

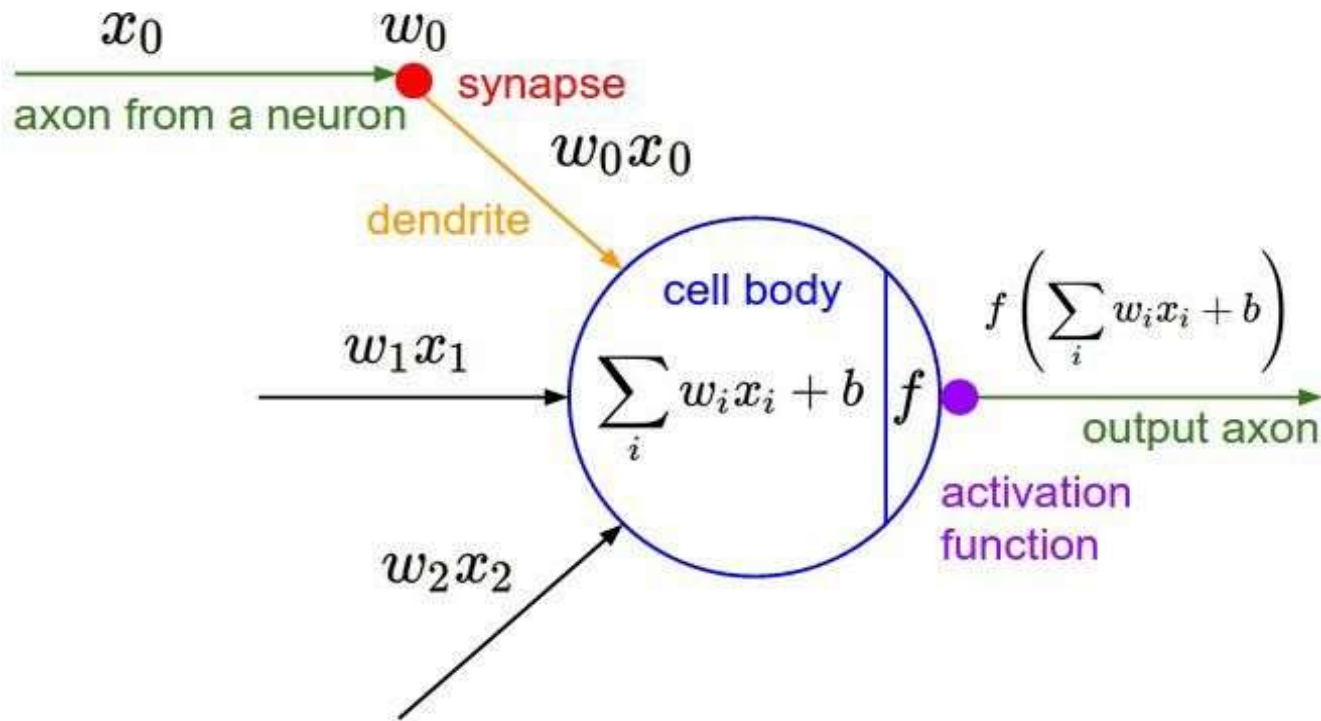
# Machine Learning

---

CNN

Dr. Shuang LIANG

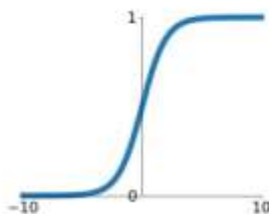
# Recall: Neuron



# Recall: Activation function

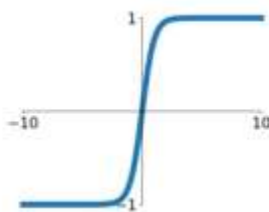
## Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



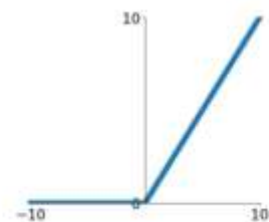
## tanh

$$\tanh(x)$$



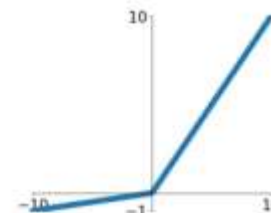
## ReLU

$$\max(0, x)$$



## Leaky ReLU

$$\max(0.1x, x)$$

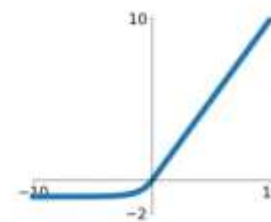


## Maxout

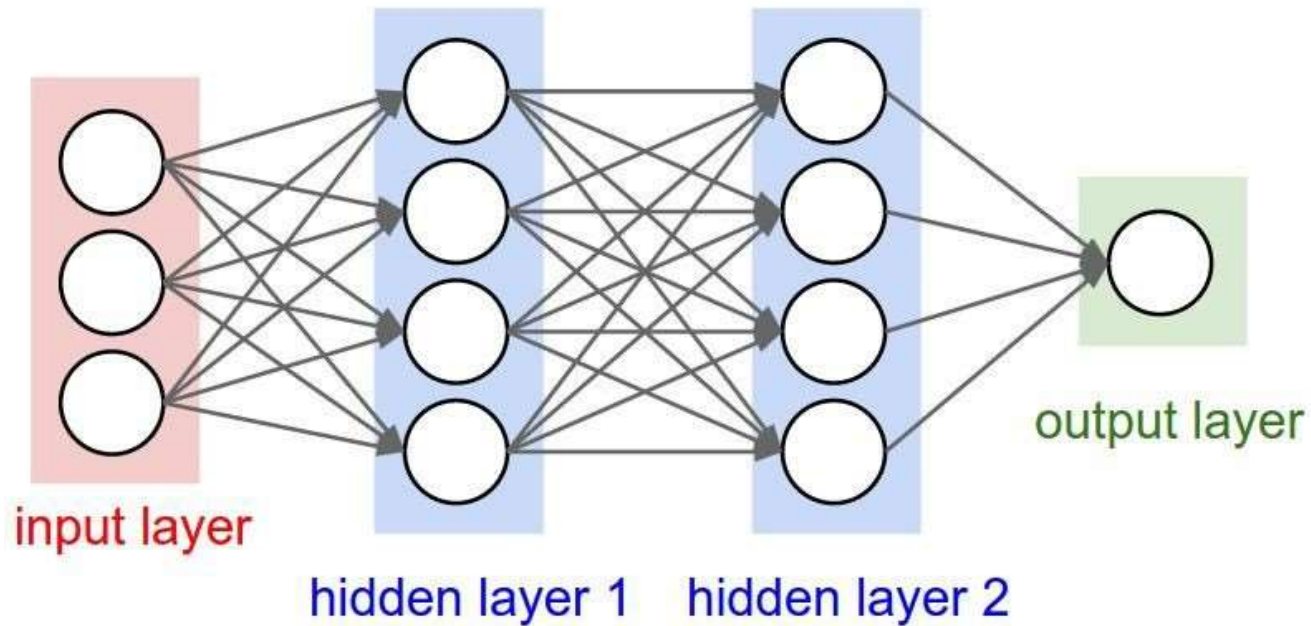
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

## ELU

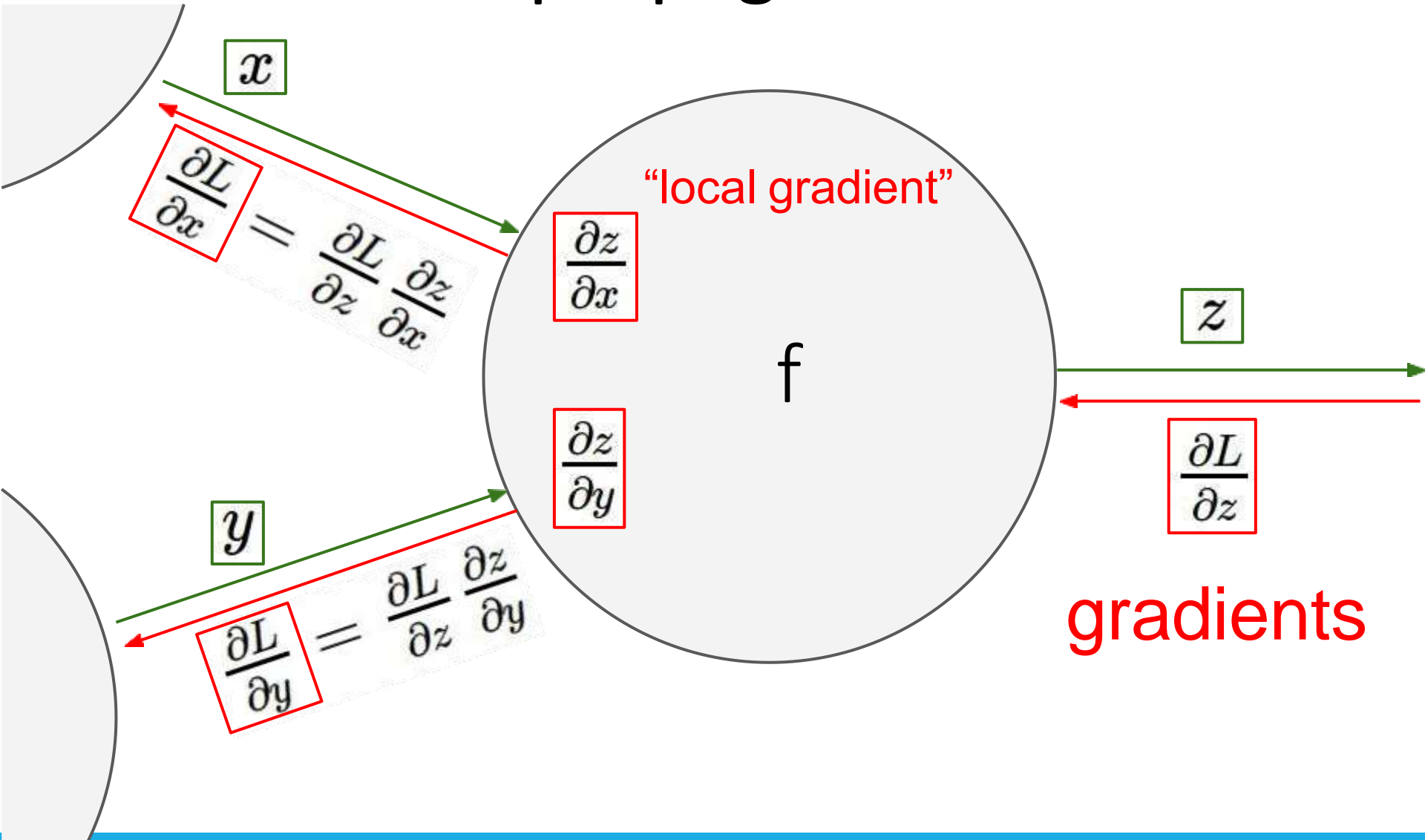
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



# Recall: Neural Network



# Recall: Backpropagation



# Today's Topics

- CNN Development
- CNN Layer
- CNN Architecture

# Today's Topics

- *CNN Development*
- CNN Layer
- CNN Architecture

# A bit of history

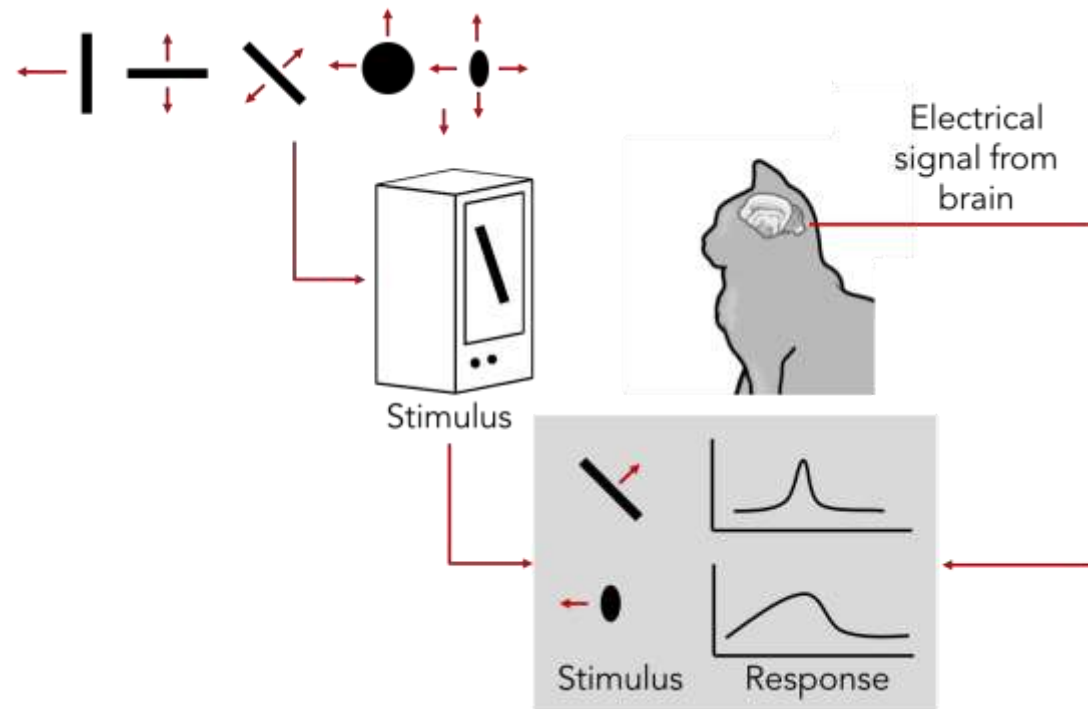
## Hubel & Wiesel, 1959

Receptive fields of single  
neurons in the cat's striate  
cortex

1962

Receptive fields, binocular  
interaction and functional  
architecture in the cat's visual  
cortex

1968...

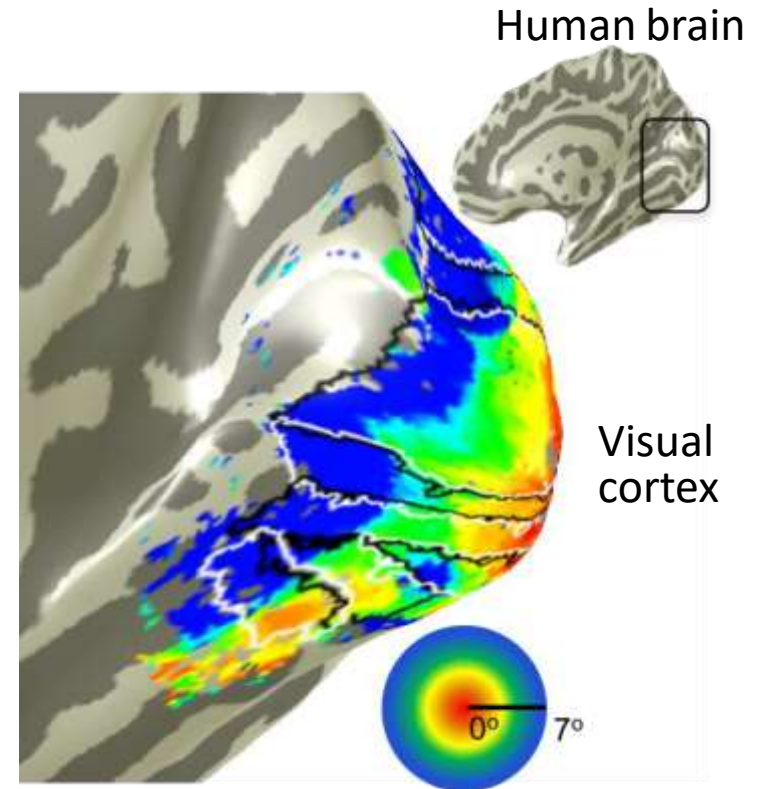
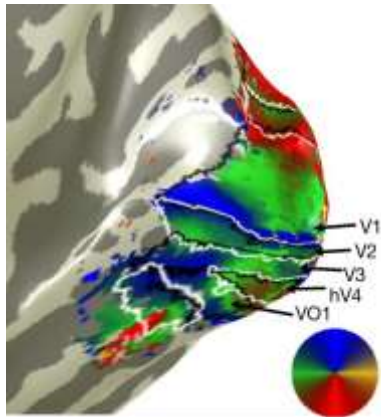


[Cat image](#) by CNX OpenStax is licensed  
under CC BY 4.0; changes made



# A bit of history

**Topographical mapping in the cortex:**  
nearby cells in cortex represent  
nearby regions in the visual field



Retinotopy images courtesy of Jesse Gomez in the  
Stanford Vision & Perception Neuroscience Lab.

