

# Deep Learning

## Lecture 7

# Convolutions

# A bit of history...

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

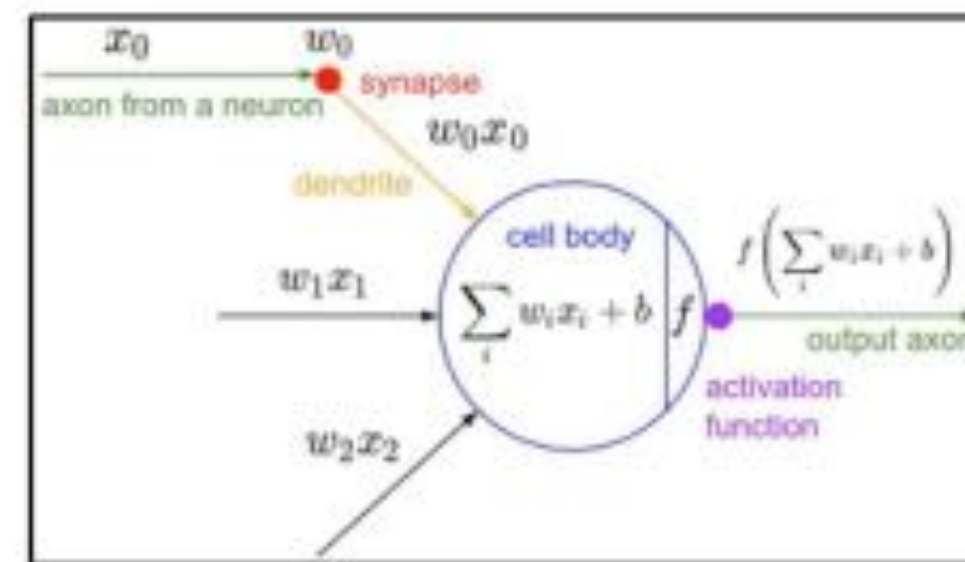
The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized  
letters of the alphabet

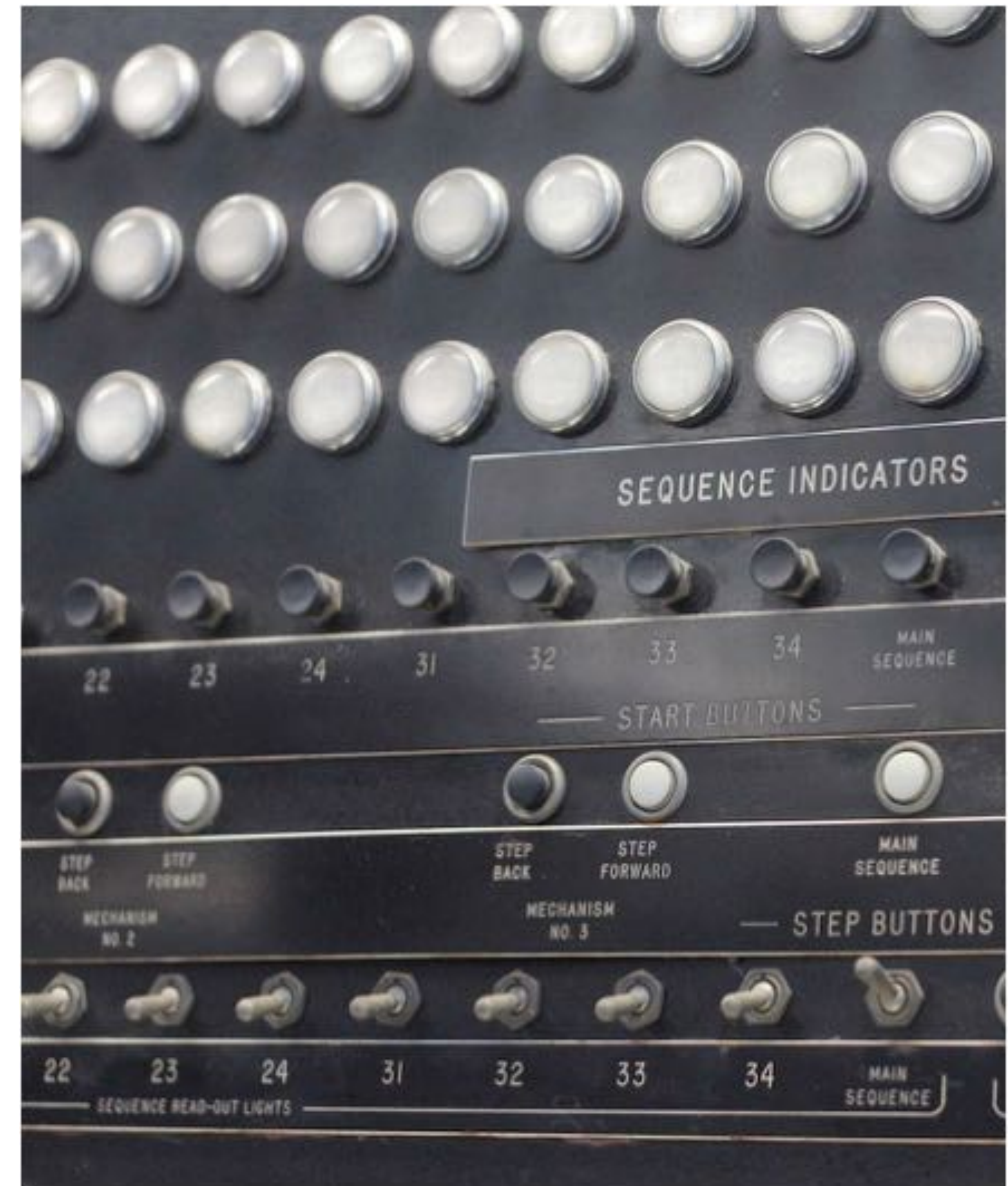
update rule:

$$w_i(t + 1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



Frank Rosenblatt, ~1957: Perceptron



[This image](#) by Rocky Acosta is licensed under [CC-BY 3.0](#)

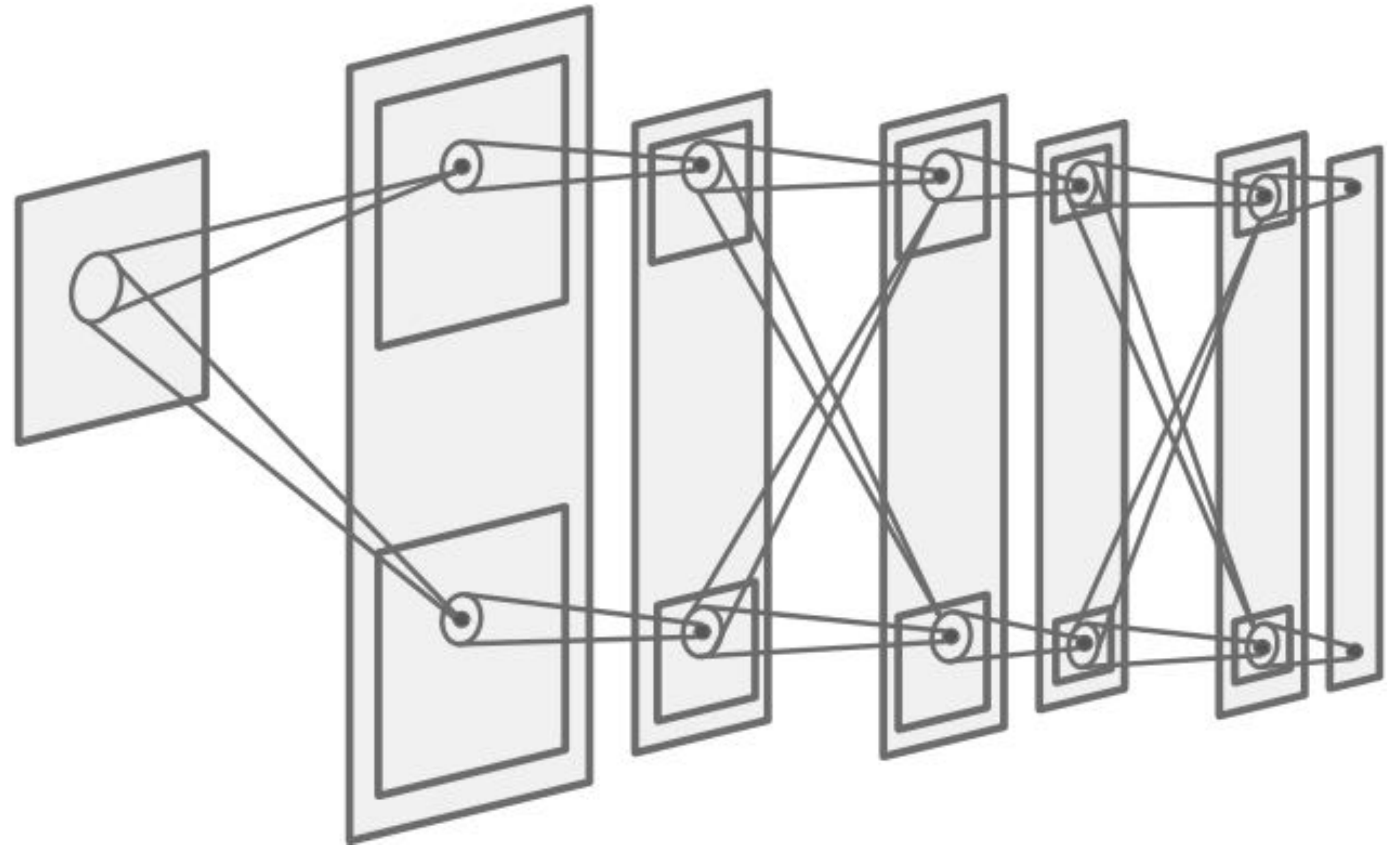


A bit of history:

# Neocognitron

*[Fukushima 1980]*

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



# A bit of history...

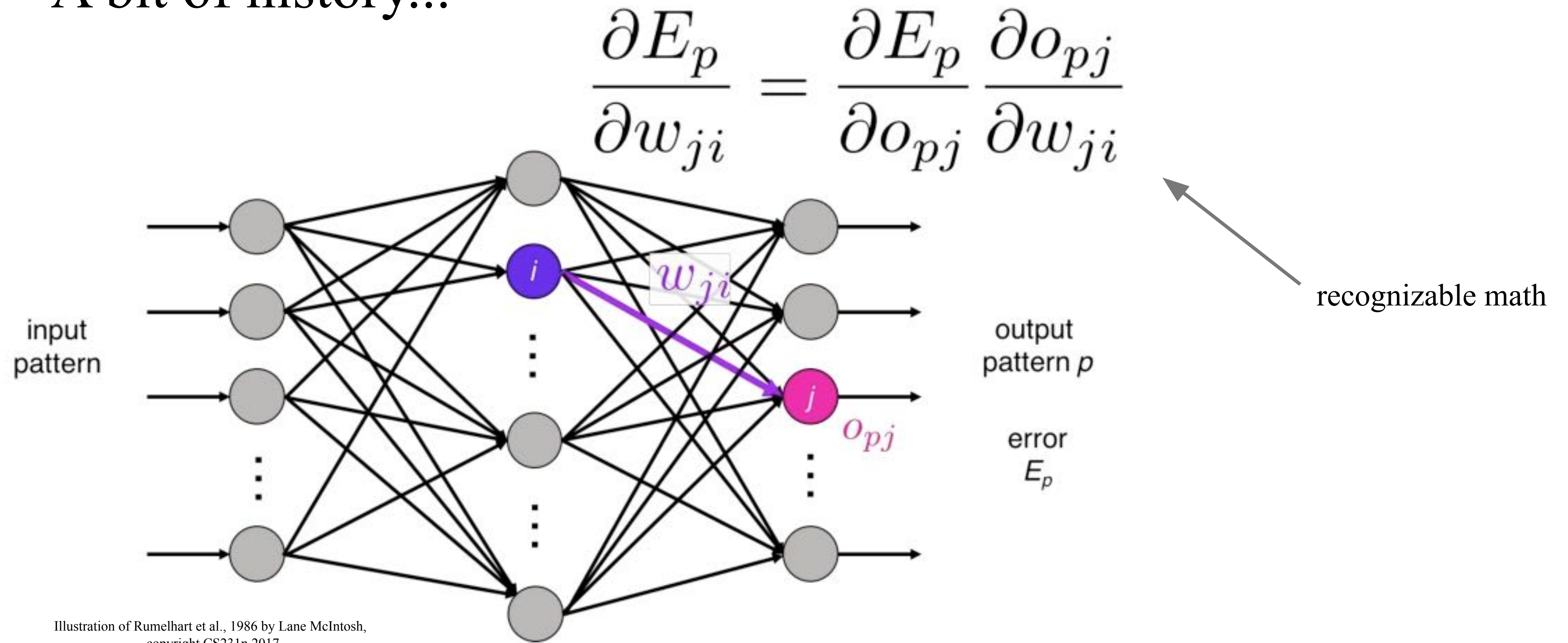


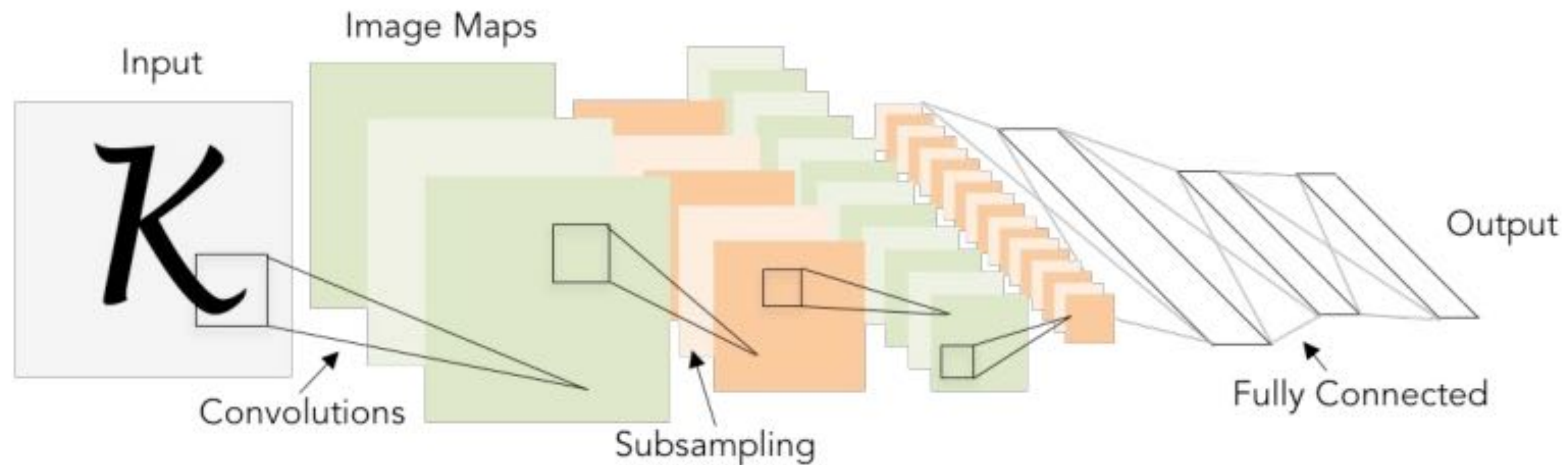
Illustration of Rumelhart et al., 1986 by Lane McIntosh,  
copyright CS231n 2017

Rumelhart et al., 1986: First time back-propagation became popular

# A bit of history:

## Gradient-based learning applied to document recognition

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



LeNet-5



# A bit of history...

[Hinton and Salakhutdinov 2006]

## Reinvigorated research in Deep Learning

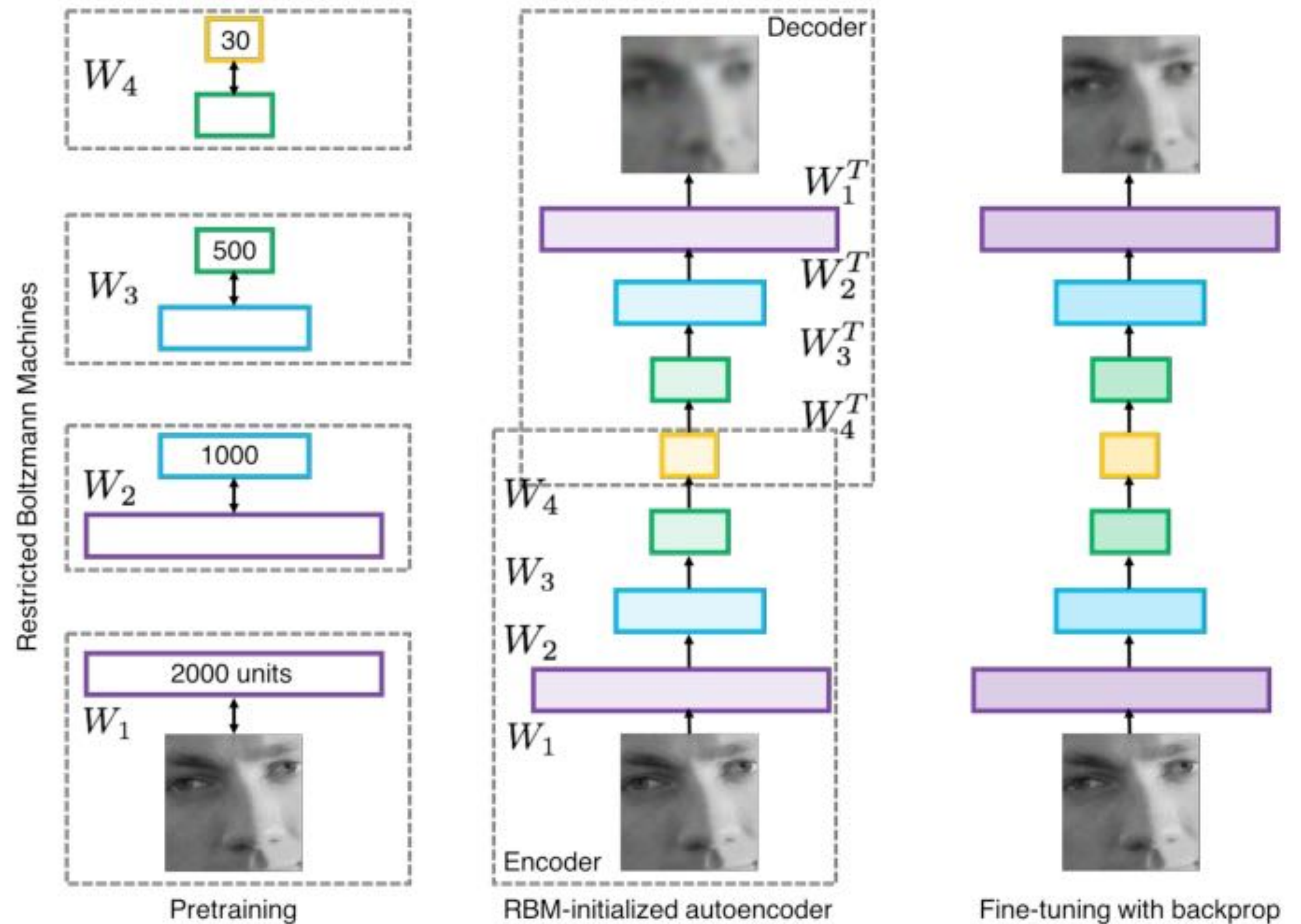


Illustration of Hinton and Salakhutdinov 2006 by Lane McIntosh, copyright CS231n 2017

