Img Classification Transfer Learning VGGnet

Fashion Items : Shoes

Itwill 12th LKYJ Team

CONTENTS

01. Datasets

02. CNN

03. VGGnet

04. Transfer Learning

05. Tuning

01

Datasets

1. Web scrolling
2. Get picture by camera
3. Pre-processing

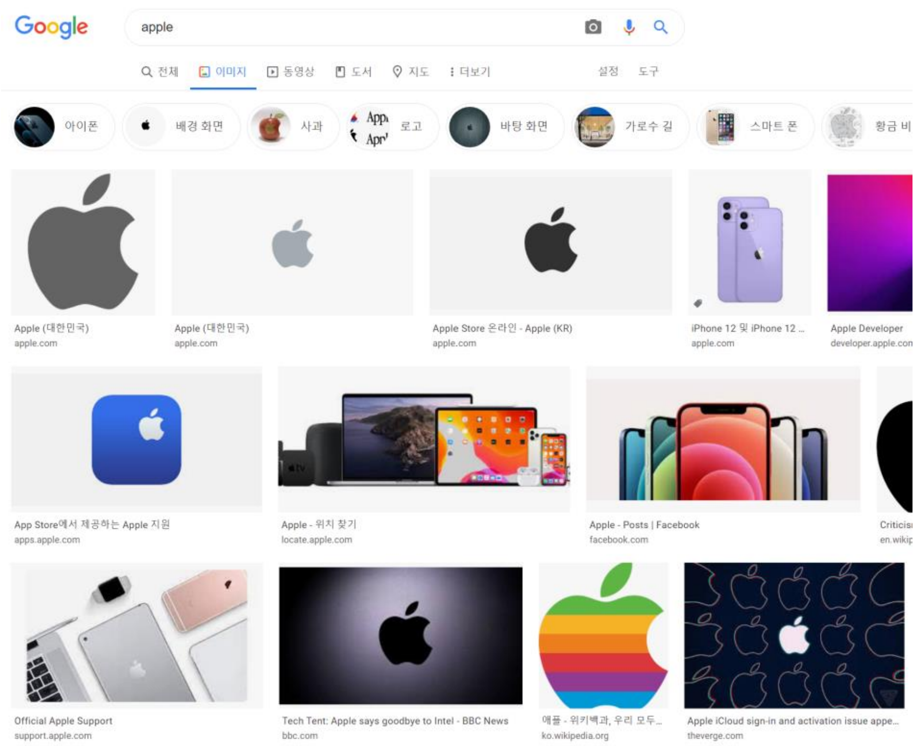
01. Web scrolling

- Web scrolling sites : Google, Pinterest, Naver, Instagram, Bing

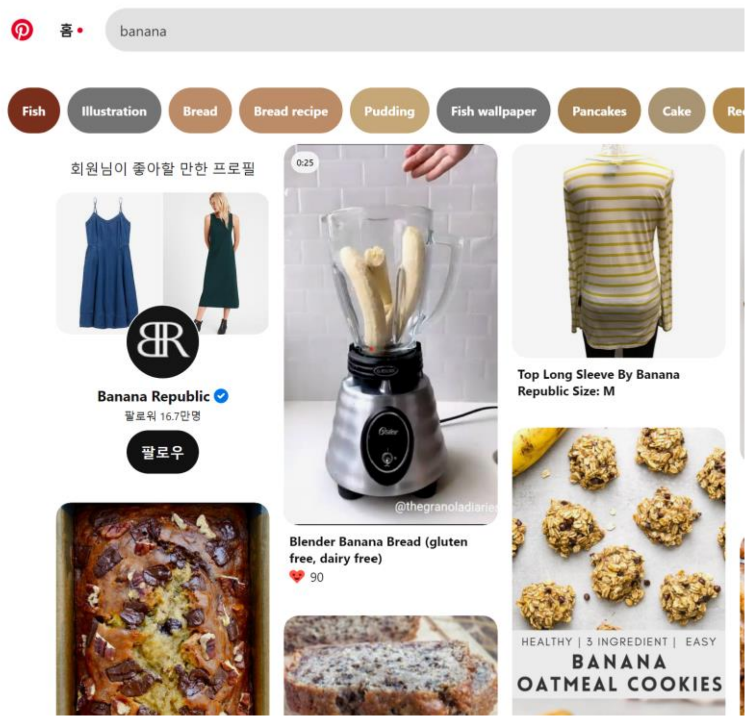
- Python code : https://github.com/LemonChocolate/Web_Scrolling_img

- Get srcs

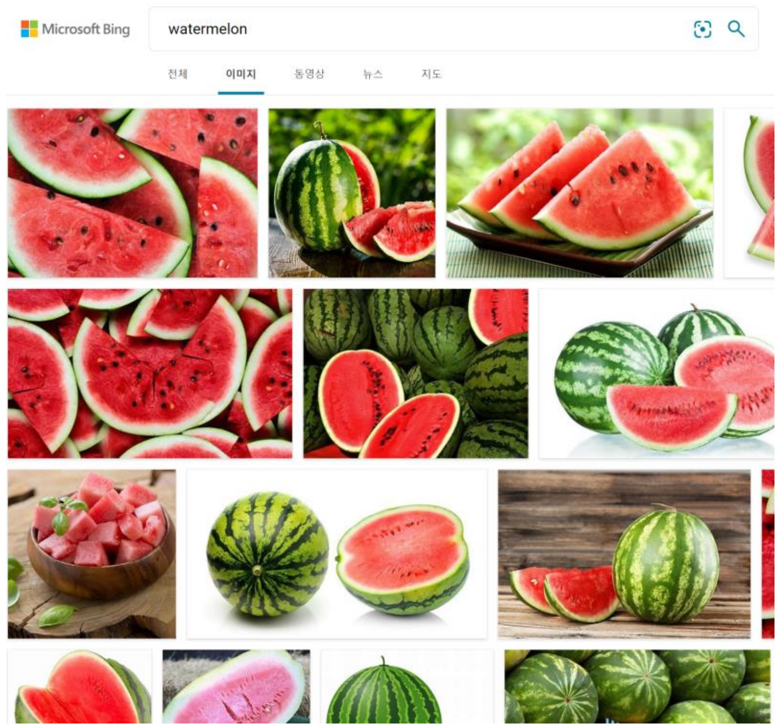
Google



Pinterest



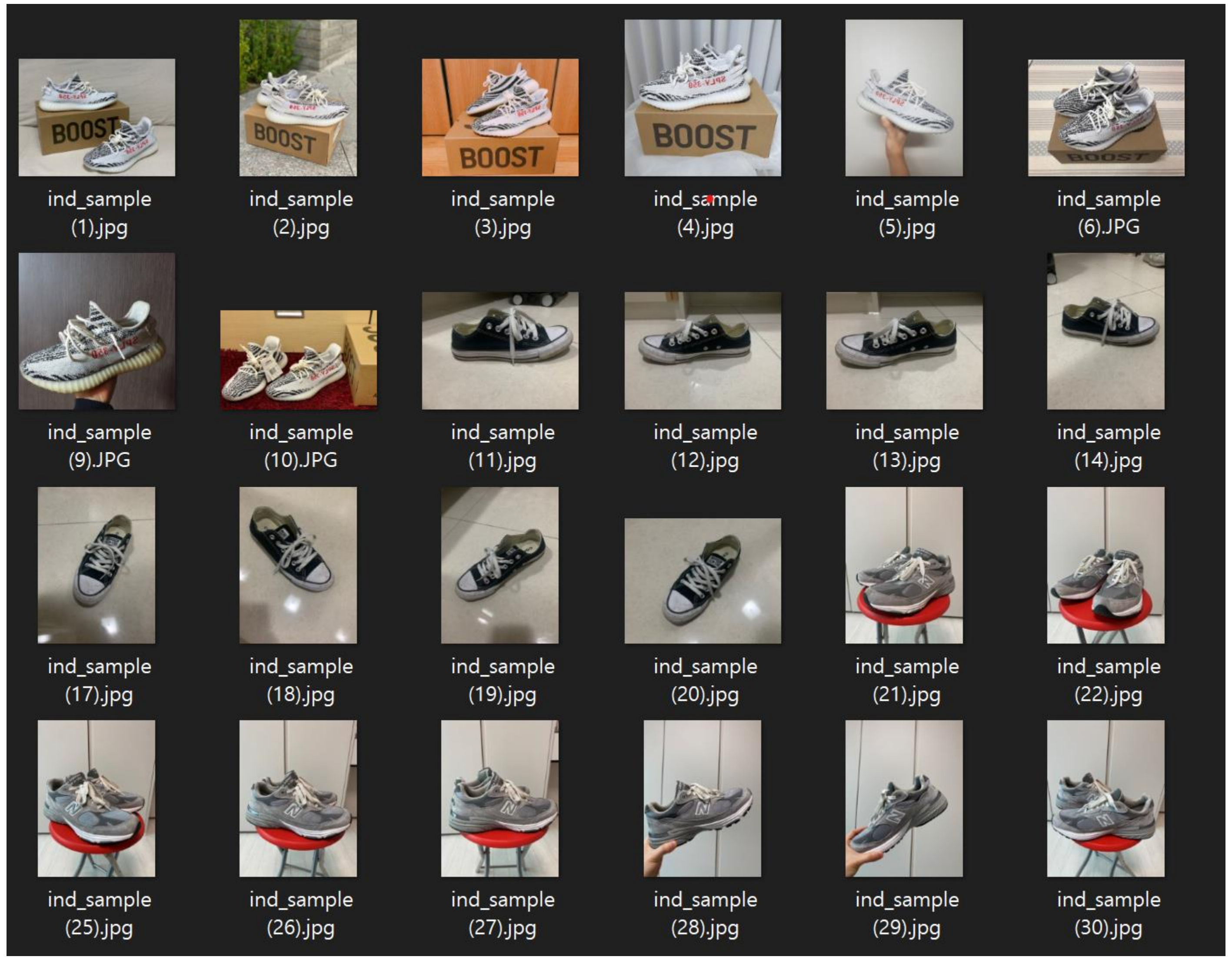
Bing



02. Get picture by camera

■ Take a picture for Testsets

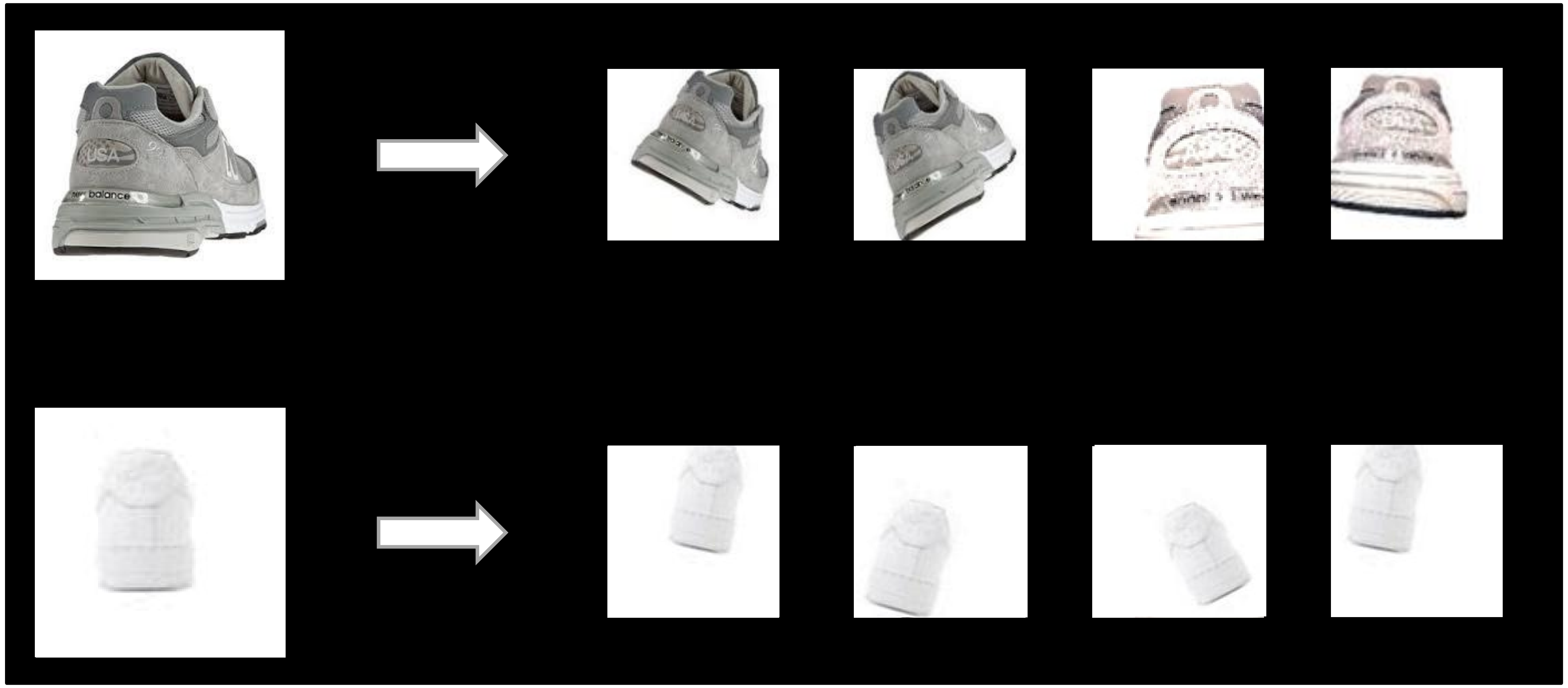
Testsets (5 classes, 50 imgs)



Trainsets (5 classes, 9000 imgs)

