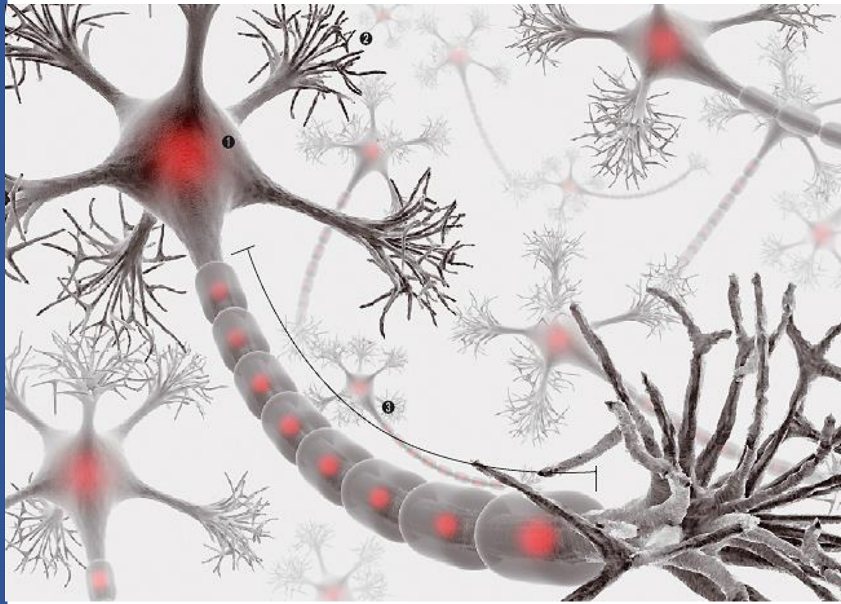
컨볼루션 신경망
(Convolutional Neural Network)

학습 목표

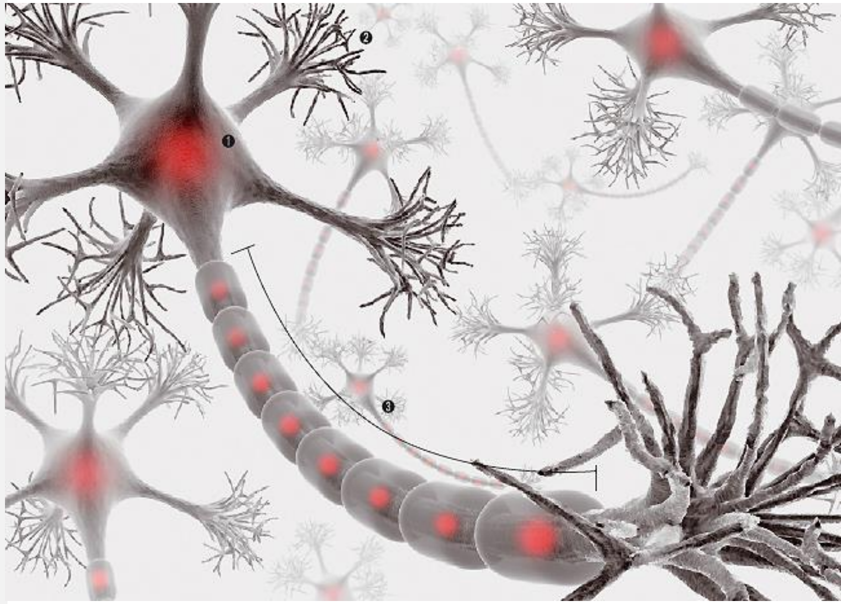
- Convolutional Neural Network의 작동 원리를 이해한다.

주요 내용

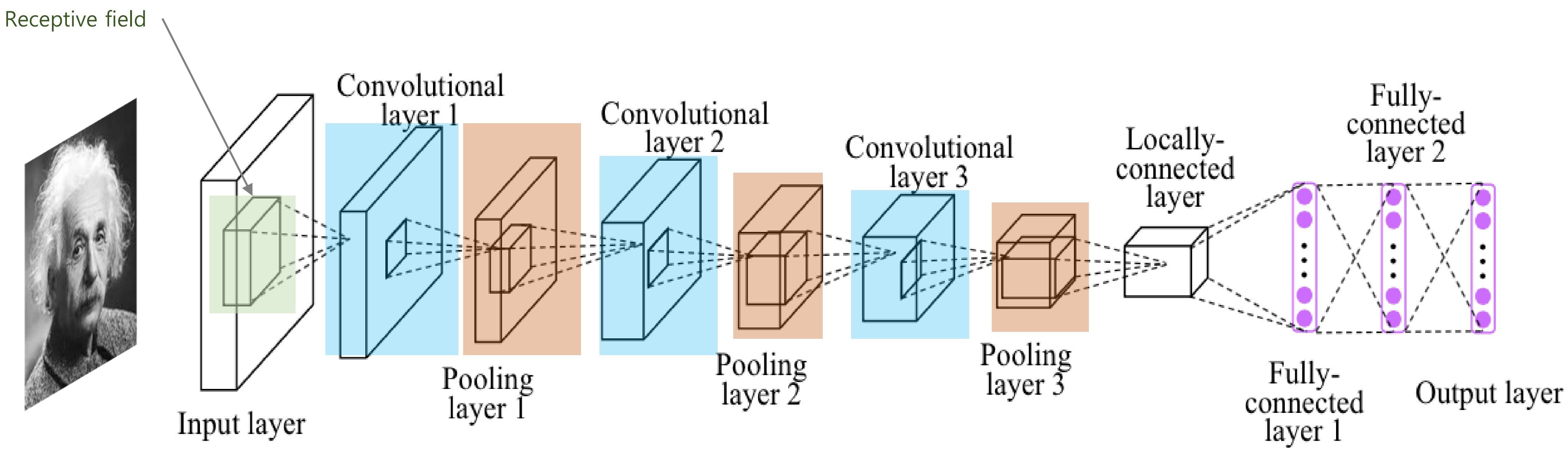
1. CNN 아키텍처
2. Convolution
3. Stride and Padding
4. CNN 가정사항



1 CNN 아키텍처



합성곱 신경망 (CNN : Convolutional Neural Network)

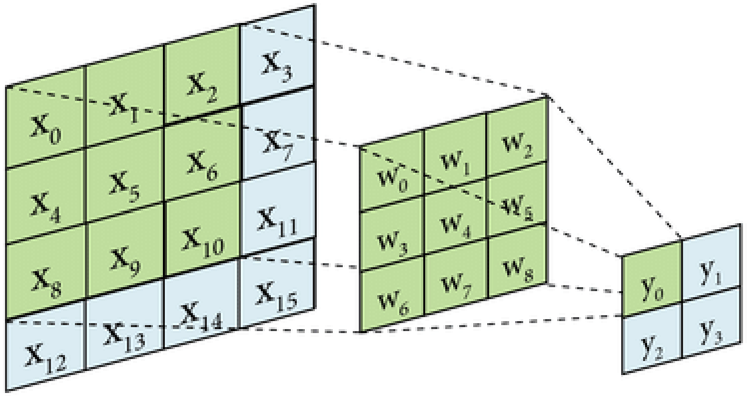


- **이미지와 같은 고차원 데이터를 처리하기 위한 신경망**
- **기본 연산이 Convolution인 신경망**으로 Convolution Layer와 Pooling Layer가 번갈아 가면서 나오는 구조
- 연산 방식과 구조는 Feedforward Network과 동일하지만 **Convolution Filter 단위로 파라미터를 공유**한다는 점이 가장 큰 차이점

Convolution

Convolution

- 이미지를 변환하거나 특징을 추출하는 연산



$$
\begin{aligned}
y_0 = w_0x_0 + w_1x_1 + w_2x_2 \\
+ w_4x_4 + w_5x_5 + w_5x_5 \\
+ w_6x_6 + w_7x_7 + w_8x_8
\end{aligned}
$$

Edge Detection

-1	0	+1
-2	0	+2
-1	0	+1

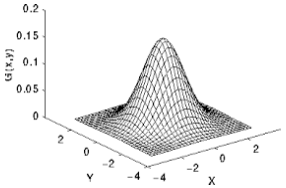
Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Smoothing

$$
\frac{1}{115}
\begin{bmatrix}
2 & 4 & 5 & 4 & 2 \\
4 & 9 & 12 & 9 & 4 \\
5 & 12 & 15 & 12 & 5 \\
4 & 9 & 12 & 9 & 4 \\
2 & 4 & 5 & 4 & 2
\end{bmatrix}
$$



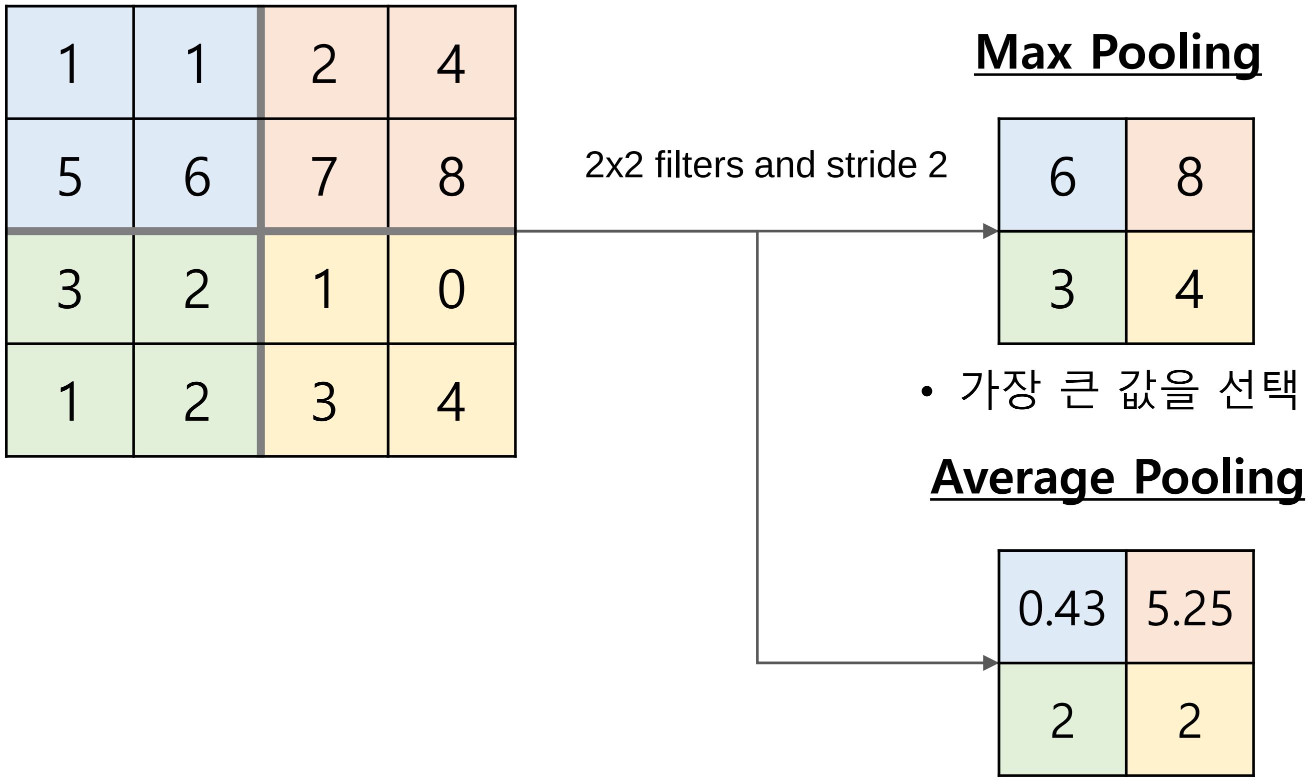
Original

Gaussian filter (σ = 10)

Sharpening

-1	-1	-1
-1	12	-1
-1	-1	-1

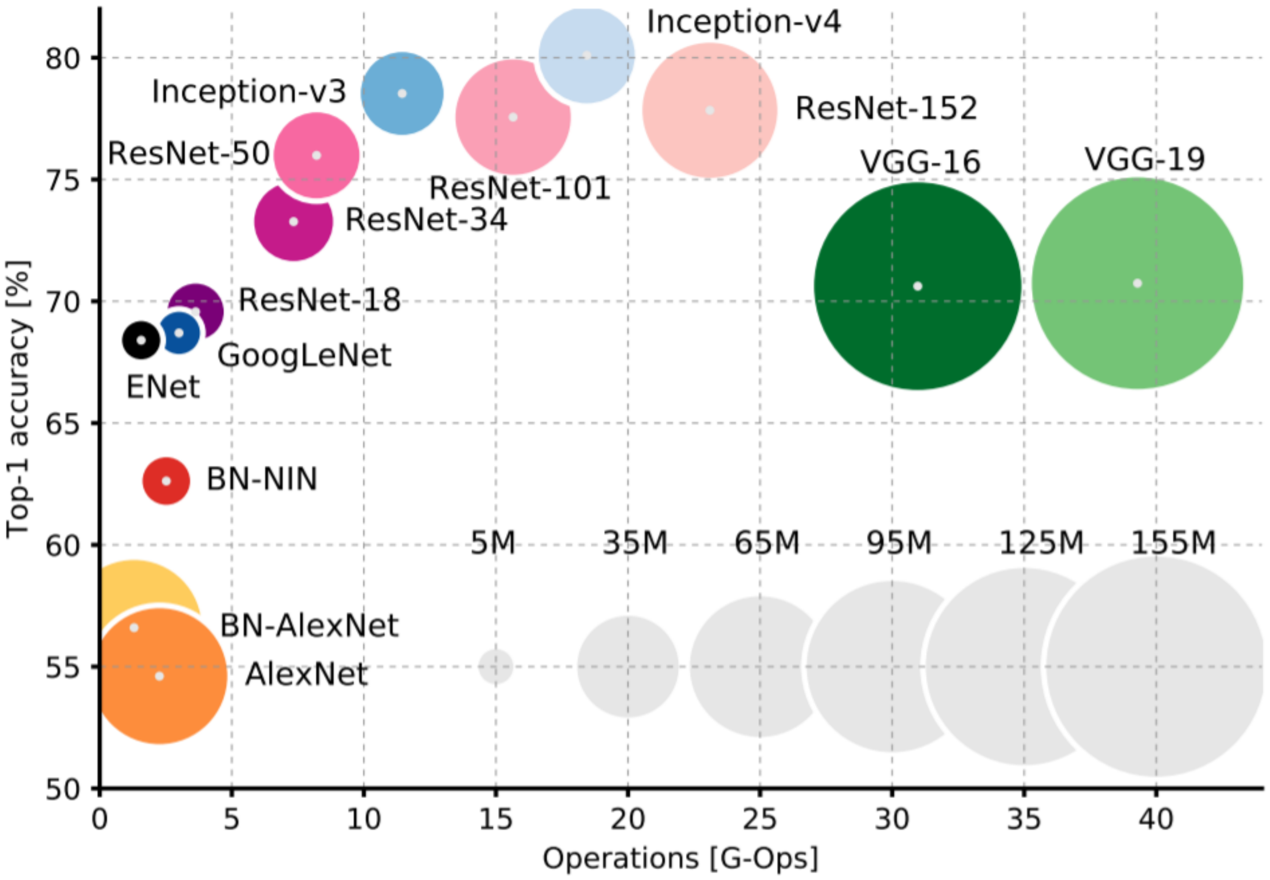
Pooling



- 가장 큰 값을 선택
- 평균 값으로 선택

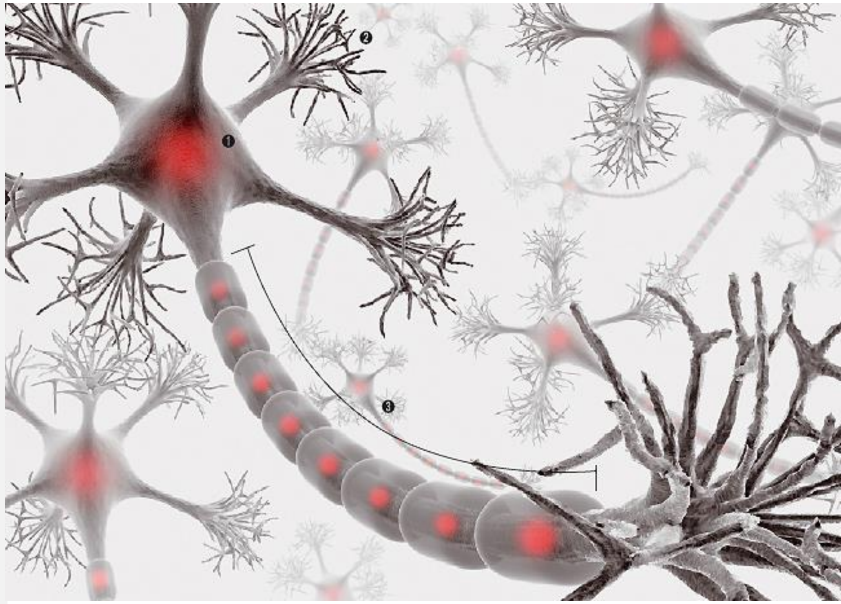
- 이미지 크기를 줄이는 연산
- 특정 영역에 대한 요약 통계량(summary statistics)을 구하는 연산
- L^2 norm이나 weighted average 등이 사용될 수 있다.

다양한 CNN 모델

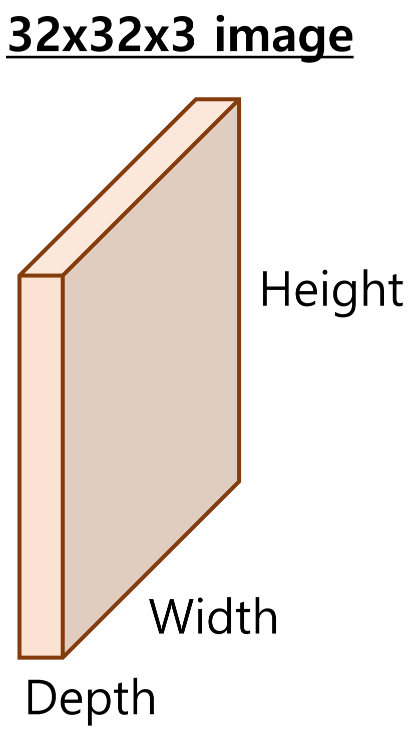


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

2 Convolution 연산

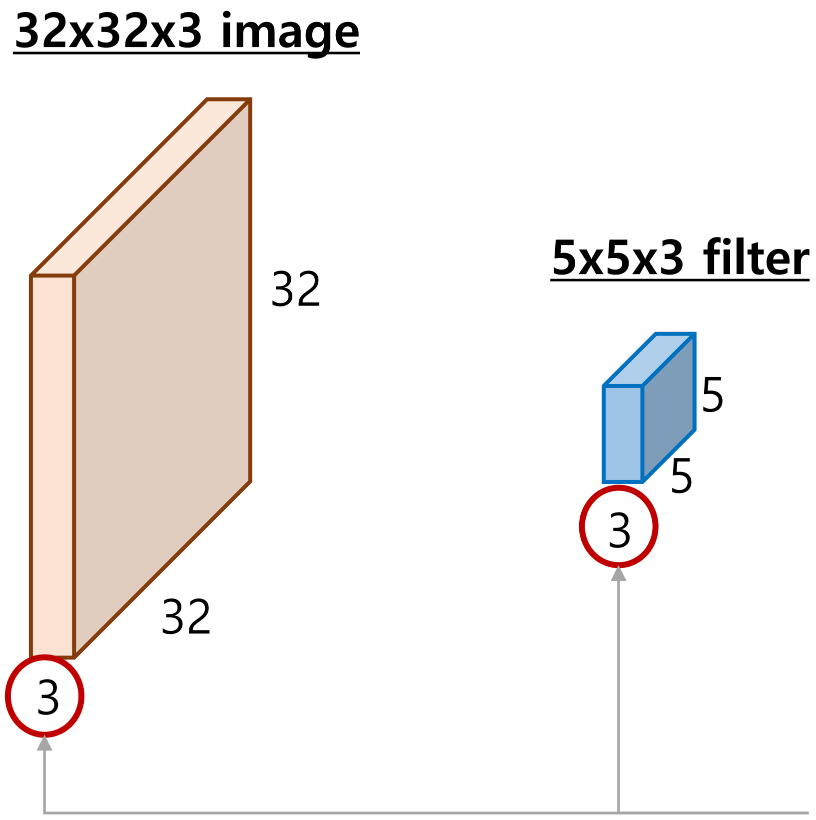


Input Image



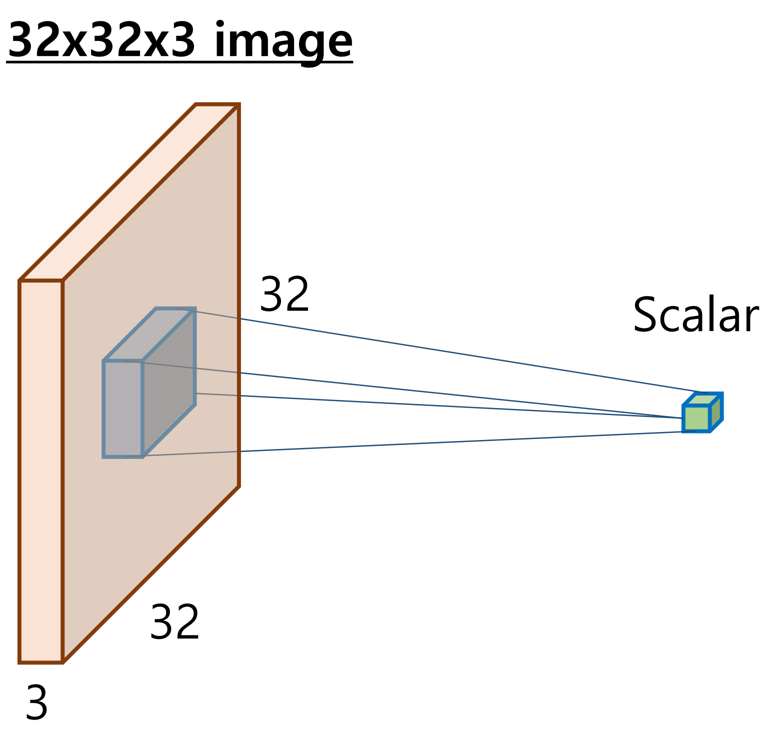
[Width] x [Height] x [Depth]

Filter

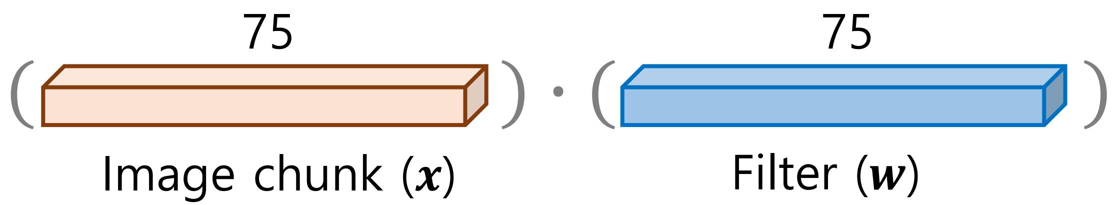


Image와 Filter의 Depth가 동일해야
Convolution 연산을 할 수 있다.

Spatial Dot Product



- Image : 5x5x3 chunk
- Filter : 5x5x3

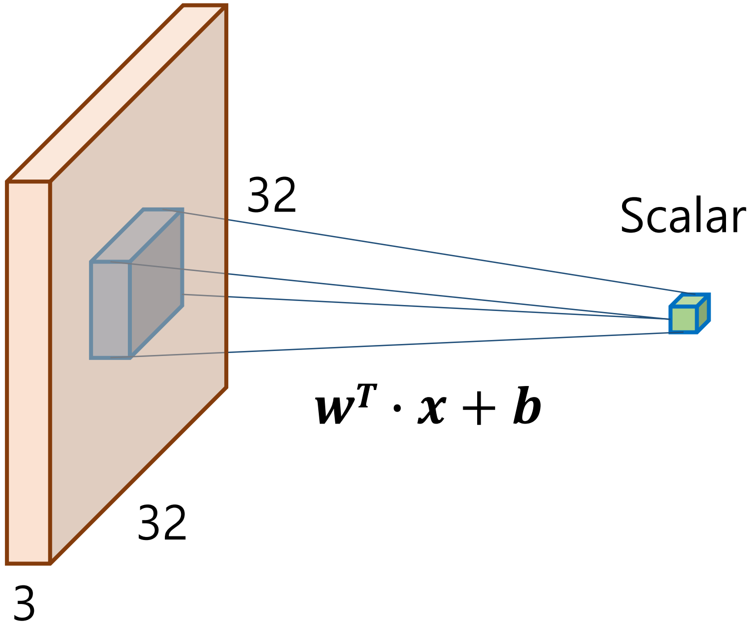


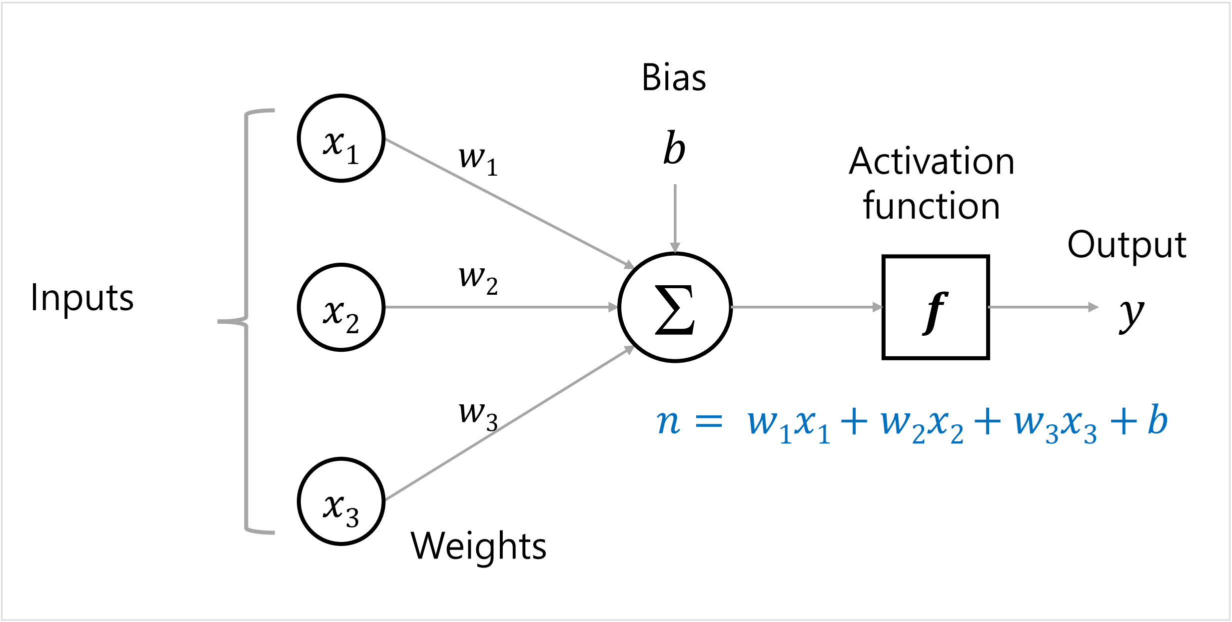
$$\boldsymbol{w}^T \cdot \boldsymbol{x} + \boldsymbol{b}$$

- Spatial dot product는 75 dimensional vector의 dot product와 동일

Spatial Dot Product

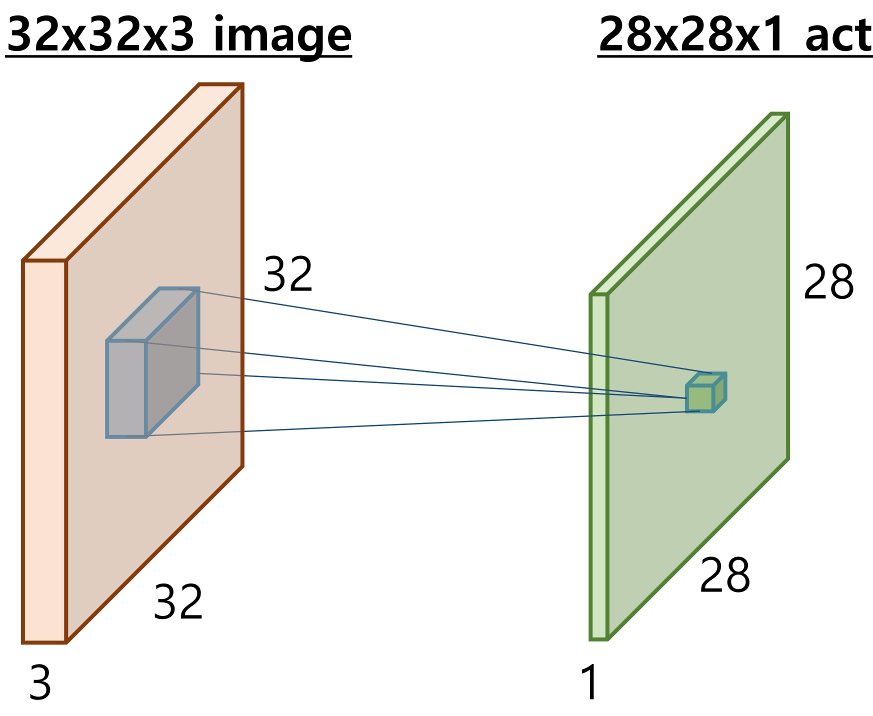
뉴런의 관점을 생각해보자!





Local Connectivity를 갖는 뉴런

Activation map



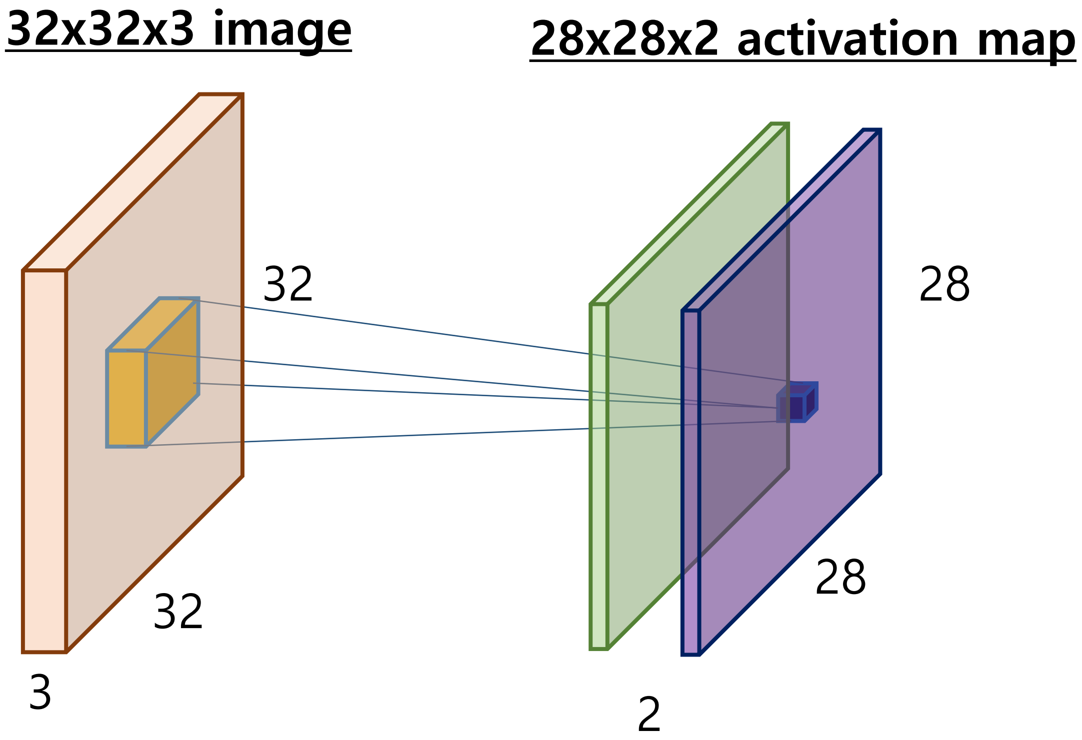
- 모든 pixel에 대해 convolution을 하면 activation map이 한 개 생성됨

따라서, activation map은 28x28 뉴런 sheet이다.

1. 각 뉴런은 입력의 작은 영역에 연결
2. 모든 뉴런은 파라미터를 공유함

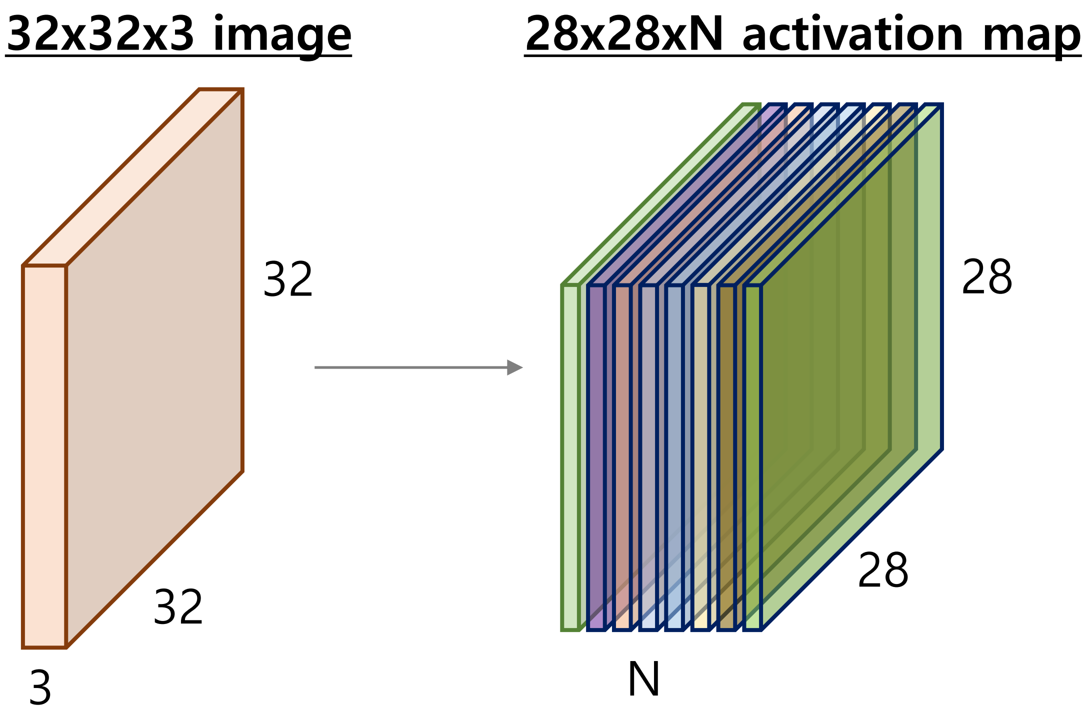
“5x5 filter” -> “5x5 receptive field for each neuron”

Activation map



서로 다른 filter 별로 activation map이 생성된다.

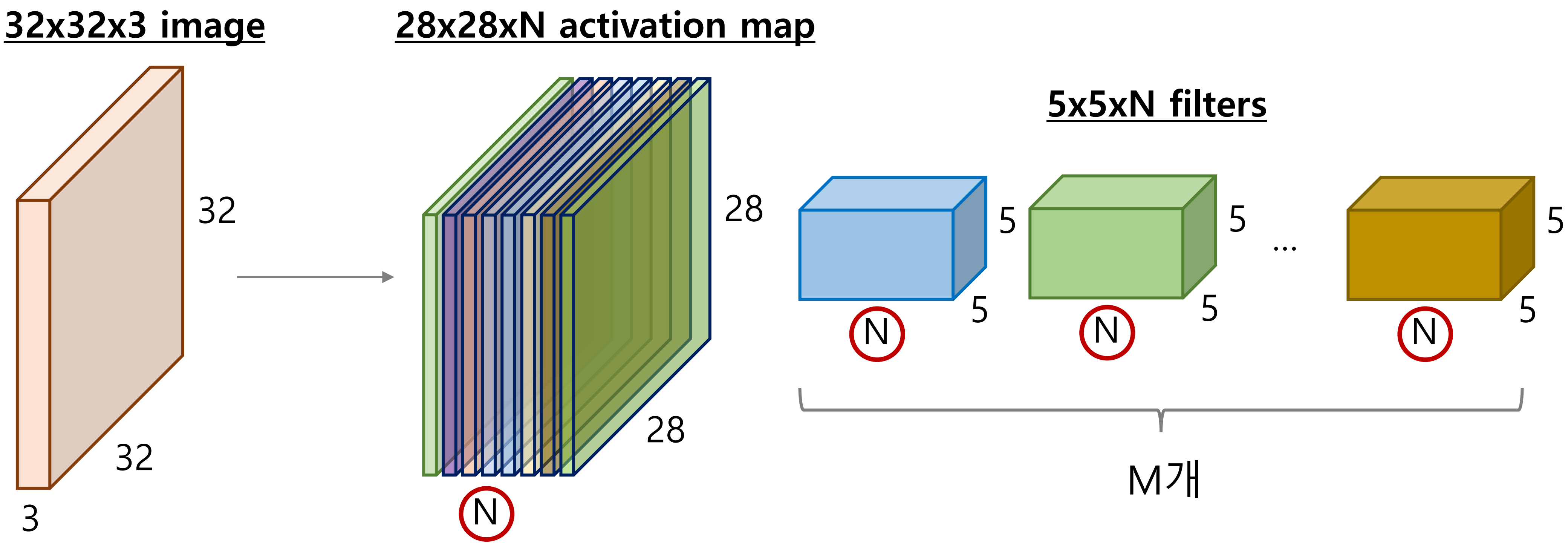
Activation map



N 개의 5x5x3 filter 적용 후

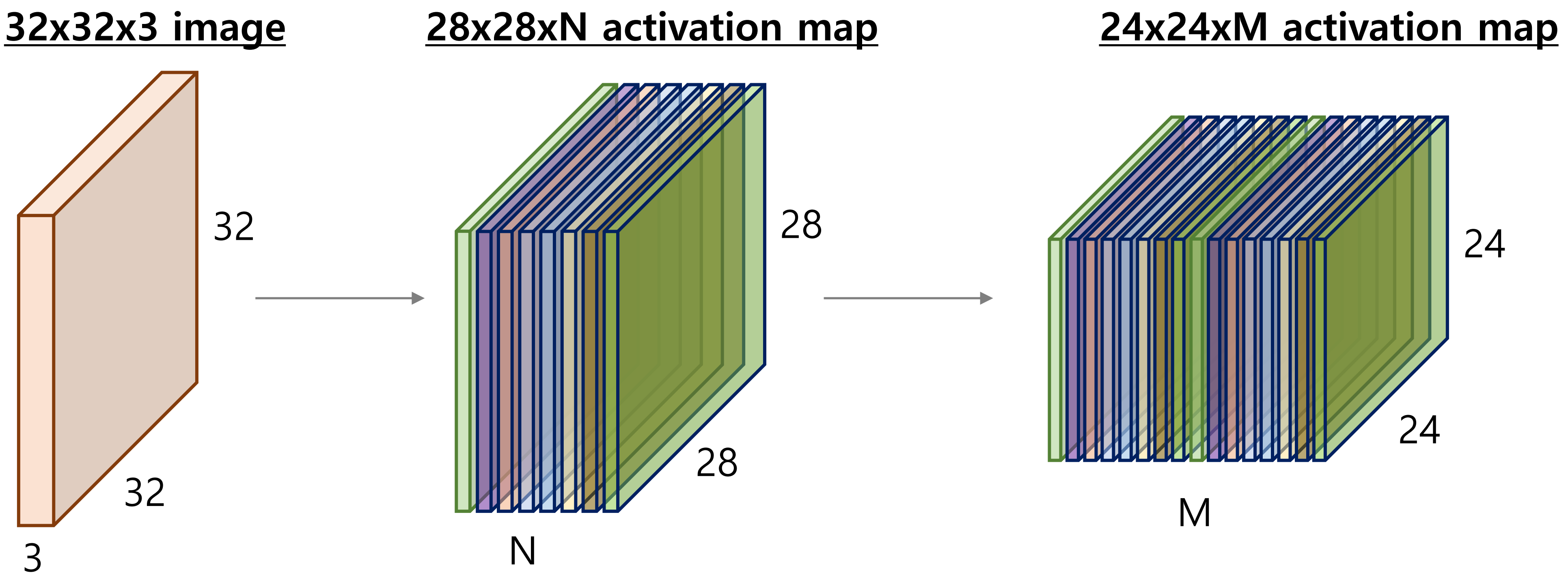
- 여러 Filter를 사용해서 다양한 feature를 학습할 수 있도록 함
- Activation map은 filter 개수만큼 depth를 갖게 됨

Filter - Again

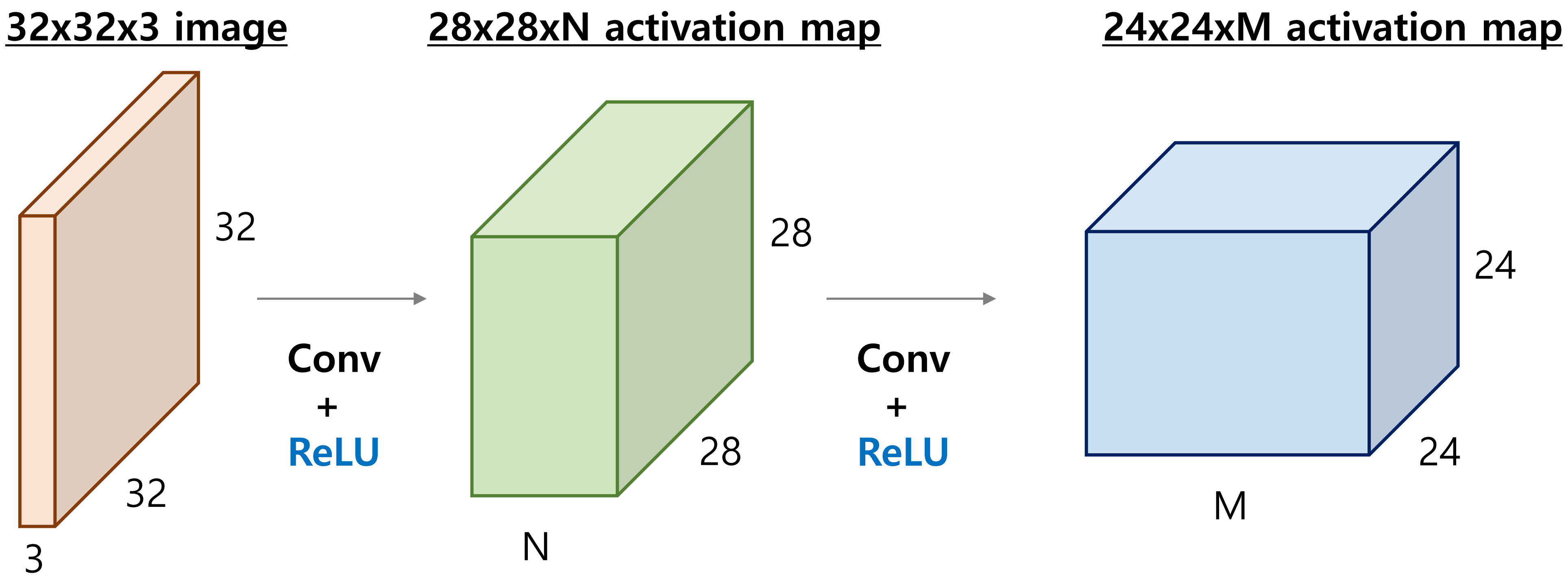


다음 layer의 filter depth
- Activation map의 depth와 동일해야 함
- 따라서, 이전 layer filter 개수와 동일

Activation map - Again

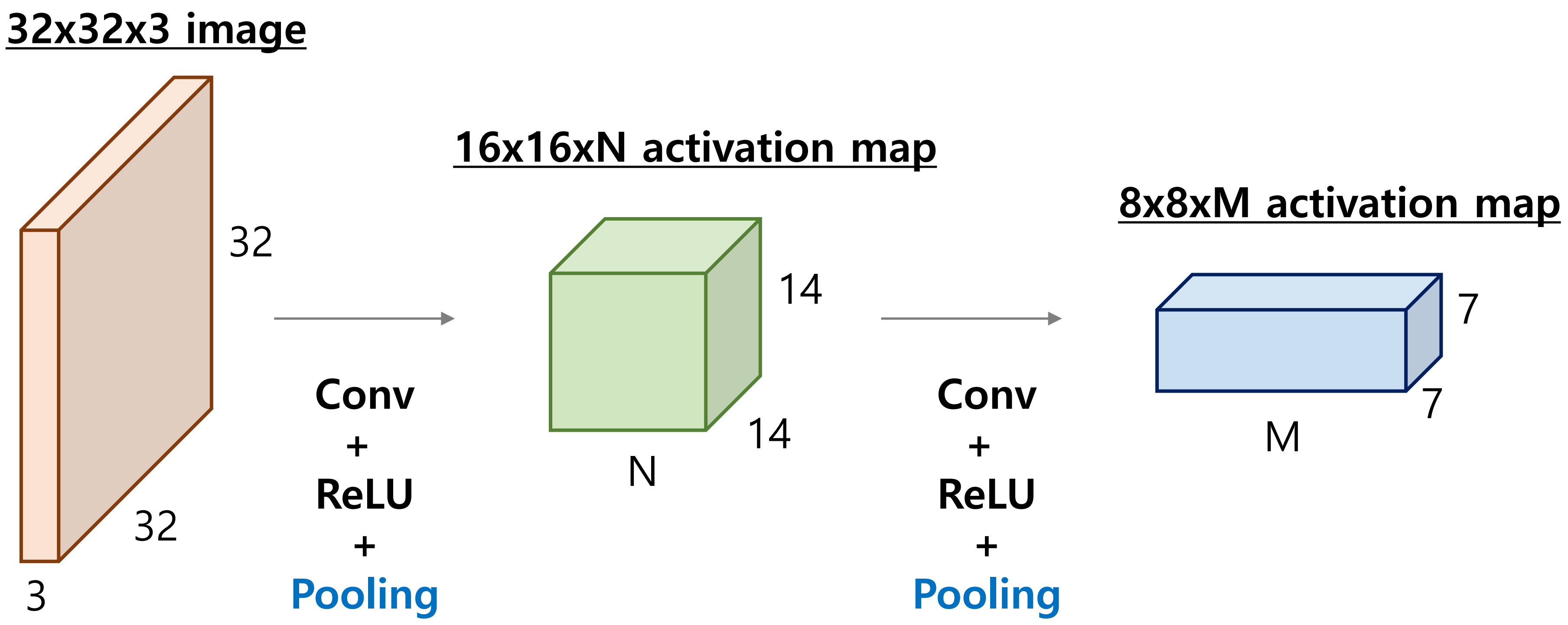


Activation 추가



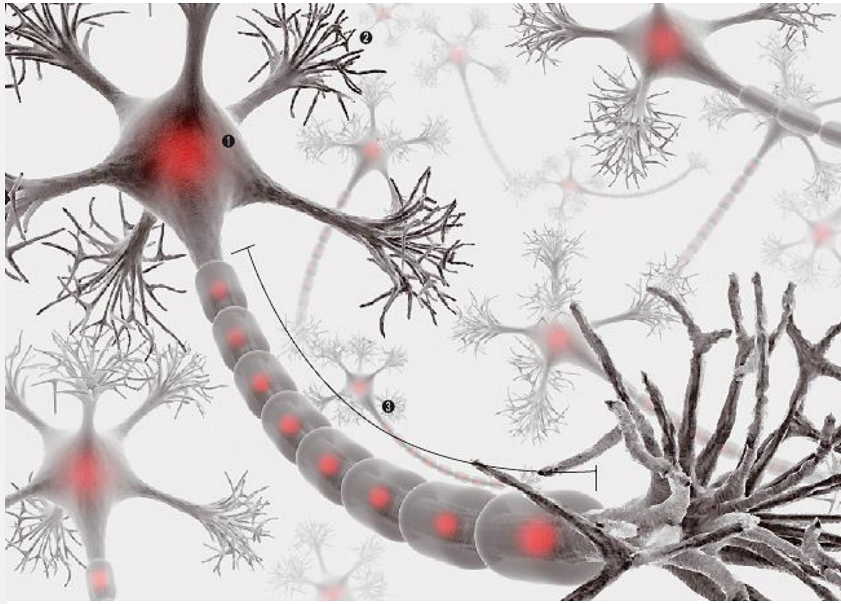
- 각 Layer 별로 convolution을 수행하고 activation function을 수행
- 비선형 연산을 통해 복잡한 분류 및 이미지 처리 연산을 수행

Pooling 추가



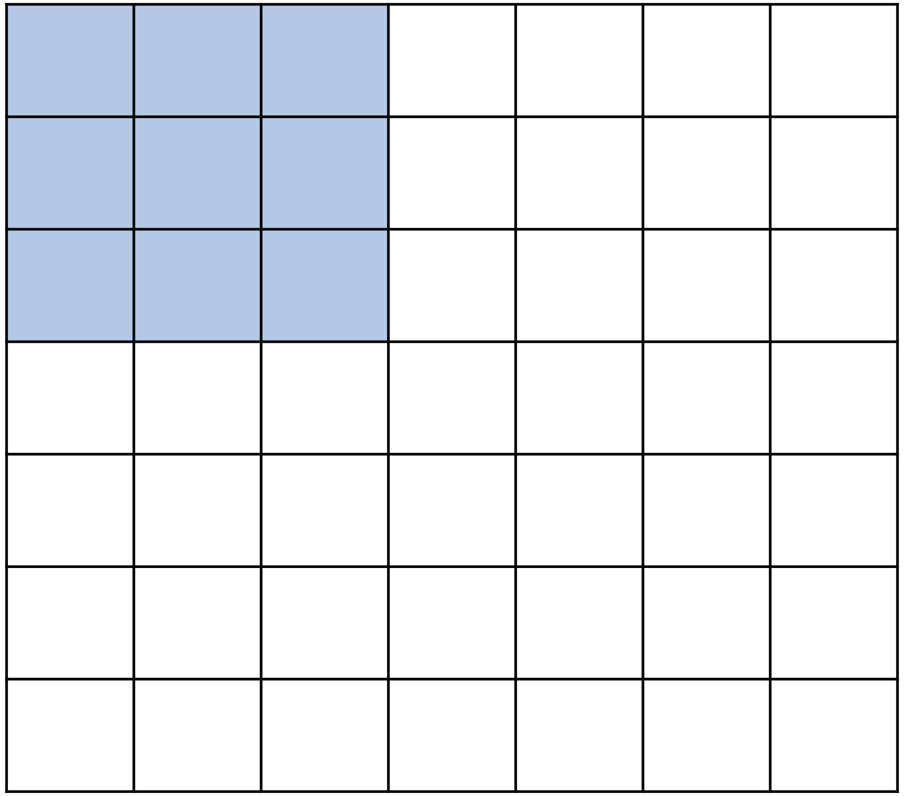
- Pooling을 통해 sub sampling을 수행함으로써 receptive field를 늘려줌

3 Stride and Padding



Convolution with Stride 1

7x7 Image, 3x3 Filter

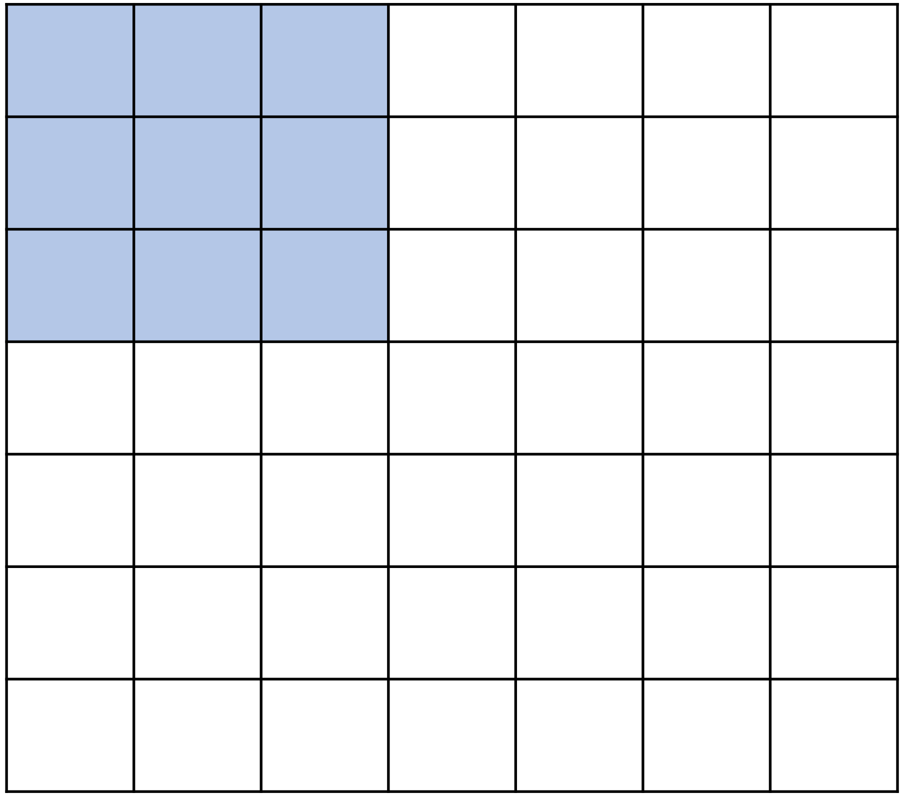


- 한 칸 씩 이동하면서 Convolution

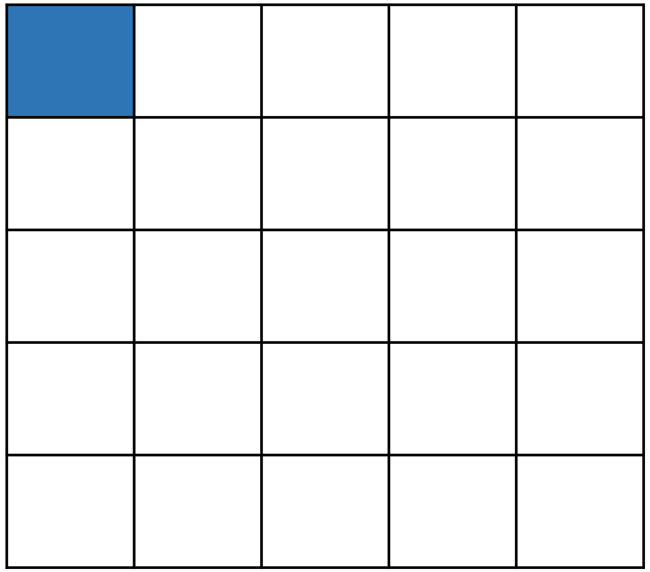
Output의 크기는?

