# Sign Language Recognition using Machine Learning

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

in

Information Technology

by

Dhanvi Mange 119A3013

Mahitha Vaidyanathan 119A3054

Sai Pavani Lanka 119A3056

Under the Guidance of:

**Prof. Mrinal Khadse**


Department of Information Technology

SIES graduate School of Technology

2021-2022

# CERTIFICATE

This is to certify that the Mini project entitled **Sign Language Recognition using Machine Learning** is a bonafide work of the following students, submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology.**

| | |
|---|---|
| Dhanvi Mange | 119A3013 |
| Mahitha Vaidyanathan | 119A3054 |
| Sai Pavani Lanka | 119A3056 |

# PROJECT REPORT APPROVAL

This IoT Mini project report entitled ***Sign Language Recognition using Machine Learning*** by following students is approved for the degree of ***Bachelor of Engineering*** in ***Information Technology.***

| | |
|---|---|
| ***Dhanvi Mange*** | ***119A3013*** |
| ***Mahitha Vaidyanathan*** | ***119A3054*** |
| ***Sai Pavani Lanka*** | ***119A3056*** |

**Name of External Examiner:** ----------

**Signature:** ----------

**Name of Internal Examiner:** ----------

**Signature:** ----------

**Date:**

**Place:** SIES GST Nerul

# DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Dhanvi Mange 119A3013

Mahitha Vaidyanathan 119A3054

Sai Pavani Lanka 119A3056

Date:

## ACKNOWLEDGEMENT

It gives us immense pleasure to thank Dr. Atul Kemker, Principal for extending his support to carry out this project. We also thank to Head of Department Dr. K. Lakshmi Sudha for her support in completing the project. We wish to express our deep sense of gratitude and thank to our Internal Guide, Ms. Mrinal Khadse for her guidance, help and useful suggestions, which helped in completing our project work in time.

Also, we would like to thank the entire faculty of Information Technology Department for their valuable ideas and timely assistance in this project, last but not the least, we would like to thank our non-teaching staff members of our college for their support, in facilitating timely completion of this project.

Dhanvi Mange 119A3013
Mahitha Vaidyanathan 119A3054
Sai Pavani Lanka 119A3056

# ABSTRACT

Sign language is the only tool of communication for the person who is not able to speak and hear anything. It is a boon for the differently abled people to express their thoughts and emotions. In this work, a novel scheme of sign language recognition has been proposed for identifying the alphabets and gestures in sign language. With the help of machine leaning and neural networks we can detect the signs and give the respective text output.

The project aims at building a machine learning model that will be able to classify the various hand gestures used for fingerspelling in sign language. In this user independent model, classification machine learning algorithms are trained using a set of image data. For the image dataset, depth images are used, which gave better results than some of the previous literatures, owing to the reduced pre-processing time. The machine learning algorithm applied on the datasets is Convolutional Neural Network (CNN).

# Contents

# Chapter 1

## Introduction:

Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, writing or through visual aids, speech being one of the most used among them. Unfortunately, for the speaking and hearing-impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency. Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign Language consists of fingerspelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. Fingerspelling is a vital tool in sign language, as it enables the communication of names, addresses and other words that do not carry a meaning in word level association. In spite of this, fingerspelling is not widely used as it is challenging to understand and difficult to use. Moreover, there is no universal sign language and very few people know it, which makes it an inadequate alternative for communication.

# Chapter 2

## REVIEW OF LITERATURE:

| SR. No. | Author | Title & Publication | Key Findings |
|---|---|---|---|
| 1. | P. Nanivadekar, V. Kulkarni | Indian Sign Language Recognition: Database Creation, Hand Tracking and Segmentation | The results demonstrate working of motion tracking, edge detection and skin color detection individually as well as their combined effect. The gesture include in database are alphabets A to Z, number 1 to 10 |
| 2. | G. Khurana, G. Joshi, J. Kaur | Static Hand Gesture Recognition System using Shape Based Features | Easy and simple approach of shape-based recognition of Indian Sign Language (ISL) has proposed. Minimum Euclidian distance is used for sign language recognition. The databases of 26 (ISL) alphabets out of which 19 distinct alphabets are consider |
| 3. | M. Mohandes | Image-Based and Sensor-Based Approach to Sign Language Recognition | Sign language continues to be the preferred method of communication among the deaf and the hearing-impaired |

.

# Chapter 3

## Report on the Present Investigation:

The systems which are currently available only provide the feature for learning Sign Language which is not very feasible as it does not help the differently abled people to communicate with everyone. Therefore, there has been an immense amount of research done to understand and implement a system for real-time recognition of Sign language which is more beneficial towards the differently abled people as well as those with whom they wish to communicate as it helps them to instantly have a conversation with everyone making it easier. However, there are very few existing systems for real-time recognition. Some of which include:

Indian Sign Language Translator App: Indian Signs App is based on Indian Sign Language; it is a Sign language learning application. Those who are interested in learning Indian sign language can utilize this application. It is suitable for beginner and parents of Deaf children. It has Indian Sign language Alphabets, Numbers, and common conversation sentences in Indian sign language anybody can carry the same in his/her pocket, it works without internet connection.

ASL Translator App: The Text-To-Sign portion of this app translates English text into ASL signs and generates sentences in "English word order". However, it is not "Signed Exact English" the translation is improved with Smart Translation Algorithm. It consists of a section to learn ASL. However, the app is not free of cost and requires an internet connection to operate.

The features of our proposed model include:

Data Pre-Processing: In this module, based on the object detected in front of the camera its binary images is being populated. Meaning the object will be filled with solid white and background will be filled with solid black. Based on the pixel's regions, their numerical value in range of either 0 or 1 is being given to next process for modules.

Scanning Single Gesture: A gesture scanner will be available in front of the end user where the user will have to do a hand gesture. Based on Pre-Processed module output, a user shall be able to see associated label assigned for each hand gestures, based on the predefined American Sign Language (ASL) standard inside the output window screen.

Creating gesture: A user will give a desired hand gesture as an input to the system with the text box available at the bottom of the screen where the user needs to type whatever he/she desires to associate that gesture with. This customize gesture will then be stored for future purposes and will be detected in the upcoming time.

Formation of a sentence: A user will be able to select a delimiter and until that delimiter is encountered every scanned gesture character will be appended with the previous results forming a stream of meaning-full words and sentences.

Exporting: A user would be able to export the results of the scanned character into an ASCII standard textual file format.

# Chapter 4

## Study of the Proposed System:

### 4.1 Problem Statement:

Sign language is a language that uses manual communication methods such as facial expressions, hand gestures and body movements to convey information. The problem arises when dumb or deaf people try to express themselves to other people with the help of these sign language. This is because normal people are usually unaware of the signs. As a result, it has been seen that communication of a dumb person is only limited within his/her family or the dumb/deaf community. At this age of Technology, the demand for a computer-based system is high for the dumb community. Our project, hence, is aimed at converting the sign language gestures into text that is readable for normal.

### 4.2 Algorithm Used:

Convolution Neural Network (CNN):

The advancements in Computer Vision with Deep Learning have been constructed and perfected with time, primarily over one particular algorithm — a Convolutional Neural Network. When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for image classification we use Convolution Neural networks.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. In CNN, we take an image as an input, assign importance to its various aspects/features in the image. The pre-processing required in CNN is much lesser as compared to other classification algorithms. The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

A CNN typically has three layers: a convolutional layer, pooling layer, and fully connected layer.

Convolution Layer:

The main objective of convolution is to extract features such as edges, colors, corners from the input. As we go deeper inside the network, the network starts identifying more complex features such as shapes, digits, face parts as well.

Pooling Layer:

This layer is solely to decrease the computational power required to process the data. It is done by decreasing the dimensions of the featured matrix even more. In this layer, we try to extract the dominant features from a restricted amount of neighborhood. Let us make it clear by taking an example.

Fully connected layer:

Till now nothing has been about classifying different images, what we have done is highlighting some features in an image and reduces the dimensions of the image drastically. From here on, the classification process is going to start.

Now that we have converted our input image into a suitable form for our multi-Level fully connected architecture, we shall flatten the image into one column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them.

**4.3 Software Requirements:**

Operating System: Windows 10

Programming Language: Python

Packages used: TensorFlow, Keras, OpenCV2, Tkinter, PyQt5, scipy, qimage2ndarray, winGuiAuto, pypiwin32, sys, keyboard, pyttsx, pillow,

IDE: VS Code

# Chapter 5

## Report on Proposed system and its implementation:

### 5.1 Existing System:

Most existing systems are only made for learning Sign Language. There are very few already existing systems, some of which include:

i. ASL Translator

ii. Indian Sign Language Translator

However, there are various Research papers written under this topic

### 5.2 Proposed System:

Our system is designed to overcome the troubles faced by the deaf people. This system is designed to translate each word that is received as input into sign language. We will use convolution neural networks which recognizes various hand gestures by capturing images and converts it into frames. The image obtained, is sent for comparison to the trained model. In our interface we will provide features which will convert sign language to text. We have also provided a feature to generate customizable gestures, so that the user can generate signs in case they use another language set apart from the one which we have trained our model such as ISL. Hence, our model can be used by multiple users even if the Language used for communication is different. Our system not only does real-time recognition of alphabets but also can form words or a stream of sentences based on the gestures made. The sentences/words that are formed can also be converted into a text file and can be saved in desired locations. This text is also converted in audio format. Thus, our system is more robust in getting exact text labels of letters, create gestures as well as form sentences on showing the particular hand-gesture.

