Transfer Learning Assignment

Download all the data in this rar_file , it contains all the data required for the assignment. When you unrar the file you'll get the files in the following format: **path/to/the/image.tif,category**

where the categories are numbered 0 to 15, in the following order:

0 letter
1 form
2 email
3 handwritten
4 advertisement
5 scientific report
6 scientific publication
7 specification
8 file folder
9 news article
10 budget
11 invoice
12 presentation
13 questionnaire
14 resume
15 memo

There is a file named as 'labels_final.csv' , it consists of two columns. First column is path which is the required path to the images and second is the class label.

In [ ]:
```
#the dataset that you are dealing with is quite large 3.7 GB and hence there are two methods to import the data to Colab
# Method 1- you can use gdown module to get the data directly from Google drive to Colab
# the syntax is as follows !gdown --id file_id , for ex - running the below cell will import the rvl-cdip.rar dataset
```

In [ ]:
```
!gdown --id 1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
```

```
Access denied with the following error:

        Cannot retrieve the public link of the file. You may need to change
        the permission to 'Anyone with the link', or have had many accesses.

You may still be able to access the file from the browser:

         https://drive.google.com/uc?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
```

In [ ]:
```
# Method -2 you can also import the data using wget function
#https://www.youtube.com/watch?v=BPUfVq7RaY8
```

In [ ]:
```
!curl --header "Host: doc-0s-8o-docs.googleusercontent.com" --header "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36" --header "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header "Accept-Language: en-US,en;q=0.9" --header "Referer: https://drive.google.com/" --header "Cookie: AUTH_nf14kr63csq6es11jms1j175edsrf0ud_nonce=hpi3nl2752t54; ab.storage.deviceId.7af503ae-0c84-478f-98b0-ecfff5d67750=%7B%22g%22%3A%22cf6f5c5c-eb07-f7a6-15ec-ee63d656f051%22%2C%22c%22%3A1641395211601%2C%22l%22%3A1645860762557%7D; ab.storage.userId.7af503ae-0c84-478f-98b0-ecfff5d67750=%7B%22g%22%3A%223334480%22%2C%22c%22%3A1645860762542%2C%22l%22%3A1645860762563%7D; ab.storage.sessionId.7af503ae-0c84-478f-98b0-ecfff5d67750=%7B%22g%22%3A%22cc78dc55-7cd4-687b-8410-9869e79ac63e%22%2C%22e%22%3A2145860768738%2C%22c%22%3A1645860762554%2C%22l%22%3A1645860768738%7D" --header "Connection: keep-alive" "https://doc-0s-8o-docs.googleusercontent.com/docs/securesc/crnf5ifhu7cm9739i4jrpvbn7r10o8dd/16ul0t7nnh691lekugi2h2arggrpbmho/1646206425000/00484516897554883881/15475546427455647259/1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu?ax=ACxEAsb9P0l7NsrkXNZE5Pej_vgNvD2gpPS1OUPxAcP6bLuiUryYiuFEzkEmYdQ3af3yUfeB1m5Koz-3QzQ2GF2fyfGhaaL6JJEJhLcBGWlHI9rfkx7MVsCZggRGTFjUoWo2sESXkA2poGjhCDpqxSgSVoh7s4CZGnFGyVrAJuztRKzGjtvG7qV2VJ7jq6LMDSpg83H8K7HifY_Kp6s4Ne-lpWkewZQEebf-hM231eJn5U7eognZlIHQpS8heWmUDbtnjKLOX02Xmh4eIkYiFXT8FHJuDqvJGZJ9l-3p3eVehD6BqcujuOTAygBsS6n8t7wshTpW5OjAepTxUYRLiohO0gWAlLo7ItlNZY8_90m67dOkk-FIfkawRJ6NLy9qgfI8bLP_LLDfQYWGmybWpyTayn_vapuaqIumBvQsloZrt8kLuhZ7qTOHFQfbXTIZQUtccAK1O1EQP6OXI999I3OXThzLTJEg6fOC9qYEsnYFuapxCF4TR9ruRz_3YslwrpNXAJOfenBrAIGEumOuHizIKstQmB_-cjFc9vbEt2CpNXiX9ZppU9xZdWR13sWPxrKP_JEa5RwPQ8MtvRpGJlaDB-67-2SHhcoSYPTXcJVDKEdnuvkGvRcOdCguRrGVehqgpGLtCckiD5qsPmYkwMv-nlMqYtsr2ZAws7-HYQPM&authuser=0&nonce=hpi3nl2752t54&user=15475546427455647259&hash=4v1rf1mg7rgc55m65vg8f3fbm4bnvl1l" -L -o "rvl-cdip.rar"
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
```

