

IMPORT LIBRARIES

In []:

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

unzip the file

In []:

```
!unzip '/content/conversation engine for deaf and dumb (8).zip'
```

Streaming output truncated to the last 5000 lines.

```
extracting: Dataset/training_set/G/1225.png
extracting: Dataset/training_set/G/1226.png
extracting: Dataset/training_set/G/1227.png
extracting: Dataset/training_set/G/1228.png
extracting: Dataset/training_set/G/1229.png
  inflating: Dataset/training_set/G/123.png
extracting: Dataset/training_set/G/1230.png
extracting: Dataset/training_set/G/1231.png
extracting: Dataset/training_set/G/1232.png
  inflating: Dataset/training_set/G/1233.png
  inflating: Dataset/training_set/G/1234.png
  inflating: Dataset/training_set/G/1235.png
  inflating: Dataset/training_set/G/1236.png
  inflating: Dataset/training_set/G/1237.png
  inflating: Dataset/training_set/G/1238.png
  inflating: Dataset/training_set/G/1239.png
  inflating: Dataset/training_set/G/124.png
  inflating: Dataset/training_set/G/1240.png
  inflating: Dataset/training_set/G/1241.png
  inflating: Dataset/training_set/G/1242.png
  inflating: Dataset/training_set/G/1243.png
  inflating: Dataset/training_set/G/1244.png
  inflating: Dataset/training_set/G/1245.png
extracting: Dataset/training_set/G/1246.png
  inflating: Dataset/training_set/G/1247.png
  inflating: Dataset/training_set/G/1248.png
  inflating: Dataset/training_set/G/1249.png
  inflating: Dataset/training_set/G/125.png
  inflating: Dataset/training_set/G/1250.png
  inflating: Dataset/training_set/G/1251.png
  inflating: Dataset/training_set/G/1252.png
  inflating: Dataset/training_set/G/1253.png
  inflating: Dataset/training_set/G/1254.png
  inflating: Dataset/training_set/G/1255.png
  inflating: Dataset/training_set/G/1256.png
  inflating: Dataset/training_set/G/1257.png
  inflating: Dataset/training_set/G/1258.png
  inflating: Dataset/training_set/G/1259.png
  inflating: Dataset/training_set/G/126.png
  inflating: Dataset/training_set/G/1260.png
  inflating: Dataset/training_set/G/1261.png
extracting: Dataset/training_set/G/1262.png
  inflating: Dataset/training_set/G/1263.png
  inflating: Dataset/training_set/G/1264.png
  inflating: Dataset/training_set/G/1265.png
  inflating: Dataset/training_set/G/1266.png
  inflating: Dataset/training_set/G/1267.png
extracting: Dataset/training_set/G/1268.png
  inflating: Dataset/training_set/G/1269.png
  inflating: Dataset/training_set/G/127.png
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

inflating: Dataset/training_set/I/950.png
inflating: Dataset/training_set/I/951.png
inflating: Dataset/training_set/I/952.png
inflating: Dataset/training_set/I/953.png
inflating: Dataset/training_set/I/954.png
inflating: Dataset/training_set/I/955.png
inflating: Dataset/training_set/I/956.png
inflating: Dataset/training_set/I/957.png
inflating: Dataset/training_set/I/958.png
inflating: Dataset/training_set/I/959.png
inflating: Dataset/training_set/I/96.png
inflating: Dataset/training_set/I/960.png
inflating: Dataset/training_set/I/961.png
inflating: Dataset/training_set/I/962.png
inflating: Dataset/training_set/I/963.png
inflating: Dataset/training_set/I/964.png
inflating: Dataset/training_set/I/965.png
inflating: Dataset/training_set/I/966.png
inflating: Dataset/training_set/I/967.png
inflating: Dataset/training_set/I/968.png
inflating: Dataset/training_set/I/969.png
inflating: Dataset/training_set/I/97.png
inflating: Dataset/training_set/I/970.png
inflating: Dataset/training_set/I/971.png
inflating: Dataset/training_set/I/972.png
extracting: Dataset/training_set/I/973.png
inflating: Dataset/training_set/I/974.png
inflating: Dataset/training_set/I/975.png
inflating: Dataset/training_set/I/976.png
inflating: Dataset/training_set/I/977.png
inflating: Dataset/training_set/I/978.png
inflating: Dataset/training_set/I/979.png
inflating: Dataset/training_set/I/98.png
inflating: Dataset/training_set/I/980.png
inflating: Dataset/training_set/I/981.png
inflating: Dataset/training_set/I/982.png
extracting: Dataset/training_set/I/983.png
inflating: Dataset/training_set/I/984.png
inflating: Dataset/training_set/I/985.png
inflating: Dataset/training_set/I/986.png
inflating: Dataset/training_set/I/987.png
inflating: Dataset/training_set/I/988.png
inflating: Dataset/training_set/I/989.png
inflating: Dataset/training_set/I/99.png
inflating: Dataset/training_set/I/990.png
inflating: Dataset/training_set/I/991.png
inflating: Dataset/training_set/I/992.png
extracting: Dataset/training_set/I/993.png
inflating: Dataset/training_set/I/994.png
inflating: Dataset/training_set/I/995.png
extracting: Dataset/training_set/I/996.png
inflating: Dataset/training_set/I/997.png
inflating: Dataset/training_set/I/998.png
inflating: Dataset/training_set/I/999.png

```

DATA AUGMENTATION

In []:

```

from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale = 1./255, shear_range=0.2, 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_size=(64,64), batch_size=100,
class_mode='categorical', color_mode = "grayscale")

```

Found 15750 images belonging to 9 classes.

In []:

```
x_test = test_datagen.flow_from_directory("/content/Dataset/test_set", target_size=(64,64),batch_size=100,
                                         class_mode='categorical', color_mode ="grayscale")
```

Found 2250 images belonging to 9 classes.

In []:

```
len(x_train)
```

Out[]:

158

In []:

```
len(x_test)
```

Out[]:

23

In []:

```
x_train.class_indices
```

Out[]:

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

MODEL BUILDING

In []:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

In []:

```
#Creating the model
model=Sequential()
#Adding the layers
model.add(Convolution2D(32, (3,3), input_shape=(64,64,1), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

#adding hidden layers
model.add(Dense(400, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(100, activation='relu'))

#Adding the output layer
model.add(Dense(9, activation='softmax'))
```

In []:

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

In []:

```
model.fit_generator(x_train, steps_per_epoch=30, epochs=10, validation_data=x_test, validation_steps=50)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
```

```
"""Entry point for launching an IPython kernel.
```

```
Epoch 1/10
```

```
30/30 [=====] - ETA: 0s - loss: 1.0544 - accuracy: 0.6410
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 50 batches). You may need to use the repeat() function when building your dataset.
```

```
30/30 [=====] - 19s 595ms/step - loss: 1.0544 - accuracy: 0.6410  
- val_loss: 0.4580 - val_accuracy: 0.8564
```

```
Epoch 2/10
```

```
30/30 [=====] - 14s 464ms/step - loss: 0.3143 - accuracy: 0.9017
```

```
Epoch 3/10
```

```
30/30 [=====] - 13s 429ms/step - loss: 0.1582 - accuracy: 0.9587
```

```
Epoch 4/10
```

```
30/30 [=====] - 13s 446ms/step - loss: 0.0724 - accuracy: 0.9797
```

```
Epoch 5/10
```

```
30/30 [=====] - 13s 421ms/step - loss: 0.0763 - accuracy: 0.9776
```

```
Epoch 6/10
```

```
30/30 [=====] - 13s 425ms/step - loss: 0.0923 - accuracy: 0.9670
```

```
Epoch 7/10
```

```
30/30 [=====] - 14s 470ms/step - loss: 0.0590 - accuracy: 0.9857
```

```
Epoch 8/10
```

```
30/30 [=====] - 13s 424ms/step - loss: 0.0327 - accuracy: 0.9910
```

```
Epoch 9/10
```

```
30/30 [=====] - 14s 461ms/step - loss: 0.0225 - accuracy: 0.9943
```

```
Epoch 10/10
```

```
30/30 [=====] - 13s 426ms/step - loss: 0.0269 - accuracy: 0.9920
```

```
Out[ ]:
```

```
<keras.callbacks.History at 0x7fad22e75c90>
```

```
In [ ]:
```

```
model.save('Real_time.h5')
```

TEST THE MODEL

```
In [ ]:
```

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

```
In [ ]:
```

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

```
In [ ]:
```

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

```
Out[ ]:
```



```
In [ ]:
```

```
from skimage.transform import resize
```



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

In []:

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

```
1/1 [=====] - 0s 110ms/step
THE PREDICTED LETTER IS  H
```

In []:

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

```
1/1 [=====] - 0s 95ms/step
THE PREDICTED LETTER IS  A
```

In []:

```
img=image.load_img('/content/Dataset/test_set/E/111.png')
detect(img)
```

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