*A Mini Project Report on*

# Hand Sign Language Detection using Deep Learning

**B.E. - I.T**
**Engineering**

**Submitted By**

**Atharv Joshi 19104036**

**Anjali Singh 20204006**

**AbhayPratap Singh 19104037**

**DEPARTMENT OF INFORMATION TECHNOLOGY**
A.P.SHAH INSTITUTE OF TECHNOLOGY
G.B. Road, Kasarvadavali, Thane (W), Mumbai-400615
UNIVERSITY OF MUMBAI

**Academic Year : 2020-21**

# CERTIFICATE

This to certify that the Mini Project report on Intrusion Detection System using PIR sensor and ESP32 Camera has been submitted by **Atharv Joshi** (19104036), **Anjali Singh** (20204006) and **AbhayPratap Singh** (19104037) who are bonafide students of A.P Shah Institute of Technology, Thane, Mumbai, as a partial fulfillment of the requirement for the degree in **Information Technology**, during the academic year **2022-2023** in the satisfactory manner as per the curriculum laid down by University of Mumbai.

| | |
|---|---|
| Prof. Sonal Balpande | Dr. Kiran Deshpande |
| Subject In-charge | Head Department of Information Technology |

## TABLE OF CONTENTS

# Chapter 1

# Introduction

In this world, there is nothing like equality but all we can do is try and make this place a little bit live-able and bearable. One of the things in our control is how we help other people around us. For this, we should start small but starting is the key for change. Normal human beings do not have much difficulty interacting with each other and can express themselves easily through speech, gestures, body language, reading, writing, speech being widely used among them. Deaf and mute people are perfectly capable of almost every task that a normal human can perform. To bring them on the level of the layman and also give them an opportunity to shine is one of the first steps in making this world an equal place. One of the ways in which they communicate through sign-language. This is something that has to be taught to them from a very young age. This is also something that is very exclusive to them rather a normal person is usually not proficient or even familiar with sign language. At times like these, it is really difficult to find a translator and there is only so much that universal signs can do. Hand signs are an effective form of human-to-human communication that has a number of possible applications.

## Purpose

Deaf and mute people cannot communicate in a way that physically abled people can. There are limited ways of communication for the deaf and mute, especially when they are young and/or are not capable of writing things but know sign language. We need a way for them to communicate and try to help them be more sociable. We want to incorporate everyone into our society but it is not possible when there are some physical limitations. The problem comes when the differently abled people have no way to communicate, especially when they are not literate in written language but know sign language. We plan on making a real-time hand sign detection system which will detect and display the Indian sign language on the screen as it is detected.

### Objectives

These are the objectives that we hope to satisfy:

1. To detect hand sign language and display the letter on the screen. This will help a person who does not know how to sign to see what the deaf and mute person is trying to convey through the sign language.

2. To make use of leading machine learning algorithms to detect and recognize hand signs. This will be done through the use of CNN for building, training and testing the model which is faster and more efficient.

3. To teach young children how to sign and evaluate by checking their form. Children who want to learn how to sign can check their form against this model to check how they perform.

4. To use this anytime, anywhere. This will be done with the help of cloud hosting the model. Thus, it will give us the ability to use the same model for multiple devices/modules.

## Scope

Communication through signs has consistently been a significant way for communication among hearing and speech impaired humans, generally called deaf and dumb. It is the only mode of communicating for such individuals to pass on their messages to other human beings, and hence other humans need to comprehend their language. In this project, sign language detection or recognition web framework is proposed with the help of image processing. This application would help in recognizing Sign Language. The dataset used is the Indian Sign Language dataset. This application could be used in schools or any place, which would make the communication process easier between the impaired and non-impaired people. The proposed method can be used for the ease of recognition of sign language. The method used is Deep Learning for image recognition and the data is trained using Convolution Neural Network. Using this method, we would recognize the gesture and predict which sign is shown on the system.