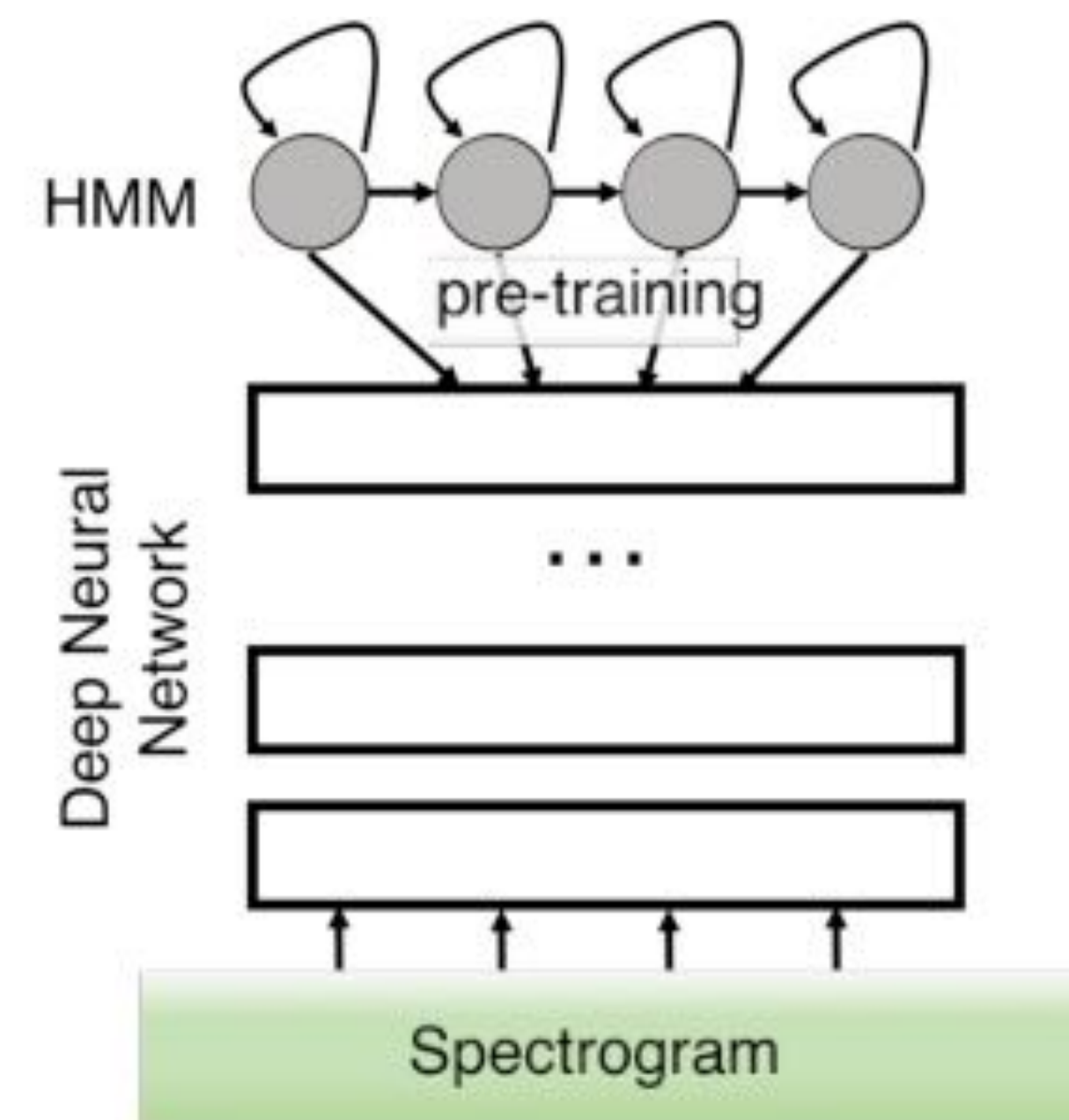
# First strong results

## *Acoustic Modeling using Deep Belief Networks*

*Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010*

## *Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition*

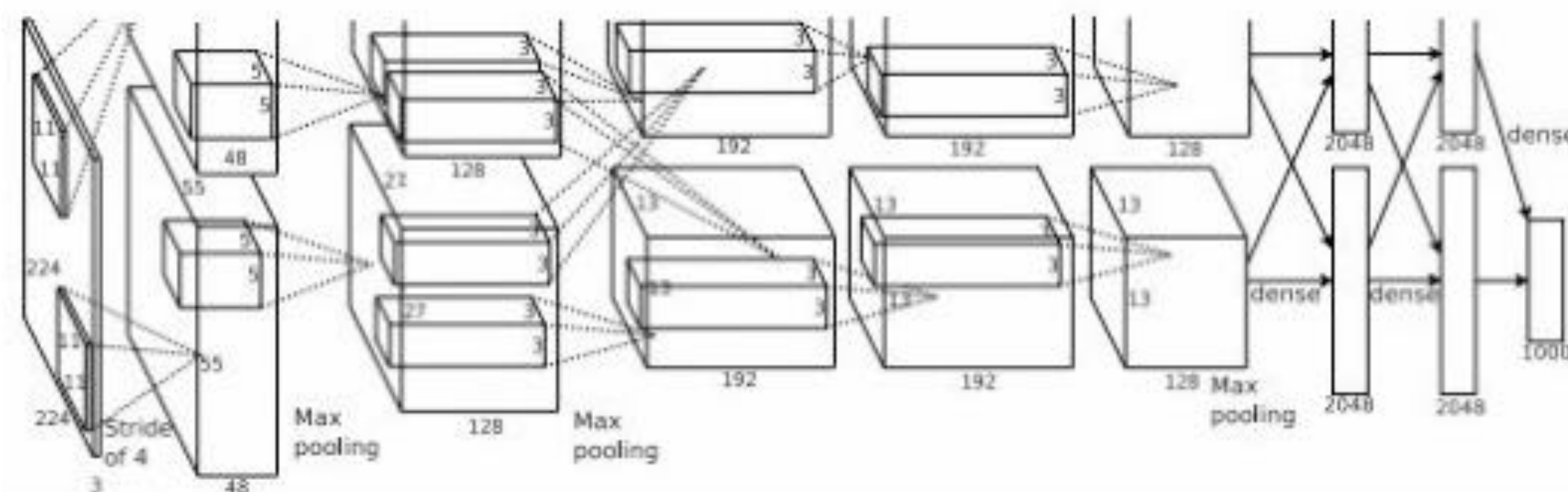
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012



copyright

## *Imagenet classification with deep convolutional neural networks*

Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012



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



# A bit of history:

## ImageNet Classification with Deep Convolutional Neural Networks

*[Krizhevsky, Sutskever, Hinton, 2012]*

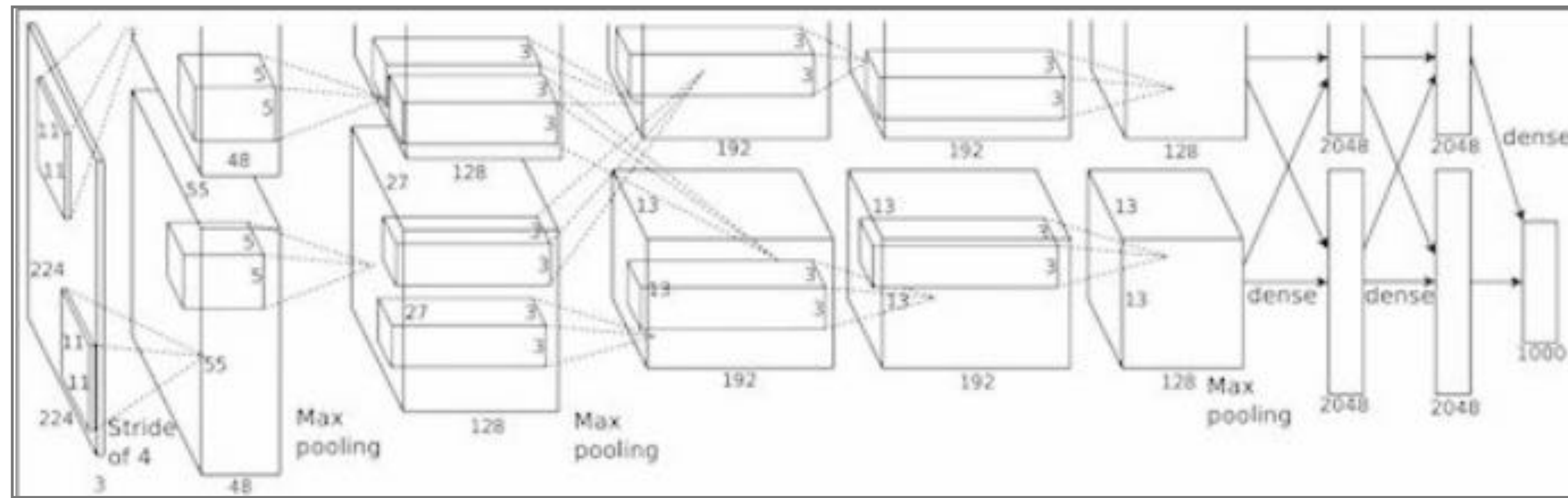


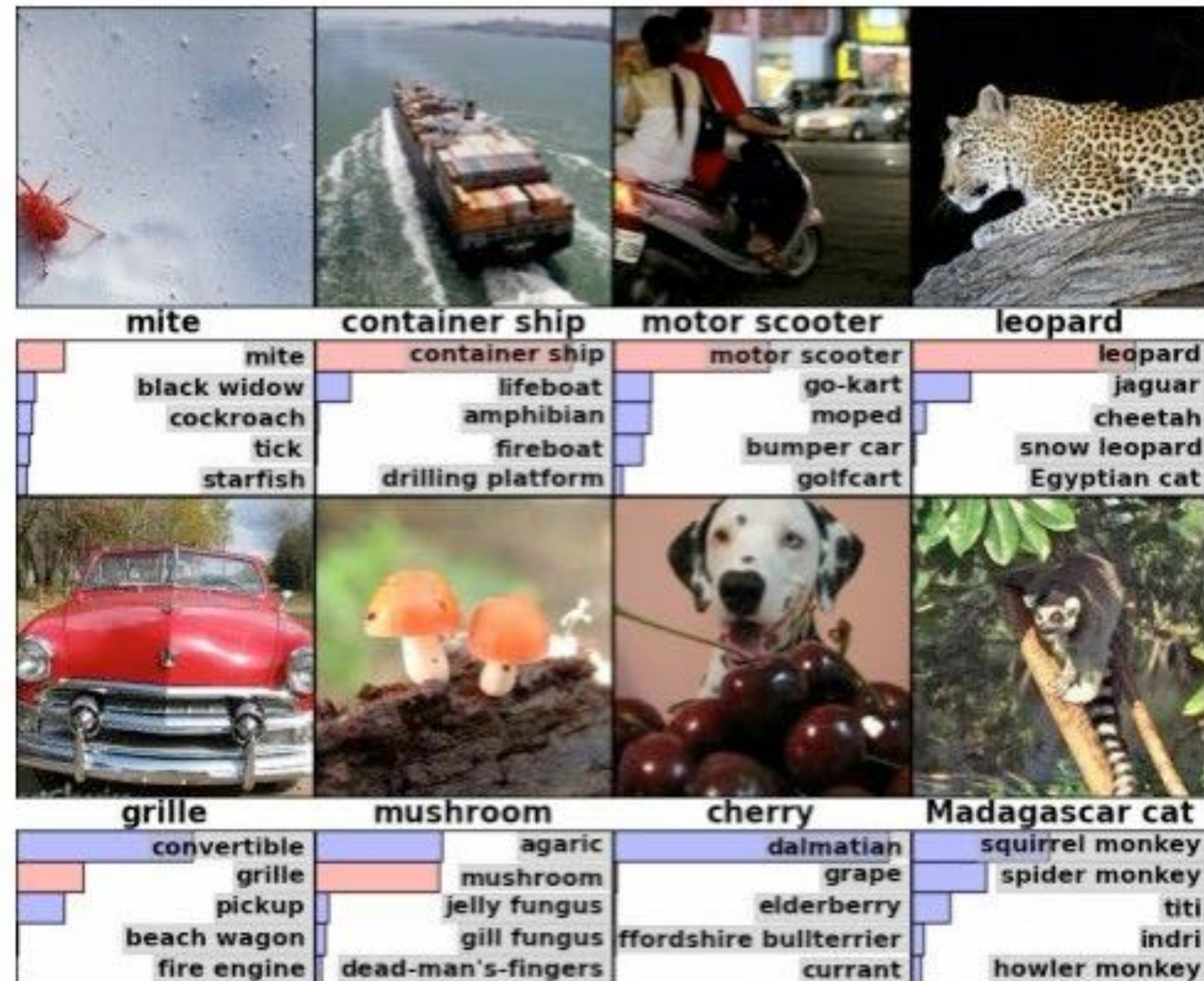
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”



# Fast-forward to today: ConvNets are everywhere

## Classification



## Retrieval

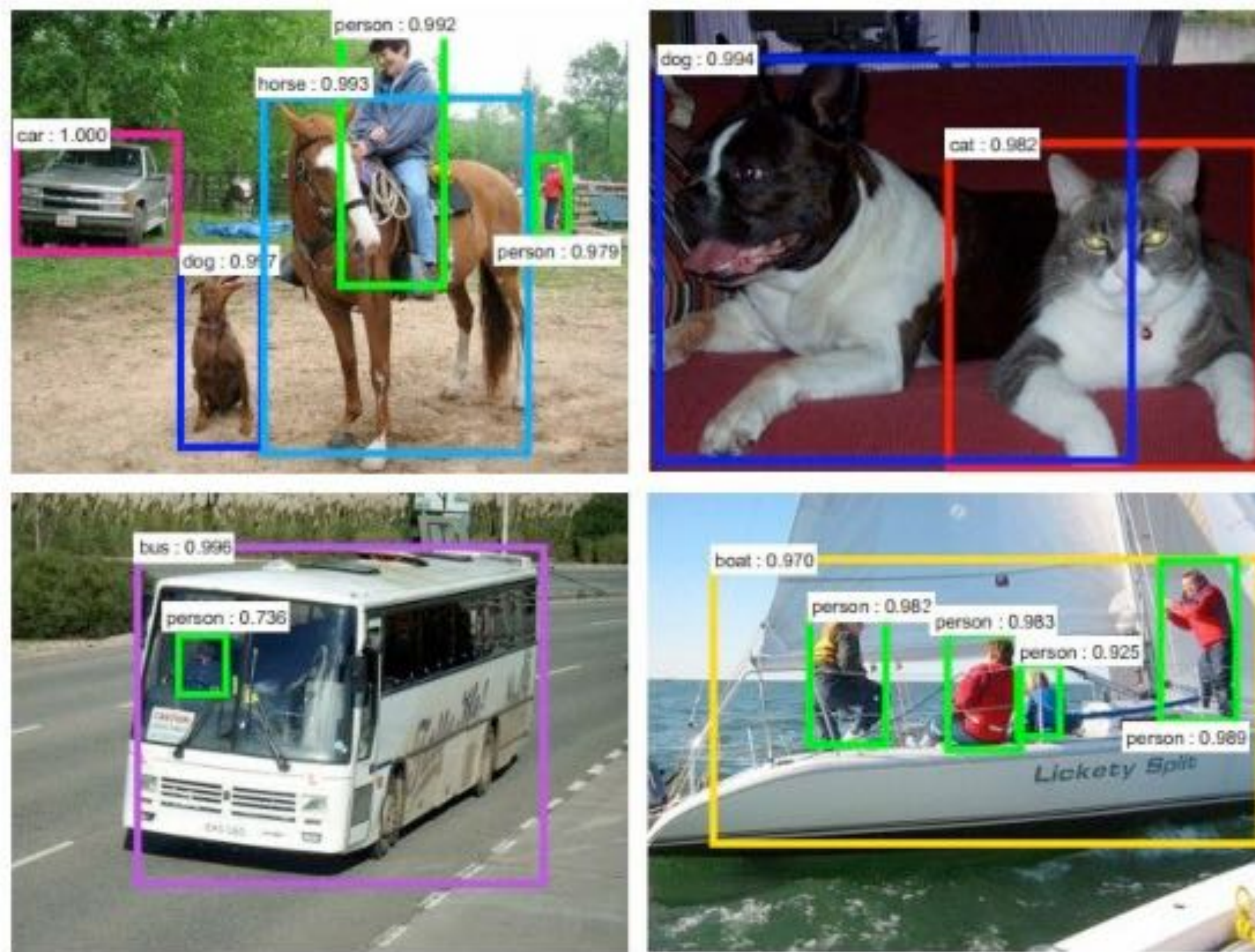


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.



# Fast-forward to today: ConvNets are everywhere

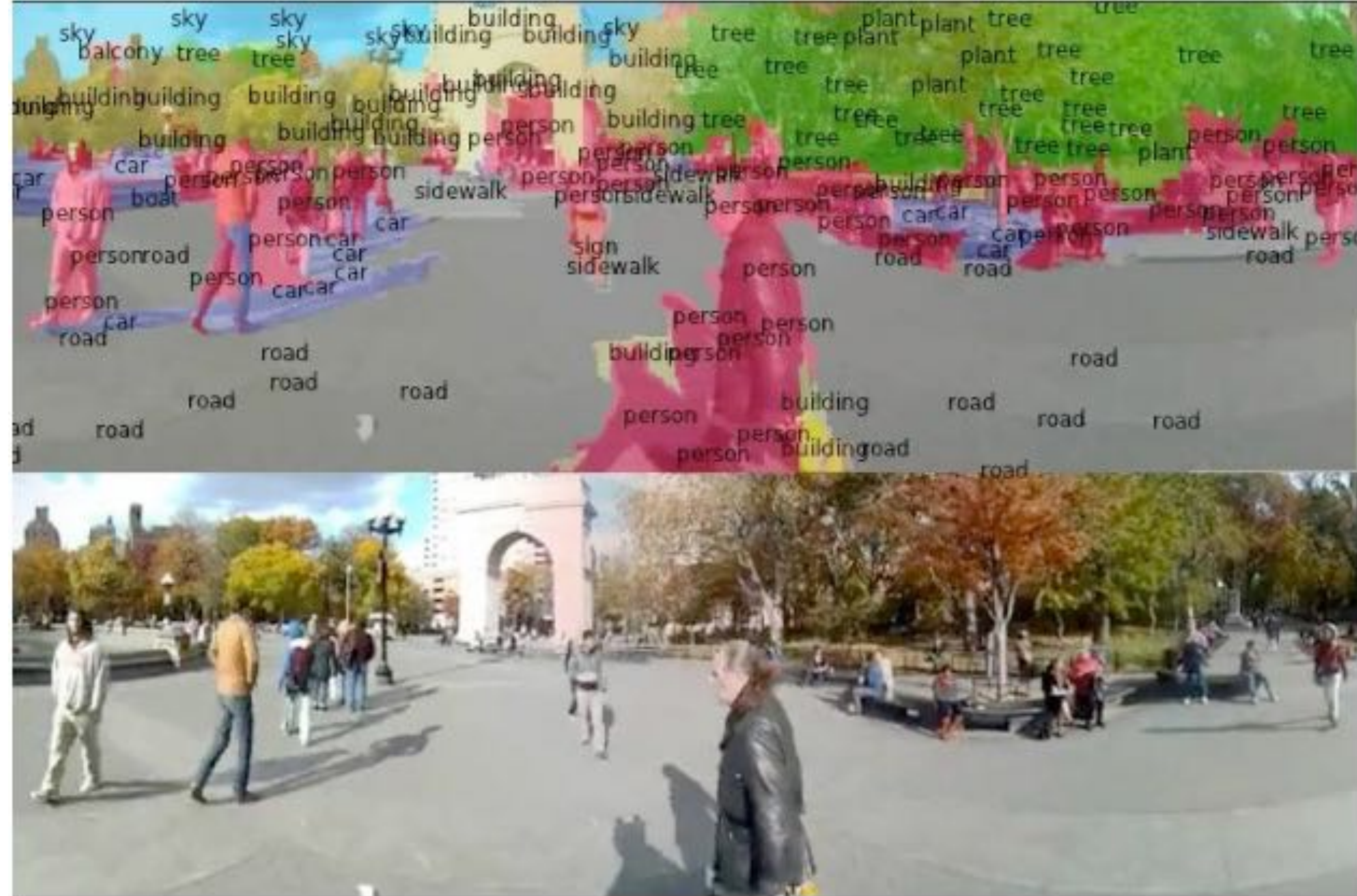
# Detection



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



self-driving cars

Photo by Lane McIntosh. Copyright CS231n 2017.



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

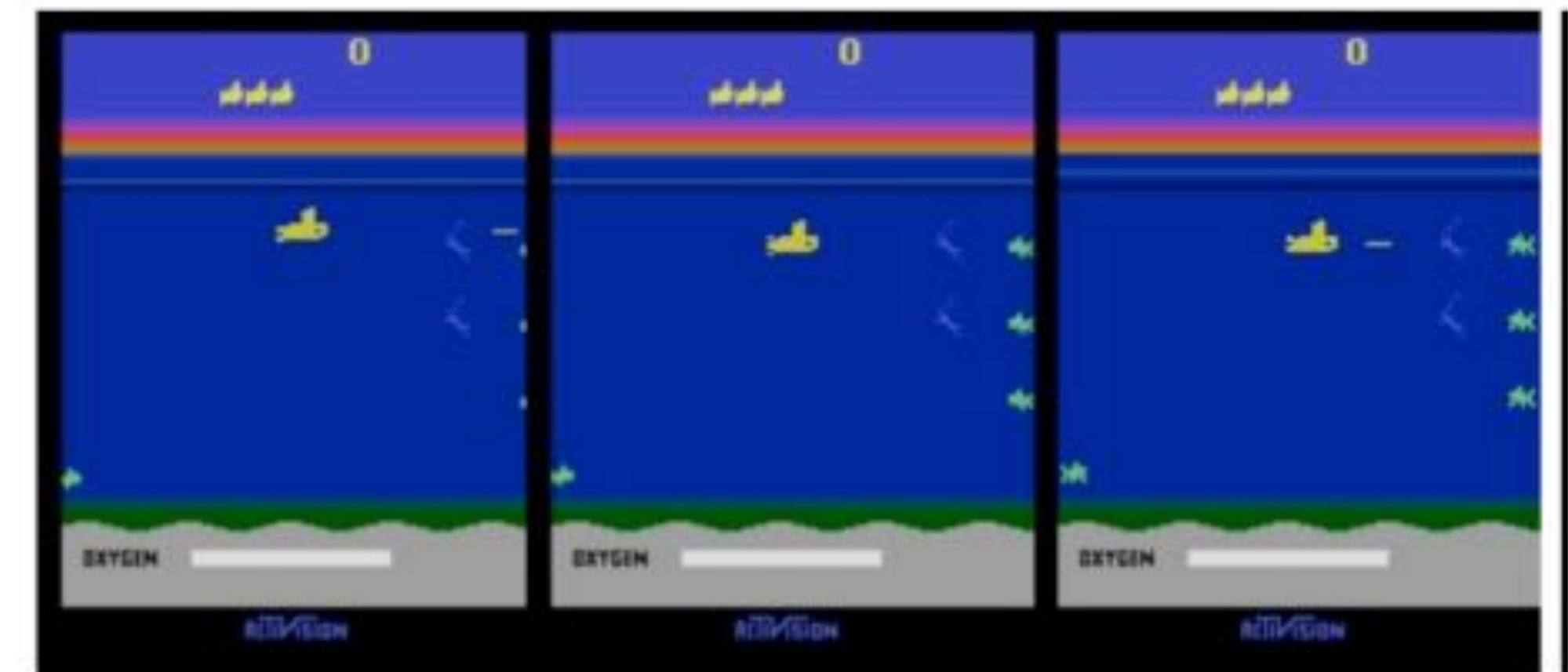
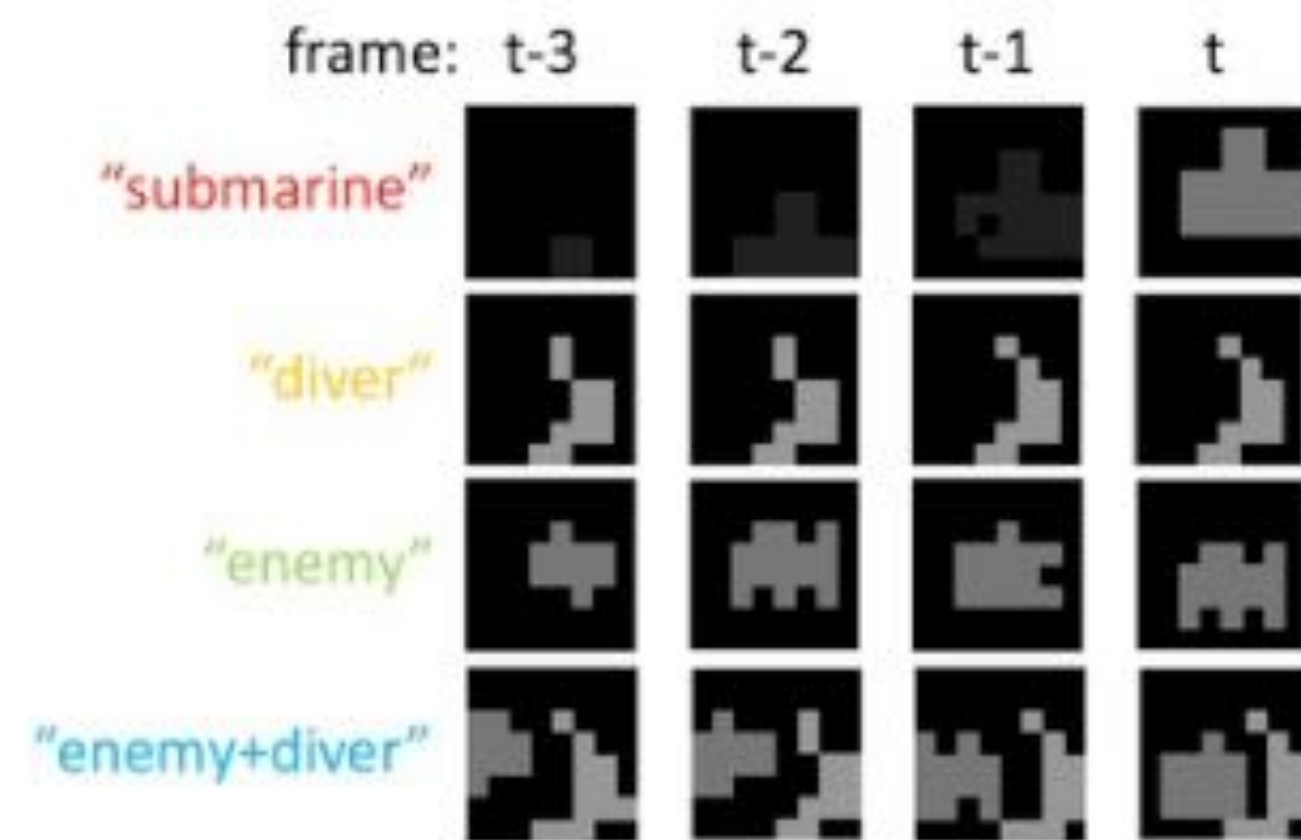


# Fast-forward to today: ConvNets are everywhere



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

*[Toshev, Szegedy 2014]*

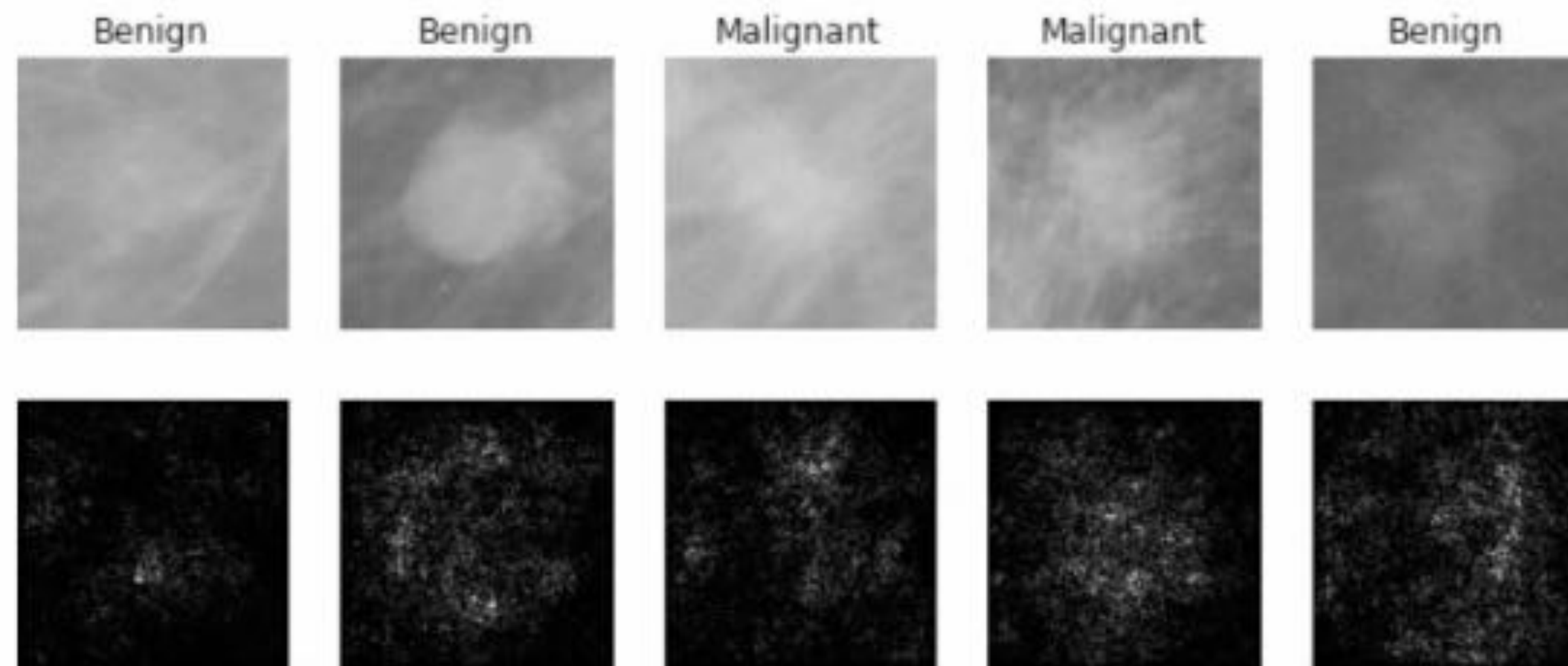


*[Guo et al. 2014]*

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.



[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*



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*



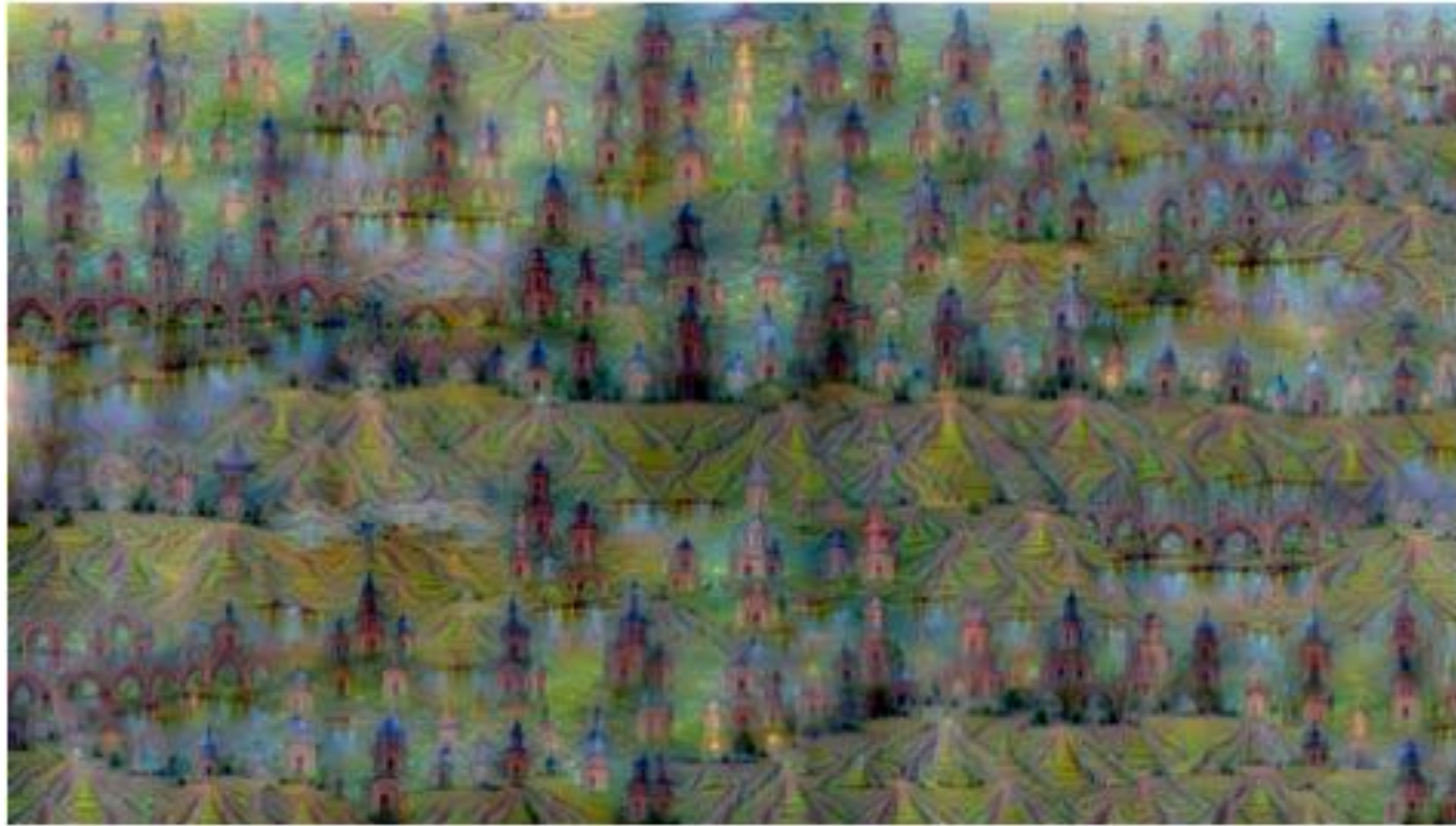
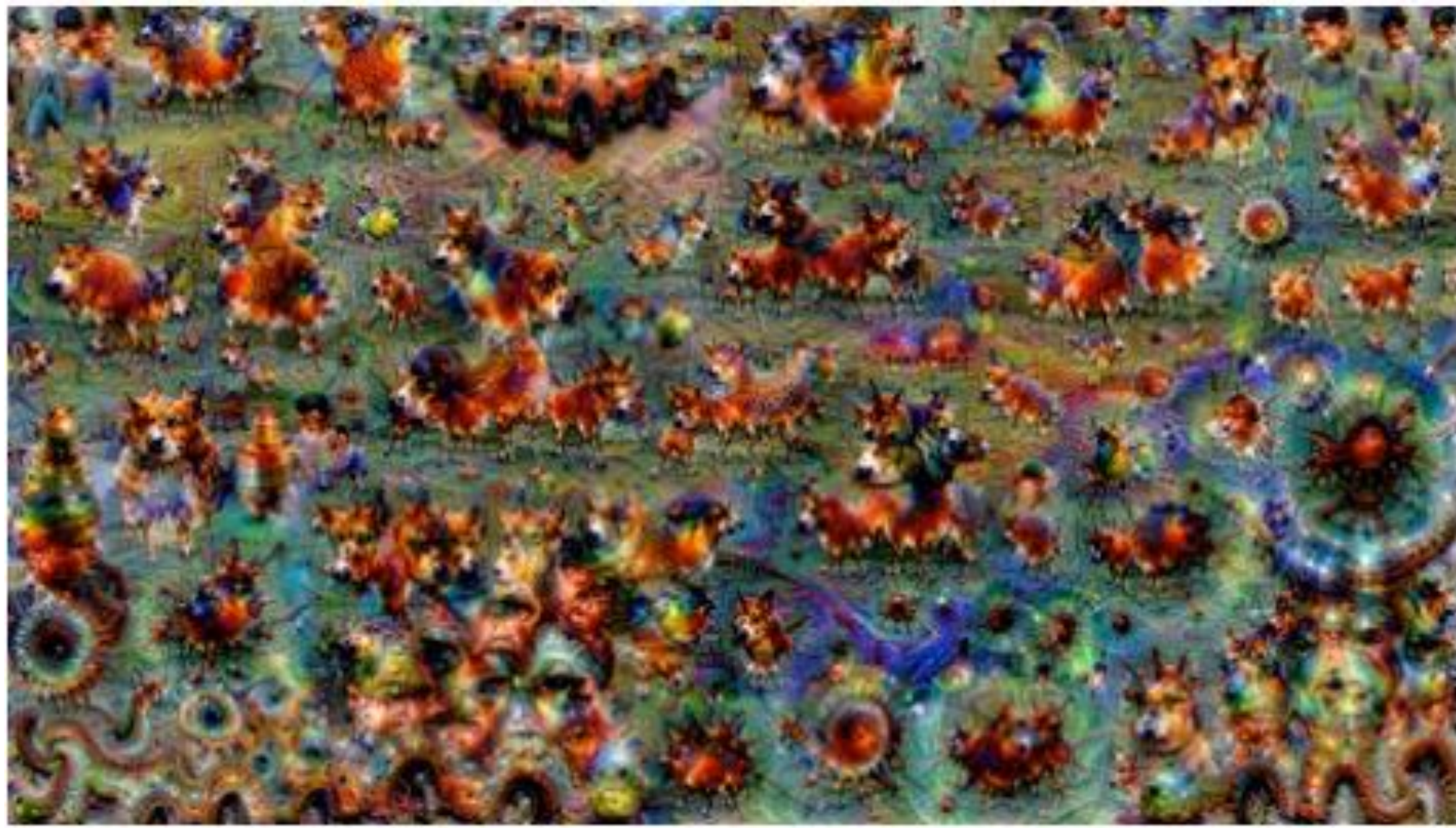
*A woman standing on a beach holding a surfboard*

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](#)





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

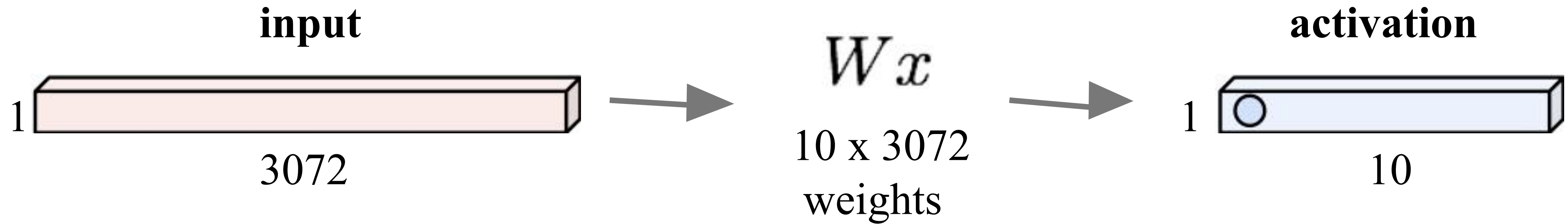


Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016  
 Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017



