REAL-TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED USING IBM WATSON

Submitted by

MALLIPEDDI DEVIKA (19UK1A0538)

KOMAKULA SHIVANI(19UK1A0561)

RAMIDI SRICHARAN(19UK1A0556)

AMBALA RAMESH(19UK1A0550)

1.INTRODUCTION :

1.1. Overview :

In our society, we have people with disabilities. The technology is developing day by day but no significant developments are undertaken for the betterment of these people. Communications between deaf-mute and a normal person has always been a challenging task. It is very difficult for mute people to convey their message to normal people. Since normal people are not trained on hand sign language. In emergency times conveying their message is very difficult. The human hand has remained a popular choice to convey information in situations where other forms like speech cannot be used. Voice Conversion System with Hand Gesture Recognition and translation will be very useful to have a proper conversation between a normal person and an impaired person in any language.

1.2. Purpose:

The project aims to develop a system that converts the sign language into a human hearing voice in the desired language to convey a message to normal people. We are making use of a convolution neural network to create a model that is trained on different hand gestures. A Web Application is built which uses this model. This application enables deaf and dumb people to convey their information using signs which get converted to human- understandable language

2.LITERATURE SURVEY:

2.1.Existing problem:

Some of the existing solutions for solving this problem are:

Technology:

One of the easiest ways to communicate is through technology such as a smart phone or laptop. A deaf person can type out what they want to say and a person who is blind or has low vision can use a screen reader to read the text out loud. A blind person can also use voice recognition software to convert what they are saying in to text so that a person who isDeaf can then read it.

Interpreter:

If a sign language interpreter is available, this facilitates easy communication if the person who is deaf is fluent in sign language. The deaf person and person who is blind can communicate with each other via the interpreter. The deaf person can use sign language and the interpreter can speak what has been said to the person who is blind and then translate anything spoken by the blind person into sign language for the deaf person.

Just Speaking:

Depending on the deaf person's level of hearing loss, they may be able to communicate with a blind person who is using speech. For example, a deaf person may have enough residual hearing (with or without the use of an assistive hearing device such as a hearing

aid) to be able to decipher the speech of the person who is blind or has low vision. However, this is often not the most effective form of communication, as it is very dependent on the individual circumstances of both people and their environment (for example, some places may have too much background noise).

2.2.Proposed solution:

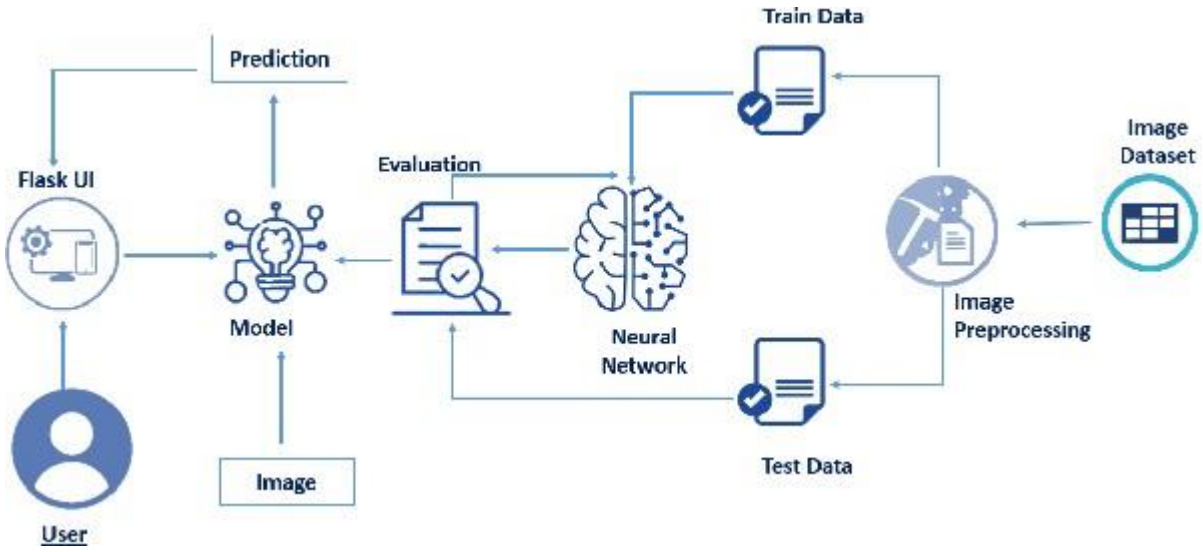
This paper describes the system that overcomes the problem faced by the speech and hearing impaired. The objectives of the research are as follow:

1. To design and develop a system which lowers the communication gap between speech hearing impaired and normal world.
2. To build a communication system that enables communications between deaf-dumb person and a normal person.
3. A convolution neural network is being used to develop a model that is trained on various hand movements. This model is used to create an app. This program allows deaf and hard of hearing persons to communicate using signs that are then translated into human readable text

3. THEORITICAL ANALYSIS:

3.1. Block Diagram:

Architecture:



3.2. Hardware / Software designing :

Hardware Requirements:

Operating System	Windows, Mac, Linux
CPU (for training)	Multi Core Processors (i3 or above/equivalent)
GPU (for training)	NVIDIA AI Capable / Google's TPU
Web Cam	Integrated or External with Full HD Support

Software Requirements:

Python	v3.10.0 or Above
Python Packages	flask, tensorflow, opencv-python, keras, numpy,pandas, virtualenv, pillow
Web Browser	Mozilla Firefox, Google Chrome or any modern web browser
IBM Cloud (for training)	Watson Studio - Model Training & Deployment asMachine Learning Instance

4. EXPERIMENTAL INVESTIGATIONS:

Training the train dataset:

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
# Image Augmentation

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [3]:
```
# Loading train and test set

X_train = train_datagen.flow_from_directory(r"C:\Users\shiva\Downloads\real time communications\real time communications\Dataset\
X_test = test_datagen.flow_from_directory(r"C:\Users\shiva\Downloads\real time communications\real time communications\Dataset\te
```

```
Found 15750 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [4]:
```
# checking indices

X_train.class_indices
```

Out[4]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Model Building

In [5]:
```
# Importing Libraries

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten
```

In [6]:
```
# Initializing the Model

model = Sequential()
```

In [7]:
```
# Adding Convolution Layer

model.add(Convolution2D((32), (3,3), input_shape = (64, 64, 3), activation = 'relu'))
```

In [8]:
```
# Adding Pooling Layer

model.add(MaxPooling2D(pool_size = (2, 2)))
```

In [9]:
```
# Adding Flatten Layer

model.add(Flatten())
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten
```

In [6]:
```
# Initializing the Model

model = Sequential()
```

In [7]:
```
# Adding Convolution Layer

model.add(Convolution2D((32), (3,3), input_shape = (64, 64, 3), activation = 'relu'))
```

In [8]:
```
# Adding Pooling Layer

model.add(MaxPooling2D(pool_size = (2, 2)))
```

In [9]:
```
# Adding Flatten Layer

model.add(Flatten())
```

In [10]:
```
# Adding Hidden Layer

model.add(Dense(units = 512, kernel_initializer = 'random_uniform', activation = 'relu'))
```

In [11]:
```
# Adding Output Layer
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [11]:
```
# Adding Output Layer

model.add(Dense(units = 9, kernel_initializer = 'random_uniform', activation = 'softmax'))
```

In [12]:
```
# Compile the model

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

In [13]:
```
# Fiiting the model

model.fit(X_train, steps_per_epoch = 24, epochs = 10, validation_data = X_test, validation_steps = 40)
```

```
Epoch 1/10
24/24 [==============================] - 6s 219ms/step - loss: 1.4599 - accuracy: 0.5469 - val_loss: 0.5981 - val_accuracy: 0.8
406
Epoch 2/10
24/24 [==============================] - 5s 224ms/step - loss: 0.4523 - accuracy: 0.8555 - val_loss: 0.4918 - val_accuracy: 0.8
984
Epoch 3/10
24/24 [==============================] - 5s 200ms/step - loss: 0.2731 - accuracy: 0.9271 - val_loss: 0.4464 - val_accuracy: 0.8
977
Epoch 4/10
24/24 [==============================] - 5s 200ms/step - loss: 0.2642 - accuracy: 0.9180 - val_loss: 0.4157 - val_accuracy: 0.9
203
Epoch 5/10
24/24 [==============================] - 5s 191ms/step - loss: 0.2434 - accuracy: 0.9206 - val_loss: 0.2446 - val_accuracy: 0.9
```

Out[13]: <keras.callbacks.History at 0x212719ab130>

In [14]:
```
# Saving the model

model.save('aslpng1.h5')
```

Testing the test dataset:

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
# Importing Libarries

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import cv2
```

In [2]:
```
# Loading model

model = load_model('aslpng1.h5')
```

In [3]:
```
from skimage.transform import resize
def detect(frame):
    img = resize(frame, (64, 64, 3))
    img = np.expand_dims(img, axis = 0)
    if np.max(img) > 1:
        img = img/255.0
    prediction = model.predict(img)
    print(prediction)
    return prediction
```

In [19]:
```
frame = cv2.imread(r"C:\Users\shiva\Downloads\real time communications\real time communications\Dataset\training_set\D\16.png")
data = detect(frame)
```

```
1/1 [==============================] - 0s 60ms/step
```

3

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [20]:
```
index = ['A','B','C','D','E','F','G','H','I']
index[np.argmax(data)]
```

Out[20]:
```
'D'
```

OpenCV

In [21]:
```
# Importing Libraries

import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [28]:
```
from tensorflow.keras.preprocessing.image import array_to_img
```

In [23]:
```
# Loading Model

model = load_model("aslpng1.h5")
```

In [24]:
```
video = cv2.VideoCapture(0)
```

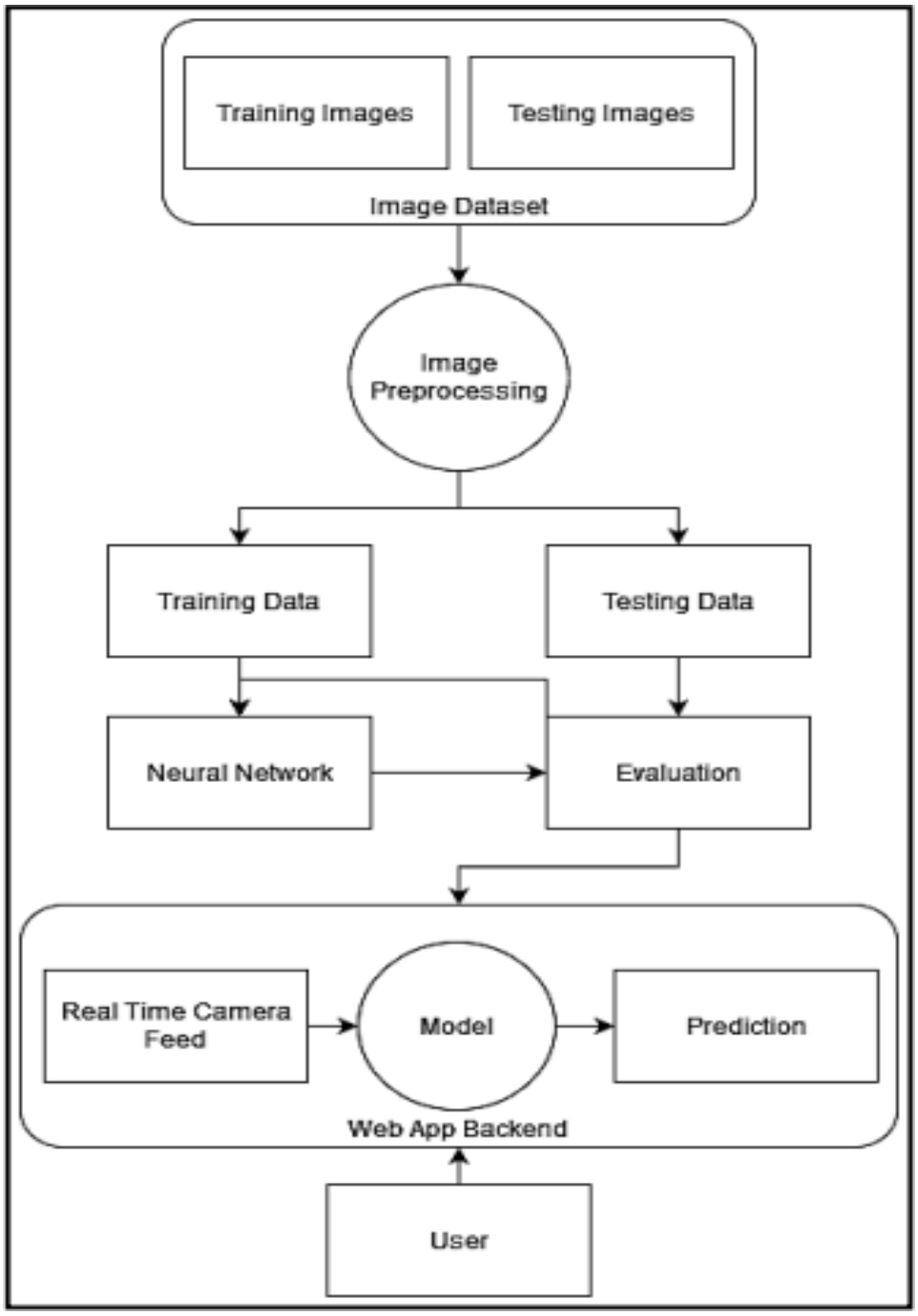
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [31]:
```
while True:
    success, frame = video.read()
    cv2.imwrite('frame.jpg', frame)
    img = image.load_img('frame.jpg', target_size = (64, 64))

    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    pred = np.argmax(model.predict(x), axis = 1)

    y = pred[0]

    copy = frame.copy()


    cv2.rectangle(copy, (320, 100), (620, 400), (255, 0, 0), 5)
    cv2.putText(frame, "The Predicted Alphabet : " + str(index[y]), (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 4)
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
```

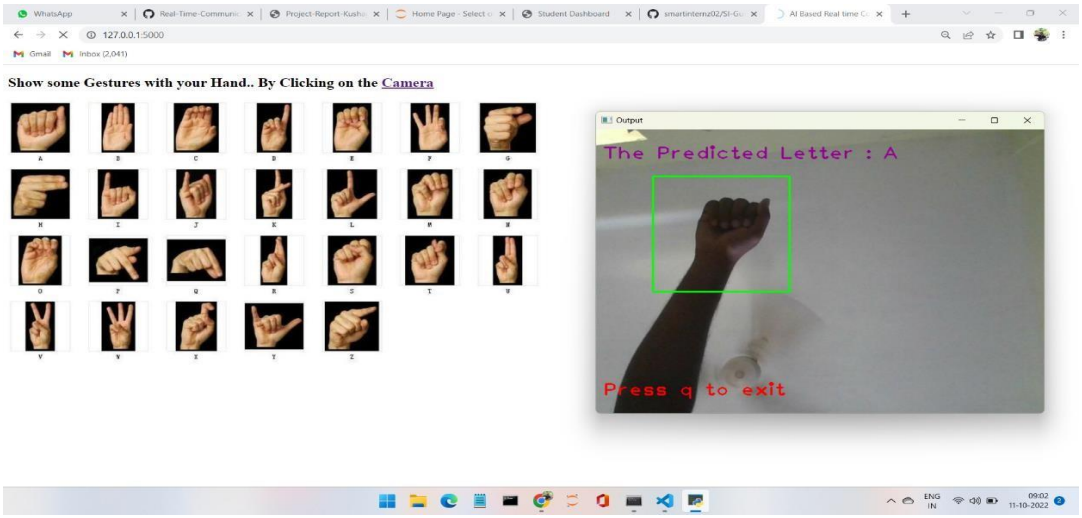
5. FLOWCHART:

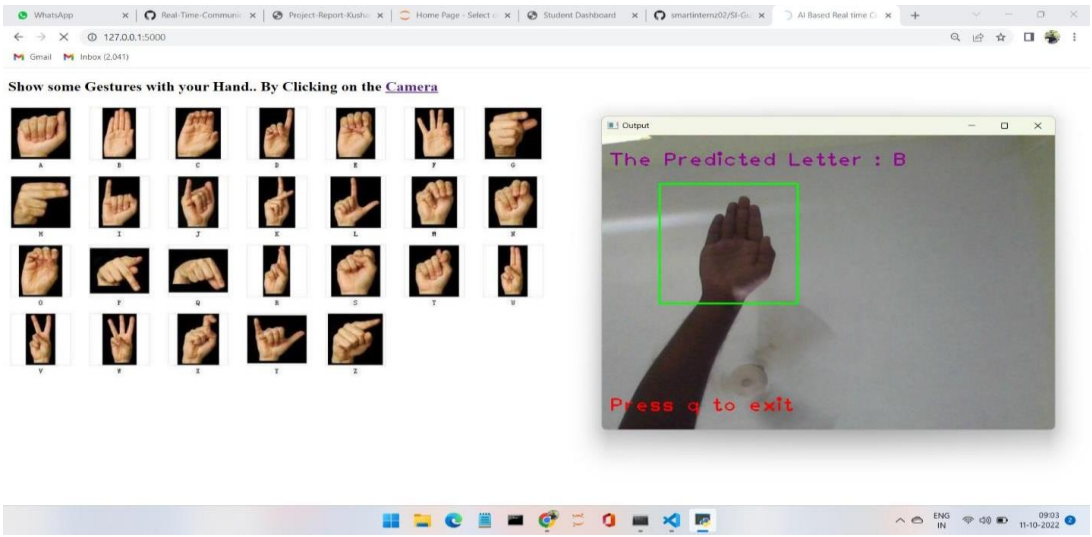

Training Images
Testing Images
Image Dataset
Image Preprocessing
Training Data
Testing Data
Neural Network
Evaluation
Real Time Camera Feed
Model
Prediction
Web App Backend
User

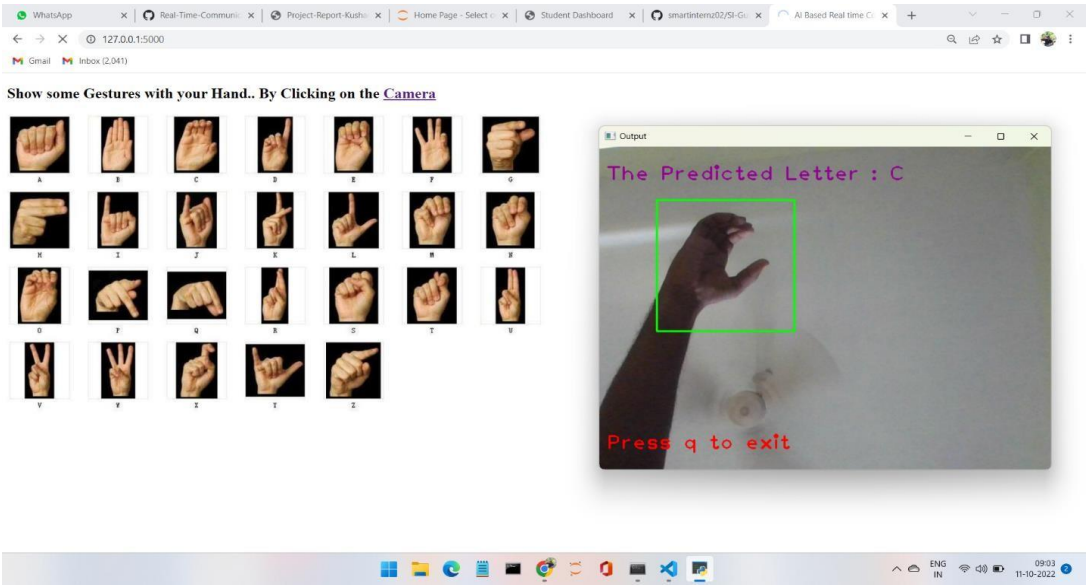
6. RESULT:

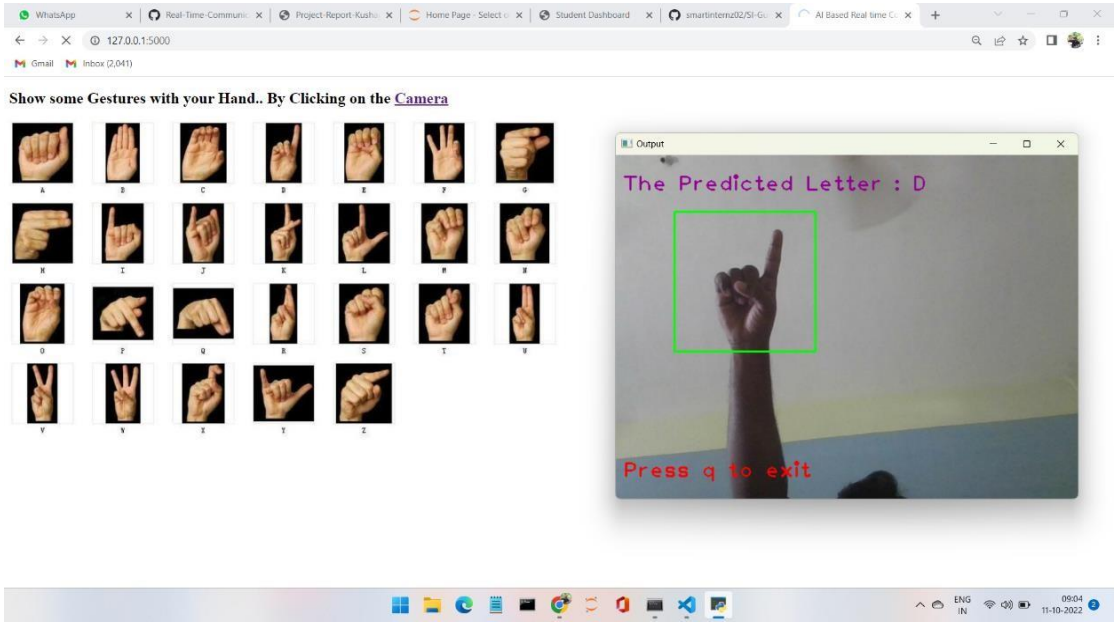
The proposed procedure was implemented and tested with set of images. The set of 15750 images of Alphabets from “A” to “I” are used for training database and a set of 2250 images of Alphabets from “A” to “I” are used for testing database. Once the gesture is recognize the equivalent Alphabet is shown on the screen.

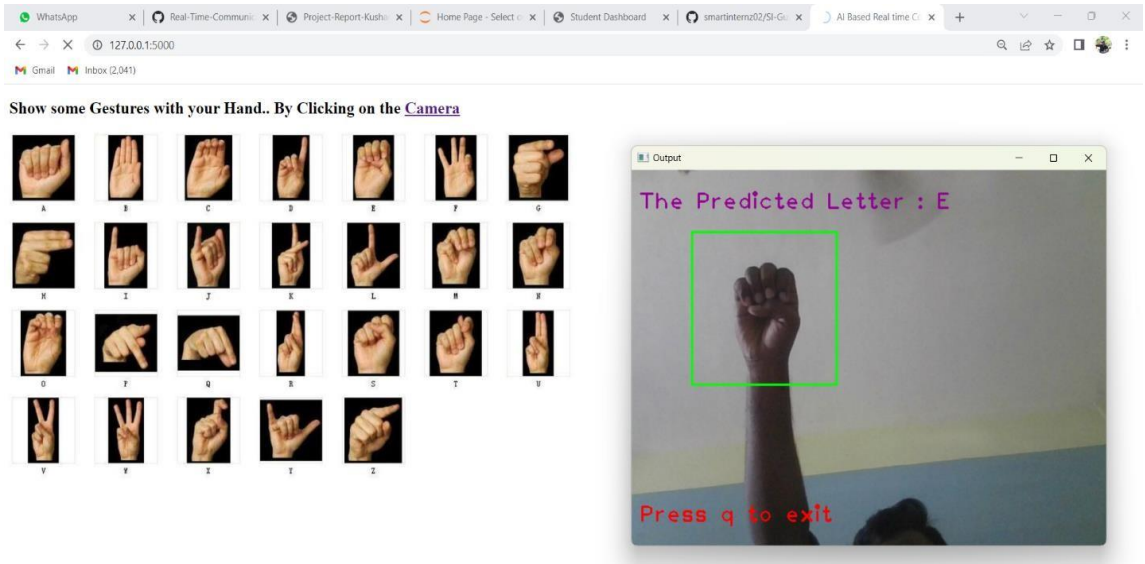
Some sample images of the output are provided below:

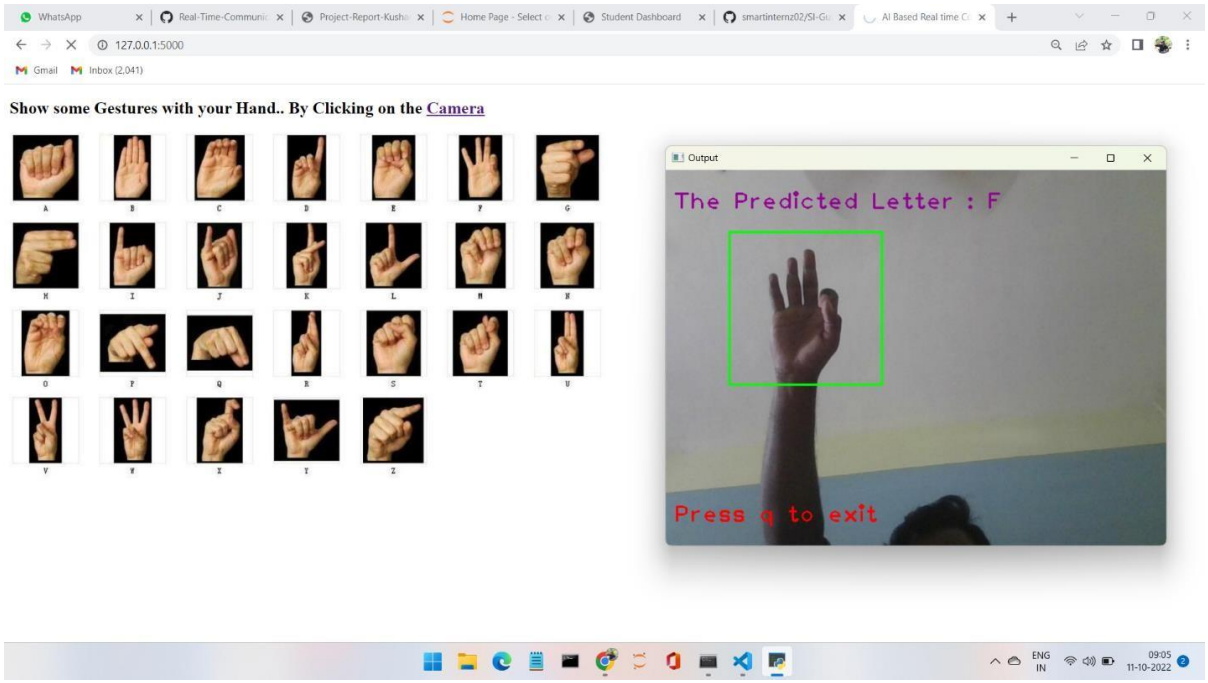


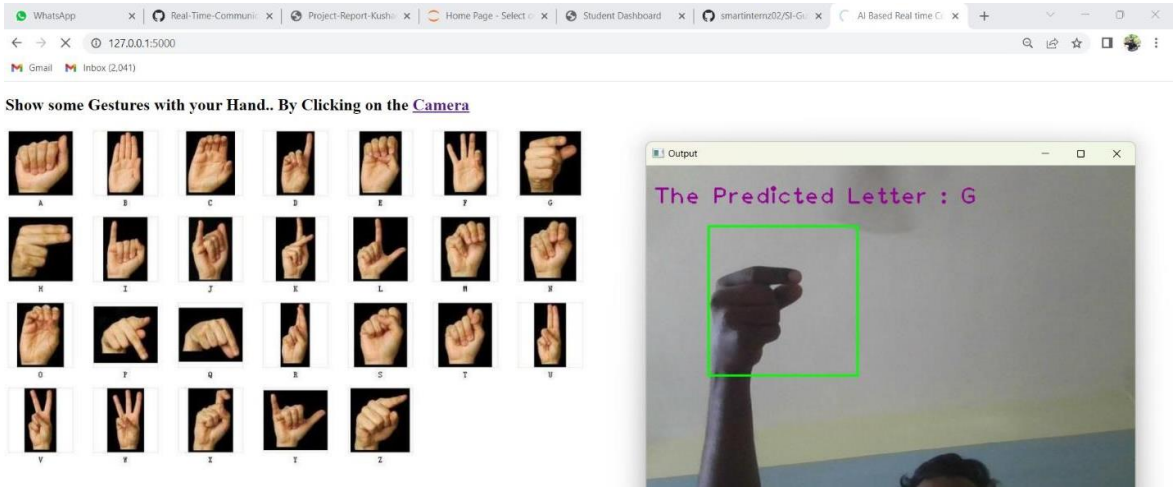


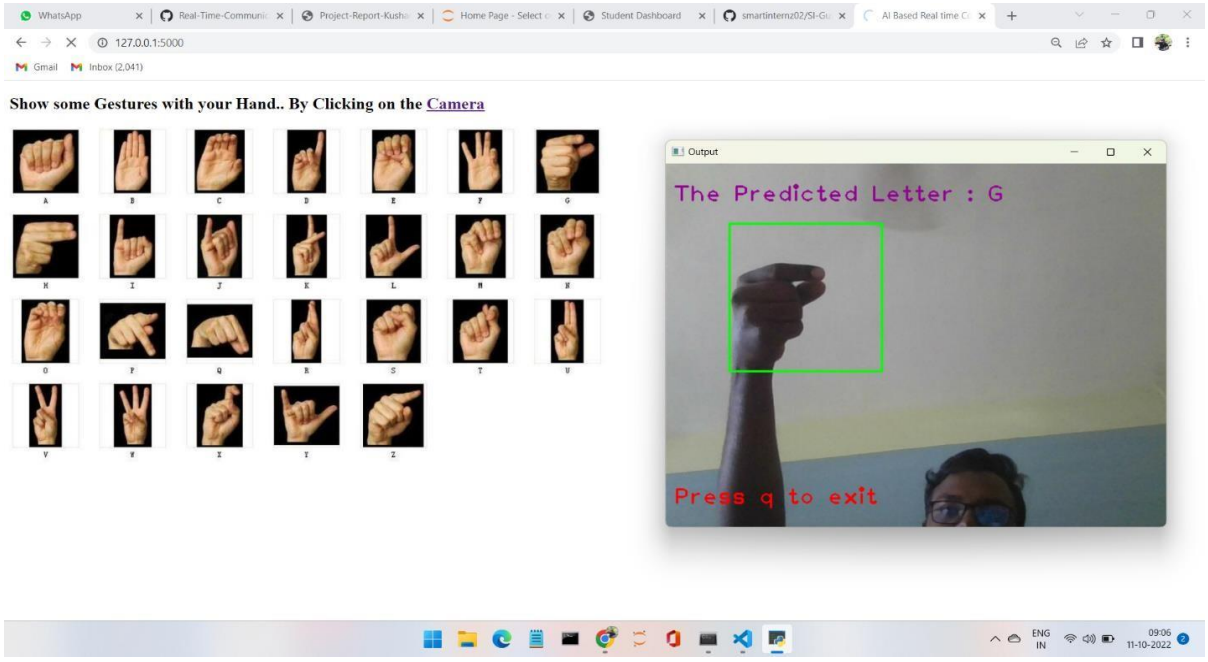

127.0.0.1:5000
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : C
Press q to exit

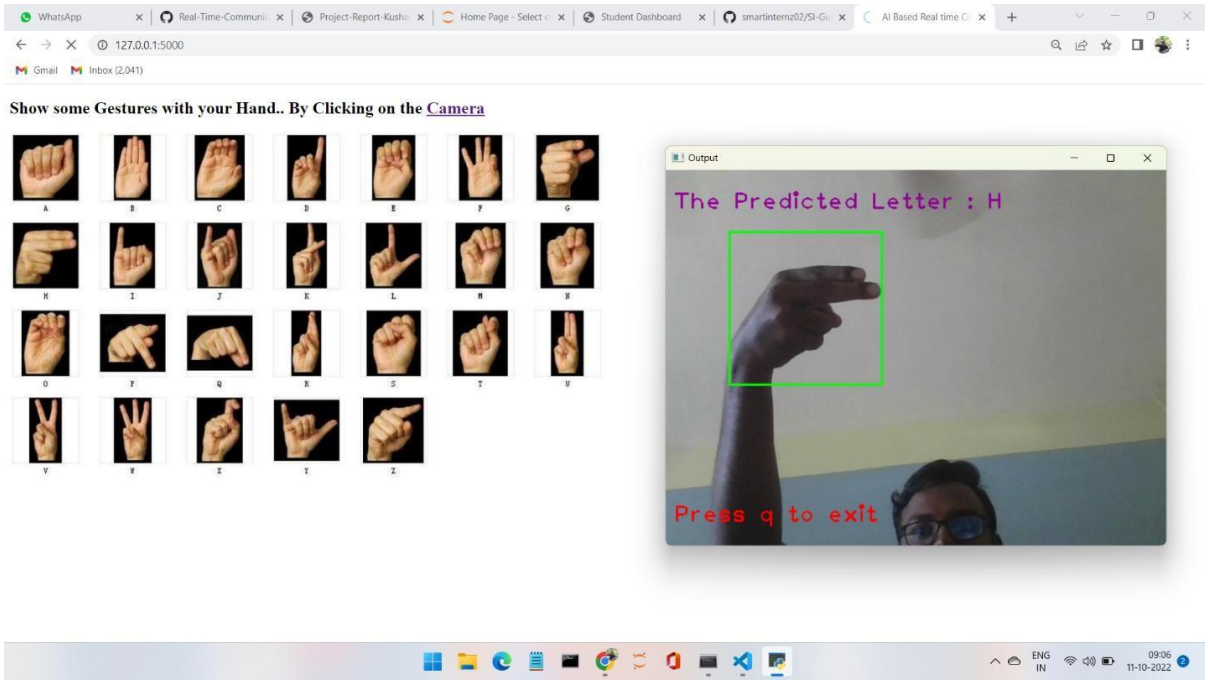

127.0.0.1:5000
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : D
Press q to exit

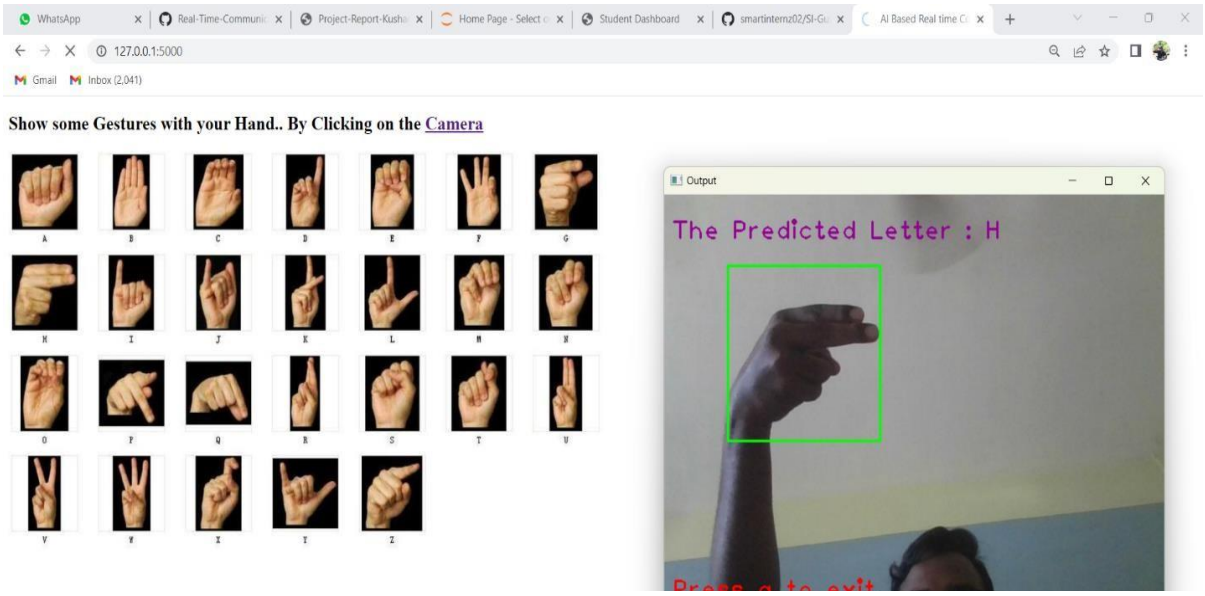

127.0.0.1:5000
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : E
Press q to exit

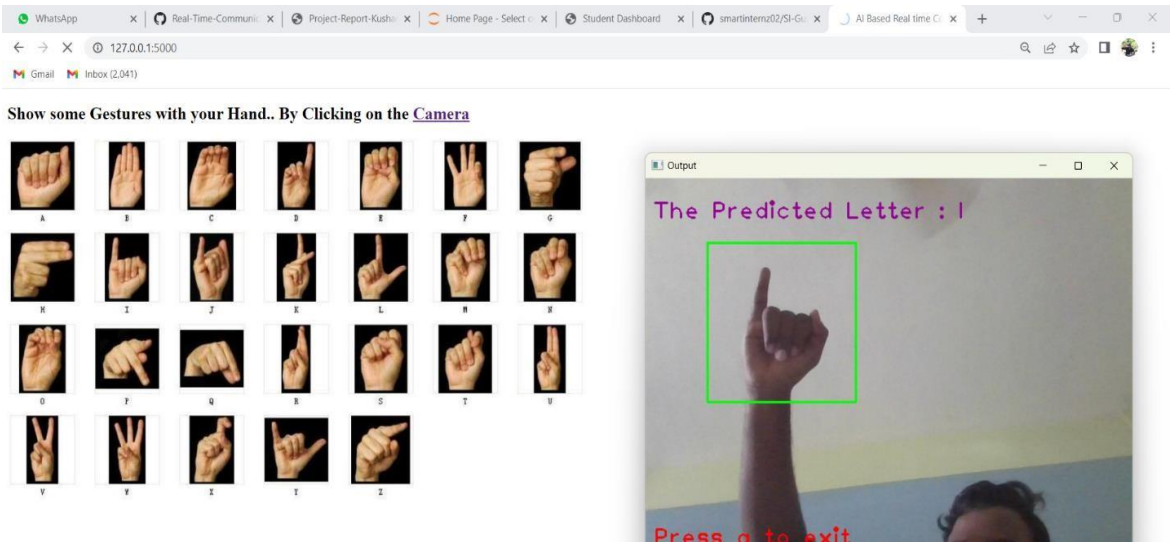

127.0.0.1:5000
Gmail
Inbox (2,041)
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : F
Press q to exit
ENG
IN
09:05
11-10-2022


127.0.0.1:5000
Gmail
Inbox (2,041)
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : G


127.0.0.1:5000
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : G
Press q to exit


127.0.0.1:5000
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : H
Press q to exit


127.0.0.1:5000
Gmail
Inbox (2,041)
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : H


127.0.0.1:5000
Gmail
Inbox (2,041)
Show some Gestures with your Hand.. By Clicking on the Camera
Output
The Predicted Letter : I

7. ADVANTAGES AND DISADVANTAGES:

Advantages:

1. It is possible to create a mobile application to bridge the communication gap between deaf and dumb persons and the general public.
2. As different sign language standards exist, their dataset can be added, and the user can choose which sign language to read.

Disadvantages:

1. The current model only works from alphabets A to I.
2. In absence of gesture recognition, alphabets from J cannot be identified as they require some kind of gesture input from the user.
3. As the quantity/quality of images in the dataset is low, the accuracy is not great, but that can easily be improved by change in dataset.

8.APPLICATIONS:

1. It will contribute to the development of improved communication for the deafened. The majority of people are unable to communicate via sign language, which creates a barrierto communication.
2. As a result, others will be able to learn and comprehend sign language and communicate with the deaf and dumb via the web app.
3. According to scientific research, learning sign language improves cognitive abilities,attention span, and creativity.

9.CONCLUSION:

Sign language is a useful tool for facilitating communication between deaf and hearing peop le. Because it allows for two-way communication, the system aims to bridge the communication gap between deaf people and rest of society. The proposed methodology translates language into English alphabets that are understandable to humans.

This system sends hand gestures to the model, who recognises them and displays the equivalent Alphabet on the screen. Deaf-mute people can use their hands to perform sign language, which will then be converted into alphabets.

10.FUTURE SCOPE

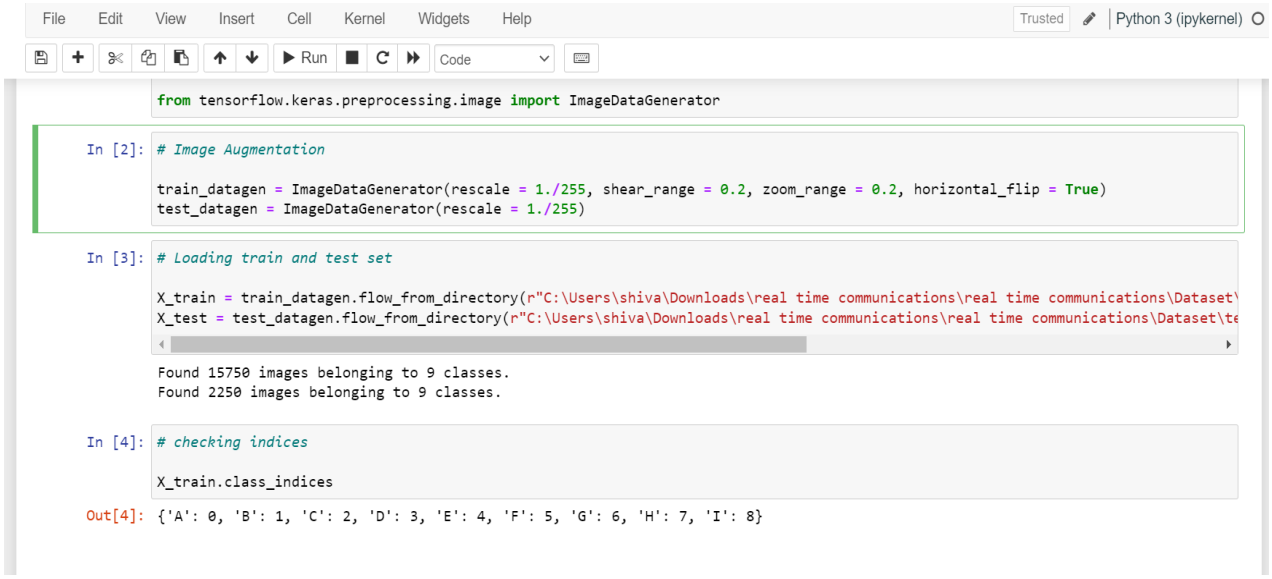
Having a technology that can translate hand sign language to its corresponding alphabet is a game changer in the field of communication and AI for the specially abled people such as deaf and dumb. With introduction of gesture recognition, the web app can easily be expanded to recognize letters beyond ‘I’, digits and other symbols plus gesture recognition can also allow controlling of software/hardware interfaces.

11.BIBILOGRAPHY

1. Environment Setup: https://www.youtube.com/watch?v=5mDYijMfSzs
2. Sign Languages Dataset: https://drive.google.com/file/d/1CSTYNw3pbvPozlFxhNOuDyRCgm6A5vid/view?usp=sh aring
3. Keras Image Processing Doc: https://keras.io/api/preprocessing/image/
4. Keras Image Dataset From Directory Doc: https://keras.io/api/preprocessing/image/#imagedatasetfromdirectory-function
5. CNN using Tensorflow: https://www.youtube.com/watch?v=umGJ30-15_A
6. OpenCV Basics of Processing Image: https://www.youtube.com/watch?v=mjKd1Tzl70I
7. Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0
8. IBM Academic Partner Account Creation: https://www.youtube.com/watch?v=x6i43M7BAqE
9. CNN Deployment and Download through IBM Cloud: https://www.youtube.com/watch?v=BzouqMGJ41k

APPENDIX:

Training and testing the dataset:



```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

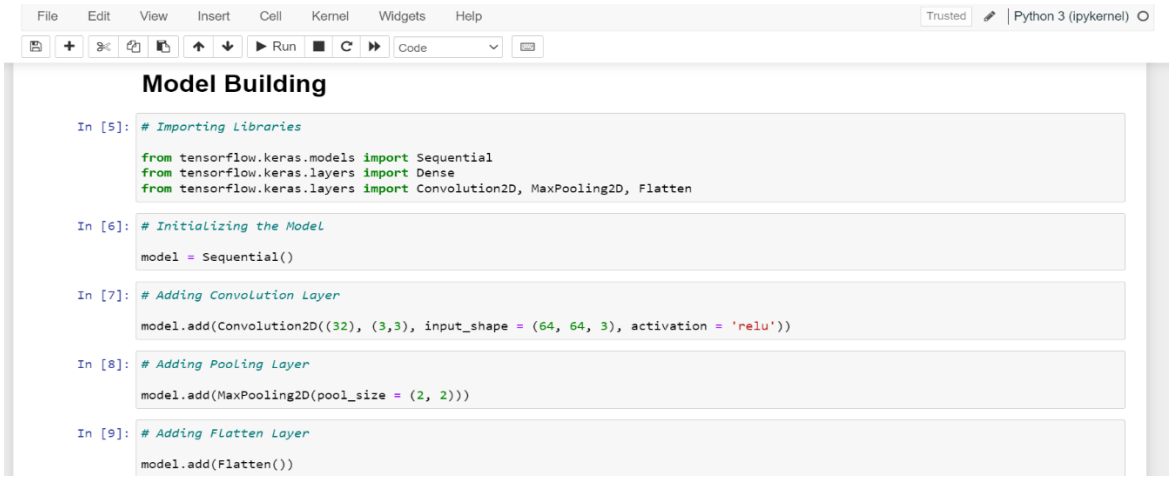
In [2]:
```
# Image Augmentation

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [3]:
```
# Loading train and test set

X_train = train_datagen.flow_from_directory(r"C:\Users\shiva\Downloads\real time communications\real time communications\Dataset\
X_test = test_datagen.flow_from_directory(r"C:\Users\shiva\Downloads\real time communications\real time communications\Dataset\te
```

```
Found 15750 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [4]:
```
# checking indices

X_train.class_indices
```

Out[4]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```



Model Building

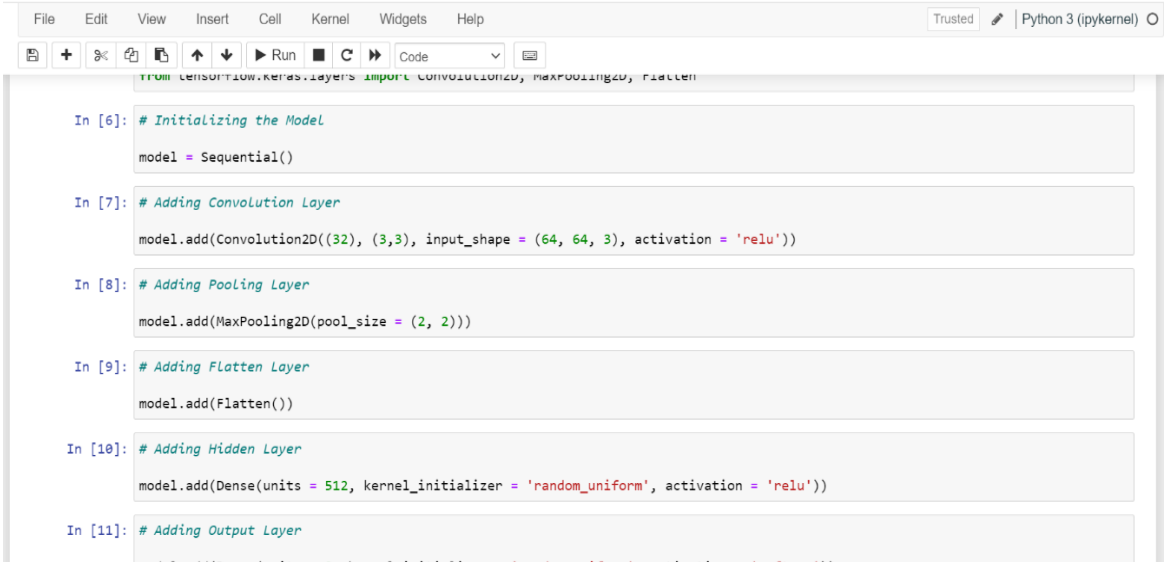
In [5]:
```
# Importing Libraries

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten
```

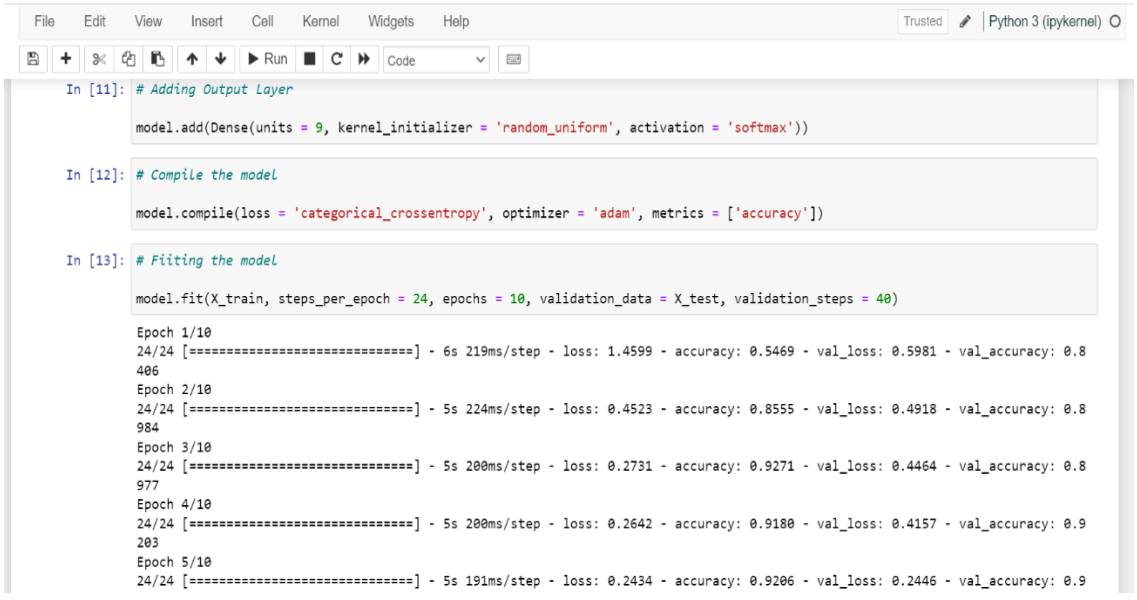
In [6]:
```
# Initializing the Model

model = Sequential()
```

In [7]:
```
# Adding Convolution Layer

model.add(Convolution2D((32), (3,3), input_shape = (64, 64, 3), activation = 'relu'))
```

In [8]:
```
# Adding Pooling Layer

model.add(MaxPooling2D(pool_size = (2, 2)))
```

In [9]:
```
# Adding Flatten Layer

model.add(Flatten())
```


File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [6]:
# Initializing the Model
model = Sequential()
In [7]:
# Adding Convolution Layer
model.add(Convolution2D((32), (3,3), input_shape = (64, 64, 3), activation = 'relu'))
In [8]:
# Adding Pooling Layer
model.add(MaxPooling2D(pool_size = (2, 2)))
In [9]:
# Adding Flatten Layer
model.add(Flatten())
In [10]:
# Adding Hidden Layer
model.add(Dense(units = 512, kernel_initializer = 'random_uniform', activation = 'relu'))
In [11]:
Adding Output Layer


File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [11]:
# Adding Output Layer
model.add(Dense(units = 9, kernel_initializer = 'random_uniform', activation = 'softmax'))
In [12]:
# Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
In [13]:
# Fiiting the model
model.fit(X_train, steps_per_epoch = 24, epochs = 10, validation_data = X_test, validation_steps = 40)
Epoch 1/10
24/24 [==============================] - 6s 219ms/step - loss: 1.4599 - accuracy: 0.5469 - val_loss: 0.5981 - val_accuracy: 0.8
406
Epoch 2/10
24/24 [==============================] - 5s 224ms/step - loss: 0.4523 - accuracy: 0.8555 - val_loss: 0.4918 - val_accuracy: 0.8
984
Epoch 3/10
24/24 [==============================] - 5s 200ms/step - loss: 0.2731 - accuracy: 0.9271 - val_loss: 0.4464 - val_accuracy: 0.8
977
Epoch 4/10
24/24 [==============================] - 5s 200ms/step - loss: 0.2642 - accuracy: 0.9180 - val_loss: 0.4157 - val_accuracy: 0.9
203
Epoch 5/10
24/24 [==============================] - 5s 191ms/step - loss: 0.2434 - accuracy: 0.9206 - val_loss: 0.2446 - val_accuracy: 0.9

Out[13]: <keras.callbacks.History at 0x212719ab130>

In [14]:
```
# Saving the model

model.save('aslpng1.h5')
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
# Importing Libarries

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import cv2
```

In [2]:
```
# Loading model

model = load_model('aslpng1.h5')
```

In [3]:
```
from skimage.transform import resize
def detect(frame):
    img = resize(frame, (64, 64, 3))
    img = np.expand_dims(img, axis = 0)
    if np.max(img) > 1:
        img = img/255.0
    prediction = model.predict(img)
    print(prediction)
    return prediction
```

In [19]:
```
frame = cv2.imread(r"C:\Users\shiva\Downloads\real time communications\real time communications\Dataset\training_set\D\16.png")
data = detect(frame)
```

```
1/1 [==============================] - 0s 60ms/step
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [20]:
```
index = ['A','B','C','D','E','F','G','H','I']
index[np.argmax(data)]
```

Out[20]: 'D'

OpenCV

In [21]:
```
# Importing Libraries

import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [28]:
```
from tensorflow.keras.preprocessing.image import array_to_img
```

In [23]:
```
# Loading Model

model = load_model("aslpng1.h5")
```

In [24]:
```
video = cv2.VideoCapture(0)
```

jupyter Real Time Communication System - Model Testing Last Checkpoint: 28 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [31]:
```
while True:
    success, frame = video.read()
    cv2.imwrite('frame.jpg', frame)
    img = image.load_img('frame.jpg', target_size = (64, 64))

    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    pred = np.argmax(model.predict(x), axis = 1)

    y = pred[0]

    copy = frame.copy()


    cv2.rectangle(copy, (320, 100), (620, 400), (255, 0, 0), 5)
    cv2.putText(frame, "The Predicted Alphabet : " + str(index[y]), (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 4)
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
```

Training Model in IBM Watson:

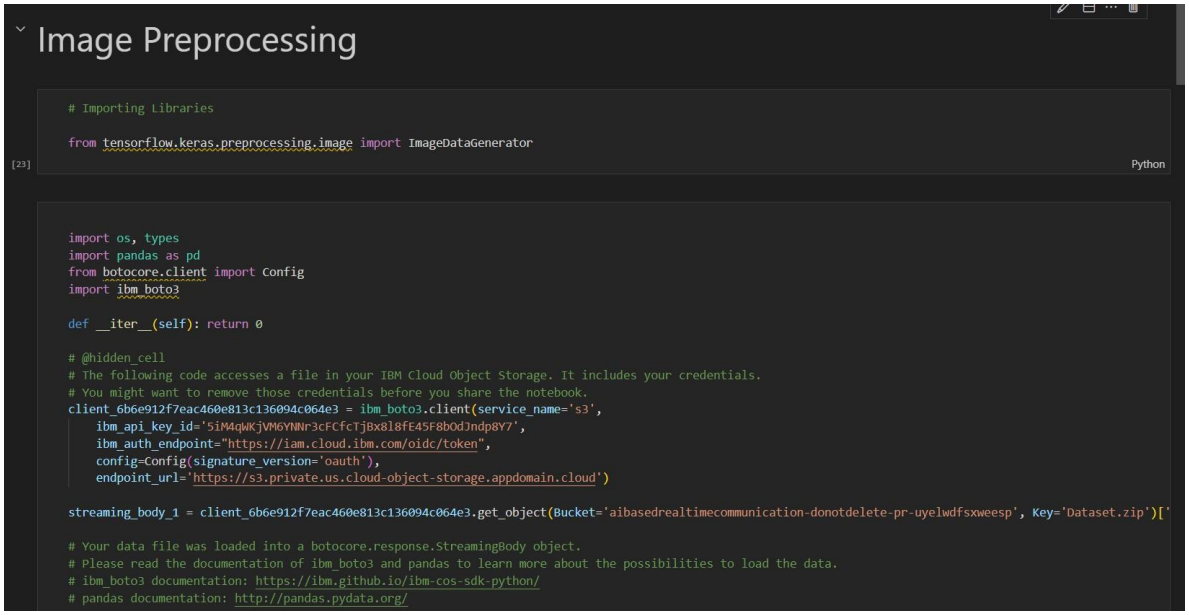
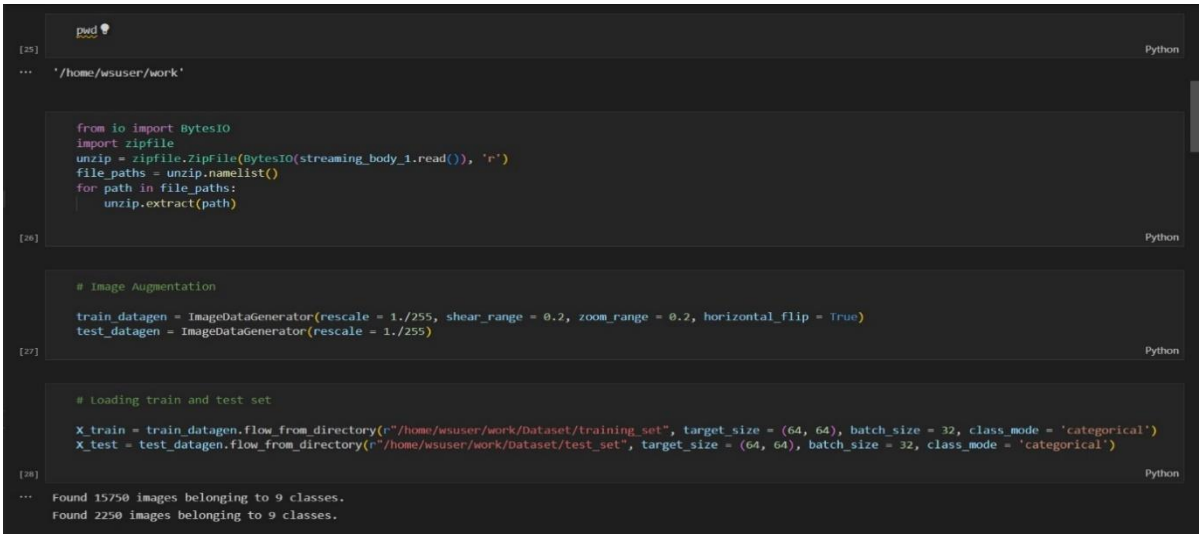


Image Preprocessing

```
# Importing Libraries

from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[23] Python

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_6b6e912f7eac460e813c136094c064e3 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='5iM4qWKjVM6YNNr3cFCfcTjBx8l8fE45F8bOdJndp8Y7',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

streaming_body_1 = client_6b6e912f7eac460e813c136094c064e3.get_object(Bucket='aibasedrealtimecommunication-donotdelete-pr-uyelwdfsxweesp', Key='Dataset.zip')['

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```



```
pwd
```

[25] Python

```
'/home/wsuser/work'
```

```
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

[26] Python

```
# Image Augmentation

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

[27] Python

```
# Loading train and test set

X_train = train_datagen.flow_from_directory(r"/home/wsuser/work/Dataset/training_set", target_size = (64, 64), batch_size = 32, class_mode = 'categorical')
X_test = test_datagen.flow_from_directory(r"/home/wsuser/work/Dataset/test_set", target_size = (64, 64), batch_size = 32, class_mode = 'categorical')
```

[28] Python

```
Found 15750 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

Model Building

```
# Importing Libraries

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten
```
[5] Python

```
# Initializing the Model

model = Sequential()
```
[6] Python

```
# Adding Convolution Layer

model.add(Convolution2D((32), (3,3), input_shape = (64, 64, 3), activation = 'relu'))
```
[7] Python

```
# Adding Pooling Layer

model.add(MaxPooling2D(pool_size = (2, 2)))
```
[33] Python

```
# Adding Flatten Layer

model.add(Flatten())
```
[34] Python

```
# Adding Hidden Layer

model.add(Dense(units = 512, kernel_initializer = 'random_uniform', activation = 'relu'))
```
[35] Python

```
# Adding Output Layer

model.add(Dense(units = 9, kernel_initializer = 'random_uniform', activation = 'softmax'))
```
[36] Python

```
# Compile the model

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```
[37] Python

```
# Fiiting the model

model.fit_generator(X_train, steps_per_epoch = 24, epochs = 10, validation_data = X_test, validation_steps = 40)
```
[38] Python

```
/tmp/wsuser/ipykernel_236/1270027362.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  model.fit_generator(X_train, steps_per_epoch = 24, epochs = 10, validation_data = X_test, validation_steps = 40)

Epoch 1/10
24/24 [==============================] - 7s 282ms/step - loss: 1.2710 - accuracy: 0.5703 - val_loss: 0.6611 - val_accuracy: 0.7641
Epoch 2/10
24/24 [==============================] - 6s 251ms/step - loss: 0.4401 - accuracy: 0.8438 - val_loss: 0.4736 - val_accuracy: 0.8648
Epoch 3/10
24/24 [==============================] - 6s 272ms/step - loss: 0.2849 - accuracy: 0.9219 - val_loss: 0.3455 - val_accuracy: 0.9195
Epoch 4/10
24/24 [==============================] - 7s 275ms/step - loss: 0.1931 - accuracy: 0.9414 - val_loss: 0.3100 - val_accuracy: 0.9164
Epoch 5/10
24/24 [==============================] - 7s 274ms/step - loss: 0.1410 - accuracy: 0.9570 - val_loss: 0.2939 - val_accuracy: 0.9281
Epoch 6/10
24/24 [==============================] - 7s 277ms/step - loss: 0.1432 - accuracy: 0.9622 - val_loss: 0.2978 - val_accuracy: 0.9438
Epoch 7/10
24/24 [==============================] - 6s 264ms/step - loss: 0.1128 - accuracy: 0.9648 - val_loss: 0.2513 - val_accuracy: 0.9414
Epoch 8/10
24/24 [==============================] - 6s 260ms/step - loss: 0.0925 - accuracy: 0.9674 - val_loss: 0.3209 - val_accuracy: 0.9461
Epoch 9/10
24/24 [==============================] - 7s 273ms/step - loss: 0.1017 - accuracy: 0.9766 - val_loss: 0.3081 - val_accuracy: 0.9555
Epoch 10/10
24/24 [==============================] - 6s 253ms/step - loss: 0.0656 - accuracy: 0.9779 - val_loss: 0.2222 - val_accuracy: 0.9711

<keras.callbacks.History at 0x7f0b498b0f70>
```

```
# Saving the model

model.save('aslpng1.h5')
```
[39] Python

```
!tar -zcvf ai-based-real-time-classification-model.tgz aslpng1.h5
```
[52] Python

```
aslpng1.h5
```

```
!pip install watson-machine-learning-client
```
[41] Python

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 23.2 MB/s eta 0:00:01
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.6.15)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client)
(1.21.41)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client)
```

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url" : "https://us-south.ml.cloud.ibm.com",
    "apikey" : "T7rpH1KfTn-s-zfDCmTyeArxYrcvFGHFV21qTDW5pf5x"
}
client = APIClient(wml_credentials)
```
[42] Python

```
def guid_space_name(client, ai_based_real_time_communication_deploy_space):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name'] == ai_based_real_time_communication_deploy_space)['metadata']['id'])
```
[43] Python

```
client.spaces.get_details()
```
[45] Python

```
Output exceeds the size limit. Open the full output data in a text editor
{'resources': [{'entity': {'compute': [{'crn': 'crn:v1:bluemix:public:pm-20:us-south:a/5be23fa7fba94c8aa2e3db0b2a4db8d2:04f159f4-9ffb-4e0d-b70b-2a1f3b216970::',
      'guid': '04f159f4-9ffb-4e0d-b70b-2a1f3b216970',
      'name': 'Watson Machine Learning-av',
      'type': 'machine_learning'}],
    'description': '',
    'name': 'ai_based_real_time_communication_deploy_space',
    'scope': {'bss_account_id': '5be23fa7fba94c8aa2e3db0b2a4db8d2'},
    'stage': {'production': False},
    'status': {'state': 'active'},
    'storage': {'properties': {'bucket_name': '0a42d73b-35af-4f9d-92da-4a84147fcb1c',
      'bucket_region': 'us-south',
```

Downloading model from IBM Cloud:

```
pip install ibm_watson_machine_learning
```

[1] Python

```
Requirement already satisfied: ibm_watson_machine_learning in c:\users\mahes\anaconda3\lib\site-packages (1.0.253)
Requirement already satisfied: urllib3 in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: requests in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2.26.0)
Requirement already satisfied: certifi in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2021.10.8)
Requirement already satisfied: ibm-cos-sdk==2.11.* in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: tabulate in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (0.9.0)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (1.3.4)
Requirement already satisfied: packaging in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (21.0)
Requirement already satisfied: importlib-metadata in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (4.8.1)
Requirement already satisfied: lomond in c:\users\mahes\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in c:\users\mahes\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning)
(0.10.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in c:\users\mahes\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning)
(2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in c:\users\mahes\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*-
>ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\mahes\anaconda3\lib\site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*-
>ibm_watson_machine_learning) (2.8.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\mahes\anaconda3\lib\site-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_machine_learning) (1.20.3)
Requirement already satisfied: pytz>=2017.3 in c:\users\mahes\anaconda3\lib\site-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_machine_learning) (2021.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mahes\anaconda3\lib\site-packages (from requests->ibm_watson_machine_learning) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\mahes\anaconda3\lib\site-packages (from requests->ibm_watson_machine_learning) (2.0.4)
Requirement already satisfied: zipp>=0.5 in c:\users\mahes\anaconda3\lib\site-packages (from importlib-metadata->ibm_watson_machine_learning) (3.6.0)
Requirement already satisfied: six>=1.10.0 in c:\users\mahes\anaconda3\lib\site-packages (from lomond->ibm_watson_machine_learning) (1.16.0)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\mahes\anaconda3\lib\site-packages (from packaging->ibm_watson_machine_learning) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url" : "https://us-south.ml.cloud.ibm.com",
    "apikey" : "T7rpH1KfTn-s-zfDCmIyeArxYrcvFGHFV21qTDW5pf5x"
}
client = APIClient(wml_credentials)
```

[2] Python

```
def guid_space_name(client, ai_based_real_time_communication_deploy_space):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name'] == ai_based_real_time_communication_deploy_space)['metadata']['id'])
```

[3] Python

```
space_id = guid_space_name(client, 'ai_based_real_time_communication_deploy_space')
space_id
```

[4] Python

```
'1853d74e-ca3c-4075-81e3-d5cdd0741a52'
```

```
client.set.default_space(space_id)
```

[5] Python

```
'SUCCESS'
```

```
client.repository.download('59b18265-3a03-47d3-b2d8-d9a0c5106f05', 'ai-based-real-time-classification-model.h5')
```

[6] Python

```
Successfully saved model content to file: 'ai-based-real-time-classification-model.h5'

'D:\\Maheshfiles\\Studies\\Smart Bridge\\AI-ML-DL Project/ai-based-real-time-classification-model.h5'
```

Web Application in Flask:

webstreaming.py

```python
from flask import Flask,render_template,request
import cv2
from keras.models import load_model
import numpy as np
from gtts import gTTS
import os
from keras.preprocessing import image
from skimage.transform import resize
from playsound import playsound
app = Flask(__name__)

model=load_model("aslpng1.h5")

vals = ['A', 'B','C','D','E','F','G','H','I']

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')
@app.route('/index', methods=['GET'])
def home():
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def predict():
        print("[INFO] starting video stream...")
        vs = cv2.VideoCapture(0)

        (W, H) = (None, None)

        while True:
            (grabbed, frame) = vs.read()

            if not grabbed:
                break

            if W is None or H is None:
                (H, W) = frame.shape[:2]
            output = frame.copy()
            # r = cv2.selectROI("Slect", output)
            # print(r)
            cv2.rectangle(output, (81, 79), (276,274), (0,255,0), 2)
            frame = frame[81:276, 79:274]
            frame = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
            _, frame = cv2.threshold(frame, 95, 255, cv2.THRESH_BINARY_INV)
            frame = cv2.cvtColor(frame, cv2.COLOR_GRAY2RGB)


            img = resize(frame,(64,64,3))
            img = np.expand_dims(img,axis=0)
            if(np.max(img)>1):
                img = img/255.0


            result = np.argmax(model.predict(img))
            index=['A', 'B','C','D','E','F','G','H','I']
            result=str(index[result])


            cv2.putText(output, "The Predicted Letter : {}".format(result), (10, 50), cv2.FONT_HERSHEY_PLAIN,
                        2, (150,0,150), 2)
            cv2.putText(output, "Press q to exit", (10,450), cv2.FONT_HERSHEY_PLAIN, 2, (0,0,255), 2)


            speech = gTTS(text = result, lang = 'en', slow = False)

            cv2.imshow("Output", output)
            key = cv2.waitKey(1) & 0xFF

            if key == ord("q"):
                break
```

```
        print("[INFO] cleaning up...")
        vs.release()
        cv2.destroyAllWindows()
        return render_template("index.html")


if __name__ == '__main__':
    app.run(debug=False)
```

index.html ● webstreaming.py ●

templates > index.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>AI Based Real time Communication</title>
</head>
<body>
    <h2>Show some Gestures with your Hand.. By Clicking on the <a href="{{ url_for('predict') }}">Camera</a></h2>
    <div>
        <img src = 'https://www.researchgate.net/publication/274480405/figure/fig4/
        AS:668304138063878@1536347531535/26-sample-hand-gesture-images-for-26-letters-in-American-Sign-Language.jpg'  width = 50%>

    </div>
</body>
</html>
```

Hand Gestures in American Sign Language: