VEL TECH HIGH TECH Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE

Real-Time Communication System Powered by AI for Specially Abled

TEAM ID:PNT2022TMID22136

TEAM LEAD: MOHAMED NASSAAR.N.V.S(113019106043)

TEAM MEMBER1: MOHAMED ANAS.S(113019106041)

TEAM MEMBER2: KARTHICK.L(113019106030)

TEAM MEMBER3: DEVA.V(113019106014)

Faculty Mentor Name: Mrs. S.Jalaja

Introduction

1.1 Project Overview

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

1.2 Purpose

The project aims to develop a system that converts the sign language into a human hearing voice in the desired language to convey a message to normal people, 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.

2. LITERATURE SURVEY

2.1 Existing problem

Technology

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

Interpreter

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

Just Speaking

Depending on the deaf person's level of hearing loss, they may be able to communicate with a blind person who is using speech. For example, a deaf person may have enough residual hearing (with or without the use of an assistive hearing device such as a hearing aid) to be able to decipher the speech of the person who is blind or has low vision. However, this is often not the most effective form of communication, as it is very

dependent on the individual circumstances of both people and their environment (for example, some places may have too much background noise).

2.2 References

This paper describes the system that overcomes the problem faced by the speech and hearing impaired. The objectives of the research are as follow: 1. To design and develop a system which lowers the communication gap between speech hearing impaired and normal world.

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

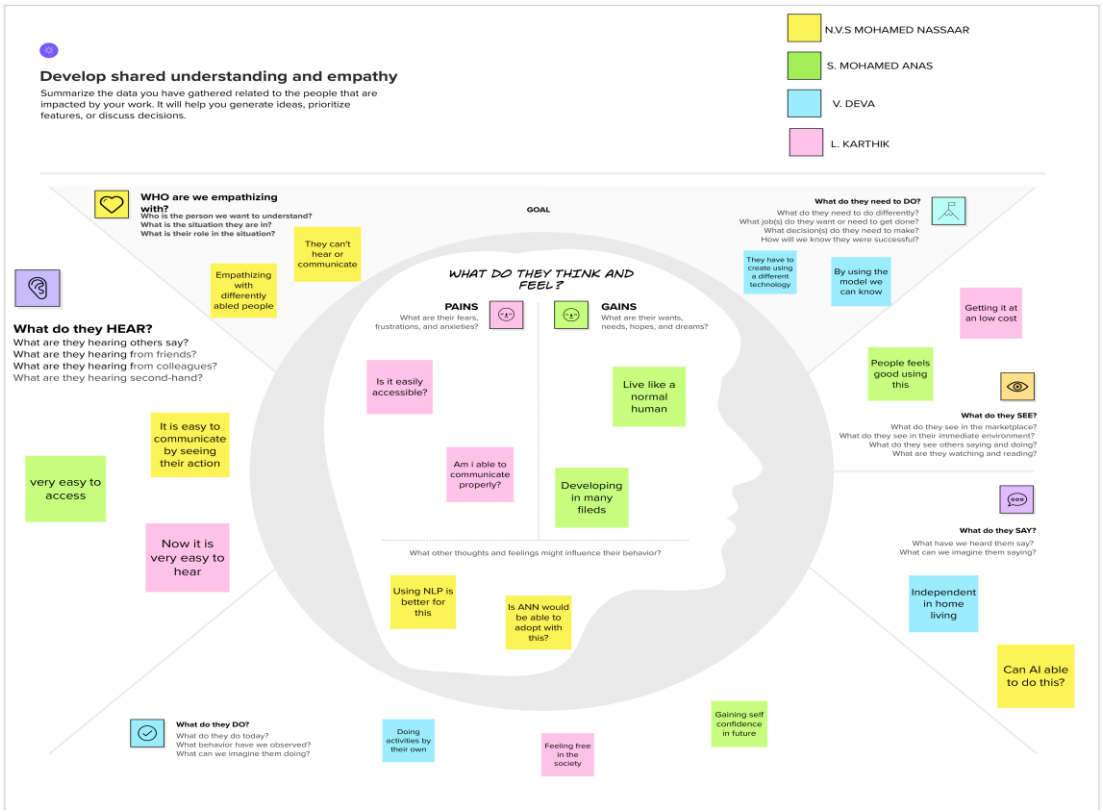
3. A convolution neural network is being used to develop a model that is trained on various hand movements. This model is used to create an app. This programme allows deaf and hard of hearing persons to communicate using signs that are then translated into human readable text.

2.3 Problem Statement Definition

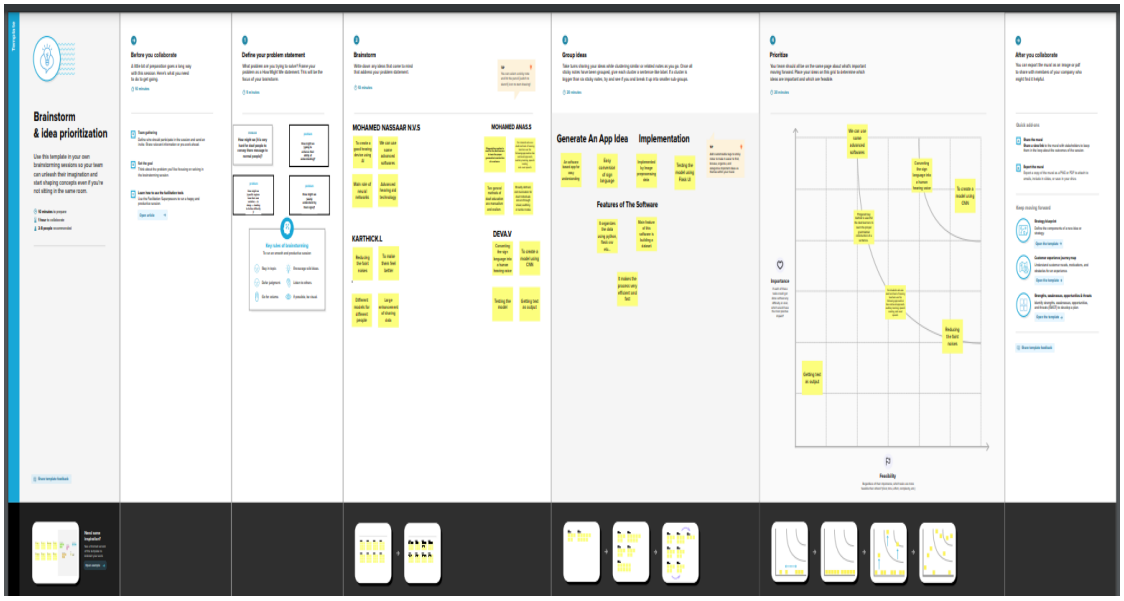
This paper describes the system that overcomes the problem faced by the speech and hearing impaired.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



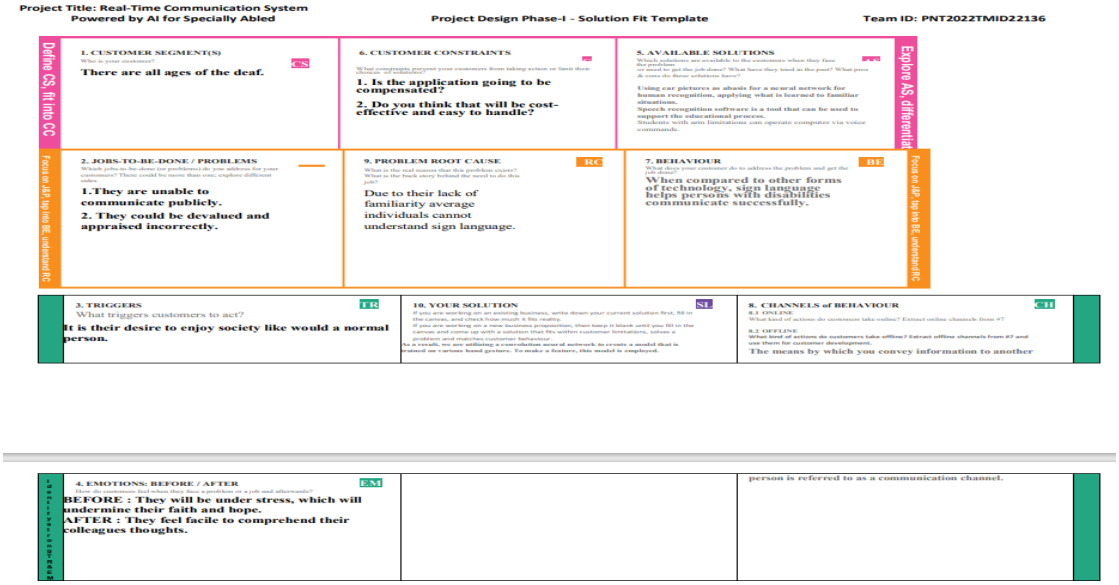
3.2 Ideation & Brainstorming



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem statement (Problem to be solved)	It has always been difficult to communicate with those who are deaf and hearing people. Deaf people find it extremely difficult to communicate with hearing people. Due to regional differences in slang there is additional compilation
2.	Idea / Solution description	As a result, we are using a convolutional neural network to built model that is trained on various hand movements, A feature is created that utilises the model.
3.	Novelty / Uniqueness	Therefore, the function allows deaf persons to communicate by employing signs that are translated into speech and human-understandable language.
4.	Social impact / customer satisfaction	It can be quite challenging for them to communicate their message in emergency situations. The human hand has remained a common method of communication. information when other formats, such as Speech is not allowed. then voice Conversion system that recognises hand gestures and A proper understanding will benefit from translation.an ordinary human and an alien in Conversation any language-impaired person.
5.	Business model (Revenue model)	Price the product as a service (model), financial sustainability, and the offer.
6.	Scalability of the solution	Even though technology is constantly evolving, there are some substantial advancements being made for the benefit of these folks.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

