PROJECT REPORT

TOPIC

Real-Time Communication System Powered by AI for Specially Abled

TEAM MEMBERS

NIRANJAN NAIR - 20BCI7037

SHREE S NADGAUDA - 20BCI7133

NALLA VINAYAK SAI - 20BCR7048

B V CHANDRAHAAS - 20BCD7081

INDEX

1.INTRODUCTION

1.1 Project Overview

Artificial intelligence is not intended to take the place of people, but rather improve our lives by assisting us in tasks that we are unable to complete on our own.Many businesses, like as Google DeepMind, IBM Watson, Apple Siri, Microsoft Cortana, etc., are engaged in this type of research.It implies that there will probably be a lot of fresh innovations soon.Because these technologies make routine tasks simpler and less time-consuming, they could have a positive impact on everyone's lives, including people who do not have disabilities.

1.2 Project purpose

Sharing ideas, thoughts, and experiences with folks in one's local vicinity helps people get to know one another. There are several ways to do this, but the gift of "Speech" is by far the most powerful. Speech makes it possible for all people to understand and effectively convey their views. Not taking into account those who are deprived of this wonderful gift—the dumb and the deaf—will be unfair. In these circumstances, human hand touch has continued to be the primary mode of communication. Things that were formerly difficult or impossible for persons with disabilities to obtain are now frequently available to them and are simple for them to access. People with disabilities can now live in a society where their difficulties are recognised and taken into account thanks to artificial intelligence (AI). Technology can now adapt and change the globe to make it a more open society thanks to technological breakthroughs. As AI directly correlates humans, including those with and without disabilities, there is a certain sensation of being human.[1] To create a model that is trained on different hand motions, we are using a convolution neural network and deep learning. Based on this model, an app is created with the help of which a person who is deaf or dumb can communicate using postures that are translated into speech and human-understandable words.

2.LITERATURE SURVEY

2.1 Existing problem

Sign language recognition has gained significant attention in recent years due to its potential in blurring the line of communication between specially abled individuals and the wider community. Many studies have revolved around this very task of enabling machines to identify the hand gesture that is represented through the video feed or image.

One of the studies by Taniya Sahana et.al [2], proposes a system that helps deaf and mute people to communicate with machines through hand gestures based on multi-scale density features. The paper uses a dataset with 10 classes with 1000 symbols of hand gesture images representing the American Sign Language. They performed orientation normalization on each image and used machine learning models to learn the dataset and predict the labels. The MLP(Multi-layer Perceptron) model procured the highest accuracy(98.2%).

Another study [3], uses blob detection, skin color recognition, template matching and image segmentation to preprocess data for hand gesture images. They use the KNN model for classifying 26 different classes of Indian hand gestures.This paper adds in complete two - sided communication in an efficient manner because the system is implemented as a mobile application.

Chung et al.[4] proposed a machine learning approach for gesture recognition, focusing on the utilization of a deep convolutional neural network. The deep convolutional neural network built by the authors incorporated two prominent architectures, namely AlexNet and VGGNet. The primary motivation behind this technique stemmed from the post-processing of hand images after tracking and recognition had been performed.

2.2 Proposed solution

In this project we aim to classify hand gestures representing letters from ‘A’ to ‘I’ using computer vision and deep learning neural network architecture. The images were also augmented by flipping, shearing, zooming and rotating to increase dataset size and to consider all orientations.

Here, we use the transfer learning method which uses VGG16 as the base model. This CNN architecture is trained on the ImageNet dataset and has 16 layers of convolution which makes it a complex and powerful architecture for image classification tasks. Transfer learning is the technique that includes using a pre-trained model that has already learned general features or patterns from a large dataset, which can be valuable for the new task at hand. By leveraging these learned representations, the model can effectively generalize and adapt to the new task with a smaller amount of labeled data. We freeze some initial layers to remove the problem of overfitting by reducing the complexity of the network as we do not need the extra layers. The number of parameters are reduced significantly. We add additional layers of dropout and dense at the output layer of the VGG16 base model after flattening the output. The output layer consists of 9 neurons(one neuron for each class) where a neuron is activated when the corresponding output is predicted. This layer has a softmax activation where the class with the highest probability is considered the predicted class.

2.3 Problem statement definition

Artificial intelligence has been creating new, more straightforward methods for organising our daily activities. With the great potential to automate routine chores AI can assist people with disabilities by significantly enhancing their capacity to move around and participate in activities of daily living. AI can do this by performing tasks that currently require cognitive

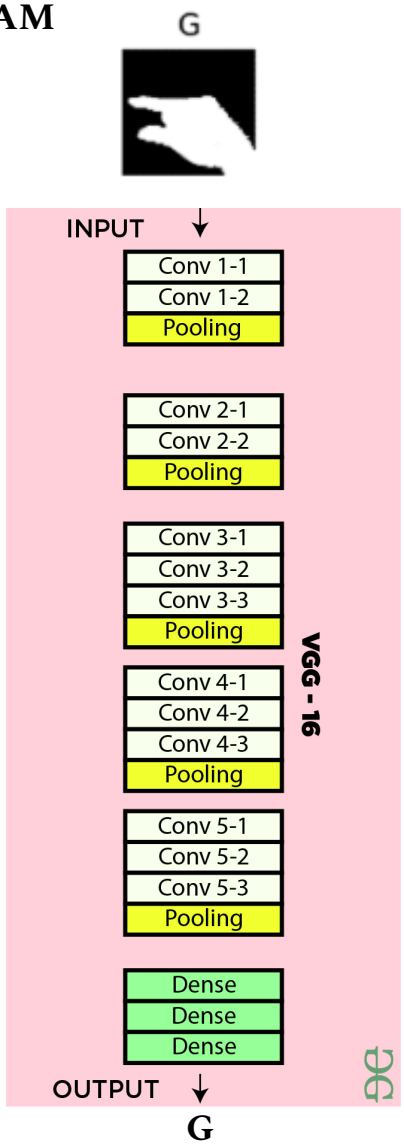
ability, such as speech and voice recognition, visual perception, predictive text functionality, decision-making, and performance of a variety of other tasks.

The Issue with AI Disabled persons can leave the house, navigate their neighbourhoods, engage with others, and even find employment by using driverless automobiles. Autonomous vehicles may make independent mobility easier and boost accessibility that is tailored to the requirements and abilities of each user after they have been fully incorporated into society.

Artificial intelligence has been creating new, more straightforward methods for organising our everyday tasks. AI can greatly improve the mobility and participation of people with disabilities by having the potential to automate tasks that would typically require human intelligence, such as speech and voice recognition, visual perception, predictive text functionality, decision-making, and performance of a variety of other tasks.

3.THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE AND SOFTWARE REQUIREMENTS

1. Hardware Requirements:
   - Medium or High quality Camera(Laptop/Webcam): A camera capturing good-quality images and video feed.This could be a webcam or even a smartphone camera.
   - Computer with GPU: A computer system with sufficient processing power and memory to handle real-time or offline gesture recognition tasks. The specifications will vary depending on the complexity of the algorithm and the desired performance. A GPU can increase the speed of training the model by a lot.
2. Software Requirements:
   - Operating System: The project should be compatible with the target operating system(s), such as Windows, Linux, or macOS.
   - Browser: Running the flask app requires a running and compatible browser.
   - IDE: IDE or Integrated Development Environment is required to write the code. In this project, Vscode and Jupyter Notebook/Colab were used writing the code for preprocessing and constructing the model and training it
   - Libraries and Frameworks: The Python programming language uses various libraries and frameworks required to facilitate image processing, machine learning, deep learning, or computer vision tasks.We use TensorFlow,scikit-learn, OpenCV, Flask etc.

