산업인공지능 개론

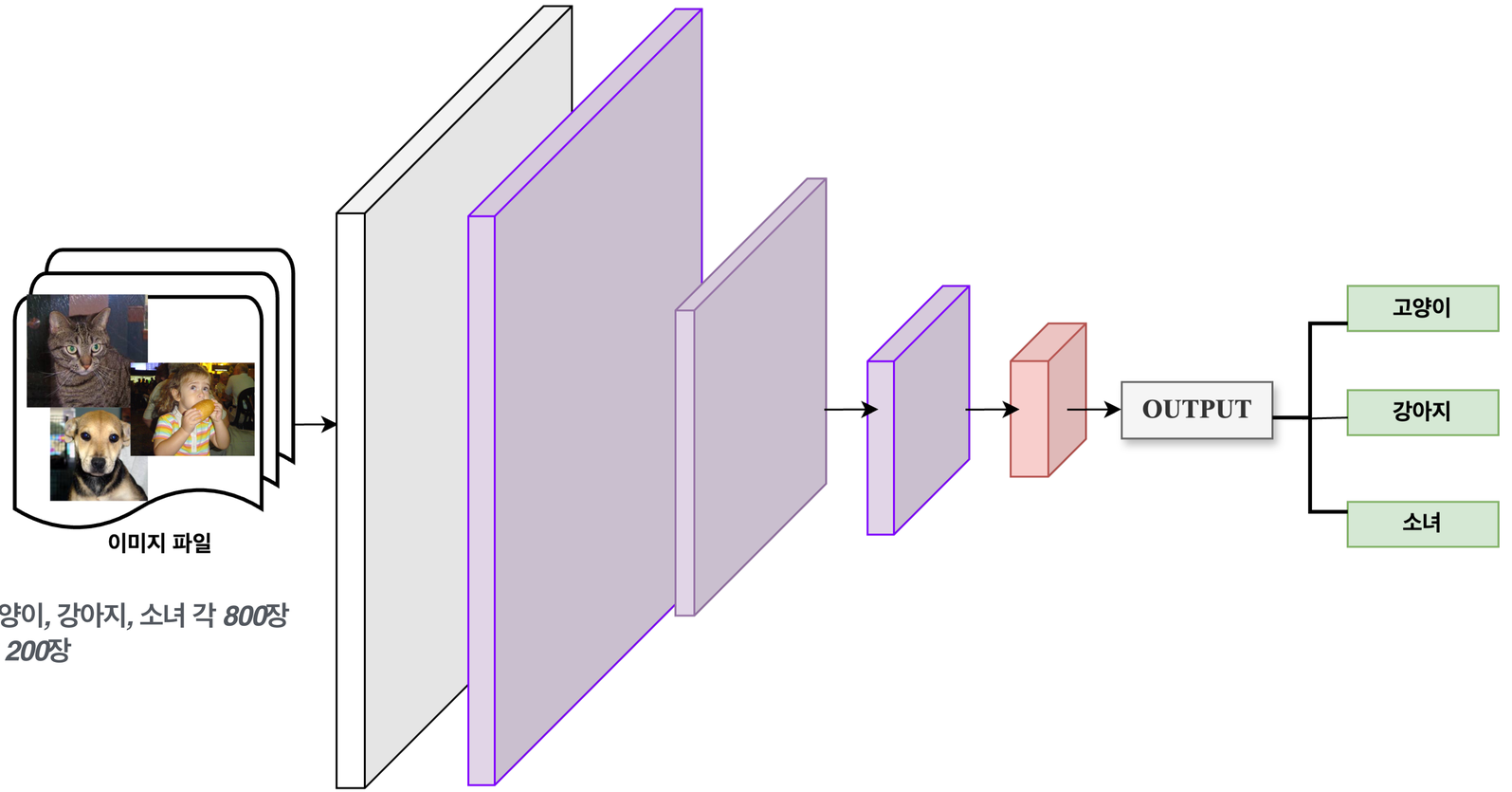
Miniproject #3
이미지 분류

학과 : 산업인공지능학과
학번 : 2024254022
이름 : 정현일

2024.06.02.

