twqycgs0l

May 28, 2023

0.1 *Import the necessary libraries*

[20]:

```
# import the libraries as shown below

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.resnet50 import ResNet50
#from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

0.2 *Resize*

- The code below sets IMAGE_SIZE variable is set to [224, 224], indicating the desired size for resizing the images. This means that all images will be resized to a shape of 224x224 pixels.
- The code snippet you provided defines these variables as paths to the directories where the training and validation images are stored. These paths will be used later when loading and processing the image data for training and evaluation.

[21]:

```
# re-size all the images to this
IMAGE_SIZE = [224, 224]

train_path = '/content/drive/MyDrive/Datasets/Train'
valid_path = '/content/drive/MyDrive/Datasets/Test'
```

- A ResNet50 model is instantiated using the Keras ResNet50 function
- By instantiating the ResNet50 model with these parameters, you create a model that is pre-trained on ImageNet and has a convolutional base that can be used as a feature extractor. This base can then be combined with additional layers to build a custom classification model or perform other tasks.

[22]:
```
# Import the Vgg 16 library as shown below and add preprocessing layer to the␣
 ↪front of VGG
# Here we will be using imagenet weights

resnet = ResNet50(input_shape=IMAGE_SIZE + [3], weights='imagenet',␣
 ↪include_top=False)
```

0.3 *don't train existing weights*

- Demonstrates how to freeze (make non-trainable) all the layers in the ResNet50 model. Additionally, it retrieves the number of output classes based on the subdirectories present in the training dataset.

[34]:
```
# don't train existing weights
for layer in resnet.layers:
    layer.trainable = False



# useful for getting number of output classes
folders = glob('/content/drive/MyDrive/Datasets/Train/*')
```

0.4 *Defining Layers*

- A new dense layer with an activation function of softmax is added to the output of the ResNet50 model. This additional layer serves as the final prediction layer for the model.
- Next, a new Model object is created by specifying the inputs as the input layer of the ResNet50 model and the outputs as the prediction layer. This allows for the creation of a model that takes the pre-trained ResNet50 model as its base and adds the custom prediction layer on top.
- To inspect the structure of the model, the summary() method is called on the model object, which provides a summary of the model's architecture, including the number of parameters in each layer.
- In short, this adds a prediction layer to the ResNet50 model, creates a new model object by specifying the inputs and outputs, and then displays the summary of the model's structure using the model.summary() method. This enables a quick overview of the model's architecture and parameter counts.

[35]:
```
# our layers - you can add more if you want
x = Flatten()(resnet.output)
```

[36]:
```
prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=resnet.input, outputs=prediction)
# view the structure of the model
model.summary()
```

```
Model: "model_4"
--------------------------------------------------------------------------------
------------------
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================
==================
 input_3 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0
['input_3[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472
['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256
['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0
['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0
['conv1_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0
['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160
['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256
['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0
['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928
['conv2_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256
['conv2_block1_2_conv[0][0]']
 ization)
 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0
['conv2_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640
['pool1_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640
['conv2_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024
['conv2_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024
['conv2_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0
['conv2_block1_0_bn[0][0]',
'conv2_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0
['conv2_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448
['conv2_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256
['conv2_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0
['conv2_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928
['conv2_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256
['conv2_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0
['conv2_block2_2_bn[0][0]']
 n)
 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640
['conv2_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024
['conv2_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0
['conv2_block1_out[0][0]',
'conv2_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0
['conv2_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448
['conv2_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256
['conv2_block3_1_conv[0][0]']
 ization)

 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0
['conv2_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928
['conv2_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256
['conv2_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0
['conv2_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640
['conv2_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024
['conv2_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0
['conv2_block2_out[0][0]',
'conv2_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 56, 56, 256)  0
```

```
['conv2_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896
['conv2_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584
['conv3_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584
['conv2_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048
['conv3_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048
['conv3_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048
['conv3_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0
['conv3_block1_0_bn[0][0]',
'conv3_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0
['conv3_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664
['conv3_block1_out[0][0]']
```

