

PROJECT TITLE: REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

TEAM ID:PNT2022TMID03716

TEAM MEMBERS

SHRUTHI R (TEAM LEAD)

SAKTHIPRIYA B

RAJA E

RADHA KRISHNAN P

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

Communication plays a significant role in making the world a better place. Communication creates bonding and relations among the people, whether persona, social, or political views. Most people communicate efficiently without any issues, but many cannot due to disability. They cannot hear or speak, which makes Earth a problematic place to live for them. Even simple basic tasks become difficult for them. Disability is an emotive human condition. It limits the individual to a certain level of performance. Being deaf and dumb pushes the subject to oblivion, highly introverted. In a world of inequality, this society needs empowerment. Harnessing technology to improve their welfare is necessary. In a tech era, no one should be limited due to his or her inability. The application of technology should create a platform or a world of equality despite the natural state of humans. On the other hand, technology is the most innovative thing on Earth for every time the clock ticks, researchers, software engineers, programmers, and information technology specialists are always coming up with bright ideas to provide convenience to everyone.

This project shows how artificial intelligence is being used to help people who are unable to do what most people do in their everyday lives. Aligned with communication, D-talk is a system that allows people who are unable to talk and hear be fully understood and for them to learn their

language easier and also for the people that would interact and communicate with them.

2. LITERATURE REVIEW

2.1 EXISTING PROBLEM

1. An AI software to communicate with deaf and mute in real time

This has been made possible by a third-year engineering student at BNMIT who has developed an Artificial Intelligence (AI) powered software application for the welfare of the deaf and mute people. The software, christened DnD Mate, does not only translate sign language into text and speech, but also translates speech into sign language, all in real time and as quick as the person speaks. Currently, there are no applications/software that facilitates a two-way communication channel.

This easy-to-use innovative digital translator works with your device's in-built cameras, reads hand and facial gestures by the deaf and mute user and translates them into text and speech. That is not all! The software will also translate your voice or text input into sign language. 'The software is based on a Deep Learning model and can work both offline and online. While in the offline mode, the deaf and mute person can communicate with you on the same device in real time; in the online mode, you can converse sitting in far off places as well, just like you talk to anyone over a video call,' says Bhargav DV, the third year Electronics and Communication Engineering student at BNMIT.

2. Sign Language Recognition System for People with Disability using Machine Learning and Image Processing by M.Saleh, R. Albeshir, M.Tariq

This paper shows how artificial intelligence is being used to help people who are unable to do what most people do in their everyday lives. Aligned with communication, D-talk is a system that allows people who are unable to talk and hear be fully understood and for them to learn their language easier and also for the people that would interact and communicate with them. This system provides detailed hand gestures that show the interpretation at the bottom so that everyone can understand them. This research allows the readers to learn the system and what it can do to people who are struggling with what they are not capable of and will provide the technical terms on how the system works.

3. Design of a Communication System using Sign Language aid for Differently Abled Peoples by Shrikant Temburwar, Payal Jaiswal, Shital Mande, Souparnika Patil

Our goal is to design a human computer interface system that can accurately identify the language of the deaf and dumb. With the use of image processing and artificial intelligence, many techniques and algorithms have been developed in this area. Each character speech recognition system is trained to recognize the characters and convert them into the required pattern. The proposed system aims to give speech speechless, a real-time character language

is captured as a series of images, and it is processed and then converted into speech and text

4. Two Way Communicator between Deaf and Dumb People and Normal People.

This system consists mainly of two modules, the first module is Indian Sign Language (ISL) gestures from real-time video and mapping it with human-Understandable speech. Accordingly, the second module is the natural language as Input and card with equivalent Indian Sign Language animated gestures.

5. Sign Language Learning based on Android for Deaf and Speech Impaired People.

This research makes an Android-based application that can directly interpret Sign language presented by deaf people in written language. Translation process Starts with the detection of hands with OpenCV and translation of and signals The K-NN classification. Tutorial features added in this application with the goal to train intensively to guide the user when using the sign language.

2.2 REFERENCES:

1. Shreyashi Narayan Sawant, "Sign Language recognition System to aid Deaf- dumb People Using PCA", IJCSET ISSN : 2229-3345 Vol. 5 No. 05 May 2014.
2. Setiawardhana, Rizky Yuniar Hakkun, Achmad Baharuddin, "Sign Language Learning based on Android For Deaf and Speech Impaired People", 978-1-4673-9345- 4/15/31.00 c 2015 IEEE
3. M. Ebrahim Al-Ahdal & Nooritawati Md Tahir," Review in Sign Language Recognition Systems" Symposium on Computer & Informatics(ISCI),pp:52-57, IEEE ,2012
4. Archana S. Ghotkar, Rucha Khatal, Sanjana Khupase, Surbhi Asati & Mithila Hadap," Hand Gesture Recognition for Indian Sign Language" International Conference on Computer Communication and Informatics (ICCCI), pp:1- 4.IEEE,Jan 2012
5. <https://www.bnmit.org/an-ai-software-to-communicate/>
6. Prof. P.G. Ahire, K.B. Tilekary,T.A. Jawake, P.B. Warale, "Two Way Communicator between Deaf and Dumb People and Normal People", 978-1-4799-6892-3/15 31.00 c 2015 IEEE.
7. Design of a Communication System using Sign Language aid for Differently Abled Peoples. Shrikant Temburwar, Payal Jaiswal, Shital Mande, Souparnika Patil International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 03 | Mar -2017

2.3 PROBLEM STATEMENT AND DEFINITION:

PROBLEM STATEMENT :

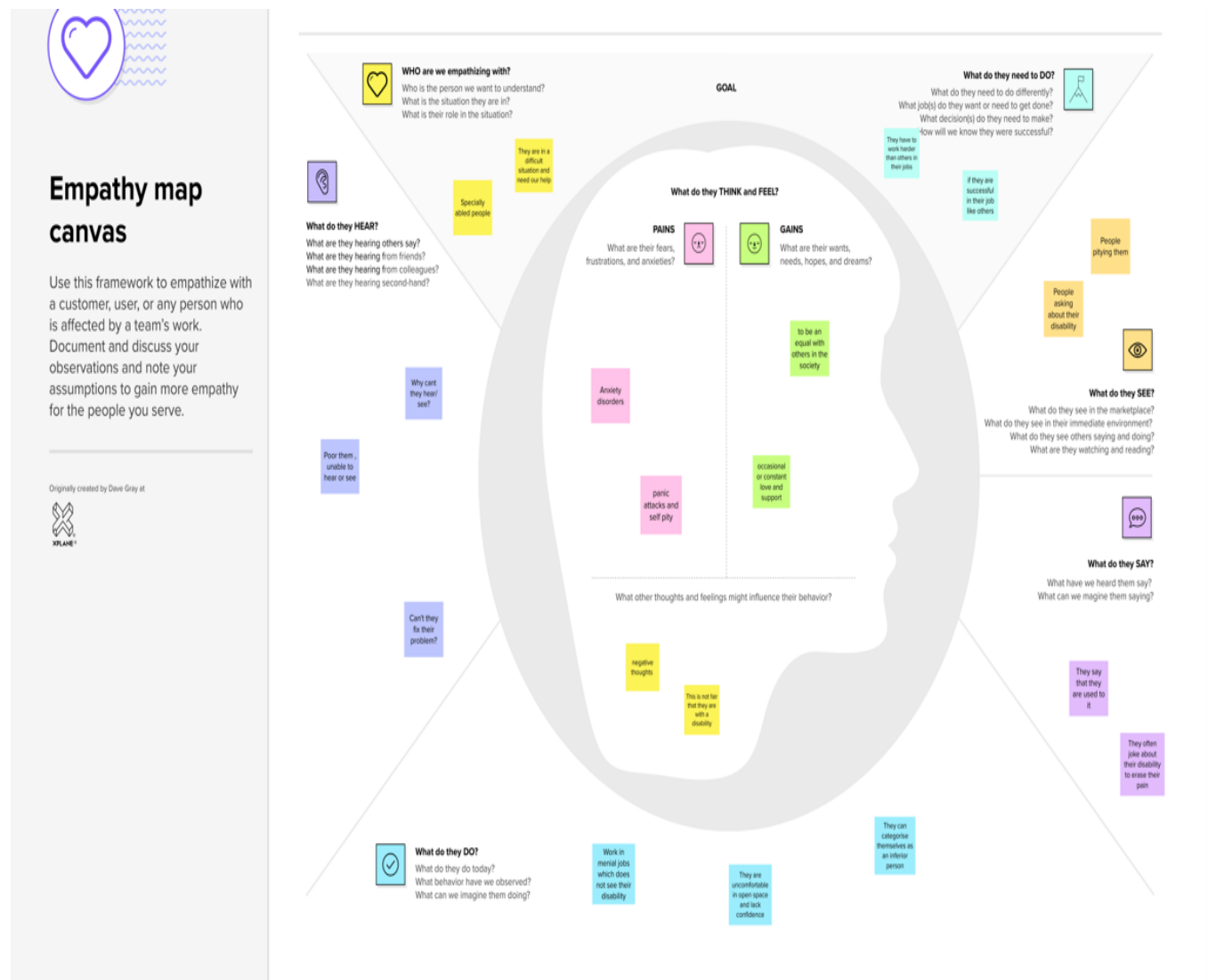
PROBLEM STATEMENTS	I AM (CUSTOMER)	I AM TRYING TO	BUT	BECAUSE	WHICH MAKES ME FEEL
WHY THE CUSTOMER CANNOT UNDERSTAND THE USER INTERFACE	VISUALLY CHALLENGED	COMMUNICATE TO OTHER PEOPLE THROUGH THE APP	I CANNOT UNDERSTAND THE INSTRUCTIONS	THE USER INTERFACE IS COMPLEX	DEVASTATED
WHAT IS THE ERROR	DEAF	CONVERT SIGN LANGUAGE TO AUDIO/SPEECH	IT IS SHOWING ERROR WHILE CONVERTING THE SIGNS	THE BACKGROUND IN THE PICTURE IS TAKEN ALONG WITH THE SIGN	UNHAPPY

PROBLEM DEFINITION:

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.

3. IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:



3.2 IDEATION AND BRAINSTORMING:

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

PROBLEM

**How might we able
to develop a real
time communication
system for specially
abled using artificial
Intelligence and
machine learning**



Key rules of brainstorming

To run an smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

We can get feedback	We need to make sure that the solution is feasible	It should be easier to understand and use
It should not contain the same elements		

Person 2

The problem should be completely addressed	The operation should be simple and easy to understand	The results from the report should be labeled properly
Having a system that can be used by the user		

Person 3

The results should be easy to understand and use	The output must be given and proper	It should be easy to use and not too complicated
It should be easy to use and not too complicated		

Person 4

First the user should be able to understand the system	The background should be easy to understand	We need to make sure the system is easy to use
We can have a system that can be used by the user	It should be easy to use and not too complicated	

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

The user should be able to easily understand how the system works. Interaction between the user and the system should be user friendly.

Speech inputs can also be included to add an option for the individuals with visual impairment.

Instant transcribing the conversation using a voice control similar to google assistant can be used.

They should be able to easily navigate throughout the system and locate their specific needs.

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

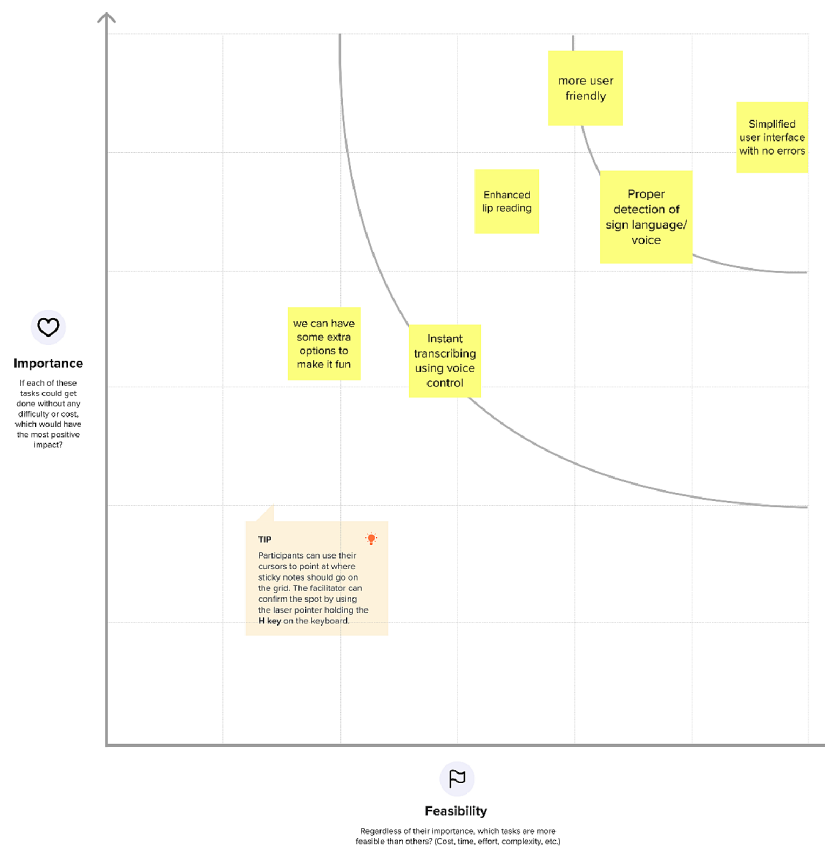
Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



3.3 PROPOSED SOLUTION:

S.NO	PARAMETER	DESCRIPTION
1	Problem Statement (Problem to be solved)	Statement - Communication between deaf-mute and a normal person has always been a challenging task. Description - The Deaf/Dumb people needs a way to communicate easily and quickly with the normal people, so that the Deaf/Dumb people feel confident enough to express there thought, ideas, and can make conversation with the normal people.
2	Idea / Solution description	1. Designing and implementing a system using artificial intelligence neural network deep Learning algorithms and image processing concepts to take input as hand gestures (or) sign language and it generates recognizable outputs in the form of text and voice. 2. We can convert the sign languages into voice or text. So that the specially abled people will convey the message to normal people.
3	Novelty / Uniqueness	The system uses neural network and machine learning to train the models and Computer vision to recognizes the video or image of sign language then smart deep learning algorithms translate it into speech or text.
4	Social Impact / Customer Satisfaction	1. As the specially abled people feel very difficult to convey their message to normal people in emergency times as well as in normal times. This project mainly focuses on their safety. 2. The main purpose of this application is to make specially abled people feel independent and more confident 3. This application can also make the blind and deaf people to communicate easily and make them to feel at ease on public
5	Business Model (Revenue Model)	The system can generate revenue through direct customers and collaborate with health care sector and government ,this generates revenue from their customers.