# Hierarchical organization

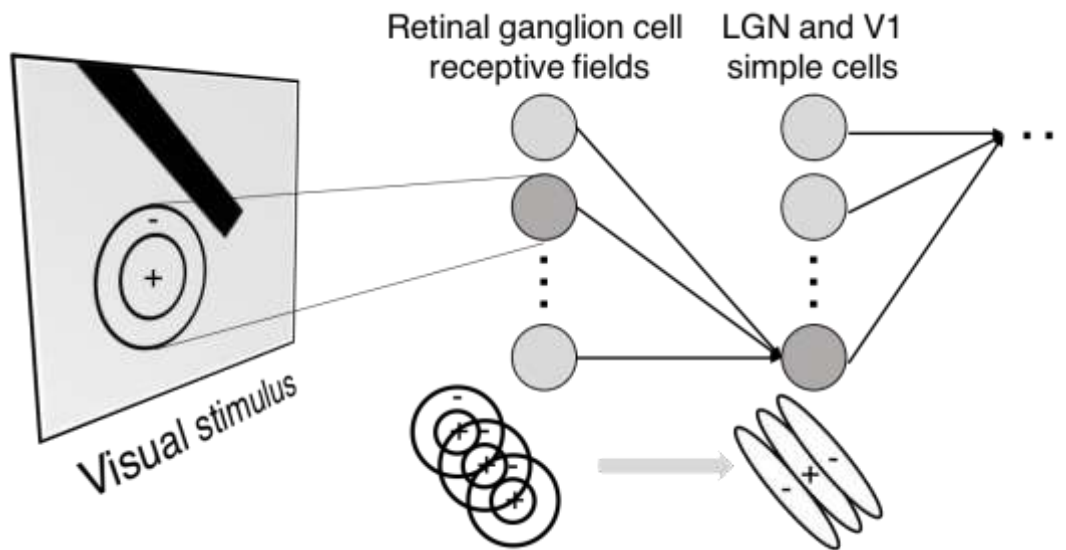


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

**Simple cells:**  
Response to light  
orientation

**Complex cells:**  
Response to light  
orientation and movement

**Hypercomplex cells:**  
response to movement  
with an end point



No response

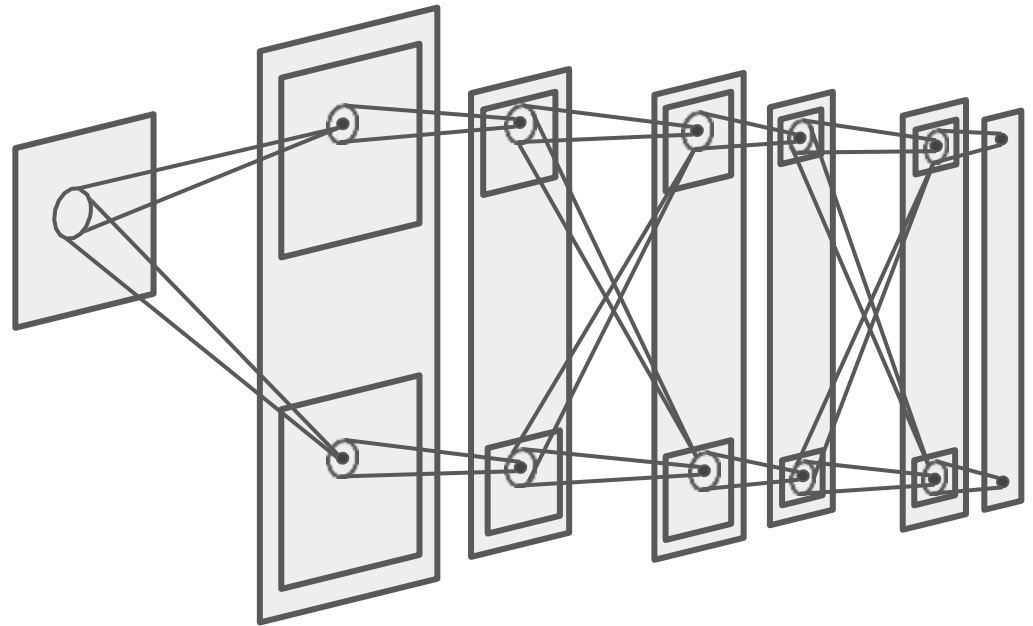


Response  
(end point)

# A bit of history

## Neocognitron *[Fukushima 1980]*

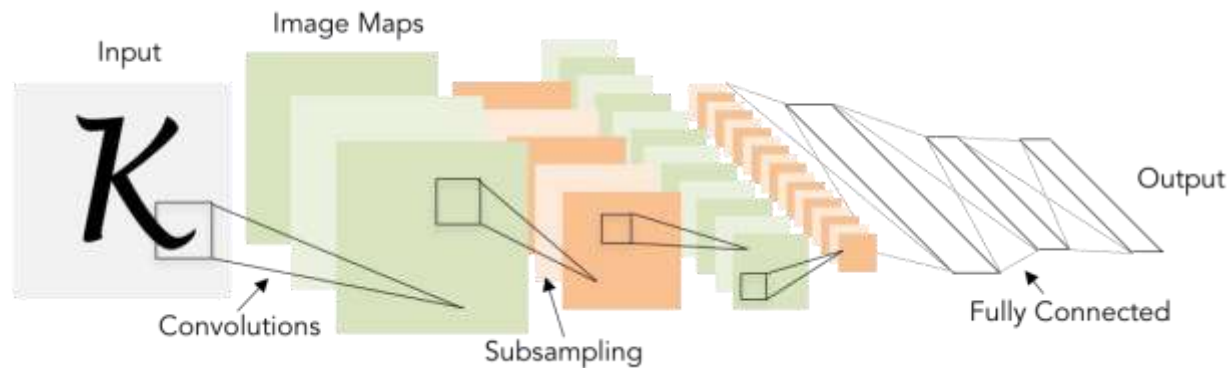
“sandwich” architecture (SCSCSC...)  
simple cells: modifiable parameters  
complex cells: perform pooling



# A bit of history

## Gradient-based learning applied to document recognition

*[LeCun, Bottou, Bengio, Haffner 1998]*



LeNet-5

# A bit of history

## ImageNet Classification with Deep Convolutional Neural Networks *[Krizhevsky, Sutskever, Hinton, 2012]*

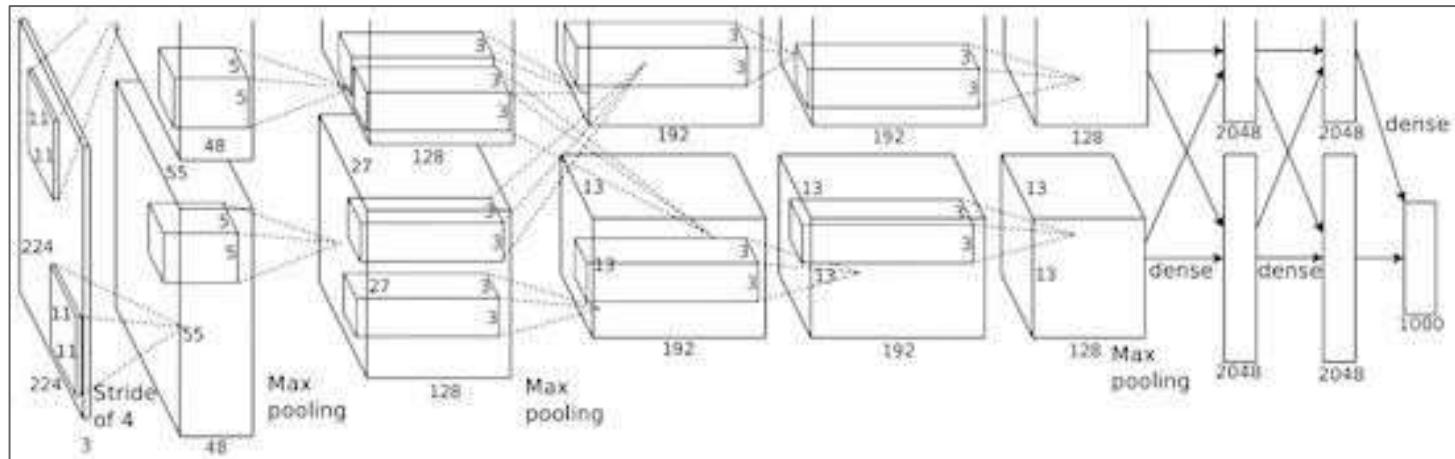


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”

# Fast-forward to today: ConvNets are everywhere

## Classification



## Retrieval

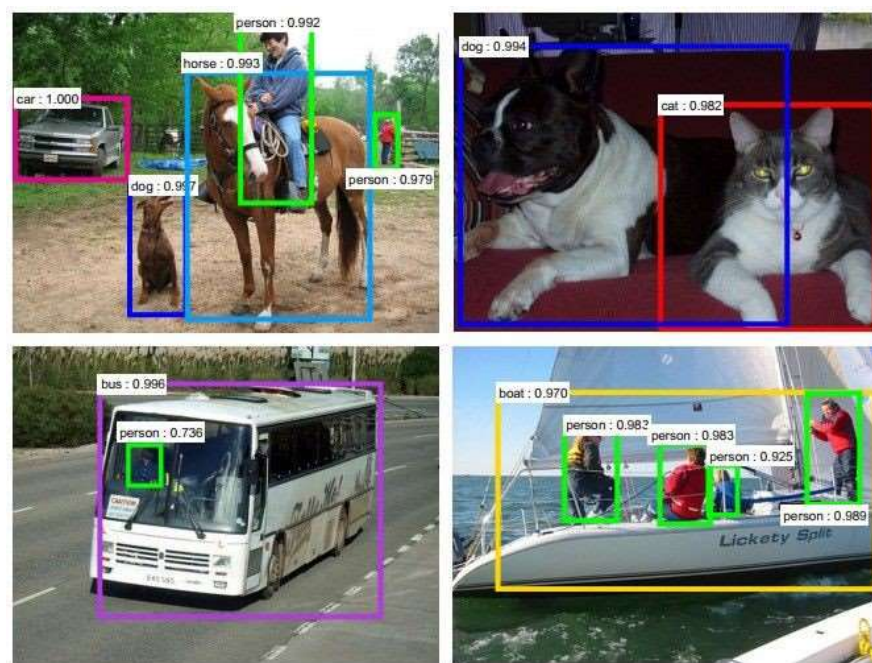


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012.  
Reproduced with permission.



# Fast-forward to today: ConvNets are everywhere

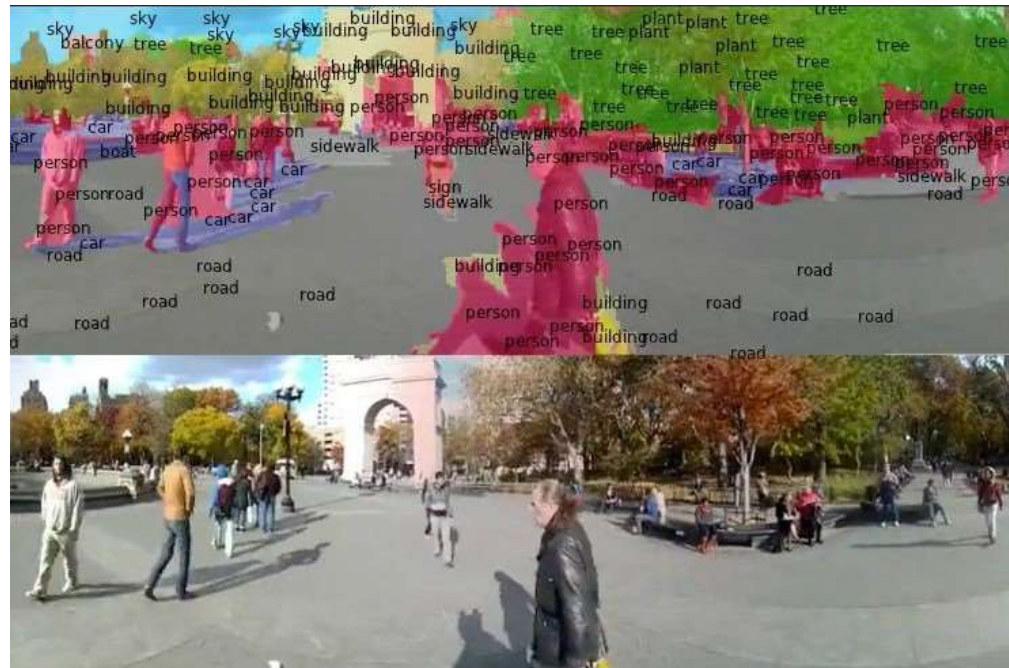
## Detection



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]**

# Fast-forward to today: ConvNets are everywhere



Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars



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

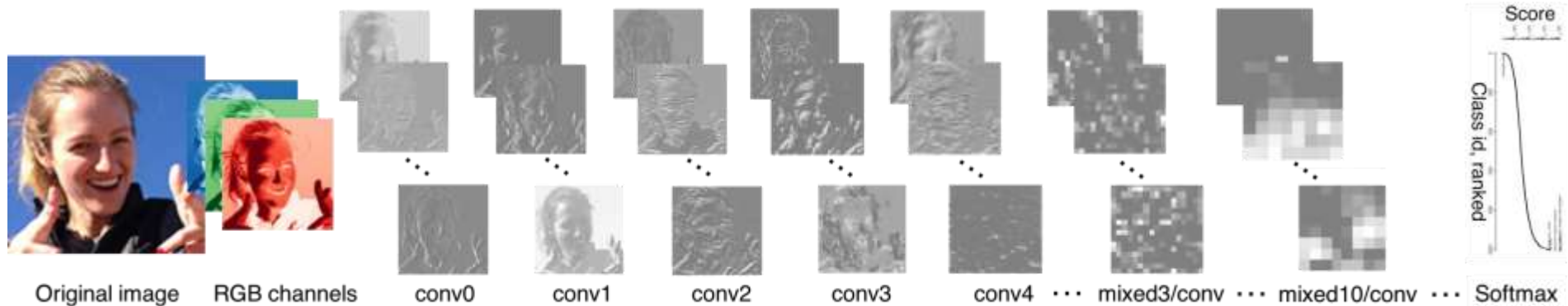
## NVIDIA Tesla line

(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.

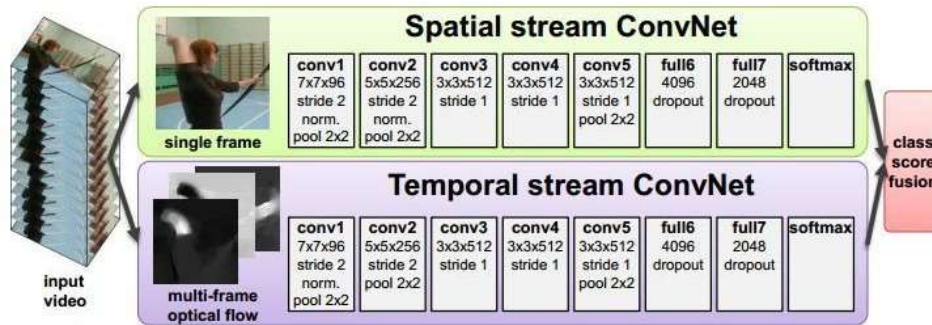


# Fast-forward to today: ConvNets are everywhere



[Taigman et al. 2014]

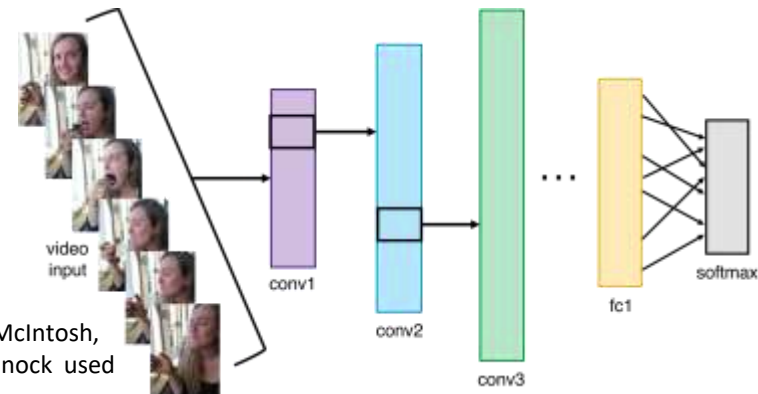
Activations of [inception-v3 architecture](#) [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

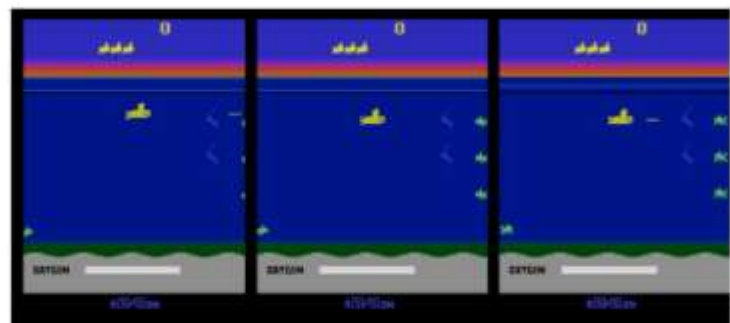
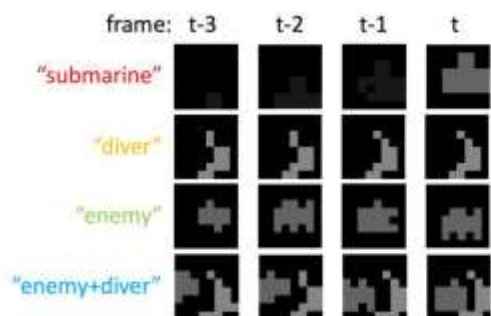


# Fast-forward to today: ConvNets are everywhere



**[Toshev, Szegedy 2014]**

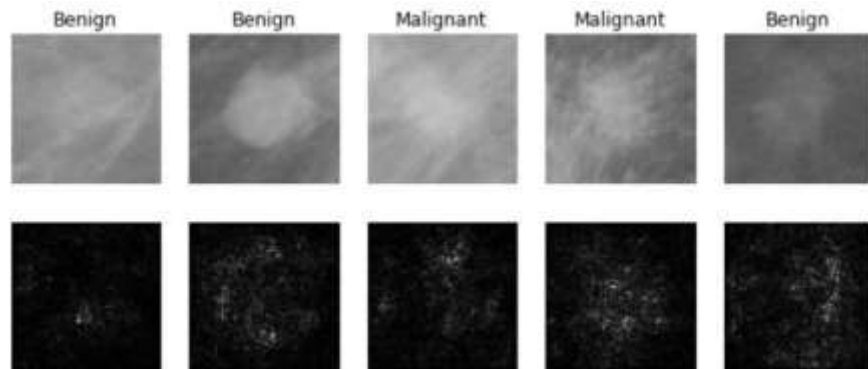
Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.



**[Guo et al. 2014]**

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

# Fast-forward to today: ConvNets are everywhere



**[Levy et al. 2016]**

Figure copyright Levy et al. 2016.  
Reproduced with permission.



**[Dieleman et al. 2014]**

From left to right:  
[public domain by NASA](#),  
usage [permitted](#) by ESA/Hubble,  
[public domain by NASA](#),  
and [public domain](#).



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

Photos by Lane McIntosh.  
Copyright CS231n 2017.

# Fast-forward to today: ConvNets are everywhere

[This image](#) by Christin Khan is in the public domain and originally came from the U.S. NOAA.



***Whale recognition, Kaggle Challenge***

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.



***Mnih and Hinton, 2010***



# Fast-forward to today: ConvNets are everywhere

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/>

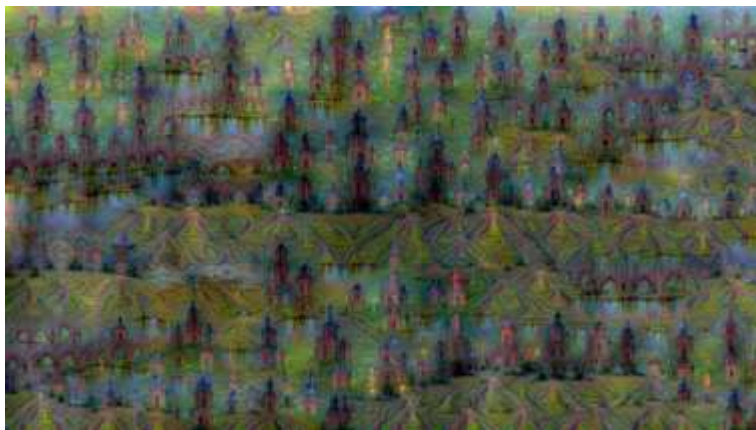
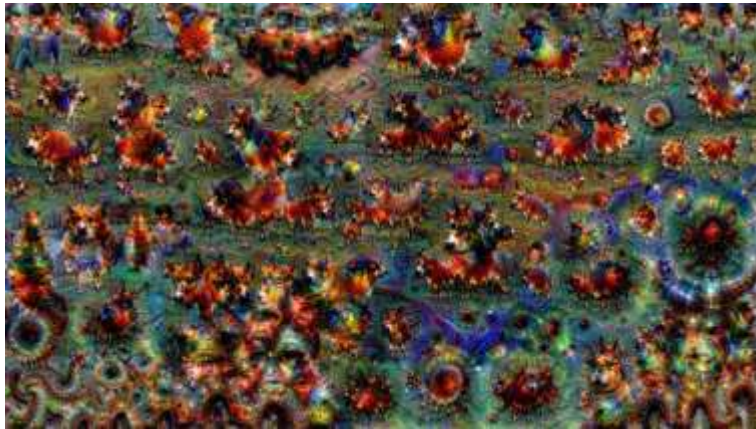
<https://pixabay.com/en/woman-female-model-portrait-adult-983967/>

<https://pixabay.com/en/handstand-lake-meditation-496008/>

<https://pixabay.com/en/baseball-player-shortstop-infield-1045263/>

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

# Fast-forward to today: ConvNets are everywhere



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a [blog post](#) by Google Research.



[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

# Today's Topics

- CNN Development
- *CNN Layer*
- CNN Architecture

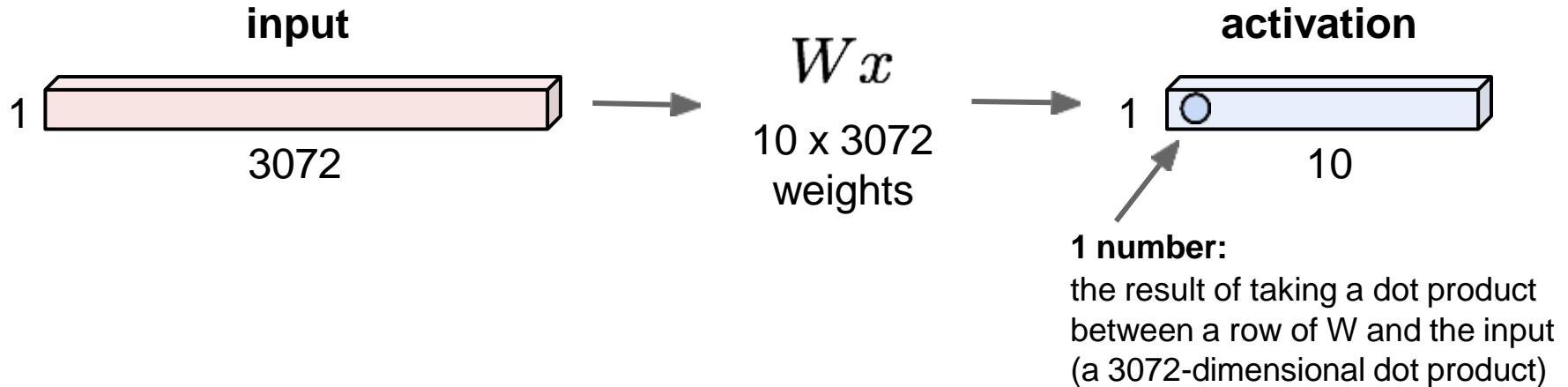
# CNN Layer

- *Fully Connected Layer*
- Convolution Layer
- Pooling Layer



# Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

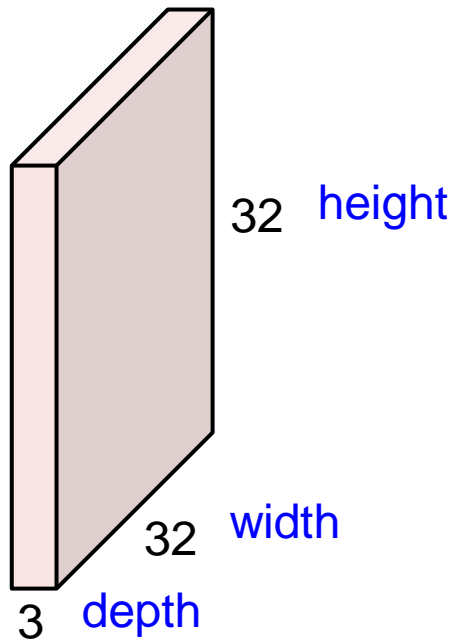


# CNN Layer

- Fully Connected Layer
- *Convolution Layer*
- Pooling Layer

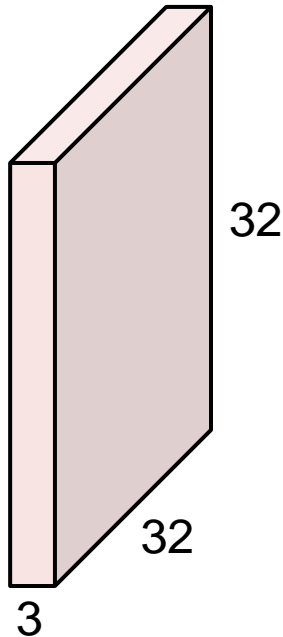
# Convolution Layer

32x32x3 image -> preserve spatial structure



# Convolution Layer

32x32x3 image

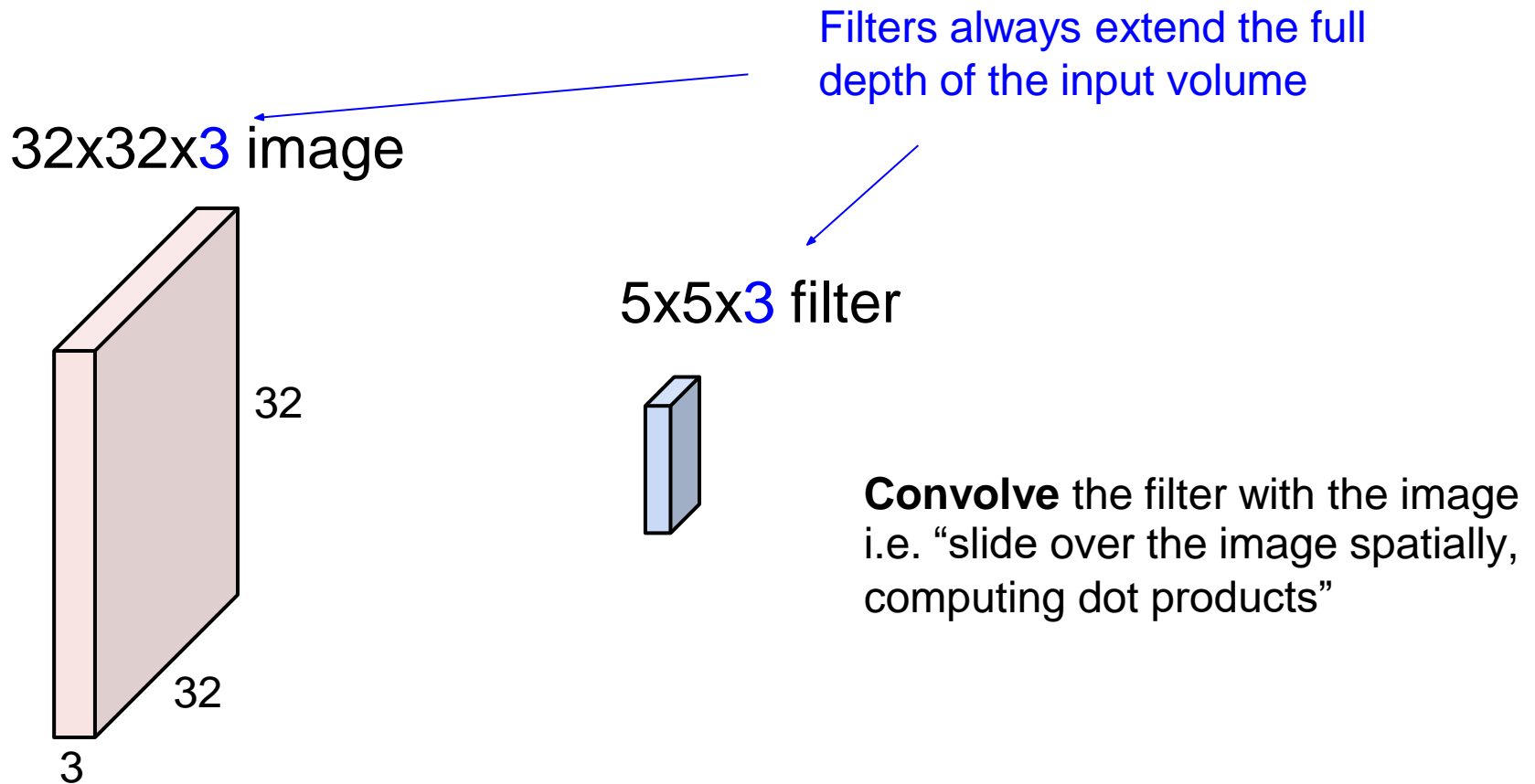


5x5x3 filter

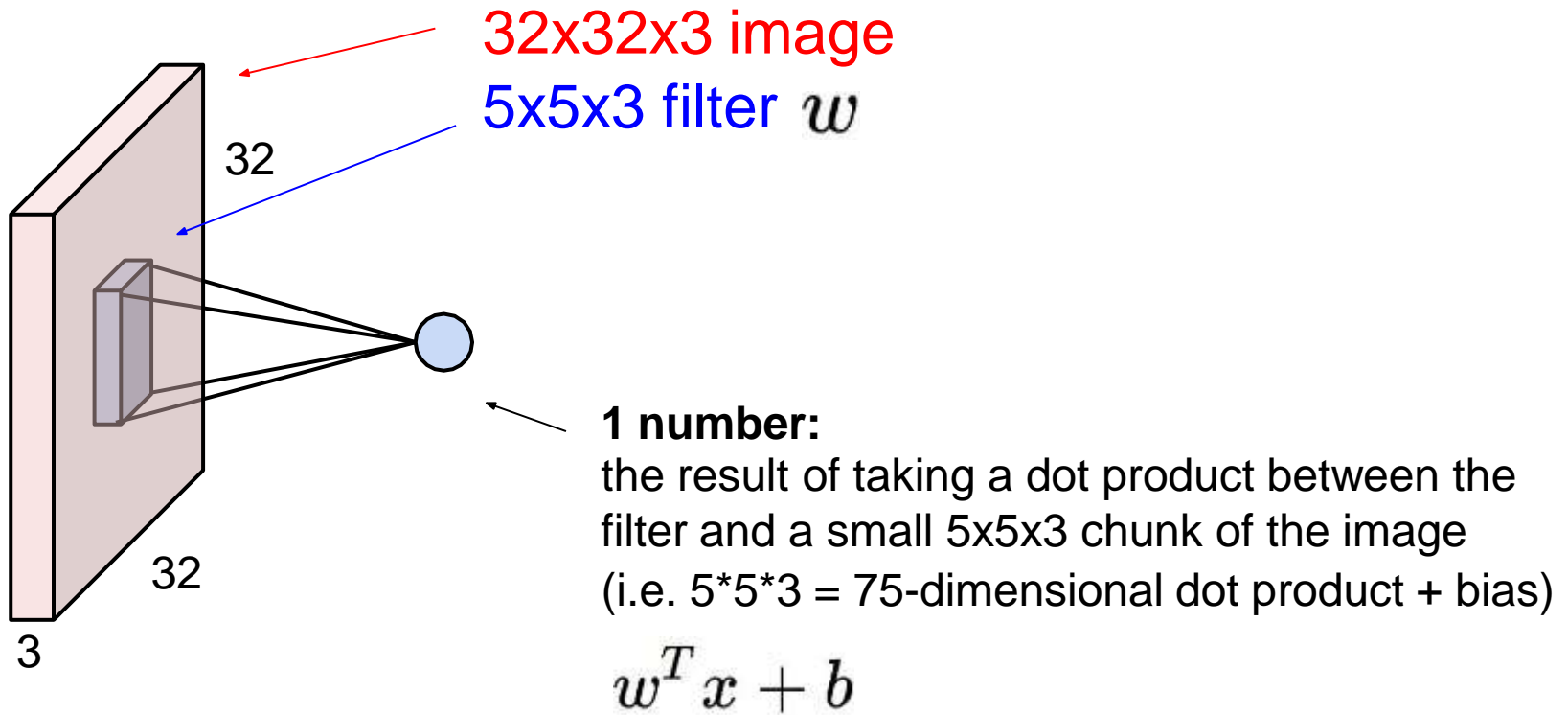


**Convolve** the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”

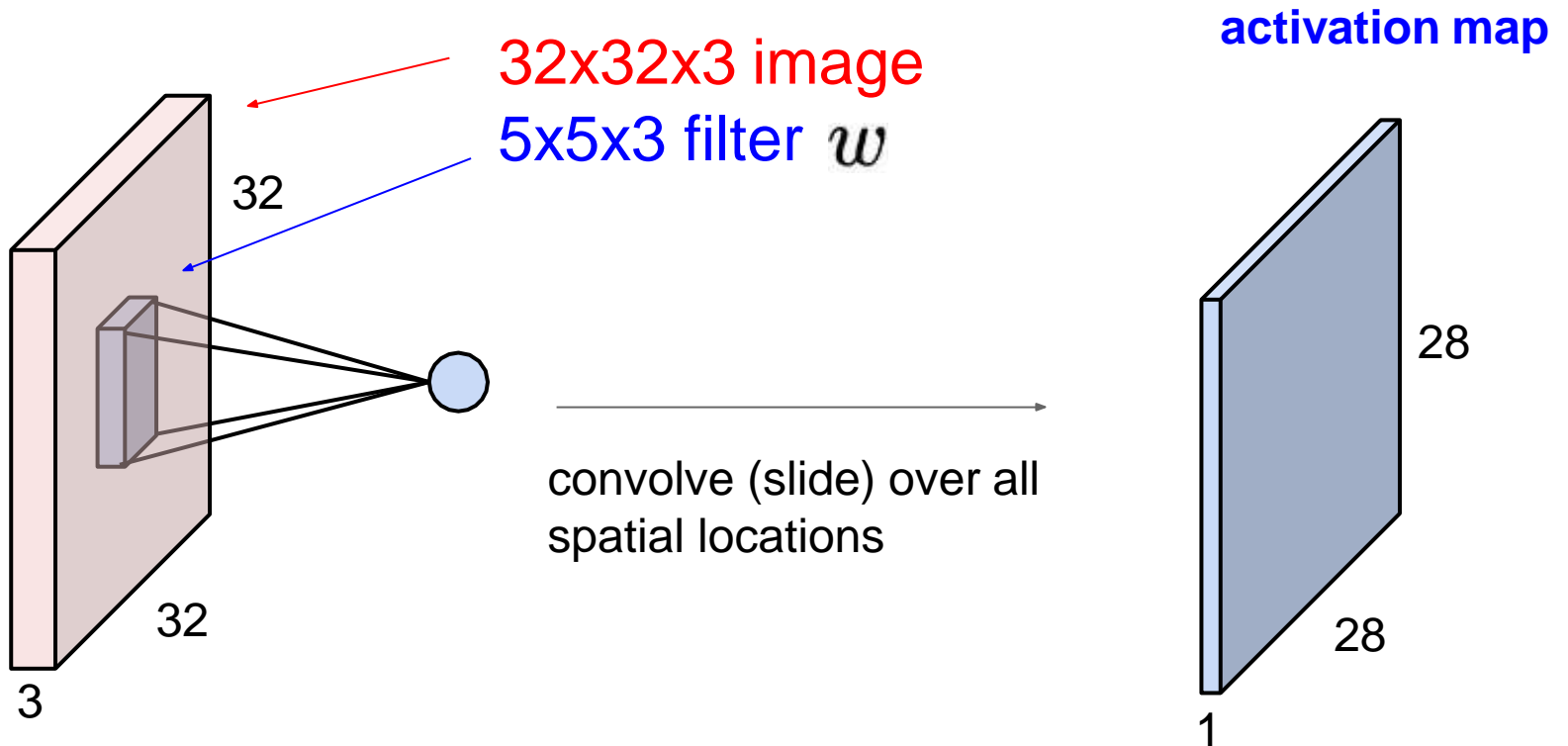
# Convolution Layer



# Convolution Layer

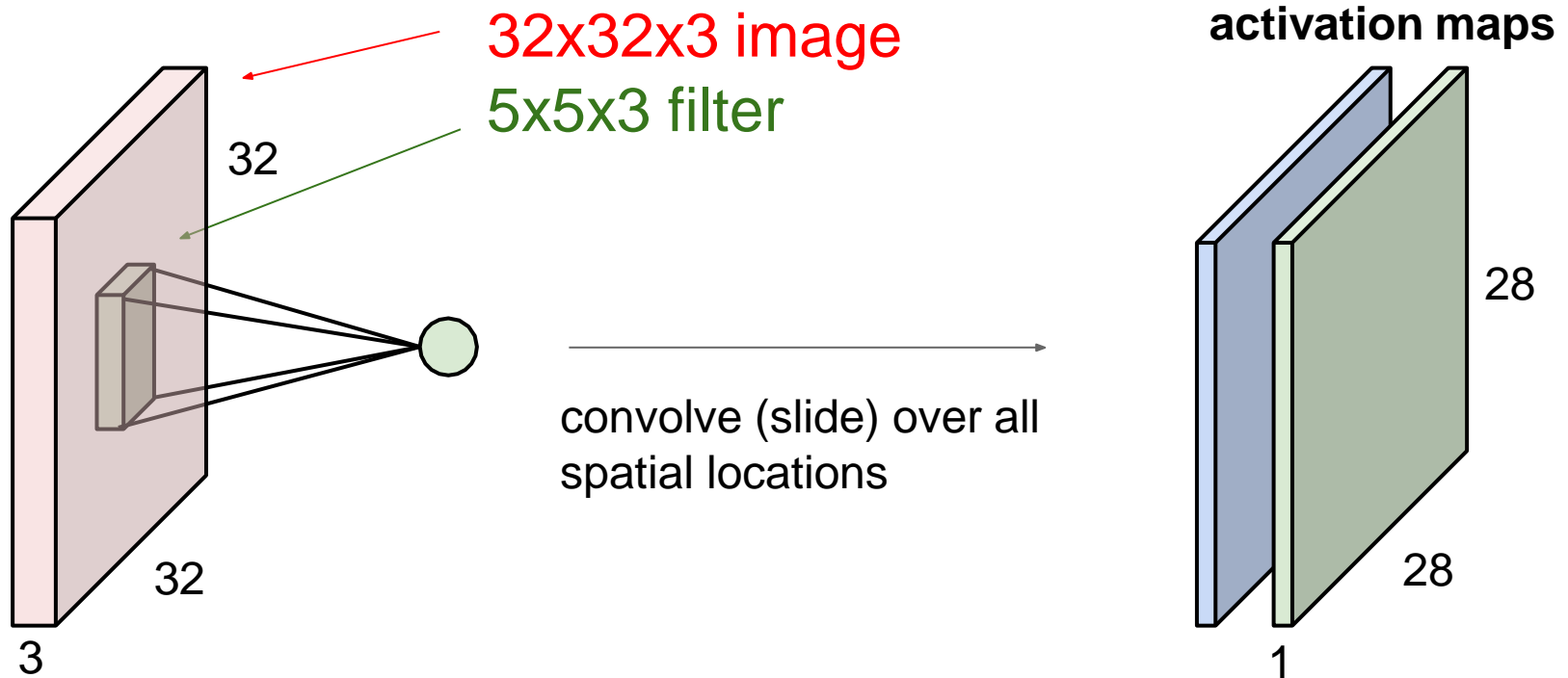


# Convolution Layer



# Convolution Layer

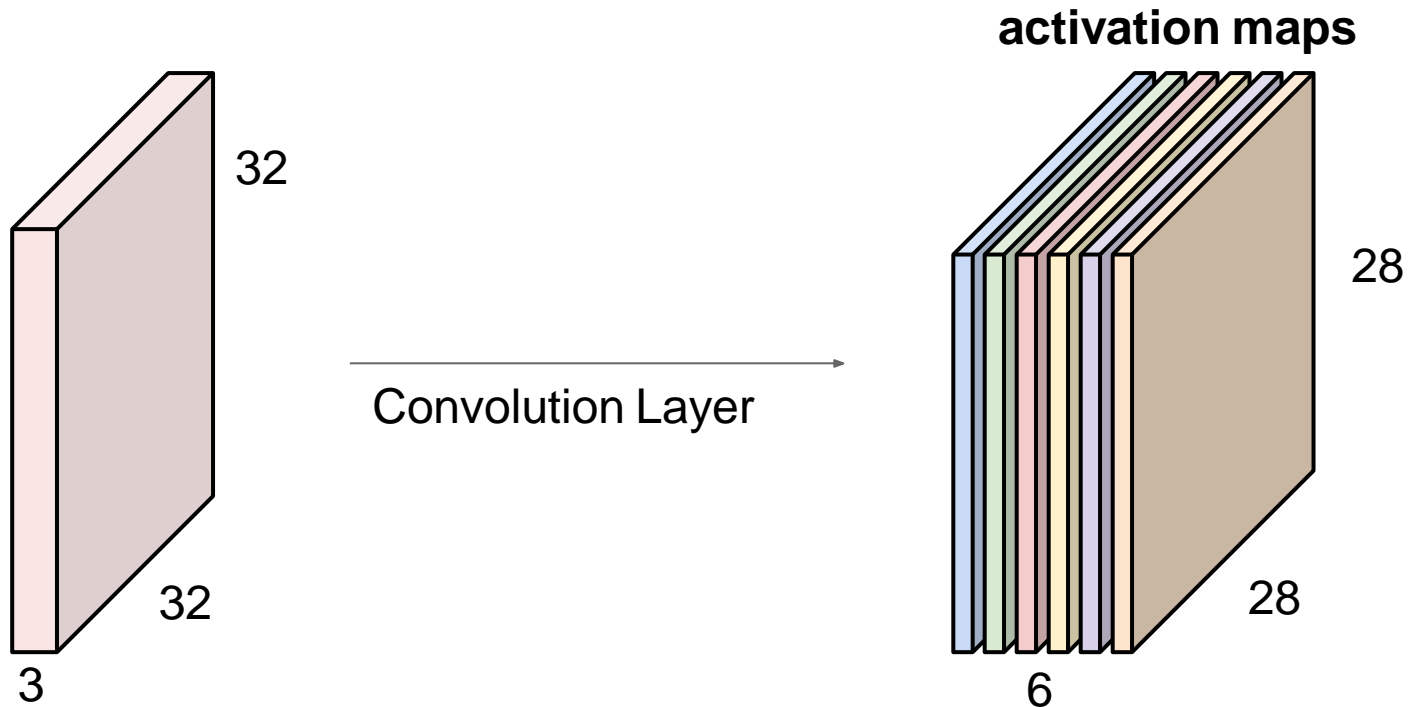
consider a second, **green** filter





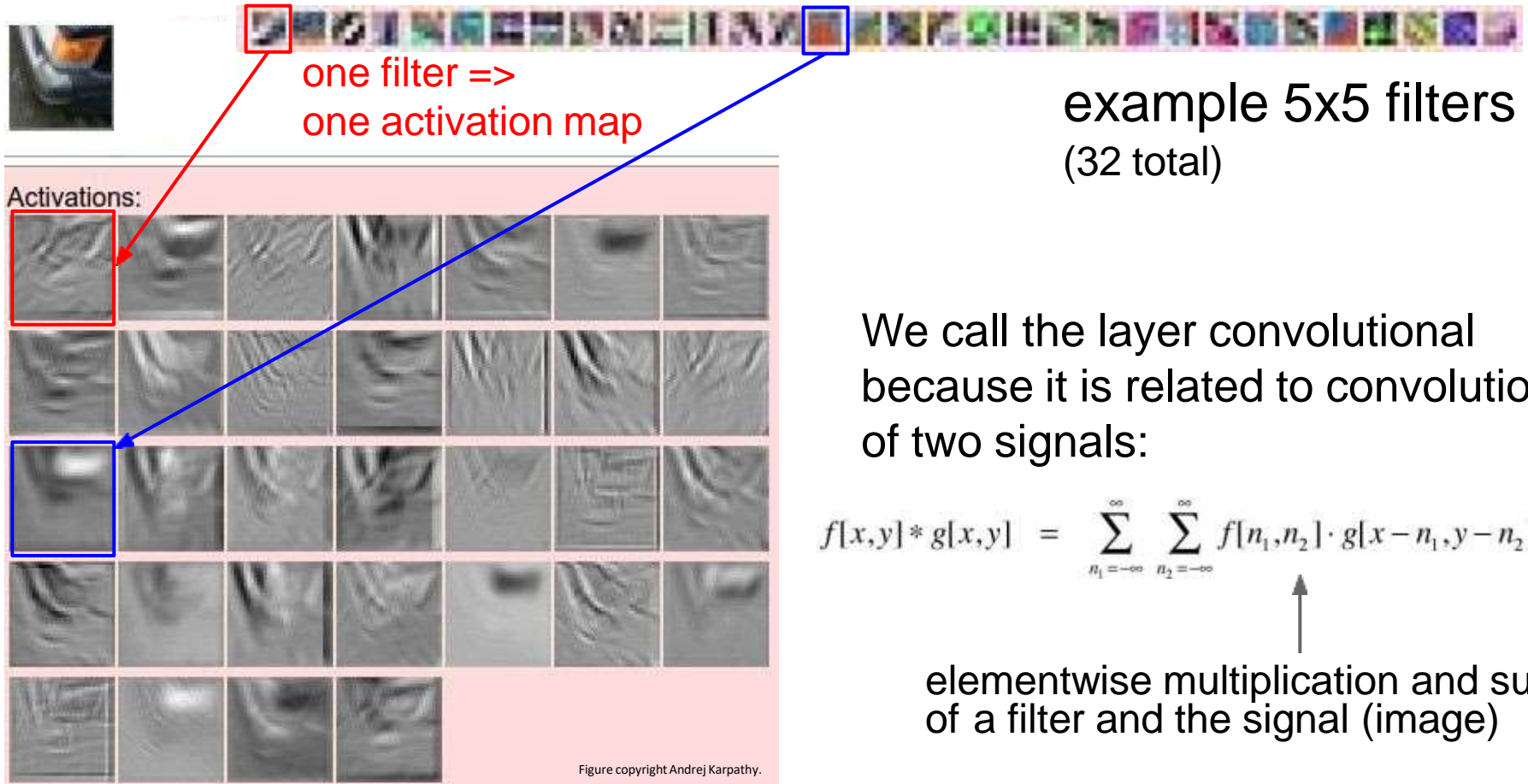
# Convolution Layer

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!

# Convolution Layer



# Convolution

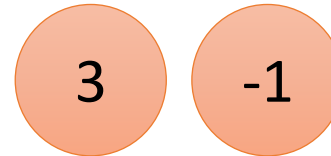
stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1



# Convolution

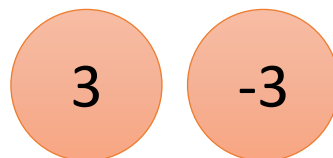
If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1



We set stride=1 below

# Convolution

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

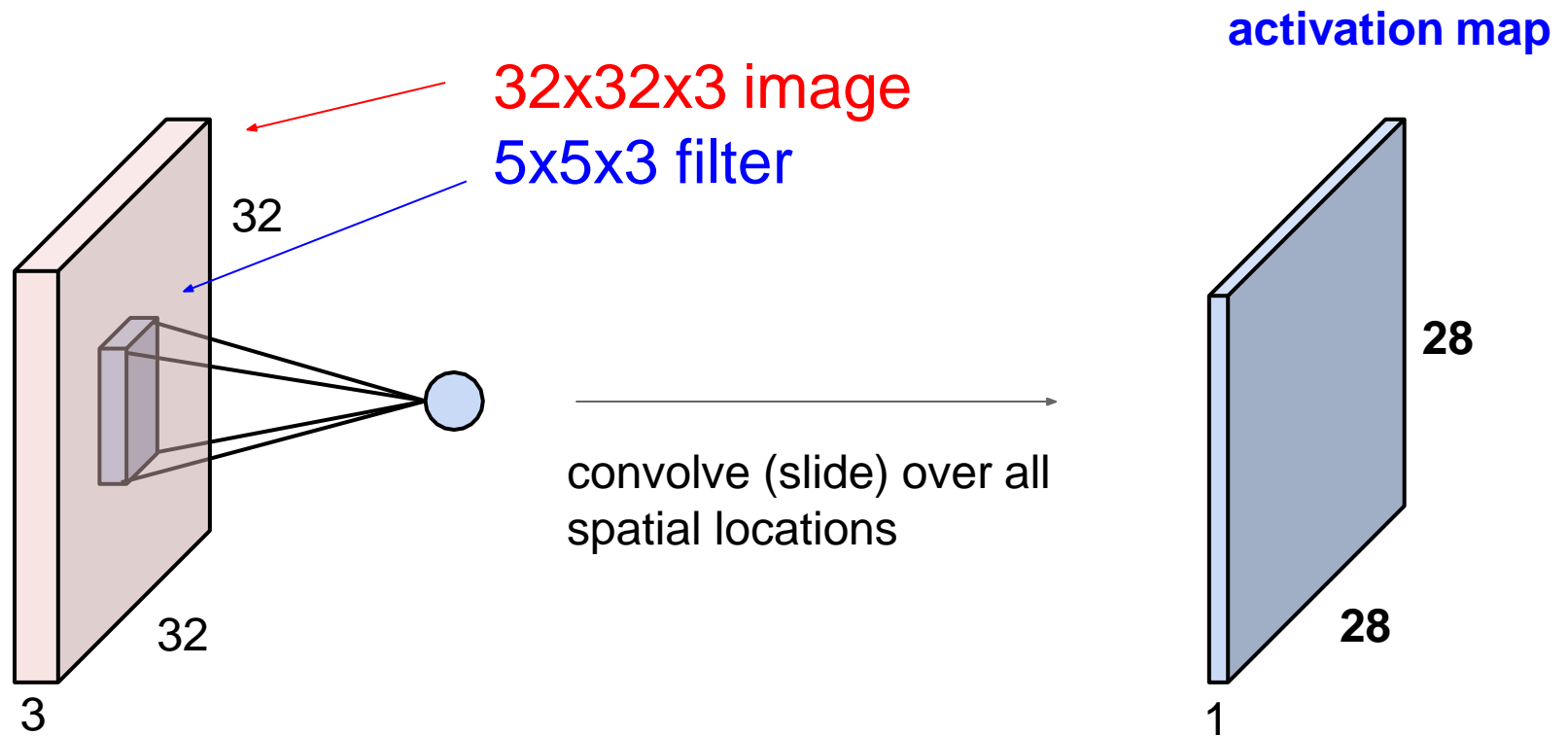
6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

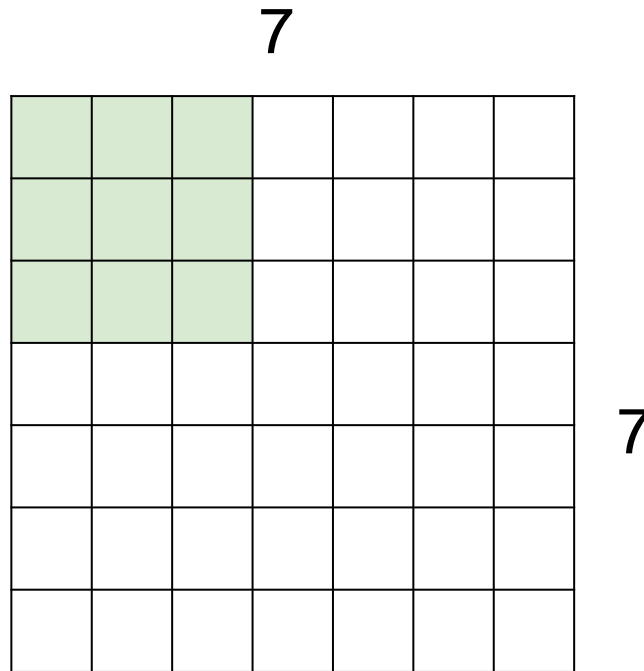
Filter 1

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

# A closer look at spatial dimensions

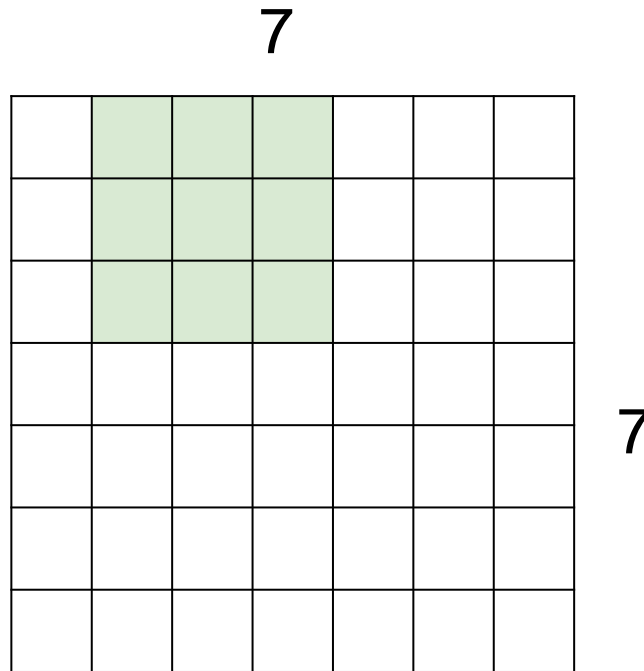


# A closer look at spatial dimensions



7x7 input (spatially)  
assume 3x3 filter

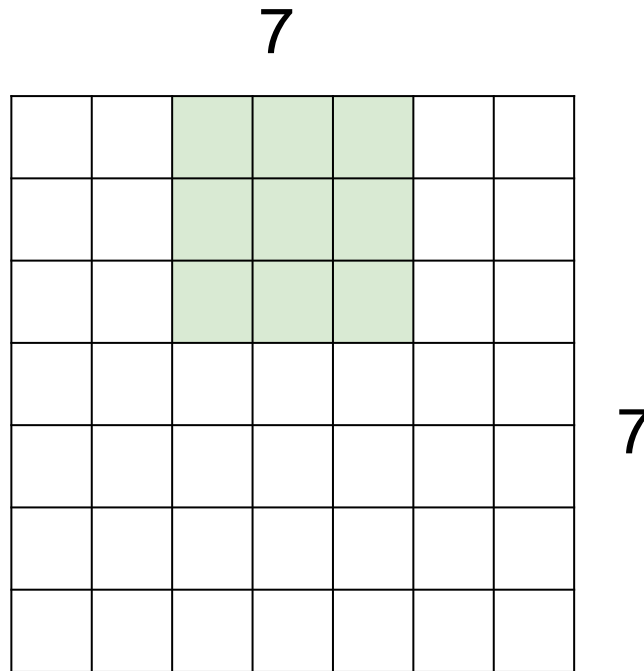
# A closer look at spatial dimensions



7x7 input (spatially)  
assume 3x3 filter

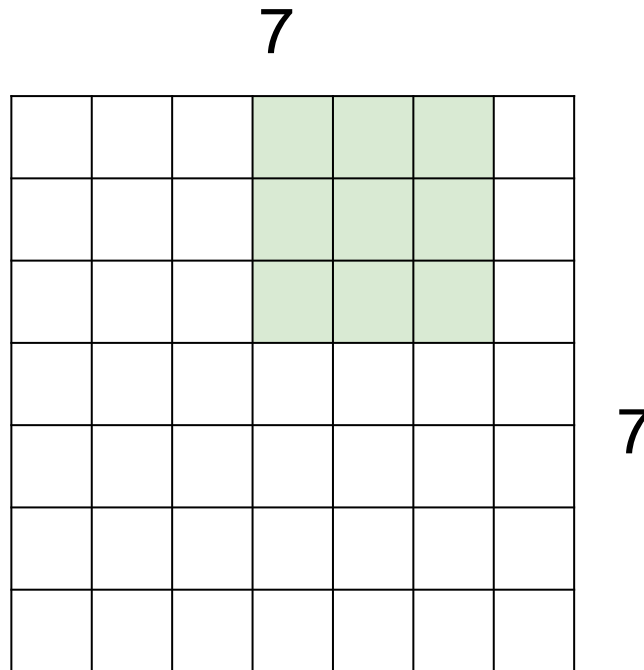


# A closer look at spatial dimensions



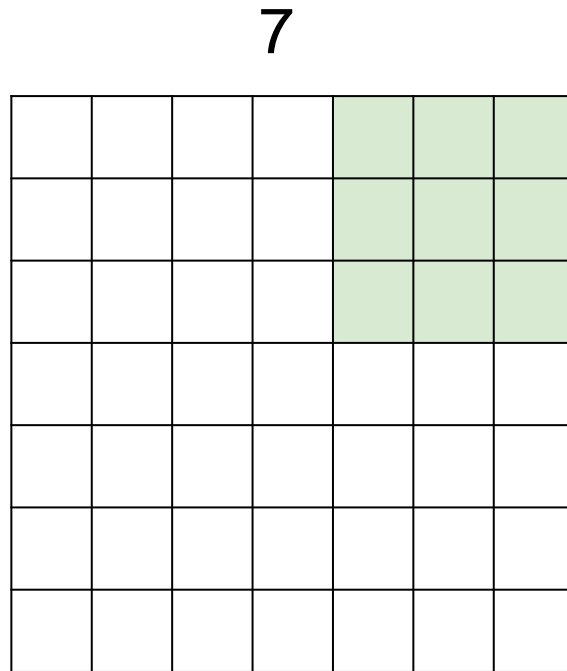
7x7 input (spatially)  
assume 3x3 filter

# A closer look at spatial dimensions



7x7 input (spatially)  
assume 3x3 filter

# A closer look at spatial dimensions

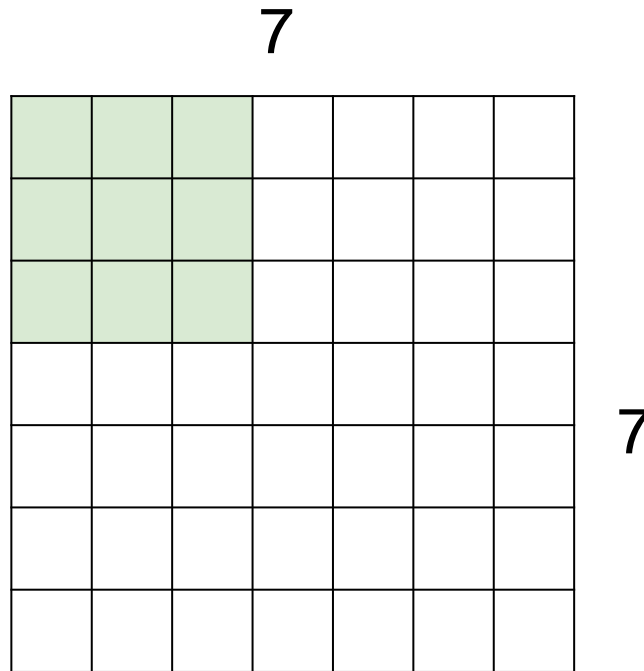


7x7 input (spatially)  
assume 3x3 filter

7

**=> 5x5 output**

# A closer look at spatial dimensions

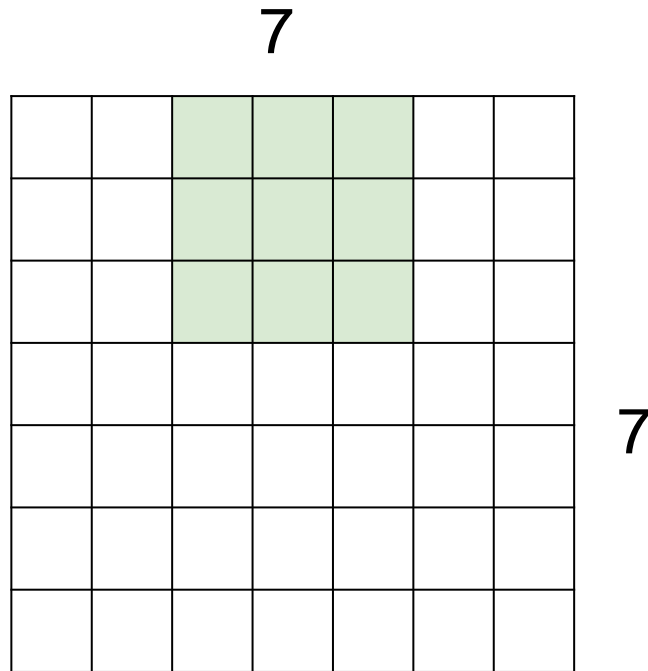


7x7 input (spatially)

assume 3x3 filter

Applied **with stride 2**

# A closer look at spatial dimensions

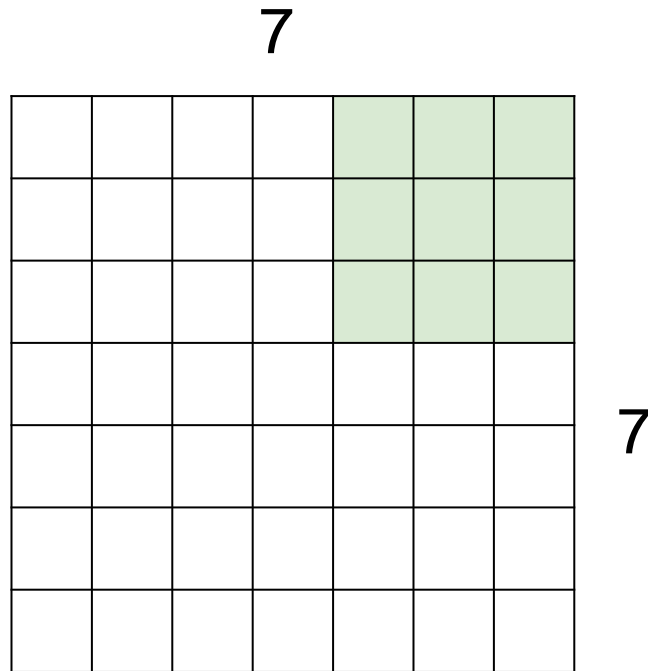


7x7 input (spatially)

assume 3x3 filter

Applied **with stride 2**

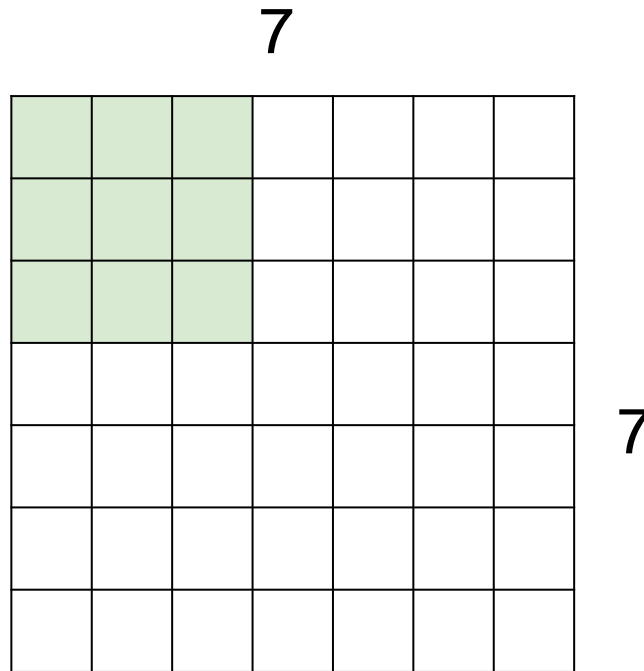
# A closer look at spatial dimensions



7x7 input (spatially)  
assume 3x3 filter

Applied **with stride 2**  
**=> 3x3 output!**

# A closer look at spatial dimensions



7x7 input (spatially)

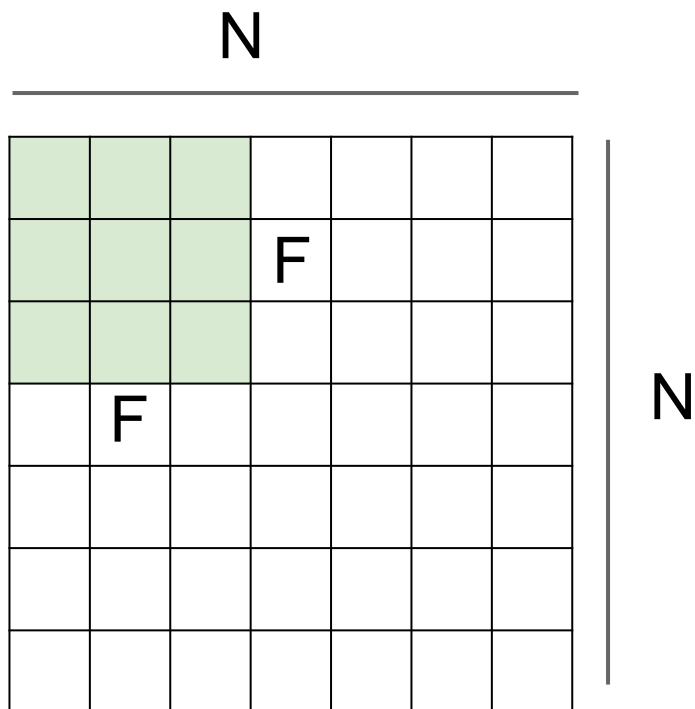
assume 3x3 filter

Applied **with stride 3?**

**doesn't fit!**

cannot apply 3x3 filter on  
7x7 input with stride 3.

# Convolution Layer



Output size:

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

e.g.  $N = 7$ ,  $F = 3$ :

$$\text{stride } 1 \Rightarrow (7 - 3) / 1 + 1 = 5$$

$$\text{stride } 2 \Rightarrow (7 - 3) / 2 + 1 = 3$$

$$\text{stride } 3 \Rightarrow (7 - 3) / 3 + 1 = 2.33 \therefore \backslash$$



# 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?

(recall:)

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

# 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 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)

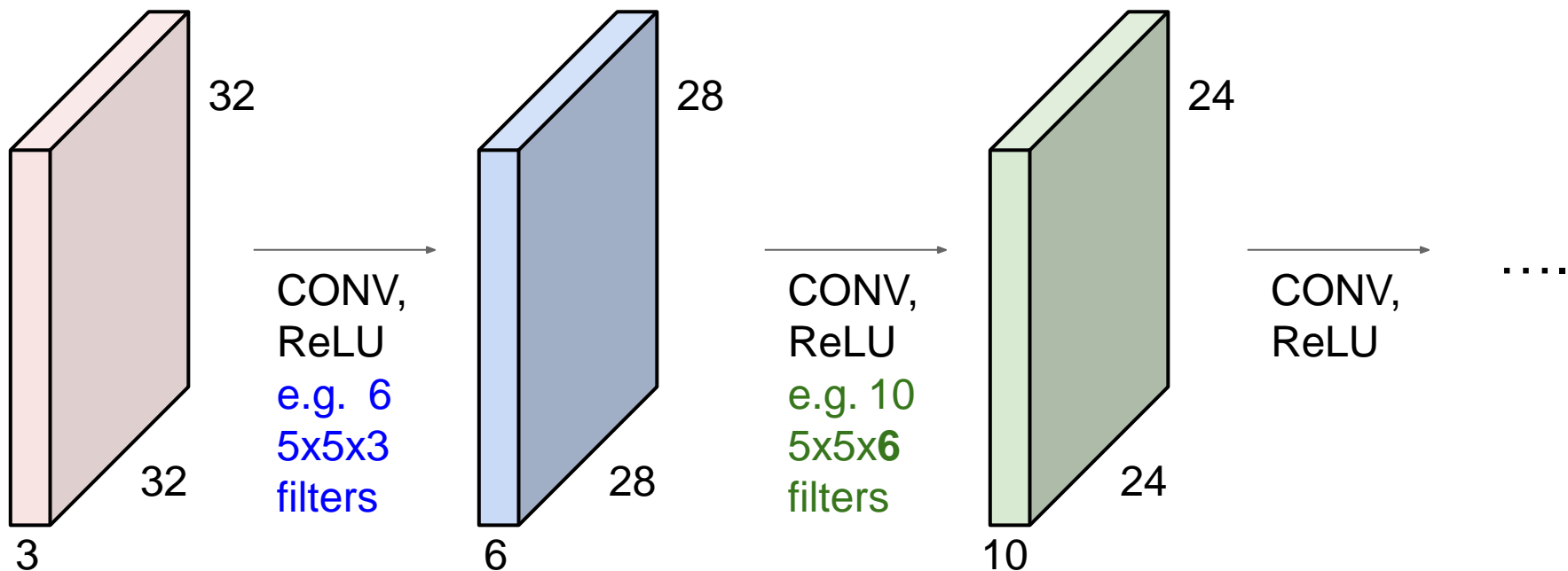
e.g.  $F = 3 \Rightarrow$  zero pad with 1

$F = 5 \Rightarrow$  zero pad with 2

$F = 7 \Rightarrow$  zero pad with 3

# Convolution Layer: Example

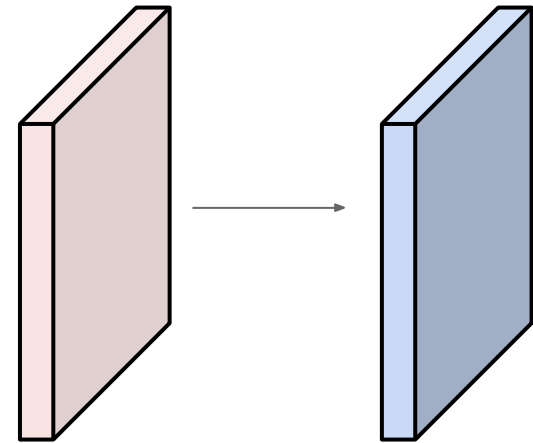
32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32  $\rightarrow$  28  $\rightarrow$  24 ...). Shrinking too fast is not good, doesn't work well.



# Convolution Layer: Example

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

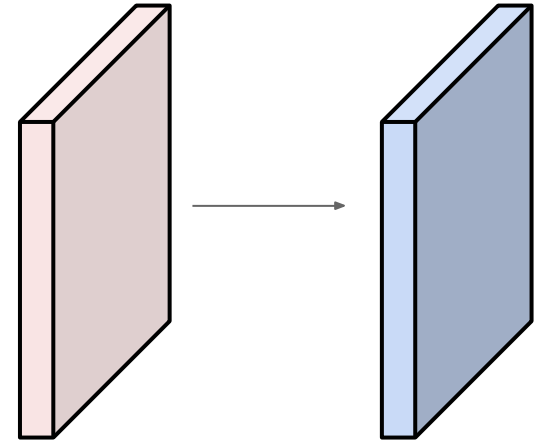
Output volume size:?



# Convolution Layer: Example

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

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

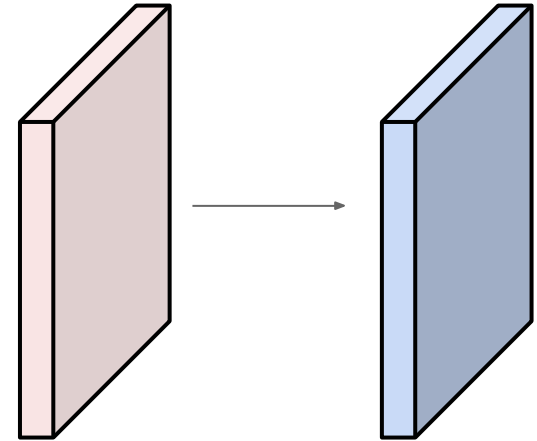


# Convolution Layer: Example

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

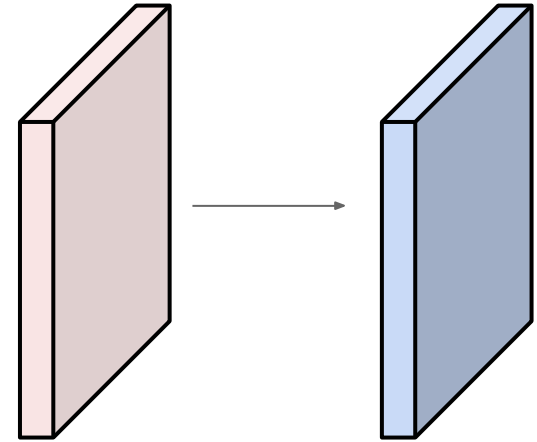
Number of parameters in this layer?



# Convolution Layer: Example

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  
 $\Rightarrow 76*10 = 760$  (+1 for bias)





# Convolution Layer: Summary

- Accepts a volume of size  $W_1 \times H_1 \times D_1$

- Requires 4 hyperparameters:

- Numbers of filters  $K$
- Their spatial extent  $F$
- The stride  $S$
- The amount of zero padding  $P$

## Common settings:

$K$  = (powers of 2, e.g. 32, 64, 128, 512)

- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ?$  (whatever fits)
- $F = 1, S = 1, P = 0$

- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:

- $W_2 = \frac{W_1 - F + 2P}{S} + 1$

- $H_2 = \frac{H_1 - F + 2P}{S} + 1$  (i.e. width and height are computed equally by symmetry)

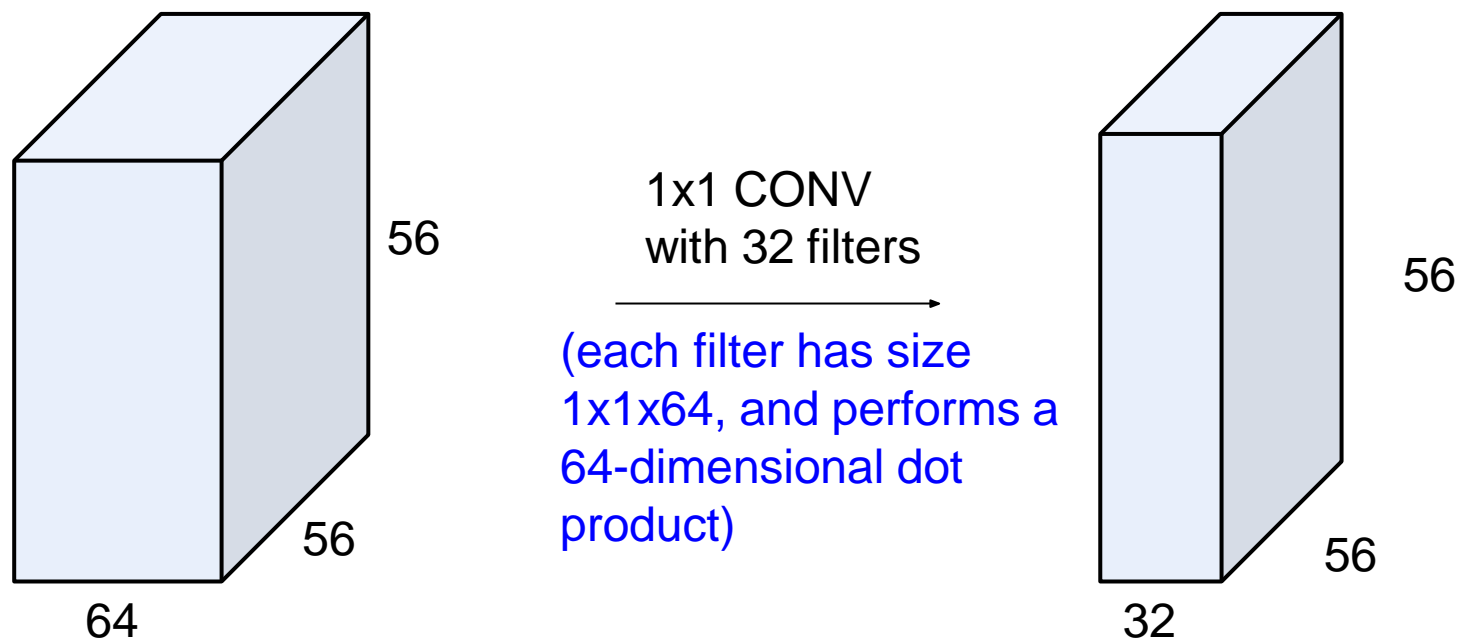
- $D_2 = K$

# Convolution Layer: Summary

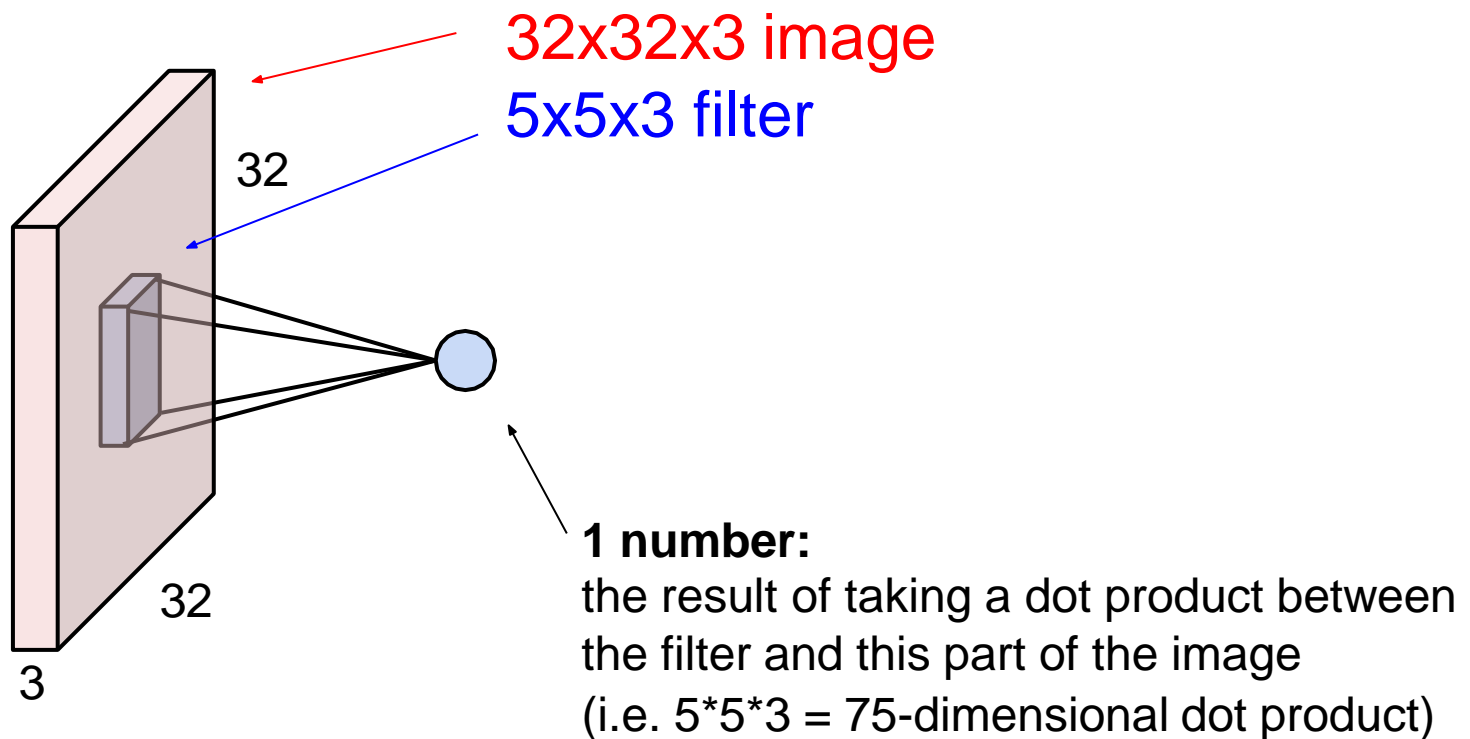
- With parameter sharing, it introduces  $F \cdot F \cdot D_1$  weights per filter, for a total of  $(F \cdot F \cdot D_1) \cdot K$  weights and  $K$  biases
- In the output volume, the  $d$ -th depth slice (of size  $W_2 \times H_2$ ) is the result of performing a valid convolution of the  $d$ -th filter over the input volume with a stride of  $S$ , and the offset by  $d$ -th bias.

