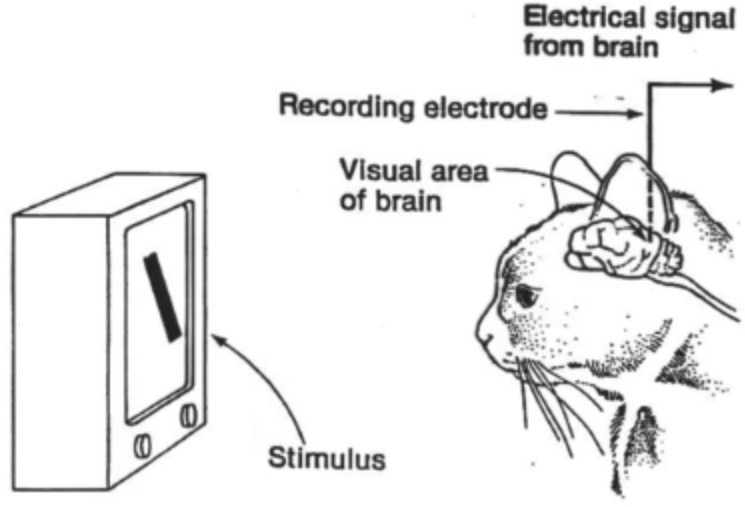
Convolution Neural Network

https://github.com/dlcjfgmlnasa/Tongmyong_College_Tensorflow_Tutorial

Created by **Choelhui lee**

Convolution Neural Network (CNN)

- **`Computer vision`** 특화된 네트워크
- 인간의 시신경의 구조를 모방 하여 **`인간의 vision정보를 처리하는 것을 흉내낸 deep learning 알고리즘`**

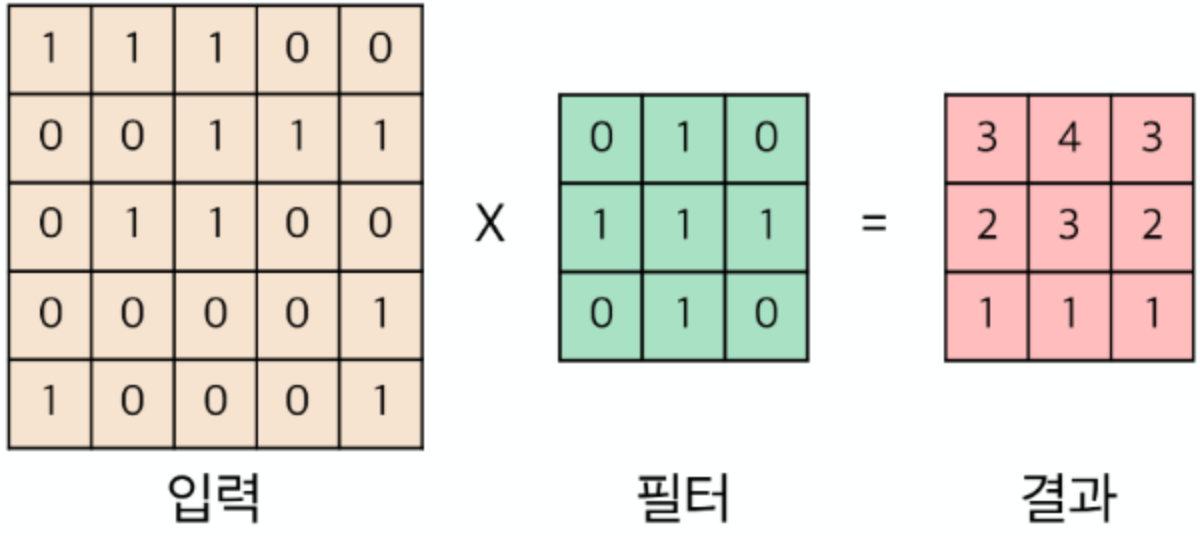


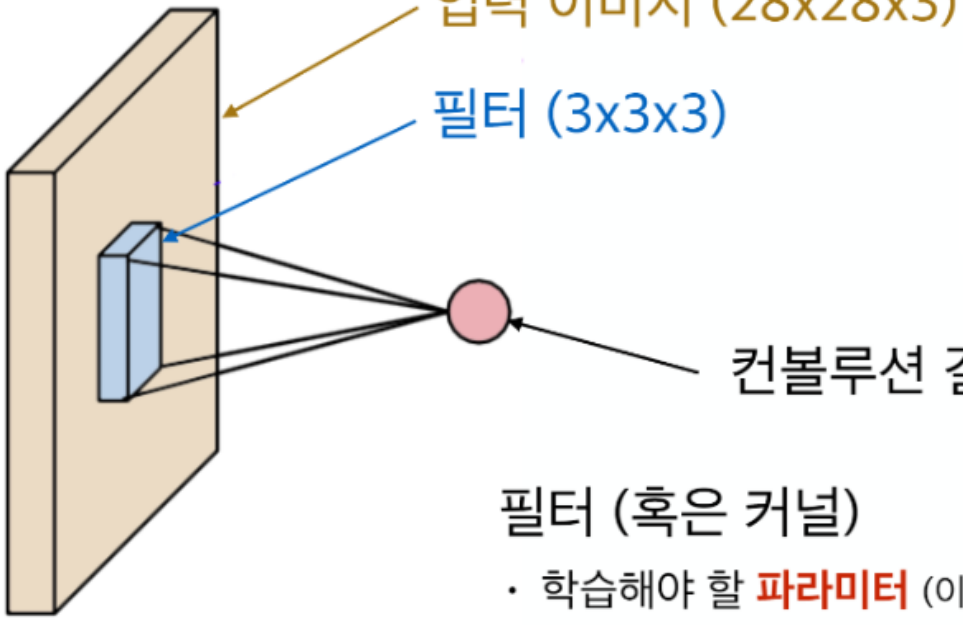
Convolution Neural Network (CNN) 의 특징

- **`전통적인 이미지 분류 프로세스`**
  - feature extrator(특징추출) 과 Classifier(분류기)가 별도로 존재
  - 특징추출 모듈 은 전통적인 computer vision 알고리즘으로 수행
  - 분류기만 학습
- **`CNN 이미지 분류 프로세스`**
  - feature extrator(특징추출) 과 classification(분류기)이 하나로 존재
  - **`feature extrator은 Convolution Layer`** + **`Classifier은 fully connected Layer`**
  - 특징 추출의 과정을 신경 쓸 필요가 없음 (`end-to-end`)

Convolution

- 입력 값과 필터를 통해 결과값을 도출하는 과정
- **`Convolution 은 어떤 filter를 사용하여 주어진 image의 적절한 feature를 뽑아내기 위해 사용한 operation == Convolution 연산은 해당 이미지의 특징을 뽑아주는 연산을 실시한다`**
- **`Convolution Nerual Network 의 핵심은 좋은 feature map을 뽑아주는 convolution filter 를 learning하는 모델을 만드는데 있음`**



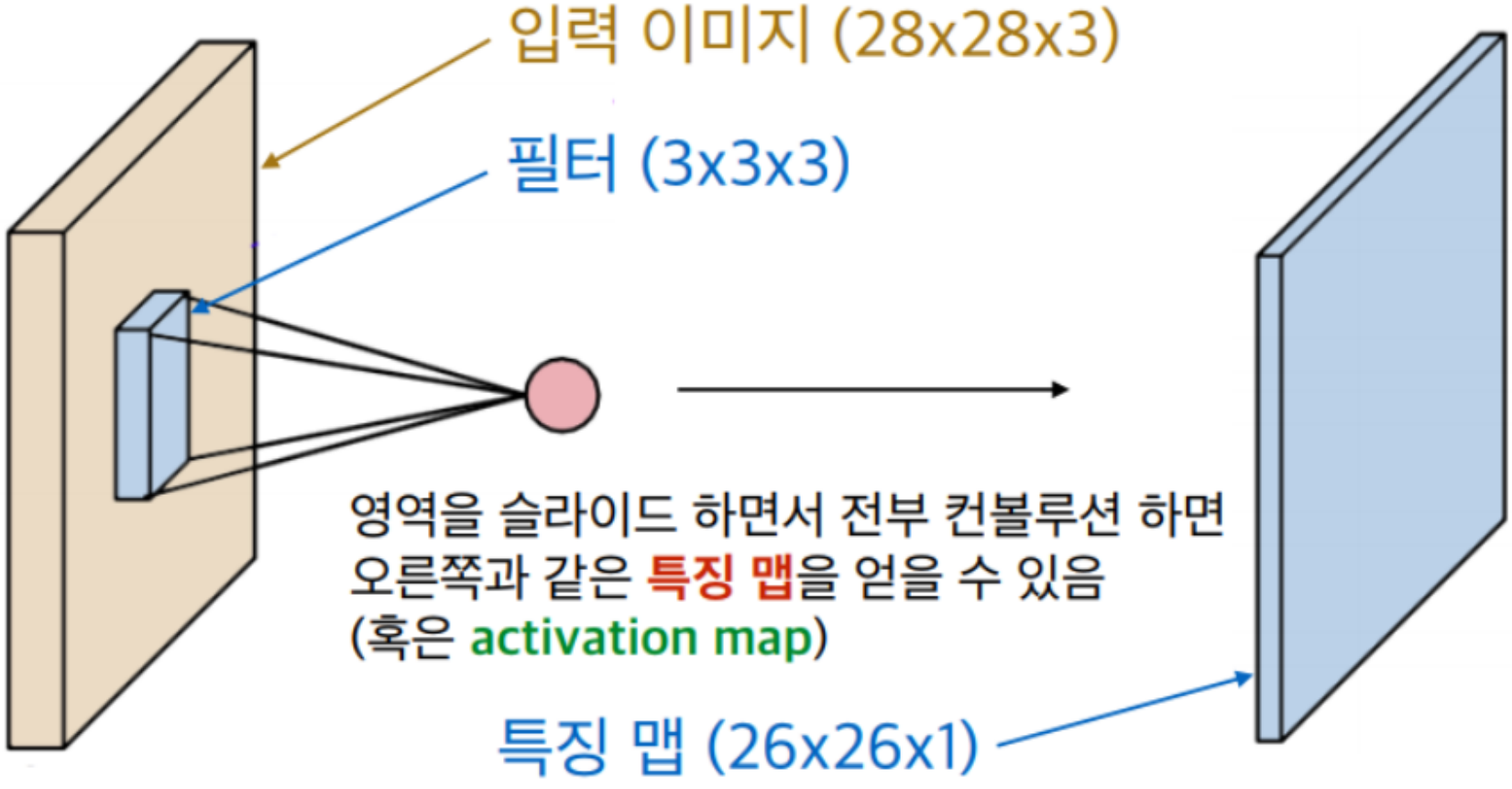


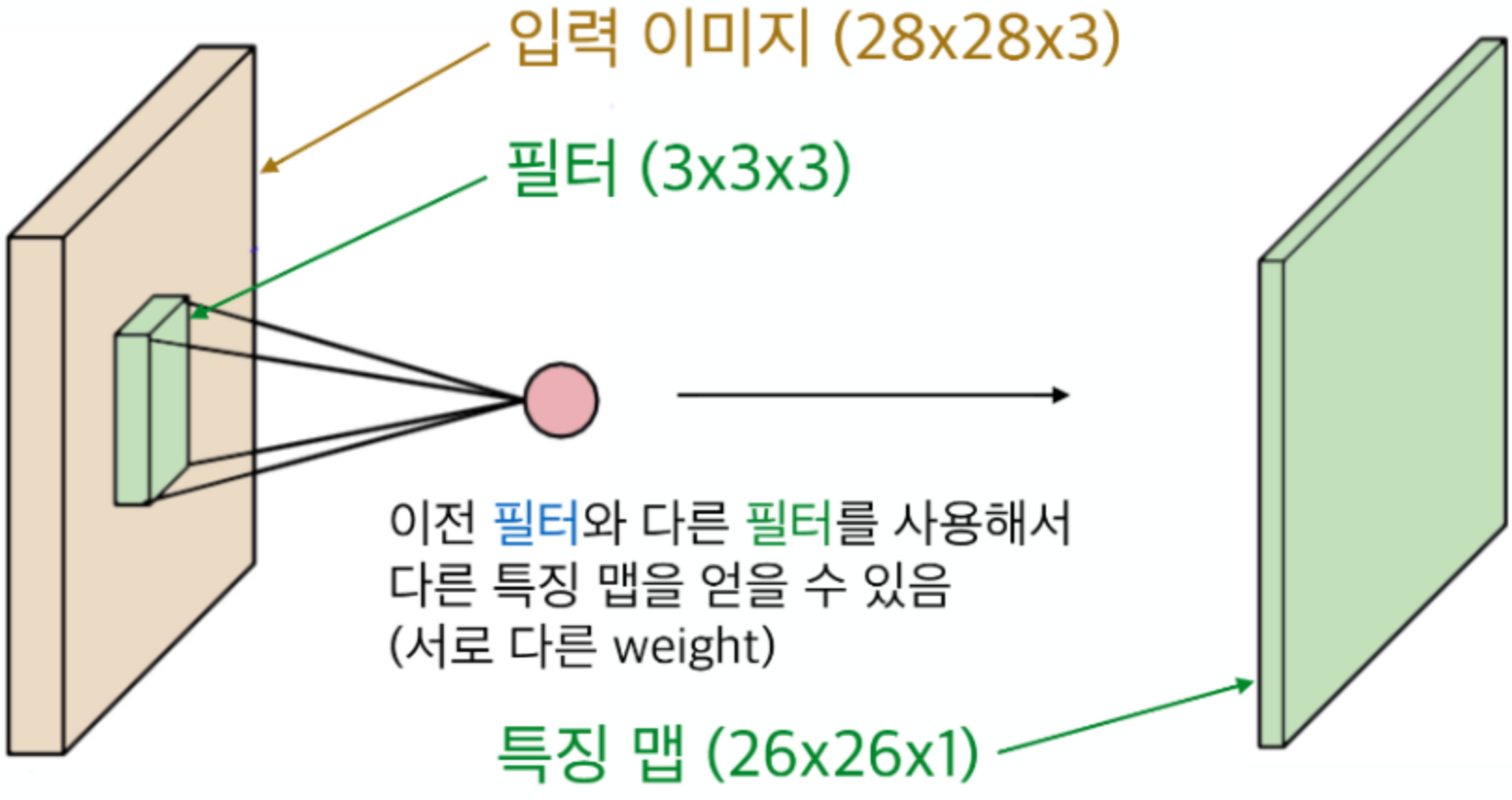
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

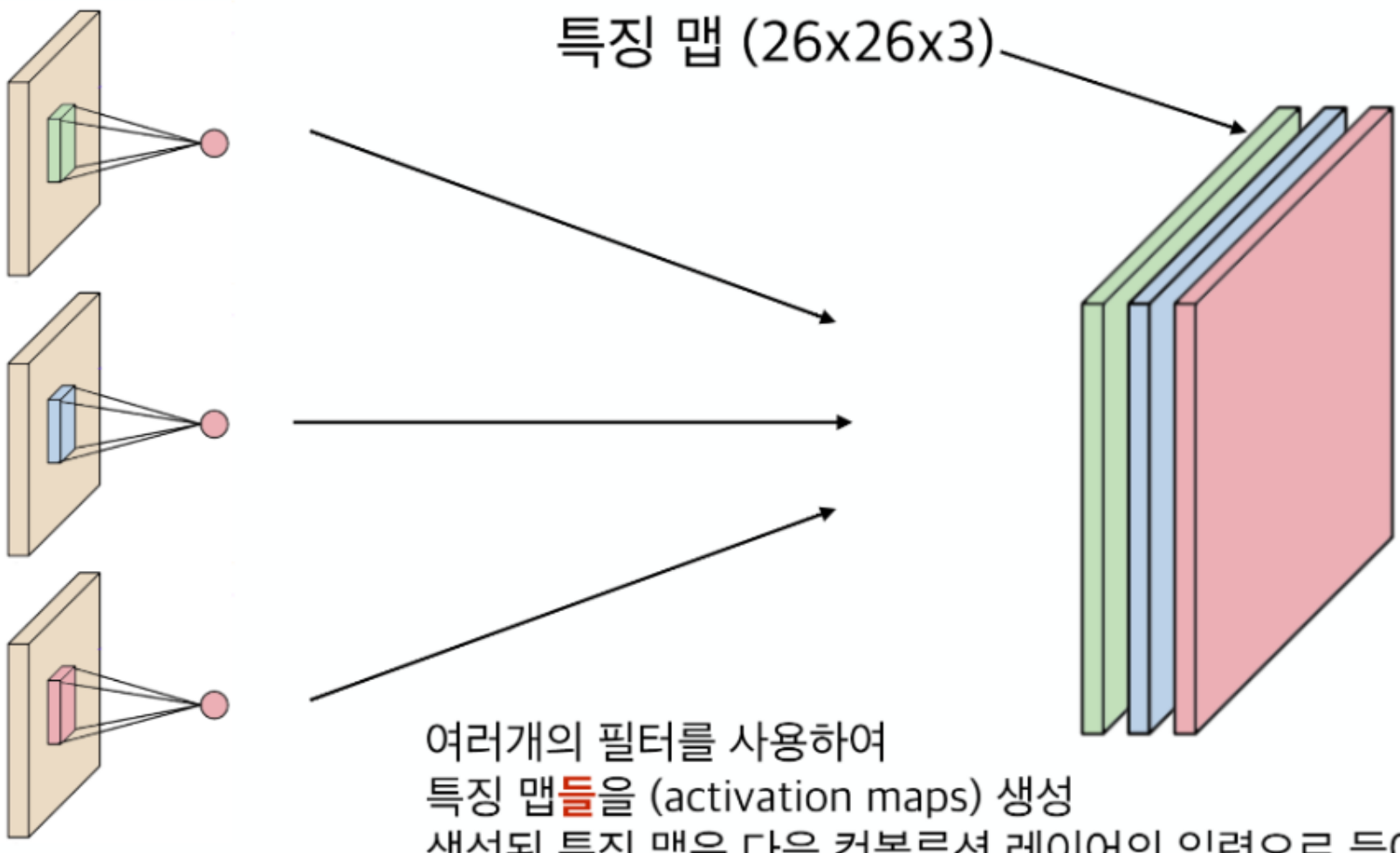
$$w^T x + b$$

필터 (혹은 커널)

