Government college of engineering Sengipatti

Thanjavur

Real-Time Communication System Powered By AI For Specially Abled

TEAM ID : PNT2022TMID47262

TEAM MEMBERS :

1. Dharshini.D (822719106013)
2. Yashini.K (822719106049)
3. DurgaDevi.A (822719106014)
4. SubhaDharshini.G (822719106043)
5. Anusaya.B.G (822719106004)

CONTENTS

1. INTRODUCTION
1.1 Abstract
1.2 Project Overview
2. LITERATURE SURVEY
2.1 Existing problem
2.2 References
2.3 Problem Statement Definition
3. IDEATION & PROPOSED SOLUTION
3.1 Empathy Map Canvas
3.2 Ideation & Brainstorming
3.3 Proposed Solution
3.4 Problem Solution fit
4. REQUIREMENT ANALYSIS
4.1 Functional requirement
4.2 Non-Functional requirements
5. PROJECT DESIGN
5.1 Data Flow Diagrams
5.2 Solution & Technical Architecture
5.3 User Stories
6. PROJECT PLANNING & SCHEDULING
6.1 Sprint Planning & Estimation
6.2 Sprint Delivery Schedule
6.3 Reports from JIRA
7. CODING & SOLUTIONING (Explain the features added in the project along withcode)
8. TESTING
8.1 Test Cases
8.2 User Acceptance Testing
9. RESULTS
9.1 Performance Metrics
10.ADVANTAGES & DISADVANTAGES
11.CONCLUSION
12.FUTURE SCOPE
13.APPENDIX
Source Code & GitHub Link

1. INTRODUCTION

1.1 ABSTRACT

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

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 PROJECT OVERVIEW

Deaf and dumb people are humans at the deepest psychological level. Many of these people are not even exposed to sign languages and it is observed that it gives a great relief on a psychological level, when they find out about signing to connect themselves with others by expressing their love or emotions. About 5% population in world are suffering from hearing loss. Deaf and dumb people use sign language as their primary means to express their thoughts and ideas to the people around them with different hand and body gestures. There are only about 250 certified sign language interpreters in India for a deaf population of around 7 million. In this work, the design of prototype of an assistive device for Deaf-mute people is presented so as to reduce this communication gap with the normal people. This device is portable and can hang over the neck. This device allows the person to communicate with sign hand postures in order to recognize different gestures based signs. The controller of this assistive device is developed for processing the images of gestures by employing various imageprocessing techniques and deep learning models to recognize the sign. This sign is converted into speech in real-time using text-to-speech module.

2. LITERATURE SURVEY

2.1 Existing problem :

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.

According to the World Health Organization, the world population experiencing hearing and speech challenges approximates over 466 million people globally [1,2]. With such disability, instead unequally distributed resources, these people are vulnerable to discrimination [3]. The fact that every human being, abled or disabled, is entitled to a good life with equal opportunities calls for affirmative action [4,10]. This society requires attention from all quarters, especially on technological enhancement, to ensure the disabled get a comfortable life [5,16,20]. With the number increasing significantly, something needs to be done. The deaf and dumb are introverts, remaining engraved in their thoughtful world. Communication, which is essential in human life, is challenging. Humans are social beings, and effective communication is necessary [6, 22,24]. The development of technology should, therefore, serve to improve their lives as well [7]. The introduction of an application that can be used by the deaf and dumb will be a great innovation. It will not only make life easier but will as well increase their life opportunities, including employability. The deaf and dumb category must be involved within technology on PC experience as they involved in technology on smartphones. D- talk application provides this experience for them by reading their hand movements and displays a certain function.

2.2 REFRENCES:

1. Anderson, R., Wiryana, F., Ariesta, M. C., & Kusuma, G. P. (2017). Sign language recognition application systems for deaf-mute people: A review based on input-process-output. Procedia computer science, 116, 441-448.
2. Raheja, J. L., Mishra, A., & Chaudhary, A. (2016). Indian sign language recognition using SVM. Pattern Recognition and Image Analysis, 26(2), 434-441.

