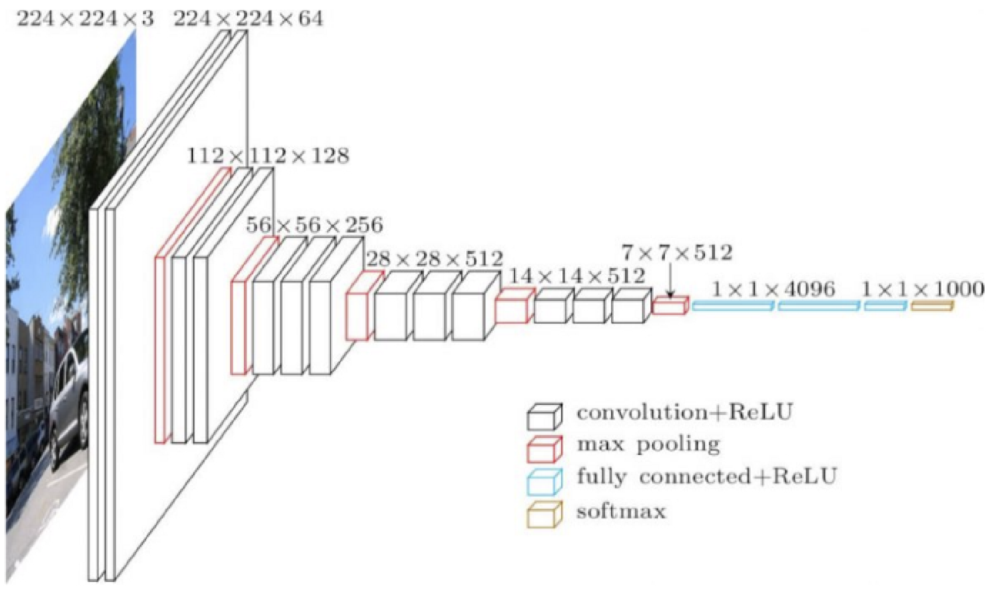
VGG-16

Very Deep Convolutional Networks for Large-Scale Image Recognition



VGG16 Architecture

Alexnet(2012)의 8-layer모델보다 깊이가 2배 이상 깊은 네트워크의 학습에 성공한 모델이다.

오차율을 Alexnet의 절반으로 줄였고, 모든 Convolution Layer에서 3 x 3 Filter를 사용하는것이 특징이다.

13 x Convolution Layer

3 x Fully Connected Layers

3 x 3 Filters

Stride, Padding : 1

2 x 2 Max Pooling (Stride : 2)

