```
!unzip '/content/drive/MyDrive/archive (1).zip'
```

```
      inflating: train_data/train_data/himbul/100_5161.JPG
      inflating: train_data/train_data/himbul/100_5162.JPG
      inflating: train_data/train_data/himbul/10774570945_9bc537c255_o.jpg
      inflating: train_data/train_data/himbul/11383838134_96e891a316_o.jpg
      inflating: train_data/train_data/himbul/11383857084_cdd7514746_o.jpg
      inflating: train_data/train_data/himbul/11387473723_464965720a_o.jpg
      inflating: train_data/train_data/himbul/9007817678_24db19fc63_o.jpg
      inflating: train_data/train_data/himgri/12029790563_2fe54a68aa_o.jpg
      inflating: train_data/train_data/himgri/12029822653_1185544e7b_o.jpg
      inflating: train_data/train_data/himgri/12029933015_802428e277_o.jpg
      inflating: train_data/train_data/himgri/12029987304_031b7a2d53_o.jpg
      inflating: train_data/train_data/himgri/12030165813_f5341e8ed0_o.jpg
      inflating: train_data/train_data/himgri/12030221124_1baaf905d1_o.jpg
      inflating: train_data/train_data/himgri/12030269234_1c3f5a8e8c_o.jpg
      inflating: train_data/train_data/himgri/12152037683_13e1556c41_o.jpg
      inflating: train_data/train_data/himgri/12152178844_dc29e3126f_o.jpg
      inflating: train_data/train_data/himgri/12152510436_23b62fed3b_o.jpg
      inflating: train_data/train_data/himgri/12265055185_138e3c7c70_o.jpg
      inflating: train_data/train_data/himgri/12265498144_d2e80faa3b_o.jpg
      inflating: train_data/train_data/himgri/12265561764_4814951031_o.jpg
      inflating: train_data/train_data/himgri/12265582724_61cb64e73a_o.jpg
      inflating: train_data/train_data/himgri/12265738206_2b9b25cc8a_o.jpg
      inflating: train_data/train_data/himgri/12265747116_d4b4fa3741_o.jpg
      inflating: train_data/train_data/himgri/12266053036_fcab92d8e8_o.jpg
      inflating: train_data/train_data/himgri/12266077576_1d7143aaf7_o.jpg
      inflating: train_data/train_data/himgri/12266086526_82cd337667_o.jpg
      inflating: train_data/train_data/himgri/IMG_5463.JPG
      inflating: train_data/train_data/hsparo/100_4757.JPG
      inflating: train_data/train_data/hsparo/100_4758.JPG
      inflating: train_data/train_data/hsparo/100_5039.JPG
      inflating: train_data/train_data/hsparo/100_5040.JPG
      inflating: train_data/train_data/hsparo/100_5041.JPG
      inflating: train_data/train_data/hsparo/100_5048.JPG
      inflating: train_data/train_data/hsparo/100_5049.JPG
      inflating: train_data/train_data/hsparo/100_5050.JPG
      inflating: train_data/train_data/hsparo/100_5572.JPG
      inflating: train_data/train_data/indvul/DSC_0502.jpg
      inflating: train_data/train_data/indvul/DSC_0571e.jpg
      inflating: train_data/train_data/indvul/DSC_0572.jpg
      inflating: train_data/train_data/indvul/DSC_0576e.jpg
      inflating: train_data/train_data/indvul/DSC_0582.jpg
      inflating: train_data/train_data/indvul/DSC_0583e.jpg
      inflating: train_data/train_data/indvul/DSC_0584.jpg
      inflating: train_data/train_data/indvul/DSC_0616c.jpg
      inflating: train_data/train_data/indvul/DSC_0617.jpg
      inflating: train_data/train_data/jglowl/12152151476_7a1524aabb_o.jpg
      inflating: train_data/train_data/jglowl/DSC01335.jpg
      inflating: train_data/train_data/jglowl/DSC01336.jpg
      inflating: train_data/train_data/jglowl/_D32_10285.jpg
      inflating: train_data/train_data/jglowl/_D32_10578.jpg
      inflating: train_data/train_data/jglowl/_D32_10583.jpg
      inflating: train_data/train_data/lbicrw/100_4037.JPG
      inflating: train_data/train_data/lbicrw/100_4912.JPG
      inflating: train_data/train_data/lbicrw/100_4913.JPG
      inflating: train_data/train_data/lbicrw/100_4914.JPG
      inflating: train_data/train_data/lbicrw/100_4915.JPG
      inflating: train_data/train_data/lbicrw/100_4916.JPG
      inflating: train_data/train_data/mgprob/100_5587.JPG
```

```
# Data Augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_gen = ImageDataGenerator(rescale=(1./255),horizontal_flip=True,shear_range=0.2)
test_gen = ImageDataGenerator(rescale=(1./255))  #--> (0 to 255) convert to (0 to 1)


train = train_gen.flow_from_directory('/content/train_data/train_data',
                                       target_size=(120, 120),
                                       class_mode='categorical',
                                       batch_size=8)
test = test_gen.flow_from_directory('/content/test_data/test_data',
                                     target_size=(120, 120),
                                       class_mode='categorical',
                                       batch_size=8)
```

```
    Found 150 images belonging to 16 classes.
    Found 157 images belonging to 16 classes.
```

```
train.class_indices
```

```
{'blasti': 0,
 'bonegl': 1,
 'brhkyt': 2,
 'cbrtsh': 3,
 'cmnmyn': 4,
 'gretit': 5,
 'hilpig': 6,
 'himbul': 7,
 'himgri': 8,
 'hsparo': 9,
 'indvul': 10,
 'jglowl': 11,
 'lbicrw': 12,
 'mgprob': 13,
 'rebimg': 14,
 'wcrsrt': 15}
```

```
# CNN
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
from tensorflow.keras.models import Sequential
```

```
model = Sequential()
model.add(Convolution2D(20,(3,3),activation='relu',input_shape=(120, 120, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(45,activation='relu'))
model.add(Dense(16,activation='softmax'))
```

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

```
model.fit(train,batch_size=16,validation_data=test,epochs=20)
```

```
Epoch 1/20
19/19 [==============================] - 89s 5s/step - loss: 3.7453 - accuracy: 0.1533 - val_loss: 2.7484 - val_accuracy: 0.1274
Epoch 2/20
19/19 [==============================] - 83s 5s/step - loss: 2.6261 - accuracy: 0.1933 - val_loss: 2.7914 - val_accuracy: 0.1847
Epoch 3/20
19/19 [==============================] - 83s 5s/step - loss: 2.3595 - accuracy: 0.3067 - val_loss: 2.7364 - val_accuracy: 0.2038
Epoch 4/20
19/19 [==============================] - 84s 5s/step - loss: 2.2175 - accuracy: 0.3000 - val_loss: 2.5962 - val_accuracy: 0.2102
Epoch 5/20
19/19 [==============================] - 83s 5s/step - loss: 2.1155 - accuracy: 0.3867 - val_loss: 2.5395 - val_accuracy: 0.1975
Epoch 6/20
19/19 [==============================] - 82s 4s/step - loss: 1.8793 - accuracy: 0.4133 - val_loss: 2.7106 - val_accuracy: 0.2229
Epoch 7/20
19/19 [==============================] - 85s 5s/step - loss: 1.6822 - accuracy: 0.5000 - val_loss: 2.8509 - val_accuracy: 0.2866
Epoch 8/20
19/19 [==============================] - 83s 5s/step - loss: 1.5243 - accuracy: 0.5400 - val_loss: 2.5815 - val_accuracy: 0.2229
Epoch 9/20
19/19 [==============================] - 83s 5s/step - loss: 1.3815 - accuracy: 0.6000 - val_loss: 2.6106 - val_accuracy: 0.1911
Epoch 10/20
19/19 [==============================] - 82s 4s/step - loss: 1.2442 - accuracy: 0.6200 - val_loss: 2.9040 - val_accuracy: 0.2229
Epoch 11/20
19/19 [==============================] - 82s 4s/step - loss: 1.0892 - accuracy: 0.6867 - val_loss: 2.9905 - val_accuracy: 0.2484
Epoch 12/20
19/19 [==============================] - 82s 4s/step - loss: 0.9467 - accuracy: 0.7000 - val_loss: 2.8541 - val_accuracy: 0.2229
Epoch 13/20
19/19 [==============================] - 82s 4s/step - loss: 0.8378 - accuracy: 0.7733 - val_loss: 3.0563 - val_accuracy: 0.2675
Epoch 14/20
19/19 [==============================] - 82s 4s/step - loss: 0.6851 - accuracy: 0.8400 - val_loss: 2.8489 - val_accuracy: 0.1975
Epoch 15/20
19/19 [==============================] - 82s 4s/step - loss: 0.5179 - accuracy: 0.8600 - val_loss: 3.1108 - val_accuracy: 0.2038
Epoch 16/20
19/19 [==============================] - 82s 5s/step - loss: 0.3999 - accuracy: 0.8800 - val_loss: 3.2028 - val_accuracy: 0.2293
Epoch 17/20
19/19 [==============================] - 82s 4s/step - loss: 0.3219 - accuracy: 0.9400 - val_loss: 3.7455 - val_accuracy: 0.2420
Epoch 18/20
19/19 [==============================] - 82s 4s/step - loss: 0.2629 - accuracy: 0.9533 - val_loss: 3.2475 - val_accuracy: 0.2420
Epoch 19/20
19/19 [==============================] - 81s 4s/step - loss: 0.1614 - accuracy: 0.9733 - val_loss: 3.6267 - val_accuracy: 0.2420
Epoch 20/20
19/19 [==============================] - 81s 4s/step - loss: 0.0939 - accuracy: 0.9933 - val_loss: 3.7223 - val_accuracy: 0.2102
<keras.callbacks.History at 0x7fe31c04c8b0>
```

```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
# Testing
import numpy as np
from tensorflow.keras.preprocessing import image
# Testing 1
img1 = image.load_img('/content/gretit.JPG',target_size=(120,120))
img1 = image.img_to_array(img1)
img1 = np.expand_dims(img1,axis=0)
pred = np.argmax(model.predict(img1))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 128ms/step
    7
    himbul
```

```
# Testing 2
img2 = image.load_img('/content/hsparo.JPG',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 19ms/step
    9
    hsparo
```

```
# Testing 3
img2 = image.load_img('/content/bonegl.jpg',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 18ms/step
    1
    bonegl
```

```
# Testing 4
img2 = image.load_img('/content/brhkyt.JPG',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 28ms/step
    13
    mgprob
```

```
# Testing 5
img2 = image.load_img('/content/himgri.jpg',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 19ms/step
    8
    himgri
```

Model Tuning

```
model = Sequential()
model.add(Convolution2D(12,(3,3),activation='relu',input_shape=(120, 120, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Convolution2D(24,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Convolution2D(36,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(62,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(16,activation='relu'))
model.add(Dense(16,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 118, 118, 12)      336       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 59, 59, 12)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 57, 57, 24)        2616      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 28, 28, 24)       0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 26, 26, 36)        7812      
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 13, 13, 36)       0         
 2D)                                                             
                                                                 
 flatten_2 (Flatten)         (None, 6084)              0         
                                                                 
 dense_4 (Dense)             (None, 62)                377270    
                                                                 
 dense_5 (Dense)             (None, 32)                2016      
                                                                 
 dense_6 (Dense)             (None, 16)                528       
                                                                 
 dense_7 (Dense)             (None, 16)                272       
                                                                 
=================================================================
Total params: 390,850
Trainable params: 390,850
Non-trainable params: 0
_________________________________________________________________
```

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

```
model.fit(train,batch_size=8,validation_data=test,epochs=50)
```

```
19/19 [==============================] - 81s 4s/step - loss: 0.3712 - accuracy: 0.8800 - val_loss: 6.6775 - val_accuracy: 0.2611
Epoch 32/50
19/19 [==============================] - 117s 6s/step - loss: 0.3936 - accuracy: 0.8733 - val_loss: 6.4761 - val_accuracy: 0.2739
Epoch 33/50
19/19 [==============================] - 81s 4s/step - loss: 0.3207 - accuracy: 0.9067 - val_loss: 6.4551 - val_accuracy: 0.2994
Epoch 34/50
19/19 [==============================] - 81s 4s/step - loss: 0.2841 - accuracy: 0.9067 - val_loss: 6.3452 - val_accuracy: 0.2994
Epoch 35/50
19/19 [==============================] - 82s 4s/step - loss: 0.2364 - accuracy: 0.9333 - val_loss: 6.5411 - val_accuracy: 0.3121
Epoch 36/50
19/19 [==============================] - 82s 4s/step - loss: 0.1808 - accuracy: 0.9400 - val_loss: 7.4355 - val_accuracy: 0.3185
Epoch 37/50
19/19 [==============================] - 116s 6s/step - loss: 0.1125 - accuracy: 0.9733 - val_loss: 8.1777 - val_accuracy: 0.3121
Epoch 38/50
19/19 [==============================] - 116s 6s/step - loss: 0.1198 - accuracy: 0.9600 - val_loss: 8.0572 - val_accuracy: 0.3121
Epoch 39/50
19/19 [==============================] - 80s 4s/step - loss: 0.2090 - accuracy: 0.9267 - val_loss: 7.2143 - val_accuracy: 0.3057
Epoch 40/50
19/19 [==============================] - 80s 4s/step - loss: 0.3865 - accuracy: 0.8733 - val_loss: 10.3709 - val_accuracy: 0.3057
Epoch 41/50
19/19 [==============================] - 115s 6s/step - loss: 0.7138 - accuracy: 0.8267 - val_loss: 6.5822 - val_accuracy: 0.2675
Epoch 42/50
19/19 [==============================] - 80s 4s/step - loss: 0.4162 - accuracy: 0.8800 - val_loss: 6.4207 - val_accuracy: 0.3248
Epoch 43/50
19/19 [==============================] - 80s 4s/step - loss: 0.3536 - accuracy: 0.8933 - val_loss: 7.0266 - val_accuracy: 0.2102
Epoch 44/50
19/19 [==============================] - 81s 4s/step - loss: 0.5137 - accuracy: 0.8467 - val_loss: 6.8031 - val_accuracy: 0.2994
Epoch 45/50
19/19 [==============================] - 81s 4s/step - loss: 0.3416 - accuracy: 0.9000 - val_loss: 6.7521 - val_accuracy: 0.3057
Epoch 46/50
19/19 [==============================] - 79s 4s/step - loss: 0.1323 - accuracy: 0.9800 - val_loss: 7.1598 - val_accuracy: 0.3439
Epoch 47/50
19/19 [==============================] - 80s 4s/step - loss: 0.0839 - accuracy: 0.9800 - val_loss: 7.8019 - val_accuracy: 0.3057
Epoch 48/50
