사전 학습된 CNN(VGG-16)을 이용한 Fine Tunig

VGG-16 Model

- University of Oxford - Visual Geometry Group
- 2014 ILSVRC 2nd Model
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

```
import warnings
warnings.filterwarnings('ignore')
```

Import Keras

```
import keras

keras.__version__
```

```
'2.4.3'
```

I. Google Drive Mount

- 'dogs_and_cats_small.zip' 디렉토리를 구글드라이브에 업로드

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1) 구글 드라이브 마운트 결과 확인

```
!ls -l '/content/drive/My Drive/Colab Notebooks/datasets/dogs_and_cats_small.zip'
```

```
-rw------- 1 root root 90618980 Mar  4 04:51 '/content/drive/My Drive/Colab Notebooks/dataset
```

2) unzip 'dogs_and_cats_small.zip'

```
!unzip /content/drive/My₩ Drive/Colab₩ Notebooks/datasets/dogs_and_cats_small.zip
```

```
Archive:  /content/drive/My Drive/Colab Notebooks/datasets/dogs_and_cats_small.zip
  inflating: test/cats/cat.1501.jpg
  inflating: test/cats/cat.1502.jpg
  inflating: test/cats/cat.1503.jpg
  inflating: test/cats/cat.1504.jpg
  inflating: test/cats/cat.1505.jpg
  inflating: test/cats/cat.1506.jpg
  inflating: test/cats/cat.1507.jpg
  inflating: test/cats/cat.1508.jpg
  inflating: test/cats/cat.1509.jpg
  inflating: test/cats/cat.1510.jpg
  inflating: test/cats/cat.1511.jpg
  inflating: test/cats/cat.1512.jpg
  inflating: test/cats/cat.1513.jpg
  inflating: test/cats/cat.1514.jpg
  inflating: test/cats/cat.1515.jpg
  inflating: test/cats/cat.1516.jpg
  inflating: test/cats/cat.1517.jpg
  inflating: test/cats/cat.1518.jpg
  inflating: test/cats/cat.1519.jpg
  inflating: test/cats/cat.1520.jpg
  inflating: test/cats/cat.1521.jpg
  inflating: test/cats/cat.1522.jpg
  inflating: test/cats/cat.1523.jpg
  inflating: test/cats/cat.1524.jpg
  inflating: test/cats/cat.1525.jpg
  inflating: test/cats/cat.1526.jpg
  inflating: test/cats/cat.1527.jpg
  inflating: test/cats/cat.1528.jpg
  inflating: test/cats/cat.1529.jpg
  inflating: test/cats/cat.1530.jpg
  inflating: test/cats/cat.1531.jpg
  inflating: test/cats/cat.1532.jpg
  inflating: test/cats/cat.1533.jpg
  inflating: test/cats/cat.1534.jpg
  inflating: test/cats/cat.1535.jpg
  inflating: test/cats/cat.1536.jpg
  inflating: test/cats/cat.1537.jpg
  inflating: test/cats/cat.1538.jpg
  inflating: test/cats/cat.1539.jpg
  inflating: test/cats/cat.1540.jpg
  inflating: test/cats/cat.1541.jpg
  inflating: test/cats/cat.1542.jpg
  inflating: test/cats/cat.1543.jpg
  inflating: test/cats/cat.1544.jpg
  inflating: test/cats/cat.1545.jpg
  inflating: test/cats/cat.1546.jpg
  inflating: test/cats/cat.1547.jpg
  inflating: test/cats/cat.1548.jpg
  inflating: test/cats/cat.1549.jpg
  inflating: test/cats/cat.1550.jpg
  inflating: test/cats/cat.1551.jpg
  inflating: test/cats/cat.1552.jpg
  inflating: test/cats/cat.1553.jpg
  inflating: test/cats/cat.1554.jpg
  inflating: test/cats/cat.1555.jpg
  inflating: test/cats/cat.1556.jpg
  inflating: test/cats/cat.1557.jpg
  inflating: test/cats/cat.1558.jpg
  inflating: test/cats/cat.1559.jpg
```

```
!ls -l
```

```
!ls -l
```

```
total 20
drwx------ 5 root root 4096 Mar 24 06:21 drive
drwxr-xr-x 1 root root 4096 Mar 18 13:36 sample_data
drwxr-xr-x 4 root root 4096 Mar 24 06:21 test
drwxr-xr-x 4 root root 4096 Mar 24 06:21 train
drwxr-xr-x 4 root root 4096 Mar 24 06:21 validation
```

II. Image_File Directory Setting

- train_dir
- valid_dir
- test_dir

```
train_dir = 'train'
valid_dir = 'validation'
test_dir = 'test'
```

III. Data Preprocessing

1) ImageDataGenerator( ) & flow_from_directory( )

- Normalization
  - ImageDataGenerator( )
- Resizing & Generator
  - flow_from_directory( )

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255)
valid_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = train_datagen.flow_from_directory(
                  train_dir,
                  target_size = (150, 150),
                  batch_size = 20,
                  class_mode = 'binary')

valid_generator = valid_datagen.flow_from_directory(
                  valid_dir,
                  target_size = (150, 150),
                  batch_size = 20,
                  class_mode = 'binary')
```

```
                 class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

IV. Import VGG-16 Model & Some Layers Freezing

1) conv_base

```
from keras.applications import VGG16

conv_base = VGG16(weights = 'imagenet',
                  include_top = False,
                  input_shape = (150, 150, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg1
58892288/58889256 [==============================] - 0s 0us/step
```