6	Scalability of the Solution	1.Specially abled people can participate in daily activities rather than being inactive and can get good job opportunities. This provides tremendous opportunities to the specially abled people to show their skills. 2.Adaptive learning platforms also provide personalised learning experiences tailored to the specific needs of students with disabilities. 3. This application aims to help specially abled(like blind and deaf) by providing them with an attractive and easy way of communication.
---	-----------------------------	---

3.4 PROBLEM SOLUTION FIT:

Problem-Solution fit canvas 2.0

TEAM ID: PNT2022TMID03716

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Who is your customer? I.e. working parents of 0-5 y.o. kids <div>The deaf and dumb, whom we collectively term as the "Specially abled" people.</div>	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. <div>The specially -abled people find difficulties in communication with others. This makes them reluctant to encounter new environment and people.</div>	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 <div>Deaf and dumb tend to write or text in order to communicate which is found unviable in absence of necessary materials. They also make use of lip-reading, gestures and pointers to communicate.</div>	Explore AS, differentiate
	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. <div>Conversion of sign language into audio and text messages.</div>	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. <div>Normal people don't take any effort to learn sign language which makes the communication with the specially-abled difficult.</div>	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) <div>They seek for interpreters and mobile applications to build communication with normal people.</div>	
Focus on J&P, tap into BE, understand RC	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. <div>The ease of communication by the normal people.</div>	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. <div>To develop a web-based application to facilitate the communication between the normal and the specially-abled people using advanced deep learning Algorithm and Natural Language Processing</div>	8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 <div>Social media application like Twitter, WhatsApp etc.</div> 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. <div>Local Community Camps conducted by NGOs, advertorial posters and interpreters.</div>	Extract online & offline CH of BE
	4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards? Before: Feeling unfair about their communication ability when compared to normal people. After: Feeling better and bridging the gaps between people. <div></div>			

Problem-Solution fit canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 license
 Created by Daria Nepriakhina / Amaltama.com

AMALTAMA

4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENT:

FR NO	FUNCTIONAL REQUIREMENT(EPIC)	SUB REQUIREMENT(STORY/SUB-TASK)
FR-1	User Registration	Registration through form registration through gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Authentication	Authenticat through Facial recognition Authentication through Password authentication protocol
FR-4	External interfaces	Robots and other tools provide home-basedcare and other assistance, allowing people with disabilities to live independently
FR-5	Transaction Processing	More application can use to translate the sign language like D talk in the system
FR-6	Reporting	There is a growing feeling that we need to do more, to help make the lives of people with disabilities easier
FR-7	Business Rules	Human augumentation and practical accuracy are responsible of AI business rules

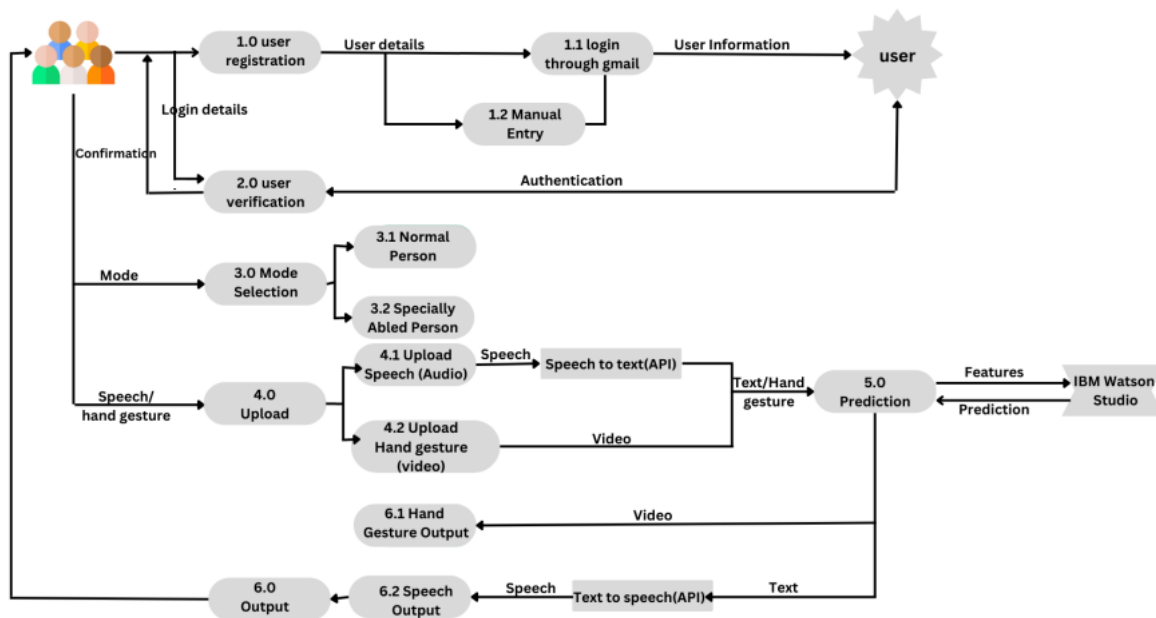
4.2 NON FUNCTIONAL REQUIREMENTS:

FR NO	NON FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR-1	Usability	Provide personalised learning experiences tailoredto the specific needs of students with disabilities
NFR-2	Security	Set the inclusion and exclusion criteria,report the results in the survey
NFR-3	Reliability	It setting the pace of the future and helping people in need
NFR-4	Performance	Enables people with disabilities to step into a world where their

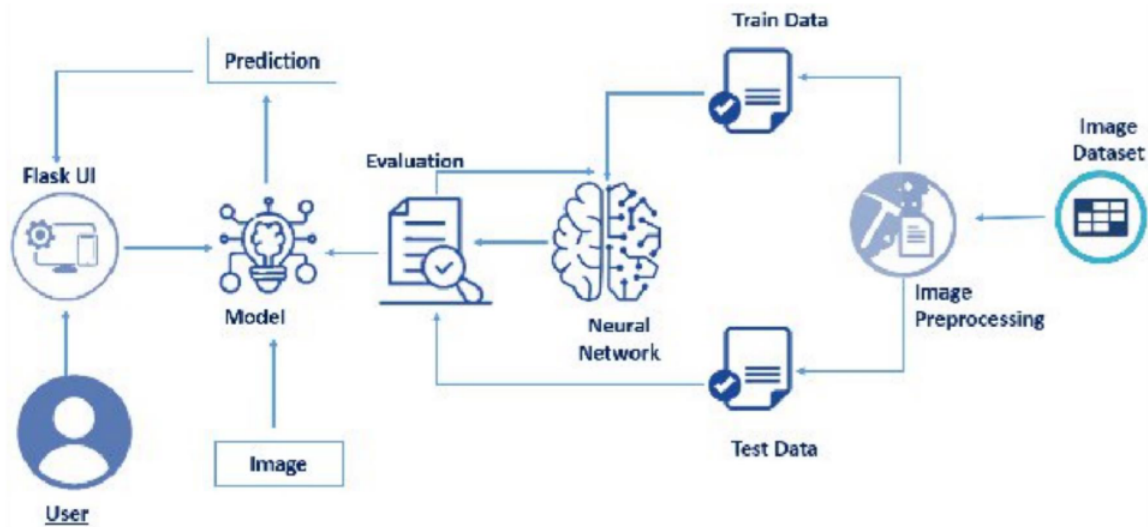
		difficulties are understood and taken into account
NFR-5	Availability	Technoloty solutions that mimic humans and use logic from playing chess to solving equations and machine learning is one of the technologies
NFR-6	Scalability	The improvement in the specially abled persons interaction with environments

5. PROJECT DESIGN:

5.1 DATA FLOW DIAGRAM:



5.2 SOLUTION AND TECHNICAL ARCHITECTURE:



5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Specially Abled Person)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account and select the mode of usage	High	Sprint-1
		USN-2	As a user, I can register for the application through Gmail		High	Sprint-1
	Confirmation	USN-3	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password	Credentials has to be matched	Medium	Sprint-2
	Mode Selection	USN-5	As a user, I will be prompted to select the mode of communication and I will select the specially abled mode (Gesture to Speech)	Either of the modes has to be chosen for further processing	High	Sprint-3
	Video Capturing	USN-6	As a user of this mode I will capture my hand gesture as video	Minimum video quality criteria has to be met	High	Sprint-1
	Gesture interpretation	USN-7	As a user of this mode, I will be able to receive and interpret the translated gestures from the other end.	Must be a valid gesture	Low	Sprint-1

Customer (Normal Person)	Registration	USN-8	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account and select the mode of usage	High	Sprint-1
		USN-9	As a user, I can register for the application through Gmail		High	Sprint-1
	Confirmation	USN-10	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Sprint-2
	Login	USN-11	As a user, I can log into the application by entering email & password	Credentials has to be matched	Medium	Sprint-2
	Mode Selection	USN-12	As a user, I will be prompted to select the mode of communication and I will select the specially abled mode (Gesture to Speech)	Either of the modes has to be chosen for further processing	High	Sprint-3
	Speech Recording	USN-13	As a user of this mode I will record the speech in order to convert it into gesture	Minimum audio quality criteria have to be met	High	Sprint-1
	Speech recognition	USN-14	As a user of this mode, I will be able to receive and interpret the translated speech from the other end.	The words must be a recognizable	Low	Sprint-1
Administrator	Application monitoring and controlling	USN-15	As an admin, I will be responsible for controlling the user activities and further upgradations of the application	Admin level privilege	Medium	Sprint-3

6. PROJECT PLANNING AND SCHEDULING:

6.1 SPRINT PLANNING AND 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.	2	High	R SHRUTHI SAKTHIPRIYA B
Sprint-2		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	RAJA E
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	Medium	RADHA KRISHNAN P
Sprint-2	Dashboard	USN-4	As a user, I can log into my account in a given Dashboard	1	High	SAKTHIPRIYA B
Sprint-1	User interface	USN-5	Professional responsible for user requirements & needs	1	High	R SHRUTHI
Sprint-3	Objective	USN-6	The goal is to describe all the inputs and outputs	1	High	RAJA E

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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA:

[illegible]

OCT								NOV								NOV								NOV							
24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20				
																								RTCSPBAF Sprint 4							

7. CODING AND SOLUTIONING:

7.1 FEATURE 1: TRAINING CNN MODEL IN IBM

This feature lets us to train our test model in IBM WATSON STUDIO for better and accurate model building and neural network prediction.

CODE:

CNN Prediction for Real Time Communication through AI for Specially Abled

Testing the model

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

model=load_model('aslpng1.h5')
img=image.load_img(r'/content/Dataset/test_set/A/28.png',
                    target_size=(64, 64))
```

img



```
x=image.img_to_array(img)
```

```
x.ndim
```

```
3
```

```
x=np.expand_dims(x,axis=0)
```

```
x.ndim
```

```
4
```

```
pred=np.argmax(model.predict(x),axis=1)
```

```
1/1 [=====] - 0s 73ms/step
```

```
pred
```



```
array([0])
```

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

```
A
```

Downloading From IBM

Connecting to IBM Cloud Storage to Get Model from Deployment

```
pip install ibm_watson_machine_learning  
from ibm_watson_machine_learning import APIClient  
wml_credentials = {  
    "url": "https://us-south.ml.cloud.ibm.com",  
    "apikey": "4y7eNmzaeDsCxie0E5b-PACwiQldF2Ock7lM6VAd28Fb"  
}
```

```
client = APIClient(wml_credentials)
```

```
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'])
```

```
space_uid = guid_from_space_name(client, 'model')  
print("Space UID : ", space_uid)
```

```
client.set.default_space(space_uid)
```

```
client.repository.download("eec72d86-8ba8-46cc-8588-  
c9ff4bf89c85", "aslpng1.tar.gz")
```

7.2 FEATURE 2: MODEL BUILDING USING IBM

CODE:

```
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)

x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 15750 images belonging to 9 classes.

x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 2250 images belonging to 9 classes.

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

model = Sequential()

model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(units=512, activation = 'relu'))

model.add(Dense(units=9, activation = 'softmax'))

