1. INTRODUCTION

1.1 Overview

By sharing their ideas, thoughts, and experiences with others around them, people come to know one another. There are several methods to do this, but the gift of "Speech" is the finest. Everyone can communicate their ideas clearly and comprehend to one another through speech. If we ignore those who don't have this wonderful gift, the deaf and dumb, it will be unjustified and stupid. The human hand has continued to be the favoured form of communication in these circumstances.

1.2 Purpose

The goal of the project is to develop a system that can translate sign language into a language that is understandable to regular people.

2.Literature Survey

2.1Existing Solution

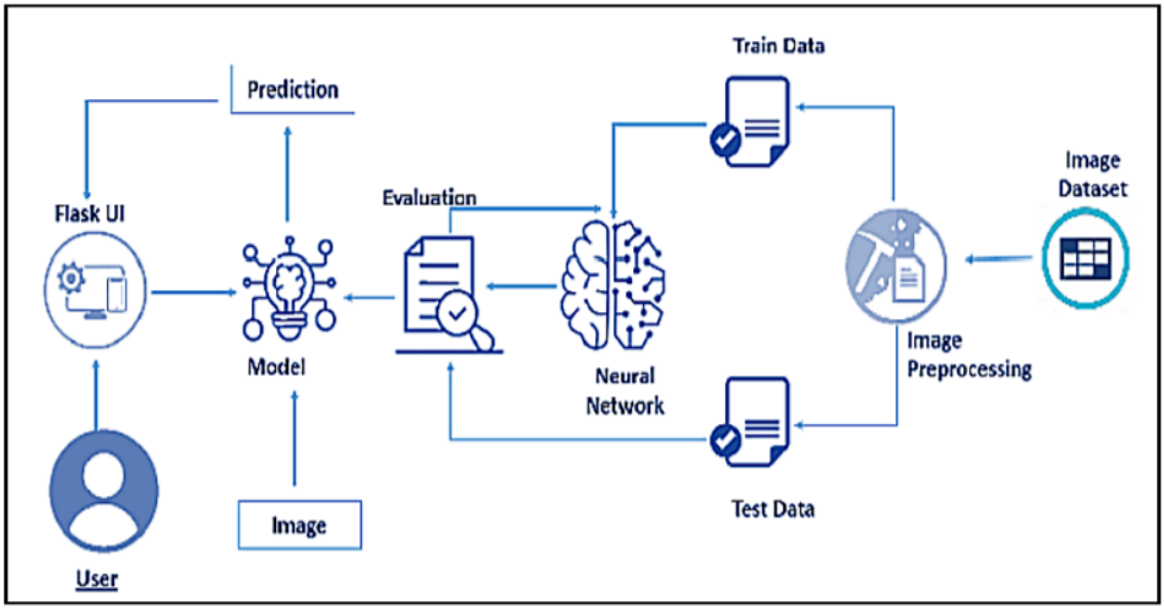
- Using a laptop or smart phone for communication is among the simplest methods. A screen reader can be used by someone who is blind or has low eyesight to hear the text being read aloud while a deaf person can type out what they wish to say. Whereas a blind person, can make use of voice to text option so that the deaf person can read the text through screen.
- Depending on how severely deaf they are, a blind person making use of speech may be able to talk to a deaf person. For instance, a deaf person might have sufficient residual hearing to understand the words of a person who is blind or has low eyesight.However, it greatly depends on the specific circumstances.

2.2 Proposed Solution

A model is trained on diverse hand gestures and is created using a convolution neural network. Using this an app can be created. Through the use of signals that are later converted into text that can be read by humans, this programme enables deaf and dumb people to communicate.

3.Theoritical Analysis

3.1 Block diagram



3.2 Hardware/Software Specifications

- Hardware

  Operating System : Windows, Mac, Linux
  CPU : Multi Core Processors(i3 or above)
  GPU : NVIDIA AI Capable / Google's TPU WebCam Integrated or External with FullHD Support

- Software

  Python : v3.9.0 or above
  Python Packages : tensorflow, flask,keras, numpy, pandas, virtualenv, pillow,opencv-python
  Web Browser :Mozilla Firefox,Internet Explorer, Google Chrome
  IBM Cloud : Watson Studio - Model Training & Deployment as Machine Learning Instance

4.Experimental Investigation

Training and Testing code using given dataset

In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [4]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'C:\Users\HP\Downloads\asl alphabets ai\ProjectFiles\Dataset\training_set',target_
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'C:\Users\HP\Downloads\asl alphabets ai\ProjectFiles\Dataset\test_set',target_size=(
```

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

In [5]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [6]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [15]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [16]:
```
# Creating Model
model=Sequential()
```

In [17]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

# Adding Hidden Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

# Adding Output Layer
model.add(Dense(9,activation='softmax'))
```

In [19]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [22]:
```
# Fitting the Model Generator
model.fit(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
18/18 [==============================] - 74s 4s/step - loss: 0.1971 - accuracy: 0.9455 - val_loss: 0.1936 - val_accuracy: 0.
9493
Epoch 2/10
18/18 [==============================] - 74s 4s/step - loss: 0.0937 - accuracy: 0.9750 - val_loss: 0.1569 - val_accuracy: 0.
9631
Epoch 3/10
18/18 [==============================] - 74s 4s/step - loss: 0.0482 - accuracy: 0.9888 - val_loss: 0.1560 - val_accuracy: 0.
9684
Epoch 4/10
18/18 [==============================] - 74s 4s/step - loss: 0.0308 - accuracy: 0.9928 - val_loss: 0.1673 - val_accuracy: 0.
9764
Epoch 5/10
18/18 [==============================] - 74s 4s/step - loss: 0.0220 - accuracy: 0.9952 - val_loss: 0.1677 - val_accuracy: 0.
9698
Epoch 6/10
18/18 [==============================] - 74s 4s/step - loss: 0.0147 - accuracy: 0.9970 - val_loss: 0.1638 - val_accuracy: 0.
9769
Epoch 7/10
18/18 [==============================] - 74s 4s/step - loss: 0.0081 - accuracy: 0.9987 - val_loss: 0.1804 - val_accuracy: 0.
9769
Epoch 8/10
18/18 [==============================] - 74s 4s/step - loss: 0.0080 - accuracy: 0.9984 - val_loss: 0.1696 - val_accuracy: 0.
9769
Epoch 9/10
18/18 [==============================] - 74s 4s/step - loss: 0.0064 - accuracy: 0.9990 - val_loss: 0.2145 - val_accuracy: 0.
9769
Epoch 10/10
18/18 [==============================] - 75s 4s/step - loss: 0.0050 - accuracy: 0.9992 - val_loss: 0.2100 - val_accuracy: 0.
9769
```

Out[22]:
```
<keras.callbacks.History at 0x230bd328550>
```

Saving the Model

In [12]:
```
model.save('speciallyabled.h5')
```

Testing the model

In [1]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [5]:
```
model=load_model('speciallyabled.h5')
img=image.load_img(r'C:\Users\HP\Downloads\asl alphabets ai\ProjectFiles\Dataset\test_set\D\3.png',target_size=(64,64))
```

In [6]:
```
img
```

Out[6]:

In [7]:
```
x=image.img_to_array(img)
```

In [8]:
```
x.ndim
```

Out[8]:
```
3
```

In [9]:
```
x=np.expand_dims(x,axis=0)
```

In [10]:
```
x.ndim
```

Out[10]: 4

In [11]:
```
pred=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 237ms/step
```

In [12]:
```
pred
```

Out[12]: array([1], dtype=int64)

In [13]:
```
index=['A','B','C','D','E','F','G','H','I']
print(index[pred[0]])
```

```
B
```

In [14]:
```
import cv2
```

In [21]:
```
img=cv2.imread(r'C:\Users\HP\Downloads\asl alphabets ai\ProjectFiles\Dataset\test_set\C\2.png',1)
```

In [22]:
```
img1=cv2.imread(r'C:\Users\HP\Downloads\asl alphabets ai\ProjectFiles\Dataset\test_set\B\2.png',0)
```

In [23]:
```
print(img.shape)
```

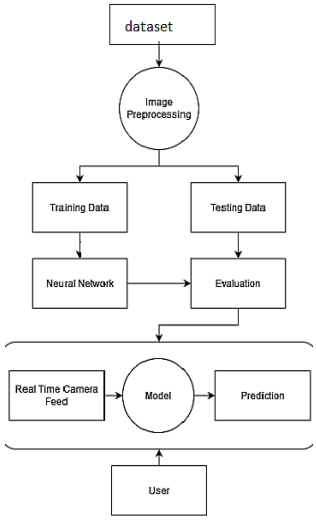
In [24]:
```
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

CNN Video Analysis

In [2]:
```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model('speciallyabled.h5')
video=cv2.VideoCapture(0)
index=['A','B','C','D','E','F','G','H','I']
while 1:
    succes,frame=video.read()
    cv2.imwrite('image.jpg',frame)
    img=image.load_img('image.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 is: '+str(index[y]),(100,100),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),4)
    cv2.imshow('image',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video.release()
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 8ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 16ms/step
```

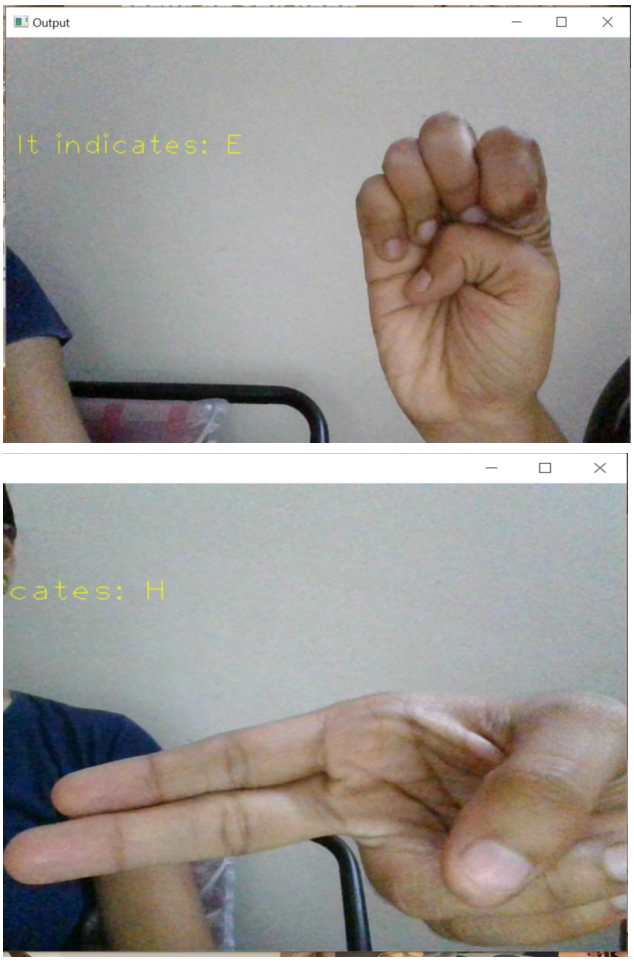
5.Flowchart



6.Result

A series of photos were used to implement and test the suggested approach. A set of 2250 photos of the alphabet from A-I are utilised for the testing database, while a set of 15750 images are used for the training database as soon as the gesture is recognised, on the screen, an equivalent alphabet is displayed.

Below are some examples of the output images:



7.Advantages and Disadvantages

Advantages:

1.An mobile application could be developed to help the hearing-impaired and the general public communicate with one another.

2.The device does not require other external devices and users only need to use their hands within the camera collection range.Also Low cost, Convenient and easy to use.

Disadvantages:

1.The present model is able to predict only the letters A - I.

2. The quality of the dataset used is not efficient, has a great impact on the accuracy.

8.Applications

It will have major role in the improvement of communciation with specially abled,

because most people are unable to converse via sign language.

It brings people together.

9.Conclusion

The use of sign language can help deaf and dumb to communicate more effectively. The technology strives to close the communication gap between the deaf community and the rest of society because it supports two-way conversation. The suggested method converts language intoEnglish alphabets that people can read and write.This system transmits the model hand movements, which it recognises and responds to by displaying the corresponding alphabet. People who are deaf-mute can utilise sign language with their hands, which be turned into alphabets.

10.Future Scope

For persons with particular needs, such as the deaf and dumb, having technology that can translate hand sign language to its appropriate alphabet is a key differentiator. The web programme may easily be developed to detect letters other than I, numbers, and other symbols with the addition of gesture recognition. Gesture recognition can also be used to control software and hardware interfaces.

11.Bibilography

1. Sign Languages Dataset: https://drive.google.com/file/d/1ITbDvhLwyTTkuUYfNjOKhcIZh7hDgi64/view?usp=sharing
2. CNN using Tensorflow: https://www.youtube.com/watch?v=umGJ30-15_A
3. Flask: https://www.youtube.com/watch?v=lj4I_CvBnt0
4. IBM Cloud Account Creation: https://www.youtube.com/watch?v=x6i43M7BAqE
5. CNN Deployment and Download through IBM Cloud: https://www.youtube.com/watch?v=BzouqMGJ41k

12.Appendix

Training and Testing the Model

In [5]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [7]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [8]:
```
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_ccf9b69a4e1443d593920fc48b9c198c = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='iDiQbFo6KMCA2-HUUkkyO9v6rAahmCSA3Aofj-zLbnvP',
    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_2 = client_ccf9b69a4e1443d593920fc48b9c198c.get_object(Bucket='communication-donotdelete-pr-rdym70nqcaeoxu', Key
='Dataset.zip')['Body']
```

In [9]:
```
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_2.read()), 'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [10]:
```
%%bash
ls Dataset
```

```
test_set
training_set
```

In [41]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/home/wsuser/work/Dataset/training_set',target_size=(64,64), class_mode='categorica
l',batch_size=5)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/home/wsuser/work/Dataset/test_set',target_size=(64,64), class_mode='categorical',batc
h_size=5)
```

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

In [30]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  124
```

Model Creation

In [32]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [66]:
```
# Creating Model
model=Sequential()
```

In [67]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

# Adding Hidden Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

# Adding Output Layer
model.add(Dense(9,activation='softmax'))
```

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

In [69]:
```
# Fitting the Model Generator
model.fit(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
18/18 [==============================] - 49s 3s/step - loss: 1.4490 - accuracy: 0.5696 - val_loss: 0.5208 - val_accuracy: 0.888
4
Epoch 2/10
18/18 [==============================] - 48s 3s/step - loss: 0.3672 - accuracy: 0.9010 - val_loss: 0.2726 - val_accuracy: 0.921
3
Epoch 3/10
18/18 [==============================] - 48s 3s/step - loss: 0.1552 - accuracy: 0.9566 - val_loss: 0.2208 - val_accuracy: 0.946
2
Epoch 4/10
18/18 [==============================] - 48s 3s/step - loss: 0.0861 - accuracy: 0.9770 - val_loss: 0.1768 - val_accuracy: 0.964
0
Epoch 5/10
18/18 [==============================] - 48s 3s/step - loss: 0.0514 - accuracy: 0.9872 - val_loss: 0.1686 - val_accuracy: 0.968
4
Epoch 6/10
18/18 [==============================] - 48s 3s/step - loss: 0.0349 - accuracy: 0.9912 - val_loss: 0.1731 - val_accuracy: 0.968
0
Epoch 7/10
18/18 [==============================] - 48s 3s/step - loss: 0.0248 - accuracy: 0.9942 - val_loss: 0.1679 - val_accuracy: 0.978
2
Epoch 8/10
```

In [33]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "TZeHfigplibCmMu2Xot8ol_mSfX0qB8reYwZ5UYCGg1d"
}

client = APIClient(wml_credentials)
```

Save to Deployment Space

In [35]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return (next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [38]:
```
space_uid = guid_from_space_name(client, 'communicate')
print("Space UID : ", space_uid)
```

```
Space UID :  1ce87514-ade5-4c48-902a-50069a8b10e9
```

```
In [39]: client.set.default_space(space_uid)

Out[39]: 'SUCCESS'

In [40]: client.software_specifications.list()

------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                      0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base

In [41]: software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
         software_spec_uid

Out[41]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'

In [44]: model_details = client.repository.store_model(model='IBM_TrainedModel.tgz', meta_props={
             client.repository.ModelMetaNames.NAME: "CNN",
             client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
             client.repository.ModelMetaNames.TYPE: "tensorflow_2.7"})
         model_id = client.repository.get_model_uid(model_details)

         This method is deprecated, please use get_model_id()

         /opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_machine_learning/repository.py:1452: UserWarning: This method is depr
         ecated, please use get_model_id()
           warn("This method is deprecated, please use get_model_id()")

In [45]: model_id

Out[45]: '052384b0-2806-4f31-8781-27585d4f3a0c'
```

Downloading trained model from IBM Cloud

```
In [3]: from ibm_watson_machine_learning import APIClient
        wml_credentials = {
        "url": "https://us-south.ml.cloud.ibm.com",
        "apikey": "TZeHfigplibCmMu2Xot8ol_mSfX0qB8reYwZ5UYCGg1d"
        }

        client = APIClient(wml_credentials)

In [4]: def guid_from_space_name(client, space_name):
            space = client.spaces.get_details()
            return (next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])

In [5]: space_uid = guid_from_space_name(client, 'communicate')
        print("Space UID : ", space_uid)

        Space UID :  1ce87514-ade5-4c48-902a-50069a8b10e9

In [6]: client.set.default_space(space_uid)

Out[6]: 'SUCCESS'

In [9]: client.repository.download("052384b0-2806-4f31-8781-27585d4f3a0c","IBM_Model_train_Download.tar.gz")

        Successfully saved model content to file: 'IBM_Model_train_Download.tar.gz'

Out[9]: 'C:\\Users\\HP\\Downloads\\asl alphabets ai\\ProjectFiles\\IBM_Deployment_Files/IBM_Model_train_Download.tar.gz'
```