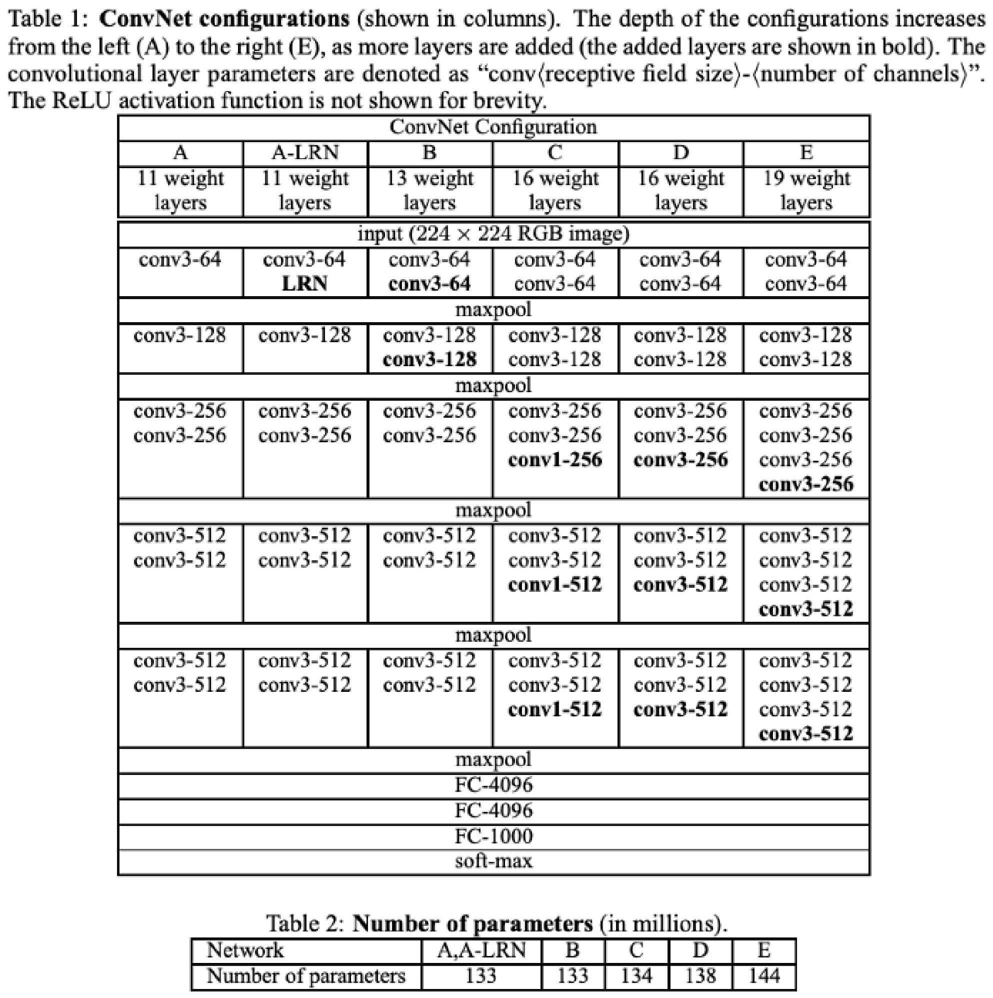


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

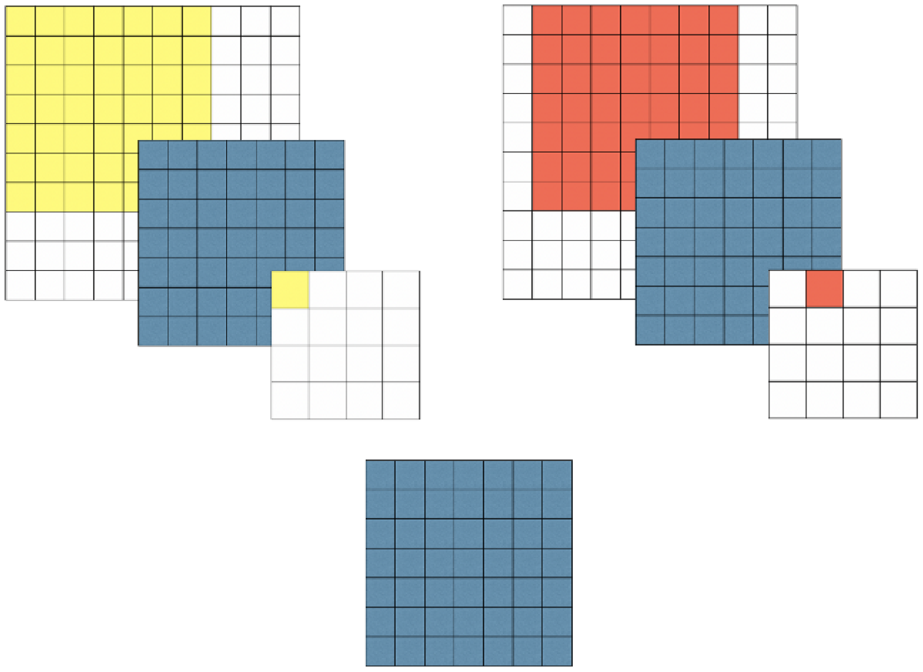
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

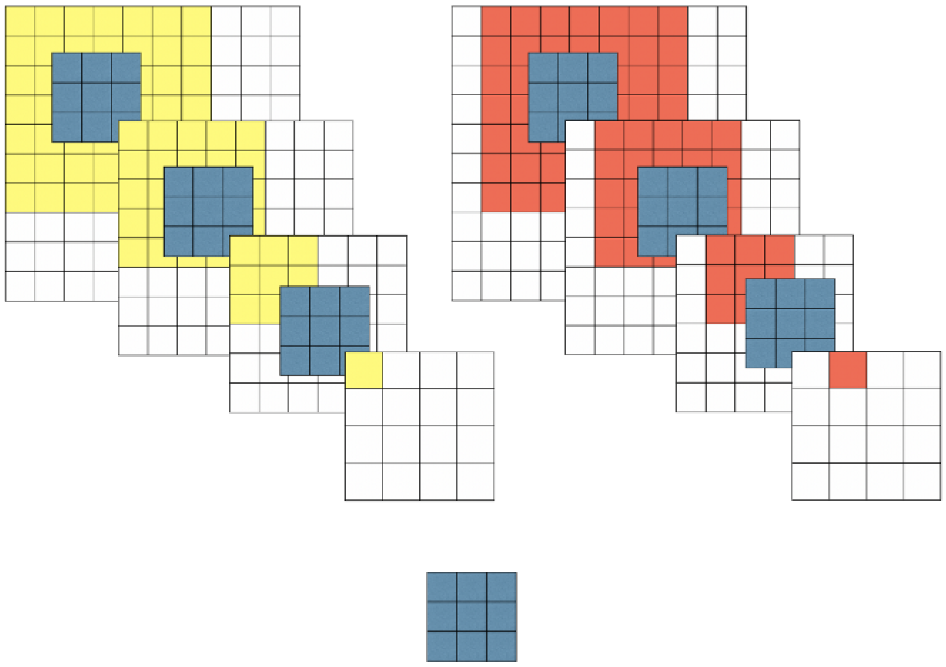
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