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

```

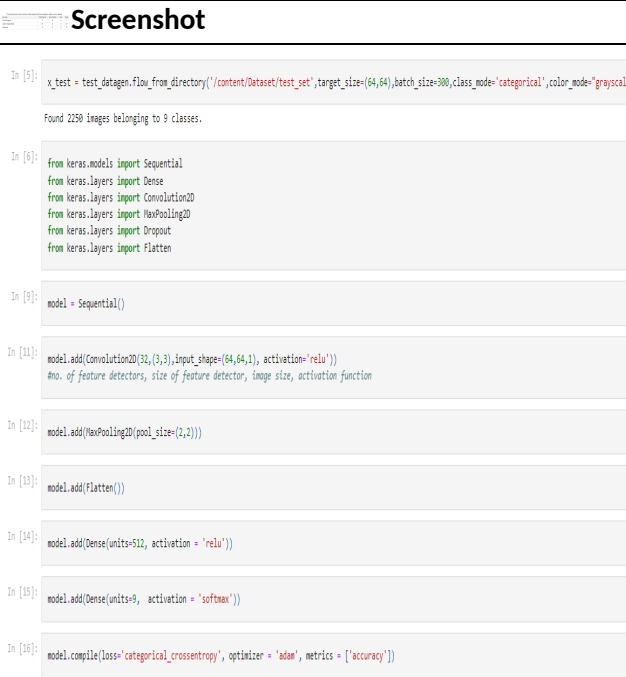
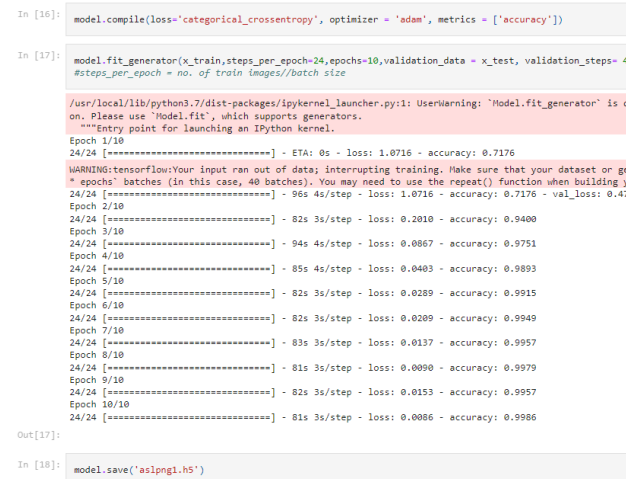
8. TESTING:

8.1 USER TESTCASES:

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

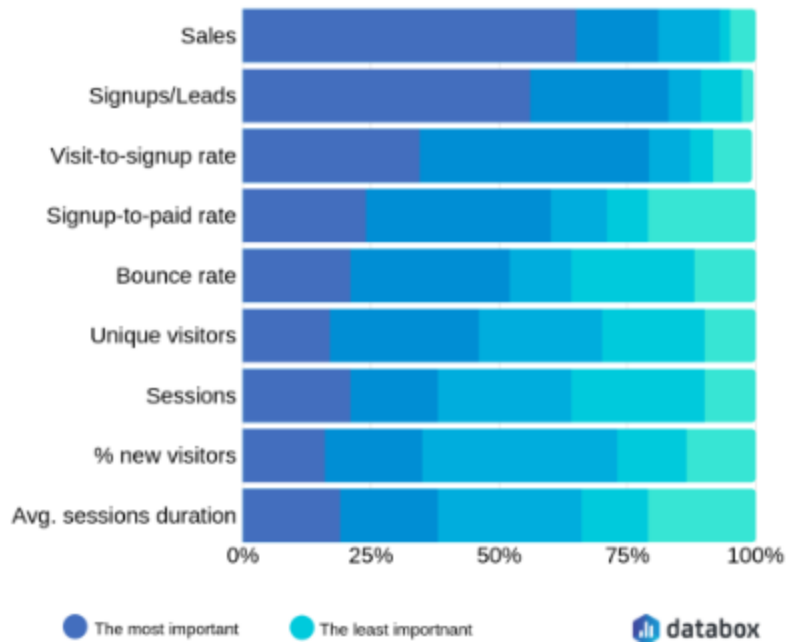
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

8.2 USER ACCEPTANCE TESTING:

S.No.	Parameter	Values	Screenshot
1	Model Summary		 <pre> In [5]: x_test = test_datagen.flow_from_directory('/content/Dataset/test_set', target_size=(64,64), batch_size=300, class_mode='categorical', color_mode='grayscale') Found 2250 images belonging to 9 classes. In [6]: 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 In [9]: model = Sequential() In [11]: model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu')) #no. of feature detectors, size of feature detector, image size, activation function In [12]: model.add(MaxPooling2D(pool_size=(2,2))) In [13]: model.add(Flatten()) In [14]: model.add(Dense(units=512, activation = 'relu')) In [15]: model.add(Dense(units=9, activation = 'softmax')) In [16]: model.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy']) </pre>
2	Accuracy	<p>Training Accuracy – 99.6%</p> <p>Validation Accuracy – 98.3%</p>	 <pre> In [16]: model.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy']) In [17]: model.fit_generator(x_train, steps_per_epoch=24, epochs=10, validation_data = (x_test, validation_steps= 40)) #steps_per_epoch = no. of train images//batch size /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: 'Model.fit_generator' is deprecated. Please use 'Model.fit', which supports generators. ***Entry point for launching an IPython kernel. Epoch 1/10 24/24 [=====] - ETA: 0s - loss: 1.0716 - accuracy: 0.7176 WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator "epochs" batches (in this case, 40 batches). You may need to use the repeat() function when building your 24/24 [=====] - 96s 46/step - loss: 1.0716 - accuracy: 0.7176 - val_loss: 0.4701 Epoch 2/10 24/24 [=====] - 82s 3s/step - loss: 0.2010 - accuracy: 0.9400 Epoch 3/10 24/24 [=====] - 94s 4s/step - loss: 0.0867 - accuracy: 0.9751 Epoch 4/10 24/24 [=====] - 85s 4s/step - loss: 0.0403 - accuracy: 0.9893 Epoch 5/10 24/24 [=====] - 82s 3s/step - loss: 0.0289 - accuracy: 0.9915 Epoch 6/10 24/24 [=====] - 83s 3s/step - loss: 0.0209 - accuracy: 0.9949 Epoch 7/10 24/24 [=====] - 83s 3s/step - loss: 0.0137 - accuracy: 0.9957 Epoch 8/10 24/24 [=====] - 81s 3s/step - loss: 0.0090 - accuracy: 0.9979 Epoch 9/10 24/24 [=====] - 82s 3s/step - loss: 0.0153 - accuracy: 0.9957 Epoch 10/10 24/24 [=====] - 81s 3s/step - loss: 0.0086 - accuracy: 0.9986 Out[17]: In [18]: model.save('as1png1.h5') </pre>

9. RESULTS:

9.1 PERFORMANCE METRICS:

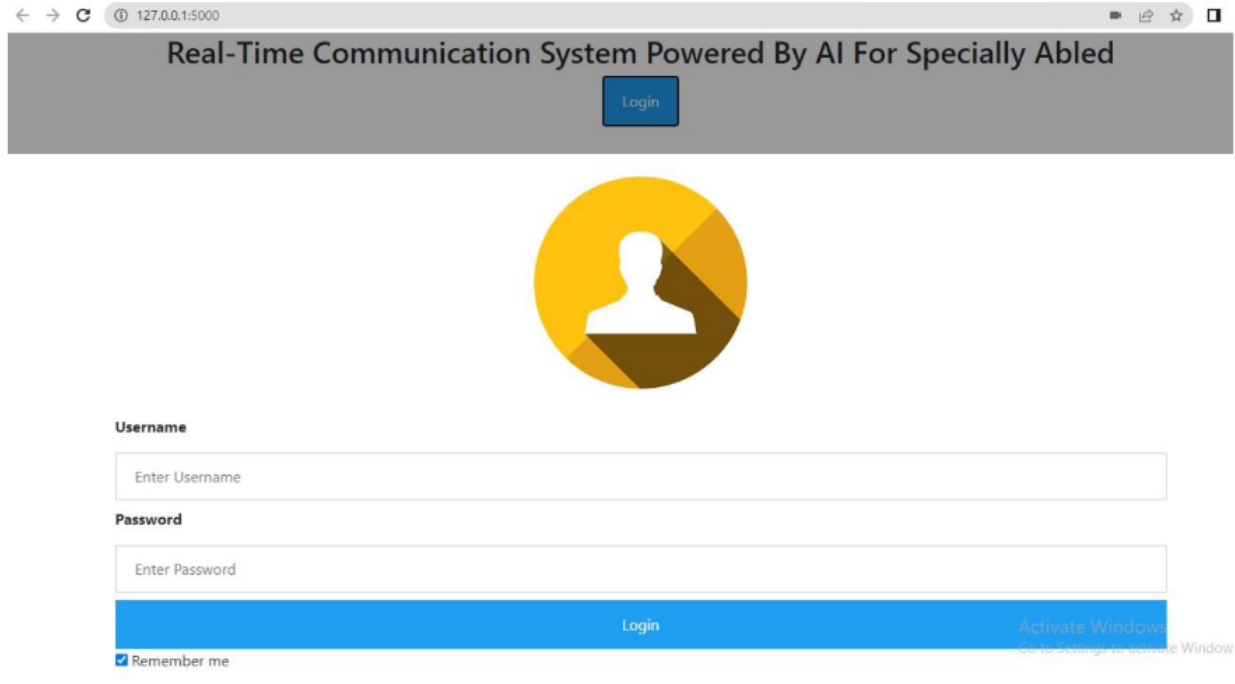


OUTPUT:

← → ↻ ⓘ 127.0.0.1:5000

Real-Time Communication System Powered By AI For Specially Abled

Login



Username

Password

Login

☒ Remember me

Activate Windows
Go to Settings to activate Windows

10. ADVANTAGES :

1. Cost of using the application is simpler
2. Easy user Interface
3. As it is simpler to use, specially abled can use this application in public easier
4. High performance speed with little to no margin of error

DISADVANTAGES:

1. Only available smartphones, so people need to have one to use this application
2. The application needs proper maintenance or upgrade per month for easy use.

11. CONCLUSION:

Thus the real time communication sysem powered by AI for specially abled application has been develeoped and their perfomances were tested and output has been verified successfully.

12. FUTURE SCOPE:

1. As the application is allowed of proper upragadation of latest model, we can develop the model into even more enhanced one with latest technologies
2. In the future we can use this application anywhere in the world as everything is now becoming digitalised.
3. This application can even include various medical supports needed by the specially abled.

13 APPENDIX:

SOURCE CODE:

DATA COLLECTION:

MODEL BUILDING:

1. ADDING OF CONVOLUTION LAYER:

```
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)

x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set', target_si
ze=(64, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 15750 images belonging to 9 classes.

x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set', target_size=(6
4, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 2250 images belonging to 9 classes.

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

model = Sequential()

model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function

```

2. ADDING OF FLATTEN LAYER:

```

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)

x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set', target_si
ze=(64, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 15750 images belonging to 9 classes.

x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set', target_size=(6
4, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 2250 images belonging to 9 classes.