# Chapter 2

## Review of Literature

| Sr. No | Research Paper | Finding from Paper |
|---|---|---|
| 1. | Indian Sign Language recognition system using SURF with SVM and CNN. | We learned that sign language can be recognized using a webcam and different types of algorithms. For e.g., SVM - Support Vector Machine, CNN- Convolutional Neural Network and their accuracy while detecting objects on the webcam. |
| 3. | Sign Language Recognition Based on Machine Learning | This work showed us how machine learning can be applied in the process of Sign Language Recognition. This study proposed an online use of gesture-based communication acknowledgment utilizing American Sign Language (ASL). The proposed online application will assist with eliminating the correspondence hole by being an instructional exercise to learn and figure out the gesture-based communication. In this, we have utilized a dataset of 57,000 pictures for both testing and preparing. Calculations, for example, Na ̈ive Bayes calculation, Support Vector Machine (SVM), k-Nearest Neighbors (KNN) and Convolutional Neural Network (CNN) are utilized for preparing the dataset and acquiring results. |

| | | |
|---|---|---|
| 3. | Real-Time Hand Detection using Convolutional Neural Networks for Costa Rican Sign Language Recognition | Used a video dataset to train their model and use it to detect the sign language in real-time. From this work we learned how to implement our hand sign language detection in real-time. |

# Chapter 3

## Problem Definition

People affected by speech impairment rely only on sign language, which makes it more difficult for them to communicate with the remainder of the majority. This implies a requirement for sign language recognition system which can recognize and convert sign language into spoken or written language and vice versa. Such identifiers, however, are limited, costly, and cumbersome to use [1]. The problem statement goes as follows:” User wants to use sign language to converse with a deaf and mute person but the user is not familiar with the sign language.” Proposed Solution: The solution we propose is making a real time hand sign language detection Software using CNN for training and testing the Model created.

# Chapter 4

## Proposed System

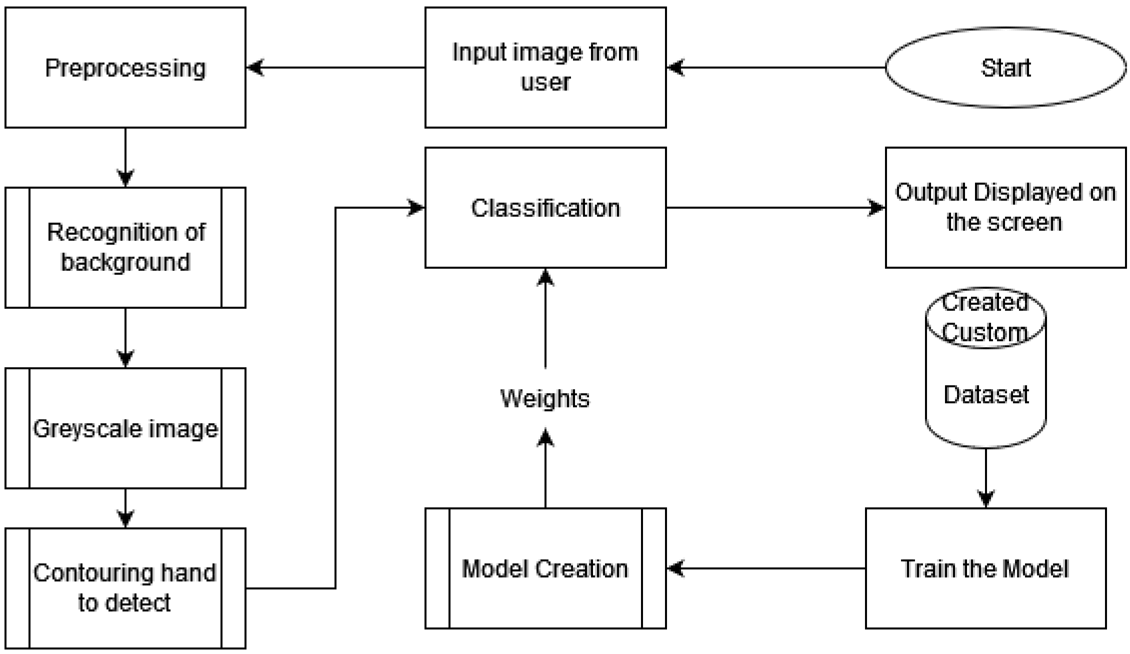


**Fig. Proposed system architecture**

## Features and Functionality:

1. Hand sign language and display the letter on the screen
2. Real time detection of the alphabets.
3. Background interference is removed.