# $1 \times 1$ conv

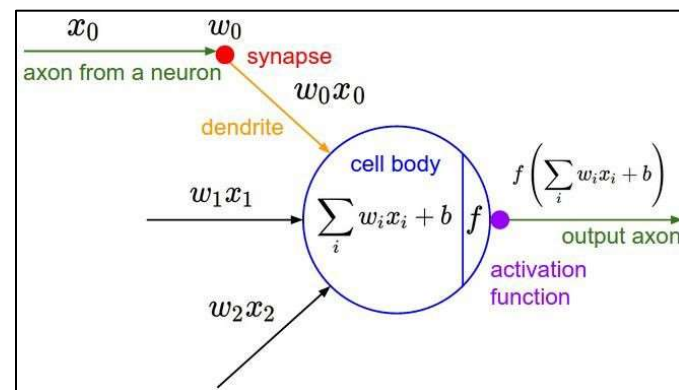
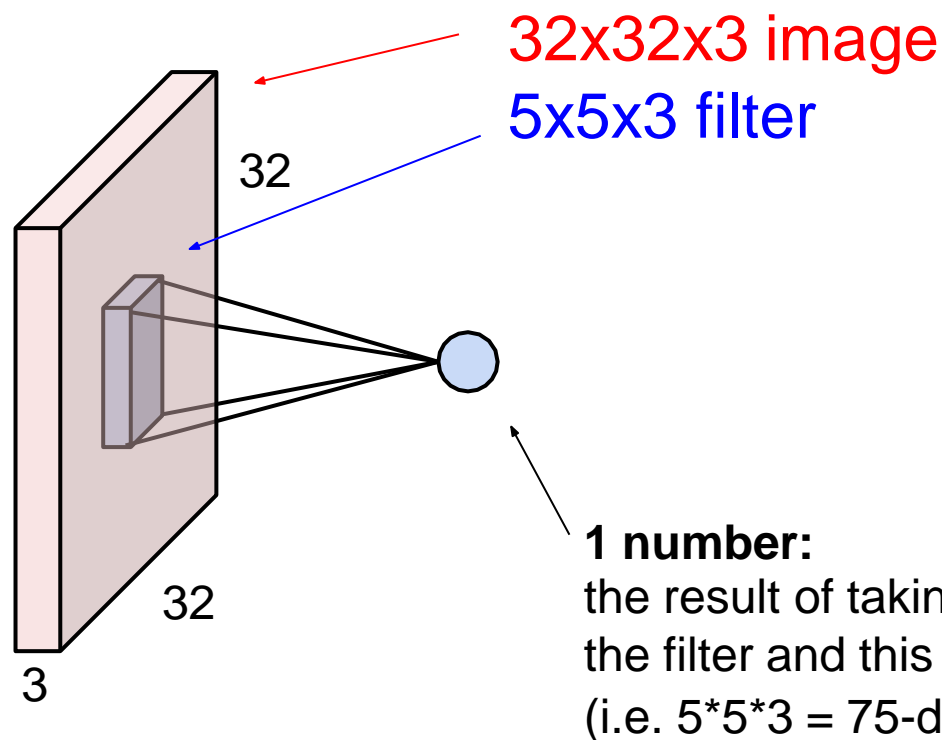
- 1x1 convolution layers make perfect sense



# The brain/neuron view of Conv Layer

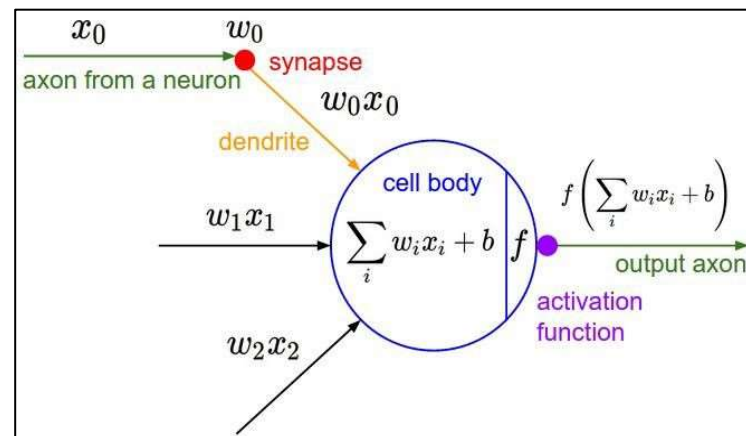
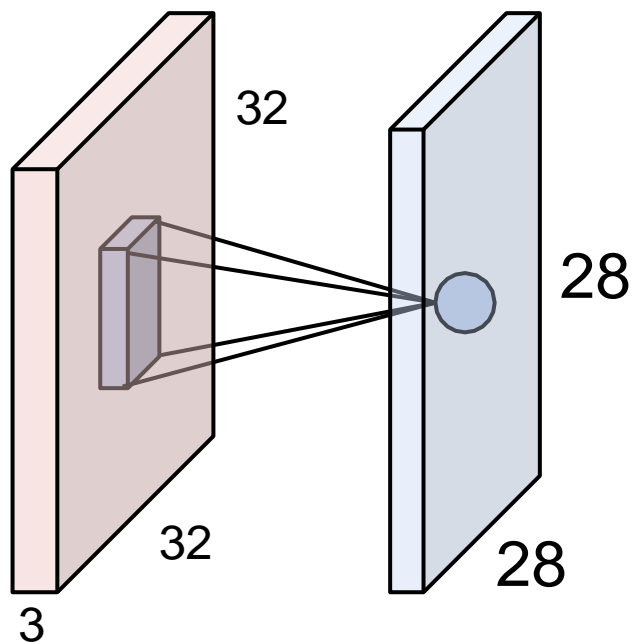


# The brain/neuron view of Conv Layer



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

# The brain/neuron view of Conv Layer

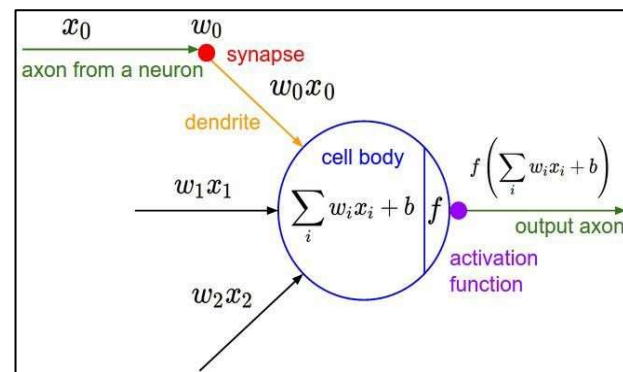
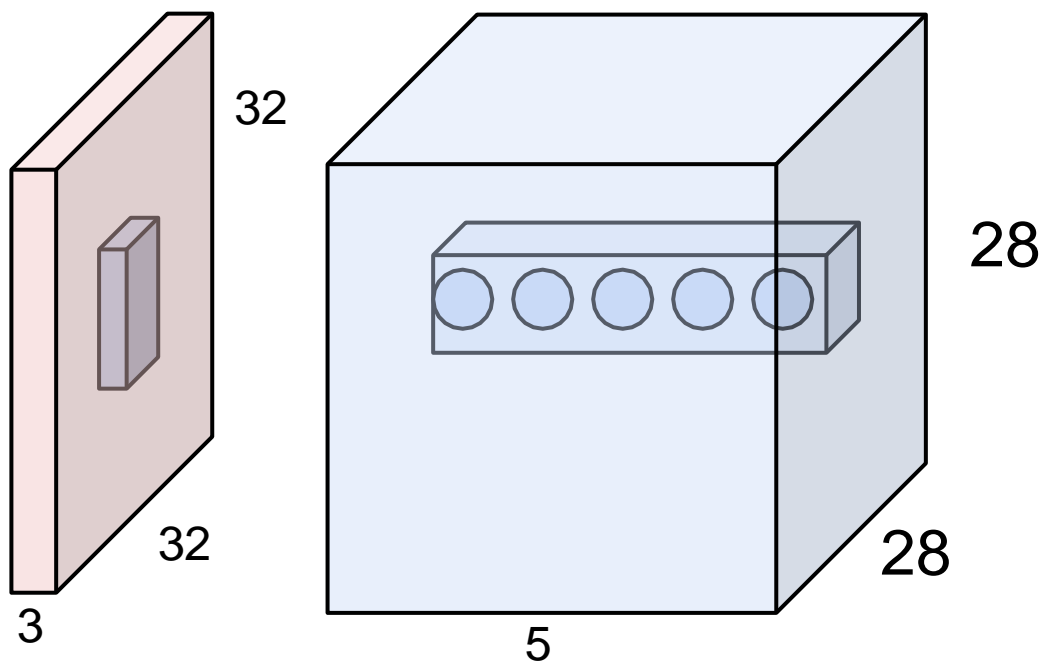


An activation map is a 28x28 sheet of neuron outputs:

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

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

# The brain/neuron view of Conv Layer



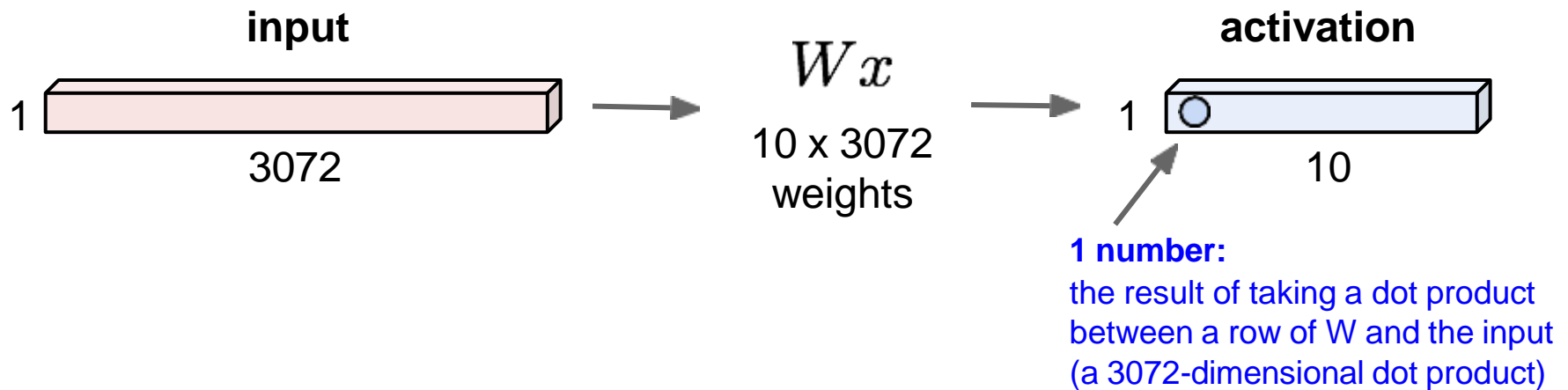
E.g. with 5 filters,  
CONV layer consists of  
neurons arranged in a 3D grid  
(28x28x5)