In [ ]:
```
#unrar the file
get_ipython().system_raw("unrar x rvl-cdip.rar")
```

2. On this image data, you have to train 3 types of models as given below You have to split the data into Train and Validation data.

1. Try not to load all the images into memory, use the gernarators that we have given the reference notebooks to load the batch of images only during the train data. or you can use this method also https://medium.com/@vijayabhaskar96/tutorial-on-keras-imagedatagenerator-with-flow-from-dataframe-8bd5776e45c1

https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c

Note- In the reference notebook you were dealing with jpg images, in the given dataset you are dealing with tiff images. Imagedatagenrator works with both type of images. If you want to use custom data pipeline then you have to convert your tiff images to jpg images.

1. You are free to choose Learning rate, optimizer, loss function, image augmentation, any hyperparameters. but you have to use the same architechture what we are asking below.
2. Use tensorboard for every model and analyse your gradients. (you need to upload the screenshots for each model for evaluation)

1. You can check about Transfer Learning in this link - https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/3426/code-example-cats-vs-dogs/8/module-8-neural-networks-computer-vision-and-deep-learning </pre>

1. Do print model.summary() and draw model_plots for each of the model.

Model-1

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weighs.
2. After VGG-16 network without FC layers, add a new Conv block (1 Conv layer and 1 Maxpooling), 2 FC layers and an output layer to classify 16 classes. You are free to choose any hyperparameters/parameters of conv block, FC layers, output layer.
3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool Layer --> 2 FC layers --> Output Layer**
4.Print model.summary() and plot the architecture of the model.
Reference for plotting model
5. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

In [ ]:

In [ ]:

Model-2

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weighs.
2. After VGG-16 network without FC layers, don't use FC layers, use conv layers only as Fully connected layer.Any FC layer can be converted to a CONV layer. This conversion will reduce the No of Trainable parameters in FC layers.
For example, an FC layer with K=4096 that is looking at some input volume of size 7×7×512 can be equivalently expressed as a CONV layer with F=7,P=0,S=1,K=4096.
In other words, we are setting the filter size to be exactly the size of the input volume, and hence the output will simply be 1×1×4096 since only a single depth column “fits” across the input volume, giving identical result as the initial FC layer. You can refer this link to better understanding of using Conv layer in place of fully connected layers.
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC layers, 1 output layer for 16 class classification. **INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC -->Output Layer**
4. 4.Print model.summary() and plot the architecture of the model.
Reference for plotting model
5. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

In [ ]:

In [ ]:

Model-3

1. Use same network as Model-2 '**INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer**' and train only Last\ 6 Layers of VGG-16 network, 2 Conv layers identical to FC layers, 1 output layer.