마지막 부분의 3개의 Fully Connected Layers는 4096,4096,1000개로 구성하고, 1000개의 Class로 구성되어 있기때문에 Softmax를 사용하였다.

모델의 특정 식별성 증가 메커니즘



7x7 Filter



3x3 Filter

10 x 10 이미지가 있다고 가정했을때, 7 x 7, 3 x 3 Filter를 비교한다.

7 x 7 Filter의 Receptive Field는 7 x 7.

3 x 3 Filter의 경우 stride가 1일때,

3×3 conv를 3번 반복한 Feature Map의 픽셀 한칸은 원본이미지를 7×7 conv 했을때의 Feature map의 픽셀을 수용할수 있다.

이때, Conv 연산이 ReLU 활성화를 거치게 되는데, 7 x 7 layer는 1번의 비선형함수가 적용되지만, 3×3 layer는 총 3번의 비선형함수가 적용되게된다.

이를 통해, 비선형성을 증가시키고 모델의 특정 식별 능력을 증가시킬수 있다. 또한 7×7 = 49, 3×3×3=27으로 Parameter의 개수 또한 감소하여 메모리 사용에 이득을 챙길 수 있다.

사용된 최적화 알고리즘

Multinomial Logisitic Regression

- 다항 로지스틱 회귀 (2개 이상의 범주)

Mini-batch gradient Descent

- 전체 Dataset에서 뽑은 Mini-batch안의 데이터 M개에 대해서 각각 데이터에 대한 기울기 M개를 구한뒤, M개의 평균 기울기를 통해 모델을 업데이트 하는 방식

Momentum(0.9)

- 일종의 가속도 개념.
- Global Minimum이 아닌, Local Minimum에 빠질 경우를 예방
- 경사하강을 운동량으로 표현하여, 운동량의 계수에 따라 바로 전 단계에 모멘텀 계수를 곱한다.
- 기존 운동량이 크다면, 그만큼 가속도를 받아 학습을 하기때문에 local minimum을 탈출하기 쉬워 학습에 용이하다

L2 Norm(Weight Decay)

- Euclidean Norm의 개념(직선거리)
- 기존의 Cost Function에 가중치의 제곱을 포함하여 더한다
- 가중치가 클 경우 overfitting에 영향을 미치기 때문에, 가중치 값이 클 경우 페널티를 크게 주고 가중치값이 작을 경우 페널티를 작게주어 모든 가중치가 모델에 고르게 반영되도록 함.

Dropout(0.5)

- 노드의 연결을 랜덤하게 끊어서 남은 노드들로만 학습을 진행
- 추후 Evaluation에서 모든 노드를 사용

Learning rate 0.01로 초기화후 서서히 줄이기

- Learning rate가 너무 클 경우, 최적점을 지나칠 수 있으며
- Learning rate가 너무 작을 경우, 학습 속도가 느려질 수 있다.
- 이러한 부분을 Control하여 overshooting을 방지한다

Code 구현 및 실습

```
import tensorflow as tf
from tensorflow.python.client import device_lib
device_lib.list_local_devices()

[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 18258968595231459162,
 name: "/device:XLA_CPU:0"
 device_type: "XLA_CPU"
 memory_limit: 17179869184
 locality {
 }
 incarnation: 9816436241720098664
 physical_device_desc: "device: XLA_CPU device",
 name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 4985044352
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 12127227614557882974
 physical_device_desc: "device: 0, name: GeForce GTX 1660 SUPER, pci bus id: 0000:06:00.0, compute capability: 7.5",
 name: "/device:XLA_GPU:0"
 device_type: "XLA_GPU"
 memory_limit: 17179869184
 locality {
 }
 incarnation: 3750131139645084680
 physical_device_desc: "device: XLA_GPU device"]
```

```
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras import optimizers, initializers, regularizers, metrics
from keras.callbacks import ModelCheckpoint
import os
from glob import glob
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
from numpy import expand_dims
from keras import models
from keras import layers
from tensorflow.python.keras.layers import Conv2D, MaxPooling2D, Flatten
from tensorflow.python.keras.layers import Dense, Dropout, Input
from tensorflow.python.keras.models import Model
from tensorflow.keras.layers import Input, Dense

model = models.Sequential()
#1층
model.add(Conv2D(input_shape=(224,224,3), filters=64, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#2층
model.add(Conv2D(input_shape=(224,224,64),filters=64, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#Max Pooling
model.add(MaxPooling2D(pool_size =(2,2), strides=(2,2)))
#3층
model.add(Conv2D(input_shape=(112,112,64),filters=128, kernel_size=(3,3), strides=(1,1), padding ='same', activation='relu'))
#4층
model.add(Conv2D(input_shape=(112,112,128),filters=128, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#Max Pooling
model.add(MaxPooling2D(pool_size =(2,2), strides=(2,2)))
#5층
model.add(Conv2D(input_shape=(56,56,128), filters=256, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#6층
model.add(Conv2D(input_shape=(56,56,256), filters=256, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#7층
model.add(Conv2D(input_shape=(56,56,256), filters=256, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
#8층
model.add(Conv2D(input_shape=(28,28,256), filters=512, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#9층
model.add(Conv2D(input_shape=(28,28,512), filters=512, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#10층
model.add(Conv2D(input_shape=(28,28,512), filters=512, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
```

```
#Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
#11층
model.add(Conv2D(input_shape=(14,14,512), filters=512, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#12층
model.add(Conv2D(input_shape=(14,14,512), filters=512, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#13층
model.add(Conv2D(input_shape=(14,14,512), filters=512, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
#Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
#14층 (fc1)
model.add(Flatten())
model.add(Dense(4096,activation='relu'))
#15층 (fc2)
model.add(Dense(4096,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2048,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2,activation='softmax'))
```

```
model.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 64)      1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 224, 224, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 112, 112, 128)     73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 128)     147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 56, 56, 256)       295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 256)       590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 28, 28, 512)       1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 512)       2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 512)       2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 512)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 4096)              102764544
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
```

```
dense_2 (Dense)              (None, 2048)              8390656
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 1026
=================================================================
Total params: 145,275,202
Trainable params: 145,275,202
Non-trainable params: 0
_________________________________________________________________
```

```
###vgg16으로 custom data training해보기###
# cats and dogs dataset 활용
train_dir = 'C:/Users/user/Desktop/catsanddogs/training_set/training_set'
test_dir = 'C:/Users/user/Desktop/catsanddogs/test_set/test_set'

# 폴더에 따라 자동 분류
train_image_generator = ImageDataGenerator(rescale=1./255)
test_image_generator = ImageDataGenerator(rescale=1./255)

# 데이터 구조 생성
train_data_gen = train_image_generator.flow_from_directory(batch_size=16,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(224, 224),
                                                           class_mode='binary')

test_data_gen = test_image_generator.flow_from_directory(batch_size=16,
                                                         directory=test_dir,
                                                         target_size=(224, 224),
                                                         class_mode='binary')
Found 8005 images belonging to 2 classes.
Found 2023 images belonging to 2 classes.
```

```
keras.optimizers.Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model.compile(optimizer='adam',
             loss = 'binary_crossentropy',
             metrics=['accuracy'])
history = model.fit_generator(
    train_data_gen, epochs=10,
    validation_data=test_data_gen,
    )

# 최종 결과 리포트
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

from matplotlib import pyplot as plt

plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='testing acc')
plt.title('Training and testing accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='testing loss')
plt.title('Training and testing loss')
plt.legend()
plt.show()
```

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
image = load_img('dog.1.jpg', target_size=(224, 224)) #Load image
image = img_to_array(image) # Convert to numpy array
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2])) # reshape data for the model
image = preprocess_input(image) # prepare the image for the VGG model
```

```
from keras.applications.vgg16 import decode_predictions
yhat = model.predict(image)
label = decode_predictions(yhat)    # convert the probabilities to class labels
label = label[0][0]                  # retrieve the most likely result, e.g. highest probability
print('%s (%.2f%%)' % (label[1], label[2]*100))    # print the classification
40960/35363 [==================================] - 0s 0us/step
Chesapeake_Bay_retriever (36.80%)
```

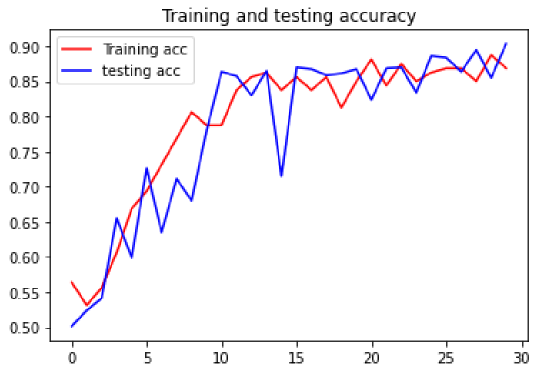
```
###전이학습으로 해보자###
train_dir = 'C:/Users/user/Desktop/catsanddogs/training_set/training_set'
test_dir = 'C:/Users/user/Desktop/catsanddogs/test_set/test_set'

