(https://cognitiveclass.ai)

Peer Review Final Assignment

Introduction

In this lab, you will build an image classifier using the VGG16 pre-trained model, and you will evaluate it and compare its performance to the model we built in the last module using the ResNet50 pre-trained model. Good luck!

Table of Contents

Download Data

Use the `wget` command to download the data for this assignment from here: https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/concrete_data_week4.zip (https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/concrete_data_week4.zip)

Use the following cells to download the data.

In [4]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.applications import ResNet50
from keras.applications.resnet50 import preprocess_input
```

In [5]:

```
!wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/concrete_data_week4.zip
```

```
--2020-05-02 11:26:14--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/concrete_data_week4.zip
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softlayer.net)... 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 261483817 (249M) [application/zip]
Saving to: 'concrete_data_week4.zip'

100%[======================================>] 261,483,817 42.1MB/s   in 6.3s

2020-05-02 11:26:21 (39.7 MB/s) - 'concrete_data_week4.zip' saved [261483817/261483817]
```

In [1]:

```
#!unzip -n concrete_data_week4.zip
```

After you unzip the data, you fill find the data has already been divided into a train, validation, and test sets.

Part 1

In this part, you will design a classifier using the VGG16 pre-trained model. Just like the ResNet50 model, you can import the model `VGG16` from `keras.applications` .

You will essentially build your classifier as follows:

1. Import libraries, modules, and packages you will need. Make sure to import the *preprocess_input* function from `keras.applications.vgg16` .
2. Use a batch size of 100 images for both training and validation.
3. Construct an ImageDataGenerator for the training set and another one for the validation set. VGG16 was originally trained on 224 × 224 images, so make sure to address that when defining the ImageDataGenerator instances.
4. Create a sequential model using Keras. Add VGG16 model to it and dense layer.
5. Compile the mode using the adam optimizer and the categorical_crossentropy loss function.
6. Fit the model on the augmented data using the ImageDataGenerators.

Use the following cells to create your classifier.

In [7]:

```
#:Import libraries, modules, and packages you will need. Make sure to import the prepro
cess_input function from keras.applications.vgg16
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.applications import vgg16
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D
#from keras.optimizers import SGD
import numpy as np
```

In [8]:

```
#Use a batch size of 100 images for both training and validation.
num_classes = 2

image_resize = 224

batch_size_training = 100
batch_size_validation = 100
```

In [9]:

```
#Construct an ImageDataGenerator for the training set and another one for the validatio
n set. VGG16 was originally trained on 224 × 224 images, so make sure to address that w
hen defining the ImageDataGenerator instances.
data_generator = ImageDataGenerator(
    preprocessing_function=preprocess_input,
)

train_generator = data_generator.flow_from_directory(
    'concrete_data_week4/train',
    target_size=(image_resize, image_resize),
    batch_size=batch_size_training,
    class_mode='categorical')

validation_generator = data_generator.flow_from_directory('concrete_data_week4/valid',
        target_size=(image_resize, image_resize), batch_size=batch_size_validation,
        class_mode='categorical')
```

```
Found 30001 images belonging to 2 classes.
Found 9501 images belonging to 2 classes.
```

In [10]:

```
#Create a sequential model using Keras. Add VGG16 model to it and dense layer.

model = Sequential()

model.add(VGG16(
    include_top=False,
    pooling='avg',
    weights='imagenet',
    ))

model.add(Dense(num_classes, activation='softmax'))
model.layers[0].trainable = False
```

```
WARNING:tensorflow:From /opt/ibm/conda/miniconda3.6/lib/python3.6/site-pac
kages/tensorflow/python/framework/op_def_library.py:263: colocate_with (fr
om tensorflow.python.framework.ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Colocations handled automatically by placer.

Downloading data from https://github.com/fchollet/deep-learning-models/rel
eases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
```

In [11]:

```
#Compile the mode using the adam optimizer and the categorical_crossentropy loss functi
on
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])

steps_per_epoch_training = len(train_generator) / batch_size_training
steps_per_epoch_validation = len(validation_generator) / batch_size_validation
num_epochs = 2
```

In [12]:

```
#Fit the model on the augmented data using the ImageDataGenerators.

fit_history = model.fit_generator(
    train_generator,
    steps_per_epoch=steps_per_epoch_training,
    epochs=num_epochs,
    validation_data=validation_generator,
    validation_steps=steps_per_epoch_validation,
    verbose=1,
)
```

```
WARNING:tensorflow:From /opt/ibm/conda/miniconda3.6/lib/python3.6/site-pac
kages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.py
thon.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

Epoch 1/2
4/3 [==============================================] - 88s 22s/step - loss: 1.203
1 - acc: 0.4950 - val_loss: 0.9854 - val_acc: 0.4500
Epoch 2/2
4/3 [==============================================] - 79s 20s/step - loss: 0.958
5 - acc: 0.4425 - val_loss: 0.7276 - val_acc: 0.5300
```

In [8]:

```
model.save('classifier_vgg16_model.h5')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Part 2

In this part, you will evaluate your deep learning models on a test data. For this part, you will need to do the following:

1. Load your saved model that was built using the ResNet50 model.
2. Construct an ImageDataGenerator for the test set. For this ImageDataGenerator instance, you only need to pass the directory of the test images, target size, and the **shuffle** parameter and set it to False.
3. Use the **evaluate_generator** method to evaluate your models on the test data, by passing the above ImageDataGenerator as an argument. You can learn more about **evaluate_generator** here (https://keras.io/models/sequential/).
4. Print the performance of the classifier using the VGG16 pre-trained model.
5. Print the performance of the classifier using the ResNet pre-trained model.

Use the following cells to evaluate your models.

In [7]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.applications import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator
from keras.layers.core import Flatten, Dense, Dropout

num_classes = 2
image_resize = 224
batch_size_training = 100
batch_size_validation = 100

data_generator = ImageDataGenerator(
    preprocessing_function=preprocess_input,
)

train_generator = data_generator.flow_from_directory(
    'concrete_data_week4/train',
    target_size=(image_resize, image_resize),
    batch_size=batch_size_training,
    class_mode='categorical')

validation_generator = data_generator.flow_from_directory('concrete_data_week4/valid',
        target_size=(image_resize, image_resize), batch_size=batch_size_validation,
        class_mode='categorical')

# evaluate_generator = data_generator.flow_from_directory('concrete_data_week4/valid',
#         target_size=(image_resize, image_resize), batch_size=batch_size_validation,
#         class_mode='categorical')

## classifier_resnet_model.h5
model2 = Sequential()
model2.add(ResNet50(
    include_top=False,
    pooling='avg',
    weights='imagenet',
    ))
model2.add(Dense(num_classes, activation='softmax'))
model2.layers[0].trainable = False

model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])

steps_per_epoch_training = len(train_generator) / batch_size_training
steps_per_epoch_validation = len(validation_generator) / batch_size_validation
num_epochs = 2

fit_history = model2.fit_generator(
    train_generator,
    steps_per_epoch=steps_per_epoch_training,
    epochs=num_epochs,
    validation_data=validation_generator,
    validation_steps=steps_per_epoch_validation,
    verbose=1,
)

model2.save('classifier_resnet50a_model.h5')
```

```
Found 30001 images belonging to 2 classes.
Found 9501 images belonging to 2 classes.
Epoch 1/2
4/3 [==============================================] - 80s 20s/step - loss: 0.608
7 - acc: 0.6800 - val_loss: 0.5806 - val_acc: 0.7500
Epoch 2/2
4/3 [==============================================] - 55s 14s/step - loss: 0.336
8 - acc: 0.8975 - val_loss: 0.4475 - val_acc: 0.8500
```

In [8]:

```
import sklearn
```

In [11]:

```
#from keras.model import load_model
#my model = loadmodel('classifier_resnet50_model.h5')
X=
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size = 0.65, test_size =
0.35, random_state=0)
#print("X_train : ", X_train)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-11-65eb635adc69> in <module>()
      3
      4 from sklearn.model_selection import train_test_split
----> 5 X_train, X_test, y_train, y_test = train_test_split(X,y, train_siz
e = 0.65, test_size = 0.35, random_state=0)
      6 #print("X_train : ", X_train)

NameError: name 'X' is not defined
```

In [ ]:

```
#Construct an ImageDataGenerator for the test set. For this ImageDataGenerator instanc
e, you only need to pass the directory of the test images, target size, and the shuffle
parameter and set it to False
```

In [ ]:

```
#Use the evaluate_generator method to evaluate your models on the test data, by passing
the above ImageDataGenerator as an argument. You can learn more about evaluate_generato
r
```

In [ ]:

```
#Print the performance of the classifier using the VGG16 pre-trained model
```

In [ ]:

```
#Print the performance of the classifier using the ResNet pre-trained model.
```

Part 3

In this model, you will predict whether the images in the test data are images of cracked concrete or not. You will do the following:

1. Use the **predict_generator** method to predict the class of the images in the test data, by passing the test data ImageDataGenerator instance defined in the previous part as an argument. You can learn more about the **predict_generator** method here (https://keras.io/models/sequential/).
2. Report the class predictions of the first five images in the test set. You should print something list this:

Positive
Negative
Positive
Positive
Negative

Use the following cells to make your predictions.

In [ ]:

```
#Use the predict_generator method to predict the class of the images in the test data,
 by passing the test data ImageDataGenerator instance defined in the previous part as a
n argument. You can learn more about the predict_generator method
```

In [ ]:

```

```

In [ ]:

```
#Report the class predictions of the first five images in the test set.
```

In [ ]:

```

```

Thank you for completing this lab!

This notebook was created by Alex Aklson.

This notebook is part of a course on **Coursera** called *AI Capstone Project with Deep Learning*. If you accessed this notebook outside the course, you can take this course online by clicking here (https://cocl.us/DL0321EN_Coursera_Week4_LAB1).