In [ ]:

In [ ]:

Please write your observations or a brief summary of the results that you get after performing transfer learning with reference to model1, model2 and model3

In [ ]:

In [ ]:
```
import os, random
import numpy as np
import pandas as pd
import PIL
import keras
import itertools
from PIL import Image
#import ibm_boto3
#from botocore.client import Config


from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from skimage import feature, data, io, measure
from sklearn.metrics import confusion_matrix

import matplotlib.pyplot as plt
from matplotlib import ticker
import seaborn as sns
%matplotlib inline

from keras import backend as K
from keras.models import Sequential
from keras.layers import Input, Dropout, Flatten, Conv2D, MaxPooling2D, Dense, Activation
```

```
#from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
#from keras.optimizers import Adam
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Input, Dropout, Flatten, Conv2D, MaxPooling2D, Dense, Activation
#from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
from keras.utils import np_utils
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
#!pip install opencv_python
import cv2
```

In [ ]:
```
!curl --header "Host: doc-0s-8o-docs.googleusercontent.com" --header "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36" --header "Accept: text/html,application/xhtml+xml,applicat
ion/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header "Accept-Language: en-U
S,en;q=0.9" --header "Referer: https://drive.google.com/" --header "Cookie: AUTH_nf14kr63csq6es11jms1j175edsrf0ud_nonce=gp0sj6pj
cq6t0" --header "Connection: keep-alive" "https://doc-0s-8o-docs.googleusercontent.com/docs/securesc/crnf5ifhu7cm9739i4jrpvbn7r1
0o8dd/iqqsvfme4p5msrebboeqif31ufgoul8q/1646342625000/00484516897554883881/15475546427455647259/1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoE
u?ax=ACxEAsalWU-gcW9vwBrEvvlQOFmV2fPt5v67H3BXrsv_vDXd05CTyTOatqhSudYckk_4MMWF880PquENPglk3sWikh1VDqVBuk1bOvequ9Eaq693-jjiG1P-5DX
-pcZUqQyr9c1XXnyLl-X3h3TItltBeE9D_5qNFBss_WHxll0cZz7VzTGIXm_2K-E_YqwlFfcORNIRLKOJJTV1caaCRSucgDdGpnp5CobHL7jMKH7einTbQIUhAhTkSmA
VUOSNg3xObWWBARq7G6-syAYpmUcPEkFsqseHKKR9kccdVGNHCSnucTrvFlYvFUUlDfvJ5P2YwX-J3jY5wYcxOU1xMu0F7A7sxv1UhSRJK89Gea3pjK-pNBjhh9bGxUu
sWOUA0XuuoOEWJ1G6SBhHDVGAfIAi8qOP8jed5rUtwmYHuVyugpQp15rNl_0nWNqpl1i_0pZLxWRmDoyn5hCLrnJfL5fydH2bMW2CStGvoovJ3Yw3A0QR3qKibG-JJfB
BvVh1cUTV24Y1fr8NMaTfPhtKuripwqqXfGdBtU6c1h2X8GaMj3YhHjA48lTupCmwbJ8lMqRZDQkiyQubf2QbkWLk5k1eG40KEX8wjJrFwIZU18Slm-pj8OdUqLcxGHY
c5fXIqQkXvl7Bz22pEJ7NPzHTTvssS_xv3V7thMXCtjsUCHg&authuser=0&nonce=gp0sj6pjcq6t0&user=15475546427455647259&hash=824dscvm6fos4k1f7
8vlef3nm2kekuek" -L -o "rvl-cdip.rar"
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 4444M  100 4444M    0     0   175M      0  0:00:25  0:00:25 --:--:--  177M
```

In [ ]:
```
#unrar the file
get_ipython().system_raw("unrar x rvl-cdip.rar")
```

In [ ]:
```
#import all the required libraries
import tensorflow as tf
import os
import numpy as np
import pandas as pd

df=pd.read_csv('labels_final.csv',dtype=str)
```

In [ ]:
```
df.head(2)
```

Out[ ]:

	path	label
0	imagesv/v/o/h/voh71d00/509132755+-2755.tif	3
1	imagesl/l/x/t/lxt19d00/502213303.tif	3

In [ ]:
```
#new model
```

In [ ]:
```
import numpy as np
import pandas as pd
import keras
import pydotplus
from sklearn.tree import export_graphviz
```

In [ ]:
```
from keras.utils.vis_utils import plot_model
```

In [ ]:
```
images_df = pd.read_csv('labels_final.csv')
images_df['label'] = images_df['label'].astype(str)
images_dir = 'data_final'
```