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

model = Sequential()

model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function

model.add(MaxPooling2D(pool_size=(2, 2)))

```



```
model.add(Flatten())
```

3. ADDING OF POOLING LAYER:

```
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)
```

```
x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set', target_si
ze=(64, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 15750 images belonging to 9 classes.
```

```
x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set', target_size=(6
4, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 2250 images belonging to 9 classes.
```

```
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
```

```
model = Sequential()
```

```
model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

4. ADDING OF DENSE LAYER:

```
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)
```

```
x_train =
```

```
train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 15750 images belonging to 9 classes.
```

```
x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 2250 images belonging to 9 classes.
```

```
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
```

```
model = Sequential()
```

```
model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
```

```
model.add(Dense(units=512, activation = 'relu'))
```

```
model.add(Dense(units=9, activation = 'softmax'))
```

5. COMPILATION OF MODEL:

```
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)
```

```
x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 15750 images belonging to 9 classes.
```

```
x_test =
```

```
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64, 64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 2250 images belonging to 9 classes.
```

```
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
```

```
model = Sequential()
```

```
model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
```

```
model.add(Dense(units=512, activation = 'relu'))
```

```
model.add(Dense(units=9, activation = 'softmax'))
```

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

6. FIT AND SAVE THE MODEL:

```
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)
```

```
x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set',target_si
ze=(64, 64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 15750 images belonging to 9 classes.
```

```
x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64, 64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

Found 2250 images belonging to 9 classes.

```
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

model = Sequential()

model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1),
activation='relu'))
#no. of feature detectors, size of feature detector, image size,
activation function

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(units=512, activation = 'relu'))

model.add(Dense(units=9, activation = 'softmax'))

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

model.fit_generator(x_train, steps_per_epoch=24, epochs=10, validation_data =
x_test, validation_steps= 40)
#steps_per_epoch = no. of train images//batch size
```

7. IMPORTING THE REQUIRED MODEL:

```
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)

x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set', target_si
ze=(64, 64), batch_size=300, class_mode='categorical', color_mode="grayscale")
Found 15750 images belonging to 9 classes.

x_test =
```

```
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 2250 images belonging to 9 classes.
```

```
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
```

8. INITIALIZING THE MODEL:

```
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)
```

```
x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 15750 images belonging to 9 classes.
```

```
x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
Found 2250 images belonging to 9 classes.
```

```
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
```

```
model = Sequential()
```

TRAINING OF CNN MODEL IBM:

1.CNN PREDICTION:

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

model=load_model('aslpng1.h5')
img=image.load_img(r'/content/Dataset/test_set/A/28.png',
                    target_size=(64,64))
```

```
img
```



```
x=image.img_to_array(img)
```

```
x.ndim
```

```
x=np.expand_dims(x,axis=0)
```

```
x.ndim
```

```
pred=np.argmax(model.predict(x),axis=1)
1/1 [=====] - 0s 73ms/step
```

```
pred
```

```
array([0])
```

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

2. DOWNLOAD MODEL FROM IBM:

```
pip install ibm_watson_machine_learning
```

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
```

```

        "apikey": "4y7eNmzaeDsCxie0E5b-PACwiQldF2Ock7lM6VAd28Fb"
    }

```

```

client = APIClient(wml_credentials)

```

```

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'])

```

```

space_uid = guid_from_space_name(client, 'model')
print("Space UID : ", space_uid)

```

```

client.set.default_space(space_uid)

```

```

client.repository.download("eec72d86-8ba8-46cc-8588-
c9ff4bf89c85", "aslpng1.tar.gz")

```

3.TRAIN MODEL:

```

pwd

```

```

pip install tensorflow==2.7.1

```

```

from tensorflow.keras.preprocessing.image import ImageDataGenerator

```

```

# Training Datagen

```

```

train_datagen =

```

```

ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, vertical_fli
p=False)

```

```

# Testing Datagen

```

```

test_datagen = ImageDataGenerator(rescale=1/255)

```

```

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='mT4yG1S3H9nBBV3UAwsgkb5FH89r-koWMhH4gnnWTjhN',
                               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-u5ptdjnvogkjlw6'
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/
```

In [7]:

```
# Unzip the Dataset Zip File
```

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

```
%%bash
ls Dataset
test_set
training_set
```

```
# Training Dataset
```

```
x_train=train_datagen.flow_from_directory(r'/home/wsuser/work/Dataset/train
ing_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
# Testing Dataset
```

```
x_test=test_datagen.flow_from_directory(r'/home/wsuser/work/Dataset/test_se
t',target_size=(64,64), class_mode='categorical',batch_size=900)
```


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

```
print("Len x-train : ", len(x_train))  
print("Len x-test : ", len(x_test))  
Len x-train : 18  
Len x-test : 3
```

```
# The Class Indices in Training Dataset  
x_train.class_indices
```

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

Model Creation

```
# Importing Libraries
```

```
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import  
Convolution2D,MaxPooling2D,Flatten,Dense
```

```
# Creating Model
```

```
model=Sequential()  
# Adding Layers  
model.add(Convolution2D(32, (3,3), activation='relu', input_shape=(64, 64, 3)))  
model.add(MaxPooling2D(pool_size=(2,2)))  
model.add(Flatten())
```

```
# Adding Hidden Layers
```

```
model.add(Dense(300, activation='relu'))  
model.add(Dense(150, activation='relu'))
```

```
# Adding Output Layer
```

```
model.add(Dense(9, activation='softmax'))
```

```
# Compiling the Model
```

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

```
# Fitting the Model Generator
```

```
model.fit_generator(x_train, steps_per_epoch=len(x_train), epochs=10, validati  
on_data=x_test, validation_steps=len(x_test))  
model.save('aslpng1.h5')  
# Current accuracy is 0.8154
```

```

# Convert the Saved Model to a Tar Compressed Format
!tar -zcvf trainedModel.tgz aslpng1.h5
aslpng1.h5

%%bash
ls -ll
total 210184
-rw-rw---- 1 wsuser wscommon 111324760 Nov  9 13:49 aslpng1.h5
drwxrwx--- 4 wsuser wscommon      4096 Nov  9 13:34 Dataset
-rw-rw---- 1 wsuser wscommon 103895281 Nov  9 13:49 trainedModel.tgz

#WATSON MODEL:

!pip install watson-machine-learning-client --upgrade

from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "4y7eNmzaeDsCxie0E5b-PACwiQldF2Ock7lM6VAd28Fb"
}

client = APIClient(wml_credentials)
Save to Deployment Space

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'])

space_uid = guid_from_space_name(client, 'model')
print("Space UID : ", space_uid)
Space UID : 7ea8348c-8baa-4f9a-bac8-f21015c4bc86

client.set.default_space(space_uid)

client.software_specifications.list()

software_spec_uid =
client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid

model_details = client.repository.store_model(model='trainedModel.tgz',

```

```

meta_props={
    client.repository.ModelMetaNames.NAME: "CNN",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
    client.repository.ModelMetaNames.TYPE: "tensorflow_2.7"})
model_id = client.repository.get_model_id(model_details)

```

In [32]:

```

model_id
client.repository.download(model_id, 'aslpng1.tar.gz')
Successfully saved model content to file: 'aslpng1.tar.gz'

```

PROJECT STRUCTURE:

Name	Date modified	Type	Size
Dataset	30-10-2022 21:44	File folder	
static	30-10-2022 21:44	File folder	
templates	30-10-2022 21:53	File folder	
aslpng1.h5	30-10-2022 21:54	H5 File	0 KB
output	30-10-2022 21:56	AVI File	0 KB
Test.ipynb	30-10-2022 21:56	IPYNB File	0 KB
text	30-10-2022 21:57	MP3 File	0 KB
Train.ipynb	30-10-2022 21:57	IPYNB File	0 KB
webstreaming	30-10-2022 21:58	Python File	0 KB

SPRINT 1:

DATASET:










..		
 A	Add files via upload	6 days ago
 B	files added	6 days ago
 C	Add files via upload	6 days ago
 D	Add files via upload	6 days ago
 E	Add files via upload	6 days ago
 F	Add files via upload	6 days ago
 G	Add files via upload	6 days ago
 H	Add files via upload	6 days ago
 I	Add files via upload	6 days ago

IMAGE PREPROCESSING:

1.APPLY IMAGE DATA GENERATOR:

```

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Training Datagen
train_datagen =
ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, vertic
al_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)

import tensorflow as tf
import os
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 IPython.display as display
from PIL import Image

```

```
import pathlib
```

```
!unzip '/content/drive/MyDrive/dataset/Dataset.zip'
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from google.colab import drive
drive.mount('/content/drive')
Drive already mounted at /content/drive; to attempt to forcibly remount,
call drive.mount("/content/drive", force_remount=True).
```

In []:

```
train_datagen =
ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True,
vertical_flip=False)
test_datagen= ImageDataGenerator(rescale=1./255)
```

In []:

```
x_train =
train_datagen.flow_from_directory('/content/Dataset/training_set', target_si
ze=(64,64), batch_size=300,
                                class_mode='categorical',
color_mode = "grayscale")
Found 15750 images belonging to 9 classes.
```

In []:

```
x_train.class_indices
```

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

```
x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set', target_size=(6
4,64), batch_size=300,
                                class_mode='categorical',
color_mode = "grayscale")
Found 2250 images belonging to 9 classes.
```

```
x_test.class_indices
```

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

SPRINT 2:

1. MODEL TESTING:

[]



```
x=image.img_to_array(img)
```

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

```
[ ] y=np.argmax(model.predict(x),axis=1)
```

```
1/1 [=====] - 0s 56ms/step
```

```
[ ] y
```







```
array([1])
```

```
[ ] index=['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']
```

```
[ ] index[y[0]]
```

```
'B'
```

SPRINT 3:

 Import_The_Packages_And_Load_The_Saved_Model.ipynb	Add files via upload	6 days ago
 Load_The_Test_Image_Pre_Process_It_And_Predict.ipynb	Add files via upload	6 days ago
 Model_Building_And_Test_The_Data.ipynb	Add files via upload	2 days ago
 OpenCV.ipynb	Add files via upload	2 days ago
 Test_&_Training_Model_Data.ipynb	Add files via upload	2 days ago
 Testing_the_model.ipynb	Add files via upload	2 days ago

SPRINT 4:

MODEL BUILDING:

```
!pip install Keras==2.2.4
!pip install tensorflow==2.7
```

```
Requirement already satisfied: Keras==2.2.4 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (2.2.4)
Requirement already satisfied: h5py in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (3.2.1)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (1.1.2)
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (1.20.3)
Requirement already satisfied: pyyaml in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (5.4.1)
Requirement already satisfied: keras-applications>=1.0.6 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (1.0.8)
Requirement already satisfied: scipy>=0.14 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (1.7.3)
Requirement already satisfied: six>=1.9.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from Keras==2.2.4) (1.15.0)
```

2.]IMPORTING LIBRARIES TO BUILD MODEL.

```
#Library to train the model
import keras
import tensorflow

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D, Flatten
```

4.]ADDING STREAMING_BODY_OBJECT FOR DATASET.ZIP

```
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='aqrH2FuH38ECUn869hHk4qyvS_iKJfrZAMUJJQ-mQKx',
                              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 = 'realtimecommunicationforspecially-donotdelete-pr-rfqndcvwgch6fu'
object_key = 'Dataset.zip'

streaming_body_4 = 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/
```


5.]UNZIPPING THE DATASET

```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_4.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

```
pwd
```

```
'/home/wsuser/work/Dataset'
```

```
#checking that the dataset is there are not
import os
filenamer = os.listdir('/home/wsuser/work/Dataset/training_set')
```

6.]TRAINING AND TESTING IMAGES UNDER CLASSES

```
x_train=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.

```
x_test=test_datagen.flow_from_directory("/home/wsuser/work/Dataset/test_set",target_size=(64,64),
class_mode='categorical', batch_size=25)
```

7.]TOTAL CLASSES UNDER TRAINING AND TESTING.

```
In [ ]: x_train.class_indices
```

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

```
In [ ]: x_test.class_indices
```

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

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

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

8.]MODEL BUILDING USING CNN

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

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

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

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

```
model.summary()
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 31, 31, 32)	0
flatten_1 (Flatten)	(None, 30752)	0

total params: 896
trainable params: 896
non-trainable params: 0

9.]ADDING LAYERS FOR MODEL TRAINING.

HIDDEN LAYERS

```
model.add(Dense(units = 300, activation='relu'))  
#model.add(Dense(units = 150, init = "uniform", activation='softmax'))
```

OUTPUT LAYERS

```
model.add(Dense(units = 5, activation='softmax'))
```

10.]OPTIMIZING THE MODEL

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

```
[ ]: len(x_train)
```

```
t[ ]: 630
```

```
[ ]: len(x_test)
```

```
t[ ]: 90
```

11.]FITTING THE MODEL

```
[ ]: #model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
# Fitting the Model Generator
model.fit_generator(x_train,steps_per_epoch=630,epochs=1,validation_data=x_test,validation_steps=90)
#model.fit(x_train, epochs=100, verbose=1)
```

12.]SAVING THE MODEL

```
: ls
Dataset/ test_set/ training_set/

: pwd

'/home/wsuser/work/Dataset'

: model.save('Dataset.h5')
```

13.]CONVERTING ZIP FILE TO TAR FILE FOR LOCAL USE.

```
: #converting the model to tar
!tar -zcvf image.Classification.model_new.tgz Dataset.h5

Dataset.h5

: ls -l

Dataset/
Dataset.h5
image.Classification.model_new.tgz
test_set/
training_set/
```

14.]INSTALLING WATSON MACHINE LEARNING CLIENT SOFTWARE

15.]IMPORTING APICLIENT FOR DEPLOYING.

```
] : from ibm_watson_machine_learning import APIClient
url_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "sqLVTXSP3nnAKfzJ1rKRKCpNzS_XZ8_HXa9FRwV7BvOP"
}
client = APIClient(url_credentials)
```

```
] : client = APIClient(url_credentials)
```

16.]CREATING API_CLIENT SPACE ID.

```
] : 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_
```

```
] : space_uid = guid_from_space_name(client, 'Image Classification')
print("space UID = " + space_uid)
```

space UID = d90f421e-9169-47e7-a58c-0e7bb0e65685

17.]STORING THE MODEL_ID FOR DATASET.H5

```
In [ ]: #store the model
model_details = client.repository.store_model(model='Image-classification-model_new.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:'CNN',
    client.repository.ModelMetaNames.TYPE:'keras_2.2.4',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}
)
model_id = client.repository.get_model_uid(model_details)
```

```
In [ ]: model_id
```

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

18.]DOWNLOADING THE TAR FILE ON CLIENT REPOSITORY

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

19.]TEST THE MODEL

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

20.]LOADING THE DATASET

20.]LOADING THE DATASET

```
#Load the model
model=load_model('Dataset.h5')
```

21.]ADDING STREAMING_BODY FOR TEST IMAGE.

```
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='aqprHZFuH38ECUn869hHk4qyvS_iKJfrZAMUJJQ-mQKx',
                              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 = 'realtimecommunicationforspecially-donotdelete-pr-rfqndcvwgch6fu'
object_key = '1.png'

streaming_body_5 = 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 []:

```
img
```

In []:

```
img1=image.load_img(r"/home/wsuser/work/Dataset/test_set/C/1.png")
```

In []:

```
img1
```

In []:

```
x=image.img_to_array(img)
```

In []:

```
x
```

In []:

```
x1=np.expand_dims(x,axis=1)
```

In []:

```
x1
```

In []:

```
y=np.argmax(model.predict(x),axis=1)
```

In []:

```
y
```

In []:

```
x_train.class_indices
```

In []:

```
index=['A','B','C','D','E','F','G','H','I']
```

In []:

```
index[y[0]]
```



```
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=fnp.argmax(model.predict(x),axis=1)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

```
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)
index=['A','B','C','D','E','F','G','H','I']
index[y[0]]
```

```
img=image.load_img(r"/content/drive/MyDrive/IBM_PROJECT/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]]
```

```
img=image.load_img(r"/content/drive/MyDrive/IBM_PROJECT/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]]
```

```
!tar -zcvf Dataset-classification-model.tgz specially.h5
```

```
import tensorflow as tf
tf.__version__
```

```
In [ ]: !tar -zcvf Dataset-classification-model.tgz specially.h5
```

```
In [ ]: import tensorflow as tf
        tf.__version__
```

```
In [ ]: !pip install keras == 2.2.4
```

23.]IBM DEPLOYMENT

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

```
In [ ]: from ibm_watson_machine_learning import APIClient
        wml_credentials={
            "url":"https://us-south.ml.cloud.ibm.com",
            "apikey":"x91C3TUTrrIfLvrXsKf8yLyI1KHb3JV0Y7Qrwy1zilb2"
        }
        client=APIClient(wml_credentials)
```

CLIENT

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

```
In [ ]: space_uid=guid_space_name(client,'animal_deploy')
        print("Space UID "+space_uid)
```

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

```
In [ ]: space_uid=guid_space_name(client, 'animal_deploy")
        print("Space UID "+space_uid)
```

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

```
In [ ]: client.software_specifications.list(200)
```

```
In [ ]: software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
```

```
In [ ]: software_space_uid
```

```
In [ ]: model_details=client.repository.store_model(model='Dataset.tgz', meta_props={
        client.repository.ModelMetaNames.NAME: "CNN Model Building",
        client.repository.ModelMetaNames.TYPE: 'tensorflow_2.7',
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_space_uid
    })
```

```
In [ ]: model_id=client.repository.get_model_id(model_details)
```

```
In [ ]: model_id
```

APPLICATION BUILDING:

MAIN.PY:

```
import
cv2

        video = cv2.VideoCapture(0)

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

```

video.release()
cv2.destroyAllWindows()

```

CAMERA.PY

```

import
cv2

```

```

import numpy as np
from keras.models import load_model
from keras.utils import load_img, img_to_array

class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.roi_start = (50, 150)
        self.roi_end = (250, 350)
        self.model = load_model('G:\IBM\Conversation engine for deaf and dumb\asl1
Model
        # self.model = load_model('IBM_Communication_Model.h5') # Execute IBM Tra
        self.index=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
        self.y = None
    def __del__(self):
        self.video.release()
    def get_frame(self):
        ret, frame = self.video.read()
        frame = cv2.resize(frame, (640, 480))
        copy = frame.copy()
        copy = copy[150:150+200, 50:50+200]
        # Prediction Start
        cv2.imwrite('image.jpg', copy)
        copy_img = load_img('image.jpg', target_size=(64,64))
        x = img_to_array(copy_img)
        x = np.expand_dims(x, axis=0)
        pred = np.argmax(self.model.predict(x), axis=1)
        self.y = pred[0]
        cv2.putText(frame, 'The Predicted Alphabet is:
'+str(self.index[self.y]), (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 3)
        ret, jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()

```

WEBSTREAMING.PY:

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

```
from camera import Video
```

```
app = Flask(__name__)
```

```
@app.route('/')  
def index():
```

```
    return render_template('index.html')
```

```
def gen(camera):
```

```
    while True:
```

```
        frame = camera.get_frame()
```

```
        yield(b'--frame\r\n'
```

```
              b'Content-Type: image/jpeg\r\n'
```

```
              b'\r\n\r\n')
```

```
@app.route('/video_feed')
```

```
def video_feed():
```

```
    video = Video()
```

```
    return Response(gen(video), mimetype='multipart/  
frame')
```

```
if __name__ == '__main__':
```

```
    app.run()
```

STYLES.CSS:

```
body
```

```
{
```

```
    background-image: linear-gradient(to top, rgba(0,0,0,0.3) , rgba(0, 0, 0, 0.6)),url(../img/  
illustration-vector.jpg);
```

```
    background-size:cover;
```

```
    background-position:top;
```

```
    background-repeat: no-repeat;
```

```
    margin: 0;
```

```

        height: 100vh;
        overflow: hidden;
        padding: 0;
        text-align: center;
    }
    .myvideo {
        background-color: floralwhite;
        margin-top: 10vh;
        align-content: center;
    }
    .title{
        text-align: center;
        color: white;
        margin-top: 10vh;

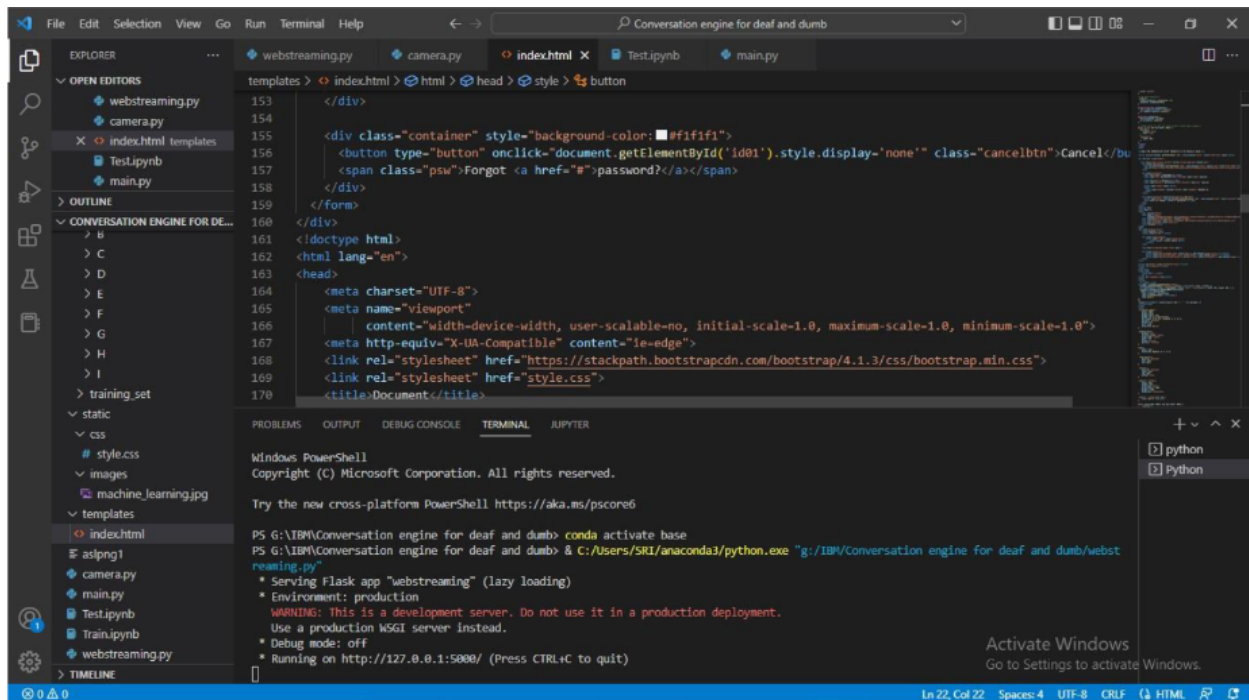
    }

    h2{
        color: white;
        text-decoration: none;
        font-size: 25px

    }
    a{
        color: white;
        font-size: 20px;
    }
    a:hover{
        transform: scale(1.1);
        transition: transform 0.3s;
        transition-duration: transform 0.2s;
    }
    p{
        color: aliceblue;
        align-content: center;
        margin: 7vw;
        font-size: 22px
    }
}

```

HTML:



GITHUB LINK: <https://github.com/IBM-EPBL/IBM-Project-3798-1658638384>

DEMO LINK:

https://drive.google.com/file/d/1iWVSaGQ1kQ9G9Pe6_KCEfkuNyvo1bKAV/view?usp=sharing