

# 인공지능개론

## Convolutional Neural Networks

# Neural Networks

## ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

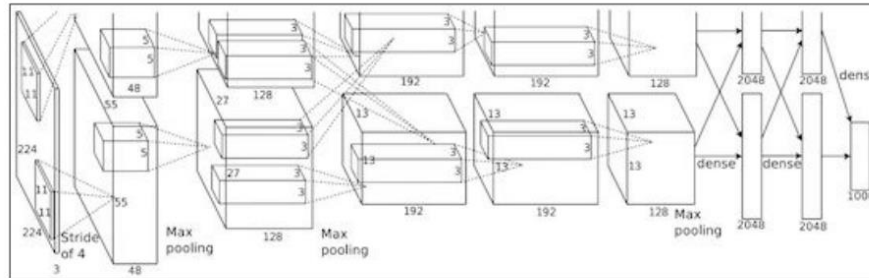
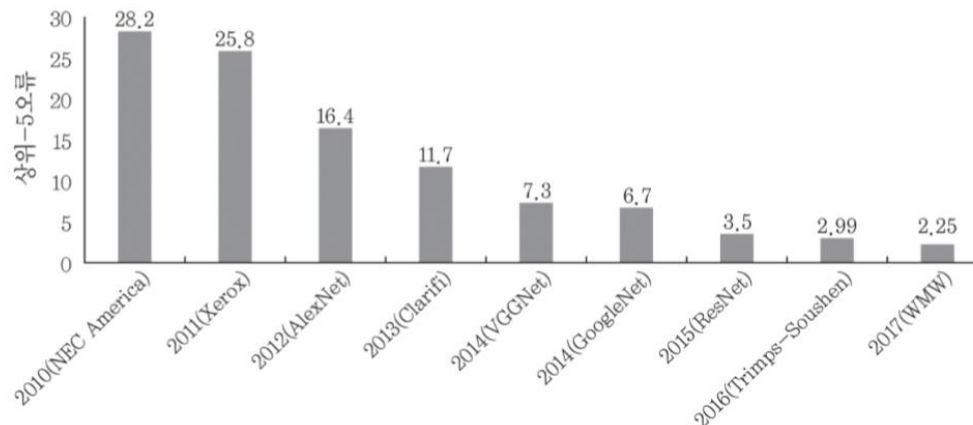


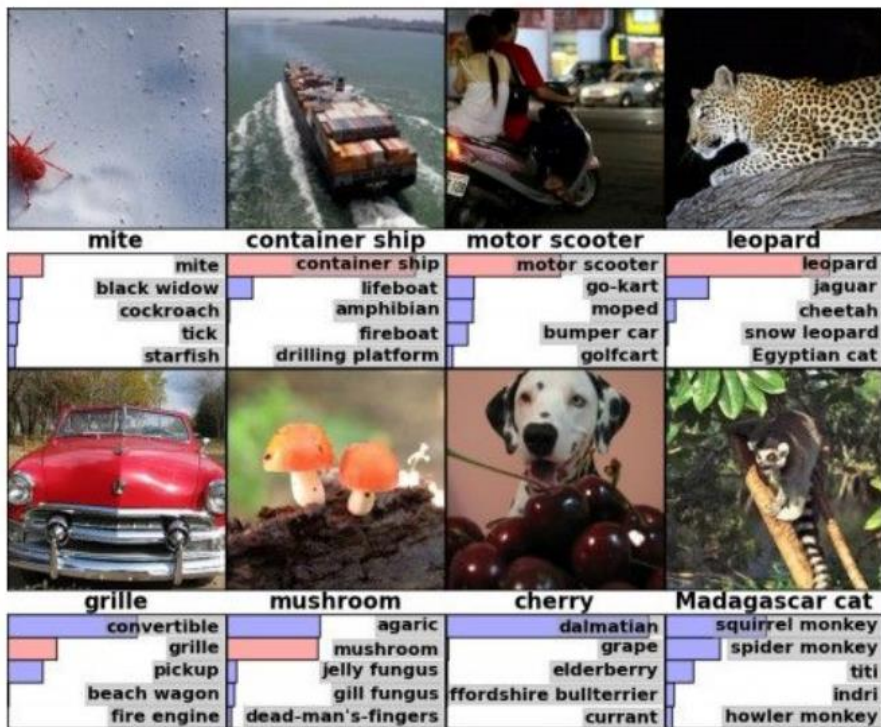
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”



# Neural Networks

## Classification



## Retrieval

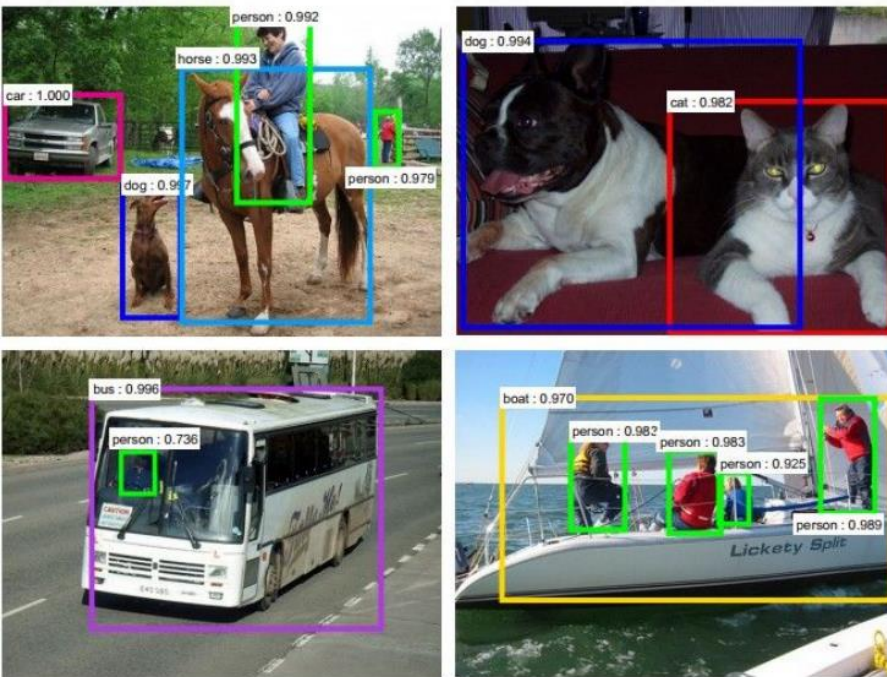


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.



# Neural Networks

## Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

*[Faster R-CNN: Ren, He, Girshick, Sun 2015]*

## Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

*[Farabet et al., 2012]*

# Neural Networks



Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars



[This image](#) by GBPublic\_PR is licensed under [CC-BY 2.0](#)

## NVIDIA Tesla line

(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.



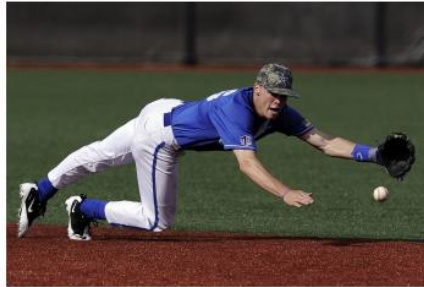
# Neural Networks

No errors



*A white teddy bear sitting in the grass*

Minor errors



*A man in a baseball uniform throwing a ball*

Somewhat related



*A woman is holding a cat in her hand*

## Image Captioning

[Vinyals et al., 2015]  
[Karpathy and Fei-Fei, 2015]



*A man riding a wave on top of a surfboard*



*A cat sitting on a suitcase on the floor*



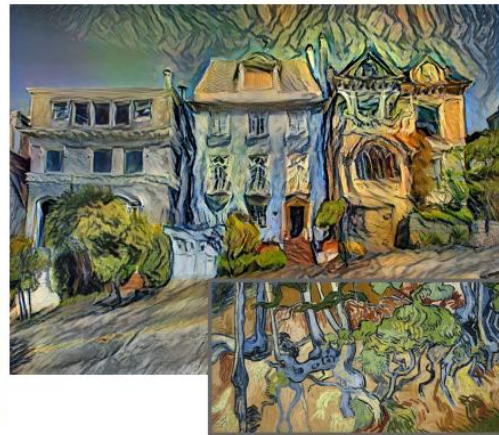
*A woman standing on a beach holding a surfboard*

All images are CC0 Public domain:

<https://pixabay.com/en/luggage-antique-cat-1643010/>  
<https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/>  
<https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/>  
<https://pixabay.com/en/woman-female-model-portrait-adult-983967/>  
<https://pixabay.com/en/handstand-lake-meditation-496008/>  
<https://pixabay.com/en/baseball-player-shortstop-infield-1045263/>

Captions generated by Justin Johnson using [NeuralTalk2](#)

# Neural Networks



[Original image](#) is CC0 public domain  
[Starry Night](#) and [Tree Roots](#) by Van Gogh are in the public domain  
[Bokeh image](#) is in the public domain  
Stylized images copyright Justin Johnson, 2017;  
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016  
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017



[Sermanet et al. 2011]  
[Ciresan et al.]

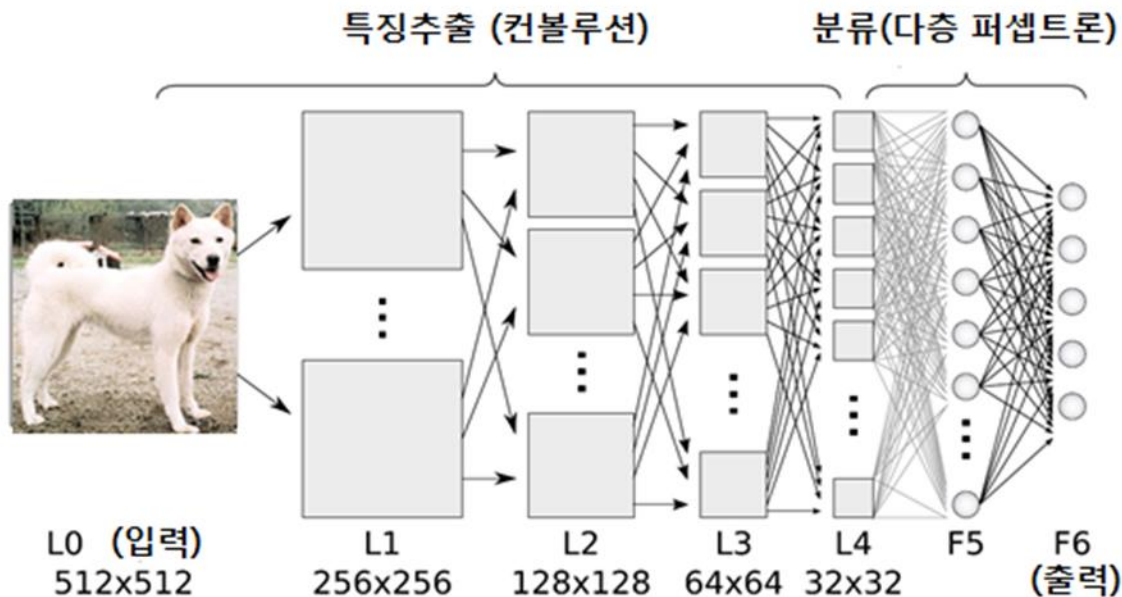
Photos by Lane McIntosh.  
Copyright CS231n 2017.



# Neural Networks

## ❖ 컨볼루션 신경망(Convolutional Neural Network, **CNN**)

- 전반부 : 컨볼루션 연산을 수행하여 **특징 추출**
- 후반부 : 특징을 이용하여 **분류**
- 영상분류, 문자 인식 등 인식문제에 높은 성능





# Neural Networks

## ❖ 컨볼루션(covolution)

- 일정 영역의 값들에 대해 가중치를 적용하여 하나의 값을 만드는 연산

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$	$x_{25}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$	$x_{35}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$	$x_{45}$
$x_{51}$	$x_{52}$	$x_{53}$	$x_{54}$	$x_{55}$

입력

$w_{11}$	$w_{12}$	$w_{13}$
$w_{21}$	$w_{22}$	$w_{23}$
$w_{31}$	$w_{32}$	$w_{33}$

컨볼루션 필터  
커널  
마스크

$y_{11}$	$y_{12}$	$y_{13}$
$y_{21}$	$y_{22}$	$y_{23}$
$y_{31}$	$y_{32}$	$y_{33}$

컨볼루션 결과

$$\begin{aligned}
 y_{11} = & w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13} \\
 & + w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23} \\
 & + w_{31}x_{31} + w_{32}x_{32} + w_{33}x_{33} \\
 & + w_0
 \end{aligned}$$

# Neural Networks

## ❖ 컨볼루션

11	10	10	00	01
00	10	10	10	00
00	00	10	10	10
00	00	10	10	00
01	10	10	00	01

입력

1	0	1
0	1	0
1	0	1

컨볼루션 필터  
커널  
마스크

4	3	4
2	4	3
2	3	4

컨볼루션 결과

$$\begin{aligned} y_{11} = & w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13} \\ & + w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23} \\ & + w_{31}x_{31} + w_{32}x_{32} + w_{33}x_{33} \\ & + w_0 \end{aligned}$$

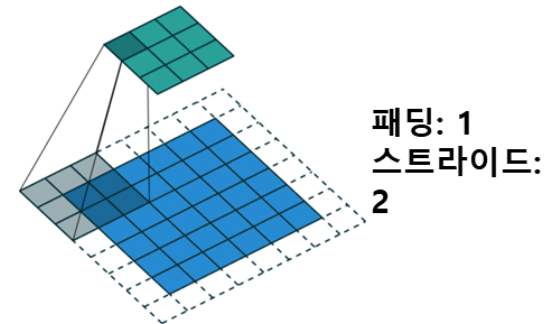
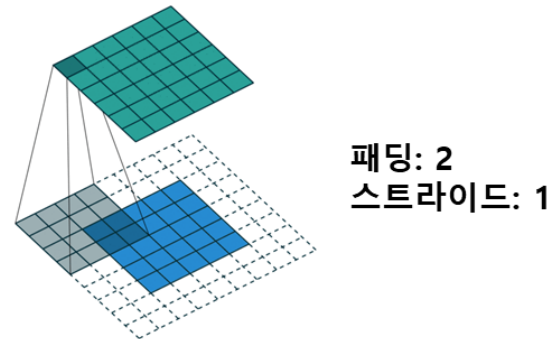
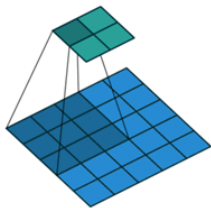
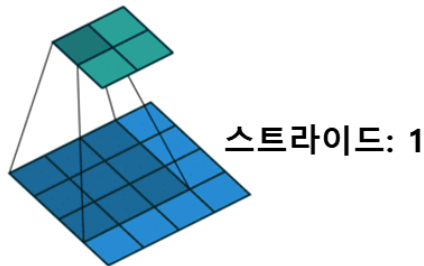
# Neural Networks

## ❖ 스트라이드(stride, 보폭)

- 커널을 다음 컨볼루션 연산을 위해 이동시키는 칸 수

## ❖ 패딩(padding)

- 컨볼루션 결과의 크기를 조정하기 위해 입력 배열의 둘레를 확장하고 0으로 채우는 연산

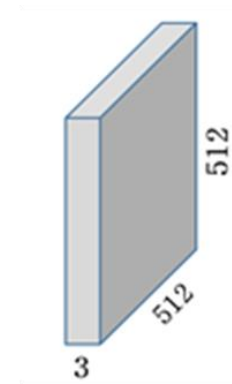
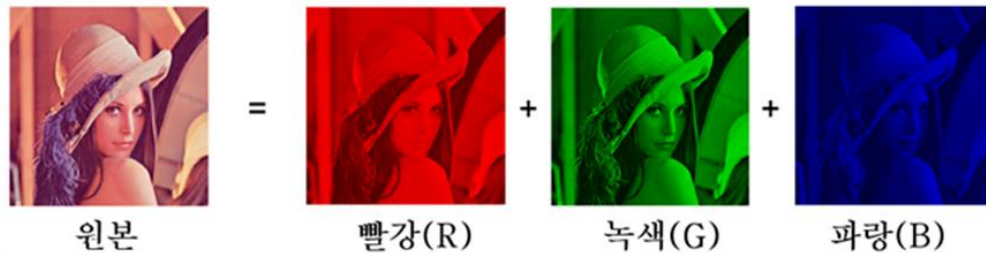




