손에 잡히는 딥러닝

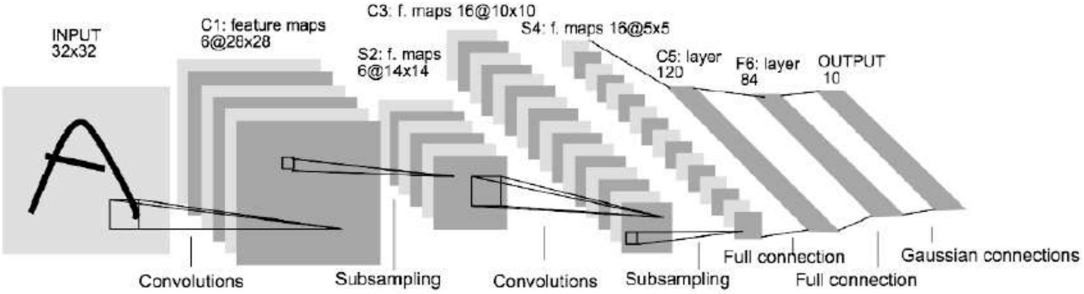
Convolutional Neural Networks

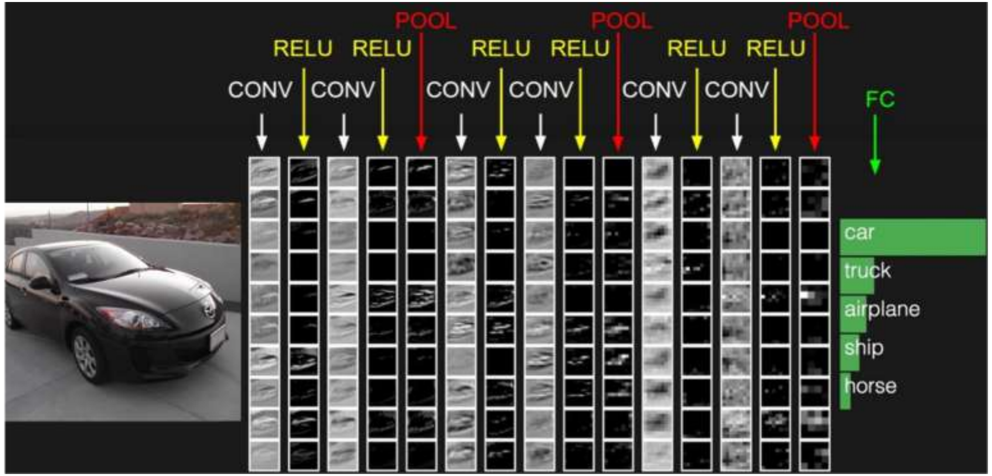
모두의연구소

박은수 Research Director

큰 그림 : 구조

- LeNet (1998년)

