📖

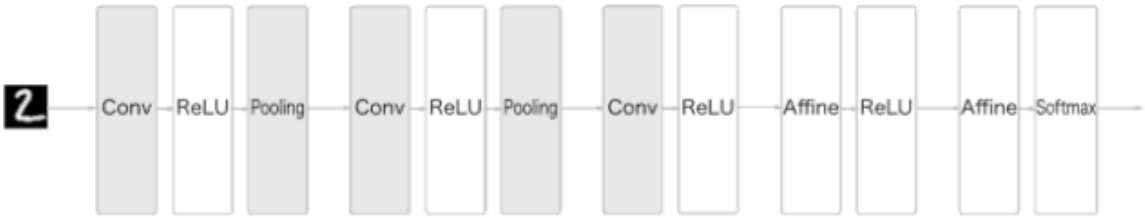
[BASE] 합성곱 신경망(CNN)

1. 합성곱 신경망(Convolutional Neural Network)

🌟 CNN은 이미지 인식 및 음성 인식등 다양한 곳에서 사용됩니다.

특히, 이미지 인식 분야에서 딥러닝을 활용한 기법들은 대부분 CNN을 기초로 하고 있습니다.

▼ 1-1. 전체 구조



대부분의 CNN의 계층은 **Conv - ReLU - (Pooling)** 흐름으로 연결됩니다.

또한, Output에 가까운 층에서는 **Affine-ReLU** 구성을 사용하며,

마지막 출력 계층에서는 **'Affine-Softmax' 조합을 그대로 사용합니다.**

▼ 1-2. CNN Layer(합성곱 계층)

FC Problem

- 데이터의 형상이 무시된다는 점입니다.

  입력 데이터가 이미지인 경우, 일반적으로 (height, width, channels)로 구성된 3차원 데이터이다. 만약 **FC**를 사용하여 3차원 데이터를 1차원 데이터로 변환하는 경우, 가까운 Pixel에 대한 **공간적 정보를 소실하게 됩니다.**

CNN(Convolutional Neural Network)

- **CNN**에서 **CNN Layer**(합성곱 계층)의 입출력 데이터를 특징 맵(feature map)이라고 합니다.

  (일반적으로 입력 데이터를 **입력 특징 맵(input feature map)**, 출력 데이터를 **출력 특징 맵(output feature map)**이라고 합니다.)

▼ 1-3. 합성곱 연산

🌟 **CNN Layer에서는 합성곱 연산을 이미지 처리의 필터 연산을 의미합니다.**

합성곱 연산은 입력 데이터에 필터를 적용합니다.

일반적으로 CNN에서는 데이터의 형상을 (height, width)로 표시합니다.

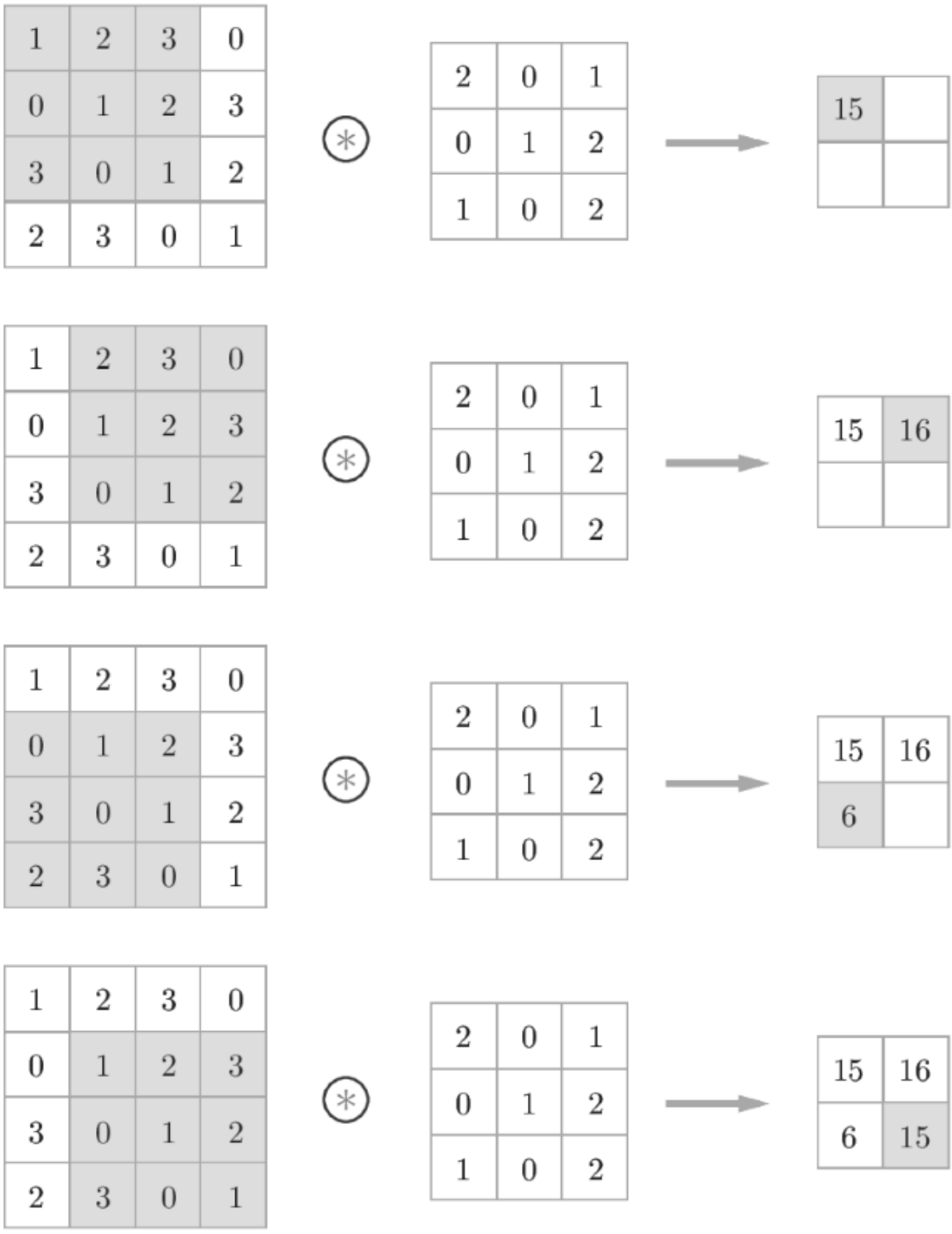
Input(4, 4) * **Filter**(3, 3) = **Output**(2, 2) 입니다. (**Filter를 Kernel 이라고도 합니다.**)

🌟 **CNN(Convolutional Neural Network)에서 Filter의 매개변수를 가중치(Weights)라고 합니다.**

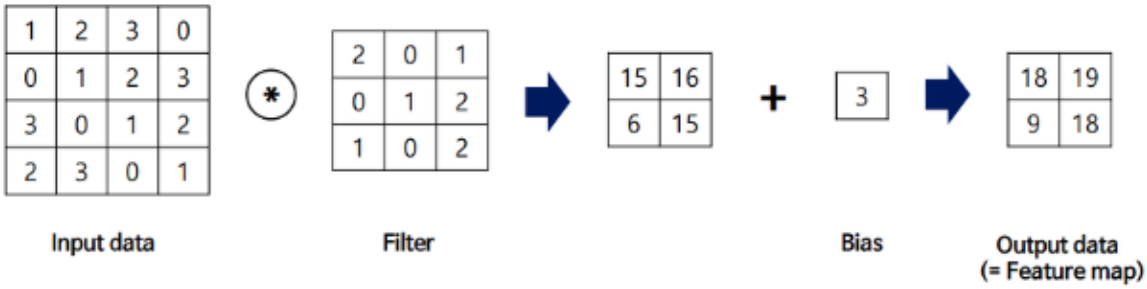
합성곱 연산은 **Filter(Kernel)의 Window를 일정 간격으로 이동하며, Input에 적용합니다.**

(Window는 하단 이미지의 회색 바탕을 의미합니다.)

하단과 같이 수행하는 적분(덧셈)방식을 **단일 곱셈-누산(fused multipliy-add, FMA)**라고 합니다.



