A Project/Dissertation Report

on

Gesticulation Recognition System

Submitted in partial fulfillment of the
requirement for the award of the degree
of

B. Tech CSE with Specialization in AIML

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

Mr. Samson Ebenezar U

Assistant
Professor

Submitted By

Akshat Sharma
19SCSE1180127

Atul Kumar Prasad
19SCSE1180130

SCHOOL OF COMPUTING SCIENCE AND ENGINEERINGDEPARTMENTOFCOMPUTERSCIENCEANDENGINEERINGGALGOTIASUNIVERSITY, GREATER NOIDA INDIA

DECEMBER
2021

Table of content

Abstract

We are aiming to build a web application in order to overcome the disabilities of deaf and dumb people to make sure they don't feel like straggling behind than the rest of the world. The problem with the current gesticulation recognition application is that they are incompetent, as a there is not one but many types of sign languages approximately 300 different types of sign languages all around the globe. The earlier systems were developed under the functioning of MATLAB, but it’s not efficient.

With the growth in globalization and emphasis on educational advancements, we want to make sure that no one should face any kind of hindrance regardless of one's in capabilities. We will be letting the user select which sign language they are comfortable with, initially however we will have a few sign languages available for the users and with further updates we will add more with given proper time.

Initially we will be starting with 2 different types of sign language datasets ASL (American Sign Language) dataset and ISL (Indian Sign Language) dataset. CNN (Convolutional Neural Network is best suited for the purpose of model training and gesture prediction which is a type of machine learning neural network. Language used is Python.

This web application will be reading user's hand gestures and after analysis it will convert that sign in text as well as speech format so they can communicate easily with everyone around them.

This project will not only be used in academic fields but also in day-to-day life as well because communication is must to survive. This gesticulation recognition system that we are building is only limited to English language as of now, but we will be looking forward to expand its linguistic basket to have multiple language conversion in the near future

Sample dataset figure

Introduction

With the growth in globalization and emphasis on educational advancements, we want to make sure that no one should face any kind of hindrance regardless of one's in capabilities. We are aiming to build a web application in order to overcome the disabilities of deaf and dumb people to make sure they don't feel like straggling behind than the rest of the world, as education define one's personality. This project will not only be used in academic fields but also in day-to-day life as well because communication is must to survive. This web application will be reading user's hand gestures and after analysis it will convert that sign in text as well as speech format so they can communicate easily with everyone around them. This gesture conversion that we are building is only limited to English language as of now, but we will be looking forward to expand its linguistic basket to have multiple language conversion in the near future.

The problem with the current gesticulation recognition application is that they are incompetent, as a there is not one but many types of sign languages approximately 300 different types of sign languages all around the globe. The earlier systems were developed under the functioning of MATLAB, but it’s not efficient.

Motivation

Need for a very efficient gesticulation recognition system is must approximately 7 million people of the whole Indian’s population are deaf and dumb. It’s very important that this part of India as well as in whole world needs to be brought and taught equally. We are hoping our web application will be helpful not only in educational sector but also in day to day life use as well i.e. in communication of mentally challenged people with people who are not mentally challenged and also between two mentally challenged people.

Problem Formulation and Remedy

There are currently a few gesticulation recognition/ sign language detection systems that are not very efficient and limited to a certain type of sign language out of hundreds of types of sign languages. In earlier sign language detection system user had to wear a green glove to separate the hand from the background in order to be detected and recognize which alphabet or word the user is trying to make.

Also because of the so many different types languages all the sign language detection systems are limited by a linguistic barrier because all different types of sign languages uses different types of symbols to represent the same alphabet , digit , or words and every different languages has a different set of grammar rules.

Our gesticulation recognition system will not need any special equipment that is needed to be worn by user/users. We will be preprocessing image by reading the image and reshaping the images to equal size and removing noise making our system work without more easily without needing any type of equipment.

Our gesticulation recognition system will initially have two different types of sign languages ASL (American Sign Language) dataset and ISL (Indian Sign Language) dataset.This gesticulation recognition system that we are building is only limited to English language as of now, but we will be looking forward to expand its linguistic basket to have multiple language conversion in the near future.

Tools and Technology Used

- Python- We will be using python as our main source coding language. We will be implementing a few specific libraries like scikitlearn and few more.
- Neural Network- Our whole gesticulation recognition system is based on neural network more specifically artificial neural network (ANN). We will be using CNN (Convolution Neural Network) which is type of ANN, best suited for image classification.
- Tensorflow- It is a very important type of library in python when it comes to implementing machine learning or artificial intelligence especially deep learning.
- IDE- Jupyter Integrated Development Environment is what we will be using for building our web application.
- Web Cam – Web camera is needed for taking and reading the image from the user.
- Keras – It is also very important python library that is needed for creating an interface for ANNs. It provides an interface for tensorflow to work.

Literature survey/ Project Design

Our work mainly focuses on static sign language detection for ASL and ISL. There are two types of sign languages static sign language and dynamic sign language, static sign language involves separate symbols like a particular alphabet or digit it is comparatively easier to recognize than dynamic sign language which involves words and often complete sentences.

Our gesticulation recognition system will be designed to initially ask the user which sign language user the user is comfortable in and then read, then recognize and classify it into its respective category and then convert that recognized symbol into respective text and speech format.

We will be starting with 2 different types of sign language datasets ASL (American Sign Language) dataset and ISL (Indian Sign Language) dataset. CNN (Convolutional Neural Network is best suited for the purpose of model training and gesture prediction which is a type of machine learning neural network. Language used is Python.

We will be letting the user select which sign language they are comfortable with, initially however we will have a few sign languages available for the users and with further updates we will add more with given proper time.

Implementation and Source Code

#Import necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

#Load the dataset

```
train = pd.read_csv('sign_mnist_train.csv')
test = pd.read_csv('sign_mnist_test.csv')
```

```
train.head()
```

Out[4]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	3	107	118	127	134	139	143	146	150	153	...	207	207	207	207	206	206	206	204	203	202
1	6	155	157	156	156	156	157	156	158	158	...	69	149	128	87	94	163	175	103	135	149
2	2	187	188	188	187	187	186	187	188	187	...	202	201	200	199	198	199	198	195	194	195
3	2	211	211	212	212	211	210	211	210	210	...	235	234	233	231	230	226	225	222	229	163
4	13	164	167	170	172	176	179	180	184	185	...	92	105	105	108	133	163	157	163	164	179

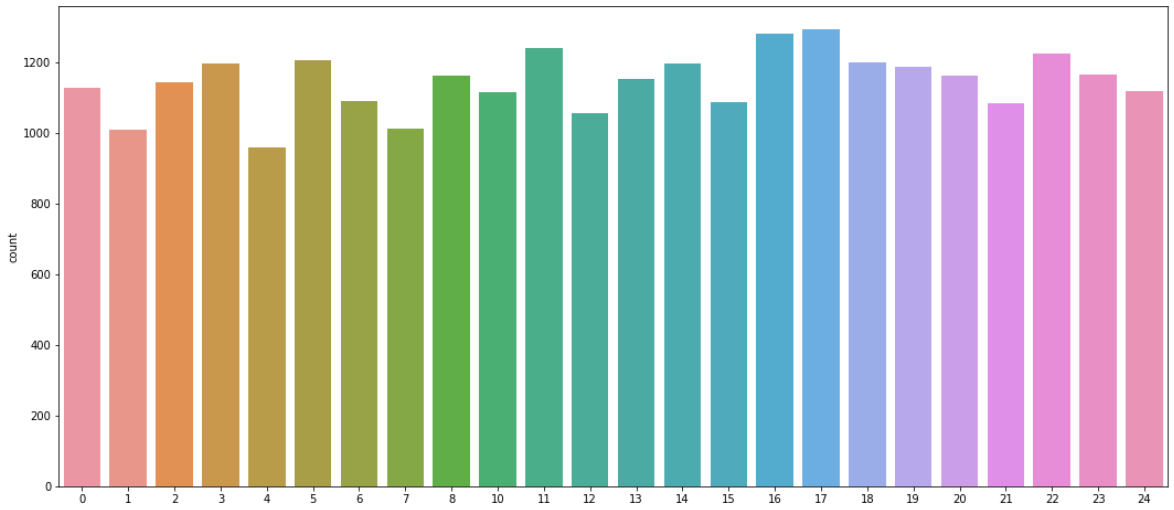
5 rows × 785 columns

```
# get our training labels
labels = train['label'].values

#get the unique labels, 24 in total
unique_val = np.array(labels)
np.unique(unique_val)
```

```
[out]array([ 0,  1,  2,  3,  4,  5,  6,  7,  8, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24], dtype=int64)
```

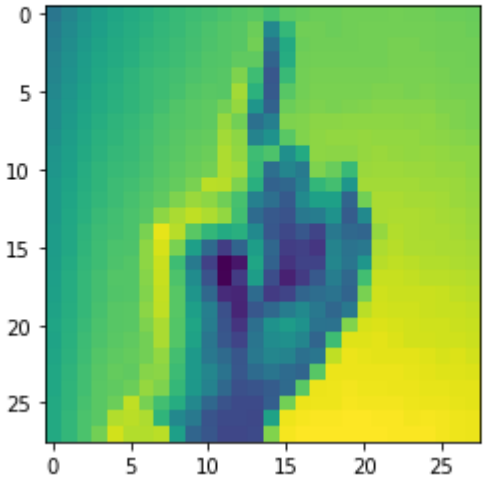
```
#plot the quanties in each class
plt.figure(figsize=(18,8))
sns.countplot(x=labels)
```

Out[7]: <AxesSubplot:ylabel='count'>




```
#drop training labels from our training data so we can separate it
train.drop('label',axis=1,inplace=True)

#extract the image data from each row in our csv, remember it's in a row
of 784 columns

images = train.values
images = np.array([np.reshape(i,(28,28)) for i in images])
images = np.array([i.flatten() for i in images])

#hot one incode our labels
from sklearn.preprocessing import LabelBinarizer

label_binrizer = LabelBinarizer()
labels = label_binrizer.fit_transform(labels)

# view our labels
labels

[out] array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0]])

# inspect an image
index = 0
print(labels[index])
plt.imshow(images[index].reshape(28,28))
```

```
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
<matplotlib.image.AxesImage at 0x243802b9a00>
```
Out[12]:




```
#use opencv to view 10 random images from our training data
import cv2
import numpy as np

for i in range(0,10):
    rand = np.random.randint(0,len(images))
    input_im = images[rand]

    sample = input_im.reshape(28,28).astype(np.uint8)
    sample = cv2.resize(sample, None, fx=10, fy=10,interpolation =
cv2.INTER_CUBIC)
    cv2.imshow("sample image",sample)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

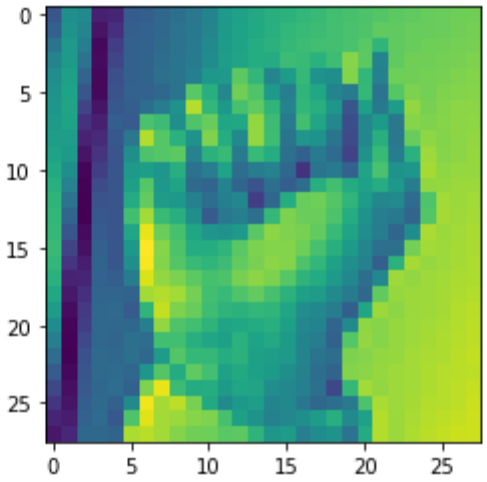
```
# split our data into x_test,y_train and y_test
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(images,labels,test_size =
0.3,random_state = 101)
```

```
# start loading our tensorflow modules and define our batch size
etc
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,Dropout

batch_size = 128
num_classes = 24
epochs = 10
```

```
#scale our image
x_train = x_train/255
x_test = x_test/255
```

```
#reshape them into the size required by tf and keras
x_train = x_train.reshape(x_train.shape[0],28,28,1)
x_test = x_test.reshape(x_test.shape[0],28,28,1)

plt.imshow(x_train[0].reshape(28,28))
```

Out[16]: <matplotlib.image.AxesImage at 0x20a1682e0d0>




```
# create our CNN Model
from tensorflow.keras.layers import Conv2D,MaxPooling2D
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Conv2D(64,kernel_size=(3,3),activation='relu',input_shape=(28,28,1)
))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.20))

model.add(Dense(num_classes,activation='softmax'))


#compile our model
model.compile(loss =
'categorical_crossentropy',optimizer=Adam(),metrics=['accuracy'])
```

```
print(model.summary())
```

[out]

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 128)               8320

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 24)                3096

=================================================================
Total params: 85,912
Trainable params: 85,912
Non-trainable params: 0
_________________________________________________________________
None
```

```
#train our model
history =
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=epochs,batch_s
ize=batch_size)
```

[out]

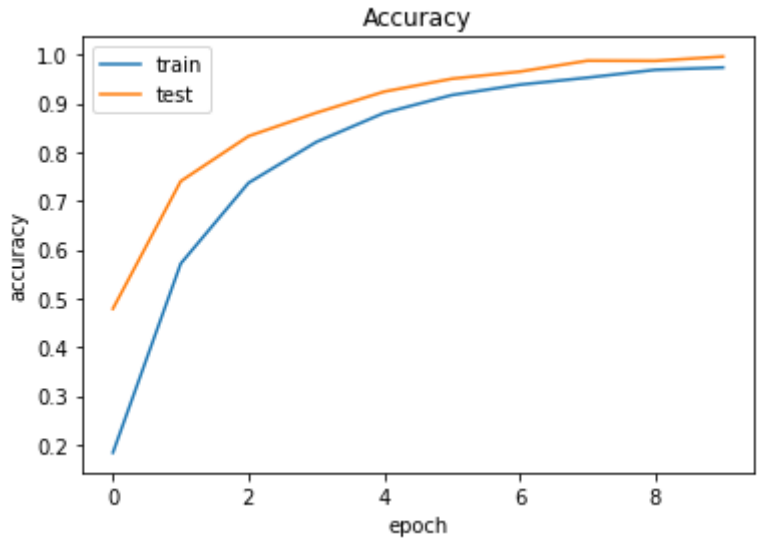
```
Epoch 1/10
151/151 [==============================] - 69s 437ms/step - loss: 2.6858 -
accuracy: 0.1835 - val_loss: 1.6268 - val_accuracy: 0.4787
Epoch 2/10
151/151 [==============================] - 65s 430ms/step - loss: 1.2733 -
accuracy: 0.5711 - val_loss: 0.8176 - val_accuracy: 0.7402
Epoch 3/10
```

```
151/151 [==============================] - 64s 421ms/step - loss: 0.7697 -
accuracy: 0.7368 - val_loss: 0.5300 - val_accuracy: 0.8326
Epoch 4/10
151/151 [==============================] - 64s 422ms/step - loss: 0.5275 -
accuracy: 0.8206 - val_loss: 0.3705 - val_accuracy: 0.8805
Epoch 5/10
151/151 [==============================] - 62s 411ms/step - loss: 0.3547 -
accuracy: 0.8803 - val_loss: 0.2434 - val_accuracy: 0.9239
Epoch 6/10
151/151 [==============================] - 61s 407ms/step - loss: 0.2566 -
accuracy: 0.9169 - val_loss: 0.1676 - val_accuracy: 0.9503
Epoch 7/10
151/151 [==============================] - 61s 405ms/step - loss: 0.1929 -
accuracy: 0.9381 - val_loss: 0.1274 - val_accuracy: 0.9649
Epoch 8/10
151/151 [==============================] - 61s 404ms/step - loss: 0.1490 -
accuracy: 0.9526 - val_loss: 0.0694 - val_accuracy: 0.9874
Epoch 9/10
151/151 [==============================] - 65s 431ms/step - loss: 0.1073 -
accuracy: 0.9686 - val_loss: 0.0631 - val_accuracy: 0.9870
Epoch 10/10
151/151 [==============================] - 66s 437ms/step - loss: 0.0918 -
accuracy: 0.9734 - val_loss: 0.0358 - val_accuracy: 0.9958
```

```
#save our model
model.save("sign_mnist_cnn_50_Epochs.h5")
print("Model Saved")
```

```
#view our training history graphically
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(['train','test'])

plt.show()
```

[out]




```
#reshape our test data so that we can evaluate it's performance on unseen data
test_labels = test['label']
test.drop('label',axis = 1,inplace=True)

test_images = test.values
test_images = np.array([np.reshape(i,(28,28)) for i in test_images])
test_images = np.array([i.flatten() for i in test_images])

test_labels = label_binrizer.fit_transform(test_labels)

test_images = test_images.reshape(test_images.shape[0],28,28,1)

test_images.shape

y_pred = model.predict(test_images)
```

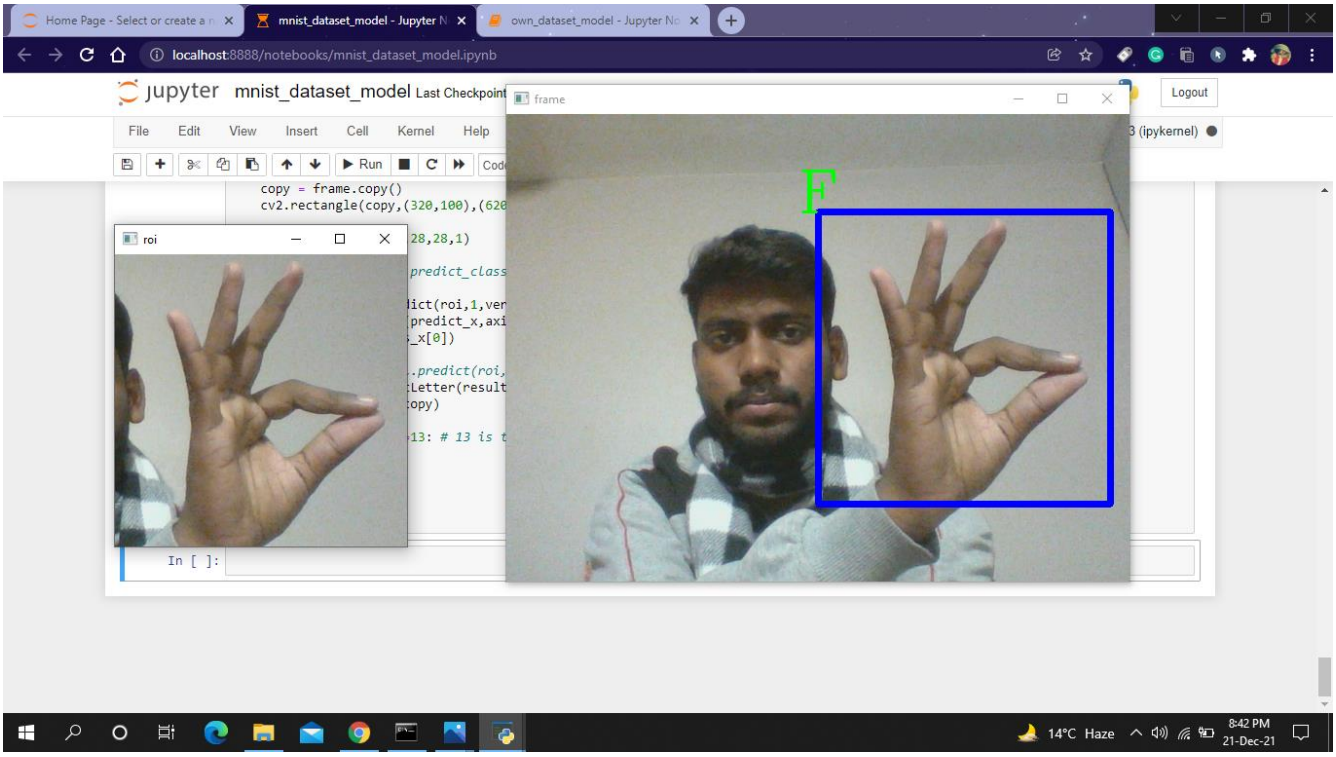
```
#get our accuracy score
from sklearn.metrics import accuracy_score
accuracy_score(test_labels,y_pred.round())

[out] 0.8113496932515337
```

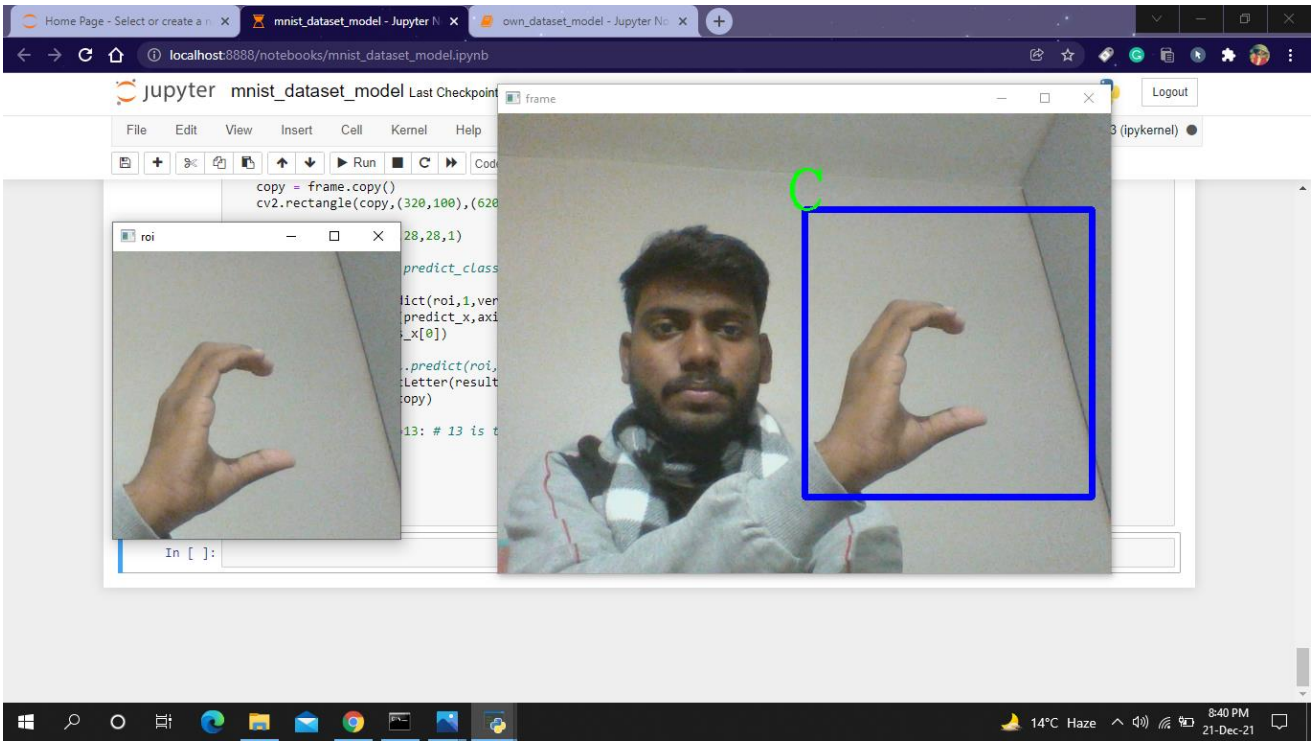
```
# create function to match label to letter
def getLetter(result):
    classLabels = {0: 'A',
                   1: 'B',
                   2: 'C',
                   3: 'D',
                   4: 'E',
                   5: 'F',
                   6: 'G',
                   7: 'H',
                   8: 'I',
                   9: 'K',
                  10: 'L',
                  11: 'M',
                  12: 'N',
                  13: 'O',
                  14: 'P',
                  15: 'Q',
                  16: 'R',
                  17: 'S',
                  18: 'T',
                  19: 'U',
                  20: 'V',
                  21: 'W',
                  22: 'X',
                  23: 'Y'}
    try:
        res = int(result)
        return classLabels[res]
    except:
        return "Error"



# test on actual webcam Input
cap = cv2.VideoCapture(0)


while True:

    ret,frame = cap.read()

    ############################
    frame = cv2.flip(frame,1)

    #define region of interest
    roi = frame[100:400,320:620]
    cv2.imshow('roi',roi)
    roi = cv2.cvtColor(roi,cv2.COLOR_BGR2GRAY)
    roi = cv2.resize(roi,(28,28),interpolation = cv2.INTER_AREA)

    cv2.imshow('roi scaled and gray',roi)
    copy = frame.copy()
    cv2.rectangle(copy,(320,100),(620,400),(255,0,0),5)

    roi = roi.reshape(1,28,28,1)
```

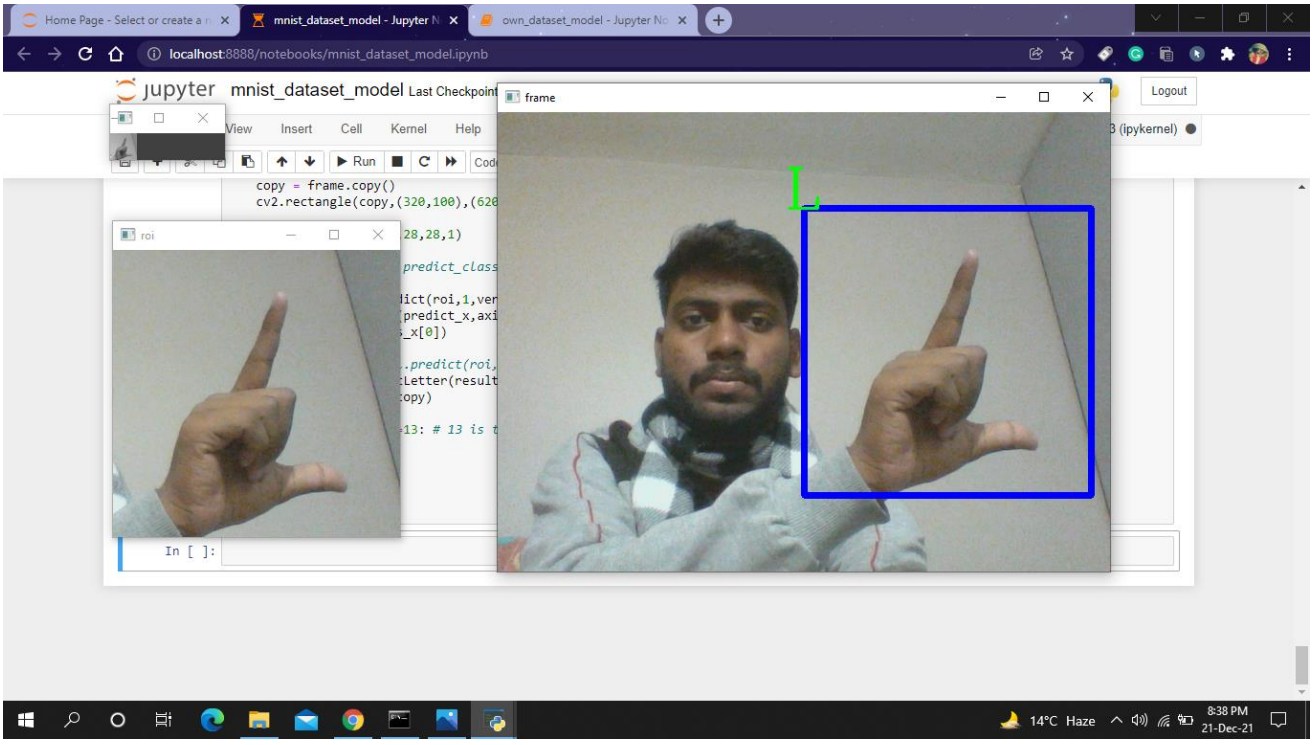
```
    #result = str(model.predict_classes(roi,1,verbose = 0)[0])

    predict_x=model.predict(roi,1,verbose = 0)
    classes_x=np.argmax(predict_x,axis=1)
    result = str(classes_x[0])

    #result = str((model.predict(roi,1,verbose=0) > 0.5).astype("int32")[0])

cv2.putText(copy,getLetter(result),(300,100),cv2.FONT_HERSHEY_COMPLEX,2,(0,255,
0),2)
    cv2.imshow('frame',copy)

    if cv2.waitKey(1) ==13: # 13 is the enter key
        break

cap.release()
cv2.destroyAllWindows()
```