# Fully Connected Layer

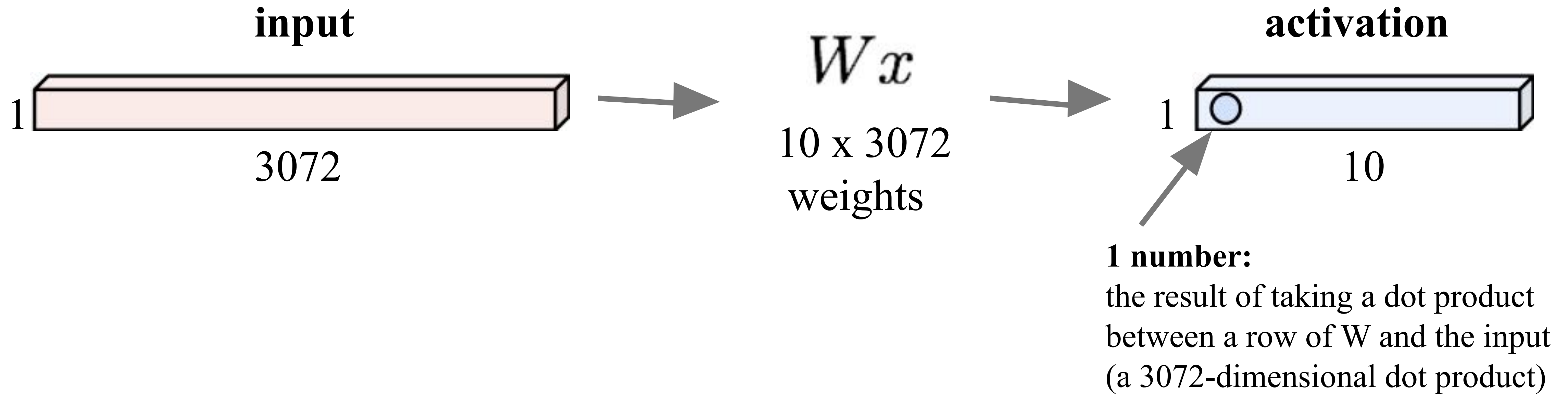
32x32x3 image -> stretch to 3072 x 1





# Fully Connected Layer

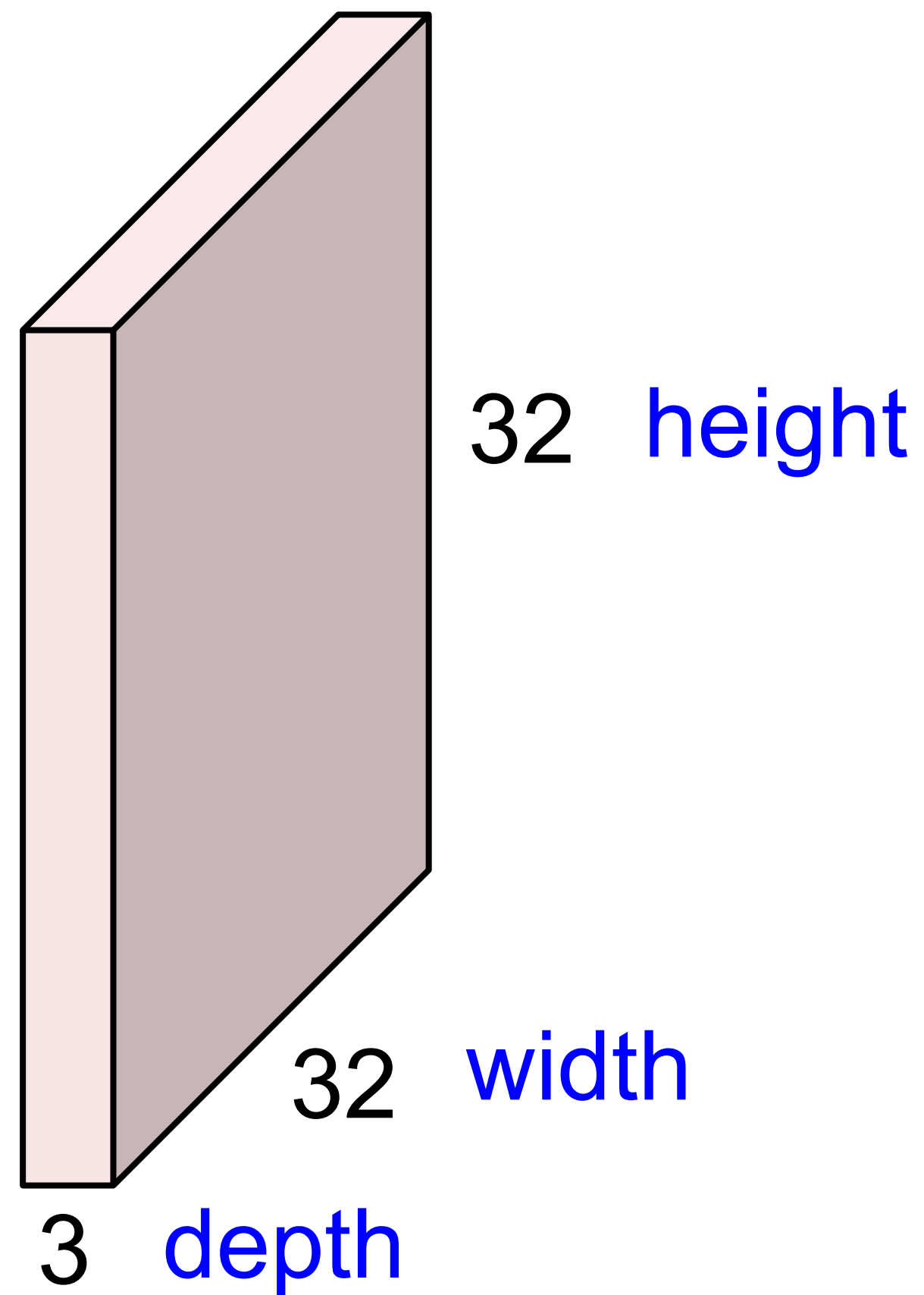
32x32x3 image -> stretch to 3072 x 1





# Convolution Layer

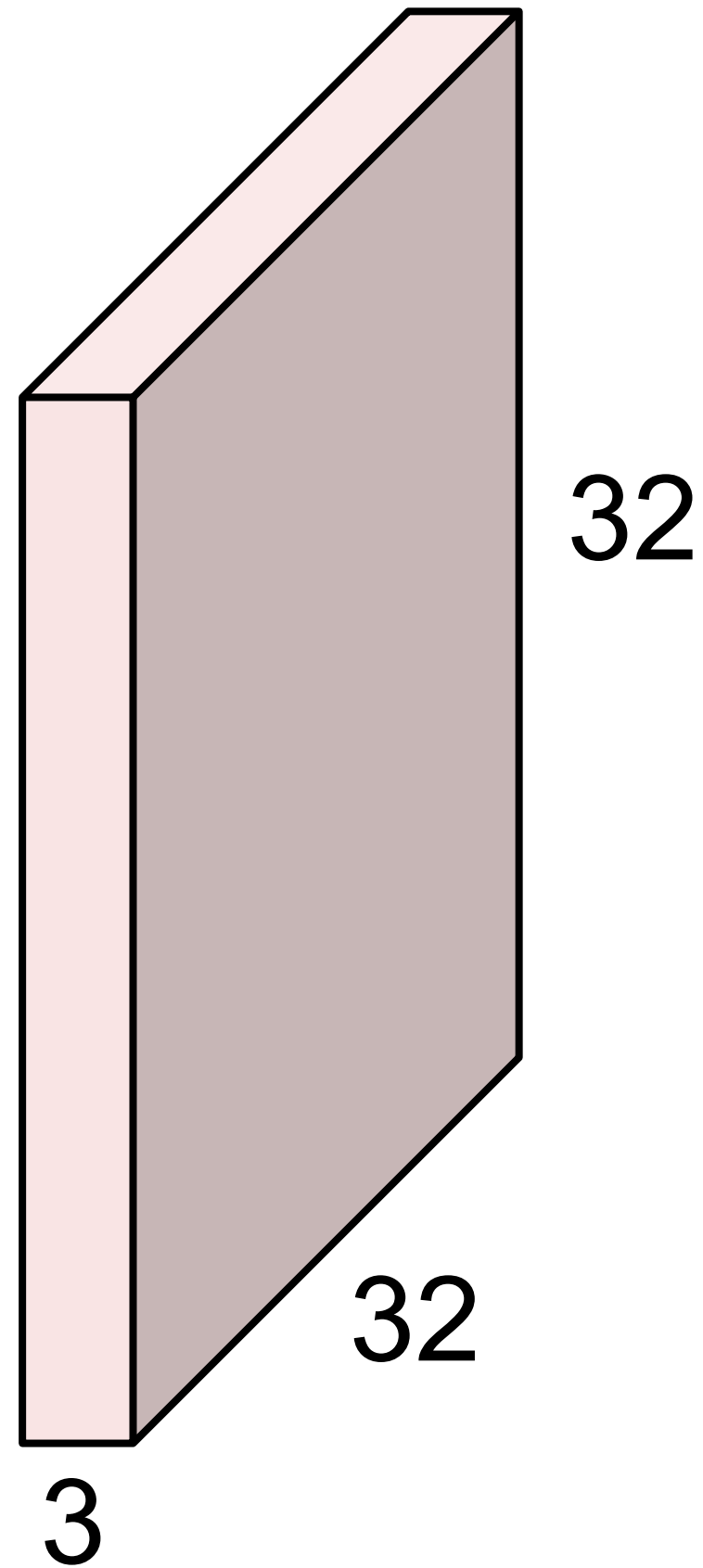
32x32x3 image



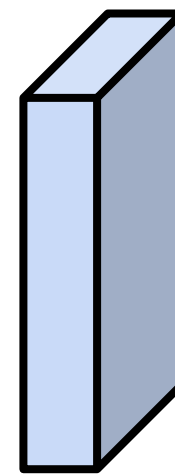


# Convolution Layer

32x32x3 image



5x5x3 filter



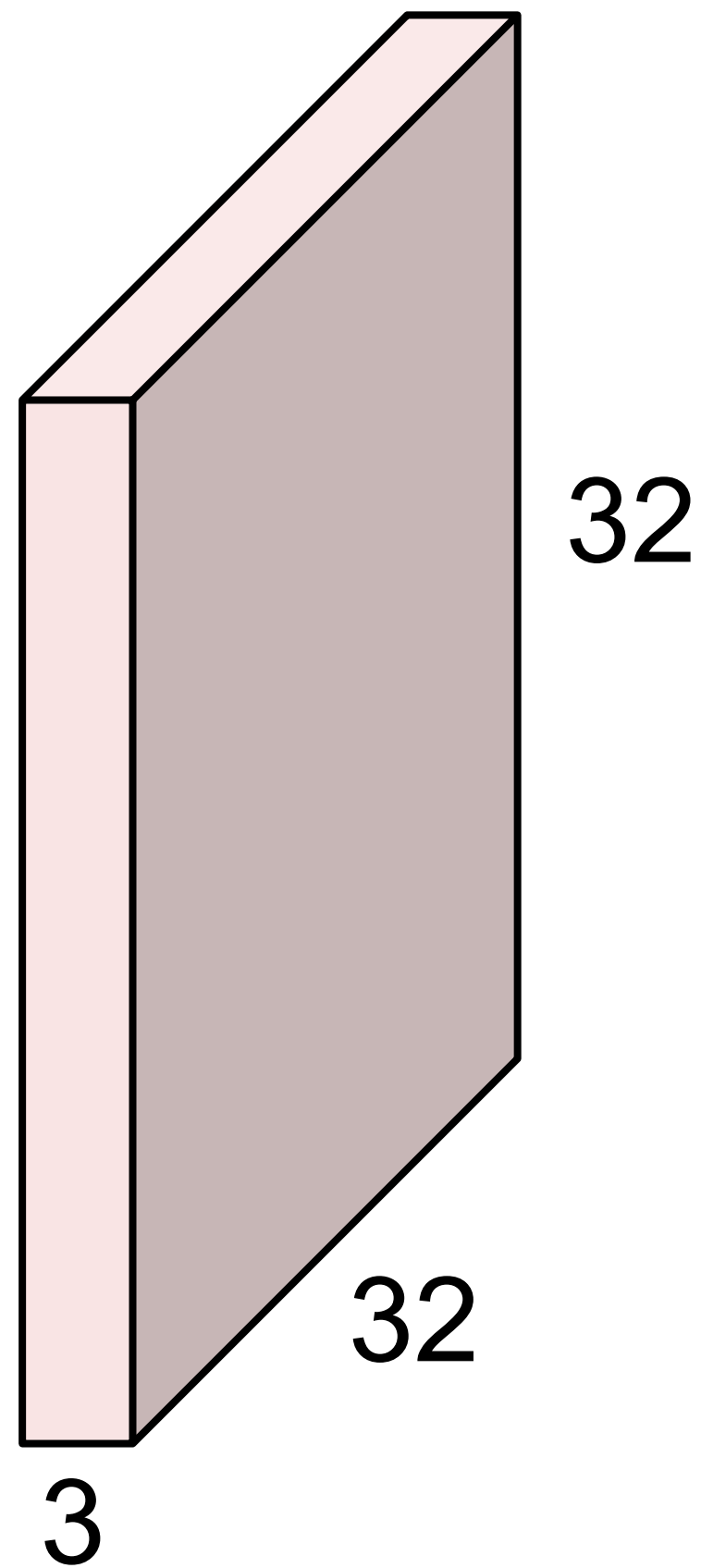
**Convolve** the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”



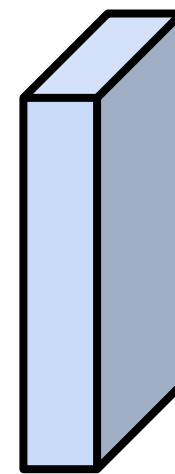
# Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



