TESTING THE MODEL

##IMPORT LIBRARIES

```
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

```
!unzip '/content/drive/MyDrive/IBMPROJECT/conversation engine for deaf and dumb.zip'
```

##DATA AUGMENTATION

```
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)

x_train = train_datagen.flow_from_directory("../DATA COLLECTION/training_set", target_size=(64,64),batch_size=100,

                              class_mode='categorical', color_mode ="grayscale")

x_test = test_datagen.flow_from_directory("../DATA COLLECTION/test_set", target_size=(64,64),batch_size=100,

                            class_mode='categorical', color_mode ="grayscale")

len(x_train)

len(x_test)

x_train.class_indices
```

```
##MODEL BUILDING
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
#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'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit_generator(x_train, steps_per_epoch=30, epochs=10, validation_data=x_test,validation_steps=50)
model.save('Real_time.h5')
```

##TEST THE MODEL

```
from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

import numpy as np

import cv2

model = load_model('Real_time.h5')

img = image.load_img('../DATA COLLECTION/test_set/test_set/H/107.png',target_size = (100,100))

img

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])

img=image.load_img("../DATA COLLECTION/test_set/test_set/H/107.png")

detect(img)

img = image.load_img('../DATA COLLECTION/test_set/test_set/A/110.png')

pred=detect(img)

img=image.load_img('../DATA COLLECTION/test_set/test_set/E/111.png')

detect(img)
```

FLASK APPLICATION

app.py

```python
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: ASL_Alphabet.jpg\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(debug=True)
```

camera.py

```python
import cv2

video = cv2.VideoCapture(0)

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

video.release()
cv2.destroyAllWindows()
```