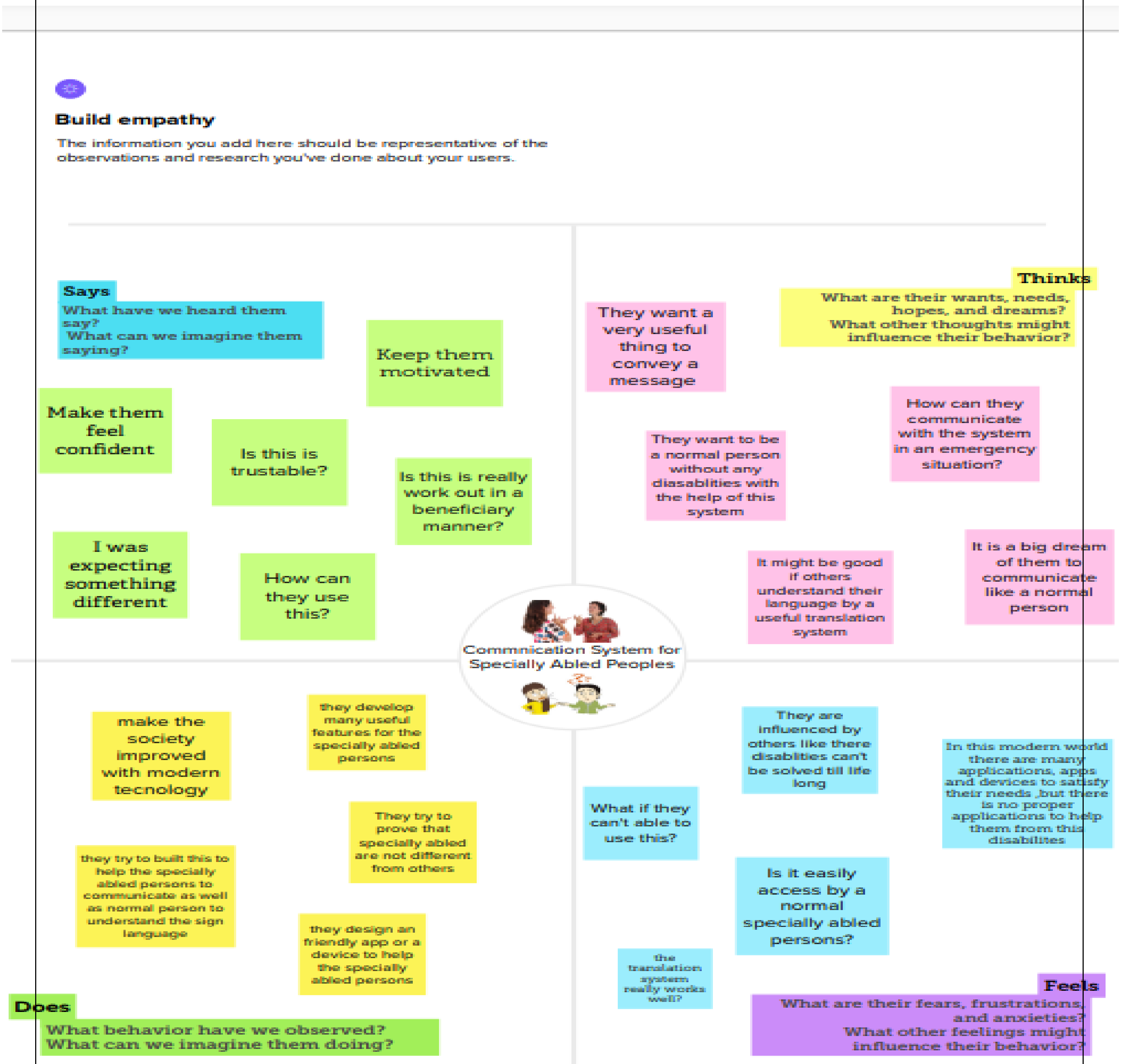
3. Jin, C. M., Omar, Z., &Jaward, M. H. (2016, May). A mobile application of American sign language translation via image processing algorithms. In 2016 IEEE Region 10 Symposium (TENSYMP) (pp. 104109). IEEE
4. Neiva, D. H., &Zanchettin, C. (2018). Gesture recognition: a review focusing on sign language in a mobile context. Expert Systems with Applications, 103, 159-183.

2.3 Problem Statement Definition :

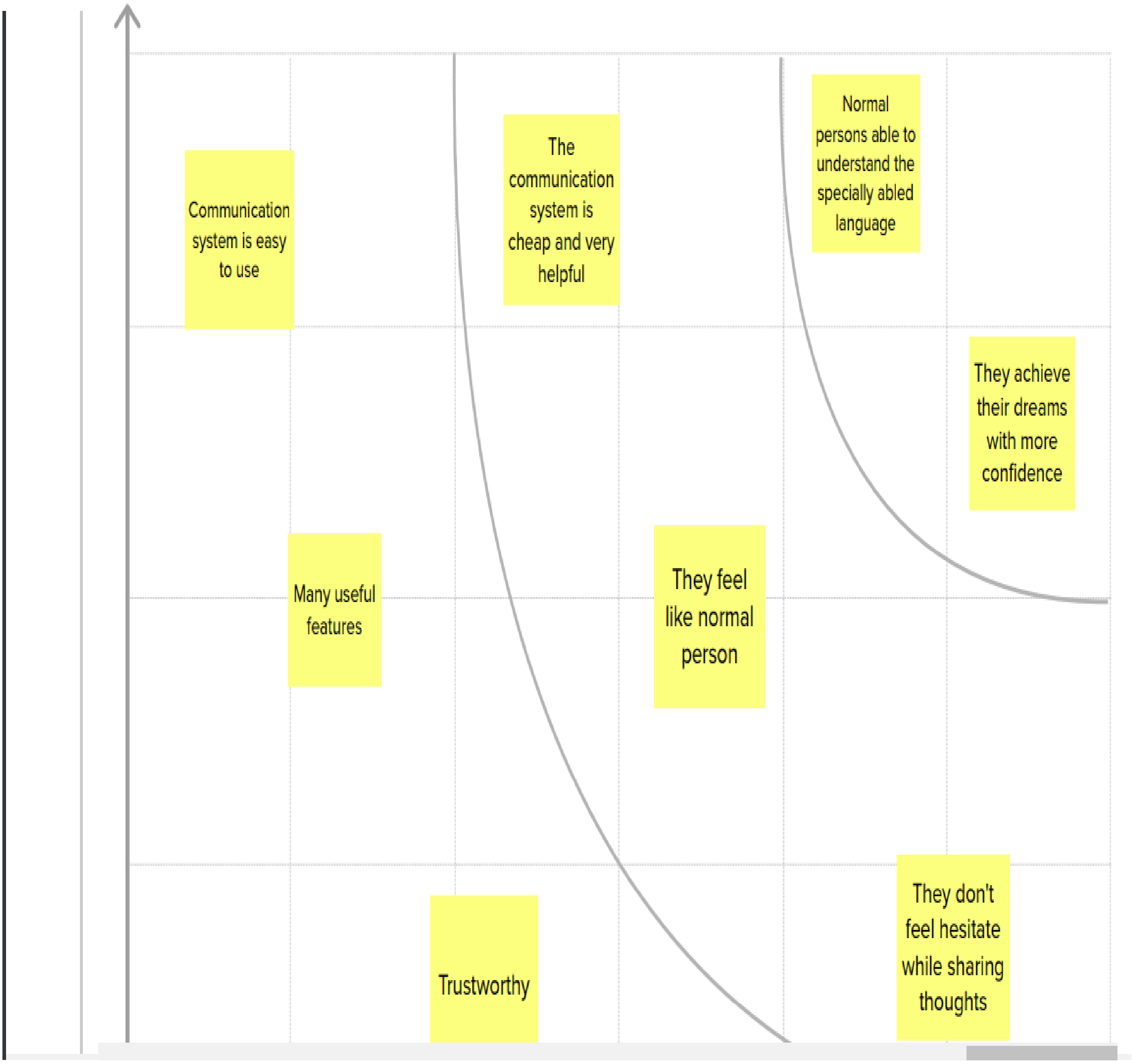
The image recognition process is a process that enables the input of the sign language into the application for necessary processing [20,31,46]. The process requires a sign to be made in front of the webcam. The computer captures the sign made via the webcam and stores the different images made. Images that come from the camera will be resized, and the resolution will change. The colors will change to grayscale image and then to black and white images while editing the images [25, 33,47]. There several techniques used to extract the image, such as SIFT, SURF, BRISK, and HSV algorithms.Compared to standard algorithms, neural networks can solve somewhat complicated issues at a much easier level about the complexity of algorithms. Neural networks can solve somewhat complicated issues at a much easier level concerning the complexity of algorithms [26, 30]. The neural network builds to mimic human brain neural function but with the mathematical functions [31, 33,38].The training phase was based on storing the images in the database. The database contained images of hands, both men and women. The training was based on identifying all possible signs that can be made using one hand. For this purpose, 30 different images with different levels of lights and duration were captured and stored in the database. These images were used as training images that will help in making the right decision for the tasks. The database contained over 1000 images of unique hands and signs.

3.IDEATION & PROPOSED SOLUTION

3.1Empathy Map Canvas



3.2 Ideation & Brainstorming



3.3 Proposed Solution

S. No:	Parameter	Description
1.	Problem Statement (Problem to besolved)	The sixth sense is a multi-platform app for aiding the people in need that is people who are handicapped in the form of lack of speech (dumb), lack of hearing (deaf), lack of sight (blind), lack of judicial power to differentiate between objects (visual agnosia) and people suffering from autism (characterized by great difficulty in communicating and forming relationships with other people and in using language and abstract concepts). Our current implementation of the product is on two platforms, namely, mobile and a web app.
2.	Idea / Solution description	The current implementation deals with object recognition and text to speech and a speech to text converter. The speech to text converter and text to speech converter utilized the Web Speech API (Application Program Interface) for the website and text to speech and speech to text library for the mobile platform. The object recognition wouldn't fetch enough use out of a website. Hence, it has been implemented on the mobile app utilizing the Firebase ML toolkit and different pre-trained models, which are both available offline as well as online.

3.	Novelty / Uniqueness	The world does not want just machine to do what they are told but even expect devices to work like us. Machine learning, a subsection of AI, is the hottest technology right now being implemented daily basis and is to be supposed to reach its peak in the next decade. Now, as the world has become a better place by providing everything at ease through technology to humans, we need to utilize the technology for differently-abled people.
4.	Social Impact / Customer Satisfaction	Thus, customer satisfaction has benefits as it helps minimize extra costs, enables industry know their repeat customer better, which could help in improving future service.Higher accuracy could be achieved in the future scope of the implementation through the use of custom models for object detection and text recognition as it could take into account the cases of objects for differently-abled people and work on those only yielding faster and accurate results.
5.	Business Model (Revenue Model)	The major contribution of the work is: • Integration of multiple modules to provide a single application to aid people of different disabilities. • All the modules are researched solely rather than have a single source for all. This is the aim of work being performed in this work. • An innovative approach for text to speech is implemented to provide a faster and convenient approach for mute to communicate through SAM (Speech Assisted for Mute).

6.	Scalability of the Solution	The text to speech and Speech to Text Engine was based on the Google Speech Engine. IT has a similar implementation, as discussed in the website counterpart. All these sections have been then integrated into a single application to provide a single solution for all.

3.4 Problem Solution fit

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	Authentication through Facial recognition Authentication through Password authentication protocol
FR-4	External interfaces	Robots and other tools provide home-based care 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 augmentation and Practical accuracy are responsible for AI business rules.

4.2 NON FUNCTIONAL REQUIREMENTS:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Provides personalised learning experiences tailored to 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**	Technology 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 the environments.

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Milestone Activity Plan.

Use the below template to create product backlog and sprint schedule

Milestone	Function (Epic)	Milestone Story Number	Story / Task
Milestone -1	Data Collection	M1	we're collecting dataset for building our project and creating two folders, one for training and another one for testing.
Milestone-2	Image preprocessing	M2	Importing image data generator libraries and applying image data generator functionality to train the test set.
Milestone-3	Model Building	M3	Importing the model building libraries, Initializing the model, Adding Convolution layers, Adding the Pooling layers, Adding the Flatten layers, Adding Dense layers, Compiling the model Fit and Save the model.
Milestone-4	Testing the model	M4	Import the packages first. Then we save the model and Load the test image, preprocess it and predict it.
Milestone-5	Application layer	M5	Build the flask application and the HTML pages.
Milestone-6	Train CNN model	M6	Register for IBM Cloud and train Image Classification Model.
Milestone-6	Final result	M7	To ensure all the activities and resulting the final output.

:

6.2 Sprint Delivery Schedule :

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	CollectDataset.	9	High	CHETHAN P GOKUL P
Sprint-1		USN-2	Image preprocessing	8	Medium	CHARLES J PUGAZAHANDHI S
Sprint-2	Model Building	USN-3	Import the required libraries, add the necessary layers and compile the model	10	High	CHETHAN P PUGAZAHANDHI S
Sprint-2		USN-4	Training the image classification model using CNN	7	Medium	GOKUL P CHARLES J
Sprint-3	Training and Testing	USN-5	Training the model and testing the model's performance	9	High	CHETHAN P GOKUL P
Sprint-4	Implementation of the application	USN-6	Converting the input sign language images into English alphabets	8	Medium	CHARLES J PUGAZHANDHI S
Sprint-4	Test the model	USN-8	Model testing	8	Medium	Madhumitha

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Model Building

Importing The Required Model Building Libraries

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

In [ ]:
```
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
dataset = pd.read_csv('E:\Datasets\Mall_Customers.csv')
```

Initializing The Model

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

In [ ]:
```
spatial_dropout=0.05
recurrent_dropout=0.1
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

Adding The Convolution Layer

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

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

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
# Let img1 be an image with no features
img1 = np.array([np.array([200, 200]), np.array([200, 200])])
img2 = np.array([np.array([200, 200]), np.array([0, 0])])
img3 = np.array([np.array([200, 0]), np.array([200, 0])])

kernel_horizontal = np.array([np.array([2, 2]), np.array([-2, -2])])
print(kernel_horizontal, 'is a kernel for detecting horizontal edges')

kernel_vertical = np.array([np.array([2, -2]), np.array([2, -2])])
print(kernel_vertical, 'is a kernel for detecting vertical edges')
```

In [ ]:
```
# We will apply the kernels on the images by
# elementwise multiplication followed by summation
def apply_kernel(img, kernel):
    return np.sum(np.multiply(img, kernel))

# Visualizing img1
plt.imshow(img1)
plt.axis('off')
plt.title('img1')
plt.show()

# Checking for horizontal and vertical features in image1
print('Horizontal edge confidence score:', apply_kernel(img1,
                                            kernel_horizontal))
print('Vertical edge confidence score:', apply_kernel(img1,
                                            kernel_vertical))
```

In [ ]:
```
# Visualizing img2
plt.imshow(img2)
plt.axis('off')
plt.title('img2')
plt.show()

# Checking for horizontal and vertical features in image2
print('Horizontal edge confidence score:', apply_kernel(img2,
                                            kernel_horizontal))
print('Vertical edge confidence score:', apply_kernel(img2,
                                            kernel_vertical))
```

In [ ]:
```
# Visualizing img3
plt.imshow(img3)
plt.axis('off')
plt.title('img3')
plt.show()

# Checking for horizontal and vertical features in image3
print('Horizontal edge confidence score:', apply_kernel(img3,
                                            kernel_horizontal))
print('Vertical edge confidence score:', apply_kernel(img3,
                                            kernel_vertical))
```

In [ ]:

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

```
Len x-train :  18
Len x-test :  3
```

In [ ]:

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

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

Model Creation

In [ ]:

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

In [ ]:

```
# Creating Model
model=Sequential()
```

In [ ]:

```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

Adding The Pooling Layer

In [ ]:

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

In [ ]:

```
import numpy as np
from keras.models import Sequential
from keras.layers import MaxPooling2D
```

In [ ]:

```
# define input image
image = np.array([[2, 2, 7, 3],
                                [9, 4, 6, 1],
                                [8, 5, 2, 4],
                                [3, 1, 2, 6]])
image = image.reshape(1, 4, 4, 1)
```

In [ ]:

```
# define model containing just a single max pooling layer
model = Sequential(
        [MaxPooling2D(pool_size = 2, strides = 2)])

# generate pooled output
output = model.predict(image)
```

In [ ]:

```
# print output image
output = np.squeeze(output)
print(output)
```

In [ ]:

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

In [ ]:

```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:

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

```
Len x-train :  18
Len x-test :  3
```

In [ ]:

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

Out[ ]:

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

Model Creation

In [ ]:

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

In [ ]:

```
# Creating Model
model=Sequential()
```

In [ ]:

```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:

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

Adding The Flatten Layer

In [ ]:

```
# importing numpy as np
import numpy as np
```

In [ ]:

```
# declare flatten np
gfg = np.array([[6, 9, 12], [8, 5, 2], [18, 21, 24]])

# using array.flatten() method
flat_gfg = gfg.flatten(order='A')
print(flat_gfg)
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:

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

```
Len x-train :  18
Len x-test :  3
```

In [ ]:

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

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

Model Creation

In [ ]:

```
model = Sequential()
for i, feat in enumerate(args.conv_f):
    if i==0:
        model.add(Conv2D(feat, input_shape=x[0].shape, kernel_size=3, padding = 'same',use_bias=False))
    else:
        model.add(Conv2D(feat, kernel_size=3, padding = 'same',use_bias=False))
        model.add(BatchNormalization())
        model.add(LeakyReLU(alpha=args.conv_act))
        model.add(Conv2D(feat, kernel_size=3, padding = 'same',use_bias=False))
        model.add(BatchNormalization())
        model.add(LeakyReLU(alpha=args.conv_act))
        model.add(Dropout(args.conv_do[i]))
```

In [ ]:

```
model.add(Flatten())

#Input code here

denseArgs = {'use_bias':False}
for i,feat in enumerate(args.dense_f):
    model.add(Dense(feat,**denseArgs))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=args.dense_act))
    model.add(Dropout(args.dense_do[i]))
model.add(Dense(1))
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
# Creating Model
model=Sequential()
```

In [ ]:

```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

Adding The Dense Layers

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

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

In [ ]:
```
print("Adding dense layer on top")
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

In [ ]:
```
print("Complete architecture of the model")
model.summary()
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

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

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

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Compile To The Model

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

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

In [ ]:
```
# Creating sample sourcecode to multiply two variables
# x and y.
srcCode = 'x = 10\ny = 20\nmul = x * y\nprint("mul =", mul)'

# Converting above source code to an executable
execCode = compile(srcCode, 'mulstring', 'exec')

# Running the executable code.
exec(execCode)
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
def compile_model_results(model, root="./"):

    listing = glob.glob(root + '/models/' + model + '/*/best_pars.pkl')

    dic_list = []
    for file in listing:
        tmp = hyper_parameters_load(file)
        dic_list.append(tmp.to_dictionary())

    df = pd.DataFrame(dic_list)
    df['diff'] = df.test_F1 - df.forecast_F1
    df['pci'] = abs(df.test_F1 - df.forecast_F1)

    if not os.path.exists(root + '/figures/' +  model ):
        os.makedirs(root + '/figures/' +  model )

    df.to_csv(root + '/figures/' +  model + '/results.csv', index=False)

    return df
```

In [ ]:
```
# Set optimizer loss and metrics
    opt = Adam(lr=args.initial_lr, beta_1=0.99, beta_2=0.999, decay=1e-6)
    if args.net.find('caps') != -1:
        metrics = {'out_seg': dice_hard}
    else:
        metrics = [dice_hard]

    loss, loss_weighting = get_loss(root=args.data_root_dir, split=args.split_num, net=args.net,
                                    recon_wei=args.recon_wei, choice=args.loss)

    # If using CPU or single GPU
    if args.gpus <= 1:
        uncomp_model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)
        return uncomp_model
    # If using multiple GPUs
    else:
        with tf.device("/cpu:0"):
            uncomp_model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)
            model = multi_gpu_model(uncomp_model, gpus=args.gpus)
            model.__setattr__('callback_model', uncomp_model)
        model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)

X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Compilation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

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

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

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [ ]:
```
# reading code from a file
f = open('main.py', 'r')
temp = f.read()
f.close()

code = compile(temp, 'main.py', 'exec')
exec(code)
```

Saving the Model

In [ ]:
```
model.save('asl_model_84_54.h5')
```

Fit And Save The Model

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

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
# Save Model Using Pickle
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
import pickle
```

In [ ]:
```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-
diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

In [ ]:
```
# Fit the model on training set
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)
```

In [ ]:

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

```
Len x-train :  18
Len x-test :  3
```

In [ ]:

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

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

Model Creation

In [ ]:

```python
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:

```python
# Creating Model
model=Sequential()
```

In [ ]:

```python
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:

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

In [ ]:

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

In [ ]:

```python
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:

```python
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [ ]:

```python
# Fitting the Model Generator
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
```

```
Epoch 1/10
18/18 [==============================] - 92s 5s/step - loss: 0.0049 - accuracy: 0.9994 - val_loss: 0.2635 - val_accuracy: 0.9773
Epoch 2/10
18/18 [==============================] - 90s 5s/step - loss: 0.0040 - accuracy: 0.9995 - val_loss: 0.2074 - val_accuracy: 0.9773
Epoch 3/10
18/18 [==============================] - 87s 5s/step - loss: 0.0041 - accuracy: 0.9995 - val_loss: 0.2460 - val_accuracy: 0.9773
Epoch 4/10
18/18 [==============================] - 91s 5s/step - loss: 0.0041 - accuracy: 0.9992 - val_loss: 0.2470 - val_accuracy: 0.9782
Epoch 5/10
18/18 [==============================] - 88s 5s/step - loss: 0.0037 - accuracy: 0.9993 - val_loss: 0.2439 - val_accuracy: 0.9782
Epoch 6/10
18/18 [==============================] - 88s 5s/step - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.2852 - val_accuracy: 0.9782
Epoch 7/10
18/18 [==============================] - 91s 5s/step - loss: 0.0023 - accuracy: 0.9997 - val_loss: 0.2589 - val_accuracy: 0.9782
Epoch 8/10
18/18 [==============================] - 93s 5s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.2523 - val_accuracy: 0.9782
Epoch 9/10
18/18 [==============================] - 92s 5s/step - loss: 0.0013 - accuracy: 0.9999 - val_loss: 0.2269 - val_accuracy: 0.9778
Epoch 10/10
18/18 [==============================] - 91s 5s/step - loss: 0.0012 - accuracy: 0.9999 - val_loss: 0.2968 - val_accuracy: 0.9782
```