In [ ]:
```
import tensorflow as tf
target_size = (128, 128)
data_generator=tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255,validation_split=0.2)
train_images_generator = data_generator.flow_from_dataframe(images_df, images_dir,x_col='path',y_col='label',class_mode='categor
ical', target_size=target_size,batch_size=128, subset='training')
valid_images_generator = data_generator.flow_from_dataframe(images_df, images_dir,x_col='path',y_col='label',class_mode='categor
ical', target_size=target_size,batch_size=128, subset='validation')
#images_df act is csv file which store files of location  image_dir.
# i was not knowning we can use csv file anf
```

```
Found 38400 validated image filenames belonging to 16 classes.
Found 9600 validated image filenames belonging to 16 classes.
```

In [ ]:
```
vgg16base_model = tf.keras.applications.VGG16(weights='imagenet', include_top=False,input_shape=(*target_size, 3))
for layer in vgg16base_model.layers:
    layer.trainable = False
conv_layer = tf.keras.layers.Conv2D(512, (3, 3), activation='relu',name='Conv')(vgg16base_model.output)
max_pool_layer = tf.keras.layers.MaxPooling2D(name='Max_Pool')(conv_layer)
flatten = tf.keras.layers.Flatten(name='Flatten')(max_pool_layer)
Fully_Connected_1 = tf.keras.layers.Dense(1024,activation='relu',name='Fully_Connected_Layer1')(flatten)
Fully_Connected_2 = tf.keras.layers.Dense(256,activation='relu',name='Fully_Connected_Layer2')(Fully_Connected_1)
Output = tf.keras.layers.Dense(16, activation='softmax',name='Output')(Fully_Connected_2)
model = tf.keras.models.Model(inputs=vgg16base_model.inputs, outputs=Output)
model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])
```

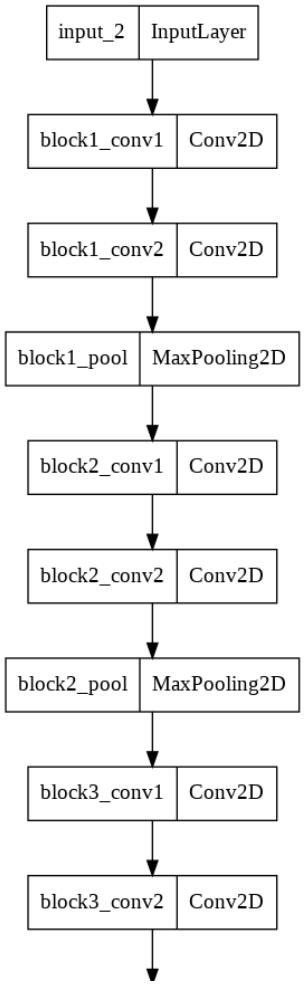
In [ ]:
```
model.summary()
```

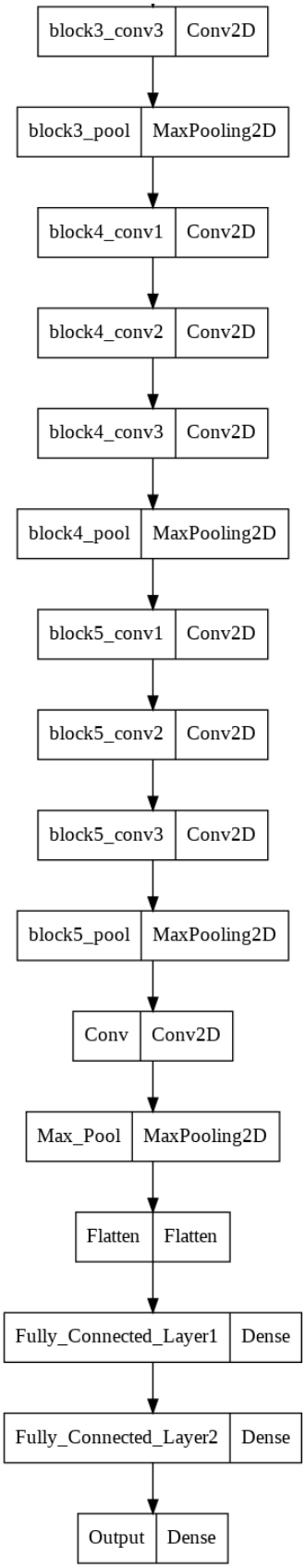
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0