이미지 데이터

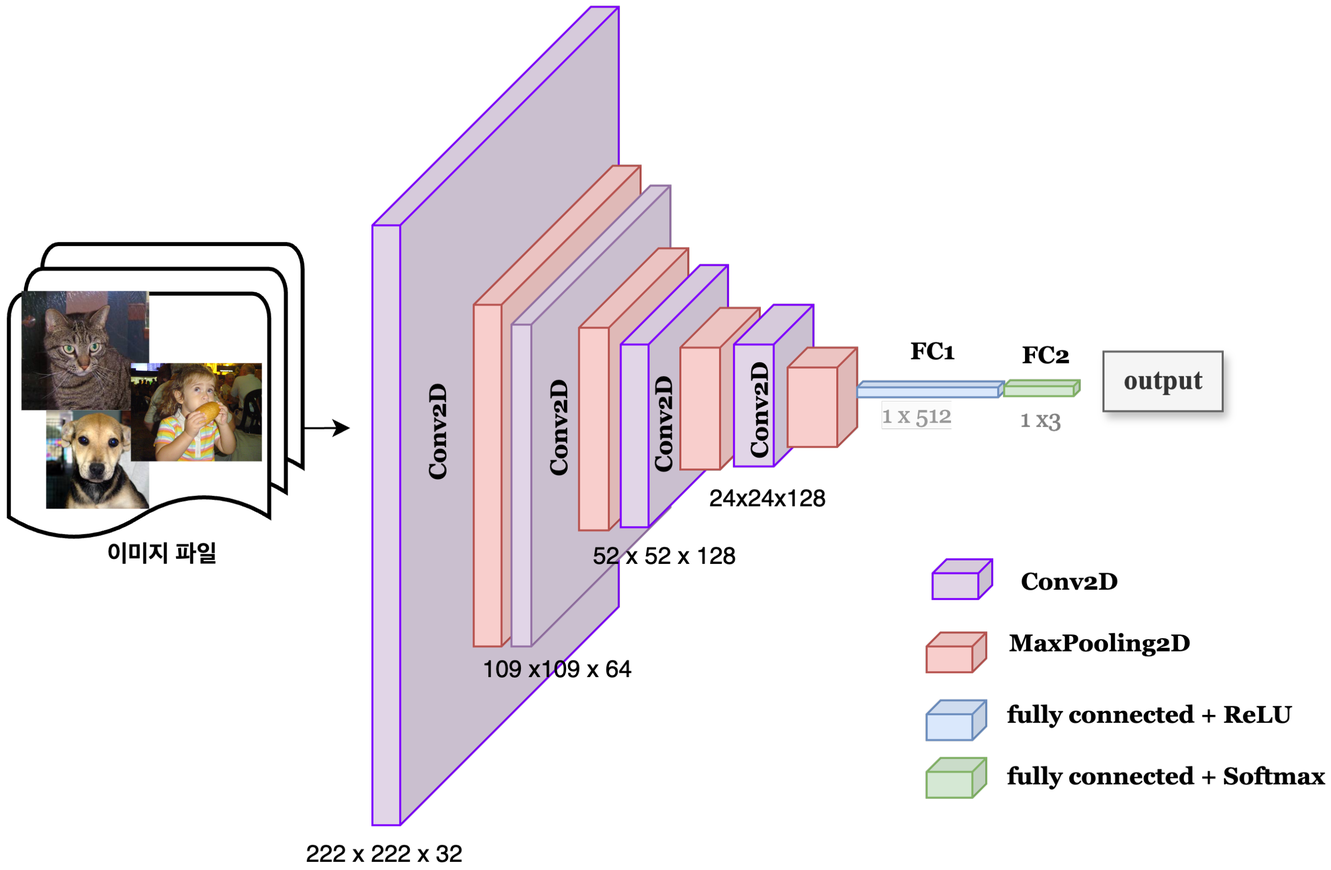
이미지 데이터는 *Kaggle*에서 제공되는 강아지, 고양이, 소녀 이미지를 활용하여 *CNN* 모델을 학습시키고, 학습된 신경망 모델을 기반으로 이미지를 분류하는 모델을 구현하였습니다.



학습데이터 고양이, 강아지, 소녀 각 *800*장
검증데이터 각 *200*장

초기 데이터 모델 설계

모델 설계



모델 요약

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_8 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_9 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_10 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_10 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_11 (Conv2D)	(None, 24, 24, 128)	147,584
max_pooling2d_11 (MaxPooling2D)	(None, 12, 12, 128)	0
flatten_2 (Flatten)	(None, 18432)	0
dense_4 (Dense)	(None, 512)	9,437,696
dense_5 (Dense)	(None, 3)	1,539

이미지 분류 모델 설명(1/2)

이미지 데이터 준비

```
imageclass/
        train/
            cats/
            dogs/
            girls/
        validation/
            cats/
            dogs/
            girls/
```

이미지 전처리 및 데이터 셋 생성

```python
# 이미지 전처리 및 증강
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1./255)

# 데이터셋 생성기
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=20,
    class_mode='categorical'
)
```

모델 정의

```python
#  CNN 모델 정의
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')

])

# Print out model summary
print(model.summary())
```

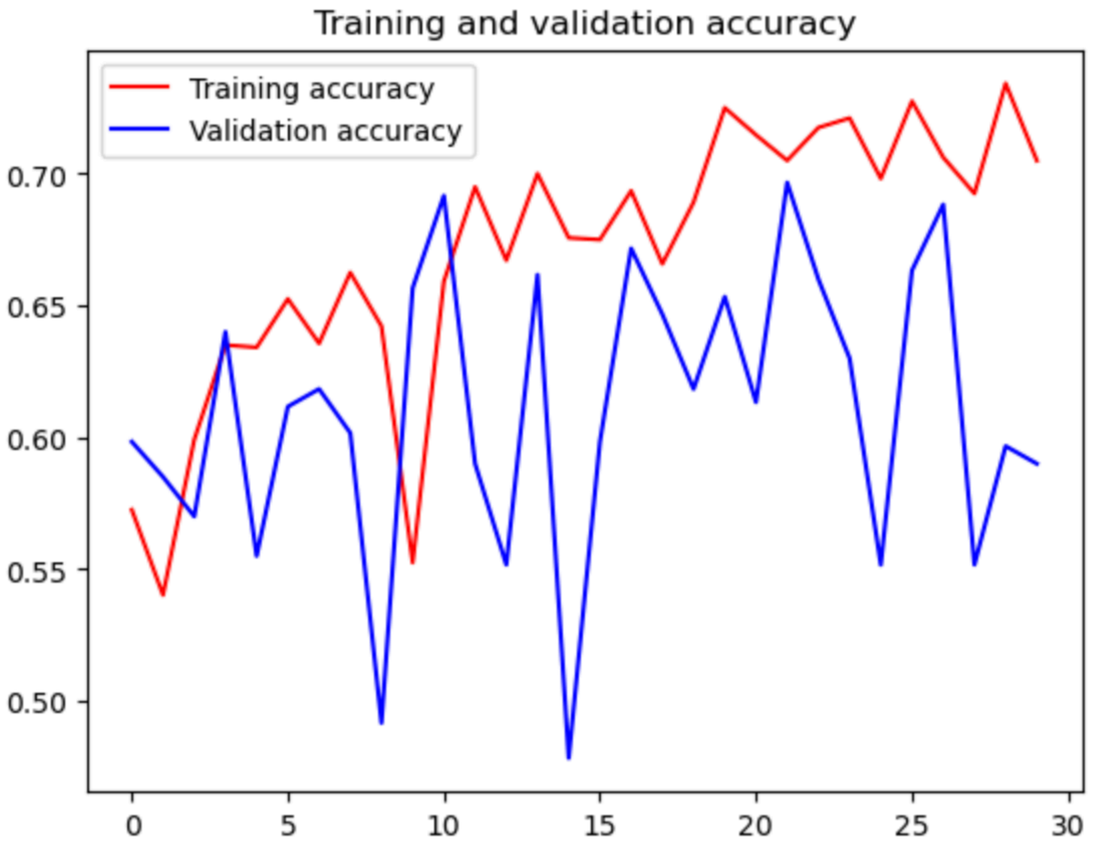
모델 학습

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

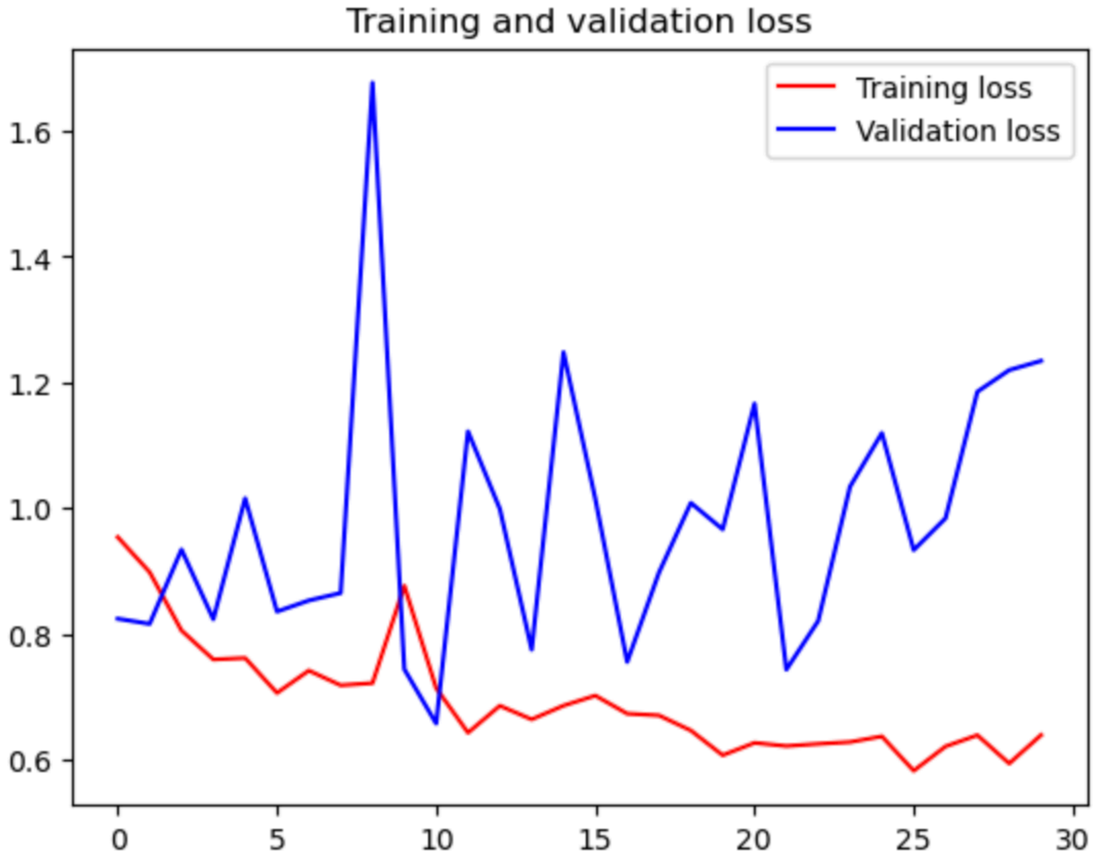
```python
# 모델 학습
history = model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=50
)
```