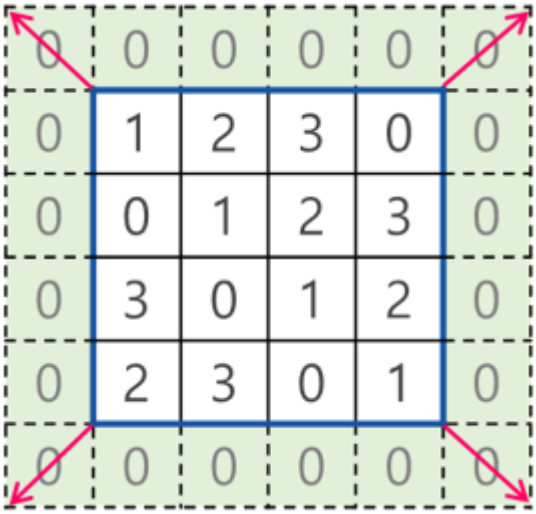
(만약 Bias(편향값)을 Filter(Kernel)에 적용한다면, 하단 이미지처럼 연산됩니다.)



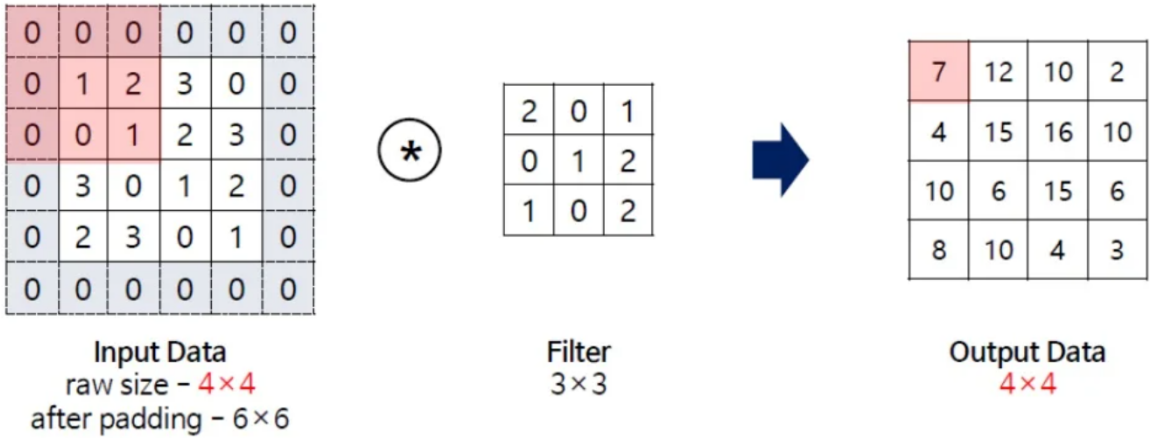
▼ 1-4. 패딩

✨ **합성곱 연산을 수행하기 전에 입력 데이터 주변을 특정 값으로 채우는 것**을 의미합니다.

EX> **Input Image(4,4) 에 Padding 1을 적용한 경우 (0으로 채우는 경우 ZeroPadding)**



EX> 처음 **Input Image**(4,4)에 **Padding**(1) 한 것에 **Filter**(3, 3) 적용한 결과 = **Output**(4, 4)

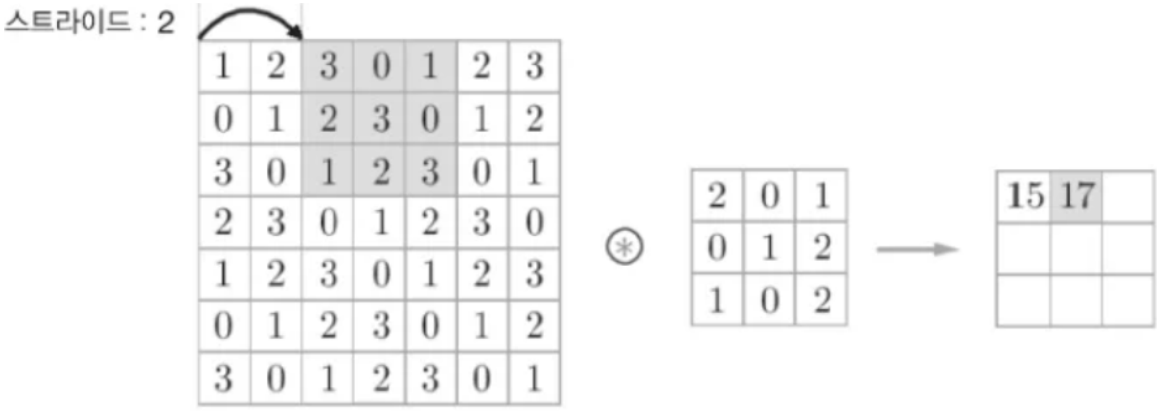


▼ 1-5. 스트라이드

Input Image에 **Filter**를 적용하는 위치의 간격을 **스트라이드(Stride)**라고 합니다.