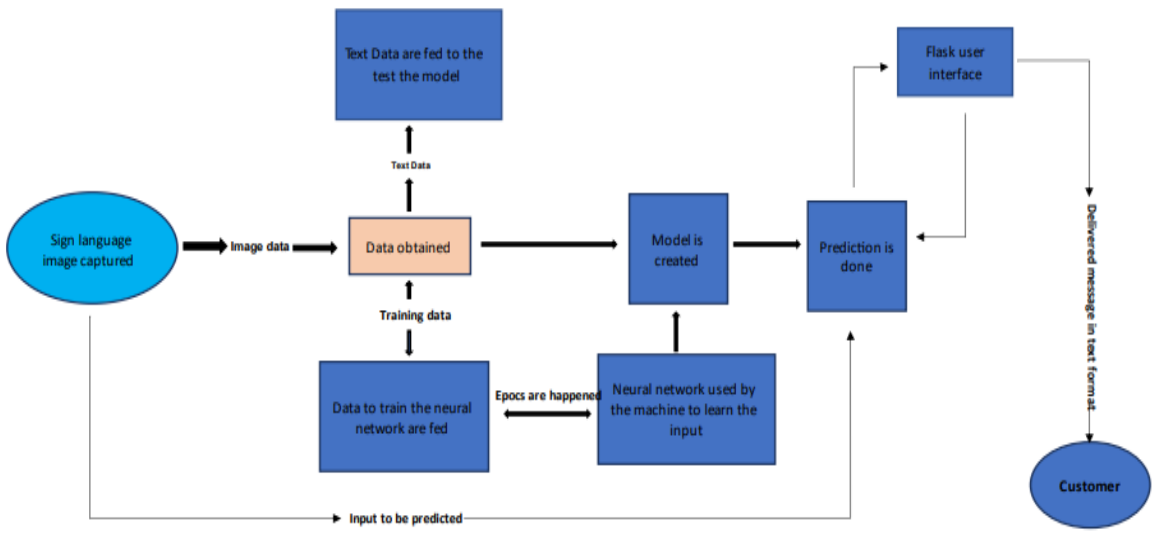
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Test)
FR-1	User Registration	To get required information from customer for registration process
FR-2	User Confirmation	Confirm the details with Email and OTP verification
FR-3	Permission access	Permit the required functions like GPS,Micro phone and Camera.
FR-4	Information provide priority based	From the frequent information provide the particular information first.
FR-5	feedback	To get feedback from customer via product or software
FR-6	troubleshoot	To solve problems in product with customer feedback.

4.2 Non-Functional requirements

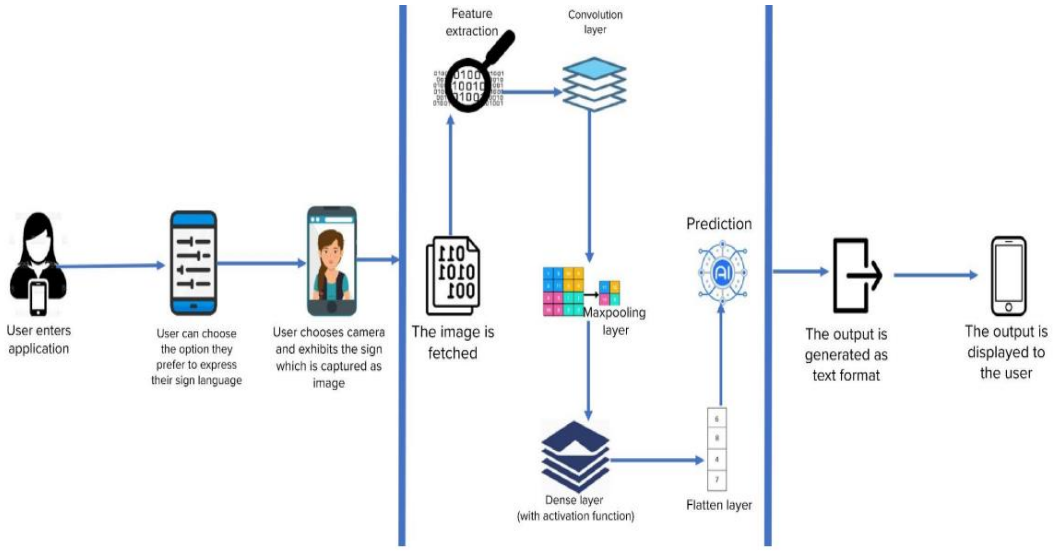
FR NO.	Non-Functional Requirement	Description
NFR-1	Usability	Easily access sign languages during process.
NFR-2	Security	To ensure customer personal information from malware and hackers.
NFR-3	Reliability	To perform sign language conversion without any interruptions
NFR-4	Performance	Sign language conversion should not load more than one minute
NFR-5	Availability	Availability describes how likely the system is accessible to a user at a given point in time.
NFR-6	Scalability	Handle many sign languages conversions without any product degradation.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Deaf-Mute people)	Registration	USN-1	As a user, I can input my sign-language to the system for processing.	The user can input sign language into the system	Low	Sprint-3
		USN-2	As a user, I can input sign-language images to the system for processing.	The user can input images into the system	High	Sprint-1
		USN-3	As a user, I can make sure the input is captured correctly by the system.	The system should capture the input correctly	Medium	Sprint-2
	Processing	USN-4	As a user, I can ensure that the sign language input is correctly getting translated into normal message and voice.	The user can ensure that the processing is done correctly.	Medium	Sprint-2
		USN-5	As a user, I can get acknowledgement from the system about the processing of the input.	The user should get an acknowledgement	High	Sprint-1
		USN-6	As a user, I will get feedback about the processing of the system.	The user should get feedback from the system	Low	Sprint-3
	System Output	USN-7	As a user, I can acknowledge the output of the system by ensuring messages are displayed.	The user should get an acknowledgement from the system	High	Sprint-2
		USN-8	As a user, I can get feedback about the system from its output	The user should get feedback from the system	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

