[5주차] Convolutional Neural Networks

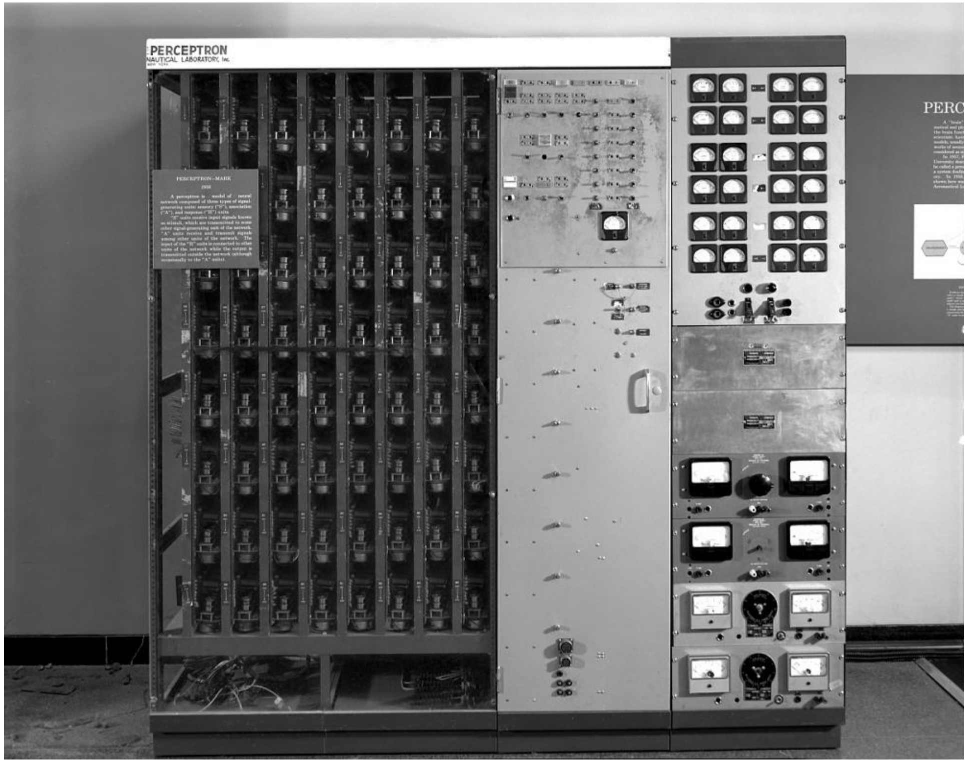
1기 강다연
1기 김지인

EURON

목차

1. ConvNet의 역사

- 1957: Frank Rosenblatt의 the Mark I Perceptron machine



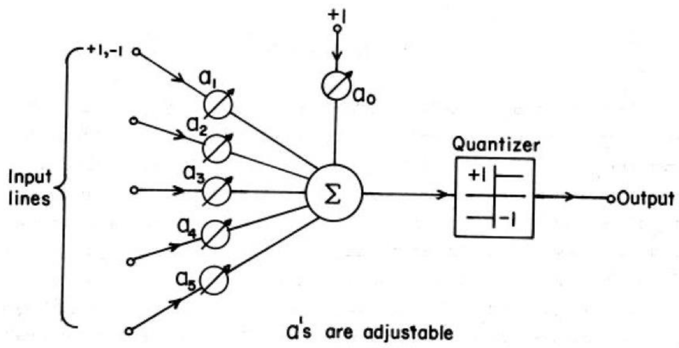
update rule:

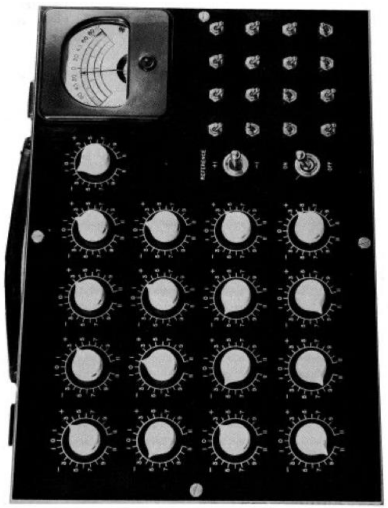
$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

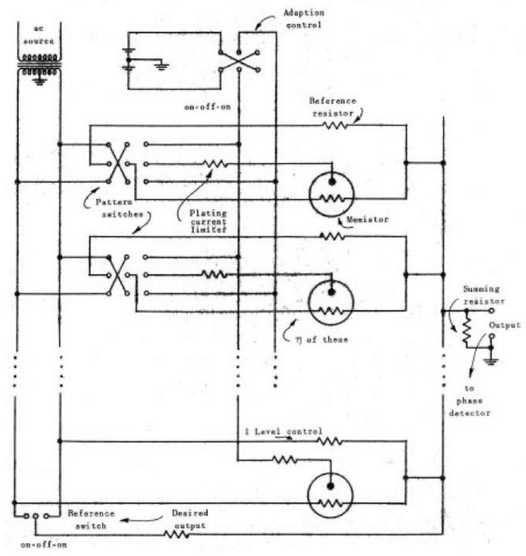
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

1. ConvNet의 역사

- 1960: Widrow and Hoff의 Adaline and Madaline

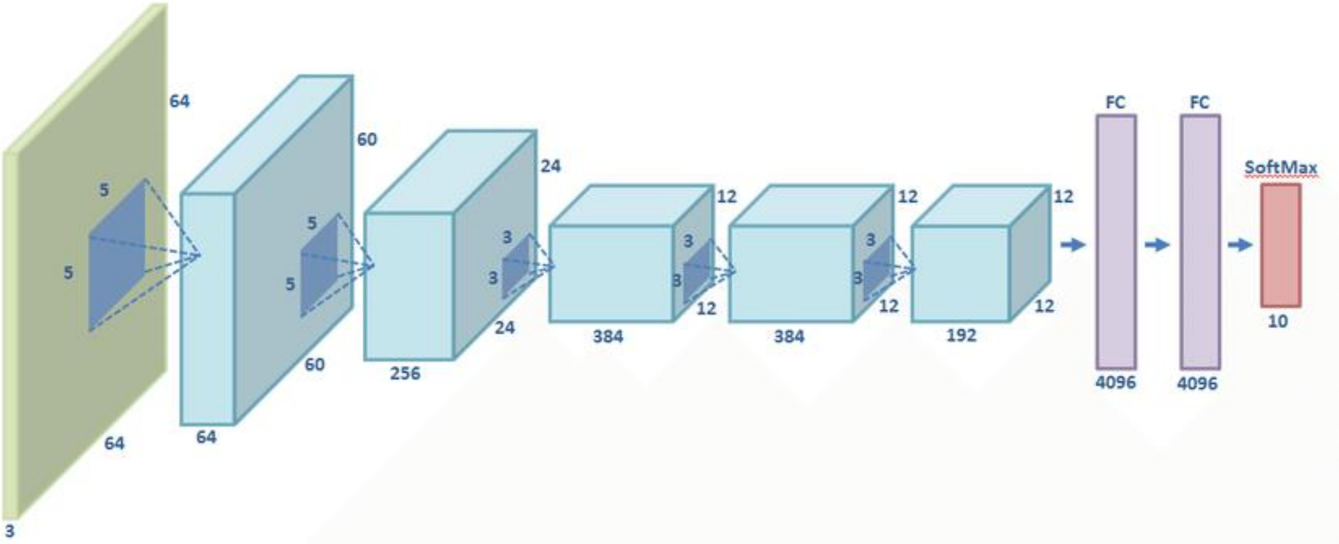






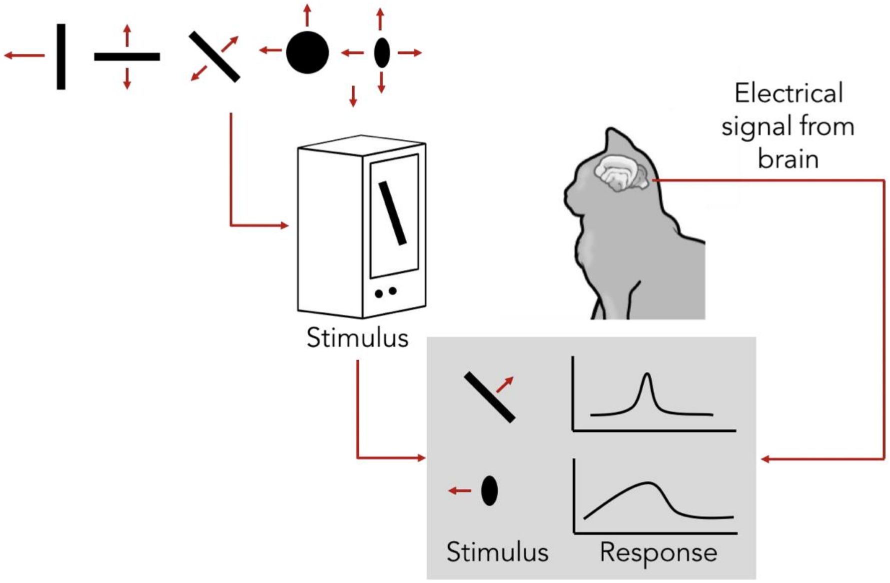
1. ConvNet의 역사

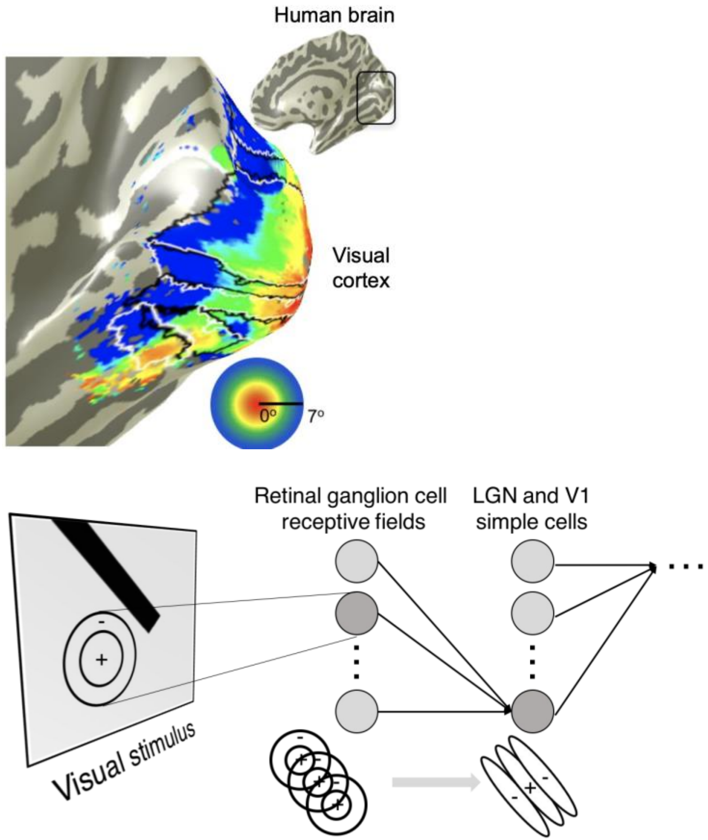
- 1986: back-propagation이 처음으로 소개됨
- 2006: 딥러닝 연구가 다시 활기를 띰
- 2012: Speech recognition & Image classification



1. ConvNet의 역사

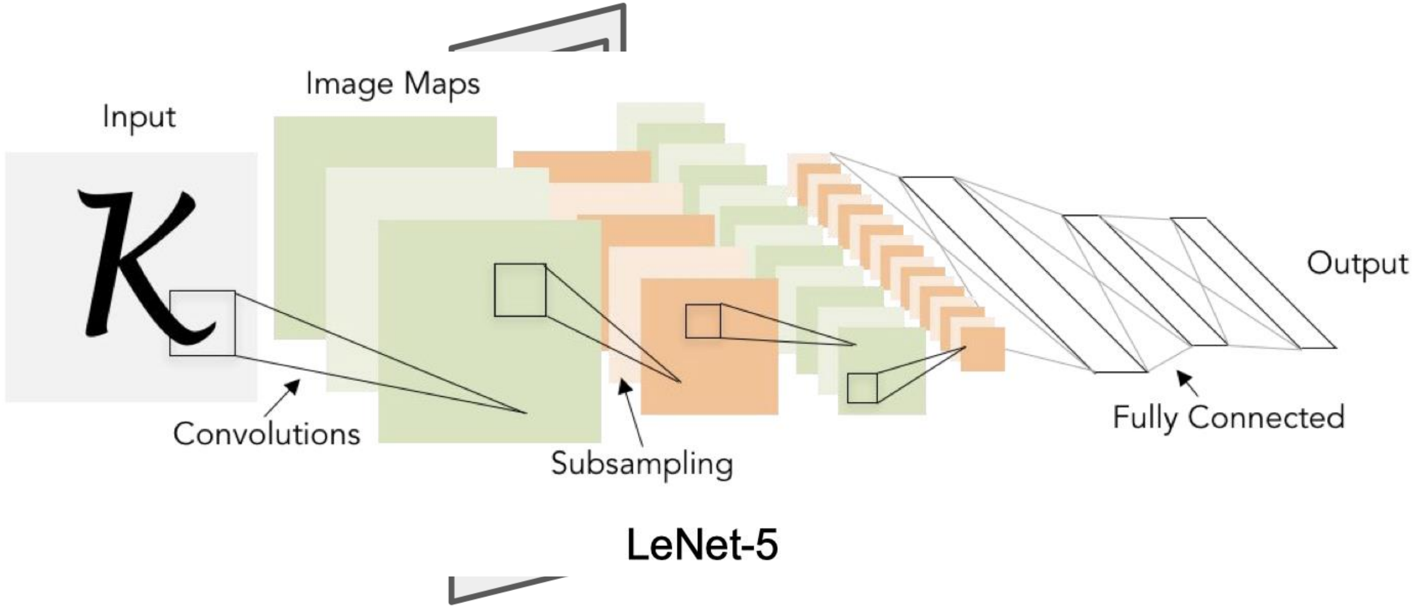
- Hubel & Wiesel





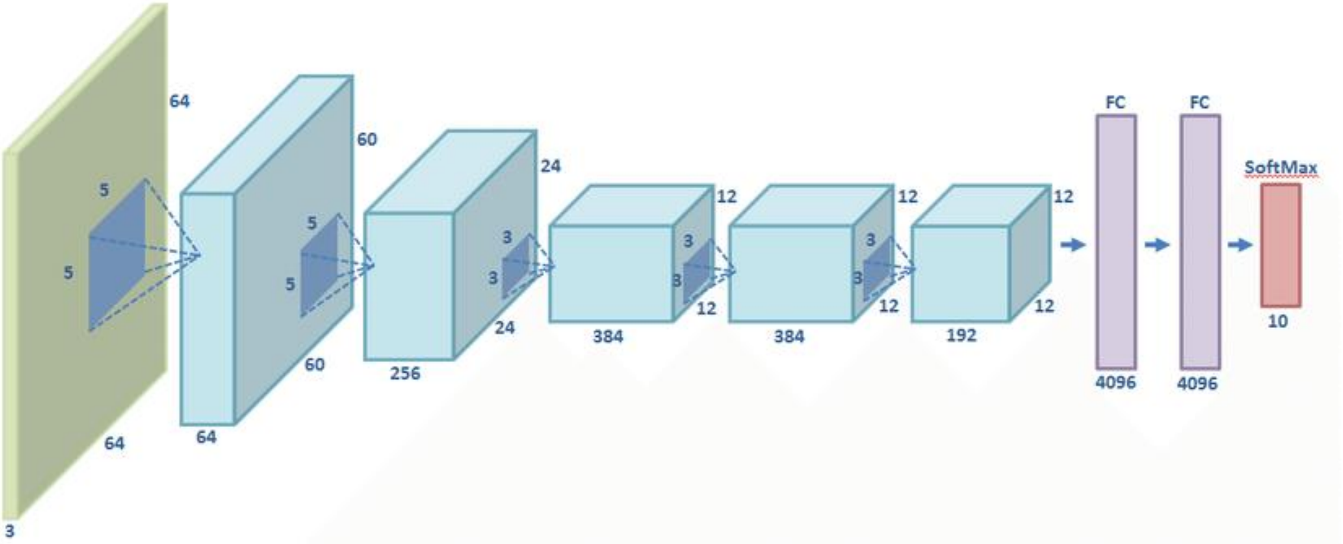
1. ConvNet의 역사

- 1980: Fukushima의 Neurocognition
- 1988: LeCun의 Gradient-based learning를 바탕으로 document recognition

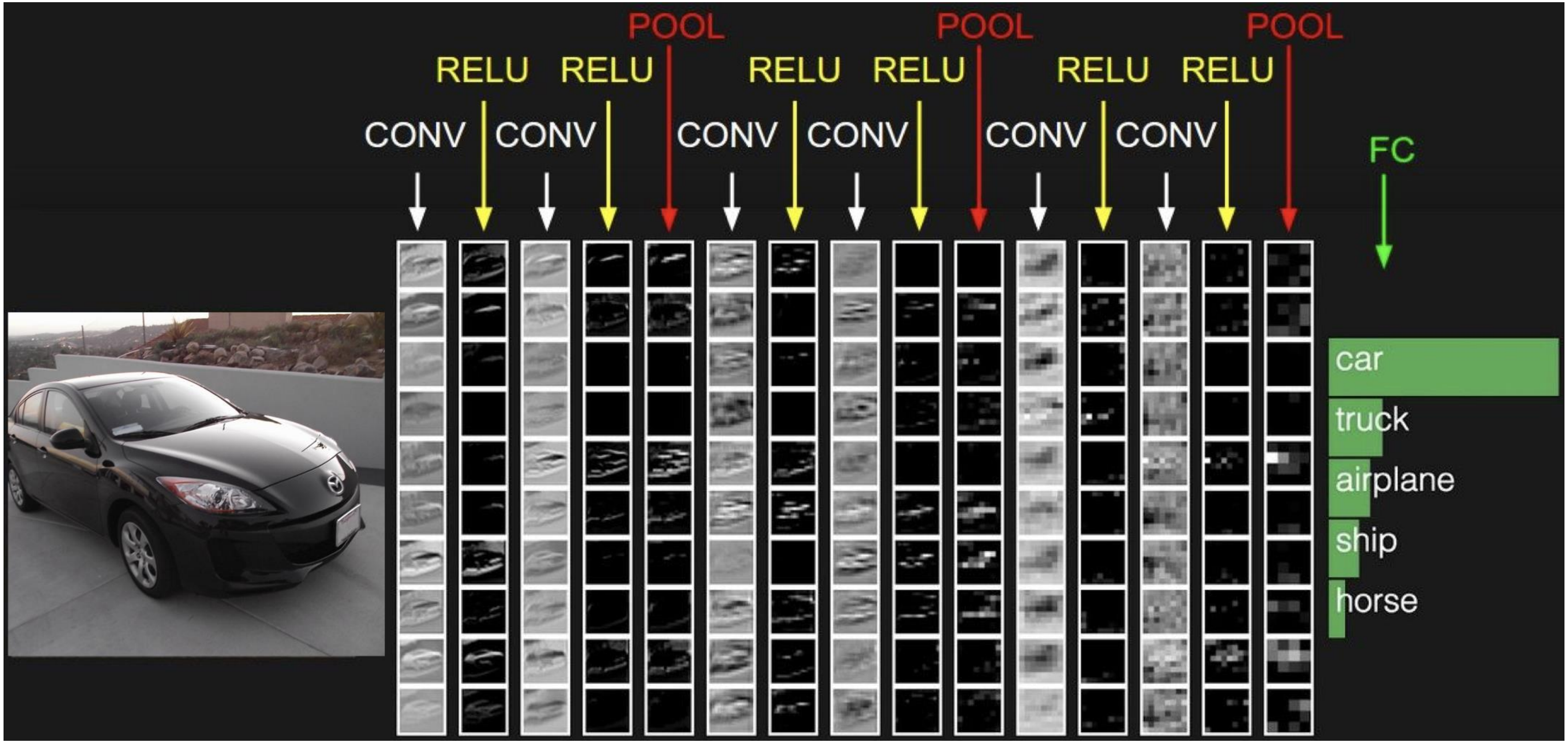


1. ConvNet의 역사

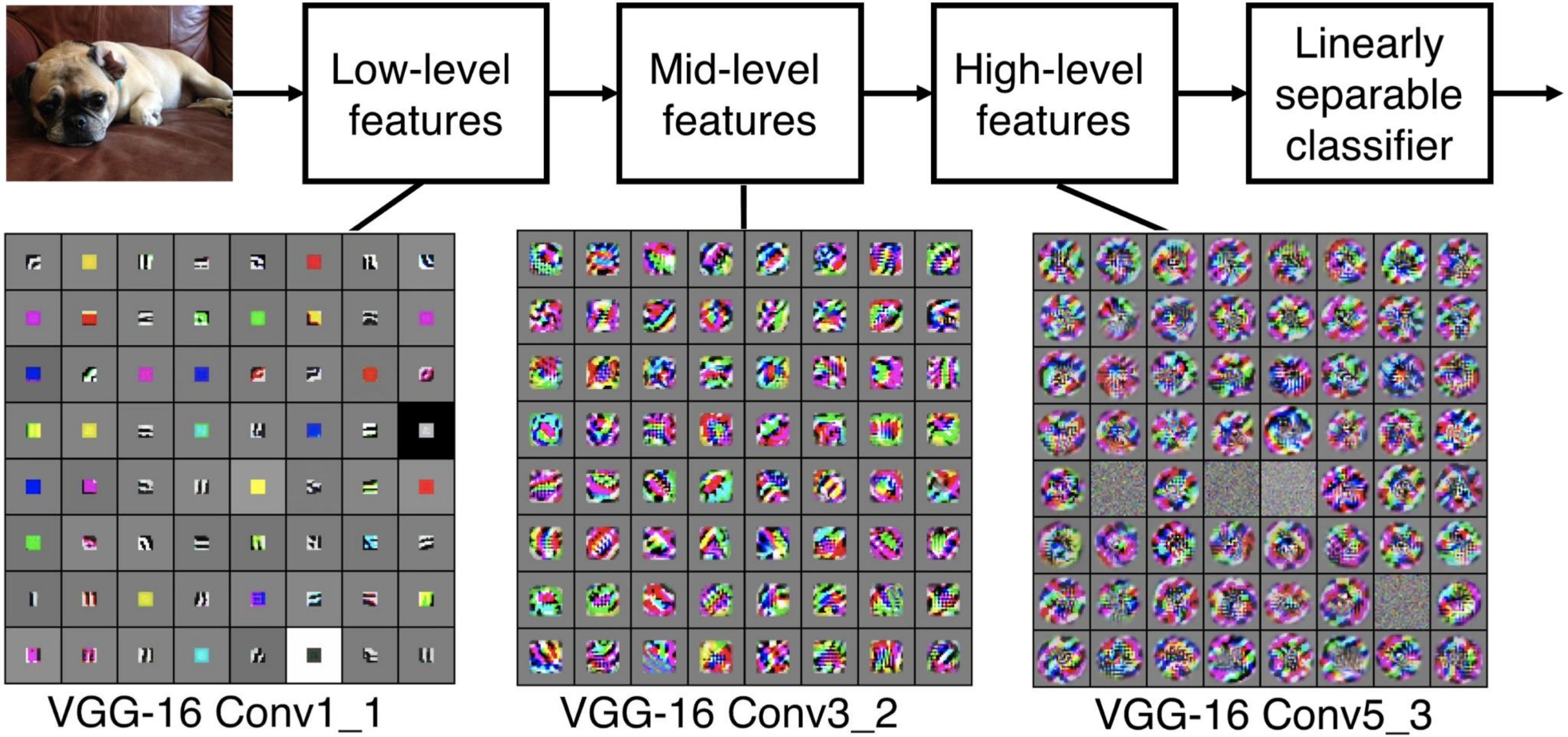
- 2012: Alec Krizhevsky의 Alexnet



Total ConvNet

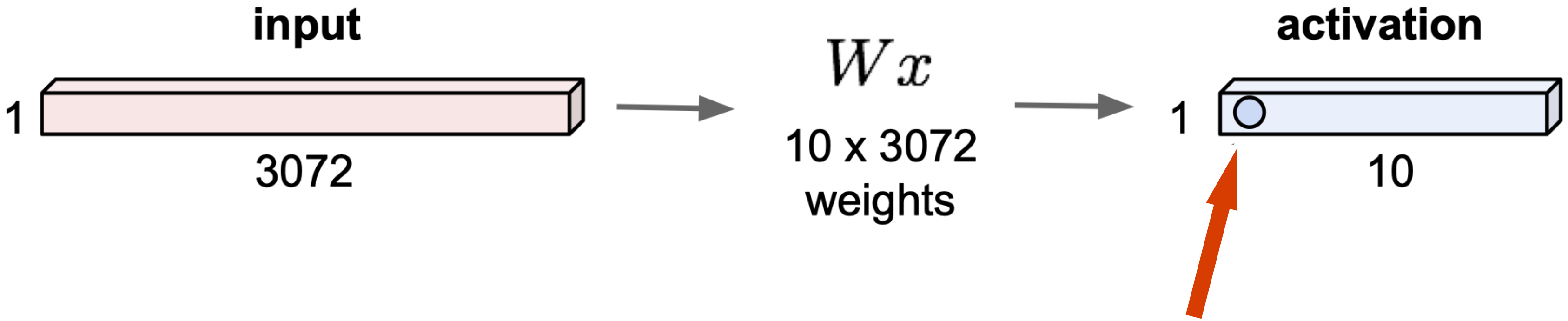


Total ConvNet



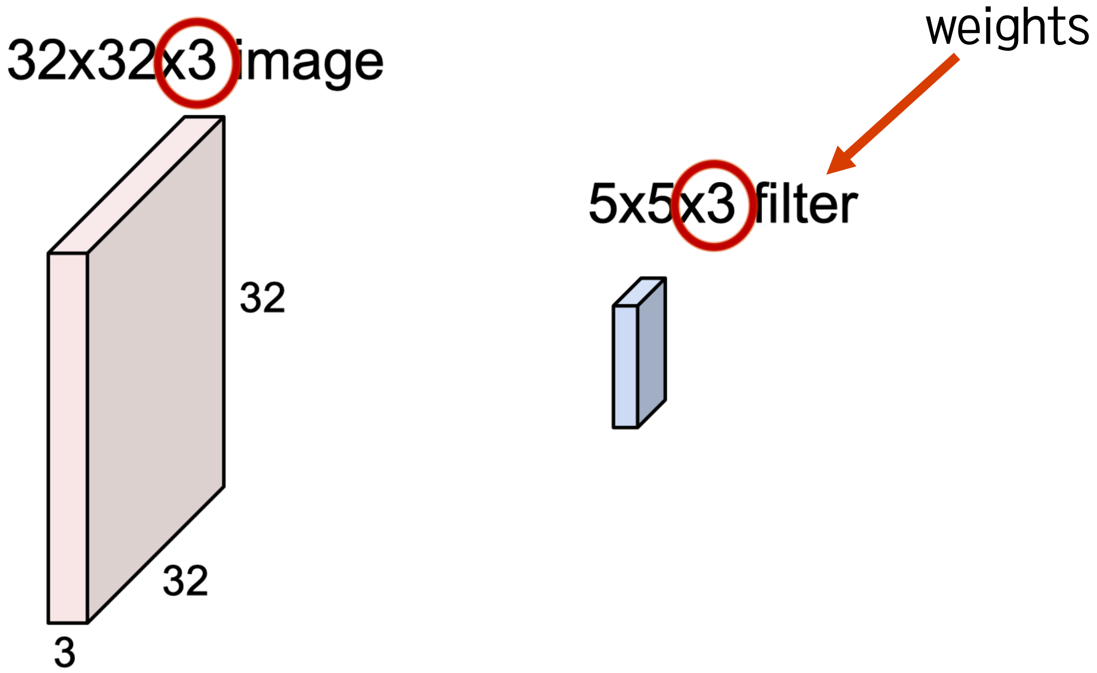
2. Fully Connected Layer

프로세스 결과를 취하여 이미지를 정의된 라벨로 분류하는데 사용



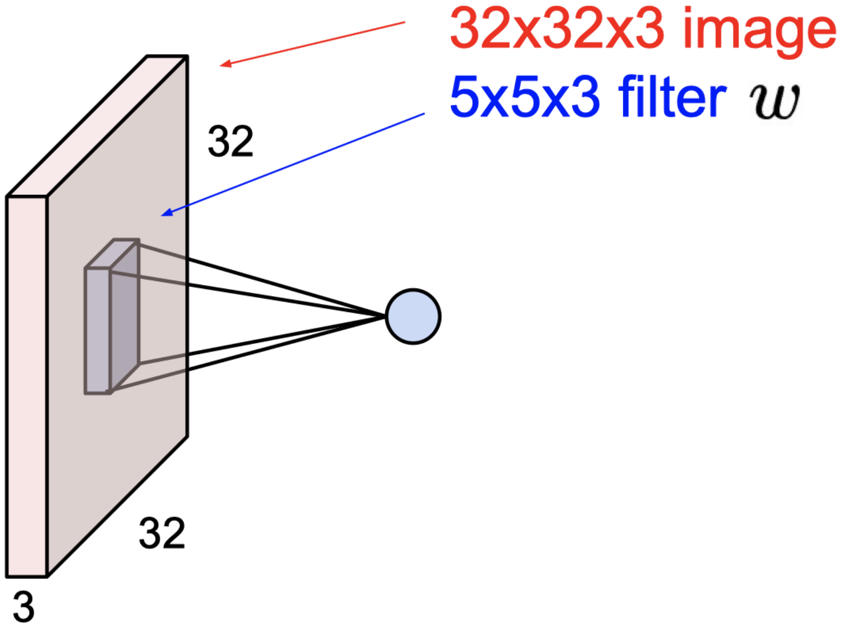
3. Convolution Layer

slide over the image spatially and
compute dot products at every spatial location



3. Convolution Layer

slide over the image spatially and
compute dot products at every spatial location



3. Convolution Layer

Input Image

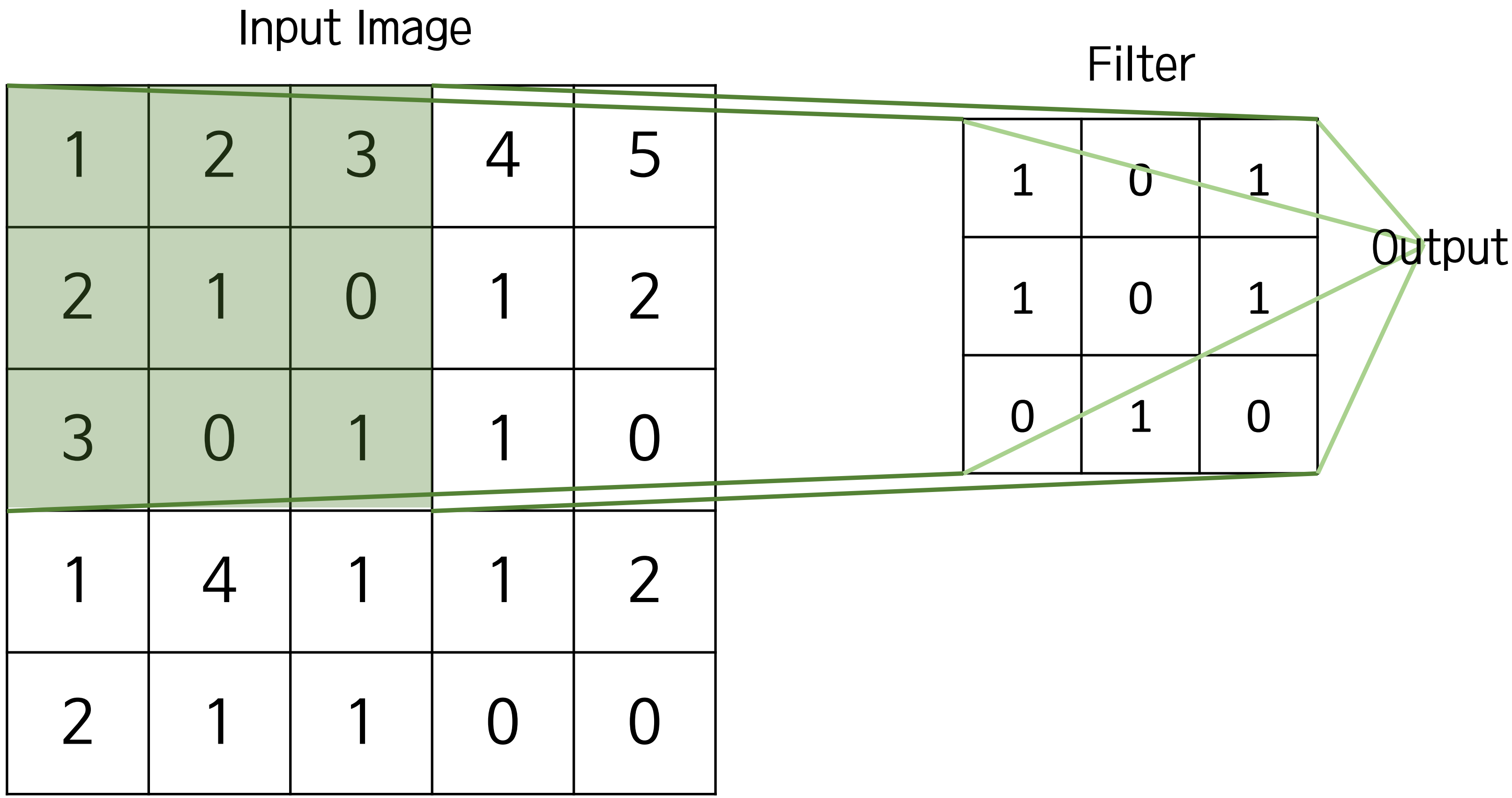
1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Parameter 개수 :
3x3 (dot product 차원) + bias
= 9 + bias

3. Convolution Layer



3. Convolution Layer

Input Image

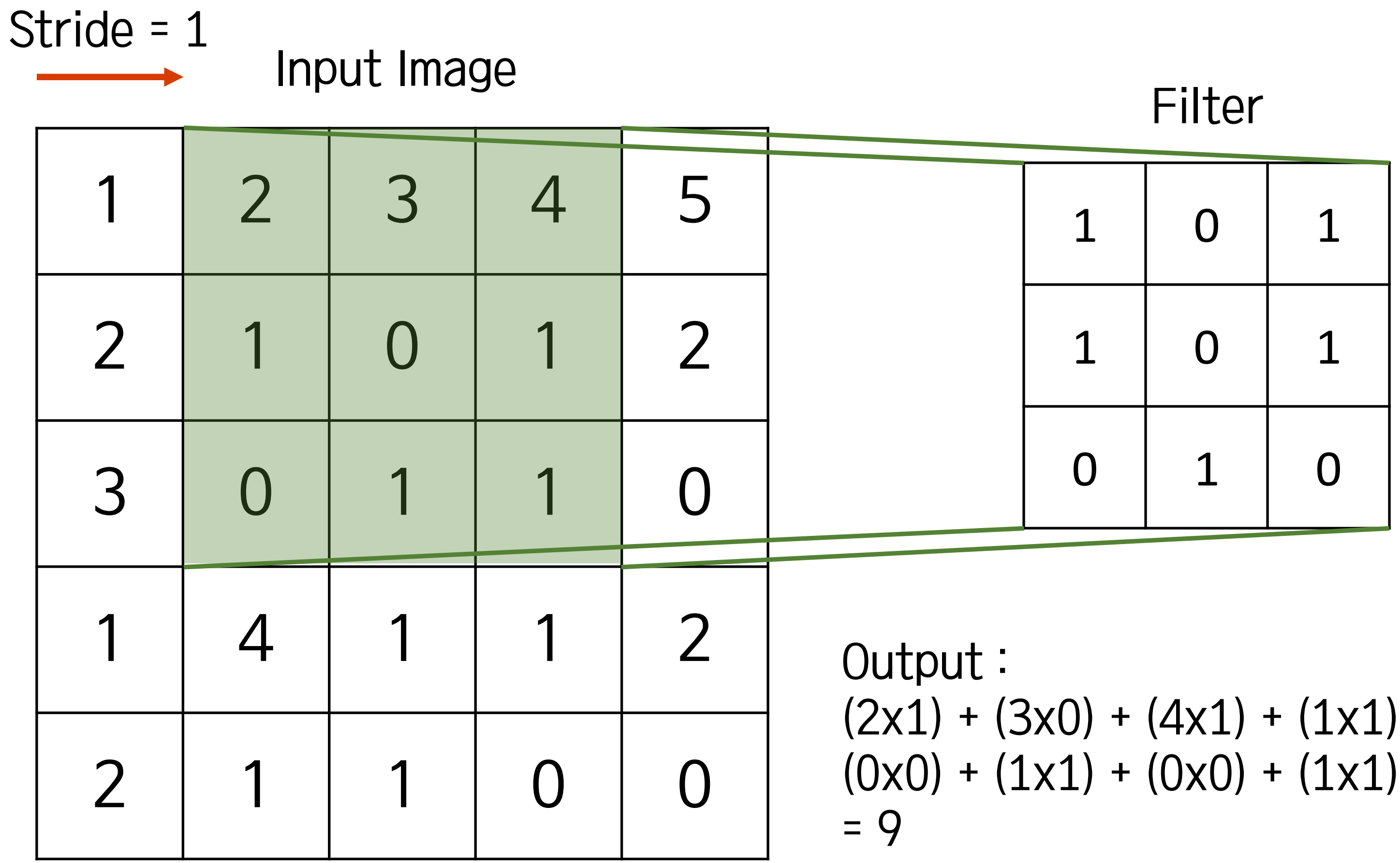
1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

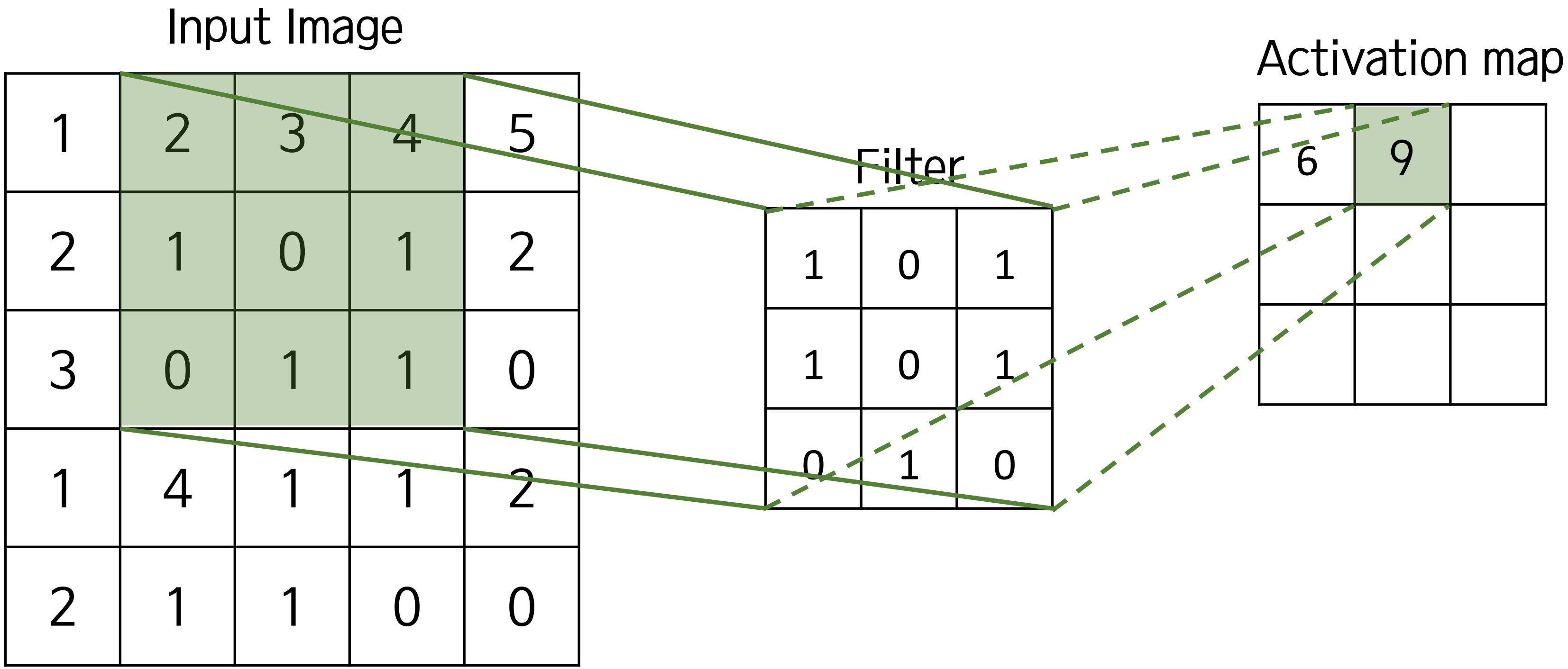
Output :
(1x1) + (2x0) + (3x1) + (2x1) +
(1x0) + (0x1) + (3x0) + (0x1) + (1x0)
= 6

3. Convolution Layer



Output :
(2x1) + (3x0) + (4x1) + (1x1) +
(0x0) + (1x1) + (0x0) + (1x1) + (1x0)
= 9

3. Convolution Layer



3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	9	11

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	9	11
10		

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	9	11
10	4	

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	9	11
10	4	4
7	7	4

3. Convolution Layer

Stride가 2이면?

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	

3. Convolution Layer

Stride = 2

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	11

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	11
7	

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Filter

1	0	1
1	0	1
0	1	0

Activation map

6	11
7	4

6	9	11
10	4	4
7	7	4

Activation map의 size는 어떻게 계산?

3. Convolution Layer

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

N

N

F

F

Filter

1	0	1
1	0	1
0	1	0

F

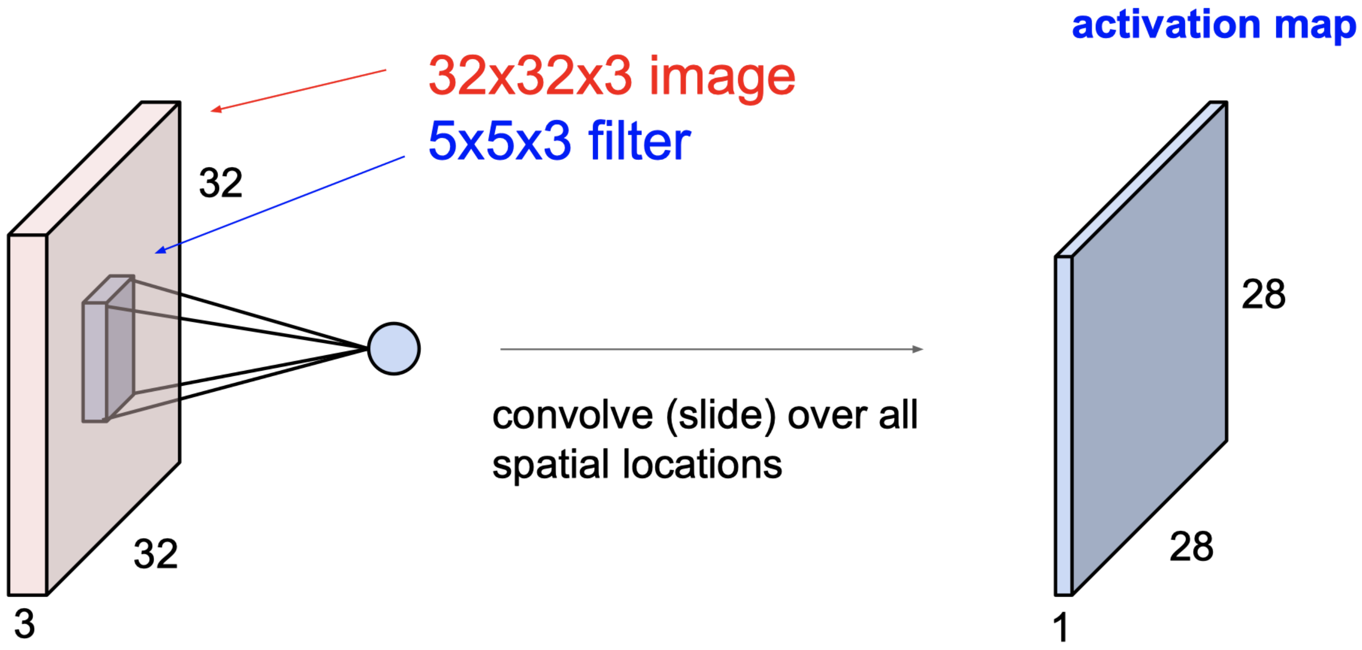
F

Output size:

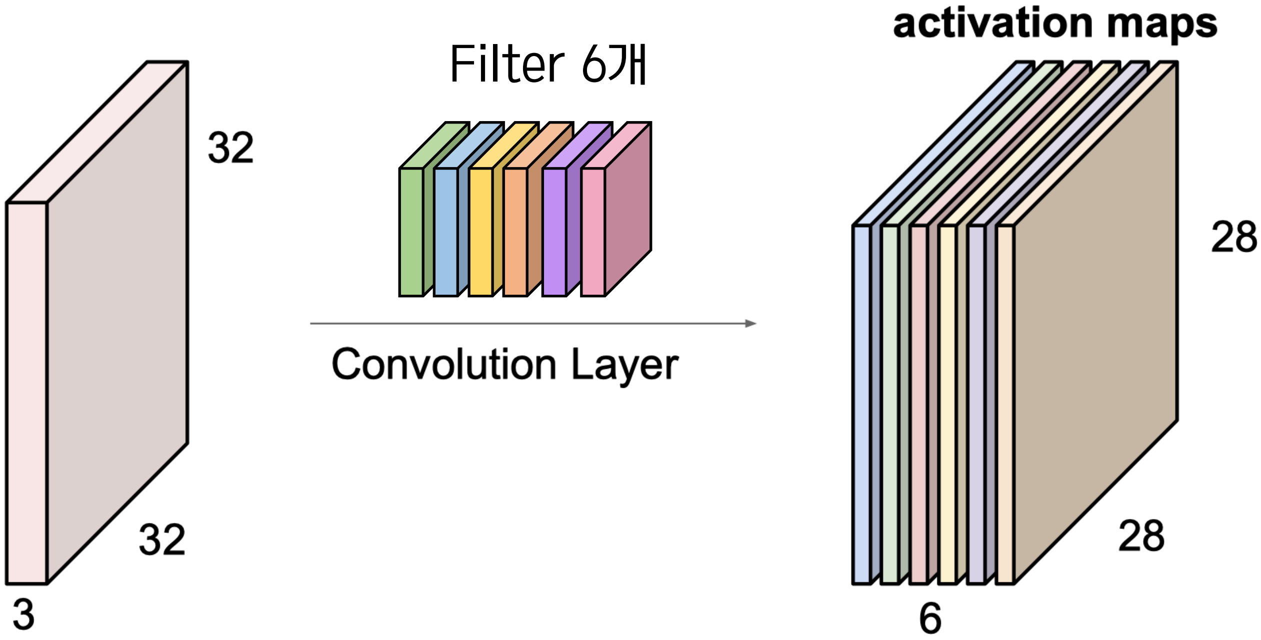
(N – F) / stride + 1

→ 2x2

3. Convolution Layer



3. Convolution Layer



3. Convolution Layer

Q) Input volume : 32x32x3

4 7x7 filters with stride 1

output activation map의 size?

A)activation map 1개의 size : (32 – 7) / 1 + 1 = 26

➔ 26x26x4

3. Convolution Layer

만약 filter가 커진다면?

Input Image

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

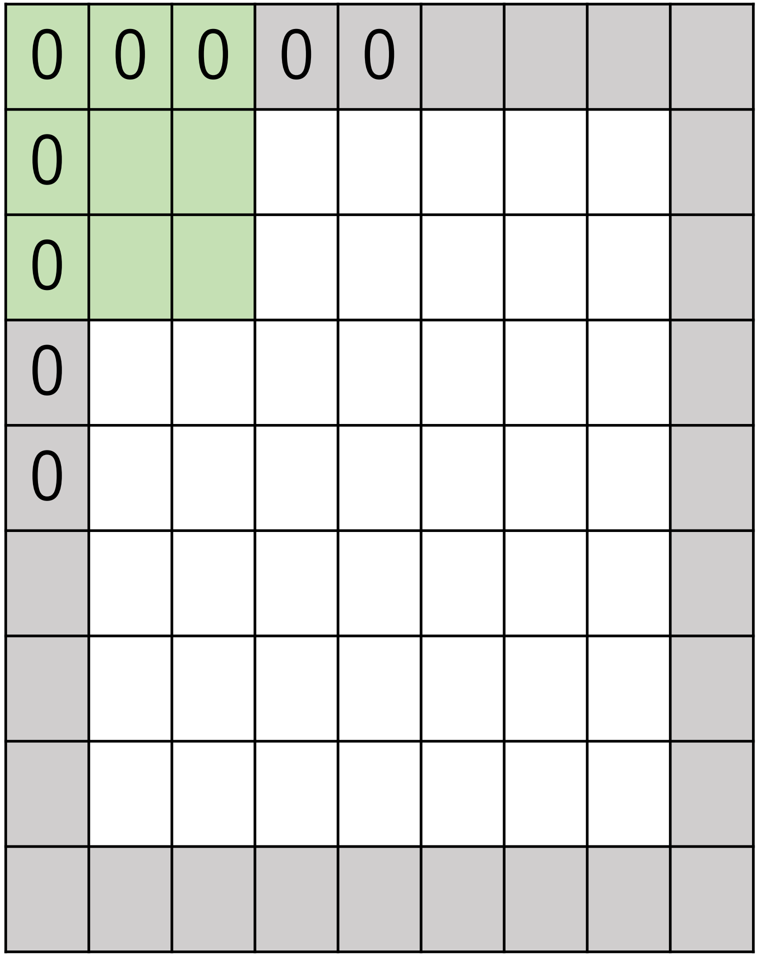
Filter 3x3 → 9번
Filter 4x4 → 4번

➔Filter가 커지면 activation map의 크기는 빠르게 작아진다

➔layer를 많이 못쓰게 된다

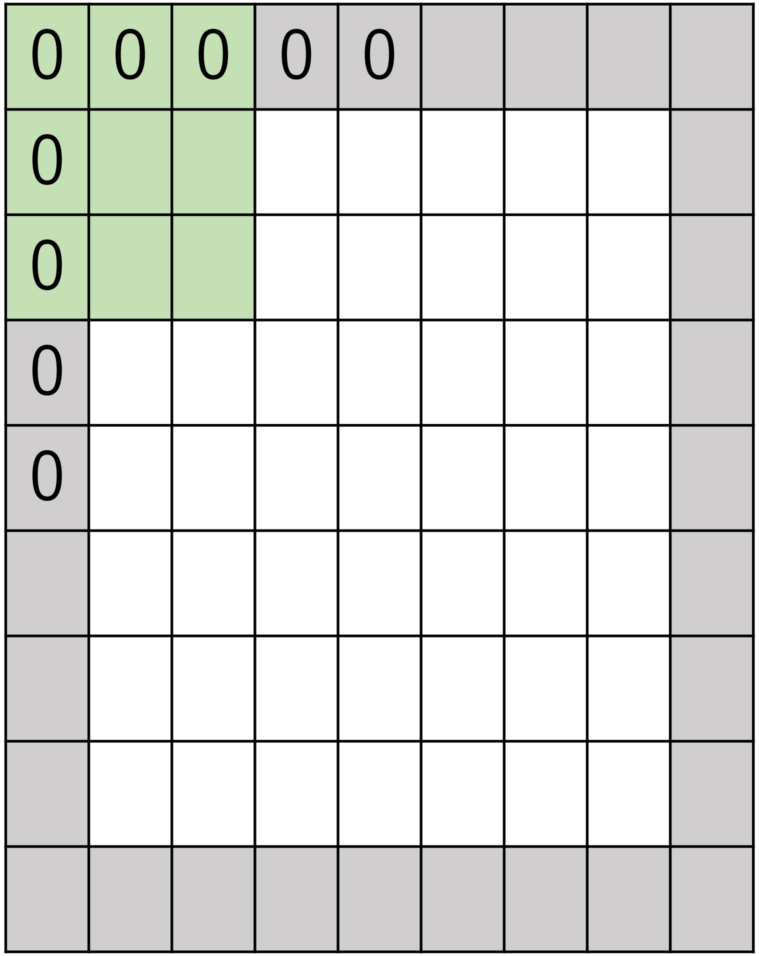
➔Padding

Padding



Output image size 유지

Padding



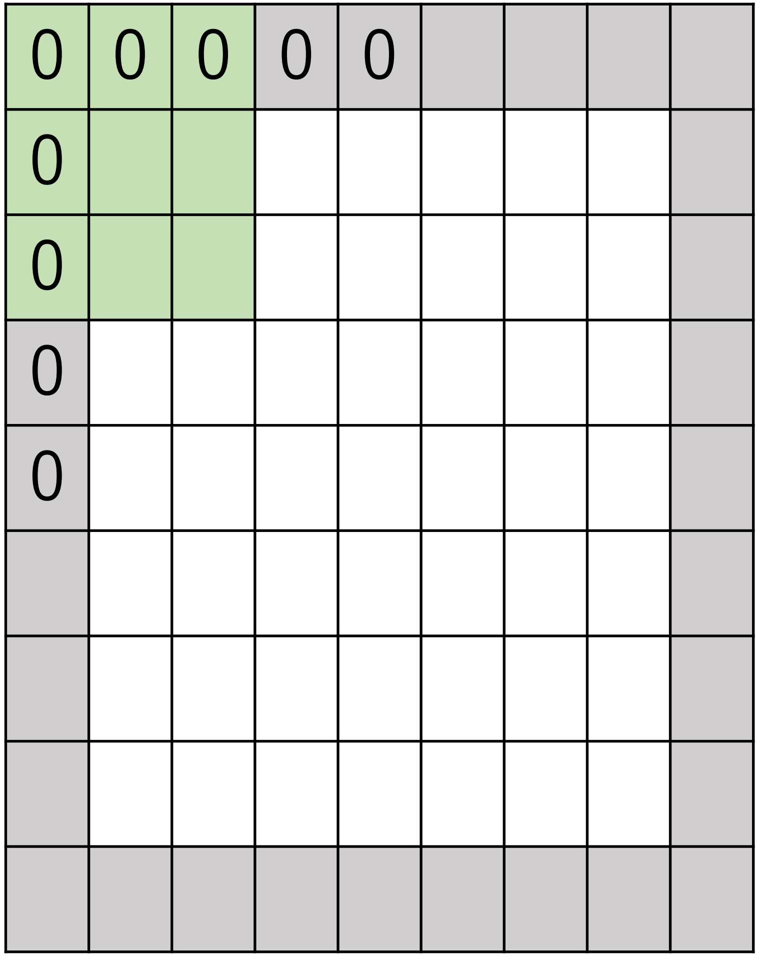
Output image size 유지

Output size

$$= (N - F + 2P) / stride + 1$$

Q) 7*7 행렬, 3*3 filter, stride 1
padding 1pixel border일 때,
output size?

Padding



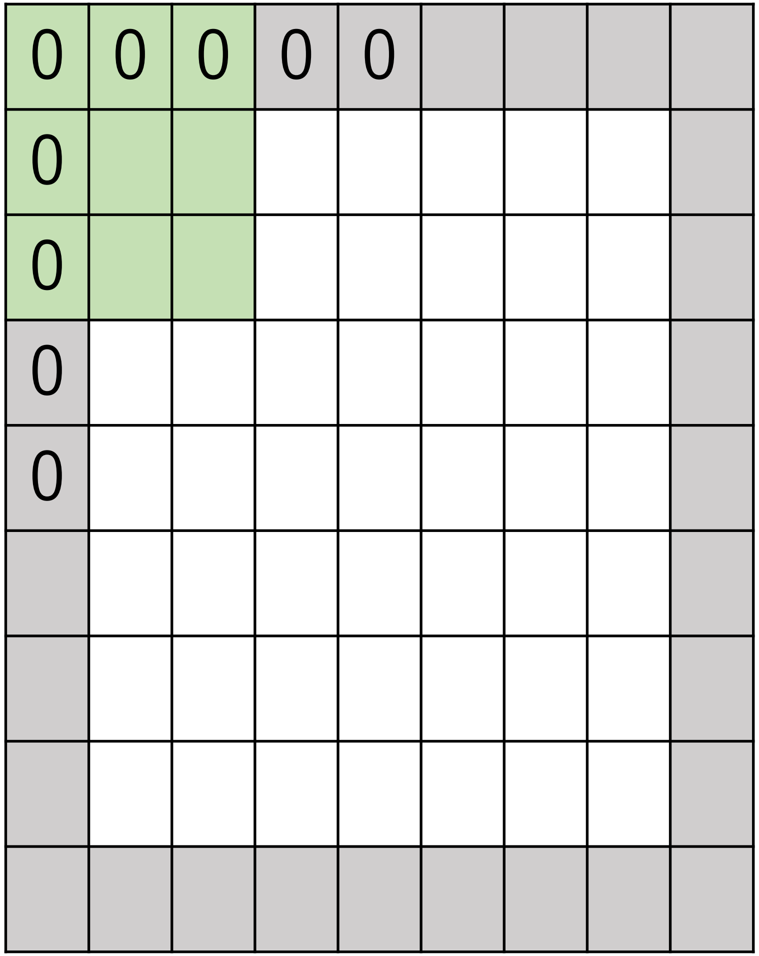
Output image size 유지

Output size
= (N – F + 2P) / stride + 1

Q) 7*7 행렬, 3*3 filter, stride 1
padding 1pixel border일 때,
output size?

A) (7 – 3 + 2) / 1 + 1 = 7

Padding

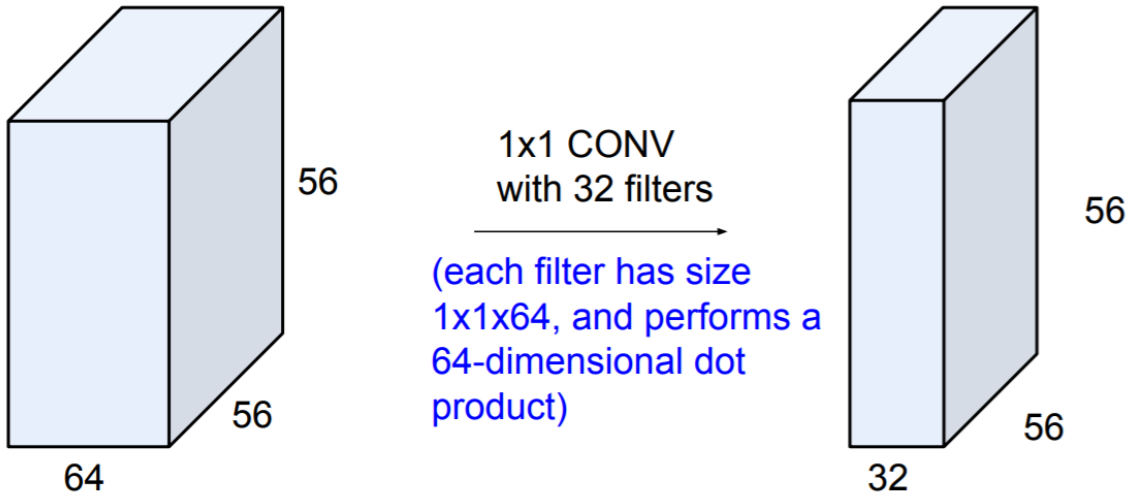


Padding Size

$$N + - F + 2P + 1 = N$$
$$P = (F - 1) / 2$$

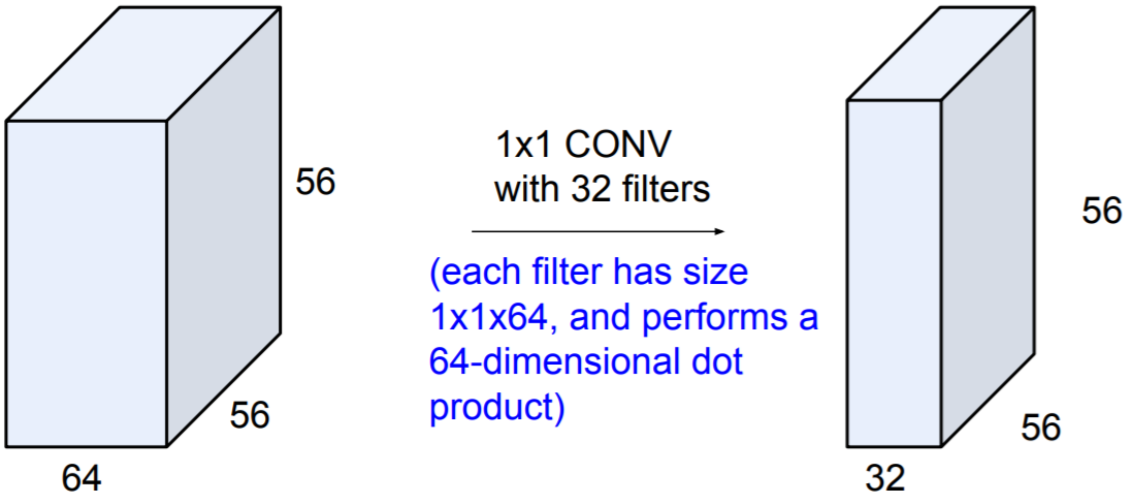
→ Map size 동일하게 유지 가능

1x1 convolution



1x1 size의 convolution filter를 이용한 Convolution Layer

1x1 convolution

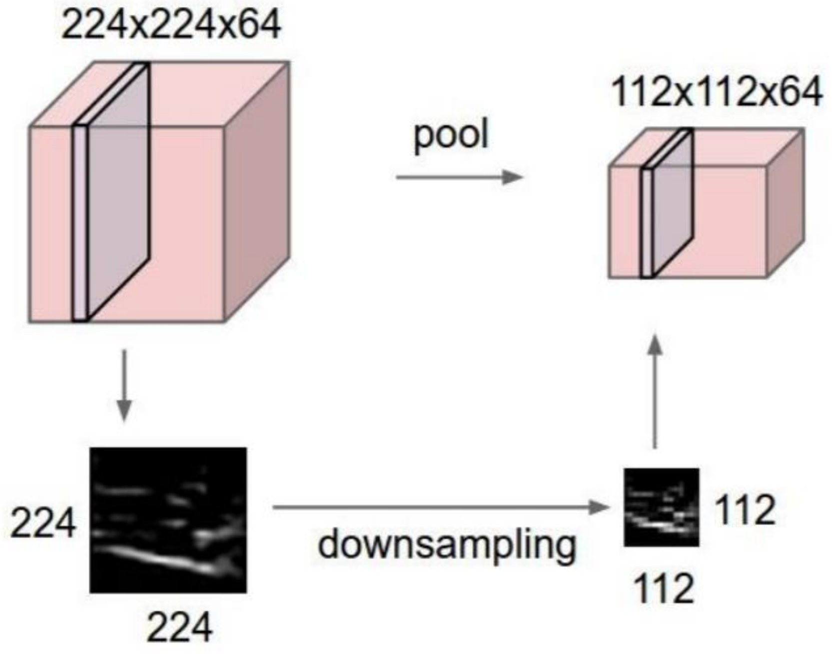


→ Channel 수 조절

→ Efficiency

→ Non-linearity

Pooling Layer



→Downsampling

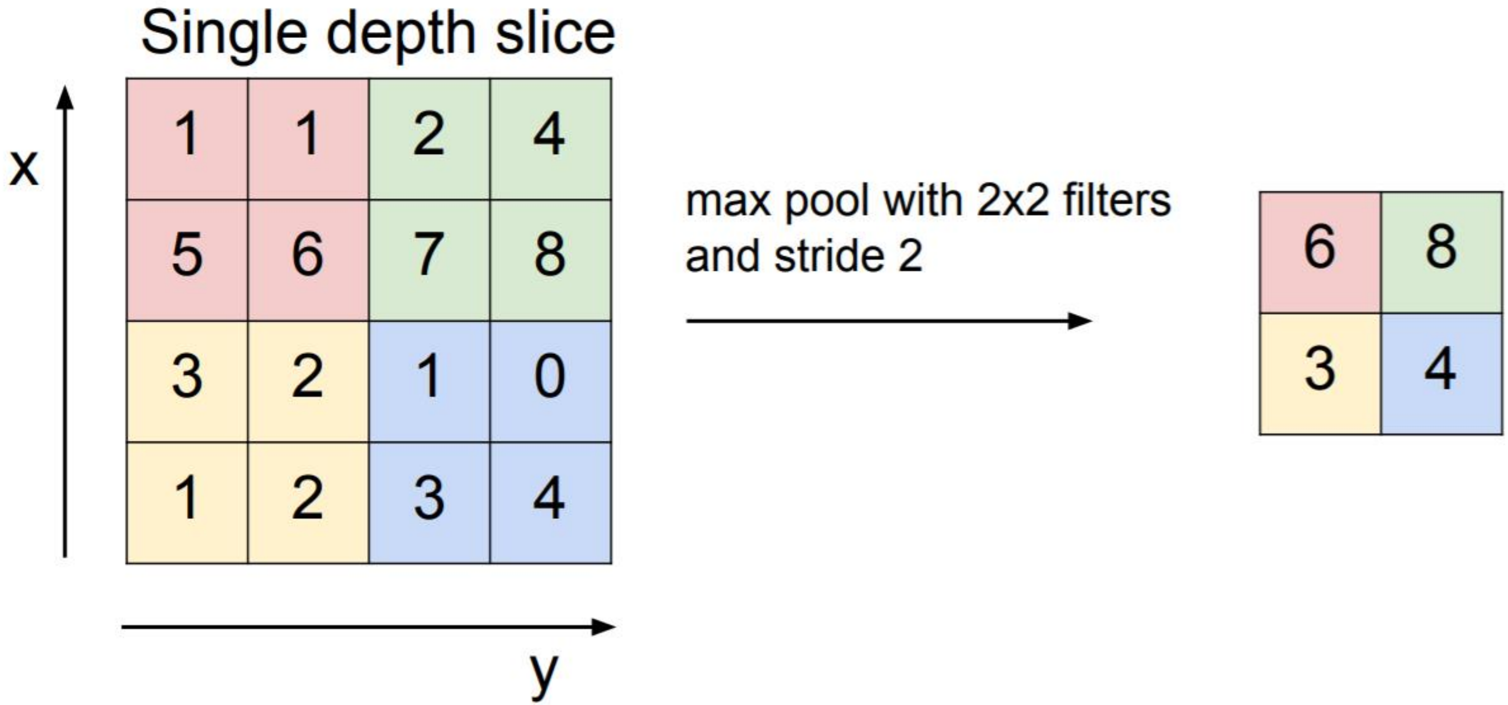
→ 각 activation map에 독립적으로 작용

→ 이미지의 차원을 공간적으로 줄여줌

→ Depth는 불변

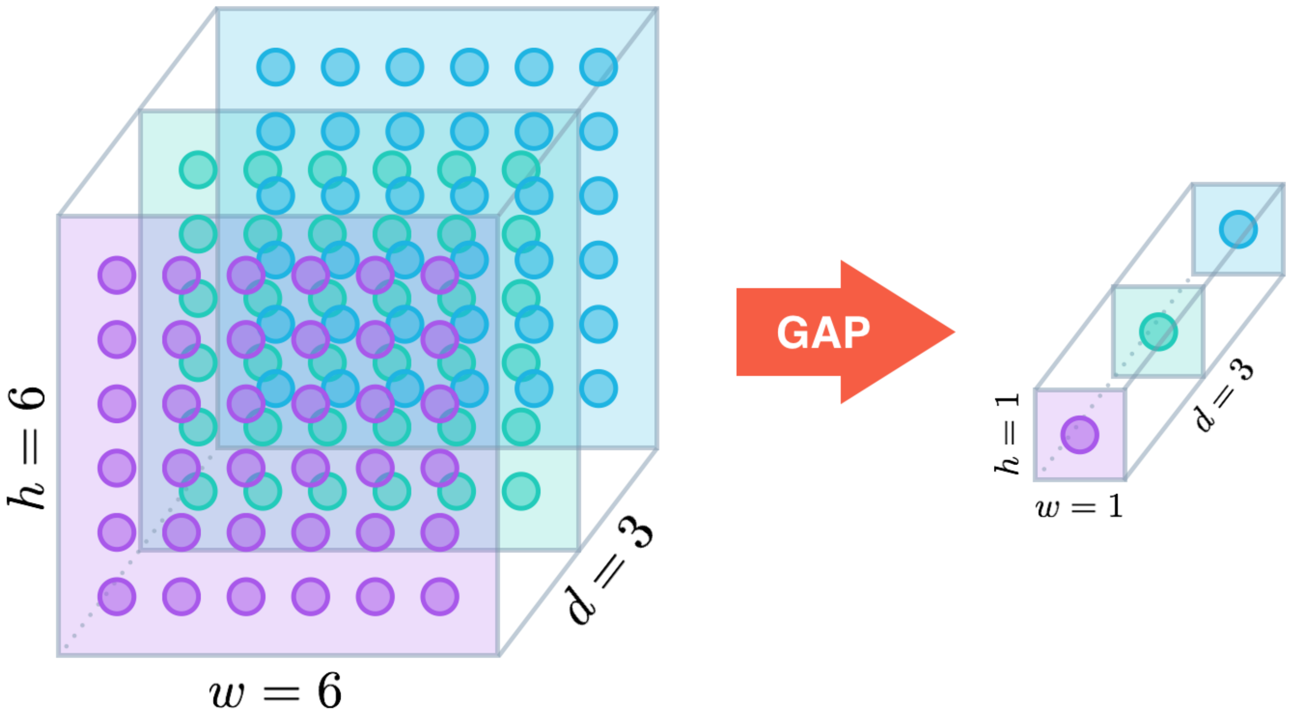
→ Padding X

