Annapoornima S

225229101

PDL Lab13. Image classification using Pre-trained CNN Models

In [1]:

```
import tensorflow as tf
import keras
```

In [2]:

```
from keras.preprocessing.image import *
```

1. IMPORT AND CREATE

In [4]:

```
from keras.applications import VGG16
model = VGG16(weights='imagenet', include_top=True)
```

In [5]:

```
print(model.summary())
```

```
Model: "vgg16"
_________________________________________________________________
 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         
                                                                 
 fc1 (Dense)                 (None, 4096)              102764544 
                                                                 
 fc2 (Dense)                 (None, 4096)              16781312  
                                                                 
 predictions (Dense)         (None, 1000)              4097000   
                                                                 
=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

2. USE METHODS

In [7]:

```
import numpy as np
from keras.applications.vgg16 import preprocess_input
# Load and preprocess the image
image1 = tf.keras.preprocessing.image.load_img('D:\download.png', target_size=(224, 224)
image1 = tf.keras.preprocessing.image.img_to_array(image1)
image1 = np.expand_dims(image1, axis=0)
image1 = preprocess_input(image1)
# Predict the probability across all output classes
yhat1 = model.predict(image1)
predictions1 = tf.keras.applications.vgg16.decode_predictions(yhat1, top=10)
```

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

3. PRINT PREDICTIONS

In [9]:

```
predictions1
```

Out[9]:

```
[[('n03109150', 'corkscrew', 0.18989675),
  ('n02879718', 'bow', 0.086401656),
  ('n04380533', 'table_lamp', 0.07809294),
  ('n04482393', 'tricycle', 0.07652949),
  ('n03532672', 'hook', 0.047865972),
  ('n04509417', 'unicycle', 0.034276113),
  ('n03127747', 'crash_helmet', 0.031239768),
  ('n03814639', 'neck_brace', 0.027444342),
  ('n02791124', 'barber_chair', 0.025835453),
  ('n07892512', 'red_wine', 0.0142781995)]]
```

In [10]:

```
#PART -II
```

In [11]:

```
model1 = tf.keras.applications.resnet50.ResNet50(include_top=True,weights='imagenet',inp
```

In [12]:

```
model1.summary()
```

```
Model: "resnet50"
______________________________________________________________________

____________________________
 Layer (type)                Output Shape                 Param #   Co
nnected to
======================================================================
============================
 input_2 (InputLayer)        [(None, 224, 224, 3)]        0         []

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)          0
['input_2[0][0]']

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

 conv1_bn (BatchNormalizati  (None, 112, 112, 64)         256
['conv1_conv[0][0]']
 on)
```

In [20]:

```
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
```

In [21]:

```
image3 =tf.keras.preprocessing.image.load_img('D:\download.png', target_size=(224, 224))
image3 = tf.keras.preprocessing.image.img_to_array(image3)
image3 = np.expand_dims(image3, axis=0)
image3 = tf.keras.applications.resnet50.preprocess_input(image3, data_format=None)
predictions = model1.predict(image3)
label = decode_predictions(predictions)
```

```
1/1 [==============================] - 0s 251ms/step
```

In [22]:

```
label
```

Out[22]:

```
[[('n04099969', 'rocking_chair', 0.22381556),
  ('n03109150', 'corkscrew', 0.11808882),
  ('n03272010', 'electric_guitar', 0.08733004),
  ('n06596364', 'comic_book', 0.03971554),
  ('n03785016', 'moped', 0.0339614)]]
```

PART - IV

In [14]:

```
import keras,os
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
```

In [25]:

```
import zipfile

# Specify the name of the zip file you want to create
zip_file_name = "my_archive.zip"

# Create a new zip file in write mode
with zipfile.ZipFile(zip_file_name, "w") as myzip:
    # Add files to the zip file
    myzip.write("file1.txt")
    myzip.write("file2.txt")
```

In [28]:

```
import zipfile
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Specify the name of the ZIP file
zip_file_name = "my_archive.zip"

# Extract the contents of the ZIP file to a temporary directory
extracted_dir = "temp_extracted"
with zipfile.ZipFile(zip_file_name, "r") as zip_ref:
    zip_ref.extractall(extracted_dir)

# Create an ImageDataGenerator for the extracted data
data_generator = ImageDataGenerator(
    rotation_range=90,
    brightness_range=[0.1, 0.7],
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0.15,
    preprocessing_function=preprocess_input
)

# Use flow_from_directory with the extracted directory
traindata = data_generator.flow_from_directory(
    directory=extracted_dir,
    target_size=(224, 224),
    subset="training"  # Use "training" or "validation" to specify the split
)

testdata = data_generator.flow_from_directory(
    directory=extracted_dir,
    target_size=(224, 224),
    subset="validation"  # Use "training" or "validation" to specify the split
)

# Now you can use traindata and testdata for training and testing your model
```

```
Found 0 images belonging to 0 classes.
Found 0 images belonging to 0 classes.
```

In [29]:

```
from keras.optimizers import Adam
opt = Adam(lr=0.001)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy, metrics=['accur
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_
rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

In [ ]:

```
hist = model.fit(testdata,epochs=100)
```

In [ ]:

In [ ]:

In [ ]: