Computer Graphics
실습 11.

2020. 11. 26

박 화 종

aqkrghkwhd@naver.com

실습 소개

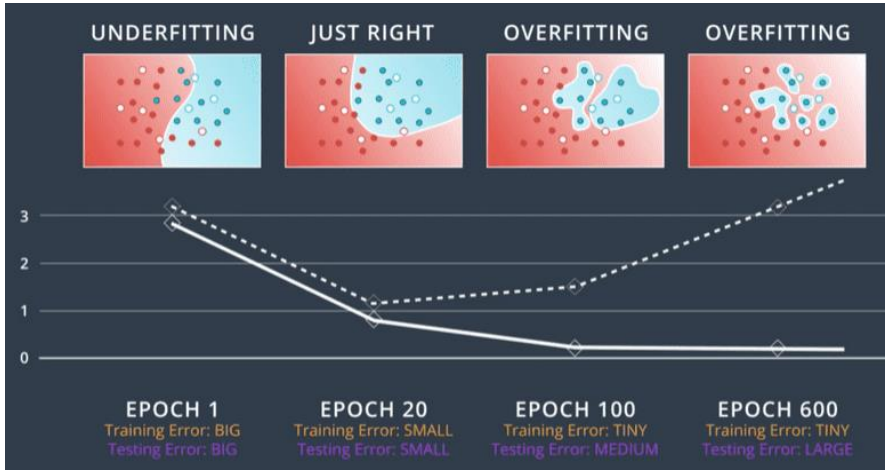
- **과목 홈페이지**
  - 충남대학교 사이버 캠퍼스 (http://e-learn.cnu.ac.kr)

- **TA 연락처**
  - 박화종
  - 공대 5호관 506호 컴퓨터비전 연구실
  - aqkrghkwhd@naver.com

- **실습 튜터**
  - 최수민(00반)
  - eocjstnals12@naver.com
  - 신준호(01반)
  - wnsgh578@naver.com

목 차

Overfitting



- **Overfitting이란?**
  - 학습데이터를 과하게 잘 학습하는 것

Train

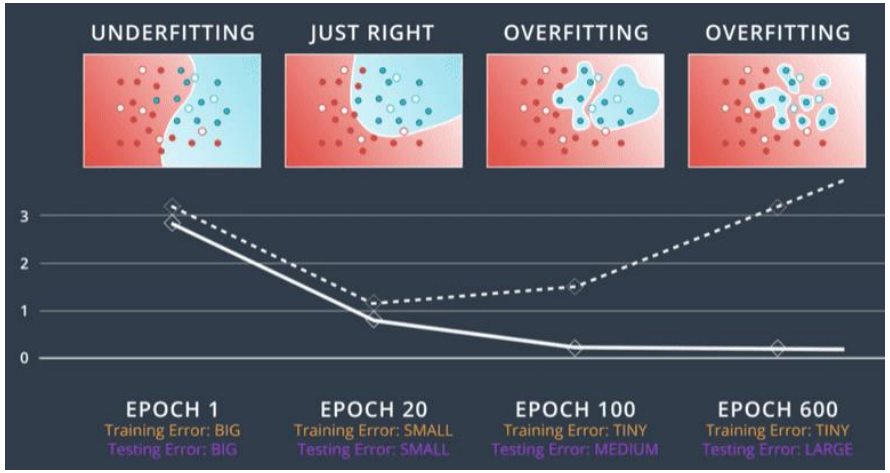
epoch가 적은 경우

epoch가 적당한 경우

epoch가 많은 경우

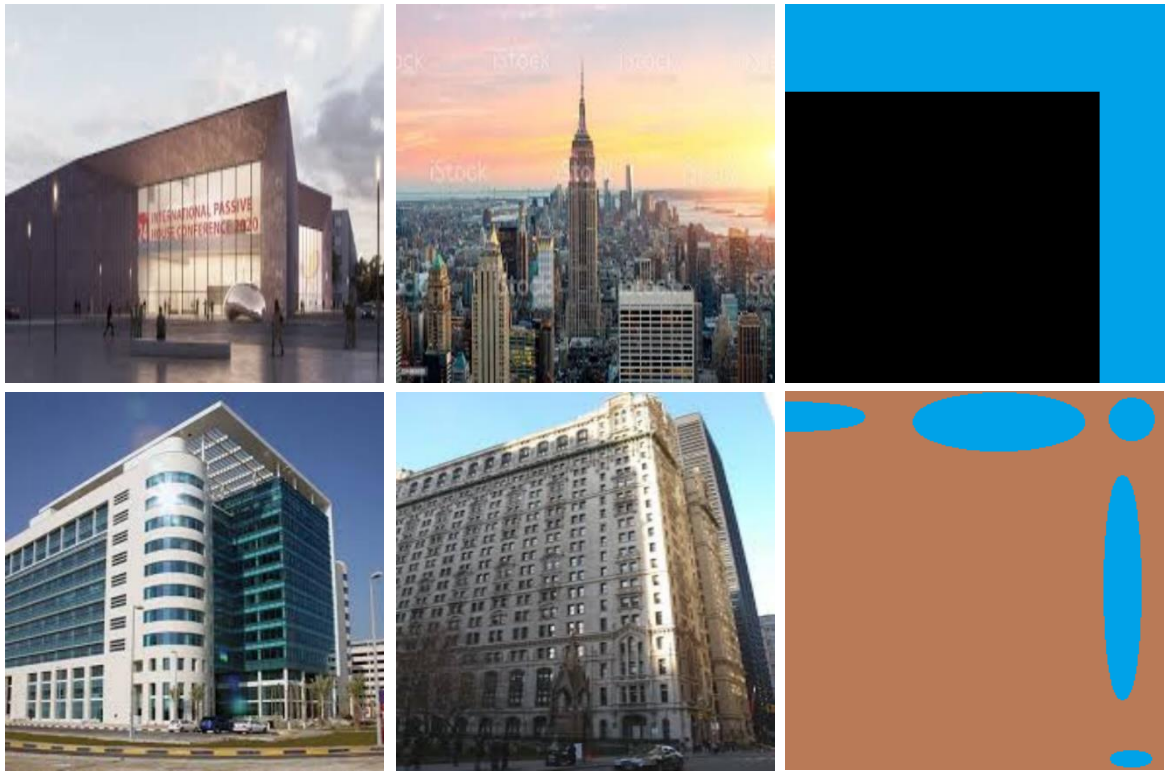
Overfitting

- **Overfitting이란?**
  - 학습데이터를 과하게 잘 학습하는 것



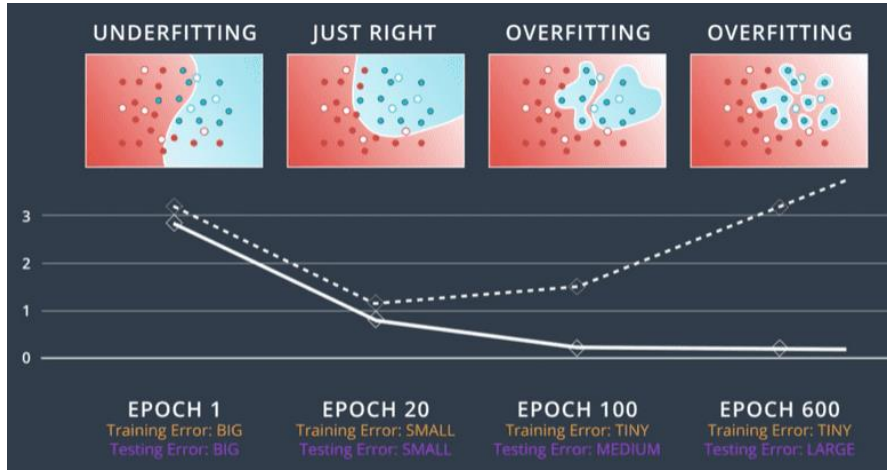
Train

epoch가 적은 경우



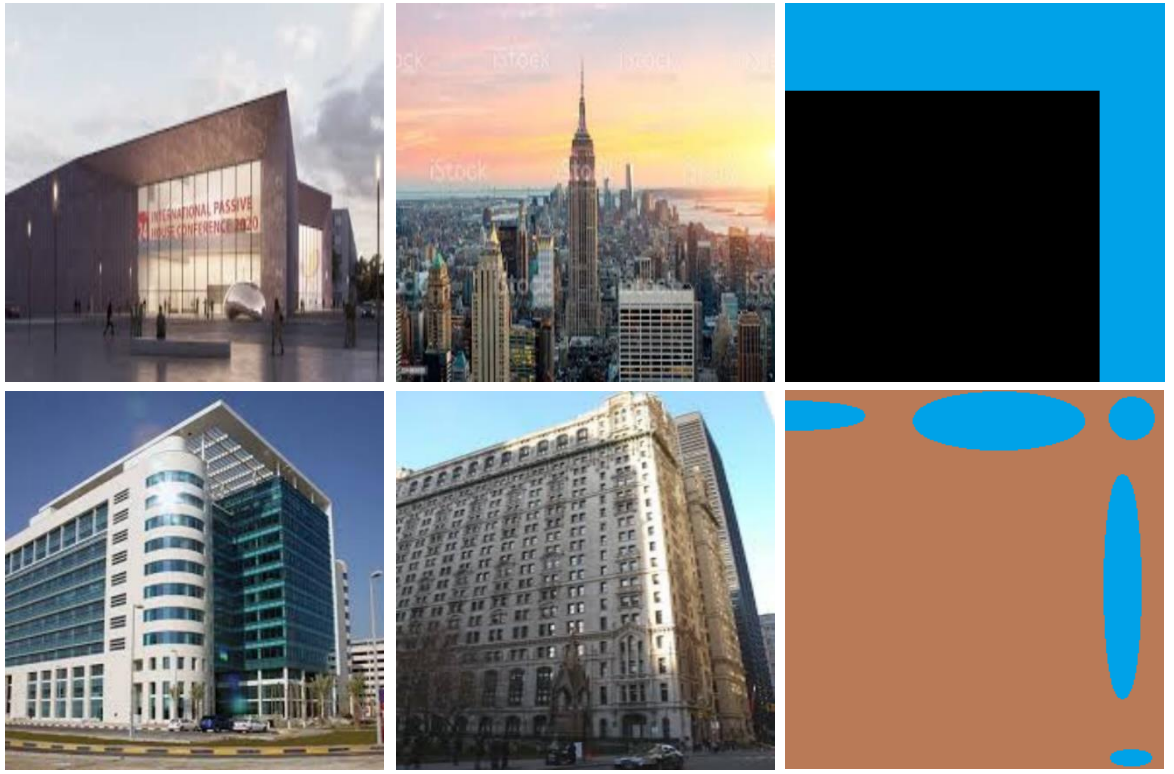
Overfitting

- **Overfitting이란?**
  - 학습데이터를 과하게 잘 학습하는 것



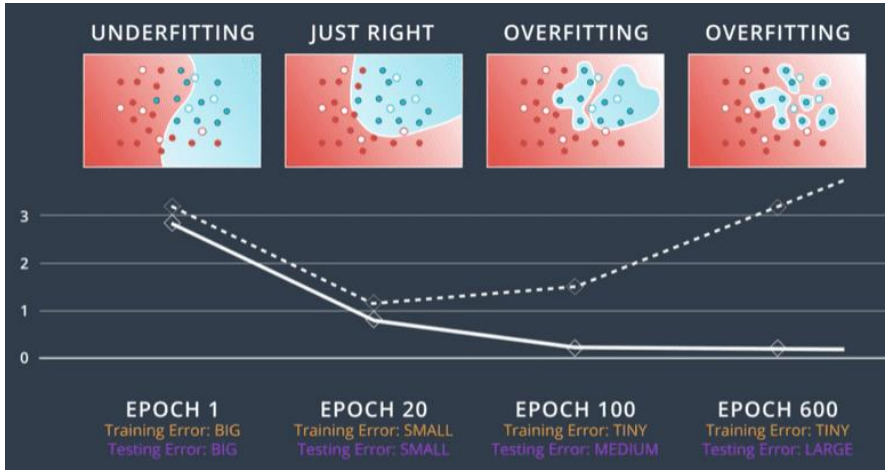
Train

epoch가 적당한 경우



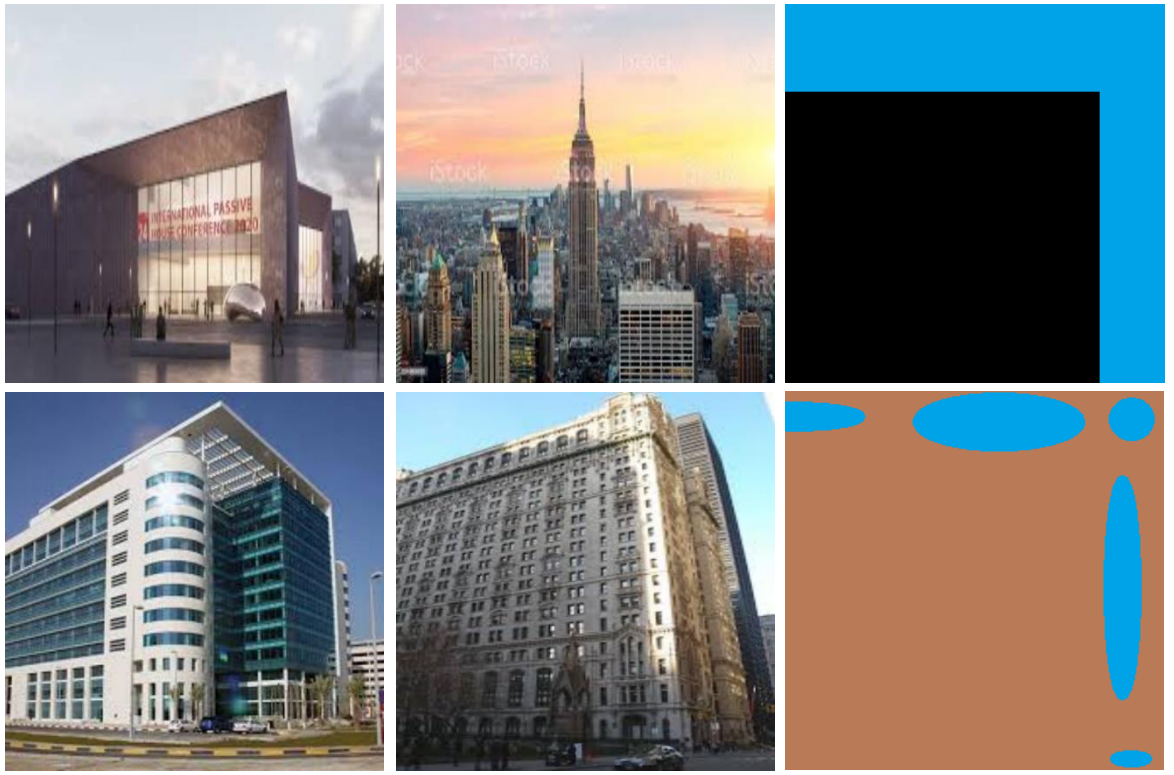
Overfitting

- **Overfitting이란?**
  - 학습데이터를 과하게 잘 학습하는 것



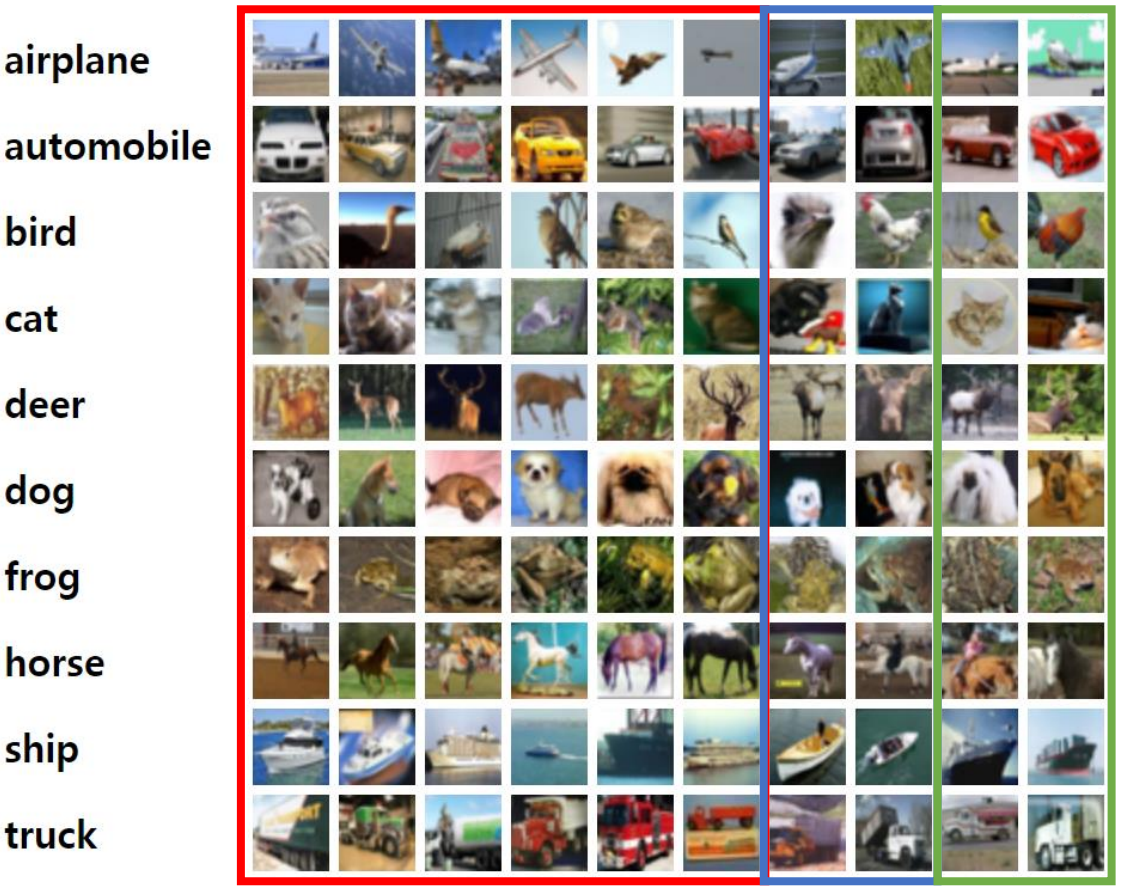
Train

epoch가 많은 경우

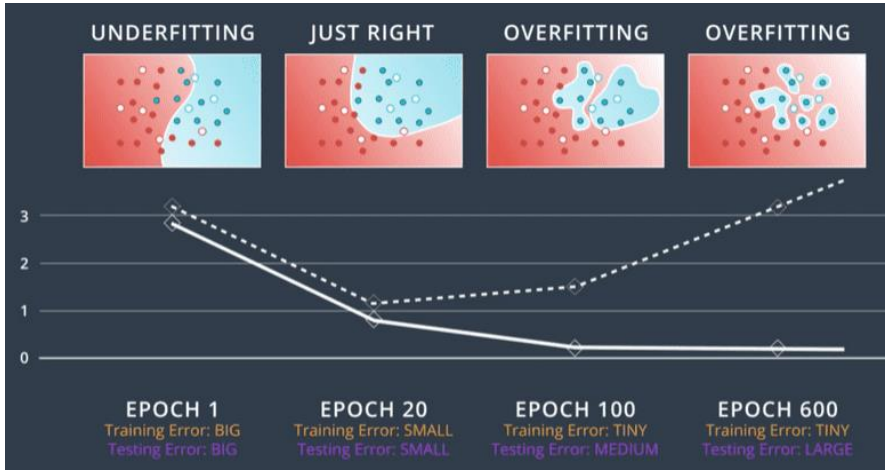


Overfitting

- **Overfitting이란?**
  - 학습데이터를 과하게 잘 학습하는 것



Train
Validation
Test



ResNet50 실습

- **ResNet50으로 학습시키기(저번주)**
  - 재난 영상 데이터셋 학습시키기

	지진	산불	폭설	산사태	낙뢰	폭우	쓰나미	태풍	화산
Img									
Data 개수	120	120	120	120	120	120	120	120	120

ResNet50 실습

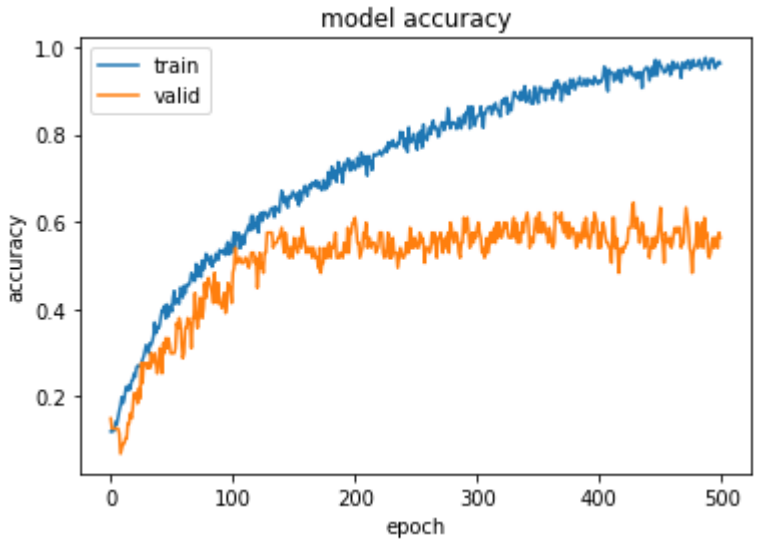
- **ResNet50으로 학습시키기(저번주)**
  - 재난 영상 데이터셋 학습시키기

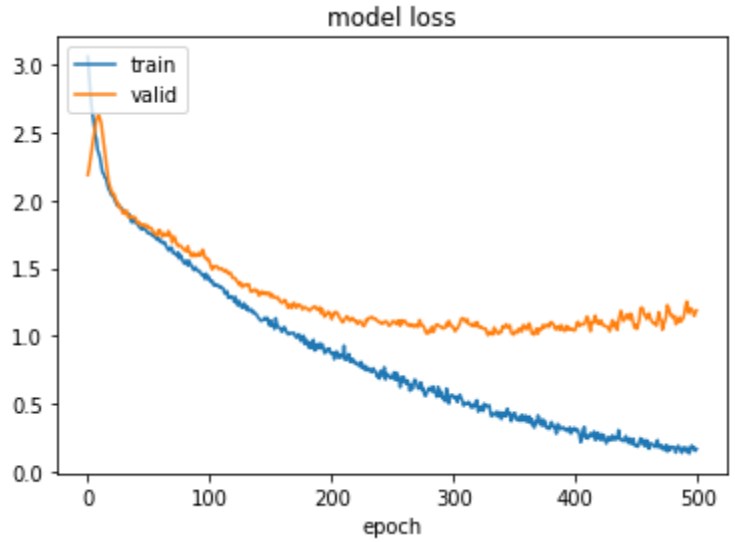
```
[8]  history = model.fit(x=x_train, y=y_train, batch_size=32, epochs=500, validation_data=(x_valid, y_valid))
```

```
Epoch 472/500
25/25 [==============================] - 8s 326ms/step - loss: 0.1674 - acc: 0.9653 - val_loss: 1.0459 - val_acc: 0.6322
Epoch 473/500
25/25 [==============================] - 8s 326ms/step - loss: 0.1818 - acc: 0.9640 - val_loss: 1.0851 - val_acc: 0.6092
Epoch 474/500
25/25 [==============================] - 8s 325ms/step - loss: 0.1626 - acc: 0.9691 - val_loss: 1.1078 - val_acc: 0.5862

...

Epoch 496/500
25/25 [==============================] - 8s 323ms/step - loss: 0.1939 - acc: 0.9511 - val_loss: 1.1947 - val_acc: 0.5632
Epoch 497/500
25/25 [==============================] - 8s 325ms/step - loss: 0.1706 - acc: 0.9614 - val_loss: 1.1834 - val_acc: 0.5632
Epoch 498/500
25/25 [==============================] - 8s 325ms/step - loss: 0.1782 - acc: 0.9588 - val_loss: 1.1474 - val_acc: 0.5402
Epoch 499/500
25/25 [==============================] - 8s 326ms/step - loss: 0.1539 - acc: 0.9665 - val_loss: 1.1750 - val_acc: 0.5747
Epoch 500/500
25/25 [==============================] - 8s 325ms/step - loss: 0.1682 - acc: 0.9627 - val_loss: 1.1867 - val_acc: 0.5632
```

ResNet50 실습

- **ResNet50으로 학습시키기(저번주)**
  - 재난 영상 데이터셋 학습시키기





<Test 데이터 정확도>
acc = 51.85%

ResNet50 실습

- **ResNet50으로 학습시키기**
  - Pre-Training(사전 학습) 이란?
    - 사전에 미리 학습을 시키는 것
    - 사전에 미리 학습한 모델을 Pre-Trained Model이라고 함.

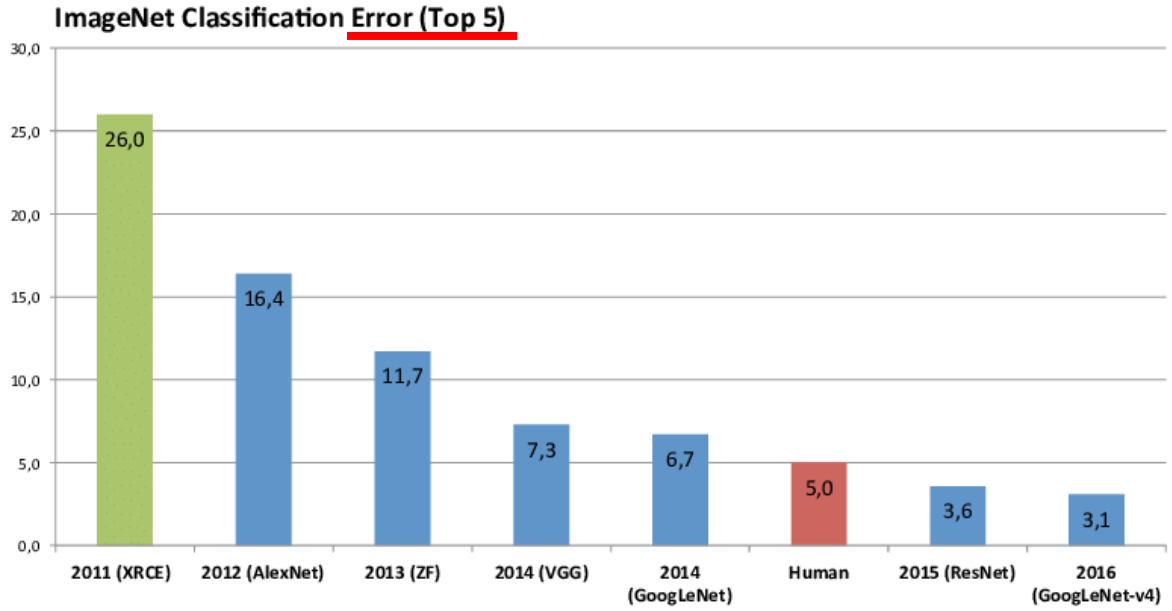
  - Why?
    - Model을 학습시킬 때 weight와 bias를 랜덤하게 초기화 시키는 것이 아니라 사전에 학습 시킨 모델의 weight와 bias로 초기화 시키기 위해

```
#base_model = tf.keras.applications.ResNet50(weights=None, input_shape=(224, 224, 3))
base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224, 224, 3))
```

Resnet50을 ImageNet 데이터셋으로 학습시킨 weight와 bias를 사용하여 base_model의 weight와 bias를 초기화 시킨다.

ResNet50 실습

- **ResNet50으로 학습시키기**
  - ImageNet이란?
    - 이미지 분류에 사용하는 데이터 셋
    - ILSVRC챌린지에서 사용
    - 1400만개 이상의 데이터가 있음



<추가설명>
ILSVRC 챌린지는 1000개의 class 이미지를 분류하는 대회입니다.
그래프를 보시면 Top 5 Error라고 나와있는데 Top 5 Error란 1000개의 클래스 중 상위 5개 안에 실제 정답이 없는 경우가 Top 5 Error입니다.
저희가 보통 했던 것은 (ex: 재난 9개중 1개만 골랐을 때 실제 정답이 아닌 경우) Top 1 Error입니다.

```
#base_model = tf.keras.applications.ResNet50(weights=None, input_shape=(224, 224, 3))
base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224, 224, 3))
```

Resnet50을 ImageNet 데이터셋으로 학습시킨 weight와 bias를 사용하여 base_model의 weight와 bias를 초기화 시킨다.

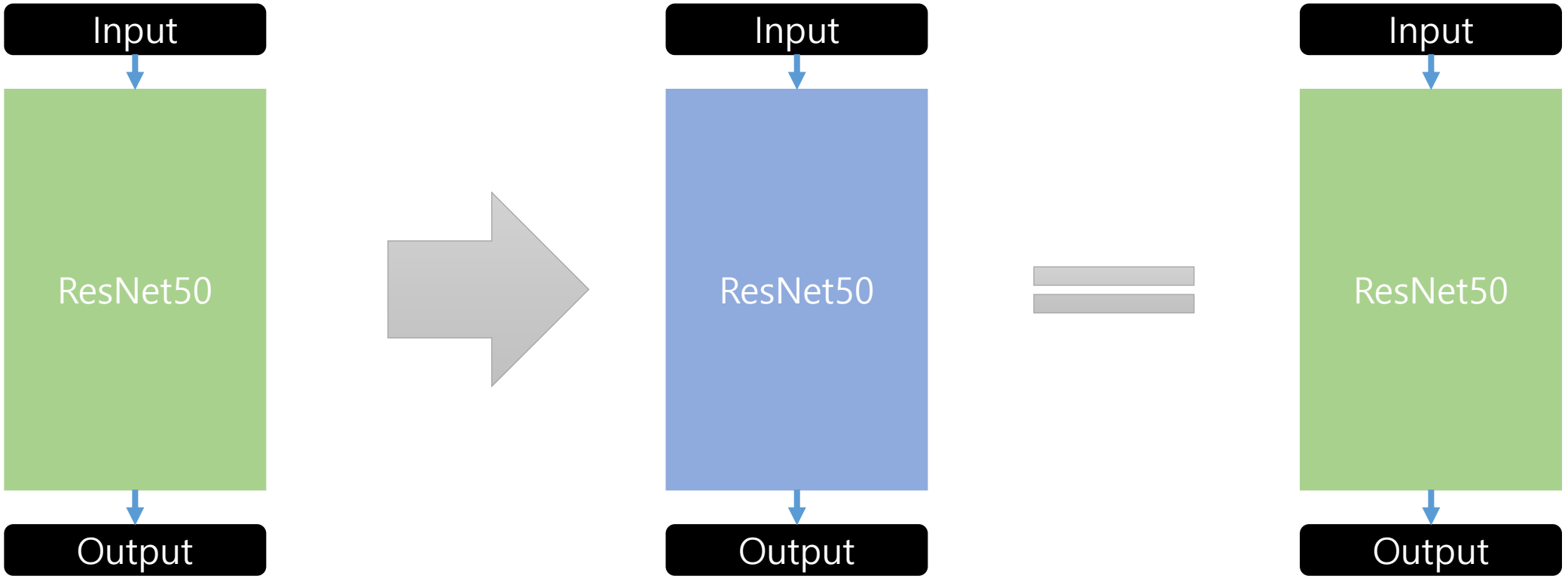
ResNet50 실습

- **ResNet50으로 학습시키기**
  - Pre-Trained Model
    - 사전에 학습된 weight와 bias를 사용하기 위해서는 Model의 구조가 동일해야 함
    - ex: VGG16의 weight, bias를 ResNet50에서 사용할 수 없음
    - ex: VGG16의 weight, bais를 VGG19에서 사용할 수 없음
    - ex: ResNet50의 weight, bais를 ResNet152에서 사용할 수 없음
    - ex: ResNet50의 input size가 다른 경우에도 사용할 수 없음
      (ImageNet 데이터셋은 size가 224x224)

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**



ImageNet으로 학습된 ResNet50

실습에서 사용할 base_model

실습에서 사용할 base_model

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**

<문제>
1. ImageNet은 Class가 1000개이다. 즉, Output이 (1, 1000)이다. 하지만 재난 데이터셋은 Output이 (1,9)이다.

2. 우리가 실습에서 사용하려고 하는 데이터셋은 재난 영상에 대한 데이터셋이다. 하지만 ImageNet은 재난 영상에 대해서는 학습이 되지 않았다.

실습에서 사용할 base_model

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**

<문제>

1. **ImageNet은 Class가 1000개이다. 즉, Output이 (1, 1000)이다. 하지만 재난 데이터셋은 Output이 (1,9)이다.**

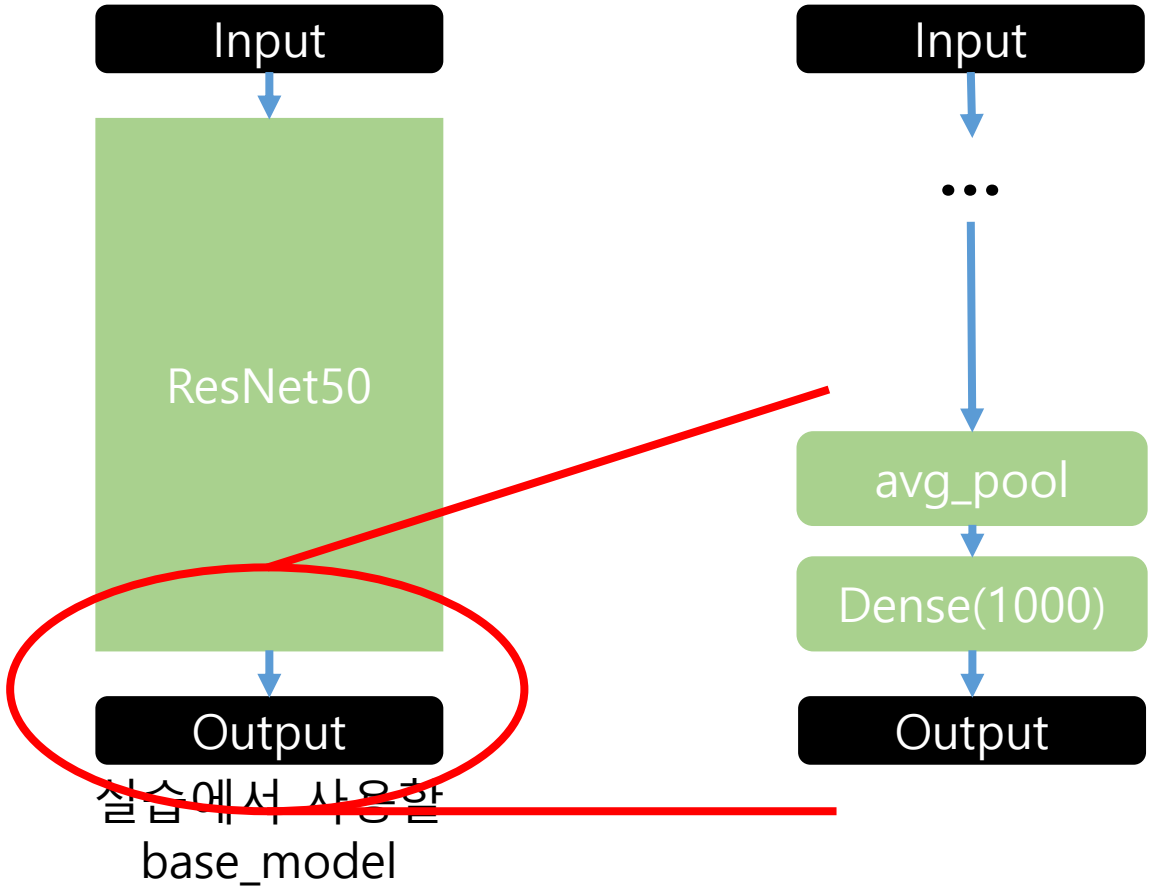
2. 우리가 실습에서 사용하려고 하는 데이터셋은 재난 영상에 대한 데이터셋이다. 하지만 ImageNet은 재난 영상에 대해서는 학습이 되지 않았다.

실습에서 사용할 base_model

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**



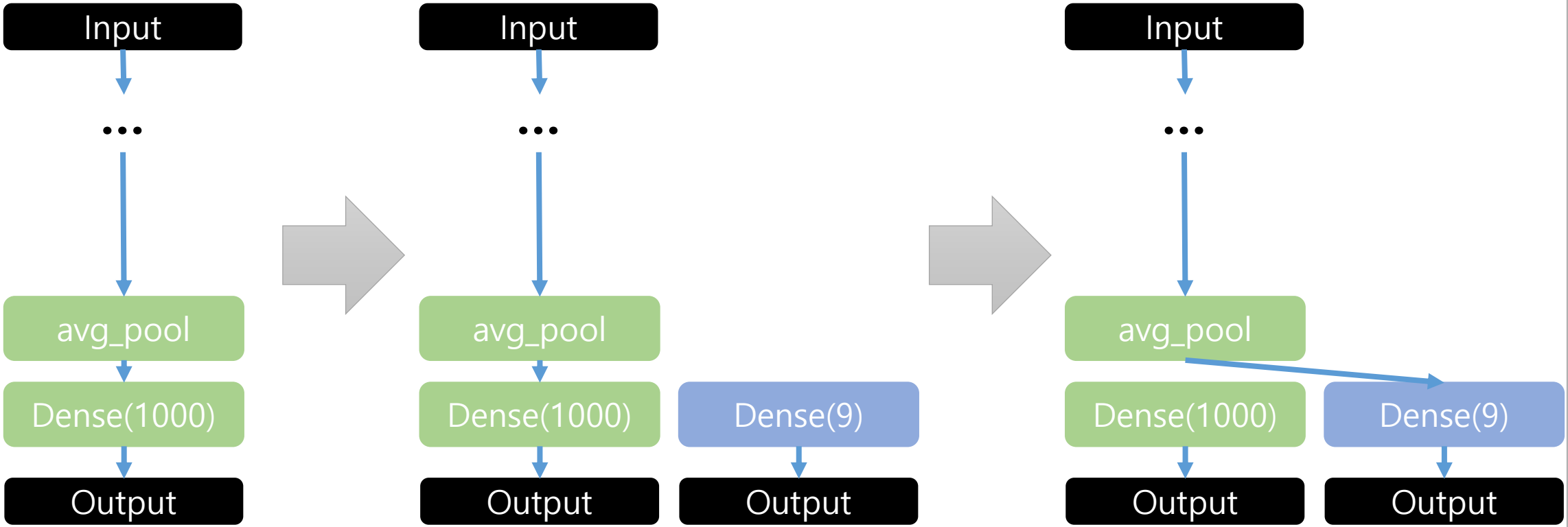
conv5_block3_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_1_conv[0][0]
conv5_block3_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][0]
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block3_1_relu[0][0]
conv5_block3_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][0]
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block3_3_conv[0][0]
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out[0][0] conv5_block3_3_bn[0][0]
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add[0][0]
avg_pool (GlobalAveragePooling2	(None, 2048)	0	conv5_block3_out[0][0]
predictions (Dense)	(None, 1000)	2049000	avg_pool[0][0]

Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**



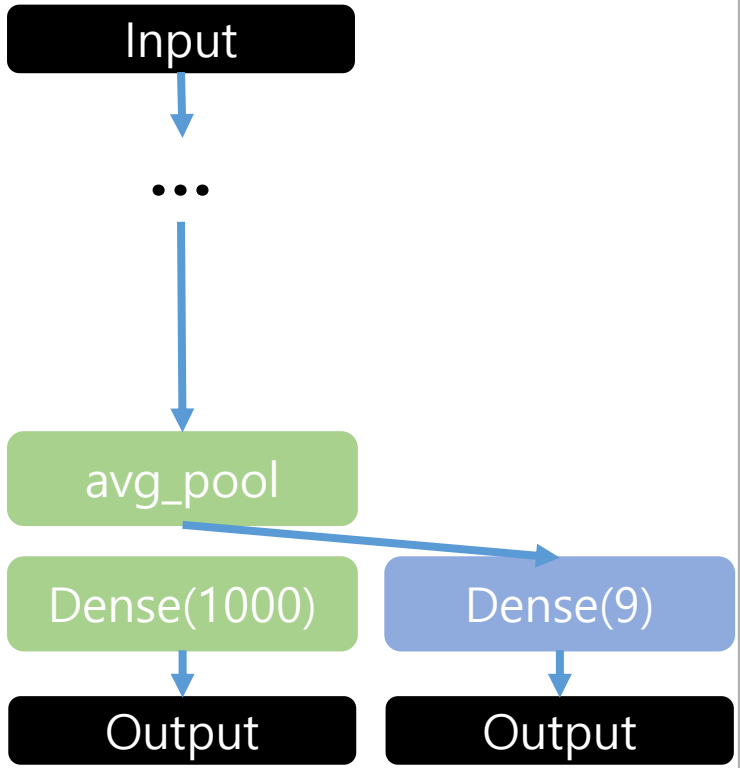
ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**

```
#base_model = tf.keras.applications.ResNet50(weights=None, input_shape=(224, 224, 3))
base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224, 224, 3))
#base_model.summary()
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)
x = base_model.output
pred = tf.keras.layers.Dense(9, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)
```

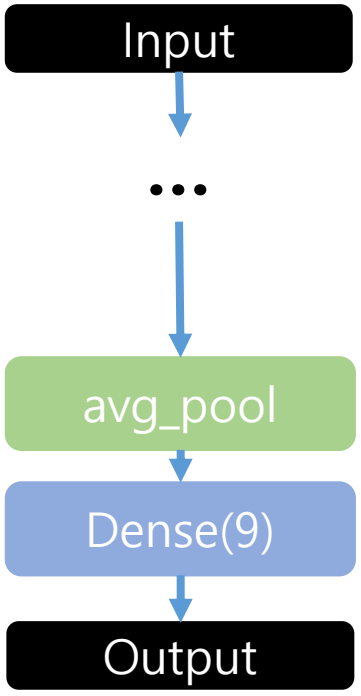
```
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 9)            18441       avg_pool[0][0]
==================================================================================================
Total params: 23,606,153
Trainable params: 23,553,033
Non-trainable params: 53,120
__________________________________________________________________________________________________
```



ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**



실습에서 사용할 모델

```
## pre-training model test
results = model.evaluate(x_test, y_test, batch_size=32)

print('test accuracy')
print(results[1])
```

```
7/7 [==============================] - 0s 71ms/step - loss: 2.9392 - acc: 0.0926
test accuracy
0.09259258955717087
```

마지막 FC Layer가 학습이 안돼서 정확도가 약 9%밖에 나오지 않음.

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Pre-Training**

<문제>

1. ImageNet은 Class가 1000개이다. 즉, Output이 (1, 1000)이다.
   하지만 재난 데이터셋은 Output이 (1,9)이다.

2. **우리가 실습에서 사용하려고 하는 데이터셋은 재난 영상에 대한 데이터셋이다. 하지만 ImageNet은 재난 영상에 대해서는 학습이 되지 않았다.**

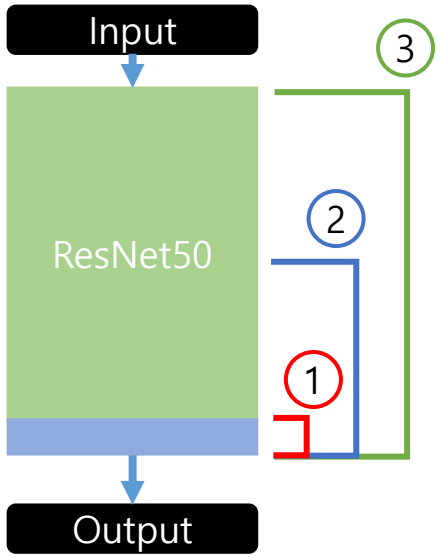
실습에서 사용할 base_model

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - Fine-Tuning이란?
    - 사전에 학습된 모델(pre-trained model)의 weight와 bias를 나의 목적에 맞게 조금 더 미세하게 조절하는 것.
  - 방법
    1. 학습이 안된 Fully Connected Layer 하나만 학습시킨다.
    2. 모델의 일부분을 다시 학습시킨다
    3. 모델의 전체를 다시 학습시킨다

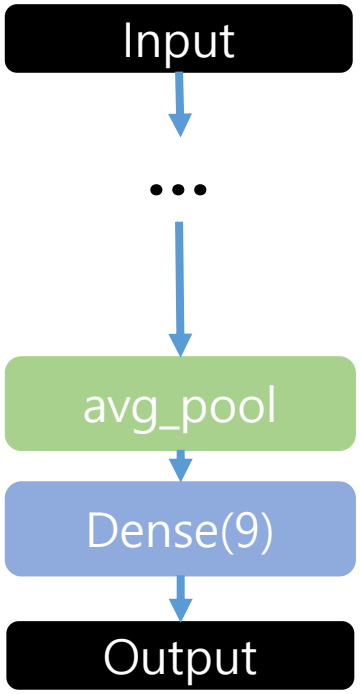
    2, 3번의 경우 learning rate를 조금 작게 설정한다.



ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 학습이 안된 Fully Connected Layer 하나만 학습시킨다.



실습에서 사용할 모델

```
#base_model = tf.keras.applications.ResNet50(weights=None, input_shape=(224, 224, 3))
base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224, 224, 3))
#base_model.summary()
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

base_model.trainable = False

x = base_model.output
pred = tf.keras.layers.Dense(9, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)
```

이미 학습이 되어있으므로 추가 학습을 시키지 않는다.

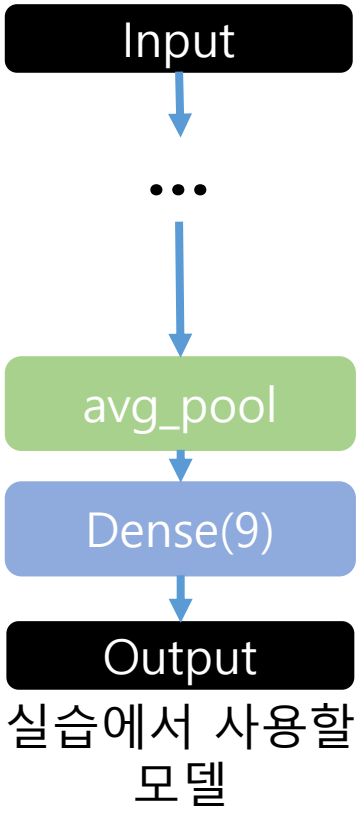
```
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 9)            18441       avg_pool[0][0]
==================================================================================================
Total params: 23,606,153
Trainable params: 18,441
Non-trainable params: 23,587,712
__________________________________________________________________________________________________
```

마지막 FC 층만 Trainable인 것을 확인

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 학습이 안된 Fully Connected Layer 하나만 학습시킨다.



실습에서 사용할 모델

```
## pre-training model test
results = model.evaluate(x_test, y_test, batch_size=32)

print('test accuracy')
print(results[1])
```

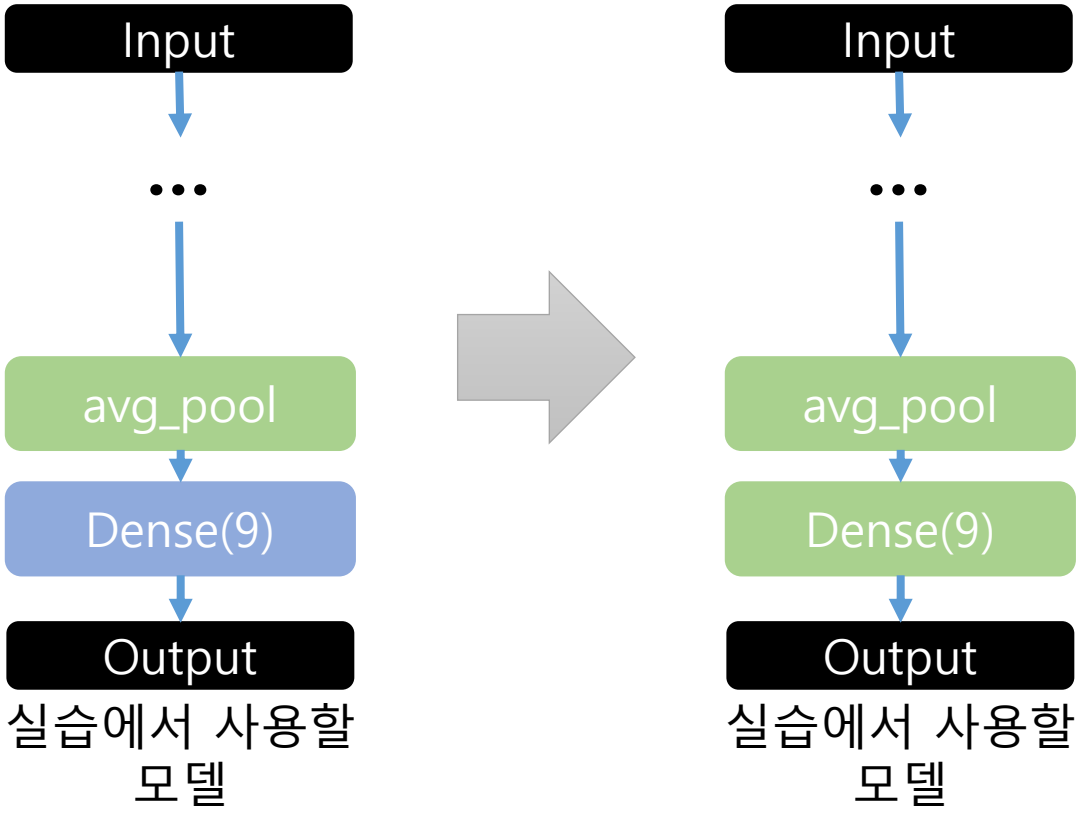
```
7/7 [==============================] - 0s 71ms/step - loss: 2.9392 - acc: 0.0926
test accuracy
0.09259258955717087
```

마지막 FC Layer가 학습이 안돼서 정확도가 약 9%밖에 나오지 않음.

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 학습이 안된 Fully Connected Layer 하나만 학습시킨다.



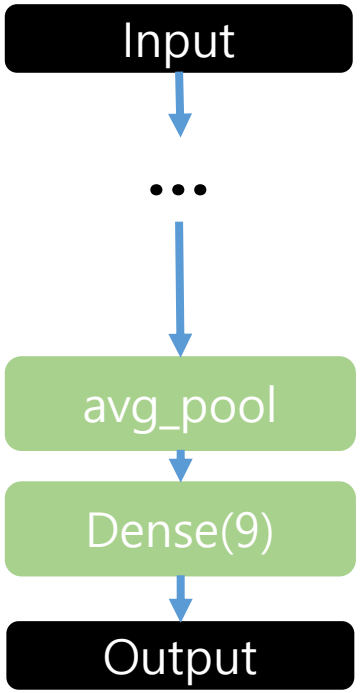
hyperparameter는 변경하지 않음.
learning rate : 0.0001
batch size : 32

FC Layer 하나만 학습 시킴.
epoch는 알아서 조절하기

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 학습이 안된 Fully Connected Layer 하나만 학습시킨다.



실습에서 사용할 모델

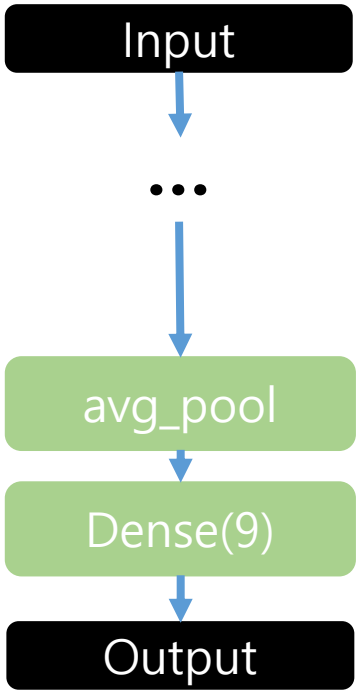
```
[9]  history = model.fit(x=x_train, y=y_train, batch_size=32, epochs=200, validation_data=(x_valid, y_valid))
```

```
25/25 [==============================] - 2s 97ms/step - loss: 0.0196 - acc: 1.0000 - val_loss: 0.5861 - val_acc: 0.7931
Epoch 172/200
25/25 [==============================] - 2s 98ms/step - loss: 0.0194 - acc: 1.0000 - val_loss: 0.5826 - val_acc: 0.7931
Epoch 173/200
25/25 [==============================] - 2s 98ms/step - loss: 0.0191 - acc: 1.0000 - val_loss: 0.5839 - val_acc: 0.7931
Epoch 174/200
25/25 [==============================] - 2s 98ms/step - loss: 0.0189 - acc: 1.0000 - val_loss: 0.5872 - val_acc: 0.7931
Epoch 175/200
25/25 [==============================] - 2s 97ms/step - loss: 0.0186 - acc: 1.0000 - val_loss: 0.5829 - val_acc: 0.7931
Epoch 176/200
25/25 [==============================] - 2s 98ms/step - loss: 0.0184 - acc: 1.0000 - val_loss: 0.5871 - val_acc: 0.8046
Epoch 177/200
25/25 [==============================] - 2s 97ms/step - loss: 0.0181 - acc: 1.0000 - val_loss: 0.5808 - val_acc: 0.7816
Epoch 178/200
25/25 [==============================] - 2s 98ms/step - loss: 0.0178 - acc: 1.0000 - val_loss: 0.5856 - val_acc: 0.7931
Epoch 179/200
25/25 [==============================] - 2s 98ms/step - loss: 0.0176 - acc: 1.0000 - val_loss: 0.5893 - val_acc: 0.8046
```

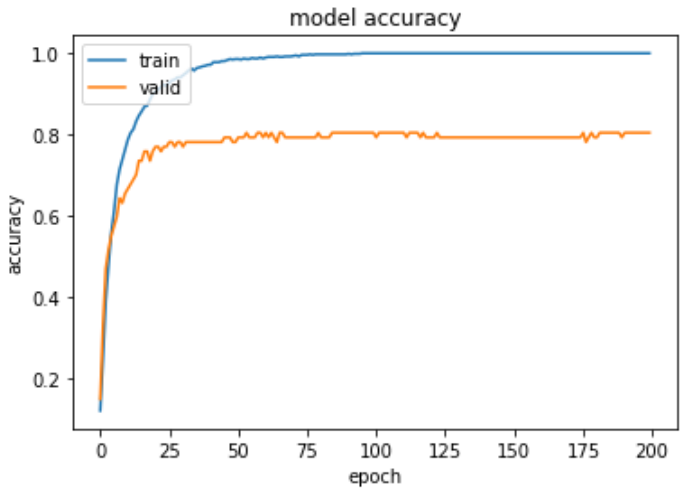
ResNet50 실습

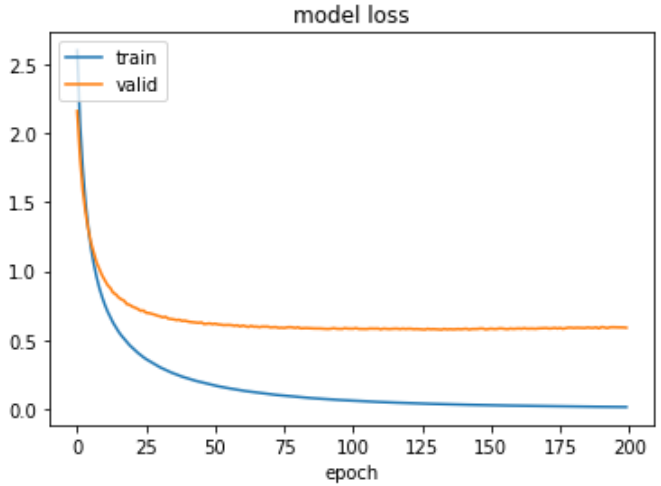
: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 학습이 안된 Fully Connected Layer 하나만 학습시킨다.



실습에서 사용할 모델





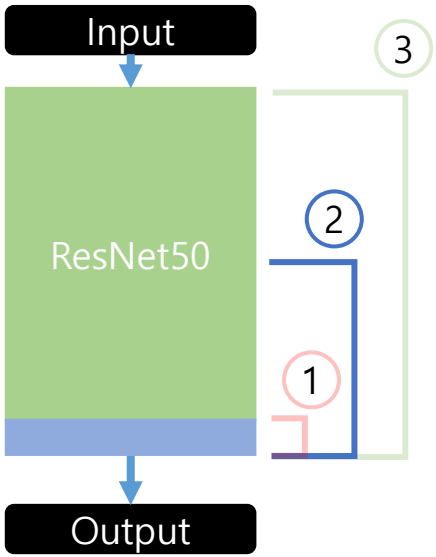

```
[12]  results = model.evaluate(x_test, y_test, batch_size=32)

      print('test accuracy')
      print(results[1])
```

```
7/7 [==============================] - 1s 77ms/step - loss: 0.6561 - acc: 0.8194
test accuracy
0.8194444179534912
```

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 일부분을 다시 학습시킨다




```
for layer in model.layers:
  layer.trainable = True

for idx, layer in enumerate(model.layers):
  print(idx, layer)
  if idx < 100:
    layer.trainable = False

model.summary()
```

(Idx, layer)

```
164 <tensorflow.python.keras.layers.core.Activation object at 0x7f2fec218710>
165 <tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f2fec230c50>
166 <tensorflow.python.keras.layers.normalization_v2.BatchNormalization object at 0x7f2fec2352e8>
167 <tensorflow.python.keras.layers.core.Activation object at 0x7f2fec235978>
168 <tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f2fec235b38>
169 <tensorflow.python.keras.layers.normalization_v2.BatchNormalization object at 0x7f2fec1ce9b0>
170 <tensorflow.python.keras.layers.core.Activation object at 0x7f2fec1cef98>
171 <tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f2fec1ceda0>
172 <tensorflow.python.keras.layers.normalization_v2.BatchNormalization object at 0x7f2fec1f1898>
173 <tensorflow.python.keras.layers.merge.Add object at 0x7f2fec1f1ef0>
174 <tensorflow.python.keras.layers.core.Activation object at 0x7f2fec1f1f28>
175 <tensorflow.python.keras.layers.pooling.GlobalAveragePooling2D object at 0x7f2fec18ad30>
176 <tensorflow.python.keras.layers.core.Dense object at 0x7f2fec18cd30>
```

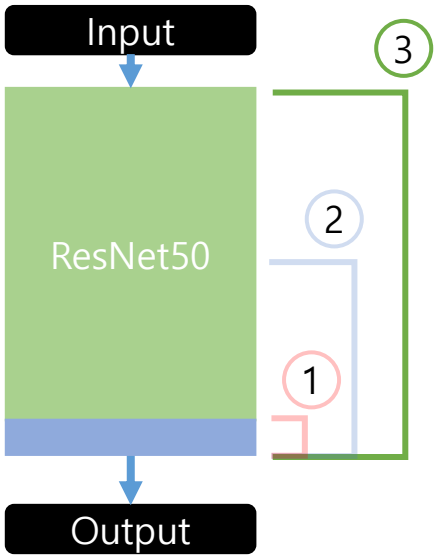
```
Model: "resnet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 100)          204900      avg_pool[0][0]
==================================================================================================
Total params: 23,792,612
Trainable params: 19,657,828
Non-trainable params: 4,134,784
__________________________________________________________________________________________________
```

FC Layer가 176번 Idx인걸 알 수 있음

ResNet50 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다




```
[6]  base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224, 224, 3))
     base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)
     x = base_model.output
     pred = tf.keras.layers.Dense(100, activation='softmax')(x)
     model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)

     opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
     model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['acc'])

     model.summary()
```

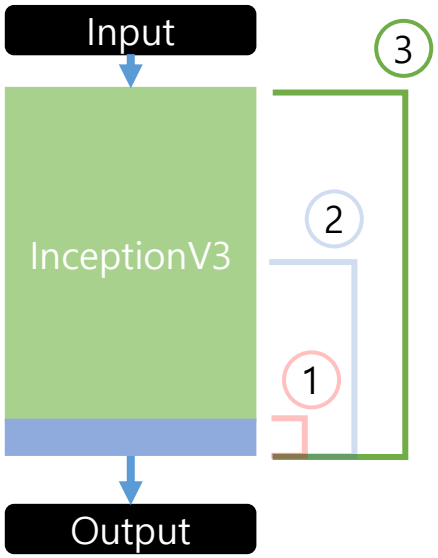
```
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0][0]
__________________________________________________________________________________________
dense (Dense)                   (None, 100)          204900      avg_pool[0][0]
==========================================================================================
Total params: 23,792,612
Trainable params: 23,739,492
Non-trainable params: 53,120
__________________________________________________________________________________________
```

```
for layer in model.layers:
  layer.trainable = True
```

InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다



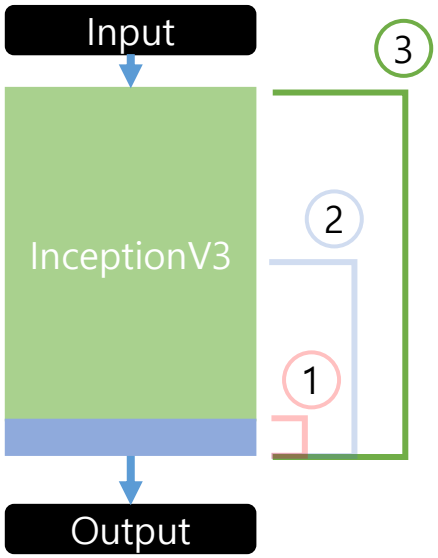

```
import cv2
import numpy as np
import tensorflow as tf

data_x = np.load('drive/My Drive/imgs.npy')
data_y = np.load('drive/My Drive/labels.npy')

print(data_x.shape)
print(data_y.shape)
```

```
(1080, 299, 299, 3)
(1080,)
```

InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다



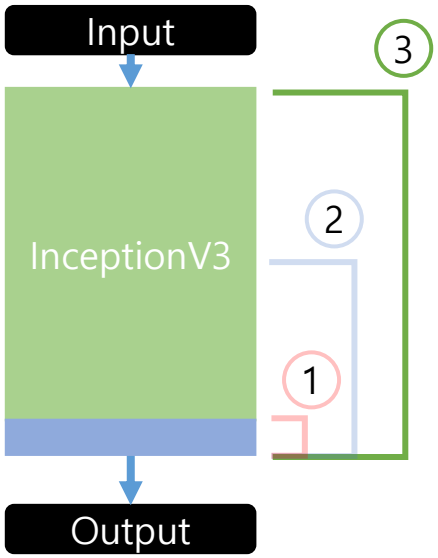

```
base_model = tf.keras.applications.InceptionV3(weights='imagenet', input_shape=(299, 299, 3))
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

x = base_model.output
pred = tf.keras.layers.Dense(9, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)

opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['acc'])

model.summary()
```

```
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 8, 8, 768)    0           activation_185[0][0]
                                                                 activation_186[0][0]
__________________________________________________________________________________________________
activation_187 (Activation)     (None, 8, 8, 192)    0           batch_normalization_187[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 8, 8, 2048)   0           activation_179[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_3[0][0]
                                                                 activation_187[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           mixed10[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 9)            18441       avg_pool[0][0]
==================================================================================================
Total params: 21,821,225
Trainable params: 21,786,793
Non-trainable params: 34,432
__________________________________________________________________________________________________
```

InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다




```
x_train = np.array(data_x)
y_train = np.reshape(data_y, newshape=(len(data_y), 1))
```

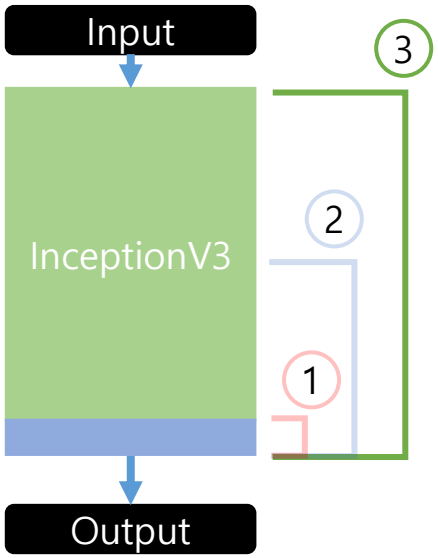
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.2, random_state=123)
x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size=0.1, random_state=123)

print('train data')
print(x_train.shape)
print(y_train.shape)

print('validation data')
print(x_valid.shape)
print(y_valid.shape)

print('test data')
print(x_test.shape)
print(y_test.shape)
```

```
train data
(777, 299, 299, 3)
(777, 1)
validation data
(87, 299, 299, 3)
(87, 1)
test data
(216, 299, 299, 3)
(216, 1)
```

InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다

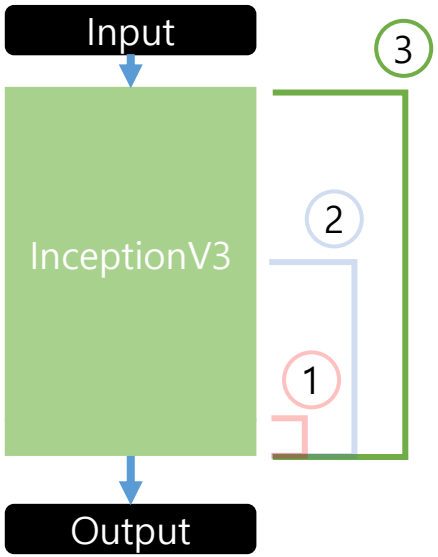



```
[12]  ## pre-training model test
      results = model.evaluate(x_test, y_test, batch_size=32)

      print('test accuracy')
      print(results[1])

      7/7 [==============================] - 2s 215ms/step - loss: 48.0268 - acc: 0.1574
      test accuracy
      0.15740740299224854
```

InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다

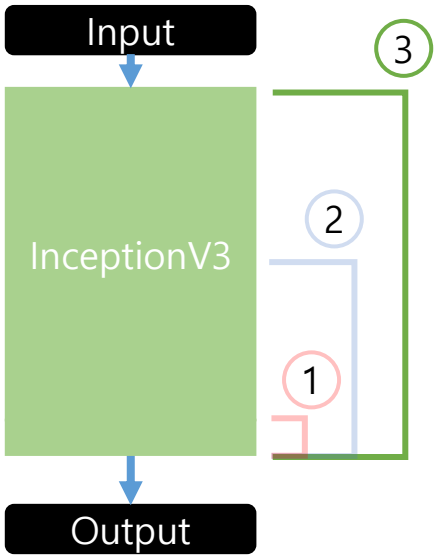



```
[14]  history = model.fit(x=x_train, y=y_train, batch_size=32, epochs=20, validation_data=(x_valid, y_valid))

Epoch 1/20
 2/25 [=>............................] - ETA: 4s - loss: 2.3300 - acc: 0.1250WARNING:tensorflow:Callbacks method `on_train
25/25 [==============================] - 13s 521ms/step - loss: 1.2812 - acc: 0.5792 - val_loss: 3.2174 - val_acc: 0.2184
Epoch 2/20
25/25 [==============================] - 11s 440ms/step - loss: 0.1786 - acc: 0.9575 - val_loss: 1.7509 - val_acc: 0.5172

                                                    ...

Epoch 14/20
25/25 [==============================] - 11s 444ms/step - loss: 0.0143 - acc: 0.9974 - val_loss: 0.5369 - val_acc: 0.8621
Epoch 15/20
25/25 [==============================] - 11s 443ms/step - loss: 0.0351 - acc: 0.9871 - val_loss: 0.6367 - val_acc: 0.8276
Epoch 16/20
25/25 [==============================] - 11s 442ms/step - loss: 0.0612 - acc: 0.9858 - val_loss: 0.5159 - val_acc: 0.8506
Epoch 17/20
25/25 [==============================] - 11s 441ms/step - loss: 0.0442 - acc: 0.9897 - val_loss: 0.6037 - val_acc: 0.8276
Epoch 18/20
25/25 [==============================] - 11s 441ms/step - loss: 0.0301 - acc: 0.9923 - val_loss: 0.5077 - val_acc: 0.8851
Epoch 19/20
25/25 [==============================] - 11s 442ms/step - loss: 0.0302 - acc: 0.9949 - val_loss: 0.4815 - val_acc: 0.8966
Epoch 20/20
25/25 [==============================] - 11s 443ms/step - loss: 0.0178 - acc: 0.9949 - val_loss: 0.6324 - val_acc: 0.8621
```

InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다

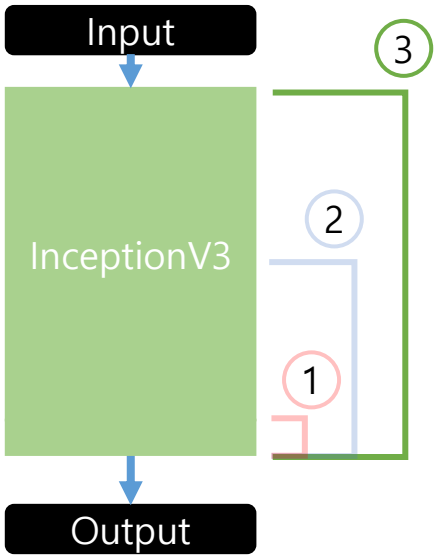


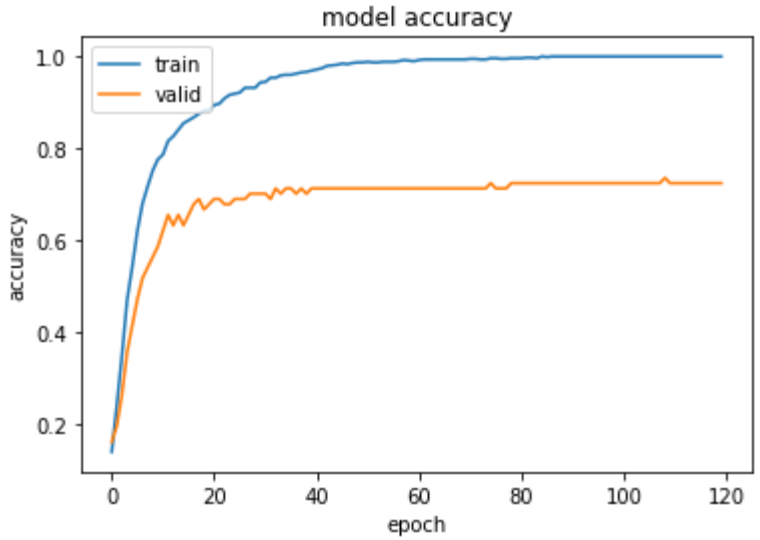

```
import matplotlib.pyplot as plt

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```

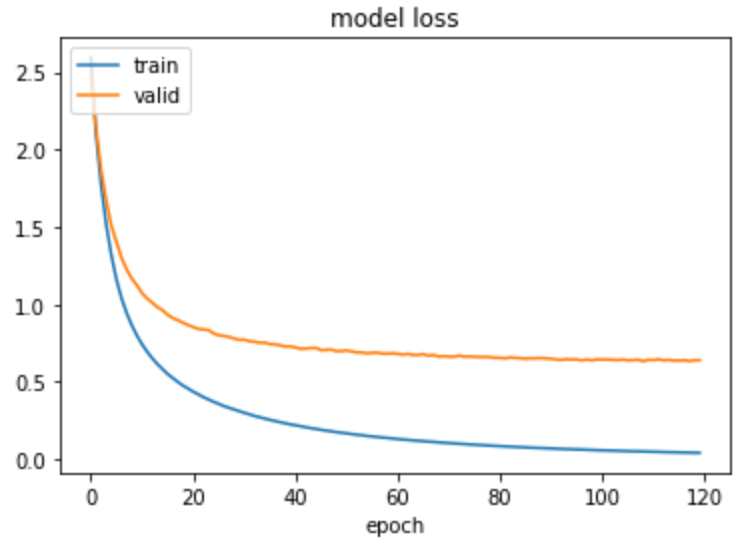
InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다



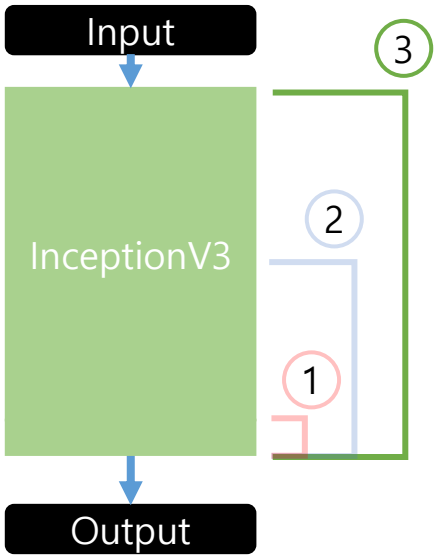




InceptionV3 실습

: ImageNet으로 학습됨
: 학습 안됨

- **Fine-Tuning**
  - 모델의 전체를 다시 학습시킨다




```
[ ]  print('validation accuracy')
     print(history.history['val_acc'][-1])
     print(np.max(history.history['val_acc']))
```

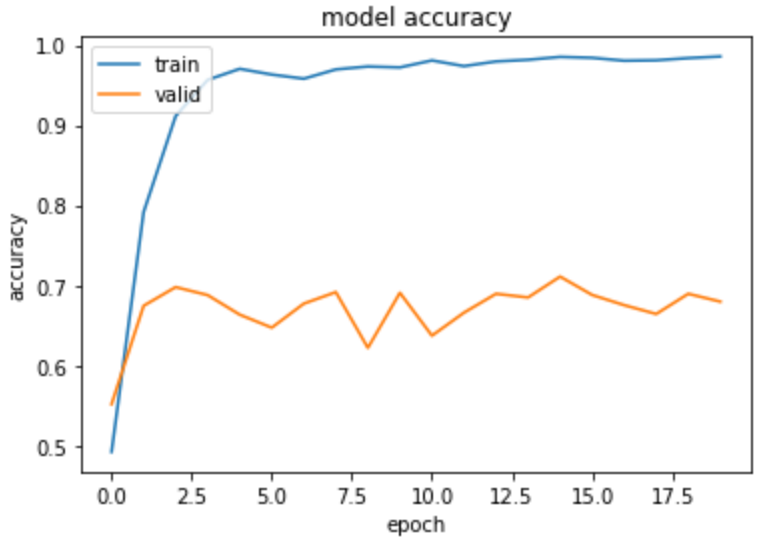
```
validation accuracy
0.7241379022598267
0.7356321811676025
```

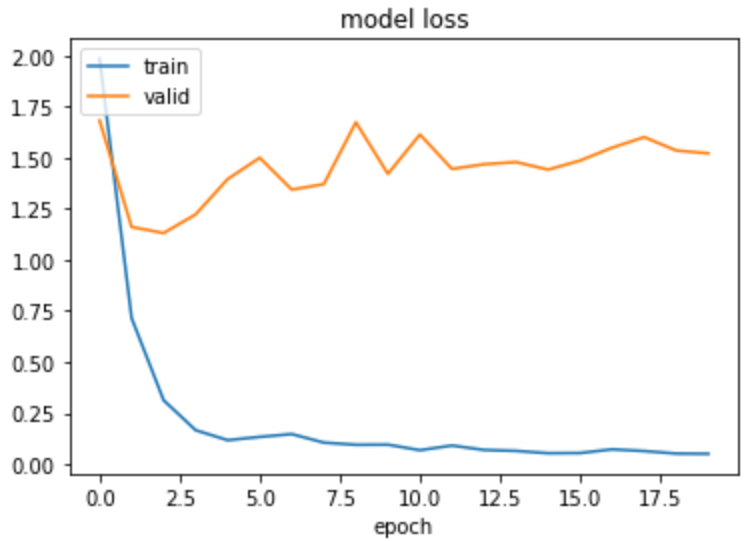
```
[ ]  results = model.evaluate(x_test, y_test, batch_size=32)

     print('test accuracy')
     print(results[1])
```

```
7/7 [==============================] - 1s 76ms/step - lo
test accuracy
0.7916666865348816
```

과제

- **ResNet50 네트워크 학습시키기**
  - 실습에서 진행한 것 처럼 보고서에 자세한 설명 필요(실습 참고)
  - Cifar-100 데이터셋 학습시키기(테스트 데이터 정확도 60% 이상)





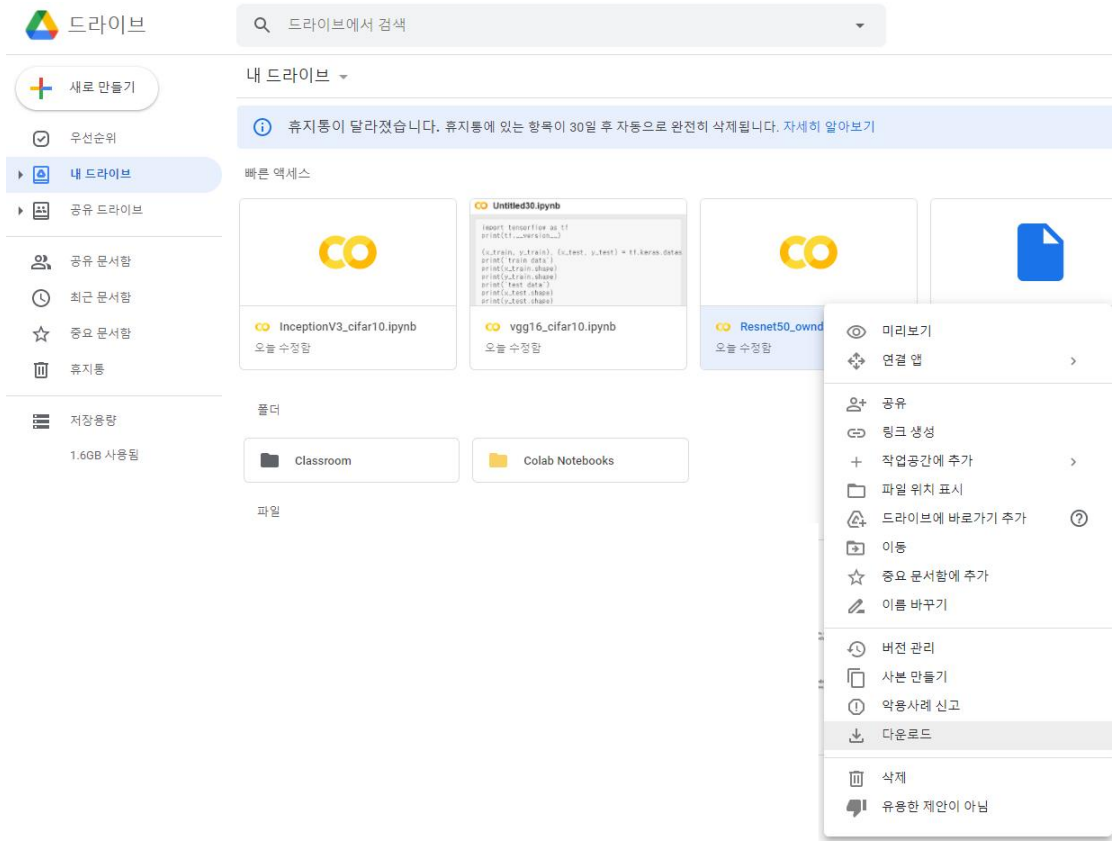

```
[ ]  results = model.evaluate(x_test, y_test, batch_size=32)

     print('test accuracy')
     print(results[1])

     188/188 [==============================] - 10s 54ms/step -
     test accuracy
     0.6658333539962769
```

과제

- **ResNet50 네트워크 학습시키기 주의할점**
  - cifar100 데이터셋의 크기를 224x224로 변경하기
    - 주의할점은 cifar100은 총 60000개의 이미지가 있는데 이를 전부 224x224로 변경하면 memory가 너무 많이 들어 colab이 멈추는 현상이 있을 수 있음. 데이터의 수를 6만개 전부 쓰지말고 조금만 쓰기(저의 경우엔 Train:19200개, validation:4800개, test:6000개를 사용했습니다.)
  - Model 잘 만들기
    - Input size, output size, pre-trained model 사용하기 등...
    - 전체를 다 재학습 시킬건지, 중간부터 재학습 시킬건지, 마지막 FC Layer만 재학습 시킬건지 알아서 선택하기.

과제 - 코드

- 자신이 실행한 코드(.ipynb) 제출
  - 구글 드라이브에서 다운 가능



과제 - 보고서

- **보고서**
  - **내용:**
    - 이름, 학번, 학과
    - 구현 코드: 구현한 코드(테스트 데이터 정확도 60% 이상)
    - 코드 설명: 구현한 코드에 대한 설명(실습에서 한 것 처럼 작성하기)
    - 느낀 점 : 결과를 보고 느낀 점, 혹은 과제를 하면서 어려웠던 점 등
    - 과제 난이도: 개인적으로 생각하는 난이도 (과제가 너무 쉬운 것 같다 등)

  - .pdf 파일로 제출 (이 외의 파일 형식일 경우 감점)

  - **파일 이름:**
    - [CG]20xxxxxxx_이름_n주차_과제.pdf

과제

- **제출 기한**
  - 12월 09일 23시 59분까지 (최대 점수 10점)
- **추가 제출 기한**
  - 12월 16일 23시 59분까지 (최대 점수 4점, 과제 총점 계산 후 -6점)
  - 12월 17일 00시 00분 이후 (점수 0점)
- **채점**
  - 구현을 못하거나(잘못 구현하거나) 보고서 내용이 빠진 경우 감점
  - 아무것도 구현하지 못해도 과제 제출하면 기본점수 있음
  - 다른 사람의 코드를 copy해서 제출시 보여준 사람, copy한 사람 둘 다 0점
- **제출 파일**
  - 아래의 파일을 압축해서 [CG]20xxxxxxx_이름_n주차_과제.zip로 제출
    - .ipynb 파일
    - .pdf 보고서 파일
    - 데이터셋은 첨부할 필요 없음

QnA