4.EXPERIMENTAL ANALYSIS

Experimental Investigations

4.1 Selecting the frameworks:

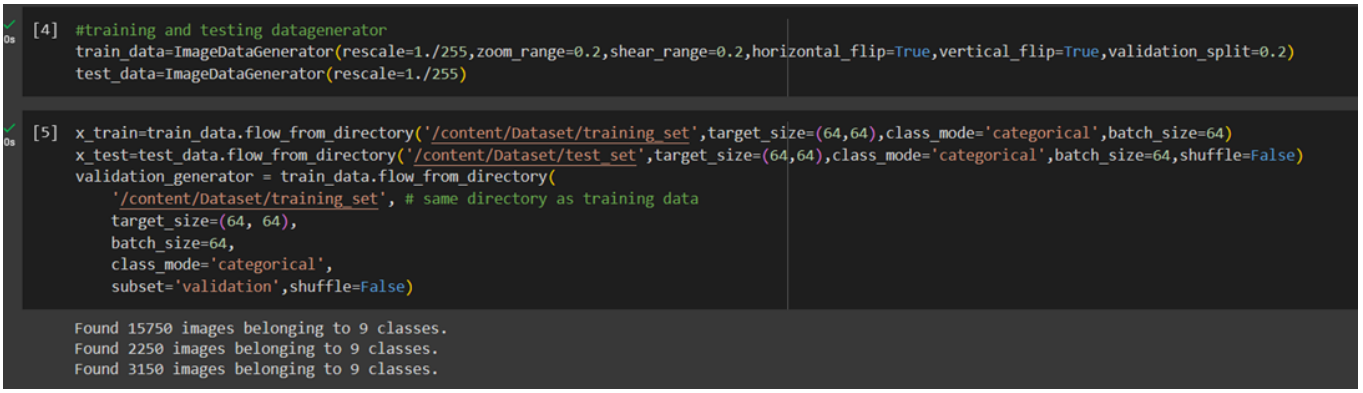
Tensorflow is the most commonly used framework used for building a custom neural network of any type in an extremely simple manner, so we select TensorFlow which has keras inbuilt which is an extremely important framework for building network architectures from scratch or importing a base model(Here, VGG16) to perform transfer learning. For pre-processing we use numpy and scikit learn preprocessing tools and to plot the results we use plotly and matplotlib libraries as they provide a simple and neat layout with many options.

```
[ ]  import pandas as pn
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sbn
     from sklearn.metrics import confusion_matrix
     import itertools
     import os
     import cv2
     import numpy as np
     import tensorflow as tf
     import pandas as pd
     import glob
     from matplotlib import pyplot as plt
     from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
     from keras.models import Sequential
     from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization
     from tensorflow.keras.optimizers import RMSprop,Adam
     from keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.preprocessing import image
     from keras.callbacks import ReduceLROnPlateau
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     !pip install plotly
     import plotly
     import plotly.graph_objects as go
     import plotly.express as px
     from plotly.subplots import make_subplots
     from sklearn.manifold import TSNE
```

4.2 Data Augmentation:

We modify and augment the images in both the train and test data. The only operation for test data we perform is rescaling the images to 1./255 to. By rescaling images in the test and training set to 1/255 within the ImageDataGenerator, the pixel values are appropriately transformed to meet the requirements for efficient and effective model training. We do not

need to augment the test data since we won't use it for training. The training set is zoomed and sheared by a factor of 0.2 which is most of the time optimal when working with multiclass classification. A shear range and zoom range of 0.2 can provide a reasonable level of diversity without introducing extreme or unrealistic transformations that might hinder the model's ability to deal with real life images. We also perform horizontal and vertical flipping to provide robust orientations that take all kinds of directions a hand can be present in front of the system into consideration. Before passing the images to the model, we set shuffle as false in the test when flowing from the directory in order to stop the model from learning wrong labels. We also create a validation set with 20% of the test data in order to test the model how it did each epoch on unseen data.