Convolution with Stride 1

7x7 Image, 3x3 Filter

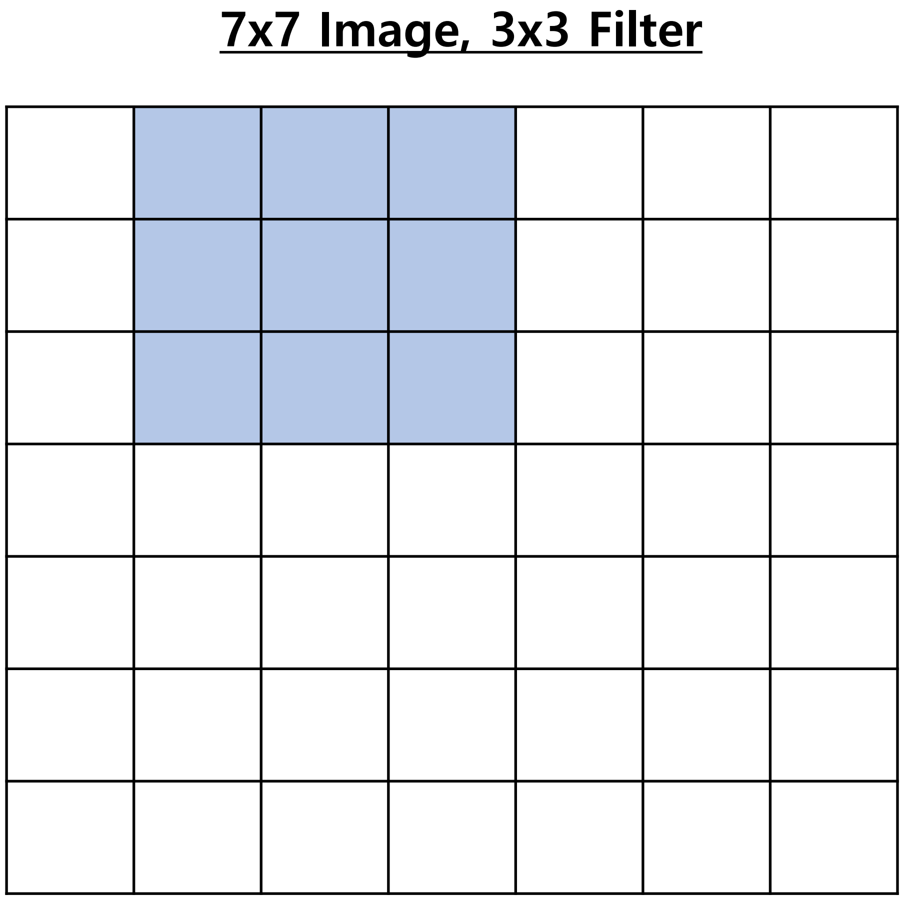


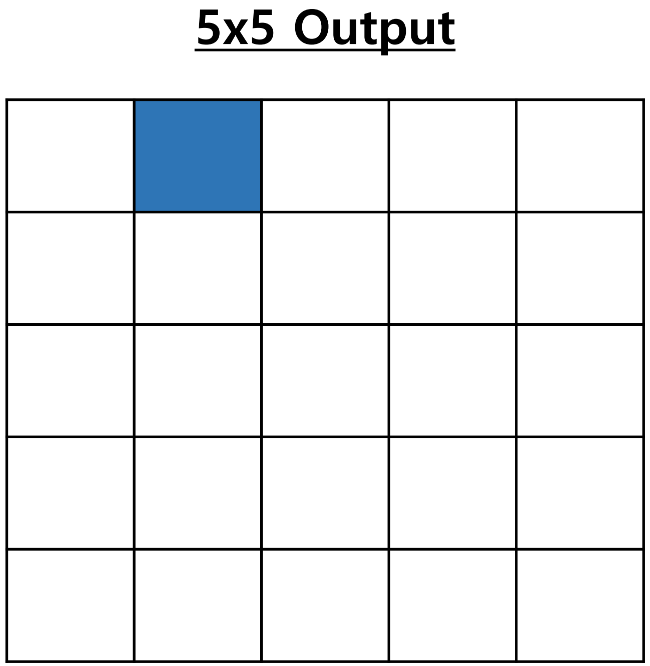
5x5 Output



- 한 칸 씩 이동하면서 Convolution

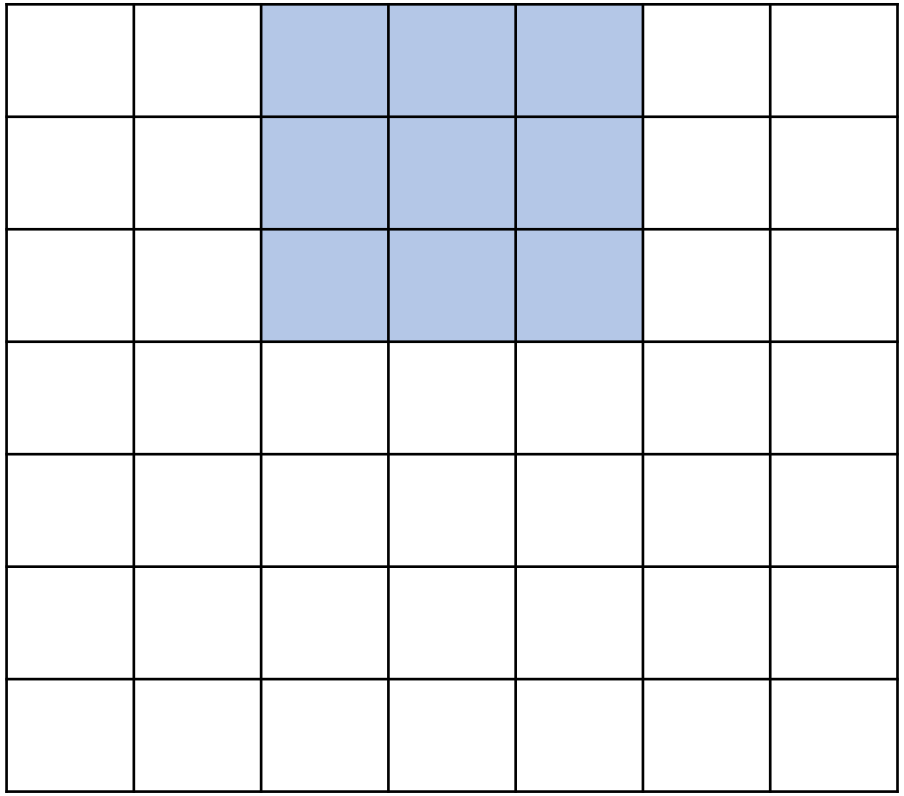
Convolution with Stride 1



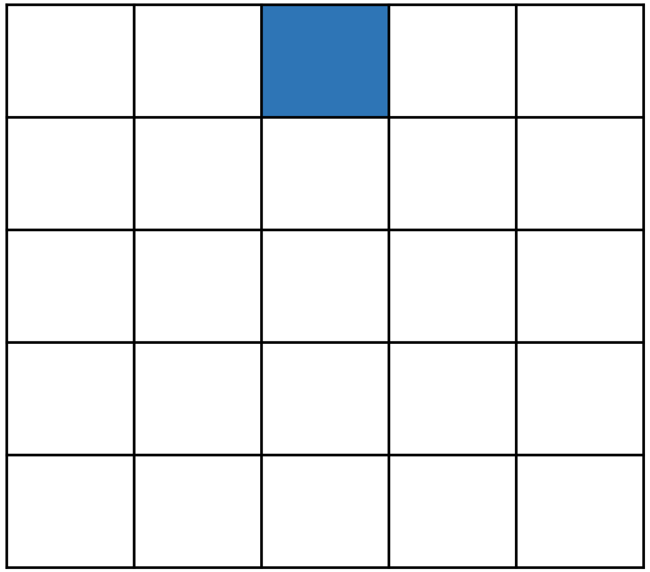


Convolution with Stride 1

7x7 Image, 3x3 Filter

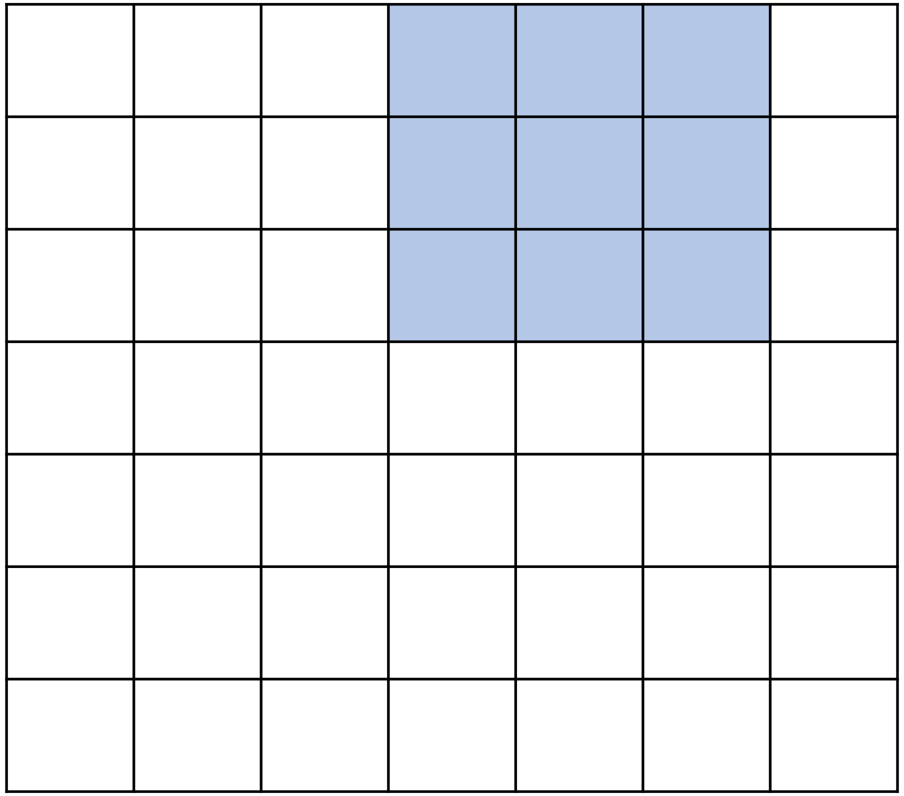


5x5 Output

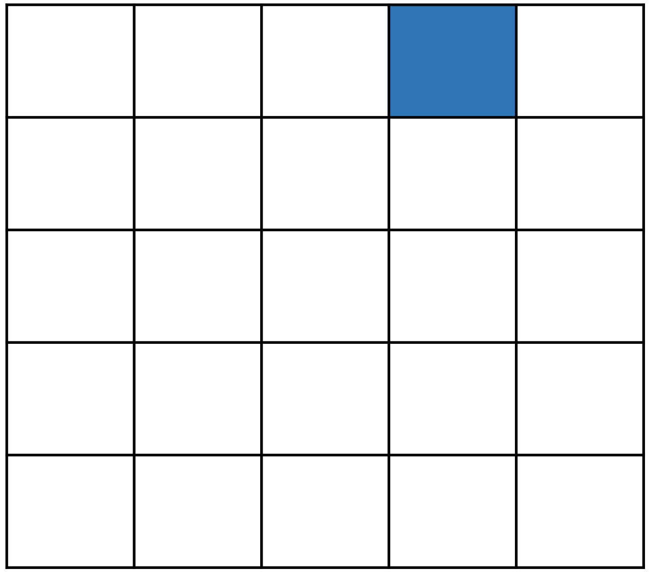


Convolution with Stride 1

7x7 Image, 3x3 Filter

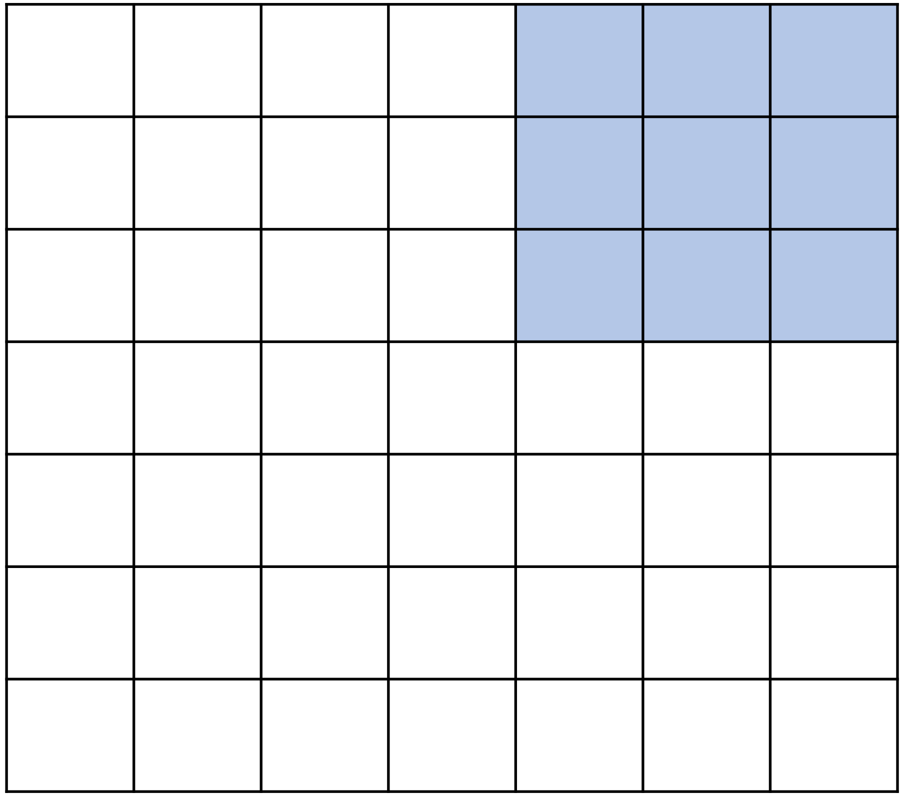


5x5 Output

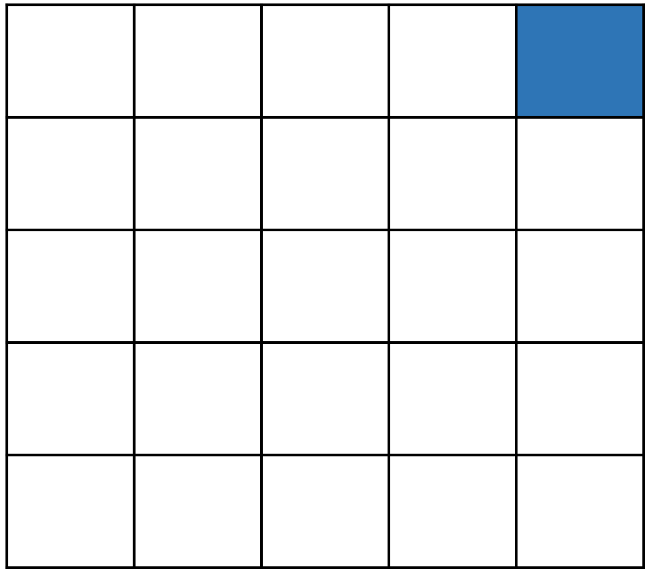


Convolution with Stride 1

7x7 Image, 3x3 Filter

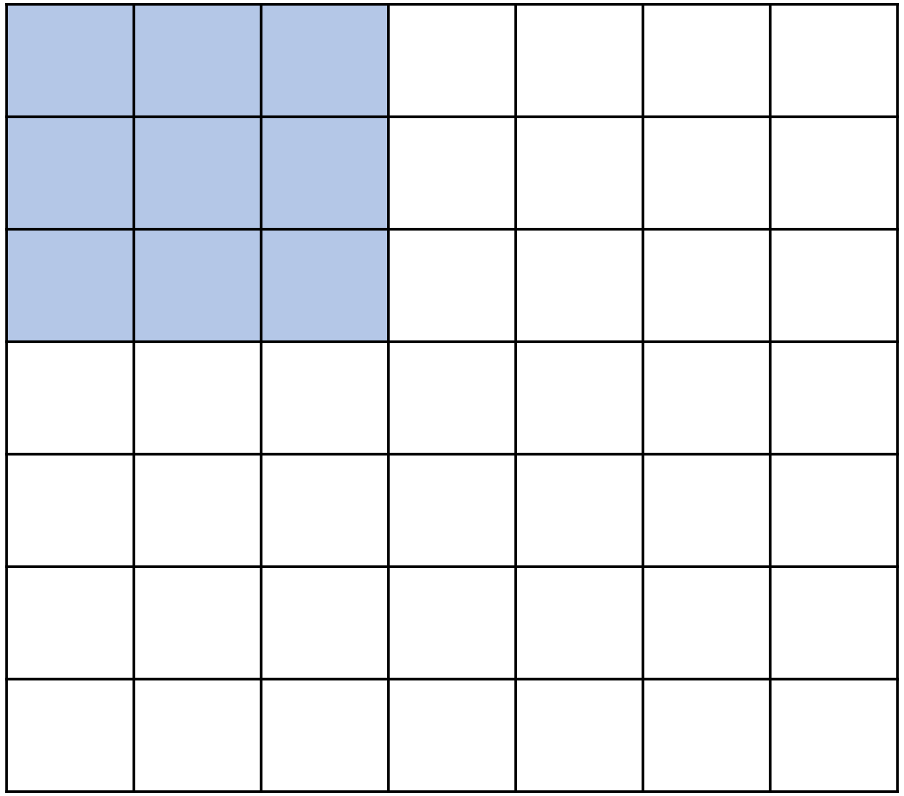


5x5 Output



Convolution with Stride 2

7x7 Image, 3x3 Filter

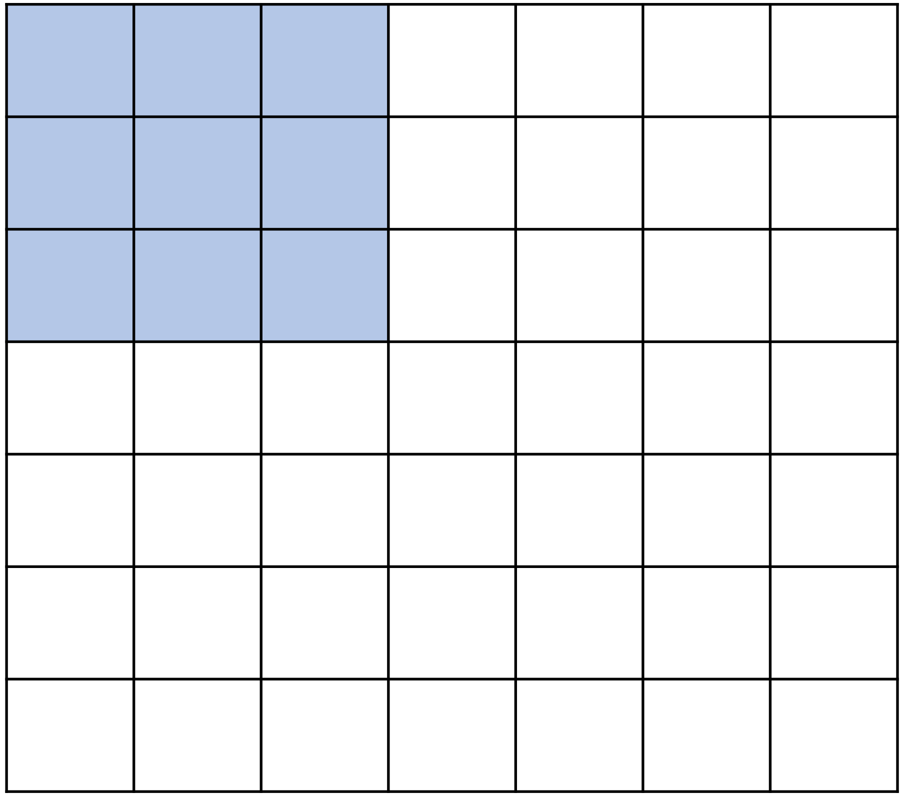


- 두 칸 씩 이동하면서 Convolution

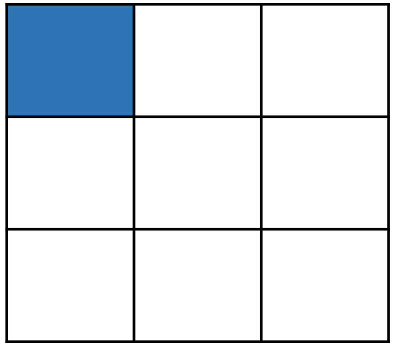
Output의 크기는?