 block1_conv1 (Conv2D)       (None, 128, 128, 64)      1792

 block1_conv2 (Conv2D)       (None, 128, 128, 64)      36928
```

```
block1_pool (MaxPooling2D)  (None, 64, 64, 64)        0

block2_conv1 (Conv2D)       (None, 64, 64, 128)       73856

block2_conv2 (Conv2D)       (None, 64, 64, 128)       147584

block2_pool (MaxPooling2D)  (None, 32, 32, 128)       0

block3_conv1 (Conv2D)       (None, 32, 32, 256)       295168

block3_conv2 (Conv2D)       (None, 32, 32, 256)       590080

block3_conv3 (Conv2D)       (None, 32, 32, 256)       590080

block3_pool (MaxPooling2D)  (None, 16, 16, 256)       0

block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

Conv (Conv2D)               (None, 2, 2, 512)         2359808

Max_Pool (MaxPooling2D)     (None, 1, 1, 512)         0

Flatten (Flatten)           (None, 512)               0

Fully_Connected_Layer1 (Den  (None, 1024)             525312
se)

Fully_Connected_Layer2 (Den  (None, 256)              262400
se)

Output (Dense)              (None, 16)                4112

=================================================================
Total params: 17,866,320
Trainable params: 3,151,632
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [ ]:
```
plot_model(model)# use to desctribe model
```

Out[ ]:





In [ ]:
```
import datetime
batch_size = 500
steps = len(train_images_generator.labels) // batch_size
early_stoping = keras.callbacks.EarlyStopping(patience=2)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard= tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True,write_grads=True)
callbacks_list = [early_stoping,tensorboard,]
history = model.fit(train_images_generator,validation_data=valid_images_generator,epochs=3,steps_per_epoch=steps,callbacks=callb
acks_list)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/3
76/76 [==============================] - 3286s 43s/step - loss: 1.9560 - accuracy: 0.3809 - val_loss: 1.5800 - val_accuracy: 0.
5140
Epoch 2/3
```

```
76/76 [==============================] - 3290s 43s/step - loss: 1.4735 - accuracy: 0.5427 - val_loss: 1.4426 - val_accuracy: 0.
5483
Epoch 3/3
76/76 [==============================] - 3307s 44s/step - loss: 1.3273 - accuracy: 0.5892 - val_loss: 1.2983 - val_accuracy: 0.
6032
```

In [ ]:
```
# Save the weights using the `checkpoint_path` format
checkpoint_path = "/content/"
# Save the weights
model.save_weights('./checkpoints/my_checkpoint')
```

In [ ]:
```
# Restore the weights
model.load_weights('./checkpoints/my_checkpoint')
```

Out[ ]:
```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f8e29bee0d0>
```

In [ ]:
```
%reload_ext tensorboard
%tensorboard --logdir {log_dir}
```

In [ ]:
```
###############  model-2
```

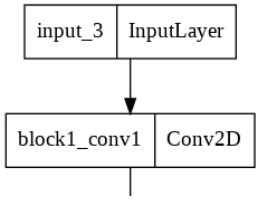
In [ ]:
```
vgg16base_model2 = tf.keras.applications.VGG16(weights='imagenet', include_top=False,input_shape=(*target_size, 3))
for layer in vgg16base_model2.layers:
    layer.trainable = False
conv_layer1 = tf.keras.layers.Conv2D(64,4,4, padding='same', activation='relu',name='Conv1')(vgg16base_model2.output)
#conv_layer1 = tf.keras.layers.Conv2D(512, kernel_size=(4, 4),padding='valid') ,4,4 BROUGHT O/P OF 1,1,64
conv_layer2 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu',padding='same',name='Conv2')(conv_layer1)
flatten = tf.keras.layers.Flatten(name='Flatten')(conv_layer2)
output = tf.keras.layers.Dense(16, activation='softmax',name='Output')(flatten)
model2 = tf.keras.models.Model(inputs=vgg16base_model2.inputs,outputs=output,name='Model_2')
model2.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])
```