4.3 Model construction:

We use the VGG16 as the basel model. It consists of 16 convolution layers which is very complex. We set the layers to non trainable to reduce the number of parameters as our dataset is not overly complex or huge. We use the imagenet weights as it consists of the information of a huge variety of objects which will help in generalizing the predictions. In the end of the base model, custom dense layers with 128 neurons and 64 neurons are added. This can be any number of neurons, this parameter will not bring a significant change in the results. However, 2 dropout layers on each dense layer with a dropout probability of 0.5 were added to reduce overfitting as the model is complex with multiple neurons. The images were fed into the model with batches of 64 images as it is an optimal number when memory is sufficient enough. The learning rate was set as 0.001(Optimal) and the Adam optimizer was used as it is the most efficient in performing loss optimization. The loss function used is categorical cross entropy which is required for multicast classification problems. The model was run for 10 epochs as the validation accuracy saturated around that point and did not have a significant increase after that.

```
[12] base_model = VGG16(weights='imagenet', include_top=False, input_shape=(64,64,3))

     for layer in base_model.layers:
         layer.trainable = False

     x = base_model.output
     x = Flatten()(x)
     x = Dense(128, activation='relu')(x)
     x = Dropout(0.5)(x)
     x = Dense(64, activation='relu')(x)
     x = Dropout(0.5)(x)
     predictions = Dense(9, activation='softmax')(x)

     model = Model(inputs=base_model.input, outputs=predictions)
```

```
[17] history=model.fit_generator(
            x_train,
            steps_per_epoch=len(x_train),
            epochs=10,
            validation_data = validation_generator,
        validation_steps = len(validation_generator)

    )

    <ipython-input-17-93725b234748>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
      history=model.fit_generator(
    Epoch 1/10
    247/247 [==============================] - 45s 180ms/step - loss: 0.8964 - accuracy: 0.6685 - val_loss: 0.1785 - val_accuracy: 0.9683
    Epoch 2/10
    247/247 [==============================] - 33s 133ms/step - loss: 0.2974 - accuracy: 0.9006 - val_loss: 0.0610 - val_accuracy: 0.9863
    Epoch 3/10
    247/247 [==============================] - 42s 169ms/step - loss: 0.1829 - accuracy: 0.9433 - val_loss: 0.0259 - val_accuracy: 0.9956
    Epoch 4/10
    247/247 [==============================] - 33s 135ms/step - loss: 0.1300 - accuracy: 0.9580 - val_loss: 0.0188 - val_accuracy: 0.9959
    Epoch 5/10
    247/247 [==============================] - 36s 145ms/step - loss: 0.1036 - accuracy: 0.9676 - val_loss: 0.0135 - val_accuracy: 0.9971
    Epoch 6/10
    247/247 [==============================] - 33s 134ms/step - loss: 0.0972 - accuracy: 0.9698 - val_loss: 0.0103 - val_accuracy: 0.9984
    Epoch 7/10
    247/247 [==============================] - 33s 135ms/step - loss: 0.0775 - accuracy: 0.9750 - val_loss: 0.0283 - val_accuracy: 0.9921
    Epoch 8/10
    247/247 [==============================] - 38s 155ms/step - loss: 0.0806 - accuracy: 0.9733 - val_loss: 0.0078 - val_accuracy: 0.9987
    Epoch 9/10
    247/247 [==============================] - 34s 136ms/step - loss: 0.0674 - accuracy: 0.9771 - val_loss: 0.0051 - val_accuracy: 0.9990
    Epoch 10/10
    247/247 [==============================] - 34s 137ms/step - loss: 0.0709 - accuracy: 0.9776 - val_loss: 0.0044 - val_accuracy: 0.9994
```