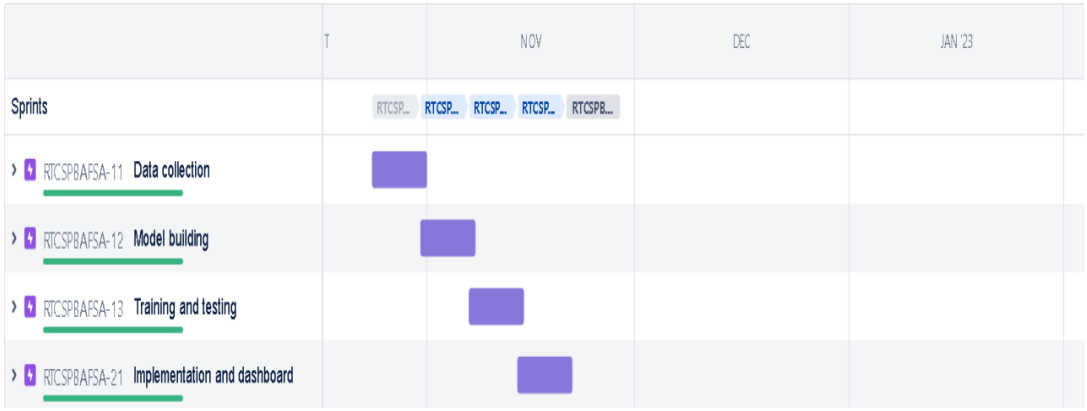
6.1 Sprint Planning & Estimation

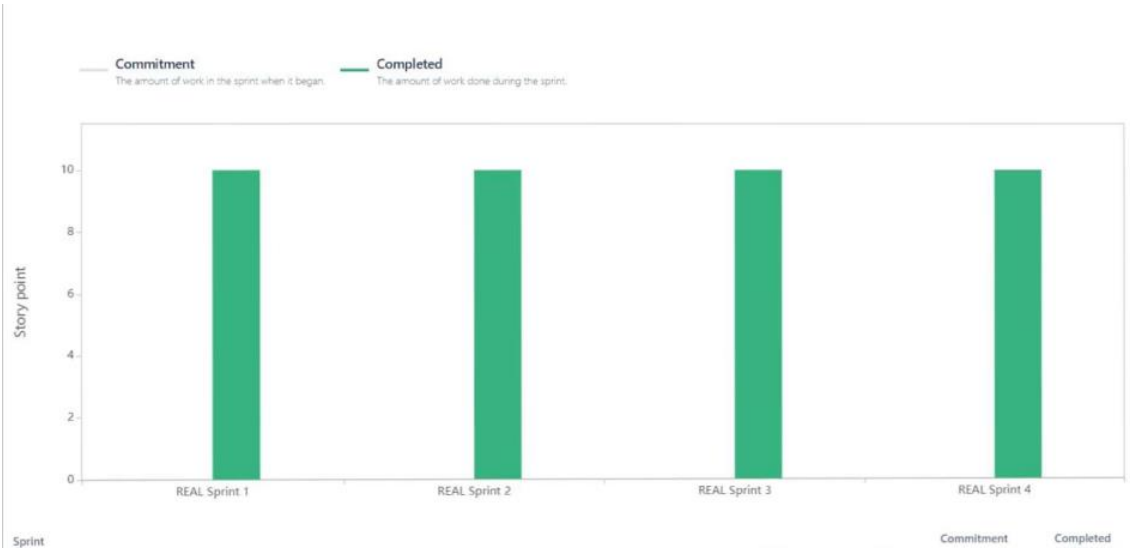
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data collection	USN-1	Collection of required data, login information from user	2	Low	MOHAMED NASSAAR
Sprint-1		USN-2	Image pre-processing	3	High	MOHAMED ANAS
Sprint-2	Model building	USN-3	Import the required libraries, add the necessary layers, and compile the model	2	Low	KARTHICK
Sprint-2		USN-4	Training the image classification model using CNN	3	High	DEVA
Sprint-3	Training and testing	USN-5	Training the model and testing the model's performance	3	High	MOHAMED NASSAAR
Sprint-3		USN-6	Converting the input sign language images into English alphabets and save model for deployment	2	Low	MOHAMED ANAS
Sprint-4	Implementation and dashboard	USN-7	As a user, I can acknowledge the output of the system by ensuring messages are displayed.	2	Low	KARTHICK
Sprint-4		USN-8	As a user, I can get and give feedback about the system from its output.	3	High	DEVA

