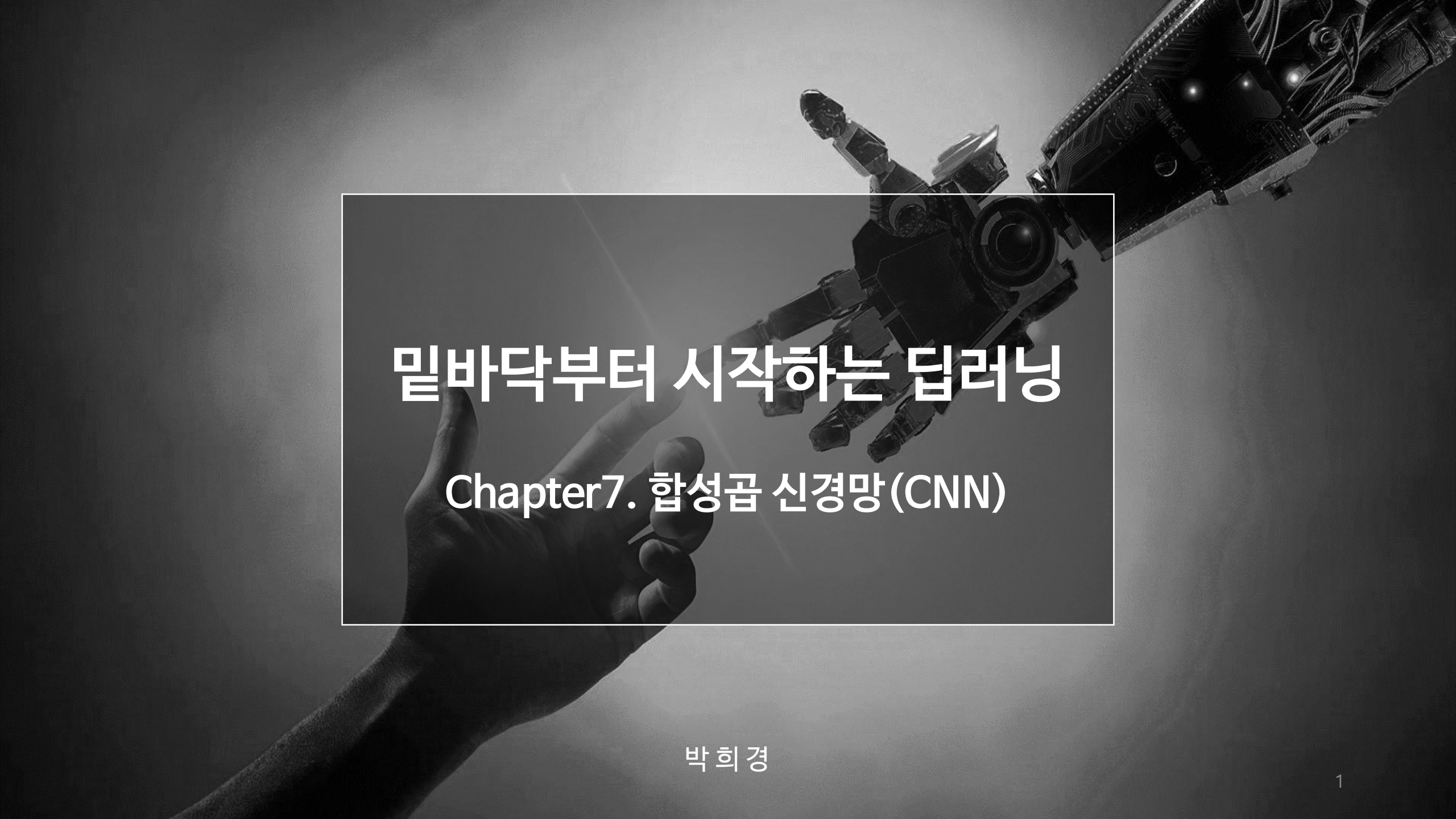
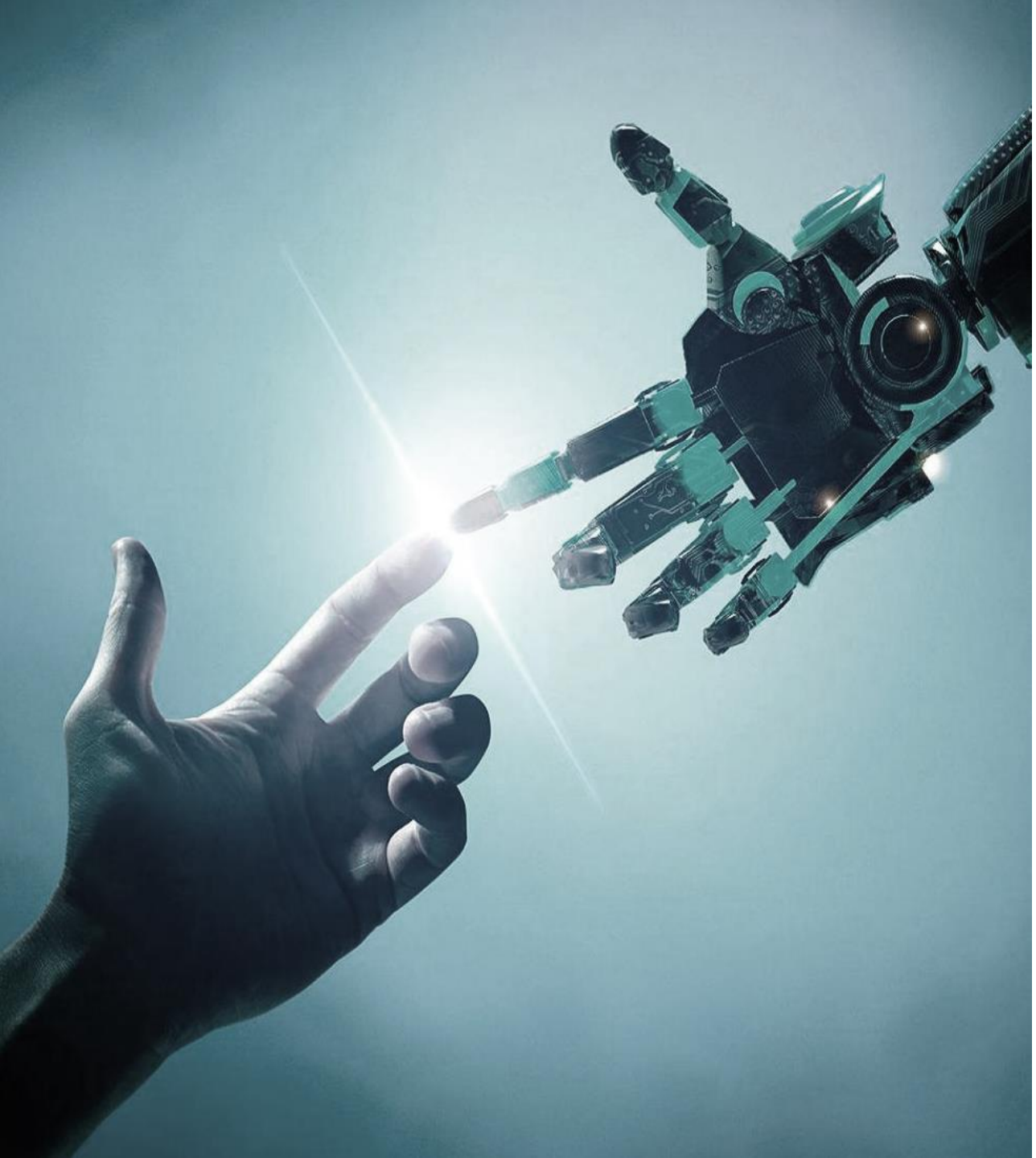


밑바닥부터 시작하는 딥러닝

Chapter7. 합성곱 신경망(CNN)

박 희 경



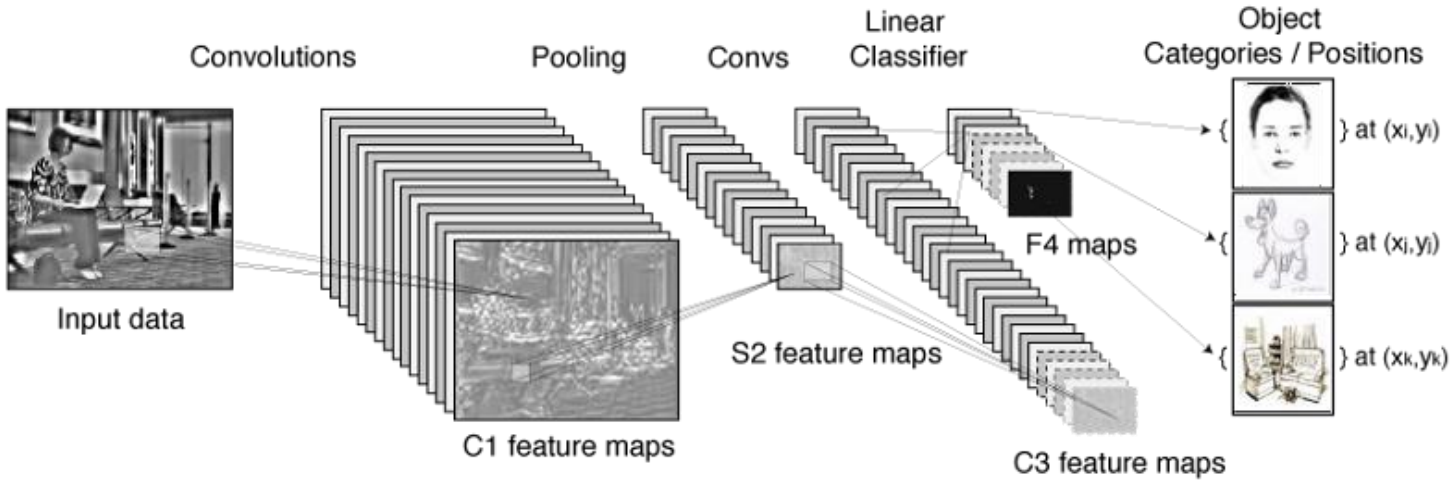
CONTENTS

01 전체 구조

CNN

CNN(Convolution Neural Network, 합성곱 신경망)이란?

: CNN은 합성곱(Convolution) 연산을 사용하는 ANN(Artificial Neural Network)의 한 종류이다.
Convolution을 사용하면 3차원 데이터의 공간적 정보를 유지한 채 다음 레이어로 보낼 수 있다.
대표적인 CNN으로는 LeNet(1998)과 AlexNet(2012)이 있으며, VGG, GoogLeNet, ResNet 등은
층을 더 깊게 쌓은 CNN기반의 DNN(Depp Neural Network, 심층 신경망)이다.

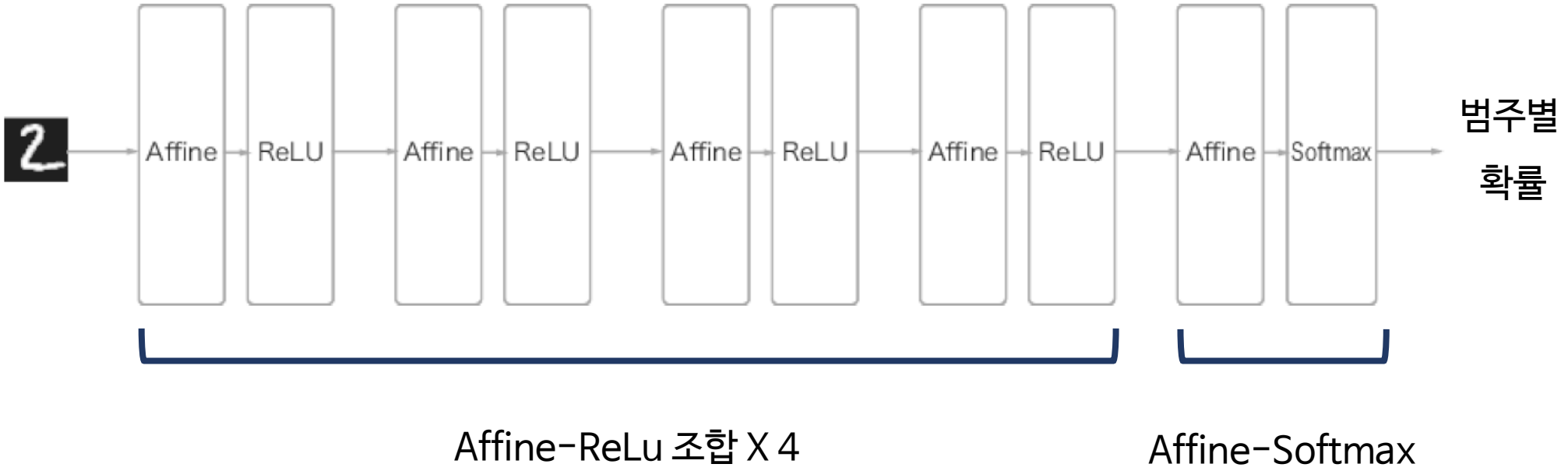


01 전체 구조

CNN

기존의 신경망(Artificial Neural Network) 구조

: 인접하는 계층의 모든 뉴런이 결합되어 있는 완전연결(fully-connected)로, Affine 계층으로 구성되어 있다.

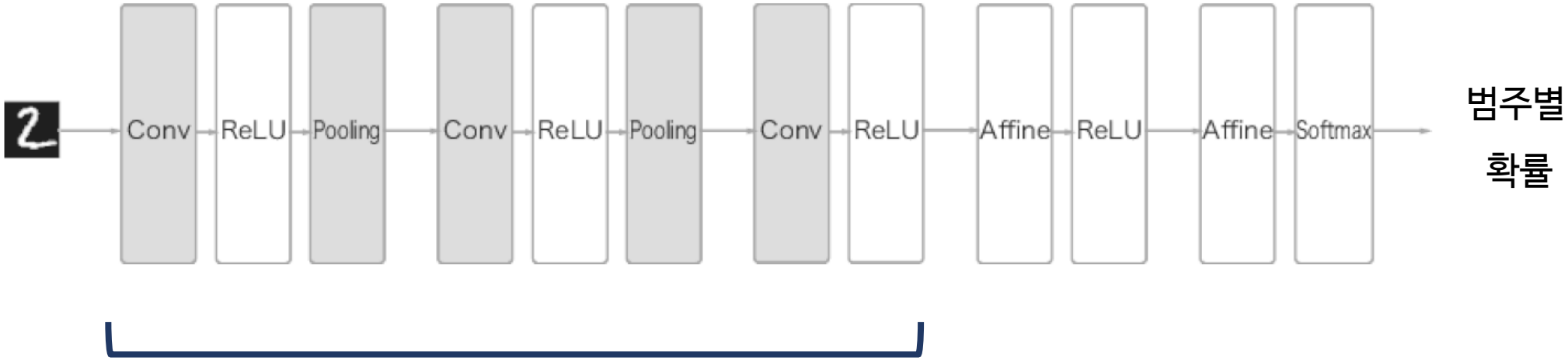


01 전체 구조

CNN

CNN 구조

: 신경망 구조에서 합성곱 계층(Conv), 풀링 계층(Pooling)이 추가된다.

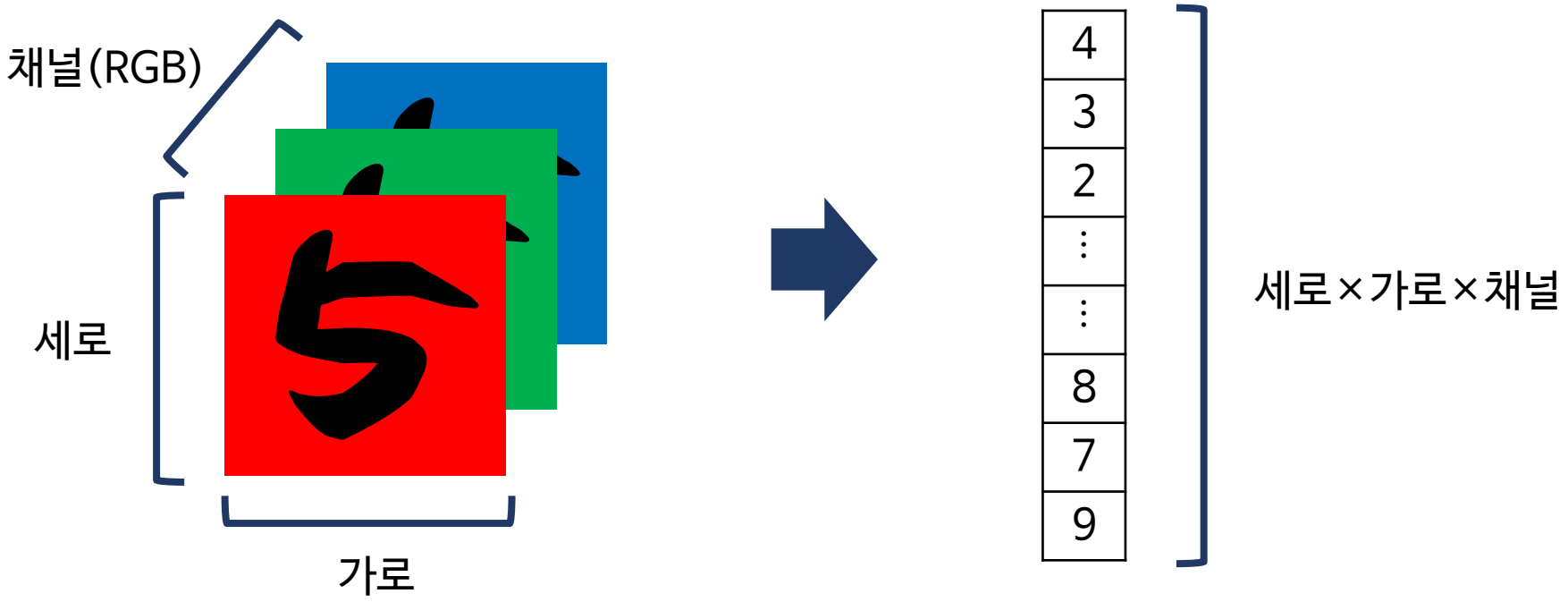


02 합성곱 계층

완전연결 계층의 문제점

데이터 형상의 무시

: 세로·가로·채널(색상)로 구성된 3차원 데이터인 이미지가 완전연결 계층(fully-connected layer)에 입력될 때, 1차원 데이터로 변형되게 되면서 공간적 정보를 잃게 된다.

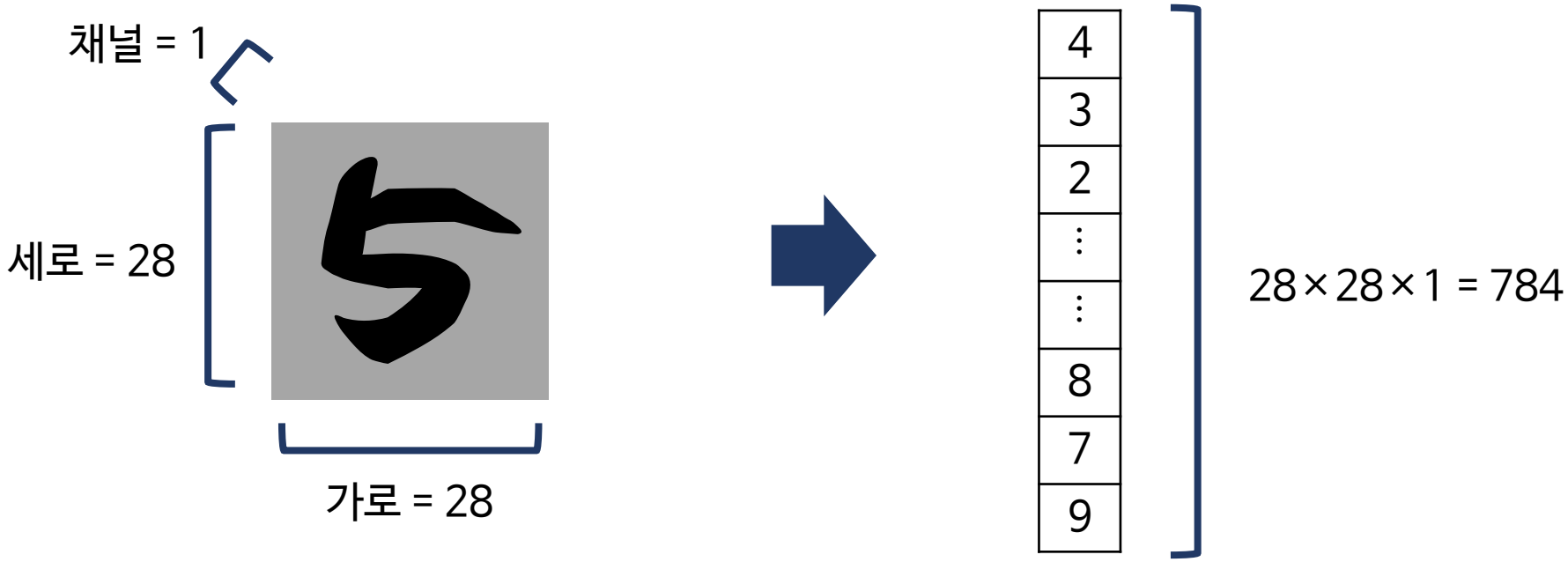


02 합성곱 계층

완전연결 계층의 문제점

데이터 형상의 무시

: 예를 들어, 공간적으로 가까운 픽셀은 값이 비슷하거나, RGB의 각 채널은 서로 밀접하게 관련되어 있거나,
거리가 먼 픽셀끼리는 별 연관이 없는 등, 3차원 속 의미를 갖는 본질적인 패턴이 무시된다.



합성곱 연산

합성곱(Convolution)

: 특정 (높이, 너비)를 갖은 필터(Filter, Kernel)를 일정 간격(Stride)으로 이동해가며 입력 데이터에 적용 (원소 곱 후, 총합 : 단일 곱셈-누산)

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

Input data

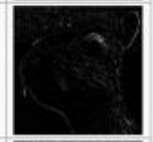
*

2	0	1
0	1	2
1	0	2

Filter

15	16
6	15

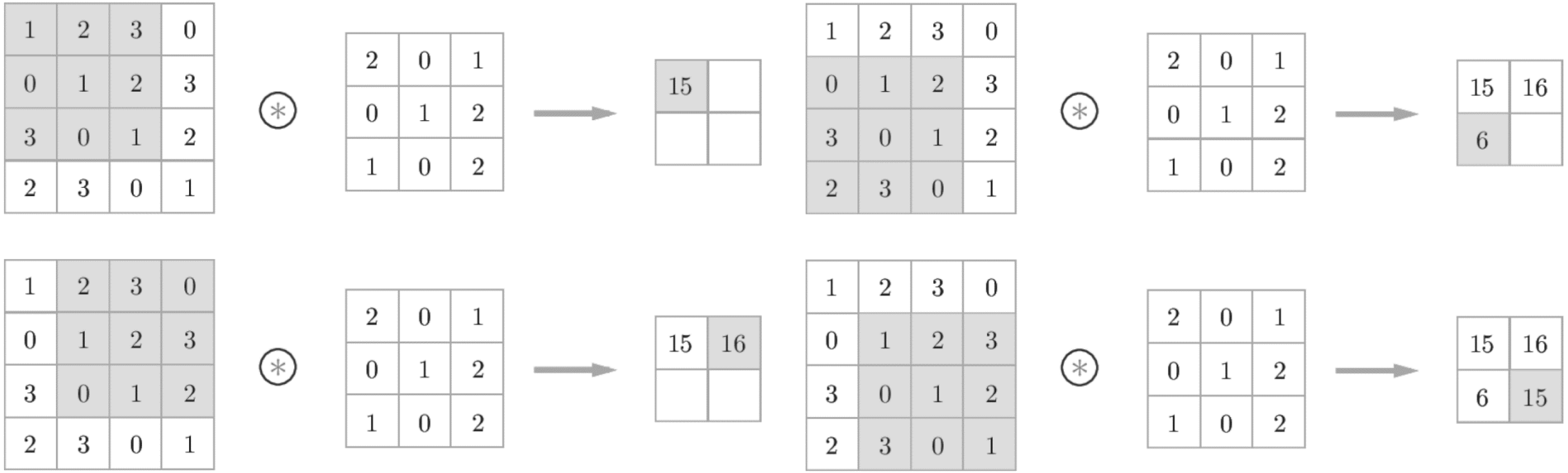
Output data

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

합성곱 연산

합성곱(Convolution) 과정

: 동일한 필터가 이미지 전체에 적용



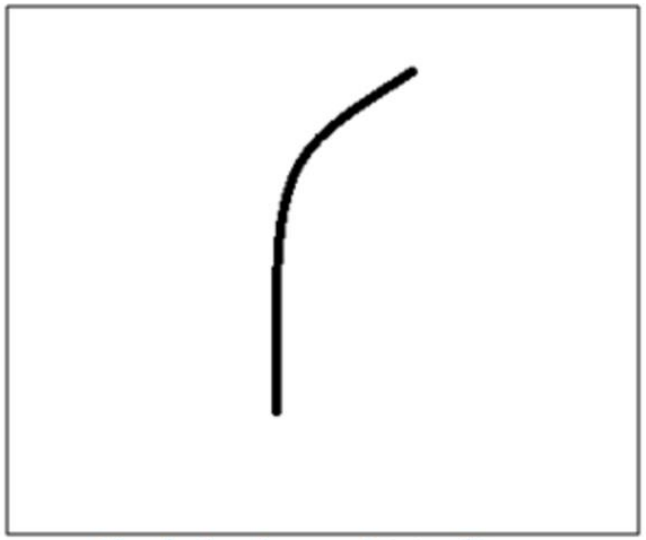
$2\times2 + 3\times0 + \cdots + 1\times0 + 2\times2 = 16$

합성곱(Convolution)의 효과

: 필터는 그 특징이 데이타에 있는지 없는지를 검출

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

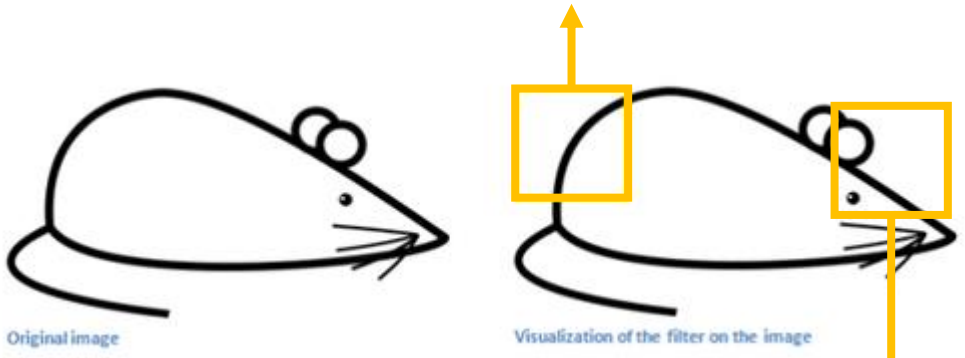
위의 곡선과 비슷한 특징들을 추출

합성곱 연산

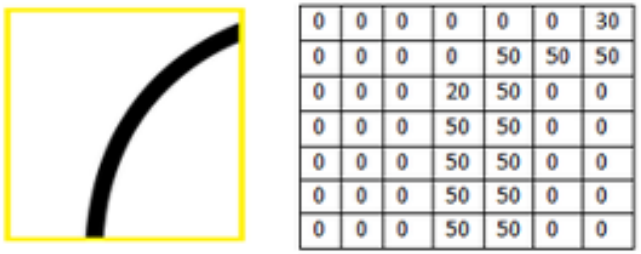
합성곱(Convolution)의 효과

: 해당 곡선의 필터를 적용했을 때의 예

필터와 비슷한 부분



필터와 비슷하지 않은 부분



0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

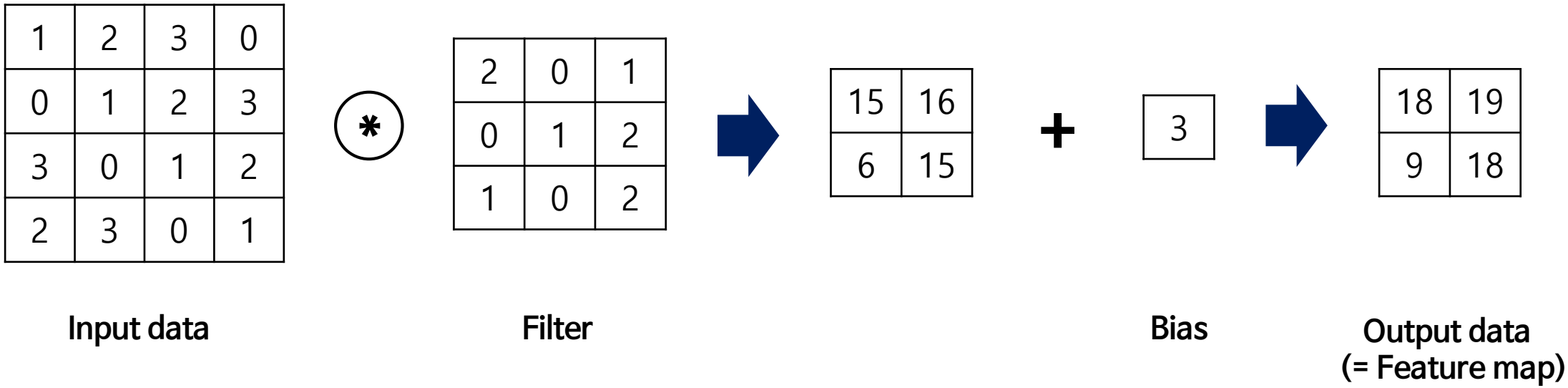
Pixel representation of filter

Multiplication and Summation = 0

합성곱 연산

합성곱(Convolution) + 편향

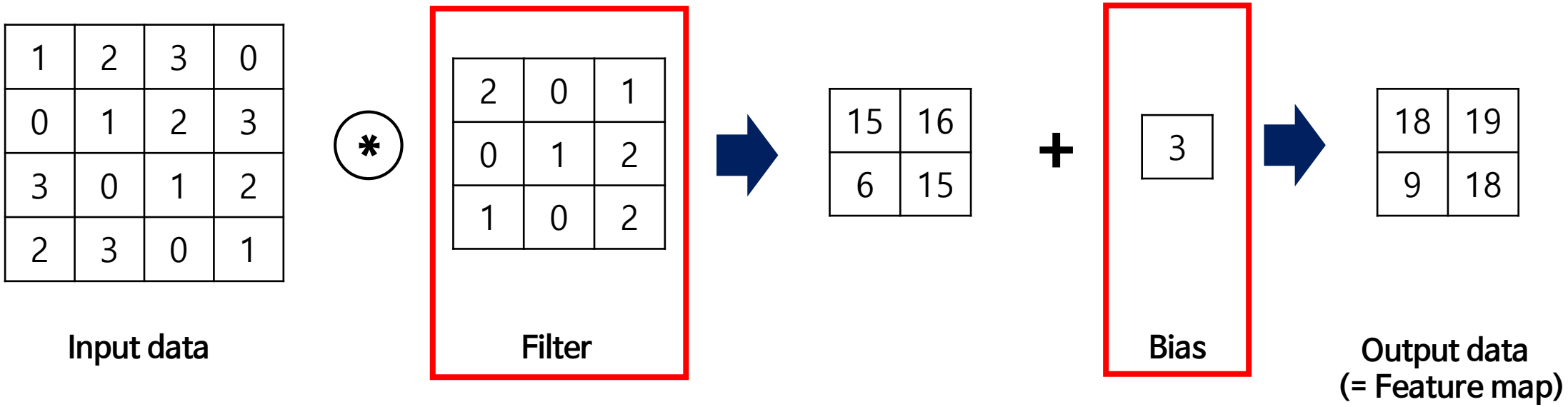
: 필터를 적용한 후, 모든 원소에 편향이 더해진다 (브로드캐스트).



합성곱 연산

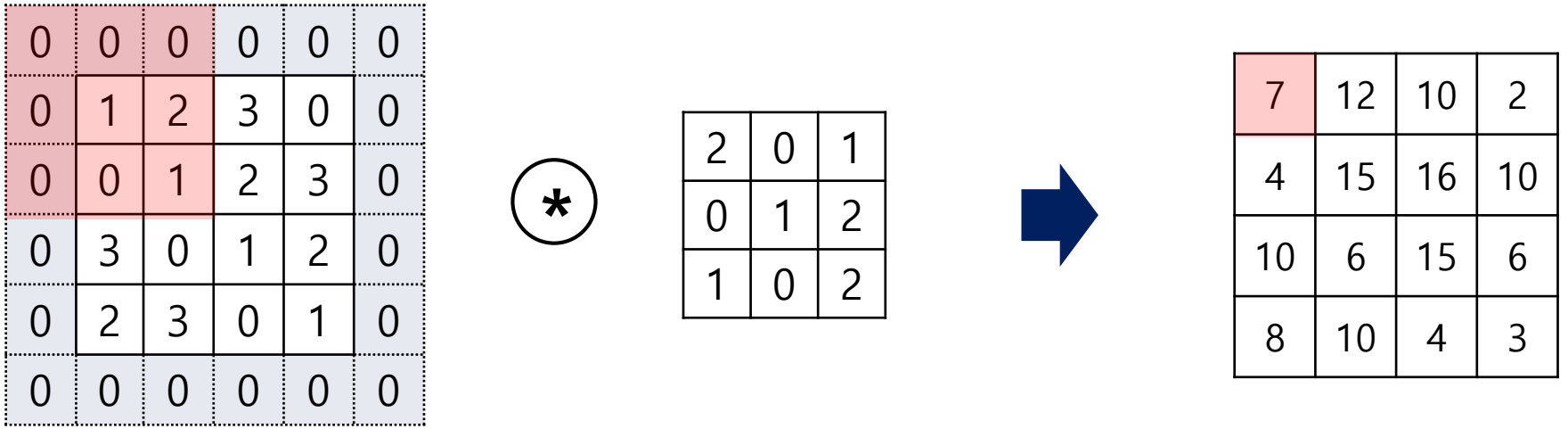
합성곱 연산에서의 매개변수

: 필터(가중치), 편향이 학습을 시킬 매개변수이다.



패딩(Padding)

: 합성곱 연산을 수행하기 전에 입력 데이터 주변을 특정 값(주로 0을 사용 - Zero paddig)을 채우는 단계로,
주로, 입력 데이터와 출력 데이터의 크기를 맞추기 위해 사용한다.



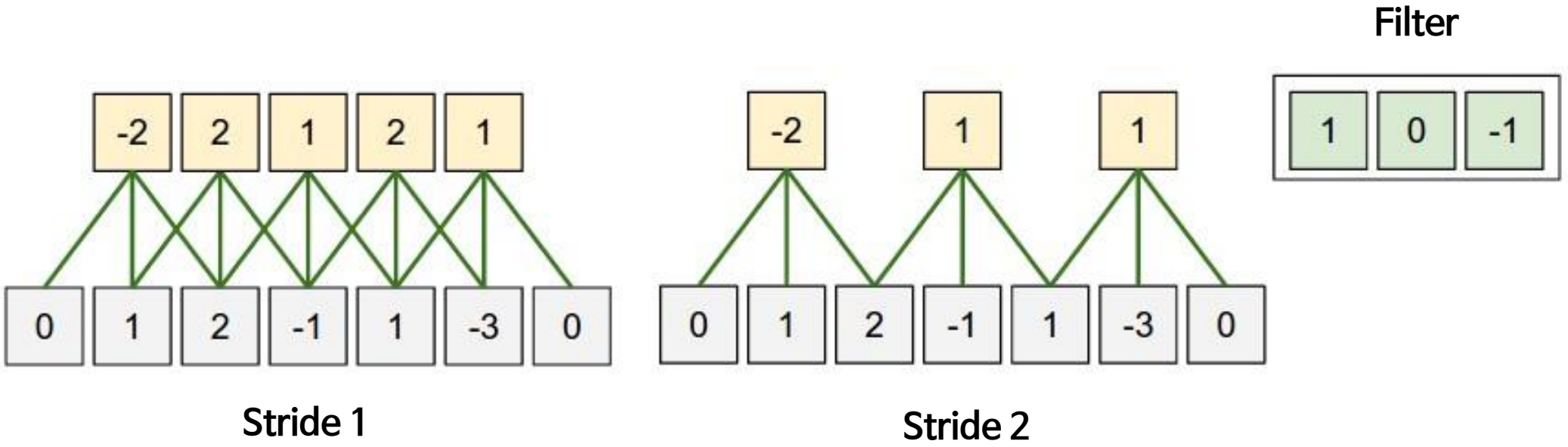
Input Data
raw size - 4×4
after padding - 6×6

Filter
3×3

Output Data
4×4

스트라이드(Stride) - 1 Dimension

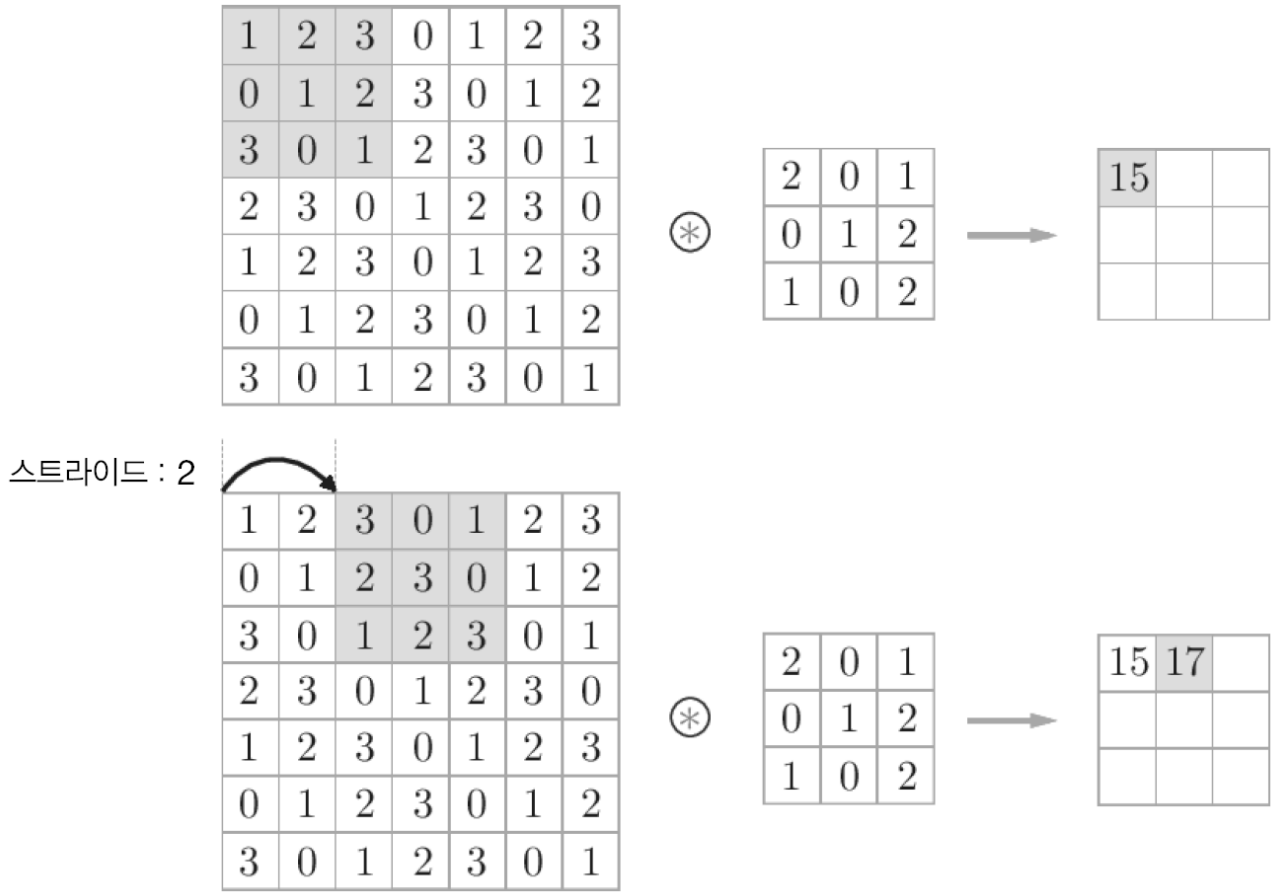
: 필터를 적용하는 위치의 간격



http://cs231n.github.io/convolutional-networks/

스트라이드(Stride) - 2 Dimension

: 필터를 적용하는 위치의 간격으로,
스트라이드가 커지면 출력크기가
작아진다.



스트라이드

출력 크기 계산

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

입력 크기 - (H, W)

필터 크기 - (FH, FW)

출력 크기 - (OH, OW)

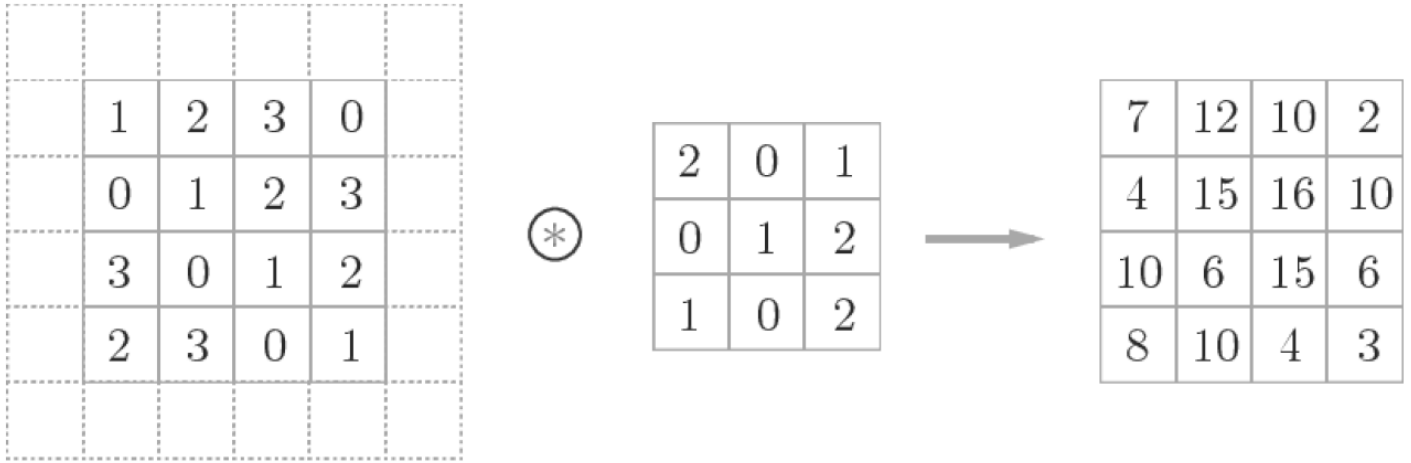
패딩 - P

스트라이드 - S

: 출력 크기(OH, OW)는 정수로 나누어 떨어지는 값이어야 한다. (∵원소의 개수)

딥러닝 프레임워크 중에는 값이 딱 나누어 떨어지지 않을 때, 가장 가까운 정수로 반올림 하는 경우도 있다.

출력 크기 계산 - 예제 1



(4, 4)
입력 데이터(패딩 : 1)
스트라이드 : 1

(3, 3)
필터

(4, 4)
출력 데이터

$$OH = \frac{H + 2P - FH}{S} + 1$$
$$OW = \frac{W + 2P - FW}{S} + 1$$

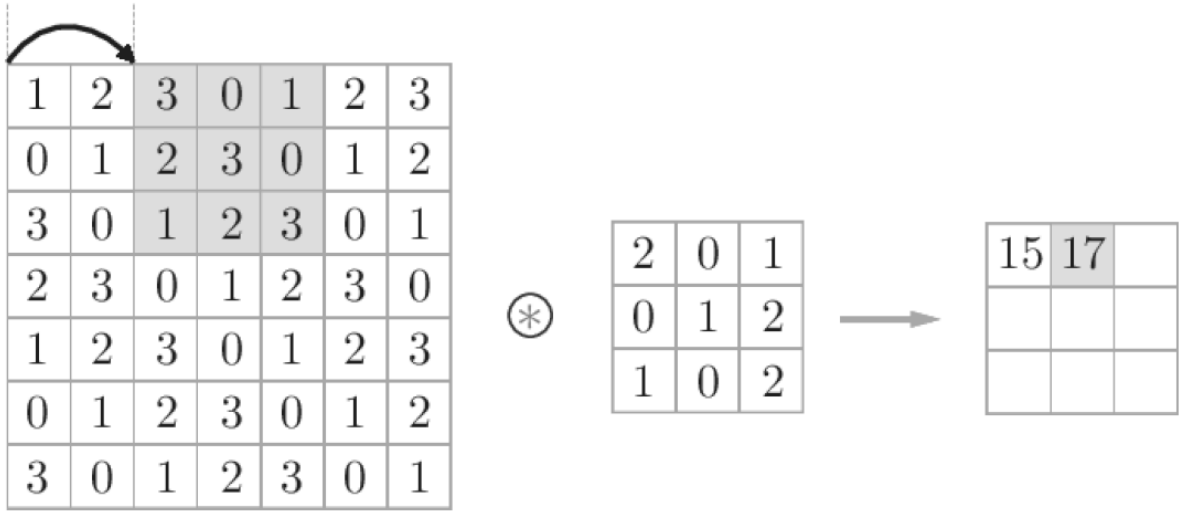
$$OH = \frac{4 + 2 \times 1 - 3}{1} + 1 = 4$$

$$OW = \frac{4 + 2 \times 1 - 3}{1} + 1 = 4$$

출력 크기 계산 - 예제 2

$$OH = \frac{H + 2P - FH}{S} + 1$$
$$OW = \frac{W + 2P - FW}{S} + 1$$

스트라이드 : 2



입력 데이터 : (7×7)
패딩 : 0

필터 : (3×3)