19/19 [==============================] - 80s 4s/step - loss: 0.0600 - accuracy: 0.9867 - val_loss: 8.8786 - val_accuracy: 0.2930
```

```
# Testing 1
img1 = image.load_img('/content/gretit.JPG',target_size=(120,120))
img1 = image.img_to_array(img1)
img1 = np.expand_dims(img1,axis=0)
pred = np.argmax(model.predict(img1))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
1/1 [==============================] - 0s 18ms/step
3
cbrtsh
```

```
# Testing 2
img2 = image.load_img('/content/hsparo.JPG',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
1/1 [==============================] - 0s 20ms/step
9
hsparo
```

```
# Testing 3
img2 = image.load_img('/content/bonegl.jpg',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
1/1 [==============================] - 0s 22ms/step
1
bonegl
```

```
# Testing 4
img2 = image.load_img('/content/brhkyt.JPG',target_size=(120,120))
img2 = image.img_to_array(img2)
```

```
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 19ms/step
    15
    wcrsrt
```

```
# Testing 5
img2 = image.load_img('/content/himgri.jpg',target_size=(120,120))
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2,axis=0)
pred = np.argmax(model.predict(img2))
print(pred)
output = ['blasti','bonegl','brhkyt','cbrtsh','cmnmyn','gretit','hilpig','himbul','himgri','hsparo','indvul','jglowl','lbicrw','mgprob','rebi
print(output[pred])
```

```
    1/1 [==============================] - 0s 25ms/step
    8
    himgri
```

Transfer Learning

```
from tensorflow.keras.layers import Dense,Flatten,Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
import numpy as np
```

```
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
# Adding the preprocessing layer to the front of vgg

vgg = VGG16(include_top=False,weights='imagenet',input_shape=(224,224,3))
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
    58889256/58889256 [==============================] - 3s 0us/step
```

```
#Train model with existing weights
for layer in vgg.layers:
  print(layer)
```

```
    <keras.engine.input_layer.InputLayer object at 0x7f4ef5a2b6a0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b1ea9b0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b1eba90>
    <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f4e6b2b0af0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b1ebee0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b2b1db0>
    <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f4e6b2b2fb0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b2b3af0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b2b30a0>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b2b15a0>
    <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f4e6b2b0430>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e6b2b2d70>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e68101f00>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e68102f50>
    <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f4e68103f10>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e68101060>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e68102860>
    <keras.layers.convolutional.conv2d.Conv2D object at 0x7f4e68111c60>
    <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f4e68112590>
```

```
# Train model with existing weights

for layer in vgg.layers:
  layer.trainable=False
```

```
x = Flatten()(vgg.output)
```

```
# output layer

prediction = Dense(4,activation='softmax')(x)


# Create Vgg16 model

model = Model(inputs=vgg.input,outputs=prediction)
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0         
                                                                 
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0         
                                                                 
 flatten (Flatten)           (None, 25088)             0         
                                                                 
 dense (Dense)               (None, 4)                 100356    
                                                                 
=================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688
_________________________________________________________________
```

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

```
model.fit_generator(train,validation_data=test,epochs=4,steps_per_epoch=len(train),validation_steps=len(test))
```

```
<ipython-input-24-9596add3a7e4>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
  model.fit_generator(train,validation_data=test,epochs=4,steps_per_epoch=len(train),validation_steps=len(test))
```