### 5.3 Implementation:

### Scanning & Recognition of Gestures

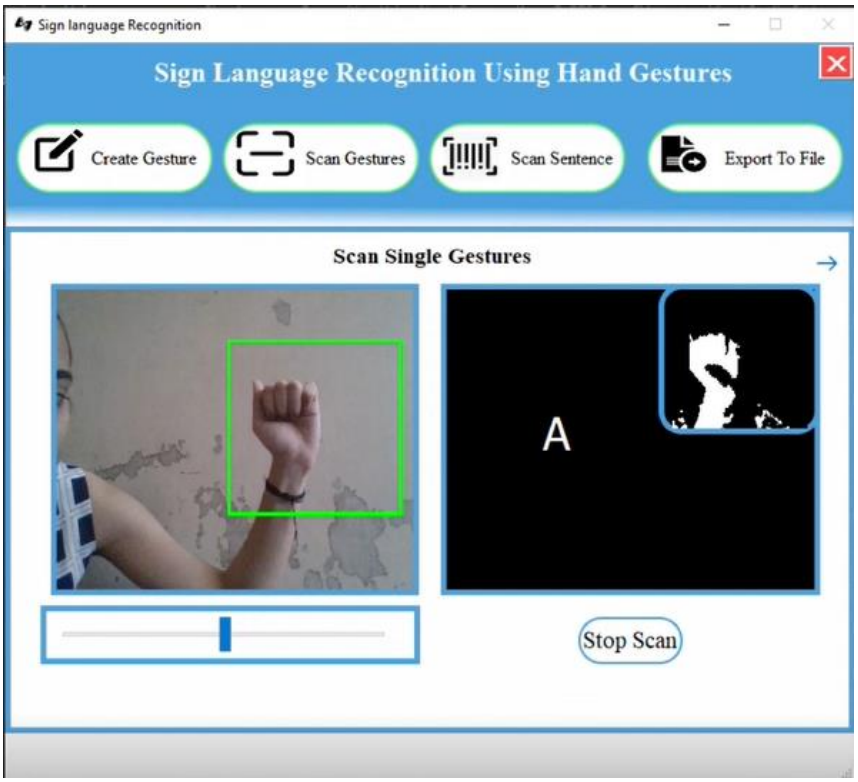



Fig 1.0: Recognition of Letter A

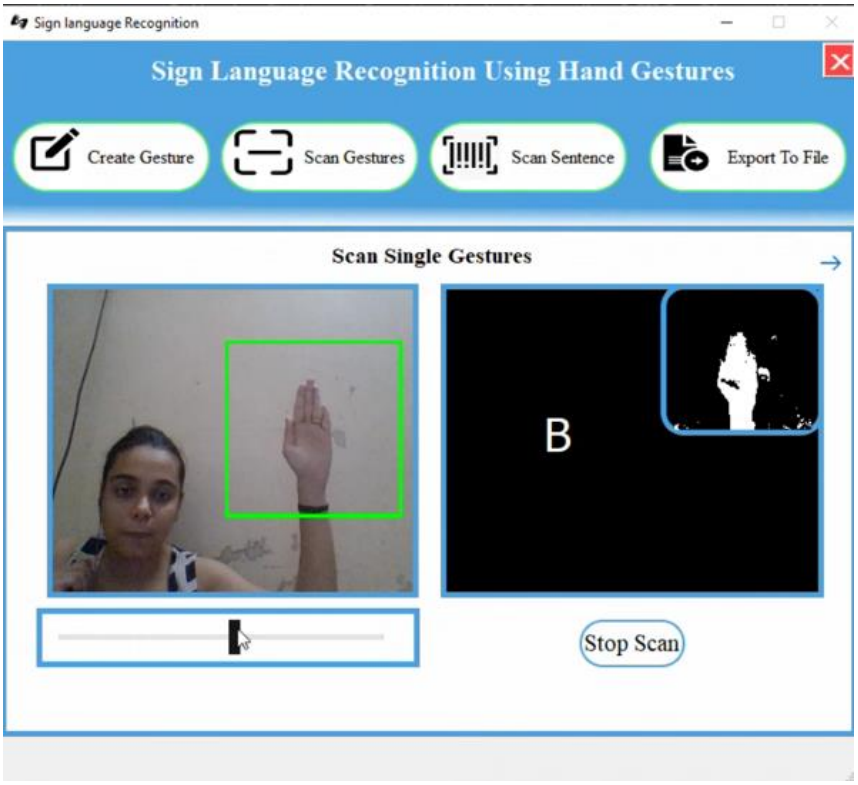



Fig 2.0: Recognition of Letter B

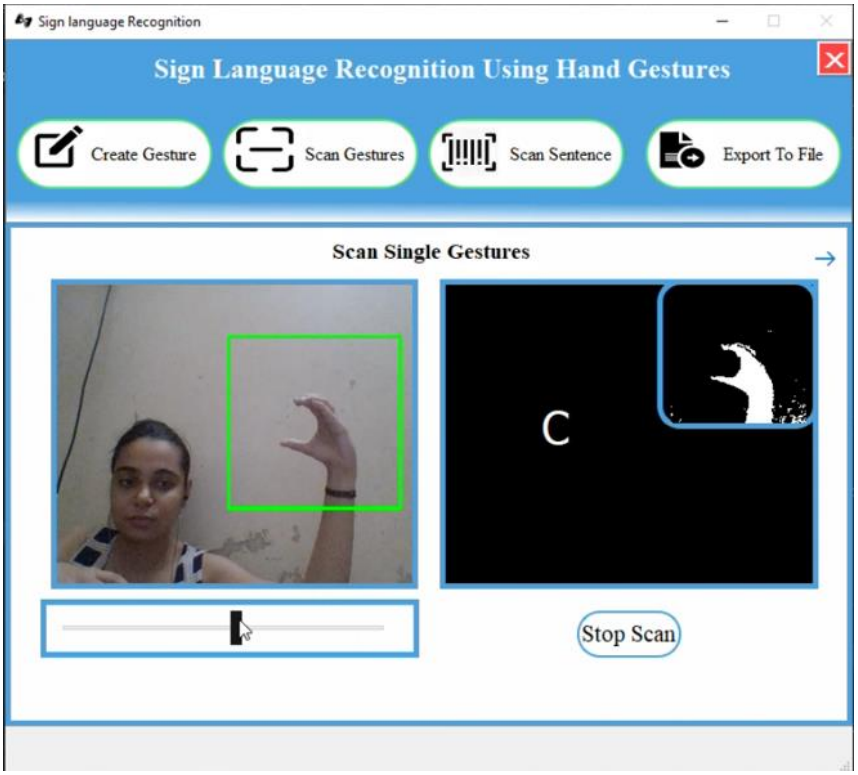



Fig 3.0: Recognition of Letter C

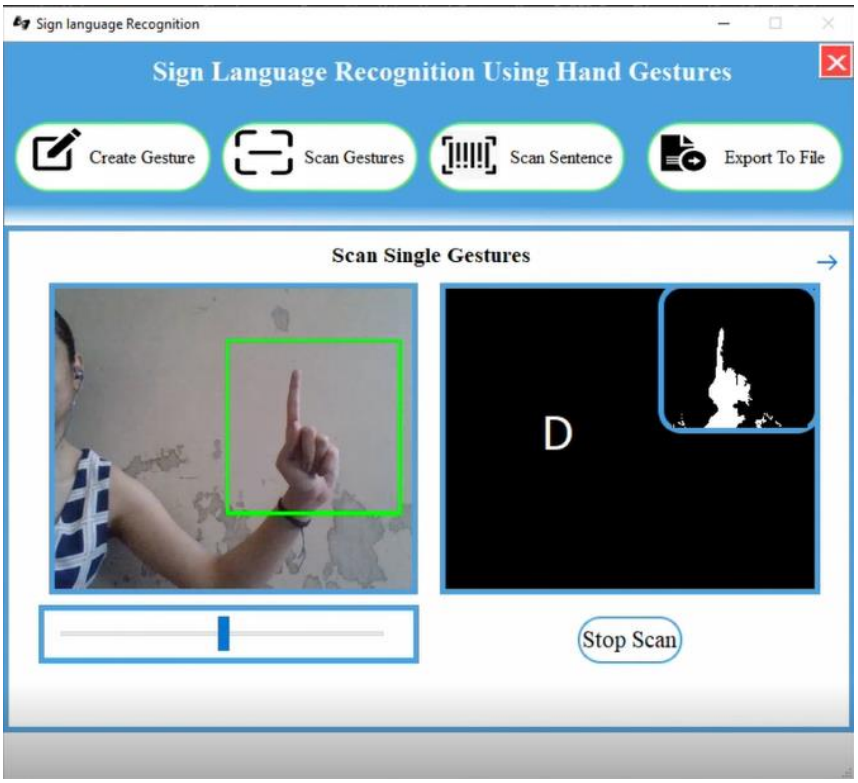



Fig 4.0: Recognition of Letter D

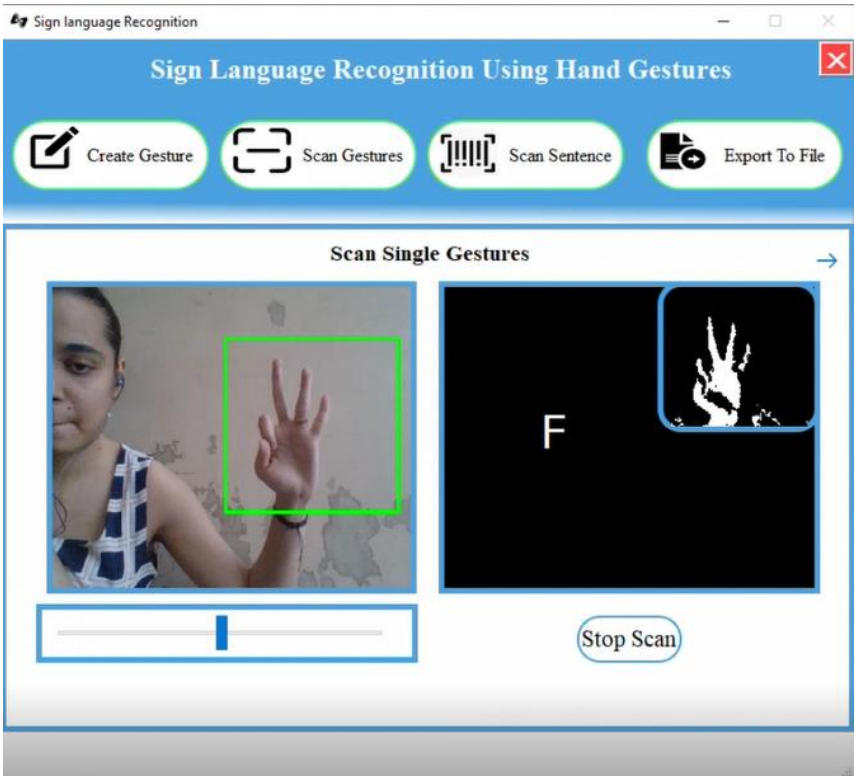



Fig 5.0: Recognition of Letter F

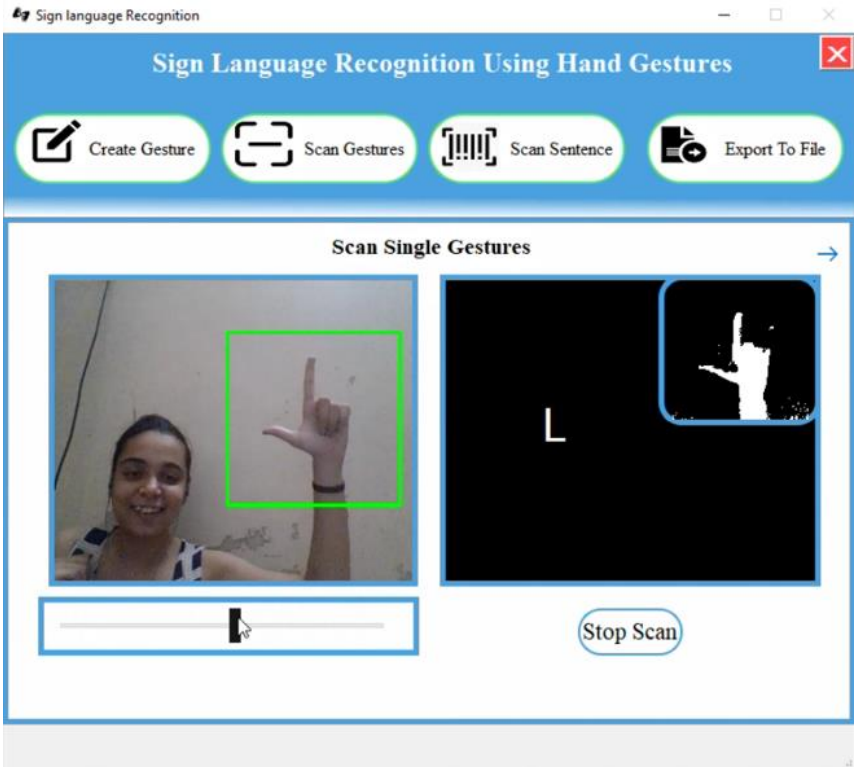



Fig 6.0: Recognition of Letter L

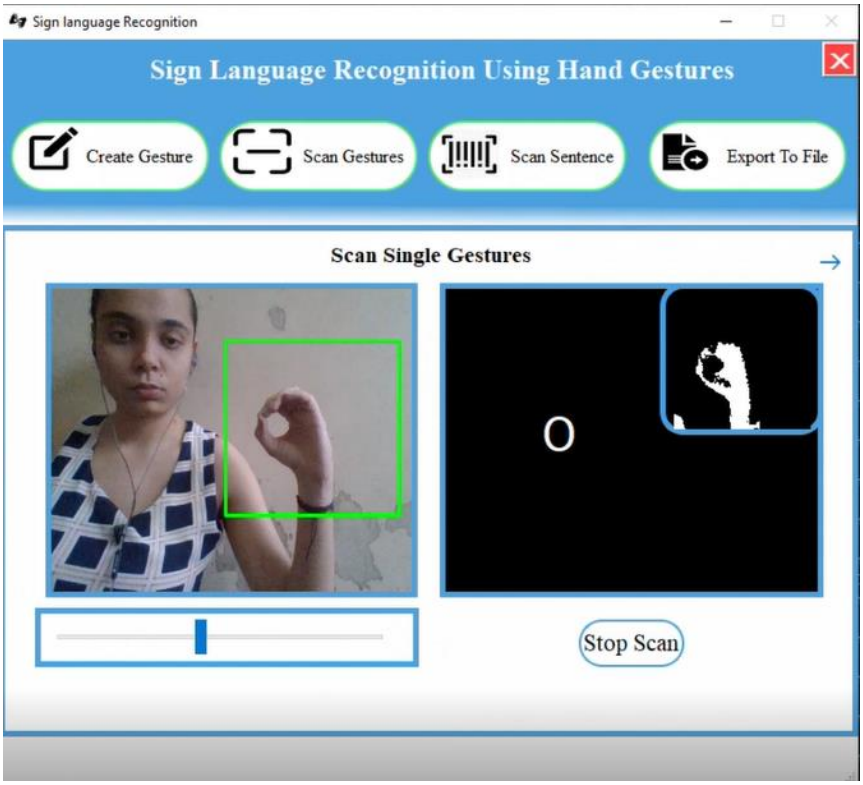



Fig 7.0: Recognition of Letter O

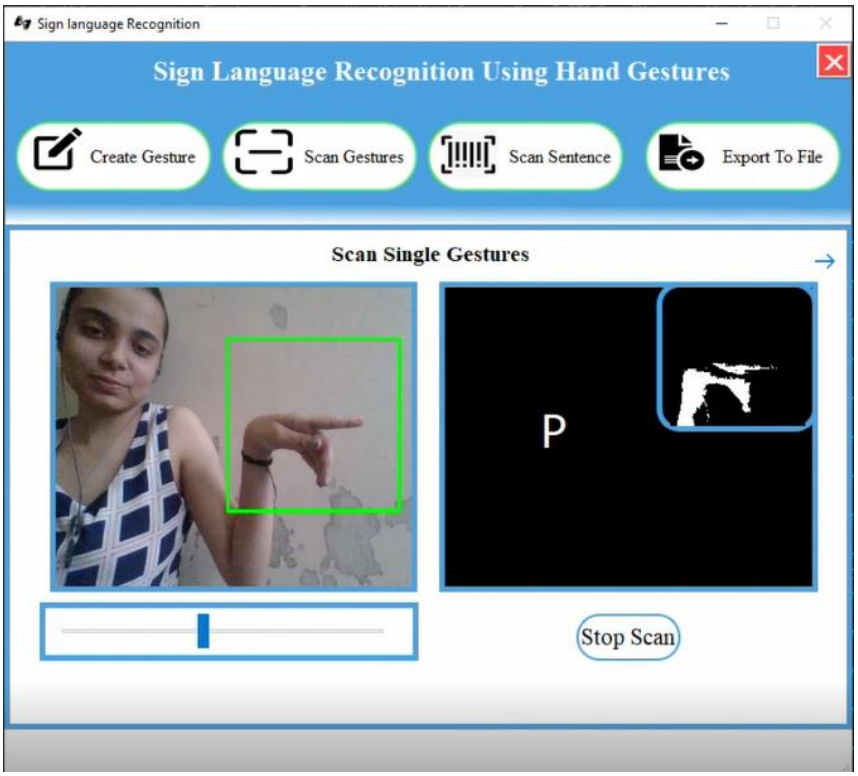



Fig 8.0: Recognition of Letter P

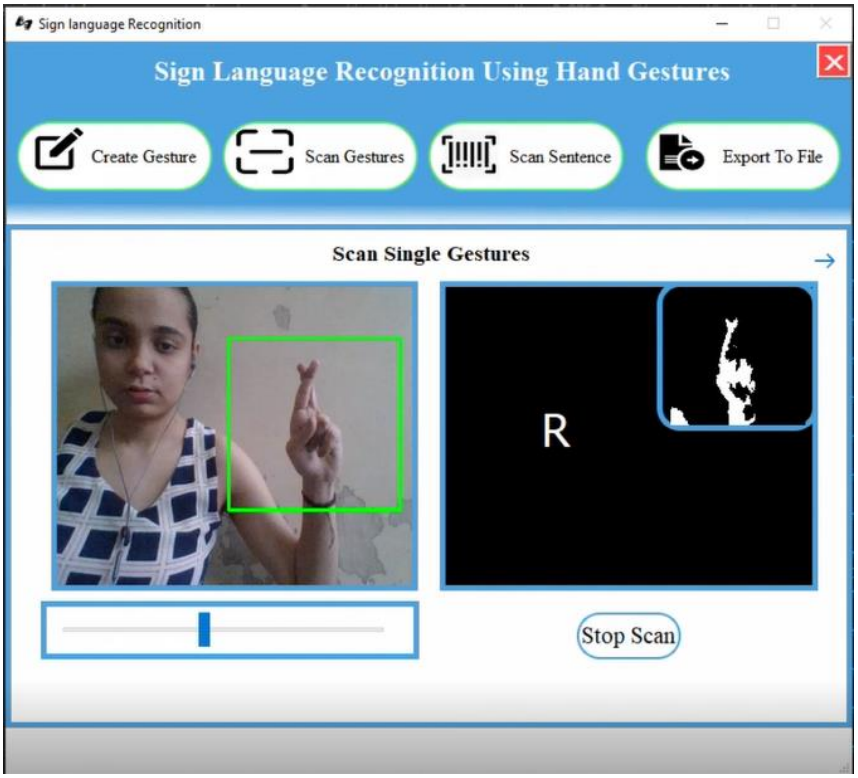



Fig 9.0: Recognition of Letter R

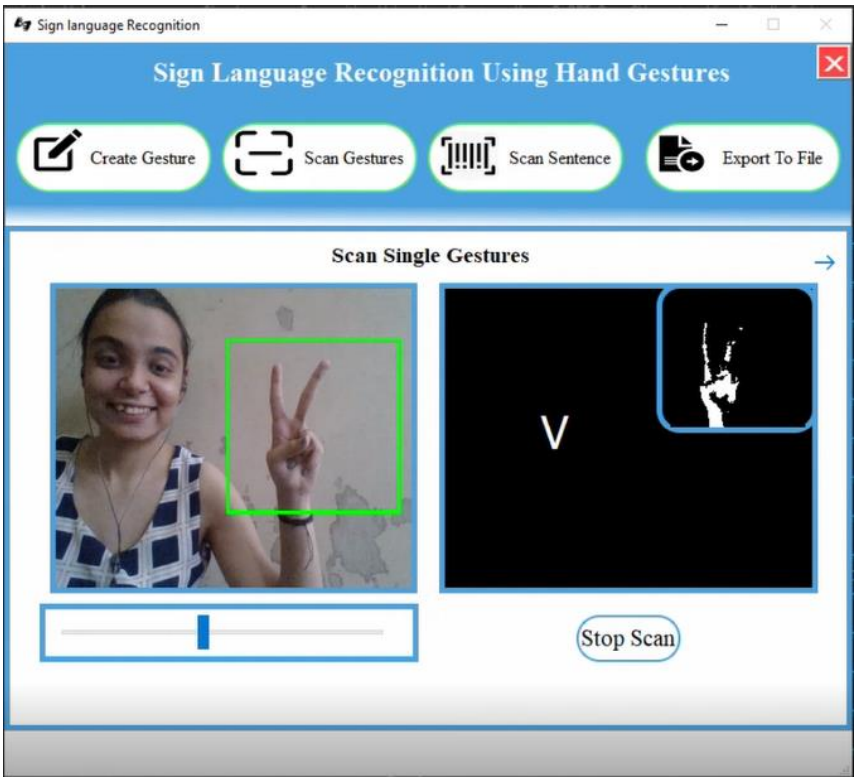



Fig 10.0: Recognition of Letter V

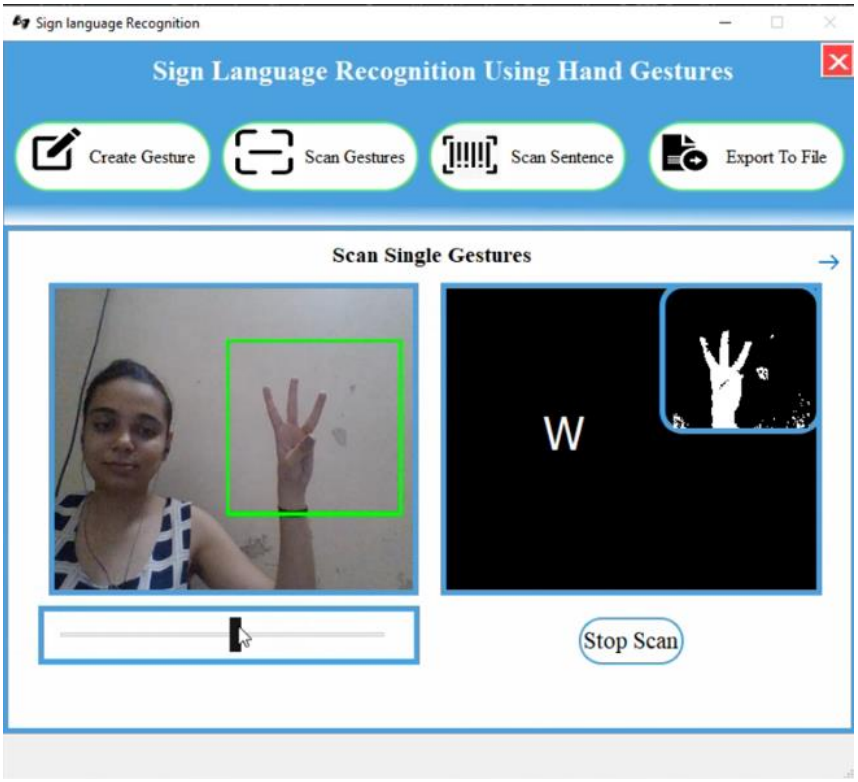



Fig 11.0: Recognition of Letter W

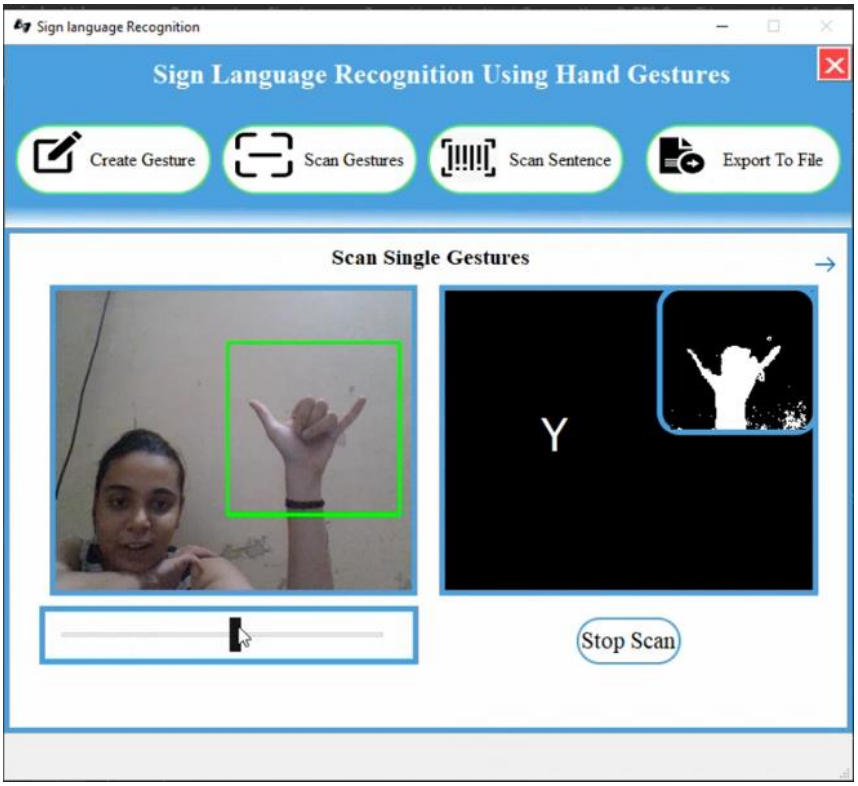



Fig 12.0: Recognition of Letter Y

**Scanning Sentence**

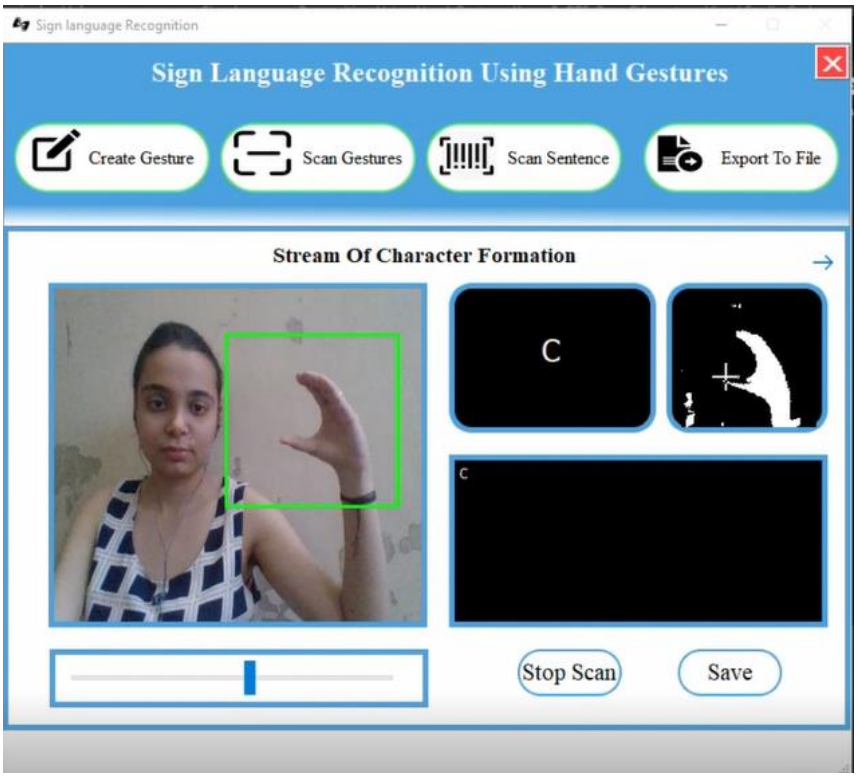