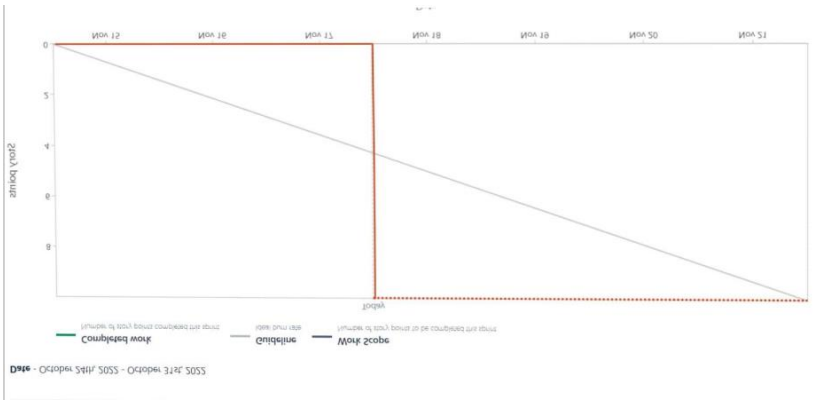
6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	05	6 Days	24 Oct 2022	29 Oct 2022	05	05 Nov 2022
Sprint-2	05	6 Days	31 Oct 2022	05 Nov 2022	05	08 Nov 2022
Sprint-3	05	6 Days	07 Nov 2022	14 Nov 2022	05	12 Nov 2022
Sprint-4	05	6 Days	14 Nov 2022	19 Nov 2022	05	19 Nov 2022

6.3 Reports from JIRA







7. CODING & SOLUTIONING

7.1 Feature 1

Import ImageDataGenerator Library and configure it

In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
%matplotlib inline
```

In [18]:
```
import abc
import functools
import multiprocessing
import sys
import threading
import warnings

import numpy as np
import six
```

In [19]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale = 1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale = 1./255)
```

In [24]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='7rF2Rxrd9ApnHSPwzdSzTE4NszW4TKK_Y4XGIi0wn2we',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'imageclassification-donotdelete-pr-tyvlh8ju3g4n38'
object_key = 'Dataset.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

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

In [25]:
```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [26]:
```
pwd
```

Out[26]:
```
'/home/wsuser/work'
```

In [27]:
```
import os
filenamer = os.listdir('/home/wsuser/work/Dataset/training_set')
```

In [28]:
```
dir1='/home/wsuser/work/Dataset/training_set'
```

In [29]:
```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
```

In [30]:
```
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [31]:
```
xtrain=train_datagen.flow_from_directory("/home/wsuser/work/Dataset/training_set",target_size=(64,64),class_mode="categorical",batch_size=25)
```

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

In [32]:
```
xtest=test_datagen.flow_from_directory("/home/wsuser/work/Dataset/test_set",target_size=(64,64),
class_mode='categorical' , batch_size=25)
```

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

In [33]:
```
xtrain.class_indices
```

Out[33]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

In [34]:
```
xtest.class_indices
```

Out[34]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

In [39]:
```
from tensorflow.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
```

In [40]:
```
model=Sequential()
```

In [41]:
```
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [42]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [43]:
```
model.add(Flatten())
```

In [44]:
```
model.add(Dense (units=512, activation='relu'))
model.add(Dense (units=9, activation='softmax'))
```

In [45]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [46]:
```
len(xtrain)
```

Out[46]:
```
630
```

In [47]:
```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

In [48]:
```
early_stopping = EarlyStopping(monitor='val_accuracy',
                       patience=5)
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy',
                       patience=5,
                       factor=0.5,min_lr=0.00001)

callback = [reduce_lr,early_stopping]
```

In [49]:
```
model.fit(xtrain,
          steps_per_epoch=len(xtrain),
          epochs=10,
          callbacks=callback,
          validation_data=xtest,
          validation_steps=len(xtest))
```

```
Epoch 1/10
630/630 [==============================] - 142s 225ms/step - loss: 0.1915 - accuracy: 0.9445 - val_loss: 0.2997 - val_accuracy: 0.9729 - lr: 0.0010
Epoch 2/10
630/630 [==============================] - 143s 226ms/step - loss: 0.0270 - accuracy: 0.9913 - val_loss: 0.4361 - val_accuracy: 0.9720 - lr: 0.0010
Epoch 3/10
630/630 [==============================] - 142s 225ms/step - loss: 0.0173 - accuracy: 0.9943 - val_loss: 0.3518 - val_accuracy: 0.9756 - lr: 0.0010
Epoch 4/10
630/630 [==============================] - 143s 227ms/step - loss: 0.0106 - accuracy: 0.9968 - val_loss: 0.4301 - val_accuracy: 0.9791 - lr: 0.0010
Epoch 5/10
630/630 [==============================] - 144s 229ms/step - loss: 0.0124 - accuracy: 0.9960 - val_loss: 0.2558 - val_accuracy: 0.9844 - lr: 0.0010
Epoch 6/10
630/630 [==============================] - 142s 226ms/step - loss: 0.0103 - accuracy: 0.9966 - val_loss: 0.5808 - val_accuracy: 0.9778 - lr: 0.0010
Epoch 7/10
630/630 [==============================] - 141s 224ms/step - loss: 0.0068 - accuracy: 0.9977 - val_loss: 0.7558 - val_accuracy: 0.9773 - lr: 0.0010
Epoch 8/10
630/630 [==============================] - 141s 224ms/step - loss: 0.0050 - accuracy: 0.9985 - val_loss: 0.6070 - val_accuracy: 0.9809 - lr: 0.0010
Epoch 9/10
[==============================] - 142s 226ms/step
Epoch 10/10
630/630 [==============================] - 142s 226ms/step - loss: 0.0029 - accuracy: 0.9990 - val_loss: 0.9111 - val_accuracy: 0.9773 - lr: 0.0010
```

Out[49]:

In [50]:
```
model.save('Sign.h5')
```

In [51]:
```
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):
    img=img/255.0
    prediction = model.predict(img)
    print(prediction)
    prediction = model.predict_classes(img)
    print(prediction)
```

In [52]:
```
!tar -zcvf image_new.tgz Sign.h5
```

```
Sign.h5
```

In [53]:
```
!pip install watson-machine-learning-client --upgrade
```

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 16.8 MB/s eta 0:00:01
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.9.24)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learni
ng-client) (1.21.41)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learnin
g-client) (0.5.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-
client) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21-
>boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,
>=1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-mac
hine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-l
earning-client) (2.11.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-lea
rning-client) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client)
(3.3)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client)
(2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client)
(1.19.5)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

In [74]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
                   "url": "https://us-south.ml.cloud.ibm.com",
                   "apikey":"UoSEdlAFnOKSoSAZ0gI8d4OJEfYBdImfc7isyLvbKp7u"
}
client=APIClient(wml_credentials)
```

In [56]:
```
client=APIClient(wml_credentials)
```

In [58]:
```
def guid_from_space_name(client, space_name):
    space=client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"]== space_name)['metadata']['id'])
```

In [75]:
```
space_uid=guid_from_space_name(client,'image classification')
print("Space UID=" + space_uid)
```

```
Space UID=d9d9fe54-f62d-4506-9b45-eccb3c0648da
```

In [76]:
```
client.set.default_space(space_uid)
```

Out[76]:
```
'SUCCESS'
```

In [77]:
```
client.software_specifications.list()
```

NAME	ASSET_ID	TYPE
default_py3.6	0062b8c9-8b7d-44a0-a9b9-46c416adcbd9	base
kernel-spark3.2-scala2.12	020d69ce-7ac1-5e68-ac1a-31189867356a	base
pytorch-onnx_1.3-py3.7-edt	069ea134-3346-5748-b513-49120e15d288	base
scikit-learn_0.20-py3.6	09c5a1d0-9c1e-4473-a344-eb7b665ff687	base
spark-mllib_3.0-scala_2.12	09f4cff0-90a7-5899-b9ed-1ef348aebdee	base
pytorch-onnx_rt22.1-py3.9	0b848dd4-e681-5599-be41-b5f6fccc6471	base
ai-function_0.1-py3.6	0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda	base
shiny-r3.6	0e6e79df-875e-4f24-8ae9-62dcc2148306	base
tensorflow_2.4-py3.7-horovod	1092590a-307d-563d-9b62-4eb7d64b3f22	base
pytorch_1.1-py3.6	10ac12d6-6b30-4ccd-8392-3e922c096a92	base
tensorflow_1.15-py3.6-ddl	111e41b3-de2d-5422-a4d6-bf776828c4b7	base
autoai-kb_rt22.2-py3.10	125b6d9a-5b1f-5e8d-972a-b251688ccf40	base
runtime-22.1-py3.9	12b83a17-24d8-5082-900f-0ab31fbfd3cb	base
scikit-learn_0.22-py3.6	154010fa-5b3b-4ac1-82af-4d5ee5abbc85	base
default_r3.6	1b70aec3-ab34-4b87-8aa0-a4a3c8296a36	base
pytorch-onnx_1.3-py3.6	1bc6029a-cc97-56da-b8e0-39c3880dbbe7	base
kernel-spark3.3-r3.6	1c9e5454-f216-59dd-a20e-474a5cdf5988	base
pytorch-onnx_rt22.1-py3.9-edt	1d362186-7ad5-5b59-8b6c-9d0880bde37f	base
tensorflow_2.1-py3.6	1eb25b84-d6ed-5dde-b6a5-3fbdf1665666	base
spark-mllib_3.2	20047f72-0a98-58c7-9ff5-a77b012eb8f5	base
tensorflow_2.4-py3.8-horovod	217c16f6-178f-56bf-824a-b19f20564c49	base
runtime-22.1-py3.9-cuda	26215f05-08c3-5a41-a1b0-da66306ce658	base
do_py3.8	295addb5-9ef9-547e-9bf4-92ae3563e720	base
autoai-ts_3.8-py3.8	2aa0c932-798f-5ae9-abd6-15e0c2402fb5	base
tensorflow_1.15-py3.6	2b73a275-7cbf-420b-a912-eae7f436e0bc	base
kernel-spark3.3-py3.9	2b7961e2-e3b1-5a8c-a491-482c8368839a	base
pytorch_1.2-py3.6	2c8ef57d-2687-4b7d-acce-01f94976dac1	base
spark-mllib_2.3	2e51f700-bca0-4b0d-88dc-5c6791338875	base
pytorch-onnx_1.1-py3.6-edt	32983cea-3f32-4400-8965-dde874a8d67e	base
spark-mllib_3.0-py37	36507ebe-8770-55ba-ab2a-eafe787600e9	base
spark-mllib_2.4	390d21f8-e58b-4fac-9c55-d7ceda621326	base
autoai-ts_rt22.2-py3.10	396b2e83-0953-5b86-9a55-7ce1628a406f	base
xgboost_0.82-py3.6	39e31acd-5f30-41dc-ae44-60233c80306e	base
pytorch-onnx_1.2-py3.6-edt	40589d0e-7019-4e28-8daa-fb03b6f4fe12	base
pytorch-onnx_rt22.2-py3.10	40e73f55-783a-5535-b3fa-0c8b94291431	base
default_r36py38	41c247d3-45f8-5a71-b065-8580229facf0	base
autoai-ts_rt22.1-py3.9	4269d26e-07ba-5d40-8f66-2d495b0c71f7	base
autoai-obm_3.0	42b92e18-d9ab-567f-988a-4240ba1ed5f7	base
pmml-3.0_4.3	493bcb95-16f1-5bc5-bee8-81b8af80e9c7	base
spark-mllib_2.4-r_3.6	49403dff-92e9-4c87-a3d7-a42d0021c095	base
xgboost_0.90-py3.6	4ff8d6c2-1343-4c18-85e1-689c965304d3	base
pytorch-onnx_1.1-py3.6	50f95b2a-bc16-43bb-bc94-b0bed208c60b	base
autoai-ts_3.9-py3.8	52c57136-80fa-572e-8728-a5e7cbb42cde	base
spark-mllib_2.4-scala_2.11	55a70f99-7320-4be5-9fb9-9edb5a443af5	base
spark-mllib_3.0	5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9	base
autoai-obm_2.0	5c2e37fa-80b8-5e77-840f-d912469614ee	base
spss-modeler_18.1	5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b	base
cuda-py3.8	5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e	base
autoai-kb_3.1-py3.7	632d4b22-10aa-5180-88f0-f52dfb6444d7	base
pytorch-onnx_1.7-py3.8	634d3cdc-b562-5bf9-a2d4-ea90a478456b	base

```
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [78]:
```
software_spec_uid= client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
```

Out[78]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'

In [87]:
```
model_details=client.repository.store_model(model='image_new.tgz',meta_props={
client.repository.ModelMetaNames.NAME: "CNN",
client.repository.ModelMetaNames. TYPE: "tensorflow_2.7",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid}
                                            )
model_id=client.repository.get_model_uid(model_details)
```

```
This method is deprecated, please use get_model_id()
```

In [80]:
```
model_id
```

Out[80]: '620e9d5d-31fe-48fb-bb5e-308f951a99ec'

In [88]:
```
client.repository.download (model_id, 'my_model.tar.gz')
```

```
Successfully saved model content to file: 'my_model.tar.gz'
```

Out[88]: '/home/wsuser/work/my_model.tar.gz'

In [66]:
```
from keras.models import load_model
from keras.preprocessing import image
```

In [67]:
```
model=load_model("Sign.h5")
```

In [68]:
```
import numpy as np
from tensorflow.keras.models import load_model
from keras.preprocessing import image
```

In [70]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='7rF2Rxrd0ApnHSPwzdSzTE4NszW4TKK_Y4XGIi0wn2we',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'imageclassification-donotdelete-pr-tyvlh8ju3g4n38'
object_key = 'Dataset.zip'

streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

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

In [71]:
```
img=image.load_img(r"/home/wsuser/work/Dataset/test_set/A/90.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

Out[71]: 'A'

In [72]:
```
img=image.load_img( "/home/wsuser/work/Dataset/test_set/D/1.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis = 1)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

Out[72]: 'D'

In [73]:
```
img=image.load_img(r"/home/wsuser/work/Dataset/test_set/G/1.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x), axis=1)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

Out[73]: 'G'

8. TESTING

8.1 Test Cases

In [2]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [8]:
```
model=load_model("sign.h5")
```

In [11]:
```
img =image.load_img('15.png',target_size=(64,64))
```

In [12]:
```
import numpy as np
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
```

In [13]:
```
x.shape
```

Out[13]:
```
(1, 64, 64, 3)
```

In [14]:
```
img=image.load_img("15.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

```
1/1 [==============================] - 10s 10s/step
```

Out[14]:
```
'A'
```

In [16]:
```
img=image.load_img("6.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

```
1/1 [==============================] - 0s 211ms/step
```

Out[16]:
```
'A'
```

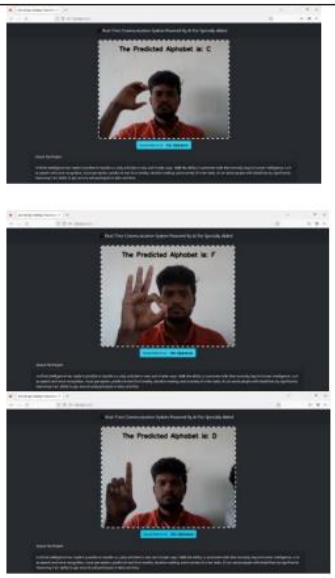
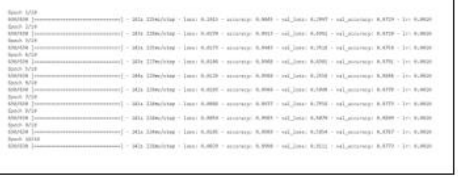
8.2 User Acceptance Testing

Date	18th -Nov-22
Team ID	PNT2022TMID22136
Project Name	Real-Time Communication System
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Webpage_TC_OO1	Functional	Home Page	Check that the user can see the application's interface, which includes a camera window and a reference button.		1. Enter URL 2. Paste the URL in Local Browser 3.Wait till the camera opens 1.Enter URL 2.Paste the URL in Local Browser	http://127.0.0.1:5000/	A window containing the website's camera interface should open. 1. The camera should open with a reference button, the project description and the developer.	working as expected	Pass	Got the expected output			MOHAMED NASLAAR..N.V.S, MOHAMED ANAS.S,DEVA.V,KARTHICK.L
Webpage_TC_OO2	UI	Home Page	Check the website's interface's UI elements.		3. Wait till the camera opens 4. Showing the hand gestures now by selecting the "American Alphabet" using the reference button.	http://127.0.0.1:5000/	2. Now the Image processing should detect the user's hand gestures.	Working as expected	Pass	Got the expected output			MOHAMED NASLAAR..N.V.S, MOHAMED ANAS.S,DEVA.V,KARTHICK.L

9. RESULTS

9.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Model Summary	The project aims to develop a system that converts the sign language into a human hearing voice in the desired language to convert speech into understandable sign language for the deaf . We are making use of aconvolution 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 people to convey their information using signs which get converted to human-understandable language and speech is given as output.	
2.	Accuracy	Training Accuracy - 0 . 9990 Validation Accuracy -0.9773	

10. ADVANTAGES

Advantages:

1. It is feasible to develop a mobile application to close the communication gap between the hearing-impaired and the rest of society.

2. The user may select which sign language to read by adding the dataset when new sign language standards are created.

3. The disabled people who have hearing impairment will not have to be socially anxious anymore. They can communicate with great confidence

11. Disadvantages:

1. The present model is limited to the letters A through I.

2. Alphabets from J cannot be recognised in the absence of gesture recognition because they need user input in the form of a gesture.

3. The accuracy isn't excellent because there aren't many or high-quality photographs in the dataset, but that can be fixed by changing the datas.

12. Conclusion

The use of sign languages can help normal and deaf individuals communicate more effectively. Our approach strives to reduce the communication gap between the deaf community and the rest of society since it supports two-way conversation. Our suggested technology converts sign languages into human-understandable English language. With the help of this technology, the model receives hand gestures, recognises them, and then shows the corresponding Alphabet on the screen. This initiative allows deaf-mute persons to perform sign language with their hands, which will later be translated into alphabets.

13. Future Scope

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

14. APPENDIX

Source Code for Model Building

Fit and save the model

In [25]:
```
model.fit_generator(x_train, steps_per_epoch=24, epochs=10, validation_data= x_test, validation_steps=40)
```

```
/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 supports generators.
  """Entry point for launching an IPython kernel.
Epoch 1/10
24/24 [==============================] - ETA: 0s - loss: 0.0041 - accuracy: 0.9994
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * ep
ochs` batches (in this case, 40 batches). You may need to use the repeat() function when building your dataset.
24/24 [==============================] - 43s 2s/step - loss: 0.0041 - accuracy: 0.9994 - val_loss: 0.2699 - val_accuracy: 0.9764
Epoch 2/10
24/24 [==============================] - 35s 1s/step - loss: 0.0044 - accuracy: 0.9989
Epoch 3/10
24/24 [==============================] - 31s 1s/step - loss: 0.0035 - accuracy: 0.9994
Epoch 4/10
24/24 [==============================] - 32s 1s/step - loss: 0.0018 - accuracy: 0.9999
Epoch 5/10
24/24 [==============================] - 31s 1s/step - loss: 0.0026 - accuracy: 0.9996
Epoch 6/10
24/24 [==============================] - 31s 1s/step - loss: 0.0020 - accuracy: 0.9999
Epoch 7/10
24/24 [==============================] - 30s 1s/step - loss: 0.0016 - accuracy: 0.9999
Epoch 8/10
24/24 [==============================] - 31s 1s/step - loss: 0.0022 - accuracy: 0.9997
Epoch 9/10
24/24 [==============================] - 33s 1s/step - loss: 0.0016 - accuracy: 0.9999
Epoch 10/10
24/24 [==============================] - 31s 1s/step - loss: 0.0015 - accuracy: 0.9997
```

Out[25]:

In [26]:
```
model.save('comm.h5')
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-9062-1658950298

Project Demo Link: