REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

(TEAM ID:PNT2022TMID43024)

IBM PROJECT REPORT

Submitted by

SANJAY HANANTH V	**(713119104017)**
KEERTHIGA M	**(713119104006)**
KEERTHI P	**(713119104005)**
SUMATHI V	**(713119104019)**

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

SRI SAI RANGANATHAN ENGINEERING COLLEGE

COIMBATORE-641 109

ANNA UNIVERSITY: CHENNAI 600 025

TABLE OF CONTENTS

ABSTRACT

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. INTRODUCTION

1.1 Project Overview:

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, as well as convert speech into understandable sign language for the deaf and dumb. We are making use of a convolution neural network to create a model that is trained on different hand gestures. An app is built which uses this model. This app enables deaf and dumb people to convey their information using signs which get converted to human-understandable language and speech is given as output.

1.2 Purpose:

we are making use of a convolution neural network to create a model that is trained on different hand gestures. An app is built which uses this model. This app enables deaf and dumb people to convey their information using sings which get converted to human-understandable language and speech is given as output.

2. LITERATURE SURVEY

2.1 Existing problem:

- A recommendation isn't what you though it'd be. Your new hire has the skills, but somehow, their personality doesn't fit with the current team.
- Dealing with a charismatics referee.
- you could lack diversity and ideas when hiring via referrals only.

2.2 References:

REFERENCE PAPER 1:

TOPIC: "A Mechanism for Seamless Cryptographic Rekeying in Real Time Communication Systems"

AUTHOR:" Heiko HYPERLINK"

ABSTRACT:/

Cryptographic protection of messages requires frequent updates of the symmetric cipher key for encryption and decryption,respectively.protocols of legacy ITsacurity,TLS,SSH,or MAC sec implement rekeying under the assumption and,second,dedicated control messages to orchestrate the process can be exchanged in real-time automation applications, the first is generally prohibitive,while the second may induce problematic traffic patterns on the network we present a novel seamless rekeying approach ,which can be embedded into cyclic application data exchanges.Although,being agnostic to the underlying real-time communication system,we developed a demonstrator emulating the widespread industrial Ethernet systemPROFINET IO and

REFERENCE PAPER 2

TOPIC: “Expertise referrals using a real-time communication system”

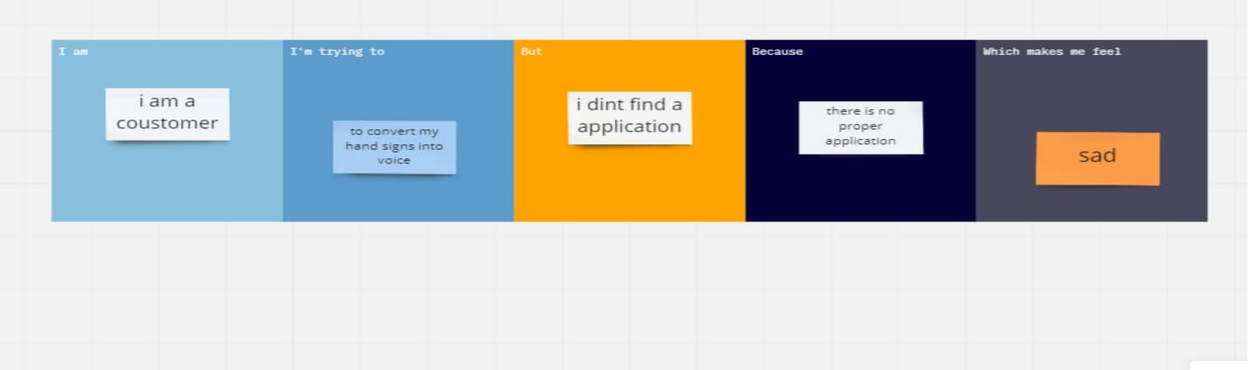
AUTHOR:” Marc Dreyfus”

ABSTRACT:

A computer-implement method of providing expertise based referrals can include receiving, from user, a voucher specifying a second user seeking expertise and a third user as a potential subject matter expert. Responsive to execution of the voucher, an instant messaging session between the second user and the third user can be established and an input from the second user indicating whether a posed question from the second user is resolved can be received. When the posed question is resolved, a role of maven can be assigned to the first user and a role of subject matter expert can be assigned to the third user. A transcript of the instant messaging session between the second user and the third user, a reference to the first user with the assigned role, and a reference to the third user with the assigned role can be stored as part of a referrals transaction.

2.3 Problem Statement Definition

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.



I am	I'm trying to	But	Because	Which makes me feel
i am a coustomer	to convert my hand signs into voice	i dint find a application	there is no proper application	sad

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

THINK AND FEEL:

- Will it detect gestures
- Will it progress?
- Do invent better things
- Conversion of voice is effective for deaf and dumb

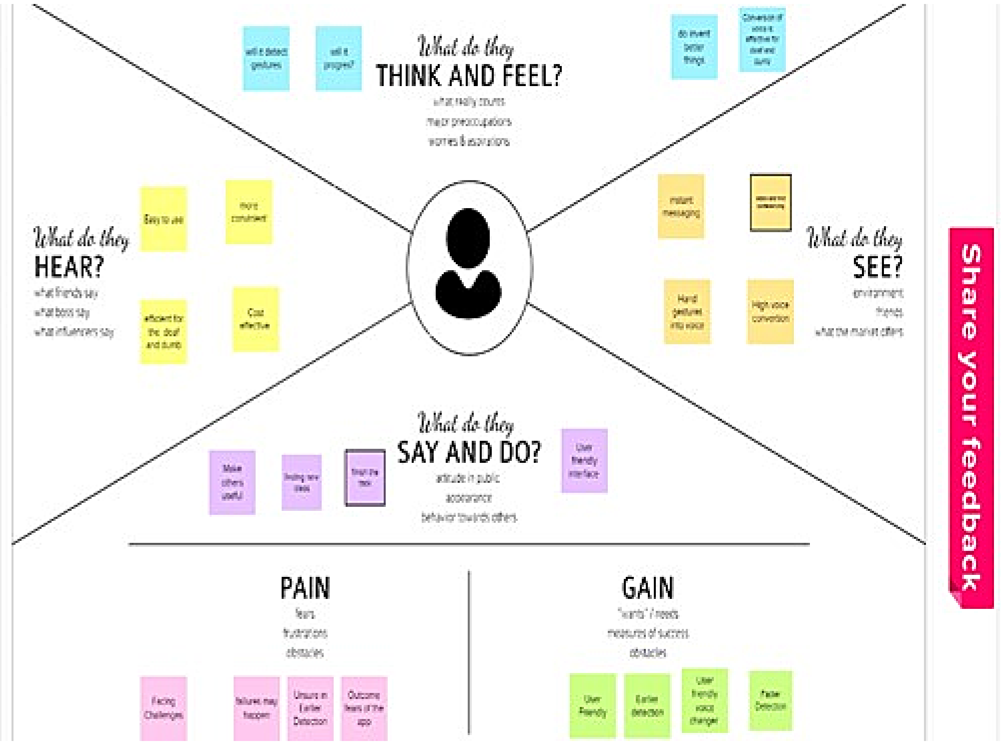
HEAR:

- Easy to use
- More convenient
- Efficient for the deaf and dumb
- Cost effective

SEE:

- Instant messaging
- Hand gestures into voice
- High voice conversion

- Video and see conferencing



PAIN:

- Facing challenges
- Failures may happen
- Unsure in earlier detection
- Outcome tears of the app

GAIN:

- User friendly
- Earlier detection
- User friendly voice changer
- Faster detection

SAY AND DO:

- Make others useful
- Finding new ideas
- Finish the task
- User friendly interface

3.2 Ideation & Brainstorming

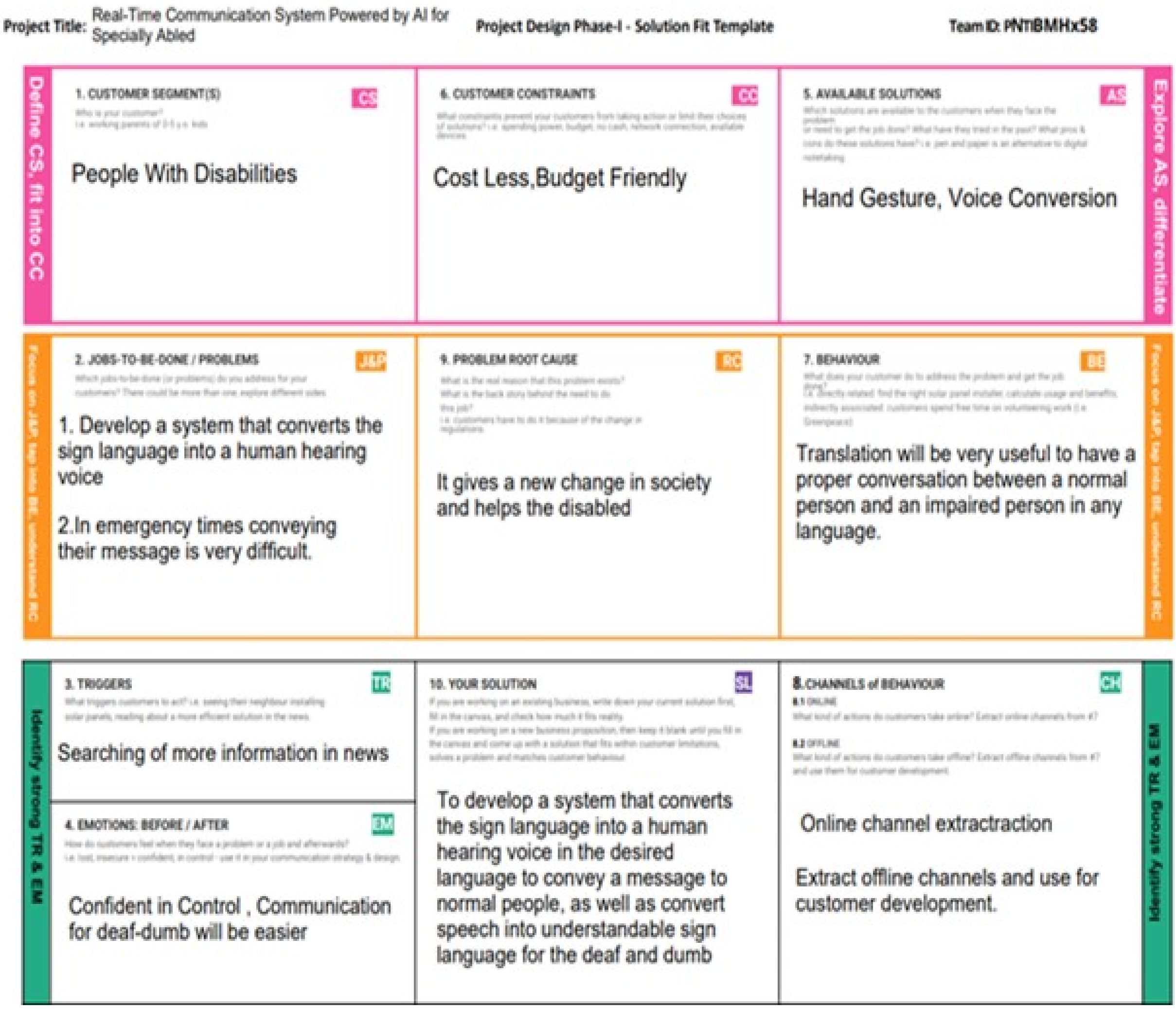
3.3 Proposed Solution:

Proposed Solution Template: Project team shall fill the following information in proposed solution template

S.NO	**PARAMETER**	**Description**
●	Problem Statement (Problem to be solved)	Differently able like dump and mute people can communicate through the sign language normal people those who do not know the sign language feels difficult to communicate with them.
●	Idea / Solution description	To over come this problem we have an idea that an application is created to communicate with the normal people
●	Novelty / Uniqueness	This process the image of a person who is using sign language voice by analyzing the sign used and convert it into
●	Social Impact/ Customer Satisfaction	Differently abled people feel free to communicate and it bring a huge difference comparing to past.
●	Business Model (Revenue Model)	There are any people in the world who is differently able this application will become more popular among them and it will be installed by all and it will be used and so it will produce more money.

●	Scalability of the Solution	Thus this would bring a new evolution in real-time communication system powered by Ai for specially able with less time and safe enough resources.

3.4 Problem Solution fit:



Project Title: Real-Time Communication System Powered by AI for Specially Abled

Project Design Phase-I - Solution Fit Template

Team ID: PNTIBMHx58

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer? i.e. working parents of 0-5 y.o. kids

People With Disabilities

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Cost Less,Budget Friendly

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Hand Gesture, Voice Conversion

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

1. Develop a system that converts the sign language into a human hearing voice

2.In emergency times conveying their message is very difficult.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.

It gives a new change in society and helps the disabled

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Translation will be very useful to have a proper conversation between a normal person and an impaired person in any language.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Searching of more information in news

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Confident in Control , Communication for deaf-dumb will be easier

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

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, as well as convert speech into understandable sign language for the deaf and dumb

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Online channel extractraction

Extract offline channels and use for customer development.

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional requirement:

Following are the functional requirements of the proposed solution.

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Registration	Registration through Gmail using mobile or laptop
FR-2	User Confirmation	Conformation via email through mobile or laptop
FR-3	User information gathering	User will be shown with registration box which has Name, Gender, disabilities ,mobile number
FR-4	User otp conformation	The page will be opened with conformation box with OTP- conformation by email/mobile number.
FR-5	User access conformation	The camera/microphone allowance will be conformed by the user.
FR-6	User screen	User will be allowed to enact and the hand gestures will be show on the screen
FR-7	User tools	There are many tools used for gestures such as automatic hand shape tools and hearing tools such as high loud less loud beam volume etc.
FR-8	User microphone access	The the voice will be audible in microphone.
FR-9	User feedback	The user will be asked to fill the feedback form.
FR-10	User log out	The user will exit the app after logout

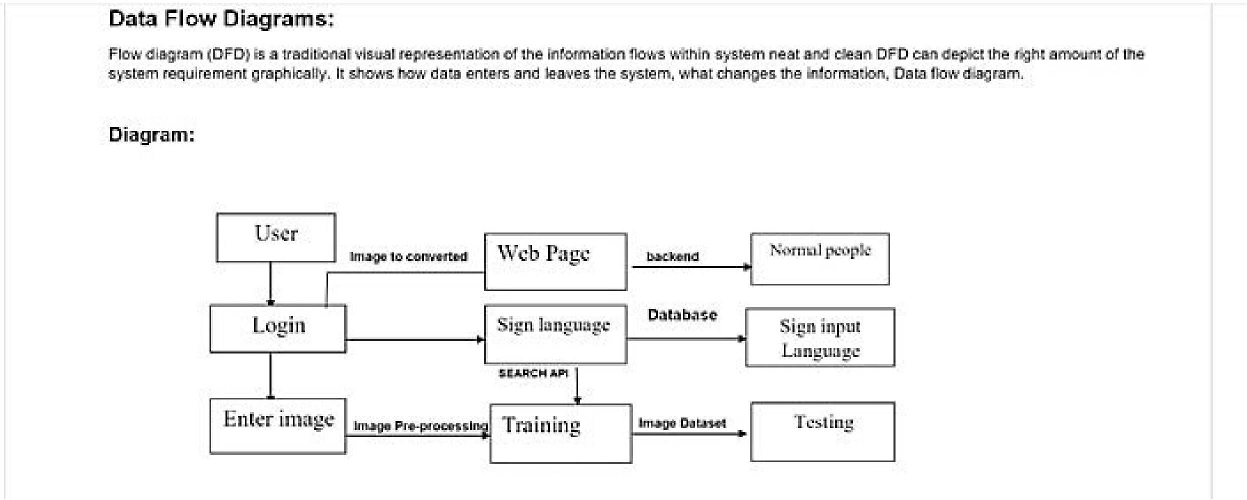
4.2 Non-functional requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The user will be easily able to convert their hand gestures into sign language using sign convertor.
NFR-2	Security	It is highly secured using conformation of mail and mobile number using OTP.
NFR-3	Reliability	If mobile or system issues happen it will be resolved by sending an email the speed of the conversion will be faster.
NFR-4	Performance	It is highly audible and with high quality screening so the user can easily hear and convert their hand gestures into voice the conversion
NFR-5 A	Availability	Most of the tools are available for converting hand gestures and give high support for the user for hearing the voice clearly.
NFR-6	Scalability	There is a large selection of device of help people with deaf and dumb disabilities such as hearing machine, voice convertor etc.

5. PROJECT DESIGN

5.1 Data flow Diagrams:

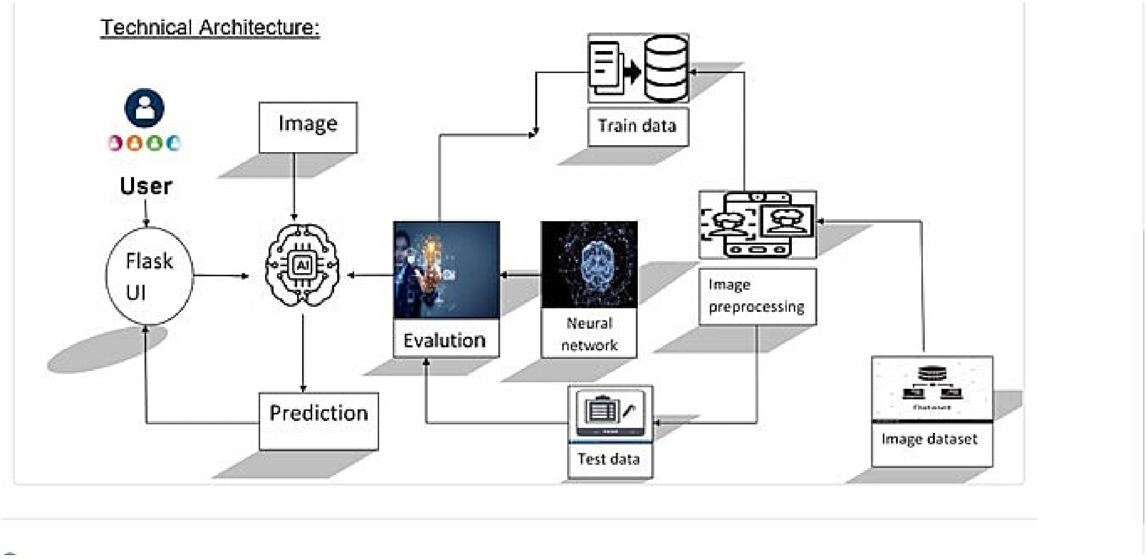


Data Flow Diagrams:

Flow diagram (DFD) is a traditional visual representation of the information flows within system neat and clean DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, Data flow diagram.

Diagram:

5.2 Solution & Technical Architecture



Technical Architecture:

5.3 User Stories

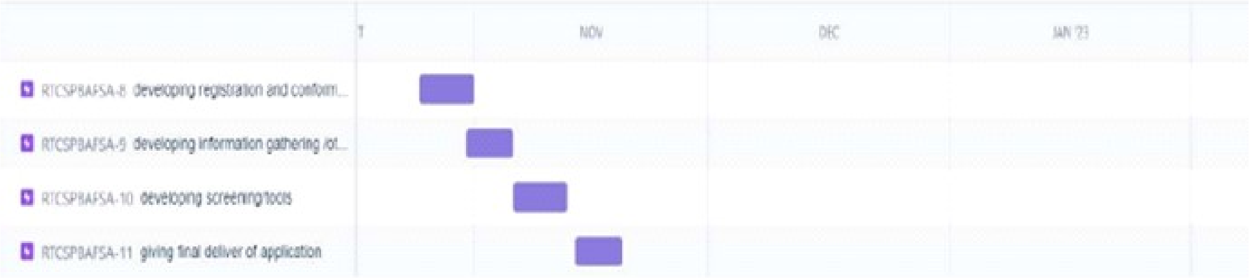
Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Hand gestures	USN-1	As a user, who are deaf and dumb I want to be able to convert hand gestures into sign language.	I can access my account / dashboard	High	Sprint-1
Customer (sign language)	USN-2	As a user, going to convert hand gestures into sign language for disabled with deaf and dumb so it will be useful for understanding	I can receive confirmation email & click confirm	High	Sprint-1
Customer (impaired user)	USN-3	As a user who is hearing the audio will hear the sign languages into voice.	I can register & access the dashboard with sign convertor.	Low	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password	1	High	Keerthi Sumathi Keerthiga
Sprint-1	conformation	USN-2	As a user, I will receive confirmation email once I have registered for the application	2	High	Keerthi Sumathi Keerthi
Sprint-2	Information gathering/ OTP conformation	USN-3	A a user i will give my information in conformation box. Sending the OTP to registered mobile member	1	Low	Keerthi Sumathi Keerthiga
Sprint-2	Access conformation	USN-3	As a user i can give allowance for the microphone and camera.	2	Medium	Keerthi Sumathi Keerthiga
Sprint-3	screen	USN-5	As a user i can See my gestures Which i am using in the screen.	2	Medium	Keerthi Sumathi Keerthiga
Sprint-3	tools	USN-6	As a user i can access all tools for the gestures and voice level such as high level and low level sound .	2	Medium	Keerthi Sumathi Keerthiga
Sprint-4	Final deliver	USN-7	The final application will be delivered	2	Medium	Keerthi Sumathi Keerthiga

6.2 Sprint Delivery Schedule6.3 Reports from JIRA:



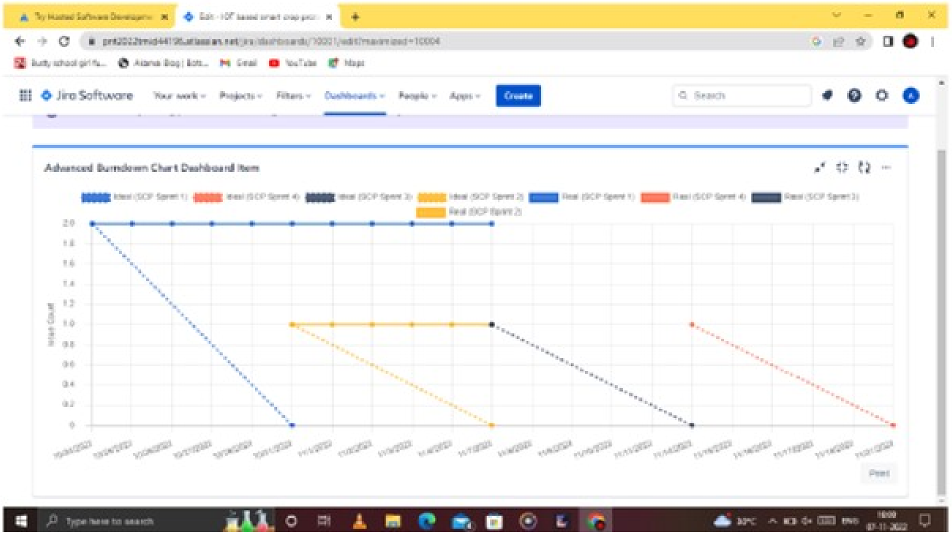
RTCSPBAFSA-8 developing registration and conform...

RTCSPBAFSA-9 developing information gathering iot...

RTCSPBAFSA-10 developing screening/tools

RTCSPBAFSA-11 giving final deliver of application

Burndown Chart:



7.CODING & SOLUTIONING

7.1 Feature 1:

SPRINT 1:

IMPORTING NECESSARY LIBRARIES

In [1]:

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
```

RENAMING DATA FILES

In [26]:

```
def rename_imgs(file_name):
    folder_path = r'test_dataset/'+file_name


    num = 0
    for file in os.listdir(folder_path):
        # if num%10 == 0:
        #     print(f'Renamed {num} files...')
        # os.rename(folder_path+'\\'+file, folder_path+'\\'+file_name+'_'+str(num)+'.jpeg')
        num += 1
fn = 'Space'
```

```
rename_imgs(fn)
```

In [7]:

```
file_names = '0123456789'+'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for fn in file_names:
    rename_imgs(fn)
```

DISPLAYING SAMPLE IMAGES FROM DATASET

In [8]:

```
train_data_path = 'train_dataset/'
test_data_path = 'test_dataset/'
```

In [9]:

```
def display(img,sign=None):

    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    fig = plt.figure(figsize=(7,7))
    ax = fig.add_subplot(111)
    plt.title(sign)
    ax.imshow(img)
```

Training Data Images

In [10]:

```
sign_img = cv2.imread(train_data_path+'O/O_234.jpeg')
display(sign_img,'a')
sign_img = cv2.imread(train_data_path+'A/A_204.jpeg')
display(sign_img,'A')
sign_img = cv2.imread(train_data_path+'3/3_340.jpeg')
display(sign_img,'3')
sign_img = cv2.imread(train_data_path+'M/M_100.jpeg')
display(sign_img,'M')
```

```
sign_img = cv2.imread(train_data_path+'S/S_10.jpeg')
display(sign_img,'Space')
```

Test Data Images

In [15]:

```
sign_img = cv2.imread(test_data_path+'S/S_15.jpeg')
display(sign_img,'S')
sign_img = cv2.imread(test_data_path+'Z/Z_1.jpeg')
display(sign_img,'Z'
sign_img = cv2.imread(test_data_path+'7/7_8.jpeg')
display(sign_img,'7')

)
```

AUGMENTATION AND PREPROCESSING THE DATASET

Creating ImageDataGenerator

In [18]:

```
image_gen = ImageDataGenerator(rotation_range=30,
                    width_shift_range=0.1,
                    height_shift_range=0.1,
                    shear_range=0.2,
                    zoom_range=0.2,
                    rescale=1/255,
                    horizontal_flip=True,
                    fill_mode='nearest',
                    validation_split=0.25)
```

Original Image

In [19]:

```
sign_img = cv2.imread(train_data_path+'3/3_100.jpeg')

display(sign_img,'3')
```

Augmented Images

In [20]:

```
display(image_gen.random_transform(sign_img))

display(image_gen.random_transform(sign_img))
```

SPLITING INTO TRAIN AND VALIDATION DATASET

Train Data Generator

In [22]:

```
train_data_gen = image_gen.flow_from_directory(train_data_path,
                                               target_size=(250,250),
                                               batch_size=16,
                                               shuffle=True,
                                               class_mode='binary',
                                               subset='training')
```

```
Found 41625 images belonging to 37 classes.
```

Validation Data Generator

In [23]:

```
validation_data_gen = image_gen.flow_from_directory(train_data_path,
                                               target_size=(250,250),
                                               batch_size=16,
                                               shuffle=True,
                                               class_mode='binary',
                                               subset='validation')
```

```
Found 13875 images belonging to 37 classes.
```

Test Data Generator

In [30]:

```
test_data_gen = image_gen.flow_from_directory(test_data_path,
                                    target_size=(250,250),
                                    batch_size=8,
                                    shuffle=True,
                                    class_mode='categorical',
                                    )
```

```
Found 2586 images belonging to 37 classes.
```

In [31]:

```
train_data_gen.class_indices
```

SPRINT 2

TEST THE MODEL

In [ ]:

```
!unzip '/content/drive/MyDrive/IBMPROJECT/conversation engine for deaf and dumb.zip'
```

In [1]:

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

In [8]:

```
model = load_model('/content/Real_time.h5')
```

In [9]

```
img = image.load_img('/content/Dataset/test_set/H/107.png',target_size = (100,100))
img
```

```
from skimage.transform import resize
def detect(frame):
    img=image.img_to_array(frame)
    img = resize(img,(64,64,1))
    img = np.expand_dims(img,axis=0)
    pred=np.argmax(model.predict(img))
    op=['A','B','C','D','E','F','G','H','I']
    print("THE PREDICTED LETTER IS ",op[pred])
```

In [11]:

```
img=image.load_img("/content/Dataset/test_set/H/107.png")
detect(img)
```

```
1/1 [==============================] - 0s 412ms/step
THE PREDICTED LETTER IS  H
```

In [12]:

```
img = image.load_img('/content/Dataset/test_set/A/110.png')
pred=detect(img)
```

```
1/1 [==============================] - 0s 23ms/step
THE PREDICTED LETTER IS  A
```

In [14]:

```
img=image.load_img('/content/Dataset/test_set/F/108.png')
detect(img)
```

```
1/1 [==============================] - 0s 25ms/step


THE PREDICTED LETTER IS  F
```

SPRINT 2

```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:

```
#create model
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [10]:

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

In [11]:

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

IMPORTING tensorflow

In [12]:

```
from keras.layers import Dropout
from keras.layers import Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Feature 2

Sprint 3

```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [10]:

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

In [11]:

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

IMPORTING tensorflow

In [12]:

```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:

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

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import cv2
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Image processing

In [ ]:

```
# Create a image


img1 = np.zeros((400,600,3),np.uint8)
```

```
plt.imshow(img1)

# Drawing Functions
```

In [ ]:

```
# Draw a circle

circle = cv2.circle(img1, (300,200), 50, (255,0,0), -1)   # (0,0,0)--->(R,G,B)
plt.imshow(img1) # Drawing rectangle

rectangle = cv2.rectangle(img1,(200,100),(400,300),(0,255,0),6)
plt.imshow(img1)


# Drawing line

line1 = cv2.line(img1,(200,100),(400,300),(0,0,255),4)
line2 = cv2.line(img1,(200,300),(400,100),(0,0,255),4)
plt.imshow(img1)


circle = cv2.circle(img1, (300,200), 50, (255,255,0), -1)   # (0,0,0)--->(R,G,B)
plt.imshow(img1)


# Text on image

text = cv2.putText(img1, 'openCV', (200,50), cv2.FONT_HERSHEY_SIMPLEX, 2,
(255,255,255),5)
plt.imshow(img1)
```

Out[ ]:

```
# Reading the image
```

```
img = cv2.imread('/content/boy.jpg',1)
plt.imshow(img)
```

In [ ]:

```
# Convert BGR to RGB

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

In [ ]:

```
# Convert BGR to Gray

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray)
```

In [ ]:

```
# Finding shape

img_rgb.shape
```

Out[ ]:

```
(983, 736, 3)
```

In [ ]:

```
img_gray.shape
```

Out[ ]:

```
(983, 736)
```

In [ ]:

```
# Resize the image

resize = cv2.resize(img_rgb,(500,1000))
print(resize.shape)
plt.imshow(resize)
```

In [ ]:

```
# Image crop

crop = resize[130:370,150:300]
plt.imshow(crop)
```

In [ ]:

```
# Edge Detection

edge = cv2.Canny(img_rgb,100,200)
plt.imshow(edge)
```

In [ ]:

```
# Blur image

r = resize[130:370,150:300]
blur = cv2.GaussianBlur(r,(13,13),cv2.BORDER_DEFAULT)
plt.imshow(resize)
plt.imshow(blur)
```

TEST THE MODEL

In [ ]:

```
!unzip '/content/drive/MyDrive/IBMPROJECT/conversation engine for deaf and dumb.zip'
```

In [1]:

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

In [8]:

```
model = load_model('/content/Real_time.h5')
```

In [9]:

```
img = image.load_img('/content/Dataset/test_set/H/107.png',target_size = (100,100))
img
```

```
from skimage.transform import resize
def detect(frame):
    img=image.img_to_array(frame)
    img = resize(img,(64,64,1))
    img = np.expand_dims(img,axis=0)
    pred=np.argmax(model.predict(img))
    op=['A','B','C','D','E','F','G','H','I']
    print("THE PREDICTED LETTER IS ",op[pred])
```

In [11]:

```
img=image.load_img("/content/Dataset/test_set/H/107.png")
detect(img)
```

```
1/1 [==============================] - 0s 412ms/step
THE PREDICTED LETTER IS  H
```

In [12]:

```
img = image.load_img('/content/Dataset/test_set/A/110.png')
pred=detect(img)
```

```
1/1 [==============================] - 0s 23ms/step
THE PREDICTED LETTER IS  A
```

In [14]:

```
img=image.load_img('/content/Dataset/test_set/F/108.png')
detect(img)
```

```
1/1 [==============================] - 0s 25ms/step
THE PREDICTED LETTER IS  F
```

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
```

Define DATA FILES

In [ ]:

```
def rename_imgs(file_name):
    folder_path = r'test_dataset/'+file_name


    num = 0
    for file in os.listdir(folder_path):
        # if num%10 == 0:
        #     print(f'Renamed {num} files...')
        # os.rename(folder_path+'\\'+file, folder_path+'\\'+file_name+'_'+str(num)+'.jpeg')
        num += 1
```

In [ ]:

```
fn = 'Space'
rename_imgs(fn)
```

In [ ]:

```
file_names = '0123456789'+'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for fn in file_names:
    rename_imgs(fn)
```

SAMPLE IMAGES FROM DATASET

In [ ]:

```
train_data_path = 'train_dataset/'
test_data_path = 'test_dataset/'
```

In [ ]:

```
def display(img,sign=None):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    fig = plt.figure(figsize=(7,7))
    ax = fig.add_subplot(111)
    plt.title(sign)
    ax.imshow(img)
```

Training Data Set

In [ ]:

```
sign_img = cv2.imread(train_data_path+'A/A_204.jpeg')
display(sign_img,'A')
sign_img = cv2.imread(train_data_path+'3/3_340.jpeg')
display(sign_img,'3')


sign_img = cv2.imread(train_data_path+'S/S_10.jpeg')
display(sign_img,'Space')
```

Test Data Set

In [ ]:

```
sign_img = cv2.imread(test_data_path+'S/S_15.jpeg')
display(sign_img,'S')


sign_img = cv2.imread(test_data_path+'Z/Z_1.jpeg')
display(sign_img,'Z')
```

Image Data Generator

In [ ]:

```
image_gen = ImageDataGenerator(rotation_range=30,
                               width_shift_range=0.1,
                               height_shift_range=0.1,
                               shear_range=0.2,
                               zoom_range=0.2,
                               rescale=1/255,
                               horizontal_flip=True,
                               fill_mode='nearest',
                               validation_split=0.25)
```

Original Image

In [ ]:

```
sign_img = cv2.imread(train_data_path+'3/3_100.jpeg')
display(sign_img,'3')
```

Augmented Images

In [ ]:

```
display(image_gen.random_transform(sign_img))
```

Split into Test & Validation dataset

Train Data Generator

In [ ]:

```
train_data_gen = image_gen.flow_from_directory(train_data_path,
                                      target_size=(250,250),
                                      batch_size=16,
                                      shuffle=True,
                                      class_mode='binary',
                                      subset='training')
```

```
Found 41625 images belonging to 37 classes.
```

Validation Data Generator

In [ ]:

```
validation_data_gen = image_gen.flow_from_directory(train_data_path,
                                      target_size=(250,250),
                                      batch_size=16,
                                      shuffle=True,
                                      class_mode='binary',
                                      subset='validation')
```

```
Found 13875 images belonging to 37 classes.
```

Test Data Generator

In [ ]:

```
test_data_gen = image_gen.flow_from_directory(test_data_path,
                         target_size=(250,250),
                         batch_size=8,
                         shuffle=True,
                         class_mode='categorical',
                         )
```

```
Found 2586 images belonging to 37 classes.
```

In [ ]:

```
train_data_gen.class_indices
```

SPRINT 4

```
import cv2
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
import numpy as np
import tensorflow as tf
import keras
```

```
C:\Users\ryans\Anaconda3\lib\site-packages\numpy\_distributor_init.py:32: UserWarning:
loaded more than 1 DLL from .libs:
C:\Users\ryans\Anaconda3\lib\site-
packages\numpy\.libs\libopenblas.noijjg62emaszi6nyurl6jbkm4evbgm7.gfortran-win_amd64.dll
```

```
C:\Users\ryans\Anaconda3\lib\site-packages\numpy\.libs\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-win_amd64.dll
  stacklevel=1)
```

In [2]:

```
model = keras.models.load_model("asl_classifier.h5")
```

In [3]:

```
labels_dict = {0:'0',
               1:'A',
               2:'B',
               3:'C',
               4:'D',
               5:'E',
               6:'F',
               7:'G',
               8:'H',
               9:'I',
               10:'J',
               11:'K',
               12:'L',
               13:'M',
               14:'N',
               15:'O',
               16:'P',
               17:"Q",
               18:'R',
               19:'S',
               20:'T',
               21:'U',
              22:'V',
              23:'W',
              24:'X',
              25:'Y',
              26:'Z'}
color_dict=(0,255,0)
x=0
y=0
w=64
h=64
```

Fully Real-Time

In [4]:

```
img_size=128
minValue = 70
source=cv2.VideoCapture(0)
count = 0
string = " "
prev = " "
prev_val = 0
while(True):
    ret,img=source.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    #cv2.rectangle(img,(x,y),(x+w,y+h),color_dict,2)
    cv2.rectangle(img,(24,24),(250 , 250),color_dict,2)
    crop_img=gray[24:250,24:250]
crop_img=gray[24:250,24:250]
    count = count + 1
    if(count % 100 == 0):
        prev_val = count
    cv2.putText(img, str(prev_val//100), (300,
150),cv2.FONT_HERSHEY_SIMPLEX,1.5,(255,255,255),2)
    blur = cv2.GaussianBlur(crop_img,(5,5),2)
    th3 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BIN
ARY_INV,11,2)
    ret, res = cv2.threshold(th3, minValue, 255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    resized=cv2.resize(res,(img_size,img_size))
    normalized=resized/255.0
    reshaped=np.reshape(normalized,(1,img_size,img_size,1))
    result = model.predict(reshaped)
    #print(result)
    label=np.argmax(result,axis=1)[0]
    if(count == 300):
        count = 99
        prev= labels_dict[label]
        if(label == 0):
               string = string + " "
            #if(len(string)==1 or string[len(string)] != " "):

        else:
                string = string + prev

    cv2.putText(img, prev, (24, 14),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
    cv2.putText(img, string, (275, 50),cv2.FONT_HERSHEY_SIMPLEX,0.8,(200,200,200),2)
    cv2.imshow("Gray",res)
    cv2.imshow('LIVE',img)
    key=cv2.waitKey(1)


    if(key==27):#press Esc. to exit
        break
print(string)
cv2.destroyAllWindows()
source.release()

cv2.destroyAllWindows()
```