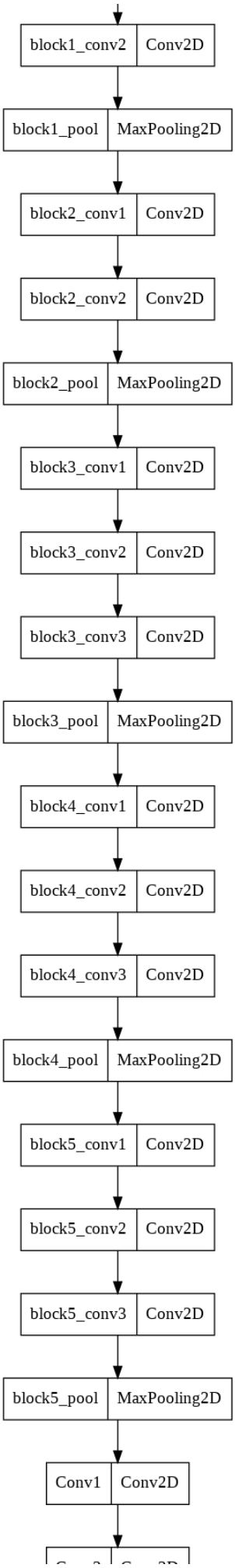
In [ ]:
```
model2.summary()
```

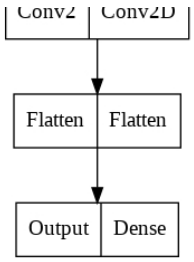
```
Model: "Model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 128, 128, 3)]     0

 block1_conv1 (Conv2D)       (None, 128, 128, 64)      1792

 block1_conv2 (Conv2D)       (None, 128, 128, 64)      36928

 block1_pool (MaxPooling2D)  (None, 64, 64, 64)        0

 block2_conv1 (Conv2D)       (None, 64, 64, 128)       73856

 block2_conv2 (Conv2D)       (None, 64, 64, 128)       147584

 block2_pool (MaxPooling2D)  (None, 32, 32, 128)       0

 block3_conv1 (Conv2D)       (None, 32, 32, 256)       295168

 block3_conv2 (Conv2D)       (None, 32, 32, 256)       590080

 block3_conv3 (Conv2D)       (None, 32, 32, 256)       590080

 block3_pool (MaxPooling2D)  (None, 16, 16, 256)       0

 block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

 block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 Conv1 (Conv2D)              (None, 1, 1, 64)          524352

 Conv2 (Conv2D)              (None, 1, 1, 128)         73856

 Flatten (Flatten)           (None, 128)               0

 Output (Dense)              (None, 16)                2064

=================================================================
Total params: 15,314,960
Trainable params: 600,272
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [ ]:
```
plot_model(model2)
```

Out[ ]:




block1_conv2 Conv2D
block1_pool MaxPooling2D
block2_conv1 Conv2D
block2_conv2 Conv2D
block2_pool MaxPooling2D
block3_conv1 Conv2D
block3_conv2 Conv2D
block3_conv3 Conv2D
block3_pool MaxPooling2D
block4_conv1 Conv2D
block4_conv2 Conv2D
block4_conv3 Conv2D
block4_pool MaxPooling2D
block5_conv1 Conv2D
block5_conv2 Conv2D
block5_conv3 Conv2D
block5_pool MaxPooling2D
Conv1 Conv2D



In [ ]:
```
history = model2.fit(train_images_generator,validation_data=valid_images_generator,epochs=3,steps_per_epoch=steps,callbacks=call
backs_list)
```

```
Epoch 1/3
76/76 [==============================] - 3298s 44s/step - loss: 2.1096 - accuracy: 0.3419 - val_loss: 1.7485 - val_accuracy: 0.
4548
Epoch 2/3
76/76 [==============================] - 3266s 43s/step - loss: 1.6078 - accuracy: 0.4972 - val_loss: 1.5021 - val_accuracy: 0.
5415
Epoch 3/3
76/76 [==============================] - 3274s 43s/step - loss: 1.4304 - accuracy: 0.5565 - val_loss: 1.4017 - val_accuracy: 0.
5696
```

In [ ]:
```
# Save the weights using the `checkpoint_path` format
checkpoint_path = "/content/"
# Save the weights
model2.save_weights('./checkpoints2/my_checkpoint')
```

In [ ]:
```
%tensorboard --logdir {log_dir}
```

```
Reusing TensorBoard on port 6006 (pid 811), started 2:45:22 ago. (Use '!kill 811' to kill it.)
```

In [ ]:
```
vgg16base_model3 = tf.keras.applications.VGG16(weights='imagenet', include_top=False,input_shape=(*target_size, 3))
for layer in vgg16base_model3.layers[0:-6]:
    layer.trainable = False
#conv_layer1 = tf.keras.layers.Conv2D(512, (3, 3), activation='relu',padding='same',name='Conv1')(vgg16base_model3.output)
conv_layer1 = tf.keras.layers.Conv2D(64,4,4, activation='relu',padding='same',name='Conv1')(vgg16base_model3.output)
conv_layer2 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu',padding='same',name='Conv2')(conv_layer1)
flatten = tf.keras.layers.Flatten(name='Flatten')(conv_layer2)
output = tf.keras.layers.Dense(16, activation='softmax',name='Output')(flatten)
model3 = tf.keras.models.Model(inputs=vgg16base_model3.inputs,outputs=output,name='Model_3')
model3.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kerne
ls_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
```

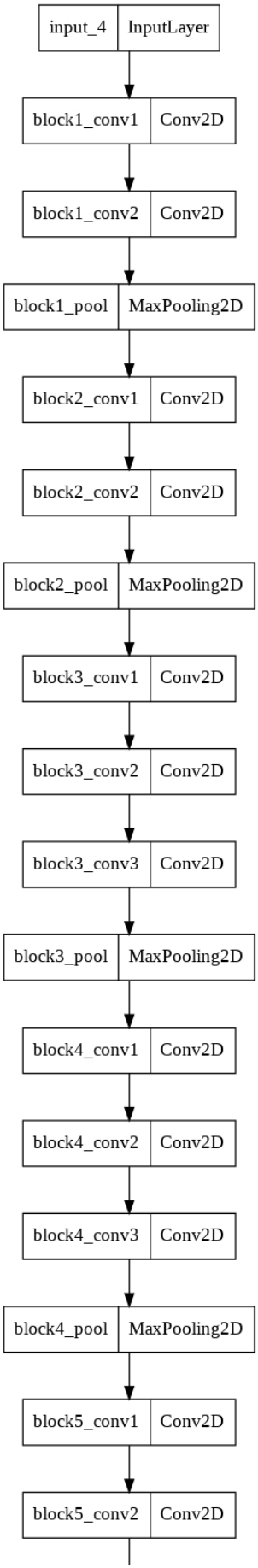
In [ ]:
```
model3.summary()
```

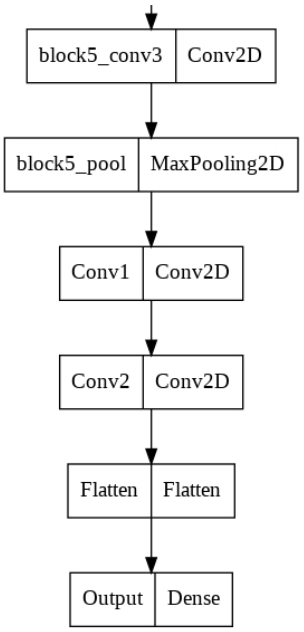
```
Model: "Model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 128, 128, 3)]     0