2) Model Information

```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160   
```

```
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

3) 'block5_conv1' Freezing

- Before 'weight' Freezing

```
print('conv_base 동결 전 훈련 가능 가중치의 종류:', len(conv_base.trainable_weights))
```

```
conv_base 동결 전 훈련 가능 가중치의 종류: 26
```

- 'weight' Freezing

```
set_trainable = False

for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True

    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

- After 'weight' Freezing

```
print('conv_base 동결 후 훈련 가능 가중치의 종류:', len(conv_base.trainable_weights))
```

```
conv_base 동결 후 훈련 가능 가중치의 종류: 6
```

```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 7,079,424
Non-trainable params: 7,635,264
_________________________________________________________________
```

V. Keras CNN Modeling with VGG-16 Freezed Layers

1) Model Define

- 'conv_base' & 'Classification' Network
- Dropout Layer

```
from keras import models, layers

model = models.Sequential()
model.add(conv_base)

model.add(layers.Flatten())
model.add(layers.Dense(256, activation = 'relu'))
model.add(layers.Dropout(0.4))
model.add(layers.Dense(1, activation = 'sigmoid'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 9,177,089
Non-trainable params: 7,635,264
_________________________________________________________________
```

2) Model Compile

- 모델 학습방법 설정
  - 이미 학습된 Weight 값을 Tuning
  - 매우 작은 Learnig Rate 지정
  - optimizers.Adam(lr = 0.000005)

```
from keras import optimizers

model.compile(loss = 'binary_crossentropy',
              optimizer = optimizers.Adam(lr = 0.000005),
              metrics = ['accuracy'])
```

3) Model Fit

- 약 35분(K80)

```
%%time

Hist_dandc = model.fit(train_generator,
                       steps_per_epoch = 100,
                       epochs = 100,
                       validation_data = valid_generator,
                       validation_steps = 50)
```

```
Epoch 1/100
100/100 [==============================] - 42s 101ms/step - loss: 0.7055 - accuracy: 0.5559 -
Epoch 2/100
100/100 [==============================] - 10s 96ms/step - loss: 0.4409 - accuracy: 0.7999 -
Epoch 3/100
100/100 [==============================] - 9s 95ms/step - loss: 0.3136 - accuracy: 0.8827 - v
Epoch 4/100
100/100 [==============================] - 9s 94ms/step - loss: 0.2263 - accuracy: 0.9214 - v
Epoch 5/100
100/100 [==============================] - 10s 95ms/step - loss: 0.1770 - accuracy: 0.9437 -
Epoch 6/100
100/100 [==============================] - 10s 95ms/step - loss: 0.1518 - accuracy: 0.9485 -
Epoch 7/100
100/100 [==============================] - 9s 95ms/step - loss: 0.1095 - accuracy: 0.9632 - v
Epoch 8/100
100/100 [==============================] - 9s 94ms/step - loss: 0.1009 - accuracy: 0.9653 - v
Epoch 9/100
100/100 [==============================] - 10s 96ms/step - loss: 0.0851 - accuracy: 0.9781 -
Epoch 10/100
100/100 [==============================] - 10s 95ms/step - loss: 0.0747 - accuracy: 0.9802 -
Epoch 11/100
100/100 [==============================] - 10s 95ms/step - loss: 0.0617 - accuracy: 0.9834 -
Epoch 12/100
100/100 [==============================] - 10s 95ms/step - loss: 0.0436 - accuracy: 0.9945 -
Epoch 13/100
 28/100 [=======>......................] - ETA: 4s - loss: 0.0373 - accuracy: 0.9936
```

4) 학습 결과 시각화

- Loss Visualization

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_dandc.history['loss']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_dandc.history['loss'])
plt.plot(epochs, Hist_dandc.history['val_loss'])

plt.title('Training & Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.grid()
plt.show()
```

- Accuracy Visualization

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_dandc.history['loss']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_dandc.history['accuracy'])
plt.plot(epochs, Hist_dandc.history['val_accuracy'])

plt.title('Training & Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['Training Accuracy', 'Validation Accuracy'])
plt.grid()
plt.show()
```

5) Model Evaluate

- test_generator

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

test_generator = test_datagen.flow_from_directory(
                 test_dir,
                 target_size = (150, 150),
                 batch_size = 20,
                 class_mode = 'binary')
```

- Loss & Accuracy

```
loss, accuracy = model.evaluate(test_generator,
                                steps = 50)

print('Loss = {:.5f}'.format(loss))
print('Accuracy = {:.5f}'.format(accuracy))
```

IV. Model Save & Load to Google Drive

1) Google Drive Mount

```
from google.colab import drive

drive.mount('/content/drive')
```

2) Model Save

```
model.save('/content/drive/My Drive/Colab Notebooks/models/005_dogs_and_cats_fine_tuning.h5')
```

```
!ls -l /content/drive/My₩ Drive/Colab₩ Notebooks/models
```

3) Model Load

```
from keras.models import load_model

model_google = load_model('/content/drive/My Drive/Colab Notebooks/models/005_dogs_and_cats_fine_tu
```

```
loss, accuracy = model_google.evaluate(test_generator,
                                       steps = 50)

print('Loss = {:.5f}'.format(loss))
print('Accuracy = {:.5f}'.format(accuracy))
```

#

#

#

The End

#

#

#