Fig 13.0: Scanning the Letter C to form the word CAR

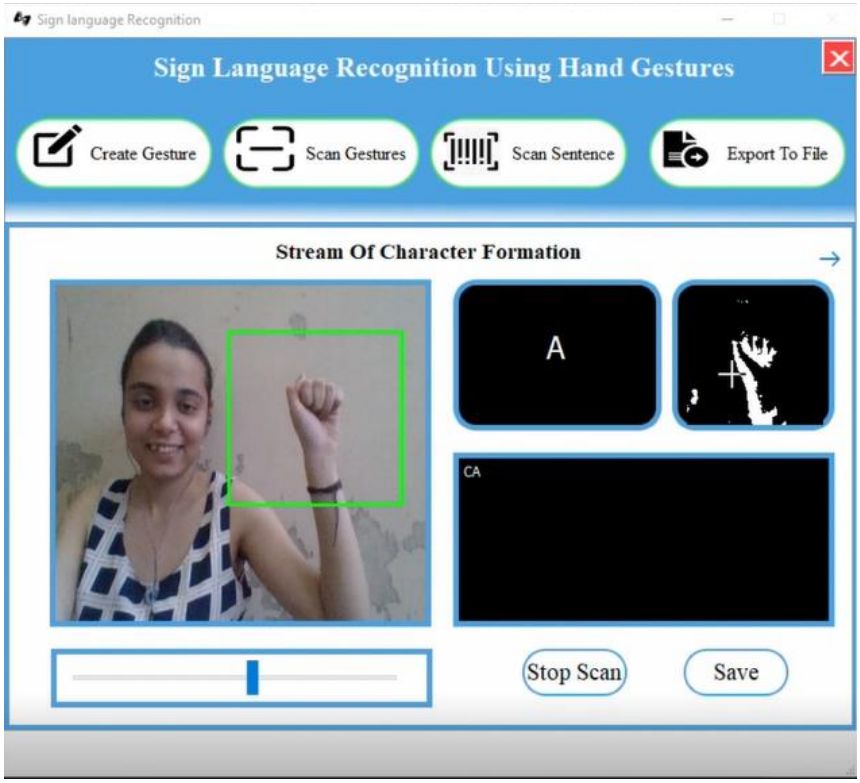



Fig 14.0: Scanning the Letter A to form the word CAR

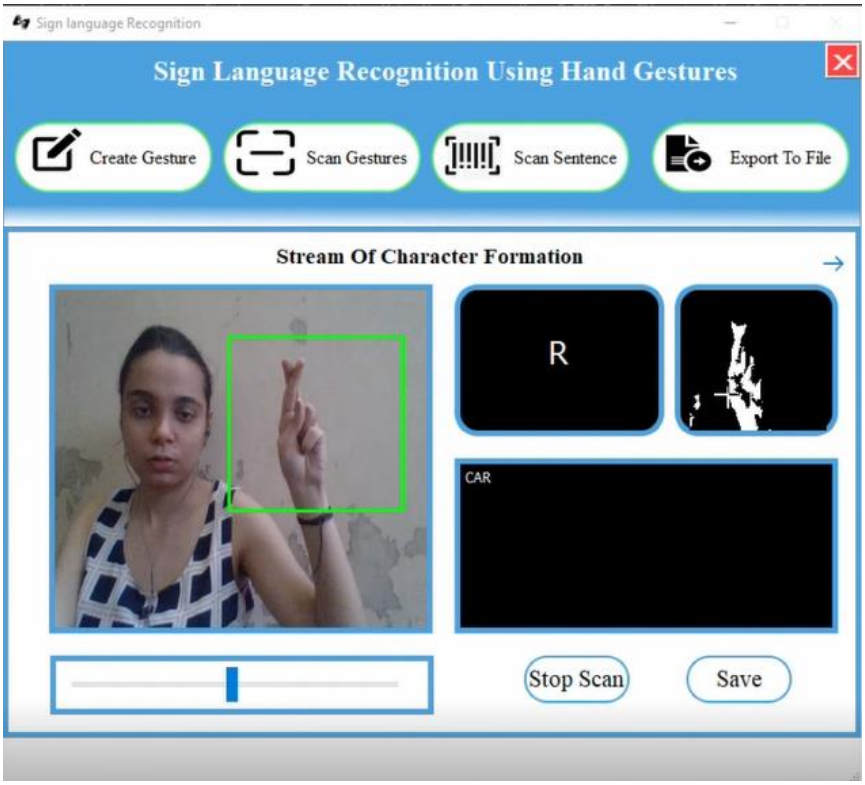



Fig 15.0: Scanning the Letter R to form the word CAR

**Exporting to File**

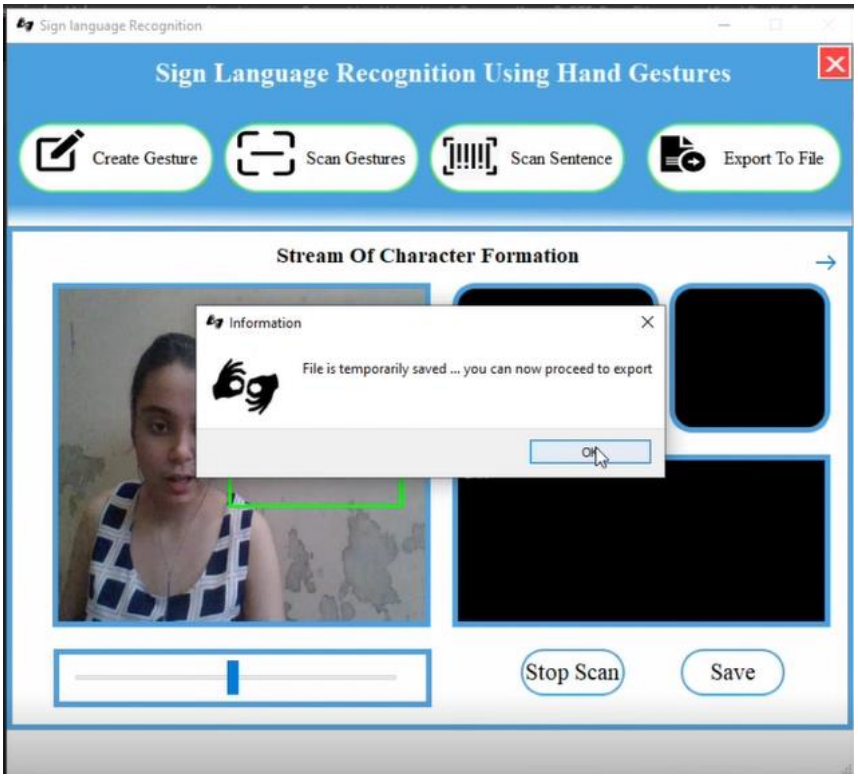



Fig 16.0: Saving the scanned word CAR

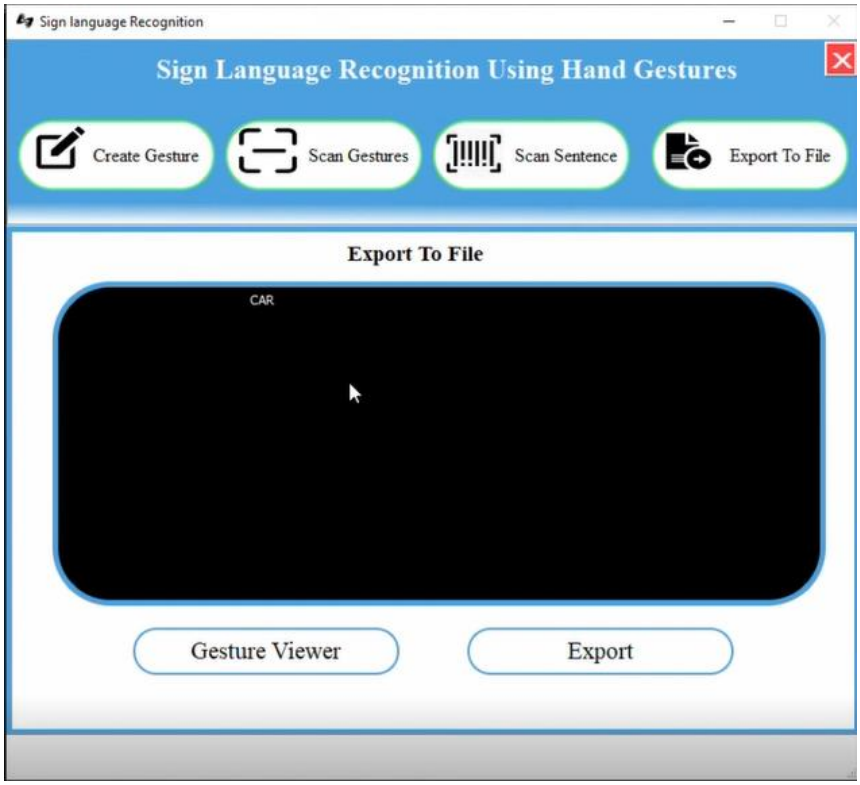



Fig 17.0: Exporting the file containing the scanned word CAR

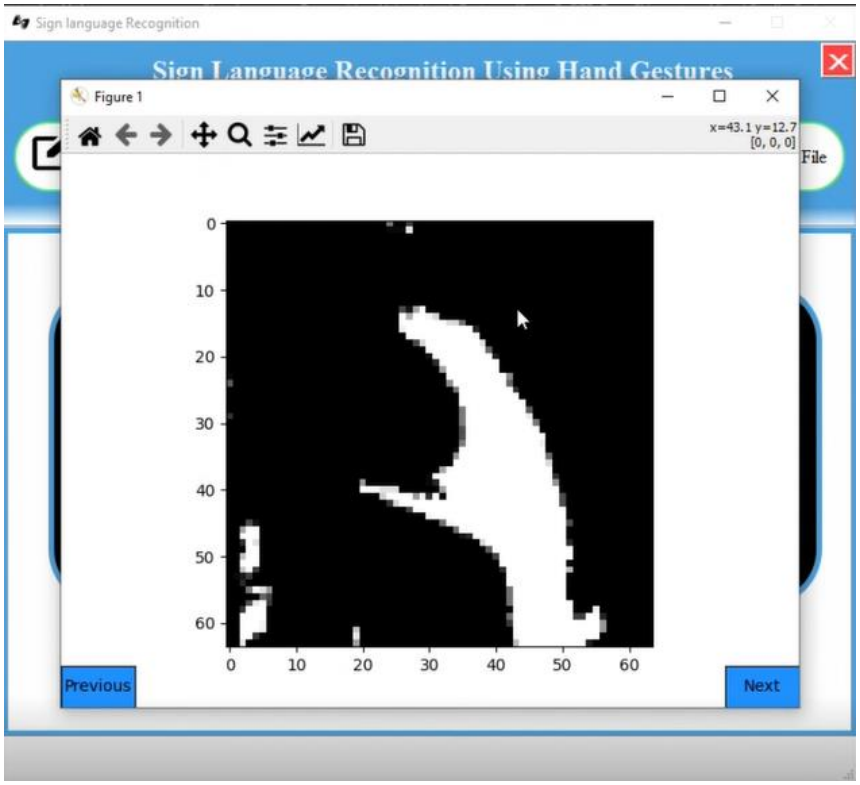



Fig 18.0: Viewing the Gestures saved at the time of scanning the word

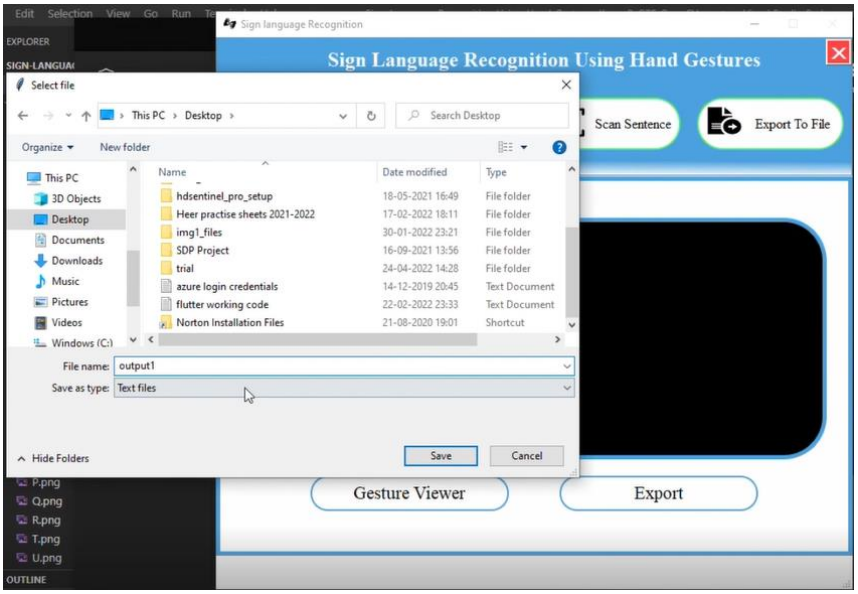



Fig 19.0: Exporting the text file to the system

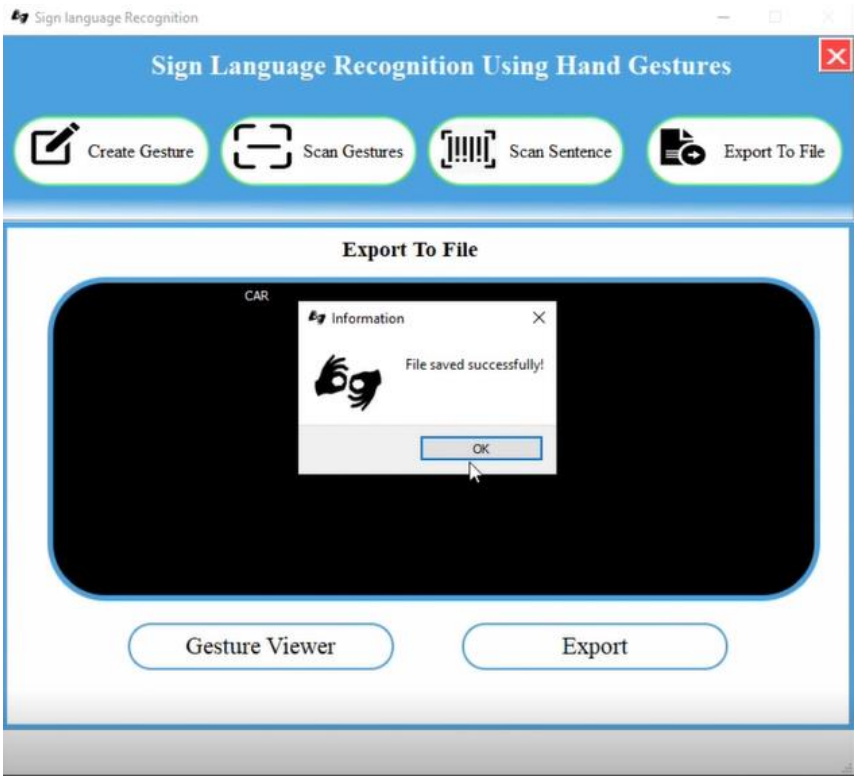



Fig 20.0: File saved Successfully

# Chapter 6

## Code:

cnn_model.py

Source-Code > cnn_model.py > ...

```python
# Part 1 - Building the CNN
#importing the Keras libraries and packages
from keras.models import Sequential
import tensorflow as tf
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense, Dropout
from keras import optimizers