 block1_conv1 (Conv2D)       (None, 128, 128, 64)      1792

 block1_conv2 (Conv2D)       (None, 128, 128, 64)      36928

 block1_pool (MaxPooling2D)  (None, 64, 64, 64)        0

 block2_conv1 (Conv2D)       (None, 64, 64, 128)       73856

 block2_conv2 (Conv2D)       (None, 64, 64, 128)       147584

 block2_pool (MaxPooling2D)  (None, 32, 32, 128)       0

 block3_conv1 (Conv2D)       (None, 32, 32, 256)       295168

 block3_conv2 (Conv2D)       (None, 32, 32, 256)       590080

 block3_conv3 (Conv2D)       (None, 32, 32, 256)       590080

 block3_pool (MaxPooling2D)  (None, 16, 16, 256)       0

 block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

 block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 Conv1 (Conv2D)              (None, 1, 1, 64)          524352

 Conv2 (Conv2D)              (None, 1, 1, 128)         73856

 Flatten (Flatten)           (None, 128)               0

 Output (Dense)              (None, 16)                2064
```

```
=================================================================
Total params: 15,314,960
Trainable params: 10,039,504
Non-trainable params: 5,275,456
_________________________________________________________________
```

In [ ]:
```
plot_model(model3)
```

Out[ ]:





In [ ]:
```
import datetime
batch_size = 800
steps = len(train_images_generator.labels) // batch_size
early_stoping = keras.callbacks.EarlyStopping(patience=2)
tensorboard= tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True,write_grads=True)
callbacks_list = [early_stoping,tensorboard,]
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
history = model3.fit(train_images_generator,validation_data=valid_images_generator,epochs=3,steps_per_epoch=steps,callbacks=call
backs_list)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/3
48/48 [==============================] - 3134s 66s/step - loss: 2.6222 - accuracy: 0.1421 - val_loss: 2.3290 - val_accuracy: 0.
2473
Epoch 2/3
48/48 [==============================] - 3099s 65s/step - loss: 2.1020 - accuracy: 0.3328 - val_loss: 1.8509 - val_accuracy: 0.
4126
Epoch 3/3
48/48 [==============================] - 3150s 66s/step - loss: 1.7419 - accuracy: 0.4430 - val_loss: 1.6255 - val_accuracy: 0.
4923
```

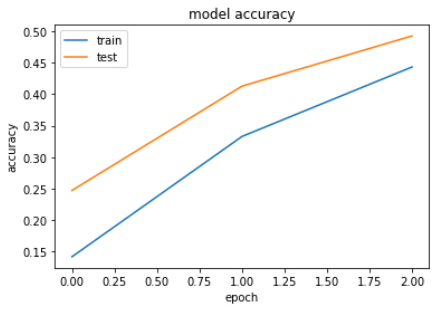
In [ ]:
```
#%tensorboard --logdir {log_dir}
%load_ext tensorboard
%tensorboard --logdir {log_dir}
```

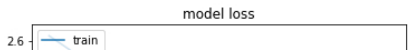
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

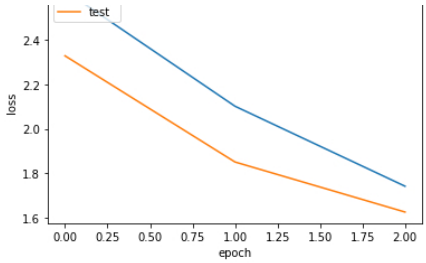
```
Reusing TensorBoard on port 6006 (pid 326), started 0:00:26 ago. (Use '!kill 326' to kill it.)
```

In [ ]:
```
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```






test
2.4
2.2
2.0
1.8
1.6
loss
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
epoch