Convolution with Stride 2

<u>7x7 Image, 3x3 Filter</u>



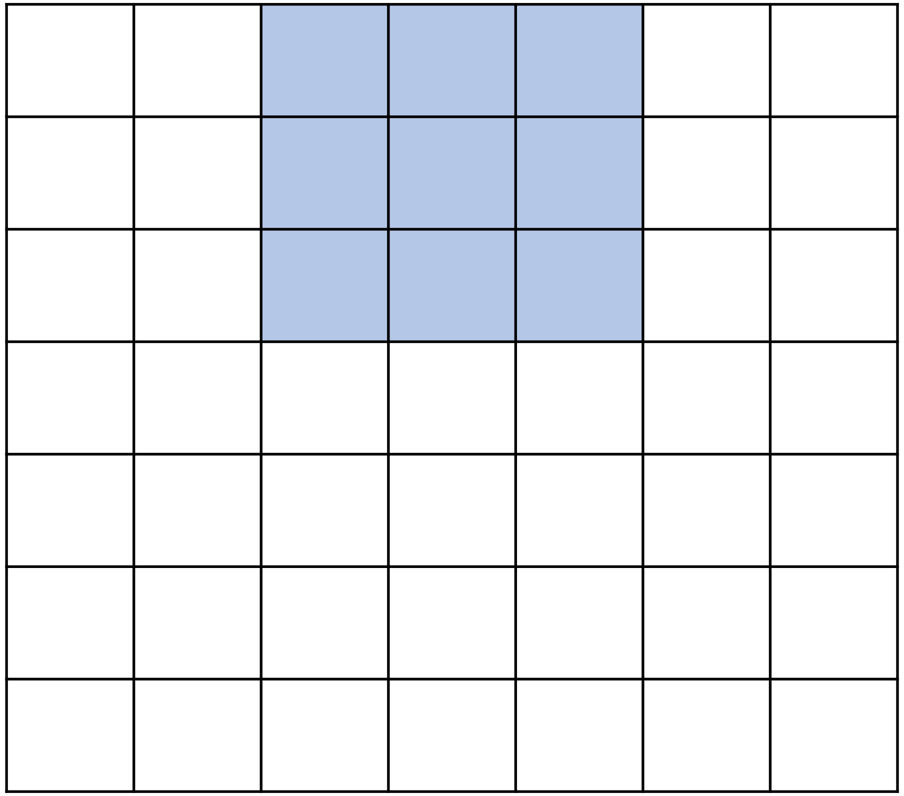
<u>3x3 Output</u>



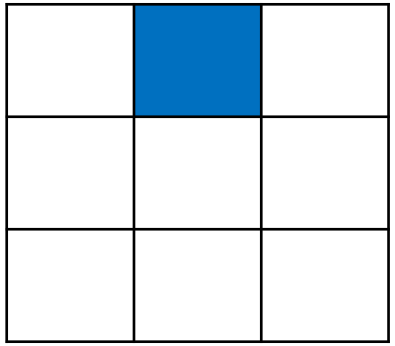
- 두 칸 씩 이동하면서 Convolution

Convolution with Stride 2

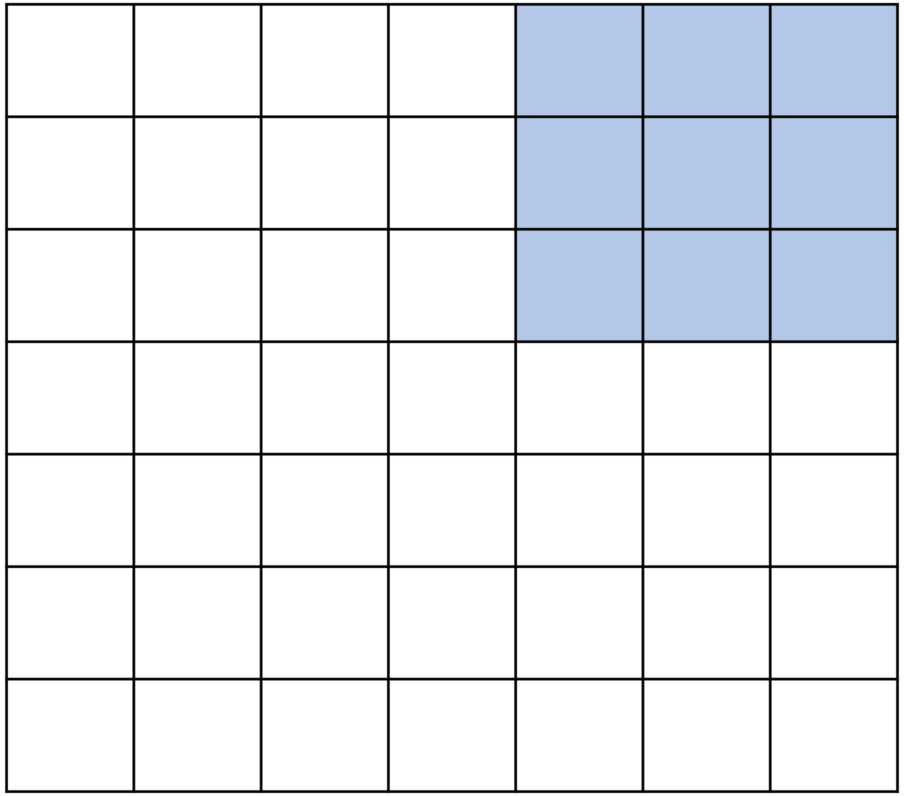
7x7 Image, 3x3 Filter

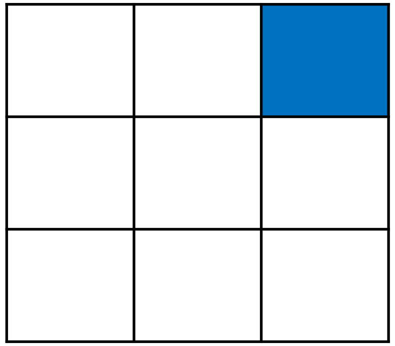


3x3 Output



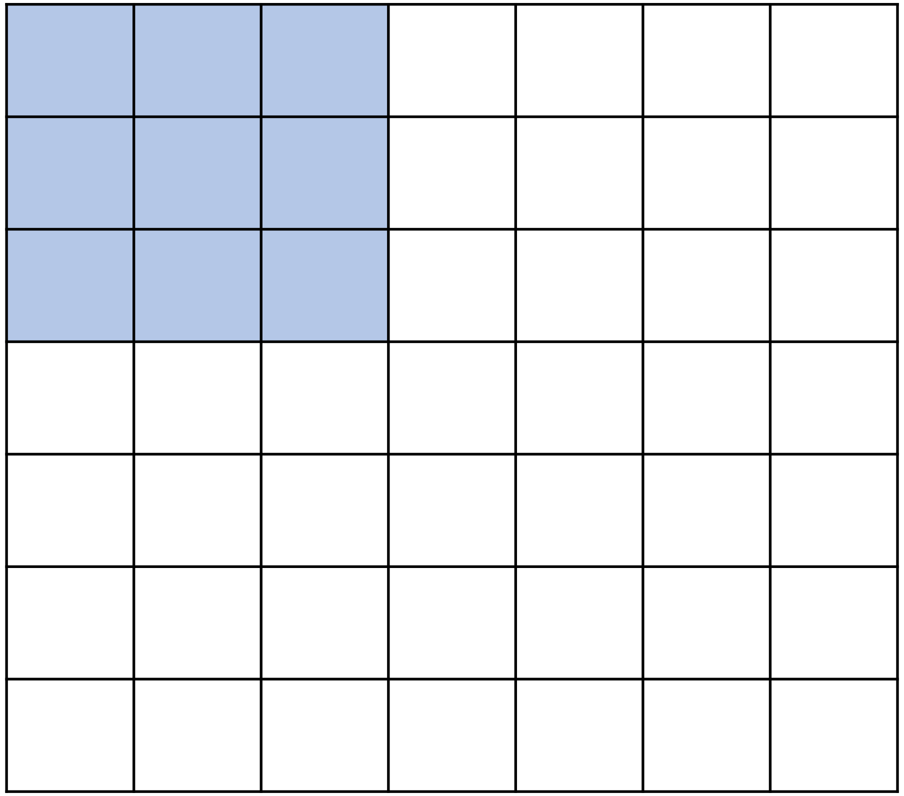
Convolution with Stride 2





Convolution with Stride 3

7x7 Image, 3x3 Filter

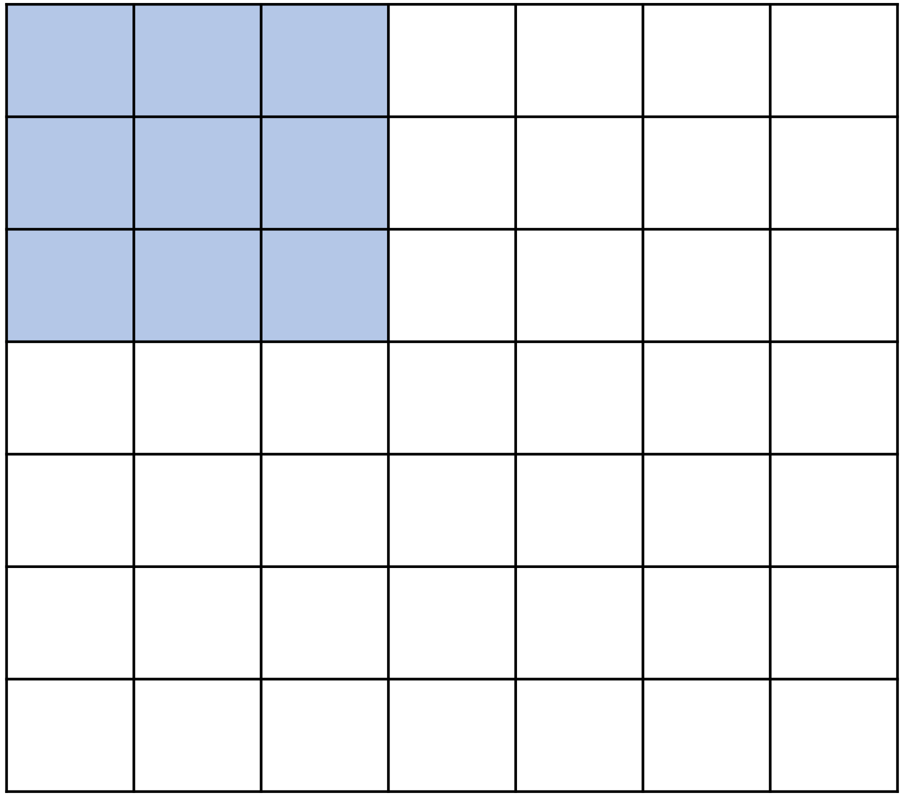


- 세 칸 씩 이동하면서 Convolution

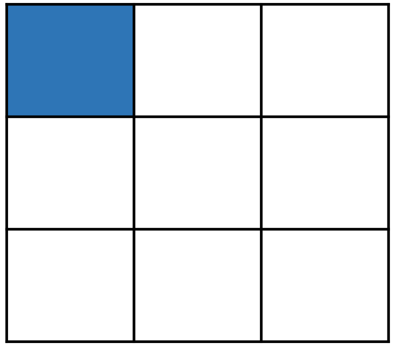
Output의 크기는?

Convolution with Stride 3

7x7 Image, 3x3 Filter



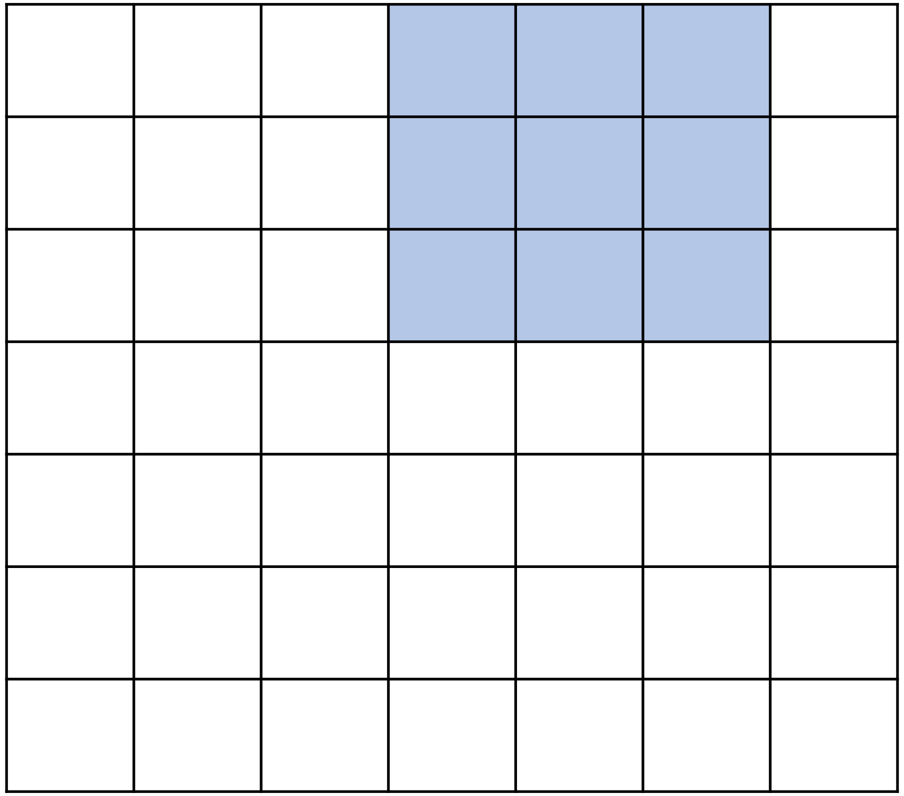
3x3 Output



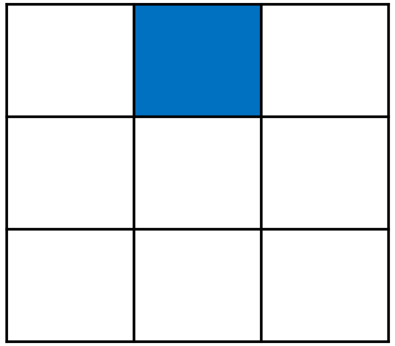
- 세 칸 씩 이동하면서 Convolution

Convolution with Stride 3

7x7 Image, 3x3 Filter

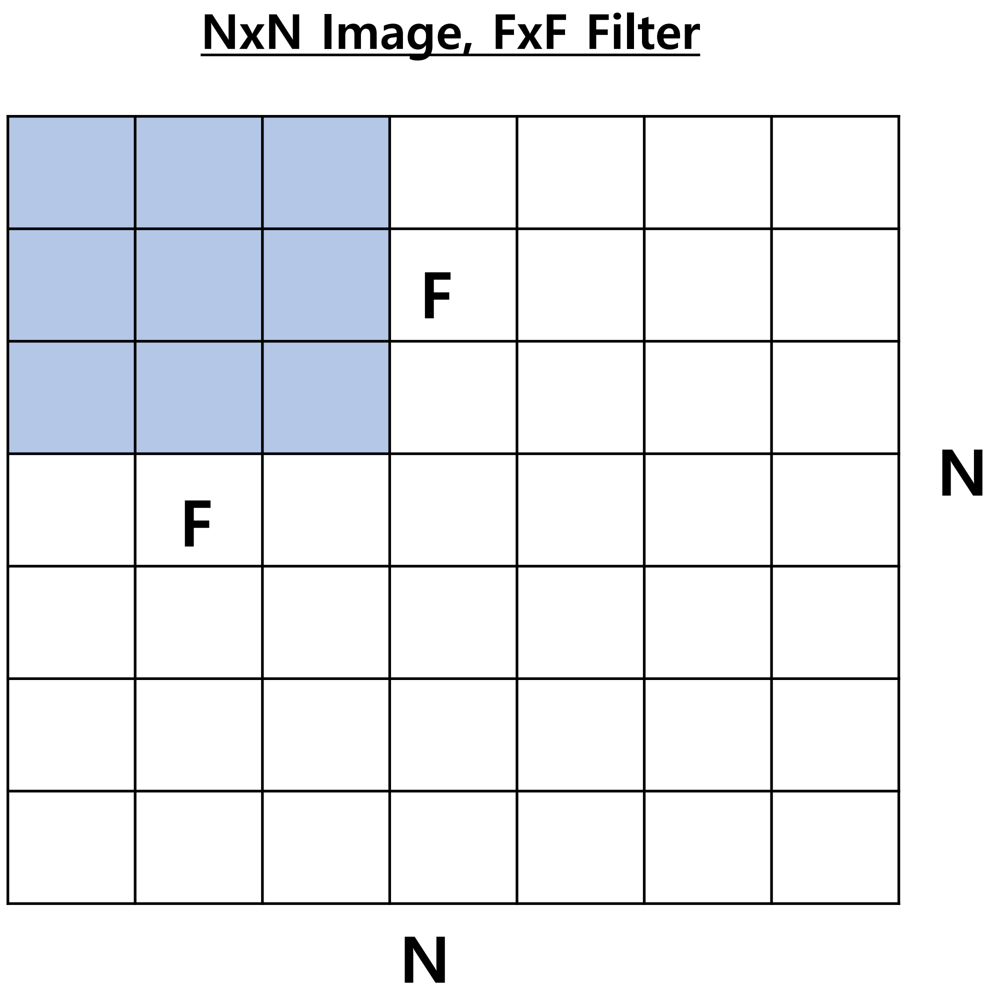


3x3 Output



- 더 이상 이동하지 못함!!
- Stride 3은 유효한 Stride가 아니다.

Output 크기 계산법



O = (N-F)/Stride + 1

예시 1) 7x7 Image, 3x3 Filter, Stride 1인 경우

5 = (7-3)/1 + 1

따라서, 5x5 Output 생성

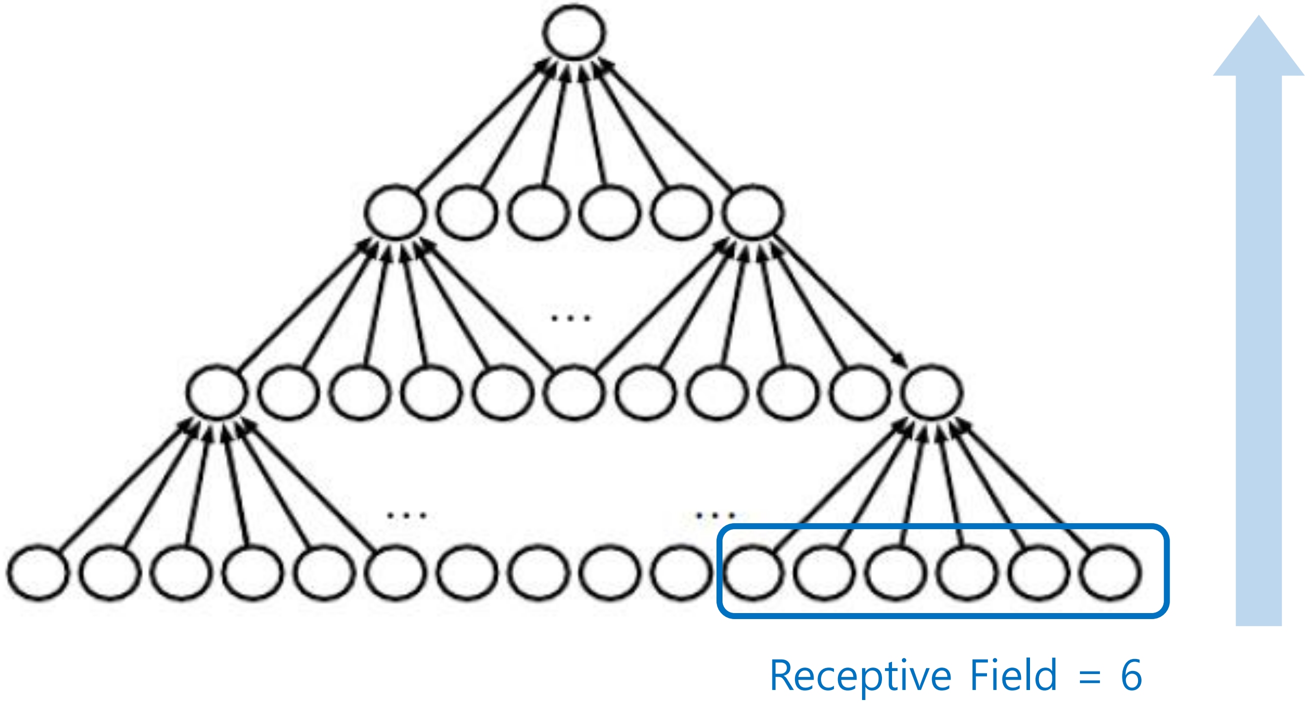
예시 2) 7x7 Image, 3x3 Filter, Stride 2인 경우

3 = (7-3)/2 + 1

따라서, 3x3 Output 생성

Convolution 연산 반복

Layer가 깊어질수록 크기가 점점 줄어드는 현상



Layer 마다 5 픽셀 씩 줄어든다.

- Kernel Size – 1 만큼씩 크기가 줄어들게 됨

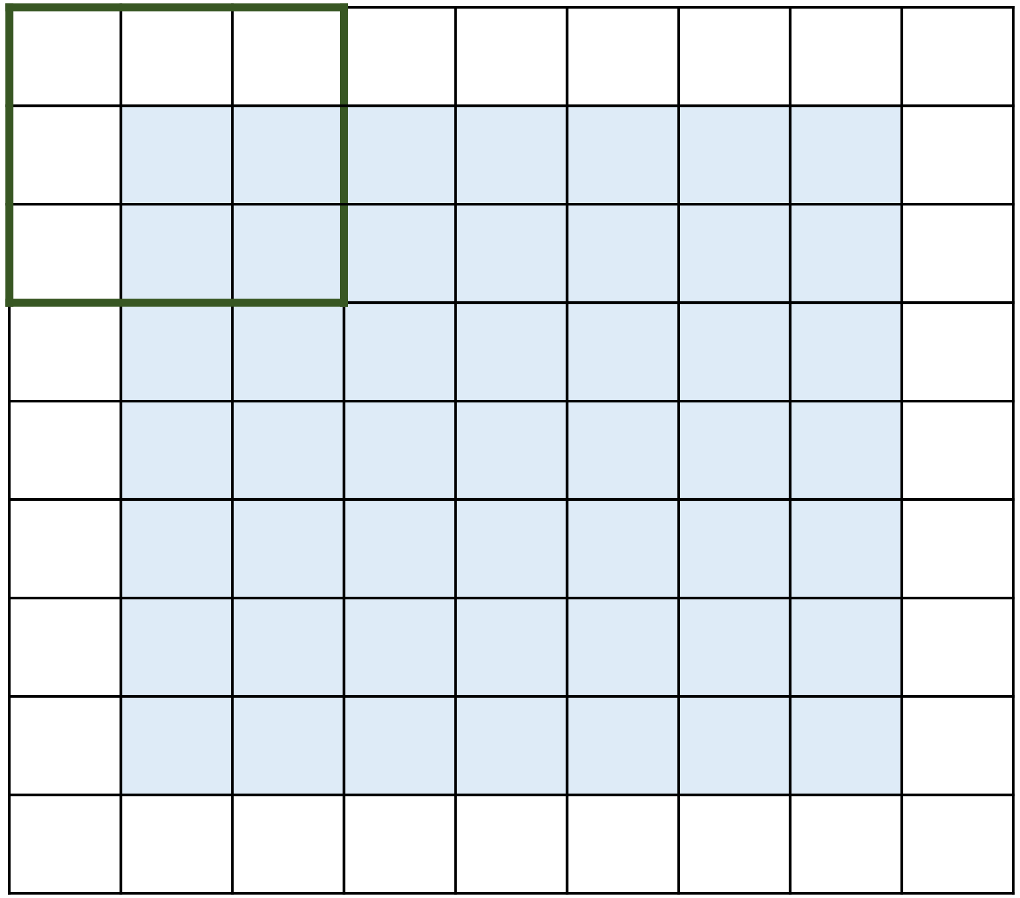
Convolution 연산 반복

크기가 유지되도록 Padding을 해 줌

- Kernel Size – 1 만큼 Padding을 하면 크기가 유지 됨

Padding - Same Size

<u>**상하좌우로 1 픽셀 씩 Padding**</u>



7x7 Image, Padding 1, 3x3 Filter, Stride 1인 경우 Output의 크기는 얼마일까?

$$O = (N-F)/Stride + 1$$

$$7 = (9-3)/1 + 1$$

크기가 유지될 수 있음

Padding - Same Size

$$O = ((N+2*P)-F)/Stride + 1$$

P : Padding

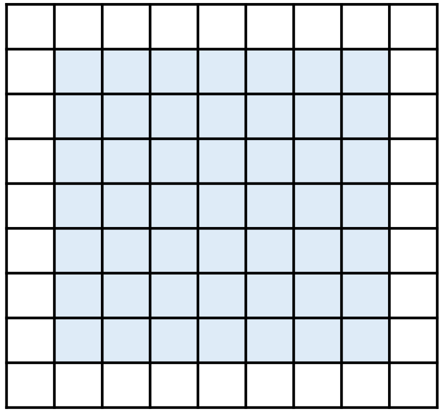
$$P = \frac{((O - 1) * Stride - (N - F))}{2}$$

Padding - Same Size

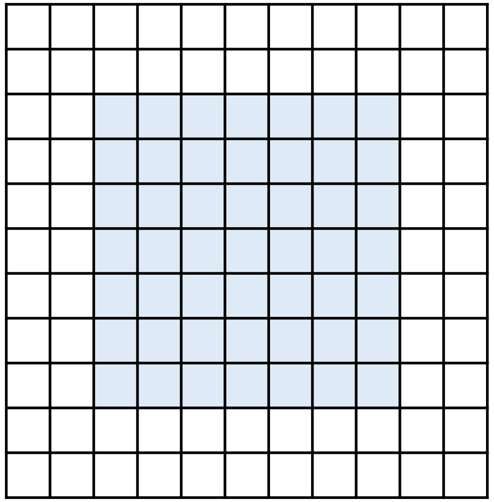
[7x7 Image, Stride 1] Same Size를 위한 Padding의 크기는?

$$\mathbf{P} = \frac{((O - 1) * Stride - (N - F))}{2}$$

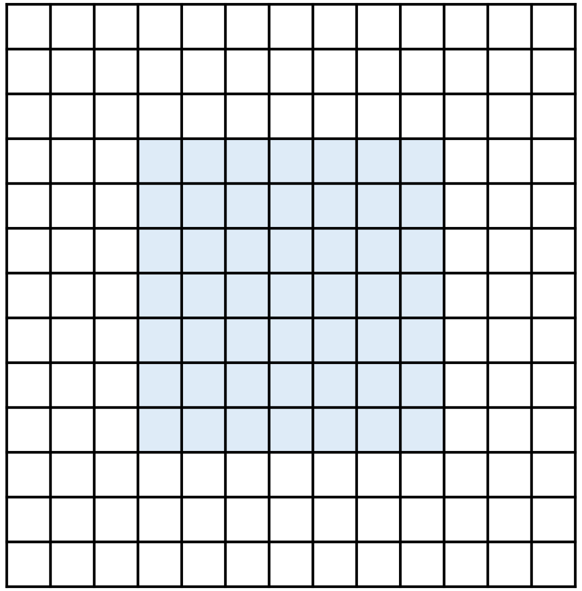
3x3 Filter



5x5 Filter



7x7 Filter

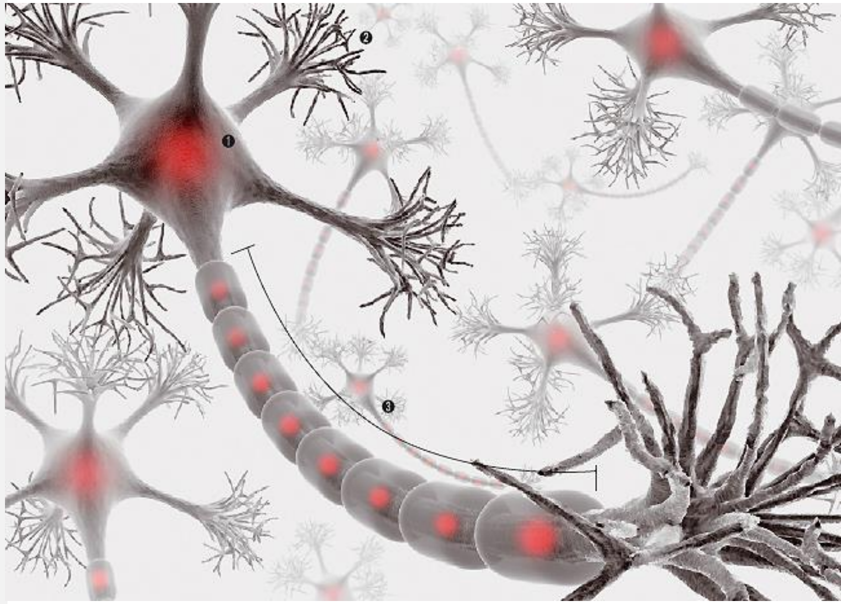


$\mathbf{P} = ((7 - 1) * 1 - (7 - 3))/2$
$= (6 - 4)/2$
$= 1$

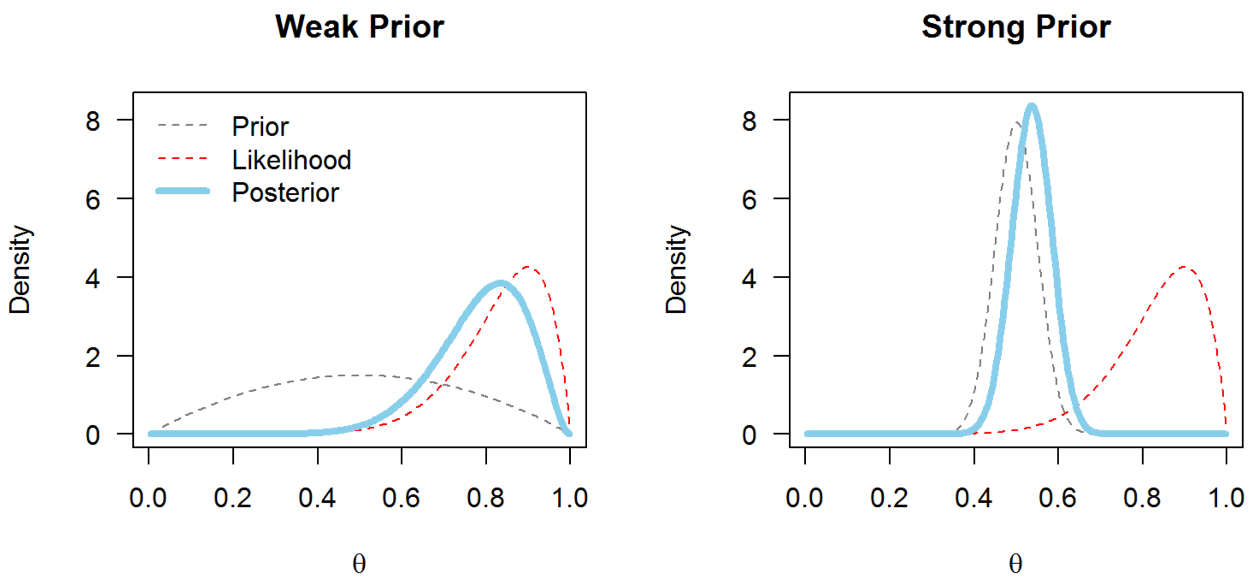
$\mathbf{P} = ((7 - 1) * 1 - (7 - 5))/2$
$= (6 - 2)/2$
$= 2$

$\mathbf{P} = ((7 - 1) * 1 - (7 - 7))/2$
$= (6 - 0)/2$
$= 3$

4 CNN 가정사항



모델의 가정사항



모델에 대한 믿음(Belief)을 파라미터의 사전 분포(Prior)로 표현

Weak Prior

- 높은 엔트로피를 갖는 분포
- Gaussian with high variance
- 데이터에 의해 파라미터가 자유롭게 변화함

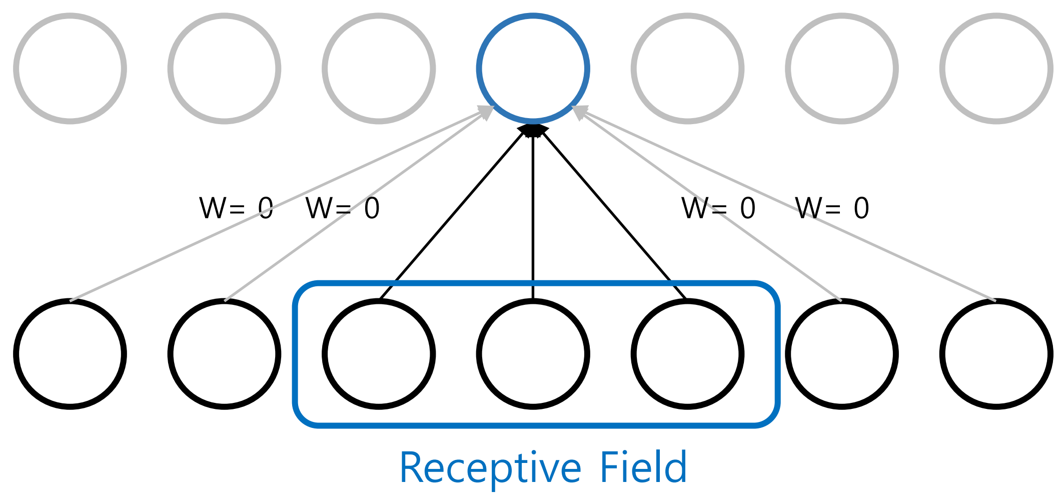
Strong Prior

- 낮은 엔트로피를 갖는 분포
- Gaussian with low variance
- 파라미터를 결정할 때 사전 분포의 영향력이 매우 큼

Infinitely Strong Prior

- 일부 파라미터의 확률이 zero
- 데이터에 상관없이 확률이 zero인 파라미터는 사용할 수 없음

Convolution as infinitely strong prior



- 계층 별로 가중치에 Infinitely Strong Prior를 적용
- Receptive Field 이외의 가중치는 0
- 모든 Hidden Unit에 대해 동일한 파라미터 사용

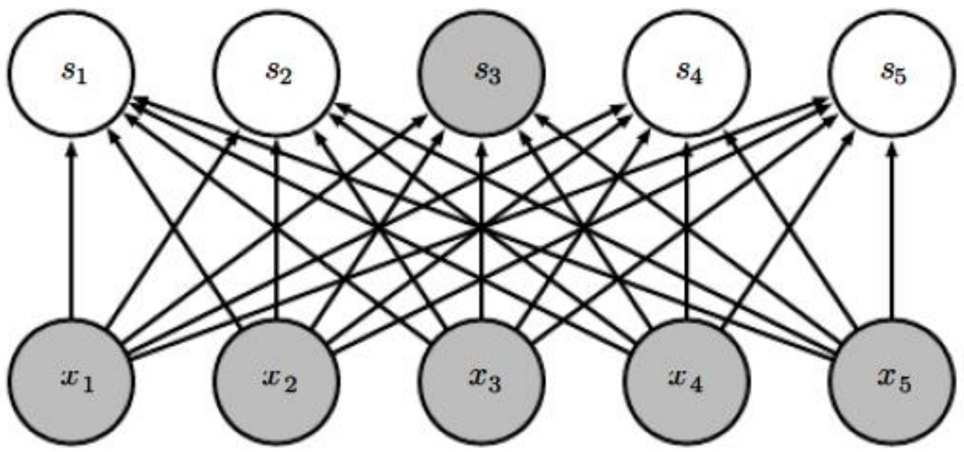
Convolution의 infinitely strong prior에 따라 다음 세가지 성질을 갖게 됨

- Sparse Interaction
- Parameter Sharing
- Equivariant

딥러닝의 성능을 향상시키는 중요한 아이디어!

