Real-Time Communication System Powered By AI For Specially Abled

TEAM ID - 2022TMID21808

Image Preprocessing

Import ImageDataGenerator Library And Configure It

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

In [3]:
```
train_datagen=ImageDataGenerator(rescale=1./255,horizontal_flip=True,vertical_flip=True,zoom_range=0.2)
```

In [4]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply ImageDataGenerator Functionality To Train And Test Set

In [5]:
```
x_train=train_datagen.flow_from_directory(r"C:\Users\rajes\Desktop\Dataset\training_set",target_size=(64,64
                                          class_mode="categorical",batch_size=30)
```

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

In [6]:
```
x_test=test_datagen.flow_from_directory(r"C:\Users\rajes\Desktop\Dataset\test_set",target_size=(64,64),
                                        class_mode="categorical",batch_size=30)
```

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

Model Building

Import The Required Model Building Libraries

In [7]:
```
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 [12]:
```
model=Sequential()
print("Model Initialized Successfully")
```

```
Model Initialized Successfully
```

Add The Convolution Layer

In [13]:
```
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 [16]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

Add The Flatten Layer

In [17]:
```
model.add(Flatten())
```

Adding The Dense Layers

In [21]:
```
model.add(Dense(500,activation='relu'))
```

In [22]:
```
model.add(Dense(300,activation='relu'))
```

In [23]:
```
model.add(Dense(9,activation="softmax"))
```

Compile The Model

In [28]:
```
model.compile(loss="categorical_crossentropy",metrics=["accuracy"],optimizer='adam')
```

In [29]:
```
len(x_train)
```

Out[29]:
```
525
```

In [30]:
```
len(x_test)
```

Out[30]:
```
75
```

In [31]:
```
x_train.class_indices
```

Out[31]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

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")
```