Delivery of Sprint 3

Date	06 November 2022
Team ID	PNT2022TMID21808
Project Name	Real time Communication System Powered by AI for Specially Abled

Model Building

Import The Required Model Building Libraries

In [6]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

Initialize The Model

In [7]:
```
model=Sequential()
```

Add The Convolution Layer

In [10]:
```
model.add(Convolution2D(32,(3,3),activation="relu",input_shape=(64,64,3)))
#No of feature detectors, size of feature detector, image size, activation function
```

Add The Pooling Layer

In [11]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

Add The Flatten Layer

In [12]:
```
model.add(Flatten())
```

Adding The Dense Layers

In [13]:
```
model.add(Dense(200,activation='relu'))
```

In [15]:
```
model.add(Dense(9,activation="softmax"))
```

Compile The Model

In [16]:
```
model.compile(loss="categorical_crossentropy",metrics=["accuracy"],optimizer='adam')
```

In [17]:
```
len(x_train)
```

Out[17]: 525

In [18]:
```
len(x_test)
```

Out[18]: 75

Fit And Save The Model

In [32]:
```
model.fit(x_train,epochs=8,validation_data=x_test,steps_per_epoch=len(x_train),validation_steps=len(x_test)
```

```
Epoch 1/8
525/525 [==============================] - 620s 1s/step - loss: 0.6116 - accuracy: 0.7691 - val_loss: 0.3
006 - val_accuracy: 0.9329
Epoch 2/8
525/525 [==============================] - 223s 425ms/step - loss: 0.1041 - accuracy: 0.9683 - val_loss:
0.0779 - val_accuracy: 0.9858
Epoch 3/8
525/525 [==============================] - 132s 250ms/step - loss: 0.0592 - accuracy: 0.9829 - val_loss:
0.1236 - val_accuracy: 0.9760
Epoch 4/8
525/525 [==============================] - 104s 198ms/step - loss: 0.0431 - accuracy: 0.9879 - val_loss:
0.2067 - val_accuracy: 0.9742
Epoch 5/8
525/525 [==============================] - 107s 204ms/step - loss: 0.0322 - accuracy: 0.9912 - val_loss:
0.0713 - val_accuracy: 0.9800
Epoch 6/8
525/525 [==============================] - 113s 216ms/step - loss: 0.0348 - accuracy: 0.9895 - val_loss:
0.1267 - val_accuracy: 0.9787
Epoch 7/8
525/525 [==============================] - 101s 193ms/step - loss: 0.0293 - accuracy: 0.9926 - val_loss:
0.1558 - val_accuracy: 0.9751
Epoch 8/8
525/525 [==============================] - 107s 205ms/step - loss: 0.0222 - accuracy: 0.9940 - val_loss:
0.1998 - val_accuracy: 0.9769
```

Out[32]:
```
<keras.callbacks.History at 0x20f394a4e20>
```

In [33]:
```
model.save("C:/Users/rajes/Downloads/signlanguage-new.h5")
```

Test the Model

Test the Model

Import The Packages And Load The Saved Model

In [35]:
```
from keras.models import load_model
import numpy as np
import h5py
import cv2
```

In [36]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

In [37]:
```
model = load_model("C:/Users/rajes/Downloads/signlanguage-new.h5")
```

Load The Test Image, Pre-Process It And Predict

In [39]:
```
img =image.load_img(r"C:\Users\rajes\Desktop\Dataset\test_set\A\8.png",target_size = (64,64,1))
img
```

Out[39]:

In [41]:
```
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 [42]:
```
from skimage.transform import resize
def detect(frame):
    img=resize(frame,(64,64,1))
    img=np.expand_dims(img,axis=0)
    if(np.max(img)>1):
        img=img/255.0
        prediction=model.predict(img)
        print(prediction)
        prediction=model.predict_classes(img)
        print(prediction)
```

In [43]:
```
frame=cv2.imread(r"C:\Users\rajes\Desktop\Dataset\test_set\A\8.png")
data=detect(frame)
```

In [44]:
```
type(img)
```

Out[44]:
```
PIL.Image.Image
```

In [45]:
```
x = image.img_to_array(img)
x
```

Out[45]:
```
array([[[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       ...,

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],


          [[0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.],
           ...,
           [0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.]],

          [[0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.],
           ...,
           [0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.]]], dtype=float32)
```

In [46]:
```
x.shape
```

Out[46]:
```
(64, 64, 3)
```

In [47]:
```
x=np.expand_dims(x,axis=0)
x.shape
```

Out[47]:
```
(1, 64, 64, 3)
```

In [48]:
```
pred_prob = model.predict(x)
```

```
1/1 [==============================] - 1s 1s/step
```

In [49]:
```
pred_prob
```

Out[49]:
```
array([[9.9954236e-01, 7.8000909e-13, 7.1030300e-08, 7.4072335e-07,
        3.7532591e-04, 2.8473270e-12, 1.7780074e-05, 6.3732426e-05,
        7.7890165e-09]], dtype=float32)
```

In [50]:
```
class_name=["A","B","C","D","E","F","G","H","I"]
pred_id = pred_prob.argmax(axis=1)[0]
```

In [51]:
```
pred_id
```

Out[51]:
```
0
```

In [52]:
```
print("the alphabet is  ",str(class_name[pred_id]))
```

```
the alphabet is   A
```

In [53]:
```
background = None
accumulated_weight = 0.5

ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350
```

In [54]:
```
word_dict = { 0:'A', 1:'B',  2:'C', 3: 'D', 4:'E', 5:'F', 6:'G',7:'H', 8:'I' }
'''
predictions = model.predict(imgs, verbose=0)
print("predictions on a small set of test data--")
print("")
for ind, i in enumerate(predictions):
    print(word_dict[np.argmax(i)], end='   ')

plotImages(imgs)
print('Actual labels')
for i in labels:
    print(word_dict[np.argmax(i)], end='   ')'''
```

Out[54]:
```
'\npredictions = model.predict(imgs, verbose=0)\nprint("predictions on a small set of test data--")\nprin
t("")\nfor ind, i in enumerate(predictions):\n    print(word_dict[np.argmax(i)], end=\'   \')\n\nplotImag
es(imgs)\nprint(\'Actual labels\')\nfor i in labels:\n    print(word_dict[np.argmax(i)], end=\'   \')'
```

In [55]:
```
def cal_accum_avg(frame, accumulated_weight):

    global background

    if background is None:
        background = frame.copy().astype("float")
        return None

    cv2.accumulateWeighted(frame, background, accumulated_weight)
```

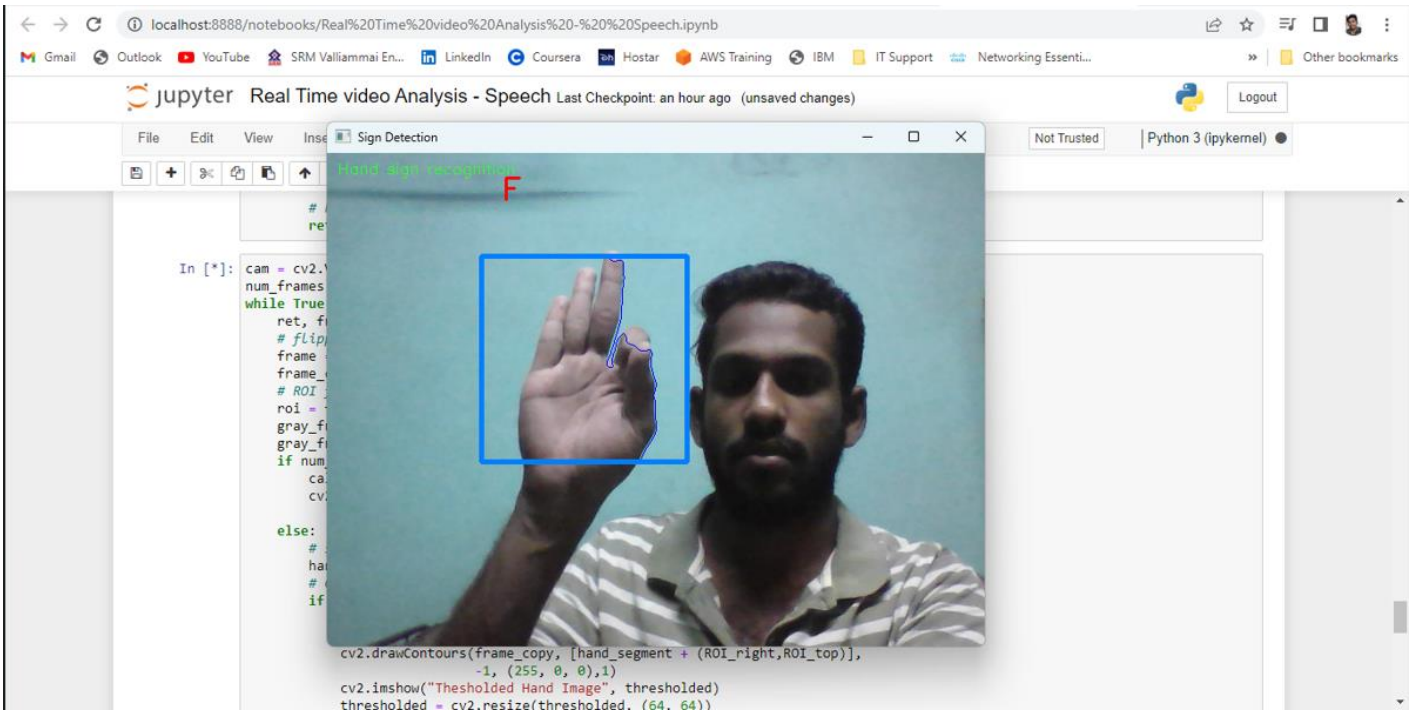
In [56]:
```
def segment_hand(frame, threshold=25):
    global background
    diff = cv2.absdiff(background.astype("uint8"), frame)
    _ , thresholded = cv2.threshold(diff, threshold,255,cv2.THRESH_BINARY)
    #Fetching contours in the frame (These contours can be of hand or any other object in foreground) …
    contours, hierarchy =cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    # If length of contours list = 0, means we didn't get any contours...
```

```
    if len(contours) == 0:
        return None
    else:
        # The largest external contour should be the hand
        hand_segment_max_cont = max(contours, key=cv2.contourArea)

        # Returning the hand segment(max contour) and the thresholded image of hand...
        return (thresholded, hand_segment_max_cont)
```

In [ ]:
```
cam = cv2.VideoCapture(0)
num_frames =0
while True:
    ret, frame = cam.read()
    # flipping the frame to prevent inverted image of captured frame...
    frame = cv2.flip(frame, 1)
    frame_copy = frame.copy()
    # ROI from the frame
    roi = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]
    gray_frame = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.GaussianBlur(gray_frame, (9, 9), 0)
    if num_frames < 70:
        cal_accum_avg(gray_frame, accumulated_weight)
        cv2.putText(frame_copy, "FETCHING BACKGROUND...PLEASE WAIT",(80, 400),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)
    else:
        # segmenting the hand region
        hand = segment_hand(gray_frame)
        # Checking if we are able to detect the hand...
        if hand is not None:
            thresholded, hand_segment = hand
            # Drawing contours around hand segment
            cv2.drawContours(frame_copy, [hand_segment + (ROI_right,ROI_top)],
                             -1, (255, 0, 0),1)
            cv2.imshow("Thesholded Hand Image", thresholded)
            thresholded = cv2.resize(thresholded, (64, 64))
            thresholded = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2RGB)
            thresholded = np.reshape(thresholded,(1,thresholded.shape[0],thresholded.shape[1],3))
            pred = model.predict(thresholded)
          cv2.putText(frame_copy, word_dict[np.argmax(pred)],(170, 45),
                      cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

    # Draw ROI on frame_copy
    cv2.rectangle(frame_copy, (ROI_left, ROI_top),
                  (ROI_right,ROI_bottom), (255,128,0), 3)
    # incrementing the number of frames for tracking
    num_frames += 1

    # Display the frame with segmented hand
    cv2.putText(frame_copy, "Hand sign recognition",
    (10, 20), cv2.FONT_ITALIC, 0.5, (51,255,51), 1)
    cv2.imshow("Sign Detection", frame_copy)
    # Close windows with Esc
    k = cv2.waitKey(1) & 0xFF
    if k == 'q':
        break

# Release the camera and destroy all the windows
cam.release()
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
```

OUTPUT:


localhost:8888/notebooks/Real%20Time%20video%20Analysis%20-%20%20Speech.ipynb
jupyter Real Time video Analysis - Speech Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View
Not Trusted
Python 3 (ipykernel)
Sign Detection
Hand sign recognition
F
In [*]:
cv2.drawContours(frame_copy, [hand_segment + (ROI_right,ROI_top)],
-1, (255, 0, 0),1)
cv2.imshow("Thesholded Hand Image", thresholded)
thresholded = cv2.resize(thresholded, (64, 64))