사전 학습된 CNN(VGG-16)을 이용한 Feature Extraction

VGG-16 Model

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

Import Keras

```
import keras

keras.__version__
```

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
```

```
!ls -l
```

```
total 20
drwx------ 5 root root 4096 Mar 24 04:27 drive
drwxr-xr-x 1 root root 4096 Mar 18 13:36 sample_data
drwxr-xr-x 4 root root 4096 Mar 24 04:27 test
```

```
drwxr-xr-x 4 root root 4096 Mar 24 04:27 train
drwxr-xr-x 4 root root 4096 Mar 24 04:27 validation
```

II. Image_File Directory Setting

- train_dir
- valid_dir
- test_dir

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

III. Import VGG-16 Model

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 [==============================] - 1s 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
_________________________________________________________________
```

IV. Feature Extraction

1) 특징추출 함수 정의 : extract_feature( )

- conv_base.predict( )

```
from keras.preprocessing.image import ImageDataGenerator
import numpy as np

datagen = ImageDataGenerator(rescale = 1./255)

batch_size = 20

def extract_features(directory, sample_count):
    features = np.zeros(shape = (sample_count, 4, 4, 512))
    labels = np.zeros(shape = (sample_count))

    generator = datagen.flow_from_directory(directory,
                                            target_size = (150, 150),
```

```
                                              target_size = (150, 150),
                                              batch_size = batch_size,
                                              class_mode = 'binary')

    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base.predict(inputs_batch)
        features[i * batch_size : (i + 1) * batch_size] = features_batch
        labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        i += 1
        if i * batch_size >= sample_count:
            break
    return features, labels
```

2) 특징추출 함수 적용

- train_dir
- valid_dir
- test_dir

- 약 1분

```
%%time

train_features, train_labels = extract_features(train_dir, 2000)
valid_features, valid_labels = extract_features(valid_dir, 1000)
test_features, test_labels = extract_features(test_dir, 1000)
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
CPU times: user 23.5 s, sys: 6.54 s, total: 30 s
Wall time: 58.9 s
```

```
train_features.shape, valid_features.shape, test_features.shape
```

```
((2000, 4, 4, 512), (1000, 4, 4, 512), (1000, 4, 4, 512))
```

3) Reshape Features

- For 'Classification' Network

```
train_features = np.reshape(train_features, (2000, 4 * 4 * 512))
valid_features = np.reshape(valid_features, (1000, 4 * 4 * 512))
test_features = np.reshape(test_features, (1000, 4 * 4 * 512))

train_features.shape, valid_features.shape, test_features.shape
```

```
((2000, 8192), (1000, 8192), (1000, 8192))
```

V. Keras CNN Modeling with VGG-16 Featured Data

1) Model Define

- 'Classification' Network Only
- Dropout Layer

```
from keras import models, layers

model = models.Sequential()
model.add(layers.Dense(256, activation = 'relu', input_dim = 4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation = 'sigmoid'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 2,097,665
Trainable params: 2,097,665
Non-trainable params: 0
_________________________________________________________________
```

2) Model Compile

- 모델 학습방법 설정

```
model.compile(loss = 'binary_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])
```

3) Model Fit

- 약 1분

```
%%time

Hist_dandc = model.fit(train_features, train_labels,
                       epochs = 100,
                       batch_size = 20,
                       validation_data = (valid_features, valid_labels))
```

```
Epoch 1/100
100/100 [==============================] - 1s 7ms/step - loss: 0.6388 - accuracy: 0.7273 -
Epoch 2/100
100/100 [==============================] - 0s 4ms/step - loss: 0.2610 - accuracy: 0.8973 -
Epoch 3/100
100/100 [==============================] - 0s 4ms/step - loss: 0.2137 - accuracy: 0.9179 -
Epoch 4/100
100/100 [==============================] - 0s 4ms/step - loss: 0.1737 - accuracy: 0.9267 -
Epoch 5/100
100/100 [==============================] - 0s 4ms/step - loss: 0.1366 - accuracy: 0.9454 -
Epoch 6/100
100/100 [==============================] - 0s 4ms/step - loss: 0.1293 - accuracy: 0.9475 -
Epoch 7/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0886 - accuracy: 0.9740 -
Epoch 8/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0885 - accuracy: 0.9667 -
Epoch 9/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0611 - accuracy: 0.9806 -
Epoch 10/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0518 - accuracy: 0.9806 -
Epoch 11/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0662 - accuracy: 0.9736 -
Epoch 12/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0483 - accuracy: 0.9804 -
Epoch 13/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0372 - accuracy: 0.9879 -
Epoch 14/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0395 - accuracy: 0.9874 -
Epoch 15/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0302 - accuracy: 0.9924 -
Epoch 16/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0169 - accuracy: 0.9940 -
Epoch 17/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0251 - accuracy: 0.9894 -
Epoch 18/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0244 - accuracy: 0.9911 -
Epoch 19/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0166 - accuracy: 0.9964 -
Epoch 20/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0158 - accuracy: 0.9949 -
Epoch 21/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0146 - accuracy: 0.9968 -
Epoch 22/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0264 - accuracy: 0.9931 -
Epoch 23/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0430 - accuracy: 0.9810 -
Epoch 24/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0311 - accuracy: 0.9884 -
Epoch 25/100
```

```
100/100 [==============================] - 0s 4ms/step - loss: 0.0280 - accuracy: 0.9923 -
Epoch 26/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0455 - accuracy: 0.9841 -
Epoch 27/100
100/100 [==============================] - 0s 3ms/step - loss: 0.0274 - accuracy: 0.9891 -
Epoch 28/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0412 - accuracy: 0.9798 -
Epoch 29/100
100/100 [==============================] - 0s 4ms/step - loss: 0.0420 - accuracy: 0.9840 -
```

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

- Loss & Accuracy

```
loss, accuracy = model.evaluate(test_features, test_labels)

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

```
32/32 [==============================] - 0s 3ms/step - loss: 0.8565 - accuracy: 0.8820
Loss = 0.85651
Accuracy = 0.88200
```

IV. Model Save & Load to Google Drive

1) Google Drive Mount

```
from google.colab import drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
```

2) Model Save

```
model.save('/content/drive/My Drive/Colab Notebooks/models/004_dogs_and_cats_feature_extraction.h5'
```

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

```
total 440289
-rw------- 1 root root     34592 Mar 18 07:49 001_Model_iris.h5
-rw------- 1 root root  41498896 Mar 23 07:15 002_dogs_and_cats_small.h5
-rw------- 1 root root  41499744 Mar 23 07:29 003_dogs_and_cats_augmentation.h5
-rw------- 1 root root  25199032 Mar 24 04:39 004_dogs_and_cats_feature_extraction.h5
-rw------- 1 root root 140748400 Mar  9 06:31 005_dogs_and_cats_fine_tuning.h5
-rw------- 1 root root 201873880 Mar  9 07:17 006_dogs_and_cats_VGG16.h5
```

3) Model Load

```
from keras.models import load_model

model_google = load_model('/content/drive/My Drive/Colab Notebooks/models/004_dogs_and_cats_feature
```

```
loss, accuracy = model_google.evaluate(test_features, test_labels)

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

```
32/32 [==============================] - 0s 3ms/step - loss: 0.8565 - accuracy: 0.8820
Loss = 0.85651
Accuracy = 0.88200
```

#

#

#

The End

#

#

#