Chap.07
합성곱 신경망(CNN)

<Deep Learning from scratch>

by Alfred.D.Cui

7.1 전체 구조

7.2 합성곱 계층

7.3 풀링 계층

7.4 합성곱/풀링 계층 구현하기

7.5 CNN 구현하기

7.6 CNN 시각화하기

7.7 대표적인 CNN

Lena Forsen (1972 PLAYBOY)

7.1 전체 구조

CNN (Convolutional Neural Network)

합성곱

- 7.2 합성곱 계층 Convolutional layer
- 7.3 풀링 계층 Pooling layer

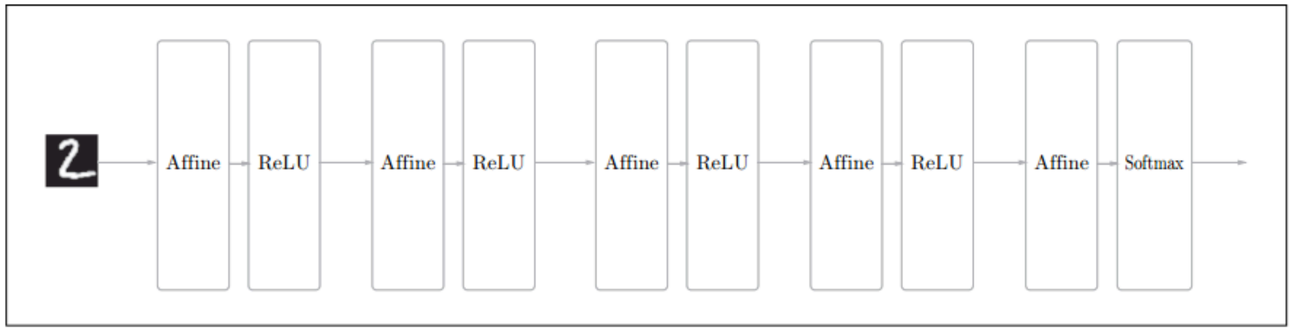


图7-1 基于全连接层（Affine层）的网络的例子

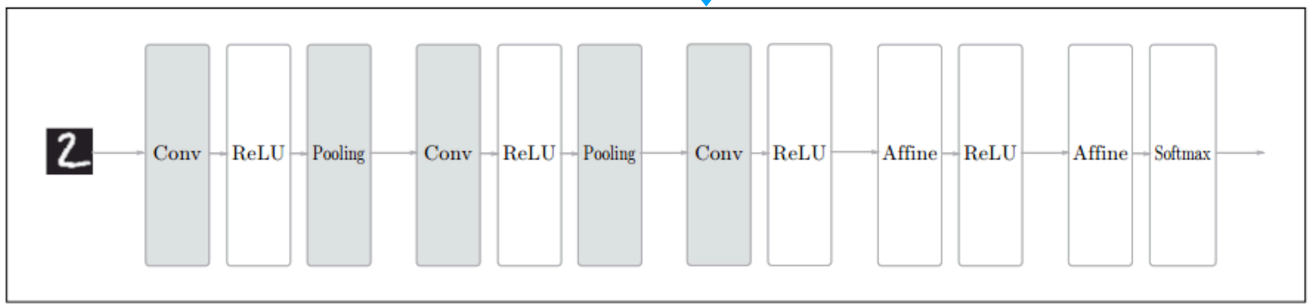


图7-2 基于CNN的网络的例子：新增了Convolution层和Pooling层（用灰色的方块表示）

7.2 합성곱 계층

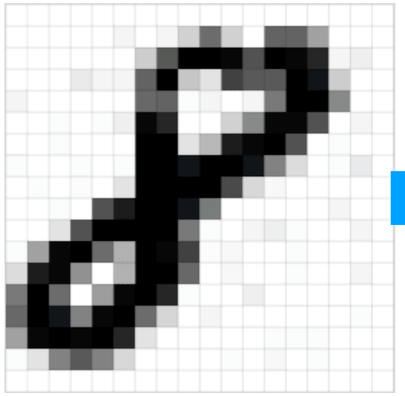
Convolutional Layer

이미지를 입력 -> 이미지의 특징을 출력

Sharpen by Photoshop

7.2.1 완전연결 계층(Affine 층) 의 문제점

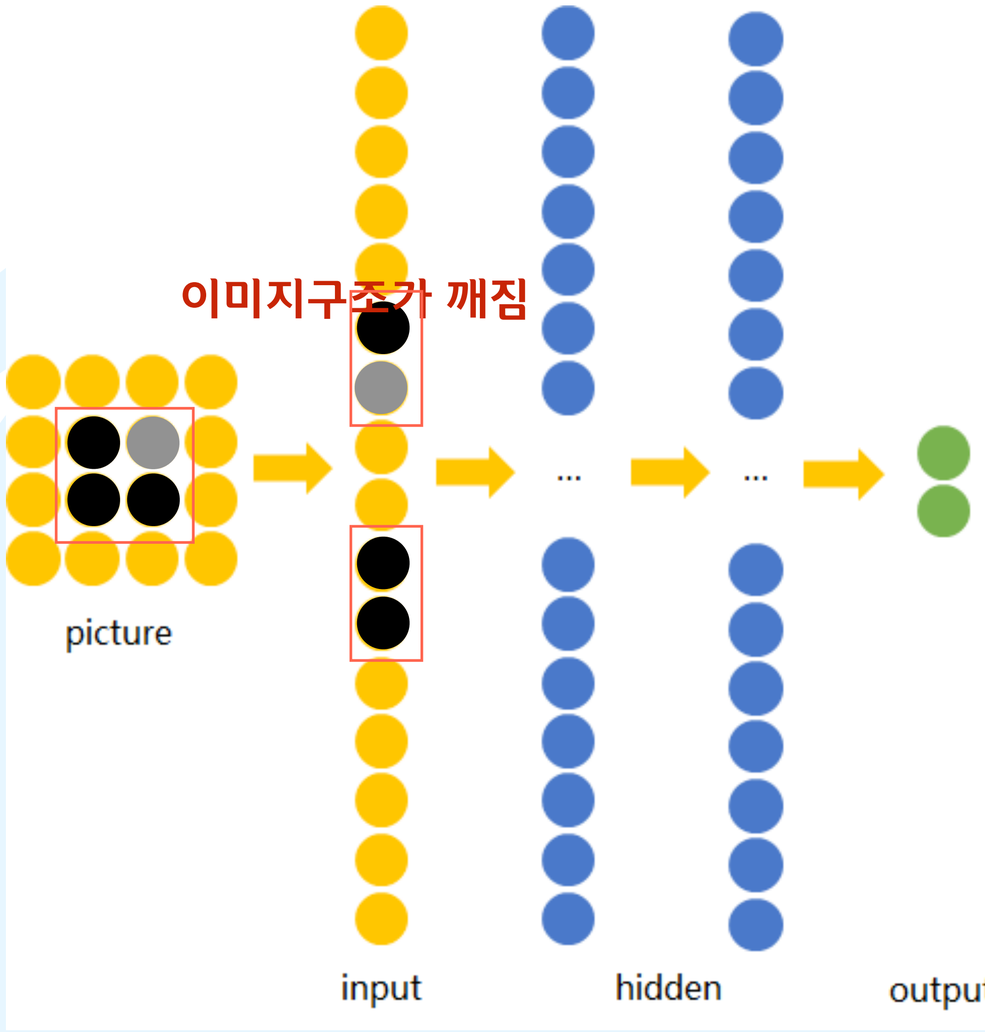
MNIST: Image => 28 * 28 Pixels => 784 Array



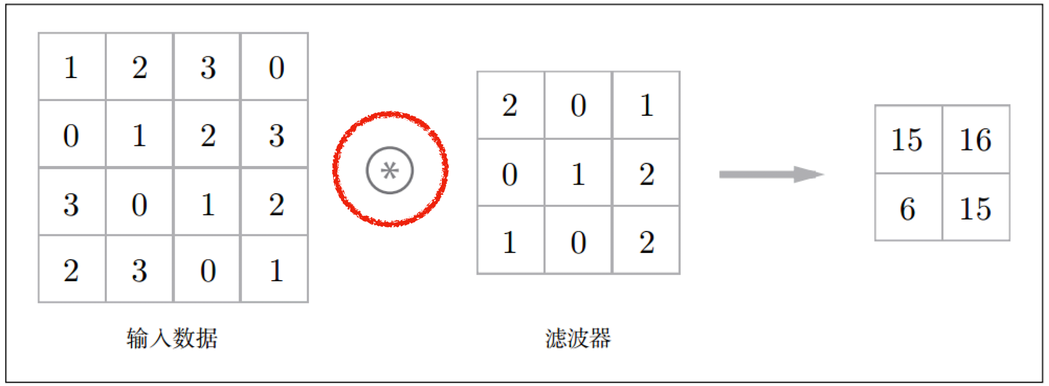
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
def load_mnist(normalize=True, flatten=True, one_hot_label=False):
```

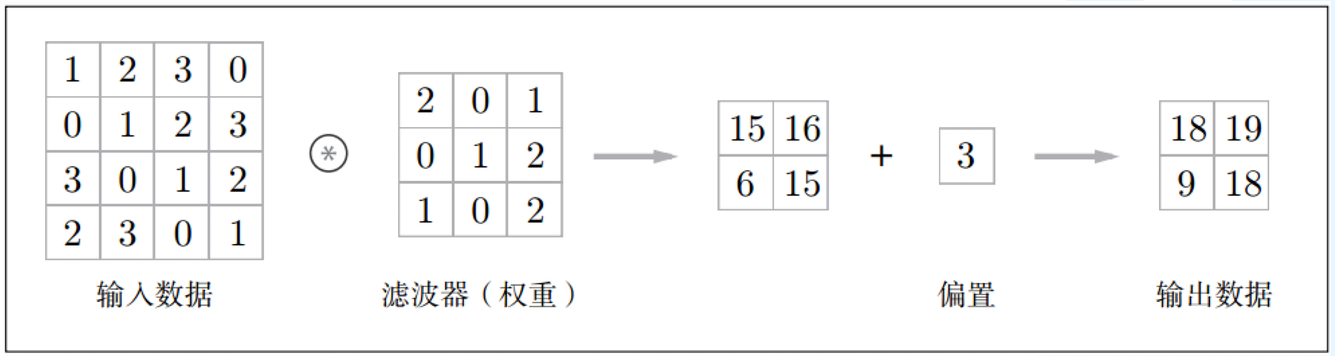
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 12 0 11 39 137 37 0 152 147 84 0 0 0

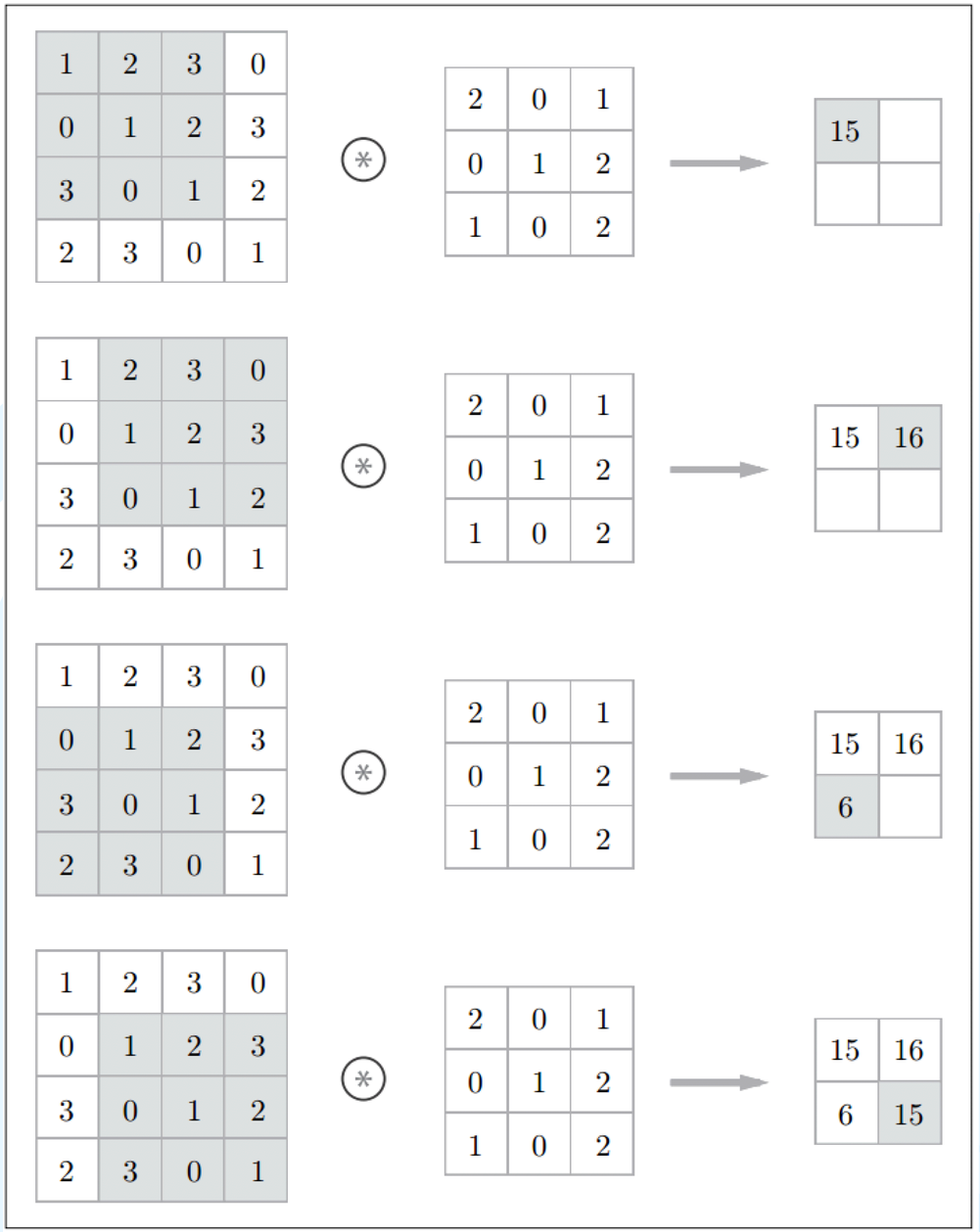


7.2.2 합성곱 연산

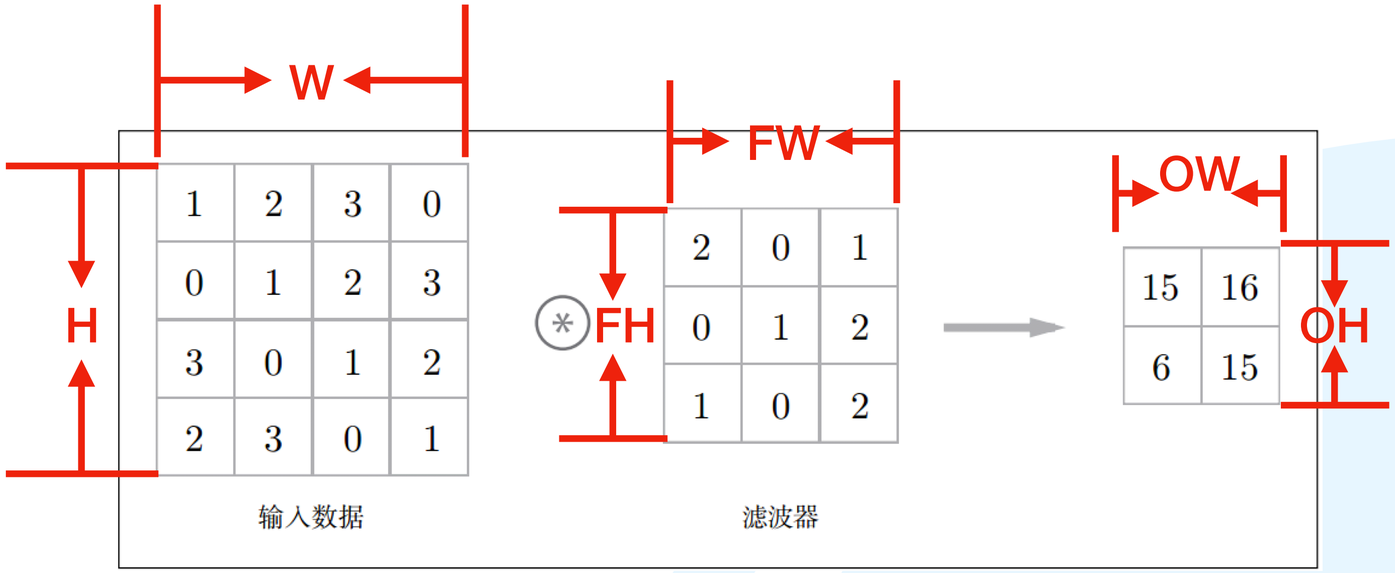


1*2 + 2*0 + 3*1 + 0*0 + 1*1 + 2*2 + 3*1 + 0*0 + 1*2