EX > Stride가 2인 경우 (Window가 2칸씩 이동합니다.)

✨ **Input Image(7,7)에 Window(2)로 Filter(3, 3)를 적용한 결과 ⇒ Output Image(3, 3)**



수식(Formulation)

- **Input ImageSize(H, W)**
- **Filter Size(FH, FW)**
- **Padding(P)**
- **Stride(S)**
- **Output ImageSize(OH, OW)**

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

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

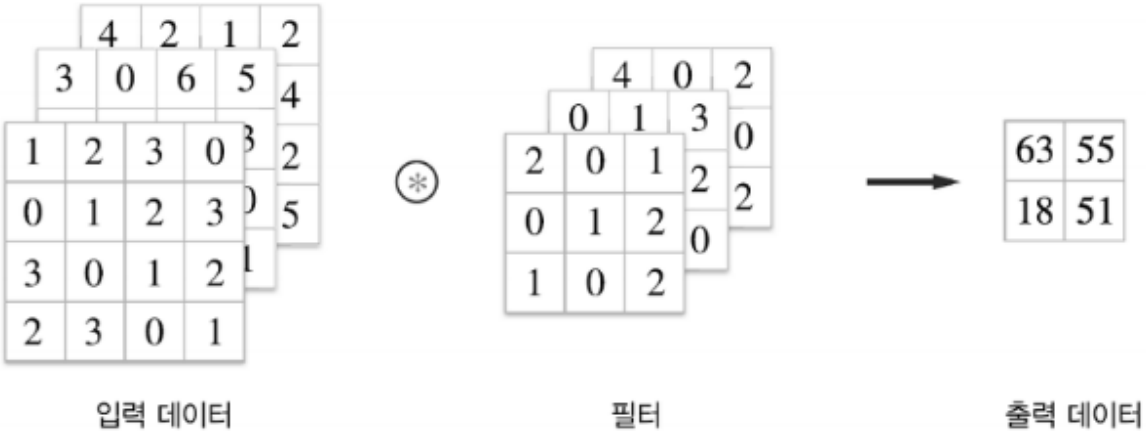
etc) 정수로 떨어지지 않는다면, 일반적으로 반올림을 하지만,

웬만하면, 정수로 표현되게 유도합니다.