Out[ ]:

Saving the Model

In [ ]:

```python
model.save('asl_model_84_54.h5')
```

8. TESTING

8.1 Test Cases

Loading the Dataset & Image Data Generation

In [14]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [15]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [25]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'C:\Users\india\Desktop\Final_Project\Dataset\test_set',target_size=(64,64), class_mode='categorical',batch
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'C:\Users\india\Desktop\Final_Project\Dataset\training_set',target_size=(64,64), class_mode='categorical',bat
```

```
Found 4969 images belonging to 9 classes.
Found 4969 images belonging to 9 classes.
```

In [26]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  6
Len x-test :  6
```

In [27]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[27]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [28]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [29]:
```
# Creating Model
model=Sequential()
```

In [30]:
```
# 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'))
```

In [31]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [32]:
```
# Fitting the Model Generator
model.fit(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
6/6 [==============================] - 23s 4s/step - loss: 5.1206 - accuracy: 0.1690 - val_loss: 3.6505 - val_accuracy: 0.3119
Epoch 2/10
6/6 [==============================] - 22s 4s/step - loss: 2.3945 - accuracy: 0.3266 - val_loss: 1.5087 - val_accuracy: 0.4991
Epoch 3/10
6/6 [==============================] - 22s 4s/step - loss: 1.4384 - accuracy: 0.4037 - val_loss: 1.0430 - val_accuracy: 0.5836
Epoch 4/10
6/6 [==============================] - 23s 4s/step - loss: 1.0761 - accuracy: 0.6488 - val_loss: 0.7109 - val_accuracy: 0.7955
Epoch 5/10
6/6 [==============================] - 27s 5s/step - loss: 0.7835 - accuracy: 0.7774 - val_loss: 0.4046 - val_accuracy: 0.9501
Epoch 6/10
6/6 [==============================] - 25s 5s/step - loss: 0.5470 - accuracy: 0.8756 - val_loss: 0.2540 - val_accuracy: 0.9752
Epoch 7/10
6/6 [==============================] - 22s 4s/step - loss: 0.4018 - accuracy: 0.9090 - val_loss: 0.1675 - val_accuracy: 0.9799
Epoch 8/10
6/6 [==============================] - 22s 4s/step - loss: 0.2862 - accuracy: 0.9406 - val_loss: 0.1185 - val_accuracy: 0.9847
Epoch 9/10
6/6 [==============================] - 22s 4s/step - loss: 0.2108 - accuracy: 0.9612 - val_loss: 0.0880 - val_accuracy: 0.9863
Epoch 10/10
6/6 [==============================] - 22s 4s/step - loss: 0.1548 - accuracy: 0.9738 - val_loss: 0.0736 - val_accuracy: 0.9843
```

Out[32]:

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	11	2	3	2	18
Duplicate	1	3	4	0	8
External	3	5	0	0	8
Fixed	12	2	5	22	41
Not Reproduced	0	1	0	0	1
Skipped	0	0	1	2	3
Won't Fix	0	4	1	1	7
Totals	27	17	14	27	86

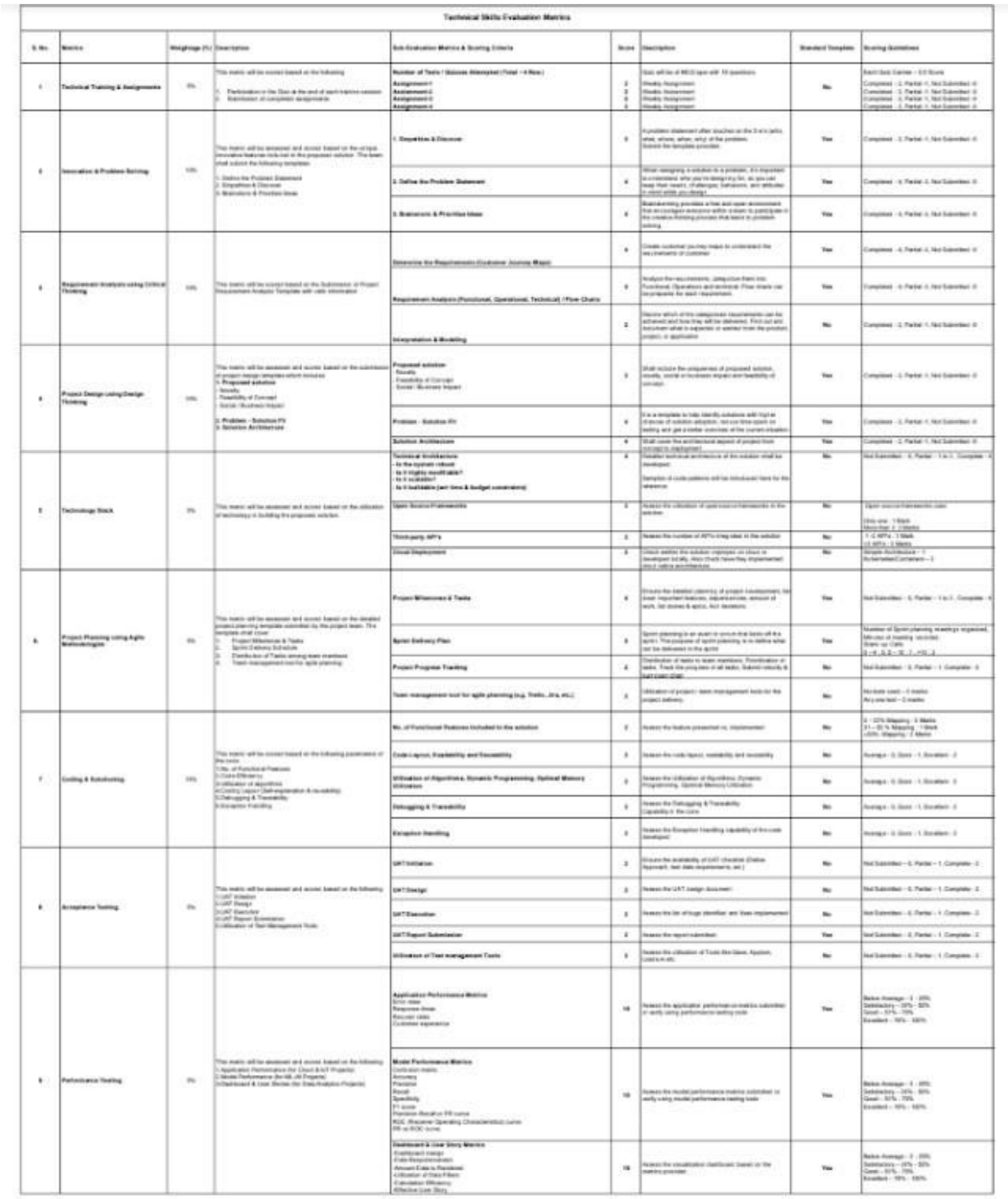
3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	49	0	0	49
Security	4	0	0	4
Outsource Shipping	4	0	0	4
Exception Reporting	11	0	0	11
Final Report Output	2	0	0	2
Version Control	1	0	0	1

