Real-Time Communication System Powered By AI For Specially Abled

Submitted by

RAJASURYA B
(1905111 - SREC)

ROSHAN S
(1905115 - SREC)

VARATHA SRI RAJAN S A
(1905162-SREC)

VETRI K
(1905163- SREC)

1 INTRODUCTION

1.1 Overview

People get to know one another by sharing their ideas, thoughts, and experiences with those around them. There are numerous ways to accomplish this, the best of which is the gift of "Speech." Everyone can very convincingly transfer their thoughts and understand each other through speech. It will be unjust if we overlook those who are denied this priceless gift: the deaf and dumb. In such cases, the human hand has remained the preferred method of communication.

1.2 Purpose

The project's purpose is to create a system that translates sign language into a human-understandable language so that ordinary people may understand it.

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 is Deaf 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.

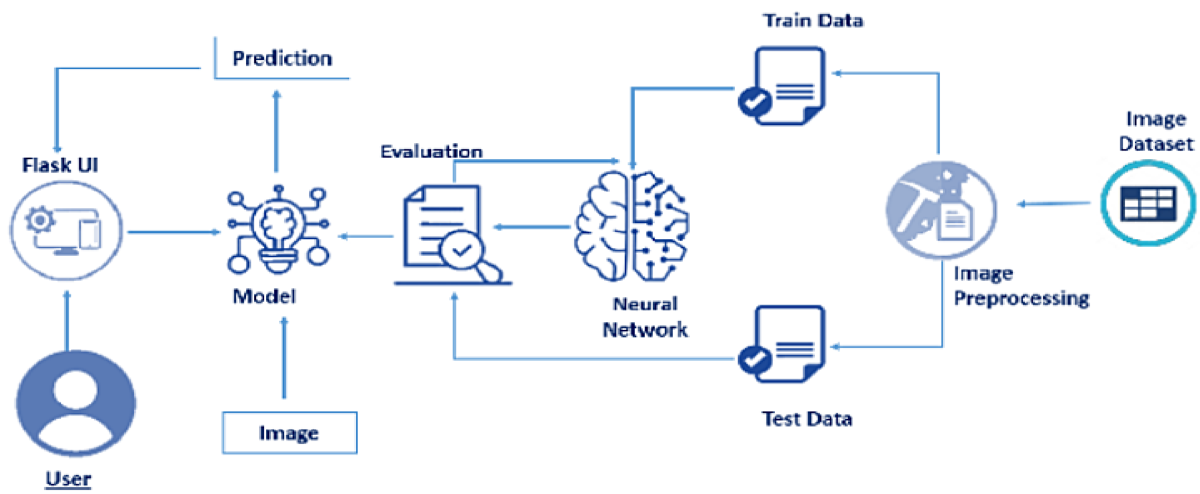
2) To build a communication system that enables communications between deaf-dumb person and a normal person.

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 programme 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
WebCam	Integrated or External with FullHD Support

Software Requirements:

Python	v3.9.0 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 as Machine Learning Instance

4 EXPERIMENTAL INVESTIGATIONS

File Edit View Run Kernel Tabs Settings Help

Externship Project.ipynb

```
[3]: test_datagen = ImageDataGenerator(rescale=1/255)

[4]: x_train=train_datagen.flow_from_directory(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\training_set',target_size=(64,64),
                                              class_mode='categorical',batch_size=900)

     Found 15750 images belonging to 9 classes.

[5]: len(x_train)

[5]: 18

[6]: x_test=test_datagen.flow_from_directory(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set',target_size=(64,64),
                                            class_mode='categorical',batch_size=900)

     Found 2250 images belonging to 9 classes.

[7]: len(x_test)

[7]: 3

[8]: x_train.class_indices

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

Importing Libraries

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

Creating Model

```
[10]: model=Sequential()
```

Adding Layers

Fit the model

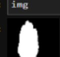
```
[17]: model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))

C:\Users\LEGION\AppData\Local\Temp/ipykernel_24492/234118701.py:1: 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=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
Epoch 1/10
18/18 [==============================] - 50s 3s/step - loss: 1.3134 - accuracy: 0.5780 - val_loss: 0.4696 - val_accuracy: 0.8787
Epoch 2/10
18/18 [==============================] - 49s 3s/step - loss: 0.3442 - accuracy: 0.9060 - val_loss: 0.3401 - val_accuracy: 0.9191
Epoch 3/10
18/18 [==============================] - 50s 3s/step - loss: 0.1466 - accuracy: 0.9592 - val_loss: 0.2585 - val_accuracy: 0.9418
Epoch 4/10
18/18 [==============================] - 50s 3s/step - loss: 0.0797 - accuracy: 0.9782 - val_loss: 0.2345 - val_accuracy: 0.9502
Epoch 5/10
18/18 [==============================] - 49s 3s/step - loss: 0.0494 - accuracy: 0.9865 - val_loss: 0.2327 - val_accuracy: 0.9631
Epoch 6/10
18/18 [==============================] - 48s 3s/step - loss: 0.0317 - accuracy: 0.9929 - val_loss: 0.2524 - val_accuracy: 0.9760
Epoch 7/10
18/18 [==============================] - 49s 3s/step - loss: 0.0267 - accuracy: 0.9932 - val_loss: 0.2736 - val_accuracy: 0.9702
Epoch 8/10
18/18 [==============================] - 50s 3s/step - loss: 0.0168 - accuracy: 0.9965 - val_loss: 0.2574 - val_accuracy: 0.9716
Epoch 9/10
18/18 [==============================] - 49s 3s/step - loss: 0.0119 - accuracy: 0.9976 - val_loss: 0.2519 - val_accuracy: 0.9764
Epoch 10/10
18/18 [==============================] - 51s 3s/step - loss: 0.0104 - accuracy: 0.9981 - val_loss: 0.2918 - val_accuracy: 0.9640

[17]: <keras.callbacks.History at 0x2cf71c82550>
```

Save the model

File Edit View Run Kernel Tabs Settings Help

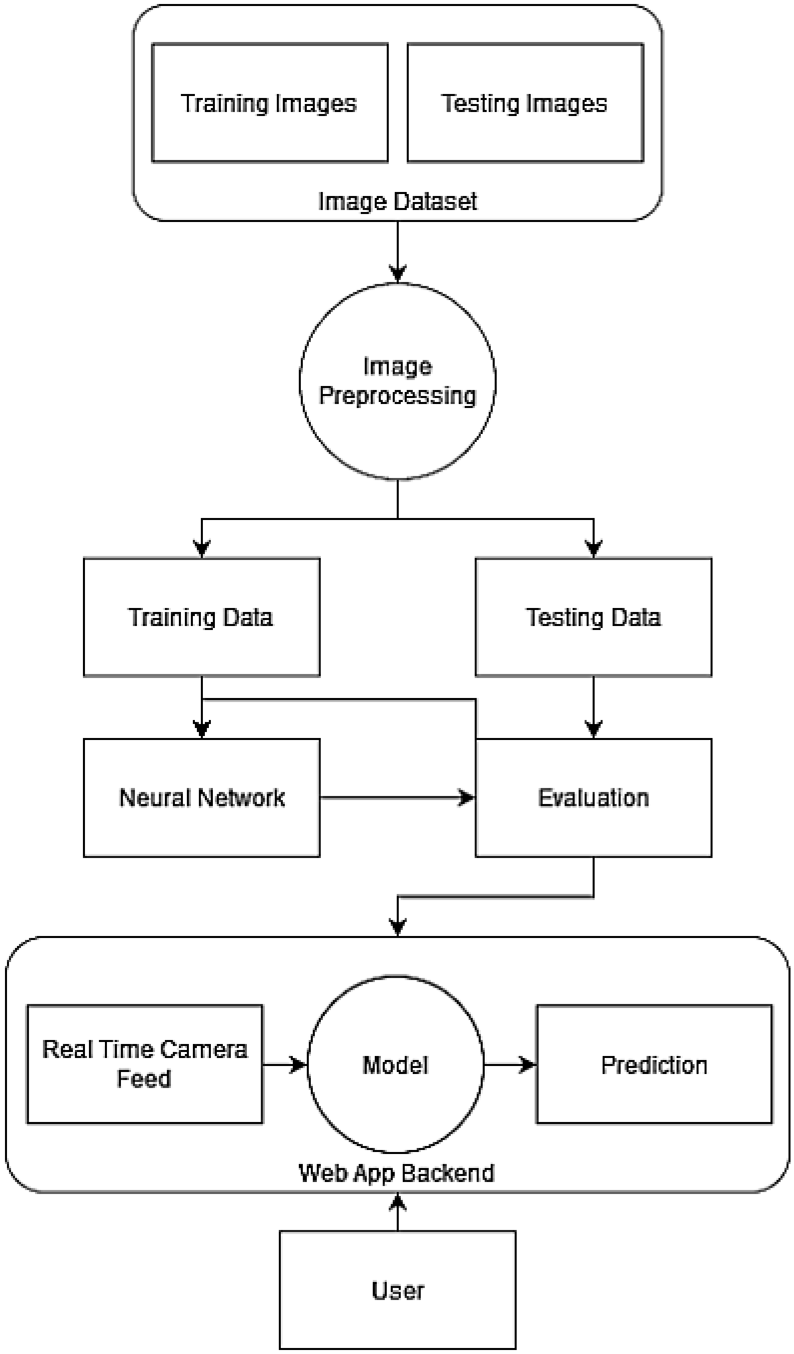
Externship Project.ipynb

```
[20]: model=load_model('HandSign.h5')
      img=image.load_img(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set\B\2.png',
                        target_size=(64,64))

[21]: img

[21]:
```



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

[23]: x

[23]: array([[[[0., 0., 0.],
              [0., 0., 0.],
              [0., 0., 0.],
              ...,
              [0., 0., 0.],
              [0., 0., 0.],
              [0., 0., 0.]],

             [[0., 0., 0.],
              [0., 0., 0.],
              [0., 0., 0.],
              ...,
              [0., 0., 0.],
              [0., 0., 0.],
              [0., 0., 0.]],

             [[0., 0., 0.],
              [0., 0., 0.],
              [0., 0., 0.],
              ...,
```

5 FLOWCHART

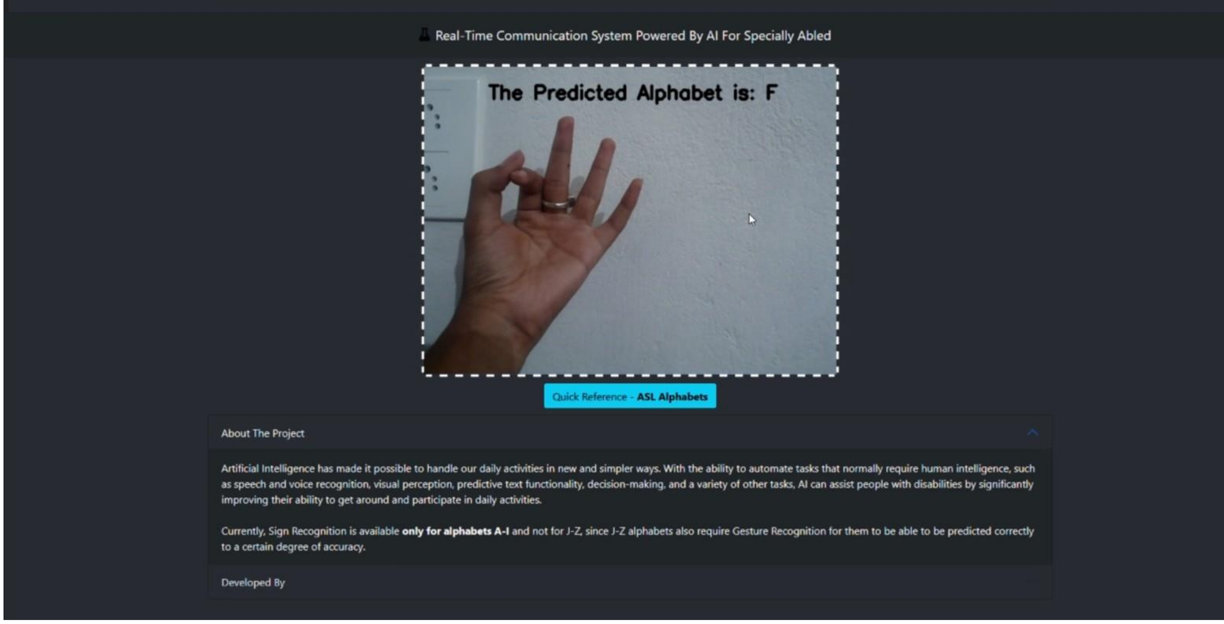


6 RESULT

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 recognised;

the equivalent Alphabet is shown on the screen.

Some sample images of the output are provided below:



7 ADVANTAGES & DISADVANTAGES

Advantages:

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

Disadvantages:

- The current model only works from alphabets A to I.
- In absence of gesture recognition, alphabets from J cannot be identified as they require some kind of gesture input from the user.
- 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

- 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 barrier to communication.

- As a result, others will be able to learn and comprehend sign language and communicate with the deaf and dumb via the web app.
- 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 people. Because it allows for two-way communication, the system aims to bridge the communication gap between deaf people and the 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, thanks to this project.

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

- https://youtu.be/5mDYijMfSzs
- https://drive.google.com/file/d/1ITbDvhLwyTTkuUYfNjOKhclZh7hDgi64/view?usp=sharing
- https://keras.io/api/preprocessing/image/
- https://keras.io/api/preprocessing/image/#imagedatasetfromdirectory-function
- https://youtu.be/umGJ30-15_A
- https://www.youtube.com/watch?v=mjKd1Tzl70I

- https://www.youtube.com/watch?v=lj4I_CvBnt0
- https://www.youtube.com/watch?v=bzX_auqvePs
- https://youtu.be/x6i43M7BAqE
- https://youtu.be/BzouqMGJ41k

12 APPENDIX

Source Code for model:

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

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

[3]: test_datagen = ImageDataGenerator(rescale=1/255)

[4]: x_train=train_datagen.flow_from_directory(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\training_set',target_size=(64,64),
                                              class_mode='categorical',batch_size=900)

     Found 15750 images belonging to 9 classes.

[5]: len(x_train)

[5]: 18

[6]: x_test=test_datagen.flow_from_directory(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set',target_size=(64,64),
                                              class_mode='categorical',batch_size=900)

     Found 2250 images belonging to 9 classes.

[7]: len(x_test)

[7]: 3

[8]: x_train.class_indices
```

```
[8]: x_train.class_indices

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

Importing Libraries

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

Creating Model

```
[10]: model=Sequential()
```

Adding Layers

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

[12]: model.add(MaxPooling2D(pool_size=(2,2)))

[13]: model.add(Flatten())
```

[14]:
```python
##Hidden Layer 1
model.add(Dense(300,activation='relu'))
##Hidden Layer 2
model.add(Dense(150,activation='relu'))
```

Output Layer

[15]:
```python
model.add(Dense(9,activation='softmax'))
```

Compile the model

[16]:
```python
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Fit the model

Fit the model

[17]:
```python
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
C:\Users\LEGION\AppData\Local\Temp/ipykernel_24492/234118701.py:1: 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=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
Epoch 1/10
18/18 [==============================] - 50s 3s/step - loss: 1.3134 - accuracy: 0.5780 - val_loss: 0.4696 - val_accuracy: 0.8787
Epoch 2/10
18/18 [==============================] - 49s 3s/step - loss: 0.3442 - accuracy: 0.9060 - val_loss: 0.3401 - val_accuracy: 0.9191
Epoch 3/10
18/18 [==============================] - 50s 3s/step - loss: 0.1466 - accuracy: 0.9592 - val_loss: 0.2585 - val_accuracy: 0.9418
Epoch 4/10
18/18 [==============================] - 50s 3s/step - loss: 0.0797 - accuracy: 0.9782 - val_loss: 0.2345 - val_accuracy: 0.9502
Epoch 5/10
18/18 [==============================] - 49s 3s/step - loss: 0.0494 - accuracy: 0.9865 - val_loss: 0.2327 - val_accuracy: 0.9631
Epoch 6/10
18/18 [==============================] - 48s 3s/step - loss: 0.0317 - accuracy: 0.9929 - val_loss: 0.2524 - val_accuracy: 0.9760
Epoch 7/10
18/18 [==============================] - 49s 3s/step - loss: 0.0267 - accuracy: 0.9932 - val_loss: 0.2736 - val_accuracy: 0.9702
Epoch 8/10
18/18 [==============================] - 50s 3s/step - loss: 0.0168 - accuracy: 0.9965 - val_loss: 0.2574 - val_accuracy: 0.9716
Epoch 9/10
18/18 [==============================] - 49s 3s/step - loss: 0.0119 - accuracy: 0.9976 - val_loss: 0.2519 - val_accuracy: 0.9764
Epoch 10/10
18/18 [==============================] - 51s 3s/step - loss: 0.0104 - accuracy: 0.9981 - val_loss: 0.2918 - val_accuracy: 0.9640
```

[17]: <keras.callbacks.History at 0x2cf71c82550>

Save the model

[18]:
```python
model.save('HandSign.h5')
```

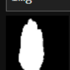
Testing the model

[19]:
```python
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

[20]:
```python
model=load_model('HandSign.h5')
img=image.load_img(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set\B\2.png',
                   target_size=(64,64))
```

[21]:
```python
img
```

[21]:



[22]:
```python
x=image.img_to_array(img)
```

[23]:
```python
x
```

```
          [0., 0., 0.],
          [0., 0., 0.],
          [0., 0., 0.]],

         [[0., 0., 0.],
          [0., 0., 0.],
          [0., 0., 0.],
          ...,
          [0., 0., 0.],
          [0., 0., 0.],
          [0., 0., 0.]],

         [[0., 0., 0.],
          [0., 0., 0.],
          [0., 0., 0.],
          ...,
          [0., 0., 0.],
          [0., 0., 0.],
          [0., 0., 0.]]], dtype=float32)

[24]: x.ndim

[24]: 3

[25]: x=np.expand_dims(x,axis=0)

[26]: x

[26]: array([[[[0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.],
           ...,
           [0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.]],
          [0., 0., 0.]]]], dtype=float32)

[27]: x.ndim

[27]: 4

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

[29]: pred

[29]: array([1], dtype=int64)

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

      B
```

Open CV

```
[31]: import cv2

[32]: img=cv2.imread(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set\B\2.png',1)

[33]: img

[33]: array([[[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

[34]: img1=cv2.imread(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set\B\2.png',0)

[35]: img1

[35]: array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)

[36]: print(img.shape)

      (64, 64, 3)

[37]: img=cv2.imread(r'C:\Users\LEGION\Desktop\Project Externship\Dataset\test_set\B\2.png',1)
      cv2.imshow('image',img)
      cv2.waitKey(0)
      cv2.destroyAllWindows()
```

CNN Video Analysis

CNN Video Analysis

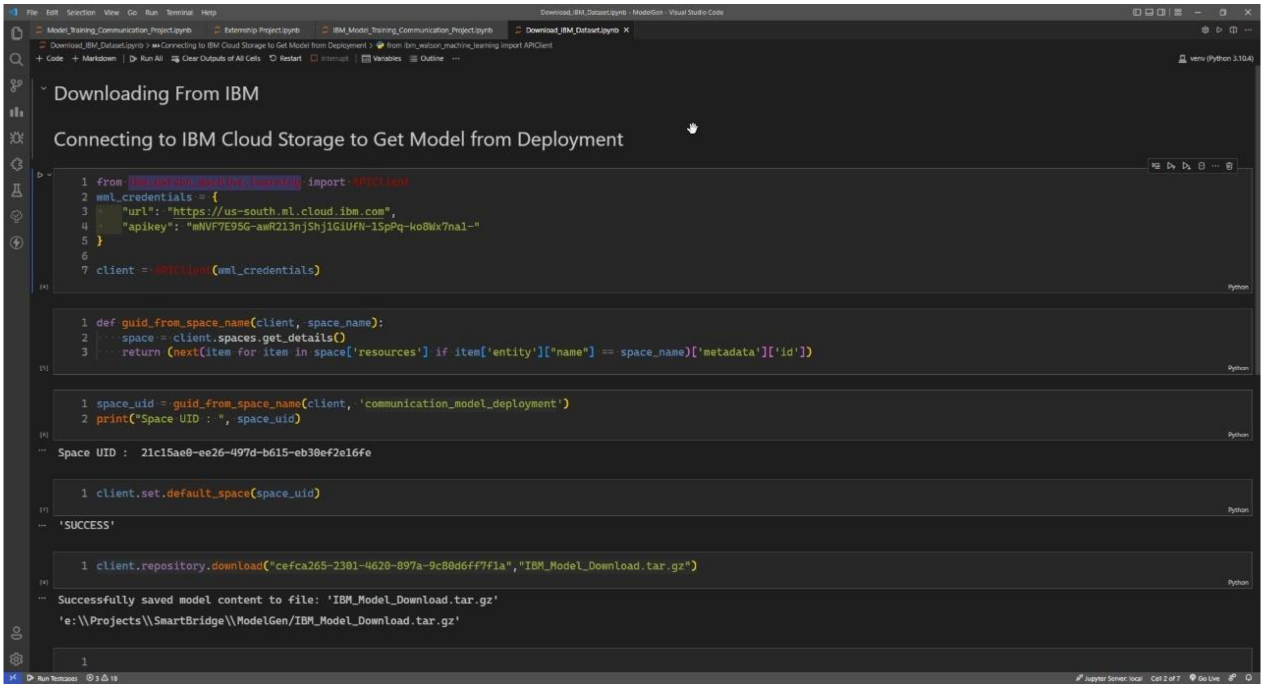
```
[1]: import cv2
     import numpy as np
     from tensorflow.keras.models import load_model
     from tensorflow.keras.preprocessing import image
     model=load_model('HandSign.h5')
     video=cv2.VideoCapture(0)
     index=['A','B','C','D','E','F','G','H','I']
     while 1:
         succes,frame=video.read()
         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]
         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()
```

Web app code:

```python
from flask import Flask, Response, render_template
from flask_socketio import SocketIO, emit
from camera import Video

app = Flask(__name__)
index=['A','B','C','D','E','F','G','H','I']
ll = None
@app.route('/')
def index():
    return render_template('index.html', predict_result=ll)

def gen(camera):
    global ll
    while True:
        frame = camera.get_frame()
        ll = camera.y
        yield(b'--frame\r\n'
            b'Content-Type: image/jpeg\r\n\r\n' + frame +
            b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    video = Video()
    return Response(gen(video), mimetype='multipart/x-mixed-replace; boundary = frame')
```

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

class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.roi_start = (50, 150)
        self.roi_end = (250, 350)
        # self.model = load_model('asl_model.h5') # Execute Local Trained Model
        self.model = load_model('IBM_Communication_Model.h5') # Execute IBM Trained Model
        self.index=['A','B','C','D','E','F','G','H','I']
        self.y = None
    def __del__(self):
        self.video.release()
    def get_frame(self):
        ret,frame = self.video.read()
        frame = cv2.resize(frame, (640, 480))
        copy = frame.copy()
        copy = copy[150:150+200,50:50+200]
        # Prediction Start
        cv2.imwrite('image.jpg',copy)
        copy_img = image.load_img('image.jpg', target_size=(64,64))
        # copy_img = image.load_img('image.jpg', target_size=(28,28))
        x = image.img_to_array(copy_img)
        x = np.expand_dims(x, axis=0)
        pred = np.argmax(self.model.predict(x), axis=1)
        self.y = pred[0]
        cv2.putText(frame,'The Predicted Alphabet is: '+str(self.index[self.y]),(100,50),
            cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),3)
        ret,jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()
```

IBM trained and downloaded model:



<u>**Sign language reference:**</u>