# Chapter 5
# Software Requirements

## Python:

The libraries used in python are:

- OpenCV
- Mediapipe
- Tensorflow
- Keras
- Numpy
- Matplotlib

# Chapter 6

## Implementation

### Creation of Dataset:

```
import cv2
import numpy as np

background = None
accumulated_weight = 0.5

ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350


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 = "a"
num_imgs_taken = 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 = 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 <= 300:

        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)

            cv2.putText(frame_copy, str(num_frames)+"For" + str(element), (70, 45),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

            # 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) + '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 <= 300:
                cv2.imwrite(r"C:\\gesture\\train\\"+str(element)+"\\" + str(num_imgs_taken+300) + '.jpg',
thresholded)
                print("Yess")
                # cv2.imwrite(r"C:\\gesture\\x"+"\\" + str(num_imgs_taken) + '.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()
```

**Training of CNN**

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D,
Dropout
from keras.optimizers import Adam, SGD
from keras.metrics import categorical_crossentropy
from keras.preprocessing.image import ImageDataGenerator
import itertools
import random
import warnings
import numpy as np
import cv2
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import ModelCheckpoint, EarlyStopping
from matplotlib import pyplot as plt
warnings.simplefilter(action='ignore', category=FutureWarning)


train_path = r'C:\Users\athar\Downloads\dataset/Train'
test_path = r'C:\Users\athar\Downloads\dataset/Test'

train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_from
_directory(directory=train_path, target_size=(64,64), class_mode='categorical',
batch_size=10,shuffle=True)
test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input).flow_from
_directory(directory=test_path, target_size=(64,64), class_mode='categorical', batch_size=10,
shuffle=True)

imgs, labels = next(train_batches)
```

```
#Plotting the images...
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(30,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()


plotImages(imgs)
print(imgs.shape)
print(labels)

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(Dropout(0.2))
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.3))
model.add(Dense(35,activation ="softmax"))


# In[23]:


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_generator(train_batches, epochs=25, 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(f'{model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of {scores[1]*100}%')


#model.save('best_model_dataflair.h5')
model.save('best_model_dataflair1.h5')

print(history2.history)

imgs, labels = next(test_batches)

model = keras.models.load_model(r"best_model_dataflair1.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}%')

model.summary()

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


word_dict = {0:'One', 1:'Two', 2:'Three', 3:'Four', 4:'Five', 5:'Six', 6:'Seven', 7:'Eight', 8:'Nine', 9:'A',
10:'B', 11: 'C', 12:'D', 13:'E', 14:'F', 15:'G', 16:'H',
    17:'I', 18:'J', 19:'K', 20:'L', 21:'M', 22:'N', 23:'O', 24:'P', 25:'Q', 26:'R', 27:'S', 28:'T', 29:'U', 30:'V',
31:'W', 32:'X', 33:'Y', 34:'Z'}

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)

history2.history
```

### Testing with live Data

```
import numpy as np
import cv2
import keras
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
import urllib.request

model = keras.models.load_model(r"best_model_dataflair1.h5")

word_dict = {0:'One', 1:'Two', 2:'Three', 3:'Four', 4:'Five', 5:'Six', 6:'Seven', 7:'Eight', 8:'Nine', 9:'A',
10:'B', 11: 'C', 12:'D', 13:'E', 14:'F', 15:'G', 16:'H',
    17:'I', 18:'J', 19:'K', 20:'L', 21:'M', 22:'N', 23:'O', 24:'P', 25:'Q', 26:'R', 27:'S', 28:'T', 29:'U', 30:'V',
31:'W', 32:'X', 33:'Y', 34:'Z'}

background = None
accumulated_weight = 0.5

ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350



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)

    #Fetching contours in the frame (These contours can be of hand or any other object in foreground) ...
    contours, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # If length of contours list = 0, means we didn't get any contours...
    if len(contours) == 0:
        return None
    else:
        # The largest external contour should be the hand
        hand_segment_max_cont = max(contours, key=cv2.contourArea)

        # Returning the hand segment(max contour) and the thresholded image of hand...
        return (thresholded, hand_segment_max_cont)

#cam = cv2.VideoCapture(0)
url='http://192.168.231.185/cam-hi.jpg'

num_frames =0


while True:
    img_resp=urllib.request.urlopen(url)
    imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
    frame=cv2.imdecode(imgnp,-1)
    # 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)


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

      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.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:
    break

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

# Chapter 7
# Result

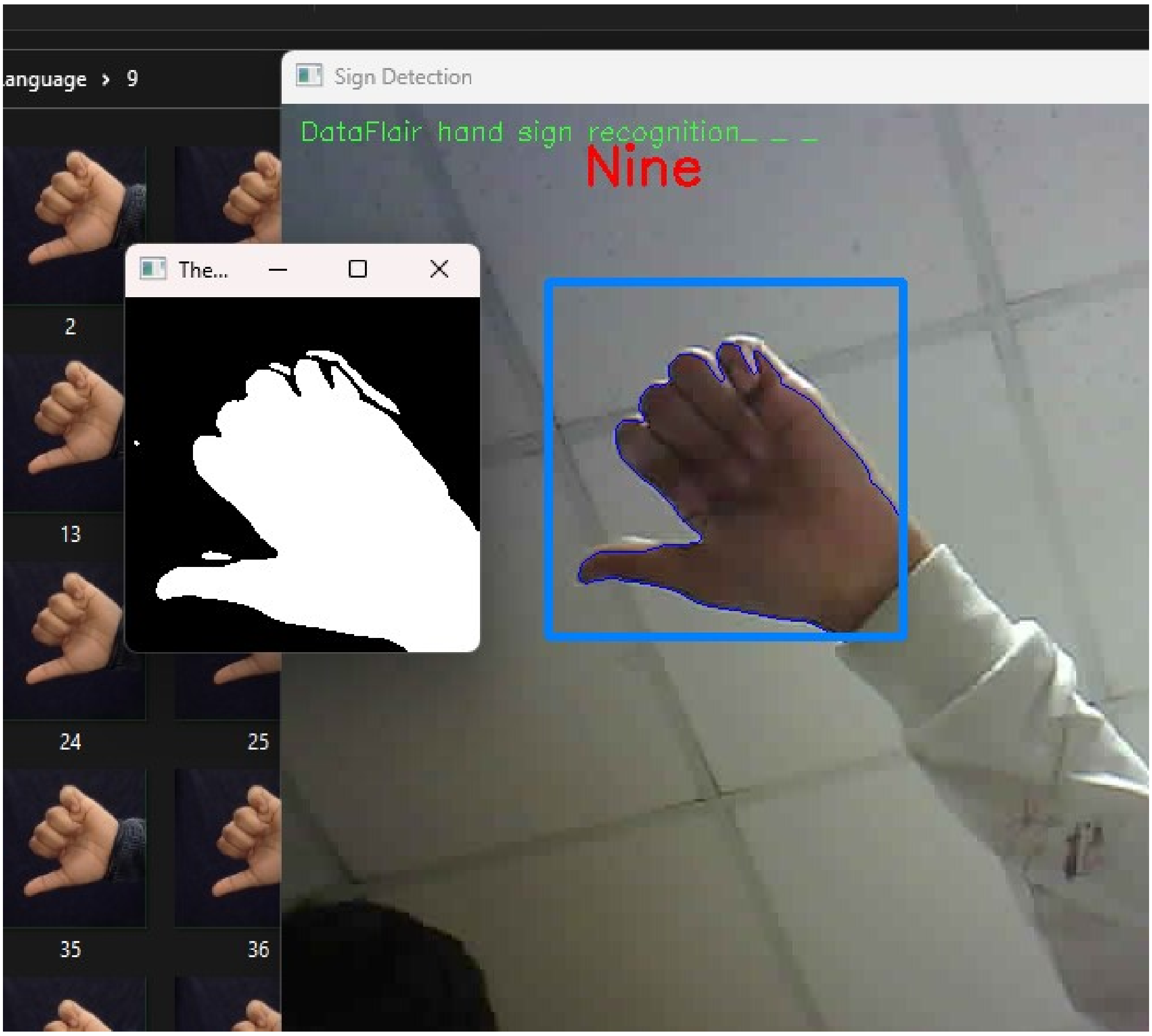


**Fig. Nine**

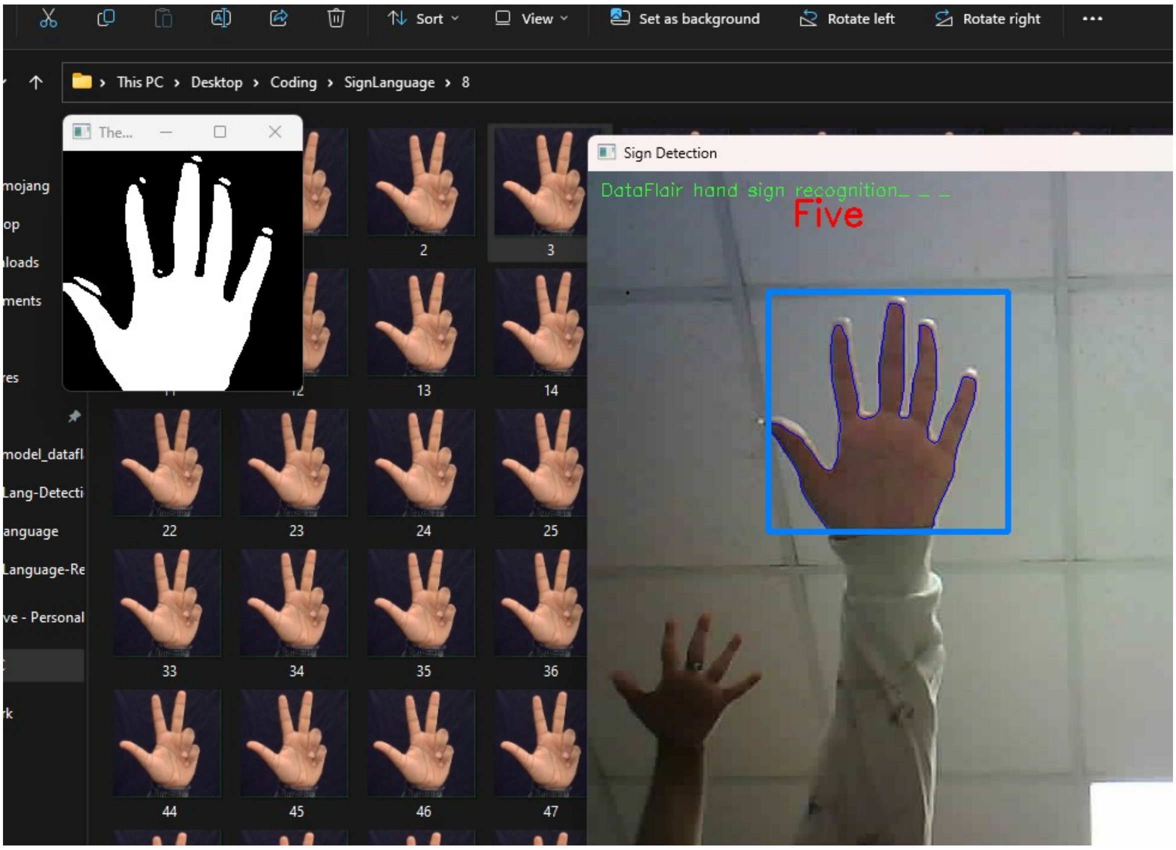


**Fig. Five is displayed**

# Chapter 8

# Conclusion

The ESP 32 camera is used to detect the image. It is programmed and then powered by NodeMCU (esp8266). A CNN model in the backend which is made in python is used to detect and recognize the image and the detected character is displayed on the screen.

## Future Scope

The system can be used in hospitals and a portable model can be made. It can also be trained with different sign languages and even more phrases can be added to make it more efficient and usable.

## References

[1] Katoch, S., Singh, V., Tiwary, U. S. (2022). Indian Sign Language recognition system using SURF with SVM and CNN. Array, 14, 100141. https://doi.org/10.1016/j.array.2022.100141

[2]https://www.hackster.io/ArnovSharmamakes/esp32−cam−web−server−and−getting − started − guide − f 1a04a

[3] Prof. Anjali M. Dalvi1, Shivam Sonawane2, Shruti Degaonkar3, Suraj Kulkarni4, Gauri Chavan5, "Sign Language Recognition Based on Machine Learning", IJIRE-V3I03-137-143

[4] J. Zamora-Mora and M. Chac ́on-Rivas," Real-Time Hand Detection using Convolutional Neural Networks for Costa Rican Sign Language Recognition," 2019 International Conference on Inclusive Technologies and Education (CONTIE), 2019, pp. 180-1806, doi: 10.1109/CONTIE49246.2019.00042.

[5] The dataset used to train the model:
https://www.kaggle.com/datasets/vaishnaviasonawane/indian-sign-language-datase