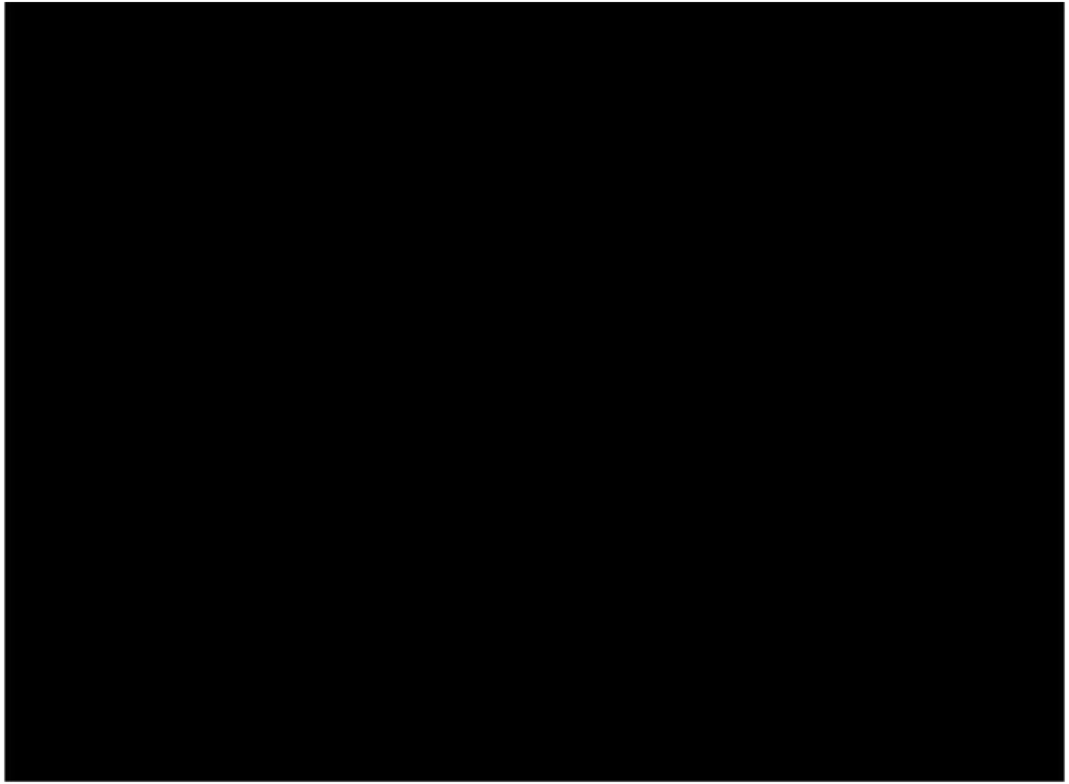




전형적인 구조

CONV POOL FC

LeNet 1

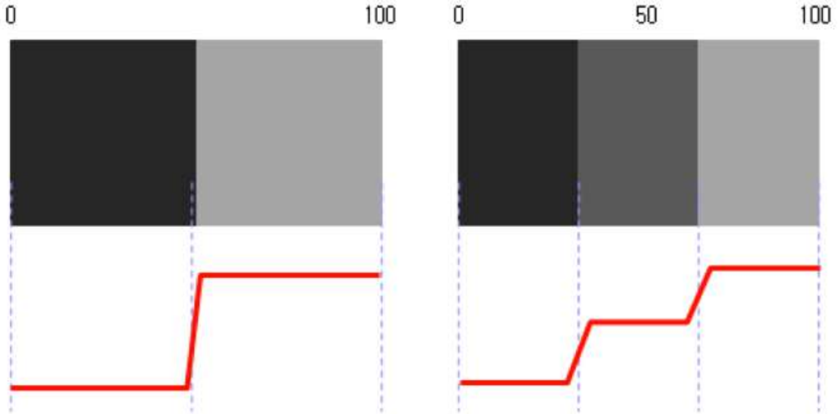


큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다

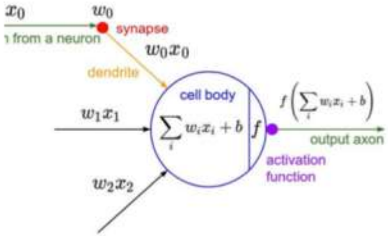
-1	0	1
-1	0	1
-1	0	1

수직 에지를 찾는
컨볼루션 필터

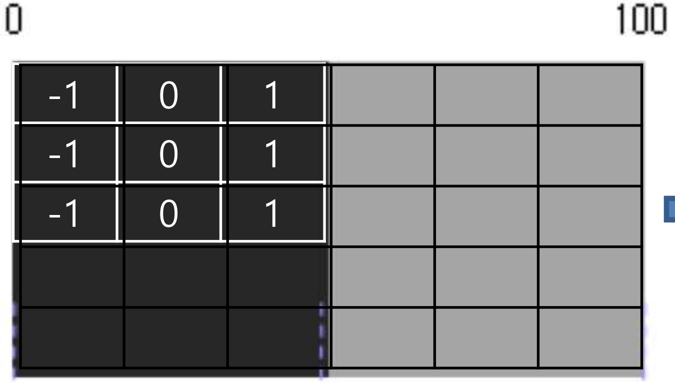


큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



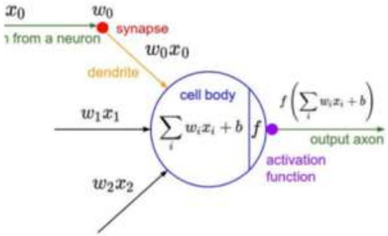
수직 에지를 찾는
컨볼루션 필터



0			

큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다

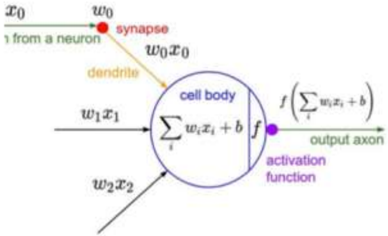


수직 에지를 찾는
컨볼루션 필터

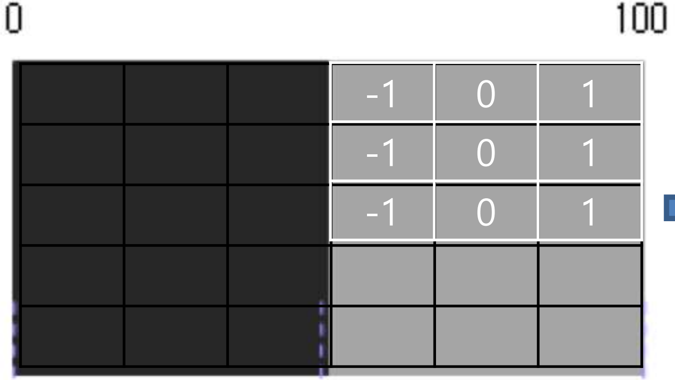
0	300		

큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



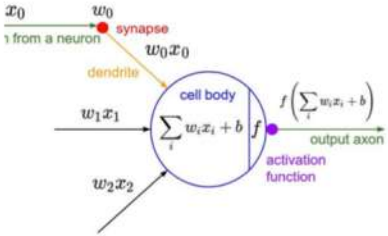
수직 에지를 찾는
컨볼루션 필터



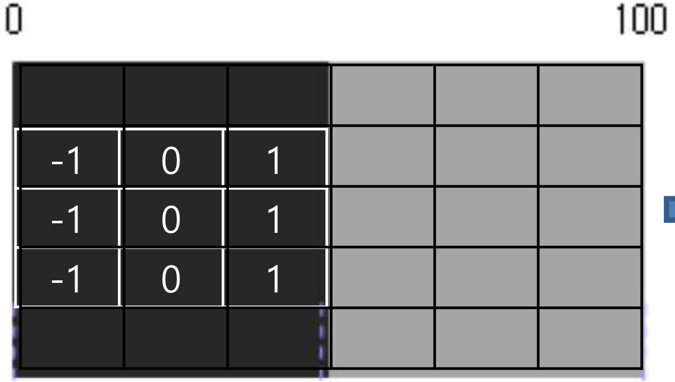
0	300	300	0

큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



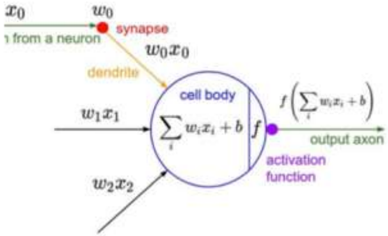
수직 에지를 찾는
컨볼루션 필터



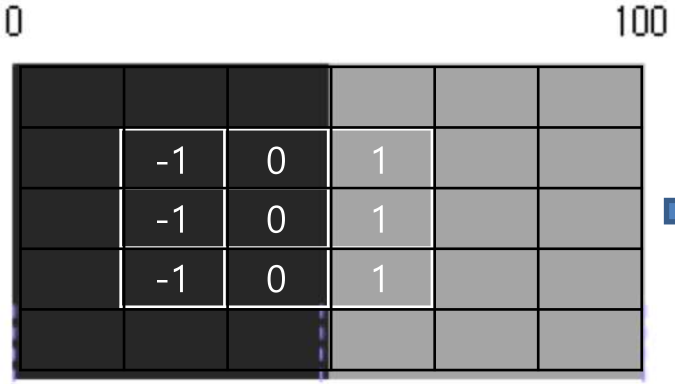
0	300	300	0
0			

큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



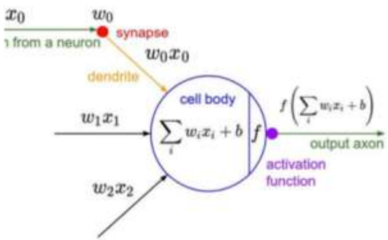
수직 에지를 찾는
컨볼루션 필터



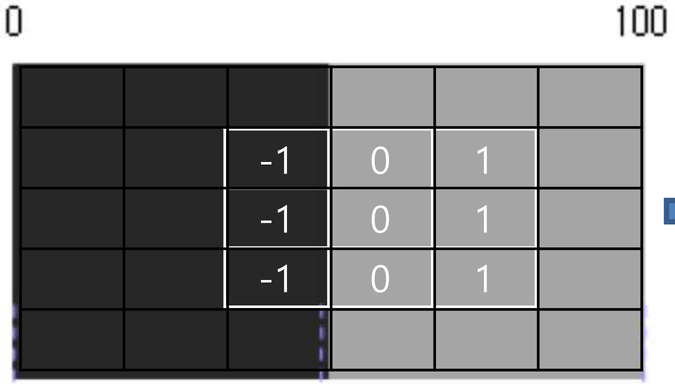
0	300	300	0
0	300		

큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



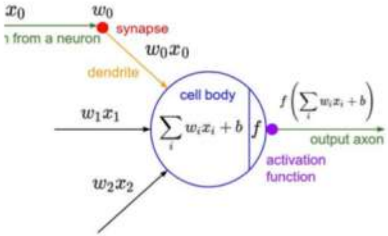
수직 에지를 찾는
컨볼루션 필터



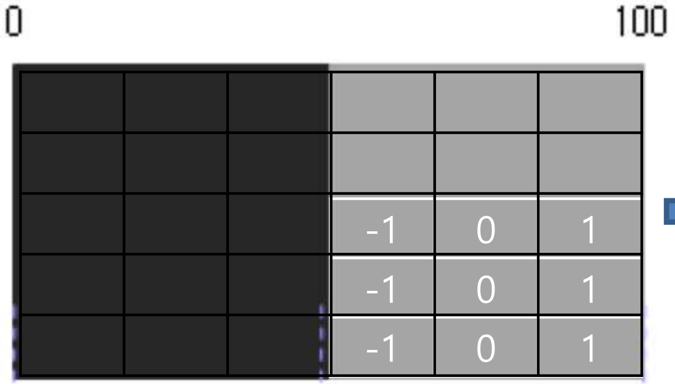
0	300	300	0
0	300	300	

큰 그림 : 컨볼루션은 ...

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

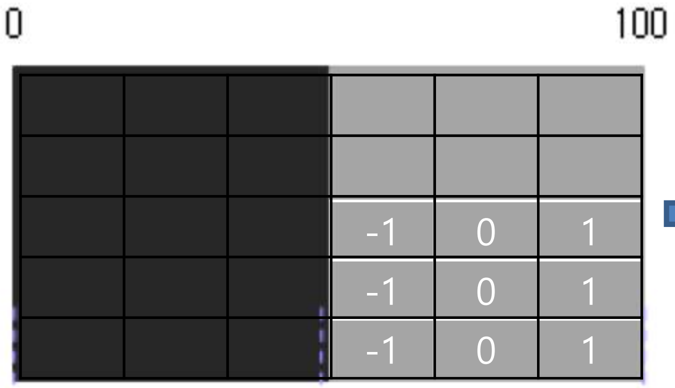


0	300	300	0
0	300	300	0
0	300	300	0

큰 그림 : 컨볼루션은 ...

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



0	300	300	0
0	300	300	0
0	300	300	0

수직 에지 검출

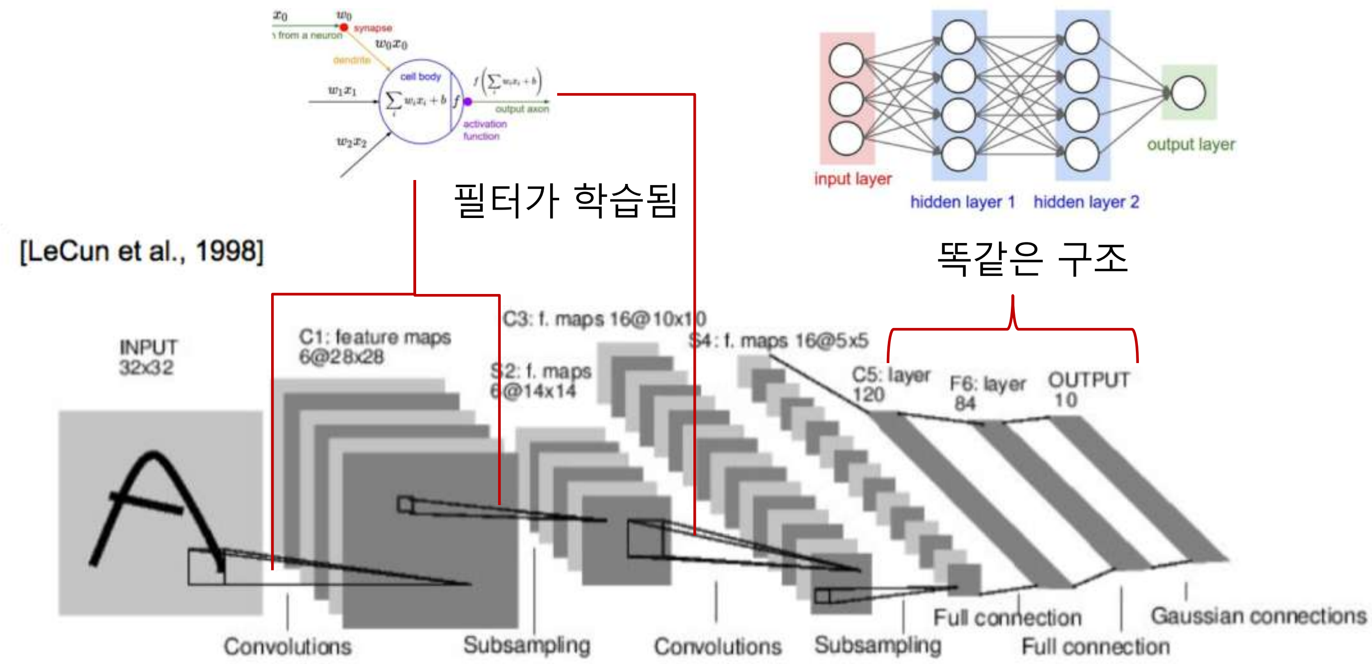
다양한 필터로 다양한 특징을 추출할 수 있다

- 위에 보여드린 예시의 구현 입니다

3_conv2d_example.ipynb

컨볼루셔널 뉴럴넷 (Convolutional Neural Network)