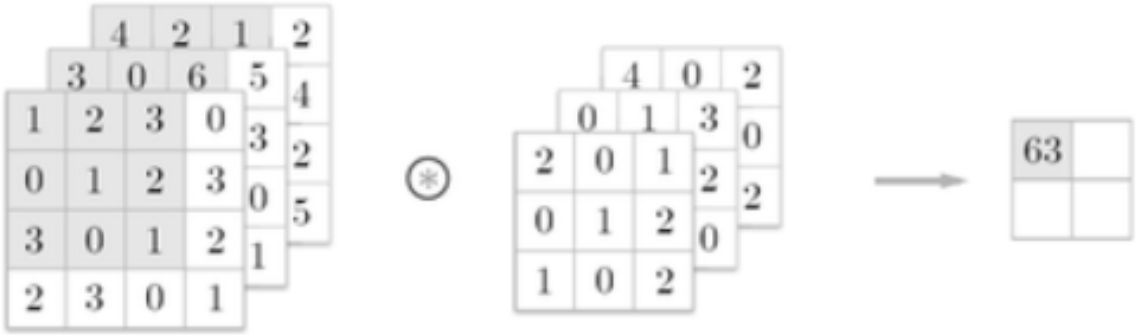
▼ 1-6. 3차원 데이터의 합성곱 연산

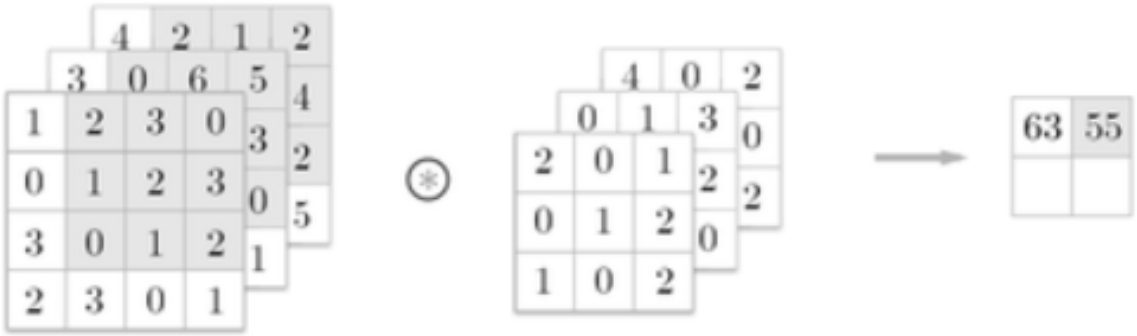
✨ 이미지의 경우, (H, W, C) [세로, 가로, 채널]까지 고려한 3차원 데이터이다.

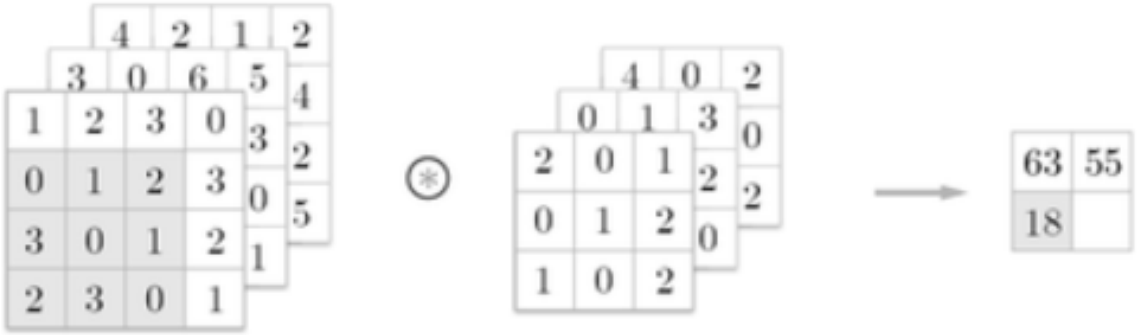
주의점으로는, Input Image와 Filter의 channel 수가 동일해야 합니다.

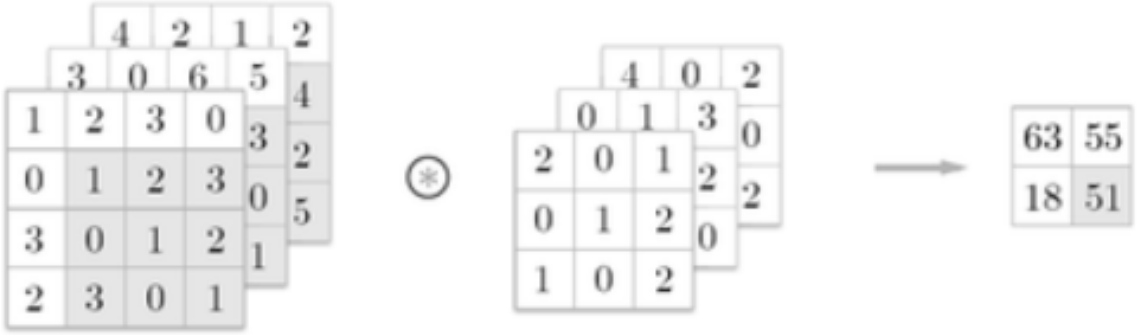


EX 계산과정>


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

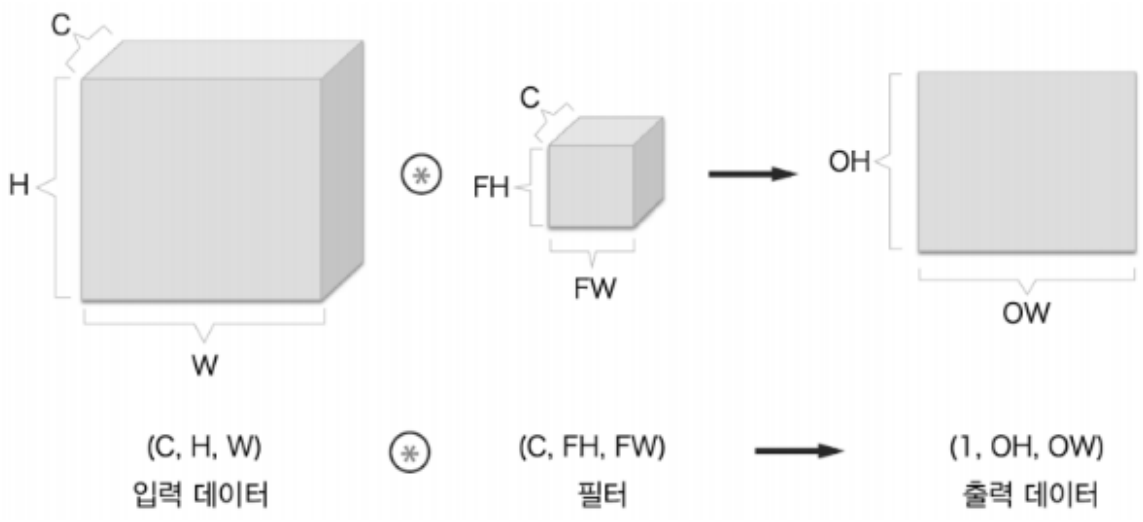

4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 3 0 5
3 0 1 2 1
2 3 0 1
4 0 2
0 1 3 0
2 0 1 2 2
0 1 2 0
1 0 2
63 55


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

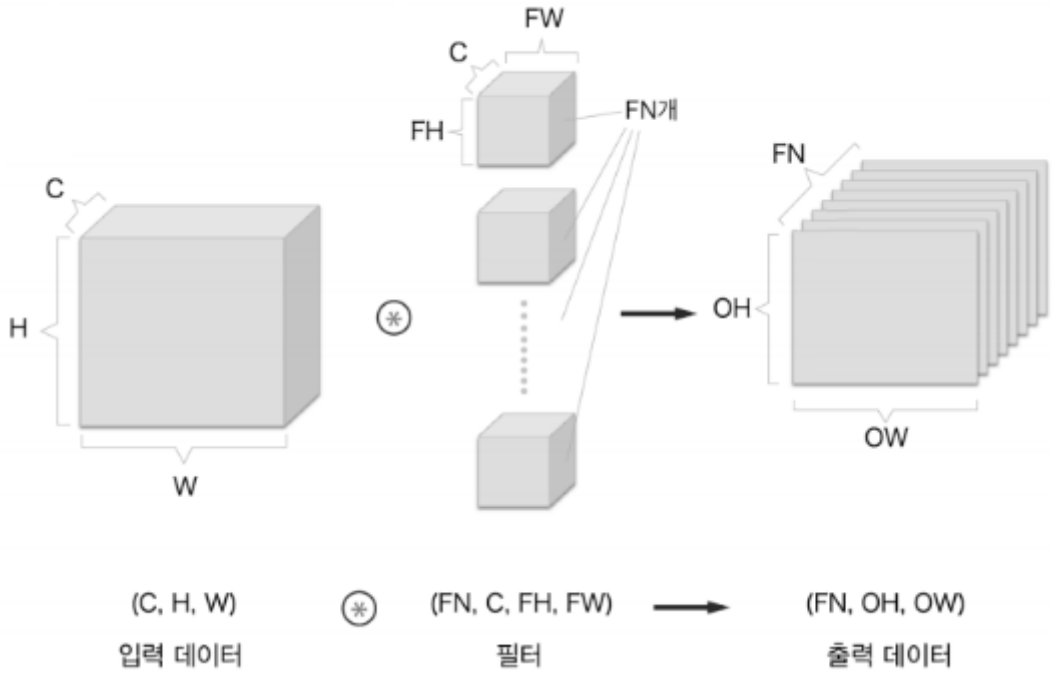

4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 3 0 5
3 0 1 2 1
2 3 0 1
4 0 2
0 1 3 0
2 0 1 2 2
0 1 2 0
1 0 2
63 55
18 51