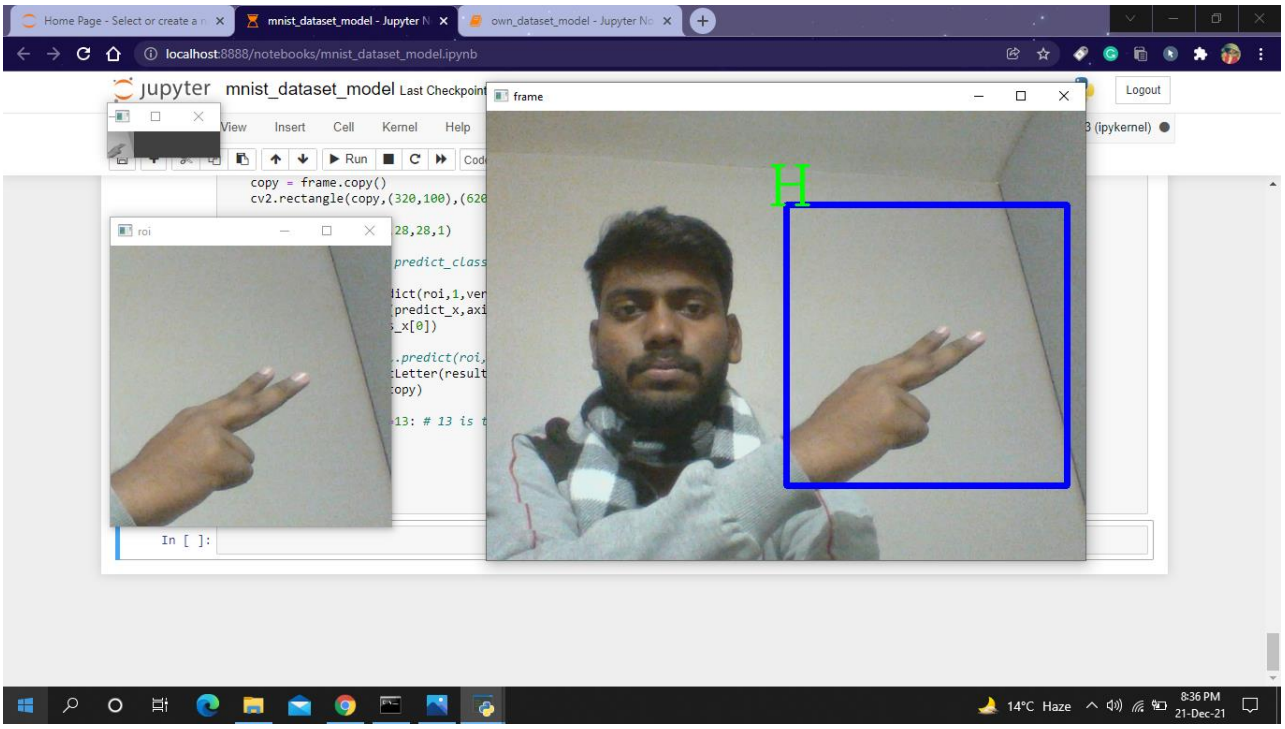
Output and Result

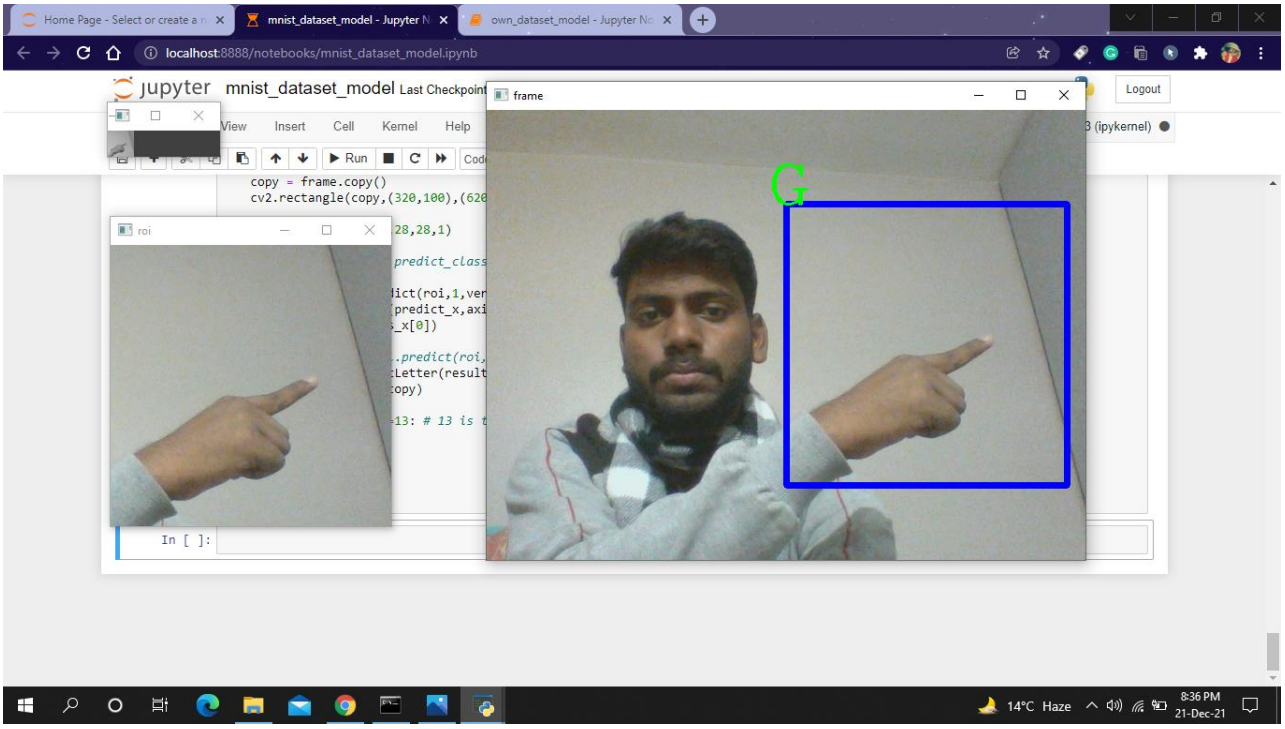
We have successfully implemented our project with the accuracy of around 80% to 85%




