IBM Developer
SKILLS NETWORK

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

Use the following cells to download the data.

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

In [ ]:
```
!unzip -q 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 [1]:
```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.applications import VGG16
from keras.applications.vgg16 import preprocess_input
```

In [2]:
```
generator = ImageDataGenerator(preprocessing_function=preprocess_input)

training_generator = generator.flow_from_directory(
    "concrete_data_week4/train",
    target_size=(224,224),
    batch_size=100,
    class_mode="categorical",
)
validation_generator = generator.flow_from_directory(
    "concrete_data_week4/valid",
    target_size=(224,224),
    batch_size=100,
    class_mode="categorical",
)
```

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

In [3]:
```
model_vgg16 = Sequential()

model_vgg16.add(VGG16(include_top=False, pooling="avg", weights="imagenet",))
model_vgg16.add(Dense(2, activation="softmax"))

model_vgg16.layers[0].trainable = False

model_vgg16.compile(
    optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"]
)
```

In [5]:
```
num_epochs = 1
steps_per_epoch_training = len(training_generator)
steps_per_epoch_validation = len(validation_generator)

history_vgg16 = model_vgg16.fit_generator(
    training_generator,
    steps_per_epoch=steps_per_epoch_training,
    epochs=num_epochs,
    validation_data=validation_generator,
    validation_steps=steps_per_epoch_validation,
    verbose=1,
)
```

```
/home/hj14/Anaconda/envs/gnn-bw/lib/python3.7/site-packages/keras/engine/training.py:1915: UserWarning: `Model.fit_generator
` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
301/301 [==============================] - 837s 3s/step - loss: 0.0997 - accuracy: 0.9728 - val_loss: 0.0332 - val_accuracy:
0.9946
```

In [ ]:
```
model_vgg16.save("vgg16_model.h5")
```

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.
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 [ ]:
```
from keras.models import load_model

model_resnet50 = load_model("resnet_model.h5")
```

In [14]:
```
performance_vgg16 = model_vgg16.evaluate_generator(testing_generator)
print("Performance of the VGG16-trained model")
print("Loss: {}".format(round(performance_vgg16[0], 5)))
print("Accuracy: {}".format(round(performance_vgg16[1], 5)))
```

```
Performance of the VGG16-trained model
Loss: 0.00741
Accuracy: 0.996
```

In [15]:
```
performance_resnet50 = model_resnet50.evaluate_generator(testing_generator)
print("Performance of the ResNet50-trained model")
print("Loss: {}".format(round(performance_resnet50[0], 5)))
print("Accuracy: {}".format(round(performance_resnet50[1], 5)))
```

```
Performance of the ResNet50-trained model
Loss: 0.11661
Accuracy: 0.952
```

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.
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

In [17]:
```
predictions_vgg16 = model_vgg16.predict_generator(testing_generator, steps=1)


def print_prediction(prediction):
    if prediction[0] > prediction[1]:
        print("Negative ({}% certainty)".format(round(prediction[0] * 100, 1)))
    elif prediction[1] > prediction[0]:
        print("Positive ({}% certainty)".format(round(prediction[1] * 100, 1)))
    else:
        print("Unsure (prediction split 50–50)")


print("First five predictions for the VGG16-trained model:")
for i in range(5):
    print_prediction(predictions_vgg16[i])
```

```
First five predictions for the VGG16-trained model:
Negative (99.7% certainty)
Negative (92.1% certainty)
Negative (97.0% certainty)
Negative (98.7% certainty)
Negative (98.1% certainty)
```

Use the following cells to make your predictions.

In [18]:
```
predictions_resnet50 = model_resnet50.predict_generator(testing_generator, steps=1)
print("First five predictions for the ResNet50-trained model:")
for i in range(5):
    print_prediction(predictions_resnet50[i])
```

```
First five predictions for the ResNet50-trained model:
Negative (100.0% certainty)
Negative (100.0% certainty)
Negative (100.0% certainty)
Negative (99.9% certainty)
Negative (99.4% certainty)
```

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.