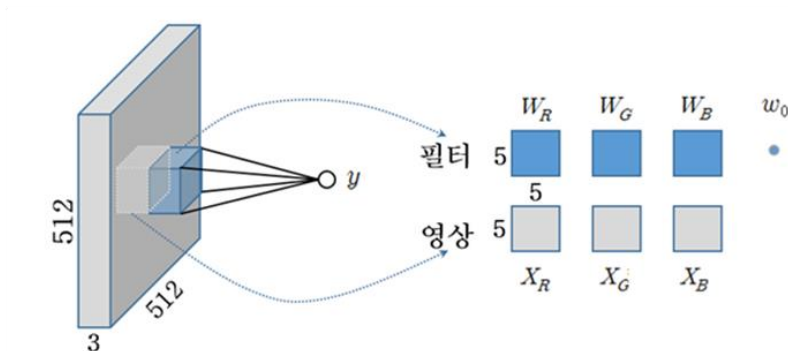
# Neural Networks

## ❖ 컬러 영상의 컨볼루션

- 컬러 영상의 다차원 행렬 표현



- 컬러영상의 컨볼루션

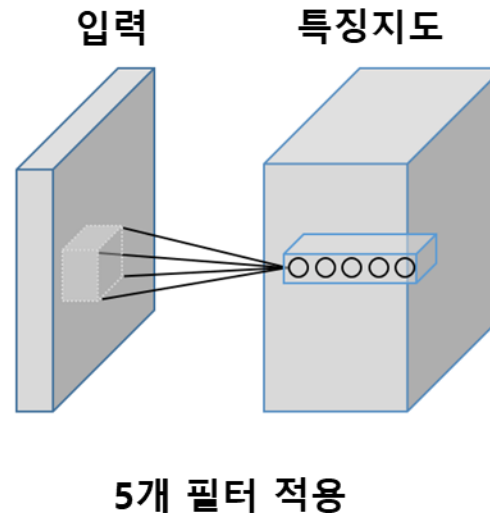
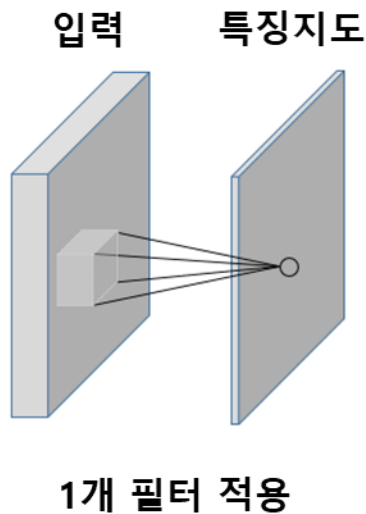


$$y = X_R^* W_R + X_G^* W_G + X_B^* W_B + w_0$$

# Neural Networks

## ❖ 특징지도(feature map)

- 컨볼루션 필터의 적용 결과로 만들어지는 2차원 행렬
- 특징지도의 원소값
  - 컨볼루션 필터에 표현된 특징을 대응되는 위치에 포함하고 있는 정도
- k개의 컨볼루션 필터를 적용하면 k의 2차원 특징지도 생성




# Neural Networks

## ❖ 풀링(pooling)

- 일정 크기의 블록을 통합하여 하나의 대푯값으로 대체하는 연산
- **최대값 풀링(max pooling)**
  - 지정된 블록 내의 원소들 중에서 최대값을 대푯값으로 선택


1	1	2	3
4	6	6	8
3	1	1	0
1	2	2	4



6	8
3	4

- **평균값 풀링(average pooling)**
  - 블록 내의 원소들의 평균값을 대푯값으로 사용

1	1	2	3
4	6	6	8
3	1	1	0
1	2	2	4



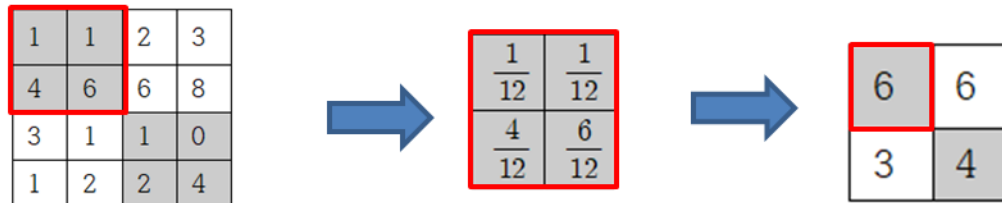
3	4.75
1.75	1.75



# Neural Networks

- **확률적 풀링(stochastic pooling)**

- 블록 내의 각 원소가 원소값의 크기에 비례하는 선택 확률을 갖도록 하고, 이 확률에 따라 원소 하나를 선택



- 학습시: 확률적 풀링

$$p_i = \frac{a_i}{\sum_{k \in R_j} a_k} \quad p_i : \text{블록 } R_j \text{에서 원소 } a_i \text{가 선택될 확률}$$

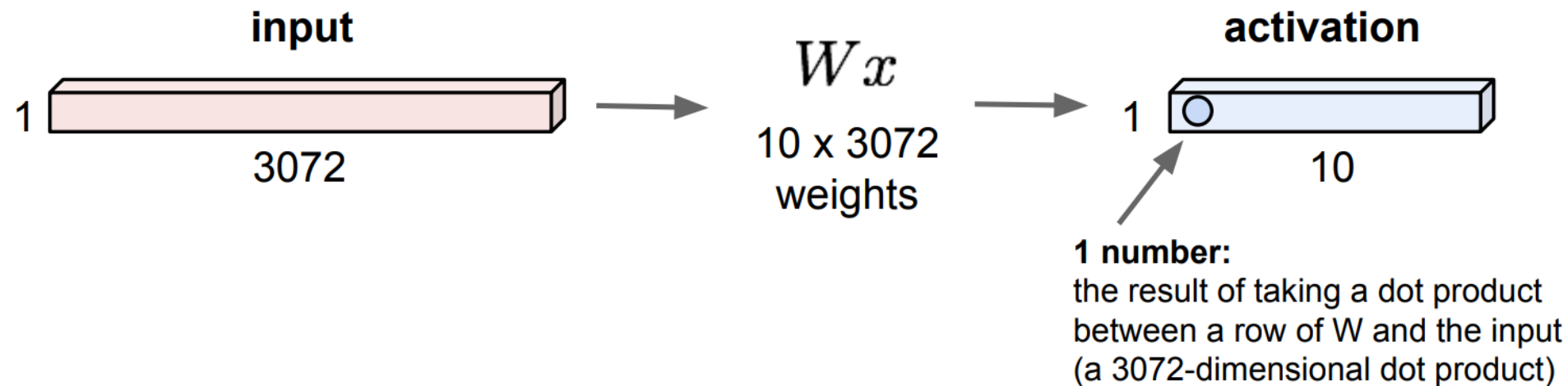
- 추론시 : 확률적 가중합 사용

$$s_j = \sum_{i \in R_j} p_i a_i$$

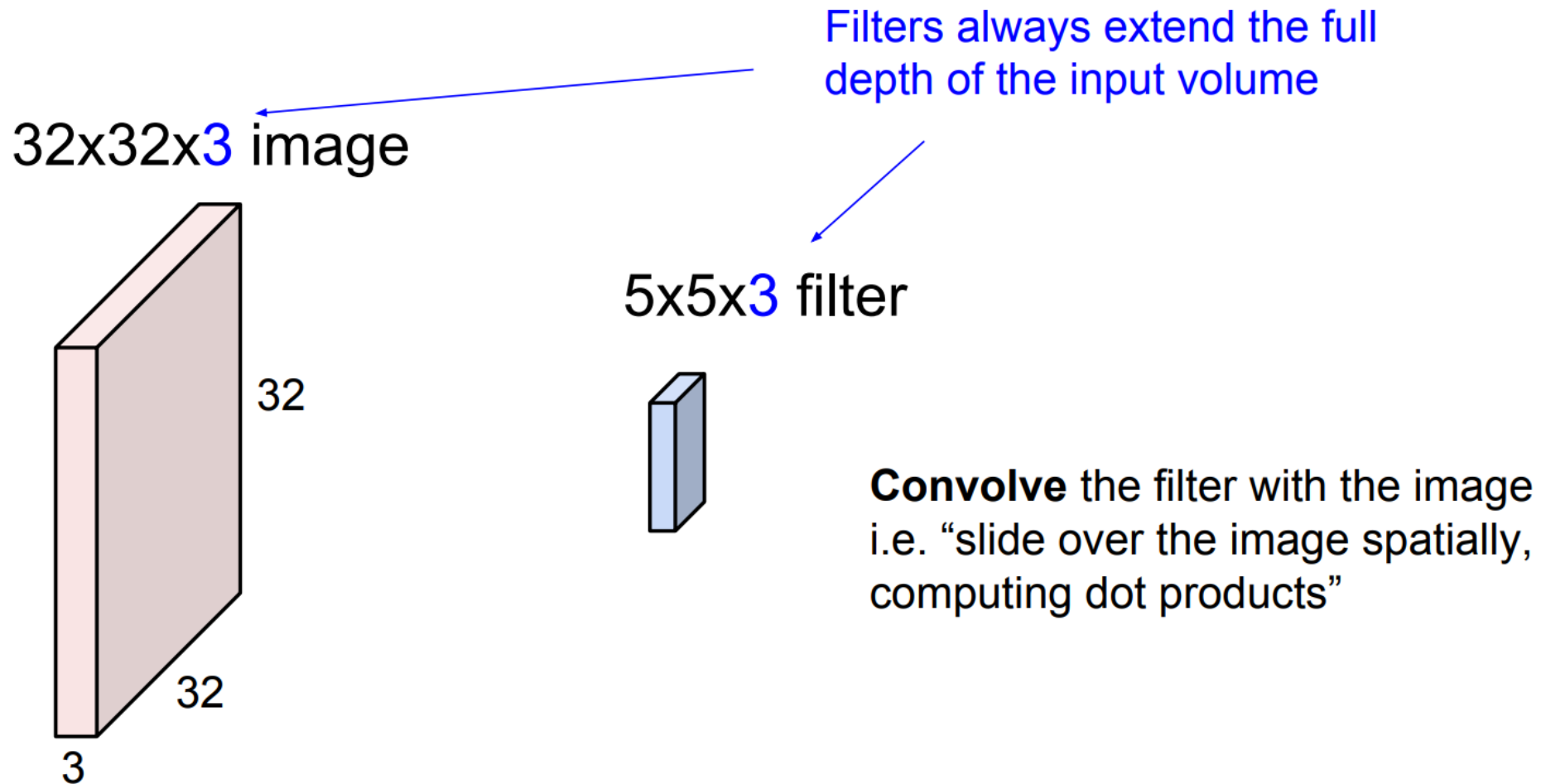
# Neural Networks

## Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

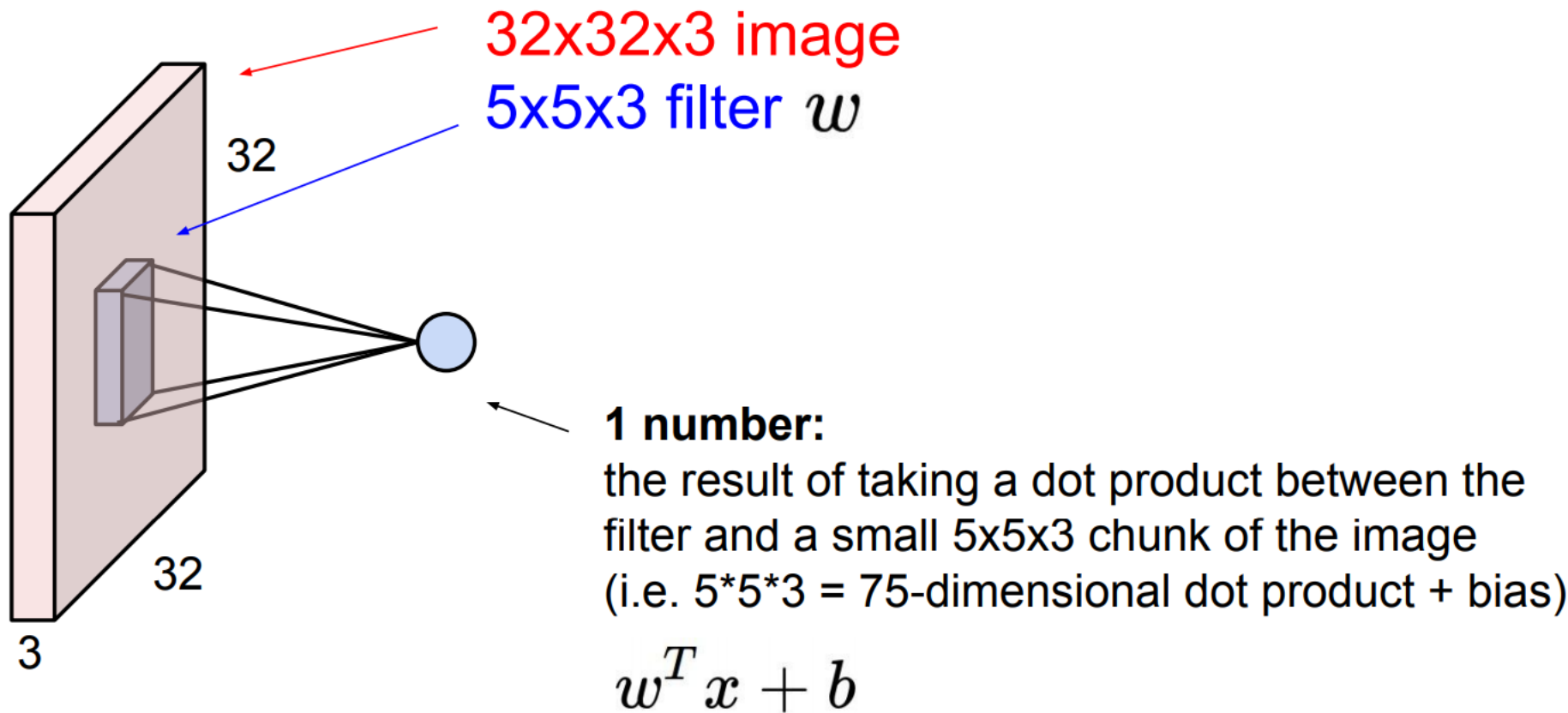


# Neural Networks

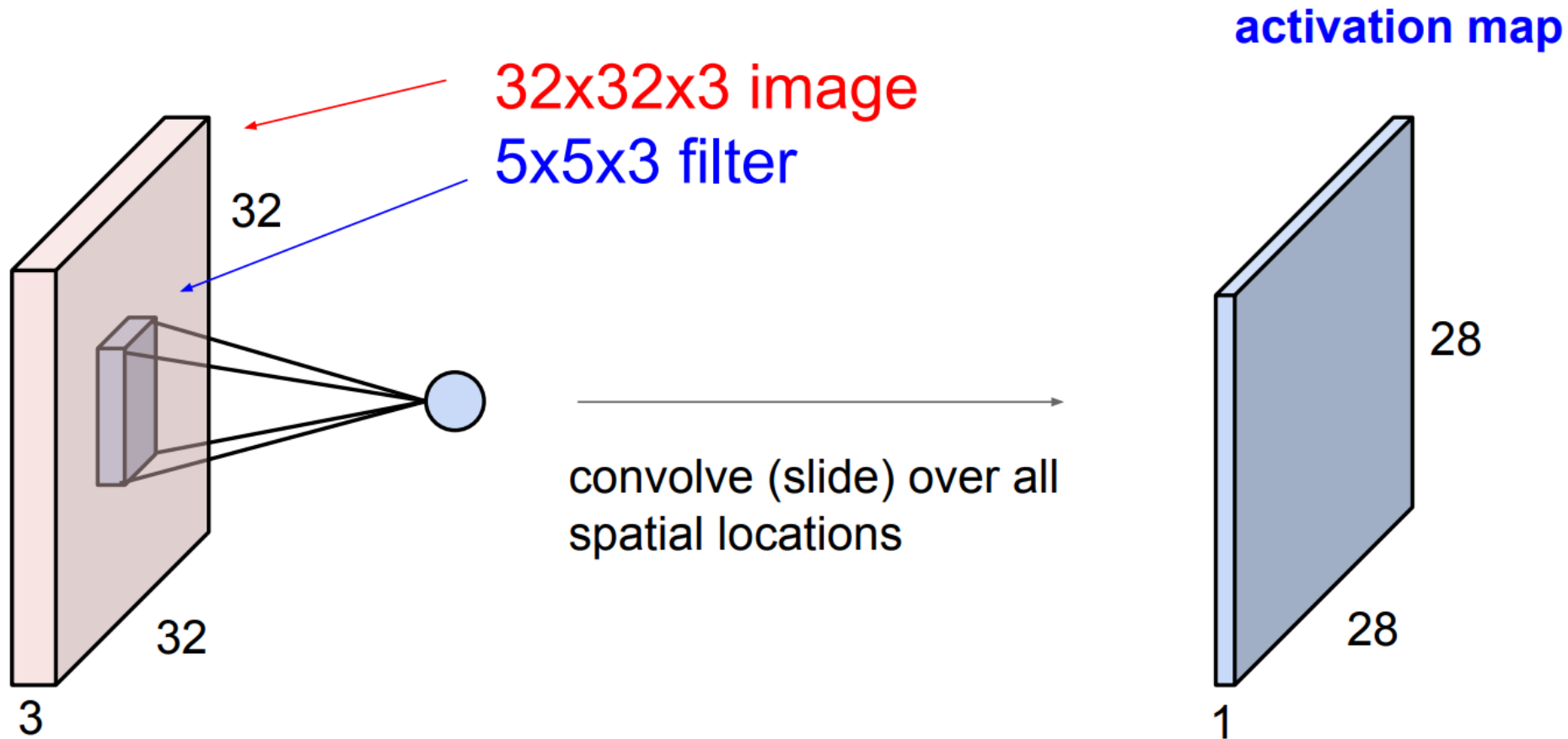




# Neural Networks

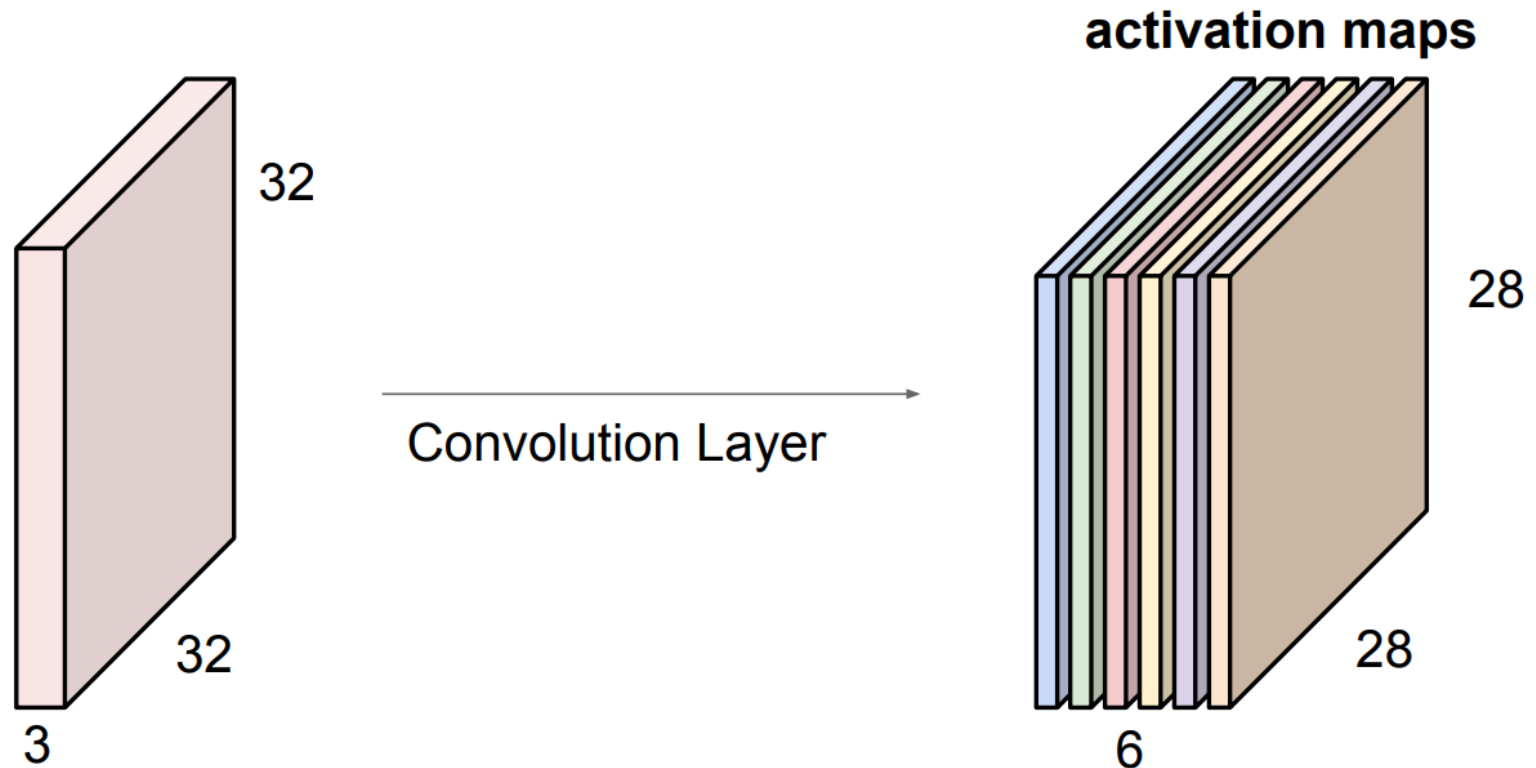


# Neural Networks



# Neural Networks

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

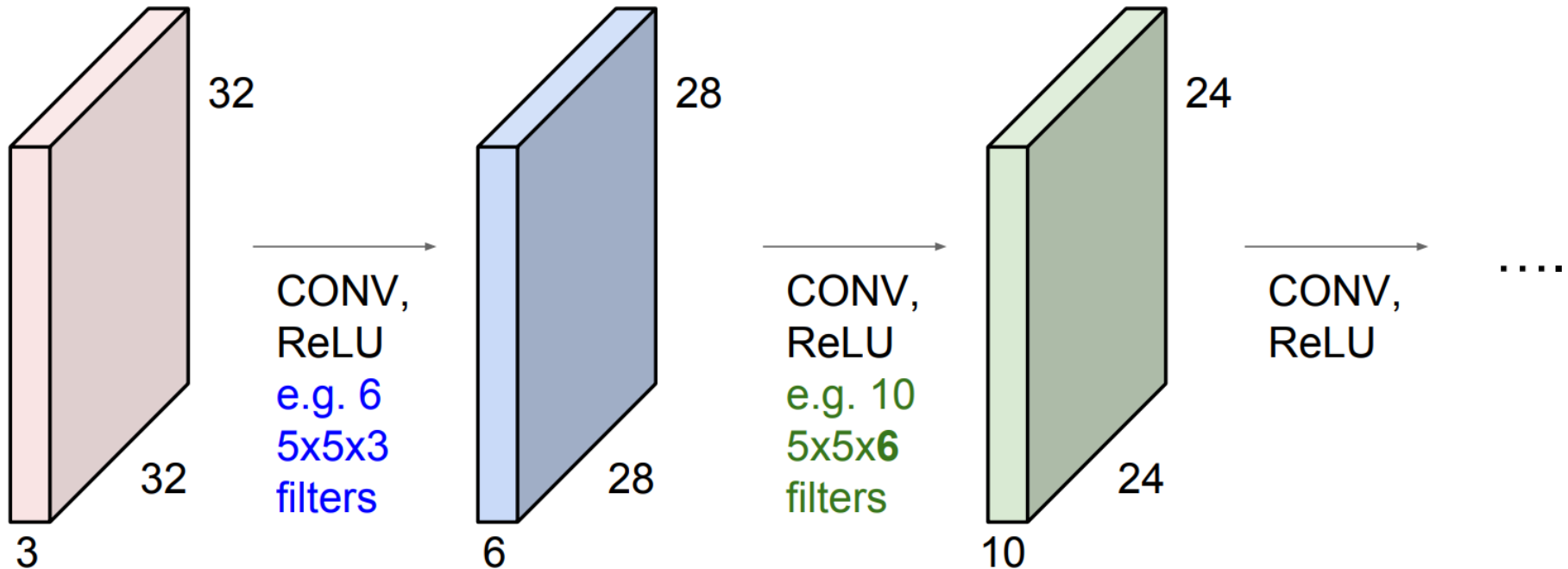


We stack these up to get a “new image” of size 28x28x6!

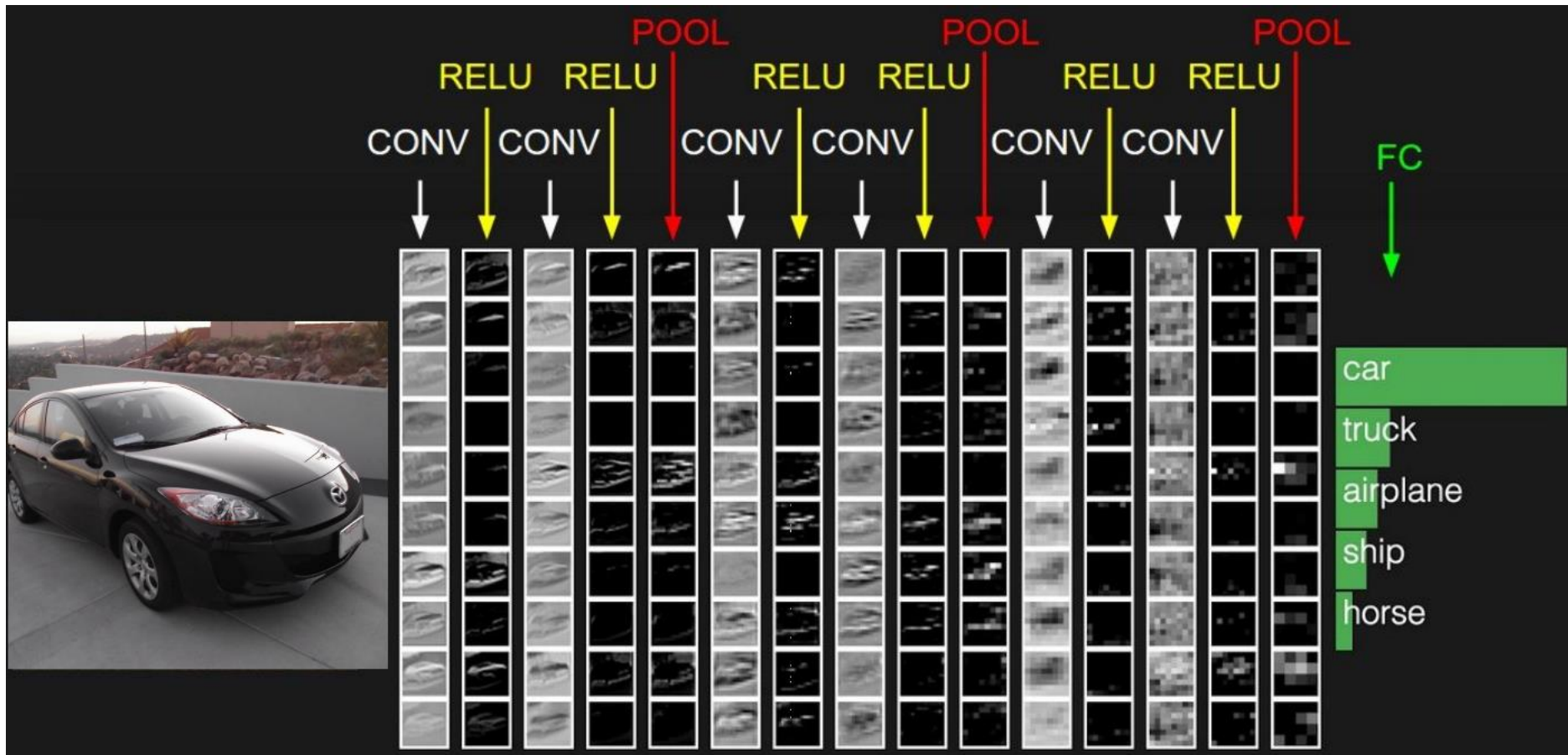


# Neural Networks

**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



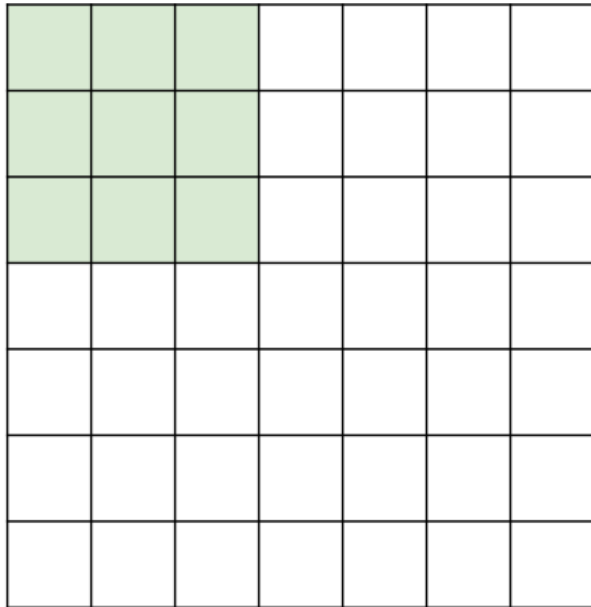
# Neural Networks



# Neural Networks

A closer look at spatial dimensions:

7



7x7 input (spatially)  
assume 3x3 filter

7

With stride 1 : 5x5 output  
With stride 2 : 3x3 output  
With stride 3 : ?



# Neural Networks

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

**3x3** filter, applied with **stride 1**

**pad with 1 pixel** border => what is the output?

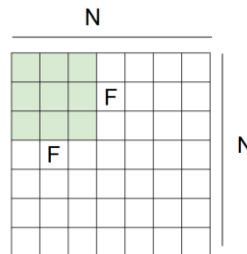
**7x7 output!**

in general, common to see CONV layers with stride 1, filters of size  $F \times F$ , and zero-padding with  $(F-1)/2$ . (will preserve size spatially)

e.g.  $F = 3 \Rightarrow$  zero pad with 1

$F = 5 \Rightarrow$  zero pad with 2

$F = 7 \Rightarrow$  zero pad with 3



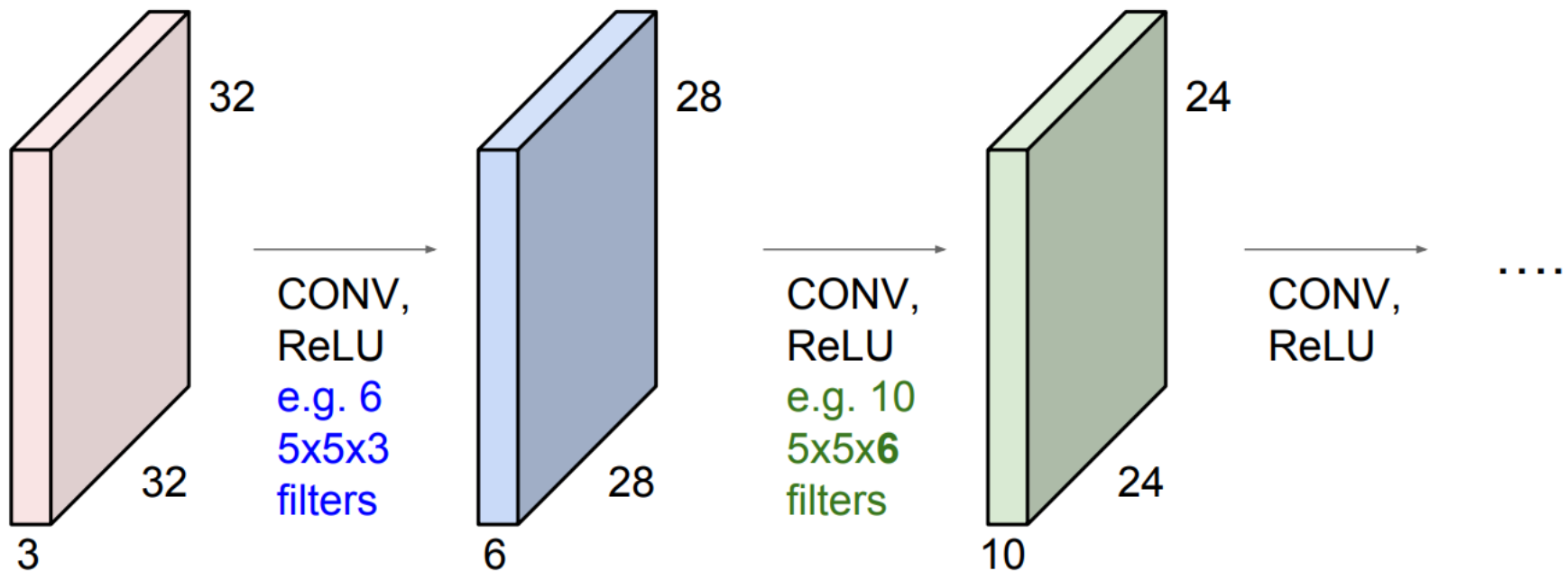
(recall:)

$$(N - F) / \text{stride} + 1$$

# Neural Networks

## Remember back to...

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

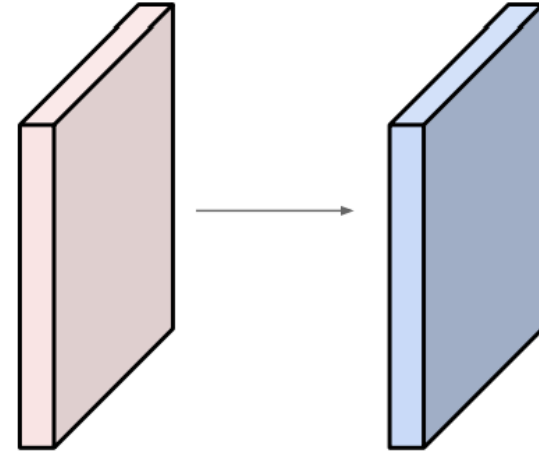


# Neural Networks

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Output volume size: ?  $(32+2*2-5)/1+1 = 32$  spatially, so  
**32x32x10**

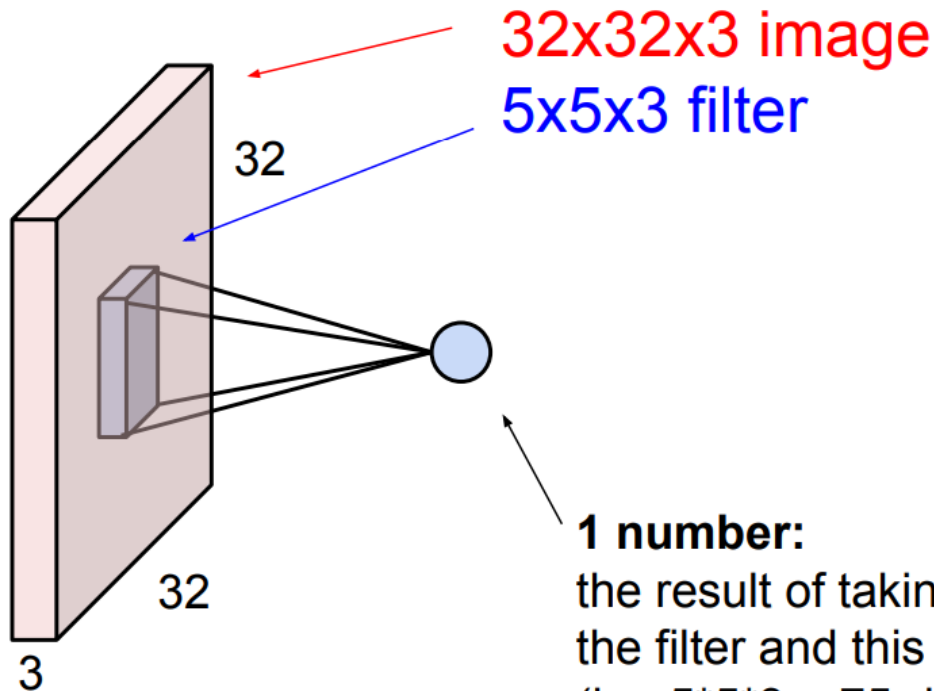
Number of parameters in this layer?

each filter has  $5*5*3 + 1 = 76$  params (+1 for bias)

=>  $76*10 = 760$

# Neural Networks

## The brain/neuron view of CONV Layer

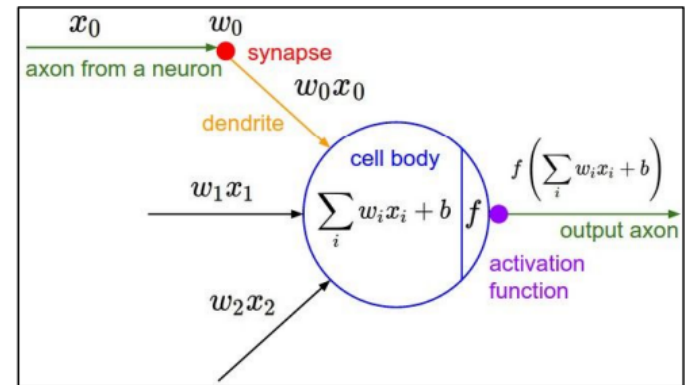


### 1 number:

the result of taking a dot product between the filter and this part of the image (i.e.  $5 \times 5 \times 3 = 75$ -dimensional dot product)

An activation map is a  $28 \times 28$  sheet of neuron outputs:

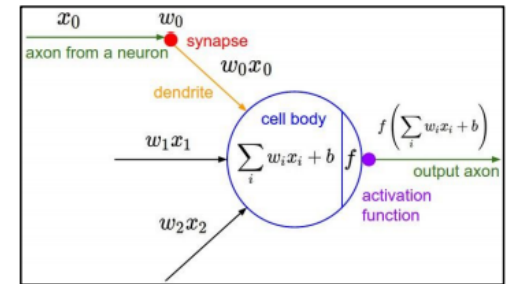
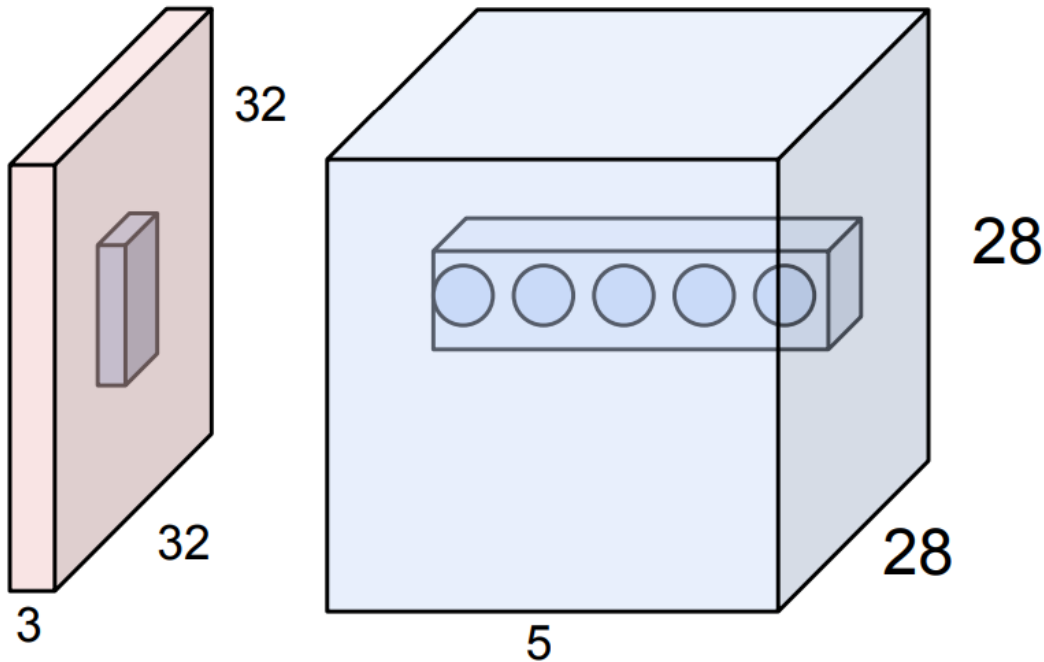
1. Each is connected to a small region in the input
2. All of them share parameters



It's just a neuron with local connectivity...

# Neural Networks

## The brain/neuron view of CONV Layer



E.g. with 5 filters,  
CONV layer consists of  
neurons arranged in a 3D grid  
(28x28x5)

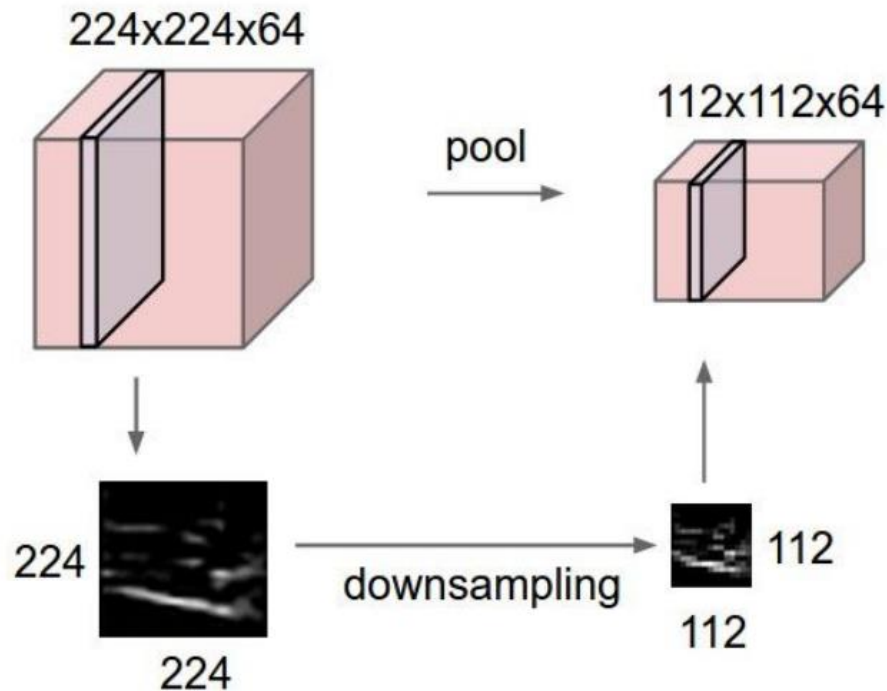
There will be 5 different  
neurons all looking at the same  
region in the input volume



# Neural Networks

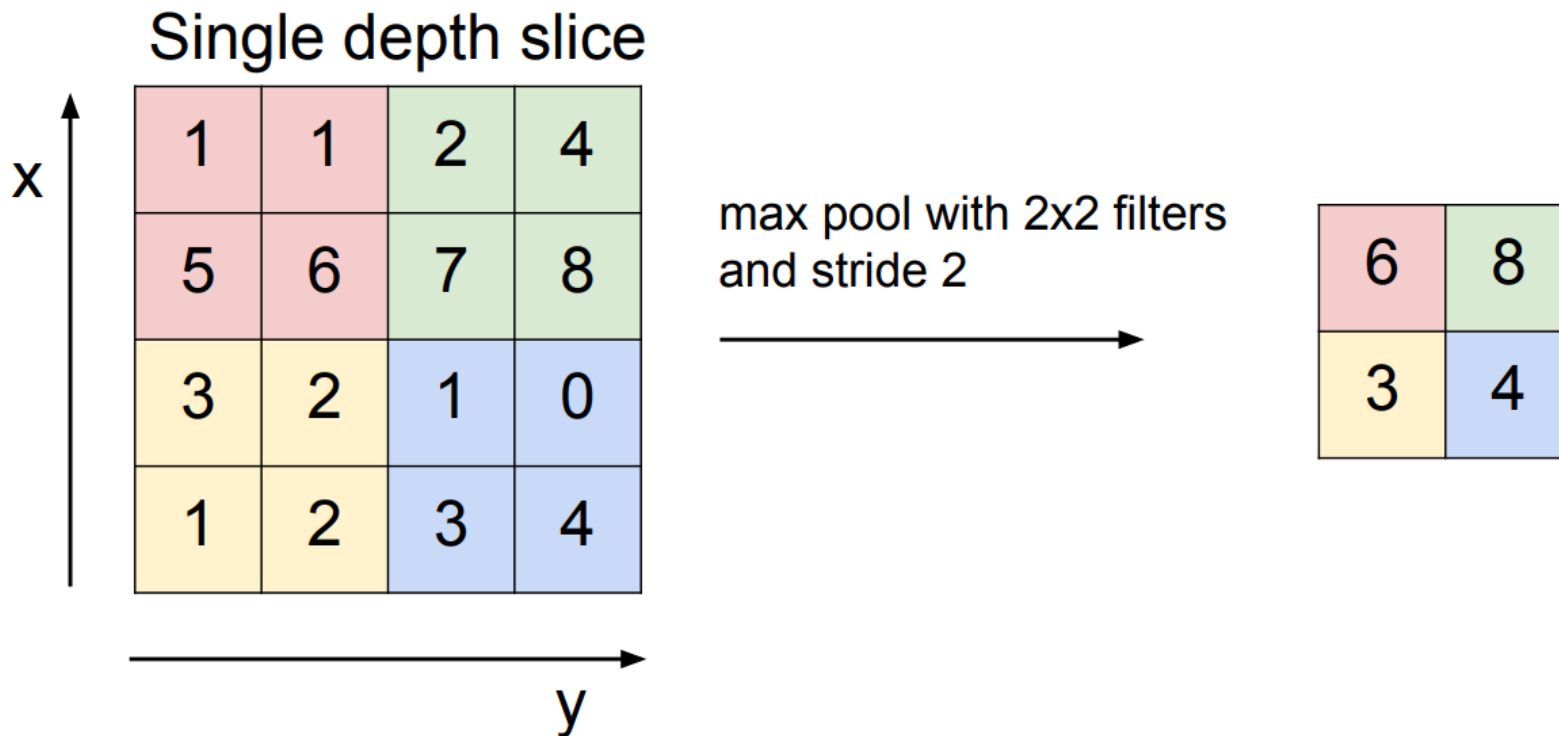
## Pooling layer

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



# Neural Networks

## MAX POOLING



# Neural Networks

## ❖ 컨볼루션 신경망의 구조

- 특징 추출을 위한 컨볼루션 부분
  - 컨볼루션 연산을 하는 **Conv**층
  - ReLU 연산을 하는 **ReLU**
  - 풀링 연산 **Pool**(선택)]
- 추출된 특징을 사용하여 분류 또는 회귀를 수행하는 다층 퍼셉트론 부분
  - 전방향으로 전체 연결된(fully connected) **FC**층 반복
  - 분류의 경우 마지막 층에 소프트맥스(softmax)을 하는 SM 연산 추가
    - 소프트맥스 연산 : 출력의 값이 0이상하면서 합은 1로 만들
- 컨볼루션 신경망 구조의 예
  - Conv-**ReLU**-**Pool**-Conv-**ReLU**-**Pool**-Conv-**ReLU**-**Pool**-**FC**-**SM**
  - Conv-**Pool**-Conv-**Pool**-Conv-**FC**-**FC**-**SM**
  - Conv-**Pool**-Conv-**Pool**-Conv-Conv-Conv-**Pool**-**FC**-**FC**-**SM**
  - Conv-**ReLU**-**Pool**-Conv-**ReLU**-**Pool**-Conv-**ReLU**-**Pool**-**FC**-**FC**-**SM**

} 반복

# Neural Networks

## ❖ 컨볼루션 신경망의 구조 예

- Conv:1-Pool:1-Conv:2-Pool:2-Conv:3-Conv:4-Conv:5-Pool:4-FC:6-FC:7-FC:8