# Initialing the CNN
classifier = Sequential()

# Step 1 - Convolutio Layer
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))

#step 2 - Pooling
classifier.add(MaxPooling2D(pool_size =(2,2)))

# Adding second convolution layer
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size =(2,2)))

#Adding 3rd Convolution Layer
# classifier.add(Convolution2D(64, 3, 3, activation = 'relu'))
# classifier.add(MaxPooling2D(pool_size =(2,2)))


#Step 3 - Flattening
classifier.add(Flatten())

#Step 4 - Full Connection
classifier.add(Dense(256, activation = 'relu'))
classifier.add(Dropout(0.5))
classifier.add(Dense(26, activation = 'softmax'))

#Compiling The CNN
classifier.compile(
        optimizer = tf.keras.optimizers.SGD(lr = 0.01),
        loss = 'categorical_crossentropy',
        metrics = ['accuracy'])

#Part 2 Fittting the CNN to the image
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

```

```
test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory(
        'Dataset/training_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

test_set = test_datagen.flow_from_directory(
        'Dataset/test_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

model = classifier.fit_generator(
        training_set,
        steps_per_epoch=800,
        epochs=25,
        validation_data = test_set,
        validation_steps = 6500
      )

'''#Saving the model
import h5py
classifier.save('Trained_model.h5')'''

print(model.history.keys())
import matplotlib.pyplot as plt
# summarize history for accuracy
plt.plot(model.history['acc'])
plt.plot(model.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss

plt.plot(model.history['loss'])
plt.plot(model.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Source-Code > Dashboard.py > ...

```python
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QMessageBox
from PyQt5.QtCore import QUrl
from PyQt5.QtGui import QImage
from PyQt5.QtGui import QPixmap
from PyQt5 import QtCore                    #importing pyqt5 libraries
from imageio import imread                  #will help in reading the images
from PyQt5.QtCore import QTimer,Qt
from PyQt5 import QtGui
from tkinter import filedialog              #for file export module
from tkinter import *
import tkinter as tk
from matplotlib import pyplot as plt        #for gesture viewer
from matplotlib.widgets import Button
import sys                                  #for pyqt
import os                                   #for removal of files
import cv2                                  #for the camera operations
import numpy as np                          #proceesing on images
import qimage2ndarray                       #convers images into matrix
import win32api
import winGuiAuto
import win32gui
import win32con                             #for removing title cv2 window and always on top
import keyboard                             #for pressing keys
import pyttsx3                              #for tts assistance
import shutil                               #for removal of directories
index = 0                                   #index used for gesture viewer
engine = pyttsx3.init()                     #engine initialization for audio tts assistance

def nothing(x):
    pass

image_x, image_y = 64,64                    #image resolution

from keras.models import load_model
classifier = load_model('ASLModel.h5')      #loading the model
```

# Chapter 7

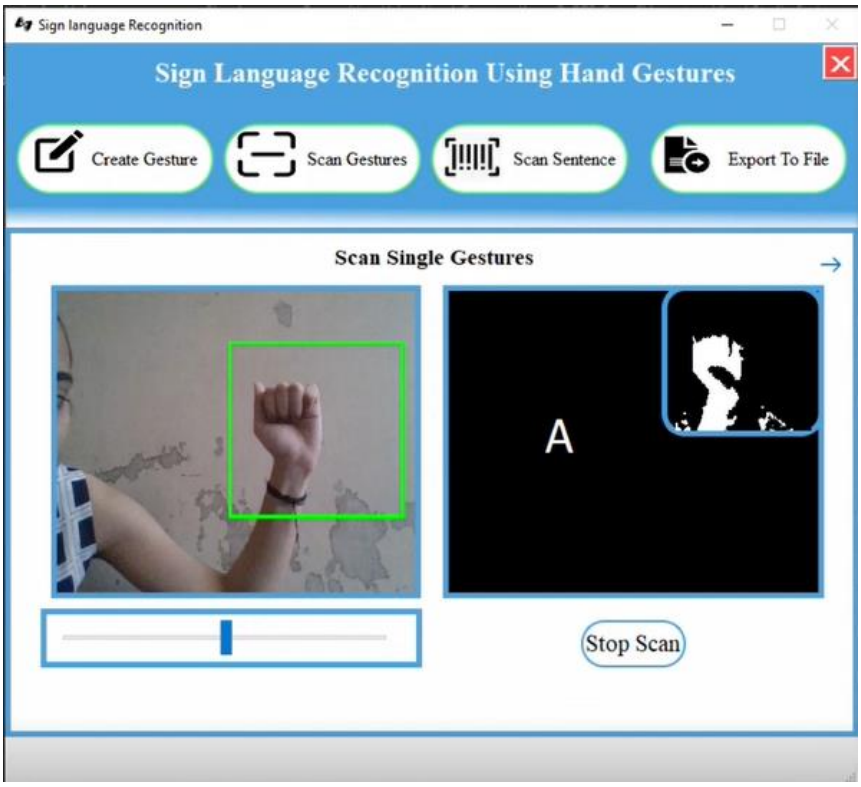
## Results and Discussions:

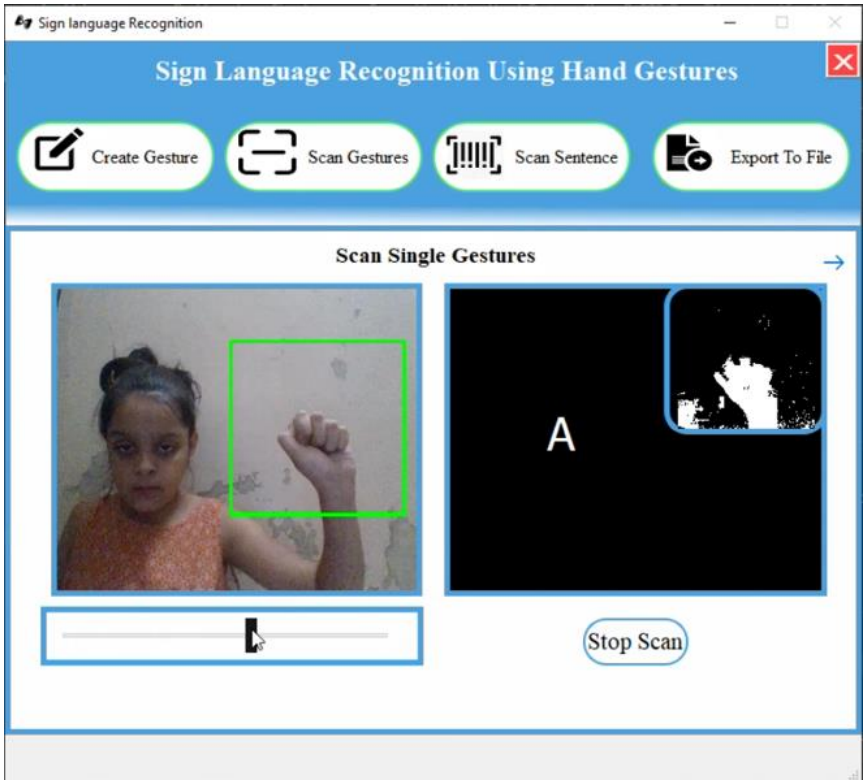
Our final classification accuracy on the test set achieved after using the second convolution layer is 63.66%. The accuracy results that we obtained were compared to the accuracy of digit recognition on the MNIST dataset and therefore we believe that our approach works relatively well.

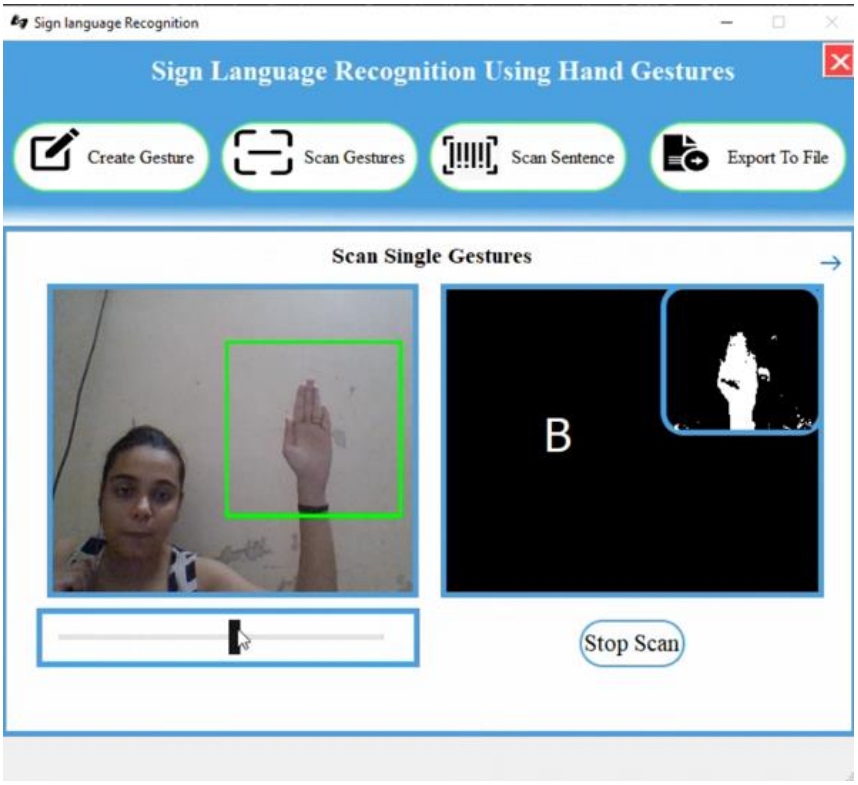
```
TERMINAL
Epoch 1/25
800/800 [==============================] - ETA: 0s - loss: 3.2437 - accuracy: 0.0550WARNING:tensorflow:Your input ran out of data; inte
rrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 650
0 batches). You may need to use the repeat() function when building your dataset.
800/800 [==============================] - 465s 579ms/step - loss: 3.2437 - accuracy: 0.0550 - val_loss: 3.2245 - val_accuracy: 0.0668
Epoch 2/25
800/800 [==============================] - 156s 195ms/step - loss: 3.1304 - accuracy: 0.0907
Epoch 3/25
800/800 [==============================] - 102s 128ms/step - loss: 2.7721 - accuracy: 0.1615
Epoch 4/25
800/800 [==============================] - 79s 98ms/step - loss: 2.3763 - accuracy: 0.2608
Epoch 5/25
800/800 [==============================] - 72s 90ms/step - loss: 2.0683 - accuracy: 0.3372
Epoch 6/25
800/800 [==============================] - 68s 84ms/step - loss: 1.8818 - accuracy: 0.3910
Epoch 7/25
800/800 [==============================] - 63s 78ms/step - loss: 1.7525 - accuracy: 0.4233
Epoch 8/25
800/800 [==============================] - 64s 80ms/step - loss: 1.6529 - accuracy: 0.4538
Epoch 9/25
800/800 [==============================] - 70s 87ms/step - loss: 1.5676 - accuracy: 0.4790
Epoch 10/25
800/800 [==============================] - 61s 76ms/step - loss: 1.4885 - accuracy: 0.5027
Epoch 11/25
800/800 [==============================] - 60s 75ms/step - loss: 1.4427 - accuracy: 0.5131
Epoch 12/25
800/800 [==============================] - 62s 77ms/step - loss: 1.3875 - accuracy: 0.5316
Epoch 13/25
800/800 [==============================] - 63s 79ms/step - loss: 1.3368 - accuracy: 0.5512
```

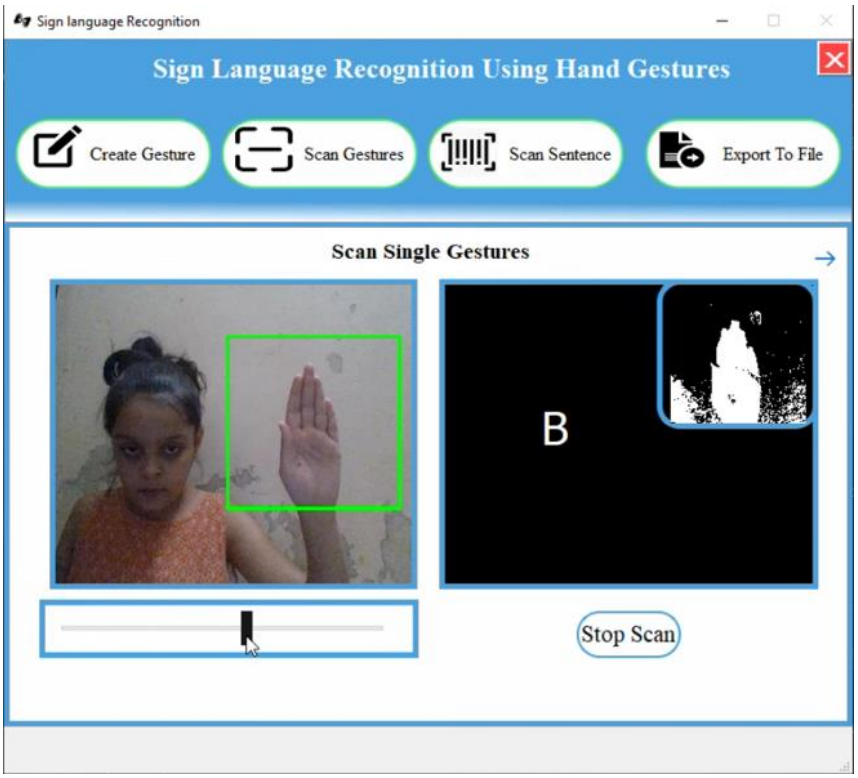
```
Epoch 14/25
800/800 [==============================] - 66s 82ms/step - loss: 1.3159 - accuracy: 0.5506
Epoch 15/25
800/800 [==============================] - 73s 91ms/step - loss: 1.2756 - accuracy: 0.5704
Epoch 16/25
800/800 [==============================] - 57s 71ms/step - loss: 1.2463 - accuracy: 0.5769
Epoch 17/25
800/800 [==============================] - 69s 86ms/step - loss: 1.2201 - accuracy: 0.5821
Epoch 18/25
800/800 [==============================] - 56s 70ms/step - loss: 1.1927 - accuracy: 0.5966
Epoch 19/25
800/800 [==============================] - 58s 73ms/step - loss: 1.1823 - accuracy: 0.5999
Epoch 20/25
800/800 [==============================] - 56s 69ms/step - loss: 1.1454 - accuracy: 0.6099
Epoch 21/25
800/800 [==============================] - 61s 77ms/step - loss: 1.1309 - accuracy: 0.6156
Epoch 22/25
800/800 [==============================] - 55s 69ms/step - loss: 1.1203 - accuracy: 0.6215
Epoch 23/25
800/800 [==============================] - 55s 69ms/step - loss: 1.0982 - accuracy: 0.6233
Epoch 24/25
800/800 [==============================] - 56s 69ms/step - loss: 1.0687 - accuracy: 0.6330
Epoch 25/25
800/800 [==============================] - 65s 81ms/step - loss: 1.0567 - accuracy: 0.6366
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

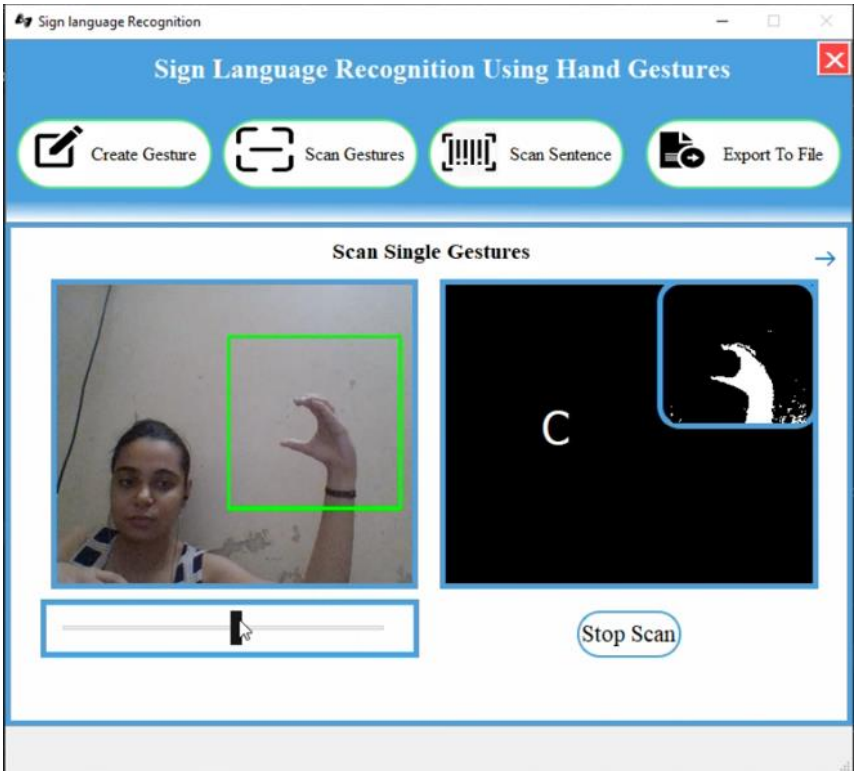
As a finishing step to our project, we have successfully created a real time implementation of our entire system, so that hand gestures made in front of our computer displays the output, which is one of the twenty-six letters in our dataset. To check the accuracy of the model, we have also tested the model with multiple users whose hand's size vary.

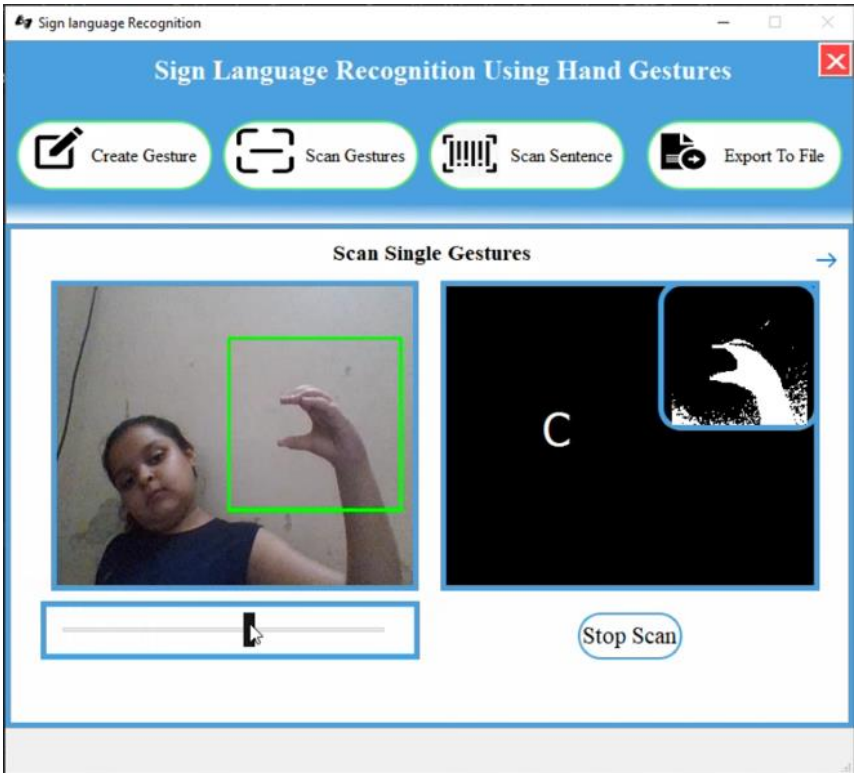


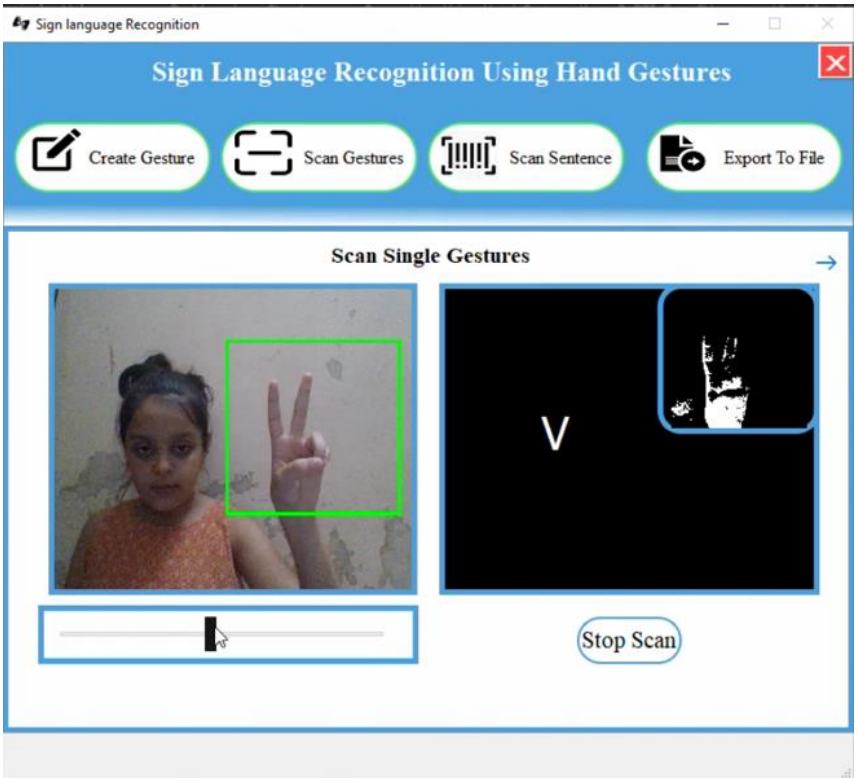



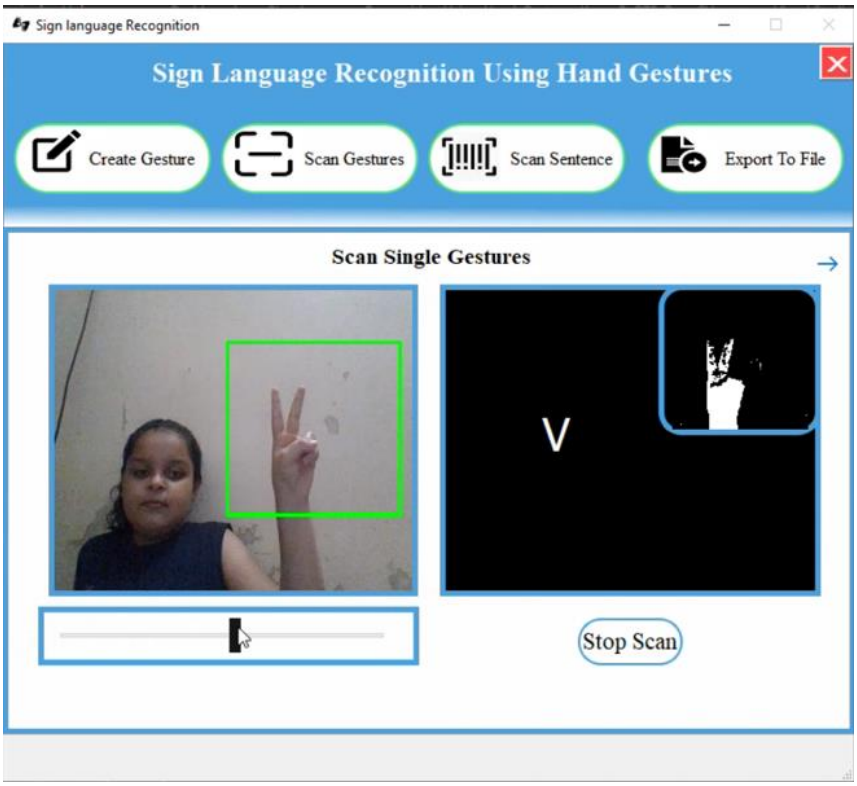

Sign language Recognition
Sign Language Recognition Using Hand Gestures
Create Gesture
Scan Gestures
Scan Sentence
Export To File
Scan Single Gestures
B
Stop Scan


Sign language Recognition
Sign Language Recognition Using Hand Gestures
Create Gesture
Scan Gestures
Scan Sentence
Export To File
Scan Single Gestures
B
Stop Scan


Sign language Recognition
Sign Language Recognition Using Hand Gestures
Create Gesture
Scan Gestures
Scan Sentence
Export To File
Scan Single Gestures
C
Stop Scan


Sign language Recognition
Sign Language Recognition Using Hand Gestures
Create Gesture
Scan Gestures
Scan Sentence
Export To File
Scan Single Gestures
C
Stop Scan


Sign language Recognition
Sign Language Recognition Using Hand Gestures
Create Gesture
Scan Gestures
Scan Sentence
Export To File
Scan Single Gestures
V
Stop Scan


Sign language Recognition
Sign Language Recognition Using Hand Gestures
Create Gesture
Scan Gestures
Scan Sentence
Export To File
Scan Single Gestures
V
Stop Scan

# Chapter 8

## Conclusion:

From this project/application we have tried to overshadow some of the major problems faced by the disabled persons in terms of talking. We found out the root cause of why they can't express more freely. The result that we got was the other side of the audience are not able to interpret what these persons are trying to say or what is the message that they want to convey.

Thereby this application serves the person who wants to learn and talk in sign languages. With this application a person will quickly adapt various gestures and their meaning as per ASL standards. They can quickly learn what alphabet is assigned to which gesture. Add-on to this custom gesture facility is also provided along with sentence formation. A user need not be a literate person if they know the action of the gesture, they can quickly form the gesture and appropriate assigned character will be shown onto the screen.

The main objective of the project was to develop a system that can translate static sign language into its corresponding word equivalent that includes letters, numbers, and basic static signs to familiarize the users with the fundamentals of sign language. The aim of this project was to find a model with highest accuracy for the task of multi-class classification of American Sign Language. The accuracy recorded for our model is 63.66%. These models can be very effective for the purpose of ASL translation, and for the future, we hope these classification networks can be used, built on further, and even combined with temporal data and recurrent neural networks to learn sequences of words and sentences.

# Appendix

## Methodologies Used include:

### OpenCV:

OpenCV (Open-Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS, and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

### Convolutional Neural Network:

CNNs use a variation of multilayer perceptron's designed to require minimal pre-processing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

### PyQt5:

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android. PyQt is a library that lets you use the Qt GUI framework from Python. Qt itself is written in C++. By using it from Python, you can build applications much more quickly while not sacrificing much of the speed of C++. PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications. PyQt5 refers to the most recent version 5 of Qt.

**Keras:**

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. Keras can be developed in R as well as Python, such that the code can be run with TensorFlow, Theano, CNTK, or MXNet as per the requirement. Keras can be run on CPU, NVIDIA GPU, AMD GPU, TPU, etc. It ensures that producing models with Keras is simple as it totally supports to run with TensorFlow serving, GPU acceleration (WebKeras, Keras.js), Android (TF, TF Lite), iOS (Native CoreML) and Raspberry Pi. It cannot handle low-level computations, so it makes use of the Backend library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

**Tkinter:**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

**TensorFlow:**

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google brain team for internal Google use. It was released under the Apache 2.0 open-source library on November 9, 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

## References

*[1] P. Nanivadekar, V. Kulkarni: Indian Sign Language Recognition: Database Creation, Hand Tracking and Segmentation*

*[2] G. Khurana, G. Joshi, J. Kaur: Static Hand Gesture Recognition System using Shape Based Features*

*[3] M. Mohandes: Image-Based and Sensor-Based Approach to Sign Language Recognition*