▼ 1-7. 블록으로 생각하기

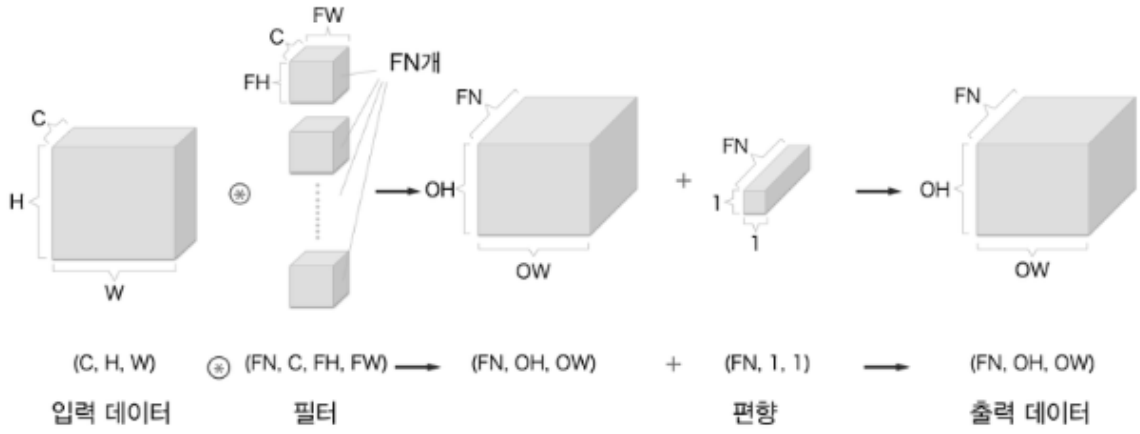
✨ **3차원 합성곱 연산은 데이터와 필터를 직육면체 블록으로 생각하면 쉽습니다.**



만약, Filter Block이 여러 개 존재한다면,



여기에, **Bias Block을 추가한다면, CNN의 흐름은 아래의 이미지와 같습니다.**

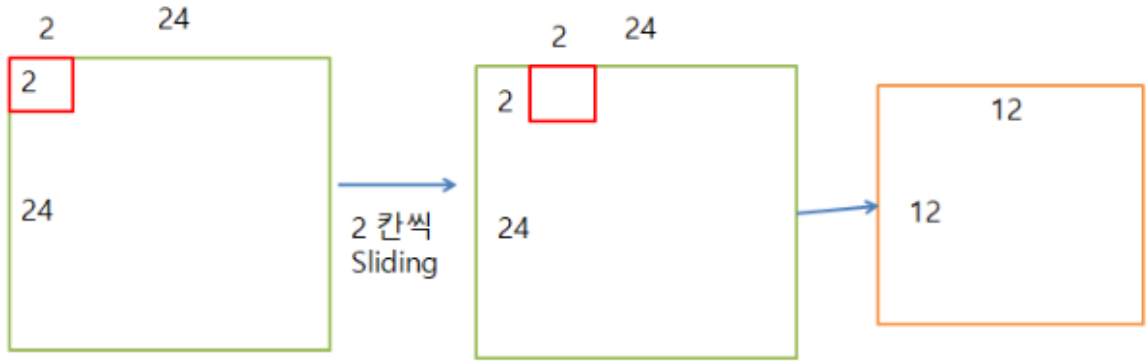


▼ 1-8. Pooling Layer(폴링 계층)

✨ **Pooling**이란? **세로, 가로 방향의 공간을 줄이는 연산을 의미**합니다.

EX Pooling (2) 와 Window (2) 인 경우>

(일반적으로, Pooling == Window 입니다.)



Pooling을 하는 이유?

- 정말 필요한 데이터만 출력할 수 있습니다.
- 데이터의 양이 적어집니다.
- Feature가 많아져, Overfitting이 발생하는 경우를 방지합니다.

Pooling의 특징

- 학습해야 할 매개 변수가 없습니다.
  - 합성곱 계층과 달리 학습해야 할 매개변수가 없습니다.
  - 폴링은 대상 영역에서 최댓값이나 평균을 취하는 명확한 처리이므로, **학습X**

- 채널 수가 변하지 않습니다.
- 입력의 변화에 영향을 적게 받습니다. (**Strong[강건하다.]**)