```
 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584
['conv3_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048
['conv3_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048
['conv3_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0
['conv3_block1_out[0][0]',
'conv3_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0
['conv3_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664
['conv3_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584
['conv3_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048
['conv3_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048
['conv3_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0
['conv3_block2_out[0][0]',
'conv3_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 28, 28, 512)  0
['conv3_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664
['conv3_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584
['conv3_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512
['conv3_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0
['conv3_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048
['conv3_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048
['conv3_block4_3_conv[0][0]']
```

```
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0
['conv3_block3_out[0][0]',
'conv3_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0
['conv3_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328
['conv3_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080
['conv4_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312
['conv3_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168
['conv4_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0
['conv4_block1_0_bn[0][0]',
                                )
'conv4_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0
['conv4_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400
['conv4_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080
['conv4_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168
['conv4_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 14, 14, 1024  0
['conv4_block1_out[0][0]',
                                )
'conv4_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0
['conv4_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400
```

```
['conv4_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080
['conv4_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168
['conv4_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0
['conv4_block2_out[0][0]',
                                )
'conv4_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0
['conv4_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400
['conv4_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block4_1_bn[0][0]']
 n)
 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080
['conv4_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168
['conv4_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0
['conv4_block3_out[0][0]',
                                )
'conv4_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0
['conv4_block4_add[0][0]']
                                )

 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400
['conv4_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080
['conv4_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168
['conv4_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block5_3_conv[0][0]']
 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0
['conv4_block4_out[0][0]',
                                )
'conv4_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0
['conv4_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400
['conv4_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080
['conv4_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024
['conv4_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0
['conv4_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168
['conv4_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096
['conv4_block6_3_conv[0][0]']
```

```
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0
['conv4_block5_out[0][0]',
                                )
'conv4_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0
['conv4_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800
['conv4_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048
['conv5_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0
['conv5_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808
['conv5_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048
['conv5_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0
['conv5_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200
['conv4_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624
['conv5_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192
['conv5_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192
['conv5_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0
```

```
['conv5_block1_0_bn[0][0]',
'conv5_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0
['conv5_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088
['conv5_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048
['conv5_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0
['conv5_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808
['conv5_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048
['conv5_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0
['conv5_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624
['conv5_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192
['conv5_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0
['conv5_block1_out[0][0]',
'conv5_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0
['conv5_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088
['conv5_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048
['conv5_block3_1_conv[0][0]']
 ization)
 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0
['conv5_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808
['conv5_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048
['conv5_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0
['conv5_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624
['conv5_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192
['conv5_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0
['conv5_block2_out[0][0]',
'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0
['conv5_block3_add[0][0]']

 flatten_4 (Flatten)            (None, 100352)       0
['conv5_block3_out[0][0]']

 dense_4 (Dense)                (None, 3)            301059
['flatten_4[0][0]']

================================================================================
==================
Total params: 23,888,771
Trainable params: 301,059
Non-trainable params: 23,587,712
--------------------------------------------------------------------------------
------------------
```

0.5 *Defining Optimizers*

- The loss function is set to 'categorical_crossentropy', which is commonly used for multi-class classification problems.

- The optimizer is set to ‘adam’, which is an efficient optimization algorithm commonly used in deep learning. Finally, the metric is set to [‘accuracy’], indicating that the model’s accuracy will be monitored during training.

[37]:
```
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

0.6 *Image Data Generator to import the images from the dataset*

- **train_datagen** is an instance of ImageDataGenerator that performs various data augmentation techniques on the training dataset. It rescales the pixel values to the range of 0 to 1, applies shear transformations, zoom transformations, and horizontal flipping.
- **test_datagen** is another instance of ImageDataGenerator that is used for the testing dataset. It simply rescales the pixel values to the range of 0 to 1.
- These **ImageDataGenerator** objects are used to preprocess and augment the image data, enhancing the model’s ability to generalize and improving its performance during training and evaluation.

[38]:
```
# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

[39]:
```
# Make sure you provide the same target size as initialied for the image size
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/
 ↪Datasets/Train',
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 64 images belonging to 3 classes.
```

[40]:
```
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/Datasets/
 ↪Test',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 58 images belonging to 3 classes.
```

- In the code below, model.fit_generator() is called to train the model. The training_set and test_set are provided as the training and validation data respectively.
- The epochs parameter is set to 50, indicating the number of times the model will iterate over the entire training dataset. The steps_per_epoch parameter is set to len(training_set), which represents the number of batches to be processed in one epoch.
- The validation_steps parameter is set to len(test_set), representing the number of batches to be processed for validation.

[41]:

```
# fit the model
# Run the cell. It will take some time to execute
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=50,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
<ipython-input-41-69229fe26ea3>:3: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  r = model.fit_generator(

Epoch 1/50
2/2 [==============================] - 32s 27s/step - loss: 7.9217 - accuracy:
0.3438 - val_loss: 5.9726 - val_accuracy: 0.3276
Epoch 2/50
2/2 [==============================] - 1s 788ms/step - loss: 5.7704 - accuracy:
0.3281 - val_loss: 7.7077 - val_accuracy: 0.1552
Epoch 3/50
2/2 [==============================] - 2s 1s/step - loss: 7.8773 - accuracy:
0.3125 - val_loss: 5.2780 - val_accuracy: 0.5172
Epoch 4/50
2/2 [==============================] - 2s 1s/step - loss: 5.1023 - accuracy:
0.3750 - val_loss: 1.7908 - val_accuracy: 0.5172
Epoch 5/50
2/2 [==============================] - 1s 781ms/step - loss: 1.7307 - accuracy:
0.4531 - val_loss: 5.5907 - val_accuracy: 0.3276
Epoch 6/50
2/2 [==============================] - 1s 816ms/step - loss: 3.4373 - accuracy:
0.4375 - val_loss: 5.5116 - val_accuracy: 0.2241
Epoch 7/50
2/2 [==============================] - 1s 825ms/step - loss: 3.1637 - accuracy:
0.4375 - val_loss: 2.5729 - val_accuracy: 0.2586
Epoch 8/50
2/2 [==============================] - 1s 773ms/step - loss: 1.1385 - accuracy:
```

```
0.6094 - val_loss: 1.9350 - val_accuracy: 0.6034
Epoch 9/50
2/2 [==============================] - 1s 787ms/step - loss: 2.5177 - accuracy:
0.4688 - val_loss: 1.8361 - val_accuracy: 0.5517
Epoch 10/50
2/2 [==============================] - 2s 1s/step - loss: 1.7923 - accuracy:
0.5000 - val_loss: 1.9295 - val_accuracy: 0.3966
Epoch 11/50
2/2 [==============================] - 2s 1s/step - loss: 1.6718 - accuracy:
0.5156 - val_loss: 3.0971 - val_accuracy: 0.1897
Epoch 12/50
2/2 [==============================] - 1s 774ms/step - loss: 1.4143 - accuracy:
0.5469 - val_loss: 1.2871 - val_accuracy: 0.5345
Epoch 13/50
2/2 [==============================] - 1s 793ms/step - loss: 0.8180 - accuracy:
0.6406 - val_loss: 1.7104 - val_accuracy: 0.5517
Epoch 14/50
2/2 [==============================] - 1s 792ms/step - loss: 1.5328 - accuracy:
0.5312 - val_loss: 1.1939 - val_accuracy: 0.5690
Epoch 15/50
2/2 [==============================] - 1s 778ms/step - loss: 0.6655 - accuracy:
0.7031 - val_loss: 1.8224 - val_accuracy: 0.3793
Epoch 16/50
2/2 [==============================] - 1s 755ms/step - loss: 1.0021 - accuracy:
0.5938 - val_loss: 1.4360 - val_accuracy: 0.5000
Epoch 17/50
2/2 [==============================] - 2s 1s/step - loss: 0.6618 - accuracy:
0.7188 - val_loss: 1.0450 - val_accuracy: 0.6379
Epoch 18/50
2/2 [==============================] - 2s 1s/step - loss: 0.6109 - accuracy:
0.7344 - val_loss: 1.0000 - val_accuracy: 0.6379
Epoch 19/50
2/2 [==============================] - 1s 780ms/step - loss: 0.4752 - accuracy:
0.7500 - val_loss: 1.1091 - val_accuracy: 0.5862
Epoch 20/50
2/2 [==============================] - 1s 798ms/step - loss: 0.5467 - accuracy:
0.7656 - val_loss: 1.0399 - val_accuracy: 0.6034
Epoch 21/50
2/2 [==============================] - 1s 751ms/step - loss: 0.4572 - accuracy:
0.7656 - val_loss: 0.9025 - val_accuracy: 0.6552
Epoch 22/50
2/2 [==============================] - 1s 795ms/step - loss: 0.3869 - accuracy:
0.8438 - val_loss: 0.9922 - val_accuracy: 0.6552
Epoch 23/50
2/2 [==============================] - 1s 804ms/step - loss: 0.4395 - accuracy:
0.8125 - val_loss: 0.9067 - val_accuracy: 0.6379
Epoch 24/50
2/2 [==============================] - 2s 1s/step - loss: 0.3702 - accuracy:
```

```
0.8438 - val_loss: 0.9397 - val_accuracy: 0.6552
Epoch 25/50
2/2 [==============================] - 2s 1s/step - loss: 0.3864 - accuracy:
0.8594 - val_loss: 0.9560 - val_accuracy: 0.6379
Epoch 26/50
2/2 [==============================] - 1s 766ms/step - loss: 0.3910 - accuracy:
0.8125 - val_loss: 0.8688 - val_accuracy: 0.6724
Epoch 27/50
2/2 [==============================] - 1s 784ms/step - loss: 0.3458 - accuracy:
0.8281 - val_loss: 0.9024 - val_accuracy: 0.6552
Epoch 28/50
2/2 [==============================] - 1s 780ms/step - loss: 0.3111 - accuracy:
0.9062 - val_loss: 0.9184 - val_accuracy: 0.6552
Epoch 29/50
2/2 [==============================] - 1s 758ms/step - loss: 0.2805 - accuracy:
0.9062 - val_loss: 0.8585 - val_accuracy: 0.6724
Epoch 30/50
2/2 [==============================] - 1s 811ms/step - loss: 0.3407 - accuracy:
0.8438 - val_loss: 0.8552 - val_accuracy: 0.6897
Epoch 31/50
2/2 [==============================] - 1s 759ms/step - loss: 0.3059 - accuracy:
0.9062 - val_loss: 0.9444 - val_accuracy: 0.6552
Epoch 32/50
2/2 [==============================] - 2s 1s/step - loss: 0.3080 - accuracy:
0.8906 - val_loss: 0.8685 - val_accuracy: 0.6724
Epoch 33/50
2/2 [==============================] - 2s 1s/step - loss: 0.2865 - accuracy:
0.8906 - val_loss: 0.8680 - val_accuracy: 0.6552
Epoch 34/50
2/2 [==============================] - 1s 757ms/step - loss: 0.2874 - accuracy:
0.9062 - val_loss: 0.8716 - val_accuracy: 0.6552
Epoch 35/50
2/2 [==============================] - 1s 773ms/step - loss: 0.2633 - accuracy:
0.9375 - val_loss: 0.8279 - val_accuracy: 0.7759
Epoch 36/50
2/2 [==============================] - 1s 787ms/step - loss: 0.2832 - accuracy:
0.8750 - val_loss: 0.9756 - val_accuracy: 0.6207
Epoch 37/50
2/2 [==============================] - 1s 753ms/step - loss: 0.3337 - accuracy:
0.8594 - val_loss: 0.8837 - val_accuracy: 0.6724
Epoch 38/50
2/2 [==============================] - 1s 777ms/step - loss: 0.2174 - accuracy:
0.9531 - val_loss: 0.8992 - val_accuracy: 0.7069
Epoch 39/50
2/2 [==============================] - 1s 783ms/step - loss: 0.3631 - accuracy:
0.8281 - val_loss: 0.8766 - val_accuracy: 0.6552
Epoch 40/50
2/2 [==============================] - 2s 1s/step - loss: 0.3194 - accuracy:
```

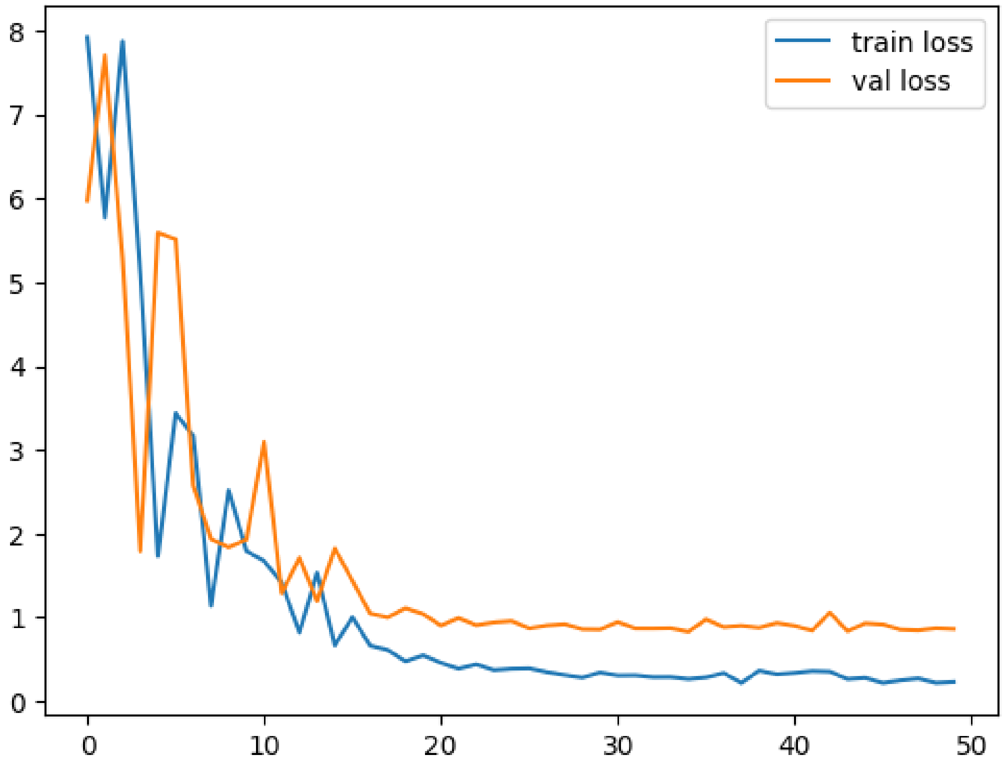
```
0.9062 - val_loss: 0.9329 - val_accuracy: 0.6552
Epoch 41/50
2/2 [==============================] - 1s 783ms/step - loss: 0.3358 - accuracy:
0.8594 - val_loss: 0.8983 - val_accuracy: 0.6897
Epoch 42/50
2/2 [==============================] - 1s 805ms/step - loss: 0.3583 - accuracy:
0.7969 - val_loss: 0.8428 - val_accuracy: 0.7241
Epoch 43/50
2/2 [==============================] - 1s 769ms/step - loss: 0.3520 - accuracy:
0.8281 - val_loss: 1.0566 - val_accuracy: 0.6379
Epoch 44/50
2/2 [==============================] - 1s 776ms/step - loss: 0.2633 - accuracy:
0.9219 - val_loss: 0.8351 - val_accuracy: 0.7414
Epoch 45/50
2/2 [==============================] - 1s 741ms/step - loss: 0.2783 - accuracy:
0.9062 - val_loss: 0.9286 - val_accuracy: 0.6379
Epoch 46/50
2/2 [==============================] - 1s 776ms/step - loss: 0.2180 - accuracy:
0.9688 - val_loss: 0.9152 - val_accuracy: 0.7069
Epoch 47/50
2/2 [==============================] - 2s 1s/step - loss: 0.2483 - accuracy:
0.9062 - val_loss: 0.8536 - val_accuracy: 0.7414
Epoch 48/50
2/2 [==============================] - 2s 1s/step - loss: 0.2705 - accuracy:
0.9062 - val_loss: 0.8451 - val_accuracy: 0.7759
Epoch 49/50
2/2 [==============================] - 1s 777ms/step - loss: 0.2186 - accuracy:
0.9531 - val_loss: 0.8717 - val_accuracy: 0.7069
Epoch 50/50
2/2 [==============================] - 1s 764ms/step - loss: 0.2300 - accuracy:
0.9375 - val_loss: 0.8623 - val_accuracy: 0.6724
```

0.7 *Plotting the loss*

- The loss plot by using the plot() function from matplotlib.pyplot. It plots the training loss (r.history['loss']) and validation loss (r.history['val_loss']) on the same graph. The label parameter is used to provide a label for each line. The legend() function is called to display the legend, and plt.show() is used to show the plot.

[44]:

```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```

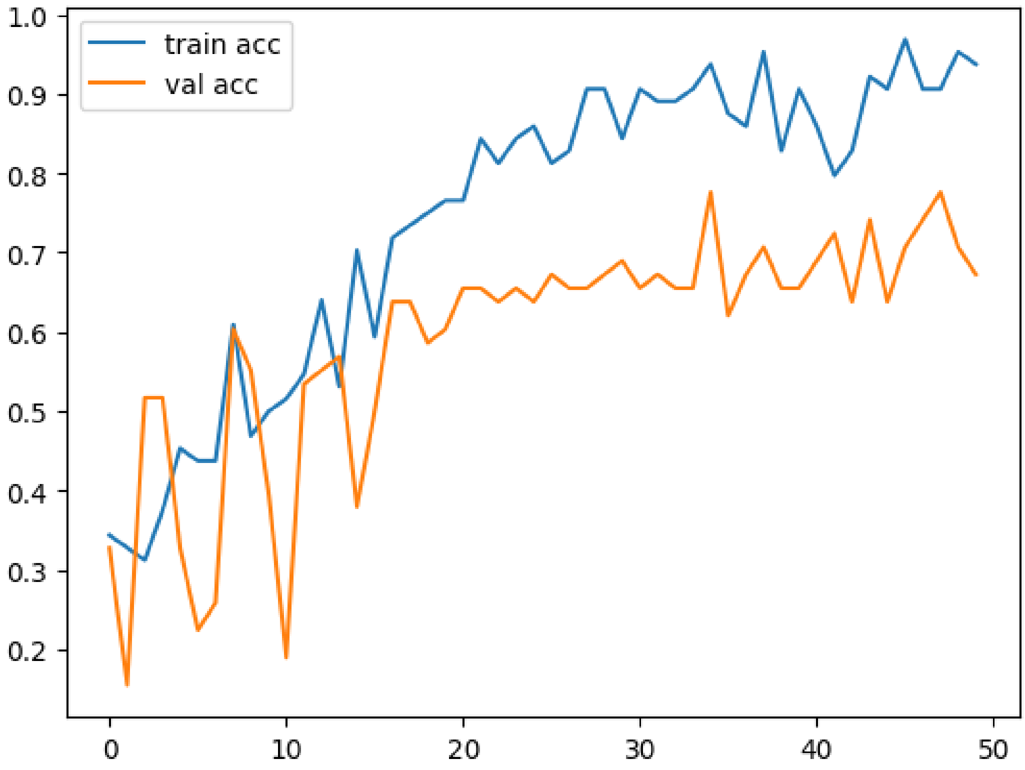


```
<Figure size 640x480 with 0 Axes>
```

0.8 *Plotting the Accuracy*

The accuracy plot by using the plot() function from matplotlib.pyplot. It plots the training accuracy (r.history['accuracy']) and validation accuracy (r.history['val_accuracy']) on the same graph. The label parameter is used to provide a label for each line. The legend() function is called to display the legend, and plt.show() is used to show the plot.

[45]:
```
# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50
train acc
val acc

```
<Figure size 640x480 with 0 Axes>
```

```
[ ]:
```