Max Pooling



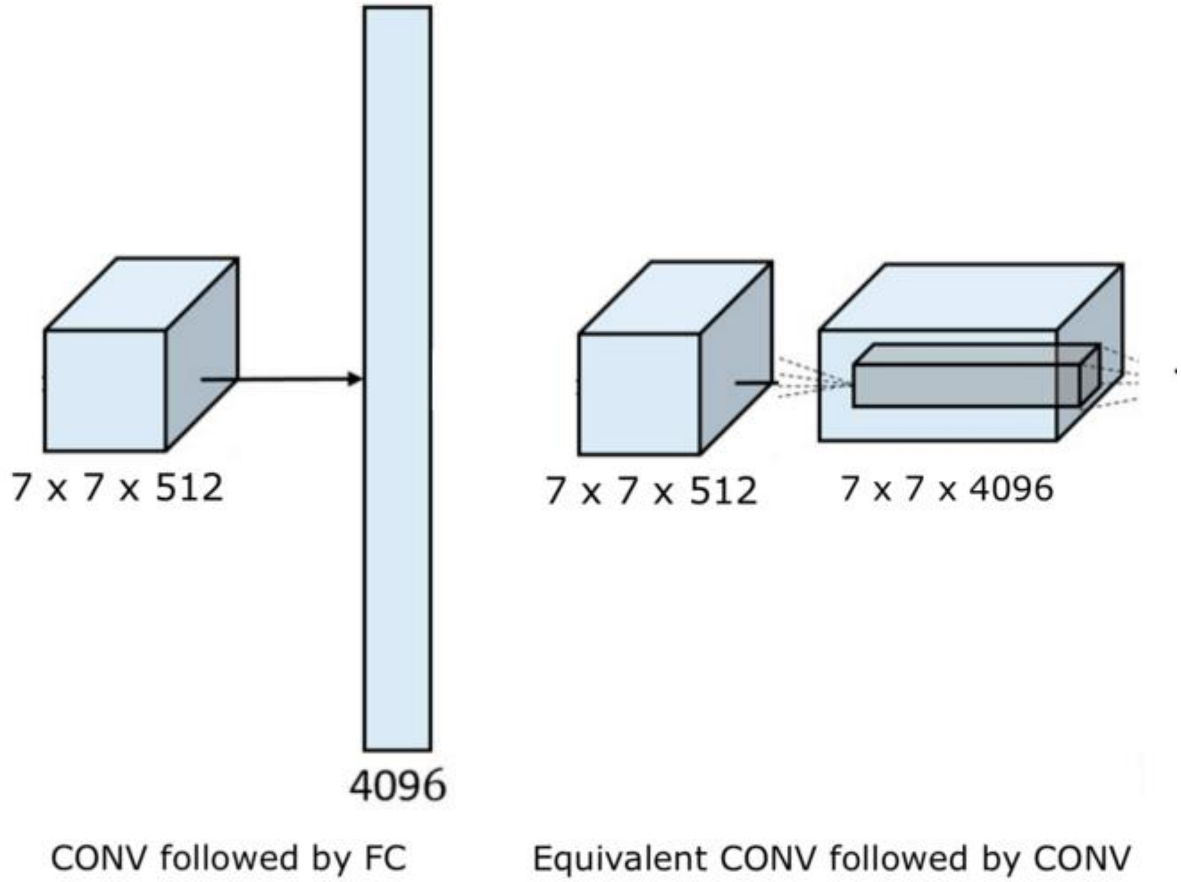
→ Max값만 이용

Global Averaging Pooling

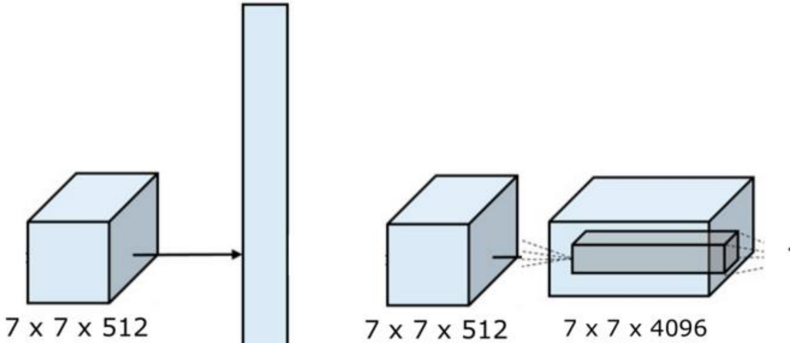


→ Feature를 1차원 벡터로 만들기 위함

FC layer vs Conv layer



FC layer to Conv layer



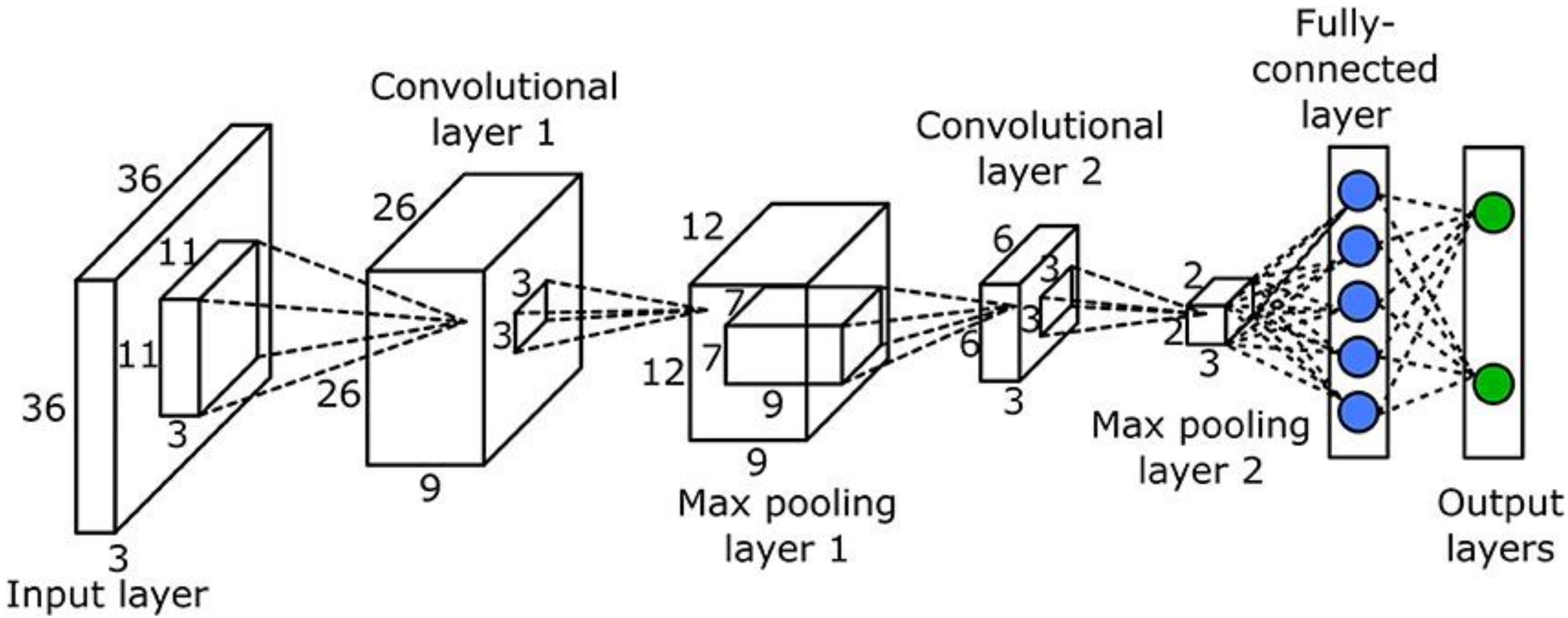
CONV followed by FC   Equivalent CONV followed by CONV

→ 모든 FC layer는 Conv layer로 변환 가능

→ FC layer의 W를 conv layer의 filter로 변환

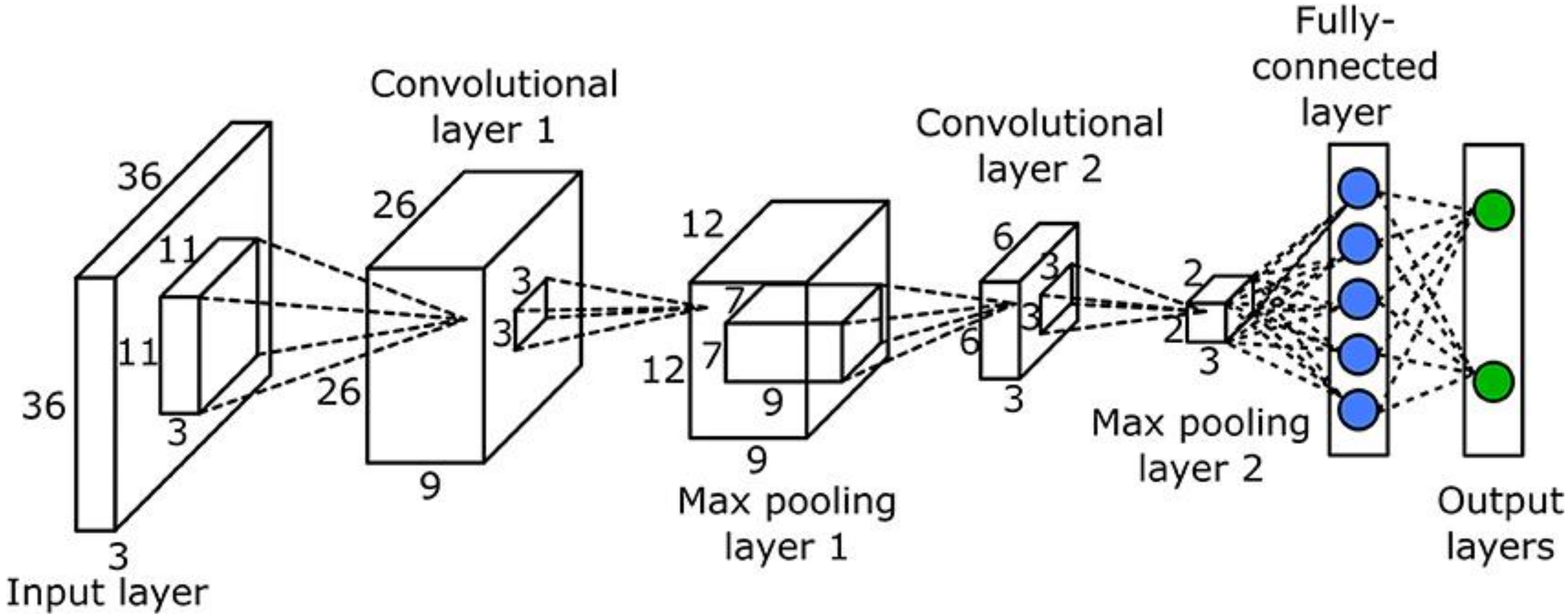
→ 1 forward pass => image "sliding"

ConvNet



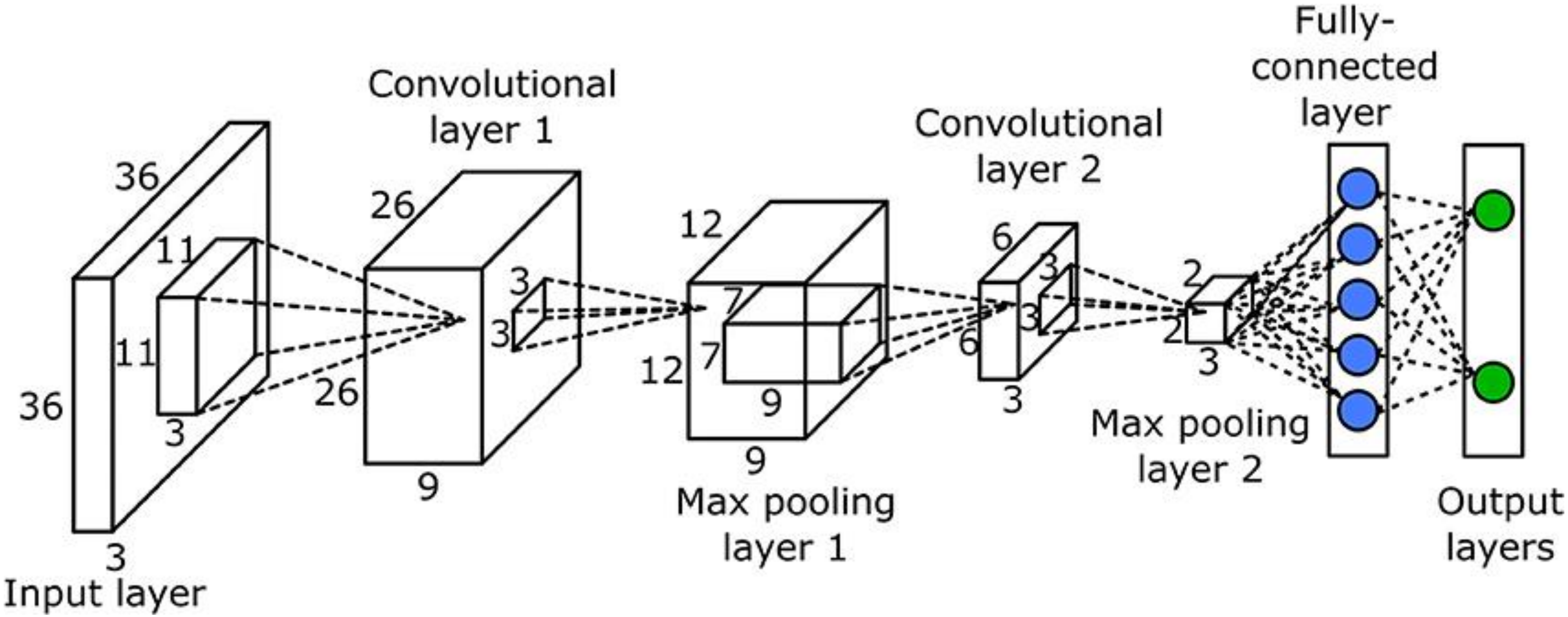
Input layer → Conv ~ Pooling layer → FC layer ~ Output layer

Layer Sizing



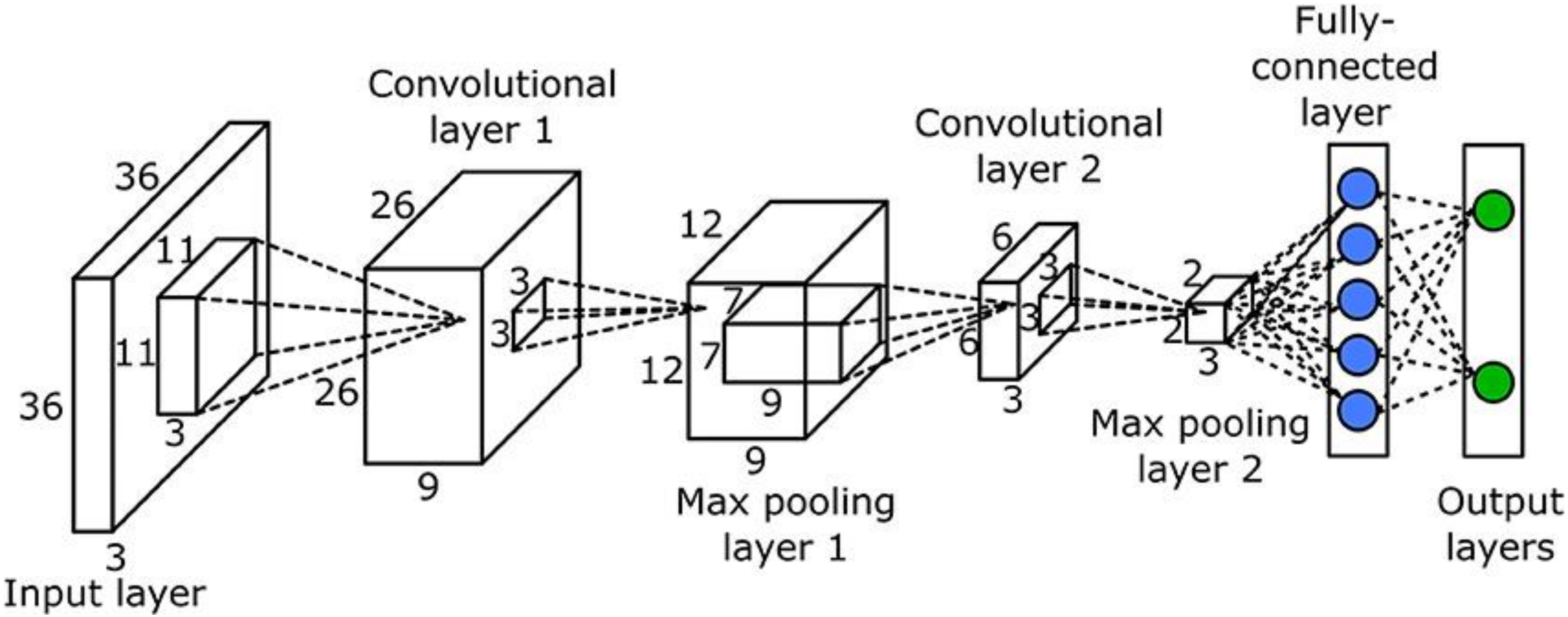
Input layer: 2로 여러 번 나눌 수 있어야 함

Layer Sizing



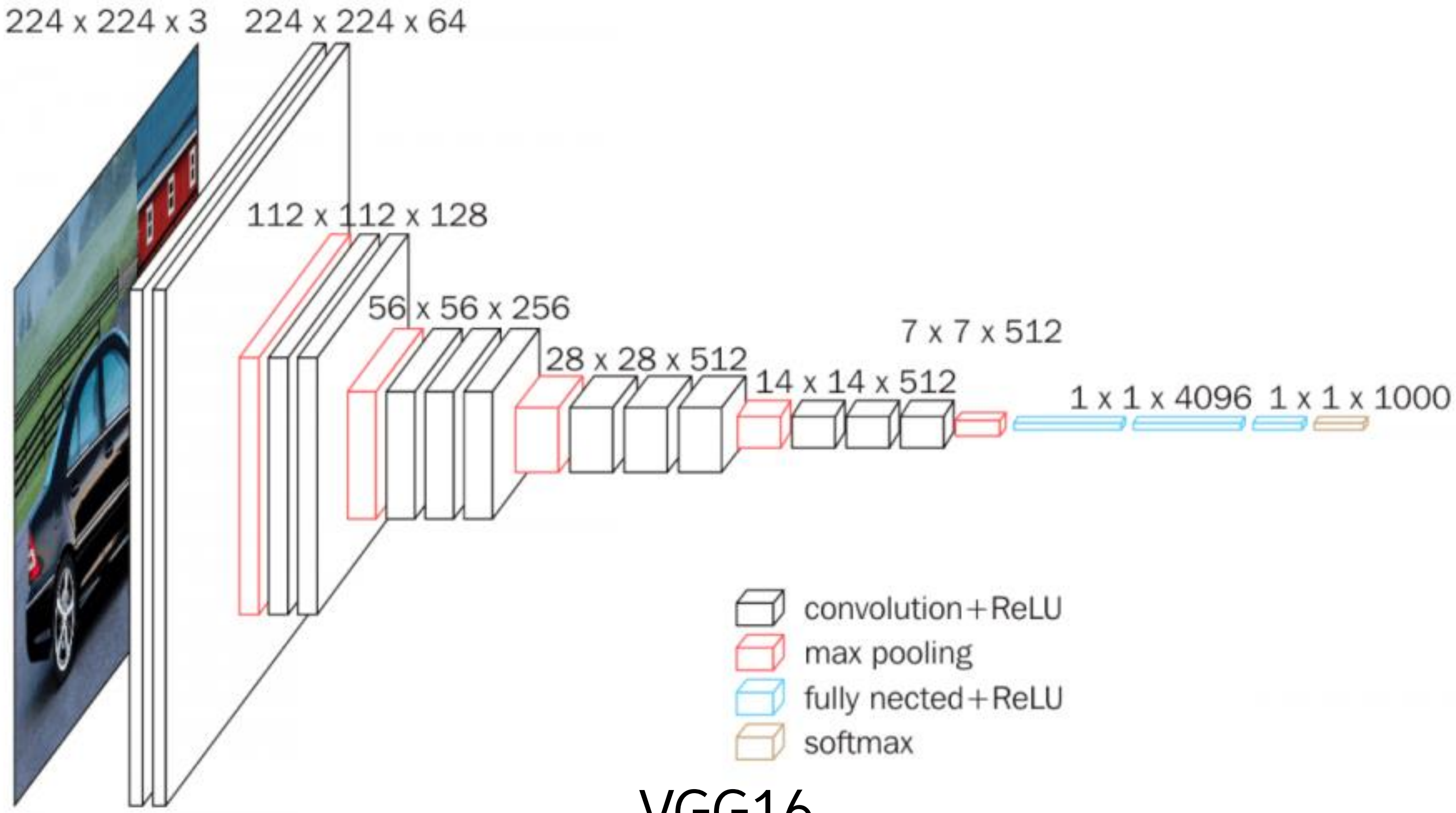
Conv layer: 최대 5x5의 작은 필터, s = 1, l = 0를 만드는 0 padding

Layer Sizing



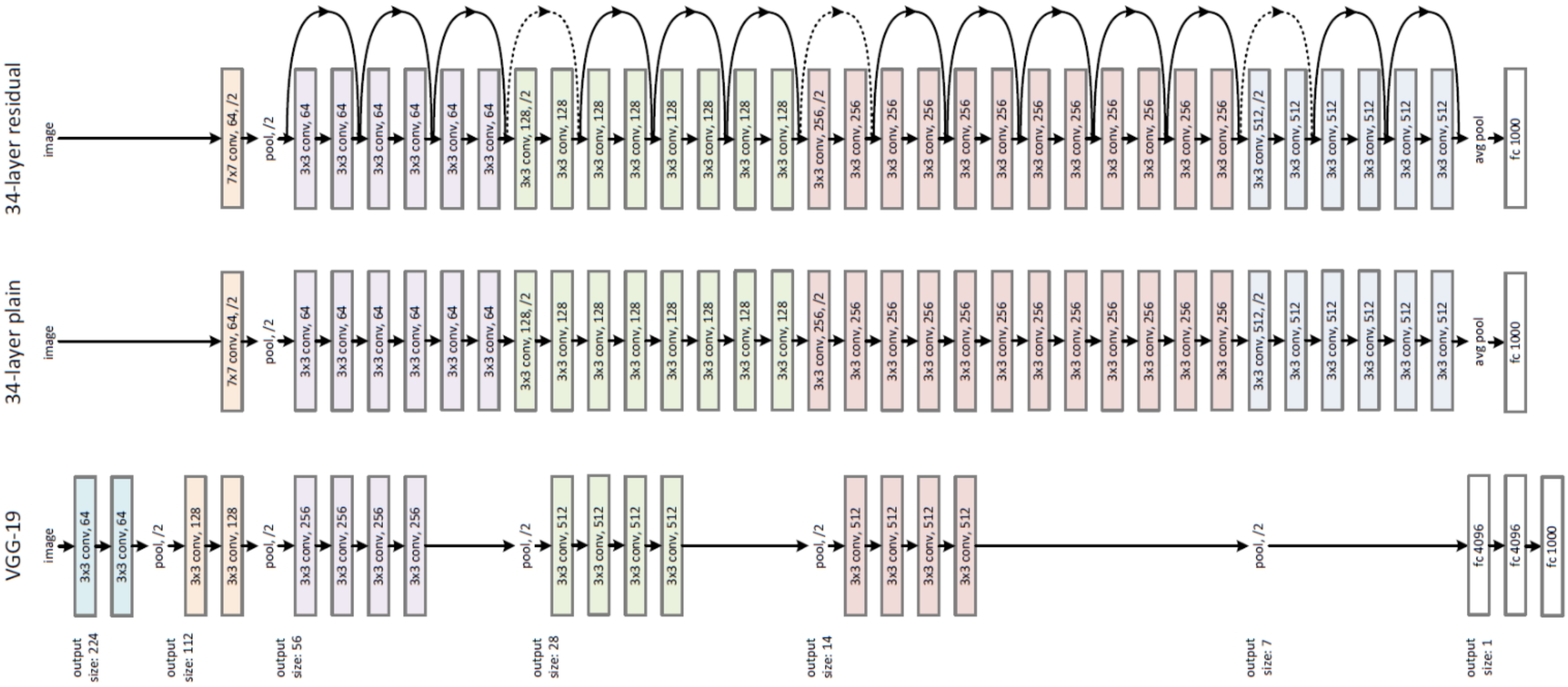
Pooling layer: (general) F = 2 or F = 3

VGGNet

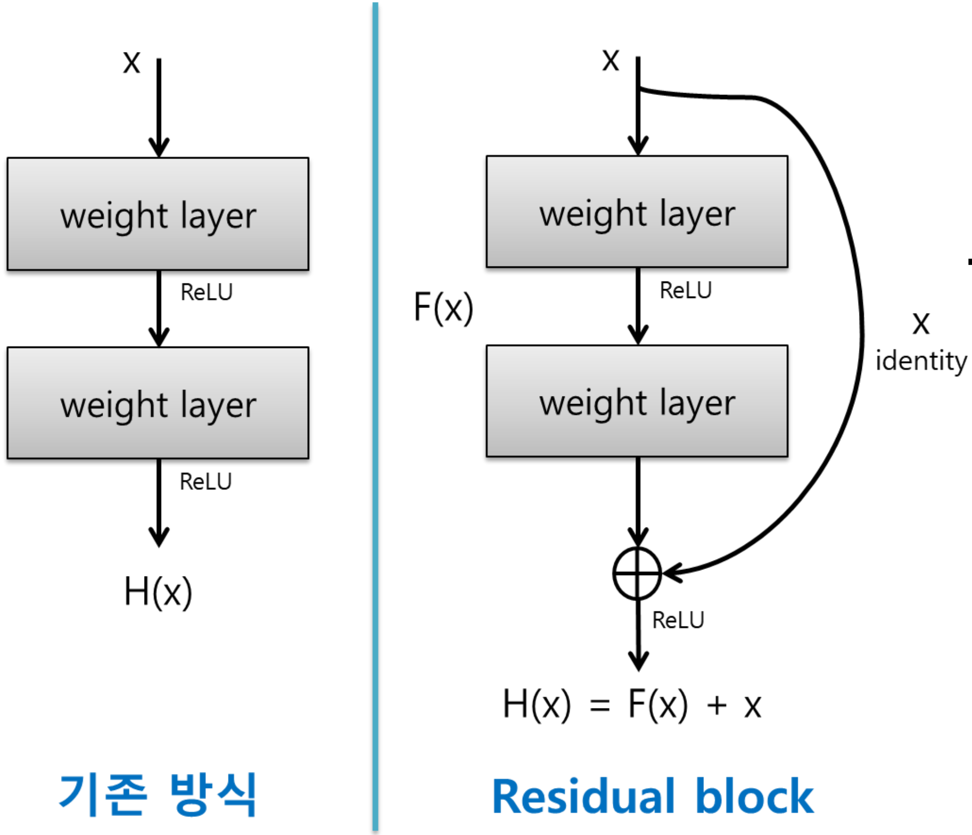


VGG16

ResNet


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

ResNet

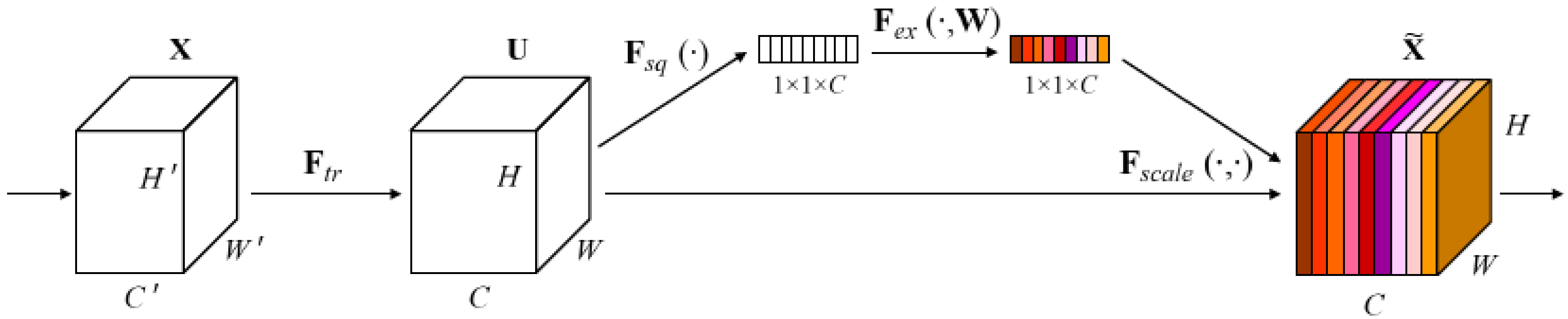


Residual Block

→ input을 output에 더해줄 수 있는 지름길을 하나 만들어 줌

residual을 최소로 해주는 것이 목표

SENet



SE block

→ VGGNet, GoogLeNet, ResNet 등에 첨가하여 성능 향상 가능

→ 목적: convolution을 통해 생성된 특성을 채널 당 중요도를 고려하여 재보정하는 것.

→ Conv 연산 뒤에 붙여 성능 향상 도모

감사합니다

Q&A