Department Of Electronics & Communication Engineering

MANIT BHOPAL

Minor Project Report

Group No. :10

Project Title :- Sign Language Recognition

SUBMITTED BY:-

S. No.	Scholar No.	Name	Signature
1	171114018	PRACHI DEVIKAR	Prachi
2	171114022	URVASHI TOMAR	Tomar
3	171114080	ISHIKA SINGH	[illegible]
4	171114117	PALAK AGRAWAL	[illegible]
5	171114150	SHREYA SOOD	[illegible]

MINOR PROJECT MENTOR

Dr. LALITA GUPTA

Signature:___________

INDEX

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 commonly used among them. However, 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.

The goal of this project is to build a neural network able to classify which letter of the American Sign Language(ASL) alphabet is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written language.

Our problem consists of three tasks to be done in real time:

1. Obtaining video of the user signing (input)
2. Classifying each frame in the video to a letter
3. Reconstructing and displaying the most likely word from classification scores (output)

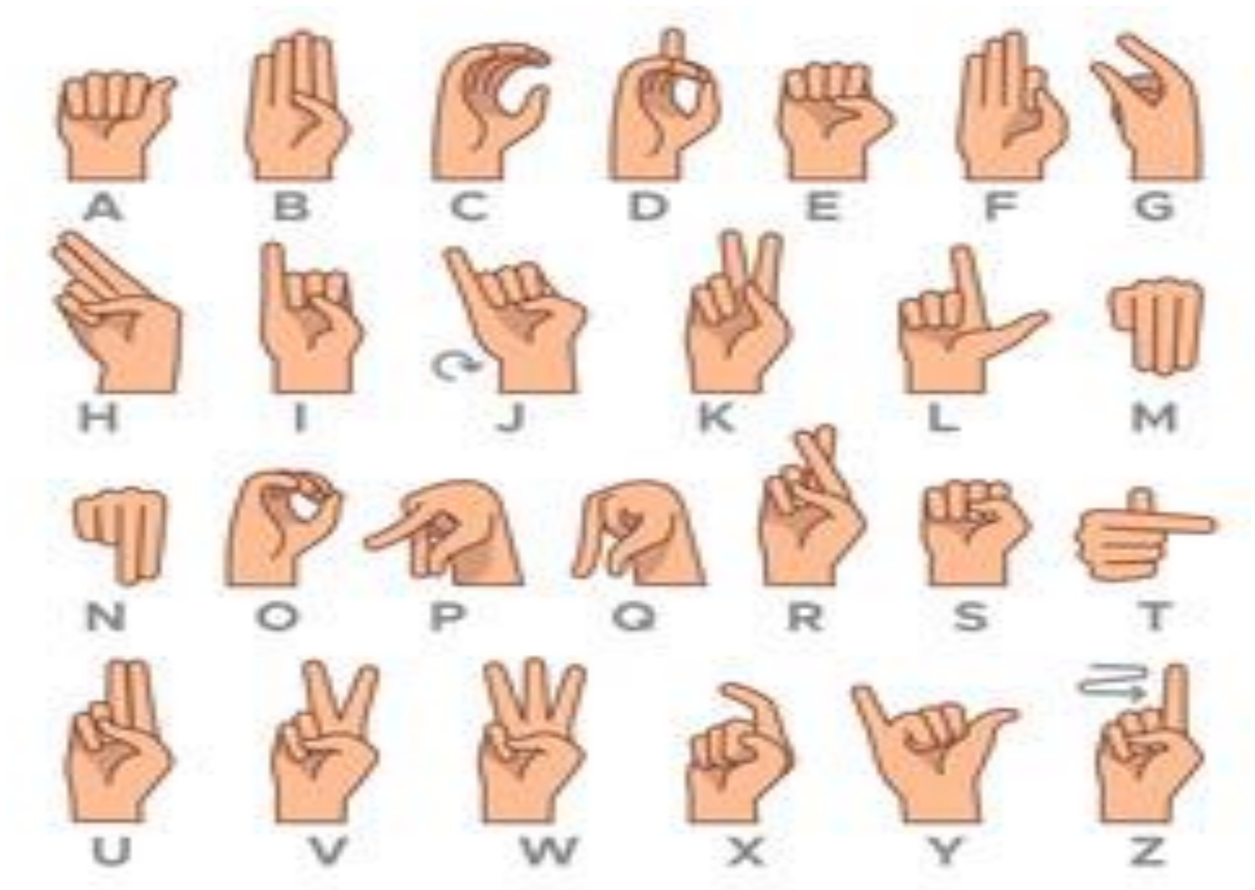


Fig 1. An illustrative Representation Of American Sign Language(ASL)

From vision perspective of a Computer, this problem represents a significant challenge due to a number of considerations, which includes:

- Environmental issues (e.g. sensitivity of light, background, and camera position)
- Occlusion (e.g. some or all fingers, or an entire hand can be out of the field of view)
- Sign boundary detection (when a sign ends and the next begins)
- Co-articulation (when a sign is affected by the preceding or succeeding sign)

Our system features a pipeline that takes video of a user signing a word as input through a web application. We then extract individual frames of the video and

generate letter probabilities for each using a CNN (letters *a* through *y*, excluding *j* and *z* since they require movement). With the use of a variety of heuristics, we group the frames based on the character index that each frame is suspected to correspond to. Finally, we use a language model in order to output a likely word to the user. Our ASL letter classification is done using a convolutional neural network . CNNs are machine learning algorithms that have seen incredible success in handling a variety of tasks related to processing
videos and images. Since 2012, the field has experienced an explosion of growth and applications in image classification, object localization, and object detection.

OBJECTIVE :

Aim of this project is to make a program for detecting American sign language.

BLOCK DIAGRAM:

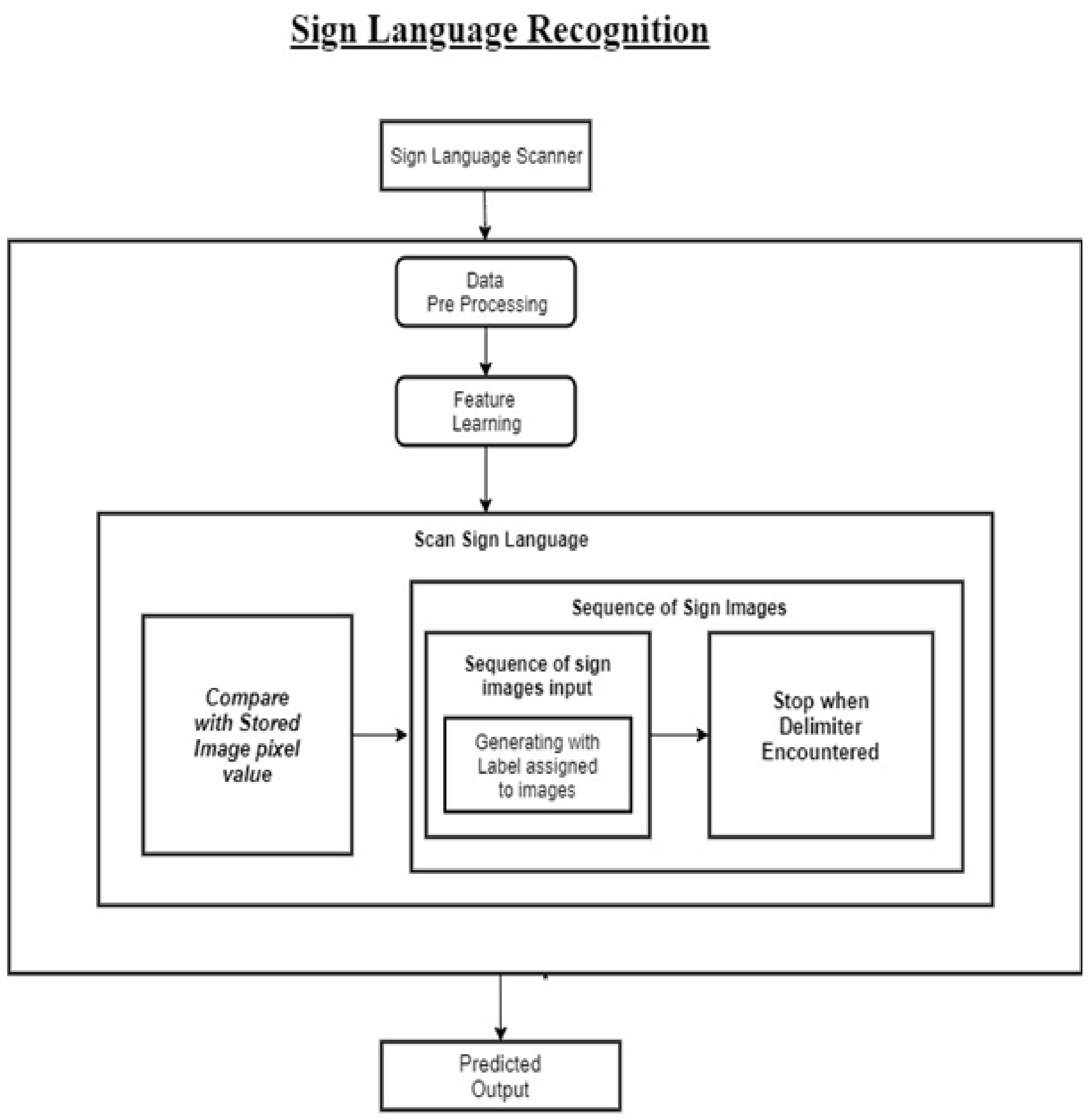

Sign Language Recognition
Sign Language Scanner
Data Pre Processing
Feature Learning
Scan Sign Language
Sequence of Sign Images
Compare with Stored Image pixel value
Sequence of sign images input
Generating with Label assigned to images
Stop when Delimiter Encountered
Predicted Output

Fig 2.Block Diagram showing the necessary steps involved in Sign Language Recognition

DESCRIPTION OF BLOCK DIAGRAM:

Above Block Diagram depicts the ordered set of steps involved in Sign Language Recognition Process.

The detailing of the Sign Language Scanner which is the central part of our Project has been elaborated in order to specify the importance of every process.

Data Pre-Processing – Based on the gesture detected in front of the camera, its binary image is populated (i.e. gesture image will be filled with solid white and background will be filled with solid black pixels). Based on the pixel's regions, their numerical value in range of either 0 or 1 is given to next process.

Feature Learning-Convolutional Neural Network (CNN) is used to build an appropriate model which is trained with the available dataset.

Scan Sign Language – A sign language scanner is available in front of the end user where the user will have to show a sign language gesture. Based on Pre-Processed part output, a user will 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.

DESCRIPTION:

In this project static sign language data in the form of images have been used. We trained a Convolutional Neural Network (CNN) to identify the signs represented by each of these images.

Software Requirements:

- Microsoft Windows XP or later
- Python Interpreter(3.7.2)
- Tensorflow Framework
- Keras API
- Python OpenCV2, scipy, qimage2ndarray, winGuiAuto, pypiwin32, sys, keyboard, pyttsx3, pillow libraries.
- PyQT5

For the detection of movement of gesture, cv2 library is used and an external camera as a hardware requirement is needed.
The frontend is built on PyQT5 which comprise of a module that simply detects the gesture and displays appropriate alphabet.

Following are the details of the steps used for the recognition of Sign Language with desired accuracy:

1)<u>Data Acquisition & Pre-Processing</u>

Sign Language MNIST dataset from Kaggle.com has been used in this project to evaluate models, inorder to classify hand gestures for each letter of the alphabet. The data includes approximately 35,000 28x28 pixel images of the 24 letters of the alphabet.

<u>Pre-Processing</u> is done to remove any unwanted artifacts in the recognition process using appropriate libraries.

The data is classified into training and test dataset,part of which is used to train our Convolutional Neural Network(CNN) Model and the other part is used to predict the Sign Language and hence act as an aid to determine the accuracy of this model which turns out to be 92.3%.

One of the training and test dataset image for the Alphabet ‘A’ is as shown below:

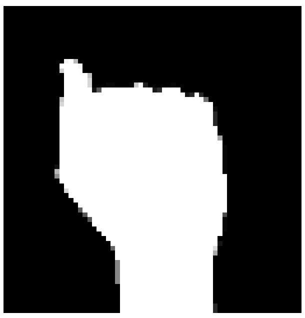


Fig 3. Sample Image of

Letter ‘A’ from training Dataset

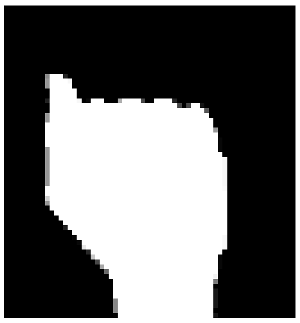


Fig 4. Sample Image of

Letter ‘A’ from test Dataset.

2.Feature Learning

We have used a Convolutional Neural Network, or CNN, model to classify theimages in our dataset.
Our first goal was to define an input layer for our neural network.

A 28x28 image contains 784 pixels each represented by a grayscale value ranging from 0 i.e. black to 1 i.e. white.
By converting each image to a series of numbers, we transformed the data into a format the computer can understand.
Next after preparing the input layer,it was processed by the hidden layers.

The first hidden layer is composed of several nodes each of which takes a weighted sum of the 784 input values. The weighted sum of inputs is then input into an activation function. For our network, we used a rectified linear unit, or ReLU.

As the data moves through more and more hidden layers, the accuracy in the Sign gesture prediction increases.
After passing the data through the Convolution and MaxPool layers,it enters the Flatten and Dense layers of the neural network ,which are responsible for reducing the data to one dimension.

Next we went for the selection of appropriate epoch number(which is the single pass through all the training data),which results in more accurate and less complex model.

Below is the detailed Description of CNN in general and above steps in particular:

CNN(convolutional neural network)

- Convolutional neural network is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organisation of animal visual cortex.

In case of CNN the neuron in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner.

- **Layers in CNN**
  1) Covolutional layer
  2) Relu layer
  3) Pooling
  4) Fully connected layer

- **Problem with normal techniques**

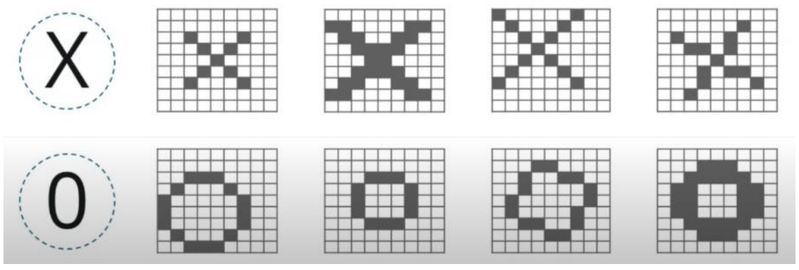


Fig 5

X and O won’t always have same image but can have these deformed images.If the technique is applied for one of the deformed X as shown below

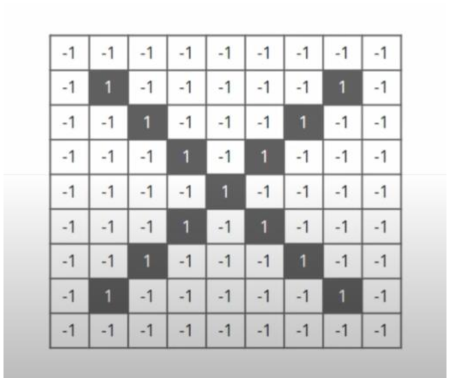


Fig 6

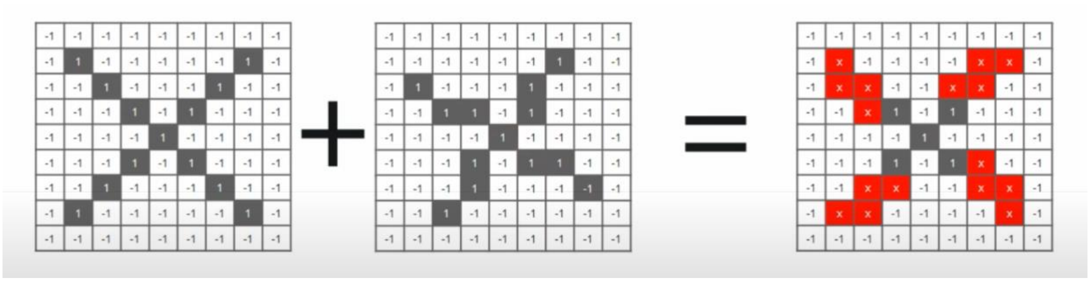


Fig 7

This X will not be detected.

CNN compare the images by features. By finding feature matches in the same position in two images, CNN gets a lot better at seeing similarity than whole-image matching schemes.

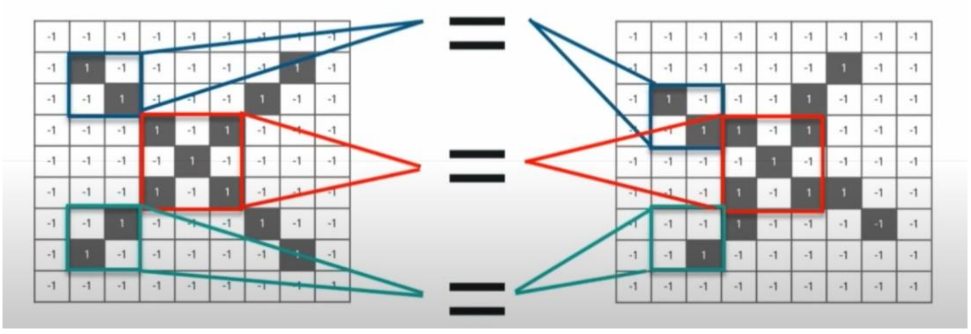
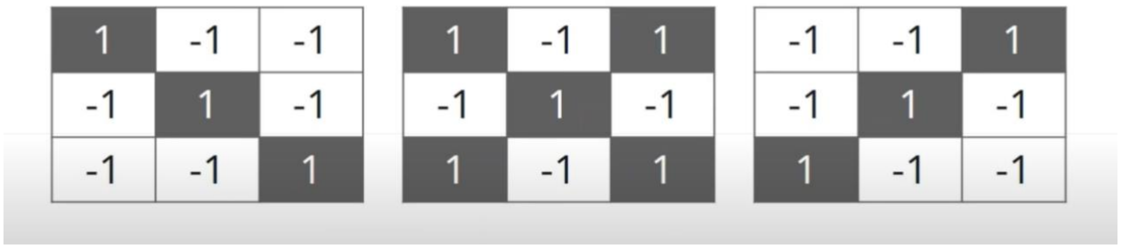


Fig 8

Now the same deformed X can be detected.

