

Real-Time Communication System Powered by AI for Specially Abled

Submitted By

TEAM ID: PNT2022TMID27708

- TEAM LEADER : VASUGI E
- TEAM MEMBER 1 : THRISHIKA P
- TEAM MEMBER 2 : VAISHNAVI V
- TEAM MEMBER 3 : VIDHYA T S

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 into 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

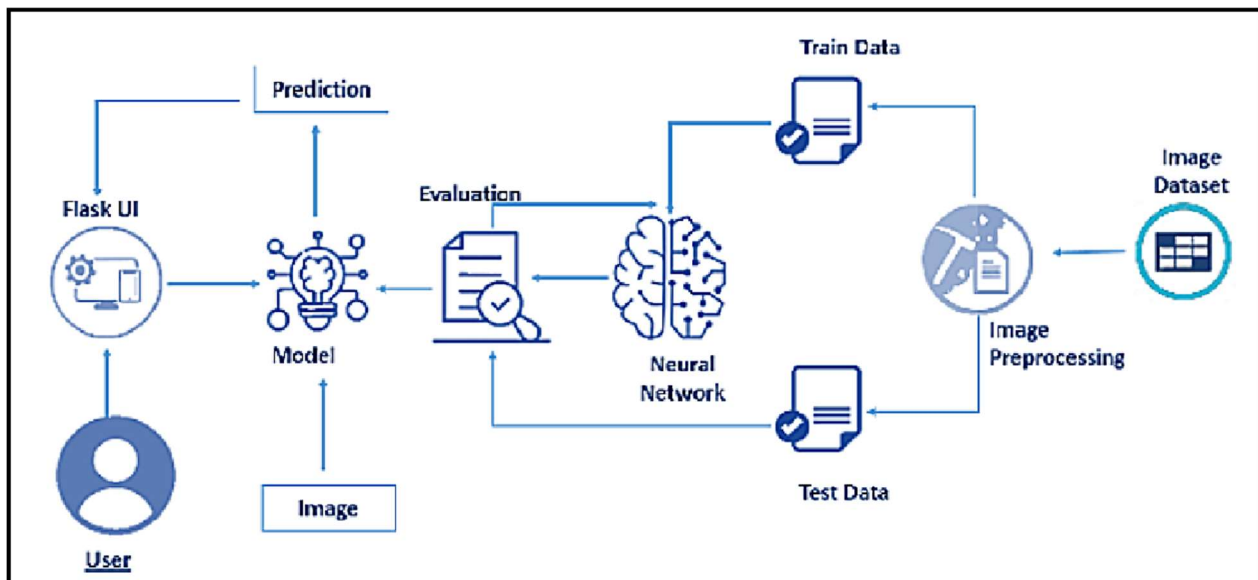
This paper describes the system that overcomes the problem faced by the speech and hearing impaired. The objectives of the research are as follow:

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-dumbperson 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:

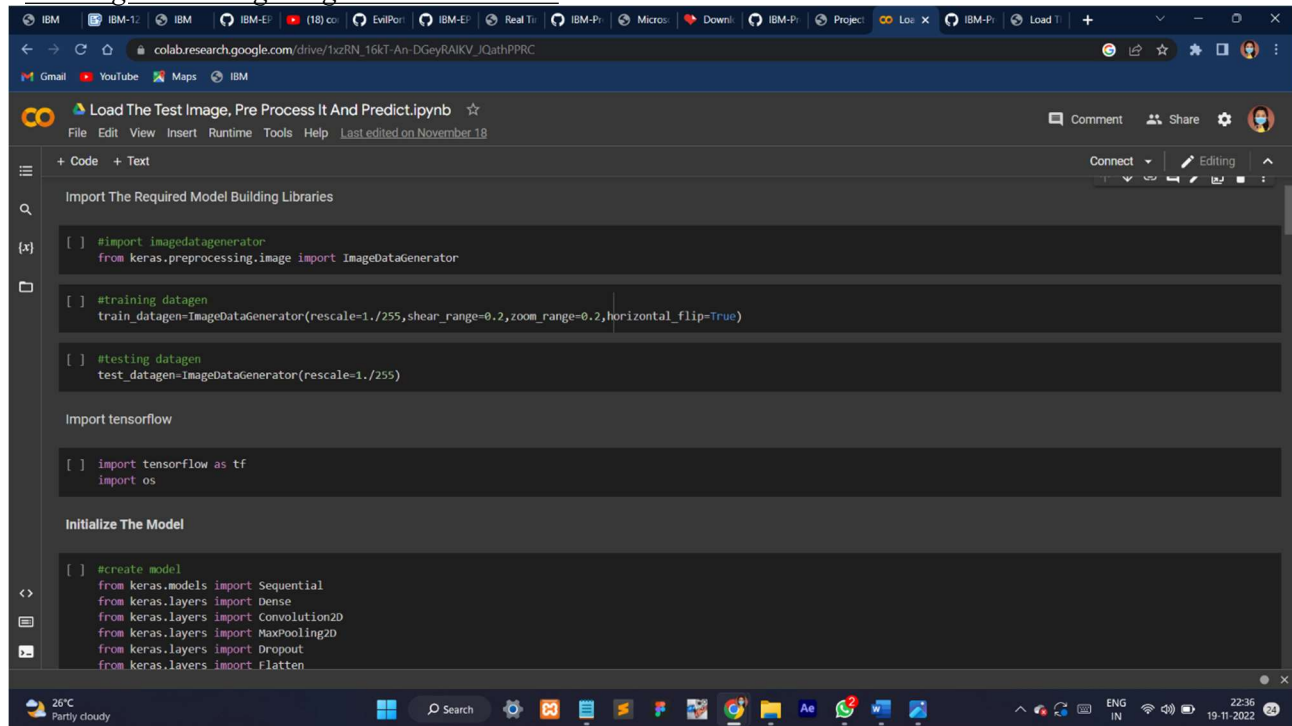
Operating System	Windows, Mac, Linux
CPU (for training)	Multi Core Processors (i3 or above/equivalent)
GPU (for training)	NVIDIA AI Capable / Google's TPU
WebCam	Integrated or External with FullHD Support

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

Training and Testing using Dataset Provided:



The screenshot displays a Jupyter Notebook titled "Load The Test Image, Pre Process It And Predict.ipynb" within a Google Colab environment. The notebook is open to a code cell containing the following Python code:

```
Import The Required Model Building Libraries

[ ] #import imagedatagenerator
    from keras.preprocessing.image import ImageDataGenerator

[ ] #training datagen
    train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

[ ] #testing datagen
    test_datagen=ImageDataGenerator(rescale=1./255)

Import tensorflow

[ ] import tensorflow as tf
    import os

Initialize The Model

[ ] #create model
    from keras.models import Sequential
    from keras.layers import Dense
    from keras.layers import Convolution2D
    from keras.layers import MaxPooling2D
    from keras.layers import Dropout
    from keras.layers import Flatten
```

The interface includes a top navigation bar with various icons, a left sidebar with file and code views, and a bottom status bar showing system information like temperature (26°C) and time (22:36 on 19-11-2022).

colab.research.google.com/drive/1xzRN_16kT-An-DGeyRAIKV_JQathPPRC

Load The Test Image, Pre Process It And Predict.ipynb

File Edit View Insert Runtime Tools Help Last edited on November 18

+ Code + Text

Unzipping the dataset

```
[ ] !unzip '/content/conversation engine for deaf and dumb (1).zip'
```

inflating: Dataset/training_set/g/1310.png
inflating: Dataset/training_set/g/1317.png
inflating: Dataset/training_set/g/1318.png
inflating: Dataset/training_set/g/1319.png
inflating: Dataset/training_set/g/132.png
inflating: Dataset/training_set/g/1320.png
inflating: Dataset/training_set/g/1321.png
inflating: Dataset/training_set/g/1322.png
inflating: Dataset/training_set/g/1323.png
inflating: Dataset/training_set/g/1324.png
inflating: Dataset/training_set/g/1325.png
inflating: Dataset/training_set/g/1326.png
inflating: Dataset/training_set/g/1327.png
inflating: Dataset/training_set/g/1328.png
inflating: Dataset/training_set/g/1329.png
inflating: Dataset/training_set/g/133.png
inflating: Dataset/training_set/g/1330.png
inflating: Dataset/training_set/g/1331.png
inflating: Dataset/training_set/g/1332.png
inflating: Dataset/training_set/g/1333.png
inflating: Dataset/training_set/g/1334.png
inflating: Dataset/training_set/g/1335.png
inflating: Dataset/training_set/g/1336.png
inflating: Dataset/training_set/g/1337.png
inflating: Dataset/training_set/g/1338.png
inflating: Dataset/training_set/g/1339.png
inflating: Dataset/training_set/g/134.png
inflating: Dataset/training_set/g/1340.png
inflating: Dataset/training_set/g/1341.png

25°C Partly cloudy

colab.research.google.com/drive/1xzRN_16kT-An-DGeyRAIKV_JQathPPRC

Load The Test Image, Pre Process It And Predict.ipynb

File Edit View Insert Runtime Tools Help Last edited on November 18

+ Code + Text

Applying ImageDataGenerator to test set

```
[ ] x_test=x_test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
class_mode='categorical',color_mode="grayscale")
```

Found 2250 images belonging to 9 classes.

```
[ ] a=len(x_train)
b=len(x_test)
```

Length of training set

```
[ ] print(a)
```

79

Length of test set

```
[ ] print(b)
```

12

Add Layers

25°C Partly cloudy

colab.research.google.com/drive/1xzRN_16kT-An-DGeyRAIKV_JQathPPRC

Load The Test Image, Pre Process It And Predict.ipynb

File Edit View Insert Runtime Tools Help Last edited on November 18

+ Code + Text

Fit The Model

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


/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which is the entry point for launching an IPython kernel.

Epoch 1/10
79/79 [=====] - 90s 1s/step - loss: 0.3965 - accuracy: 0.8746 - val_loss: 0.2797 - val_accuracy: 0.9529
Epoch 2/10
79/79 [=====] - 86s 1s/step - loss: 0.0419 - accuracy: 0.9884 - val_loss: 0.2846 - val_accuracy: 0.9751
Epoch 3/10
79/79 [=====] - 84s 1s/step - loss: 0.0195 - accuracy: 0.9947 - val_loss: 0.3436 - val_accuracy: 0.9751
Epoch 4/10
79/79 [=====] - 87s 1s/step - loss: 0.0083 - accuracy: 0.9982 - val_loss: 0.3722 - val_accuracy: 0.9751
Epoch 5/10
79/79 [=====] - 83s 1s/step - loss: 0.0066 - accuracy: 0.9983 - val_loss: 0.4095 - val_accuracy: 0.9756
Epoch 6/10
79/79 [=====] - 88s 1s/step - loss: 0.0072 - accuracy: 0.9979 - val_loss: 0.3874 - val_accuracy: 0.9756
Epoch 7/10
79/79 [=====] - 86s 1s/step - loss: 0.0059 - accuracy: 0.9985 - val_loss: 0.3891 - val_accuracy: 0.9747
Epoch 8/10
79/79 [=====] - 86s 1s/step - loss: 0.0027 - accuracy: 0.9992 - val_loss: 0.4429 - val_accuracy: 0.9756
Epoch 9/10
79/79 [=====] - 84s 1s/step - loss: 0.0073 - accuracy: 0.9981 - val_loss: 0.4907 - val_accuracy: 0.9756
Epoch 10/10
79/79 [=====] - 85s 1s/step - loss: 0.0048 - accuracy: 0.9987 - val_loss: 0.4866 - val_accuracy: 0.9702
<keras.callbacks.History at 0x7f445adcd7d0>

Save The Model

```
[ ] #load the model  
model=load_model('asl.png2.h5')
```

```
[ ] img=image.load_img('/content/Dataset/test_set/A/10.png',target_size=(400,500))  
img
```



25°C Partly cloudy

colab.research.google.com/drive/1xzRN_16kT-An-DGeyRAIKV_JQathPPRC

Load The Test Image, Pre Process It And Predict.ipynb

File Edit View Insert Runtime Tools Help Last edited on November 18

+ Code + Text

Connect Editing

IBM IBM-12 IBM IBM-EP (18) co EvilPort IBM-EP Real Ti IBM-Pi Micro: Downl IBM-Pi Project Lo: x IBM-Pi Load T

colab.research.google.com/drive/1xzRN_16kT-An-DGeyRAIKV_JQathPPRC

Gmail YouTube Maps IBM

Load The Test Image, Pre Process It And Predict.ipynb

File Edit View Insert Runtime Tools Help Last edited on November 18

Comment Share Settings


+ Code + Text Connect Editing

Load The Test Image, Pre-Process It And Predict

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

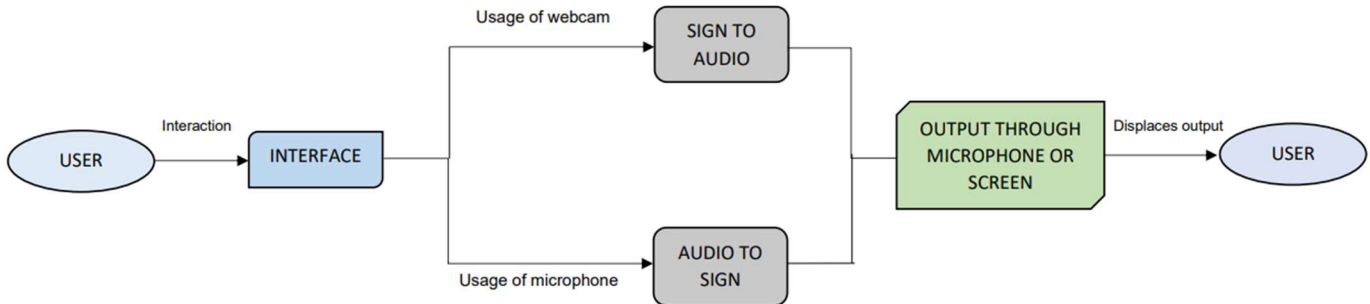
[ ] arr= image.img_to_array(img)

[ ] frame=cv2.imread('/content/Dataset/test_set/A/10.png')
data=detect(frame)
from google.colab.patches import cv2_imshow
cv2_imshow(frame)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



25°C Partly cloudy Search 22:37 19-11-2022

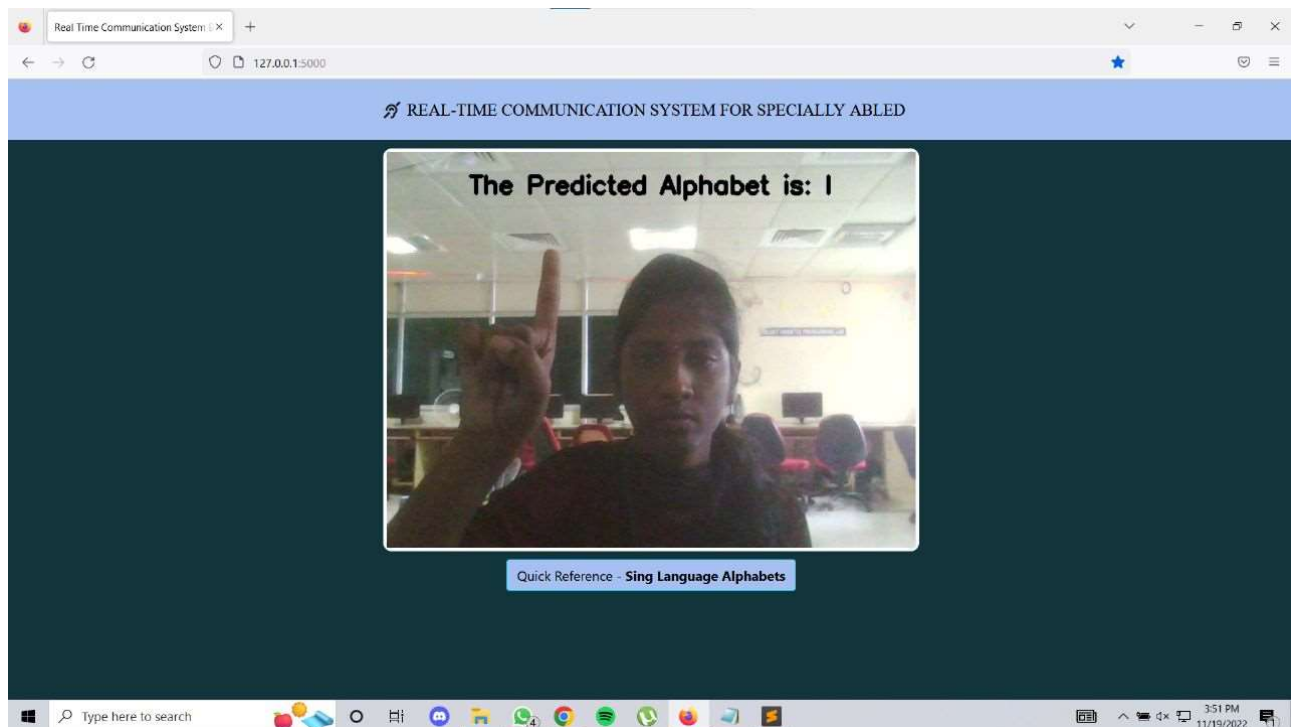
5. FLOWCHART

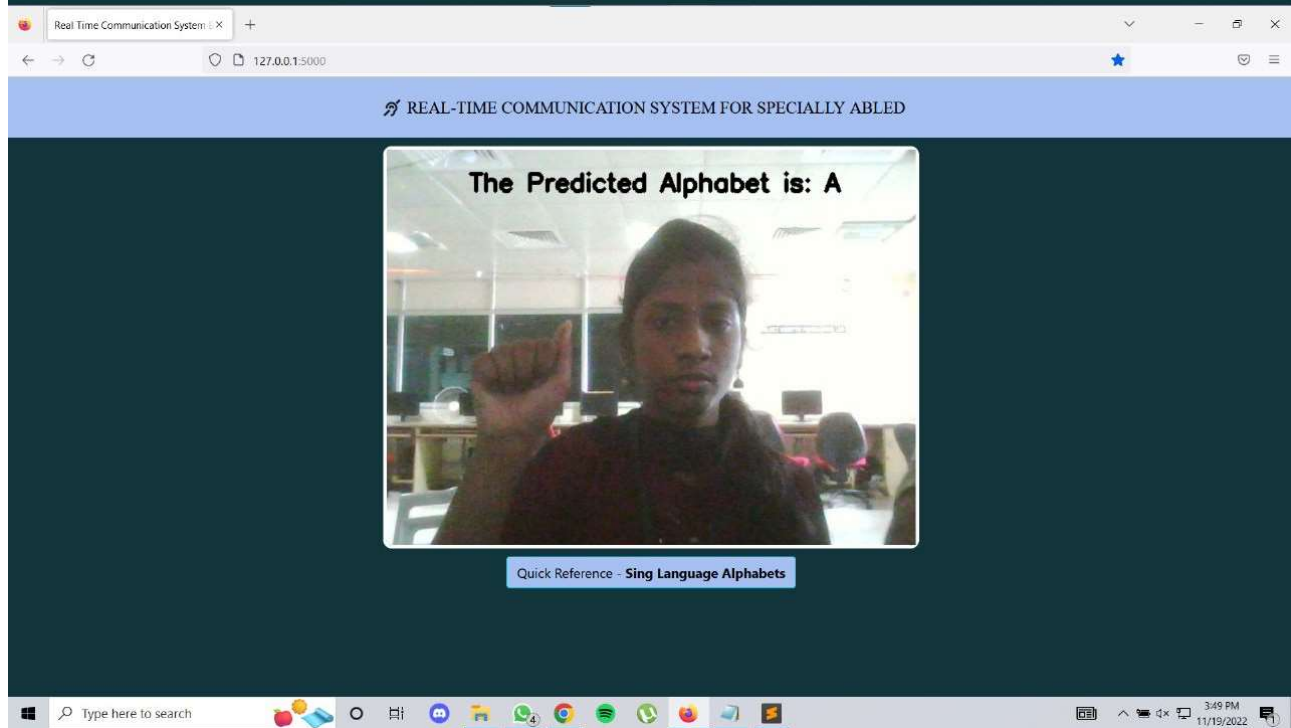
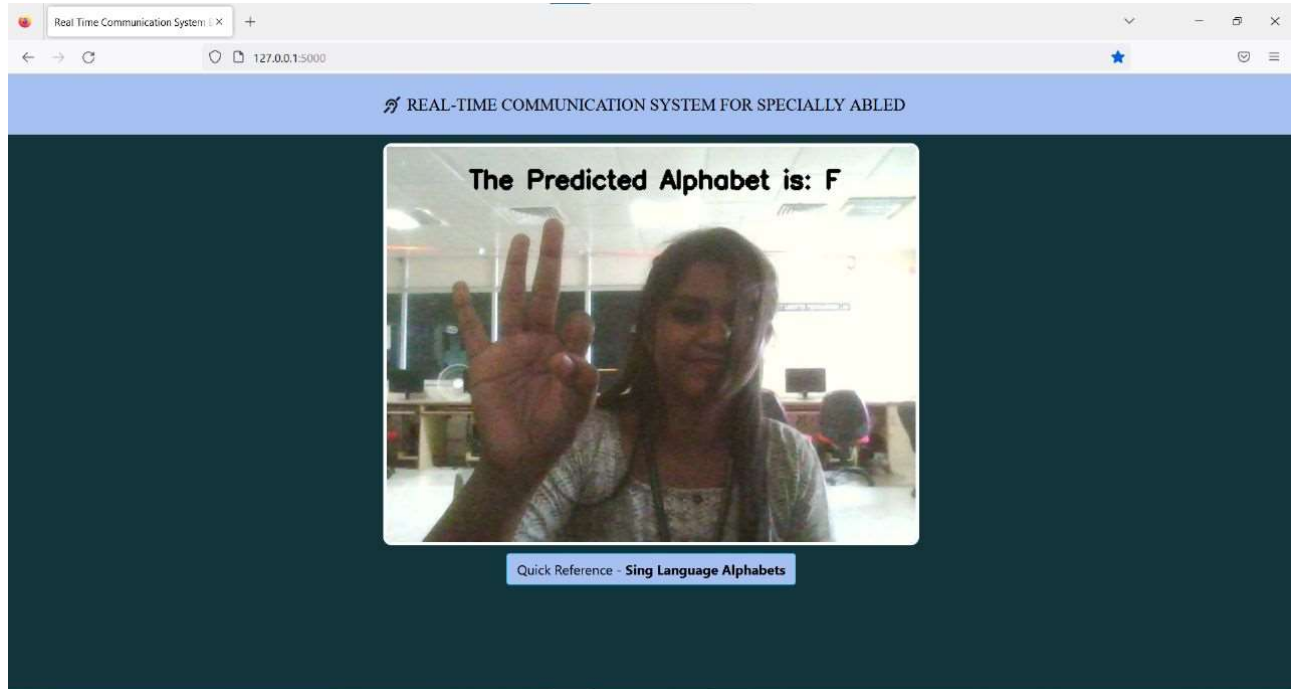


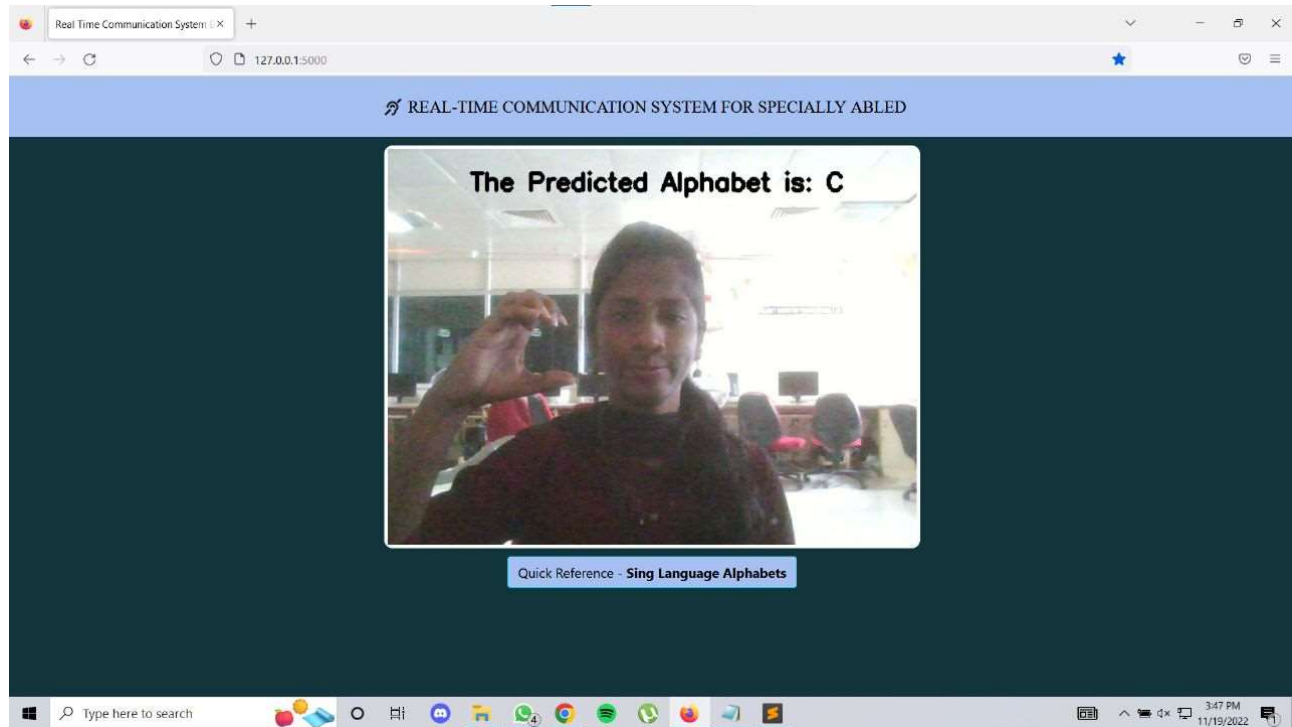
6. RESULT

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 recognise the equivalent Alphabet is shown on the screen.

Some sample images of the output are provided below:







7. ADVANTAGES & 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 barrier to 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 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

1. Environment Setup: <https://www.youtube.com/watch?v=5mDYijMfSzs>
2. Sign Languages Dataset: <https://drive.google.com/file/d/1ITbDvhLwyTTkuUYfNjOKhcIZh7hDgi64/view?usp=sharing>
3. Keras Image Processing Doc: <https://keras.io/api/preprocessing/image/>
4. Keras ImageDataset 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>

12. APPENDIX

Source Code:

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
  <title>Real Time Communication System By AI</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
  <link rel="stylesheet" href="assets/css/Banner-Heading-Image.css">
  <link rel="stylesheet" href="assets/css/Navbar-Centered-Brand.css">
  <link rel="stylesheet" href="assets/css/styles.css">
</head>

<style type="text/css">
.column {
  float: left;
  width: 33.33%;
}

/* Stops the float property from affecting content after the columns */
.columns:after {
  content: "";
  display: table;
  clear: both;
}
.column {
  width: 100%;
}
@media (min-width: 48em) {
  .column {
    float: left;
    width: 33.33%;
  }
}
/* Stops the float property from affecting content after the columns */
.columns:after {
  content: "";
  display: table;
  clear: both;
}
}
</style>

<body style="background-color: #12343b">

  <nav class="navbar navbar-light navbar-expand-md py-3" style="background: #a7c0f2;">
    <div class="container">
      <div></div><a class="navbar-brand d-flex align-items-center" href="#">
        <span class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-content-center align-items-center me-2 bs-icon">
          <i class="fas fa-deaf" ></i></span><span style="color: #000; font-family: serif;">REAL-TIME COMMUNICATION
```

```

        SYSTEM&nbsp;FOR SPECIALLY ABLED</span></a>
    </div></div>
</div>
</nav>
<div class="column">
<section>
    <div class="d-flex flex-column justify-content-center align-items-center">
        <div class="d-flex flex-column justify-content-center align-items-center" id="div-video-feed"
            style="width: 640px;height: 480px;margin: 10px;min-height: 480px;min-width: 640px;border-radius: 10px;border: 4px #fff;border-style:
solid;">
            
            </div>
        </div>
        <div class="d-flex flex-column justify-content-center align-items-center" style="margin-bottom: 10px;"><button
            class="btn btn-info" type="button" data-bs-target="#modal-1" data-bs-toggle="modal" style="background: #a7c0f2;">Quick Reference
            <strong> Sing Language Alphabets</strong></button></div>
    </section>
</div></div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

style.css

```

.bs-icon {
    --bs-icon-size: .75rem;
    display: flex;
    flex-shrink: 0;
    justify-content: center;
    align-items: center;
    font-size: var(--bs-icon-size);
    width: calc(var(--bs-icon-size) * 2);
    height: calc(var(--bs-icon-size) * 2);
    color: var(--bs-primary);
}

.bs-icon-xs {
    --bs-icon-size: 1rem;
    width: calc(var(--bs-icon-size) * 1.5);
    height: calc(var(--bs-icon-size) * 1.5);
}

.bs-icon-sm {
    --bs-icon-size: 1rem;
}

.bs-icon-md {
    --bs-icon-size: 1.5rem;
}

.bs-icon-lg {
    --bs-icon-size: 2rem;
}

.bs-icon-xl {
    --bs-icon-size: 2.5rem;
}

.bs-icon.bs-icon-primary {
    color: var(--bs-white);
    background: var(--bs-primary);
}

.bs-icon.bs-icon-primary-light {
    color: var(--bs-primary);
    background: rgba(var(--bs-primary-rgb), .2);
}

.bs-icon.bs-icon-semi-white {
    color: var(--bs-primary);
    background: rgba(255, 255, 255, .5);
}

```

```
.bs-icon.bs-icon-rounded {
    border-radius: .5rem;
}

.bs-icon.bs-icon-circle {
    border-radius: 50%;
}
```

App.py

```
from flask import Flask, Response, render_template
from camera import Video

app = Flask(__name__)
@app.route("/")
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        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')

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

camera.py

```
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))
        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()
```

main.py

```
import cv2

video = cv2.VideoCapture(0)

while True:
    ret, frame = video.read()
    cv2.imshow("Frame", frame)
    k = cv2.waitKey(1)
    if k == ord('q'):
        break

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

American Sign Language Standard Reference:

