DIFFERENT HAND GESTURE RECOGNITION USING ANN

by

SANYAM JAIN (20BIT0428)

TABLE OF CONTENTS

Introduction:

The necessity of Hand Gesture Recognition is having a well-defined dataset which have sufficient number of images for every hand gesture by which we can predict the hand gesture more efficiently. Hand Gesture Recognition idea begin with the invention of glove-based control interface for the computer control, then over many years this technology was upgraded and now we are able to have a smart touchless hand gesture recognition product which reduce human efforts indefinitely.

For Real-time problems like:

i) Controlling a laptop/personal mobile using hand gesture recognition technique.

ii) For the people who are deaf or dumb understanding the talk using hand gesture recognition technique.

For the controlling a laptop/personal mobile we had to make some fix hand notations by using which the user can perform that specific task, and by using notations which are the user make we can use CNN which can help us to predict the notation user wants to convey thereby for that we can have a dataset for the notations which can help our model to predict the notation more accurately, hence the computer/mobile can do the right operation which user wants. As, 3D-CNN can be useful for detecting a set of action (i.e., the action which require hand moments) which can make the system more efficient and hence provide us with many new gesture notations.

For people who are deaf or can’t speak the hand gesture recognition can help others to understand their talk with hands and them to understand the other people’s talk using hands. For this we can have some common actions which a person uses by expressing their feelings and based on that we can create some notations by which a general talk of normal people can also be understood by using this technique. This will let the disabled people to live a life like a normal human by understanding other’s and making the other’s understanding them.

As, we get the notations for the following cases we can create a dataset by including all the types of images a user can use to express a specific notation, then we can train the dataset using CNN or 3D CNN as in general a notation may have more than one actions so by using it, we can predict the notations and its meaning based on the user actions accurately.

Hand gestures are powerful human to human communication channel which convey a major part of information transfer in our everyday life. Hand gesture recognition is a process of understanding and classifying meaningful movements by the human hands.

We need a well defined dataset is the necessity which has images for hand gestures by which we can predict the hand gestures more efficiently.

The history of hand gesture recognition for computer control **started with the invention of glove-based control interfaces**. Researchers realized that gestures inspired by sign language can be used to offer simple commands for a computer interface.

Real Time Problems:-

i) In the automotive industry, this capability allows drivers and passengers to interact with the vehicle

— usually to control the infotainment system without touching any buttons or screens.

ii) Sign Language Interpreter for disabled people

i) A gesture recognition system starts with a camera pointed at a specific three-dimensional zone within the vehicle, capturing frame-by-frame images of hand positions and motions. This camera is typically mounted in the roof module or other vantage point that is unlikely to be obstructed. The system illuminates the area with infrared LEDs or lasers for a clear image even when there is not much natural light. Those images are analysed in real time by computer vision and machine learning technologies, which translate the hand motions into commands, based on a predetermined library of signs.

ii) **How does sign language recognition work**?

Identification of sign gesture is mainly performed by the following methods:

1. Glove-based method in which the signer has to wear a hardware glove, while the hand movements are getting captured.
2. Vision-based method, further classified into static and dynamic recognition.

Dataset Specification:

We are going to use a dataset which contains images of many different types of gesture (namely one finger notation gesture, two finger notation gesture, left notation gesture, etc.) and around 1000 images for each gesture it is the Hand Gesture Recognition Training Dataset. It contains images of gestures at different angles and clarity which helps our model to predict the hand gesture even in less clarity.

Literature Review (all papers considered in a team):

Review on Various Schemes:

Hand Gesture Recognition Using CNN for Post-Stroke People:

The main idea was to make a Hand Recognition system for a post-stroke or disabled people by which they can convey their message easily, for that the 3D-CNN was used and a dataset of 7 different hand gesture representing some message which we feel difficult to understand in real life. The dataset consists of hand gesture with less distance from camera and many different types of hands (young people hands, old people's hand etc.)

Citation - Alnaim, Norah, Maysam Abbod, and Abdulrahman Albar. "Hand gesture recognition using convolutional neural network for people who have experienced a stroke." *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2019.

Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation:

Recent research has confirmed the superiority of Convolutional Neural Network (CNN) for image likeness and categorization. Since, CNN can determine complex and non-linear friendships with figures, in this paper, a changeless help sign recognition system utilizing CNN was projected. Data augmentation like re-measuring, zooming, shearing, turn, breadth and height fluctuating was used to the dataset. They also pre-processed the data dataset to reduce the computational complication and achieve better efficiency.

Citation – Islam, Md Zahirul, et al. "Static hand gesture recognition using convolutional neural network with data augmentation." *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, 2019.

IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition:

They used a live cam input and 3D-CNN model to predict the hand gesture and to increase the usage of hand gestures they also introduce hand gestures in the dataset which make more than one actions to do a task as this can increase the functionality of the system. For example, throw-left, double-click, zoom-in, zoom-out etc. This dataset is named as IPN Hand with sufficient size, variety, and real- world elements able to train and evaluate deep neural networks.

Citation – Benitez-Garcia, Gibran, et al. "IPN hand: A video dataset and benchmark for real-time continuous hand gesture recognition." *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.

Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks: They secondhand a sliding dormer accompanying a stride that run through arriving broadcast frames where indicator sequence placed at the starting point of classifier queue. If the indicator understands an action/action, then the classifier is mobilized. The indicator's output is post-treated for a more robust acting, and the indispensable content is made utilizing distinct-time incitement block where alone incitement occurs per acted gesture.

Citation – Köpüklü, Okan, et al. "Real-time hand gesture detection and classification using convolutional neural networks." *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. IEEE, 2019.

Hand Recognition Using Geometric Classifiers:
Classification, verification and identification of the input images were done using two simple geometric classifiers. They have also tested system using nearest box classifier and MEB classifier. They were getting good results in MEB classifier and they used geometric classifier because of the fact, this method was designed to work on data that are heavily skewed by a preponderance of duplicate values.

Citation – Bulatov, Yaroslav, et al. "Hand recognition using geometric classifiers." *International Conference on Biometric Authentication*. Springer, Berlin, Heidelberg, 2004.

IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition:
The idea here is to make a hand gesture recognition system that can recognize all the hand gesture signs. By using the dataset, we will be recognizing the hand gesture signs while playing the video from the dataset, and also, we will be making the system efficient to recognize the

gesture in

different types of environments which can help us to avoid making a specific environment for the system to be active. Remembering the hand sign of every recognition can be difficult for the user, so we will be using some common signs that users use in their daily life and can easily be remembered.

How it is going to be done:
For Hand gesture recognition we will be using Artificial neural network technique and mainly the Convolution neural network (CNN) and for implementation we will be using python.
The dataset we will be using is IPN Hand.
The dataset contains RGB videos, which are recorded in the resolution of 640 × 480 with a frame rate of 30 fps.
We will be dividing the hand gesture recognition into two stages
i. Gesture spotting: This aims to detect temporal video segments which contain gesture instances.
ii. Gesture classification: This aims to classify the gesture of each spotted segment.

Citation - Benitez-Garcia, G., Olivares-Mercado, J., Sanchez-Perez, G., & Yanai, K. (2021, January). IPN hand: A video dataset and benchmark for real-time continuous hand gesture recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 4340-4347). IEEE.

Real-time hand tracking and gesture recognition system:
In this paper, hand gesture recognition system to recognize real time gesture in unconstrained environments. The system consists of three modules: real time hand tracking, training gesture and gesture recognition using pseudo two dimension hidden Markov models (P2-DHMMs). Using a Kalman filter and hand blobs analysis for hand tracking to obtain motion descriptors and hand region. uses skin color for hand gesture tracking and recognition. The basic idea lies in the real-time generation of gesture model for hand gesture recognition in the content analysis of video sequence from CCD camera. Since hand images are two dimensional, it is natural to believe that the 2-DHMM, an extension to the standard HMM, will be helpful and offer a great potential for analyzing and recognizing gesture patterns. The complete system works at about 25 frames/sec. If the hand moves too fast Kalman filter at this will not find hand pixels correctly, the tracker starts going out of track due to the translation and scale parameters get distorted values. Beside this if initialization is not good, the tracker can not archived the result of tracking.

Citation - Binh, Nguyen Dang, Enokida Shuichi, and Toshiaki Ejima. "Real-time hand tracking and gesture recognition system." *Proc. GVIP* (2005): 19-21.

Development of hand gesture recognition system using machine learning. Journal of Ambient Intelligence and Humanized Computing:

Hand gesture segmentation usually involves a pre processing stage that removes the unwanted noise. To remove noise, filters like morphological filter Kalman filter and other common filters like median and Gaussian filters have been used in many researches. The actual process of gesture segmentation involves separating the hand gesture from its background image. Skin colour is a very distinctive characteristic of the human hand which can be used to separate it from the background. The proposed HGR system was trained and tested using Sebastian Marcel static hand posture database (Marcel 2019). The dataset has hand gestures from 10 different persons and on complex as well as plain backgrounds. Different pictures are subjected to different illumination and scale conditions. For each gesture, it was made sure that different background and different rotation of the gesture was taken into consideration.

Algorithm:-

Hand gesture recognition involves multiple stages of processing so it will be necessary to develop suitable algorithms for each stage of the process. The main stages can be divided into: hand segmentation, feature extraction and classification.

1. The first stage of the HGR system is the image segmentation where the hand will be isolated from its background and then the image will be prepared for its further stages.
2. In this step, the colour distribution is separated into skin colour and non-skin colour and then the threshold value is calculated using the histogram method.
3. Feature extraction methodology

1. The integral of the image is taken.
2. Key points are localized using Fast-Hessian.
3. Orientation assignment is performed.
4. Descriptors are extracted from the identified key
 points Gesture classification

After successfully extracting the required features and obtaining a fixed size input vector, the classification is implemented using a multiclass support vector machine. The system was tested with a wide range of images which are rotated as well as scaled and it has shown to perform with an overall accuracy of 96.5% with a very fast recognition time of 0.024 s. This clearly shows that the HGR system is rotation and scale invariant and can accept gestures from complex backgrounds. Due

to the fast recognition time, this system can be enhanced by employing real time gesture images and developing the classifier accordingly.

Citation - Parvathy, Priyanka, et al. "Development of hand gesture recognition system using machine learning." *Journal of Ambient Intelligence and Humanized Computing* 12.6 (2021): 6793-6800.

Real-time hand tracking and gesture recognition system using neural networks:

A simple and fast algorithm using orientation histograms is developed. It will recognize a subset of static hand gestures. A pattern recognition system will be using a transform that converts an image into a feature vector, which will be compared with the feature vectors of a training set of gestures. The final system will be Perceptron implementation in MATLAB. This paper includes experiments of 33 hand postures and discusses the results. Experiments shows that the system can achieve a 90% recognition average rate and is suitable for real time applications. The dataset is Various Myanmar Alphabet Language databases on the Internet and photographs that are taken with a digital camera. This meant that they have different sizes, different resolutions and some times almost completely different angles of shooting.

Citation - Maung, Tin Hninn Hninn. "Real-time hand tracking and gesture recognition system using neural networks." *World Academy of Science, Engineering and Technology* 50 (2009): 466-470.

An automatic hand gesture recognition system based on Viola-Jones method and SVMs:
The system consists of two modules: hand gesture detection module and hand gesture recognition module. The detection module could accurately locate the hand regions with a blue rectangle; this is mainly based on Viola-Jones method, which is currently considered the fastest and most accurate learning-based method for object detection. In the recognition module, the Hu invariant moments feature vectors of the detected hand gesture are extracted and a Support Vector Machines (SVMs) classifier is trained for final recognition, due to its high generalization performance without the need to add a priori knowledge. The performance of the proposed system is tested through a series of experiments, the proposed system tries to apply Viola-Jones method to real-time hand gesture detection. The method was mainly realized by three parts

- Using a set of Haar-like features and integral image representation for feature extraction.
- During training, using AdaBoost algorithm to select the critical features among a large number of extracted features stage by stage and finally yields an extremely efficient classifier.

• Combining the strong classifiers in a cascade structure which efficiently increases the speed of the detector by focusing attention on promising regions of the image.

Citation – Yun, Liu, and Zhang Peng. "An automatic hand gesture recognition system based on Viola-Jones method and SVMs." *2009 Second International Workshop on Computer Science and Engineering*. Vol. 2. IEEE, 2009.

Comparative study on various subtitles:

(Title, Year, Authors)	**Methodology or Techniques used (Mention specific algorithms or recent technologies)**	**Advantages**	**Issues**	**Metrics used (those are used to justify the performance of the used scheme)**
Hand Gesture Recognition Using CNN for Post-Stroke People, 2019, Norah Alnaim, Abdulrahman Albar, and Maysam F Abbod	CNN	The accuracy result of training is 100% compared to that of testing and specificity in training is 100% whereas in testing is 99.89%. Its execution time is approximate 15,598 seconds, which is duration to train and test the system using seven hand gestures which are used in the experiment. Overall, training has the best values in most parameters.	It fails to identify hand gesture if the distance between hand and camera is more.	execution time, accuracy, sensitivity, specificity, positive and negative predictive value, likelihood and root mean square.
Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation , 2019, Md. Zahirul Islam, Mohammad	CNN	The model with augmented data achieved accuracy 97.12% which is nearly 4% higher than the model without augmentation (92.87%).	Recognition of gestures made with both hands is not possible by this system.	Loss, and Accuracy

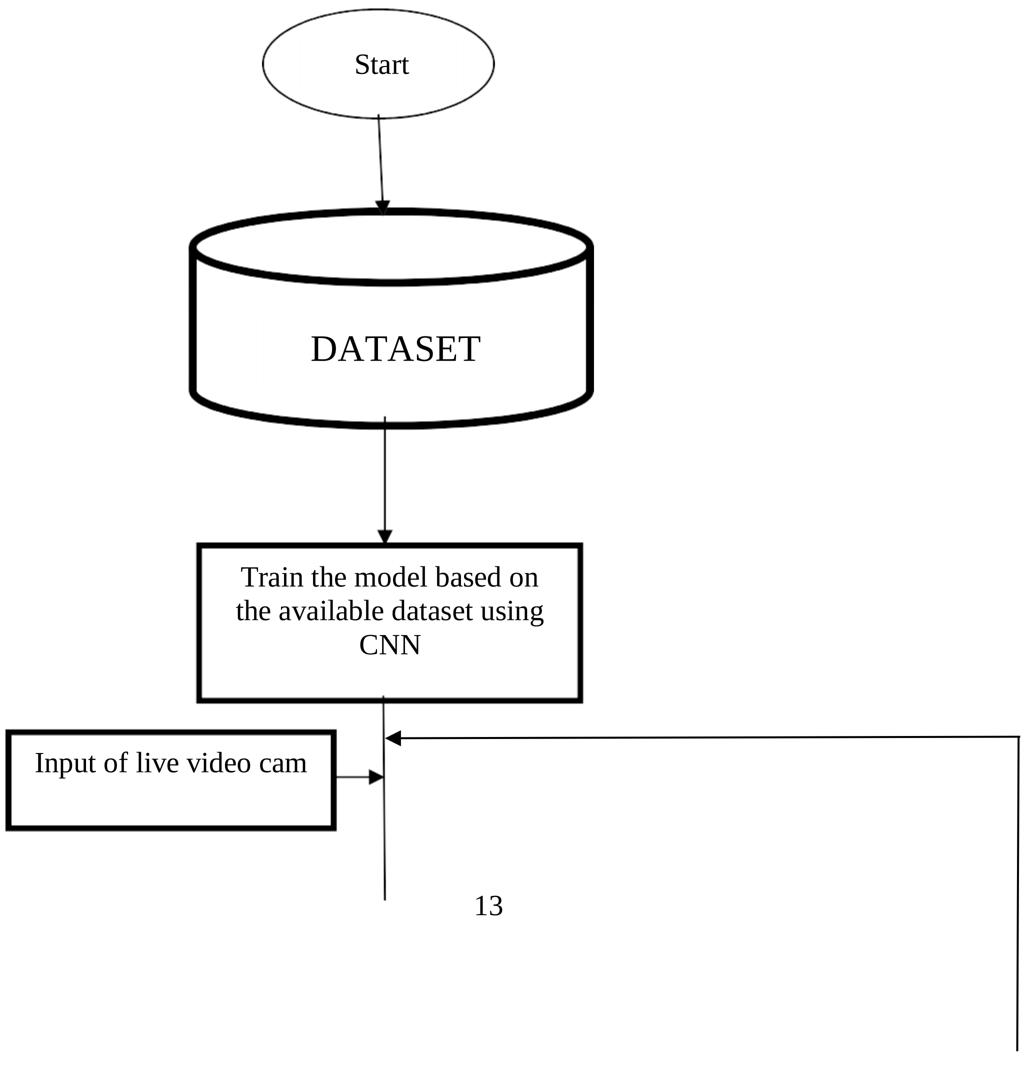
Shahadat Hossain, Raihan Ul Islam, and Karl Andersson				
IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition, 2020, Gibran Benitez-Garcia, Jesus Olivares-Mercado , Gabriel Sanchez-Perez, and Keiji Yanai	CNN	It can group the frame (i.e., more than one action) and find which notation it makes. More commonly called as complete animation instead of a picture.	The model sizes are much bigger and it requires more processor for predicting the gesture.	conventional classification, Levenshtein accuracy, and binary classification accuracy
Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks, 2019, Okan Kop ukl u, Ahmet Gunduz , Neslihan Kose , and Gerhard Rigoll	CNN	We obtain considerable early detections while achieving performances close to offline operation.	It can only predict whether the gesture is present in the image or not, nothing else.	Levenshtein distance
Hand Recognition Using Geometric Classifiers, 2004, Yaroslav Bulatov, Sachin	Geometric Classifiers	We describe a novel minimum enclosing ball classifier which performs well for hand recognition and could be of interest for other applications. Our system is easier to	It can only use for medium security applications.	$L\infty$ metric

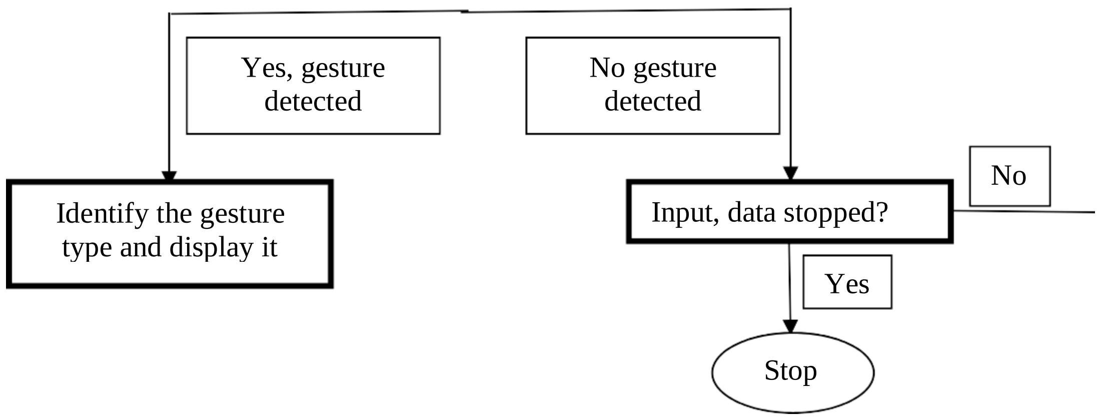
Jambawalikar, Piyush Kumar, and Saurabh Sethia1		use, cheaper to build and more accurate than previous systems.		
IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition, 2020, Gibran Benitez-Garcia, Jesus Olivares-Mercado , Gabriel Sanchez-Perez, and Keiji Yanai	Convolutional Neural Network	It can group the frame (i.e., more than one action) and find which notation it makes. More commonly called as complete animation instead of a picture.	The model sizes are much bigger and it requires more processor for predicting the gesture.	Classification Confusion Matrix of Prediction Metric and conventional classification-Levenshtein accuracy, and binary classification accuracy
Real-Time Hand Tracking and Gesture Recognition System-2009 Nguyen Dang Binh, Enokida Shuichi	Kalman filter and hand blobs analysis for hand tracking	The users do not need to wear a glove, neither is there need for a uniform background. Experiments on a single hand database have been carried out and recognition accuracy of up to 98% has been achieved.	Currently, tracking method is limited to 2D. If the hand moves too fast Kalman filter at this will not find hand pixels correctly, the tracker starts going out of track	Classification Accuracy metric
Development of hand gesture recognition system using machine learning(2020)- Priyanka Parvathy Kamalraj Subramaniam	Using median or Gaussian filters. Histogram method for skin-color detection Bag of features method for feature extraction and classification	HGR system is rotation and scale invariant and can accept gestures from complex backgrounds. Has fast recognition time	‘Stop’ gesture showed maximum misclassification. This is because of the similarity of the Stop gesture with the Spread; many gestures have been misclassified between these two classes.	Classification ROC curve Metric

"Real-time hand tracking and gesture recognition system using neural networks."-2009 Maung, Tin Hninn Hninn	Hidden Markow methods and orientation Histogram for gesture Recognition.	Another advantage of using neural networks is that you can draw conclusions from the network output. If a vector is not classified we can check its output and work out a solution	Implemented with MATLAB which tends to slow down with loops so sometimes back-tracing was avoided in code	Classification Accuracy metric
"An automatic hand gesture recognition system based on Viola-Jones method and SVMs."-Yun, Liu, and Zhang Peng.	Viola Jones Method, Haar like features sets, Adaptive boosting algorithm	It uses SVMs which always find a global minimum because it usually tries to minimize a bound on the structural risk, rather than the empirical risk.	the detection speed for the classifiers using the extended Haar-like feature is somewhat slow down due to the extra computation,	Classification Accuracy Metric

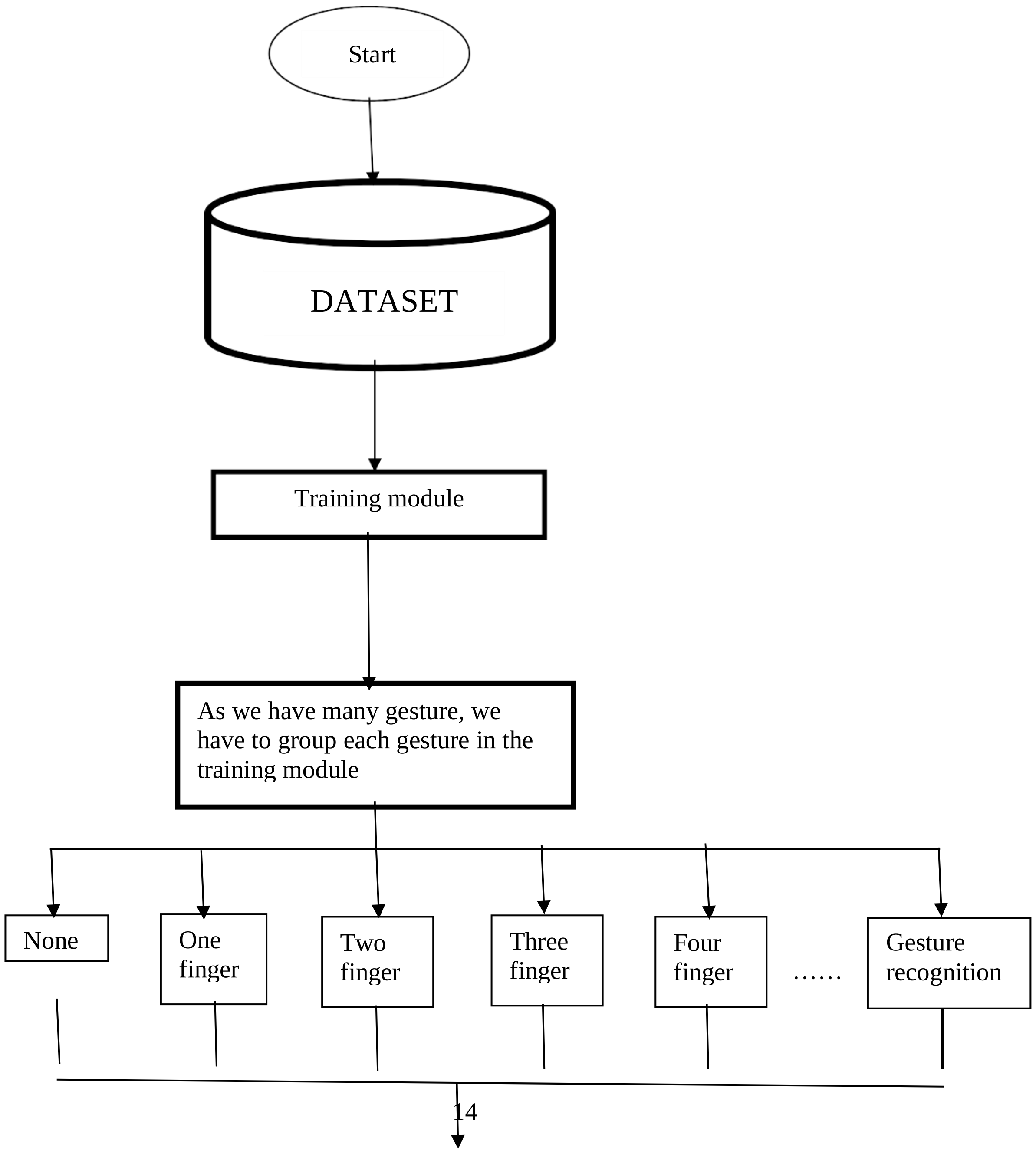
System design:

Architecture Diagram / Flow Diagram / Flowchart / …





Detailed Description of Modules



Gesture + label of
gesture added in
training model and
make a validation
model (to validate the
input)

Using CNN Model

Using CNN, find
the training and
validation accuracy
and make the most
accurate set as the
training model and
validation model.

Input of live video cam

Yes, gesture
detected

No,
ge
stu
re
no
t
de
tec
te
d

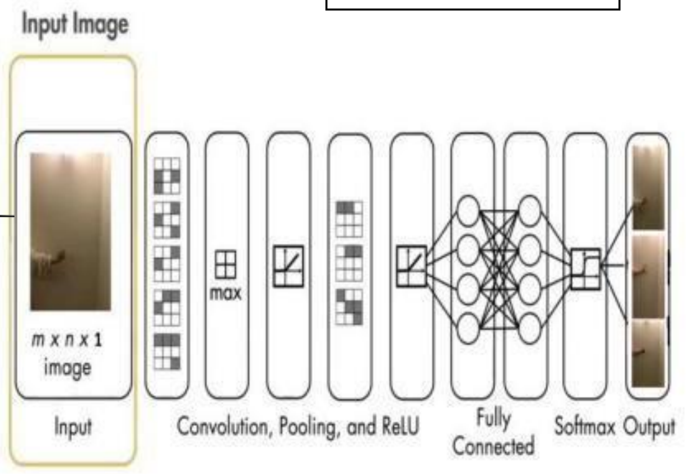
No

Input, data stopped?

Identify the gesture type
and display it

Y
e
s

Stop



Software Requirement Specifications:

Soft Computing technique: Convolutional Neural Network
(CNN) Implementation tool:
Operating system: Windows
Coding platform: Pycharm (IDLE Python)

Modules: TensorFlow, OpenCV, Keras, and NumPy.

Experimental Results & Discussion:

Source code:

traning.py:

```
# Check the accuracy

import
os
import
math
import
matplotlib.image
import tensorflow as
tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D,
MaxPool2D from keras.optimizers import Adam, SGD
from keras.preprocessing.image import
ImageDataGenerator import matplotlib.pyplot as plt
import warnings
import numpy as
np import cv2
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import ModelCheckpoint,
EarlyStopping import json

warnings.simplefilter(action='ignore',

category=FutureWarning) file = open("checkpoints.json")
data = json.load(file)

train_path =
data['paths'][0]['path_train_data'] test_path
= data['paths'][0]['path_test_data']

train_batches = ImageDataGenerator(
preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_from_directory(direct

ory
=train_path,
                                                            target_size=(64, 64),
                                                            class_mode='categori
                                                            cal',
                                                        batch_size=
                                                        10,
                                                        shuffle=Tru
                                                        e)

test_batches = ImageDataGenerator(

preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_from_directory(directo
ry
=test_path,
                                                        target_size=(64, 64),
                                                        class_mode='categori
                                                        cal', batch_size=10,
                                                        shuffle=True)

imgs, labels =
next(train_batches) labels =
os.listdir(train_path)


# Plotting the images...
def plotImages(images_arr):
  fig = plt.figure(figsize=(10,
  7)) rows =
  math.ceil(len(labels) / 2)
  cols = math.ceil(len(labels) /
  5)

  images = []
  folders =
  os.listdir(train_path) for i in
  folders:
    files = os.listdir(os.path.join(train_path, i))
    paths = [os.path.join(train_path, os.path.join(i, filename)) for filename in
    files] images.append(max(paths, key=os.path.getctime))

  for i in images:
    img = matplotlib.image.imread(i)
    img = cv2.cvtColor(img,
    cv2.COLOR_BGR2RGB) fig.add_subplot(rows,
    cols, images.index(i) + 1) plt.imshow(img)
    plt.axis('off')
    plt.title(labels[images.index(
    i)])

  plt.subplots_adjust(hspace=0.5, wspace=0)
  plt.savefig("Result.png", bbox_inches='tight')
```

```
    plt.show()


plotImages(i
mgs)
print(imgs.sha
pe)
print(labels)

word_dict = {}
```

```
for i in range(0,
    len(labels)):
    word_dict[i] =
    labels[i]

model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(64, 64, 3)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(len(labels), activation="softmax"))

model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0,
mode='auto')

model.compile(optimizer=SGD(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0005)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0,
mode='auto')

history2 = model.fit(train_batches, epochs=10, callbacks=[reduce_lr,
                    early_stop], validation_data=test_batches) # , checkpoint])
imgs, labels = next(train_batches) # For getting next batch of imgs...

imgs, labels = next(test_batches) # For getting next batch of imgs...
scores = model.evaluate(imgs, labels, verbose=0)
print(model.metrics_names, scores)
print(f'{model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of {scores[1] *

100}%') model.save('model.h5')
```

```
print(history2.history)

imgs, labels = next(test_batches)
```

```
model = keras.models.load_model(r"model.h5")

scores = model.evaluate(imgs, labels, verbose=0)
print(f'{model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of {scores[1] *
100}%')

scores # [loss, accuracy] on test data...
model.metrics_names

predictions = model.predict(imgs, verbose=0)
print("predictions on a small set of test data--
") print("")
for ind, i in enumerate(predictions):
    print(word_dict[np.argmax(i)], end=' ')

# plotImages(imgs)
print('Actual
labels') for i in
labels:
    print(word_dict[np.argmax(i)], end=' ')

print(imgs.shape)

# Model accuracy graph
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accurac
y']) plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper
left') plt.show()

# Model loss graph
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_los
s']) plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper
left') plt.show()


history2.history
```

main.py:

```
import os
import numpy as np
import mediapipe as
mp import cv2
```

```
import
keras
import
json

file =
open('checkpoints.json')
data = json.load(file)

model =

keras.models.load_model(data['paths'][0]['path_model'])

labels = os.listdir(data['paths'][0]['path_train_data'])

background = None
accumulated_weight =
0.5

ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left =

350

word_dict =

{}

for i in range(0,
    len(labels)):
    word_dict[i] =
    labels[i]


def cal_accum_avg(frame,
    accumulated_weight): global background

    if background is None:
        background =
        frame.copy().astype("float") return None

    cv2.accumulateWeighted(frame, background, accumulated_weight)


def segment_hand(frame,
    threshold=25): global background

    diff = cv2.absdiff(background.astype("uint8"), frame)

    _, thresholded = cv2.threshold(frame, threshold, 255, cv2.THRESH_BINARY_INV)

    contours, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) ==
        0: return
        (thresholded)
    else:
        hand_segment_max_cont = max(contours,
        key=cv2.contourArea) return (thresholded,
        hand_segment_max_cont)


cam = cv2.VideoCapture(0,
cv2.CAP_DSHOW) num_frames = 0

while True:
    ret, frame = cam.read()

    # filpping the frame to prevent inverted image of captured frame...
    frame = cv2.flip(frame, 1)

    frame_copy =

    frame.copy() # ROI

    from the frame
    roi = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]

    gray_frame = cv2.cvtColor(roi,
    cv2.COLOR_BGR2GRAY) gray_frame =
    cv2.GaussianBlur(gray_frame, (9, 9), 0)

    if num_frames < 70:
        cal_accum_avg(gray_frame, accumulated_weight)

        cv2.putText(frame_copy, "FETCHING BACKGROUND...PLEASE WAIT", (80, 400),
cv2.FONT_HERSHEY_SIMPLEX, 0.9,
                    (0, 0, 255), 2)

    else:
        # segmenting the hand region
        hand = segment_hand(gray_frame)

        if len(hand) == 2:
            thresholded, hand_segment = hand

            # Drawing contours around hand segment
            cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1, (255, 0, 0), 1)

        else:
            thresholded = hand

        cv2.imshow("Thesholded Hand Image",
        thresholded) thresholded =

        cv2.resize(thresholded, (64, 64))
        thresholded = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2RGB)
        thresholded = np.reshape(thresholded, (1, thresholded.shape[0], thresholded.shape[1], 3))
```

```
        pred = model.predict(thresholded)
        cv2.putText(frame_copy, word_dict[np.argmax(pred)], (170, 45),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    # Draw ROI on frame_copy
    cv2.rectangle(frame_copy, (ROI_left, ROI_top), (ROI_right, ROI_bottom), (255, 128, 0), 3)

    # incrementing the number of frames for
    tracking num_frames += 1

    # Display the frame with segmented hand
    cv2.putText(frame_copy, "DataFlair hand sign recognition_ _ _", (10, 20),
cv2.FONT_ITALIC, 0.5, (51, 255, 51), 1)
    cv2.imshow("Sign Detection", frame_copy)

    # Close windows with
    Esc k = cv2.waitKey(1)
    & 0xFF

    if k ==
        27:
        brea
        k

# Release the camera and destroy all the
windows cam.release()
cv2.destroyAllWindows()
```

create_gesture_data.py:

```
import
os
import
cv2
import
glob
import
json
import numpy as
np import
tensorflow as tf

background = None
accumulated_weight =
0.5

ROI_top = 100
ROI_bottom = 300
```

```
ROI_right = 150
ROI_left = 350

file =
open('checkpoints.json')
data = json.load(file)

test_path =
data['paths'][0]['path_test_data'] path =
data['paths'][0]['path_train_data']
```

```
traindata = glob.glob(path + "\*")
noOfGestures = len(traindata)
print("Number of Gestures already available: ", noOfGestures)


def cal_accum_avg(frame,
    accumulated_weight): global background

    if background is None:
        background =
        frame.copy().astype("float") return None

    cv2.accumulateWeighted(frame, background, accumulated_weight)


def segment_hand(frame,
    threshold=25): global background

    diff = cv2.absdiff(background.astype("uint8"), frame)

    _, thresholded = cv2.threshold(diff, threshold, 255,

    cv2.THRESH_BINARY) # Grab the external contours for the image
    contours, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if len(contours)
        == 0: return
        None
    else:
        hand_segment_max_cont = max(contours,
        key=cv2.contourArea) return thresholded,
        hand_segment_max_cont


cam =

cv2.VideoCapture(0)

num_frames = 0
element = 10
num_imgs_taken
= 0 st = ''
res = []

def get_gesture_name(path):
    n = input("Enter the Gesture
    name: ") path = path + "\\" + n
    if not
        os.path.exists(path):
        os.mkdir(path)
    if not os.path.exists(os.path.join(test_path, n)):
            os.mkdir(os.path.join(test_path,
        n)) return n

    name =

    get_gesture_name(path)

    while True:
        ret, frame = cam.read()

        # flipping the frame to prevent inverted image of captured frame...
        frame = cv2.flip(frame, 1)

        frame_copy = frame.copy()

        roi = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]

        gray_frame = cv2.cvtColor(roi,
        cv2.COLOR_BGR2GRAY) gray_frame =
        cv2.GaussianBlur(gray_frame, (9, 9), 0)

        if num_frames < 60:
            cal_accum_avg(gray_frame,
            accumulated_weight) if num_frames <= 59:
                cv2.putText(frame_copy, "FETCHING BACKGROUND...PLEASE WAIT", (80, 400),
    cv2.FONT_HERSHEY_SIMPLEX, 0.9,
                        (0, 0, 255), 2)
                # cv2.imshow("Sign Detection",frame_copy)

        # Time to configure the hand specifically into the
        ROI... elif num_frames <= 299:

            hand = segment_hand(gray_frame)

            cv2.putText(frame_copy, "Adjust hand...Gesture for" + str(element), (200,
    400), cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (0, 0, 255), 2)

            # Checking if hand is actually detected by counting number of contours
            detected... if hand is not None:
                thresholded, hand_segment = hand

                # Draw contours around hand segment
                cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1, (255, 0, 0), 1)

                # Also display the thresholded image
                cv2.imshow("Thresholded Hand Image",
        thresholded)
    else:
            # Segmenting the hand region...
            hand =
            segment_hand(gray_frame)

            # Checking if we are able to detect the
            hand... if hand is not None:

                # unpack the thresholded img and the
                max_contour... thresholded, hand_segment =
                hand

                # Drawing contours around hand segment
                cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1, (255, 0, 0), 1)

                cv2.putText(frame_copy, str(num_frames), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
                (0,
    0, 255), 2)
                # cv2.putText(frame_copy, str(num_frames)+"For" + str(element), (70,
    45), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
                cv2.putText(frame_copy, str(num_imgs_taken + 1) + 'images For' + str(element), (200,
                        400), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

                # Displaying the thresholded image
                cv2.imshow("Thresholded Hand Image",
                thresholded) if num_imgs_taken <= 299:
                    cv2.imwrite(path + "\\" + name + "\\gest" + str(noOfGestures) + "_" +
                    str(num_imgs_taken
    + 1) + '.jpg',
                            thresholded)
                else:
                    break
                num_imgs_taken
            += 1 else:
                cv2.putText(frame_copy, 'No hand detected...', (200, 400),
    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        # Drawing ROI on frame copy
        cv2.rectangle(frame_copy, (ROI_left, ROI_top), (ROI_right, ROI_bottom), (255, 128, 0), 3)

        cv2.putText(frame_copy, "DataFlair hand sign recognition_ _ _", (10, 20),
    cv2.FONT_ITALIC, 0.5, (51, 255, 51), 1)

        # increment the number of frames for
        tracking num_frames += 1

        # Display the frame with segmented hand
        cv2.imshow("Sign Detection", frame_copy)
```

Closing windows with Esc key...(any other key with ord can be used too.) k = cv2.waitKey(1) & 0xFF

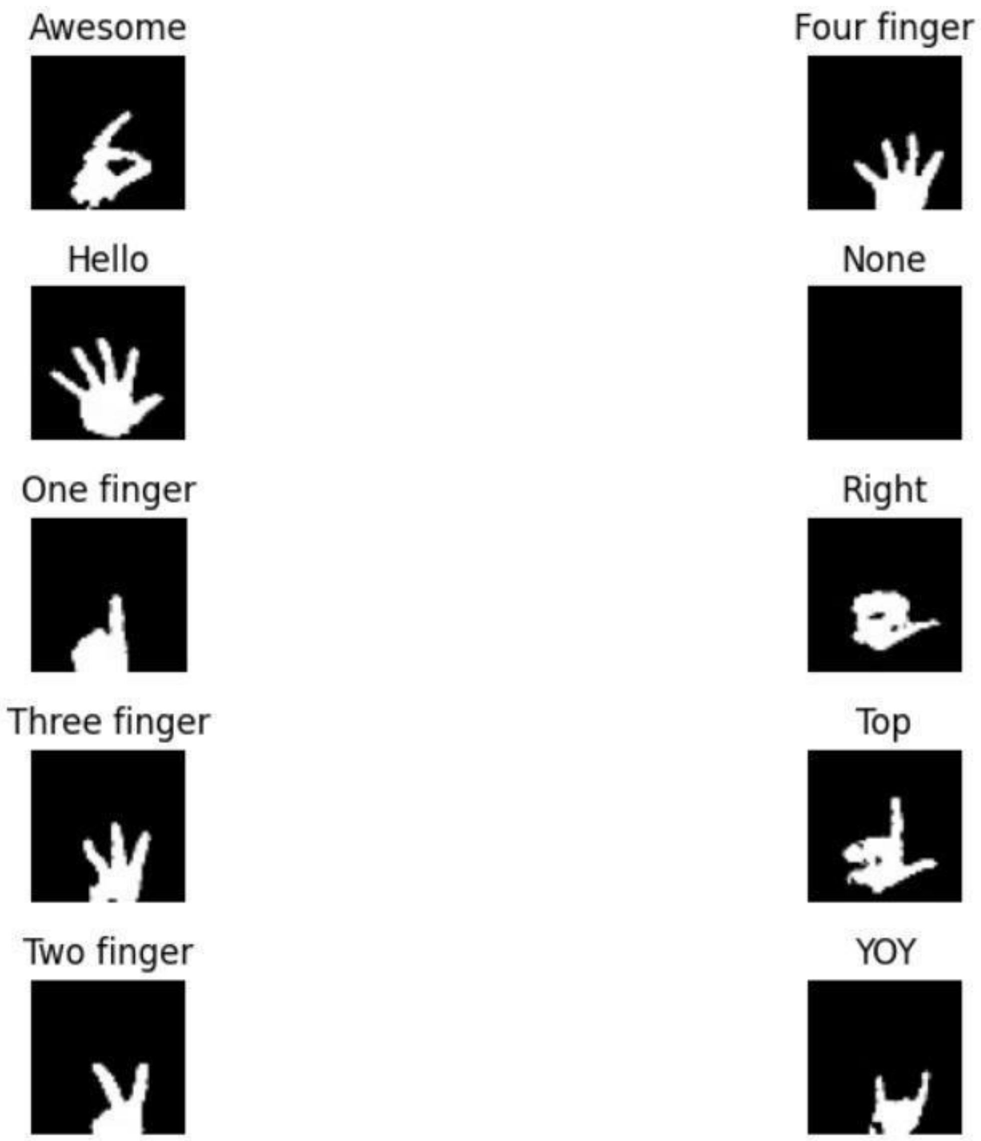
if k == 27:

```
        break

# Releasing camera & destroying all the

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

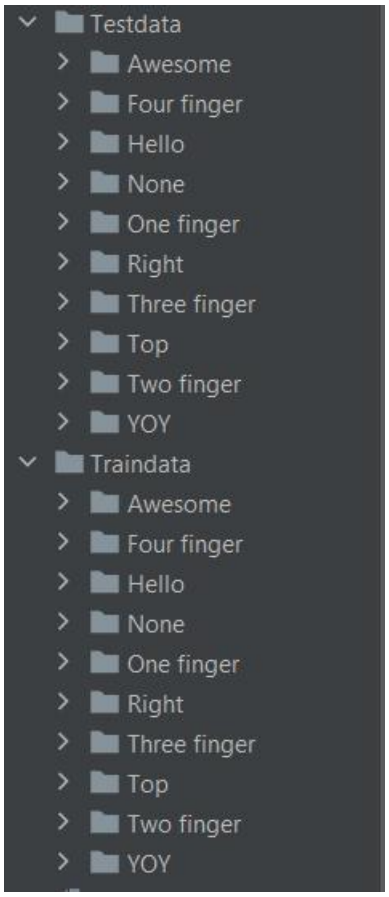
checkpoint.json:

```
{
  "paths": [
    {
      "path_train_data": "E:/PyCharm/PycharmProjects/SC-Project/Different-Hand-
Gesture- Recognition-Using-CNN/Traindata",
      "path_test_data": "E:/PyCharm/PycharmProjects/SC-Project/Different-Hand-
Gesture- Recognition-Using-CNN/Testdata",
      "path_model": "E:/PyCharm/PycharmProjects/SC-Project/Different-Hand-Gesture-
Recognition- Using-CNN/model.h5",
      "path_dir": "E:/PyCharm/PycharmProjects/SC-Project/Different-Hand-Gesture-
Recognition- Using-CNN"
    }
  ]
}
```

Hand Gesture available in Dataset:



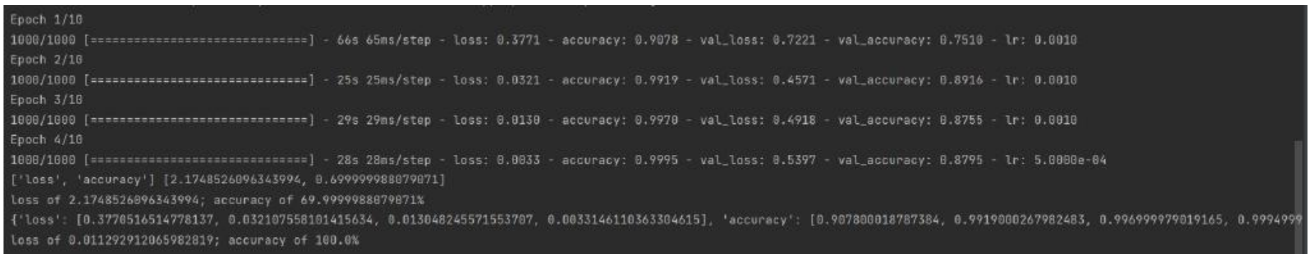
Traindata and Testdata folders:



```
Testdata
    Awesome
    Four finger
    Hello
    None
    One finger
    Right
    Three finger
    Top
    Two finger
    YOY
Traindata
    Awesome
    Four finger
    Hello
    None
    One finger
    Right
    Three finger
    Top
    Two finger
    YOY
```

Screenshots with Explanation:

We have run the training.py file which train our dataset and we have made the estimation that from the best initial epochs, once the accuracy is reached approximately to 100 then we will just stop the training by which we can save some time and the model can be made faster and the loss was also made lesser for the model.



```
Epoch 1/10
1000/1000 [==============================] - 66s 65ms/step - loss: 0.3771 - accuracy: 0.9078 - val_loss: 0.7221 - val_accuracy: 0.7510 - lr: 0.0010
Epoch 2/10
1000/1000 [==============================] - 25s 25ms/step - loss: 0.0321 - accuracy: 0.9919 - val_loss: 0.4571 - val_accuracy: 0.8916 - lr: 0.0010
Epoch 3/10
1000/1000 [==============================] - 29s 29ms/step - loss: 0.0130 - accuracy: 0.9970 - val_loss: 0.4918 - val_accuracy: 0.8755 - lr: 0.0010
Epoch 4/10
1000/1000 [==============================] - 28s 28ms/step - loss: 0.0033 - accuracy: 0.9995 - val_loss: 0.5397 - val_accuracy: 0.8795 - lr: 5.0000e-04
['loss', 'accuracy'] [2.1748526096343994, 0.699999988079071]
loss of 2.1748526096343994; accuracy of 69.9999988079071%
{'loss': [0.3770516514778137, 0.032107558101415634, 0.013048245571553707, 0.0033146110363304615], 'accuracy': [0.907800018787384, 0.9919000267982483, 0.996999979019165, 0.9994999
loss of 0.011292912065982819; accuracy of 100.0%
```

The graphs plotted for accuracy and loss form the epochs are:

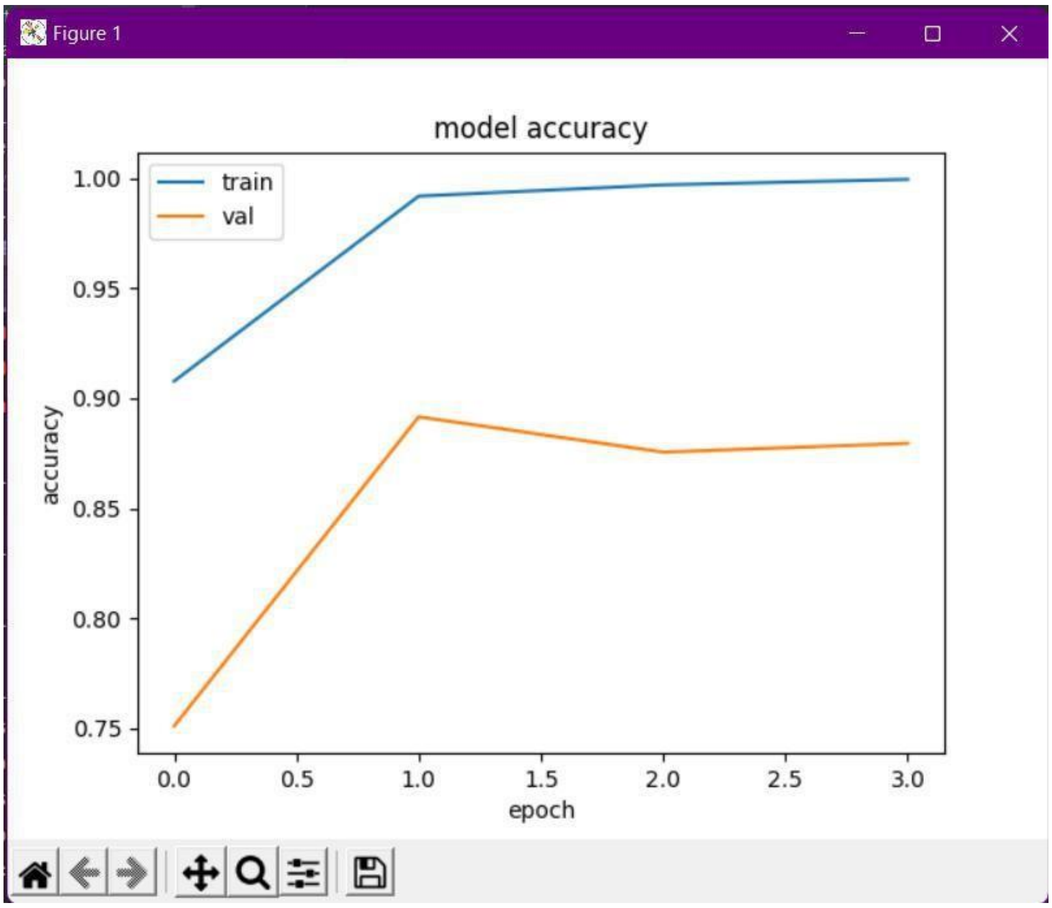
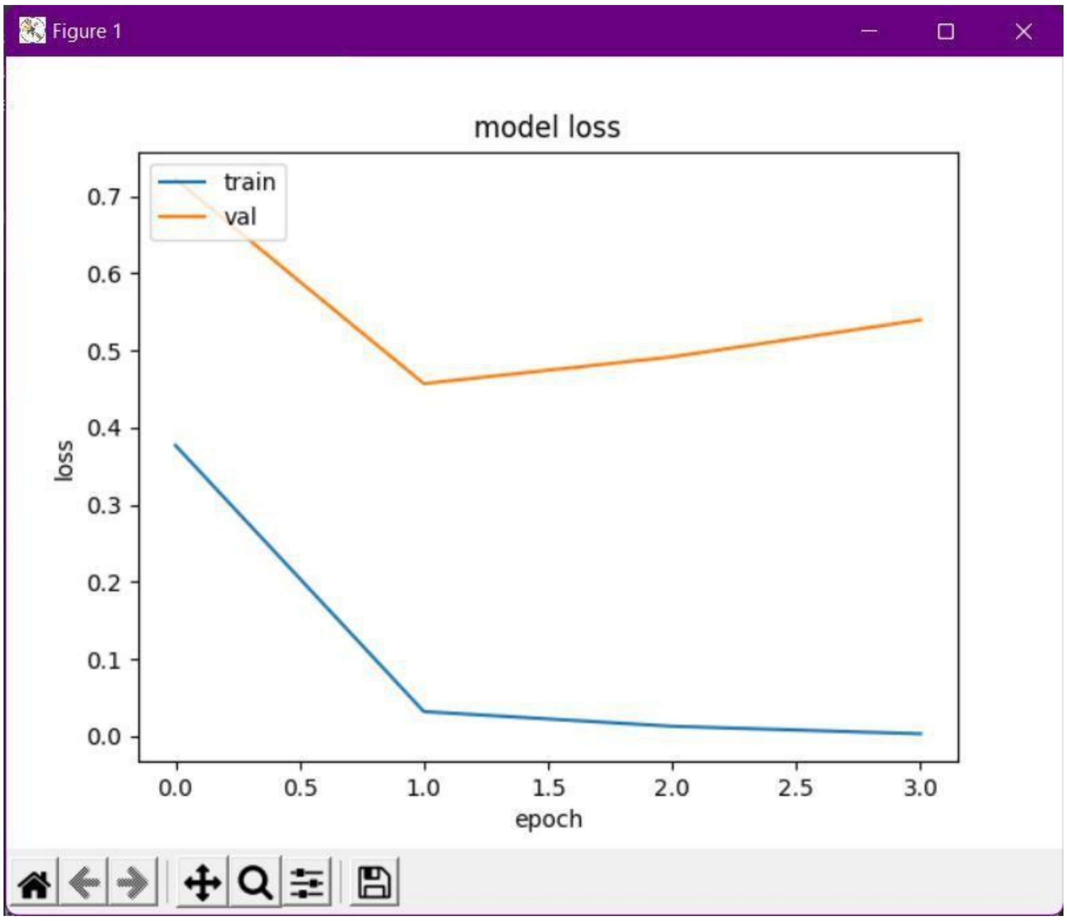
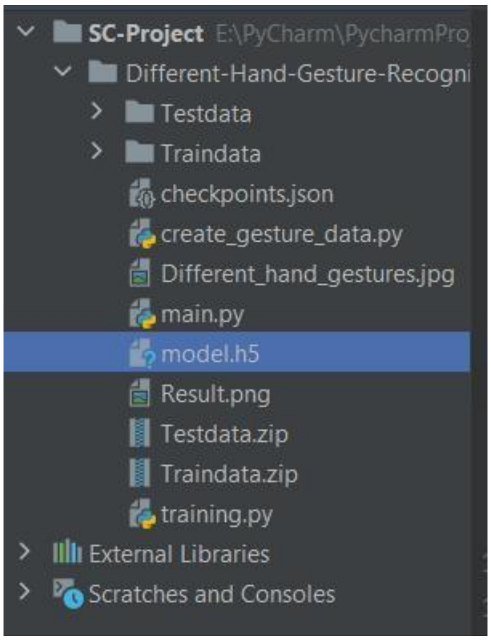


Figure 1
model accuracy
train
val
accuracy
epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.0
0.5
1.0
1.5
2.0
2.5
3.0



So, the graph represents the accuracy and loss for the training and testing of the model.

We save the training model as .h5 file such that we can directly use it to predict the gesture in the main.py instead of training it again in the main.py file

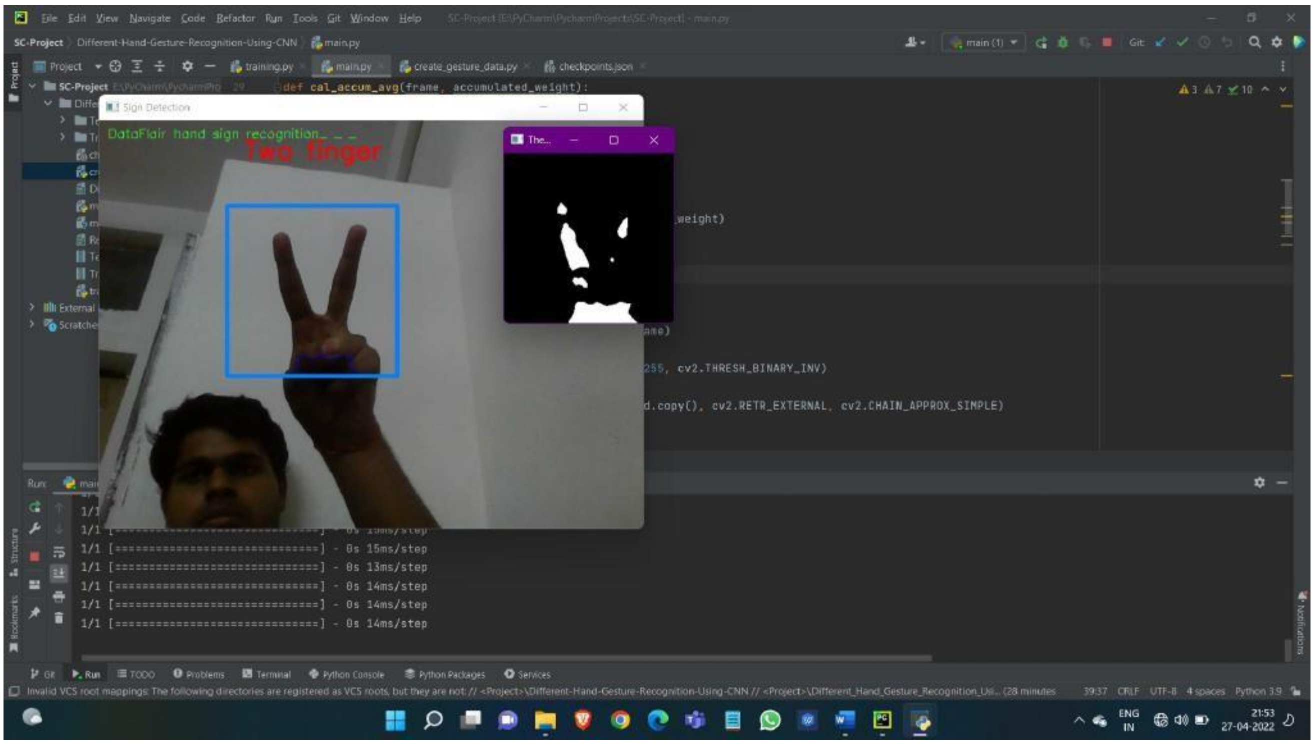


```
SC-Project E:\PyCharm\PycharmPro
  Different-Hand-Gesture-Recogni
    Testdata
    Traindata
    checkpoints.json
    create_gesture_data.py
    Different_hand_gestures.jpg
    main.py
    model.h5
    Result.png
    Testdata.zip
    Traindata.zip
    training.py
External Libraries
Scratches and Consoles
```

The predictions screenshot from the model

are:

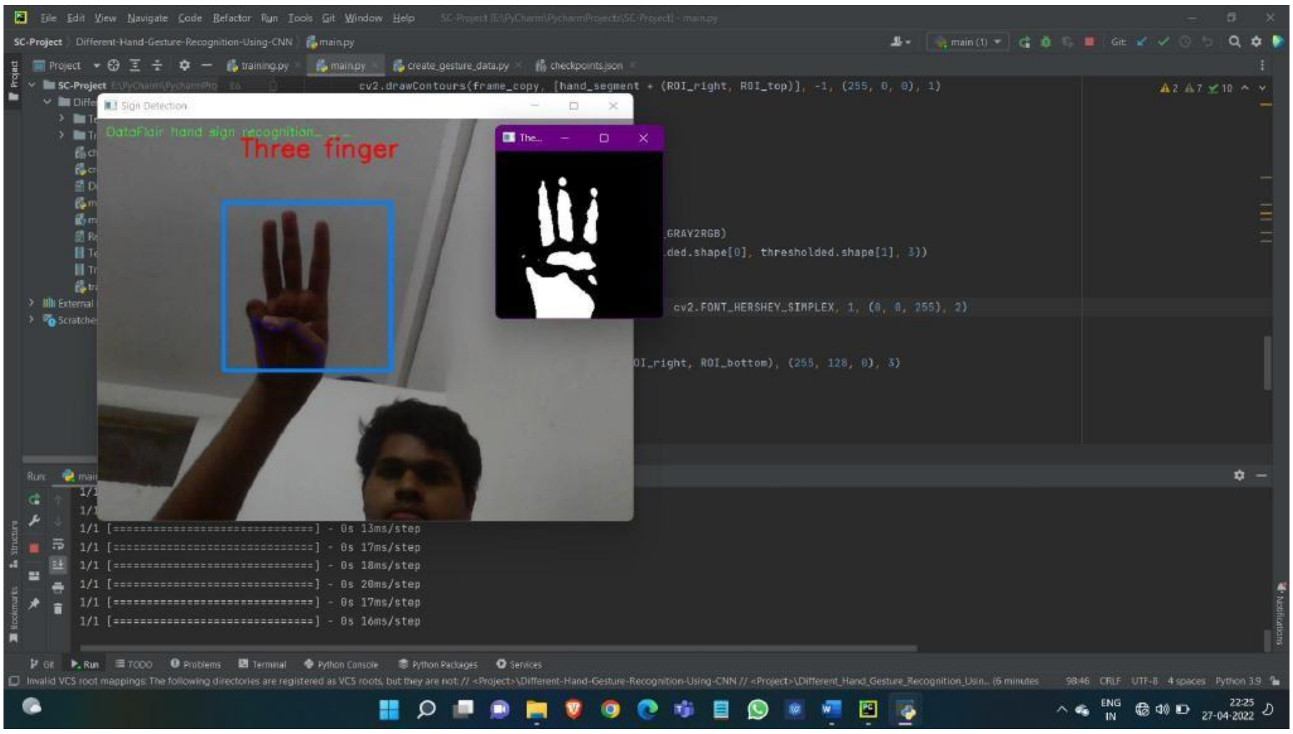
Two finger:



```
def cal_accum_avg(frame, accumulated_weight):
weight)
ame)
255, cv2.THRESH_BINARY_INV)
d.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
```

Three finger:



```
cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1, (255, 0, 0), 1)
GRAY2RGB)
ded.shape[0], thresholded.shape[1], 3))
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
OI_right, ROI_bottom), (255, 128, 0), 3)
```

```
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
```

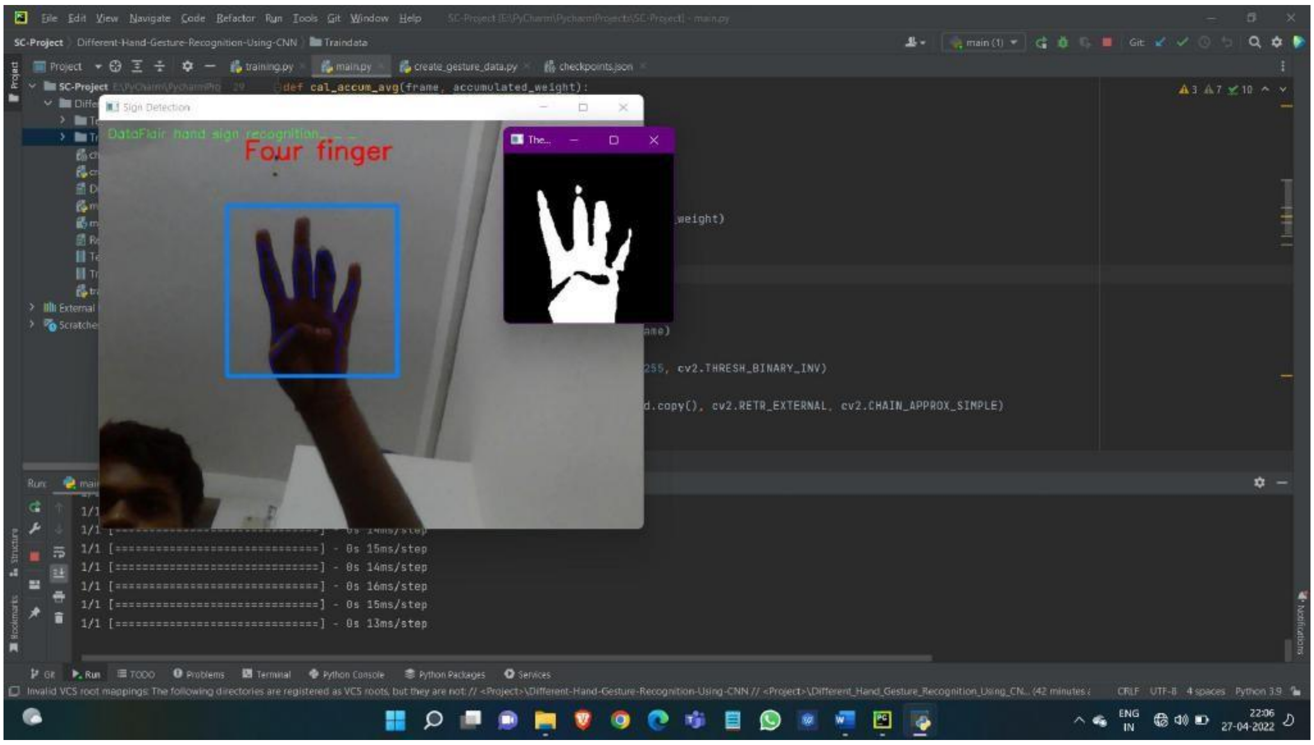
Fo

ur

fin

ger

:

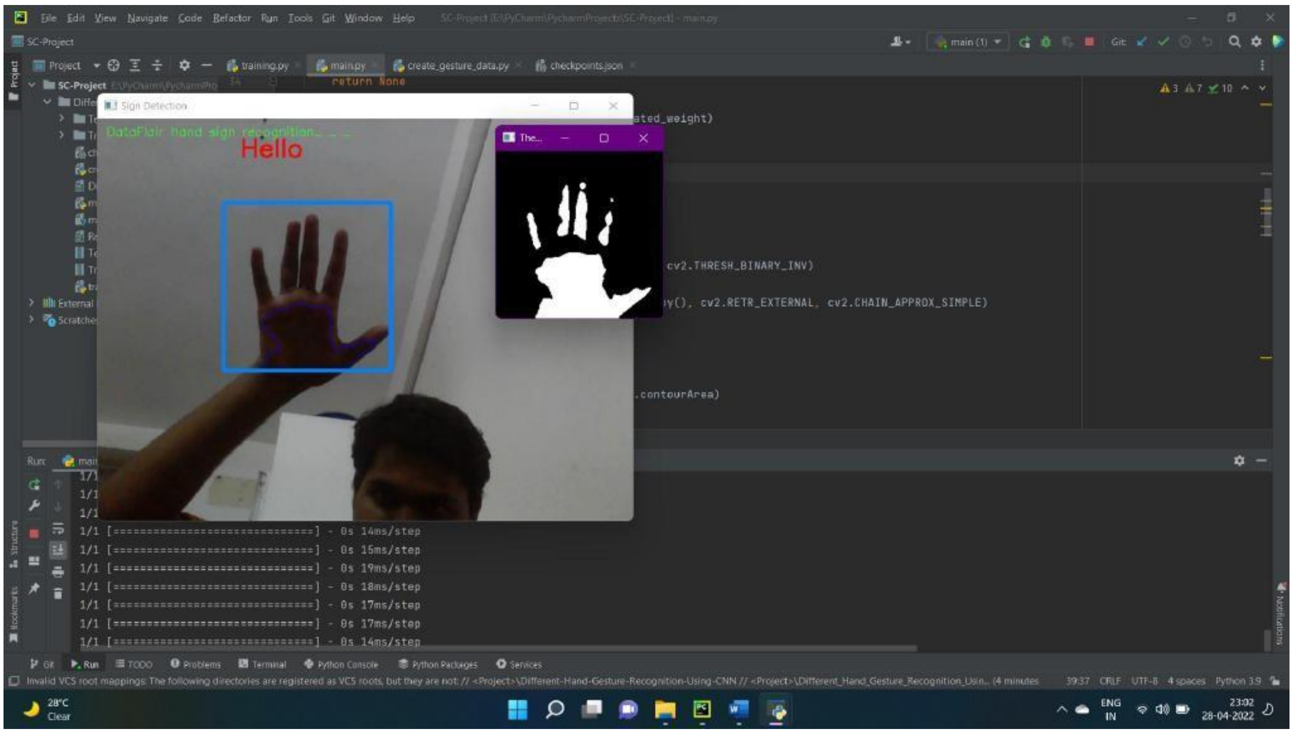


```
def cal_accum_avg(frame, accumulated_weight):
weight)
ame)
255, cv2.THRESH_BINARY_INV)
d.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 13ms/step
```

Invalid VCS root mappings: The following directories are registered as VCS roots, but they are not: // <Project>\Different-Hand-Gesture-Recognition-Using-CNN // <Project>\Different_Hand_Gesture_Recognition_Using_CN... (42 minutes a

Hello:

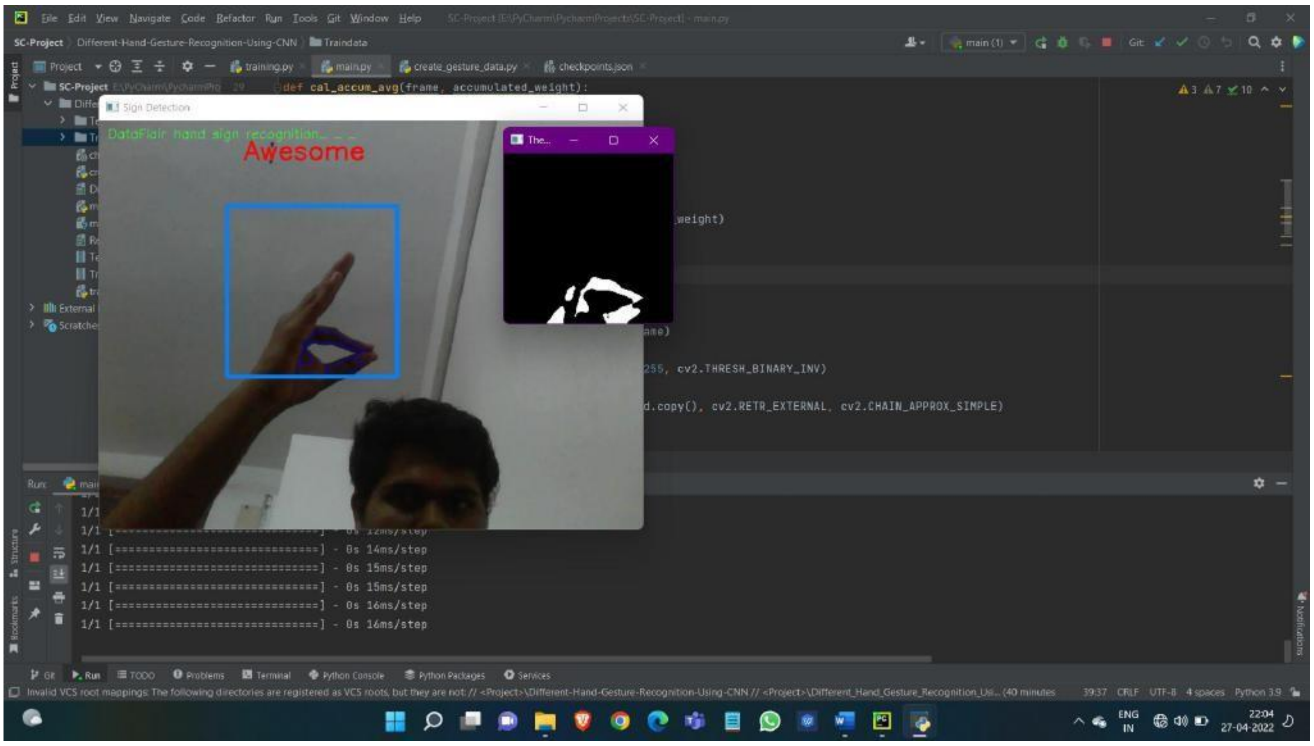


```
    return None
ated_weight)
cv2.THRESH_BINARY_INV)
y(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
.contourArea)
```

```
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 14ms/step
```

Invalid VCS root mappings: The following directories are registered as VCS roots, but they are not: // <Project>\Different-Hand-Gesture-Recognition-Using-CNN // <Project>\Different_Hand_Gesture_Recognition_Usin... (4 minutes

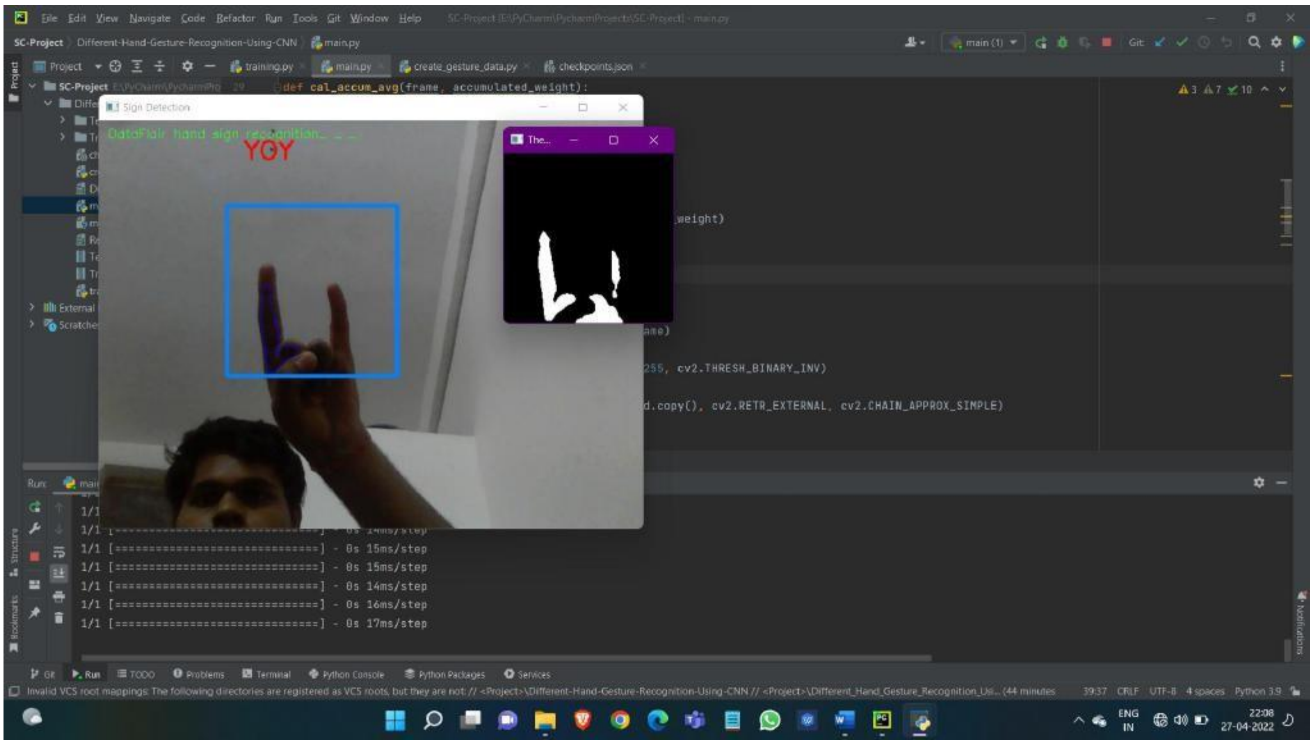
Awesome:



```
def cal_accum_avg(frame, accumulated_weight):
weight)
ame)
255, cv2.THRESH_BINARY_INV)
d.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
1/1
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
```

YOY:

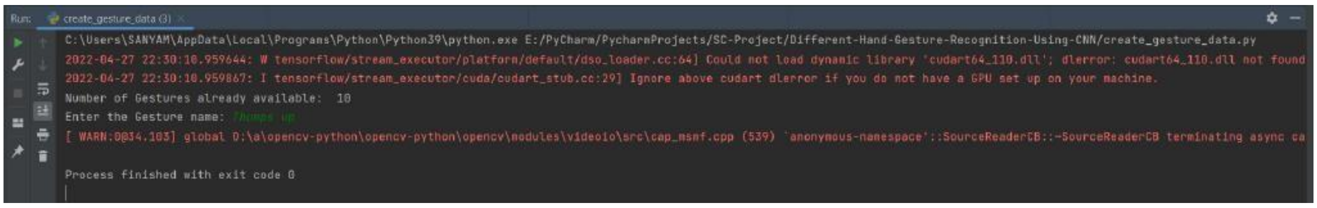


```
def cal_accum_avg(frame, accumulated_weight):
weight)
ame)
255, cv2.THRESH_BINARY_INV)
d.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

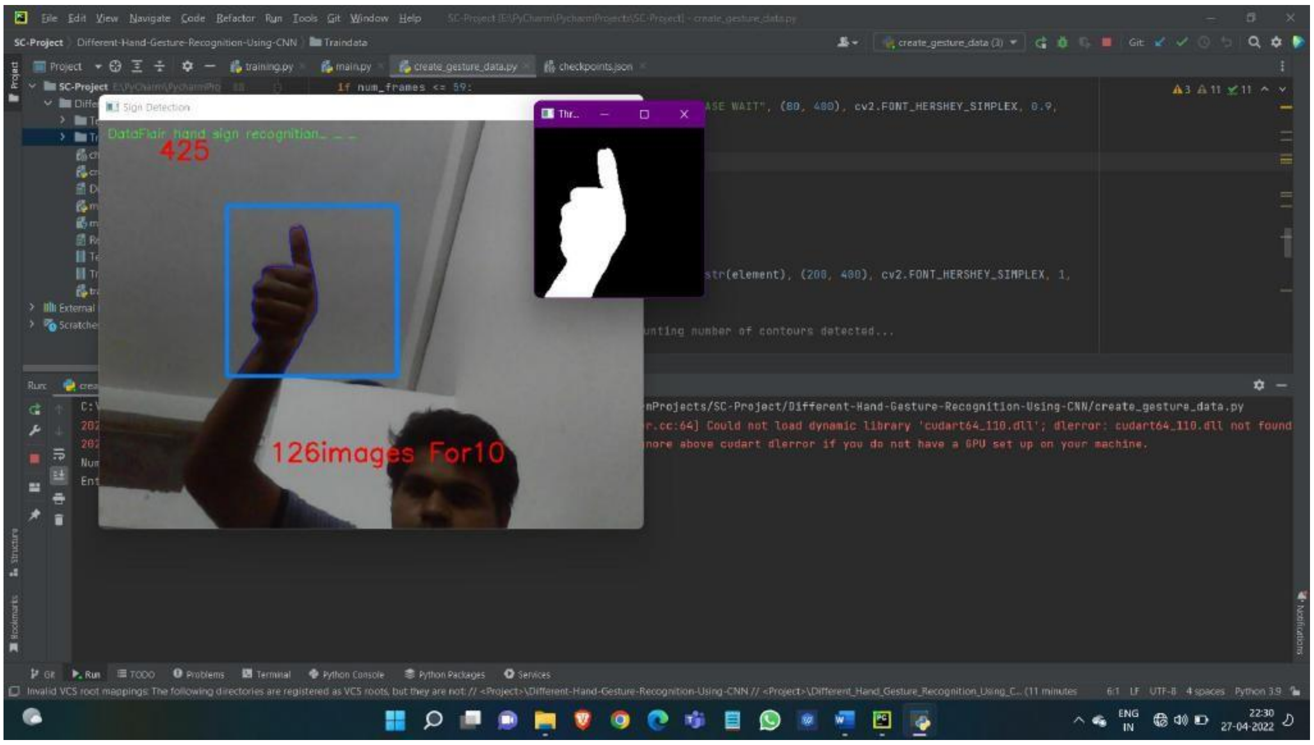
```
1/1
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
```

Due to background disturbances the threshold image for the hand gesture (The one in right window) was not clear but our model was able to predict the correct sign based on the training.

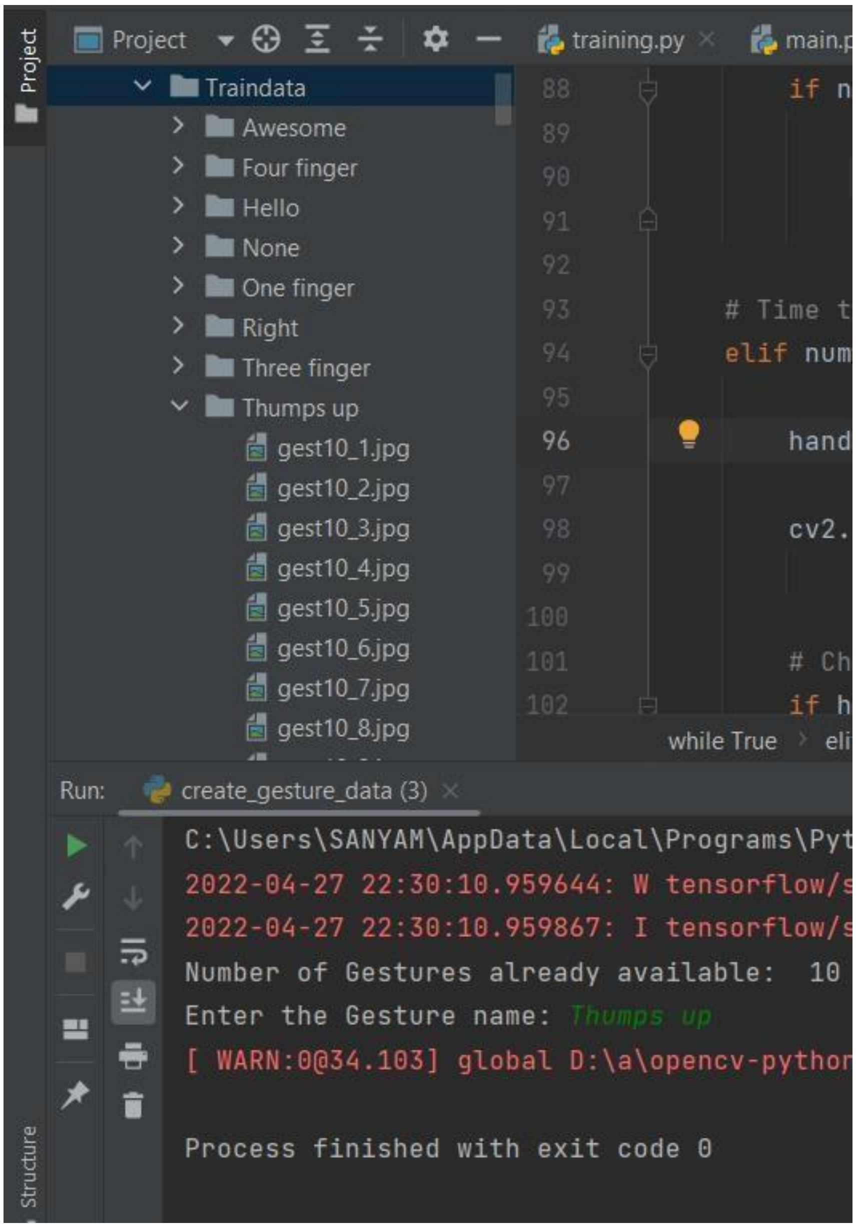
In the create_gesture_data.py we are asking the user to add a data in the dataset for that we will open a video cam, firstly the user has to enter the name of gesture (sign) such that specific folder for that can be create in the dataset and then user has to show the sign which he want to add in that folder for some time by which we can get approximately 300 images for that gesture and store it in the dataset.



```
C:\Users\SANYAM\AppData\Local\Programs\Python\Python39\python.exe E:/PyCharm/PycharmProjects/SC-Project/Different-Hand-Gesture-Recognition-Using-CNN/create_gesture_data.py
2022-04-27 22:30:10.959644: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-04-27 22:30:10.959867: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Number of Gestures already available:  10
Enter the Gesture name: Thumbs up
[ WARN:0@34.103] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (539) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async ca

Process finished with exit code 0
```



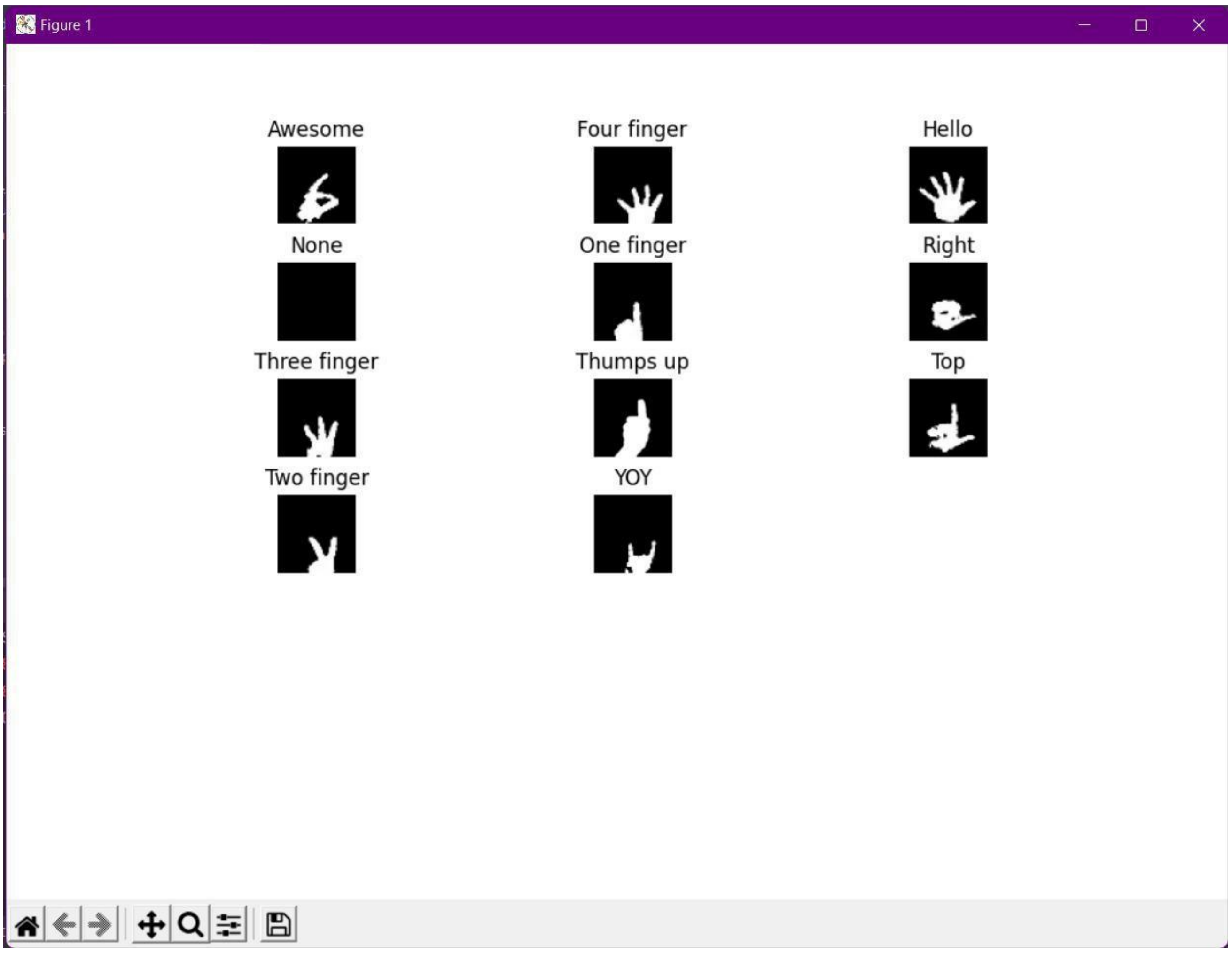
The images will be simultaneous added in the folder newly created by the program.

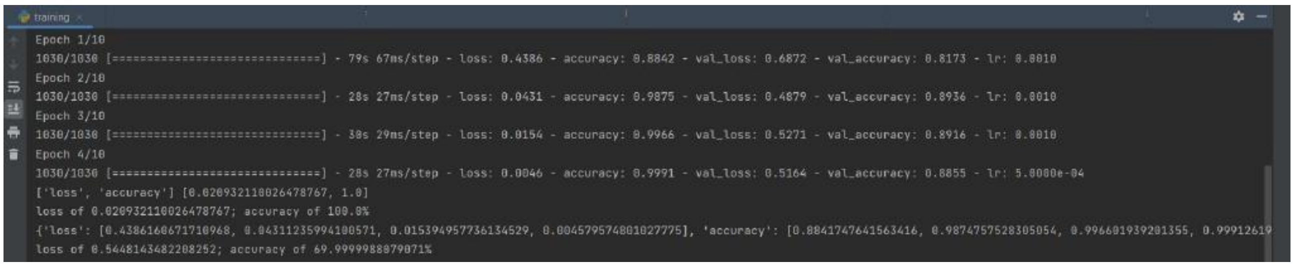


```
        if n


    # Time t
    elif num

        hand

        cv2.


        # Ch
        if h
```

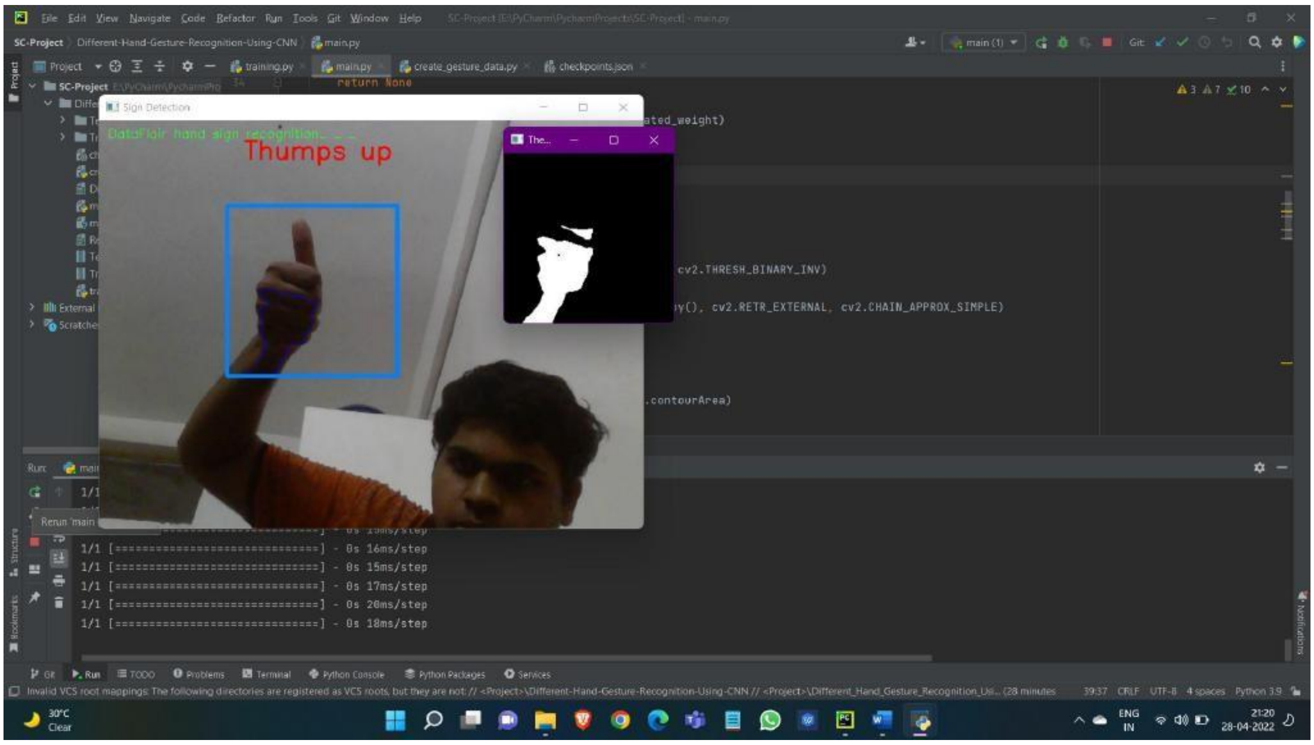
```
C:\Users\SANYAM\AppData\Local\Programs\Pyt
2022-04-27 22:30:10.959644: W tensorflow/s
2022-04-27 22:30:10.959867: I tensorflow/s
Number of Gestures already available:  10
Enter the Gesture name: Thumps up
[ WARN:0@34.103] global D:\a\opencv-pythor

Process finished with exit code 0
```

Now let’s train the model and try out with this new gesture.





```
Epoch 1/10
1030/1030 [==============================] - 79s 67ms/step - loss: 0.4386 - accuracy: 0.8842 - val_loss: 0.6872 - val_accuracy: 0.8173 - lr: 0.0010
Epoch 2/10
1030/1030 [==============================] - 28s 27ms/step - loss: 0.0431 - accuracy: 0.9875 - val_loss: 0.4879 - val_accuracy: 0.8936 - lr: 0.0010
Epoch 3/10
1030/1030 [==============================] - 30s 29ms/step - loss: 0.0154 - accuracy: 0.9966 - val_loss: 0.5271 - val_accuracy: 0.8916 - lr: 0.0010
Epoch 4/10
1030/1030 [==============================] - 28s 27ms/step - loss: 0.0046 - accuracy: 0.9991 - val_loss: 0.5164 - val_accuracy: 0.8855 - lr: 5.0000e-04
['loss', 'accuracy'] [0.020932110026478767, 1.0]
loss of 0.020932110026478767; accuracy of 100.0%
{'loss': [0.4386160671710968, 0.04311235994100571, 0.015394957736134529, 0.004579574801027775], 'accuracy': [0.8841747641563416, 0.9874757528305054, 0.996601939201355, 0.99912619
loss of 0.5448143482208252; accuracy of 69.9999988079071%
```



```
return None
ated_weight)
cv2.THRESH_BINARY_INV)
y(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
.contourArea)
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
```

Invalid VCS root mappings: The following directories are registered as VCS roots, but they are not: // <Project>\Different-Hand-Gesture-Recognition-Using-CNN // <Project>\Different_Hand_Gesture_Recognition_Usi... (28 minutes

References:

[1] Hand Gesture Recognition Using CNN for Post-Stroke People, 2019, Norah Alnaim, Abdulrahman Albar, and Maysam F Abbod

[2] Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation, 2019, Md. Zahirul Islam, Mohammad Shahadat Hossain, Raihan Ul Islam, and Karl Andersson

[3] IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition, 2020, Gibran Benitez-Garcia, Jesus Olivares-Mercado , Gabriel Sanchez-Perez, and Keiji Yanai

[4] Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks, 2019, Okan Kop ukl u, Ahmet Gunduz , Neslihan Kose , and Gerhard Rigoll

[5] Hand Recognition Using Geometric Classifiers, 2004, Yaroslav Bulatov, Sachin Jambawalikar, Piyush Kumar, and Saurabh Sethia1

[6] IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition, 2020, Gibran Benitez-Garcia, Jesus Olivares-Mercado , Gabriel Sanchez-Perez, and Keiji Yanai

[7] Real-Time Hand Tracking and Gesture Recognition System-2009 Nguyen Dang Binh, Enokida Shuichi

[8] Development of hand gesture recognition system using machine learning(2020)- Priyanka Parvathy Kamalraj Subramaniam

[9] "Real-time hand tracking and gesture recognition system using neural networks."- 2009 Maung, Tin Hninn Hninn

[10] "An automatic hand gesture recognition system based on Viola-Jones method and SVMs."- Yun, Liu, and Zhang Peng.