# 폴더에 따라 자동 분류
train_image_generator = ImageDataGenerator(rescale=1./255)
test_image_generator = ImageDataGenerator(rescale=1./255)

# 데이터 구조 생성
train_data_gen = train_image_generator.flow_from_directory(batch_size=16,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(224, 224),
                                                           class_mode='binary')

test_data_gen = test_image_generator.flow_from_directory(batch_size=16,
                                                         directory=test_dir,
                                                         target_size=(224, 224),
                                                         class_mode='binary')
```

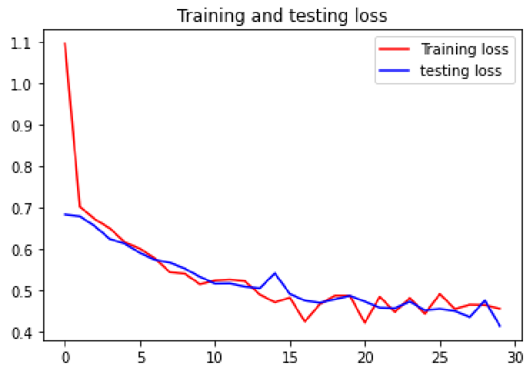
```
#설계된 모델의 w,b를 업데이트하지않게하기

for layer in model.layers:
    layer.trainable = False

#block5_pool까지가 물체의 특징을 뽑아내는 층이라서 여기까지만 사용
vgg_maxpool5 = model.get_layer('block5_pool').output

FeatureFlatten = Flatten()(vgg_maxpool5)

dense = Dense(10,name='dense', activation='relu')(FeatureFlatten)
predictions = Dense(1,activation = 'sigmoid')(dense)

New_VGGmodel = Model(inputs=model.input, outputs=predictions)
New_VGGmodel.compile(optimizer = 'rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
New_VGGmodel.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
```

```
block1_conv2 (Conv2D)         (None, 224, 224, 64)     36928

block1_pool (MaxPooling2D)    (None, 112, 112, 64)     0

block2_conv1 (Conv2D)         (None, 112, 112, 128)    73856

block2_conv2 (Conv2D)         (None, 112, 112, 128)    147584

block2_pool (MaxPooling2D)    (None, 56, 56, 128)      0

block3_conv1 (Conv2D)         (None, 56, 56, 256)      295168

block3_conv2 (Conv2D)         (None, 56, 56, 256)      590080

block3_conv3 (Conv2D)         (None, 56, 56, 256)      590080

block3_pool (MaxPooling2D)    (None, 28, 28, 256)      0

block4_conv1 (Conv2D)         (None, 28, 28, 512)      1180160

block4_conv2 (Conv2D)         (None, 28, 28, 512)      2359808

block4_conv3 (Conv2D)         (None, 28, 28, 512)      2359808

block4_pool (MaxPooling2D)    (None, 14, 14, 512)      0

block5_conv1 (Conv2D)         (None, 14, 14, 512)      2359808

block5_conv2 (Conv2D)         (None, 14, 14, 512)      2359808

block5_conv3 (Conv2D)         (None, 14, 14, 512)      2359808

block5_pool (MaxPooling2D)    (None, 7, 7, 512)        0

flatten_3 (Flatten)           (None, 25088)            0

dense (Dense)                 (None, 10)               250890

dense_8 (Dense)               (None, 1)                11

=================================================================
Total params: 14,965,589
Trainable params: 250,901
Non-trainable params: 14,714,688
_________________________________________________________________
```

```python
history = New_VGGmodel.fit_generator(train_data_gen, steps_per_epoch=10,
                                    epochs=30, validation_data = test_data_gen, validation_steps=50)
# 최종 결과 리포트
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

from matplotlib import pyplot as plt

plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='testing acc')
plt.title('Training and testing accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='testing loss')
plt.title('Training and testing loss')
plt.legend()

plt.show()
```





```
10/10 [==============================] - 9s 854ms/step - loss: 0.4551 - accuracy: 0.8687 - val_loss: 0.4138 - val_accuracy: 0.9038
```