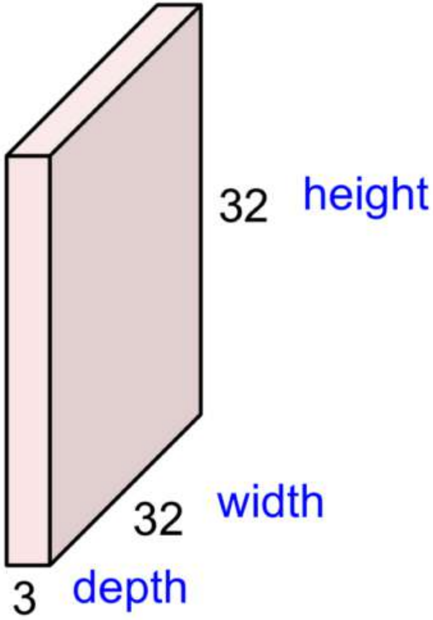
- 필터를 학습하는 구조다



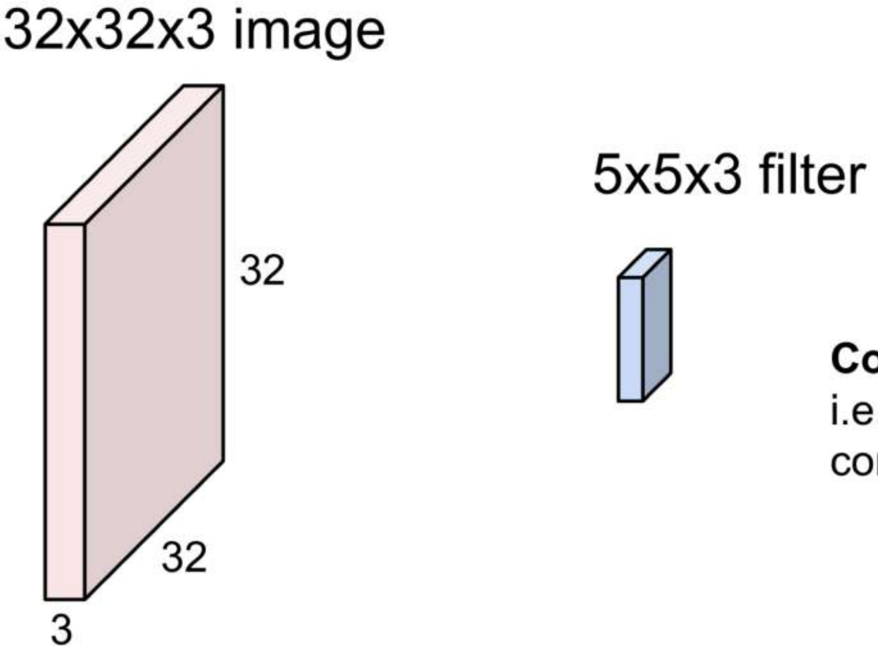
이제부터
자세히 살펴봅시다

Convolution Layer

32x32x3 image -> preserve spatial structure



Convolution Layer

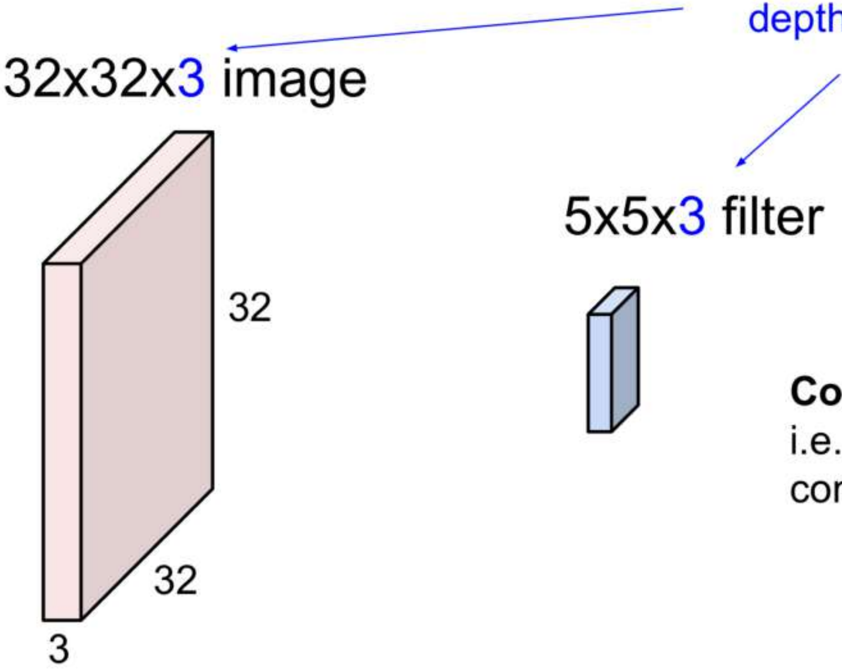


Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

Convolution Layer

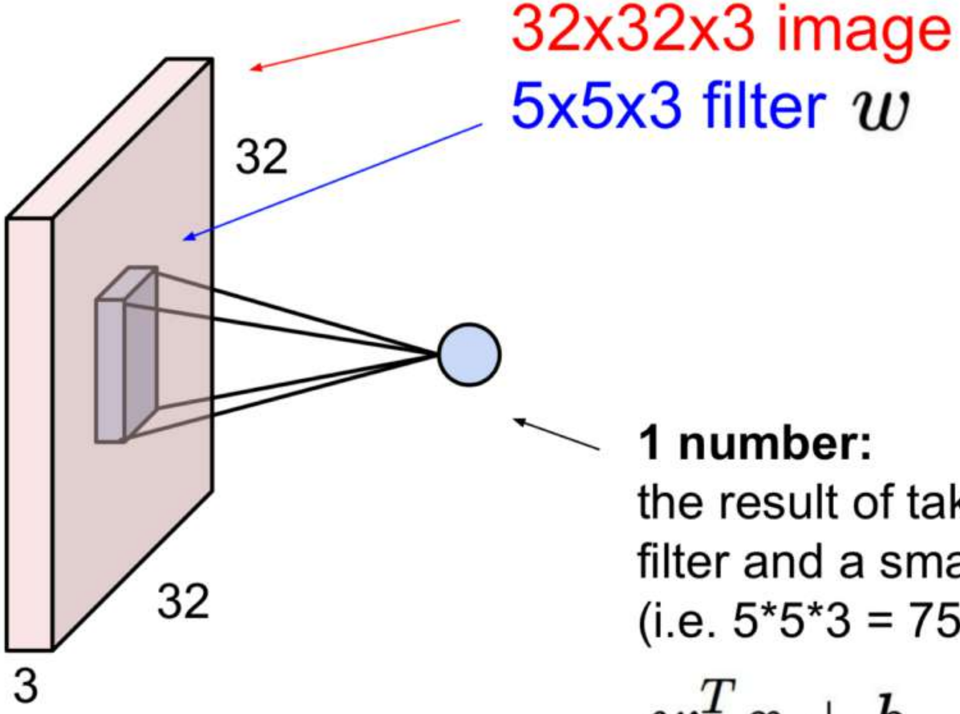
모두의연구소

Filters always extend the full depth of the input volume



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

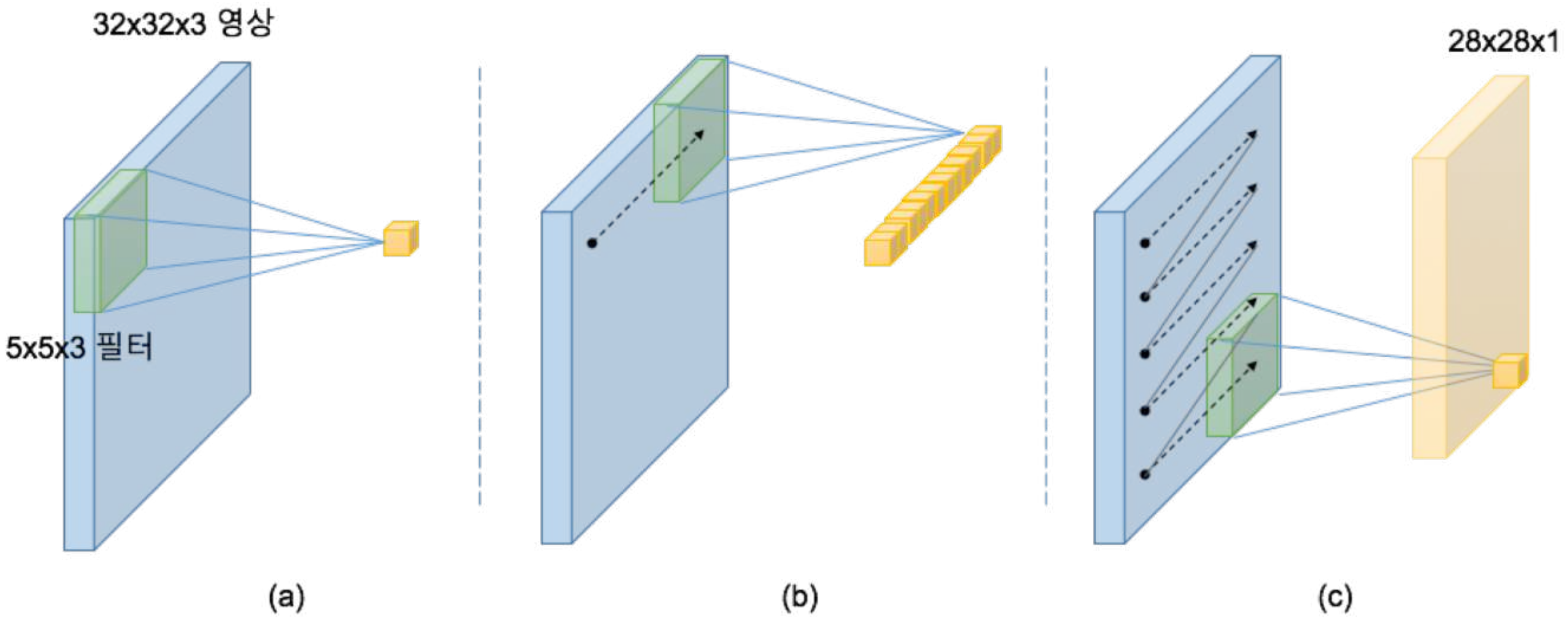
Convolution Layer



1 number:
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)