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

# Reminder: FC Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron  
looks at the full  
input volume



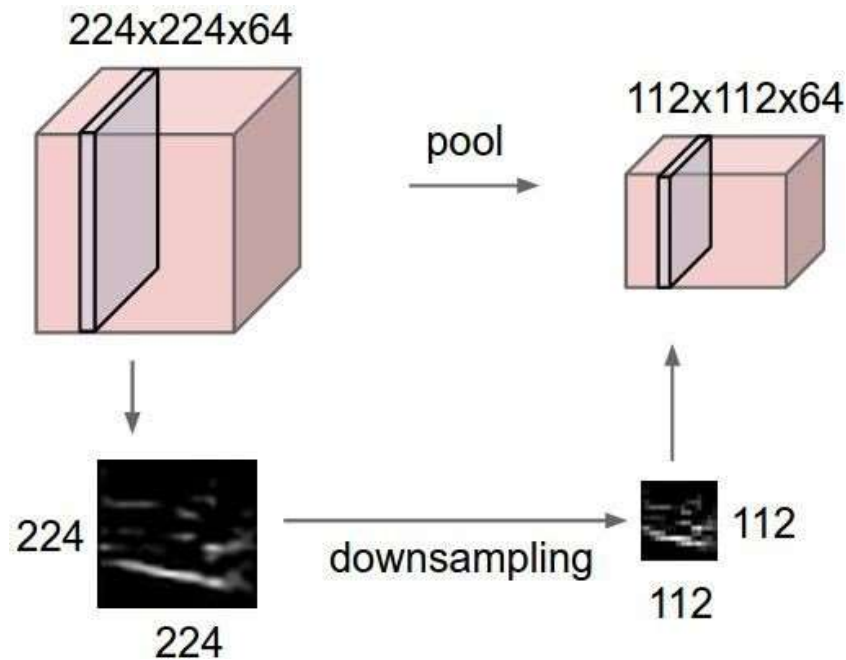


# CNN Layer

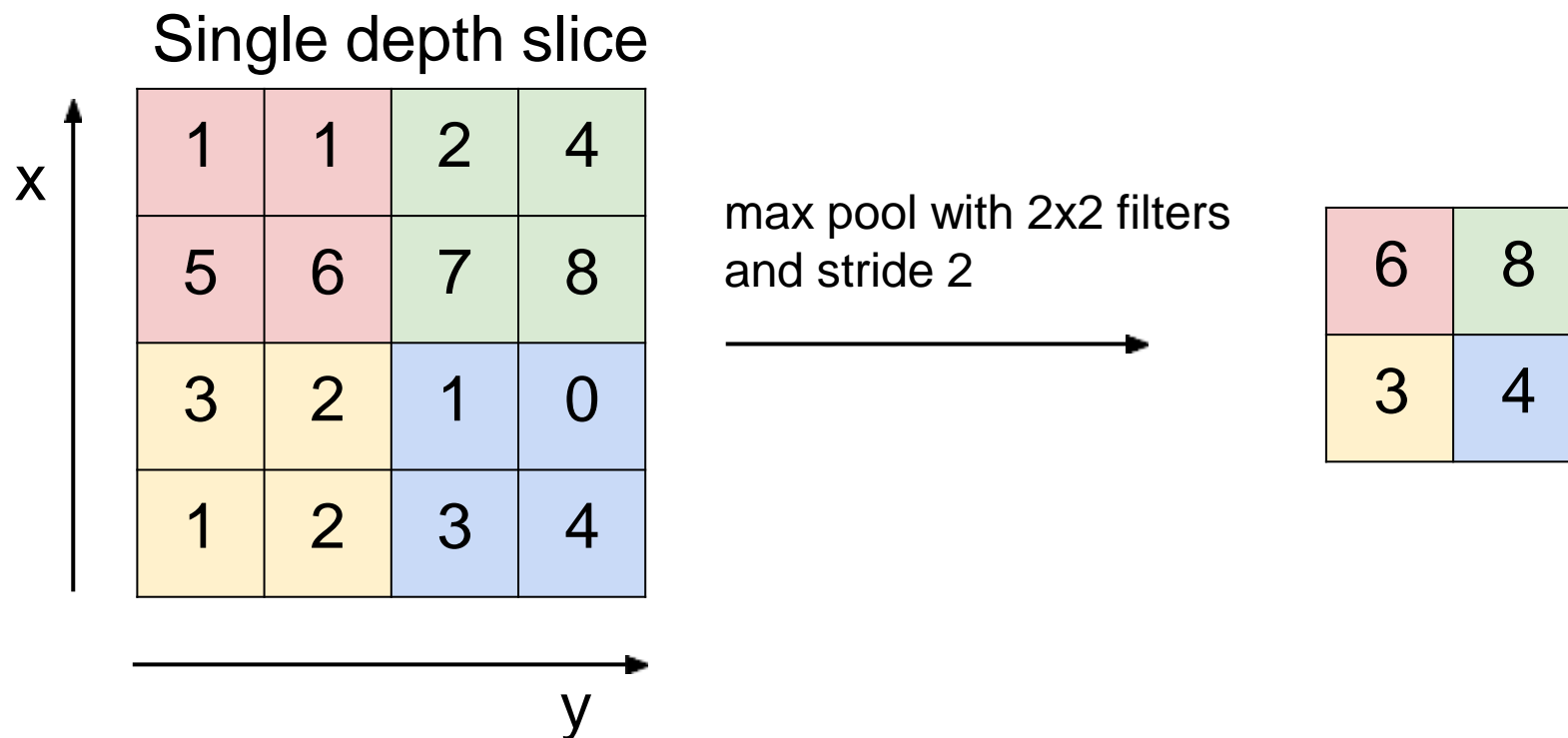
- Fully Connected Layer
- Convolution Layer
- *Pooling Layer*

# Pooling Layer

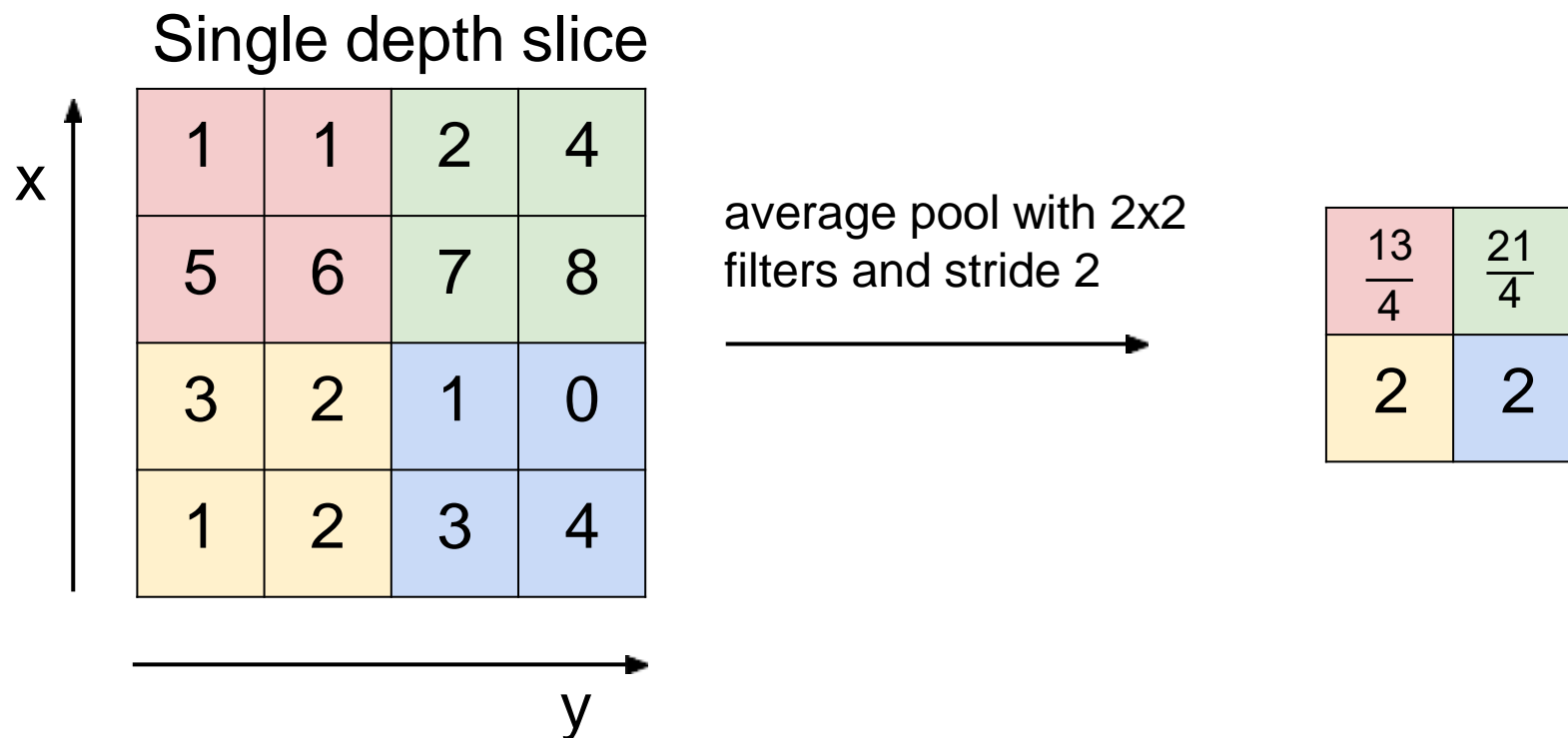
- makes the representations smaller and more manageable
- operates over each activation map independently:



# Max Pooling



# Average Pooling



# Pooling Layer: Summary

Common settings:

$F = 2, S = 2$

$F = 3, S = 2$

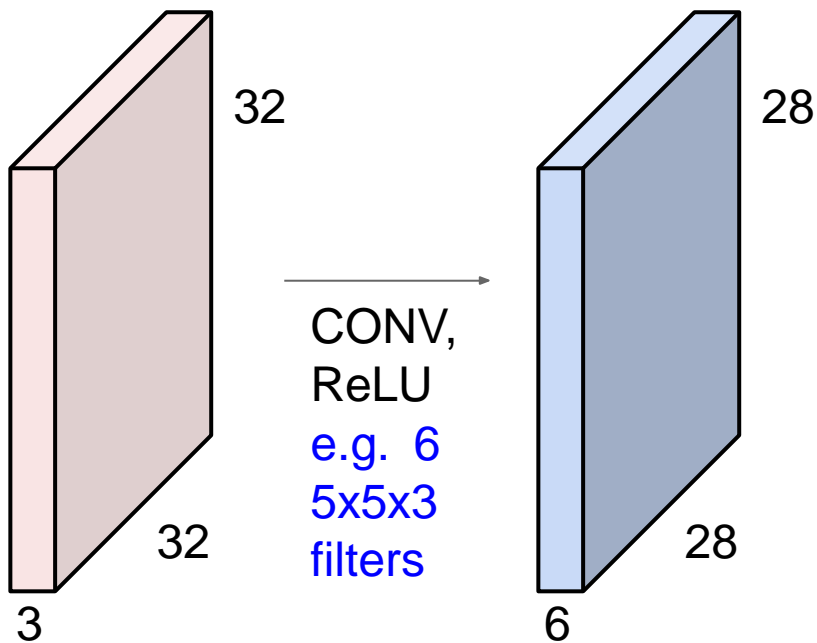
- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires 2 hyperparameters:
  - Their spatial extent  $F$
  - The stride  $S$
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = \frac{W_1 - F}{S} + 1$
  - $H_2 = \frac{H_1 - F}{S} + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

# Today's Topics

- CNN Development
- CNN Layer
- ***CNN Architecture***

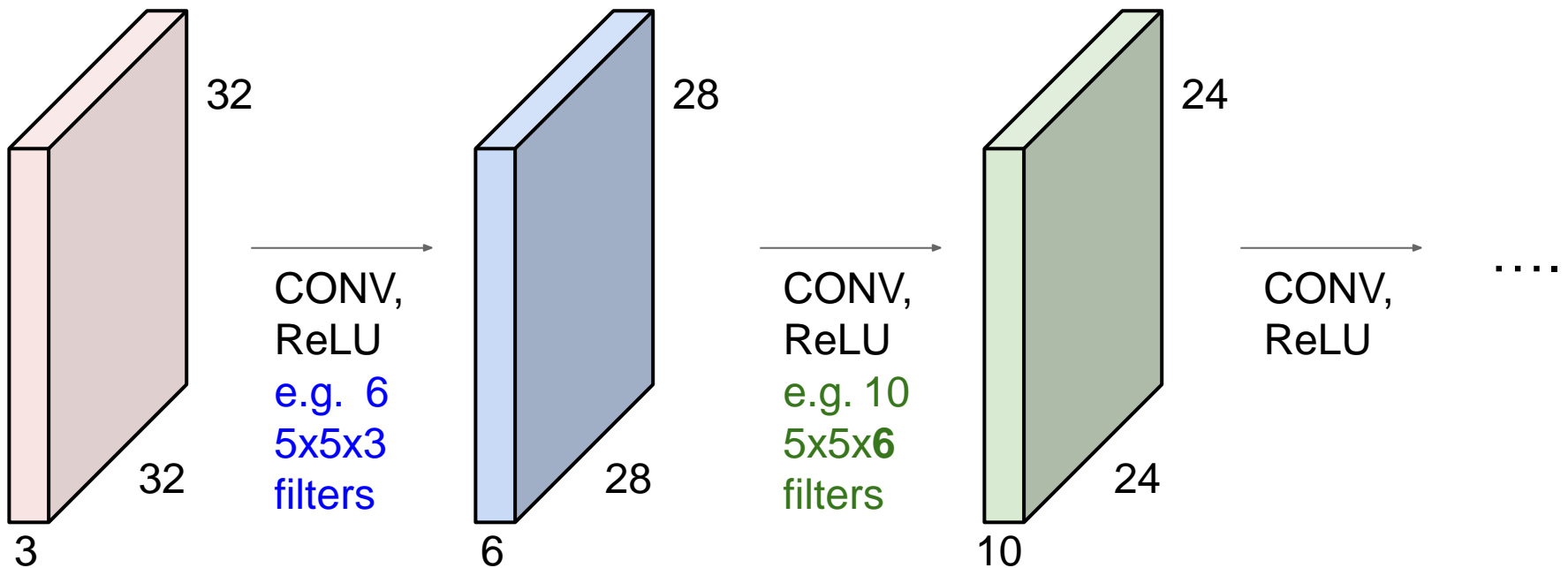
# CNN Architecture

- ConvNet is a sequence of Convolution Layers, interspersed with activation functions



# CNN Architecture

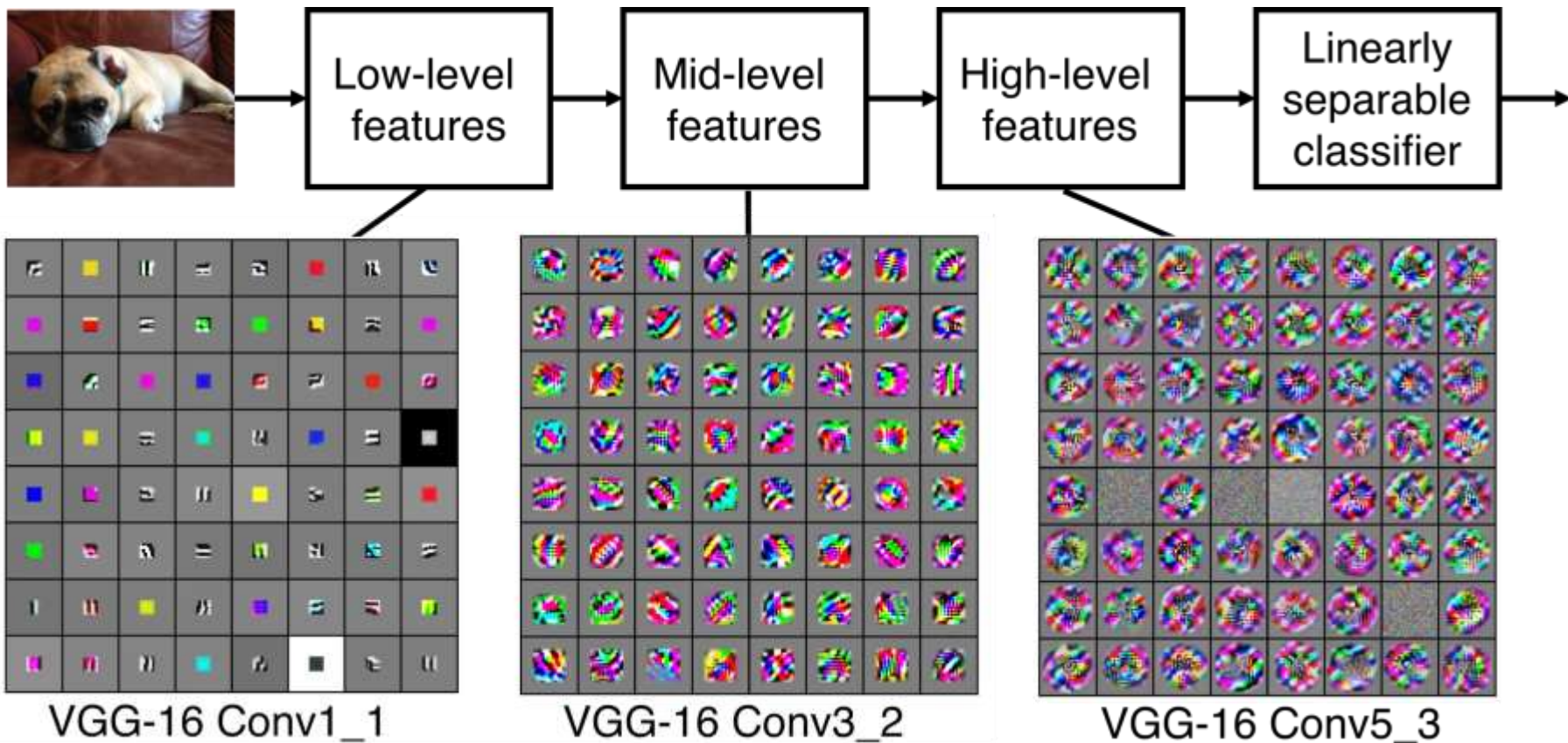
- ConvNet is a sequence of Convolution Layers, interspersed with activation functions





# CNN Architecture

- ConvNet is a sequence of Convolution Layers, interspersed with activation functions



# CNN Architecture

- ConvNet is a sequence of Convolution Layers, interspersed with activation functions



# CNN Summary

- ConvNets stack CONV, POOL, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like
  - **$[(\text{CONV-RELU}) * N - \text{POOL?}] * M - (\text{FC-RELU}) * K, \text{SOFTMAX}$**
  - where N is usually up to  $\sim 5$ , M is large,  $0 \leq K \leq 2$ .
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm

# Thinking

- Why do we introduce a max pooling layer instead of using a convolutional layer with a larger stride?
- What are the advantages of convolutional layers over fully connected layers for image classification?