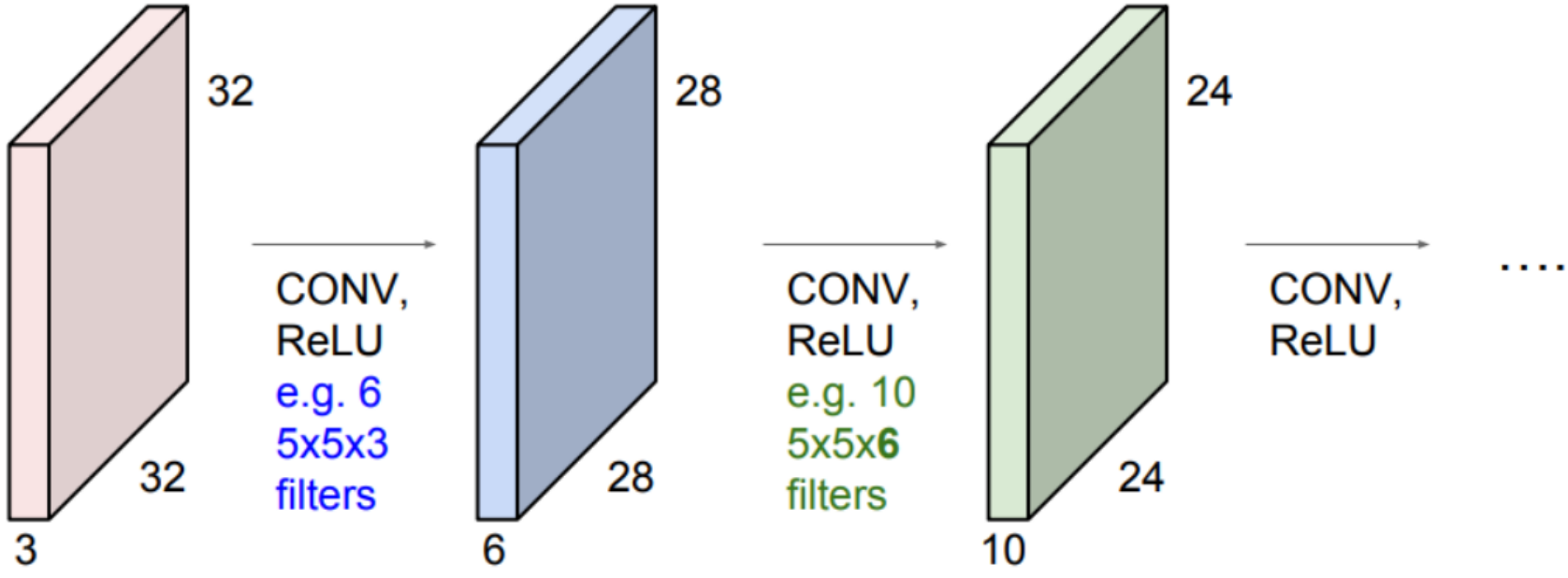
- 학습해야 할 **파라미터** (이 필터는 27개의 weight로 구성)
- 필터의 마지막 차원은 입력 이미지의 채널 차원과 같아야 함 (28x28x3 <-> 3x3x3)


입력 이미지 (28x28x3)
필터 (3x3x3)
영역을 슬라이드 하면서 전부 컨볼루션 하면
오른쪽과 같은 특징 맵을 얻을 수 있음
(혹은 activation map)
특징 맵 (26x26x1)


입력 이미지 (28x28x3)
필터 (3x3x3)
이전 필터와 다른 필터를 사용해서
다른 특징 맵을 얻을 수 있음
(서로 다른 weight)
특징 맵 (26x26x1)



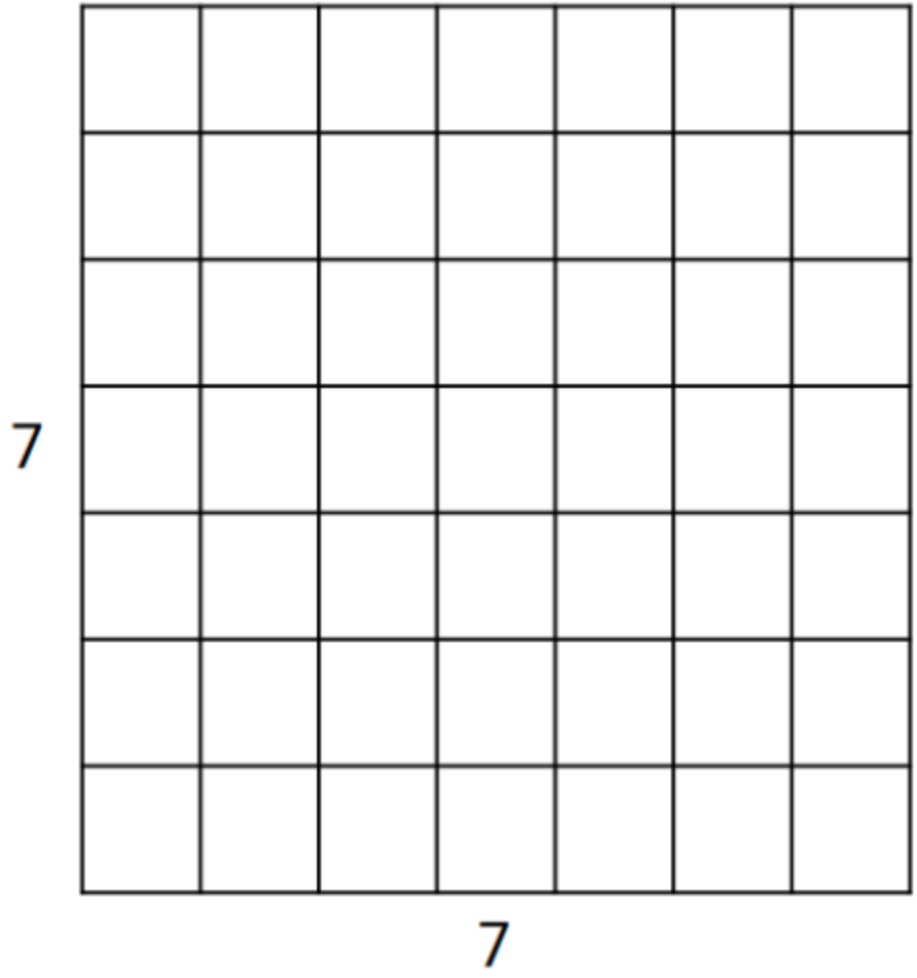
여러개의 필터를 사용하여
특징 맵들을 (activation maps) 생성
생성된 특징 맵은 다음 컨볼루션 레이어의 입력으로 들어감


32
32
3
CONV,
ReLU
e.g. 6
5x5x3
filters
28
28
6
CONV,
ReLU
e.g. 10
5x5x6
filters
24
24
10
CONV,
ReLU
....

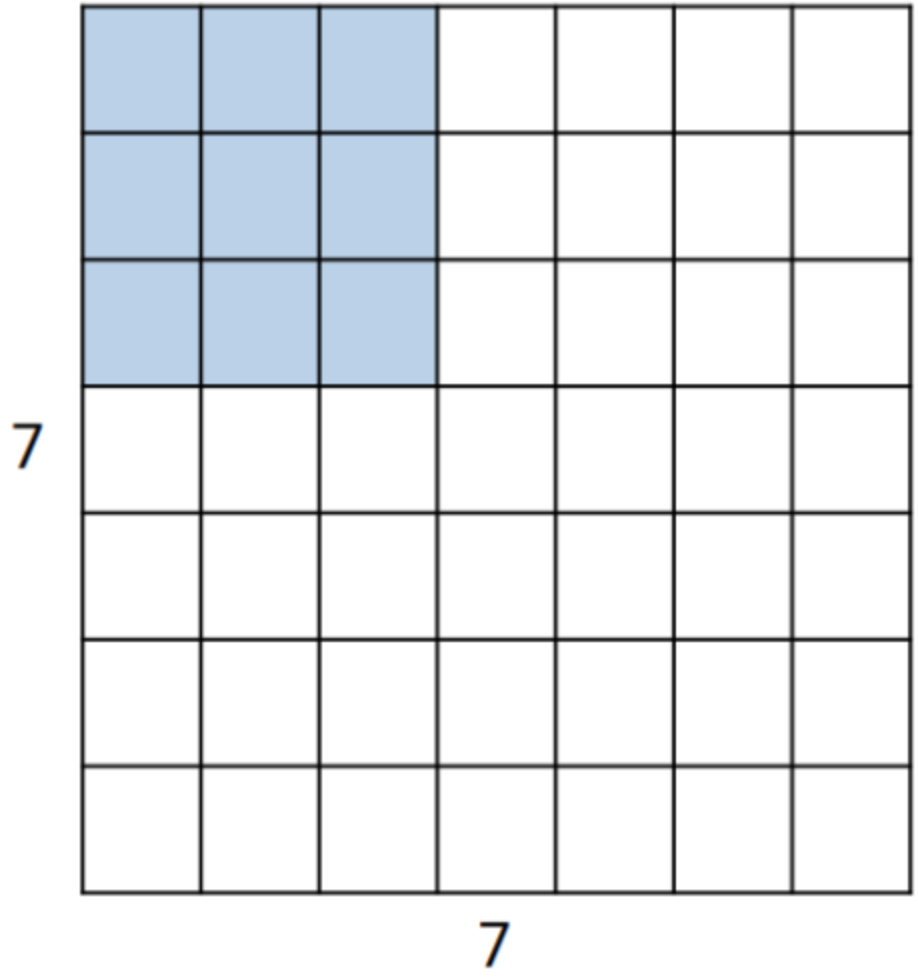
Convolution example

`7 * 7 image convolution`

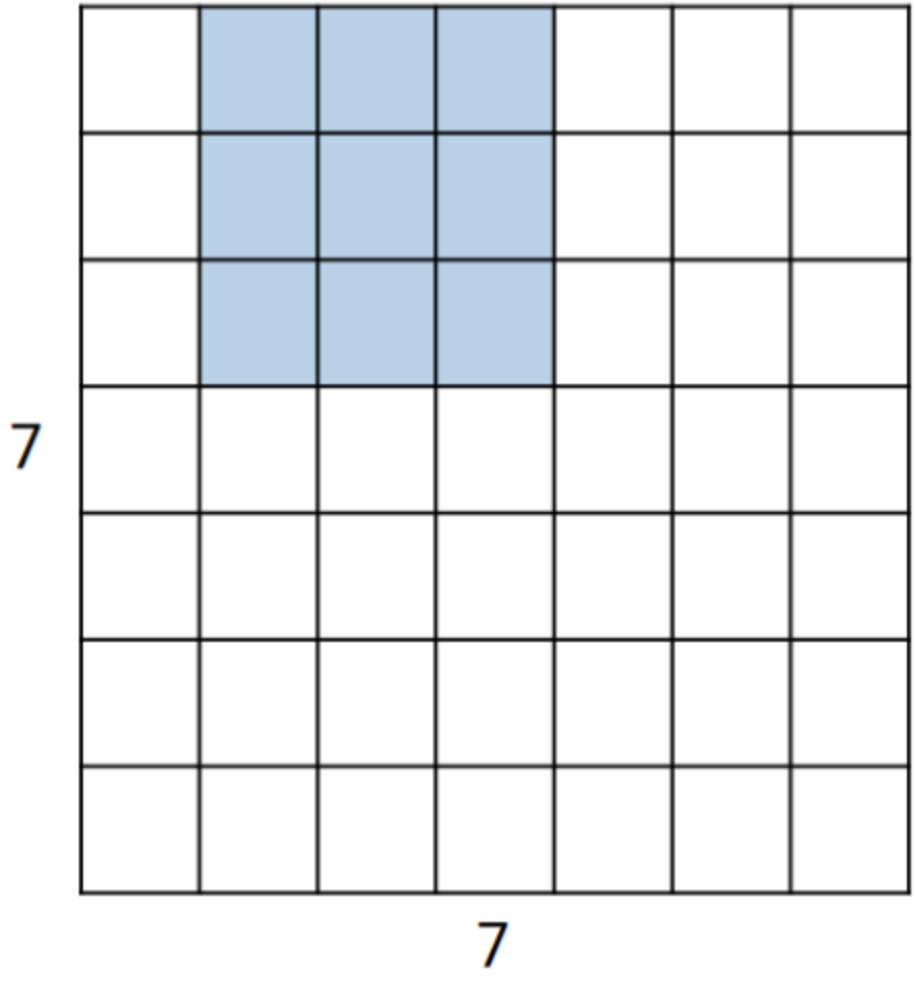
- `7 x 7 image` 를 `3 x 3 filter` 를 이용하여 Convolution



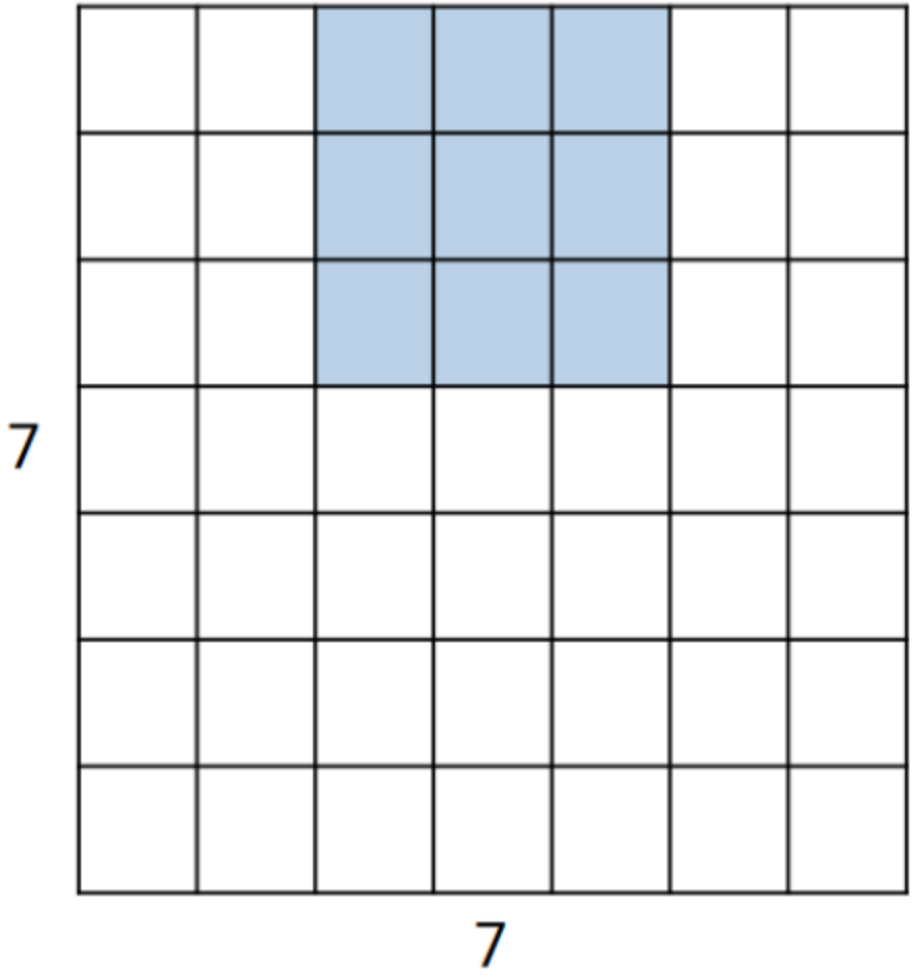
- `image 일부를 slice` 해서 `3 x 3 filter` 적용



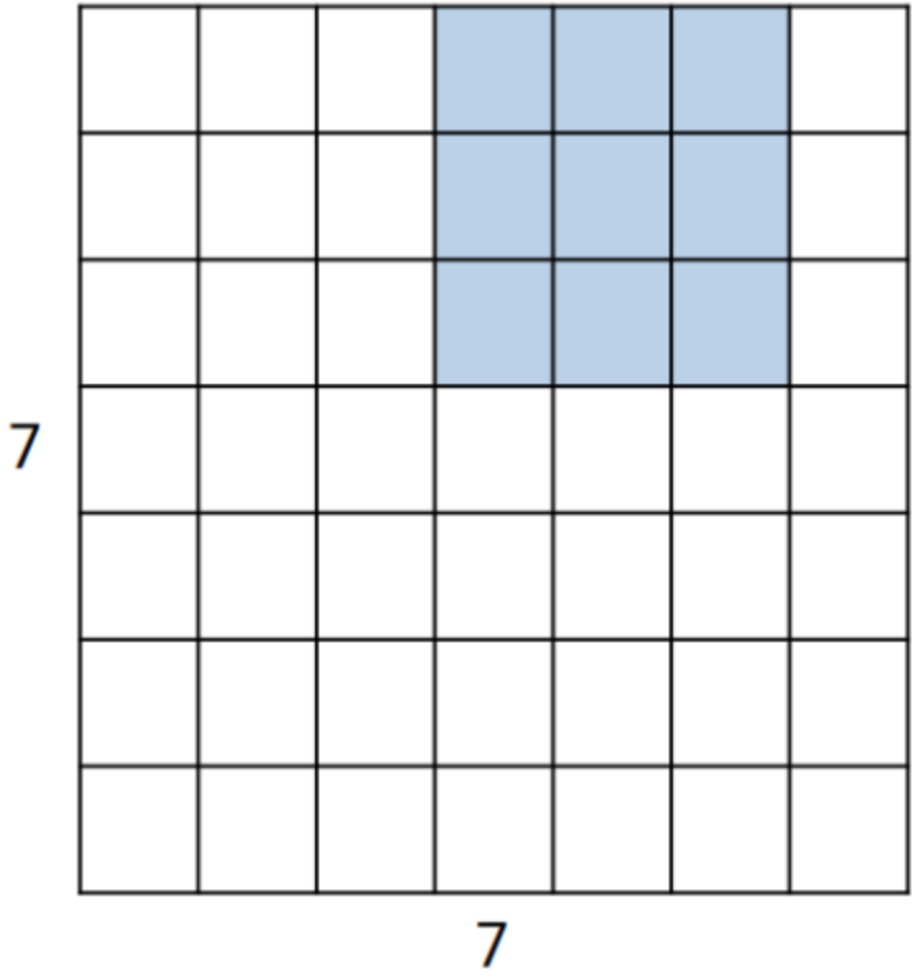
- 옆으로 한칸 이동
- 이동하는 크기를 **stride(간격)** 이라고 부르며 이 예제에서는 **stride가 1** 이다



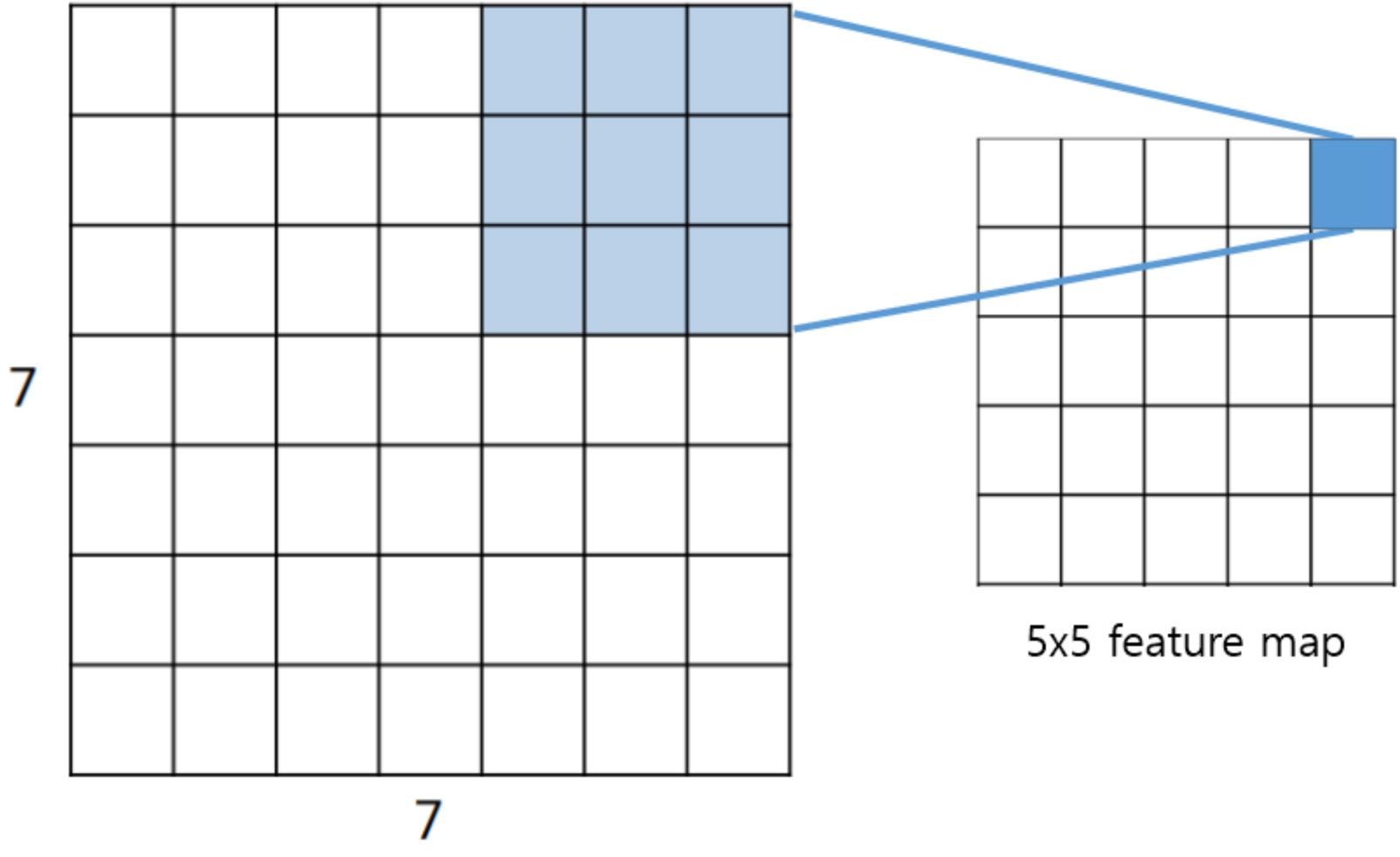
- 옆으로 한칸 이동



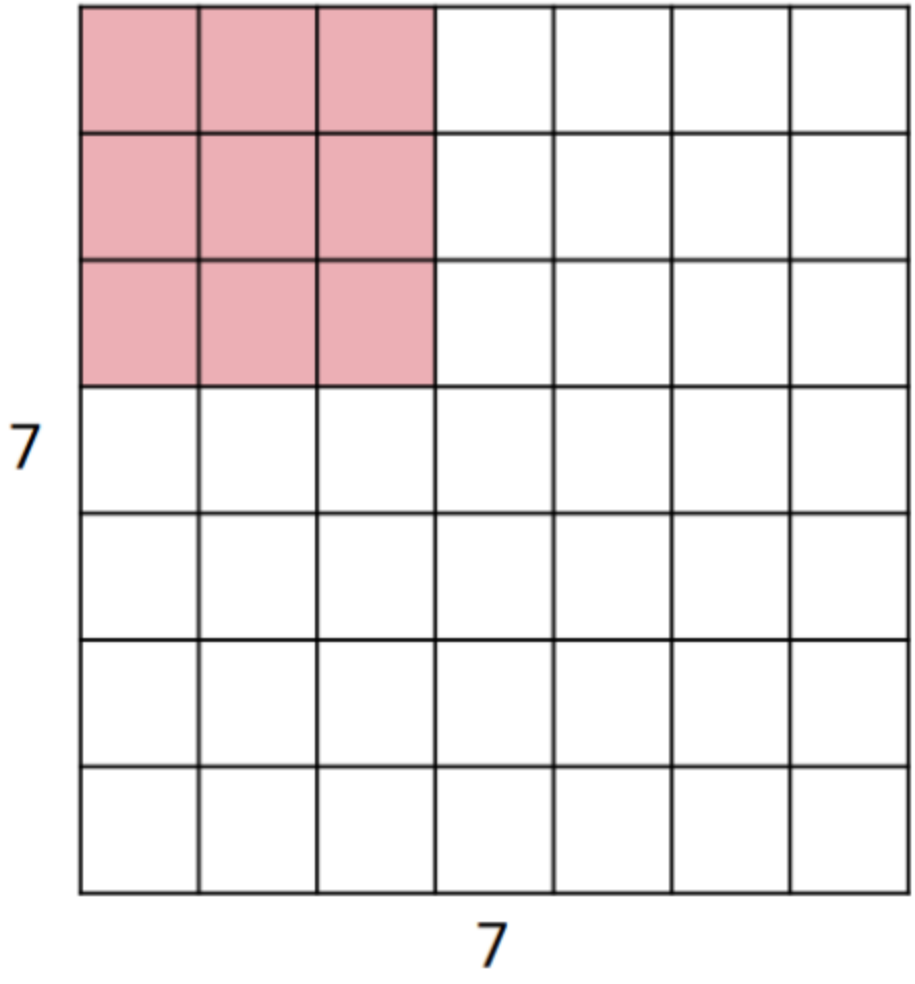
- 옆으로 한칸 이동



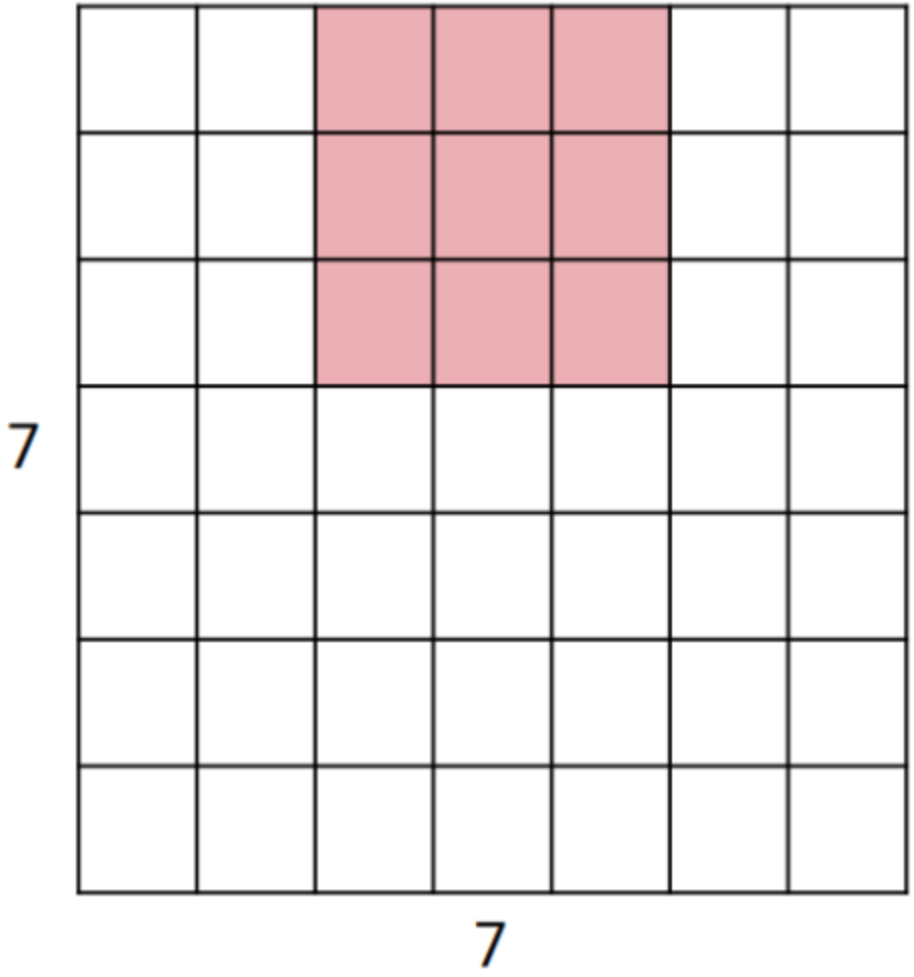
- `7 x 7 image` 에 `3 x 3 filter` 를 `stride 1` 로 적용
- `5 x 5 feature map` 이 생성



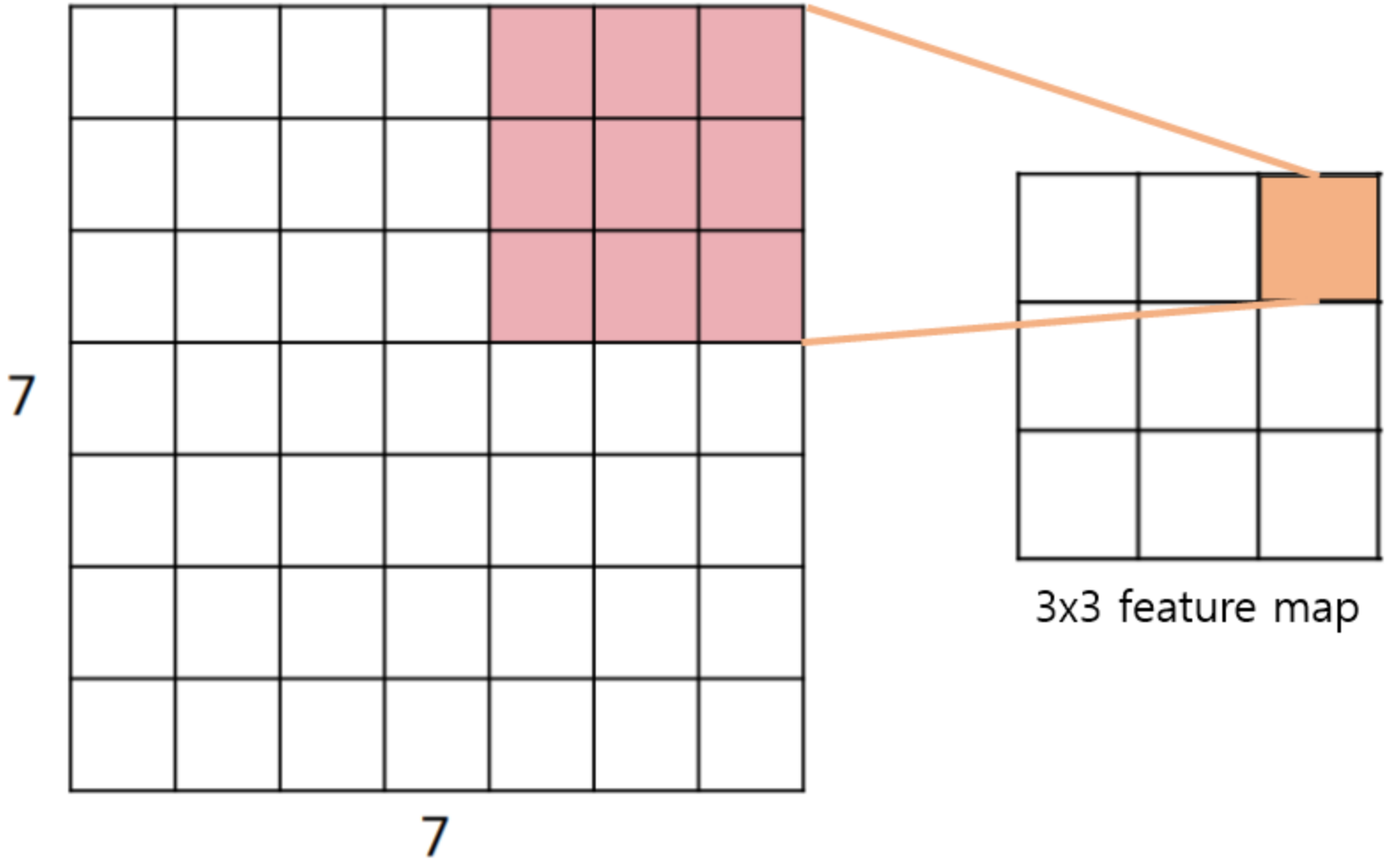
- `image 일부를 slice` 해서 `3 x 3 filter` 적용
- `stride 2`



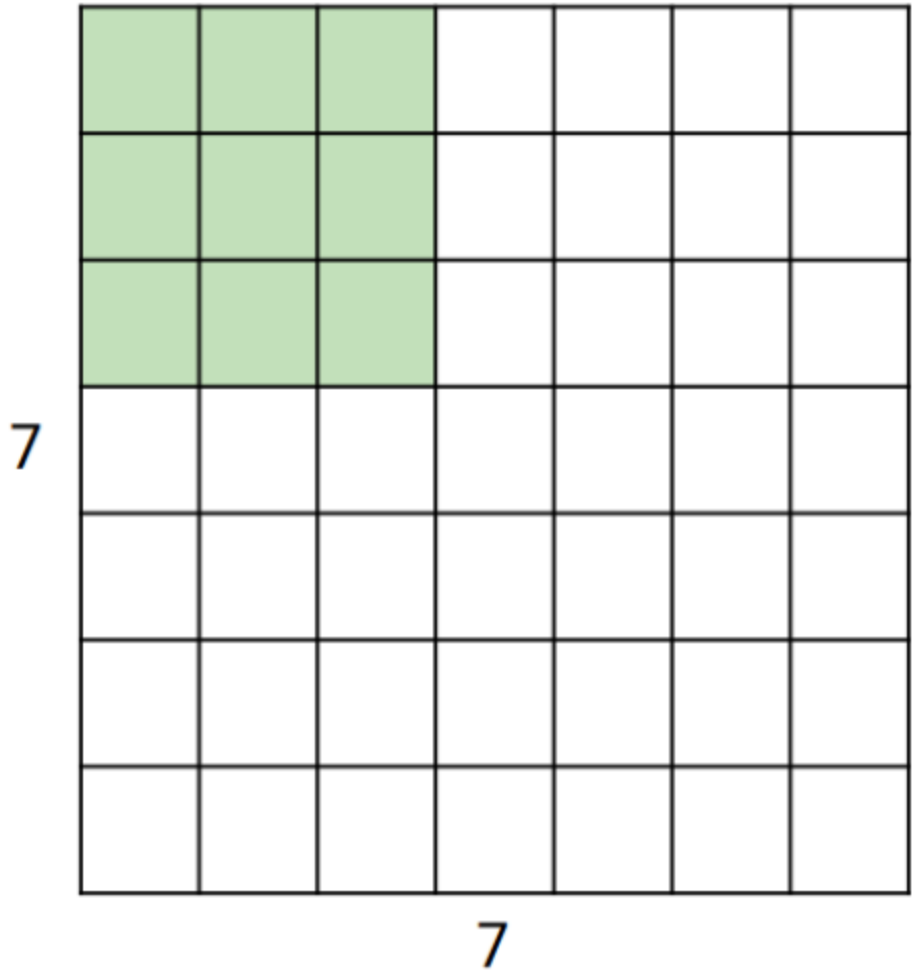
- 옆으로 두칸 이동



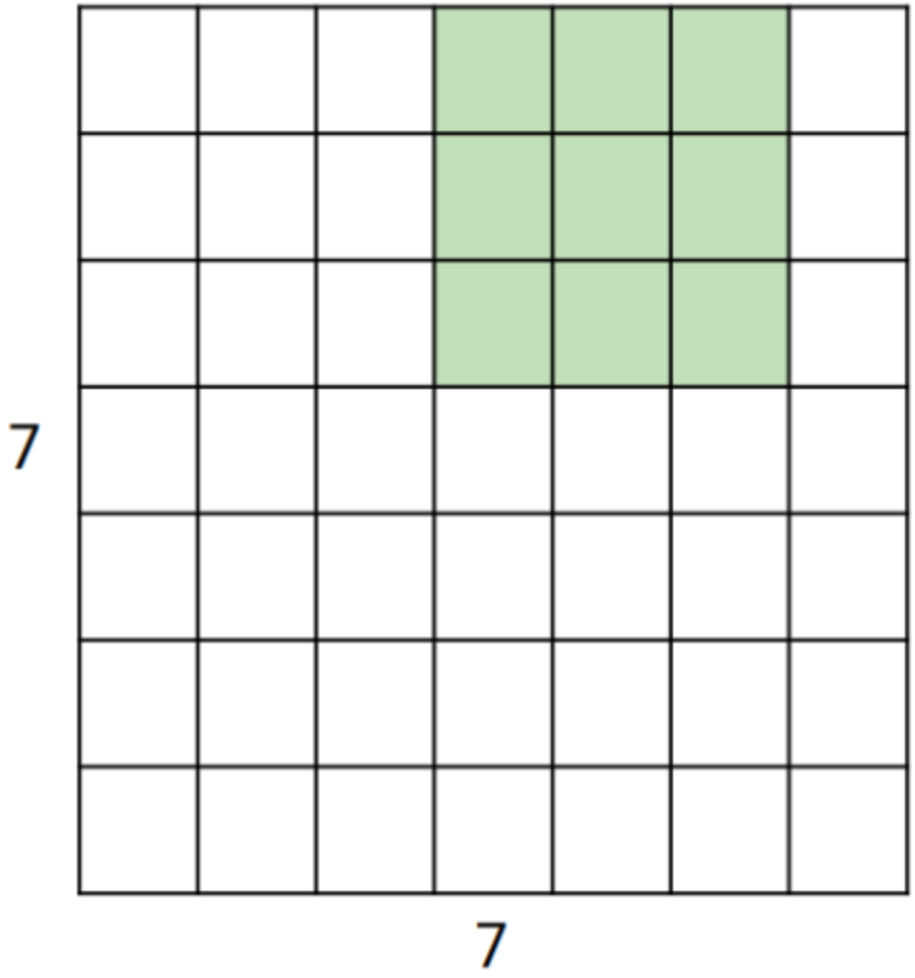
- 옆으로 두칸 이동
- `7 x 7 image` 에 `3 x 3 filter` 를 `stride 2` 로 적용
- `3 x 3 feature map` 이 생성



- `image 일부를 slice` 해서 `3 x 3 filter` 적용
- `stride 3`



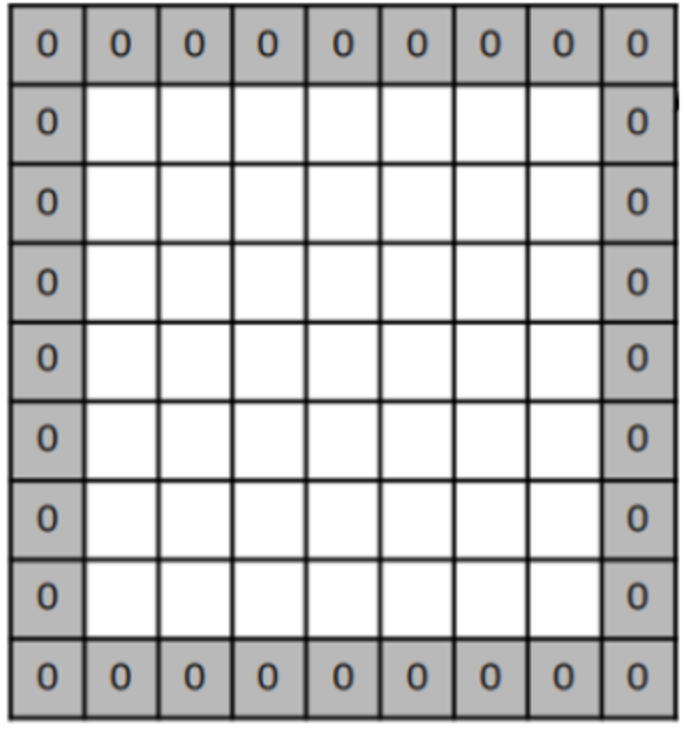
- 옆으로 세칸 이동
- **But 다음 번에 stride를 적용 불가!!!**



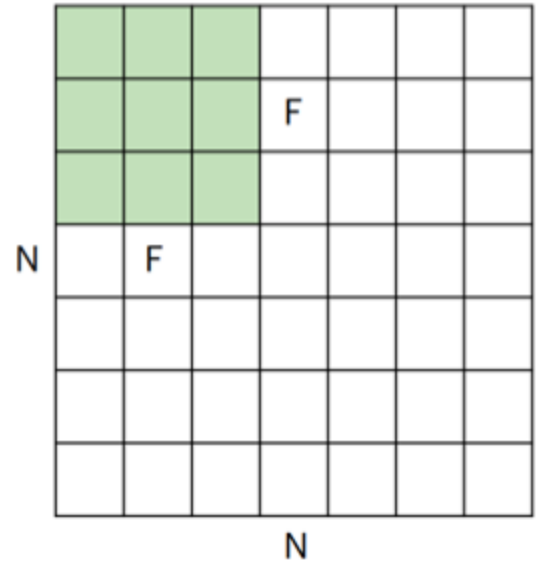
Padding

- **Convolution 의 문제점**
  - `convolution` 연산을 진행하면 `원본 image` 가 너무 빠르게 줄어듬
  - 따라서 `Layer` 를 깊게 쌓을수 없음

- **Padding**
  - `input map` **끝부분에** `0으로 이루어진 값` **을 추가**
  - 어차피 0이기 때문에 **결과를 손상시키지는 않음**
  - 0으로 표시함으로서 **모서리의 영역** 을 표시해 주는 역할을 하기도 함
  - 입력과 출력의 크기가 똑같아지는 결과를 도출해 줄 수도 있음

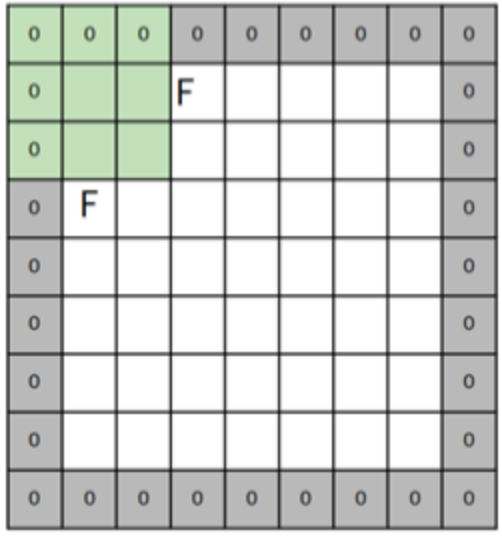


feature map 크기 계산



- **`feature Map 크기 계산 공식`**
  - **`(N - F) / stride + 1`**
  - *example*
    - stride 1 -> (7-3)/1+1 = 5
    - stride 2 -> (7-3)/2+1 = 3
    - stride 3 -> (7-3)/3+1 = 2.3 **`결과값이 정수로 나눠 떨어지지 않으면 에러!!`**

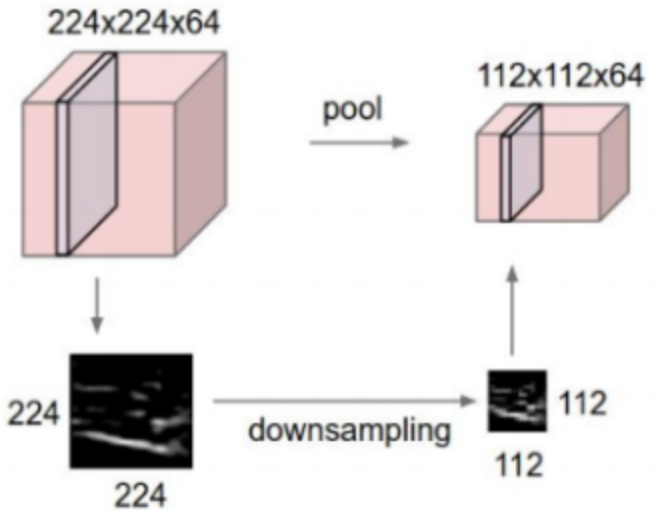
- `만약 padding 을 지정해 준다면`



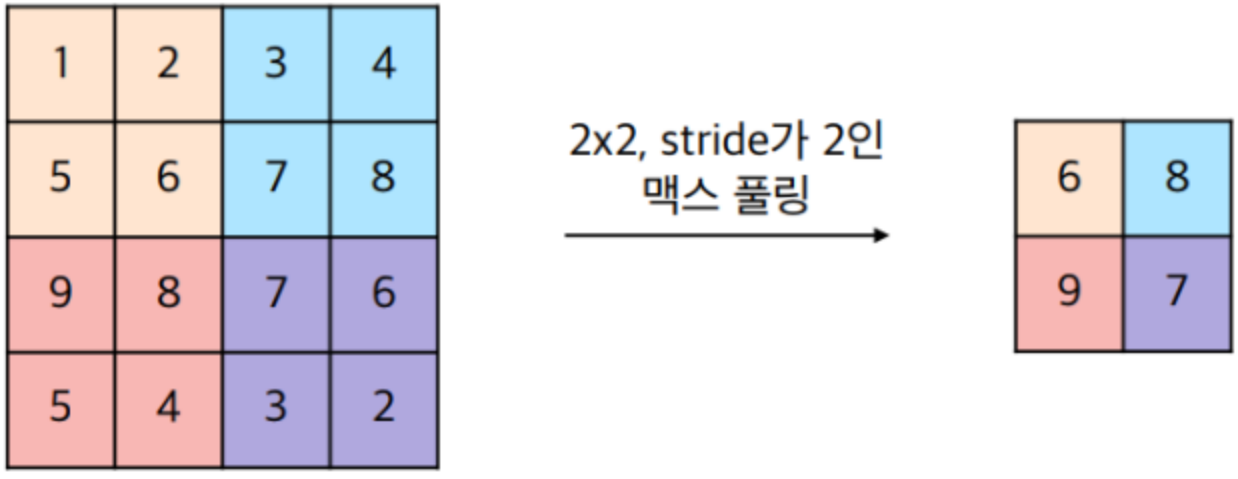
- `feature Map 크기 계산 공식`
  - `(N – F + pad * 2) / stride + 1`
  - *example*
    - 3x3 stride 1, padding 1 -> (7-3+2) / 1+1 = 7
    - 5x5 stride 1, padding 2 -> (7-3+4) / 2+1 = 7
    - 7x7 stride 1, padding 3 -> (7-3+6) / 3+1 = 7

Pooling Layer

- **입력 이미지(특징 맵)의 크기를 낮추는 레이어**
- 연산 속도를 높이고, 메모리 요구량을 낮출 수 있음
- 일반적으로 최대값을 이용하는 `Max pooling Layer`를 사용함
- `Pooling Layer` 는 파라미터가 따로 없는 단순연산

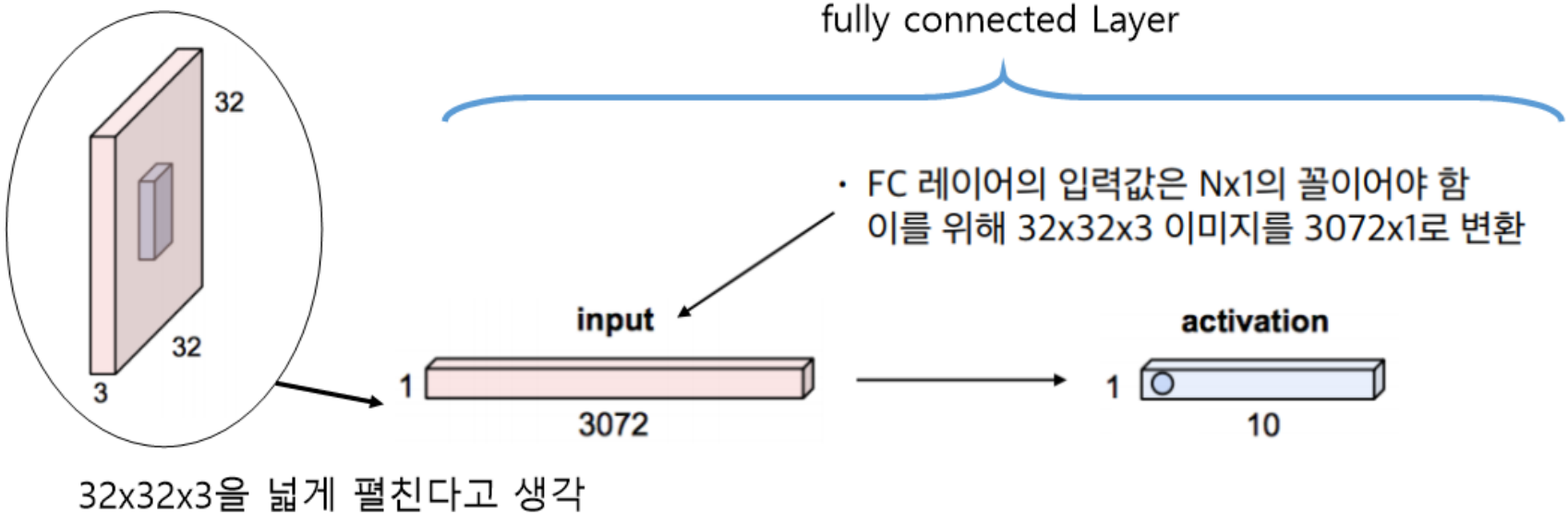


Max Pooling

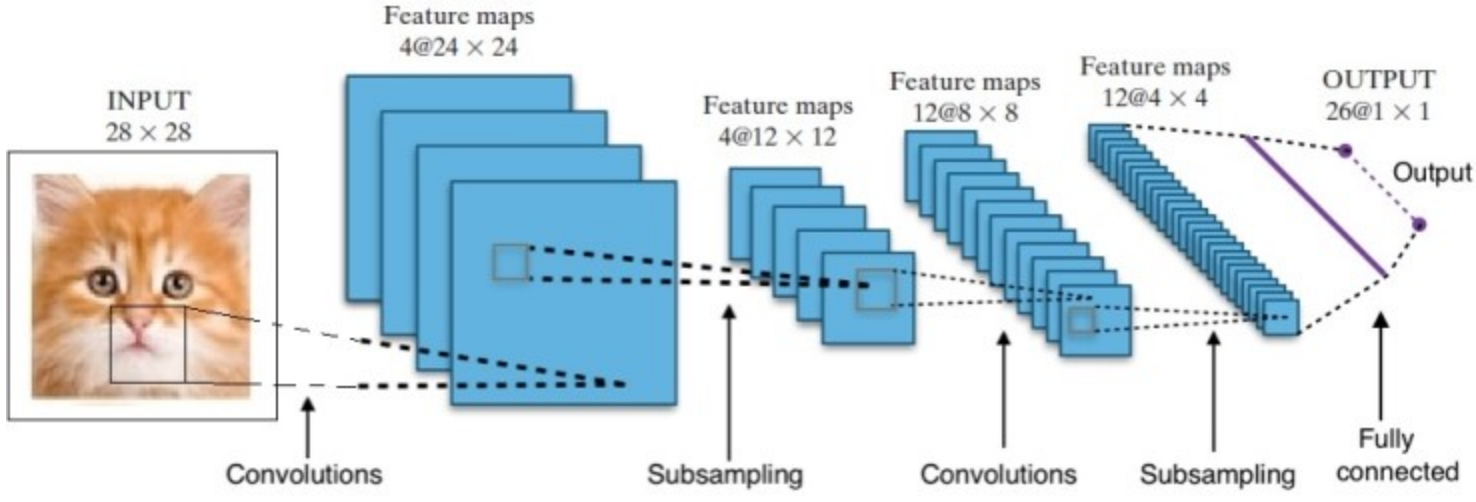


Fully connected Layer (FC Layer)

- `Fully connected Layer`
  - 일반적인 Neural Network란 완전히 동일
  - convolution layer를 flatten한 값을 입력으로 받음
  - `Classify` 역할을 한다고 보면됨



Convolution Architecture



- **Convolution Layer** : image 의 **feature extractor**
- **fully connected layer** : 문제 해결을 위한 **Classifier**