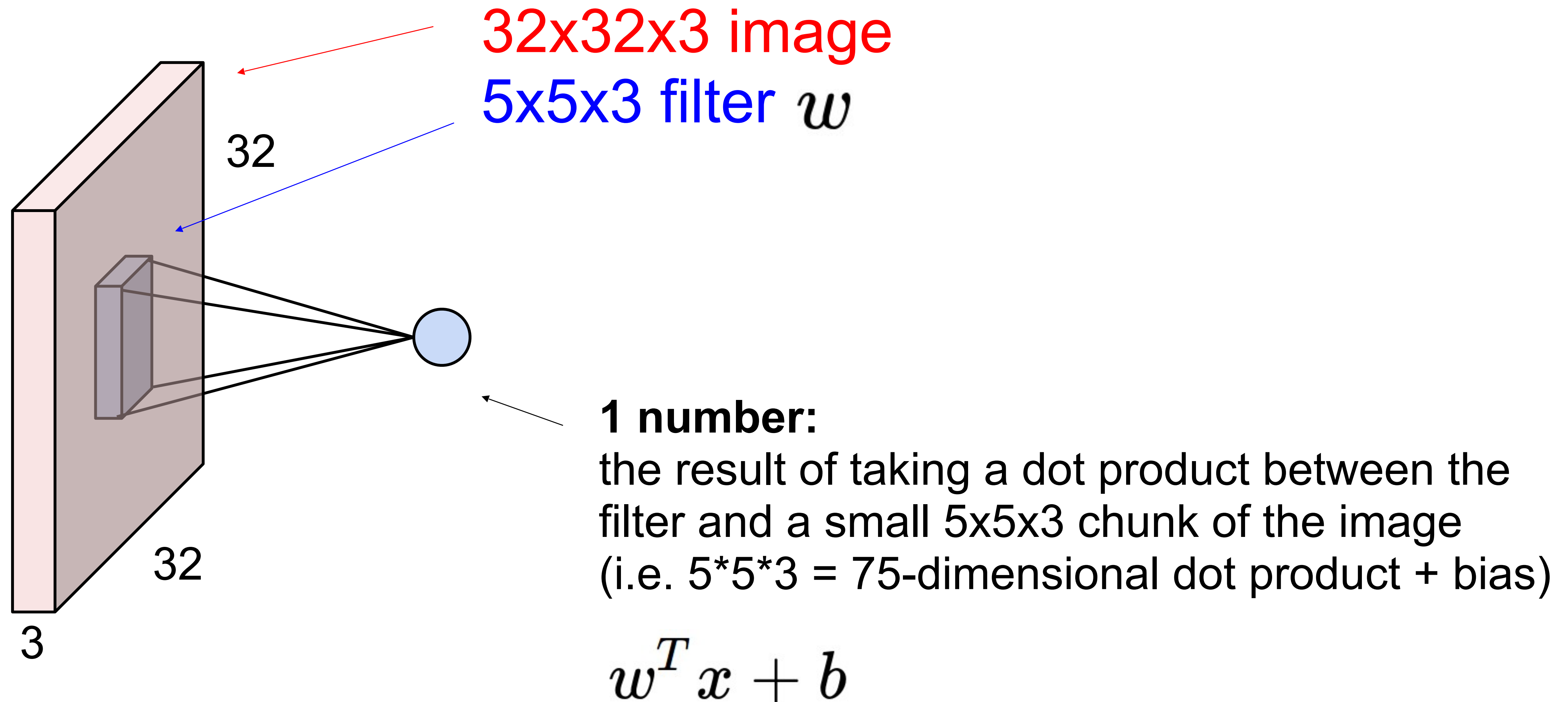
5x5x3 filter



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

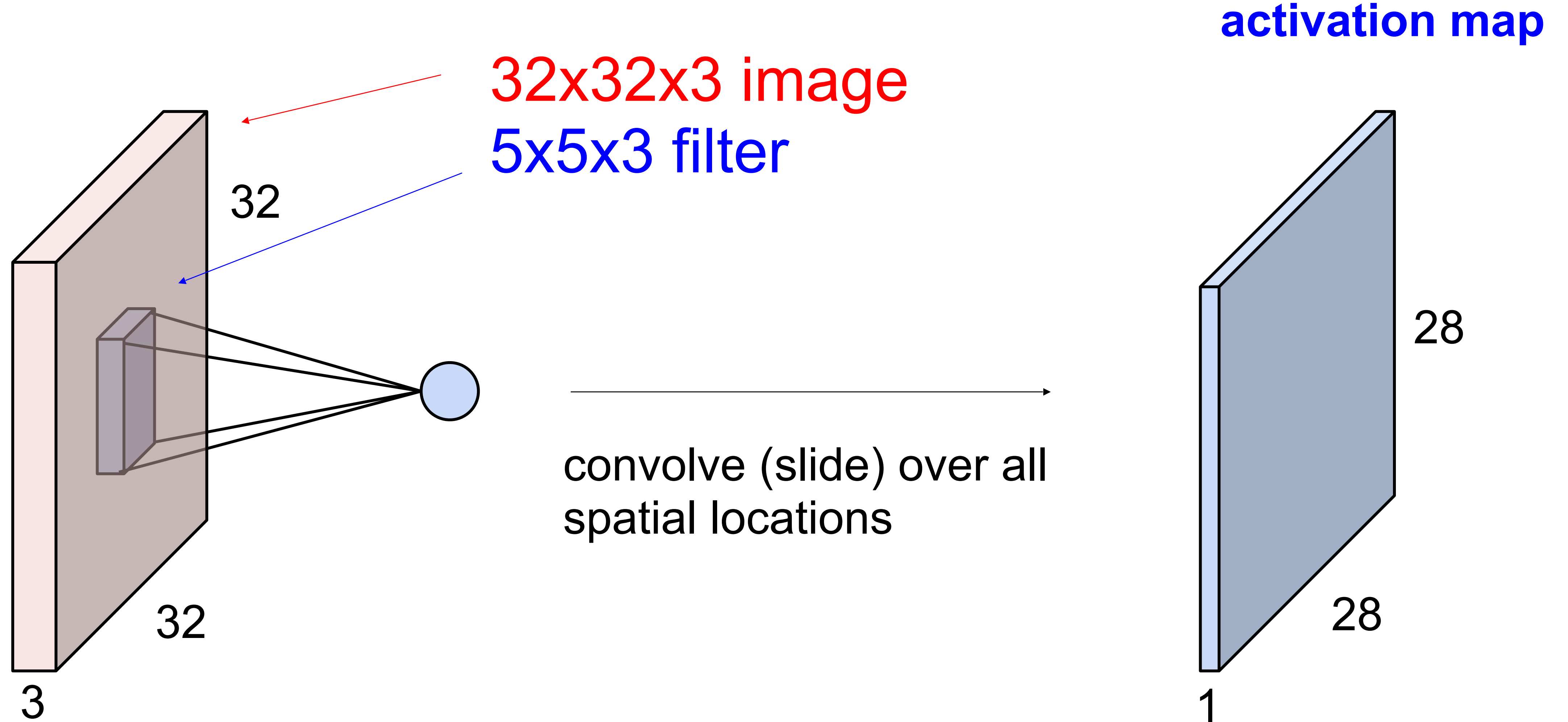


# Convolution Layer





# Convolution Layer





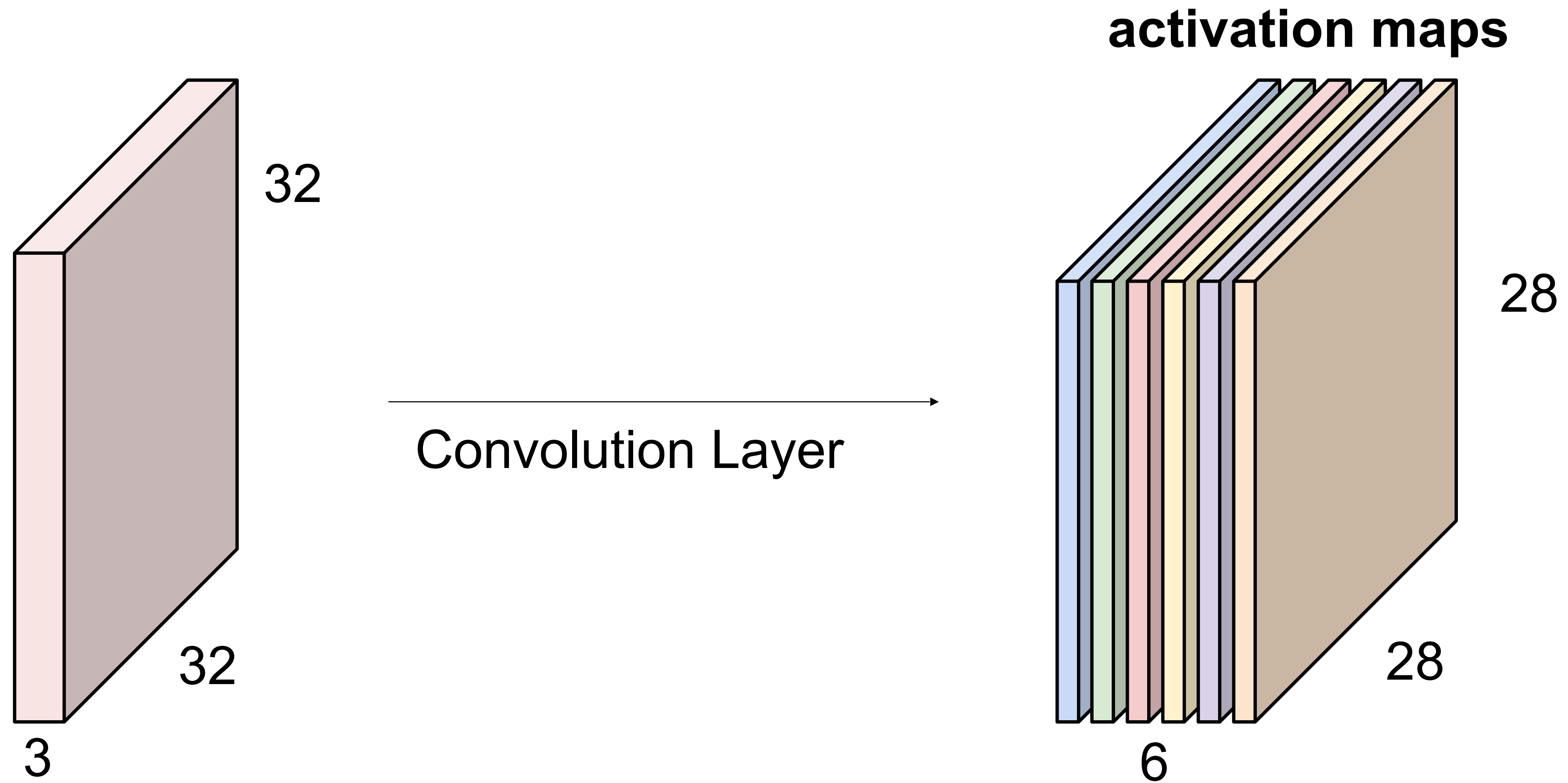
# Convolution Layer

consider a second, **green** filter





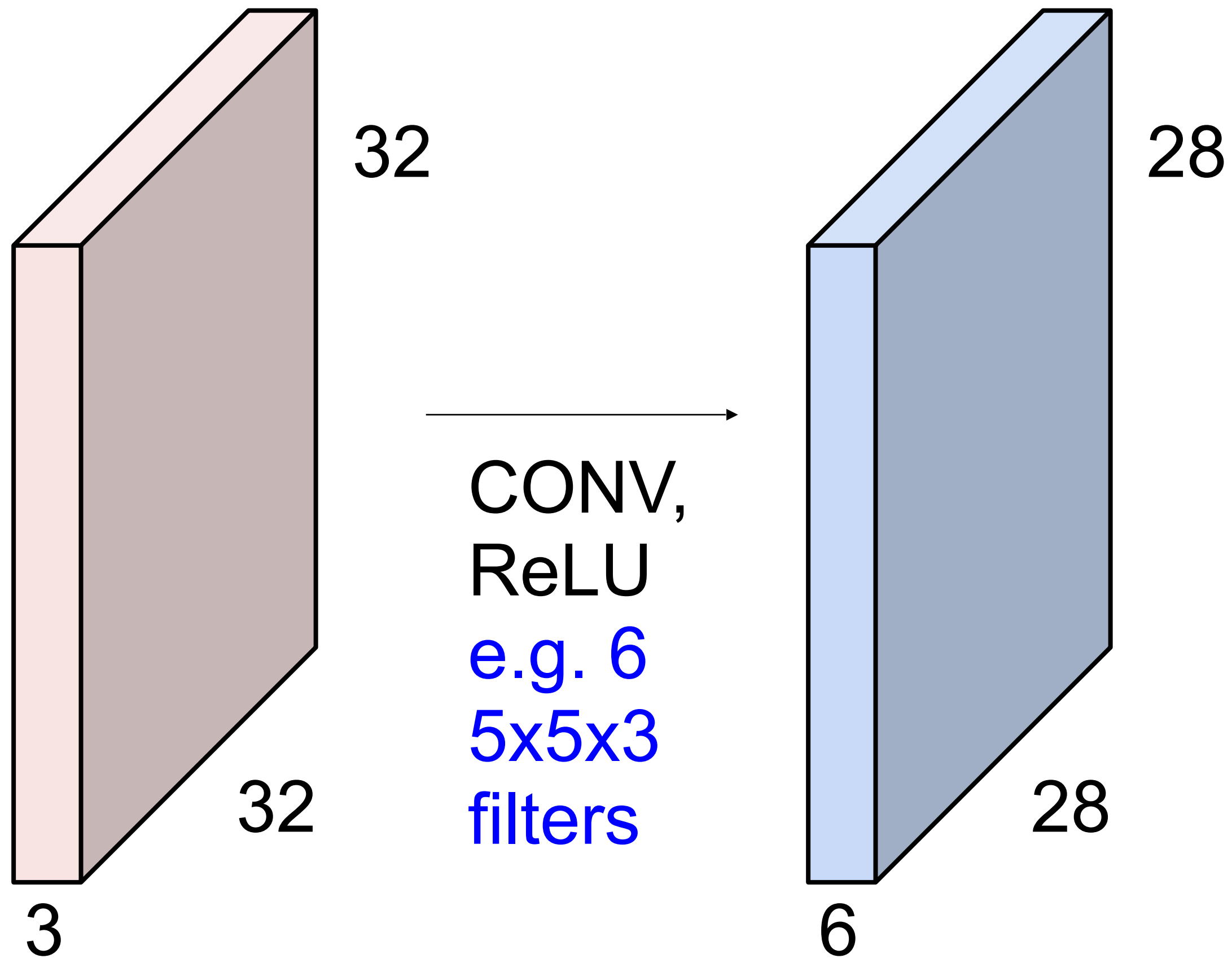
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

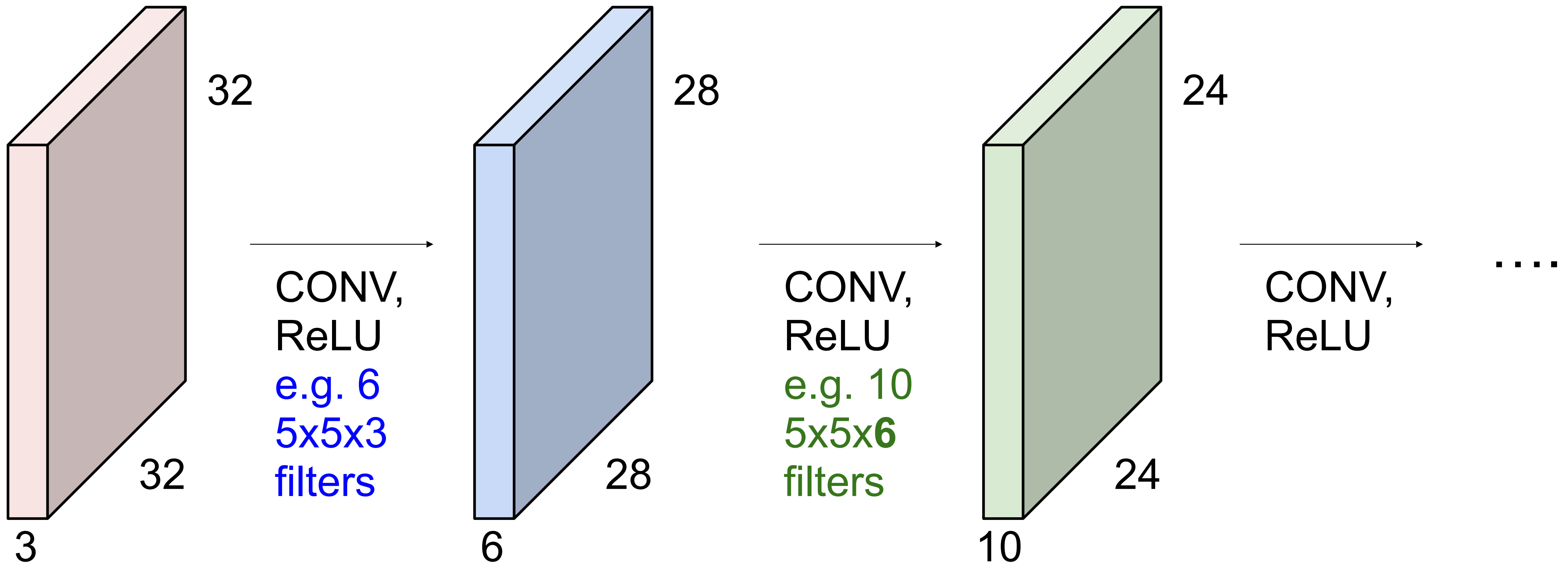


**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions





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

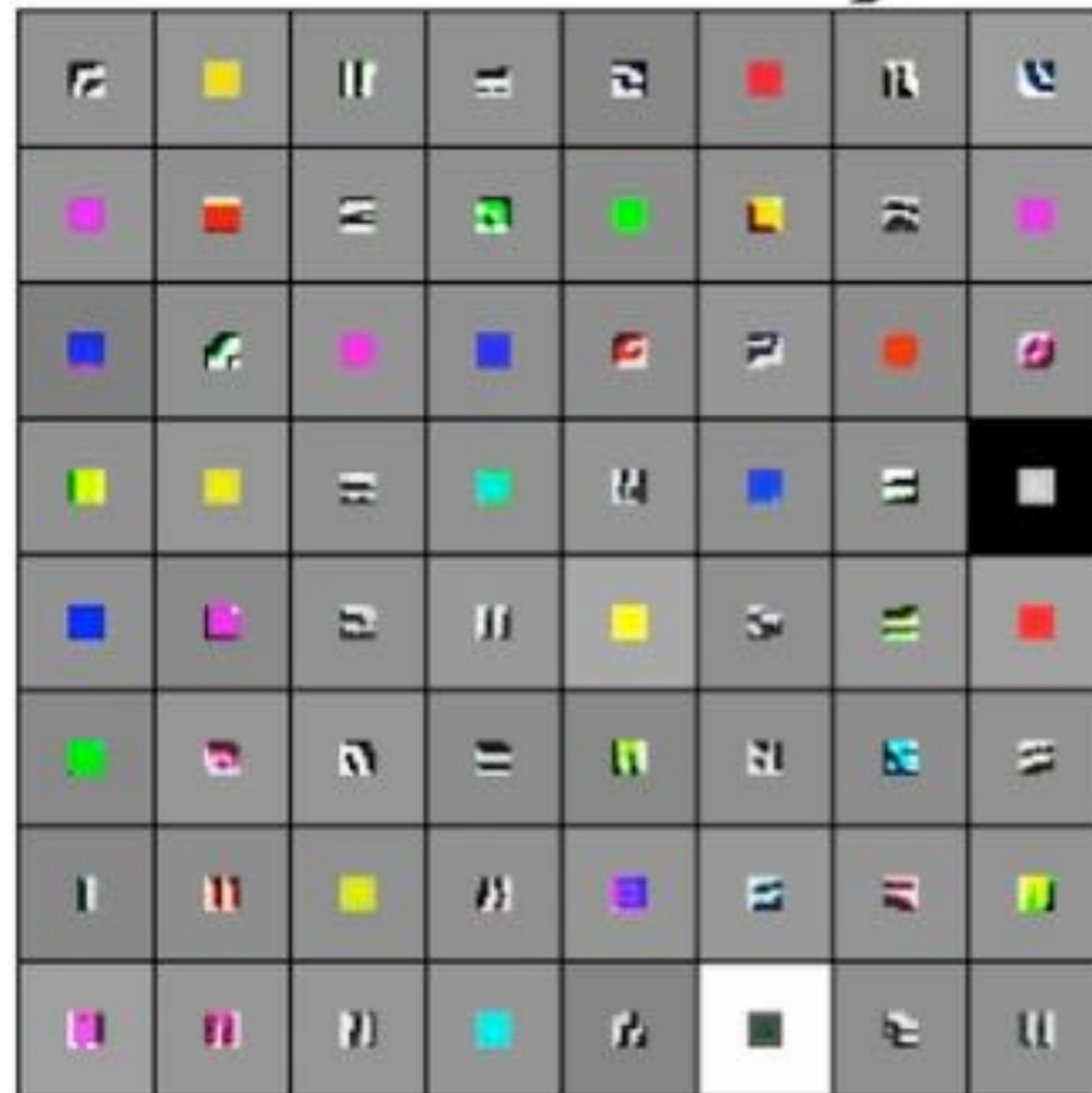
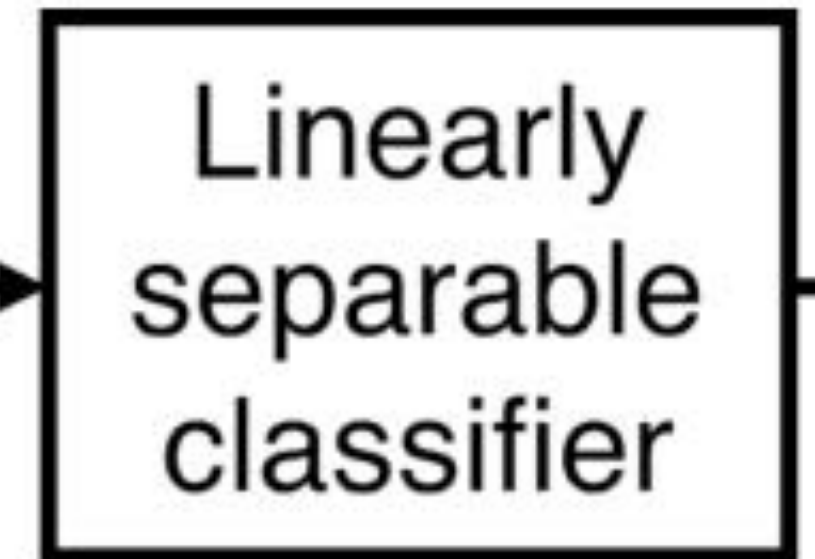




