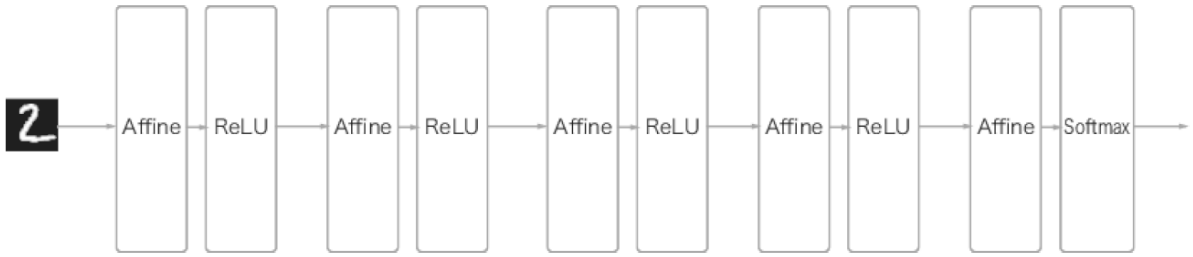
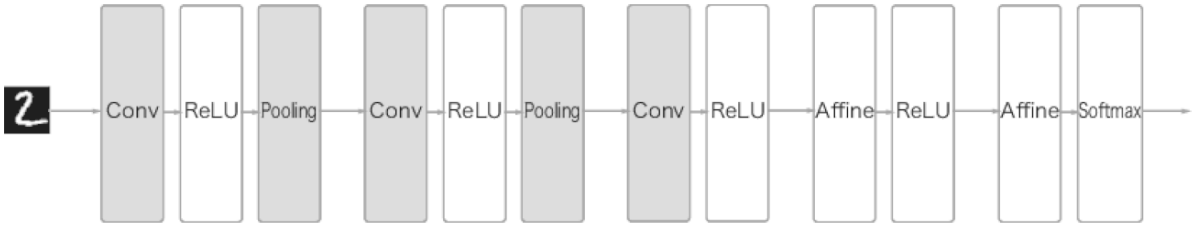
chapter 7. 합성곱 신경망 (convolutional neural networks: CNN)

7.1 전체 구조

합성곱 계층 (convolution layer), 풀링 계층 (pooling layer),

Affine 계층 (완전 연결)





7.2 합성곱 계층

convolution 연산, 패딩(padding), 스트라이드(stride), 채널(channel)

7.2.1 완전연결 계층의 문제점

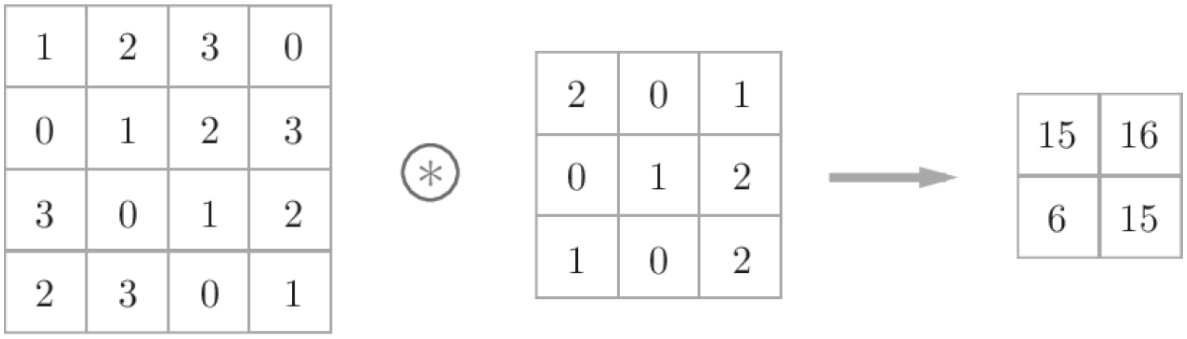
데이터의 형상이 무시 : 가로,세로,채널을 1차원으로 변환

공간적으로 가까운 픽셀, RGB 각 채널은 밀접 관련 → 1차원 어떤 위치인지 동등하게 처리하는 문제점

합성곱 계층은 형상을 유지

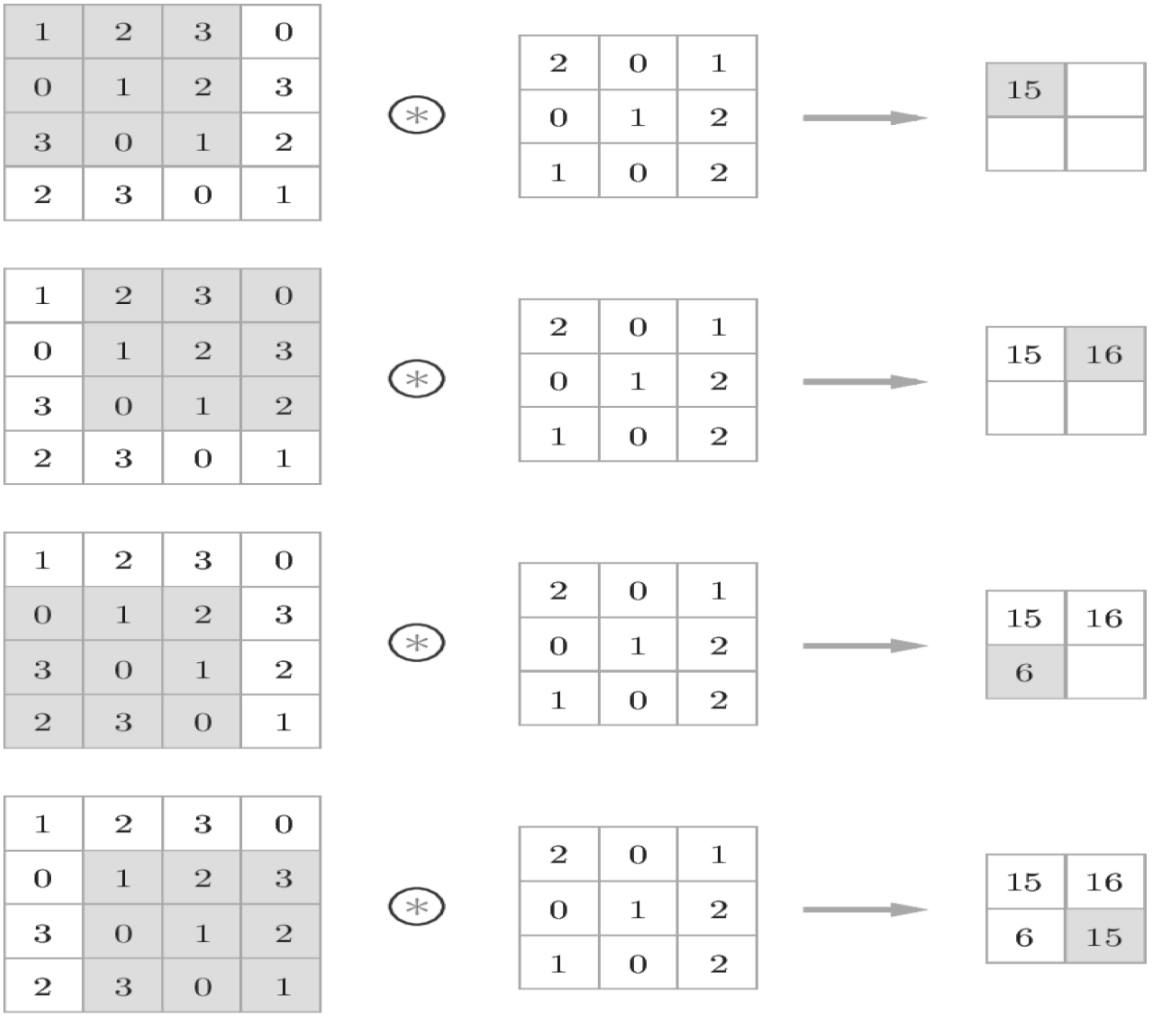
입출력 데이터를 입출력 특징, feature map 이라고도 함

7.2.2 합성곱 연산 : 필터연산, 필터를 커널이라고도 함

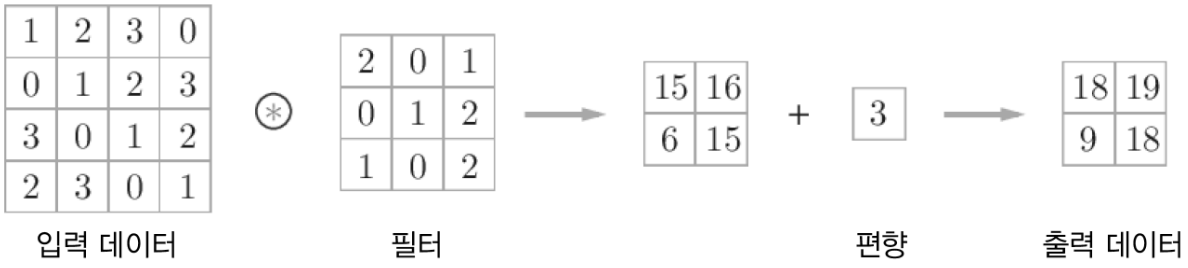


입력 데이터 필터

필터의 윈도우를 일정 간격으로 이동하며 계산



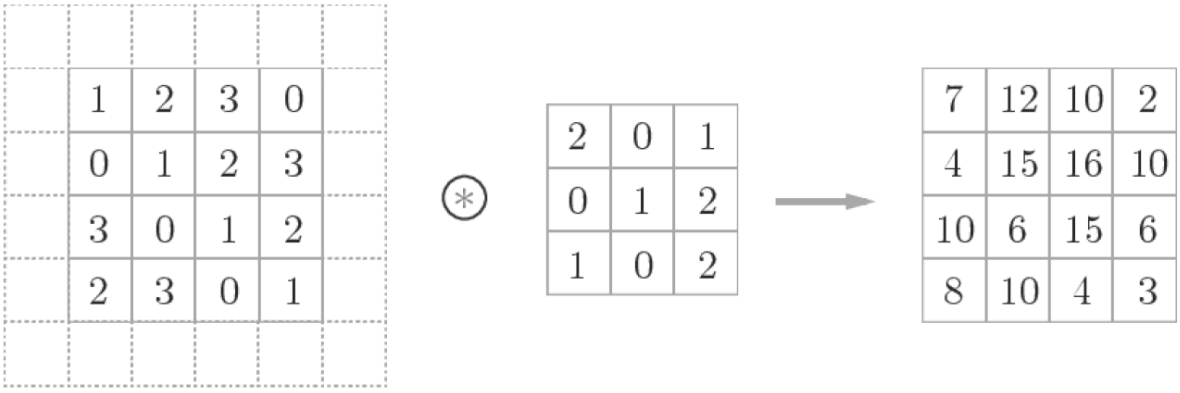
필터의 매개변수가 가중치에 해당, 편향까지 적용할 시



7.2.3 패딩 : 합성곱을 수행하기 전에 입력데이터 주변에 0을 채움

패딩에 따라 출력의 크기가 달라짐

필터가 (3,3)일 때, 패딩이 1이면 입출력 크기가 같음

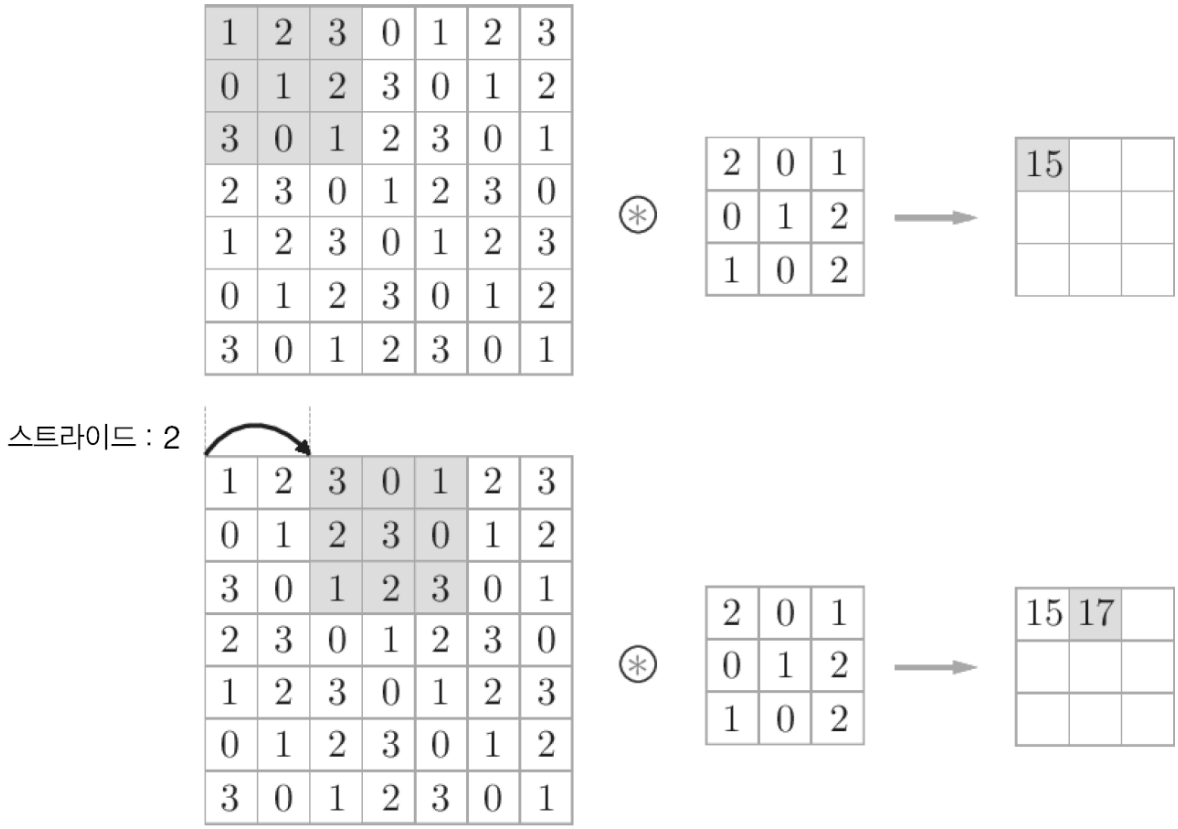


(4, 4)
입력 데이터(패딩 : 1)

(3, 3)
필터

(4, 4)
출력 데이터

7.2.4 스트라이드 : 필터를 적용하는 위치의 간격



입력 크기 (H,W), 필터 크기 (FH,FW), 출력 크기 (OH,OW), 패딩 P, 스트라이드 S 일때

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

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

윗 식의 이해: 실제 입력 크기는 H+2P, W+2P 로 생각

입력크기가 필터크기가 같으면 출력은 1 (이것 때문에 +1)

한번에 S만큼 움직이므로 1 / S 로 경우의 수만 이동

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

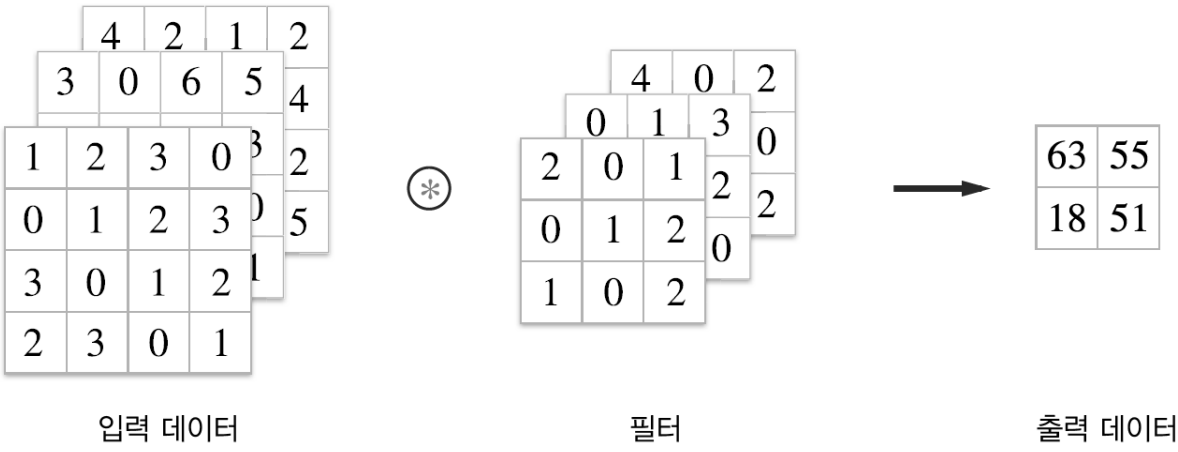
$$\frac{7+2\cdot 0-3}{2}+1=3$$

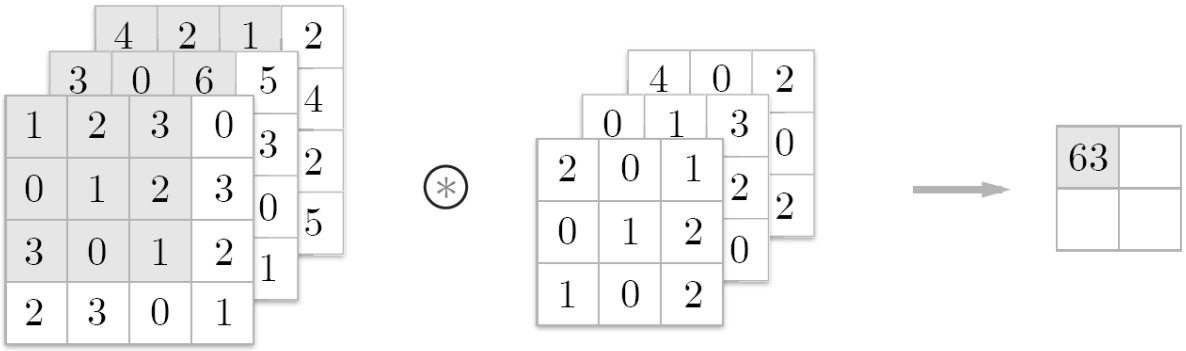
$$\frac{28+2\cdot 2-5}{3}+1=10$$

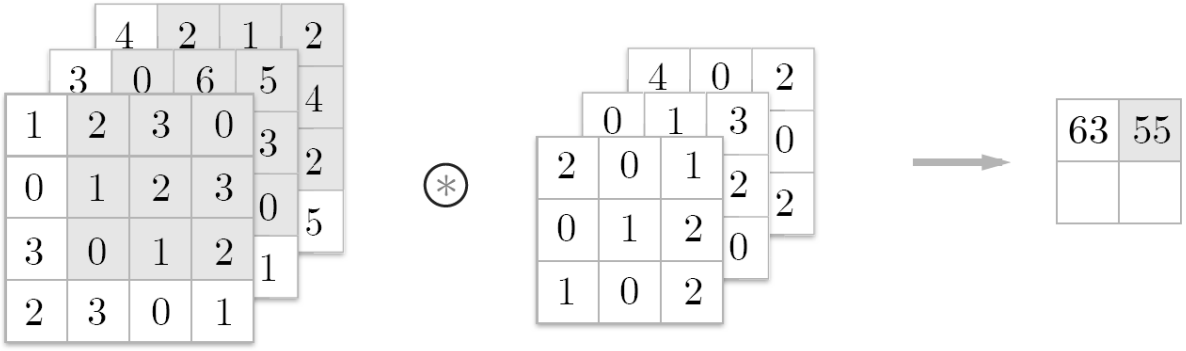
7.2.5 3차원 데이터의 합성곱 연산

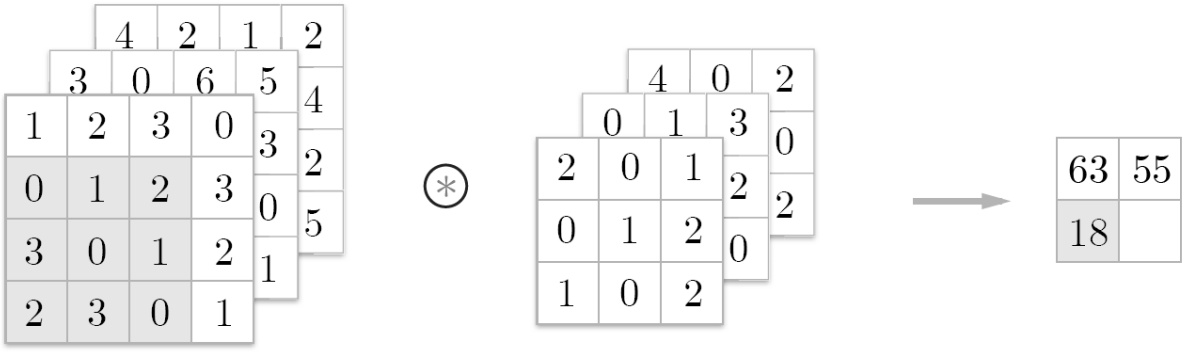
color image의 경우

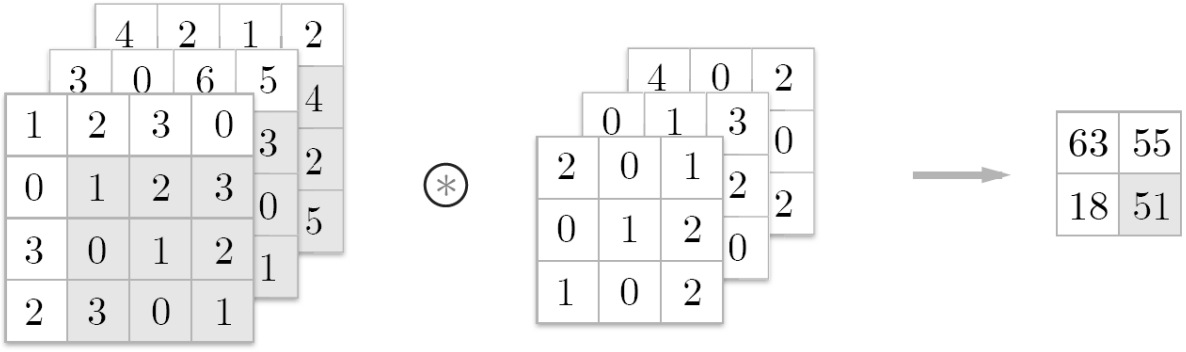
채널별로 합성곱하고 더함 : 필터의 채널수 == 입력 특징 맵 채널 수



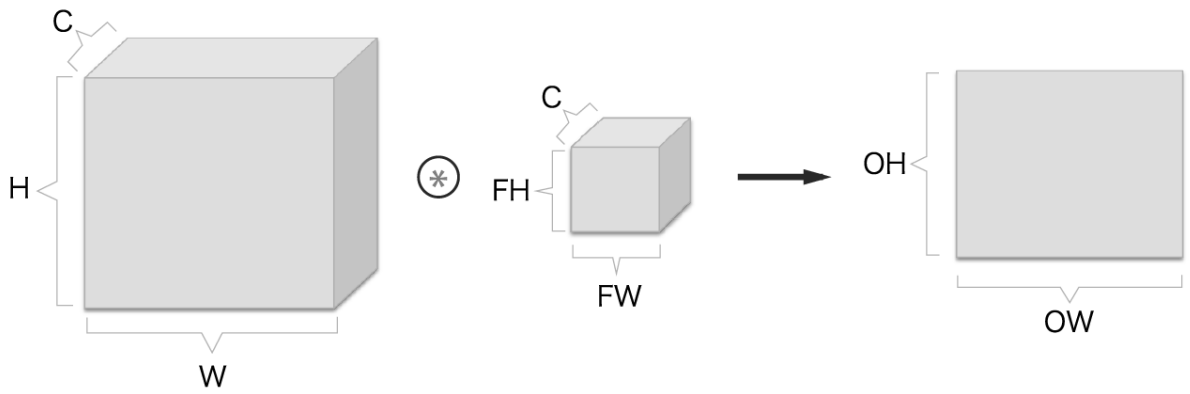

4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 3 0 5
3 0 1 2 1
2 3 0 1
⊛
4 0 2
0 1 3 0
2 0 1 2 2
0 1 2 0
1 0 2
63


4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 3 0 5
3 0 1 2 1
2 3 0 1
⊛
4 0 2
0 1 3 0
2 0 1 2 2
0 1 2 0
1 0 2
63 55


4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 3 0 5
3 0 1 2 1
2 3 0 1
⊛
4 0 2
0 1 3 0
2 0 1 2 2
0 1 2 0
1 0 2
63 55
18

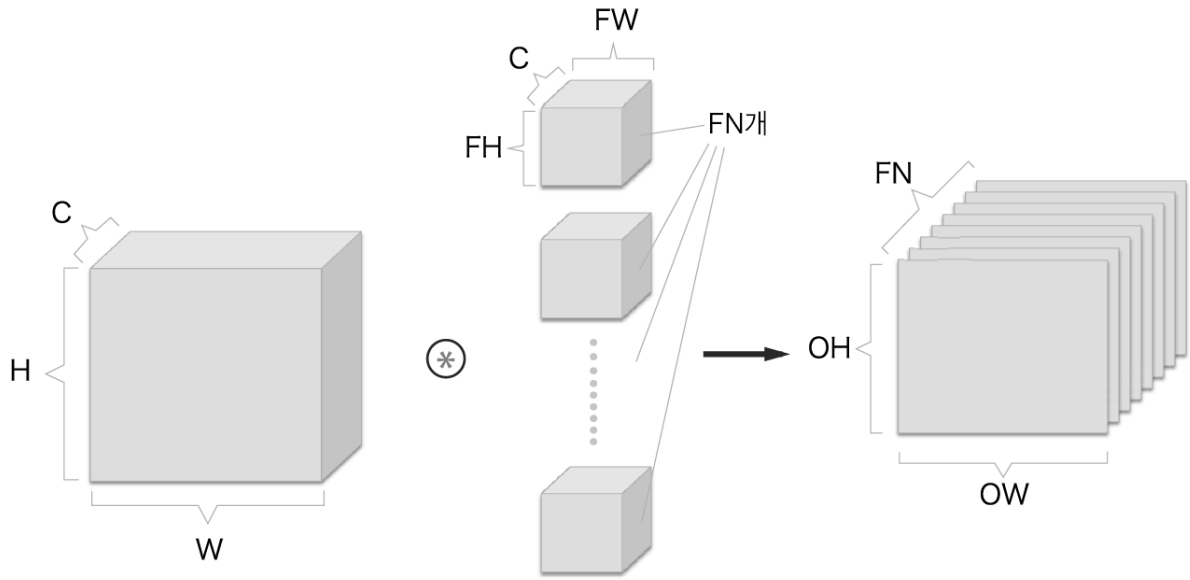

4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 3 0 5
3 0 1 2 1
2 3 0 1
⊛
4 0 2
0 1 3 0
2 0 1 2 2
0 1 2 0
1 0 2
63 55
18 51

7.2.6 블록으로 생각하기 : 3차원 합성곱 (채널,, 높이 너비)



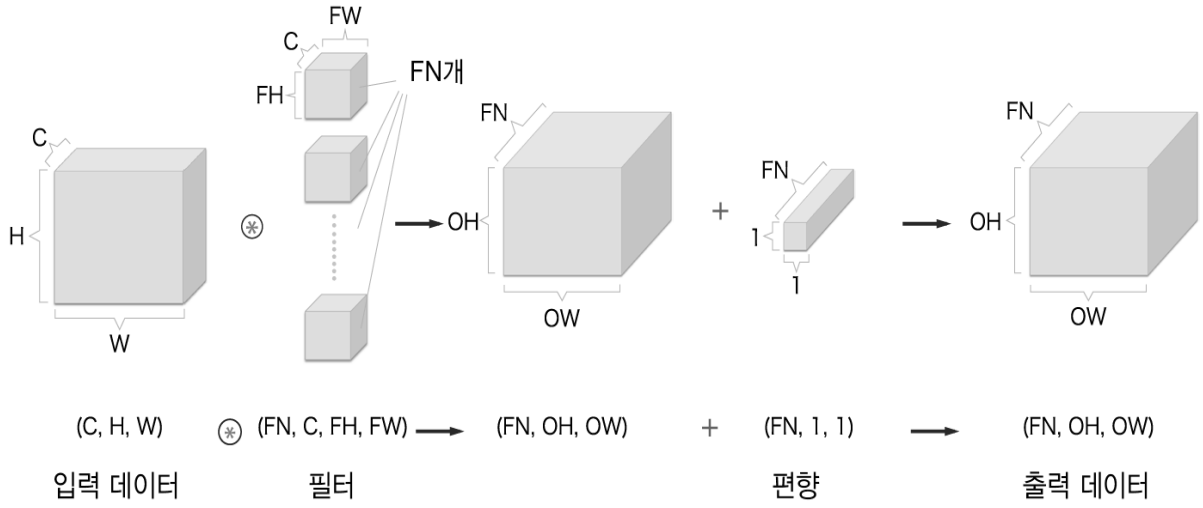
(C, H, W) 입력 데이터 ⊛ (C, FH, FW) 필터 → (1, OH, OW) 출력 데이터

여러 필터 사용시 연산



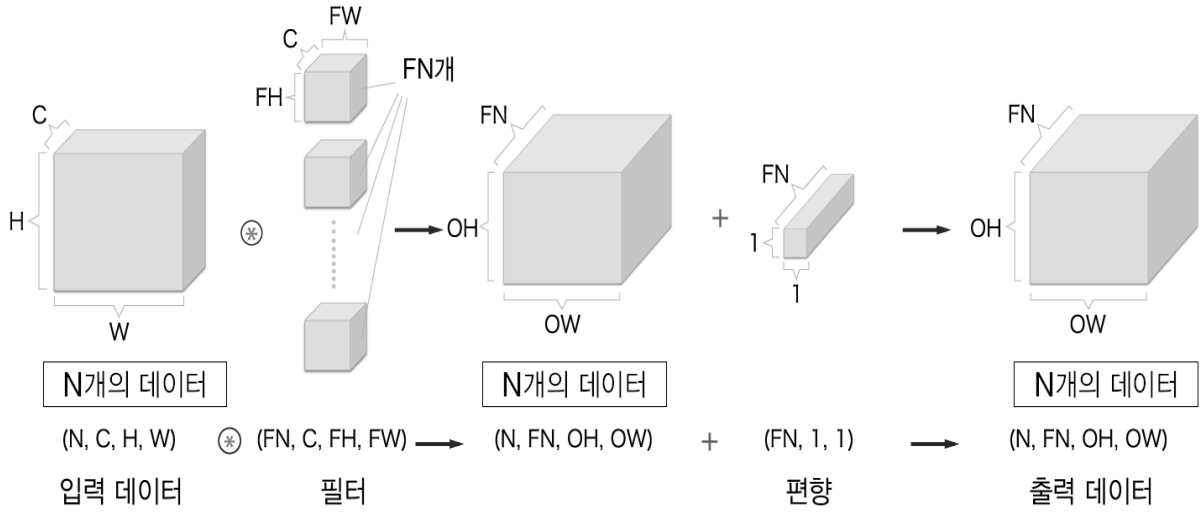
(C, H, W) 입력 데이터 ⊛ (FN, C, FH, FW) 필터 → (FN, OH, OW) 출력 데이터

편향을 추가한 연산 : 덧셈은 브로드캐스팅



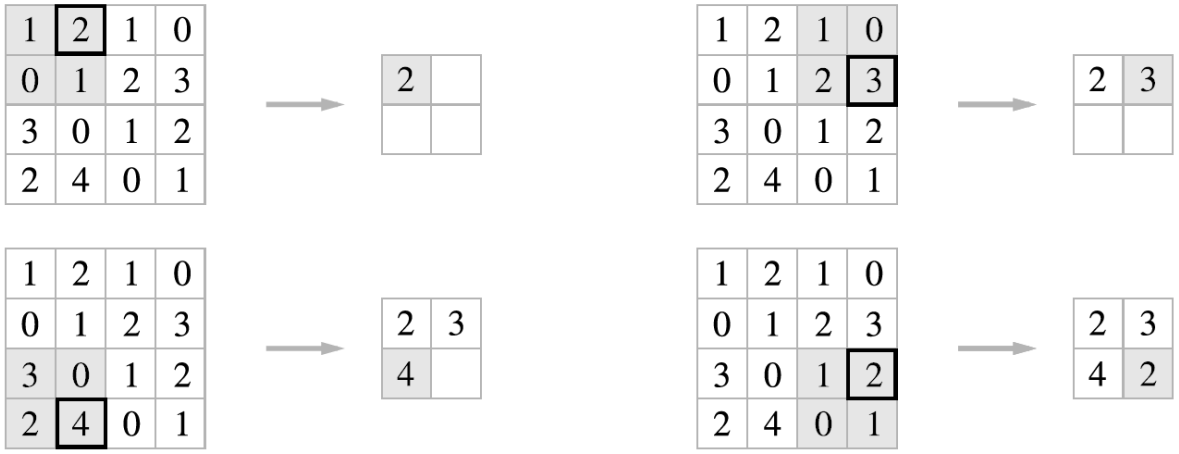
7.2.7 배치 처리 : N개 자료 동시처리, 처리효율 높음

예) GPU 여러 코어 이용 병렬 계산



7.3 풀링 계층 : 가로,세로 방향의 크기를 감소

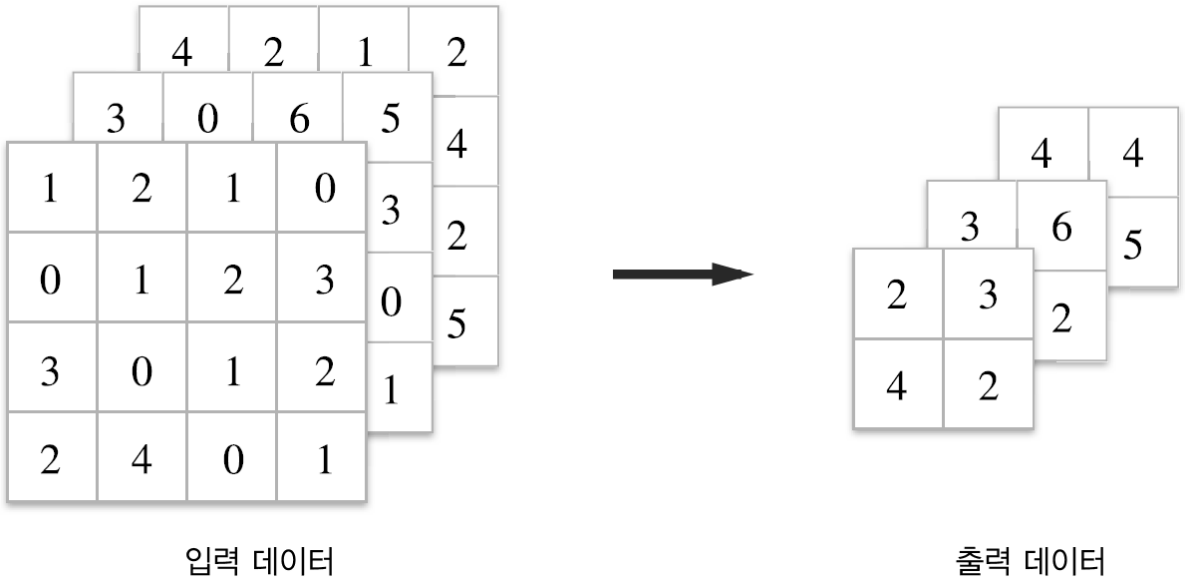
2x2 최대 풀링, 스트라이드 2, 평균 풀링도 있음



7.3.1 풀링 계층의 특징

학습해야 할 매개 변수가 없다

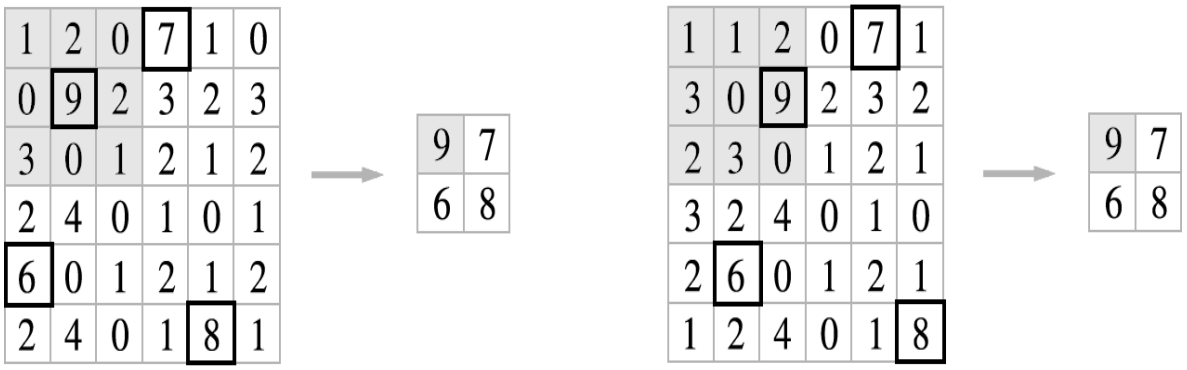
채널 수가 변하지 않는다.



입력 데이터 출력 데이터

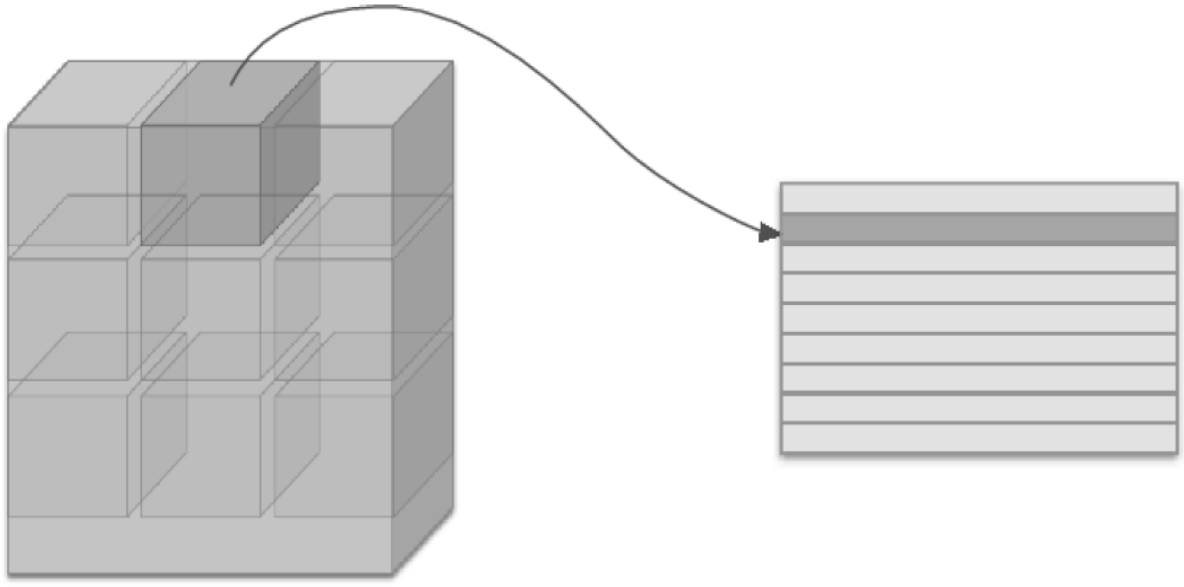
입력의 변화에 영향 적다: 한칸 이동한 입력도 같은 결과

이미지가 공간에서 조금 이동해도 비슷한 처리 결과 줌

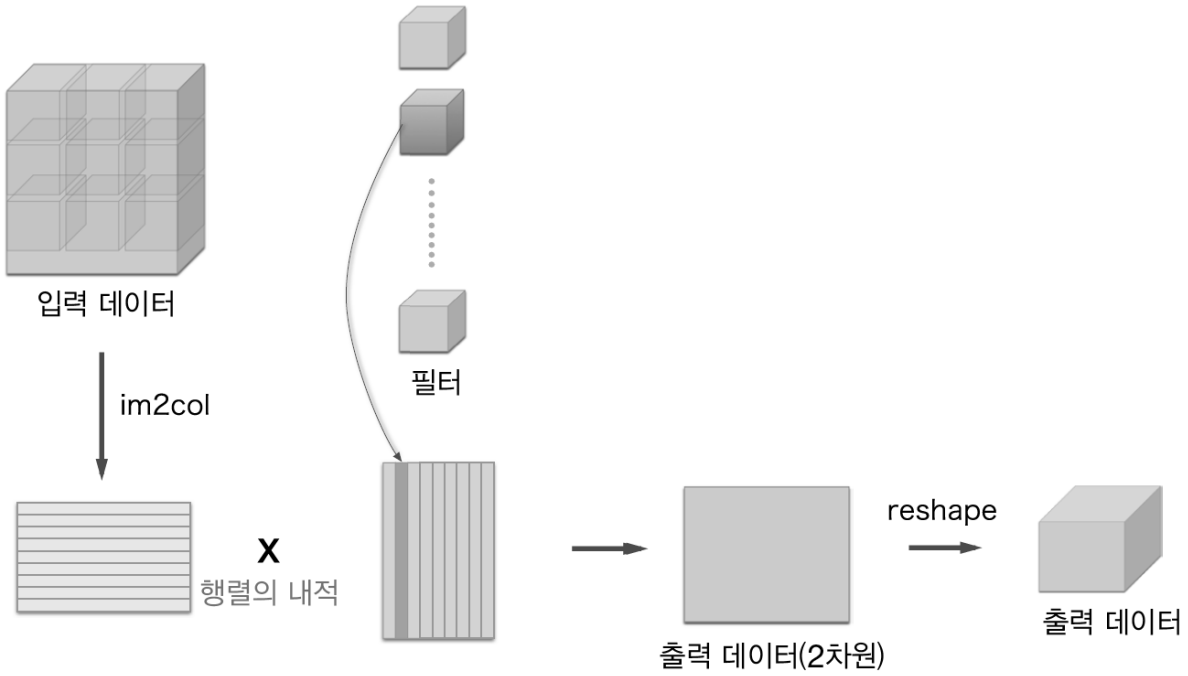


7.4 합성곱/풀링 계층 구현하기

convolution은 행렬과 행렬의 원소별 곱 → 일차원으로 변형후 계산



7.4.2 im2col로 데이터 전개하기



7.4.3 합성곱 계층 구현하기

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

x1 = np.random.rand(1,3,7,7) #데이터수, 채널 수, 높이, 너비

col1 = im2col(x1, 5, 5, 1, 0) # (9, 75)

가로 7을 필터(5)로 패딩 0, 스트라이드 1로이동 3개

마찬가지로 세로도 3개이므로 총 9개임,

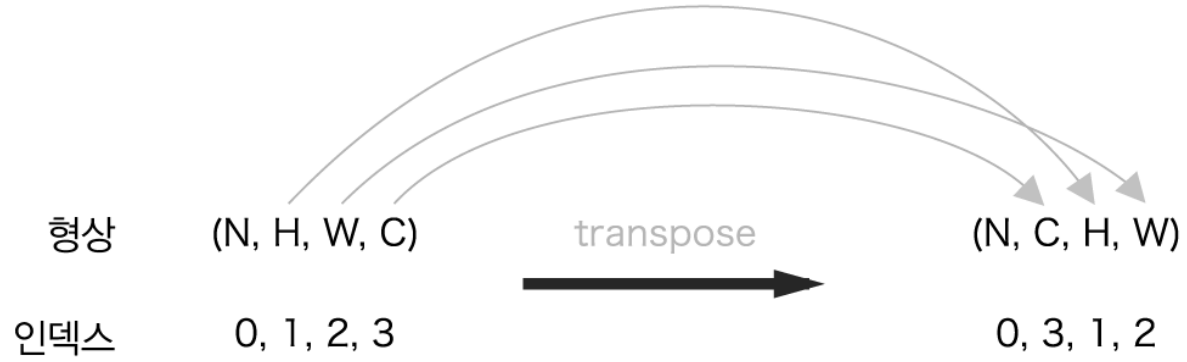
75는 데이터 수 = 채널 x 높이 x 너비 = 3 x 5 x 5

x2 = np.random.rand(10,3,7,7)

col2 = im2col(x2, 5, 5, 1, 0) # 10개 자료이므로 (90, 75)

합성곱 계층 구현은 common/layers.py내 class Convolution의 forward 보라

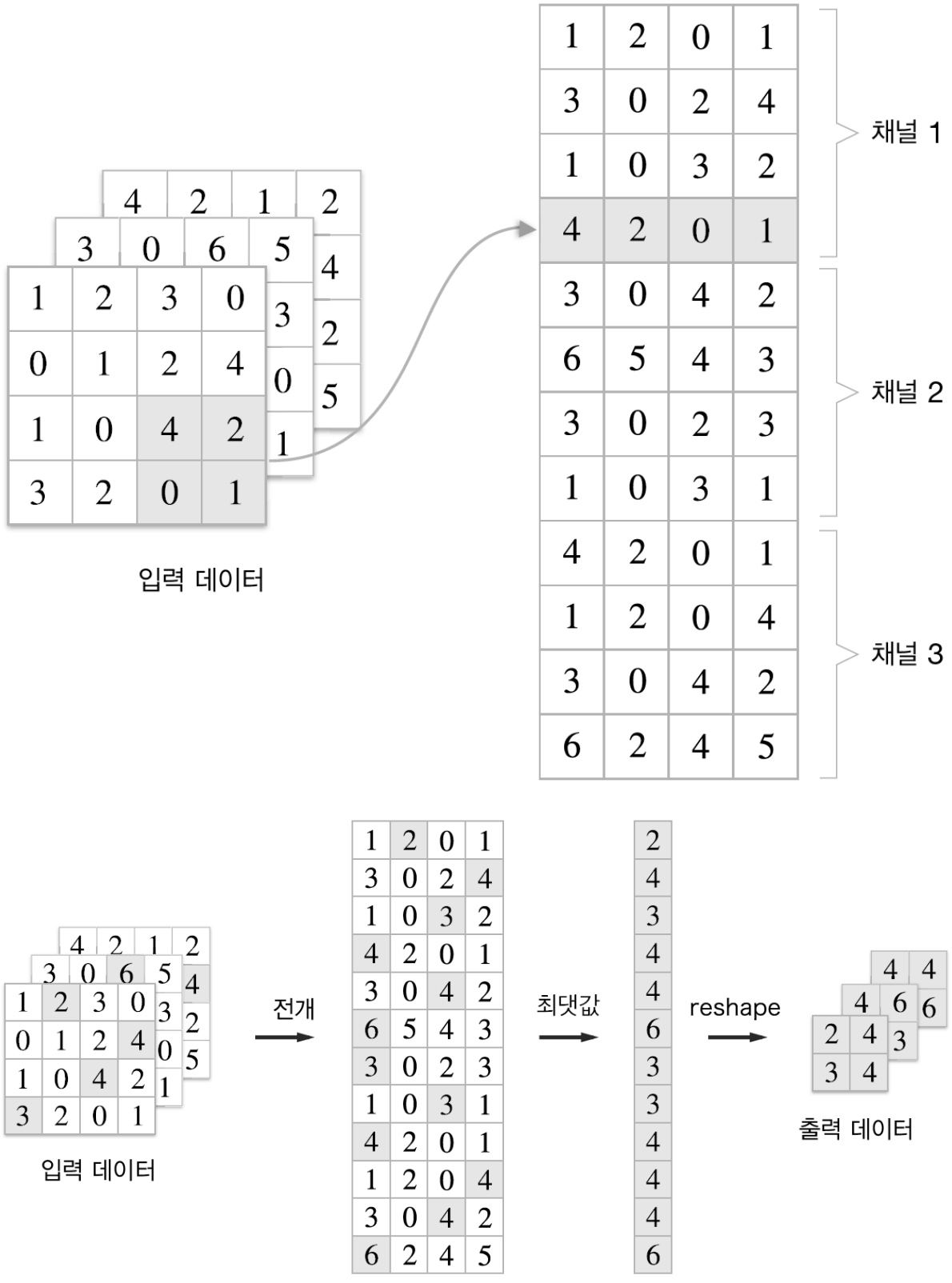
코드내용에서 transpose



backward는 Affine 계층의 행렬곱 미분과 유사 (***)

7.4.4 풀링 계층 구현하기

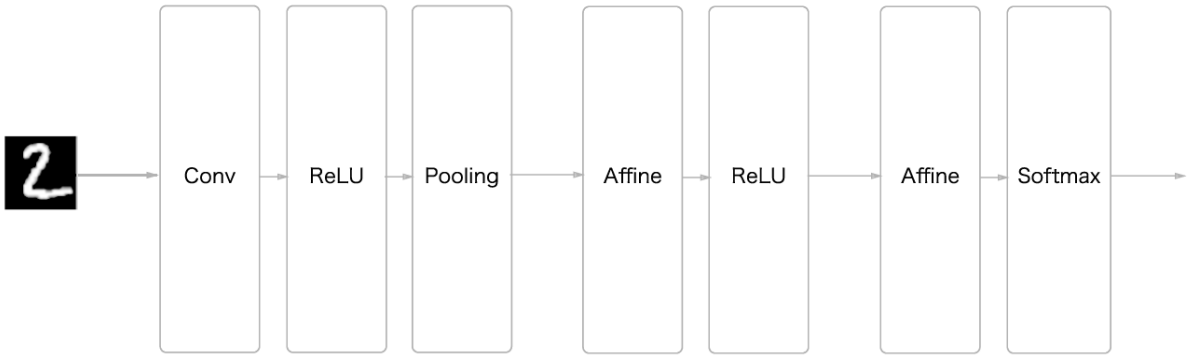
채널별로 처리



구현은 common/layers.py내 class Pooling:

backward는 relu와 유사 (***)

7.5 CNN 구현하기



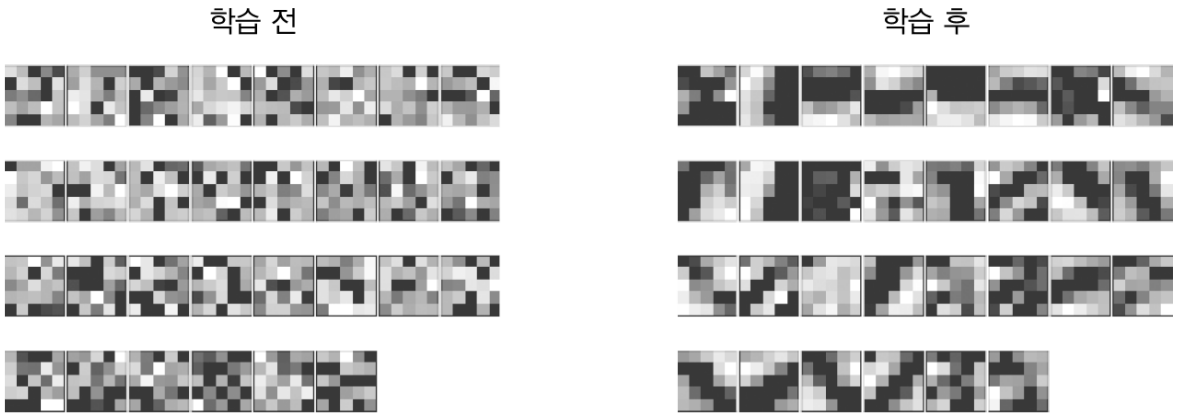
1개 합성곱 계층, 2개 완전연결 계층으로 구성

ch07/simple_convnet.py 읽기

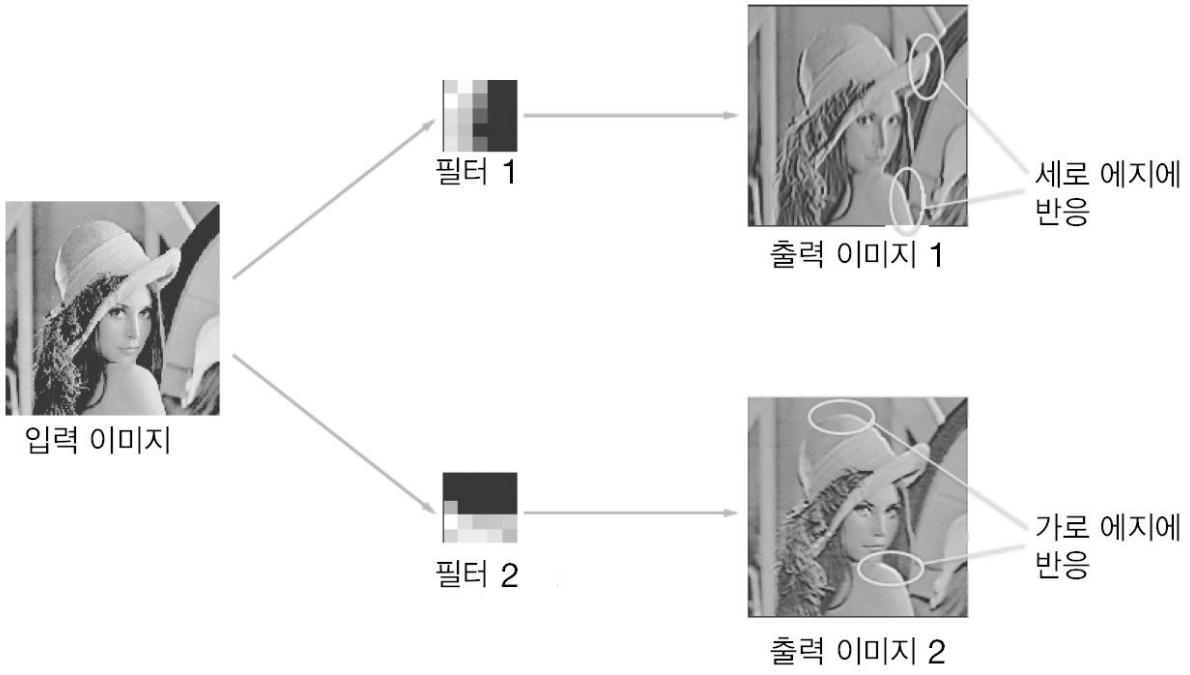
전체적인 학습은 ch07/train_convnet.py

7.6 CNN 시각화하기

7.6.1 1번째 층의 가중치 시각화하기

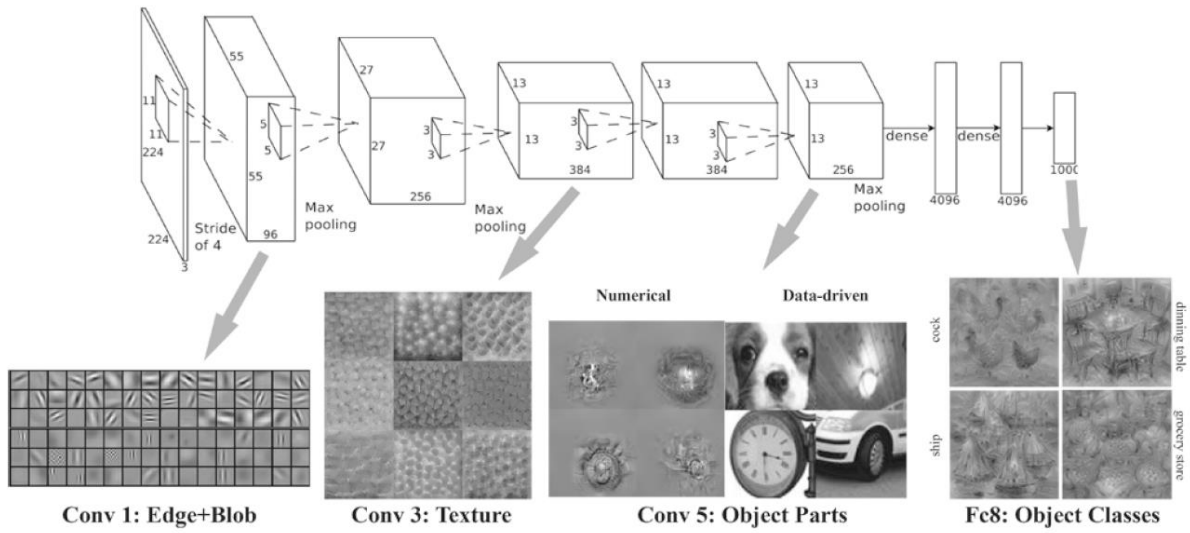


무작위 초기값은 흑백의 정도에 규칙성이 없고, 점차 학습하면서 규칙성을 가짐 → 검은색에서 점차 변화하거나, 덩어리를 가지거나 규칙을 띄는 필터 → 에지와 블롭을 보고 있음



7.6.2 층 깊이에 따른 추출 정보 변화

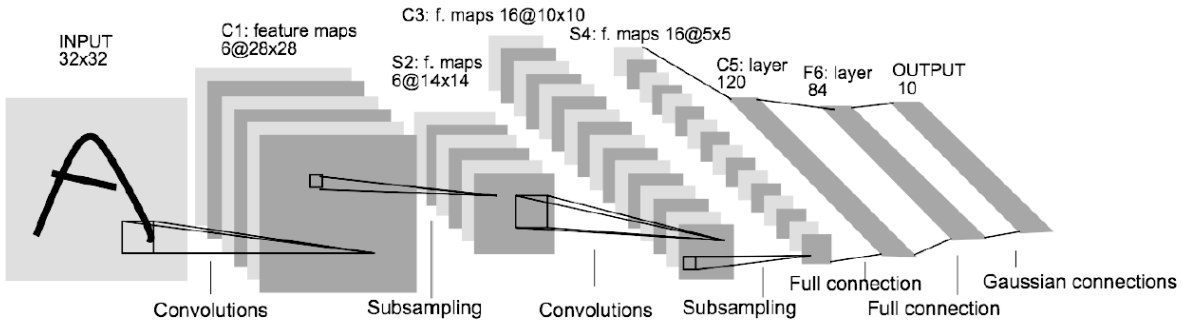
점차 추상화됨 → 처음에는 엣지와 블롭, 3번재 층은 텍스처, 5번째 층은 사물의 일부, 마지막 층은 사물의 클래스



7.7 대표적인 CNN

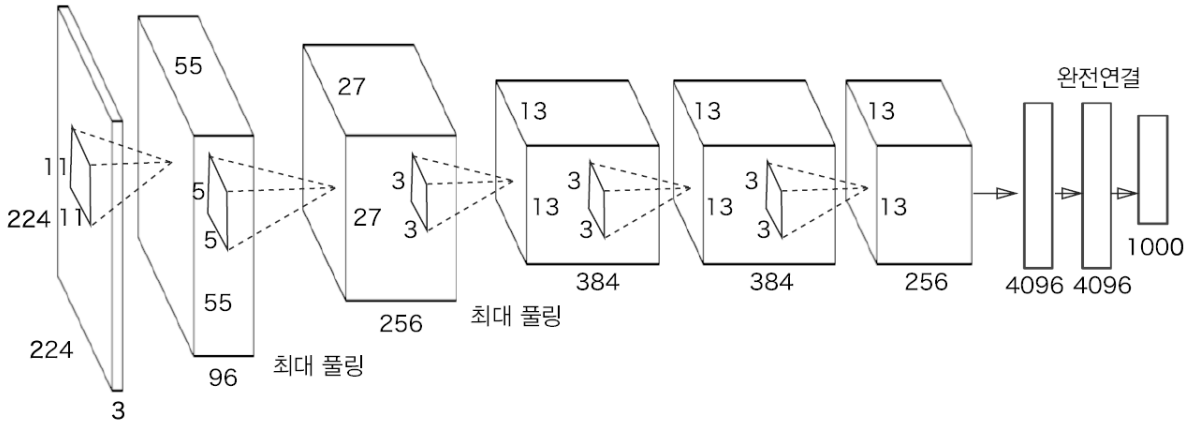
7.7.1 LeNet

1998년, 시그모이드 → ReLU, 서브샘플링 → 최대 풀링



7.7.2 AlexNet

2012년



ReLU, Local Response Normalization, dropout

➔ LeNet과 큰차이 없으나, 대량의 자료와 GPU 사용으로 발전

8장은 개괄적인 소개이므로 실제적인 이해가 어려움, 가볍게 읽고, 3학년 2학기에 tensorflow로 구현하며 공부