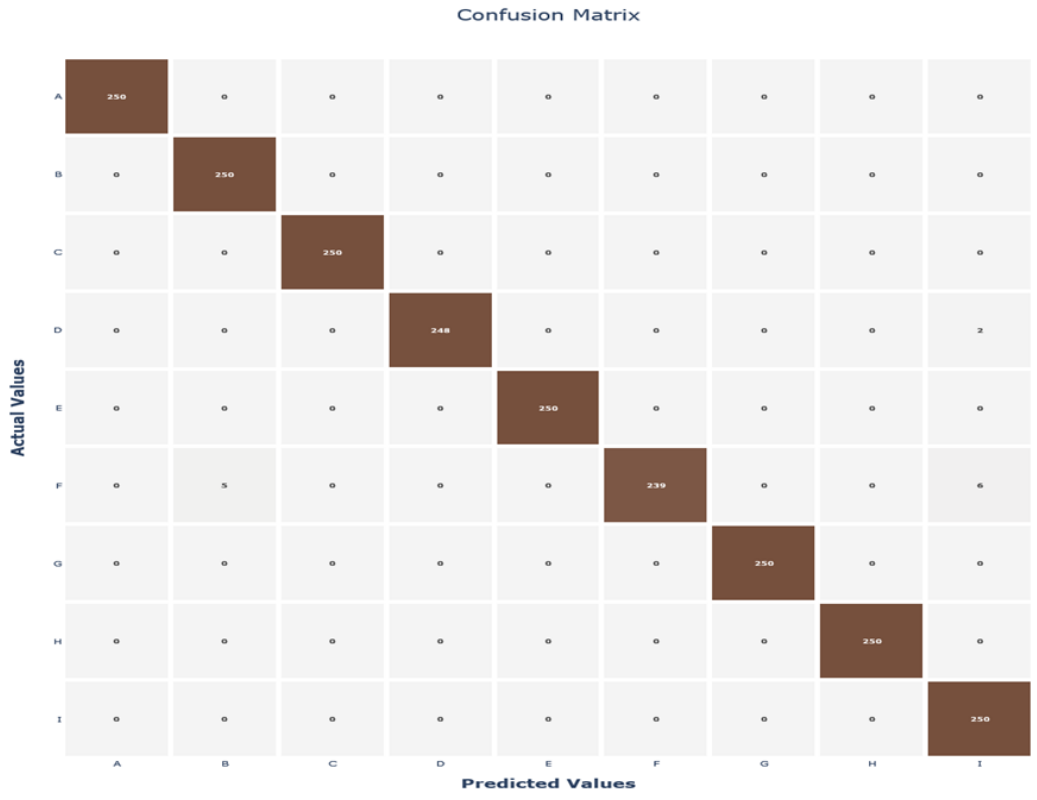
5.RESULTS

The final training accuracy at the 10th epoch came up to be 97.76% and the final validation accuracy came up to be 99.94%, so clearly there is no sign of overfitting. The validation accuracy starts at a high value as shown in the graph because the images are fairly similar to each other and the model has already learnt enough from the training dataset.

The test accuracy came up to be 99.42% which is great performance and is an indication of a reliable model. However as this is not enough we take a look at the classification report which gives us other evaluation metrics like f-score,recall and precision for each class. We can observe that the macro and weight average come up to be 99%.

	precision	recall	f1-score	support
A	1.00	1.00	1.00	250
B	0.98	1.00	0.99	250
C	1.00	1.00	1.00	250
D	1.00	0.99	1.00	250
E	1.00	1.00	1.00	250
F	1.00	0.96	0.98	250
G	1.00	1.00	1.00	250
H	1.00	1.00	1.00	250
I	0.97	1.00	0.98	250
accuracy			0.99	2250
macro avg	0.99	0.99	0.99	2250
weighted avg	0.99	0.99	0.99	2250

The confusion matrix is as follows. All classes were predicted accurately almost throughout the whole of the test data. The class F falls a little short as the model may have mistook some of the 'F's as 'I's or 'B's.



6.ADVANTAGES AND DISADVANTAGES

7.1 Advantages

- Reduces manual work
- More accurate than average human
- Capable of handling a lot of data
- Can be used anywhere from any device

7.2 Disadvantages

- Cannot handle complex data
- All the data must be in digital format
- Requires a high performance server for faster predictions
- Prone to occasional errors

7.APPLICATIONS

1. Augmentative and Alternative Communication (AAC) Systems:

- AAC systems use AI and natural language processing to assist individuals with speech or communication impairments. These systems can generate speech or convert text into speech, enabling people with disabilities to communicate effectively. [6]

2. Smart Home Automation for Accessibility:

- Real-time communication systems integrated with AI can enable individuals with disabilities to control their home environment. Voice recognition, gesture recognition, and other AI technologies allow them to operate lights, appliances, doors, and other devices through voice commands or gestures. [7]

3. Vision Assistance:

- Real-time communication systems using AI can assist individuals with visual impairments. AI-powered algorithms can identify objects, read text, describe scenes, and provide audio guidance to help visually impaired individuals navigate their surroundings more effectively. [8]

4. Hearing Assistance:

- AI-based real-time communication systems can enhance communication for individuals with hearing impairments. Speech recognition and transcription algorithms can convert spoken words into text, enabling individuals to read conversations in real-time or through visual displays. [9]

5. Cognitive Assistants:

- Real-time communication systems using AI can provide cognitive assistance for individuals with cognitive disabilities. These systems can offer reminders, prompts, and guidance for daily activities, helping individuals with cognitive impairments manage their tasks and routines. [10]

6. Mobility and Navigation:

- Real-time communication systems combined with AI can assist individuals with mobility impairments. AI algorithms can help plan accessible routes, provide navigation instructions, and offer real-time information about accessible transportation options. [11]

8.CONCLUSION

This project demonstrated a web application that uses deep learning and computer vision along with neural networks to recognize ASL. Flask , python is used to create this project . The model achieved a training accuracy of 98% and a testing accuracy of around 99% . The proposed project is scalable and can handle a huge number of users. Since it is a web application , it is compatible with any device that can run a browser. This project is extremely useful in real-world scenarios such as recognising numbers , processing bank cheques and so

on. There is so much room for improvement , which can be implemented in subsequent versions.

9.FUTURE SCOPE

This project is far from complete and there is a lot of room for improvement[5].

Some of the improvements that can be made to this project are as follows:

- Add support to detect from letters multiple images and save the results
- Add support to detect multiple digits and letters
- Add support to multi reactions function
- Improve model to detect digits and letters from complex images
- Add support to different languages to help users from all over the world

This project has endless potential and can always be enhanced to become better.Implementing this concept in the real world will benefit several industries and reduce the workload on many workers, enhancing overall work efficiency

10.BIBLIOGRAPHY

[1] N. G. Priya, D. Ravikumar, M. Safa, G. Saranya and D. Arun, "Real-Time Communication System for Specially Abled with DL-CNN Based Approach Using Image Processing," 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2022, pp. 1-8, doi: 10.1109/ICPECTS56089.2022.10046909.

[2]Taniya Sahana, Soumi Paul, Subhadip Basu, Ayatullah Faruk Mollah, Hand sign recognition from depth images with multi-scale density features for deaf mute persons, Procedia Computer Science, Volume 167, 2020, Pages 2043-2050,ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.03.243

[3]Shrikant Temburwar[1], Payal Jaiswal[2], Shital Mande[3], Souparnika Patil[4], "Design of a Communication System using Sign Language aid for Differently Abled Peoples". International Research Journal of Engineering and Technology(IRJET) Volume 4 Issue 3

[4]Chung, H. Y., Chung, Y. L., & Tsai, W. F. (2019, February). An efficient hand gesture recognition system based on deep CNN. In 2019 IEEE International Conference on Industrial Technology (ICIT) (pp. 853- 858). IEEE.

[5]Papastratis I, Chatzikonstantinou C, Konstantinidis D, Dimitropoulos K, Daras P. Artificial Intelligence Technologies for Sign Language. Sensors (Basel). 2021 Aug 30;21(17):5843. doi: 10.3390/s21175843. PMID: 34502733; PMCID: PMC8434597.

[6] https://www.aacandautism.com/

[7]https://www.accenture.com/us-en/error/pagenotfound?errSrc=%2fus-en%2fblogs%2fblogs-ai-powered-smart-home-solutions-disabilities

[8]https://www.microsoft.com/en-us/ai/seeing-ai

[9] https://www.android.com/accessibility/live-transcribe/

[10] https://cognoa.com/

[11] https://aira.io/

11.APPENDIX

Model code:

```
model=Sequential()
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(256,activation='relu'))
model.add(Dense(128,activation='relu'))

# Adding Output Layer
model.add(Dense(9,activation='softmax'))
```

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0
```

```
class callbacks(tf.keras.callbacks.Callback):
    def on_epoch_end(self,epochs,logs={}):
        if(logs.get('val_accuracy')>0.99):
            self.model.stop_training=True
```

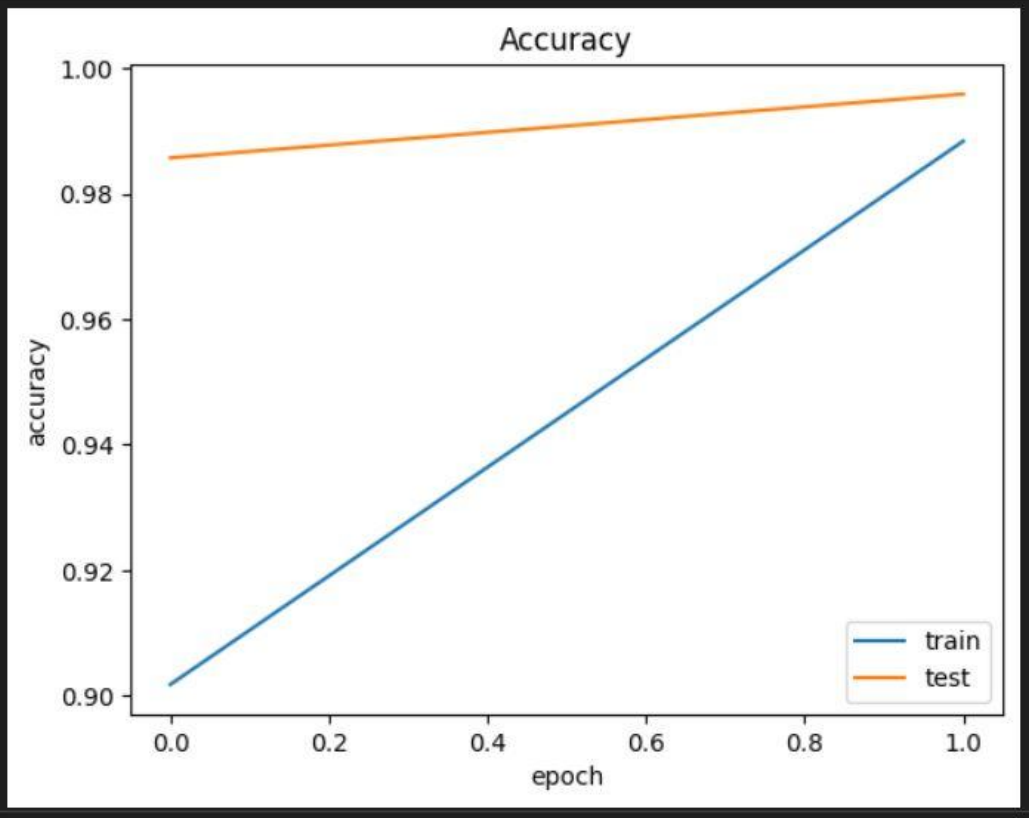
```
callback = callbacks()
```

```
history=model.fit_generator(
        x_train,
        steps_per_epoch=len(x_train),
        epochs=10,
        validation_data = validation_generator,
    validation_steps = len(validation_generator),
        callbacks=callback

)
```

```
<ipython-input-19-14758fdefc2d>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a fu
  history=model.fit_generator(
Epoch 1/10
158/158 [==============================] - 44s 219ms/step - loss: 0.3071 - accuracy: 0.9018 - val_loss: 0.0566
Epoch 2/10
158/158 [==============================] - 30s 191ms/step - loss: 0.0382 - accuracy: 0.9884 - val_loss: 0.0188
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("Accuracy")
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(['train','test'])
plt.show()
```



WEB APP CODE

```python
import gradio as gr
import cv2
import numpy as np
from keras.models import load_model

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

}
# Load the model
model = load_model("model1.h5")

# Define the function to recognize the sign
def recognize_sign(image):
    # Preprocess the grayscale image
    grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized_image = cv2.resize(grayscale_image, (64, 64))

    # Convert grayscale image to RGB format
    rgb_image = np.repeat(resized_image[..., np.newaxis], 3, axis=-1)

    # Make predictions with the model
    prediction = model.predict(np.expand_dims(rgb_image, axis=0))[0]

    predicted_class_index = np.argmax(prediction)
```

```python
# Define the input and output interfaces
iface = gr.Interface(fn=recognize_sign, inputs="image", outputs="text",
title="Real-Time Communication System Powered by Ai for Specially Abled ",
description="Upload an image of a sign here. The model will predict the corresponding alphabet in the output section.",
examples=[
        ["C:\\Users\\bvcha\\Desktop\\project\\ASL_Alphabets.png"],
    ]
)

# Launch the interface
iface.launch()
```