$$OH = \frac{7 + 2 \times 0 - 3}{2} + 1 = 3$$

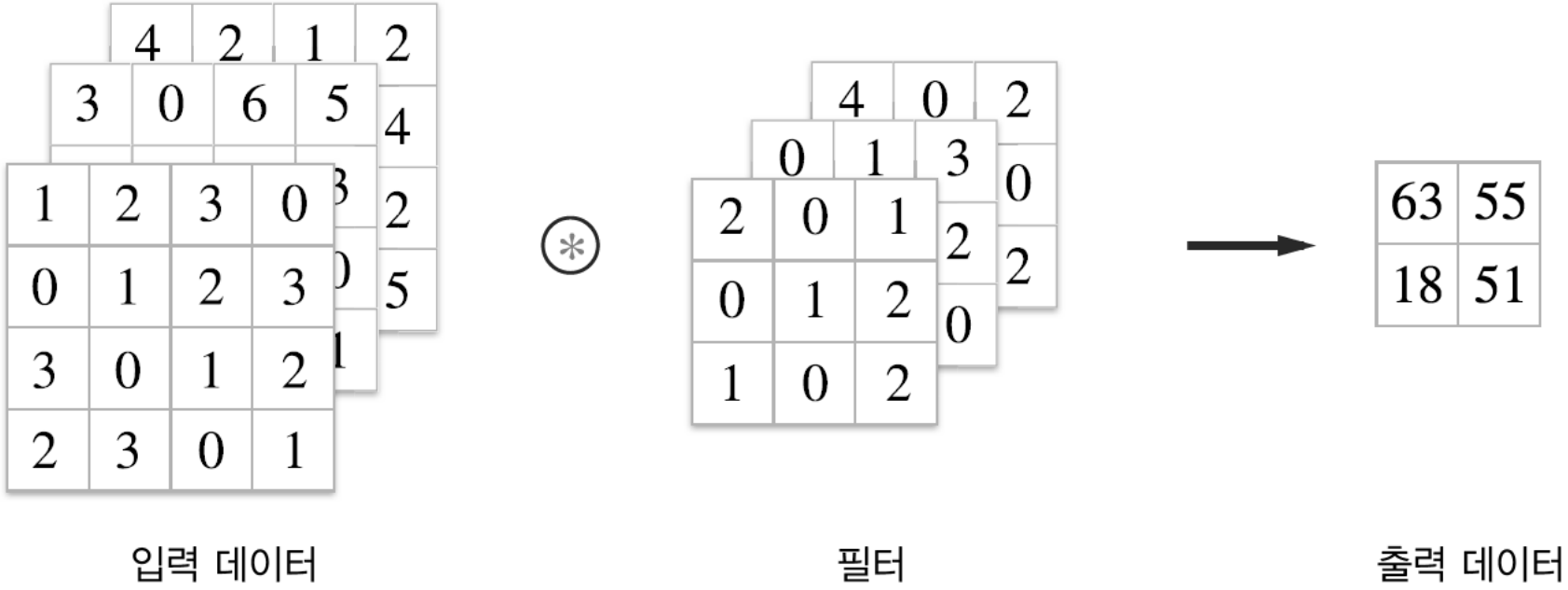
$$OW = \frac{7 + 2 \times 0 - 3}{1} + 1 = 3$$

02 합성곱 계층

3차원 데이터의 합성곱 연산

채널까지 고려한 3차원 데이터

:3차원 이미지 데이터에 대해서, 필터도 이미지와 같은 채널의 개수를 갖고 있어야 한다.

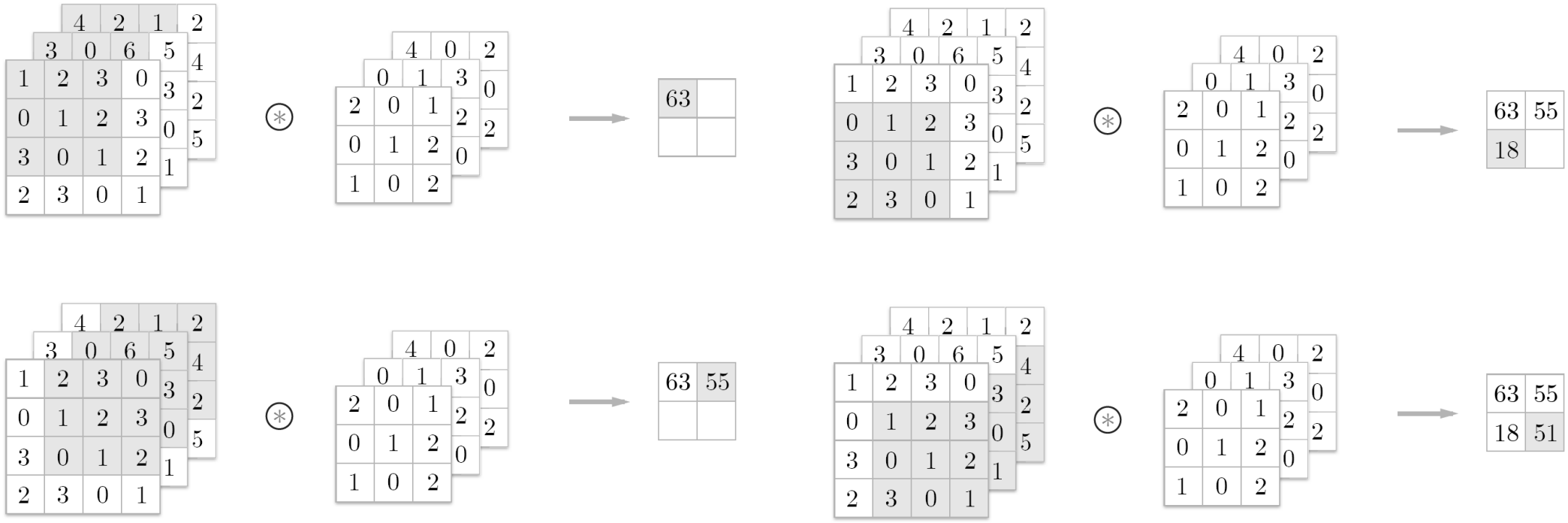


02 합성곱 계층

3차원 데이터의 합성곱 연산

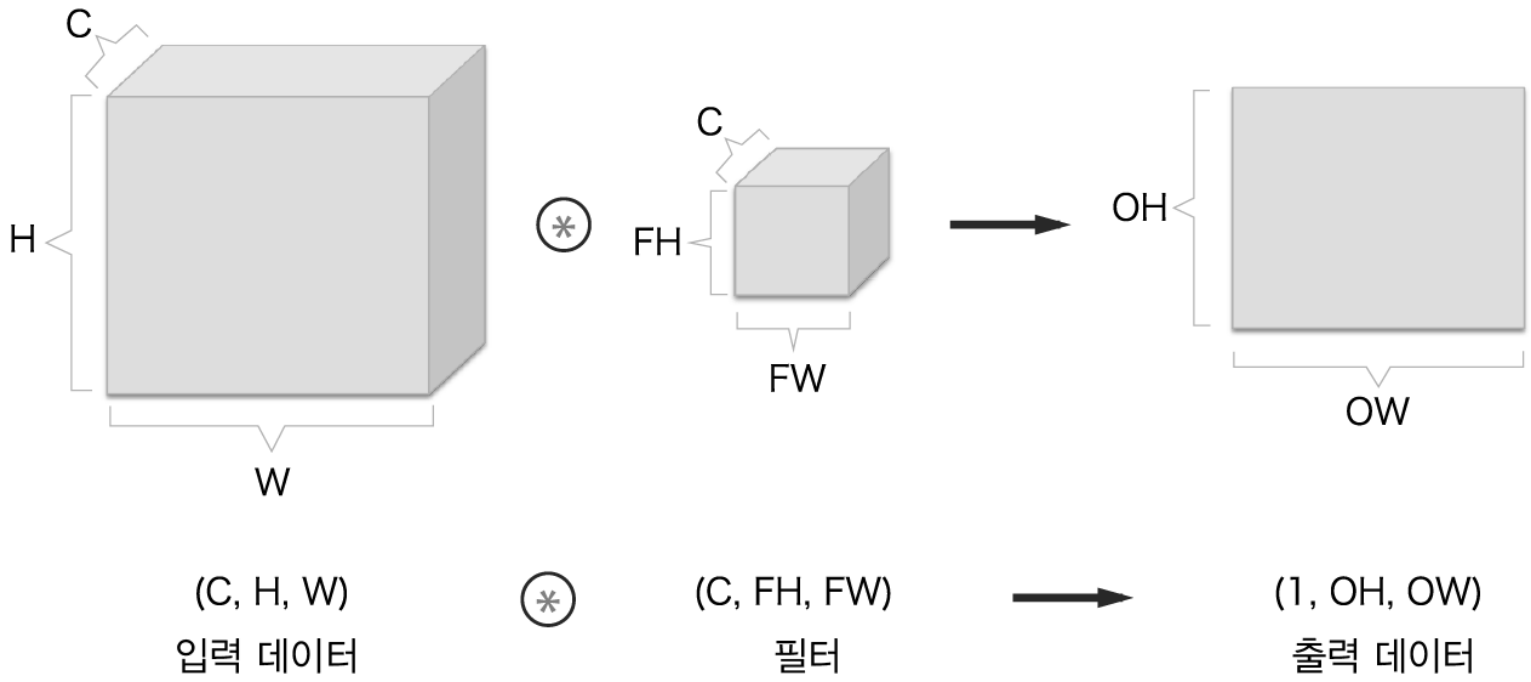
3차원 합성곱 과정

: 동일한 필터가 이미지 전체에 적용



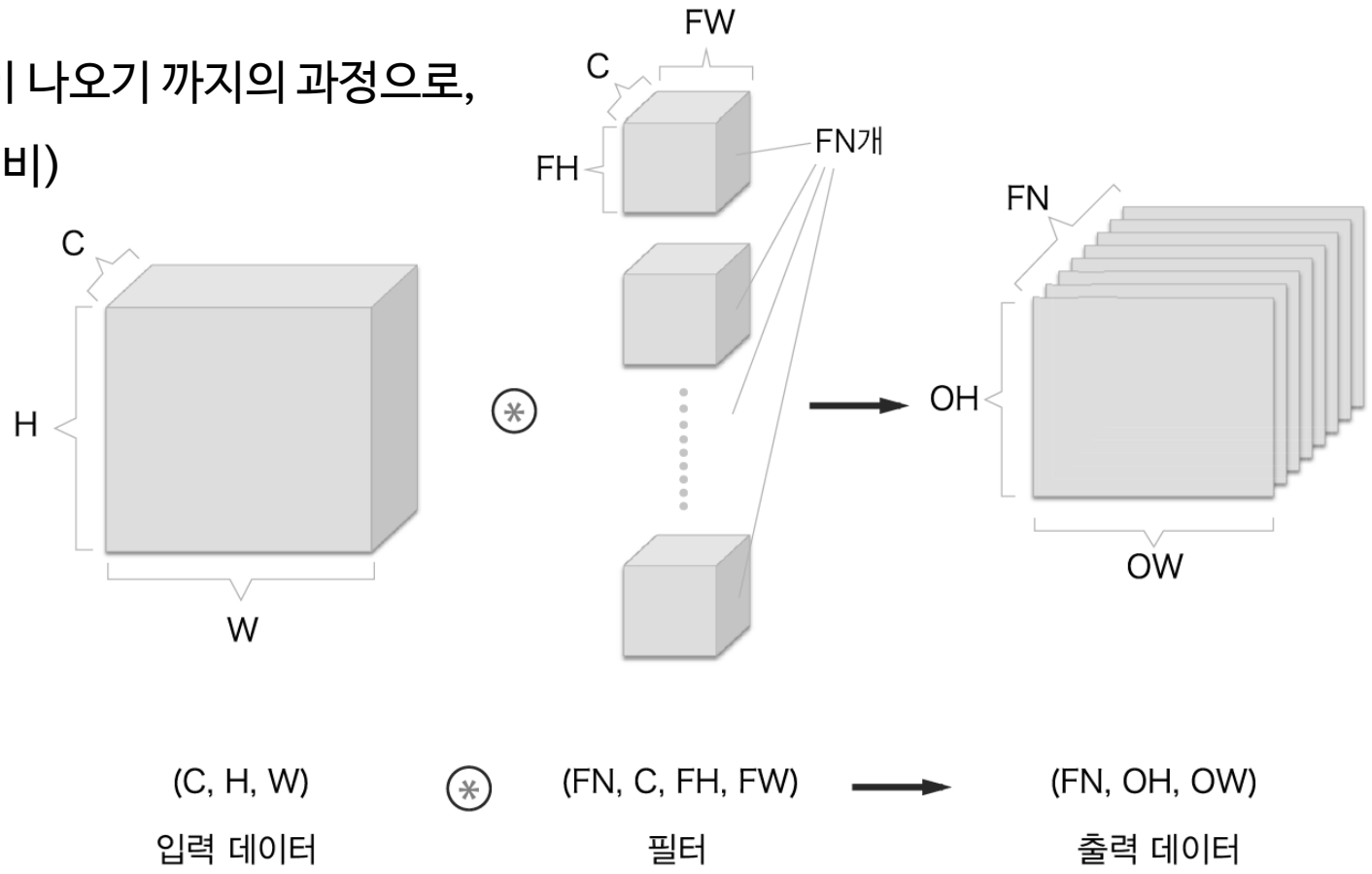
하나의 필터를 사용한 합성곱 연산

: 필터의 형식은 (채널, 높이, 너비)로 나타내며, 한 장의 특징 맵(Feature map)이 나오기 까지의 과정이다.



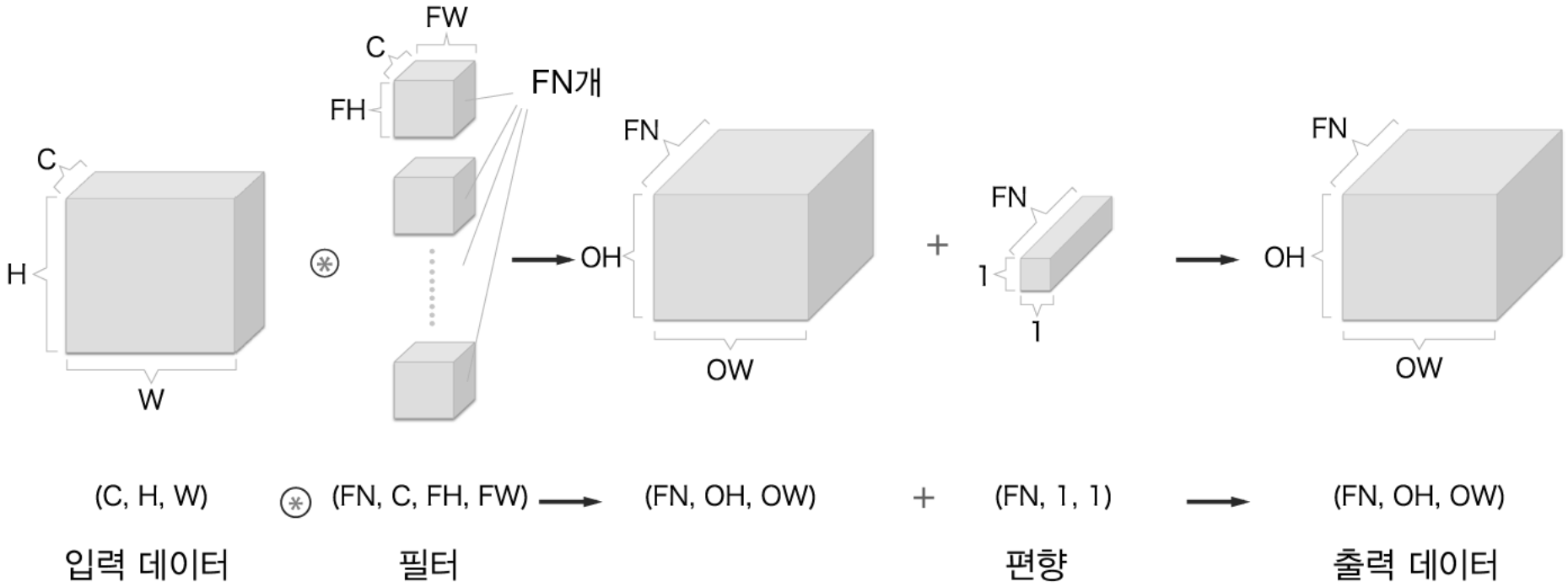
여러 필터를 사용한 합성곱 연산

: 여러 장의 특징 맵(Feature map) 이 나오기 까지의 과정으로,
(출력 채널 수, 입력 채널 수, 높이, 너비)
의 형식으로 필터를 나타낸다.



합성곱 연산의 처리 흐름 + 편향

: 채널 하나에 값 하나씩으로 구성되어, 출력인 (FN, OH, OW) 블록의 채널의 원소 모두에 더해진다 (브로드캐스트).

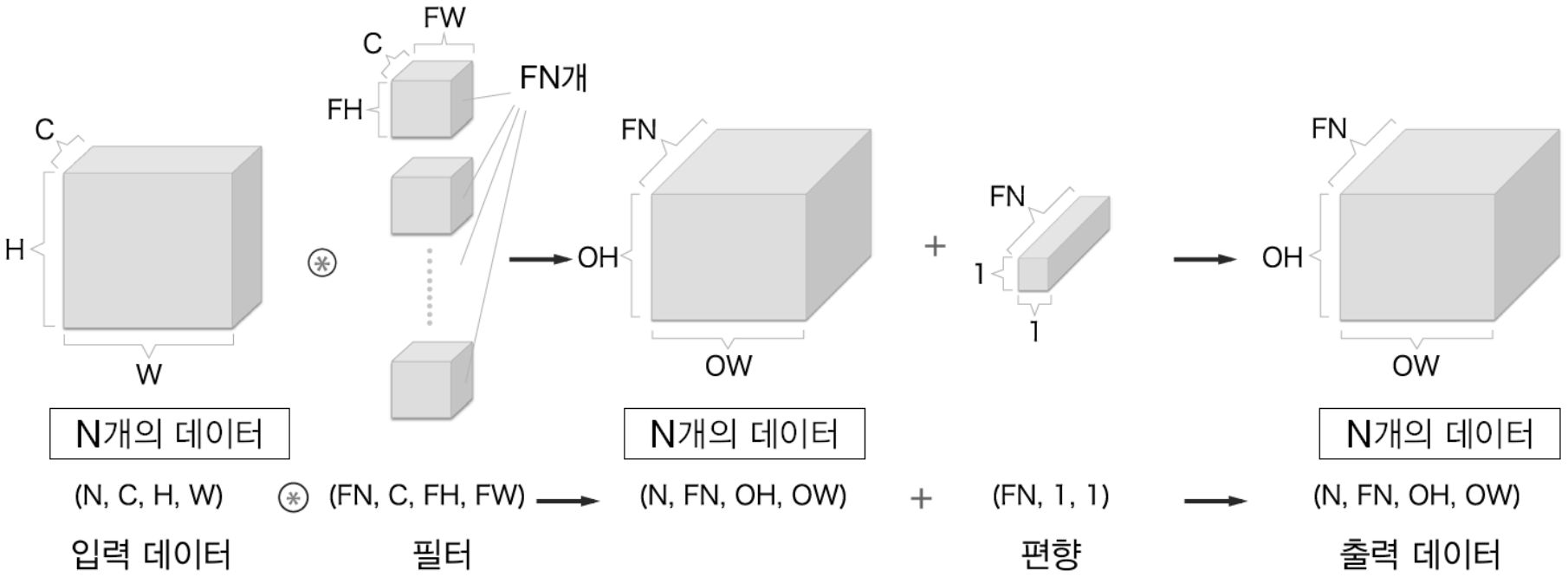


02 합성곱 계층

배치 처리

데이터 N개에 대한 합성곱 연산

: 배치 처리(각 계층을 흐르는 데이터의 차원을 하나 늘려 4차원 데이터(데이터 수, 채널 수, 높이, 너비)로 저장)를 통해, 학습을 효율적(4차원 데이터가 흐름으로써, 4회 분의 처리를 한번에 수행)으로 할 수 있다.