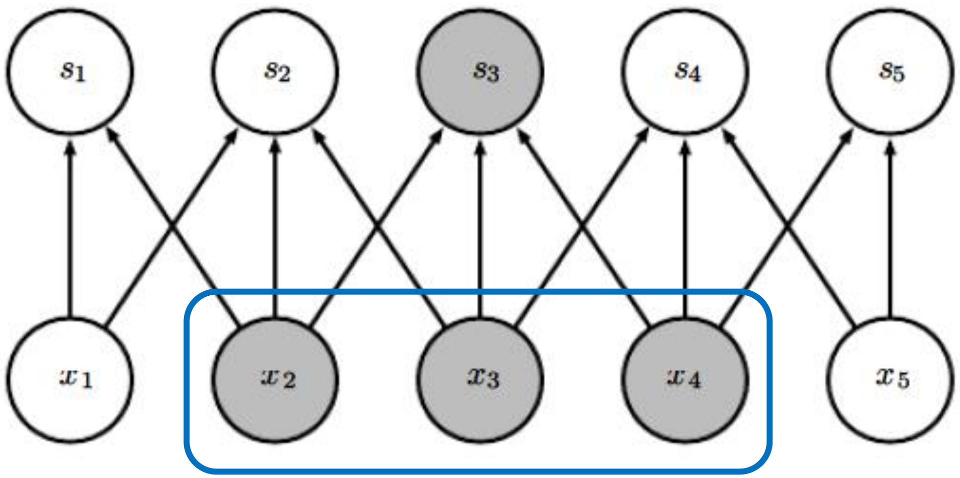
Convolution Sparse Interaction

Full Connectivity



- Output은 모든 input에 연결됨
- Parameter 수 : $O(m \times n)$
  - m : input 개수
  - n : output 개수

Sparse Connectivity



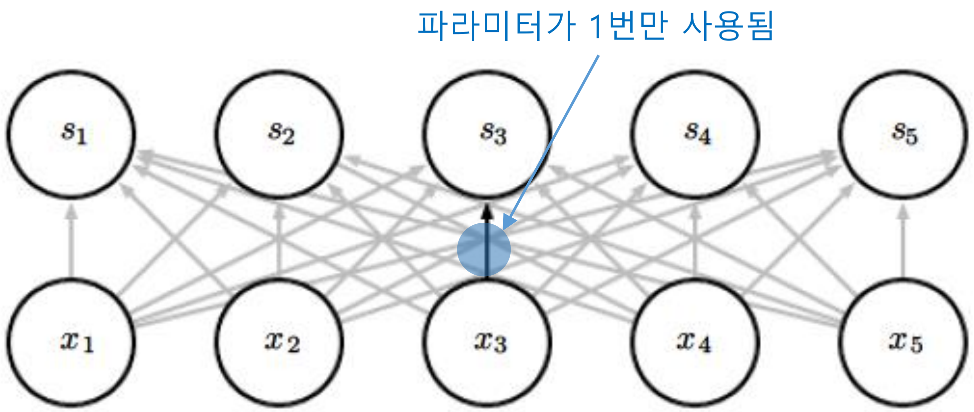
- Output에 연결된 input이 제한적
- Parameter 수 : $O(k \times n)$
  - k : output과 연결된 connection 수
  - n : output 개수

메모리 및 계산 절약, 통계적 효율 향상

Convolution

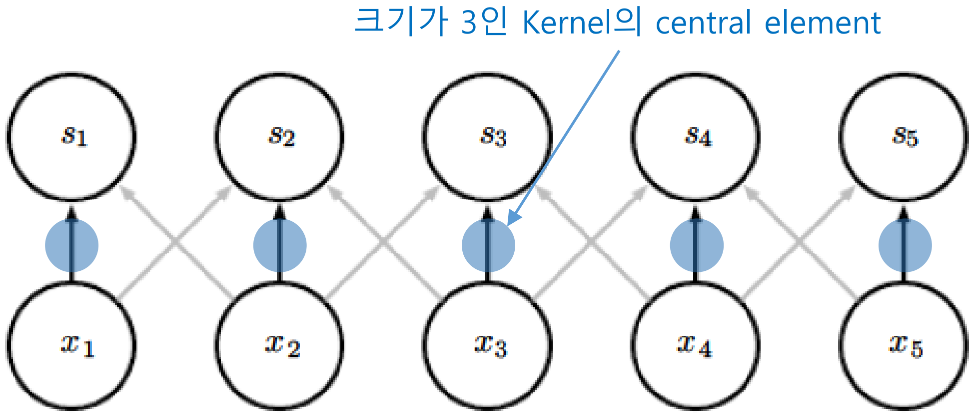
Parameter Sharing

No Parameter Sharing



- 각 파라미터는 한번만 사용됨
- Parameter 수 : $O(m \times n)$
  - m : input 개수
  - n : output 개수

Parameter Sharing

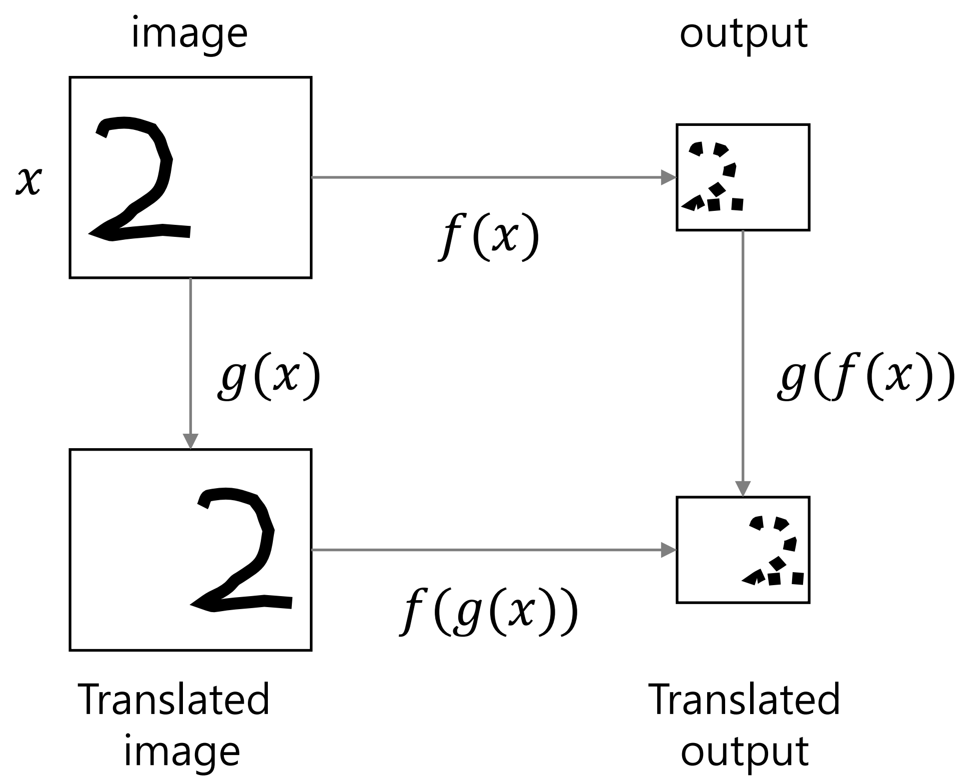


- 모든 파라미터가 재사용됨
- Parameter 수 : $O(k)$
  - k : 각 output이 갖는 connection 수
  - n : output 개수

메모리 및 계산 절약, 통계적 효율이 극적으로 향상

Equivariance

Equivariance to Translation



Input이 이동한 만큼 output도 이동하는 성질

$$f(g(x)) = g(f(x))$$

g : Translation

f : Convolution

- Parameter Sharing으로 나타나는 효과
- Convolution은 scale, rotation에 대해서는 equivariant하지 않음

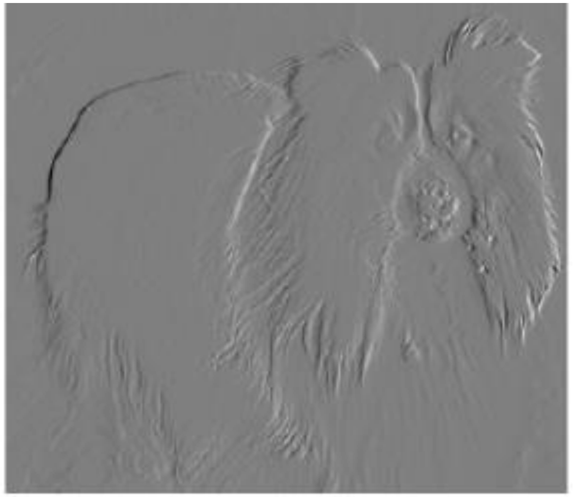
Convolution Equivariance

Input의 여러 위치에 동일한 패턴의 정보를 처리할 때 유용

<u>Edge Detection</u>

320

280



<u>Kernel</u>

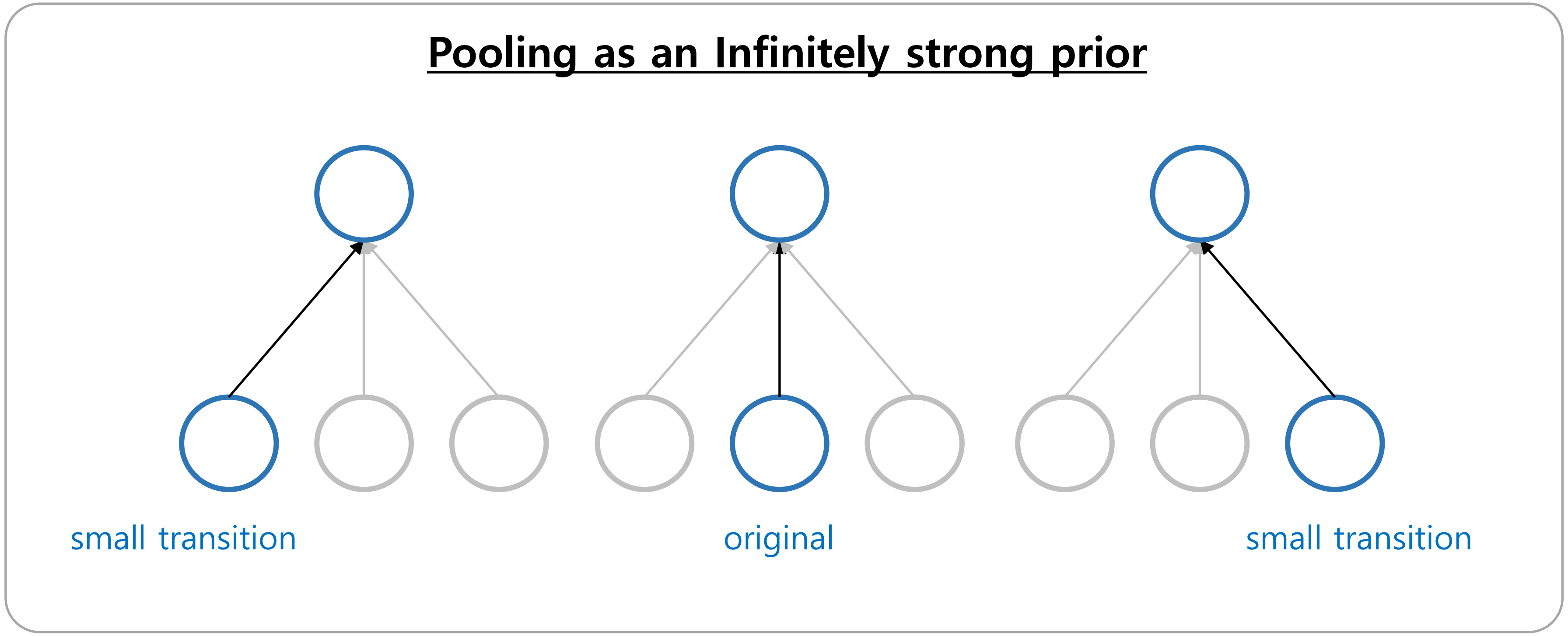
1	-1

Convolution : [319 x 280] x 3 = 267,960 float point operations

약 60,000 배 이상 계산 효율성 향상

행렬 연산 : [320 x 280] x [319 x 280] = 약 16,000,000 float point operations

Pooling as infinitely strong prior

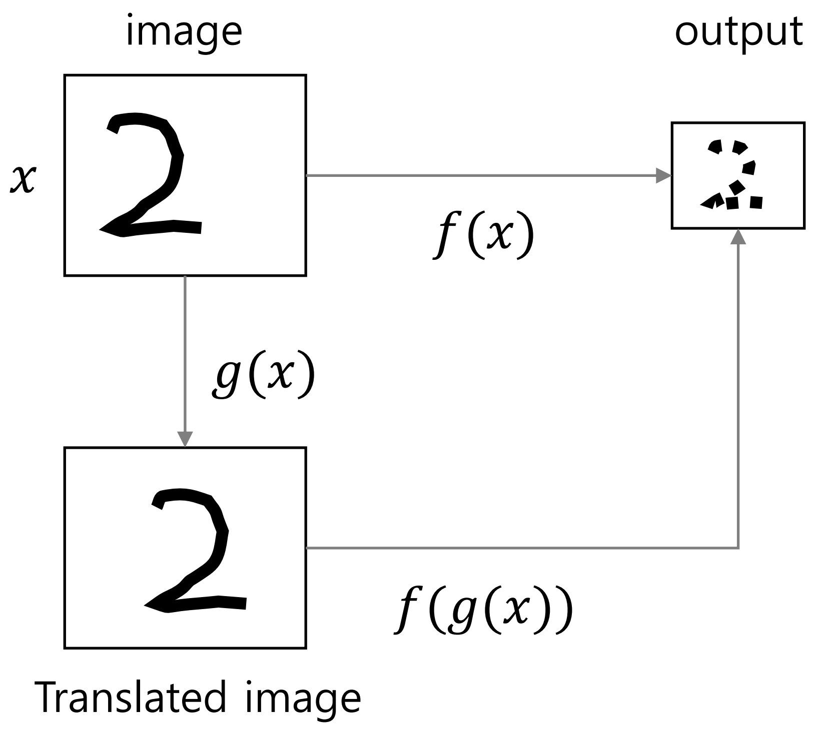


- Input에 조금의 변화가 있어도 Pooling은 동일한 결과를 얻을 수 있게 해 줌

Pooling의 Infinitely Strong Prior에 따라 Invariance to Local Translations을 갖게 됨

Invariance to Local Translation

Invariance to Local Translation



Input이 조금 이동해도 output은 바뀌지 않는 성질

$$f(x) = f\big(g(x)\big)$$

g : Translation

f : Pooling

"특징의 정확한 위치 보다 특징의 존재에 대해
더 관심이 있을 때 유용한 성질."

Thank you!