= 2 + 0 + 3 + 0 + 1 + 4 + 3 + 0 + 2
= 15





7.2.2 합성곱 연산



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

⊛

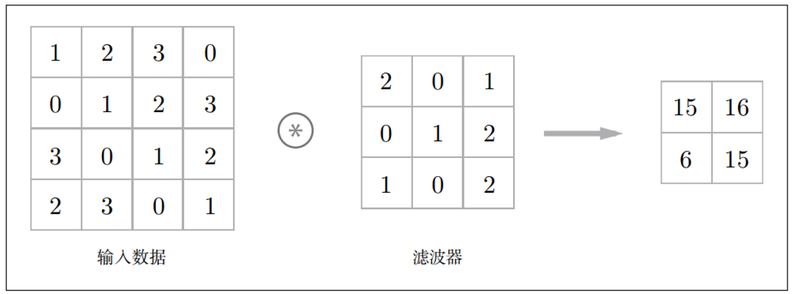
2	0	1
0	1	2
1	0	2

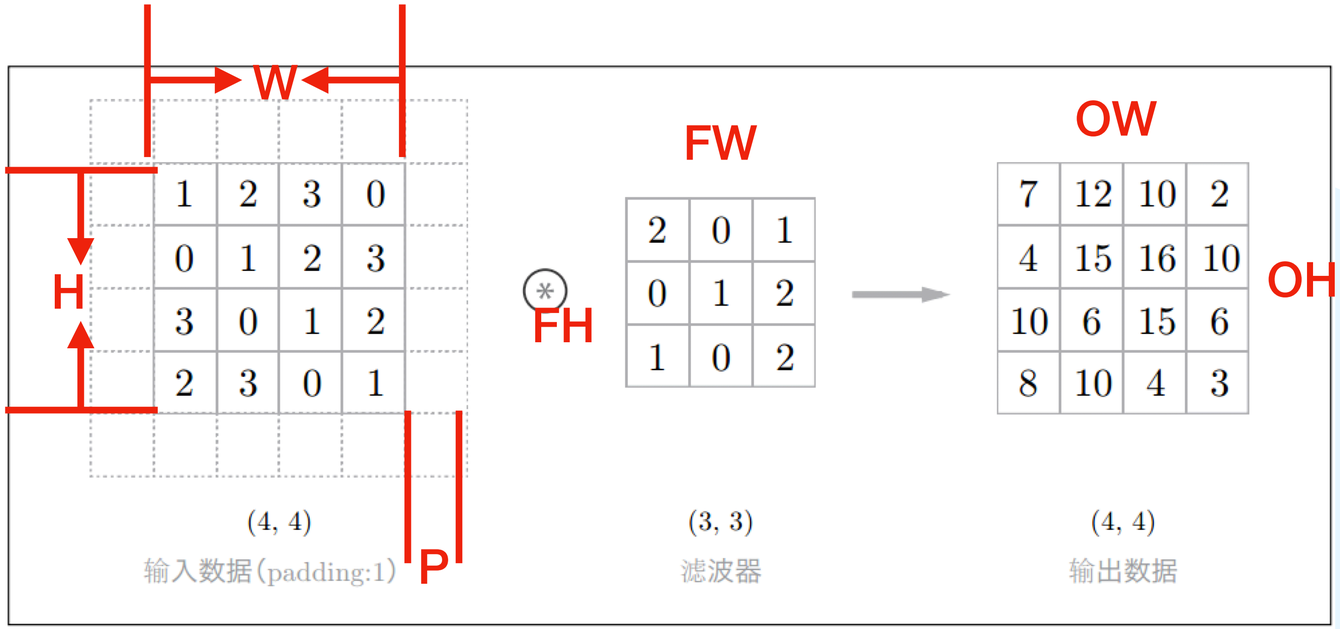
→

15		

H - FH + 1 = OH

7.2.3 패딩

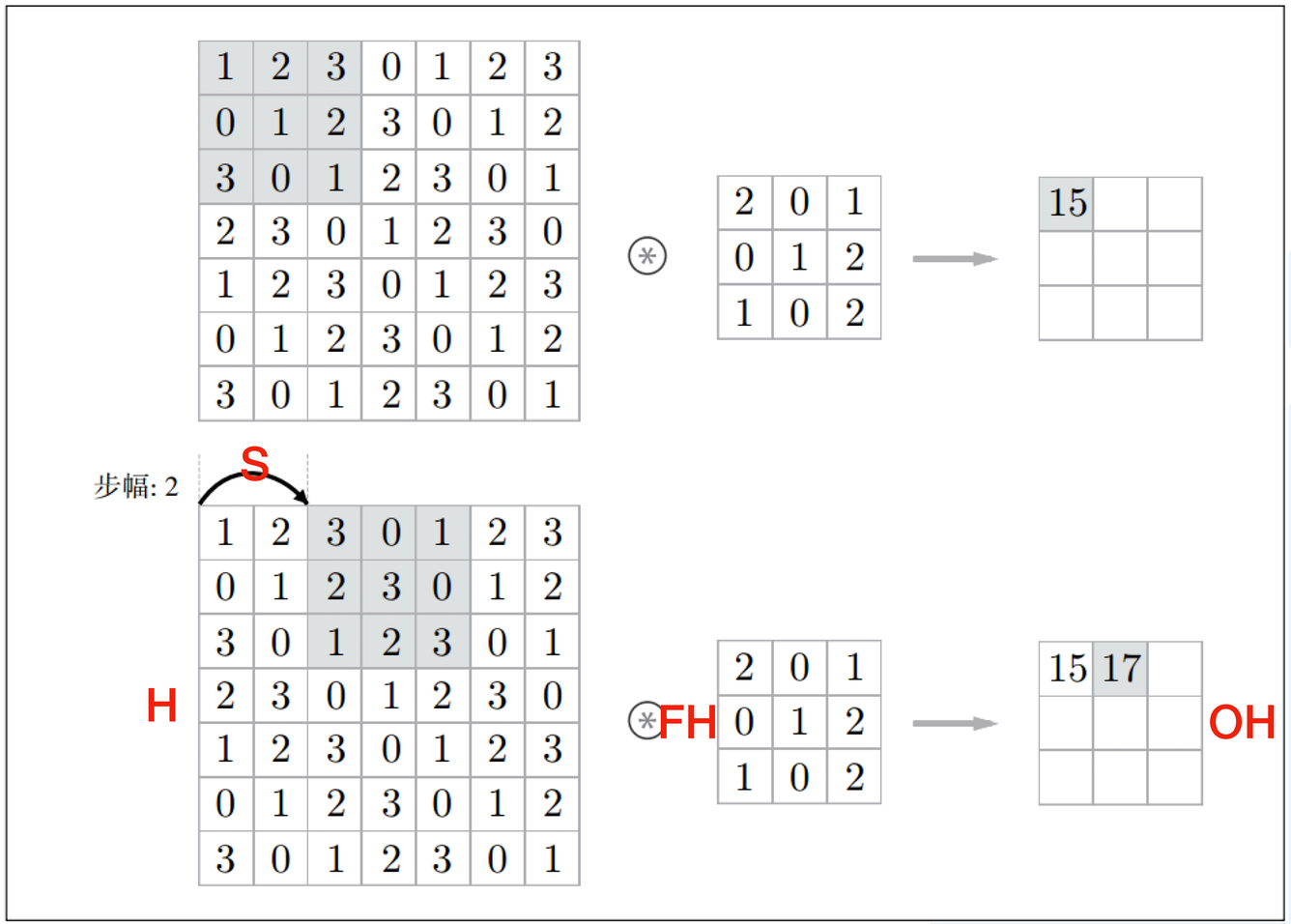




합성곱 연산의 출력 이미지는 작아진다, 여러번의 합성곱 연산을 하다보면, 이미지가 작아지면서 특징도 사라진다, 이를 막기 위해, 패딩이 필요하다.

H - FH + 1 = OH → H - FH + 1 + 2P = OH

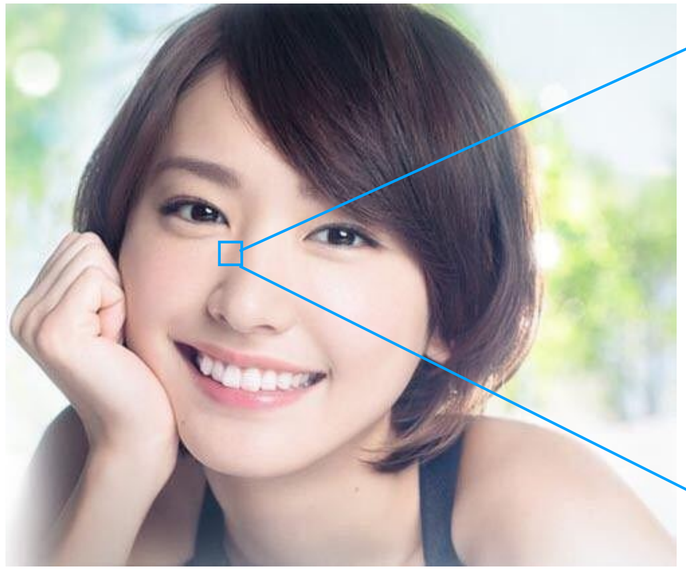
7.2.4 스트라이드 (보폭)

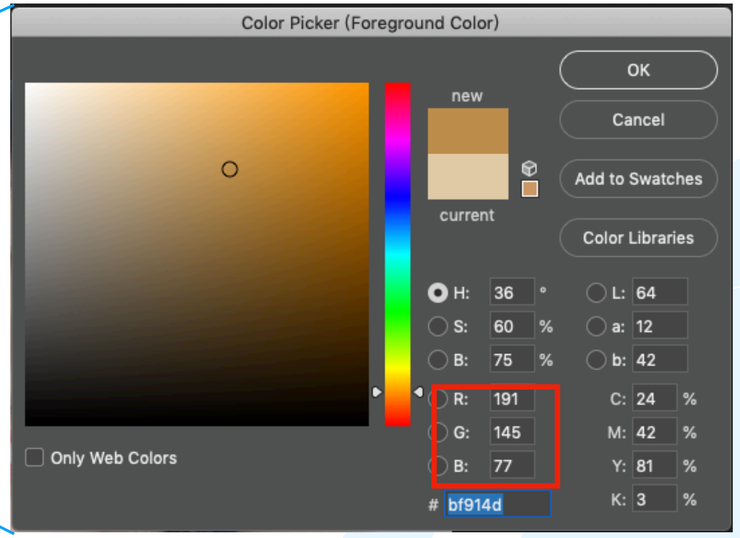


H - FH + 1 + 2P = OH

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

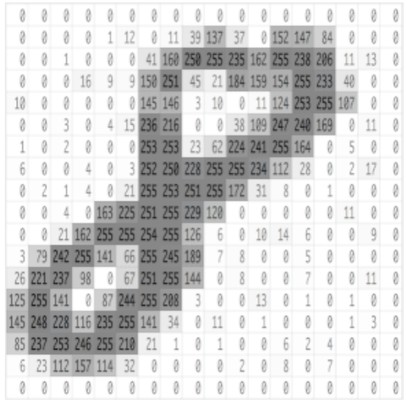
7.2.5 3차원 데이터의 합성곱 연산





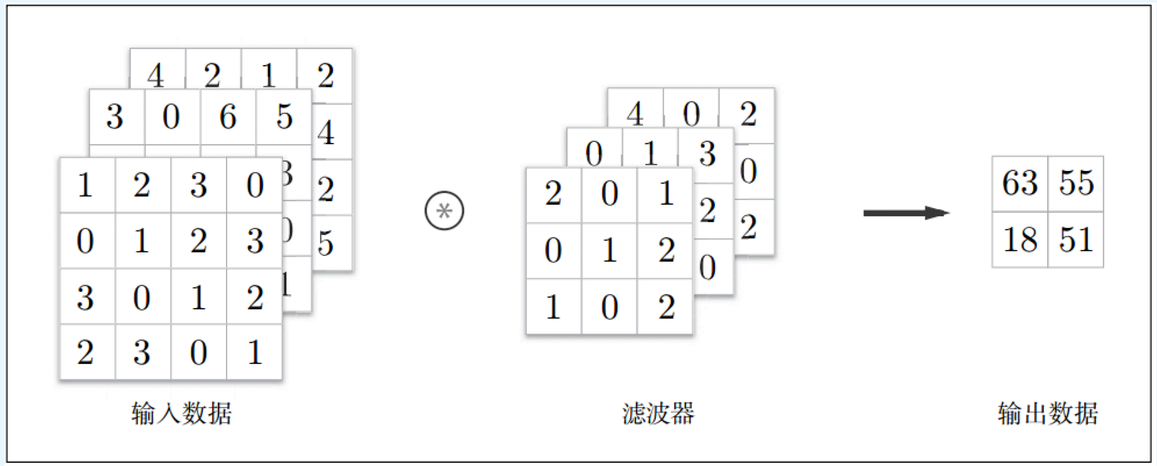
rgb(191, 145, 77)

rgba(191, 145, 77, 100)

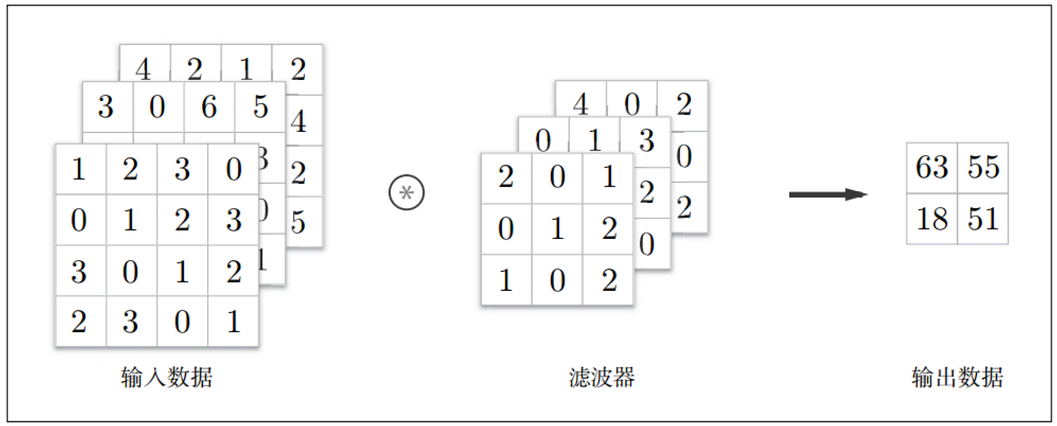


컬러 이미지의 픽셀은 RGB 세개의 색상의 값을 가지고 있다, 실제 이미지 처리 시, 각 색상의 값만 도출해 서 해당색상의 채널을 만든다.
하나의 컬러 이미지가 R,G,B 세개의 채널로 조합된다.

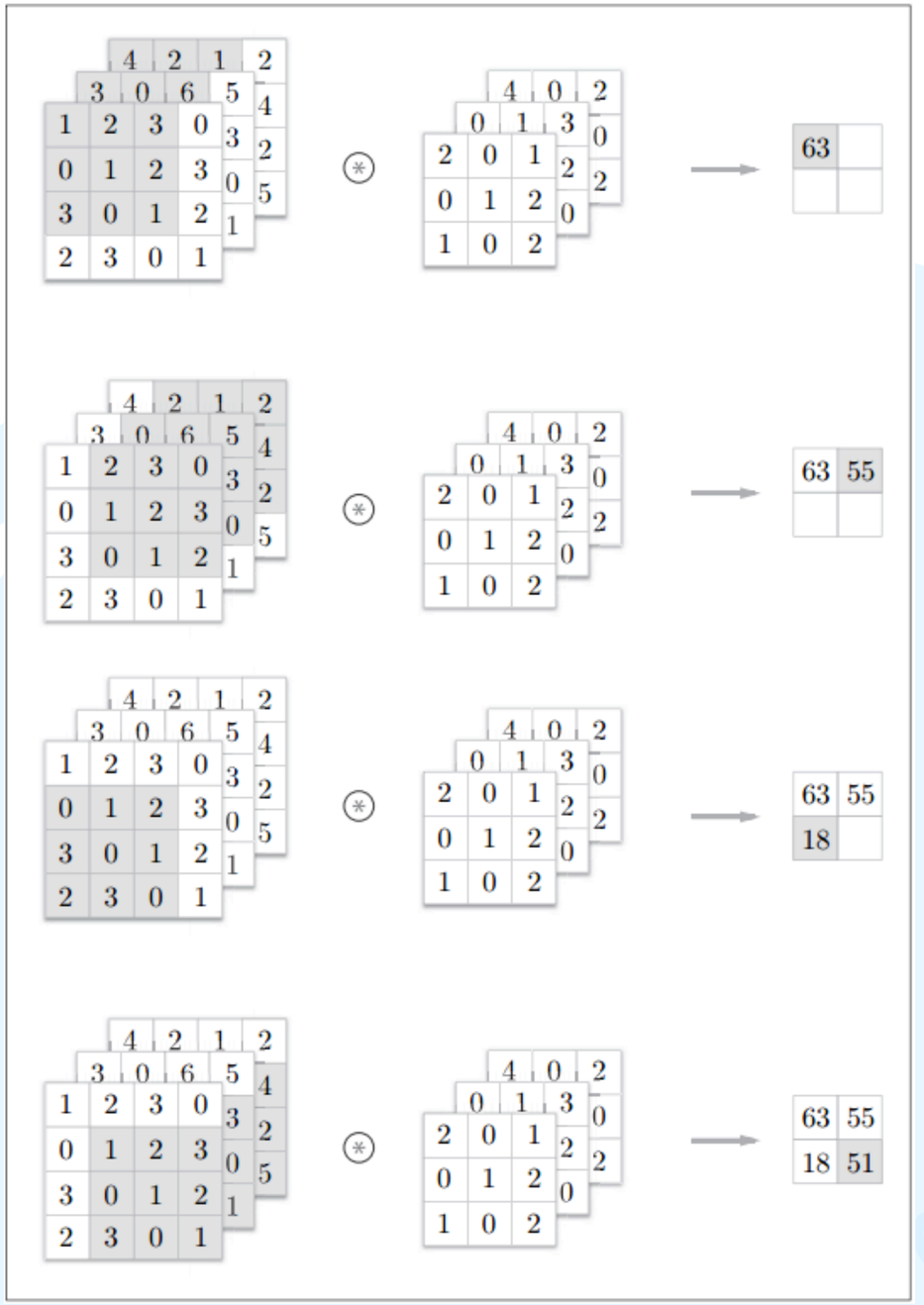
7.2.5 3차원 데이터의 합성곱 연산



7.2.5 3차원 데이터의 합성곱 연산



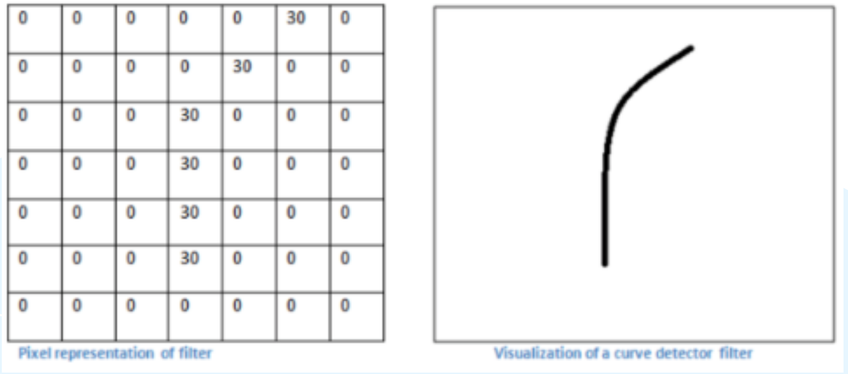
다 차원 합성곱 연산시, 각채널과 해당채널의 필터의 합성곱 값을 다시 더한다.
주의해야 할 점은, 필터의 사이즈는 전부 일치하고, 개수도 입력데이터와 같아야 한다.

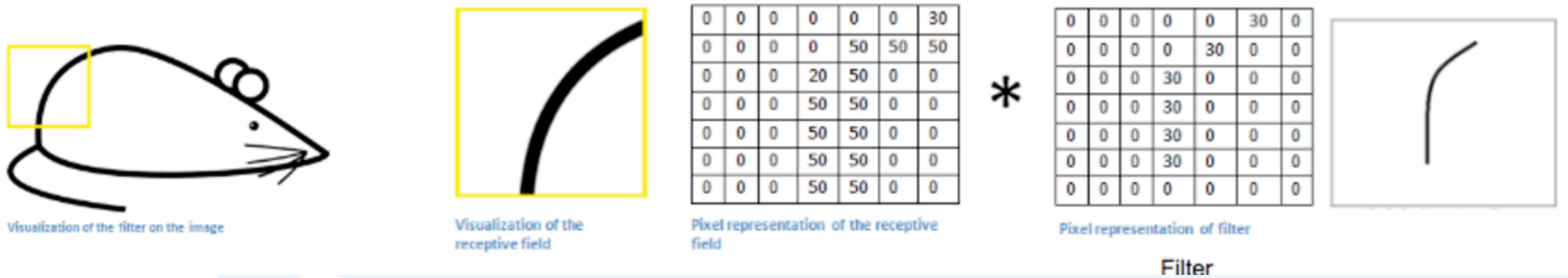


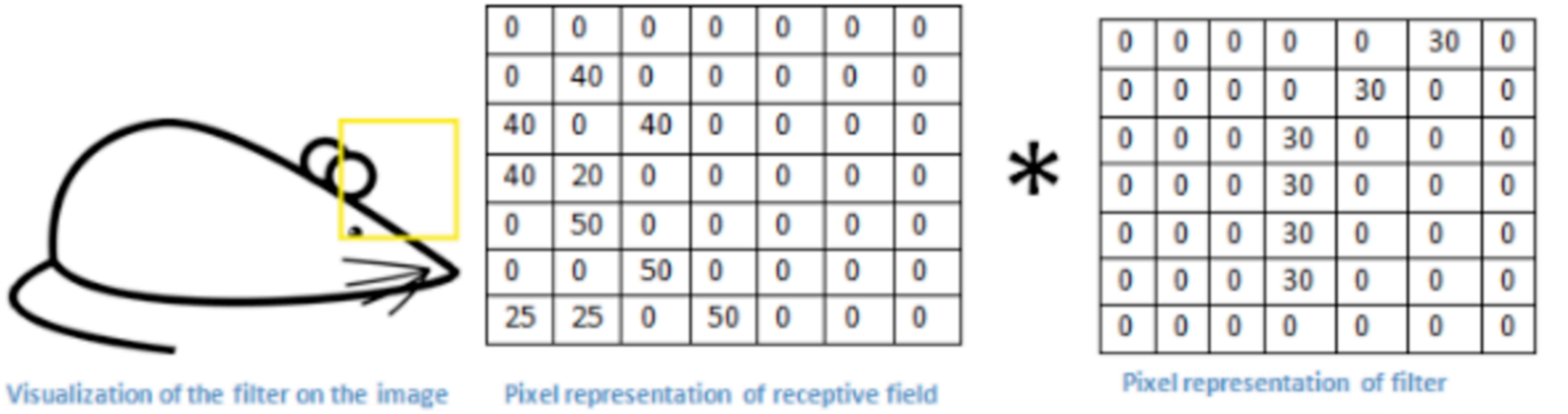
7.2.x 이쯤에서 드는 질문: 필터 넌 대체 뭐야?

weight의 matrix

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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

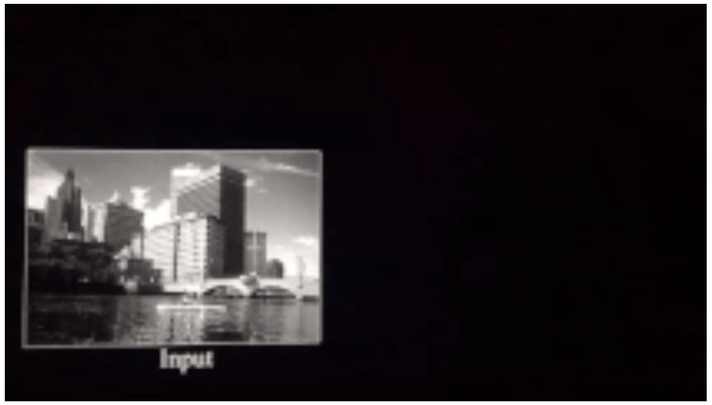






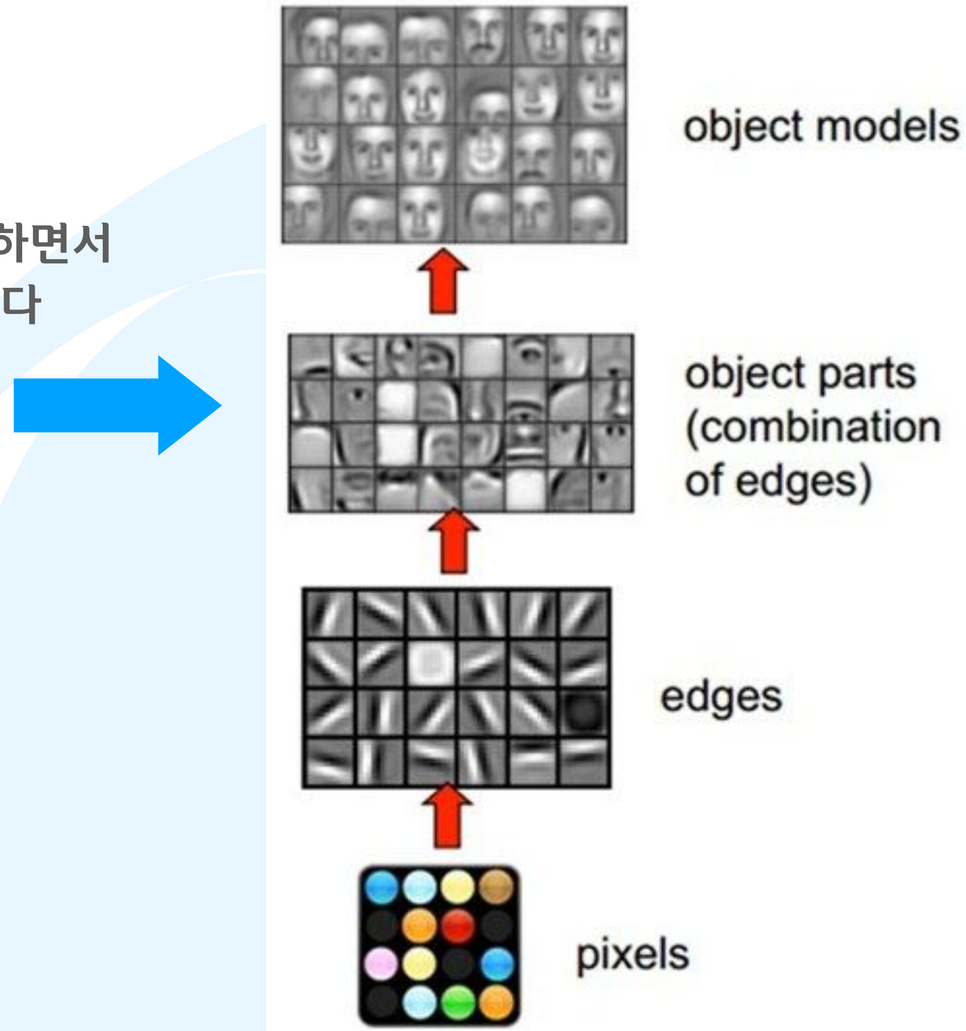
7.2.x 이쯤에서 드는 질문: 필터 넌 대체 뭐야?

weight의 matrix

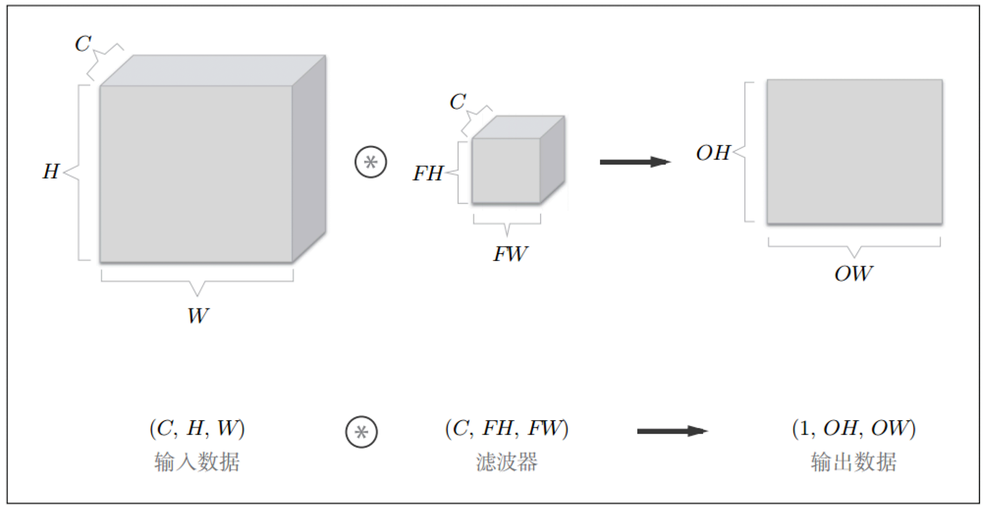


두개의 Filter 로 이미지를 관찰한 결과

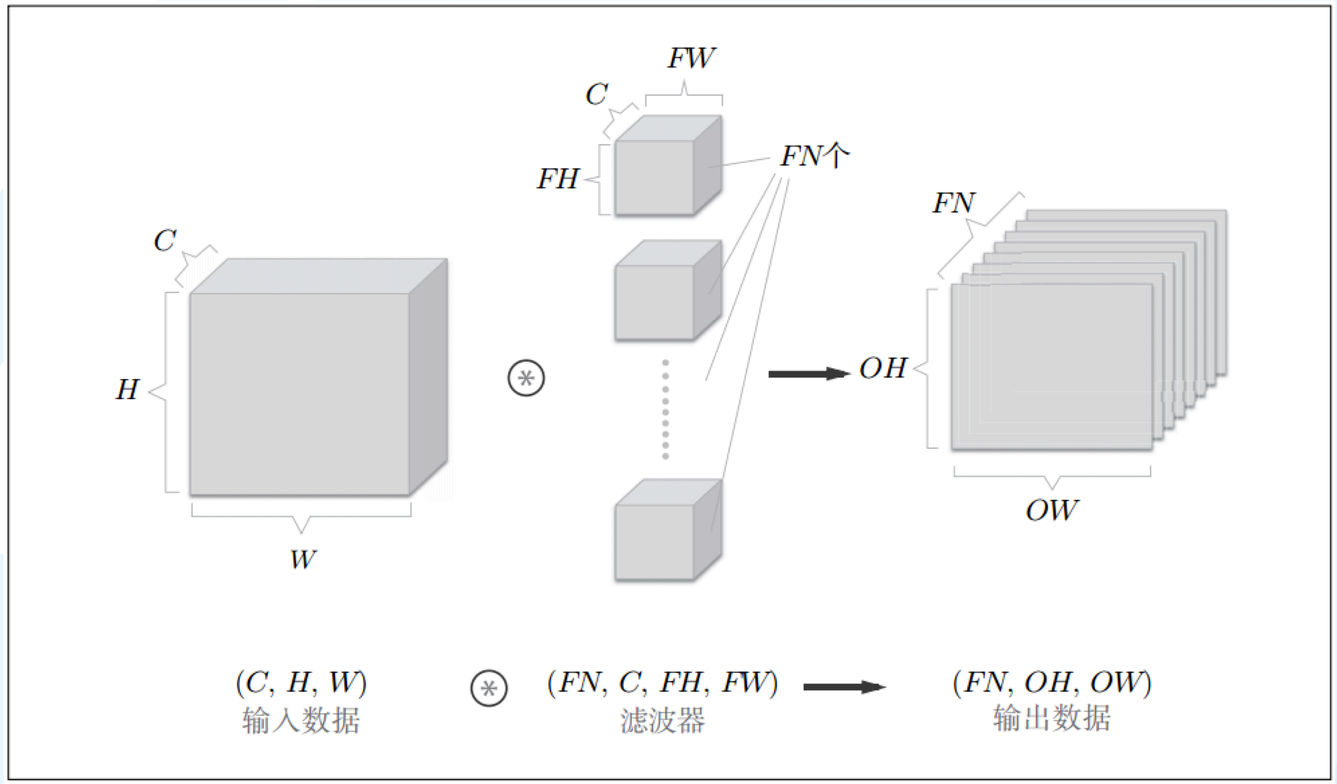
Filter 는 CNN 진행하면서
Weight 처럼 진화한다



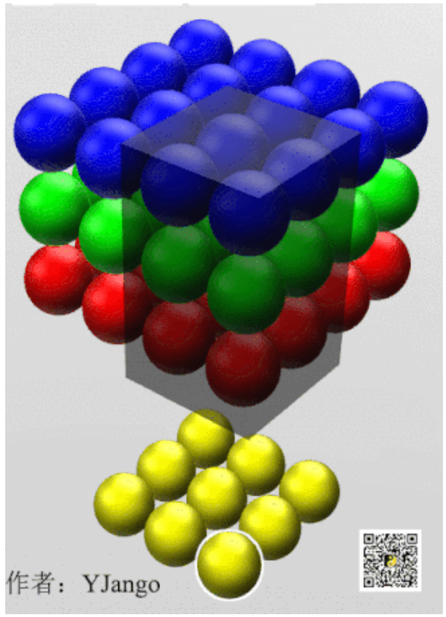
7.2.6 블록으로 생각하기

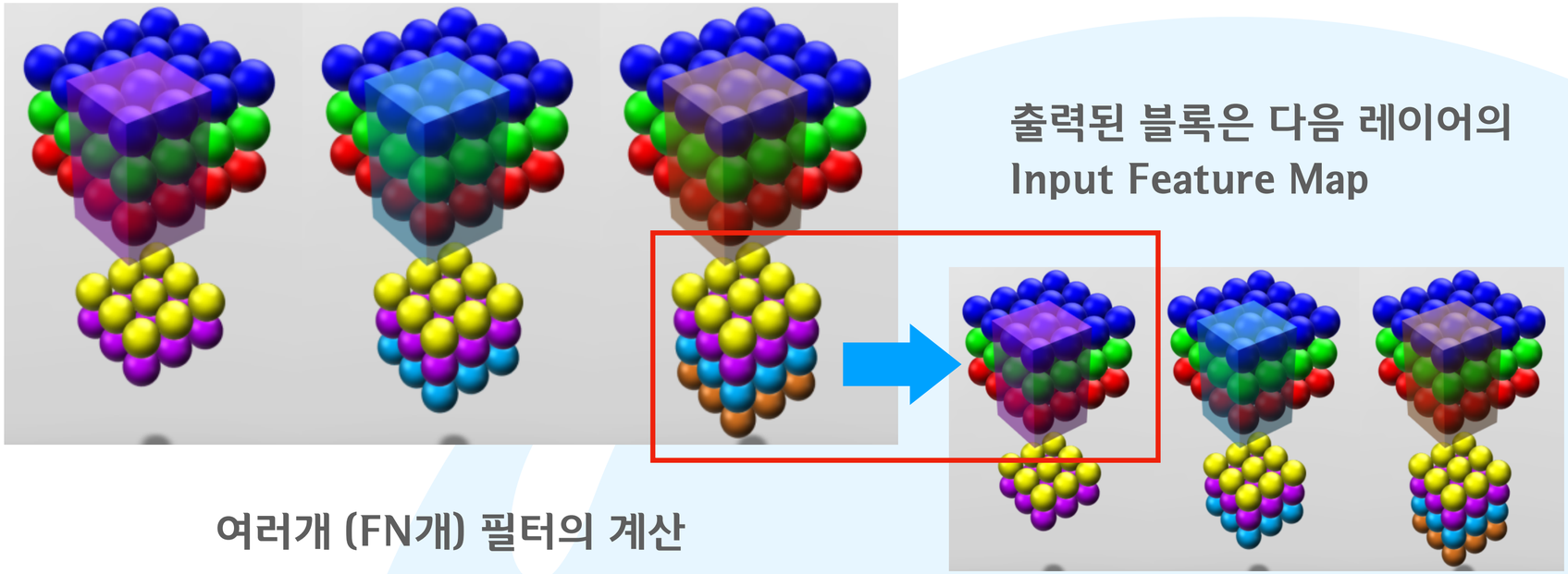


여러개 (FN개) 필터의 계산

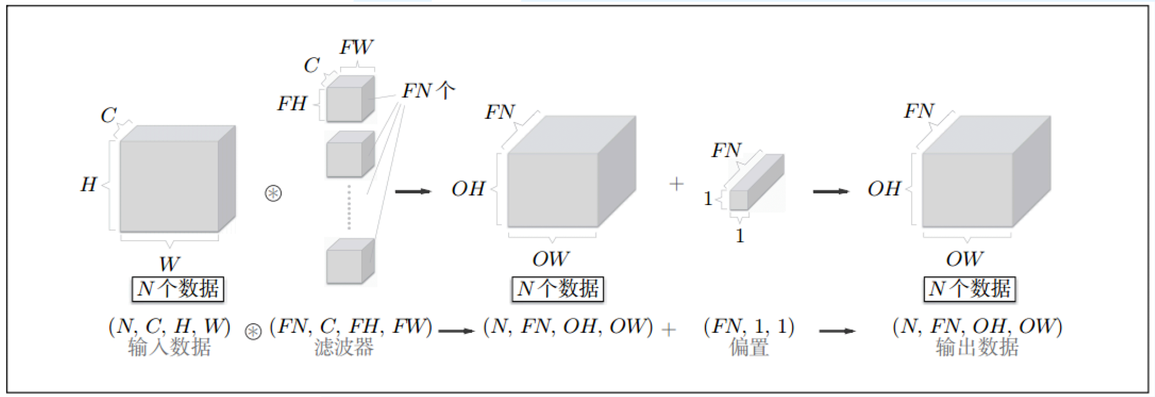


7.2.6 블록으로 생각하기

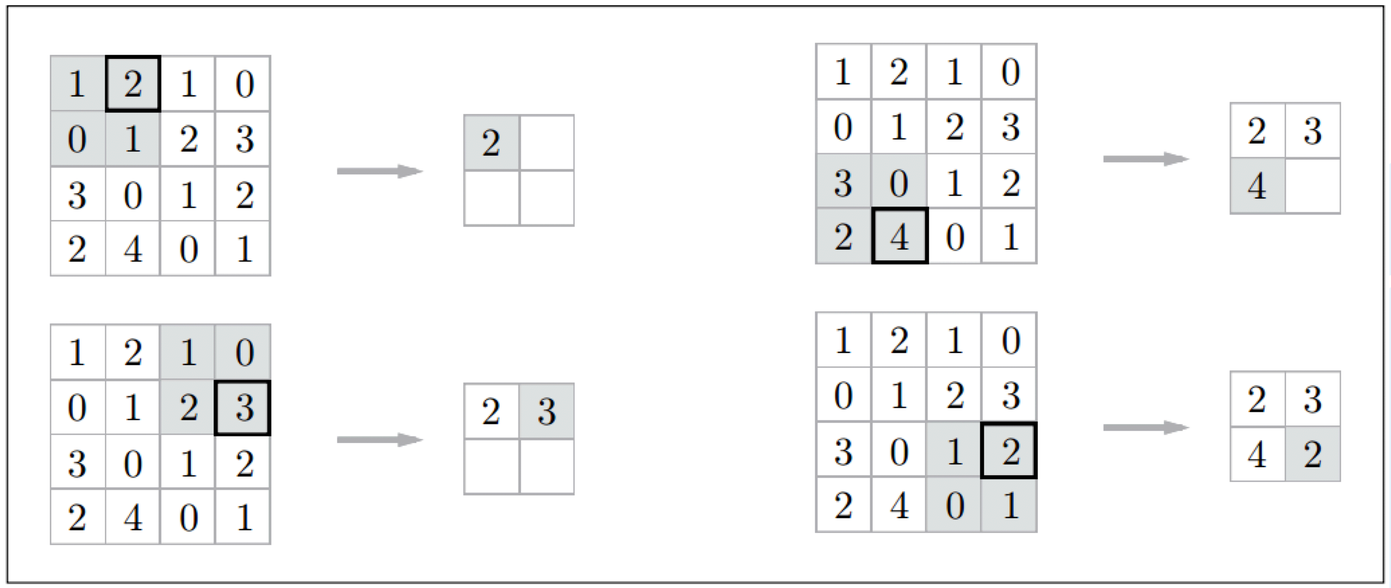




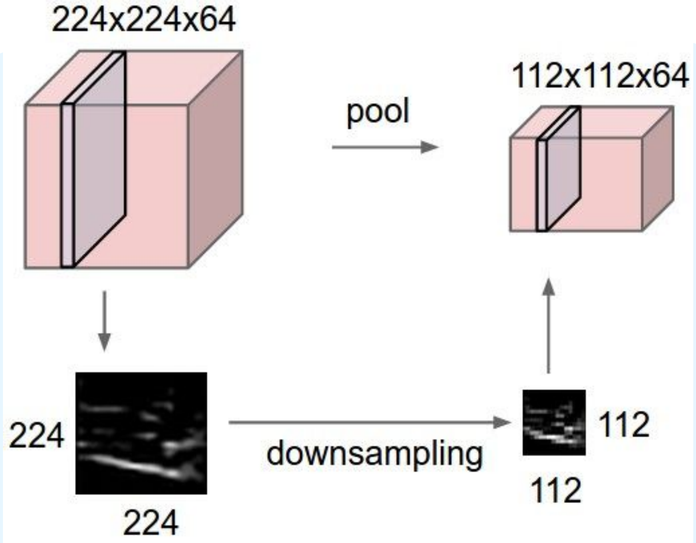
7.2.7 배치 처리



7.3 풀링 계층



max pooling

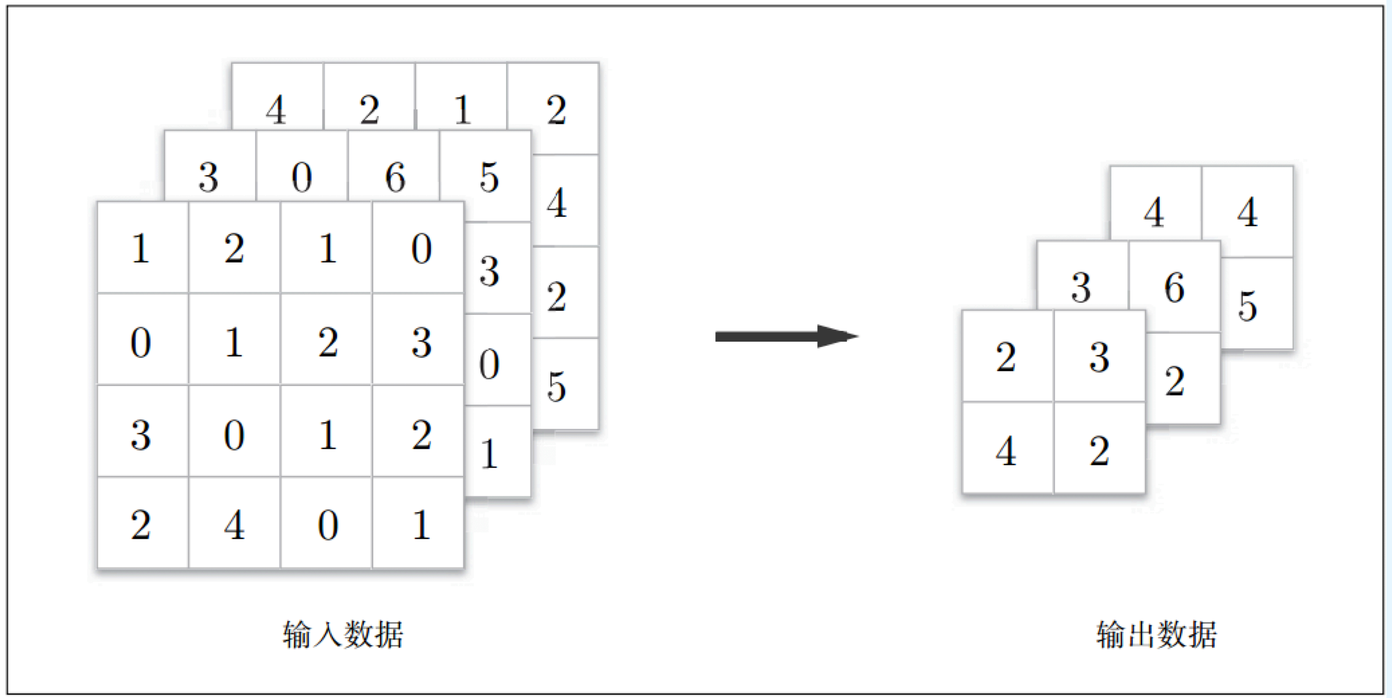


우리는 고해상도 이미지에서 물체를 인식할 수도 있지만, 저해상도 이미지에서도 물체를 인식한다. 컴퓨터가 계산을 할때 고해상도이미지일 수록 계산할 픽셀이 많아진다. 이것 뿐만 아니라 고해상도 일수록, 필요없는 특징들도 많아 진다. 그러므로 이미지의 픽셀을 줄일 필요가 있다. 이것이 Sub sampling 이고 Pooling 이라고 부르기도 한다.

max pooling, average pooling

풀링레이어는 학습 할 파라미터가 없다

풀링레이어를 거친 Feature Map 의 채널수는 변경하지 않는다



7.4 합성곱/풀링 계층 구현하기

7.4.1 4차원 배열

4차원

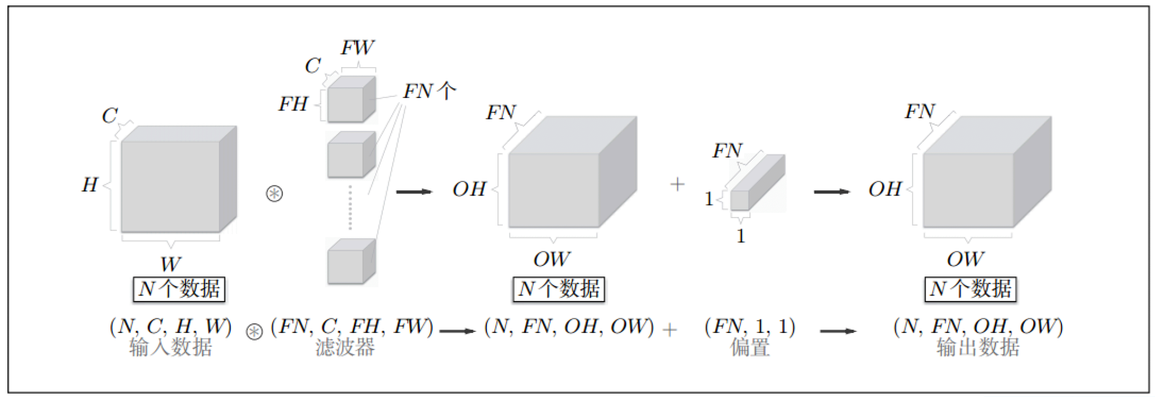