9. RESULTS

9.1 Performance Metrics



10.ADVANTAGES & DISADVANTAGES

Advantages:

- It is a cost-effective way of getting several people from different locations to attend meetings and conferences.
- It enables employees from across the world to communicate with each other 24×7 and share ideas or solve problems quickly.

Disadvantages:

- Also accuracy depends upon distance between camera and object.
- It takes a lot of time to listen, speak, read, or write to someone.

11. CONCLUSION

The proposed communication system between Deaf and Dumb people and ordinary people are aiming for it when bridging the communication gap between two societies. It provides complete two sided communication in an efficient manner between the disabled and the normal person.

For communication between deaf person and a second person, a mediator is required to translate sign language of deaf person. But a mediator is required to know the sign language used by deaf person. But this is not always possible since there are multiple sign languages for multiple languages.

So to understand all sign languages, Hand gestures of deaf peoples by normal peoples this system is proposed.

12. FUTURE SCOPE

The speech-to-text and text-to-speech technologies helped those people who had difficulties in communicating or expressing their feelings to the normal people.

This reduces the communication gap between the normal people and the specially abled people.

Using image pre-processing and Artificial Intelligence it is easy to understand the context of objects and clearly explains it to the people who use it for communication.

13.APPENDIX

SOURCE CODE:

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

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

<body style="background: rgb(39,43,48);">
    <nav class="navbar navbar-light navbar-expand-md py-3" style="background:
#212529;">
        <div class="container">
            <div></div><a class="navbar-brand d-flex align-items-center"
href="#"><span
                    class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-
content-center align-items-center me-2 bs-icon"><i
                        class="fas fa-flask"></i></span><span style="color:
rgb(255,255,255);">Real-Time Communication
                    System Powered By AI For Specially Abled</span></a>
            <div></div>
        </div>
    </nav>
    <section>
        <div class="d-flex flex-column justify-content-center align-items-center">
            <div class="d-flex flex-column justify-content-center align-items-center"
id="div-video-feed"
```

```
                style="width: 640px;height: 480px;margin: 10px;min-height:
480px;min-width: 640px;border-radius: 10px;border: 4px dashed
rgb(255,255,255) ;">
                <img src="{{ url_for('video_feed') }}" style="width: 100%;height:
100%;color: rgb(255,255,255);text-align: center;font-size: 20px;"
                    alt="Camera Access Not Provided!">
            </div>
        </div>
        <div class="d-flex flex-column justify-content-center align-items-center"
style="margin-bottom: 10px;"><button
                class="btn btn-info" type="button" data-bs-target="#modal-1" data-bs-
toggle="modal">Quick Reference
                -<strong> ASL Alphabets</strong></button></div>
    </section>
    <section>
        <div class="container">
            <div class="accordion text-white" role="tablist" id="accordion-1">
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <h2 class="accordion-header" role="tab"><button class="accordion-
button" data-bs-toggle="collapse"
                            data-bs-target="#accordion-1 .item-1" aria-expanded="true"
                            aria-controls="accordion-1 .item-1"
                            style="background: rgb(39,43,48);color:
rgb(255,255,255);">About The Project</button></h2>
                    <div class="accordion-collapse collapse show item-1"
role="tabpanel" data-bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Artificial Intelligence has made it possible to
handle our daily activities
                                in new and simpler ways. With the ability to automate tasks
that normally require human
                                intelligence, such as speech and voice recognition, visual
perception, predictive text
                                functionality, decision-making, and a variety of other tasks, AI
can assist people with
                                disabilities by significantly improving their ability to get
around and participate in
                                daily activities.<br><br>Currently, Sign Recognition is
available <strong>only for
```

```
                                    alphabets A-I</strong> and not for J-Z, since J-Z alphabets
also require Gesture
                                Recognition for them to be able to be predicted correctly to a
certain degree of
                                accuracy.</p>
                        </div>
                    </div>
                </div>
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <h2 class="accordion-header" role="tab"><button class="accordion-
button collapsed"
                            data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2"
aria-expanded="false"
                            aria-controls="accordion-1 .item-2"
                            style="background: rgb(39,43,48);color:
rgb(231,241,255);">Developed By</button></h2>
                    <div class="accordion-collapse collapse item-2" role="tabpanel" data-
bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Students at APEC
                                Program.<br><br>1. <strong>KEERTHANA</strong>
420419104029<br>2.
                                <strong>KAVIYA</strong>420419104028<br>3.
<strong>YUVASHREE</strong> 420419104061<br>4.
<strong>SUMITHRA</strong> 420419104305<br>
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
    <div class="modal fade" role="dialog" tabindex="-1" id="modal-1">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">American Sign Language -
Alphabets</h4><button type="button"
                        class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
```

```
                </div>
                <div class="modal-body"><img src="{{ url_for('static',
filename='img/ASL_Alphabets.png') }}" width="100%"></div>
                <div class="modal-footer"><button class="btn btn-secondary"
type="button"
                        data-bs-dismiss="modal">Close</button></div>
            </div>
        </div>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
></script>
</body>

</html>
```

Code:

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

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

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

@app.route('/video_feed')
def video_feed():
    video = Video()
```

return Response(gen(video), mimetype='multipart/x-mixed-replace; boundary = frame')

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

Real-Time Communication System Powered By AI For Specially Abled

Loading the Dataset & Image Data Generation

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

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

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

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

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

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [ ]:
```
# Fitting the Model Generator
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
18/18 [==============================] - 97s 5s/step - loss: 0.0054 - accuracy: 0.9991 - val_loss: 0.3700 - val_accuracy: 0.9756
Epoch 2/10
18/18 [==============================] - 97s 5s/step - loss: 0.0039 - accuracy: 0.9996 - val_loss: 0.3347 - val_accuracy: 0.9751
Epoch 3/10
18/18 [==============================] - 95s 5s/step - loss: 0.0036 - accuracy: 0.9996 - val_loss: 0.3324 - val_accuracy: 0.9756
Epoch 4/10
18/18 [==============================] - 94s 5s/step - loss: 0.0033 - accuracy: 0.9996 - val_loss: 0.3712 - val_accuracy: 0.9747
Epoch 5/10
18/18 [==============================] - 95s 5s/step - loss: 0.0033 - accuracy: 0.9995 - val_loss: 0.3011 - val_accuracy: 0.9764
Epoch 6/10
18/18 [==============================] - 95s 5s/step - loss: 0.0028 - accuracy: 0.9997 - val_loss: 0.2759 - val_accuracy: 0.9769
Epoch 7/10
18/18 [==============================] - 94s 5s/step - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.3056 - val_accuracy: 0.9769
Epoch 8/10
18/18 [==============================] - 95s 5s/step - loss: 0.0021 - accuracy: 0.9997 - val_loss: 0.3332 - val_accuracy: 0.9760
Epoch 9/10
18/18 [==============================] - 93s 5s/step - loss: 0.0019 - accuracy: 0.9997 - val_loss: 0.3236 - val_accuracy: 0.9760
Epoch 10/10
18/18 [==============================] - 93s 5s/step - loss: 0.0016 - accuracy: 0.9997 - val_loss: 0.3429 - val_accuracy: 0.9760
```

Out[ ]:

Saving the Model

In [ ]:
```
model.save('asl_model_84_54.h5')
```

Testing the model

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

In [ ]:
```
model=load_model('asl_model_84_54.h5')
img=image.load_img(r'/content/drive/MyDrive/Dataset/test_set/D/2.png',
                   target_size=(64,64))
```

In [ ]:
```
img
```

Out[ ]:

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

In [ ]:
```
x.ndim
```

Out[ ]: 3

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

In [ ]:
```
x.ndim
```

Out[ ]: 4

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

```
1/1 [==============================] - 0s 145ms/step
```

In [ ]:
```
pred
```

Out[ ]: array([3])

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

```
D
```

OPEN CV

In [ ]:
```
import cv2
```

In [ ]:
```
img=cv2.imread(r'/content/drive/MyDrive/Dataset/test_set/C/2.png',1)
```

In [ ]:
```
img1=cv2.imread(r'/content/drive/MyDrive/Dataset/test_set/B/2.png',0)
```

In [ ]:
```
print(img.shape)
```

```
(64, 64, 3)
```

In [ ]:
```
from google.colab.patches import cv2_imshow
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [3]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [4]:
```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [5]:
```
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
```

In [6]:
```
train_datagen = ImageDataGenerator(rescale = 1./255 , shear_range=0.2,
zoom_range=0.2,horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [7]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="gray
```

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

In [8]:
```
x_test=train_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.
```

Image Preprocessing

Import ImageDataGenerator Library And Configure It

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

In [3]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [4]:
```
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
```

DEMO LINK:

https://drive.google.com/file/d/1izayOTIhkYqmCMKLXa-Rx47xjo0KSFQq/view?usp=share_link

GITHUB LINK:

IBM-Project-49331-1660818038