In [8]:

```
# pip install gTTS
```

In [5]:

```
from gtts import gTTS

# This module is imported so that we can
# play the converted audio
import os

# The text that you want to convert to audio


# Language in which you want to convert
language = 'en'
# Passing the text and language to the engine,
# here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=string, lang=language, slow=False)
```

```
# Saving the converted audio in a mp3 file named
# welcome
myobj.save("welcome2121.mp3")

# Playing the converted file
os.system("welcome.mp3")
```

In [6]:

```
from playsound import playsound
playsound('welcome2121.mp3')
```

In [ ]:

```
import cv2,os

data_path='DATASET'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]

label_dict=dict(zip(categories,labels)) #empty dictionary

print(label_dict)
print(categories)
print(labels)
```

```
C:\Users\ryans\Anaconda3\lib\site-packages\numpy\_distributor_init.py:32: UserWarning:
loaded more than 1 DLL from .libs:
C:\Users\ryans\Anaconda3\lib\site-
packages\numpy\.libs\libopenblas.NOIJJG62EMASZI6NYURL6JBKM4EVBGM7.gfortran-
win_amd64.dll
C:\Users\ryans\Anaconda3\lib\site-
packages\numpy\.libs\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-
```

```
win_amd64.dll
  stacklevel=1)
{'test': 0, 'train': 1}
['test', 'train']
[0, 1]
```

In [5]:

```
data_path='DATASET/train'
classes_path=os.listdir(data_path)
classesf=os.listdir(data_path)
print(classesf)
labels_classes=[i for i in range(len(classesf))]
print(labels_classes)
```

```
['0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
'Y', 'Z']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]
```

In [81]:

```
data_path='DATASET'
```

```
label_classes_dict=dict(zip(classesf,labels_classes))
```

In [76]:

```
#print(labels_classes)
#print(categories)
print(label_classes_dict)
```

```
{'0': 0, 'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7, 'H': 8, 'I': 9, 'J': 10, 'K': 11, 'L': 12, 'M': 13,
'N': 14, 'O': 15, 'P': 16, 'Q': 17, 'R': 18, 'S': 19, 'T': 20, 'U': 21, 'V': 22, 'W': 23, 'X': 24, 'Y': 25, 'Z':
26}
```

In [77]:

```
import numpy as np
```

In [ ]:

In [82]:

```
img_size=128
data=[]
target=[]
c=0
minValue = 70
for category in categories:

    cat_path=os.path.join(data_path,category)
    print(cat_path)
    cat_names=os.listdir(cat_path)
    print(cat_names)
    for classes in cat_names:
        folder_path=os.path.join(data_path,category,classes)
        print(folder_path)
        img_names=os.listdir(folder_path)
        #print(img_names)
        for img_name in img_names:
folder_path=os.path.join(data_path,category,classes)
        print(folder_path)
        img_names=os.listdir(folder_path)
        #print(img_names)
        for img_name in img_names:
            #print(img_name)
            img_path=os.path.join(folder_path,img_name)
            img=cv2.imread(img_path)

            try:
```

```
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray,(5,5),2)
            th3 = 
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
            ret, res = cv2.threshold(th3, minValue, 255, 
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
            #res=np.array(res)
            #print(type(res))
          #Converting the image into gray scale
            resized=cv2.resize(res,(img_size,img_size))
          #resizing the gray scale into 50x50, since we need a fixed common

     datanp=np.array(data)
                                                                    In [84]:
datanp.shape
                                                                    Out[84]:
(17113, 128, 128)
                                                                    In [85]:
targetnp=np.array(target)

targetnp.shape
                                                                    Out[85]:
(17113,)
                                                                    In [86]:
import numpy as np

data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
```

```
target=np.array(target)

from keras.utils import np_utils

new_target=np_utils.to_categorical(target)
```

In [87]:

```
new_target.shape
```

Out[87]:

```
(17113, 27)
```

In [ ]

```
import numpy as np

data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
target=np.array(target)

from keras.utils import np_utils

new_target=np_utils.to_categorical(target)
```

In [87]:

```
new_target.shape
```

Out[87]:

```
(17113, 27)
```

In [ ]:

In [88]:

```
np.save('data_img',data)
np.save('target',new_target)
```

In [89]:

```
data=np.load('data_img.npy')
target=np.load('target.npy')
```

In [90]:

```
from sklearn.model_selection import train_test_split
train_data,test_data,train_target,test_target=train_test_split(data,new_target,test_size=0.2)
```

In [91]:

```
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Flatten
from keras.layers import Dense , Dropout
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "1"
sz = 128
# Step 1 - Building the CNN

# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(sz, sz, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
```

```
classifier.add(MaxPooling2D(pool_size=(2, 2)))
#classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
#classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=96, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=64, activation='relu'))
classifier.add(Dense(units=27, activation='softmax')) # softmax for more than 2

# Compiling the CNN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) # categorical_crossentropy for more than 2



# Step 2 - Preparing the train/test data and training the model
classifier.summary()
```

```
Model: "sequential_2"
```

In [92]:

```
from keras.callbacks import ModelCheckpoint
```

In [93]:

```
checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
```

```
history=classifier.fit(train_data,train_target,shuffle=True,epochs=20,callbacks=[checkpoint],validation_split=0.3)
Train on 9583 samples, validate on 4107 samples
ccuracy: 0.8519 - val_loss: 0.0425 - val_accuracy: 0.9905
print(classifier.evaluate(test_data,test_target))
N = 20
H=history
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('evaluation.png')
# serialize the model to disk
print("[INFO] saving mask detector model...")
classifier.save('asl_classifier.h5')
print("Done !")
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epochs')
plt.ylabel('Loss')
plt.legend(['train_loss','val_loss'], loc=0)
plt.show()
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('epochs')

plt.ylabel('Accuracy')
plt.legend(['train_accuracy','Val accuracy'], loc=0)
plt.show()
```

8. TESTING

8.1 Test cases:

This report shows the number of test cases that have passed, failed and untested.

Section	TotalCases	Not	Fail	Pass
● REGISTRATION/LOGIN TESTING	1	0	0	Pass
● CAMERA AND MICROPHONE TESTING	1	0	0	Pass

8.2 User Acceptance:

1. The Purpose of Document The purpose of this document is to briefly explain the test coverage and open issues of the [REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis :

This report execute our user scheduling and their approaches.

Task	Severity 1	Severity 2	Severity 3	Severity 4	Sub total
Login	5	1	2	4	12
Home page	4	1	7	5	17
Model building	1	0	3	0	4
Execute the model	1	0	0	1	2
Flask (app.py)	1	2	2	2	7
Flask (IBM app.py)	0	0	1	0	1
Deploying the model	0	0	1	1	2
Totals	12	4	16	13	45

3. Test Case Analysis:

This report show the number of test cases that have passed, Failed , untested

Section	Total cases	Not tested	Fail	Pass
REGISTERATION/LOGIN TESTING	1	0	0	Pass
CAMERA AND MICROPHONE TESTING	1	0	0	pass

9. RESULTS

9.1 Performance Metrics:

Model Performance Testing:

Project team shall fill the followinginformation in modelperformance testing template.

S.NO	Paramater	Values	Screenshot
●	Model Summary	**10**	file:///C:/Users/sanjay%20hananth/Videos/Captures/Webcam%20 Tes t%20-%20Google%20Chrome%2016-11-2022%2022 28 24%20(2).png
●	Accuracy	Training Accuracy 10 Validation Accuracy 10	file:///C:/Users/sanjay%20hananth/Videos/Captures/Webcam%20Test%20-%20Google%20Chrome%2016-11-2022%2022 28 24%20(2).png
●	Confidence Score (Only Yolo Projects)	Class Detected - 10 Confidence Score 10	file:///C:/Users/sanjay%20hananth/Videos/Captures/Webcam%20Test%20-%20Google%20Chrome%2016-11-2022%2022 28 24%20(2).png

10. ADVANTAGES & DISADVANTAGE

ADVANTAGES:

1.It defines a more powerful and more useful computer

2.It introduces a new and improved interface for human interaction

3.It introduces a new technique to solve new problems

4.It is very handles the information better than humans.

5.It is very helpful for the conversion of information into knowledge.

DISADVANTAGES:

1.The implementation cost of AI is very high.

2. The difficulties with software development for AI implementation are that the development of software is slow and expensive.Few efficient programmers are available to implement artificial intelligence.

3.A robot is one of the implementations of artificial intelligence with them replacing jobs and lead to serve unemployment.

4. Machine can easily lead to destruction if the implementation machine put in the wrong hands the results are hazardous for human begins.

11. CONCLUSION

we are making use of a convolution neural network to create a model that is trained on different hand gestures. An app is built which uses this model. This app enables deaf and dumb people to convey their information using sings which get converted to human-understandable language and speech is given as output.

12. FUTURE SCOPE

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, as well as convert speech into understandable sign language for the deaf and dumb. We are making use of a convolution neural network to create a model that is trained on different hand gestures. An app is built which uses this model. This app enables deaf and dumb people to convey their information using signs which get converted to human-understandable language and speech is given as output.

13. APPENDIX:

SOURCE CODE:

ASL_Real-Time.ipynb

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
                "C:\\Users\\ryans\\Anaconda3\\lib\\site-packages\\numpy\\_distributor_init.py:32:
```

```
UserWarning: loaded more than 1 DLL from .libs:\n",
      "C:\\Users\\ryans\\Anaconda3\\lib\\site-packages\\numpy\\.libs\\libopenblas.noijjg62emaszi6nyurl6jbkm4evbgm7.gfortran-win_amd64.dll\n",
      "C:\\Users\\ryans\\Anaconda3\\lib\\site-packages\\numpy\\.libs\\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-win_amd64.dll\n",
      "  stacklevel=1)\n"
     ]
    }
   ],
   "source": [
    "import cv2\n",
    "from keras.models import load_model\n",
    "from keras.preprocessing.image import load_img, img_to_array\n",
    "import numpy as np\n",
    "import tensorflow as tf\n",
    "import keras"
   ]
  },
  {
```

```
  "cell_type": "code",
  "execution_count": 2,
  "metadata": {},
  "outputs": [],
  "source": [
   "model = keras.models.load_model(\"asl_classifier.h5\")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 3,
  "metadata": {},
  "outputs": [],
  "source": [
   "labels_dict = {0:'0', \n",
   "              1:'A', \n",
   "              2:'B', \n",
   "              3:'C', \n",
   "              4:'D', \n",
   "              5:'E',\n",
"            6:'F',\n",
"            7:'G',\n",
"            8:'H',\n",
"            9:'I',\n",
"            10:'J',\n",
"            11:'K',\n",
"            12:'L',\n",
"            13:'M',\n",
"            14:'N',\n",
"            15:'O',\n",
"            16:'P',\n",
"            17:\"Q\",\n",
"            18:'R',\n",
"            19:'S',\n",
"            20:'T', \n",
"            21:'U', \n",
"            22:'V',\n",
"            23:'W',\n",
"            24:'X',\n",
"            25:'Y',\n",
   "                    26:'Z'}\n",
   "color_dict=(0,255,0)\n",
   "x=0\n",
   "y=0\n",
   "w=64\n",
   "h=64"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "# Fully Real-Time"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 4,
  "metadata": {},
  "outputs": [],
"source": [
 "img_size=128\n",
 "minValue = 70\n",
 "source=cv2.VideoCapture(0)\n",
 "count = 0\n",
 "string = \" \"\n",
 "prev = \" \"\n",
 "prev_val = 0\n",
 "while(True):\n",
 "    ret,img=source.read()\n",
 "    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)\n",
 "    #cv2.rectangle(img,(x,y),(x+w,y+h),color_dict,2)\n",
 "    cv2.rectangle(img,(24,24),(250 , 250),color_dict,2)\n",
 "    crop_img=gray[24:250,24:250]\n",
 "    count = count + 1\n",
 "    if(count % 100 == 0):\n",
 "        prev_val = count\n",
 "    cv2.putText(img, str(prev_val//100), (300, 150),cv2.FONT_HERSHEY_SIMPLEX,1.5,(255,255,255),2) \n",
 "    blur = cv2.GaussianBlur(crop_img,(5,5),2)\n",
 "    th3 =
 cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)\n",
 "    ret, res = cv2.threshold(th3, minValue, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)\n",
 "    resized=cv2.resize(res,(img_size,img_size))\n",
 "    normalized=resized/255.0\n",
 "    reshaped=np.reshape(normalized,(1,img_size,img_size,1))\n",
 "    result = model.predict(reshaped)\n",
 "    #print(result)\n",
 "    label=np.argmax(result,axis=1)[0]\n",
 "    if(count == 300):\n",
 "        count = 99\n",
 "        prev= labels_dict[label] \n",
 "        if(label == 0):\n",
 "               string = string + \" \"\n",
 "           #if(len(string)==1 or string[len(string)] != \" \"):\n",
 "            \n",
 "        else:\n",
 "                string = string + prev\n",
 "    \n",
 "    cv2.putText(img, prev, (24, 14),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
```

```
\n",
                                   "                              cv2.putText(img,        string,        (275,
50),cv2.FONT_HERSHEY_SIMPLEX,0.8,(200,200,200),2)\n",
    "    cv2.imshow(\"Gray\",res)    \n",
    "    cv2.imshow('LIVE',img)\n",
    "    key=cv2.waitKey(1)\n",
    "    \n",
    " \n",
    "    if(key==27):#press Esc. to exit\n",
    "        break\n",
    "print(string)        \n",
    "cv2.destroyAllWindows()\n",
    "source.release()\n",
    "\n",
    "cv2.destroyAllWindows()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
 "metadata": {},
 "outputs": [],
 "source": [
  "# pip install gTTS"
 ]
},
{
 "cell_type": "code",
 "execution_count": 5,
 "metadata": {},
 "outputs": [],
 "source": [
  "from gtts import gTTS \n",
  "  \n",
  "# This module is imported so that we can  \n",
  "# play the converted audio \n",
  "import os \n",
  "  \n",
  "# The text that you want to convert to audio \n",
  "  \n",
   "# Language in which you want to convert \n",
   "language = 'en'\n",
   "# Passing the text and language to the engine,  \n",
   "# here we have marked slow=False. Which tells  \n",
   "# the module that the converted audio should  \n",
   "# have a high speed \n",
   "myobj = gTTS(text=string, lang=language, slow=False) \n",
   "  \n",
   "# Saving the converted audio in a mp3 file named \n",
   "# welcome  \n",
   "myobj.save(\"welcome2121.mp3\") \n",
   "  \n",
   "# Playing the converted file \n",
   "os.system(\"welcome.mp3\") "
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 6,
  "metadata": {},
   "outputs": [],
   "source": [
    "from playsound import playsound\n",
    "playsound('welcome2121.mp3')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.7.4"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 4
}
```

ASL_train.ipynb:

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "C:\\Users\\ryans\\Anaconda3\\lib\\site-packages\\numpy\\_distributor_init.py:32: UserWarning: loaded more than 1 DLL from .libs:\n",
      "C:\\Users\\ryans\\Anaconda3\\lib\\site-packages\\numpy\\.libs\\libopenblas.NOIJJG62EMASZI6NYURL6JBKM4EVBGM7.gfortran-win_amd64.dll\n",
      "C:\\Users\\ryans\\Anaconda3\\lib\\site-packages\\numpy\\.libs\\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-win_amd64.dll\n",
      "  stacklevel=1)\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "{'test': 0, 'train': 1}\n",
      "['test', 'train']\n",
      "[0, 1]\n"
     ]
    }
   ],
   "source": [
    "import cv2,os\n",
    "\n",
    "data_path='DATASET'\n",
    "categories=os.listdir(data_path)\n",
    "labels=[i for i in range(len(categories))]\n",
    "\n",
    "label_dict=dict(zip(categories,labels)) #empty dictionary\n",
    "\n",
    "print(label_dict)\n",
    "print(categories)\n",
    "print(labels)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "['0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']\n",
      "[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]\n"
     ]
    }
   ],
   "source": [
   "data_path='DATASET/train'\n",
   "classes_path=os.listdir(data_path)\n",
   "classesf=os.listdir(data_path)\n",
   "print(classesf)\n",
   "labels_classes=[i for i in range(len(classesf))]\n",
   "print(labels_classes)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 81,
  "metadata": {},
  "outputs": [],
  "source": [
   "data_path='DATASET'"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 75,
  "metadata": {},
  "outputs": [],
  "source": [
   "label_classes_dict=dict(zip(classesf,labels_classes))"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 76,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "{'0': 0, 'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7, 'H': 8, 'I': 9, 'J': 10, 'K': 11, 'L': 12, 'M': 13, 'N': 14, 'O': 15, 'P': 16, 'Q': 17, 'R': 18, 'S': 19, 'T': 20, 'U': 21, 'V': 22, 'W': 23, 'X': 24, 'Y': 25, 'Z': 26}\n"
     ]
    }
   ],
   "source": [
    "#print(labels_classes)\n",
    "#print(categories)\n",
    "print(label_classes_dict)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 77,
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": 82,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "DATASET\\test\n",
      "['0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']\n",
      "DATASET\\test\\0\n",
      "DATASET\\test\\A\n",
      "DATASET\\test\\B\n",
      "DATASET\\test\\C\n",
      "DATASET\\test\\D\n",
      "DATASET\\test\\E\n",
      "DATASET\\test\\F\n",
      "DATASET\\test\\G\n",
      "DATASET\\test\\H\n",
      "DATASET\\test\\I\n",
      "DATASET\\test\\J\n",
      "DATASET\\test\\K\n",
      "DATASET\\test\\L\n",
    "DATASET\\test\\M\n",
    "DATASET\\test\\N\n",
    "DATASET\\test\\O\n",
    "DATASET\\test\\P\n",
    "DATASET\\test\\Q\n",
    "DATASET\\test\\R\n",
    "DATASET\\test\\S\n",
    "DATASET\\test\\T\n",
    "DATASET\\test\\U\n",
    "DATASET\\test\\V\n",
    "DATASET\\test\\W\n",
    "DATASET\\test\\X\n",
    "DATASET\\test\\Y\n",
    "DATASET\\test\\Z\n",
    "DATASET\\train\n",
    "['0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
'Z']\n",
    "DATASET\\train\\0\n",
    "DATASET\\train\\A\n",
    "DATASET\\train\\B\n",
    "DATASET\\train\\C\n",
    "DATASET\\train\\D\n",
    "DATASET\\train\\E\n",
    "DATASET\\train\\F\n",
    "DATASET\\train\\G\n",
    "DATASET\\train\\H\n",
    "DATASET\\train\\I\n",
    "DATASET\\train\\J\n",
    "DATASET\\train\\K\n",
   "DATASET\\train\\L\n",
   "DATASET\\train\\M\n",
   "DATASET\\train\\N\n",
   "DATASET\\train\\O\n",
   "DATASET\\train\\P\n",
   "DATASET\\train\\Q\n",
   "DATASET\\train\\R\n",
   "DATASET\\train\\S\n",
   "DATASET\\train\\T\n",
   "DATASET\\train\\U\n",
   "DATASET\\train\\V\n",
   "DATASET\\train\\W\n",
   "DATASET\\train\\X\n",
   "DATASET\\train\\Y\n",
   "DATASET\\train\\Z\n"
  ]
 }
],
"source": [
 "img_size=128\n",
 "data=[]\n",
 "target=[]\n",
 "c=0\n",
 "minValue = 70\n",
 "for category in categories:\n",
 "    \n",
 "    cat_path=os.path.join(data_path,category)\n",
 "    print(cat_path)\n",
 "    cat_names=os.listdir(cat_path)\n",
"    print(cat_names)\n",
"    for classes in cat_names:\n",
"        folder_path=os.path.join(data_path,category,classes)\n",
"        print(folder_path)\n",
"        img_names=os.listdir(folder_path)\n",
"        #print(img_names)\n",
"        for img_name in img_names:\n",
"            #print(img_name)\n",
"            img_path=os.path.join(folder_path,img_name)\n",
"            img=cv2.imread(img_path)\n",
"            \n",
"            try:\n",
"                gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) \n",
"                blur = cv2.GaussianBlur(gray,(5,5),2)\n",
"                th3 = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)\n",
"                ret, res = cv2.threshold(th3, minValue, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)\n",
"                #res=np.array(res)\n",
"                #print(type(res))\n",
"            #Converting the image into gray scale\n",
"                resized=cv2.resize(res,(img_size,img_size))\n",
"            #resizing the gray scale into 50x50, since we need a fixed common size for all the images in the dataset\n",
"                data.append(resized)\n",
"                #print(data)\n",
"                target.append(label_classes_dict[classes])\n",
"            except Exception as e:\n",
"                print('Exception:',e)\n",
"            \n",
    "         \n",
    "     \n",
    "     "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 83,
   "metadata": {},
   "outputs": [],
   "source": [
    "datanp=np.array(data)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 84,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(17113, 128, 128)"
      ]
     },
     "execution_count": 84,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "datanp.shape"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 85,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(17113,)"
      ]
     },
     "execution_count": 85,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "targetnp=np.array(target)\n",
    "\n",
    "targetnp.shape"
   ]
  },
  {
   "cell_type": "code",
 "execution_count": 86,
 "metadata": {},
 "outputs": [],
 "source": [
  "import numpy as np\n",
  "\n",
  "data=np.array(data)/255.0\n",
  "data=np.reshape(data,(data.shape[0],img_size,img_size,1))\n",
  "target=np.array(target)\n",
  "\n",
  "from keras.utils import np_utils\n",
  "\n",
  "new_target=np_utils.to_categorical(target)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 87,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(17113, 27)"
    ]
   },
   "execution_count": 87,
   "metadata": {},
   "output_type": "execute_result"
```

```
    }
   ],
   "source": [
    "new_target.shape"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": 88,
   "metadata": {},
   "outputs": [],
   "source": [
    "np.save('data_img',data)\n",
    "np.save('target',new_target)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 89,
   "metadata": {},
   "outputs": [],
   "source": [
    "data=np.load('data_img.npy')\n",
    "target=np.load('target.npy')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 90,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.model_selection import train_test_split\n",
    "train_data,test_data,train_target,test_target=train_test_split(data,new_target,test_size=0.2)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 91,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Model: \"sequential_2\"\n",
      "_________________________________________________________________\n",
      "Layer (type)                 Output Shape              Param #   \n",
      "=================================================================\n",
      "conv2d_3 (Conv2D)            (None, 126, 126, 32)      320       \n",
      "_________________________________________________________________\n",
    "max_pooling2d_3 (MaxPooling2 (None, 63, 63, 32)       0         \n",
    "_________________________________________________________________\n",
    "conv2d_4 (Conv2D)            (None, 61, 61, 32)       9248      \n",
    "_________________________________________________________________\n",
    "max_pooling2d_4 (MaxPooling2 (None, 30, 30, 32)       0         \n",
    "_________________________________________________________________\n",
    "flatten_2 (Flatten)          (None, 28800)             0         \n",
    "_________________________________________________________________\n",
    "dense_5 (Dense)              (None, 128)               3686528   \n",
    "_________________________________________________________________\n",
    "dropout_3 (Dropout)          (None, 128)               0         \n",
    "_________________________________________________________________\n",
    "dense_6 (Dense)              (None, 96)                12384     \n",
    "_________________________________________________________________\n",
    "dropout_4 (Dropout)          (None, 96)                0         \n",
    "_________________________________________________________________\n",
    "dense_7 (Dense)              (None, 64)                6208      \n",
    "_________________________________________________________________\n",
    "dense_8 (Dense)              (None, 27)                1755      \n",
    "=================================================================\n",
    "Total params: 3,716,443\n",
    "Trainable params: 3,716,443\n",
    "Non-trainable params: 0\n",
    "_________________________________________________________________\n"
   ]
  }
 ],
 "source": [
  "from keras.models import Sequential\n",
"from keras.layers import Convolution2D\n",
"from keras.layers import MaxPooling2D\n",
"from keras.layers import Flatten\n",
"from keras.layers import Dense , Dropout\n",
"import os\n",
"os.environ[\"CUDA_VISIBLE_DEVICES\"] = \"1\"\n",
"sz = 128\
"# Step 1 - Building the CNN\n",
"\n",
"# Initializing the CNN\n",
"classifier = Sequential()\n",
"\n",
"# First convolution layer and pooling\n",
"classifier.add(Convolution2D(32, (3, 3), input_shape=(sz, sz, 1), activation='relu'))\n",
"classifier.add(MaxPooling2D(pool_size=(2, 2)))\n",
"# Second convolution layer and pooling\n",
"classifier.add(Convolution2D(32, (3, 3), activation='relu'))\n",
"# input_shape is going to be the pooled feature maps from the previous convolution layer\n",
"classifier.add(MaxPooling2D(pool_size=(2, 2)))\n",
"#classifier.add(Convolution2D(32, (3, 3), activation='relu'))\n",
"# input_shape is going to be the pooled feature maps from the previous convolution layer\n",
"#classifier.add(MaxPooling2D(pool_size=(2, 2)))\n",
"\n",
"# Flattening the layers\n",
"classifier.add(Flatten())\n",
"\n",
"# Adding a fully connected layer\n",
"classifier.add(Dense(units=128, activation='relu'))\n",
"classifier.add(Dropout(0.40))\n",
    "classifier.add(Dense(units=96, activation='relu'))\n",
    "classifier.add(Dropout(0.40))\n",
    "classifier.add(Dense(units=64, activation='relu'))\n",
    "classifier.add(Dense(units=27, activation='softmax')) # softmax for more than 2\n",
    "\n",
    "# Compiling the CNN\n",
    "classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) # categorical_crossentropy for more than 2\n",
    "\n",
    "\n",
    "# Step 2 - Preparing the train/test data and training the model\n",
    "classifier.summary()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 92,
   "metadata": {},
   "outputs": [],
   "source": [
    "from keras.callbacks import ModelCheckpoint"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 93,
   "metadata": {},
   "outputs": [
    {
```

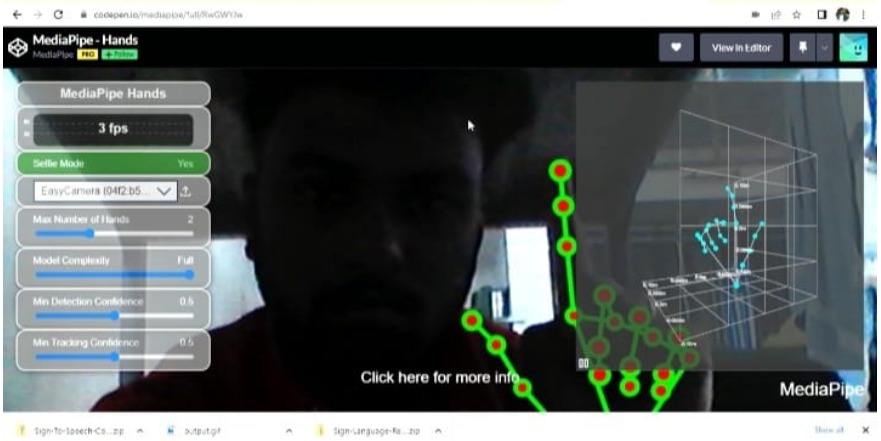
```
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Train on 9583 samples, validate on 4107 samples\n",
     "Epoch 1/20\n",
     "9583/9583 [==============================] - 108s 11ms/step - loss: 2.3023 - accuracy: 0.3101
- val_loss: 0.5760 - val_accuracy: 0.8795\n",
     "Epoch 2/20\n",
     "9583/9583 [==============================] - 98s 10ms/step - loss: 0.7736 - accuracy: 0.7359 -
val_loss: 0.1293 - val_accuracy: 0.9827\n",
     "Epoch 3/20\n",
     "9583/9583 [==============================] - 96s 10ms/step - loss: 0.4438 - accuracy: 0.8519 -
val_loss: 0.0425 - val_accuracy: 0.9905\n",
     "Epoch 4/20\n",
     "9583/9583 [==============================] - 90s 9ms/step - loss: 0.3134 - accuracy: 0.8954 -
val_loss: 0.0172 - val_accuracy: 0.9971\n",
     "Epoch 5/20\n",
     "9583/9583 [==============================] - 91s 10ms/step - loss: 0.2613 - accuracy: 0.9138 -
val_loss: 0.0131 - val_accuracy: 0.9973\n",
     "Epoch 6/20\n",
     "9583/9583 [==============================] - 91s 9ms/step - loss: 0.2097 - accuracy: 0.9323 -
val_loss: 0.0155 - val_accuracy: 0.9966\n",
     "Epoch 7/20\n",
     "9583/9583 [==============================] - 90s 9ms/step - loss: 0.1818 - accuracy: 0.9375 -
val_loss: 0.0070 - val_accuracy: 0.9985\n",
     "Epoch 8/20\n",
     "9583/9583 [==============================] - 91s 9ms/step - loss: 0.1720 - accuracy: 0.9469 -
val_loss: 0.0062 - val_accuracy: 0.9985\n",
     "Epoch 9/20\n",
     "9583/9583 [==============================] - 91s 9ms/step - loss: 0.1535 - accuracy: 0.9506 -
val_loss: 0.0093 - val_accuracy: 0.9978\n",
     "Epoch 10/20\n",
    "9583/9583 [==============================] - 91s 10ms/step - loss: 0.1428 - accuracy: 0.9562 - val_loss: 0.0064 - val_accuracy: 0.9983\n",
    "Epoch 11/20\n",
    "9583/9583 [==============================] - 94s 10ms/step - loss: 0.1251 - accuracy: 0.9590 - val_loss: 0.0059 - val_accuracy: 0.9985\n",
    "Epoch 12/20\n",
    "9583/9583 [==============================] - 90s 9ms/step - loss: 0.1283 - accuracy: 0.9624 - val_loss: 0.0048 - val_accuracy: 0.9993\n",
    "Epoch 13/20\n",
    "9583/9583 [==============================] - 91s 9ms/step - loss: 0.1034 - accuracy: 0.9691 - val_loss: 0.0087 - val_accuracy: 0.9976\n",
    "Epoch 14/20\n",
    "9583/9583 [==============================] - 91s 10ms/step - loss: 0.1145 - accuracy: 0.9663 - val_loss: 0.0054 - val_accuracy: 0.9985\n",
    "Epoch 15/20\n",
    "9583/9583 [==============================] - 92s 10ms/step - loss: 0.1025 - accuracy: 0.9686 - val_loss: 0.0066 - val_accuracy: 0.9983\n",
    "Epoch 16/20\n",
    "9583/9583 [==============================] - 90s 9ms/step - loss: 0.0927 - accuracy: 0.9713 - val_loss: 0.0036 - val_accuracy: 0.9993\n",
    "Epoch 17/20\n",
    "9583/9583 [==============================] - 90s 9ms/step - loss: 0.0880 - accuracy: 0.9723 - val_loss: 0.0054 - val_accuracy: 0.9988\n",
    "Epoch 18/20\n",
    "9583/9583 [==============================] - 101s 11ms/step - loss: 0.0864 - accuracy: 0.9729 - val_loss: 0.0042 - val_accuracy: 0.9988\n",
    "Epoch 19/20\n",
    "9583/9583 [==============================] - 94s 10ms/step - loss: 0.0839 - accuracy: 0.9750 - val_loss: 0.0040 - val_accuracy: 0.9988\n",
    "Epoch 20/20\n",
    "9583/9583 [==============================] - 93s 10ms/step - loss: 0.0817 - accuracy: 0.9745 - val_loss: 0.0038 - val_accuracy: 0.9988\n"
   ]
```

```
    }
   ],
   "source": [
    "checkpoint = ModelCheckpoint('model-
{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')\n",

"history=classifier.fit(train_data,train_target,shuffle=True,epochs=20,callbacks=[checkpoint],validation_s
plit=0.3)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 95,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "3423/3423 [==============================] - 8s 2ms/step\n",
      "[0.006450053481155843, 0.9985392689704895]\n"
     ]
    }
   ],
   "source": [
    "print(classifier.evaluate(test_data,test_target))"
   ]
  },
  {
  nw+/3Nzhsz99YCHF8mTBhAqeeemq7b7fZpJCZmcmMGTOYMWMGO3bs4JNPPuG5556jqqqK119
  "plt.savefig('evaluation.png')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 104,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "[INFO] saving mask detector model...\n",
    "Done !\n"
   ]
  }
 ],
 "source": [
  "# serialize the model to disk\n",
  "print(\"[INFO] saving mask detector model...\")\n",
  "classifier.save('asl_classifier.h5')\n",
  "print(\"Done !\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 98,
 "metadata": {},
 "outputs": [
   {
    "data": {
     "image/png":

      "<Figure size 432x288 with 1 Axes>"
     ]
    },
    "metadata": {
     "needs_background": "light"
    },
    "output_type": "display_data"
   }
  ],
  "source": [
   "import matplotlib.pyplot as plt\n",
   "plt.plot(history.history['loss'])\n",
   "plt.plot(history.history['val_loss'])\n",
   "plt.xlabel('epochs')\n",
   "plt.ylabel('Loss')\n",
   "plt.legend(['train_loss','val_loss'], loc=0)\n",
   "plt.show()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 105,
  "metadata": {},
  "outputs": [
   {
     "data": {
      "text/plain": [
       "<Figure size 432x288 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "import matplotlib.pyplot as plt\n",
    "plt.plot(history.history['accuracy'])\n",
    "plt.plot(history.history['val_accuracy'])\n",
    "plt.xlabel('epochs')\n",
    "plt.ylabel('Accuracy')\n",
    "plt.legend(['train_accuracy','val_accuracy'], loc=0)\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.7.4"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```

SCREEN SHOT



MediaPipe Hands

3 fps

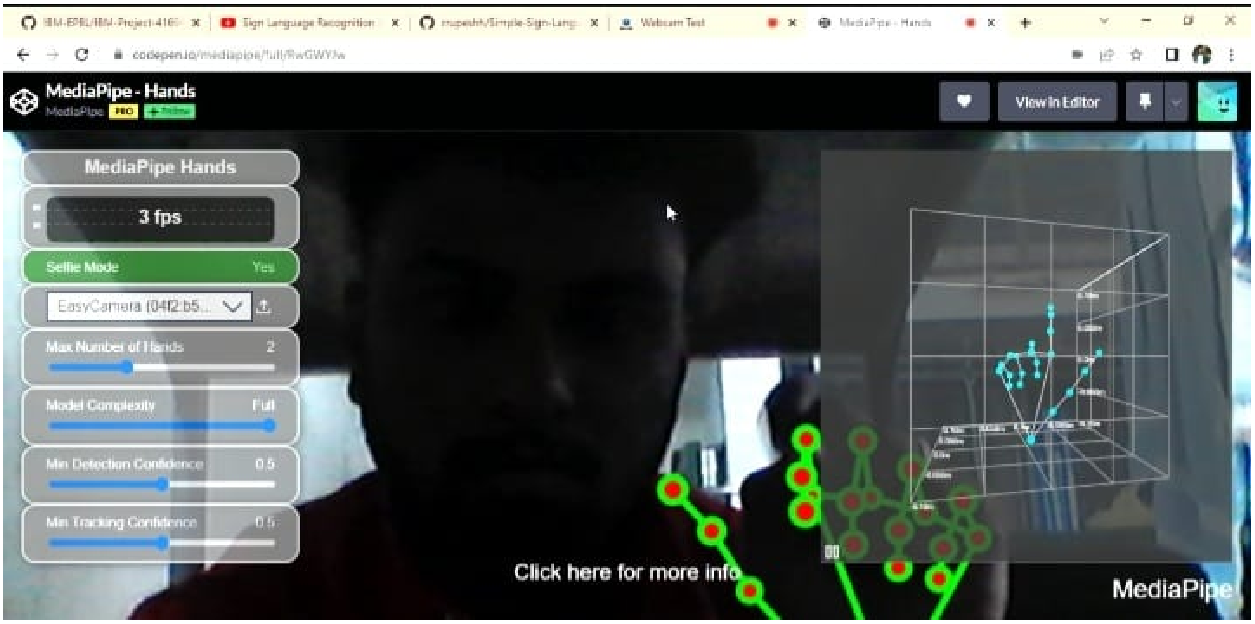
Selfie Mode Yes

Max Number of Hands 2

Model Complexity Full

Min Detection Confidence 0.5

Min Tracking Confidence 0.5



MediaPipe Hands

3 fps

Selfie Mode Yes

EasyCamera (04f2:b5...

Max Number of Hands 2

Model Complexity Full

Min Detection Confidence 0.5

Min Tracking Confidence 0.5

Appendix:

GITHUB: https://github.com/IBM-EPBL/IBM-Project-41694-1660644237

DEMOLINK:

https://drive.google.com/file/d/1-ohQkgc4d0Qt1wLnpEHtgxqzq7-ObgWT/view