Home Page - Select or create a n
mnist_dataset_model - Jupyter N
own_dataset_model - Jupyter No
localhost:8888/notebooks/mnist_dataset_model.ipynb
jupyter mnist_dataset_model Last Checkpoint
Logout
File Edit View Insert Cell Kernel Help
(ipykernel)
Run
copy = frame.copy()
cv2.rectangle(copy,(320,100),(620
roi
frame
In [ ]:
14°C Haze
8:40 PM
21-Dec-21


Home Page - Select or create a n
mnist_dataset_model - Jupyter N
own_dataset_model - Jupyter No
localhost:8888/notebooks/mnist_dataset_model.ipynb
jupyter mnist_dataset_model Last Checkpoint
Logout
View Insert Cell Kernel Help
(ipykernel)
Run
copy = frame.copy()
cv2.rectangle(copy,(320,100),(620
roi
frame
In [ ]:
14°C Haze
8:38 PM
21-Dec-21


Home Page - Select or create a n
mnist_dataset_model - Jupyter N
own_dataset_model - Jupyter No
localhost:8888/notebooks/mnist_dataset_model.ipynb
jupyter
mnist_dataset_model
Last Checkpoint
Logout
View
Insert
Cell
Kernel
Help
Run
(ipykernel)
frame
roi
H
In [ ]:
14°C Haze
8:36 PM
21-Dec-21


Home Page - Select or create a n
mnist_dataset_model - Jupyter N
own_dataset_model - Jupyter No
localhost:8888/notebooks/mnist_dataset_model.ipynb
jupyter
mnist_dataset_model
Last Checkpoint
Logout
View
Insert
Cell
Kernel
Help
Run
(ipykernel)
frame
roi
G
In [ ]:
14°C Haze
8:36 PM
21-Dec-21

Future Works

The extensions of this project are given below:

- Translate the text format into the speech for the blind people
- Convert the text into sign language
- Include the body movements as well as the expression for the conversion

Conclusion

With the growth in globalization and emphasis on educational advancements, we want to make sure that no one should face any kind of hindrance regardless of one's in capabilities. We are aiming to build a web application in order to overcome the disabilities of deaf and dumb people to make sure they don't feel like straggling behind than the rest of the world, as education define one's personality. This project will not only be used in academic fields but also in day-to-day life as well because communication is must to survive. This web application will be reading user's hand gestures and after analysis it will convert that sign in text as well as speech format so they can communicate easily with everyone around them. This gesture conversion that we are building is only limited to English language as of now, but we will be looking forward to expand its linguistic basket to have multiple language conversion in the near future. The problem with the current gesticulation recognition application is that they are incompetent, as a there is not one but many types of sign languages approximately 300 different types of sign languages all around the globe. The earlier systems were developed under the functioning of MATLAB, but it’s not efficient.

References

[1] Brill R. 1986. The Conference of Educational Administrators Serving the Deaf: A History. Washington, DC: Gallaudet University Press.

[2] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[4] L. K. Hansen and P. Salamon, "Neural network ensembles," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, Oct. 1990, doi: 10.1109/34.58871.

[5]Kang, Byeongkeun, SubarnaTripathi, and Truong Q. Nguyen. ”Real- time sign language fingerspelling recognition using convolutional neural networks from depth map.” arXiv preprintarXiv: 1509.03001 (2015).

[6] Suganya, R., and T. Meeradevi. ”Design of a communication aid for phys- ically challenged.” In Electronics and Communication Systems (ICECS), 2015 2nd International Conference on, pp. 818-822. IEEE, 2015.

[7] SruthiUpendran, Thamizharasi. A,” American Sign Language Interpreter System for Deaf and Dumb Individuals”, 2014 International Conference on Control, Instrumentation, Communication and Computa.

[8] David H. Wolpert, Stacked generalization, Neural Networks, Volume 5, Issue 2, 1992, Pages 241-259, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(05)80023-1.

[9] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks, Volume 12, Issue 10,1999, Pages 1399-1404, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(99)00073-8.

[10] MacKay D.J.C. (1995) Developments in Probabilistic Modelling with Neural Networks — Ensemble Learning. In: Kappen B., Gielen S. (eds) Neural Networks: Artificial Intelligence and Industrial Applications.Springer, London. https://doi.org/10.1007/978-1-4471-3087-1_37

[11] Polikar R. (2012) Ensemble Learning. In: Zhang C., Ma Y. (eds) Ensemble Machine Learning.Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-9326-7_1
View publication