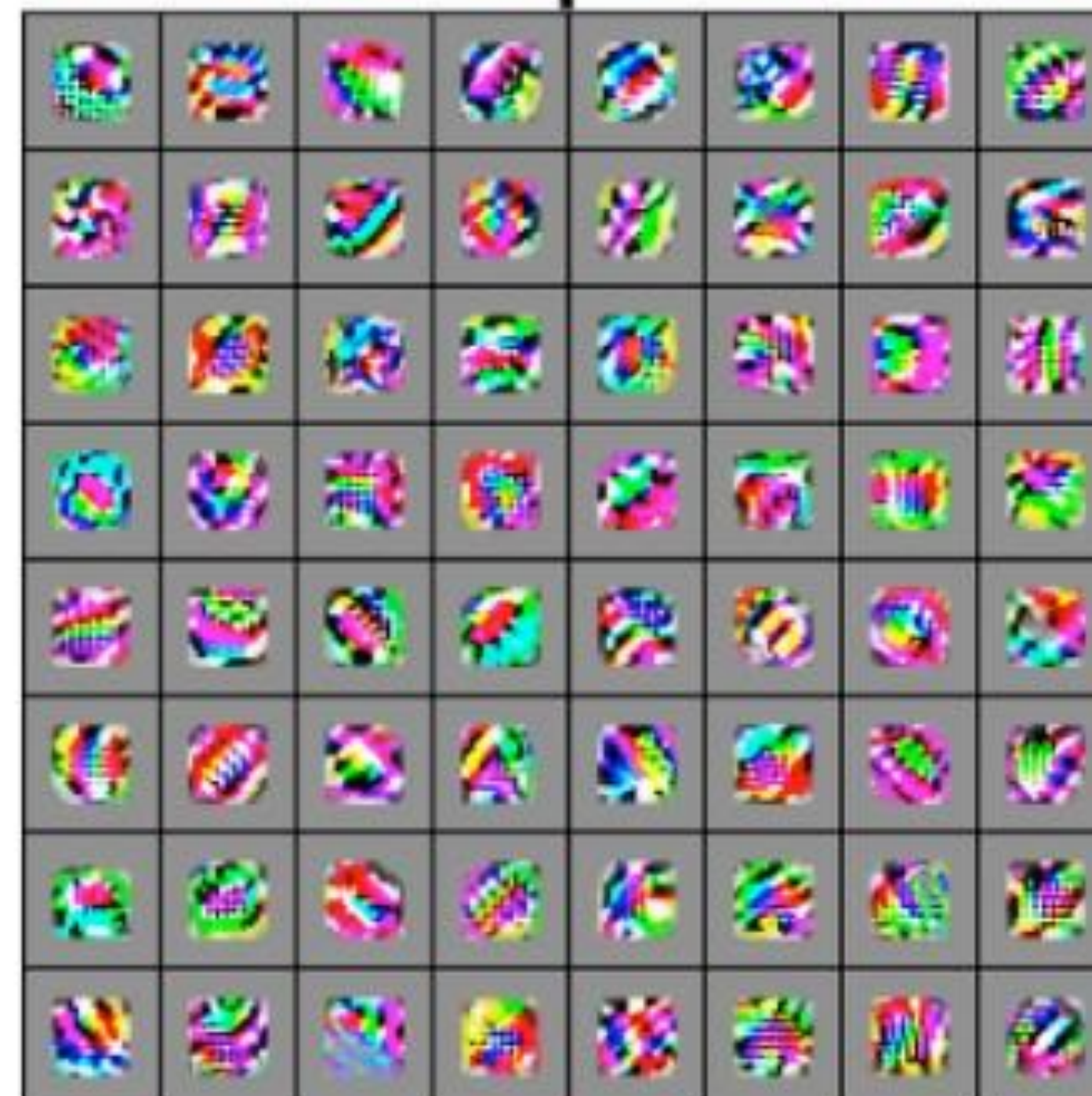
# Preview

*[Zeiler and Fergus 2013]*

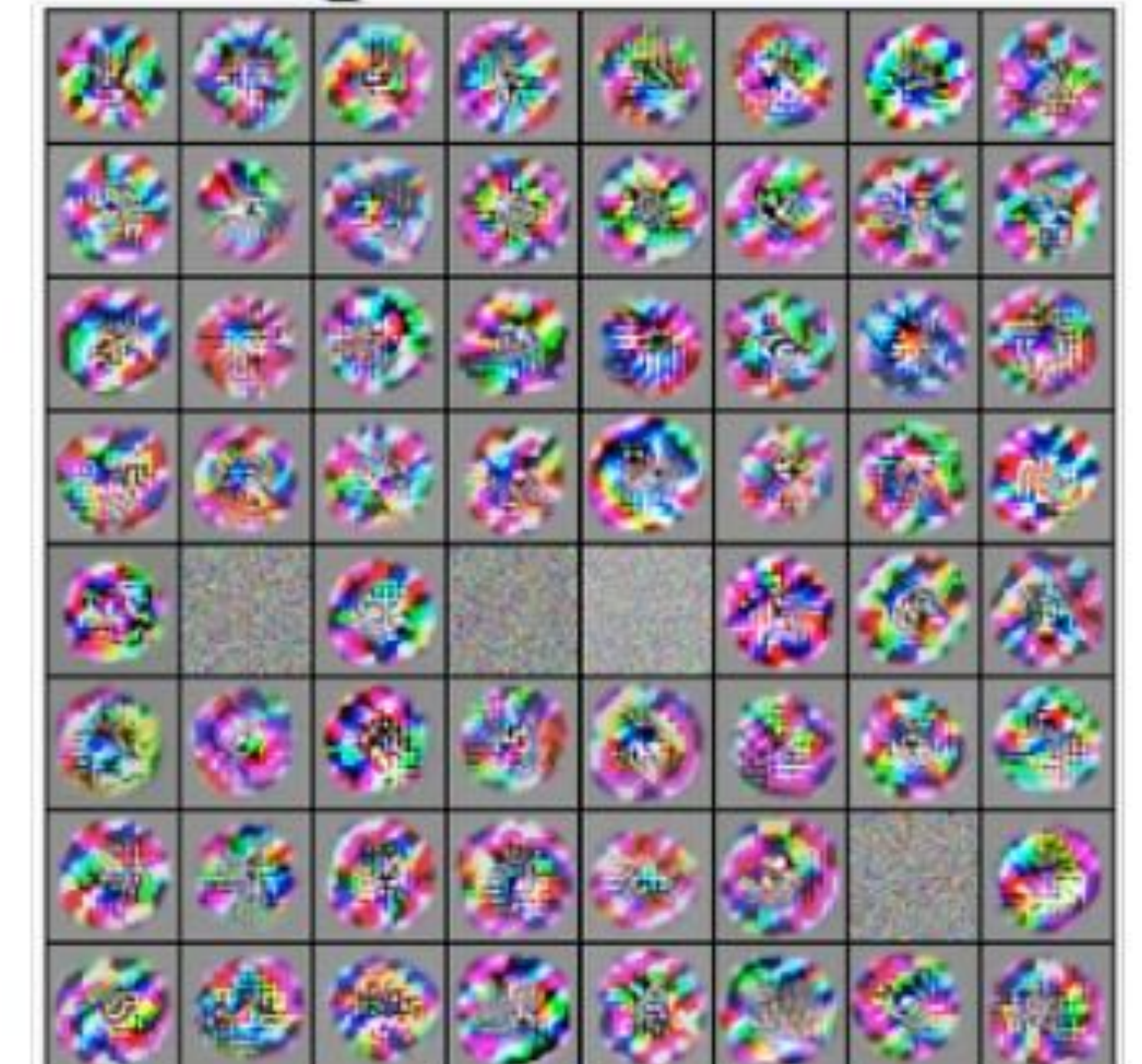
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



VGG-16 Conv1\_1



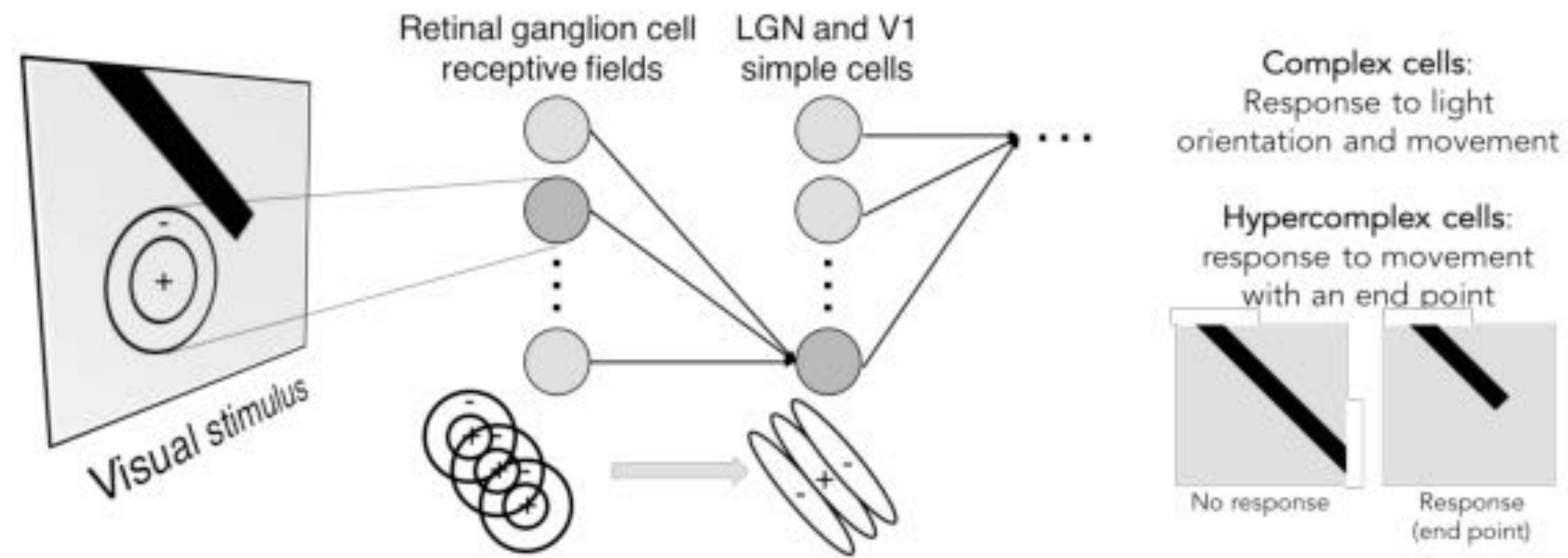
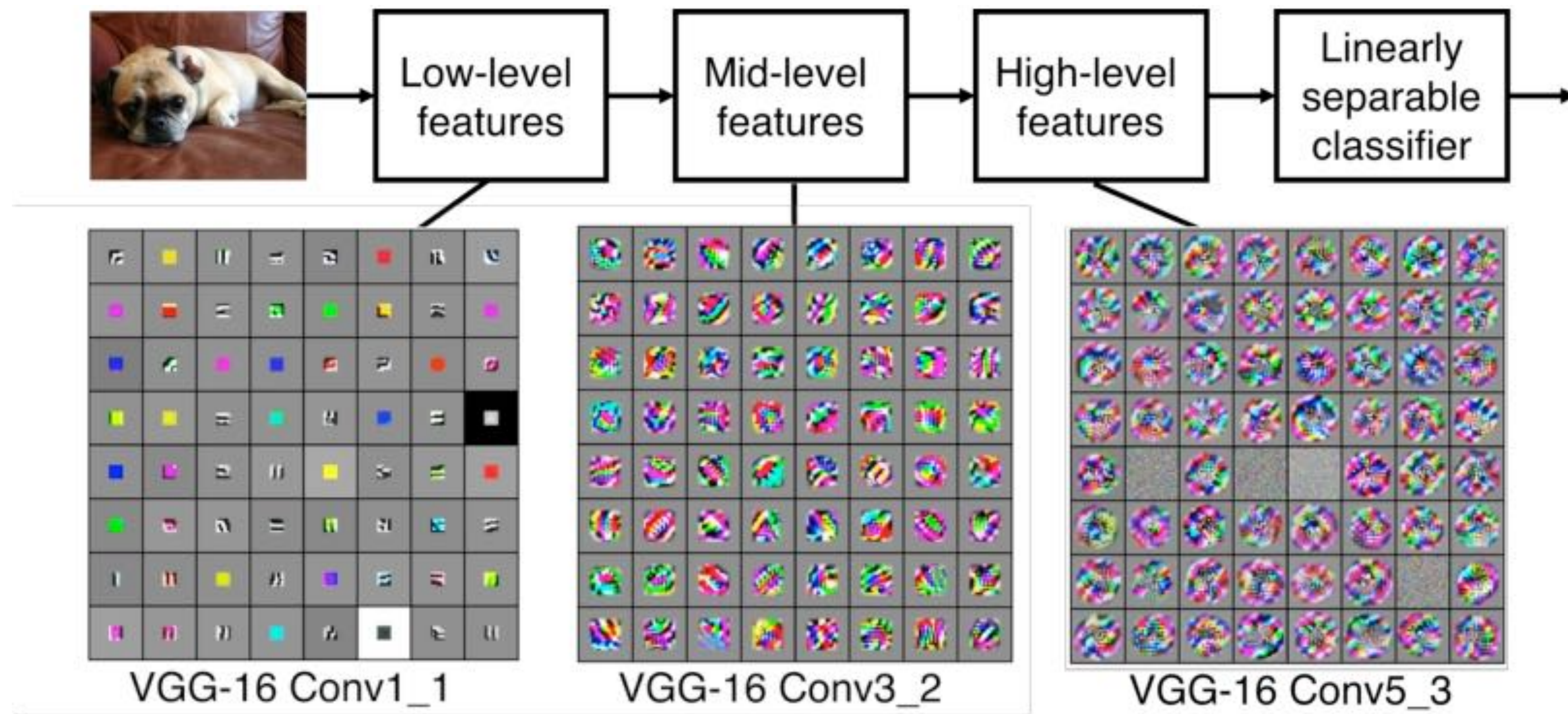
VGG-16 Conv3\_2



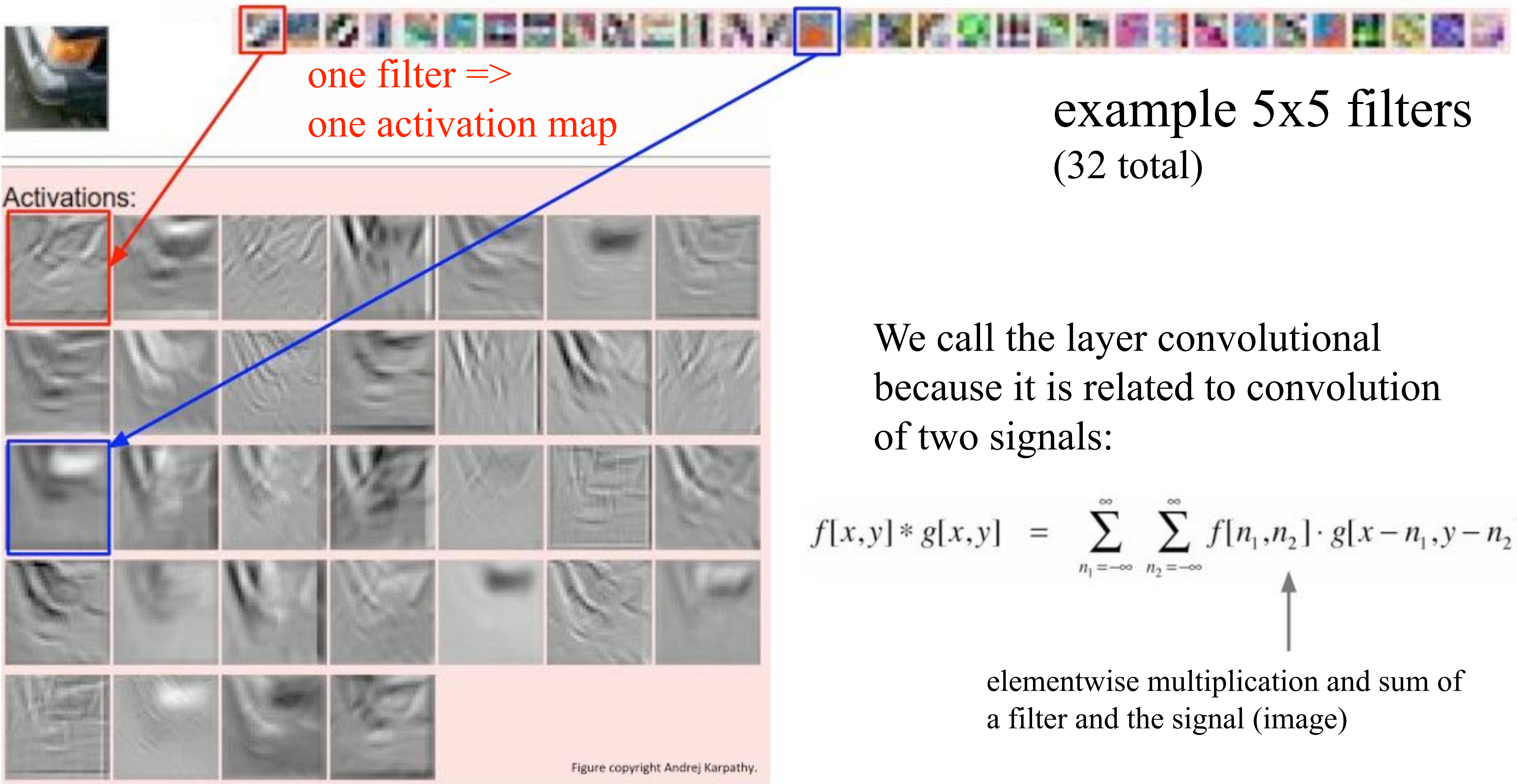
VGG-16 Conv5\_3



# Preview

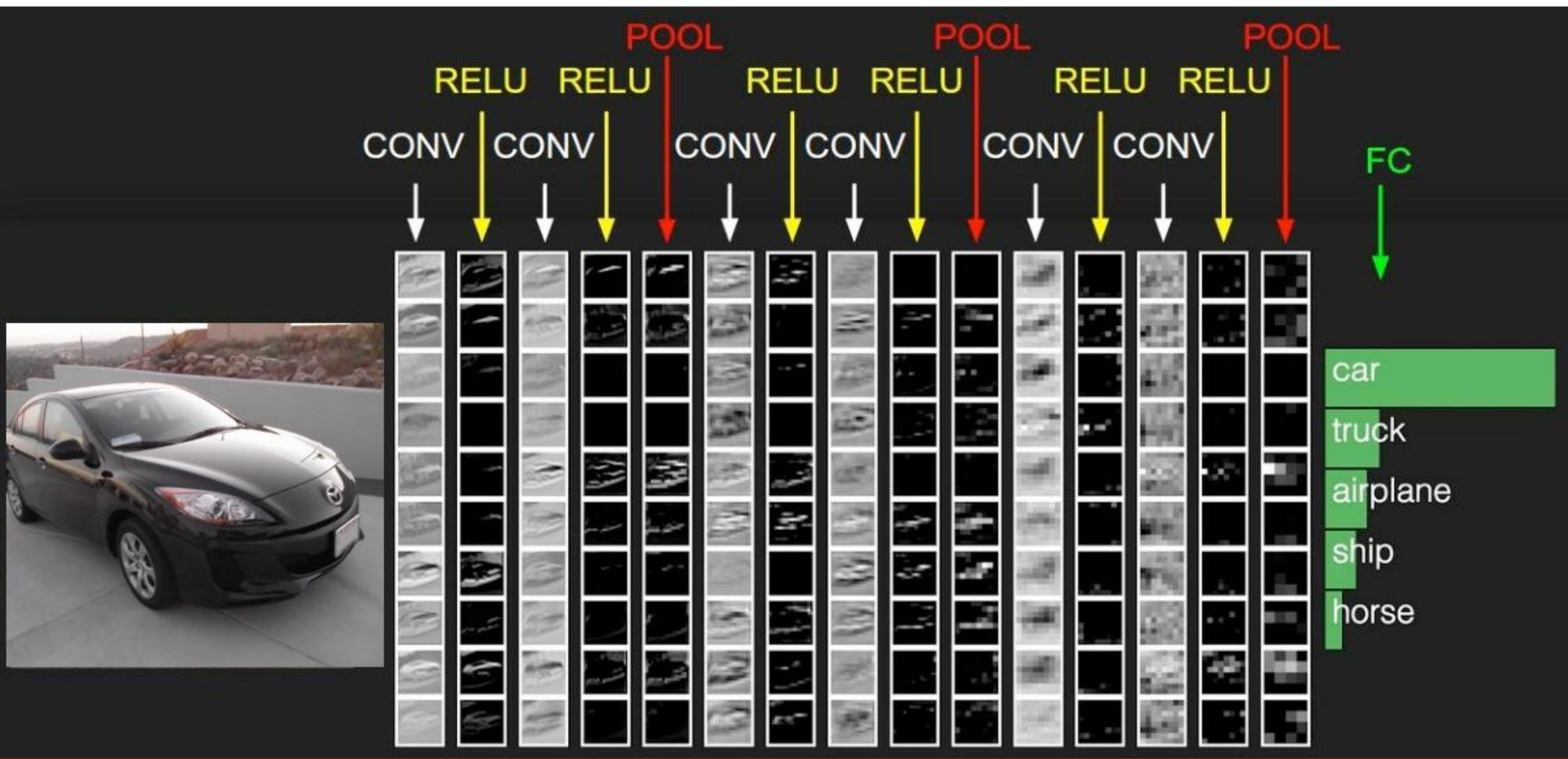








preview:

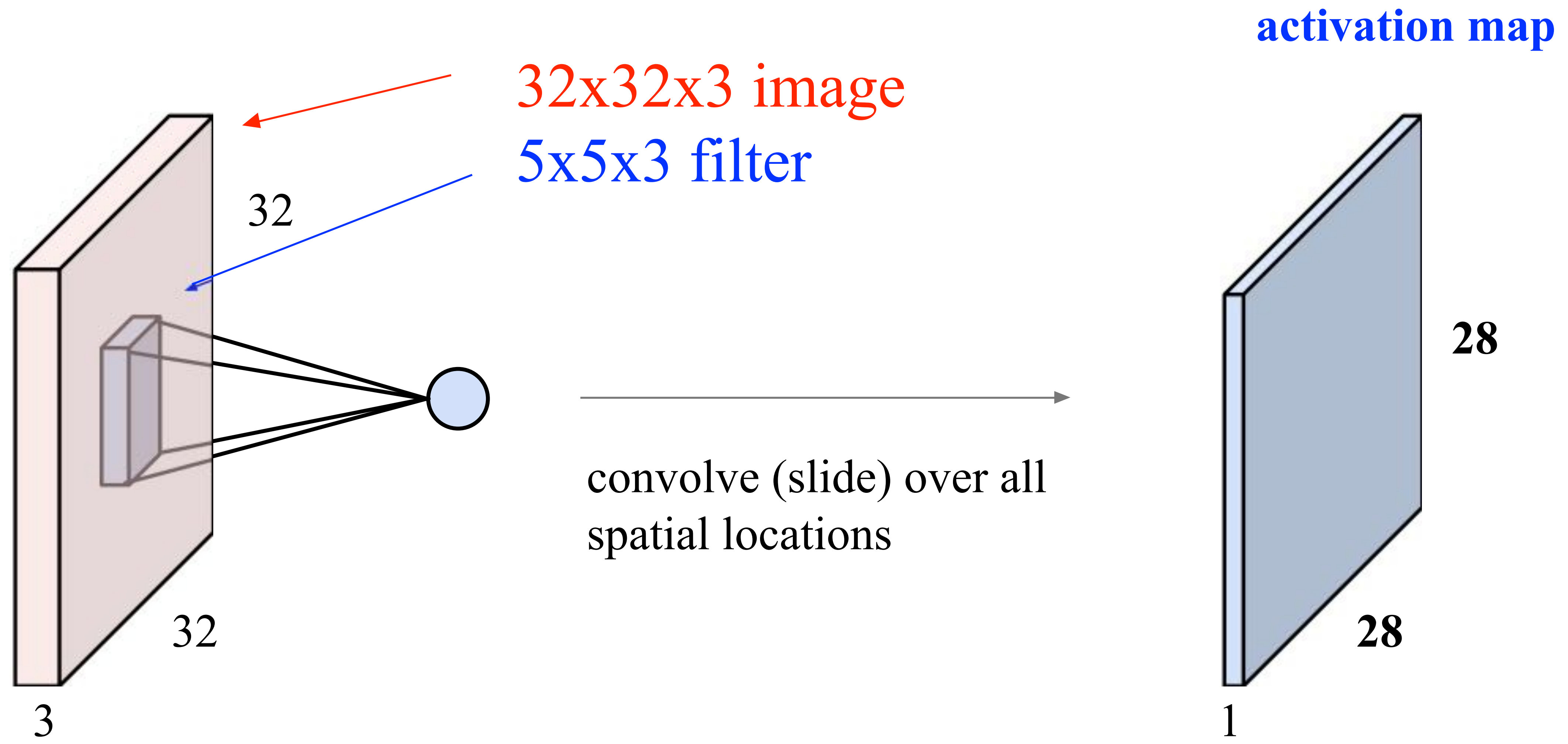




# Convolution Demo



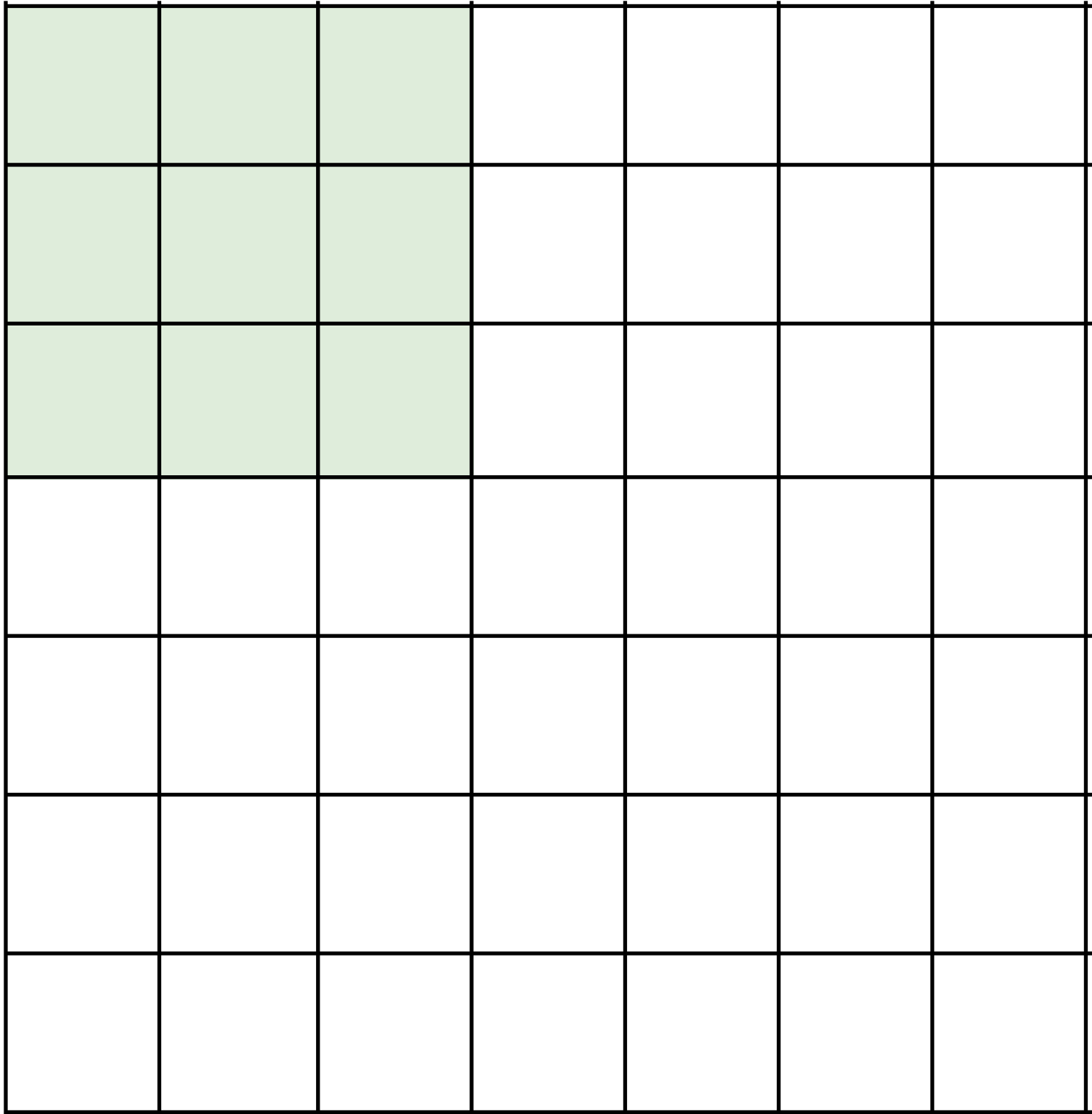
A closer look at spatial dimensions:





A closer look at spatial dimensions:

7



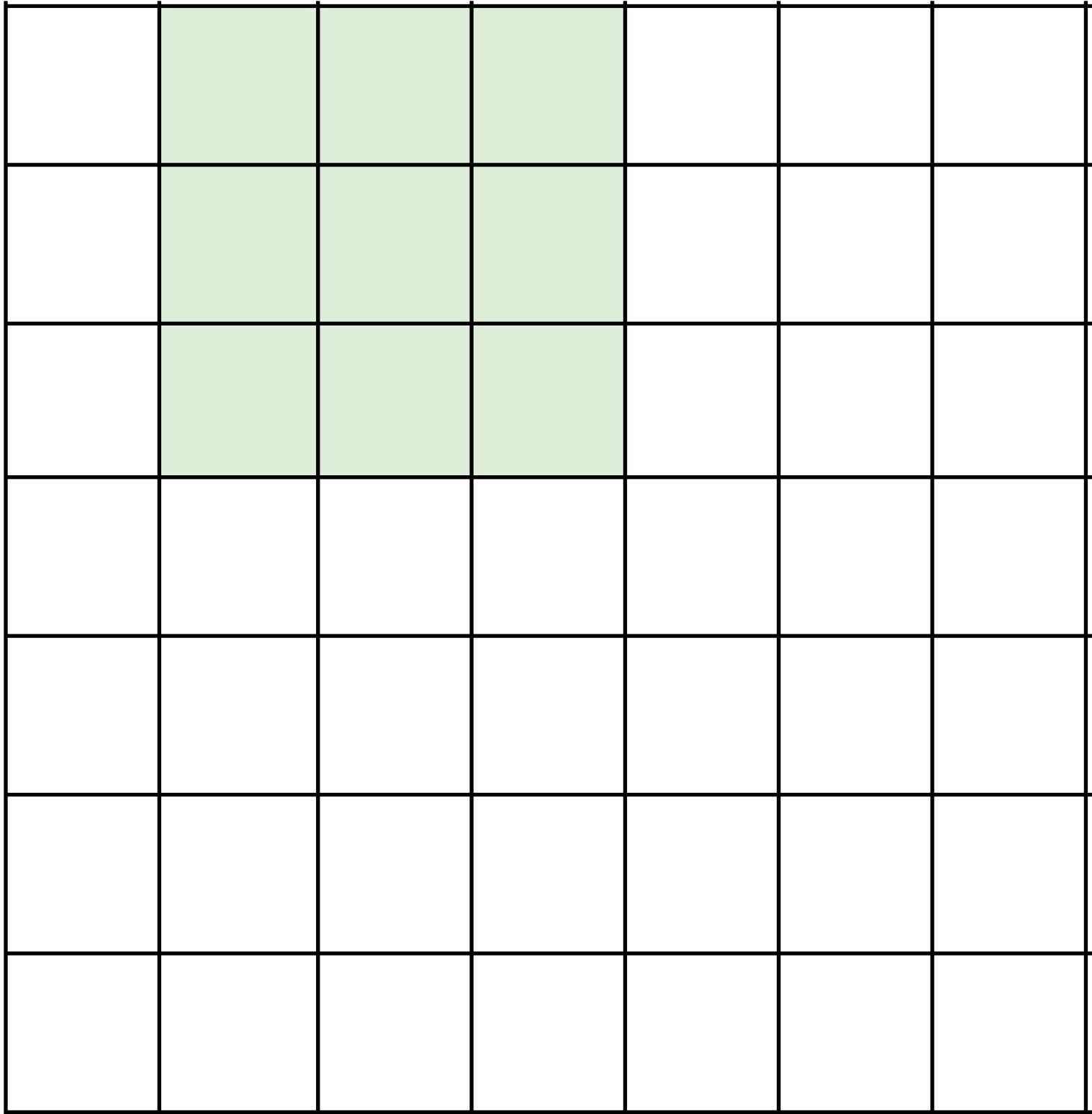
7x7 input (spatially)  
assume 3x3 filter

7



# A closer look at spatial dimensions:

7



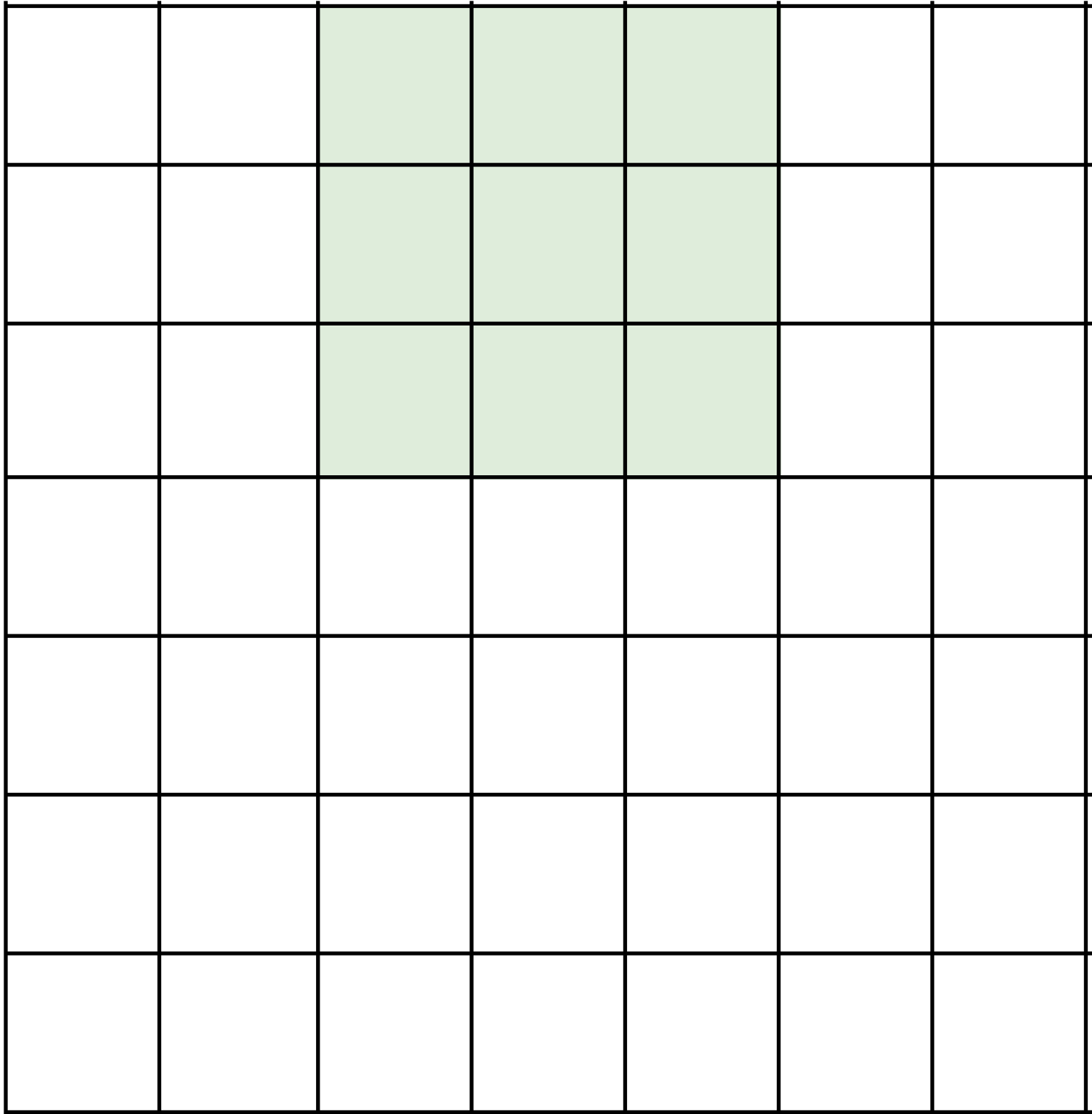
7x7 input (spatially)  
assume 3x3 filter

7



# A closer look at spatial dimensions:

7



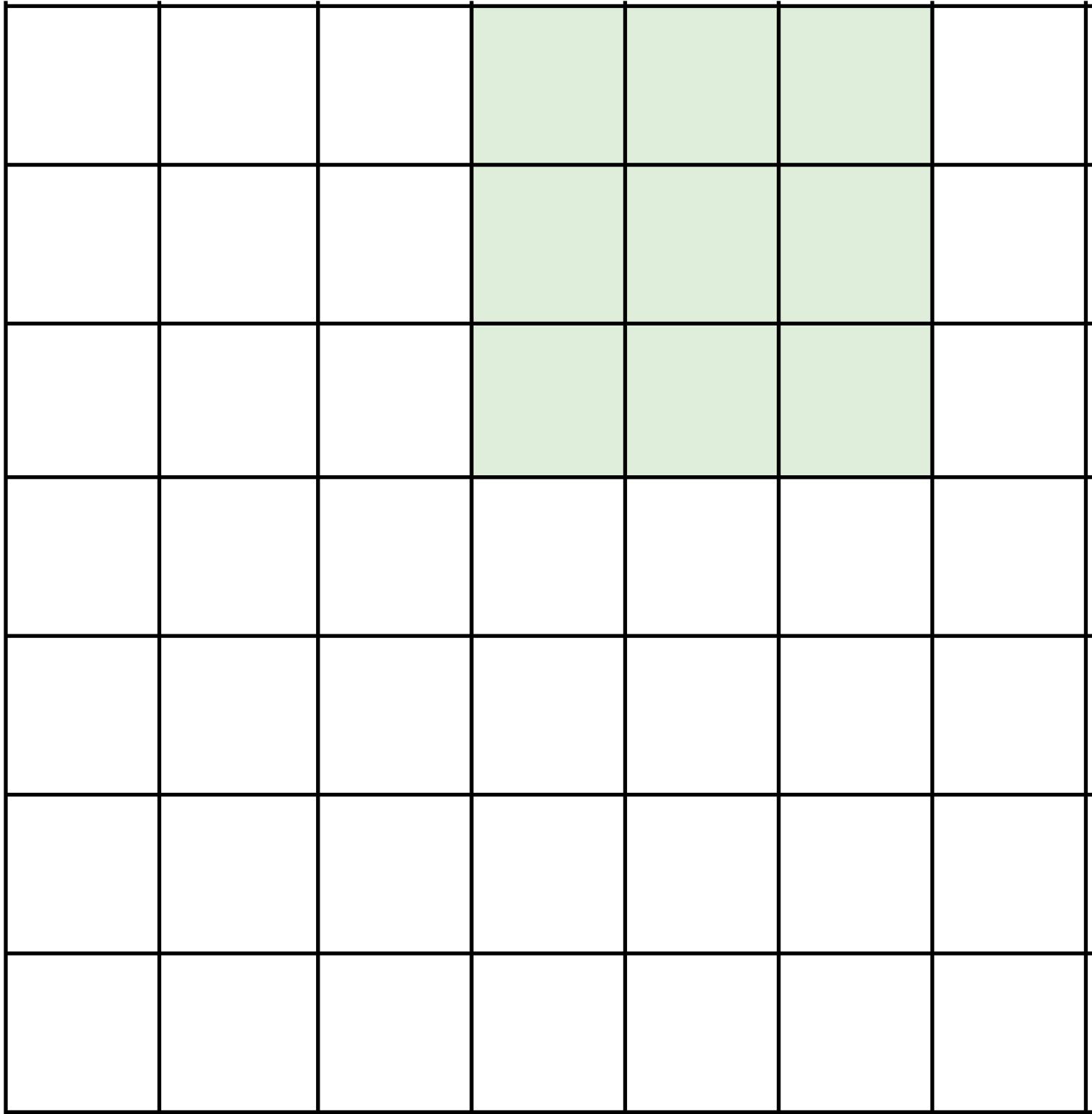
7x7 input (spatially)  
assume 3x3 filter

7



# A closer look at spatial dimensions:

7



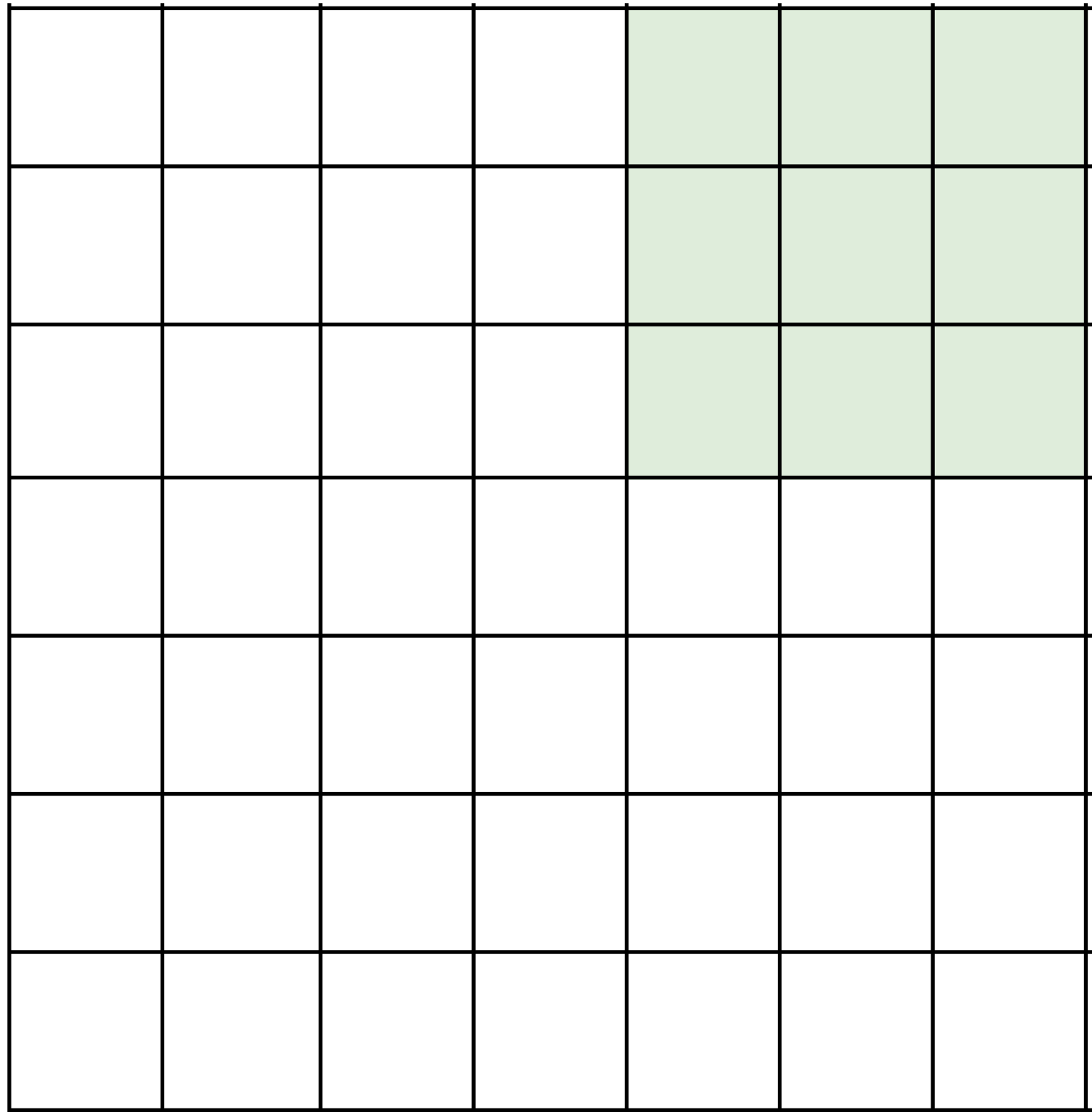
7x7 input (spatially)  
assume 3x3 filter