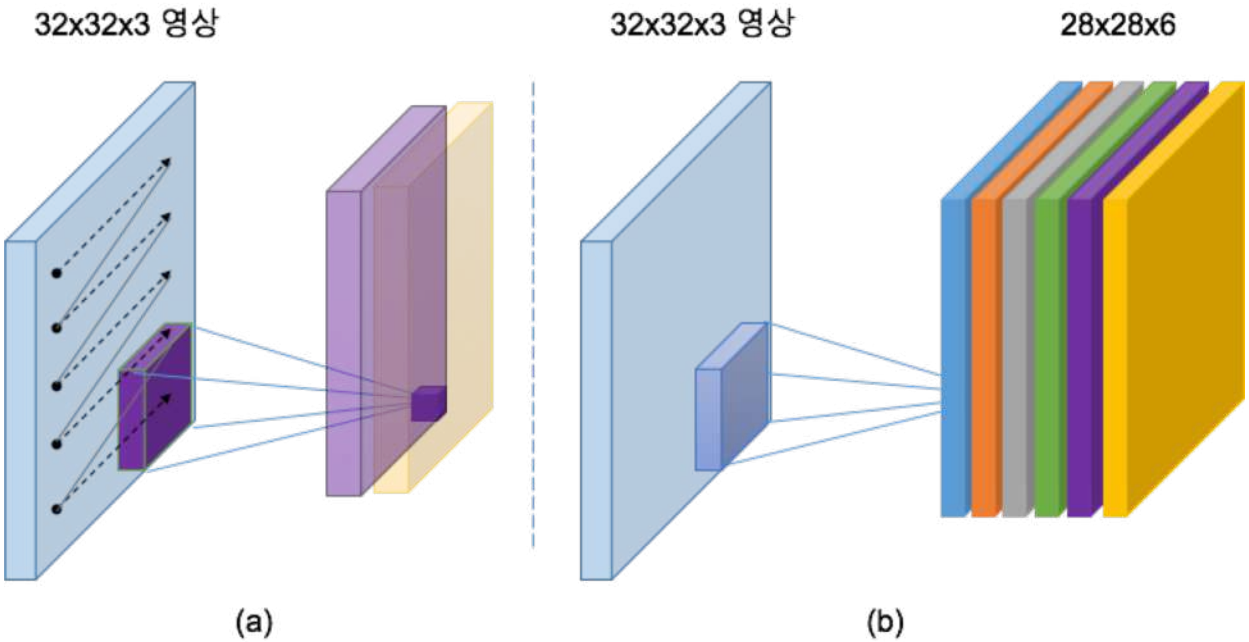
$$w^T x + b$$

Convolution Layer



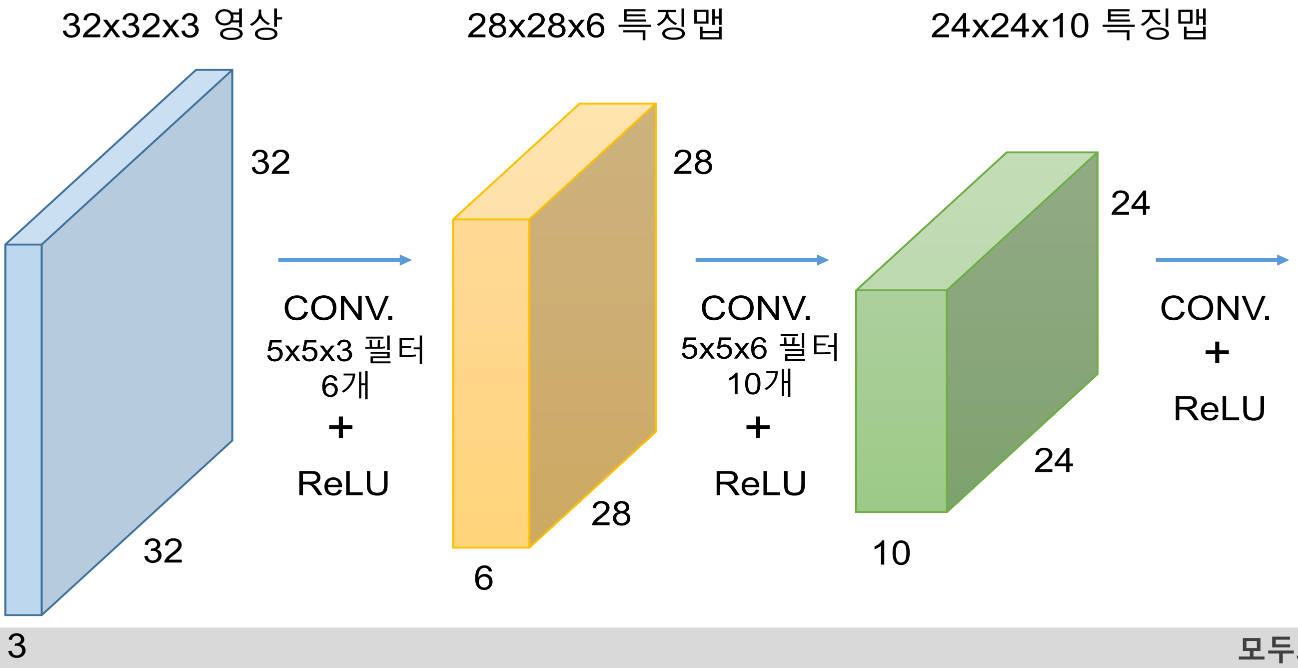
Convolution Layer

똑같은 크기의 필터 6개를 더 만들어 봅시다



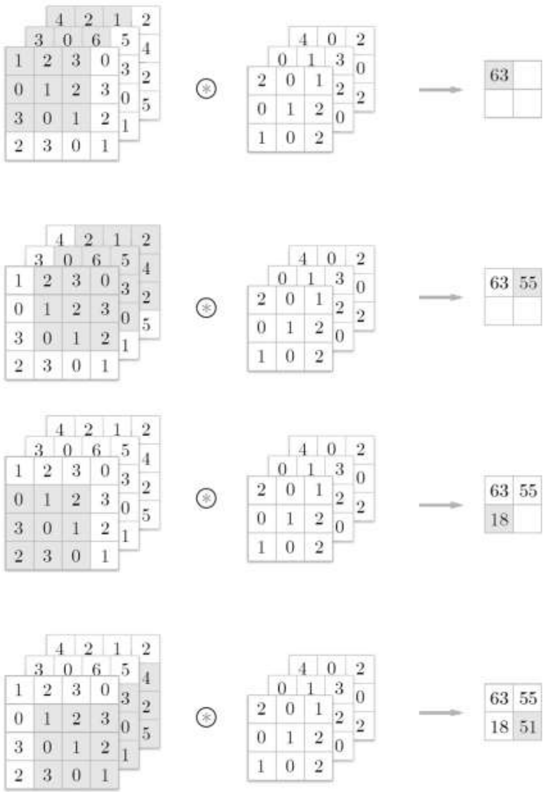
Convolution Layer

컨볼루션 네트워크는 활성화 함수를 포함한 컨볼루션 레이어의 연결 입니다



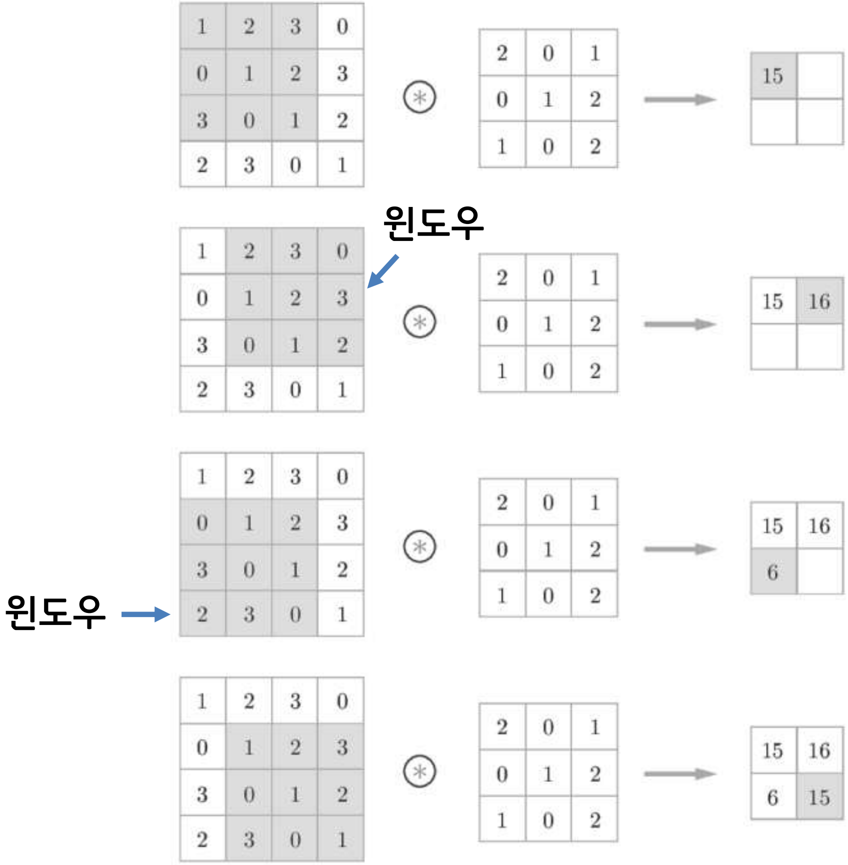
Convolution Layer

- 주의할 점은 **입력데이터의 채널 수와 필터의 채널 수가 같아야** 한다는 점
- 각 필터의 채널크기는 같아야 함



Convolution Layer

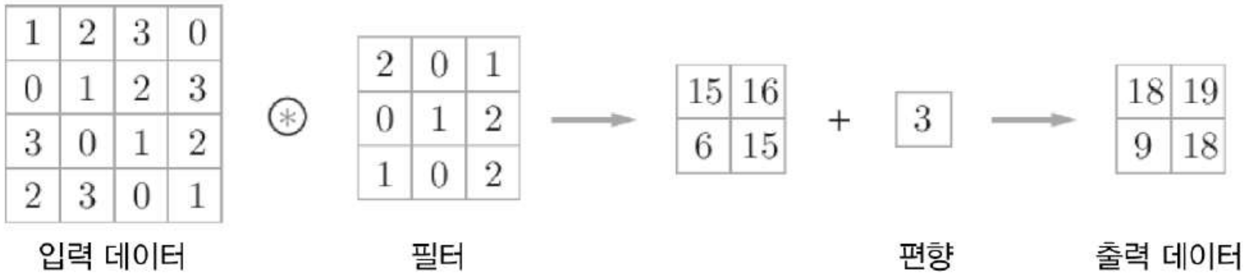
- 컨볼루션 연산
    - 1) 윈도우를 일정 간격으로 이동해가며 입력 데이터에 적용
    - 2) 입력과 필터에 대응하는 원소끼리 곱한 후 그 총합을 함 (단일 곱셈-누산 (fused multiply-add, FMA)
    - Bias 파라미터 존재함



Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolution Layer

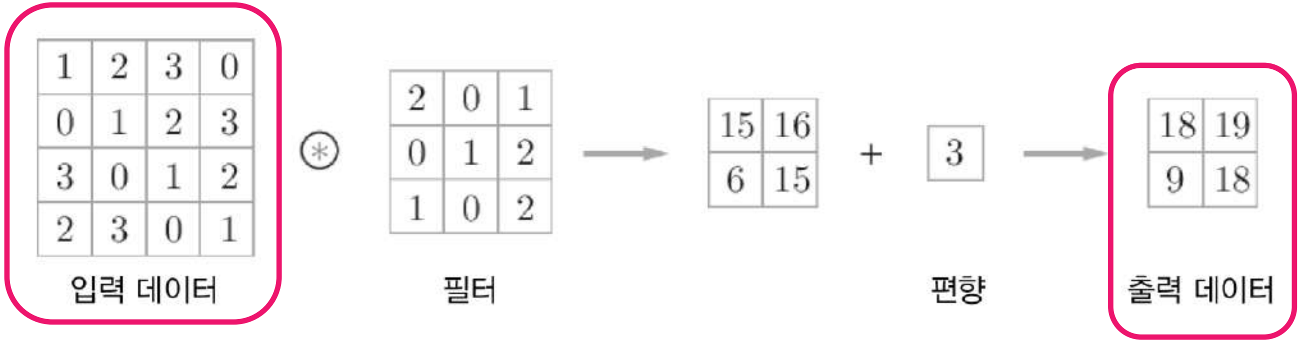
- **컨볼루션 연산**



Convolution Layer

모두의연구소

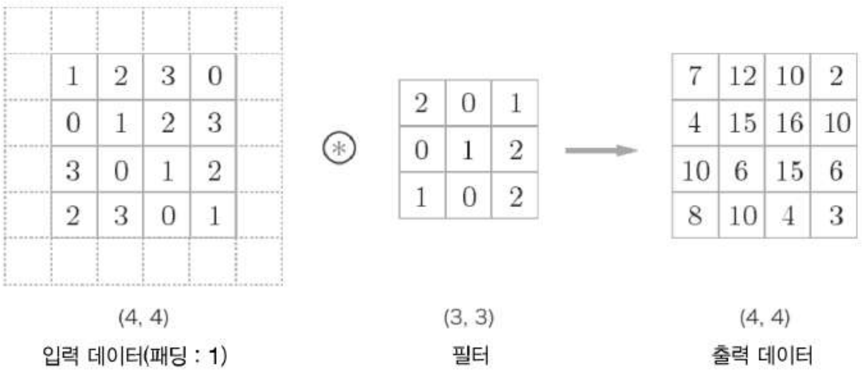
- **컨볼루션 연산**



크기가 줄어드는 군요

Convolution Layer

- 패딩 (padding)
  - 컨볼루션 연산의 패딩 처리 : 입력 데이터 주위에 0을 채운다 (패딩은 점선으로 표시했으며 그 안의 값 ‘0’은 생략함)

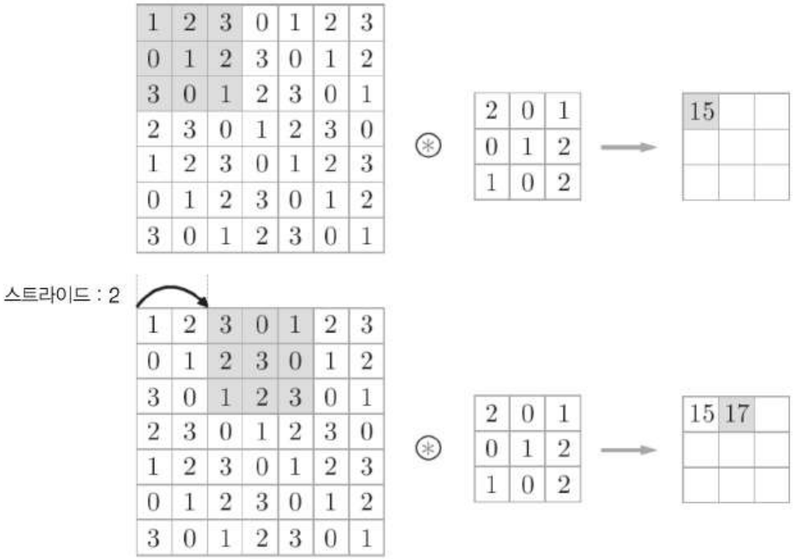


  - 패딩은 출력(특징맵(feature map))의 크기를 유지 시키고자 할때 주로 사용함

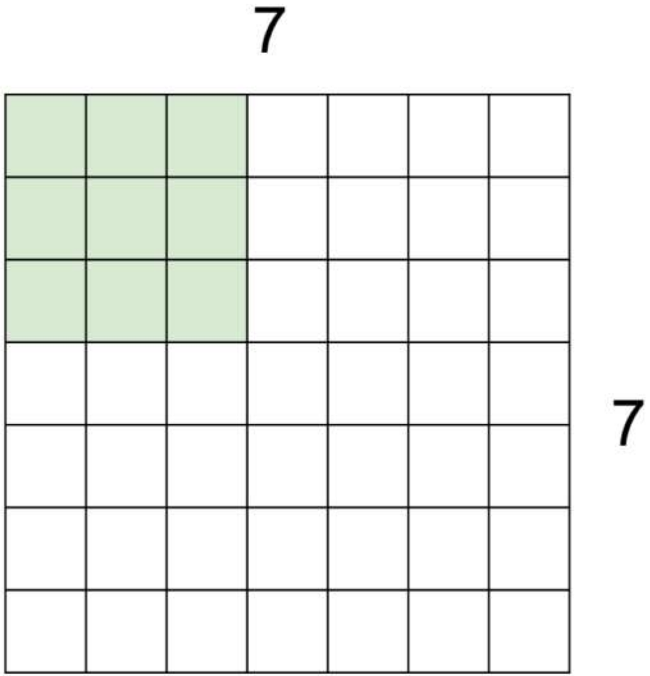
Convolution Layer

- 스트라이드 (stride)
  - 필터 적용하는 위치의 간격
    - 스트라이드를 2로 하면 필터를 적용하는 윈도우가 두 칸씩 이동함

- 스트라이드를 2로 하니 출력이 3x3이 됨



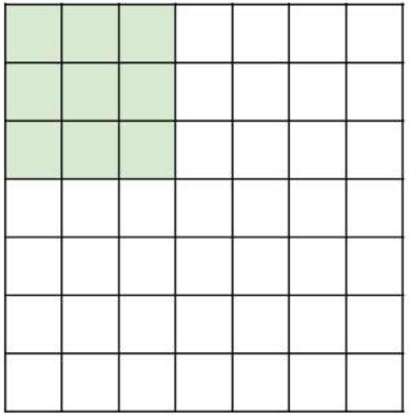
특징맵의 크기변화



7x7 input (spatially)
assume 3x3 filter

특징맵의 크기변화

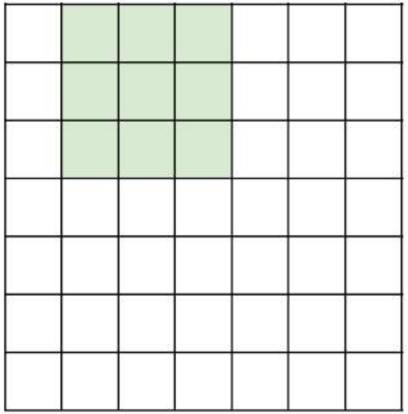
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

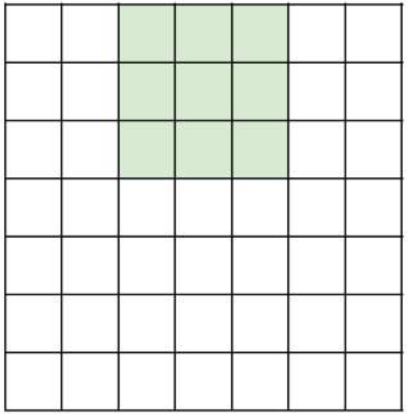
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

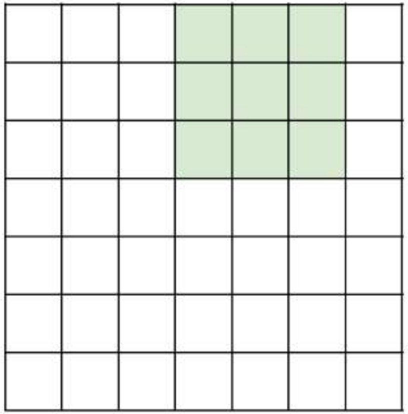
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

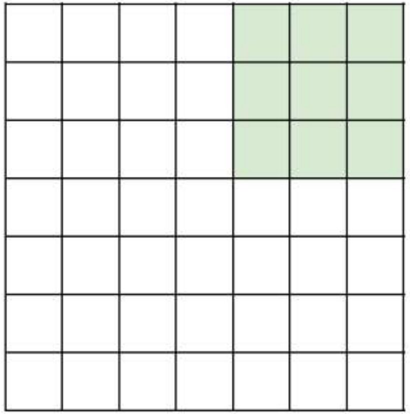
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

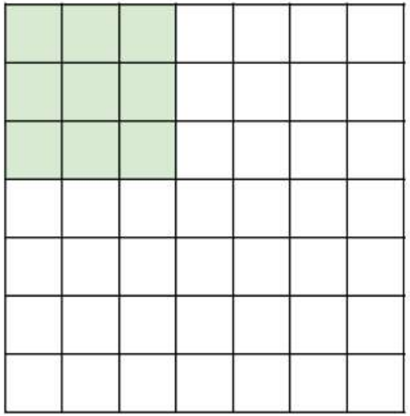
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

특징맵의 크기변화

Replicate this column of hidden neurons across space, with some **stride.**



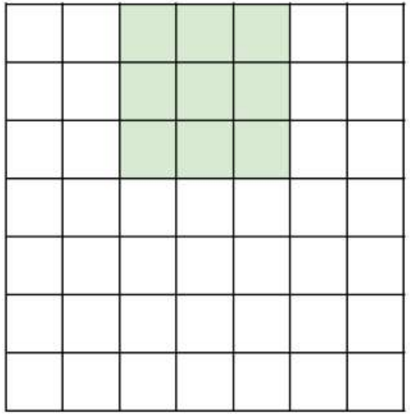
7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

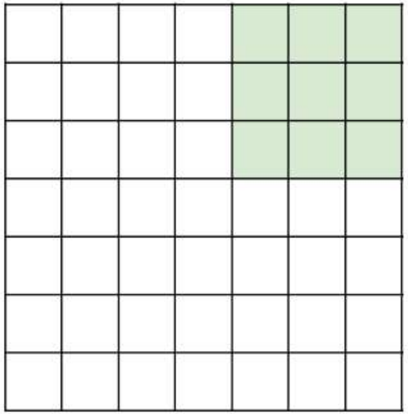


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?

특징맵의 크기변화

Replicate this column of hidden neurons across space, with some **stride.**



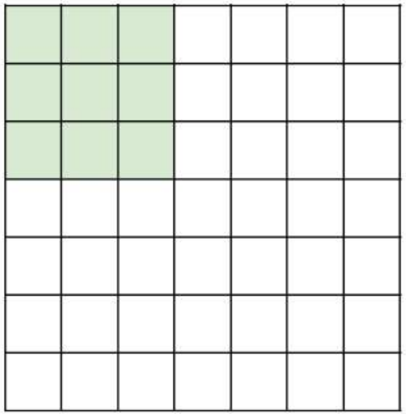
7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?
=> **3x3 output**

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

Replicate this column of hidden neurons across space, with some **stride.**

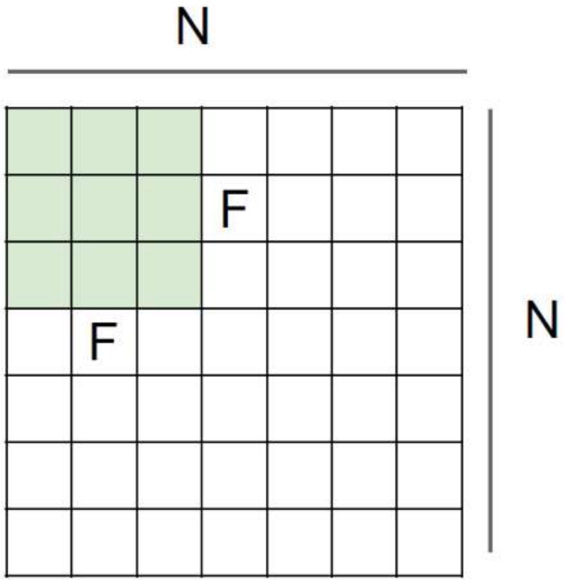


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?
=> **3x3 output**

what about stride 3? Cannot.

특징맵의 크기변화



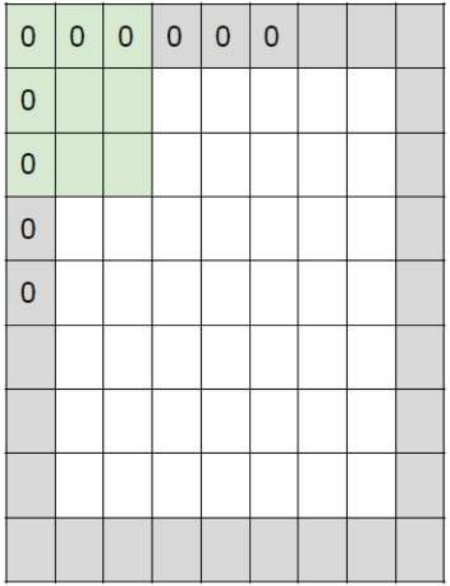
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = … :\

특징맵의 크기변화

모두의연구소

In practice: Common to zero pad the border



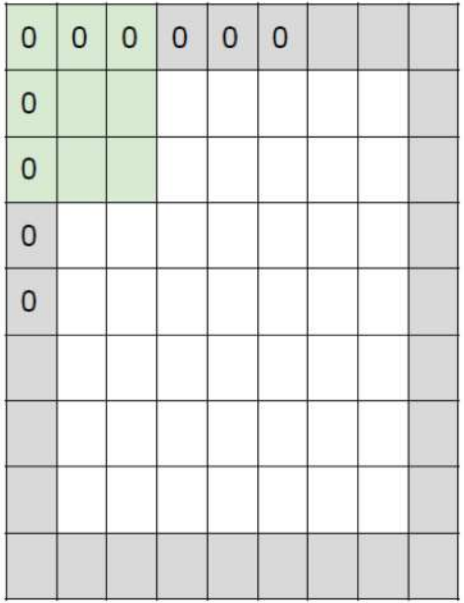
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

특징맵의 크기변화

모두의연구소

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

특징맵의 크기변화

- 출력크기 계산해 보기

- 입력크기 : (H, W)
- 필터크기 : (FH, FW)
- 출력크기 : (OH, OW)
- 패딩 : P
- 스트라이드 : S

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

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

특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

입력 : (4,4), 패딩 : 1 , 스트라이드 : 1, 필터: (3,3)

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

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

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

$$\begin{bmatrix} & & & & & \\ & 1 & 2 & 3 & 0 & \\ & 0 & 1 & 2 & 3 & \\ & 3 & 0 & 1 & 2 & \\ & 2 & 3 & 0 & 1 & \\ & & & & & \end{bmatrix} \circledast \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 7 & 12 & 10 & 2 \\ 4 & 15 & 16 & 10 \\ 10 & 6 & 15 & 6 \\ 8 & 10 & 4 & 3 \end{bmatrix}$$

(4, 4) 입력 데이터(패딩 : 1)   (3, 3) 필터   (4, 4) 출력 데이터

특징맵의 크기변화

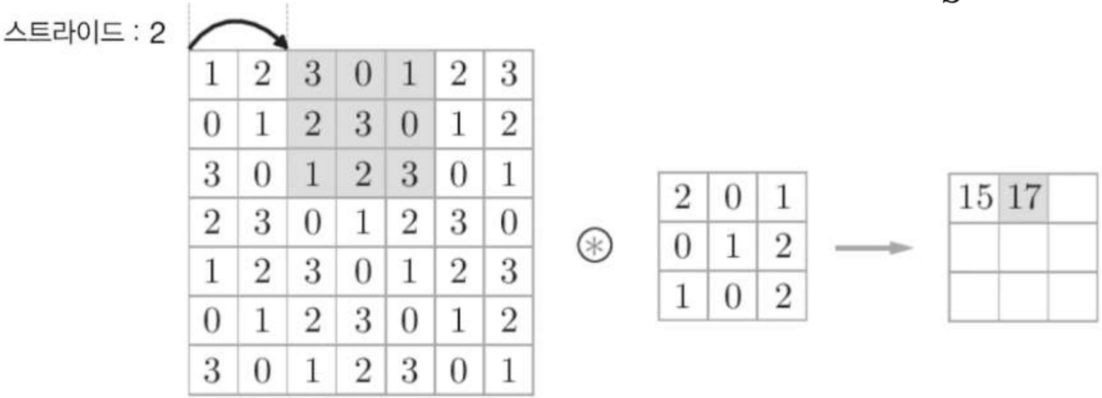
- **출력크기 계산해 보기**

입력 : (7,7), 패딩 : 0 , 스트라이드 : 2, 필터: (3,3)

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

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

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



특징맵의 크기변화

- **출력크기 계산해 보기**

입력 : (28, 31), 패딩 : 2 , 스트라이드 : 3, 필터: (5, 5)

$$OH = ?$$

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

$$OW = ?$$

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

특징맵의 크기변화

- **출력크기 계산해 보기**

입력 : (28, 31), 패딩 : 2 , 스트라이드 : 3, 필터: (5, 5)

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

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

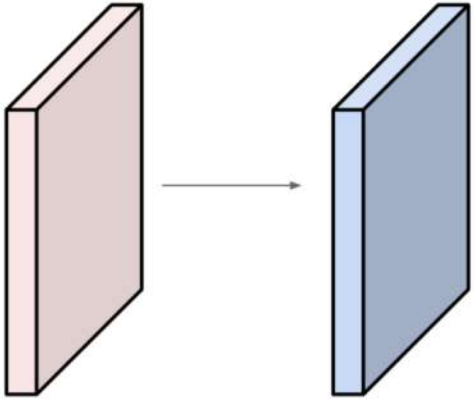
$$OH = \frac{28 + 2 \cdot 2 - 5}{3} + 1 = 10$$

$$OW = \frac{31 + 2 \cdot 2 - 5}{3} + 1 = 11$$

- ***OH, OW*가 정수가 아니면?**
  - 출력이 안나오는 것임. 오류를 내는 등의 대응 필요
  - 딥러닝 프레임웍은 가까운 정수로 내림 하는 등, 특별히 에러를 내지않고 진행되도록 구현되는 경우가 많음

특징맵의 크기변화

Examples time:

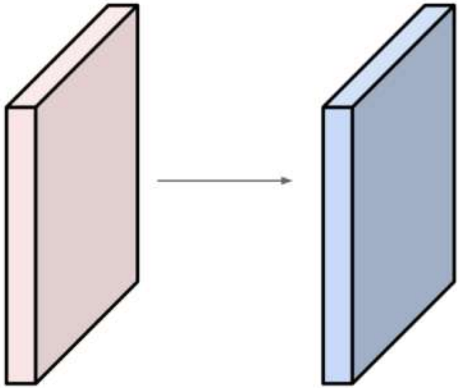


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

특징맵의 크기변화

Examples time:

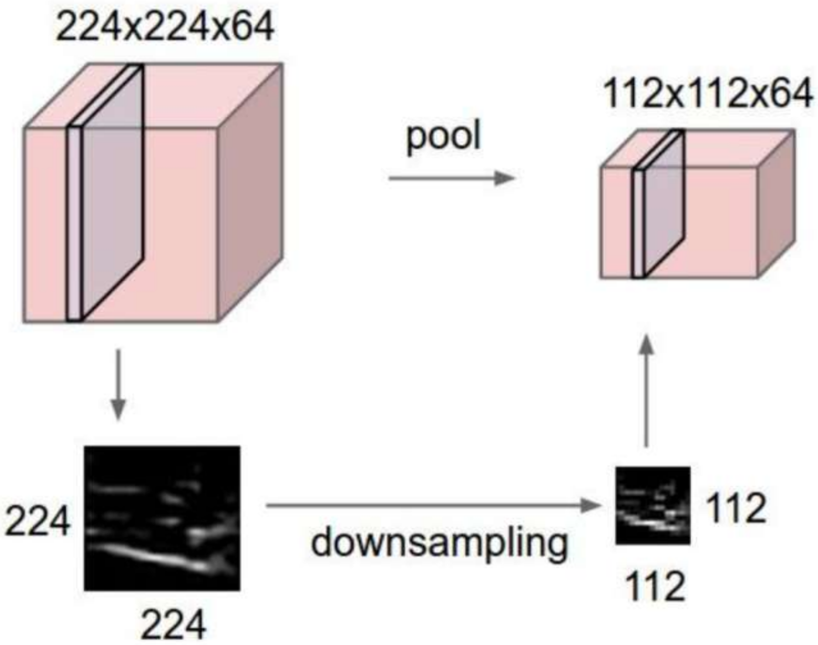


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

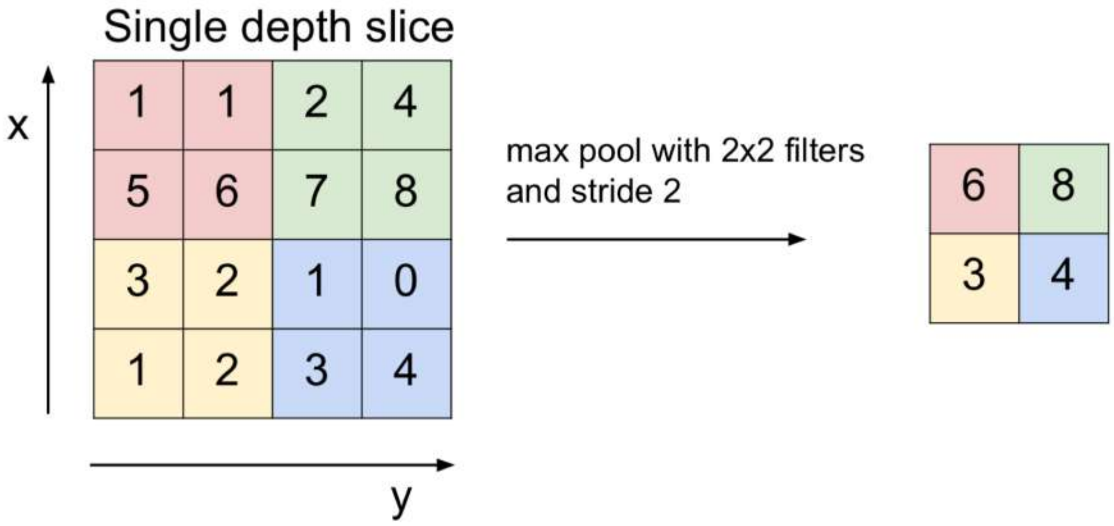
Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Pooling Layer

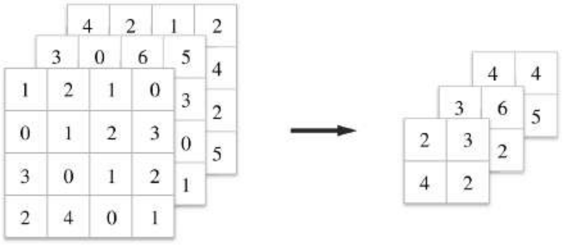
- 세로·가로 방향의 공간을 줄이는 연산
  - 2x2 최대 풀링(max pooling)을 스트라이드 2로



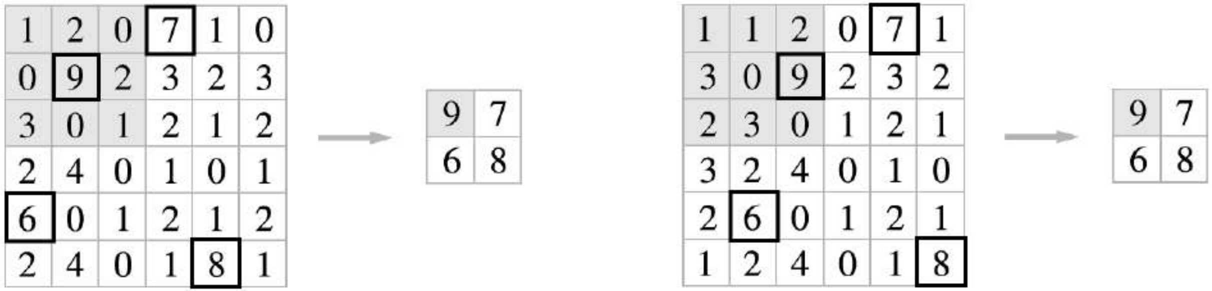
- 평균 풀링(Average pooling)도 있습니다

Pooling Layer

- 풀링 계층의 특징
  - 학습해야 할 매개변수가 없음
  - 채널 수가 변하지 않음
  - 입력의 변화에 영향을 적게 받음 (강건하다)



데이터가 오른쪽으로 1칸씩 이동한 경우



Pooling 후 특징맵 크기 변화

모두의연구소

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Common settings:

F = 2, S = 2
F = 3, S = 2

Reference : Stanford University cs231n Lecture note 6

강아지 분류기

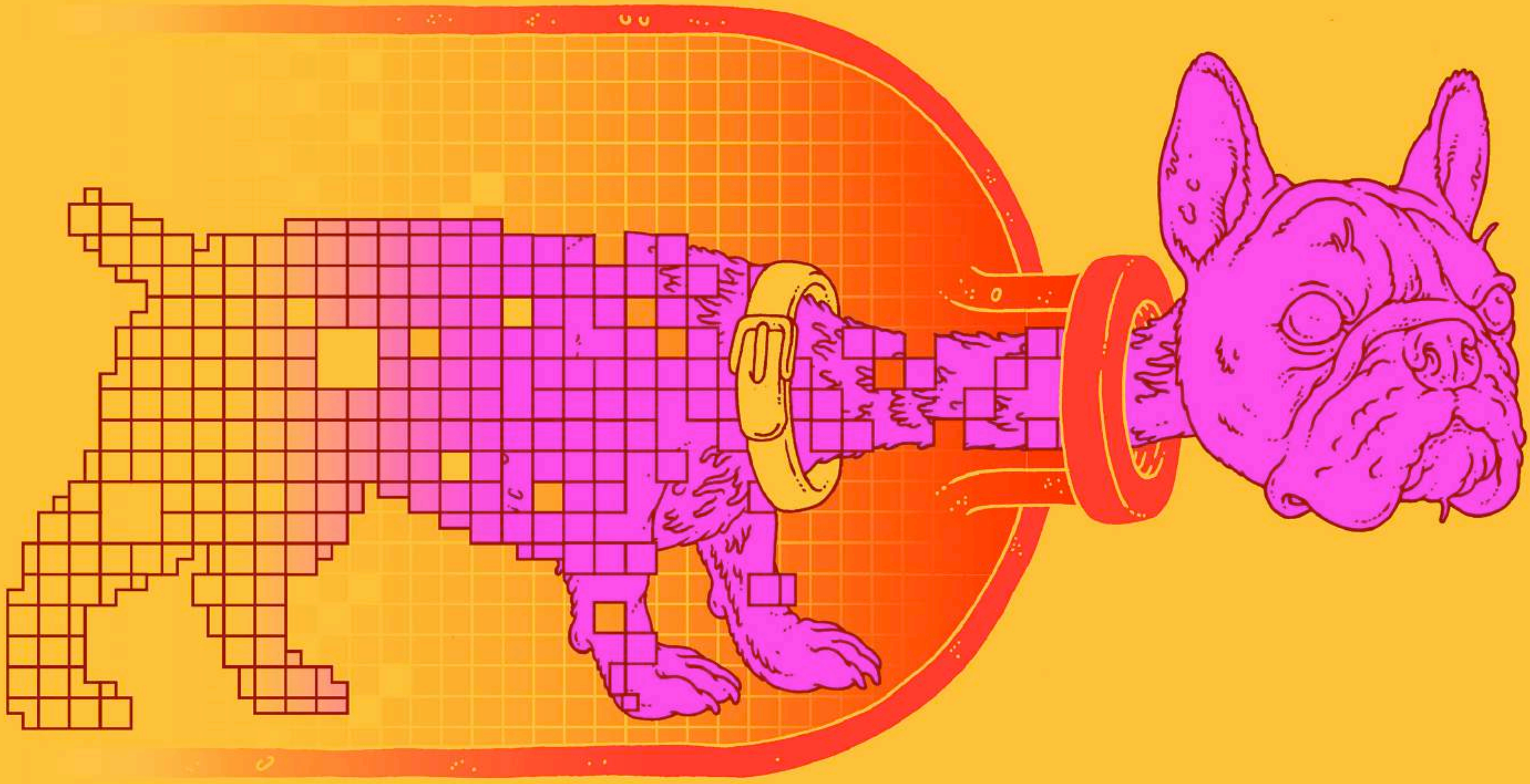


그림 : https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921/

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis

Jason Yosinski

Jeff Clune

Anh Nguyen

Thomas Fuchs

Hod Lipson

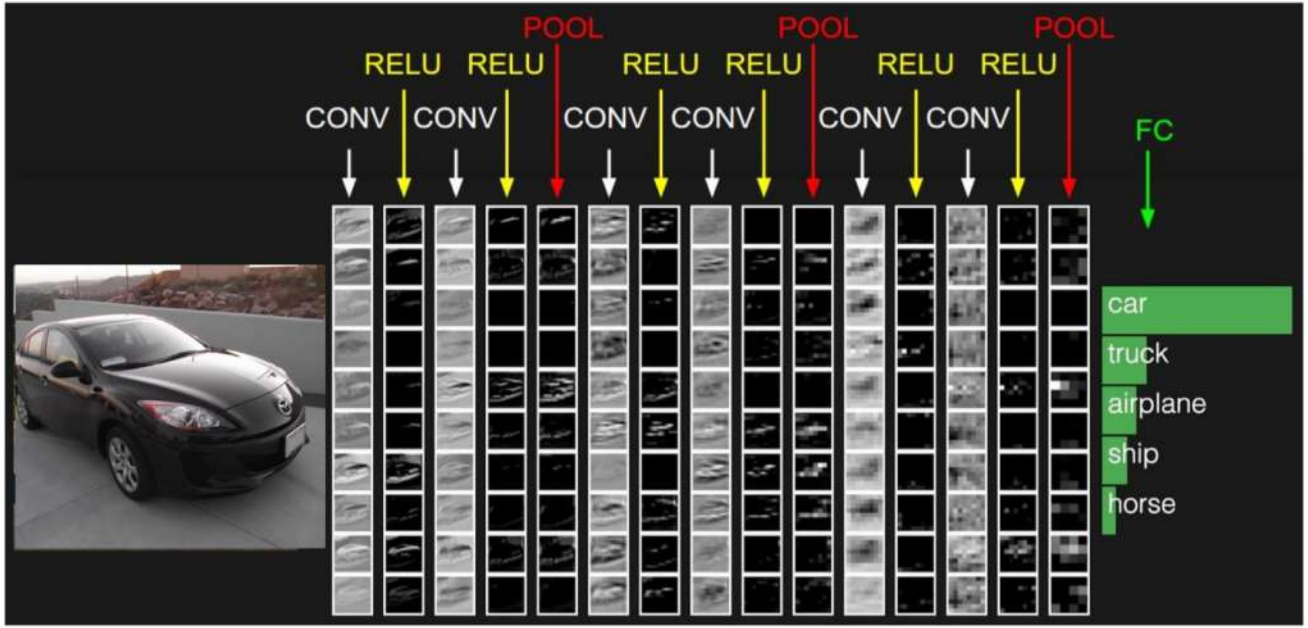
Cornell University

University of Wyoming

NASA Jet Propulsion Laboratory
California Institute of Technology

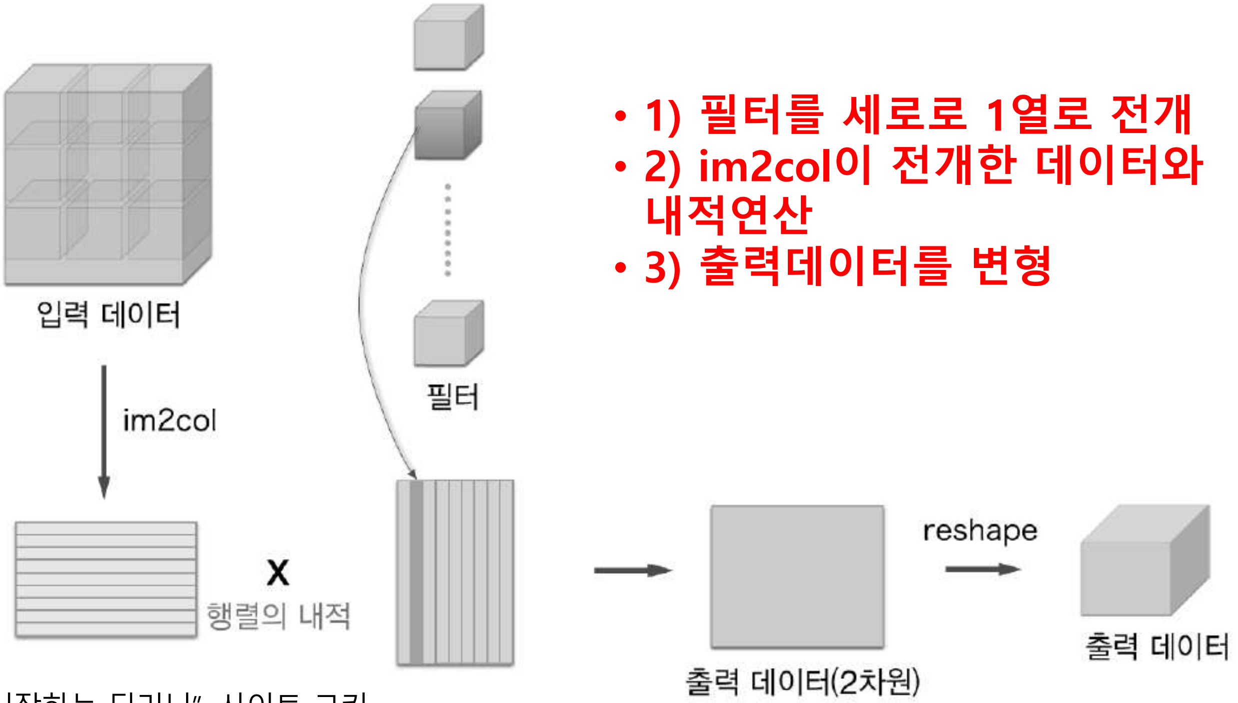
Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Convolutional Networks 구현

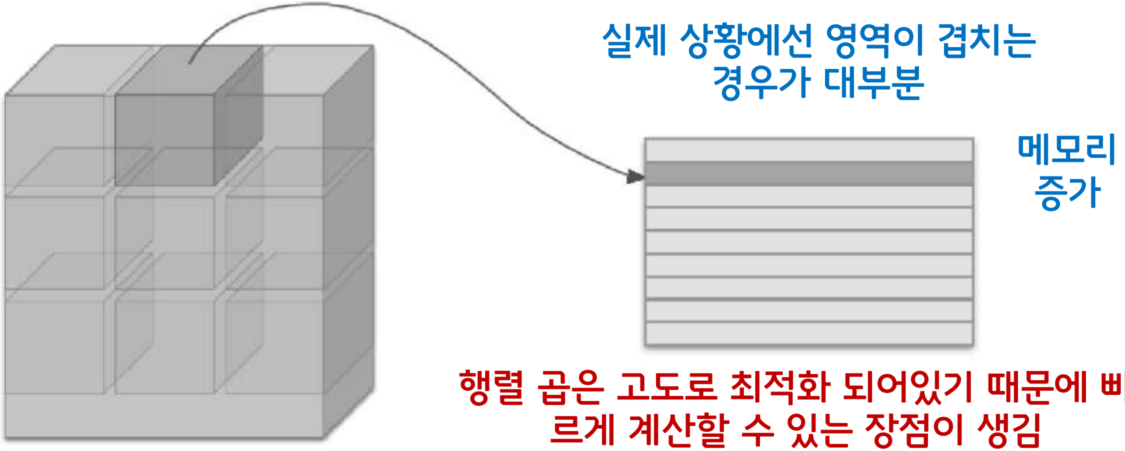
- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



- 1) 필터를 세로로 1열로 전개
- 2) im2col이 전개한 데이터와 내적연산
- 3) 출력데이터를 변형

Convolutional Networks 구현

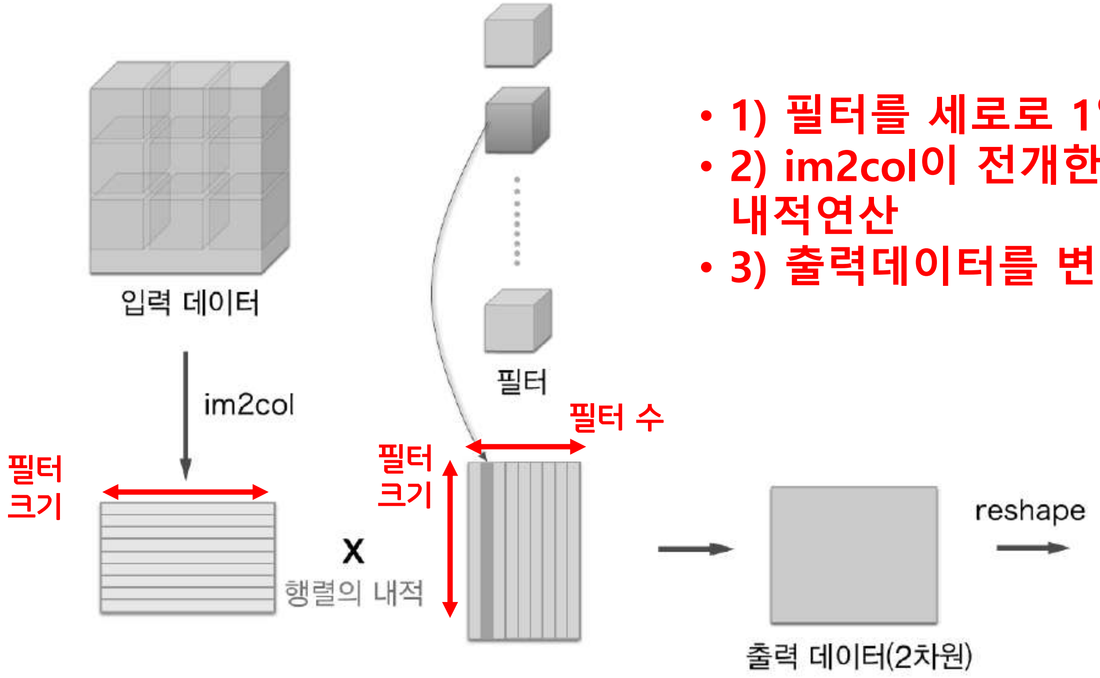
- im2col로 데이터 전개하기
  - 입력데이터에서 **필터를 적용하는 영역(3차원 블록)을 한줄**로 늘어 놓습니다
  - 이 전개를 필터를 적용하는 모든 영역에서 수행하는게 im2col입니다



필터 적용 영역을 앞에서부터 순서대로 1줄로 펼친다

Convolutional Networks 구현

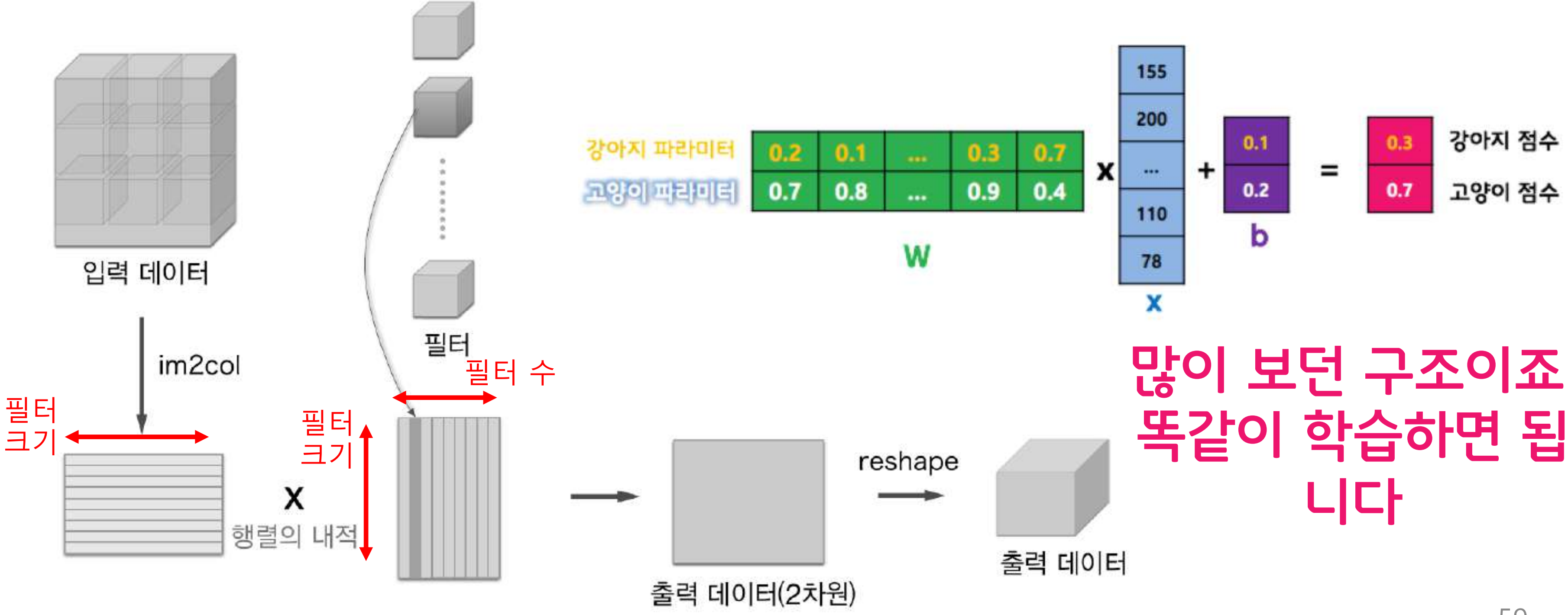
- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



- 1) 필터를 세로로 1열로 전개
- 2) im2col이 전개한 데이터와 내적연산
- 3) 출력데이터를 변형

Convolution 레이어 구현하기

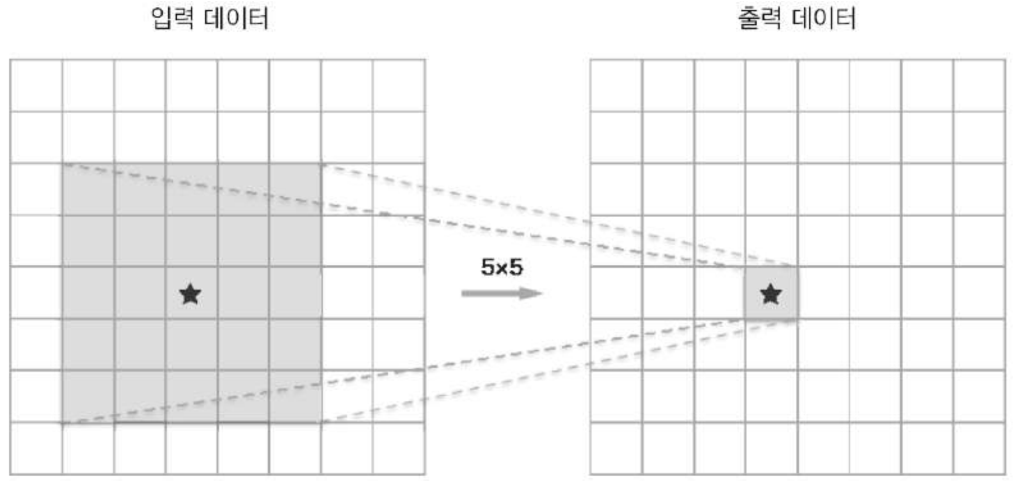
- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



많이 보던 구조이죠. 똑같이 학습하면 됩니다

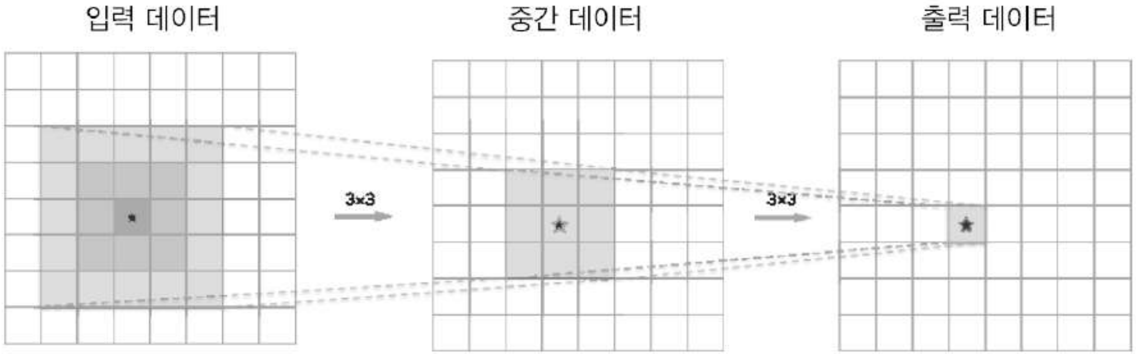
더 작은 필터

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교



더 작은 필터

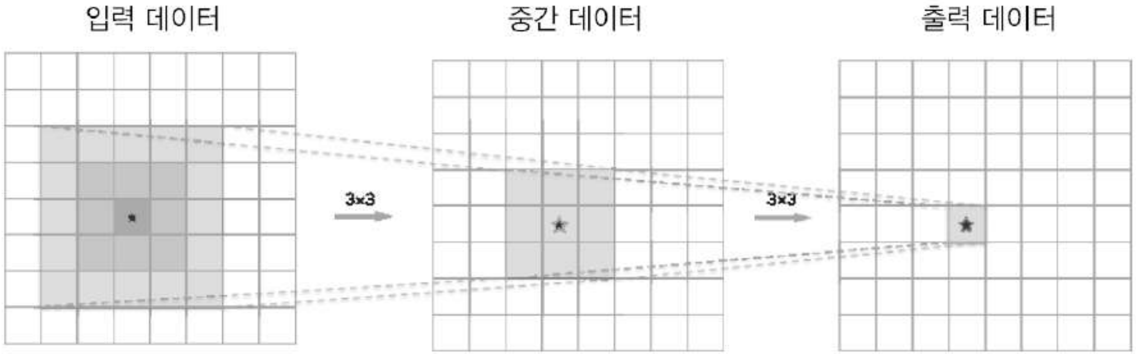
- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교



- 5x5와 같은 크기의 영역을 처리 (receptive field)
- 층이 깊어지기에 ReLU와 같은 비선형성 추가로 표현력이 개선. 비선형 함수가 겹쳐지면 더 복잡한것도 표현 가능

더 작은 필터

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교



- 매개변수 수 비교
  - 5x5 필터 1개 : 25개
  - 3x3 필터 2개 : (3x3)x2 = 18개
  - 7x7 필터 1개 : 49개
  - 3x3 필터 3개 : (3x3)x3 = 27개

CNN 예제

4_introduction-to-convnets-tensorflow.ipynb

모두의연구소

박 은 수 Research Director

E-mail : es.park@modulabs.co.kr