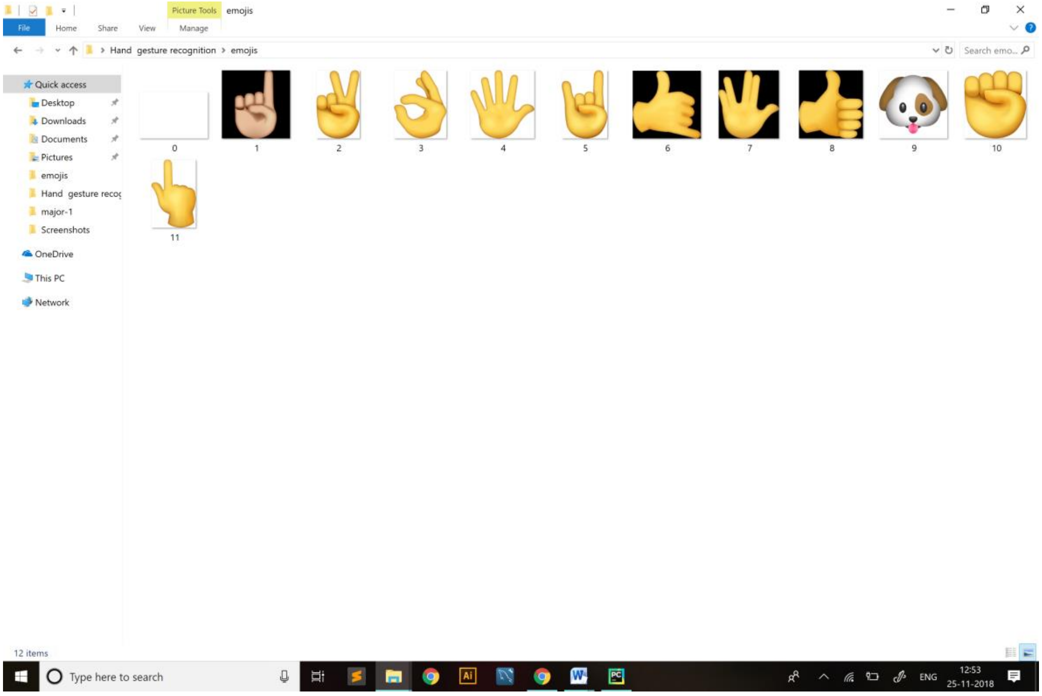


```
2.COLOR_BGR2H
p.array([2, 5
 img, mask=ma
v2.COLOR_BGR2
gray, (5, 5),

, 5), np.uint
an, kernel_sq
(dilation, cv
d(opening, 30

:x + w]
s(thresh.copy

, key=cv2.con
tour) > 2500:
.boundingRect
y:y + h1, x:x
ize(newImage,
class = keras
print(pred_class, pred_probab)
img = overlay(img, emojis[pred
```

main() › while (cap.isOpened())

```
4 0.99999285
4 0.9999937
4 0.99999297
4 0.9999826
4 0.9999862
4 0.9999856
4 0.9999856
4 0.99999034
4 0.99998415
4 0.999987
4 0.99998343
4 0.99998343
4 0.99998057
4 0.999987
```

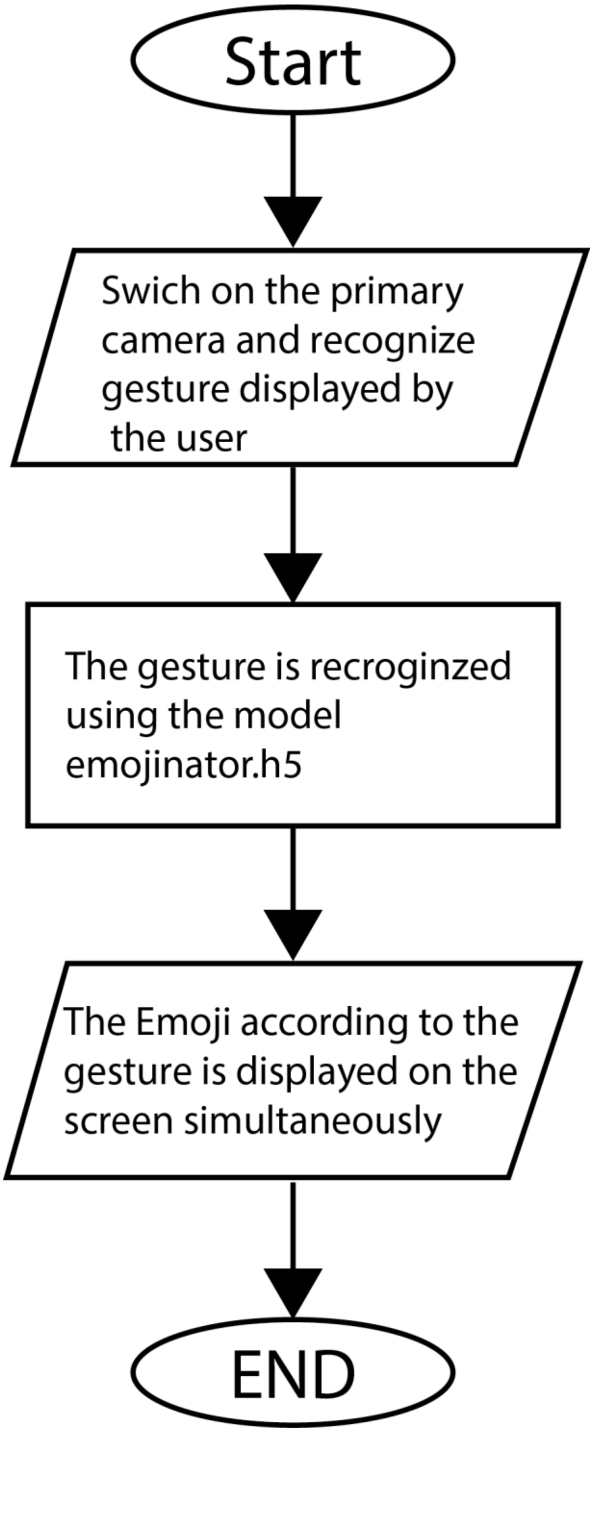
Packages installed successfully: Installed packages: 'imageio' (today 21:38)



Architecture Diagram

FlowChart

Application:


Start
Swich on the primary camera and recognize gesture displayed by the user
The gesture is recroginzed using the model emojinator.h5
The Emoji according to the gesture is displayed on the screen simultaneously
END

Backend:

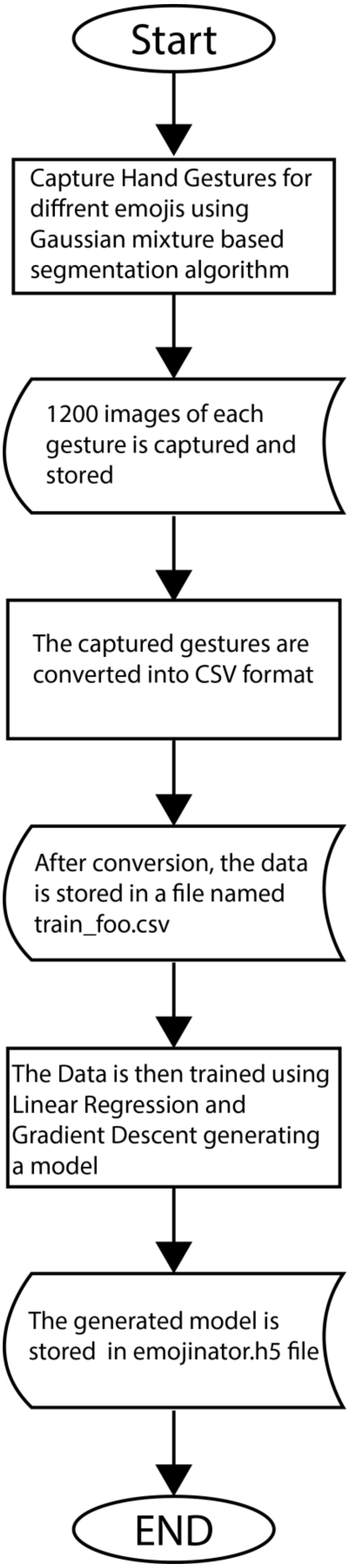
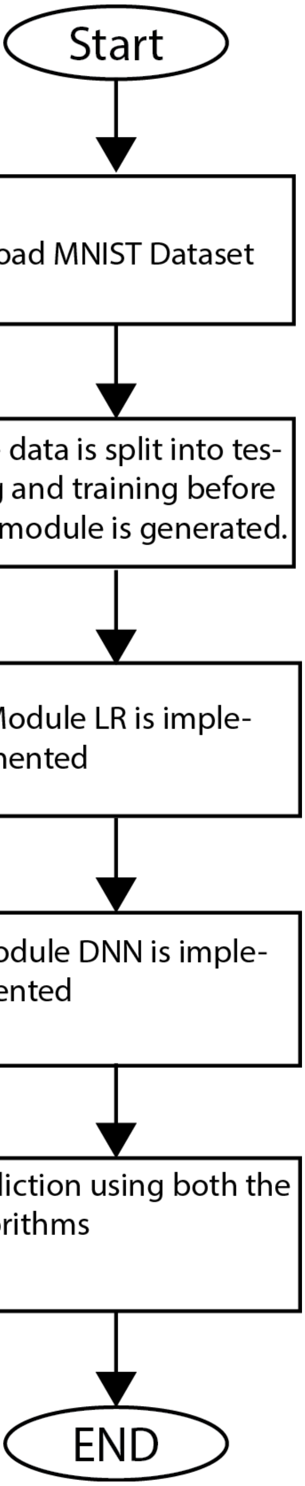
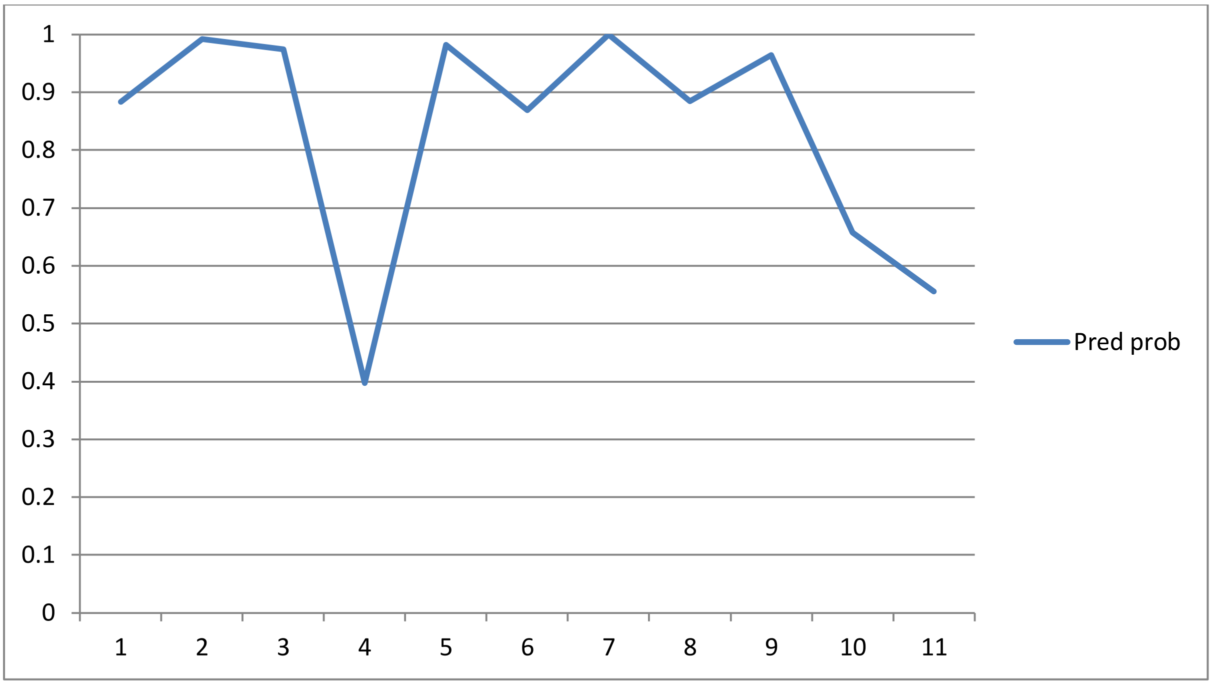

Start
Capture Hand Gestures for diffrent emojis using Gaussian mixture based segmentation algorithm
1200 images of each gesture is captured and stored
The captured gestures are converted into CSV format
After conversion, the data is stored in a file named train_foo.csv
The Data is then trained using Linear Regression and Gradient Descent generating a model
The generated model is stored in emojinator.h5 file
END

Diagram:


Start
Load MNIST Dataset
The data is split into testing and training before the module is generated.
Module LR is implemented
Module DNN is implemented
Prediction using both the algorithms
END

RESULT



Labels [Emojis]

In the above graph, I have compared all the labels/Emojis according to their prediction probability. As we can observe the prediction rate after ample amount of training is above average. for most of the labels the prediction probability is more than 80% and external factors such as proper lightining proved the result to be better.

The application is independent of any factors irrespective of size, color of the hand while we take gesture as input. The accuracy rate after training is 1.

CONCLUSION

The application can successfully determine the type of hand-emoji in run time- taking input as images of hand gesture. After the training of dataset leads to creation of model which is then used for detection of hand emojis through gesture.

The making of the application requires 4 steps over all that are creation of gesture files, convertion of the images to csv format, then the dataset are trained making a model , finally the model is used for the recognition process.

The dataset is separated into training and testing data, a particular percentage is used for both the process in our application 90% and 10% respectively.

The accuracy and probability of prediction of each class is average but are not similar the accuracy level depends on how much we have trained the model.

LIMITATION

We have limited our application to only 11 Hand-Emojis. The training of dataset is a slow process and needs better processor for fast processing.

The lighting conditions effect the application a lot. The gestures cannot be properly identified by the camera. Also the architecture of the neural network may not turn out to be correct and provide required result even after ample amount of training. Some Emojis in our application take more time to be recognized than others.

References

1. https://docs.opencv.org/3.4/db/d5c/tutorial_py_bg_subtraction.html
2. https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/
3. http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html
4. Kaew, P., 2001. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proc. 2nd European Workshop Advanced Video Based Surveillance System, 2001*.
5. Aly, A.A., Deris, S.B. and Zaki, N., 2011. Research review for digital image segmentation techniques. *International Journal of Computer Science & Information Technology*, *3*(5), p.99.
6. Miguel-Hurtado, O., Guest, R., Stevenage, S.V., Neil, G.J. and Black, S., 2016. Comparing Machine Learning Classifiers and Linear/Logistic Regression to Explore the Relationship between Hand Dimensions and Demographic Characteristics. *PloS one*, *11*(11), p.e0165521.
7. Hallén, R., 2017. A Study of Gradient-Based Algorithms.
8. https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html