학습결과 시각화

- 정확도

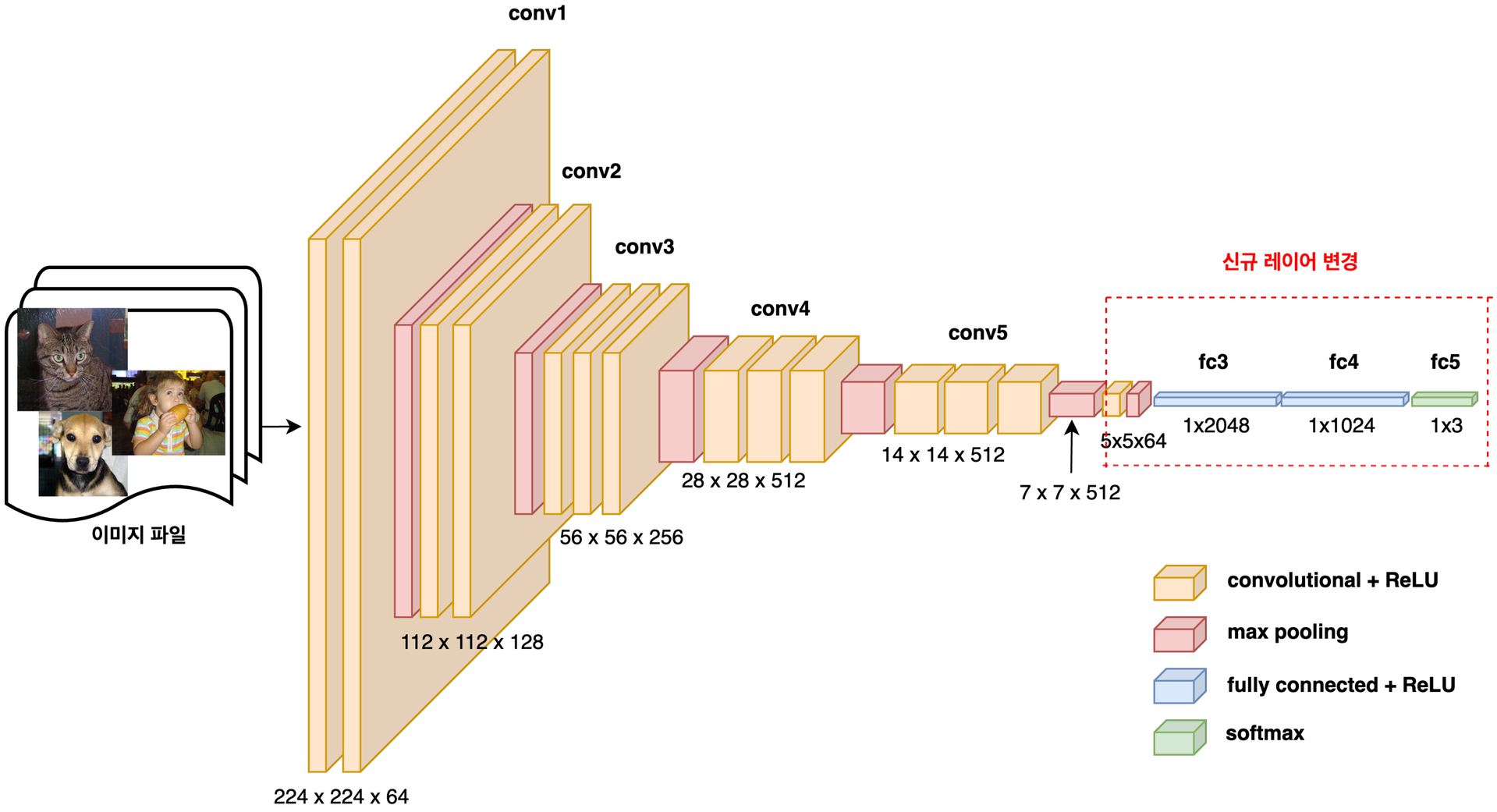


- Loss



VGG16 모델 미세 학습 *(Find-Tuning)* - 모델 설계

기존 *VGG16* 모델의 *Dense* 레이어를 제외 시키고, 신규 레이어로 변경해서 미세학습을 진행하였습니다.



VGG16 모델 미세 학습 (Find-Tuning) - 모델 정의

신규 모델 정의

```
# 모델 Layer 데이터화
layer_dict = dict([(layer.name, layer) for layer in model.layers])

# Layer 추가
x = layer_dict['block5_pool'].output
# Cov2D Layer +
x = Conv2D(filters = 64, kernel_size=(3, 3), activation='relu')(x)
# MaxPooling2D Layer +
x = MaxPooling2D(pool_size=(2, 2))(x)
# Flatten Layer +
x = Flatten()(x)
# FC Layer +
x = Dense(2048, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(3, activation='softmax')(x)

# new model 정의
new_model = Model(inputs = model.input, outputs = x)
```

모델 컴파일

```
# CNN Pre-trained 가중치를 그대로 사용할때
for layer in new_model.layers[:19] :
    layer.trainable = False

new_model.summary()

# 컴파일 옵션
new_model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
```

모델 요약

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
conv2d_1 (Conv2D)	(None, 5, 5, 64)	294,976
max_pooling2d_1 (MaxPooling2D)	(None, 2, 2, 64)	0
flatten_1 (Flatten)	(None, 256)	0
dense_3 (Dense)	(None, 2048)	526,336
dropout_2 (Dropout)	(None, 2048)	0
dense_4 (Dense)	(None, 1024)	2,098,176
dropout_3 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 3)	3,075

VGG16 모델 미세 학습 (Find-Tuning) - 모델 학습

이미지 데이터 준비

```
# 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=validation_dir,
                                                         target_size=(224, 224),
                                                         class_mode='binary')
```

Found 2398 images belonging to 3 classes.
Found 600 images belonging to 3 classes.

모델 학습

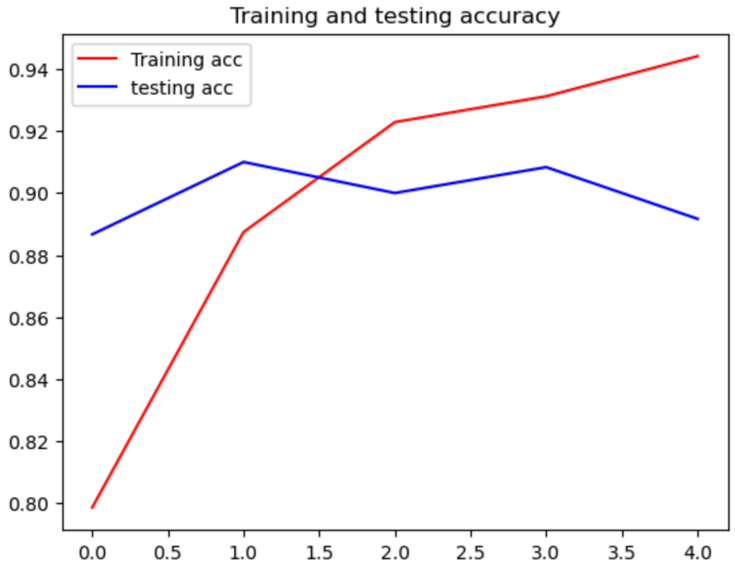
```
# 모델 학습
history = new_model.fit(train_data_gen, epochs=5,
                        validation_data=test_data_gen)
```

✓ 63m 26.8s

```
Epoch 1/5
150/150 ━━━━━━━━━━━━━━━━━━━━ 1040s 7s/step - accuracy: 0.6998 - loss: 0.6944 - val_accuracy: 0.8867 - val_loss: 0.2908
Epoch 2/5
150/150 ━━━━━━━━━━━━━━━━━━━━ 499s 3s/step - accuracy: 0.8947 - loss: 0.2974 - val_accuracy: 0.9100 - val_loss: 0.3057
Epoch 3/5
150/150 ━━━━━━━━━━━━━━━━━━━━ 1140s 8s/step - accuracy: 0.9297 - loss: 0.1894 - val_accuracy: 0.9000 - val_loss: 0.3723
Epoch 4/5
150/150 ━━━━━━━━━━━━━━━━━━━━ 618s 4s/step - accuracy: 0.9411 - loss: 0.1567 - val_accuracy: 0.9083 - val_loss: 0.3196
Epoch 5/5
150/150 ━━━━━━━━━━━━━━━━━━━━ 509s 3s/step - accuracy: 0.9471 - loss: 0.1331 - val_accuracy: 0.8917 - val_loss: 0.4767
```

VGG16 모델 미세 학습 *(Find-Tuning)* - 학습결과

- 정확도



- Loss