- **Detecting X through CNN**

Taking 3 filters as shown below



1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

Fig 9

1)<u>Convolution layer</u>

Moving these filters to every possible position on the image and multiplying the corresponding pixel values

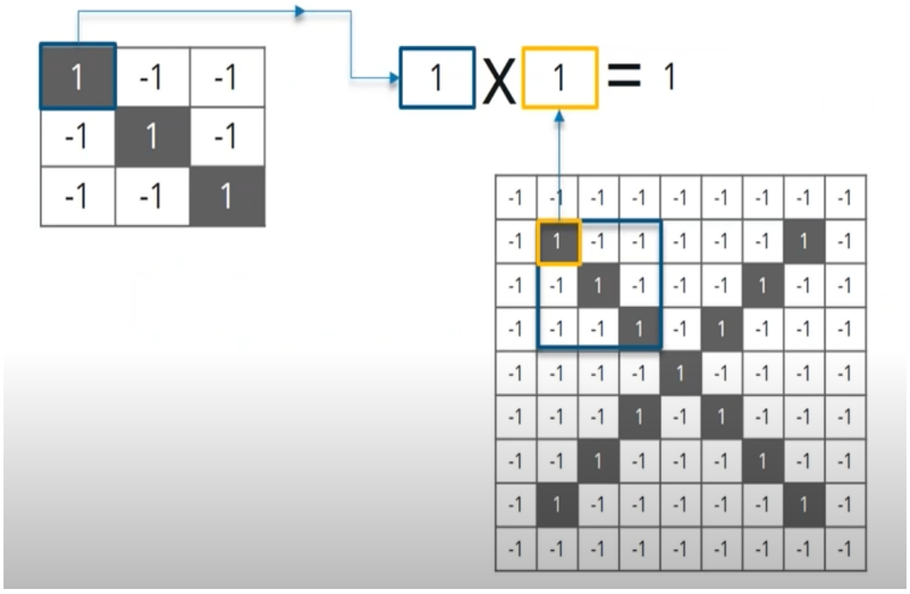


Fig 10

Adding and dividing by the total number of pixels.

Putting the value of that filter at respective place where the feature was.

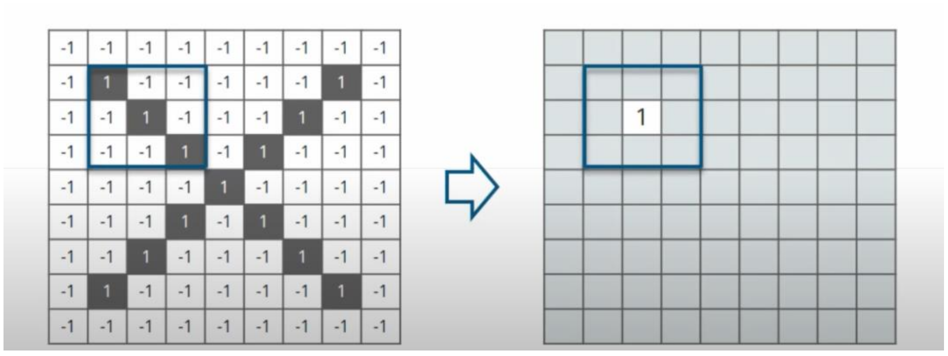


Fig 11

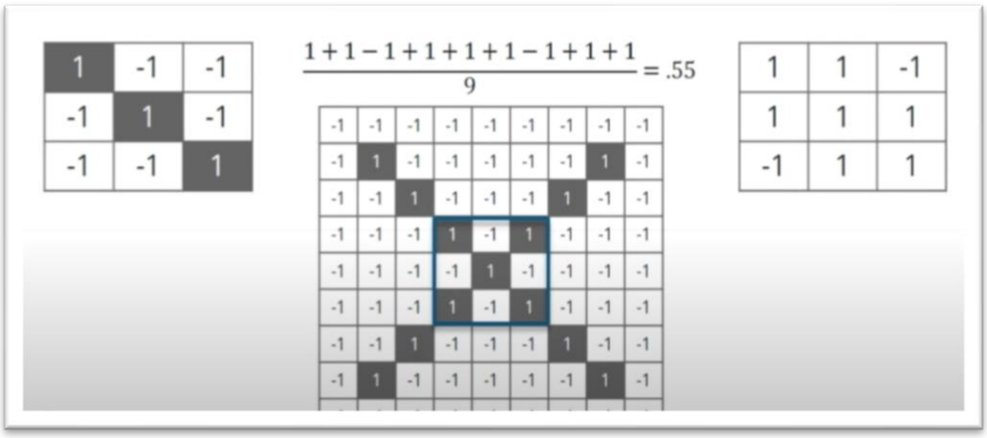
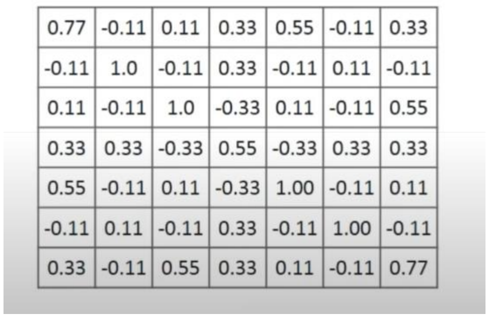


Fig 12



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

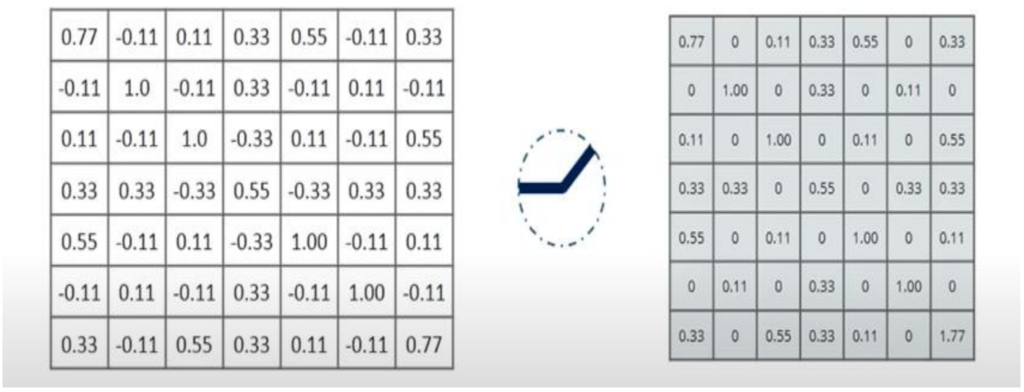
Fig 13

Convolution Output

Similarly applying all 3 filters,the output is as shown above.

2) <u>ReLU layer</u>

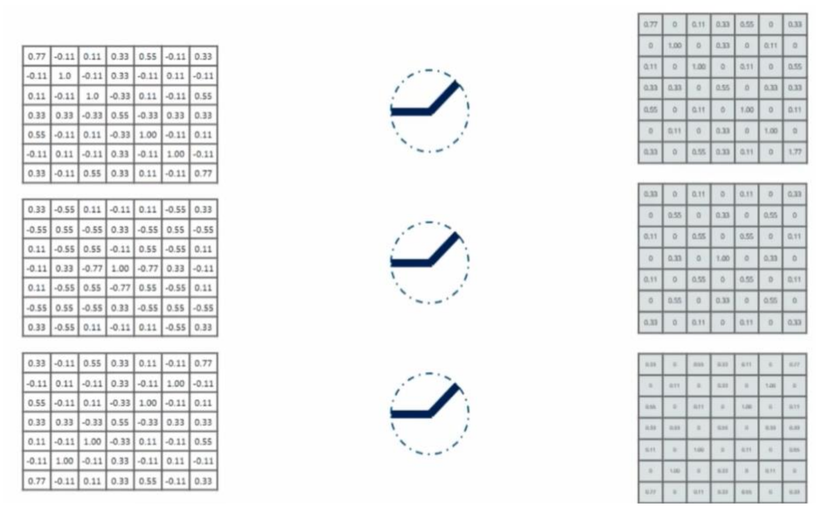
This Layer outputs 0 when the pixel value is negative,while the value remains same otherwise.



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

Fig 14



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.11	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

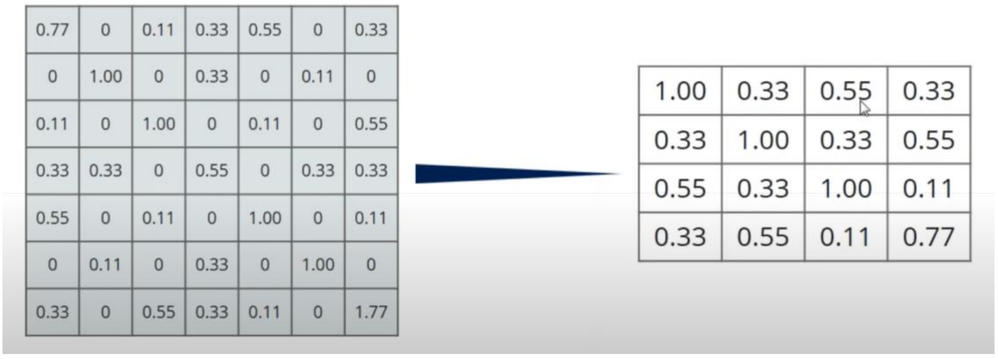
0.33	0	0.11	0	0.11	0	0.33
0	0.55	0	0.33	0	0.55	0
0.11	0	0.55	0	0.55	0	0.11
0	0.33	0	1.00	0	0.33	0
0.11	0	0.55	0	0.55	0	0.11
0	0.55	0	0.33	0	0.55	0
0.33	0	0.11	0	0.11	0	0.33

0.33	0	0.55	0.33	0.11	0	0.77
0	0.11	0	0.33	0	1.00	0
0.55	0	0.11	0	1.00	0	0.11
0.33	0.33	0	0.55	0	0.33	0.33
0.11	0	1.00	0	0.11	0	0.55
0	1.00	0	0.33	0	0.11	0
0.77	0	0.11	0.33	0.55	0	0.33

Fig 15

3) **Pooling layer**

Taking window size 2 and stride 2 and taking maximum value.



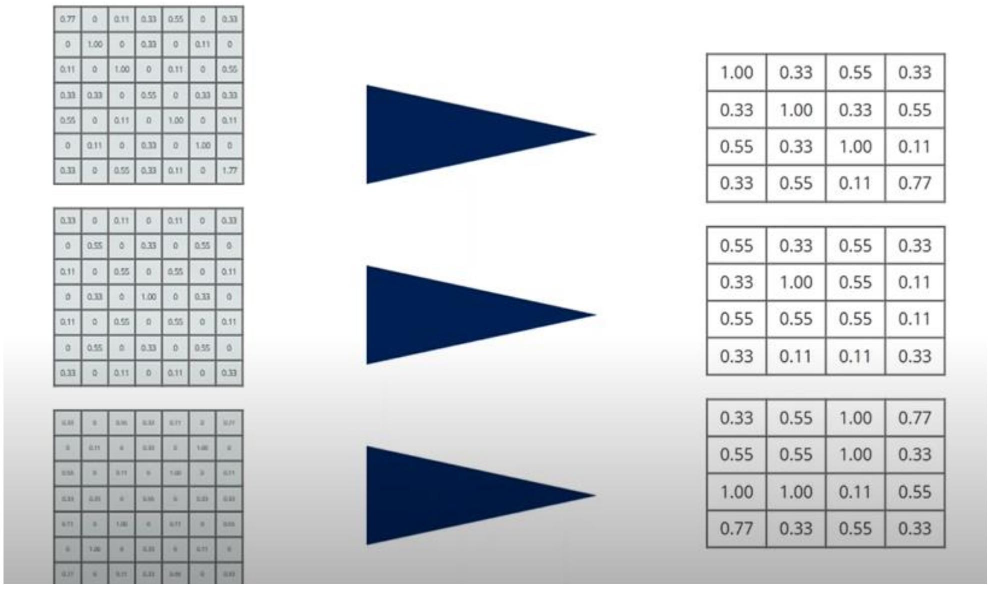
0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

Fig 16

Moving window across the entire image

Pooling output



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.33	0	0.11	0	0.11	0	0.33
0	0.55	0	0.33	0	0.55	0
0.11	0	0.55	0	0.55	0	0.11
0	0.33	0	1.00	0	0.33	0
0.11	0	0.55	0	0.55	0	0.11
0	0.55	0	0.33	0	0.55	0
0.33	0	0.11	0	0.11	0	0.33

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0	0.55	0.33	0.11	0	0.77
0	0.11	0	0.33	0	1.00	0
0.55	0	0.11	0	1.00	0	0.11
0.33	0.33	0	0.55	0	0.33	0.33
0.11	0	1.00	0	0.11	0	0.55
0	1.00	0	0.33	0	0.11	0
0.77	0	0.11	0.33	0.55	0	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Fig 17

Doing all the three steps again we get

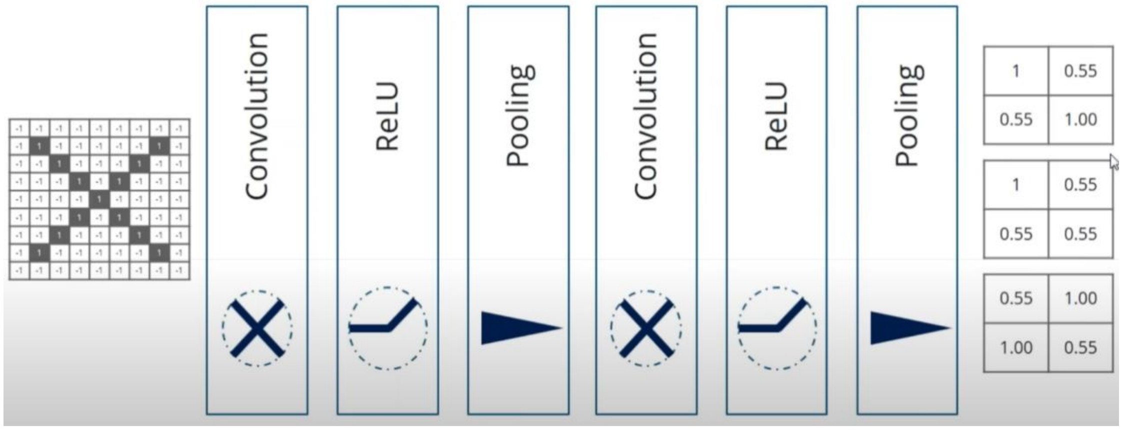


Fig 18

4) <u>Fully connected layer</u>

Taking images and putting them into a list

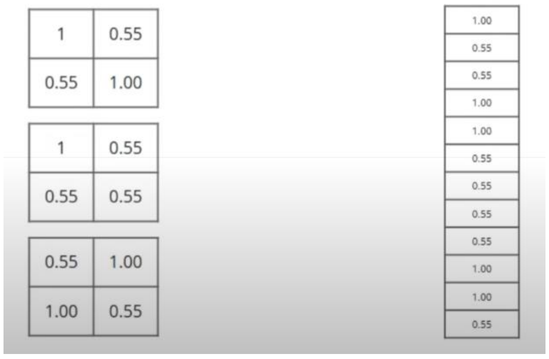


Fig 19

So now this is the data set for X

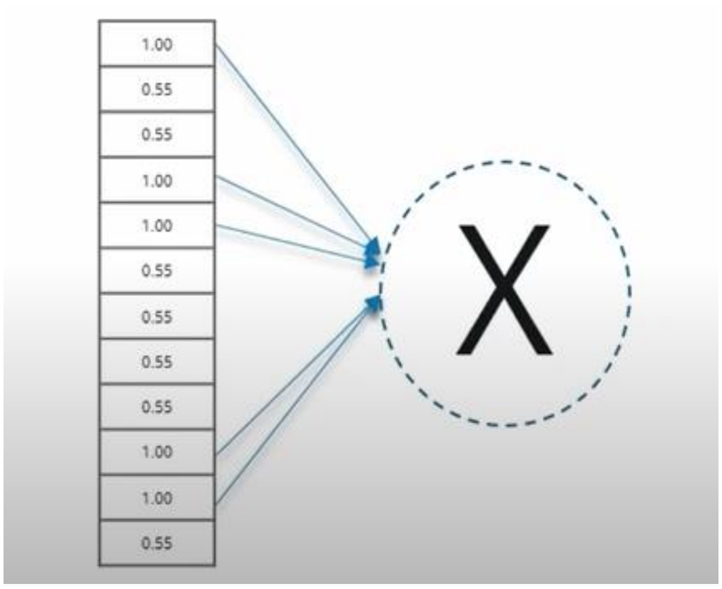


Fig 20

Similarly data set for O will come to be

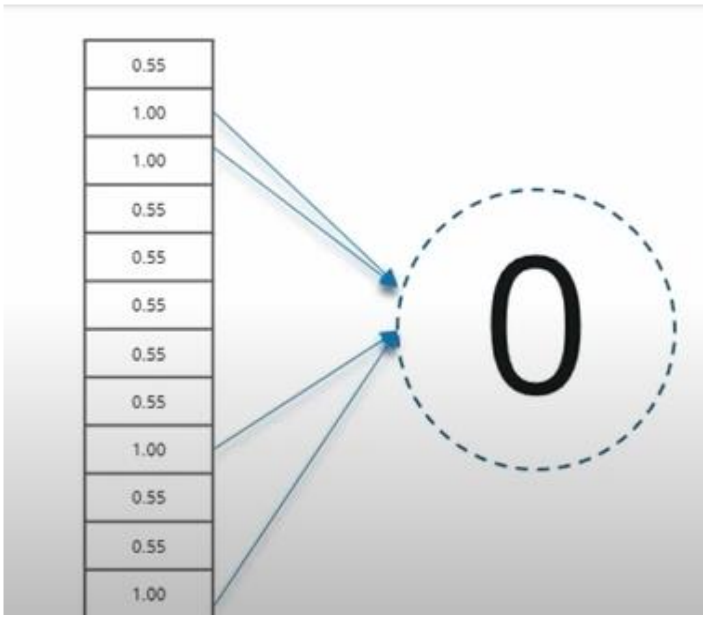
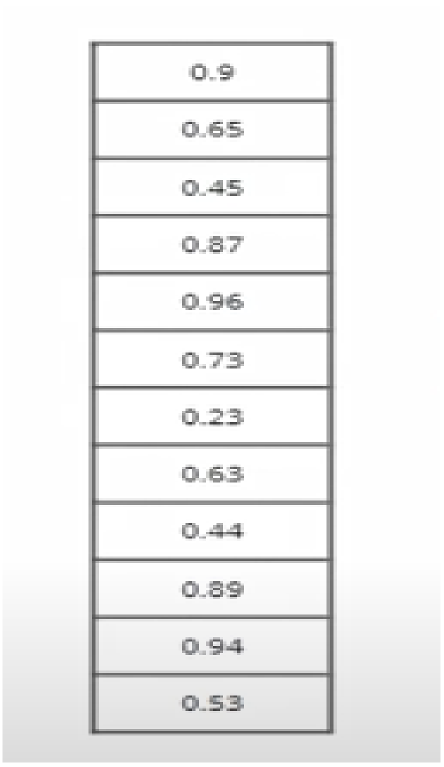


Fig 21

Detecting



0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53

Fig 22

Let this be the image to be detected

Comparison with X is as shown below

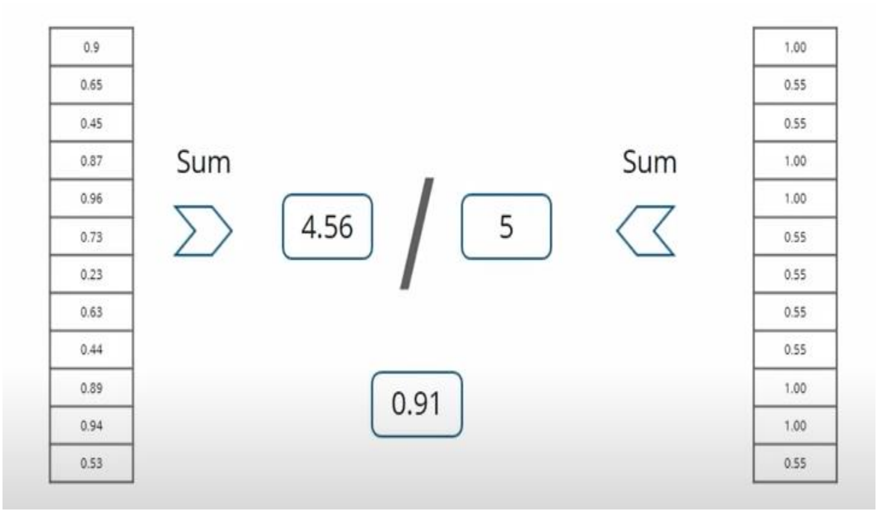


Fig 23

Comparison with O is as shown below

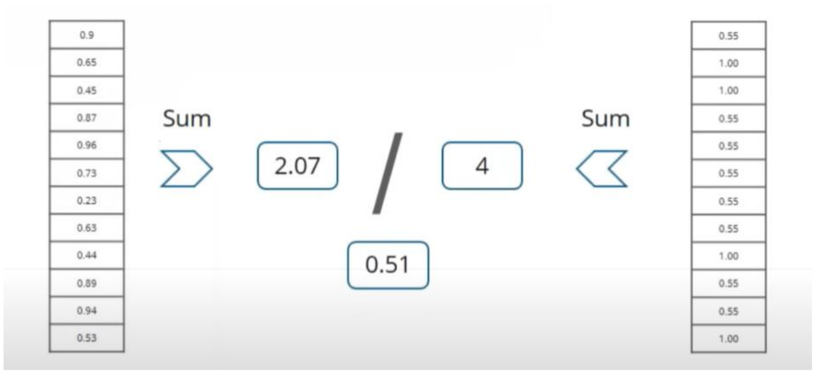


Fig 24

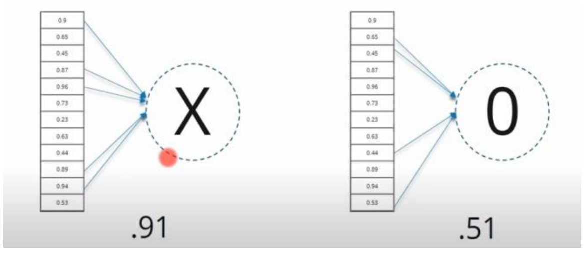


Fig 25

The alignment of data is more towards X ,hence the Input image is classified as X.

APPLICATION OF CNN IN OUR PROJECT :

```
#importing the Keras libraries and packages
from keras.models import Sequential
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()
#BUILDING THE MODEL

# Step 1 - Convolution Layer
# we have added a convolution layer of 32 filters on image size of 64*64 and
corresponding to that we have used the ‘relu’ function
classifier.add(Convolution2D(32, 3,  3, input_shape = (64, 64, 3), activation =
'relu'))

#step 2 – Pooling
# we have used the pooling window of 2*2 on the first layer output
classifier.add(MaxPooling2D(pool_size =(2,2)))

# Adding second convolution layer
# we have added a convolution layer of 32 filters on image and corresponding to
that we have used the ‘relu’ function
classifier.add(Convolution2D(32, 3,  3, activation = 'relu'))

# we have used the pooling window of 2*2 on the first layer output
classifier.add(MaxPooling2D(pool_size =(2,2)))

#Adding 3rd Convolution Layer
# we have added a convolution layer of 64 filters on image and corresponding to
that we have used the ‘relu’ function
classifier.add(Convolution2D(64, 3,  3, activation = 'relu'))

# we have used the pooling window of 2*2 on the first layer output
classifier.add(MaxPooling2D(pool_size =(2,2)))


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

#Step 4 - Full Connection
# we have used fully connected layer of 256 neurons and an activation function
‘relu’.
# we have used a dropout layer with key probability 0.5
classifier.add(Dense(256, activation = 'relu')
classifier.add(Dropout(0.5))
classifier.add(Dense(26, activation = 'softmax'))
#CNN MODEL FINISHED

#Compiling The CNN
#optimizer SGD is used to optimize our model hence to reduce loss with the
learning rate of 0.01
classifier.compile(
optimizer = 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')

#trying to fit the model, we are performing 25 iterations on the model ,we want
the maximum reduction of loss in 25 epochs.
# validation_data is perfomed to test the accuracy of the model.

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()
```

RESULT & CONCLUSION:

Our CNN model for the proposed project was able to identify sign images with 92.3% accuracy.

We believe that further variation in the ratio of training to test data could lead to a more accurate result.

We found this application to be useful to serve the person with impairement of speech. This application can even be used by anyone who want to learn to communicate in Sign Language.

Moreover, A user need not be a literate person. If they know the action of the sign language gesture, they can quickly form it and appropriate assigned character will be shown onto the screen.

Concerning to the implementation, we have used TensorFlow framework, with keras API. And for the user feasibility complete front-end is designed using PyQT5.

We conclude that this application developed by us , with further improvements and advancements could help to bridge the gap this impairement has created.

FUTURE SCOPE & ADVANCEMENTS:

- This Sign Language Recognition Application can be integrated with various search engines and texting application such as google, WhatsApp etc. so that even the illiterate people could be able to chat with others, or query something from web just with the help of gesture.

- This project currently deals with images, which can be further developed by detecting the motion of video sequence and assigning it a meaningful sentence with TTS assistance.

- The project can also help assist the impaired in public places like banks, malls etc. and Public transport as well.

REFRENCES:

- NIDCD, “american sign language”, 2017 [Online]. Available: https://www.nidcd.nih.gov/health/american-sign-language, [Accessed April 06, 2019]

- Suharjito MT, “Sign Language Recognition Application Systems for Deaf-Mute People A Review Based on Input-ProcessOutput”,2017[Online]https://www.academia.edu/35314119/Sign_Language_Recognition_Application_Systems_for_Deaf-Mute_People_A_Review_Based_on_Input-Process-Output

- Suharjito MT, “Sign Language Recognition Application Systems for Deaf-Mute People A Review Based on Input-ProcessOutput”,2017[Online]https://www.academia.edu/35314119/Sign_Language_Recognition_Application_Systems_for_Deaf-Mute_People_A_Review_Based_on_Input-Process-Output

- R. S. Thakur, R. N. Yadav and Lalita Gupta, "State-of-art analysis of image denoising methods using convolutional neural networks," in IET Image Processing, vol. 13, no. 13, pp. 2367-2380, 14 11 2019, doi:10.1049/iet-ipr.2019.0157. https://ieeexplore.ieee.org/document/8911561/references#references

- https://www.kaggle.com/datamunge/sign-language-mnist

CODE:

The Dashboard.ui and Scan.ui are made using Qt ,which is a cross platform application development Framework.

Dashboard.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>MainWindow</class>
<widget class="QMainWindow" name="MainWindow">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>741</width>
<height>642</height>
</rect>
</property>
<property name="windowTitle">
<string>MainWindow</string>
</property>
<widget class="QWidget" name="centralwidget">
<widget class="QGraphicsView" name="graphicsView_2">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>741</width>
<height>161</height>
</rect>
</property>
<property name="toolTip">
<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;br/&gt;&lt;/p&gt;&lt;/bod
y&gt;&lt;/html&gt;</string>
</property>
<property name="styleSheet">
<string notr="true">background-color: qlineargradient(spread:pad, x1:0.543136,
y1:0.074, x2:0.52, y2:1, stop:0.880682 rgba(216, 91, 216, 255), stop:1 rgba(255, 255, 255,
255));
</string>
</property>
</widget>
<widget class="QLabel" name="label">
<property name="geometry">
<rect>
<x>130</x>
<y>10</y>
<width>531</width>
```

```
<height>31</height>
</rect>
</property>
<property name="font">
<font>
<family>Times New Roman</family>
<pointsize>18</pointsize>
<weight>75</weight>
<bold>true</bold>
</font>
</property>
<property name="styleSheet">
<string notr="true">color:white;</string>
</property>
<property name="text">
<string>Sign Language Recognition</string>
</property>
</widget>
<widget class="QLabel" name="label_3">
<property name="geometry">
<rect>
<x>0</x>
<y>160</y>
<width>741</width>
<height>431</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">
font-size: 40px;
color:white;
</string>
</property>
<property name="text">
<string/>
</property>
</widget>
<widget class="QGraphicsView" name="graphicsView">
<property name="geometry">
<rect>
<x>0</x>
<y>160</y>
<width>741</width>
<height>441</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">background-color:#fefefe;
</string>
</property>
```

```
</widget>
<widget class="QPushButton" name="scan_single">
<property name="geometry">
<rect>
<x>190</x>
<y>70</y>
<width>171</width>
<height>61</height>
</rect>
</property>
<property name="font">
<font>
<family>TImes New Roman</family>
<pointsize>12</pointsize>
<weight>50</weight>
<italic>false</italic>
<bold>false</bold>
</font>
</property>
<property name="layoutDirection">
<enum>Qt::LeftToRight</enum>
</property>
<property name="autoFillBackground">
<bool>false</bool>
</property>
<property name="styleSheet">
<string notr="true">QPushButton#scan_single{
padding-left:-40px;
color:black;
background-color:#fefefe;
border: 2px solid #58e870; border-radius:30px;
font: 12pt &quot;TImes New Roman&quot;;
background-image:url(&quot;icons/scan.png&quot;);
background-repeat:none;
}
QPushButton#scan_single:hover
{
  background-color:lightgreen;
  color:white;
}</string>
</property>
<property name="text">
<string>Scan</string>
</property>
</widget>
<widget class="QPushButton" name="exit_button">
<property name="geometry">
<rect>
<x>710</x>
<y>2</y>
```

```
<width>31</width>
<height>31</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">QPushButton#exit_button{
background-image:url(&quot;icons/quit.png&quot;);
background-repeat:none;
}</string>
</property>
<property name="text">
<string/>
</property>
</widget>
</widget>
<widget class="QMenuBar" name="menubar">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>741</width>
<height>21</height>
</rect>
</property>
</widget>
<widget class="QStatusBar" name="statusbar"/>
</widget>
<resources/>
<connections/>
</ui>
```

Scan.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>MainWindow</class>
<widget class="QMainWindow" name="MainWindow">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>741</width>
<height>642</height>
</rect>
</property>
<property name="windowTitle">
<string>MainWindow</string>
</property>
<widget class="QWidget" name="centralwidget">
<widget class="QGraphicsView" name="graphicsView">
```

```
<property name="geometry">
<rect>
<x>0</x>
<y>160</y>
<width>741</width>
<height>441</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">background-color:#fefefe;
border: 5px solid #4ba1df;</string>
</property>
</widget>
<widget class="QLabel" name="label">
<property name="geometry">
<rect>
<x>280</x>
<y>170</y>
<width>181</width>
<height>31</height>
</rect>
</property>
<property name="font">
<font>
<family>Times New Roman</family>
<pointsize>14</pointsize>
<weight>75</weight>
<bold>true</bold>
</font>
</property>
<property name="styleSheet">
<string notr="true">color:black;
background-color: white;
border: 2px solid white; border-radius:10px;</string>
</property>
<property name="text">
<string>Scan</string>
</property>
</widget>
<widget class="QTextBrowser" name="textBrowser">
<property name="enabled">
<bool>true</bool>
</property>
<property name="geometry">
<rect>
<x>380</x>
<y>210</y>
<width>331</width>
<height>271</height>
</rect>
```

```
</property>
<property name="styleSheet">
<string notr="true">border: 5px solid #4ba1df;
background-color:black;
font-size: 40px;
color:white;
</string>
</property>
<property name="frameShape">
<enum>QFrame::Box</enum>
</property>
<property name="html">
<string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg 2'; font-
size:40px; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-
size:8.25pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
</property>
</widget>
<widget class="QPushButton" name="pushButton_2">
<property name="geometry">
<rect>
<x>500</x>
<y>500</y>
<width>91</width>
<height>41</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">QPushButton#pushButton_2{
color: black;
background-color:#fefefe;
border: 2px solid #4ba1df; border-radius:20px;
font: 16pt &quot;Times New Roman&quot;;
background-repeat:none;
}
QPushButton#pushButton_2:hover
{
 background-color:lightgreen;
 color:black;
}</string>
</property>
<property name="text">
<string>Stop</string>
</property>
```

```
</widget>
<widget class="QGraphicsView" name="graphicsView_3">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>741</width>
<height>161</height>
</rect>
</property>
<property name="toolTip">
<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;br/&gt;&lt;/p&gt;&lt;/bod
y&gt;&lt;/html&gt;</string>
</property>
<property name="styleSheet">
<string notr="true">background-color: qlineargradient(spread:pad, x1:0.543136,
y1:0.074, x2:0.52, y2:1, stop:0.880682 rgba(216, 91, 216, 255), stop:1 rgba(255, 255, 255,
255));
</string>
</property>
</widget>
<widget class="QLabel" name="label_2">
<property name="geometry">
<rect>
<x>130</x>
<y>10</y>
<width>531</width>
<height>31</height>
</rect>
</property>
<property name="font">
<font>
<family>Times New Roman</family>
<pointsize>18</pointsize>
<weight>75</weight>
<bold>true</bold>
</font>
</property>
<property name="styleSheet">
<string notr="true">color:white;</string>
</property>
<property name="text">
<string>Sign Language Recognition</string>
</property>
</widget>
<widget class="QLabel" name="label_3">
<property name="geometry">
<rect>
<x>40</x>
<y>210</y>
```

```xml
<width>321</width>
<height>271</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">border: 5px solid #4ba1df;
font-size: 40px;
color:white;
</string>
</property>
<property name="text">
<string>TextLabel</string>
</property>
</widget>
<widget class="QLabel" name="label_5">
<property name="geometry">
<rect>
<x>31</x>
<y>490</y>
<width>331</width>
<height>51</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">border: 5px solid #4ba1df;
font-size: 40px;
color:white;
</string>
</property>
<property name="text">
<string>TextLabel</string>
</property>
</widget>
<widget class="QSlider" name="trackbar">
<property name="geometry">
<rect>
<x>50</x>
<y>500</y>
<width>281</width>
<height>31</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">color: rgb(0, 0, 0);</string>
</property>
<property name="maximum">
<number>255</number>
</property>
<property name="singleStep">
<number>1</number>
```

```xml
</property>
<property name="orientation">
<enum>Qt::Horizontal</enum>
</property>
<property name="invertedAppearance">
<bool>false</bool>
</property>
<property name="invertedControls">
<bool>false</bool>
</property>
</widget>
<widget class="QTextBrowser" name="textBrowser_3">
<property name="enabled">
<bool>true</bool>
</property>
<property name="geometry">
<rect>
<x>570</x>
<y>210</y>
<width>141</width>
<height>131</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">border: 5px solid #4ba1df; border-radius:20px;
background-color:black;</string>
</property>
<property name="frameShape">
<enum>QFrame::Box</enum>
</property>
<property name="html">
<string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg 2'; font-
size:8.25pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
</property>
</widget>
<widget class="QCommandLinkButton" name="linkButton">
<property name="geometry">
<rect>
<x>700</x>
<y>170</y>
<width>31</width>
<height>31</height>
```

```
</rect>
</property>
<property name="text">
<string>CommandLinkButton</string>
</property>
</widget>


<widget class="QPushButton" name="scan_single">
<property name="geometry">
<rect>
<x>190</x>
<y>70</y>
<width>171</width>
<height>61</height>
</rect>
</property>
<property name="font">
<font>
<family>TImes New Roman</family>
<pointsize>12</pointsize>
<weight>50</weight>
<italic>false</italic>
<bold>false</bold>
</font>
</property>
<property name="layoutDirection">
<enum>Qt::LeftToRight</enum>
</property>
<property name="autoFillBackground">
<bool>false</bool>
</property>
<property name="styleSheet">
<string notr="true">QPushButton#scan_single{
padding-left:-40px;
color:black;
background-color:#fefefe;
border: 2px solid #58e870; border-radius:30px;
font: 12pt &quot;TImes New Roman&quot;;
background-image:url(&quot;icons/scan.png&quot;);
background-repeat:none;
}
QPushButton#scan_single:hover
{
   background-color:lightgreen;
  color:white;
}</string>
</property>
<property name="text">
<string>Scan</string>
```

```
</property>
</widget>

<widget class="QPushButton" name="exit_button">
<property name="geometry">
<rect>
<x>710</x>
<y>2</y>
<width>31</width>
<height>31</height>
</rect>
</property>
<property name="styleSheet">
<string notr="true">QPushButton#exit_button{
background-image:url(&quot;icons/quit.png&quot;);
background-repeat:none;
}</string>
</property>
<property name="text">
<string/>
</property>
</widget>
</widget>
<widget class="QMenuBar" name="menubar">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>741</width>
<height>21</height>
</rect>
</property>
</widget>
<widget class="QStatusBar" name="statusbar"/>
</widget>
<resources/>
<connections/>
</ui>
```

<u>Test.py:</u>

```
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 scipy.ndimage import imread          #will help in reading the images
from scipy import ndimage
from PyQt5.QtCore import QTimer,Qt
from PyQt5 import QtGui
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                       #converts 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 tensorflow.keras.models import load_model
classifier = load_model('model.h5')         #loading the model

def fileSearch():
"""Searches each file ending with .png in SampleGestures directory so that custom
gesture could be passed to predictor() function"""
fileEntry=[]
for file in os.listdir("SampleGestures"):
if file.endswith(".png"):
        fileEntry.append(file)
return fileEntry

def load_images_from_folder(folder):
"""Searches each images in a specified directory"""
images = []
for filename in os.listdir(folder):
      img = cv2.imread(os.path.join(folder,filename))
if img is not None:
          images.append(img)
return images

def toggle_imagesfwd(event):
```

```
"""displays next images act as a gesutre viewer"""
img=load_images_from_folder('TempGest/')
global index

  += 1

try:
if index < len(img):
      plt.axes()
      plt.imshow(img[index])
      plt.draw()
except:
pass

def toggle_imagesrev(event):
"""displays previous images act as a gesutre viewer"""
img=load_images_from_folder('TempGest/')
global index

  -= 1

try:
if index < len(img) and index>=0:
      plt.axes()
      plt.imshow(img[index])
      plt.draw()
except:
pass

def openimg():
"""displays predefined gesture images at right most window"""
cv2.namedWindow("Image", cv2.WINDOW_NORMAL )
  image = cv2.imread('template.png')
  cv2.imshow("Image",image)

cv2.setWindowProperty("Image",cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FUL
LSCREEN)
  cv2.resizeWindow("Image",298,430)
  cv2.moveWindow("Image", 1052,214)




def removeFile():
"""Removes the temp.txt and tempgest directory if any stop button is pressed oor
application is closed"""
try:
    os.remove("temp.txt")
except:
pass
```

```
  try:
    shutil.rmtree("TempGest")
except:
pass

def clearfunc(cam):
"""shut downs the opened camera and calls removeFile() Func"""
cam.release()
  cv2.destroyAllWindows()
  removeFile()

def clearfunc2(cam):
"""shut downs the opened camera"""
cam.release()
  cv2.destroyAllWindows()


def controlTimer(self):
# if timer is stopped
self.timer.isActive()
# create video capture
self.cam = cv2.VideoCapture(0)
# start timer
self.timer.start(20)


def predictor():
""" Depending on model loaded and customgesture saved prediction is made by checking
array or through SiFt algo"""
import numpy as np
from tensorflow.keras.preprocessing import image
  test_image = image.load_img('1.png', target_size=(64, 64))
  test_image = image.img_to_array(test_image)
  test_image = np.expand_dims(test_image, axis = 0)
  result = classifier.predict(test_image)
gesname=''
fileEntry=fileSearch()
for i in range(len(fileEntry)):
    image_to_compare = cv2.imread("./SampleGestures/"+fileEntry[i])
    original = cv2.imread("1.png")
    sift = cv2.xfeatures2d.SIFT_create()
    kp_1, desc_1 = sift.detectAndCompute(original, None)
    kp_2, desc_2 = sift.detectAndCompute(image_to_compare, None)

    index_params = dict(algorithm=0, trees=5)
    search_params = dict()
    flann = cv2.FlannBasedMatcher(index_params, search_params)

    matches = flann.knnMatch(desc_1, desc_2, k=2)
    good_points = []
    ratio = 0.6
for m, n in matches:
if m.distance < ratio*n.distance:
            good_points.append(m)
if(abs(len(good_points)+len(matches))>20):      #goodpoints and matcches sum from
1.png and customgestureimages is grater than 20
gesname=fileEntry[i]
      gesname=gesname.replace('.png','')
if(gesname=='sp'):                    #sp is replaced with <space>
gesname=' '
return gesname

if result[0][0] == 1:
return 'A'
elif result[0][1] == 1:
return 'B'
elif result[0][2] == 1:
return 'C'
elif result[0][3] == 1:
return 'D'
elif result[0][4] == 1:
return 'E'
elif result[0][5] == 1:
return 'F'
elif result[0][6] == 1:
return 'G'
elif result[0][7] == 1:
return 'H'
elif result[0][8] == 1:
return 'I'
elif result[0][9] == 1:

return 'J'
elif result[0][10] == 1:
return 'K'
elif result[0][11] == 1:
return 'L'
elif result[0][12] == 1:
return 'M'
elif result[0][13] == 1:
return 'N'
elif result[0][14] == 1:
return 'O'
elif result[0][15] == 1:
return 'P'
elif result[0][16] == 1:
return 'Q'
elif result[0][17] == 1:
return 'R'
```

```
elif result[0][18] == 1:
return 'S'
elif result[0][19] == 1:
return 'T'
elif result[0][20] == 1:
return 'U'
elif result[0][21] == 1:
return 'V'
elif result[0][22] == 1:
return 'W'
elif result[0][23] == 1:
return 'X'
elif result[0][24] == 1:
return 'Y'
elif result[0][25] == 1:
return 'Z'


def checkFile():
"""retrieve the content of temp.txt for export module """
checkfile=os.path.isfile('temp.txt')
if(checkfile==True):
    fr=open("temp.txt","r")
    content=fr.read()
    fr.close()
else:
    content="No Content Available"
return content

class Dashboard(QtWidgets.QMainWindow):
def __init__(self):
    super(Dashboard, self).__init__()
self.setWindowFlags(QtCore.Qt.WindowMinimizeButtonHint)


# Closes all the frames
cv2.destroyAllWindows()
self.setWindowIcon(QtGui.QIcon('icons/windowLogo.png'))
self.title = 'Sign language Recognition'
uic.loadUi('UI_Files/dash.ui', self)
self.setWindowTitle(self.title)
self.timer = QTimer()
if(self.scan_single.clicked.connect(self.scanSingle)==True):
self.timer.timeout.connect(self.scanSingle)
self.scan_single.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.exit_button.clicked.connect(self.quitApplication)
self._layout = self.layout()
self.label_3 = QtWidgets.QLabel()
    movie = QtGui.QMovie("icons/dashAnimation.gif")
self.label_3.setMovie(movie)
self.label_3.setGeometry(0,160,780,441)
      movie.start()
self._layout.addWidget(self.label_3)
self.setObjectName('Message_Window')

def quitApplication(self):
"""shutsdown the GUI window along with removal of files"""
userReply = QMessageBox.question(self, 'Quit Application', "Are you sure you want to
quit this app?", QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
if userReply == QMessageBox.Yes:
         removeFile()
         keyboard.press_and_release('alt+F4')

def gestureViewer(self):
"""gesture viewer through matplotlib """
try:
         img=load_images_from_folder('TempGest/')
         plt.imshow(img[index])
except:
         plt.text(0.5, 0.5, 'No new Gesture Available',
horizontalalignment='center',verticalalignment='center')
      axcut = plt.axes([0.9, 0.0, 0.1, 0.075])
      axcut1 = plt.axes([0.0, 0.0, 0.1, 0.075])
      bcut = Button(axcut, 'Next', color='dodgerblue', hovercolor='lightgreen')
      bcut1 = Button(axcut1, 'Previous', color='dodgerblue', hovercolor='lightgreen')

#plt.connect('button_press_event', toggle_imagesfwd)
bcut.on_clicked(toggle_imagesfwd)
      bcut1.on_clicked(toggle_imagesrev)
      plt.show()
      axcut._button = bcut      #creating a reference for that element
axcut1._button1 = bcut1
#buttonaxe._button = bcut

def scanSingle(self):
"""Single gesture scanner """
try:
         clearfunc(self.cam)
except:
pass
uic.loadUi('UI_Files/scan_single.ui', self)
self.setWindowTitle(self.title)
if(self.scan_single.clicked.connect(self.scanSingle)):
         controlTimer(self)
self.pushButton_2.clicked.connect(lambda:clearfunc(self.cam))
self.linkButton.clicked.connect(openimg)
self.scan_single.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
try:
self.exit_button.clicked.connect(lambda:clearfunc(self.cam))
except:
```

```
pass
self.exit_button.clicked.connect(self.quitApplication)
    img_text = ''
while True:
        ret, frame = self.cam.read()
        frame = cv2.flip(frame,1)
try:
            frame=cv2.resize(frame,(321,270))
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            img1 = cv2.rectangle(frame, (150,50),(300,200), (0,255,0), thickness=2,
lineType=8, shift=0)
except:
            keyboard.press_and_release('esc')

        height1, width1, channel1 = img1.shape
        step1 = channel1 * width1
# create QImage from image
qImg1 = QImage(img1.data, width1, height1, step1, QImage.Format_RGB888)
# show image in img_label
try:
self.label_3.setPixmap(QPixmap.fromImage(qImg1))
            slider1=self.trackbar.value()
except:
pass

lower_blue = np.array([0, 0, 0])
        upper_blue = np.array([179, 255, slider1])

        imcrop = img1[52:198, 152:298]
        hsv = cv2.cvtColor(imcrop, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv, lower_blue, upper_blue)

        cv2.namedWindow("mask", cv2.WINDOW_NORMAL )
        cv2.imshow("mask", mask)

cv2.setWindowProperty("mask",cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FUL
LSCREEN)
        cv2.resizeWindow("mask",118,108)
        cv2.moveWindow("mask", 894,271)

        hwnd = winGuiAuto.findTopWindow("mask")
        win32gui.SetWindowPos(hwnd, win32con.HWND_TOP,
0,0,0,0,win32con.SWP_NOMOVE | win32con.SWP_NOSIZE |
win32con.SWP_NOACTIVATE)

try:
self.textBrowser.setText("\n\n\t"+str(img_text))
except:
pass
```

```
img_name = "1.png"
save_img = cv2.resize(mask, (image_x, image_y))
        cv2.imwrite(img_name, save_img)
        img_text = predictor()

if cv2.waitKey(1) == 27:
break

self.cam.release()
      cv2.destroyAllWindows()

app = QtWidgets.QApplication([])
win = Dashboard()
win.show()
sys.exit(app.exec())
```

Recognise.py:

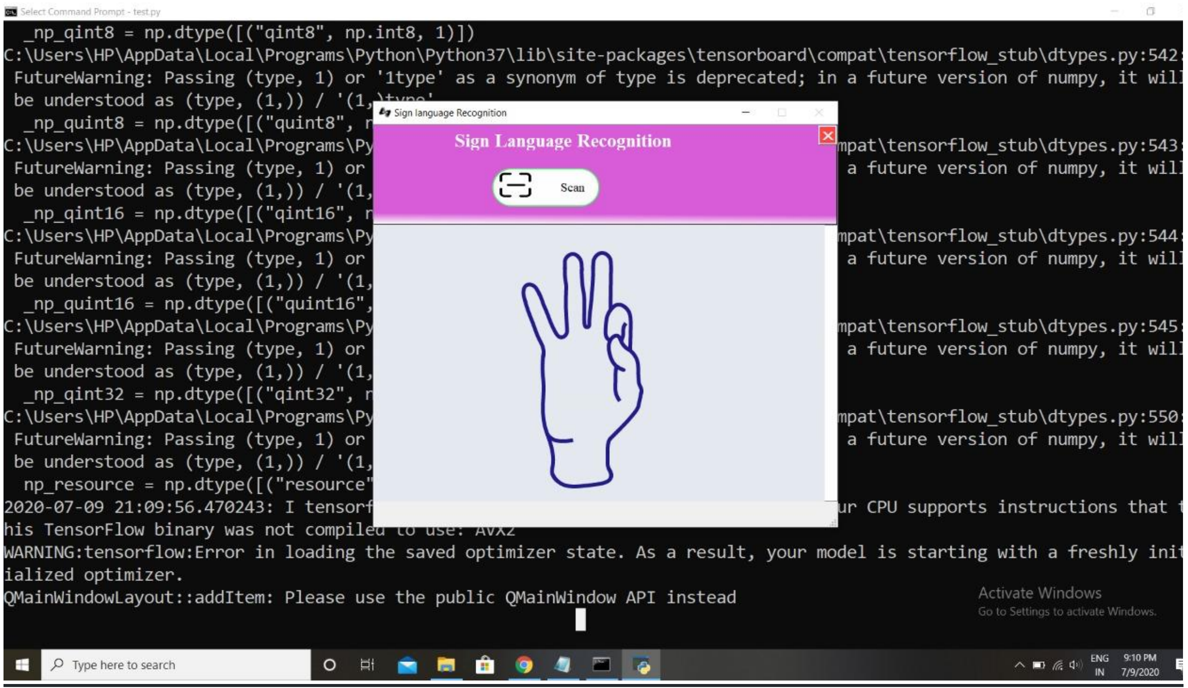
```
import cv2
import numpy as np
import os


def nothing(x):
pass


image_x, image_y = 64, 64

from keras.models import load_model

classifier = load_model('model.h5')

# original = cv2.imread("5.png")
fileEntry = []
for file in os.listdir("SampleGestures"):
if file.endswith(".png"):
       fileEntry.append(file)


def imgprocessing():
    image_to_compare = cv2.imread("./SampleGestures/space.png")
    original = cv2.imread("1.png")
    sift = cv2.xfeatures2d.SIFT_create()
    kp_1, desc_1 = sift.detectAndCompute(original, None)
    kp_2, desc_2 = sift.detectAndCompute(image_to_compare, None)

    index_params = dict(algorithm=0, trees=5)
    search_params = dict()
    flann = cv2.FlannBasedMatcher(index_params, search_params)

    matches = flann.knnMatch(desc_1, desc_2, k=2)

    good_points = []
    ratio = 0.6
for m, n in matches:
if m.distance < ratio * n.distance:
            good_points.append(m)
# print(len(good_points))
fin = len(kp_1) - len(kp_2)
    print(abs(fin))
if (abs(fin) <= 2):
return 'space'

result = cv2.drawMatches(original, kp_1, image_to_compare, kp_2, good_points, None)

# cv2.imshow("result", result)
    # cv2.imshow("Original", original)
    # cv2.imshow("Duplicate", image_to_compare)
    # cv2.destroyAllWindows()


def predictor():
import numpy as np
from keras.preprocessing import image
    test_image = image.load_img('1.png', target_size=(64, 64))
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)
    result = classifier.predict(test_image)
for i in range(len(fileEntry)):
        image_to_compare = cv2.imread("./SampleGestures/" + fileEntry[i])
        original = cv2.imread("1.png")
        sift = cv2.xfeatures2d.SIFT_create()
        kp_1, desc_1 = sift.detectAndCompute(original, None)
        kp_2, desc_2 = sift.detectAndCompute(image_to_compare, None)

        index_params = dict(algorithm=0, trees=5)
        search_params = dict()
        flann = cv2.FlannBasedMatcher(index_params, search_params)

        matches = flann.knnMatch(desc_1, desc_2, k=2)

        good_points = []
        ratio = 0.6
for m, n in matches:
if m.distance < ratio * n.distance:
                good_points.append(m)
        print(fileEntry[i])
```

```
if (abs(len(good_points) + len(matches)) >20):
            gesname = fileEntry[i]
            gesname = gesname.replace('.png', '')
return gesname
# print(good_points)
    # if(abs(fin)<=2):
    # return 'space'
if result[0][0] == 1:
return 'A'
elif result[0][1] == 1:
return 'B'
elif result[0][2] == 1:
return 'C'
elif result[0][3] == 1:
return 'D'
elif result[0][4] == 1:
return 'E'
elif result[0][5] == 1:
return 'F'
elif result[0][6] == 1:
return 'G'
elif result[0][7] == 1:
return 'H'
elif result[0][8] == 1:
return 'I'
elif result[0][9] == 1:
return 'J'
elif result[0][10] == 1:
return 'K'
elif result[0][11] == 1:
return 'L'
elif result[0][12] == 1:
return 'M'
elif result[0][13] == 1:
return 'N'
elif result[0][14] == 1:
return 'O'
elif result[0][15] == 1:
return 'P'
elif result[0][16] == 1:
return 'Q'
elif result[0][17] == 1:
return 'R'
elif result[0][18] == 1:
return 'S'
elif result[0][19] == 1:
return 'T'
elif result[0][20] == 1:
return 'U'
elif result[0][21] == 1:
```

```
return 'V'
elif result[0][22] == 1:
return 'W'
elif result[0][23] == 1:
return 'X'
elif result[0][24] == 1:
return 'Y'
elif result[0][25] == 1:
return 'Z'


cam = cv2.VideoCapture(0)

cv2.namedWindow("Trackbars")

cv2.createTrackbar("L - H", "Trackbars", 0, 179, nothing)
cv2.createTrackbar("L - S", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("L - V", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("U - H", "Trackbars", 179, 179, nothing)
cv2.createTrackbar("U - S", "Trackbars", 255, 255, nothing)
cv2.createTrackbar("U - V", "Trackbars", 255, 255, nothing)

cv2.namedWindow("ASL Recognition")

img_text = ''
img_text1 = ''
# z5=1
while True:
    ret, frame = cam.read()
    frame = cv2.flip(frame, 1)
    l_h = cv2.getTrackbarPos("L - H", "Trackbars")
    l_s = cv2.getTrackbarPos("L - S", "Trackbars")
    l_v = cv2.getTrackbarPos("L - V", "Trackbars")
    u_h = cv2.getTrackbarPos("U - H", "Trackbars")
    u_s = cv2.getTrackbarPos("U - S", "Trackbars")
    u_v = cv2.getTrackbarPos("U - V", "Trackbars")

    img = cv2.rectangle(frame, (425, 425), (625, 300), (0, 255, 0), thickness=2, lineType=8,
shift=0)

    lower_blue = np.array([l_h, l_s, l_v])
    upper_blue = np.array([u_h, u_s, u_v])
    imcrop = img[102:298, 427:623]
    hsv = cv2.cvtColor(imcrop, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, lower_blue, upper_blue)

    cv2.putText(frame, img_text, (30, 400), cv2.FONT_HERSHEY_TRIPLEX, 1.5, (0,
255, 0))
    cv2.putText(frame, img_text1, (80, 400), cv2.FONT_HERSHEY_TRIPLEX, 1.5, (0,
255, 0))
```

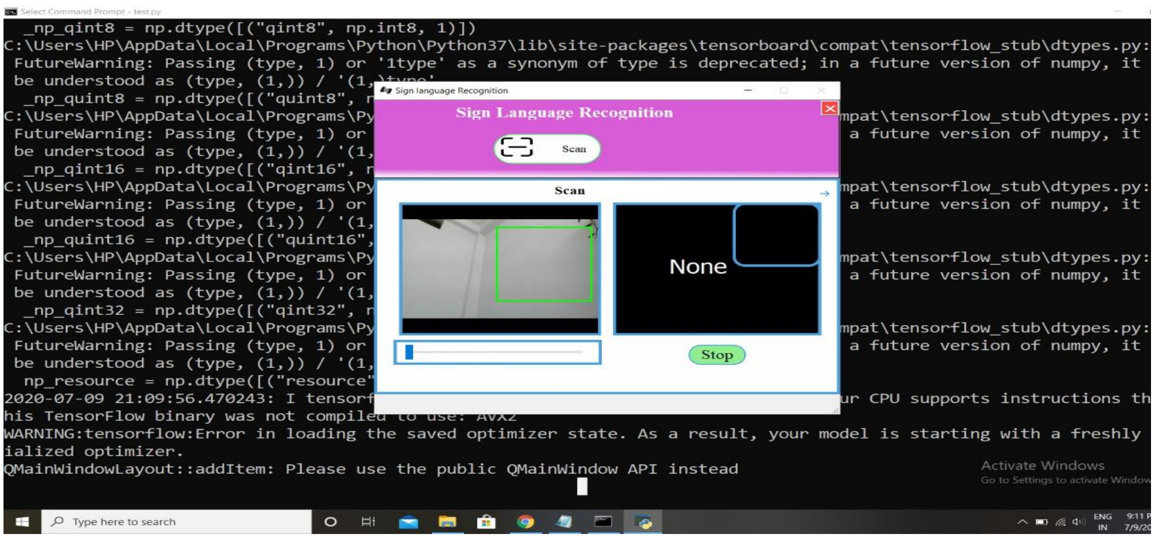
```
    cv2.imshow("ASL Recognition", frame)
    cv2.imshow("mask", mask)

    img_name = "1.png"
save_img = cv2.resize(mask, (image_x, image_y))
    cv2.imwrite(img_name, save_img)
    img_text = predictor()

if cv2.waitKey(1) == 27:
break

# img_text1 = imgprocessing()


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

SNAPSHOTS OF PROJECT:


Select Command Prompt - test.py
Sign language Recognition
Sign Language Recognition
Scan
Type here to search
Activate Windows
Go to Settings to activate Windows.
ENG IN
9:10 PM
7/9/2020

```
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\HP\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542:
 FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will
 be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8",
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:543:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it will
 be understood as (type, (1,)) / '(1,
  _np_qint16 = np.dtype([("qint16", r
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:544:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it will
 be understood as (type, (1,)) / '(1,
  _np_quint16 = np.dtype([("quint16",
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:545:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it will
 be understood as (type, (1,)) / '(1,
  _np_qint32 = np.dtype([("qint32", r
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:550:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it will
 be understood as (type, (1,)) / '(1,
  np_resource = np.dtype([("resource"
2020-07-09 21:09:56.470243: I tensorf                                      ur CPU supports instructions that t
his TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly init
ialized optimizer.
QMainWindowLayout::addItem: Please use the public QMainWindow API instead
```

Fig 26. An Introductory Graphic (gif)


Select Command Prompt - test.py
Sign language Recognition
Sign Language Recognition
Scan
Scan
None
Stop
Type here to search
Activate Windows
Go to Settings to activate Window
ENG IN
9:11 P
7/9/20

```
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\HP\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:
 FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it
 be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8",
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it
 be understood as (type, (1,)) / '(1,
  _np_qint16 = np.dtype([("qint16", r
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it
 be understood as (type, (1,)) / '(1,
  _np_quint16 = np.dtype([("quint16",
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it
 be understood as (type, (1,)) / '(1,
  _np_qint32 = np.dtype([("qint32", r
C:\Users\HP\AppData\Local\Programs\Py                                      mpat\tensorflow_stub\dtypes.py:
 FutureWarning: Passing (type, 1) or                                        a future version of numpy, it
 be understood as (type, (1,)) / '(1,
  np_resource = np.dtype([("resource"
2020-07-09 21:09:56.470243: I tensorf                                      ur CPU supports instructions th
his TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly
ialized optimizer.
QMainWindowLayout::addItem: Please use the public QMainWindow API instead
```

Fig 27.User Interface:Left(To Scan the Sign Language Gesture) Right(Predicted Sign Language ‘Output’)