4차원 배열

- 끝 -

7.4 합성곱/풀링 계층 구현하기

7.4.1 4차원 배열

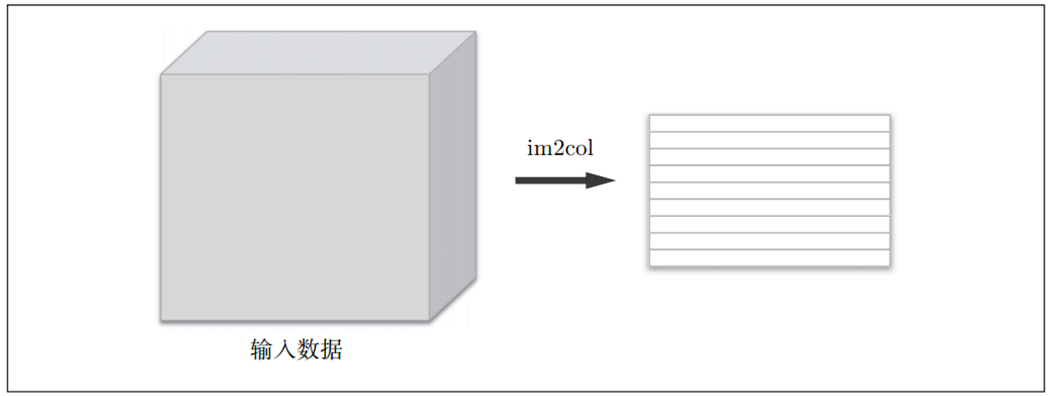
CNN 각 레이어간 전달되는 데이터는 4차원 배열이다.
(개수, 필터수, pixel height, pixel width)

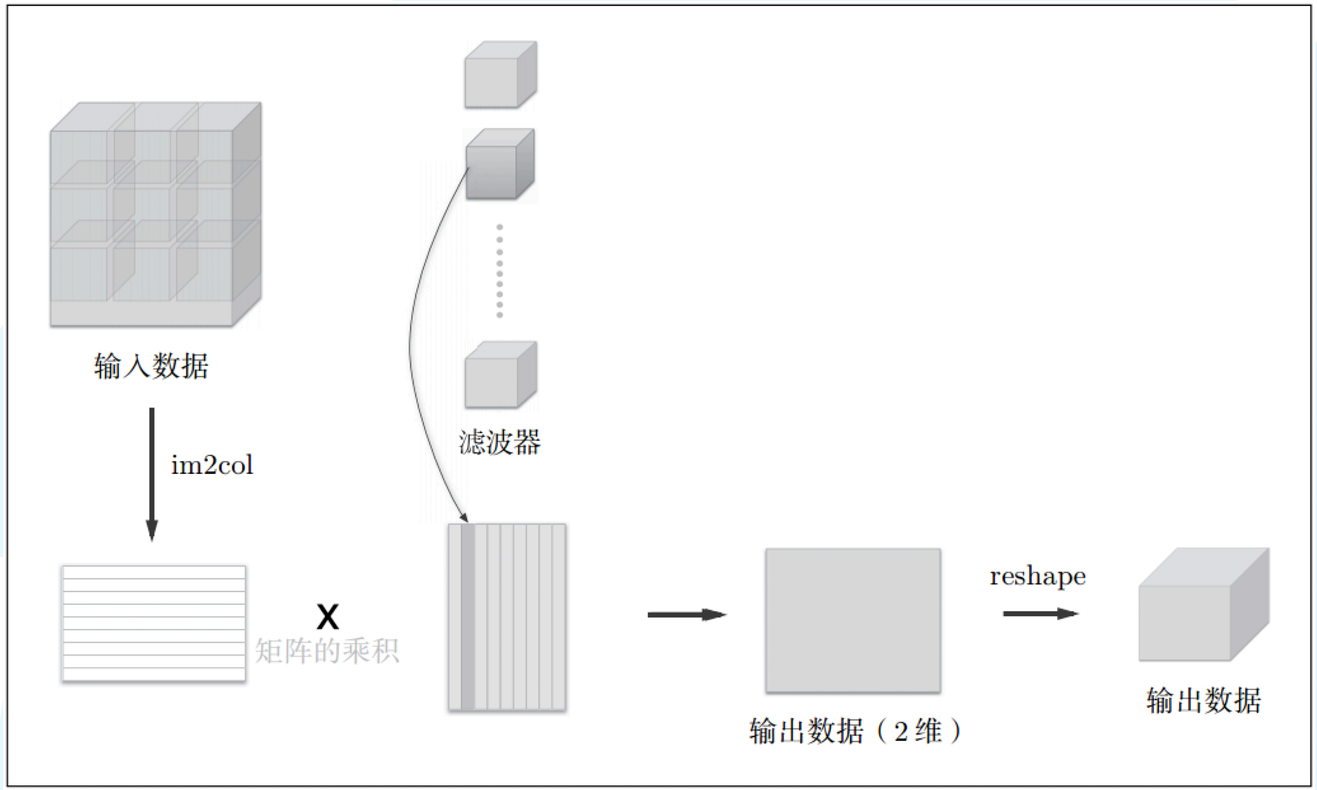



```
>>> x = np.random.rand(10, 1, 28, 28) # 随机生成数据
>>> x.shape
(10, 1, 28, 28)
```

7.4.2 im2col로 데이터 전개하기

im2col(input_data, filter_h, filter_w, stride=1, pad =0)





卷积运算的滤波器处理的细节：将滤波器纵向展开为1列，并计算和im2col展开的数据的矩阵乘积，最后转换(reshape)为输出数据的大小

여러분, 다음주도 힘차게!