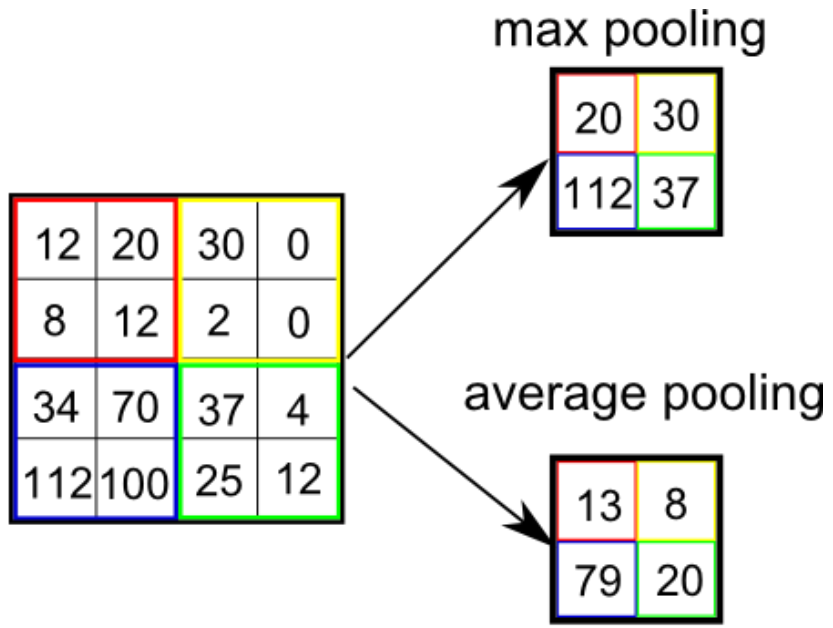
풀링(Pooling)

: 세로 ·가로 방향의 공간을 줄이는 연산으로, Sub-sampling이라고도 불린다.

주로, 풀링의 윈도우 크기와 스트라이드의 값은 같게 설정한다.

풀링 윈도우 - (2×2)

스트라이드 - 2

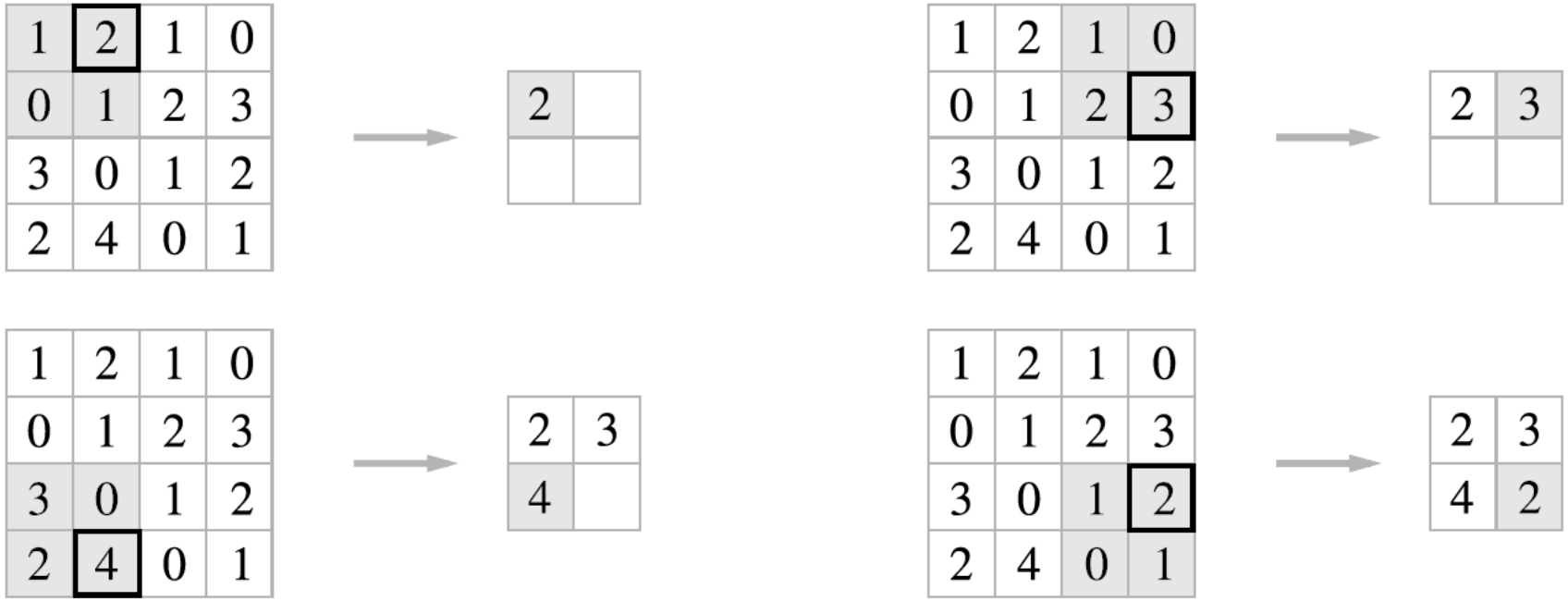


http://vaaaaaanquish.hatenablog.com/entry/2015/01/26/060622

풀링 계층의 특징

학습해야 할 매개변수가 없다

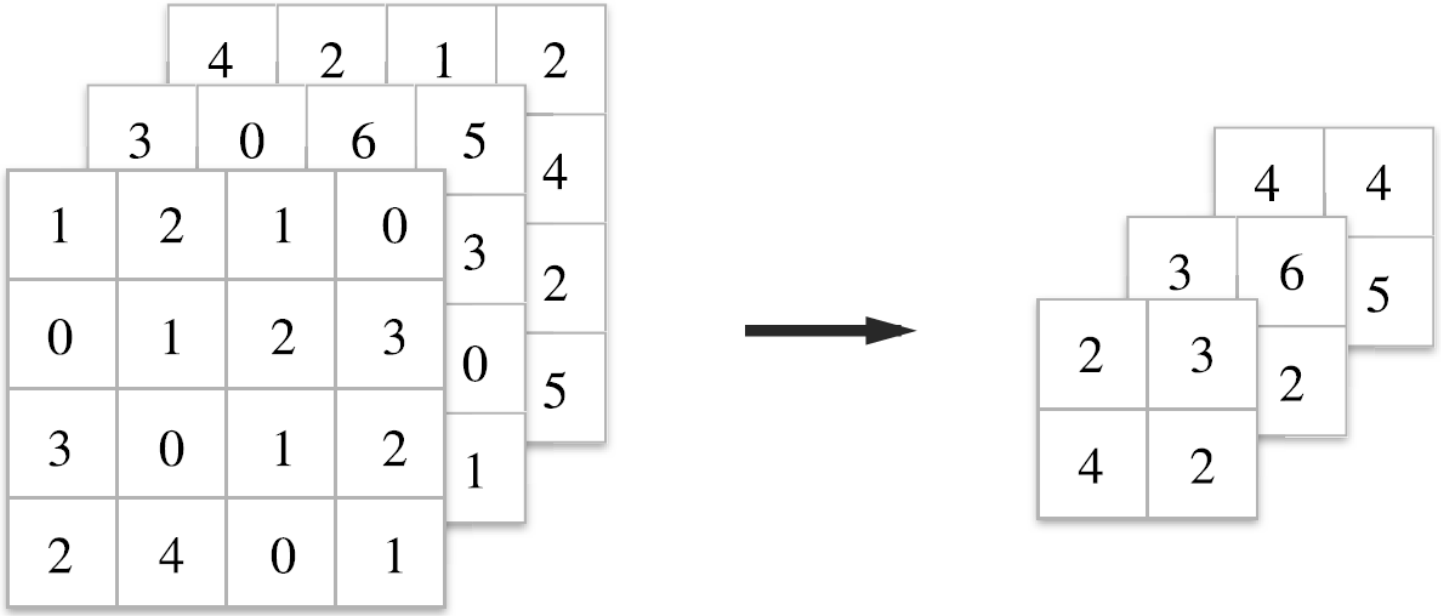
: 풀링은 대상 영역에서 최댓값이나 평균만을 취하는 명확한 처리이므로 학습해야 할 매개변수가 없다.



풀링 계층의 특징

채널 수가 변하지 않는다.

: 채널마다 독립적으로 계산하기 때문에, 입력 데이터의 채널 수 그대로 출력 데이터로 내보낸다.



입력 데이터 출력 데이터

03 풀링 계층

풀링 계층의 특징

입력의 변화에 영향을 적게 받는다 (강건하다)

: 입력 데이터가 조금 변해도 풀링의 결과는 잘 변하지 않는다.