03. Pre-processing

- Data generating in local
- 600 imgs to 1800 imgs
- Examples



02

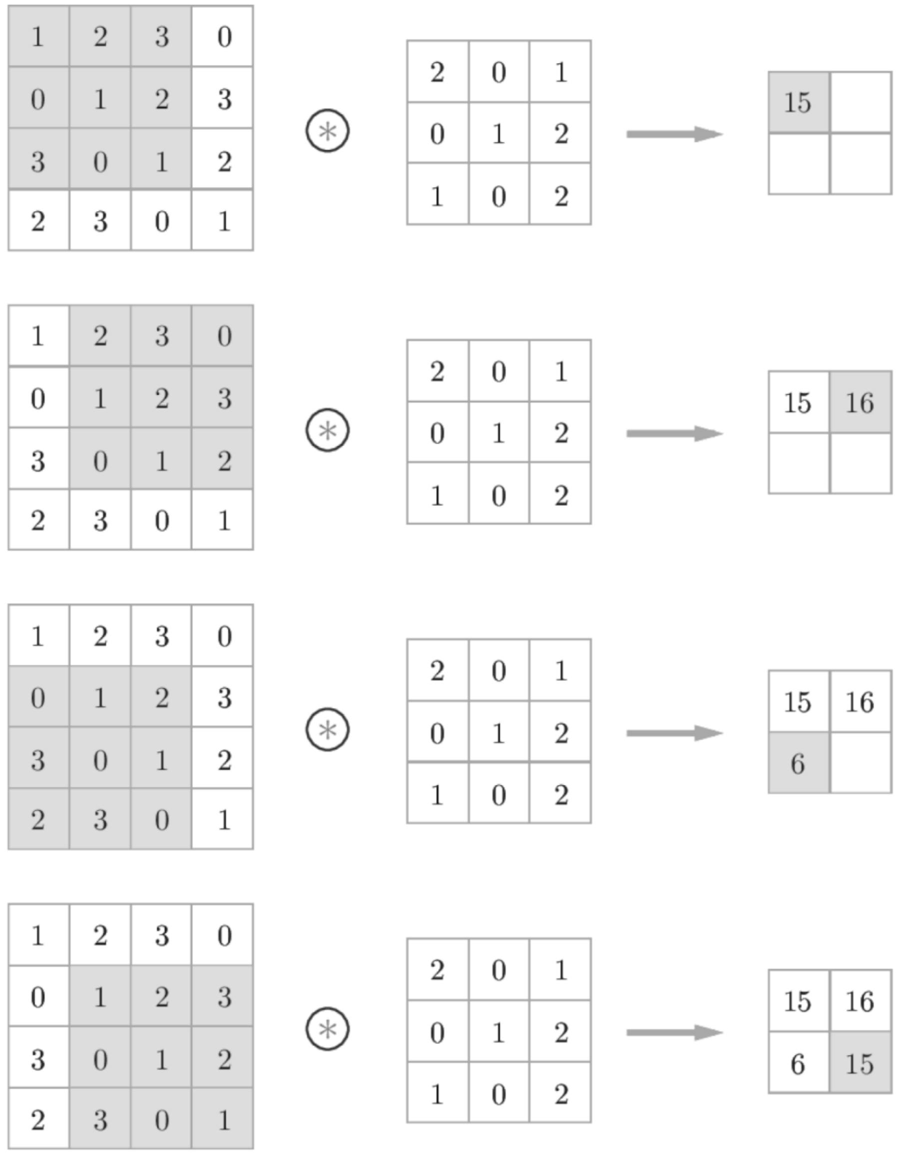
CNN

1. Layer
2. Activation
3. Paper

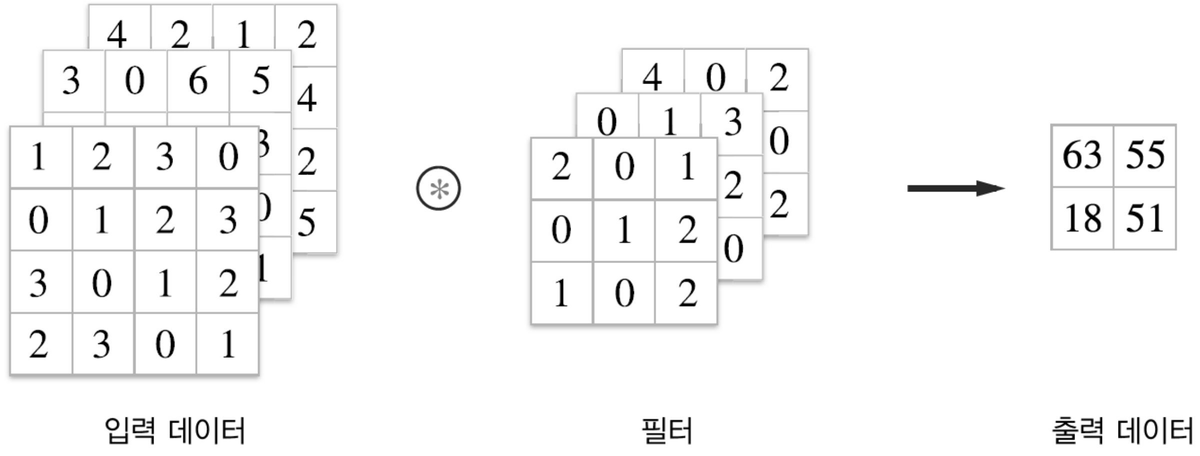
01. Layer

■ Convolution layer

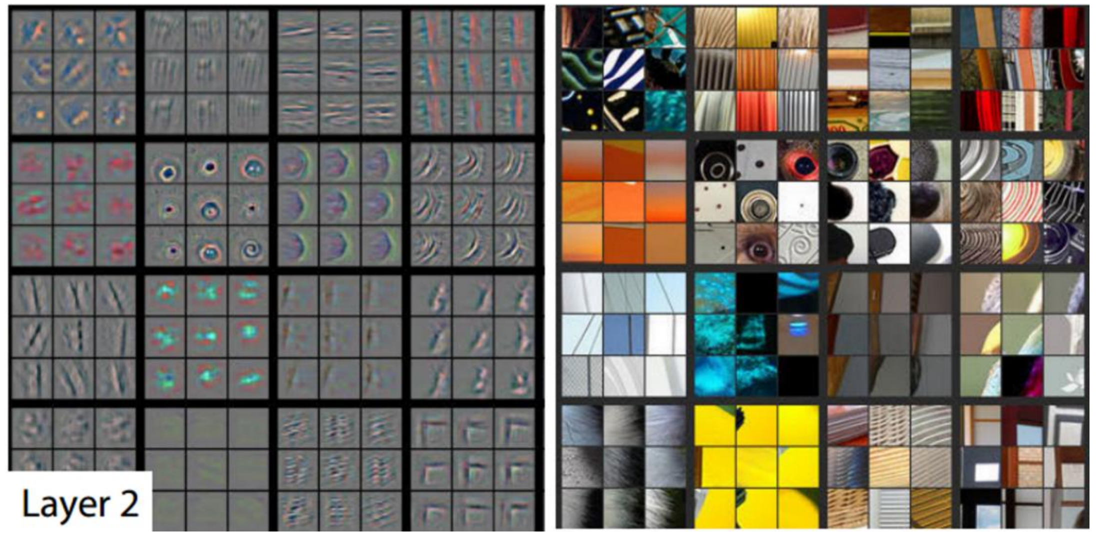
합성곱 연산 (input : 4x4, filter : 3x3)



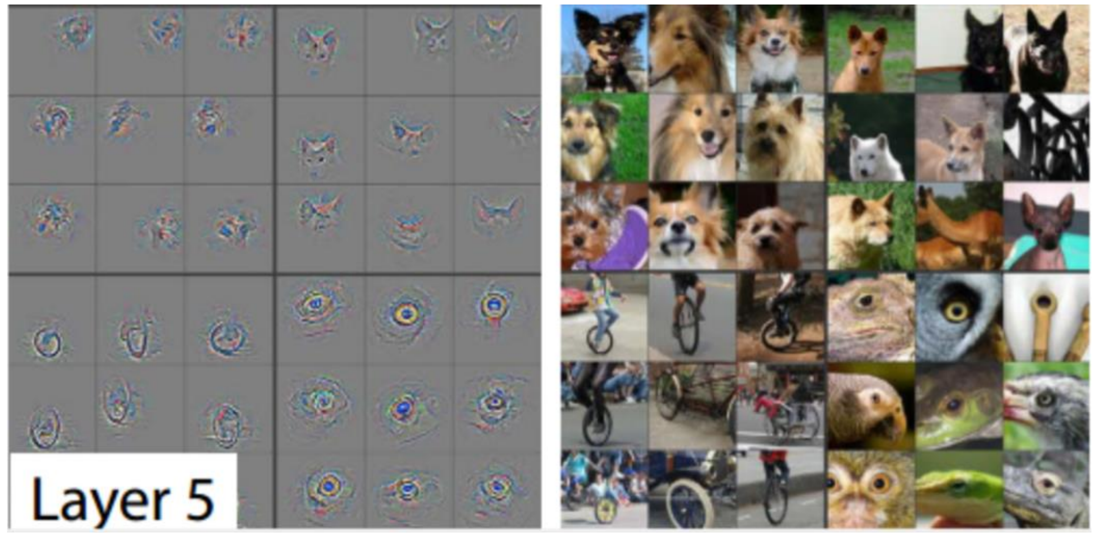
RGB img 1장의 합성곱 연산의 형태 (input : 3x4x4, filter : 3x3x3)



Layer가 깊어질수록 더 추상화된 feature map 을 추출



(edge(선), volume(덩어리) 를 인식)

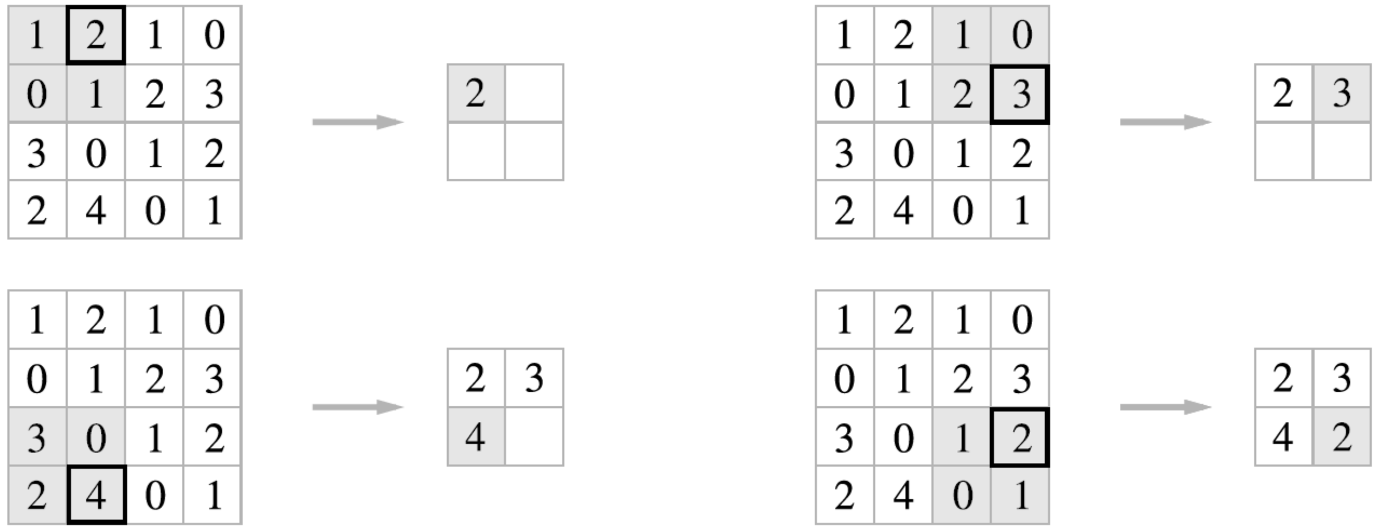


(추상화된 feature map을 인식)

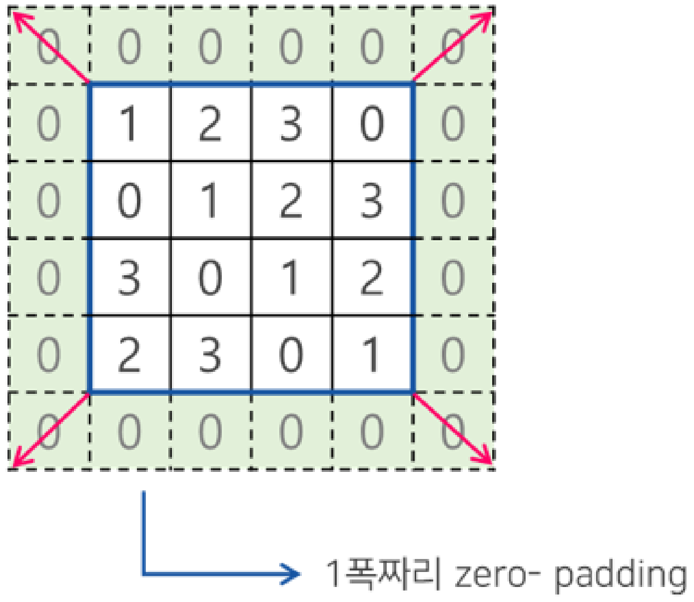
01. Layer

■ Pooling(sub sampling) : Max pooling, average pooling... etc

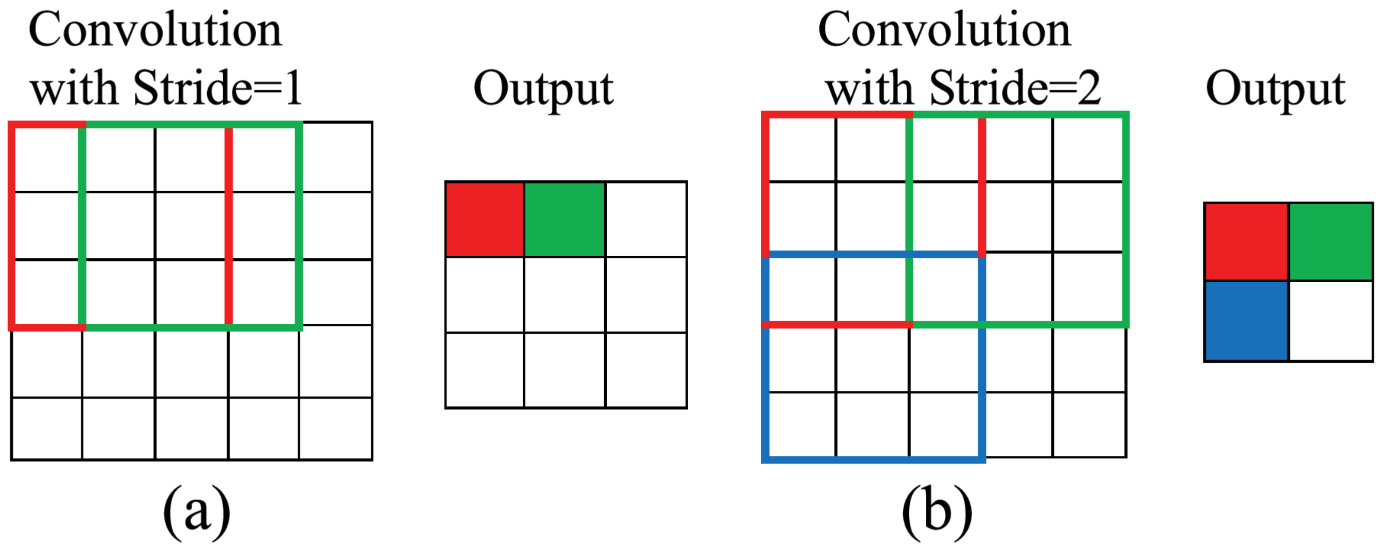
Max pooling (input : 4x4, window : 2x2, stride :2)



Zero padding : 1



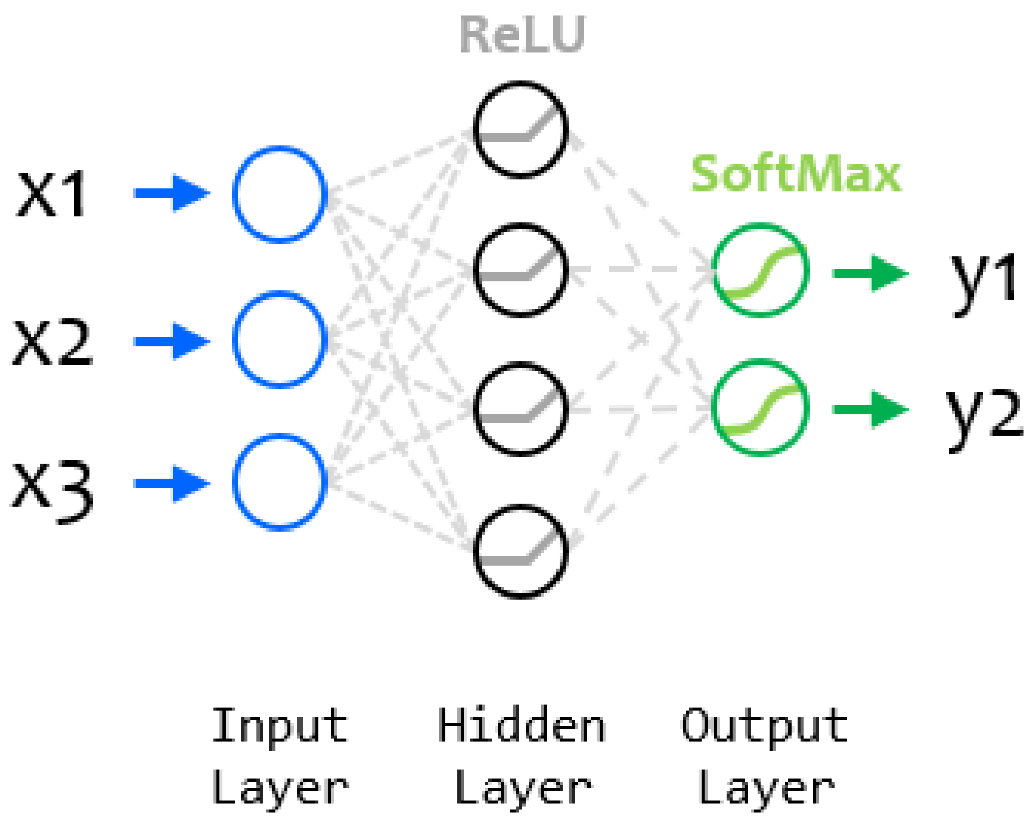
About Stride



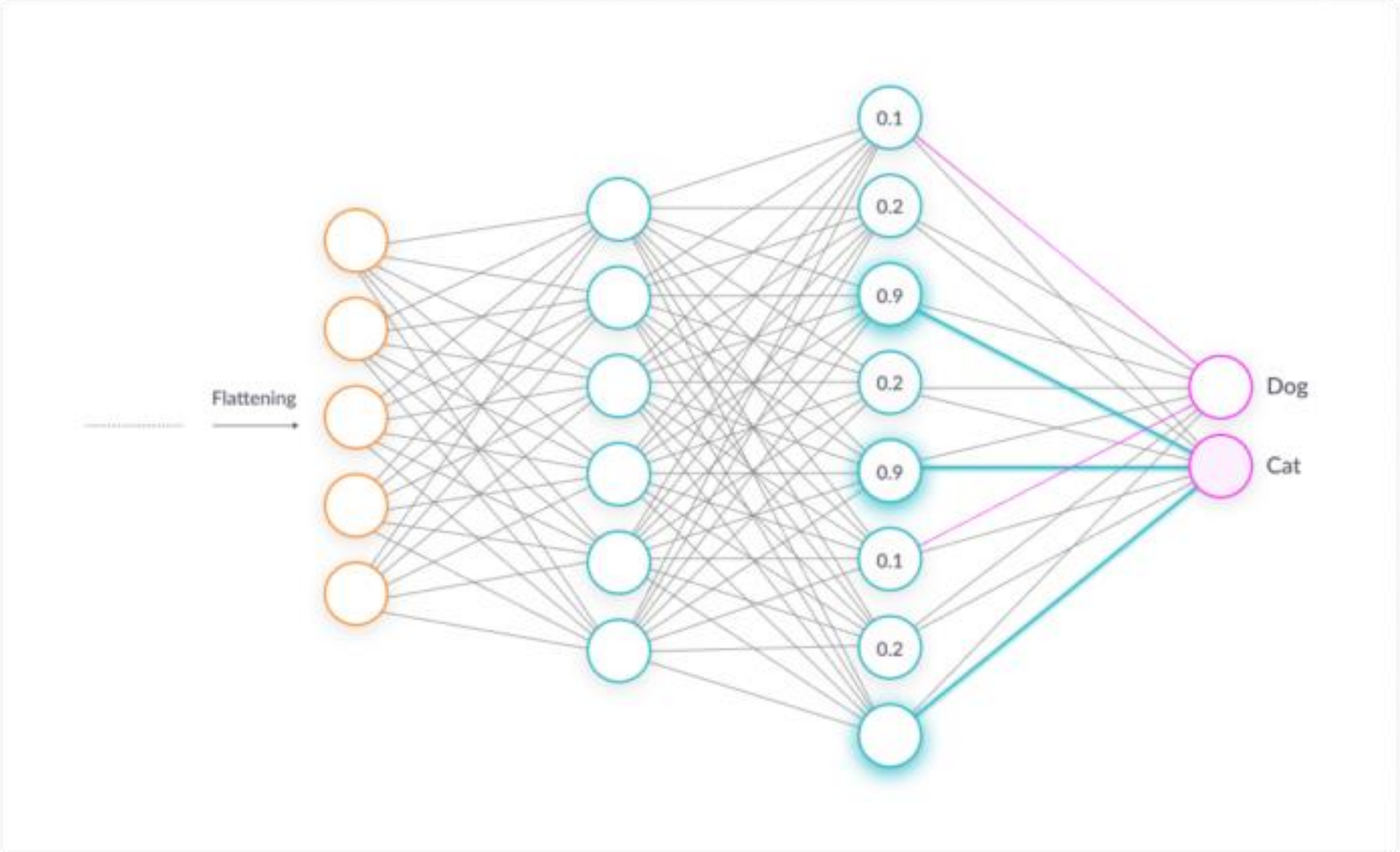
01. Layer

■ FC(fully connected) layer

Flatten(3), dense(4, activation=relu), dense(2, activation=softmax)

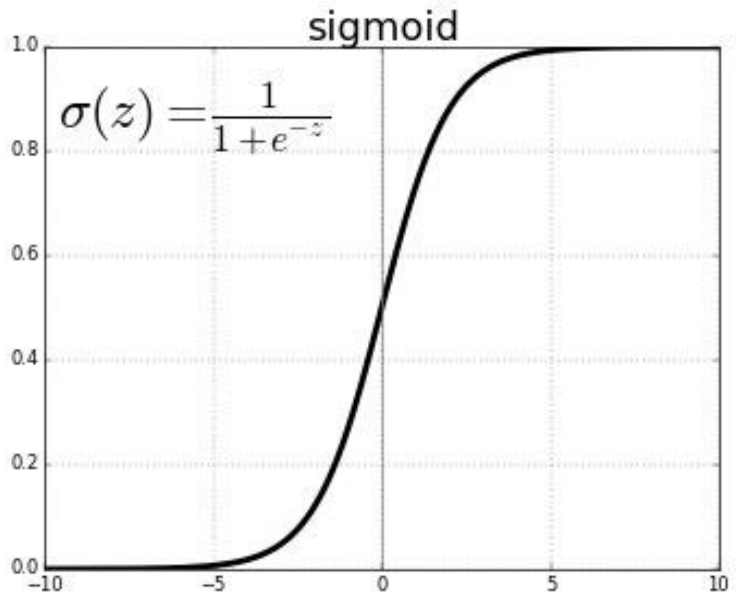


Larger version

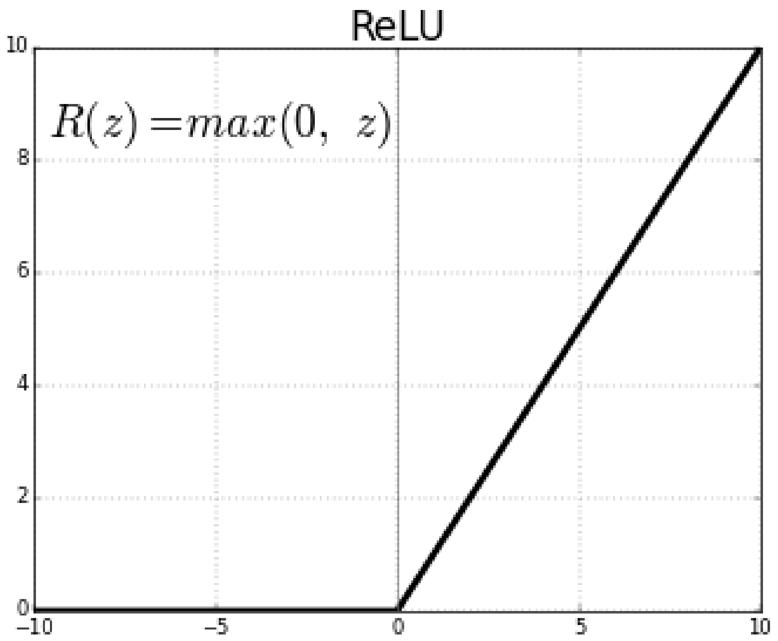


02. Activation

■ Sigmoid



■ Relu



■ Softmax

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

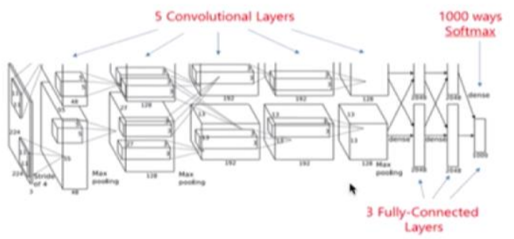
Sum(softmax(x)) = 1

03. Paper

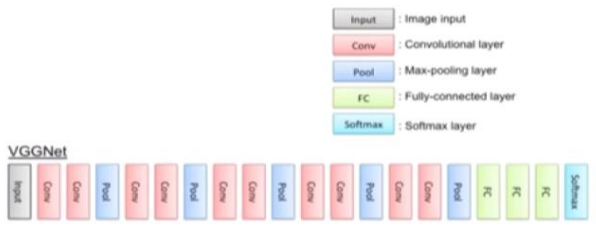
■ VGGnet : Very Deep Convolution Layer (https://arxiv.org/pdf/1409.1556.pdf)

CNN Architectures

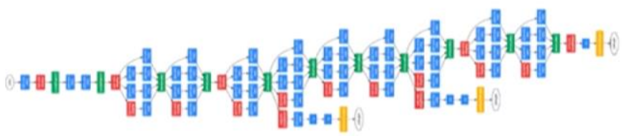
AlexNet



VGG



GoogLeNet



ResNet

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

03

VGGnet

1. VGGnet
2. Architecture

01. VGGnet

■ Paper : VGGnet

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

Table 3: **ConvNet performance at a single test scale.**

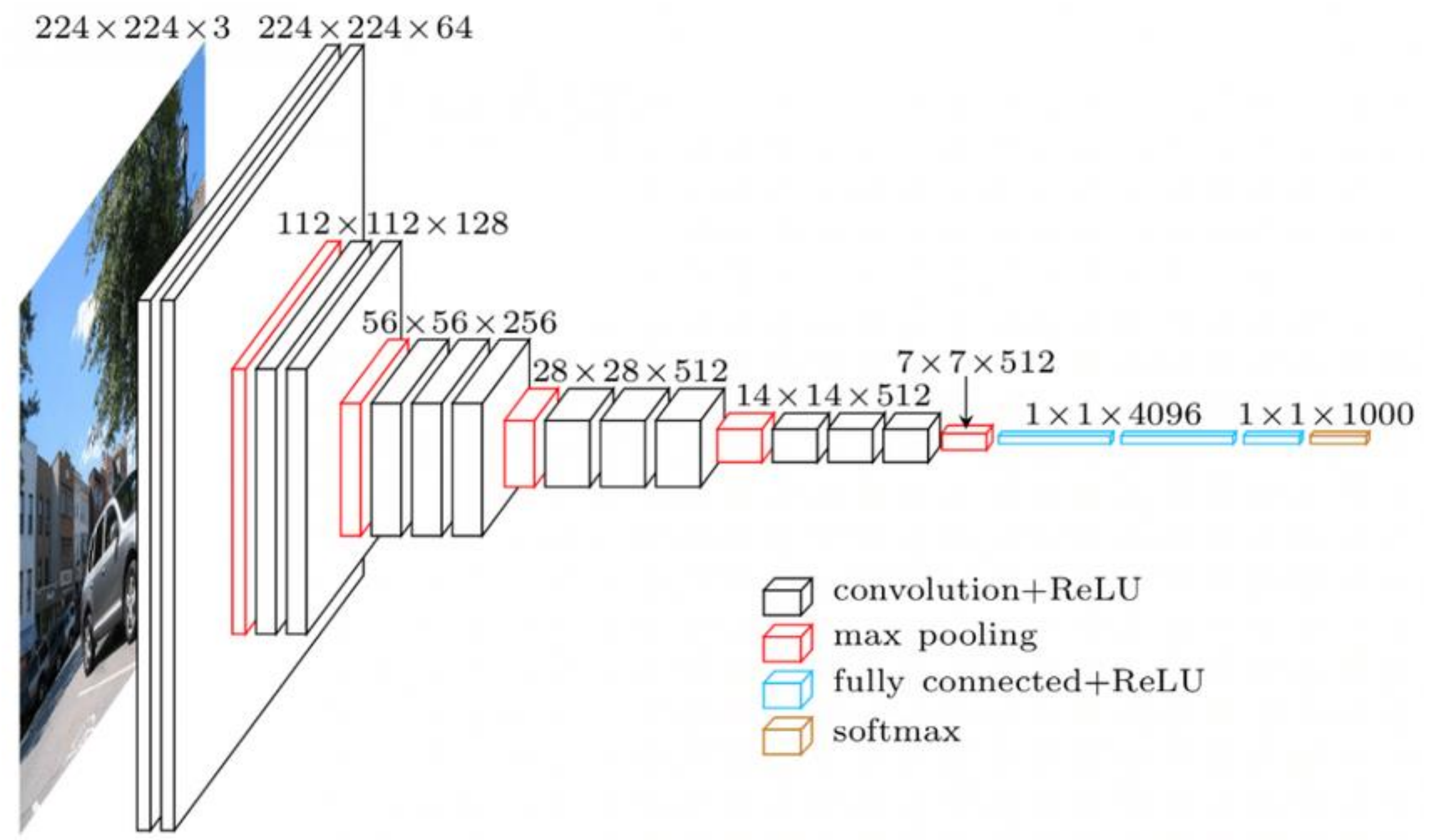
ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

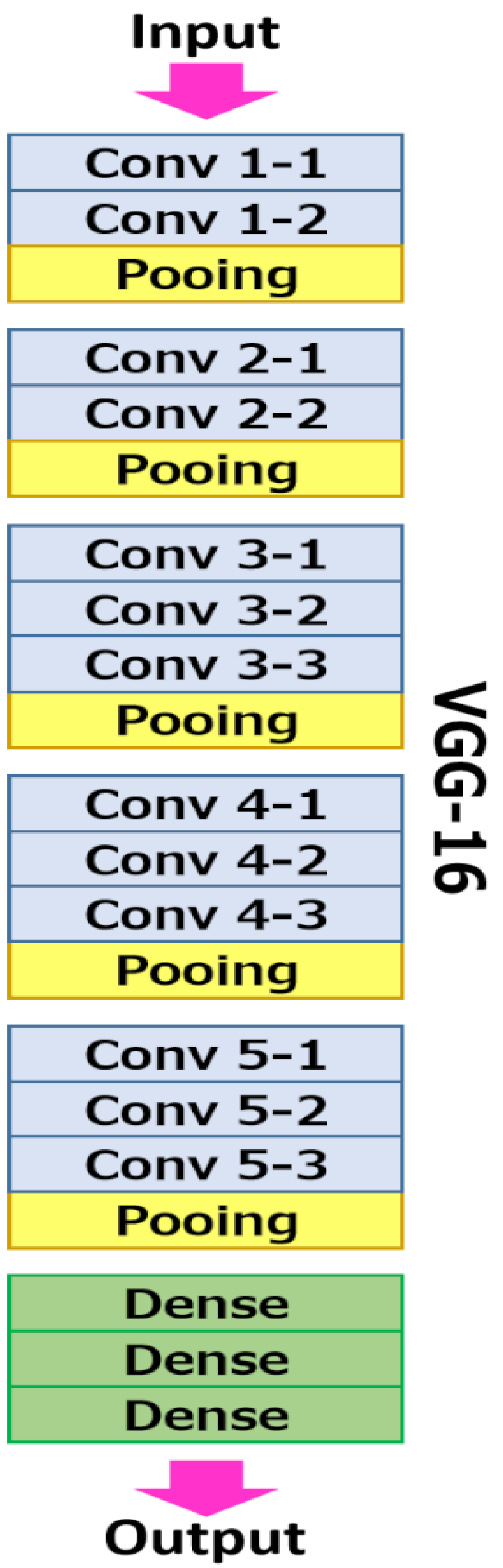
Table 5: **ConvNet evaluation techniques comparison.** In all experiments the training scale S was sampled from $[256; 512]$, and three test scales Q were considered: $\{256, 384, 512\}$.

ConvNet config. (Table 1)	Evaluation method	top-1 val. error (%)	top-5 val. error (%)
D	dense	24.8	7.5
	multi-crop	24.6	7.5
	multi-crop & dense	**24.4**	**7.2**
E	dense	24.8	7.5
	multi-crop	24.6	7.4
	multi-crop & dense	**24.4**	**7.1**

02. Architecture

■ VGG16





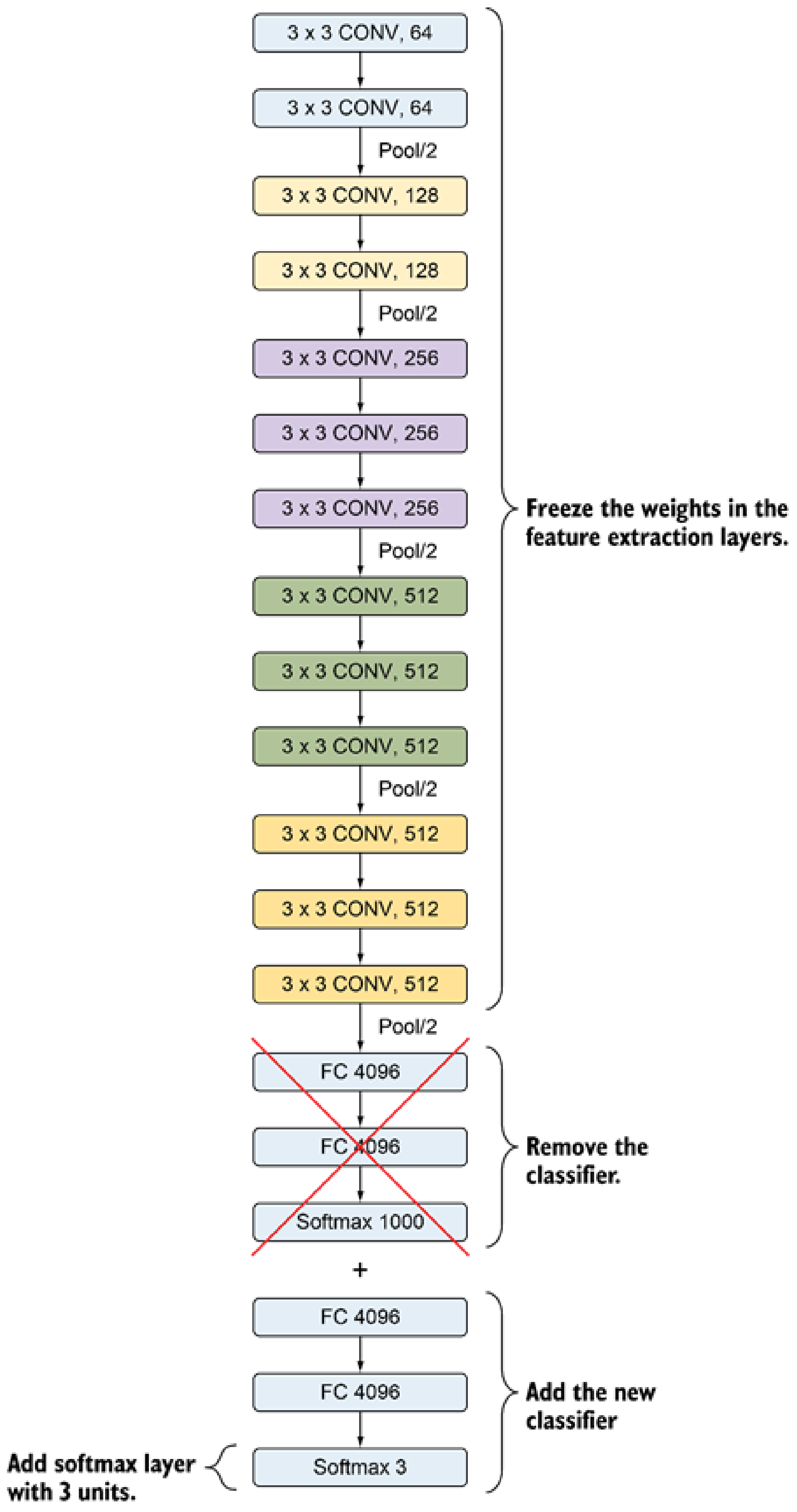
04

Transfer Learning

1. What is Transfer Learning?
2. Imgnet weights
3. Strategy

01. What is Transfer Learning?

■ Get Architecture & weights
+ training Dense
+ (high conv layer)



02. Imgnet weights

■ Initial weight : Imagenet (ILSVRC) IMAGENET

- 14,197,122 images, 21841 synsets indexed

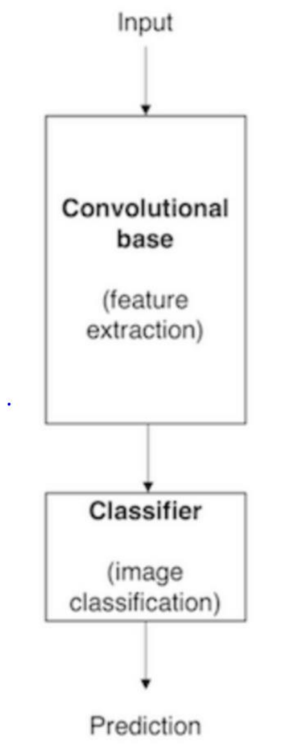
- 가중치 초기값으로 설정

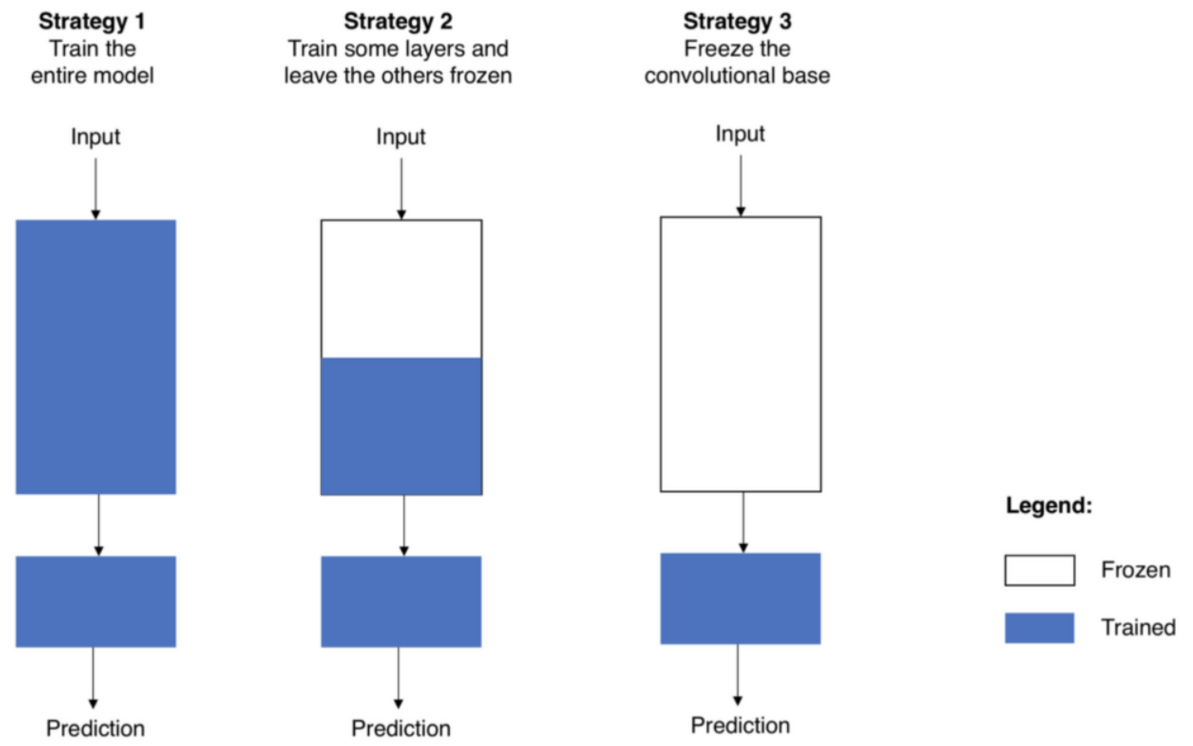
- Convolusion Layer 10 까지 가중치 동결

- custom dataset 의 feature map 추출용도로 사용

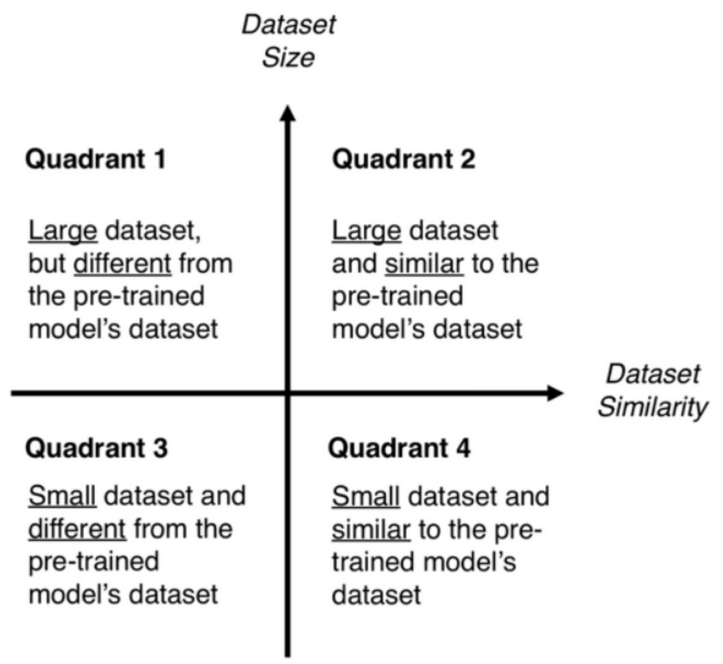
03. Strategy

■ Strategy

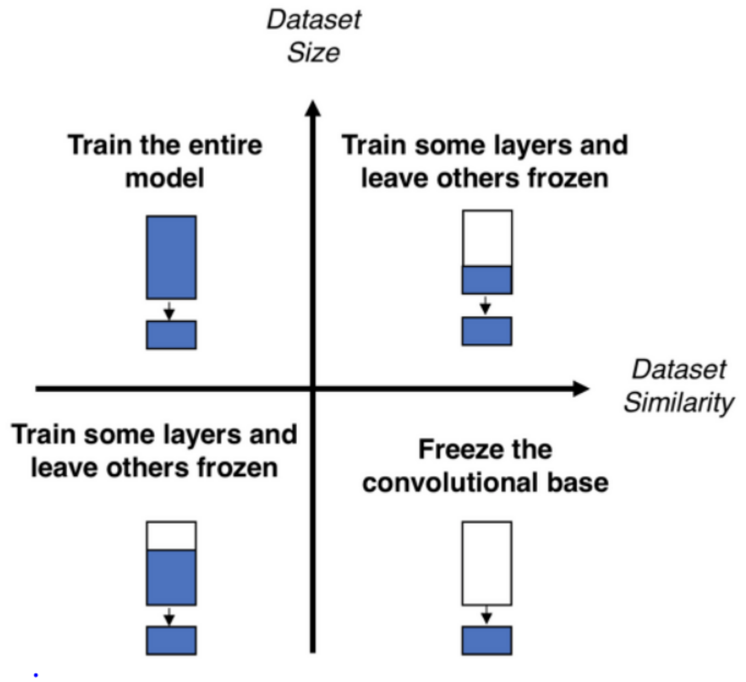




[그림 2] Fine-tuning의 세 가지 전략



[그림 3] 데이터크기-유사성 그래프



[그림 4] 각 상황에 따른 Fine-tuning 방법

05

Tuning

1. Hyper parameters
2. Overfitting regularizations
3. Graphs

01. Hyper parameters

■ Hyper parameters

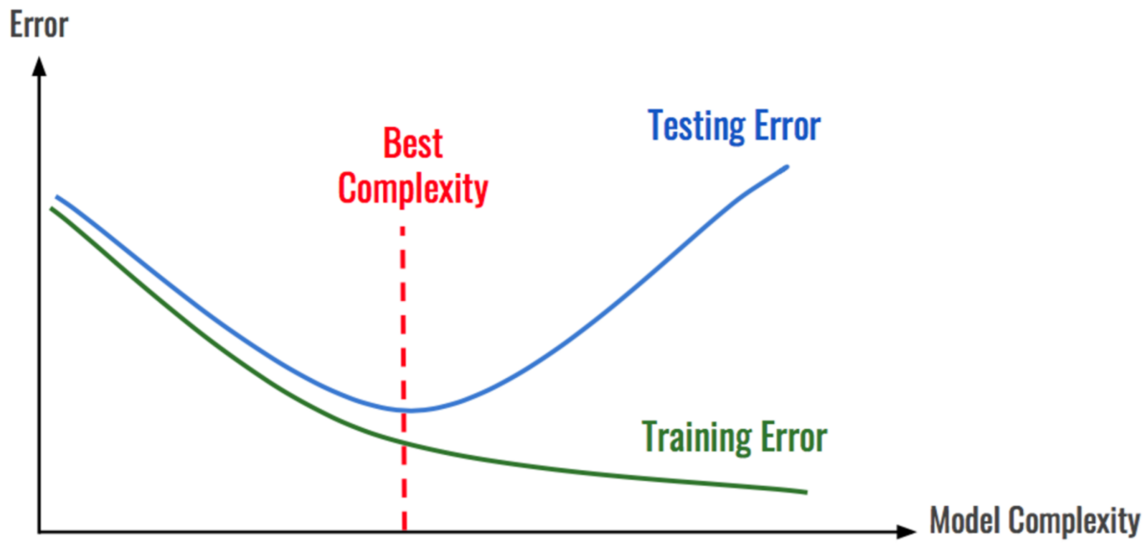
- optimizer
- metrics
- loss function
- batch size
- epochs
- img size

02. Overfitting regularizations

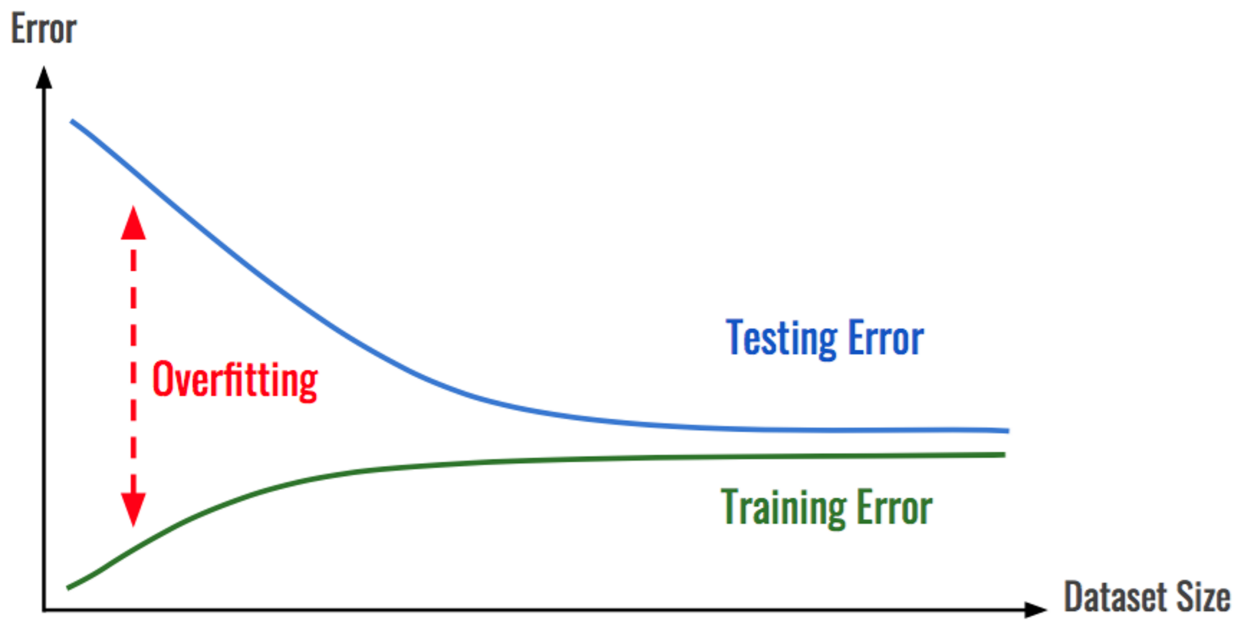
■ Overfitting regularizations

- Batch Normalization
- dropout
- 2l regularizer
- callback(early stopping)

[1] 모델복잡도에 따른 overfitting

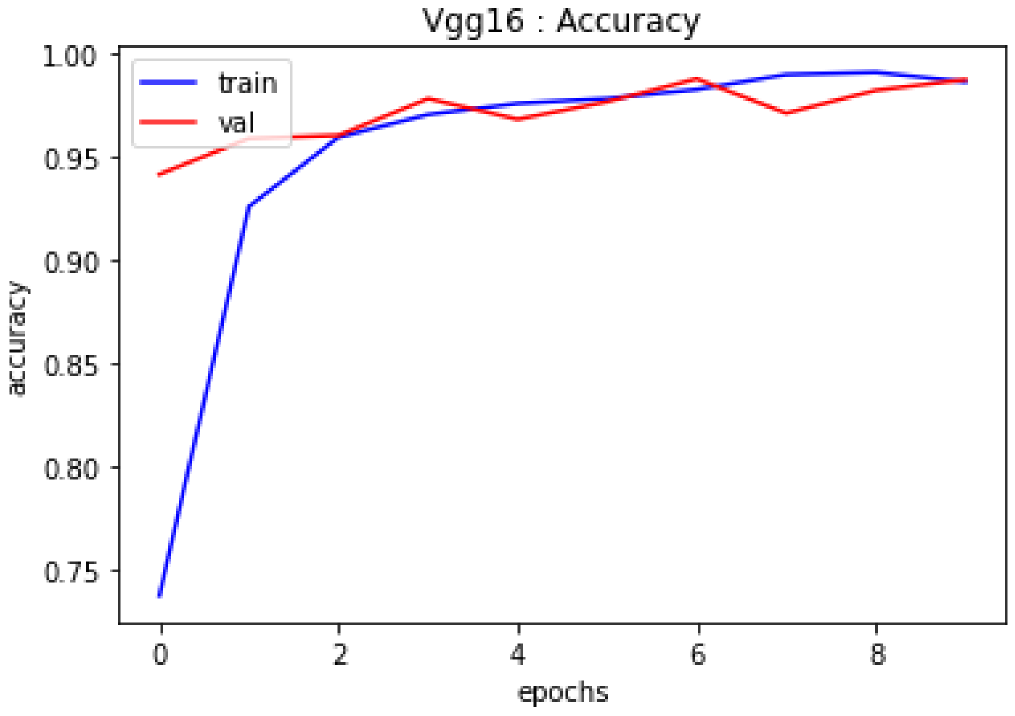


[2] Dataset size 에 따른 overfitting

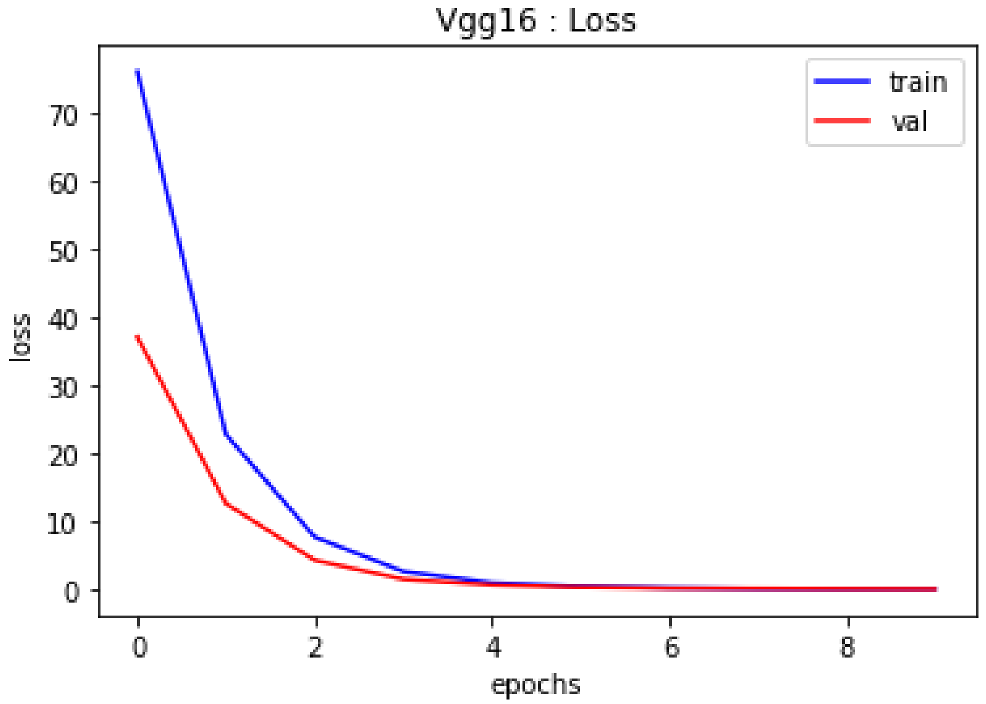


03. Graphs

■ After tuning



	train_Acc	val_Acc
0	0.737619	0.941852
1	0.926190	0.959259
2	0.959841	0.960741
3	0.970794	0.978519
4	0.976190	0.968518
5	0.978571	0.977037
6	0.983016	0.988148
7	0.990159	0.971482
8	0.991270	0.982593
9	0.986667	0.987778



	train_Loss	val_loss
0	76.151306	37.156326
1	22.837507	12.695949
2	7.801236	4.393091
3	2.742107	1.589292
4	1.068945	0.754511
5	0.528423	0.409886
6	0.314434	0.249031
7	0.188393	0.223279
8	0.145010	0.174336
9	0.147721	0.139329

Index : source for srcs

- p.8 src1, src2 : https://github.com/WegraLee
- p.9 src1 : https://github.com/WegraLee
- p.14 : https://arxiv.org/pdf/1409.1556.pdf
- p.17 : https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-6/v-8/106
- p.19 : https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

Thank you

Itwill 12th LKYJ Team