7



# A closer look at spatial dimensions:

7



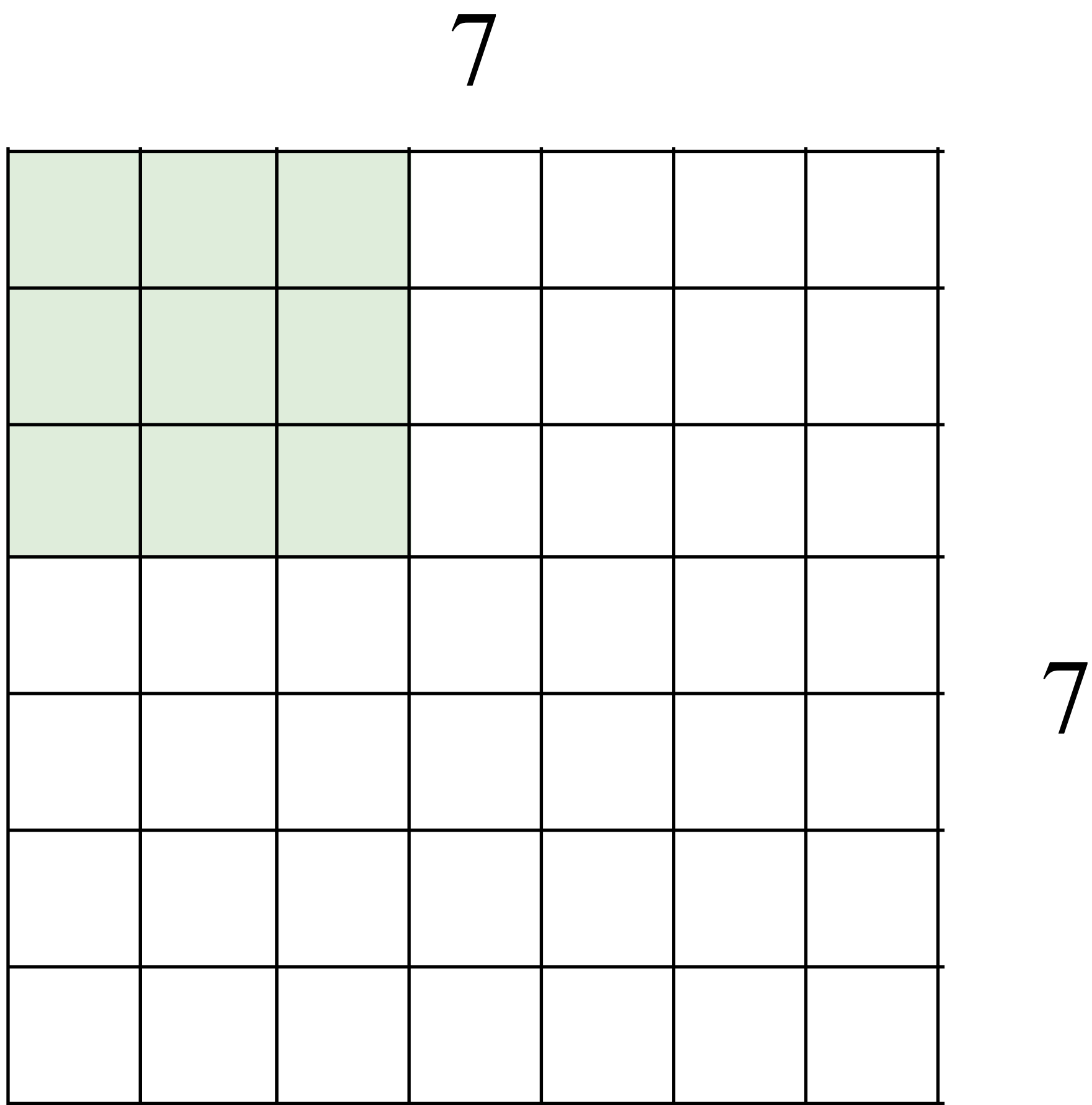
7x7 input (spatially)  
assume 3x3 filter

7

=> **5x5 output**



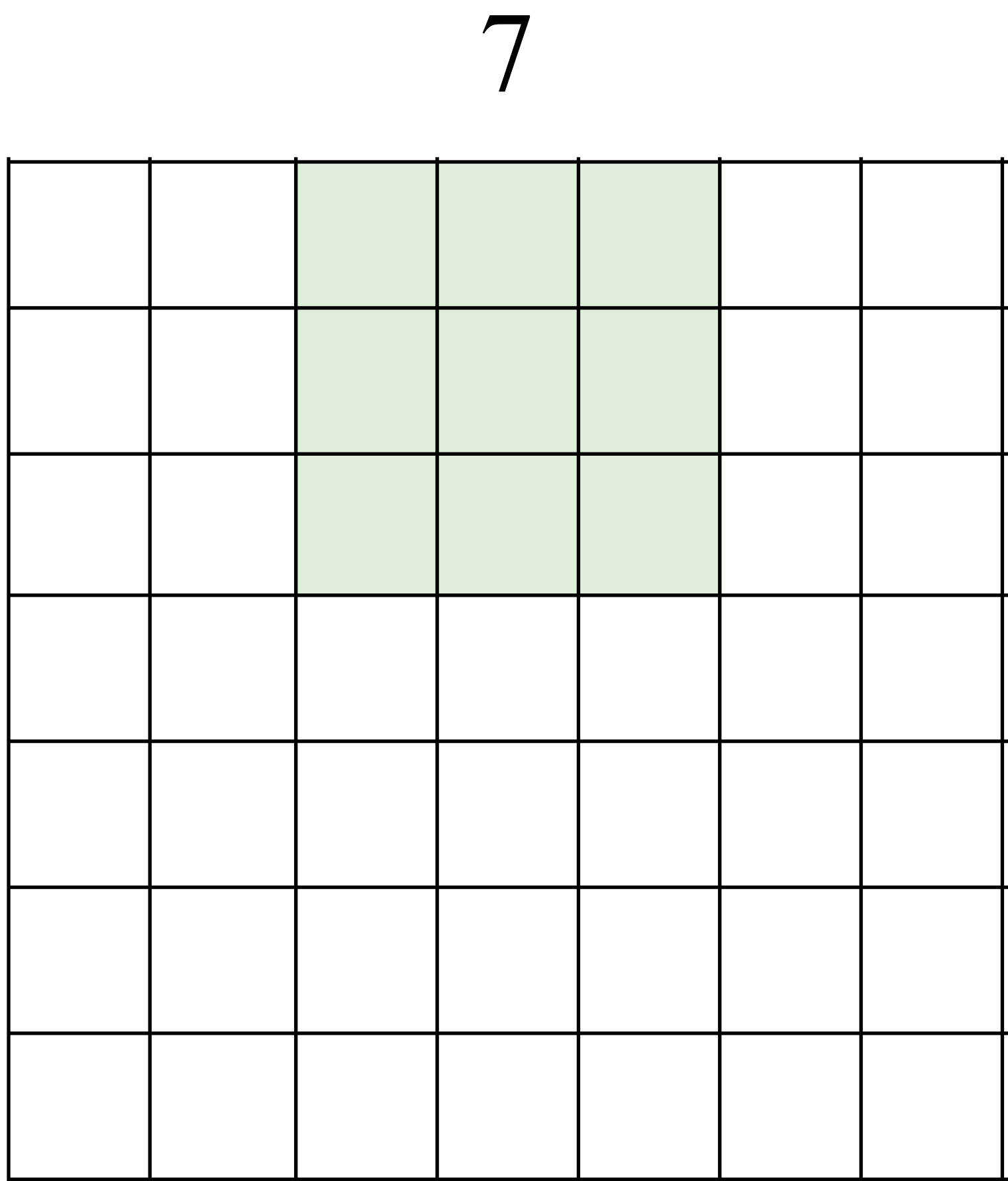
# A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter  
applied **with stride 2**



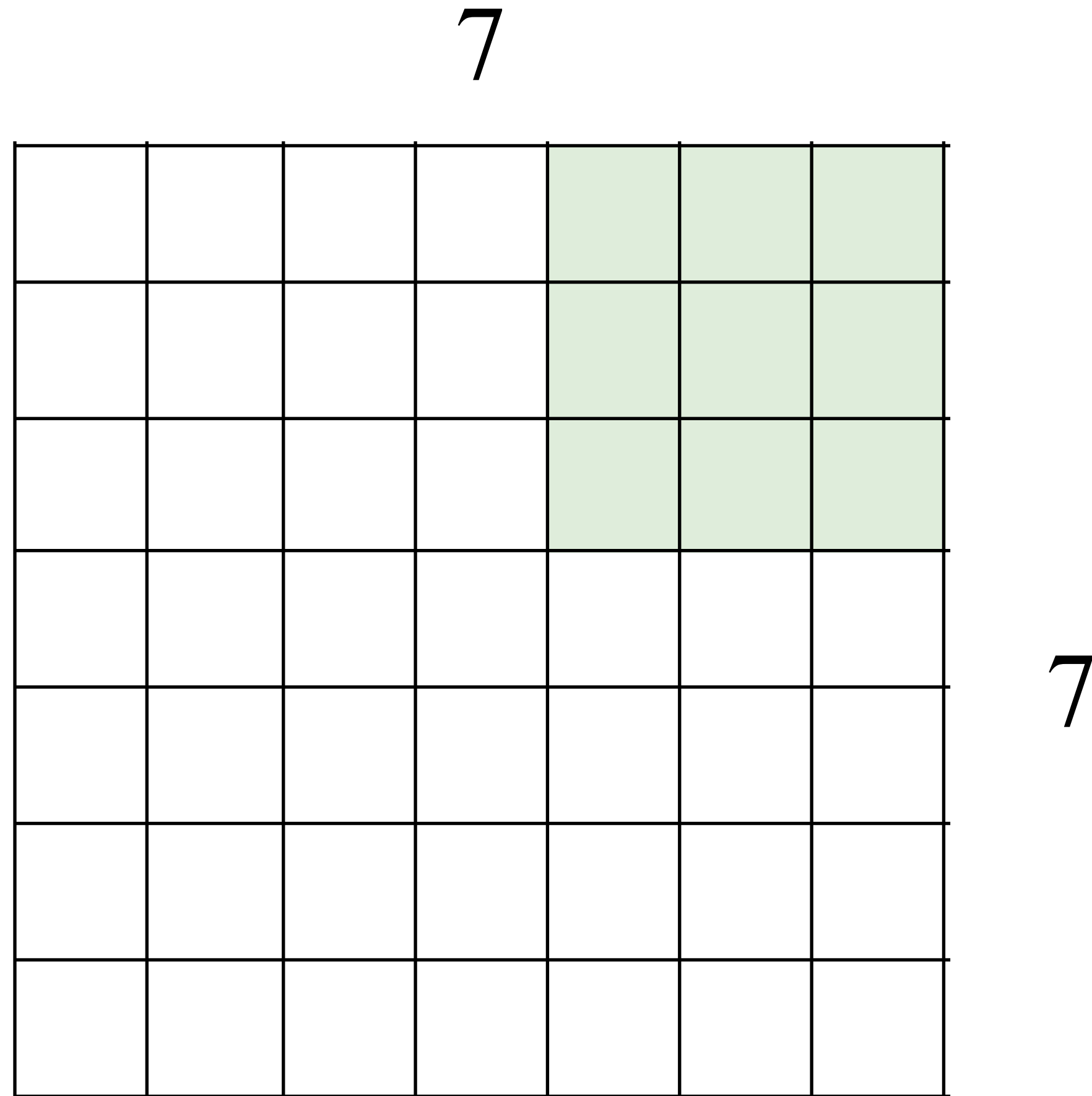
# A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter  
applied **with stride 2**



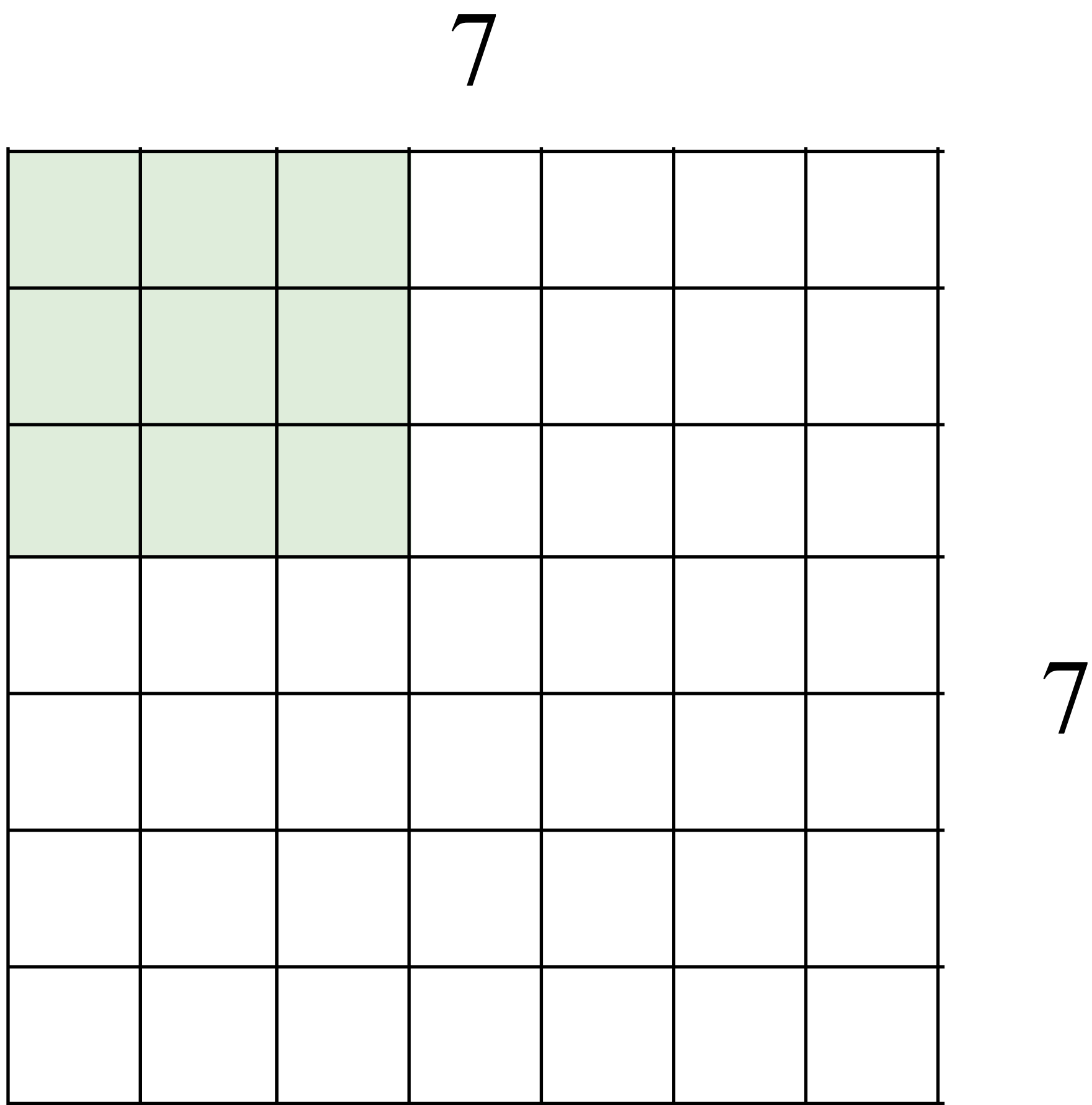
A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter  
applied **with stride 2**  
**=> 3x3 output!**

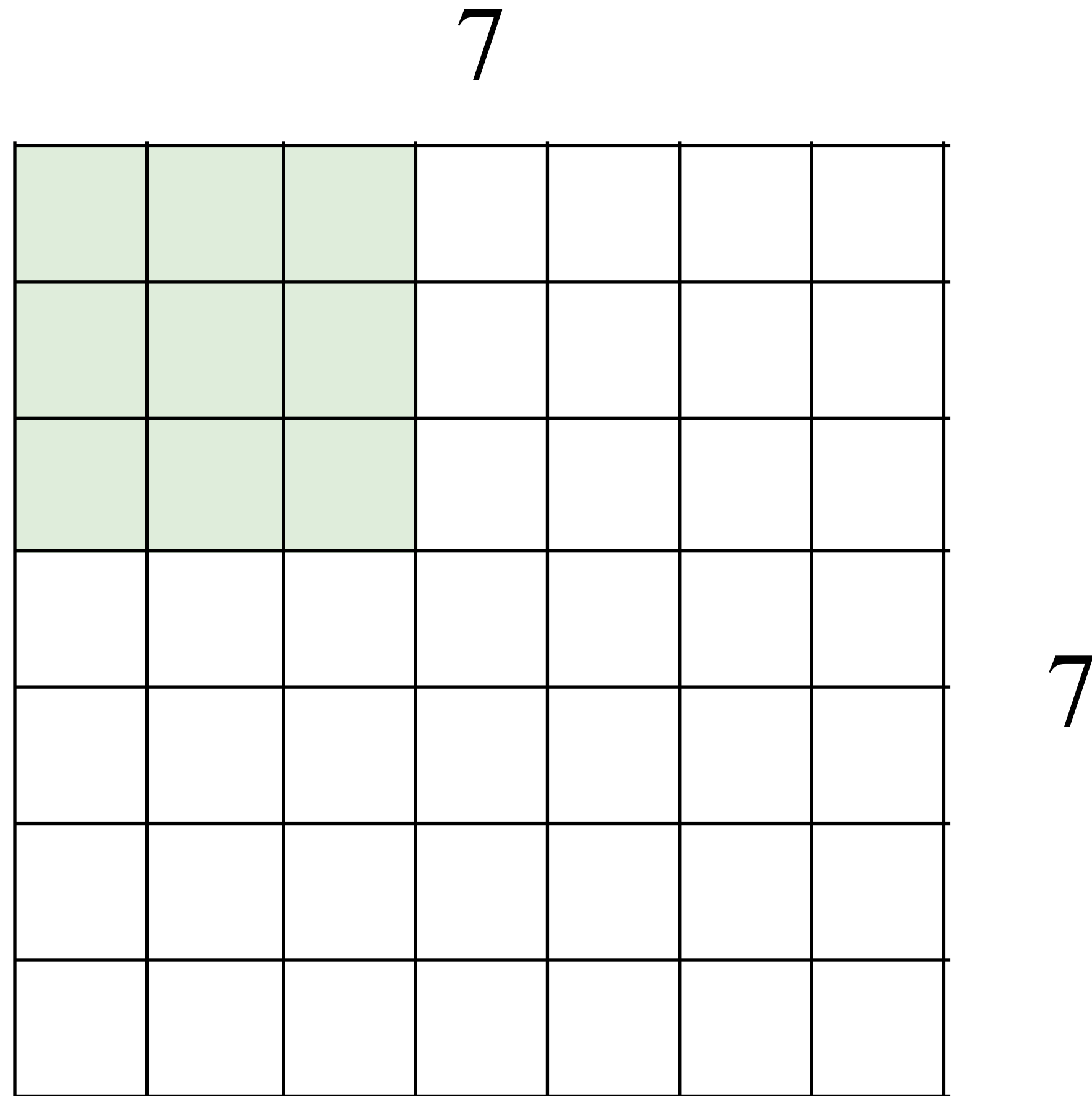


# A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter  
applied **with stride 3?**

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.



N


N

Output size:

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

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

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

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

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

# 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

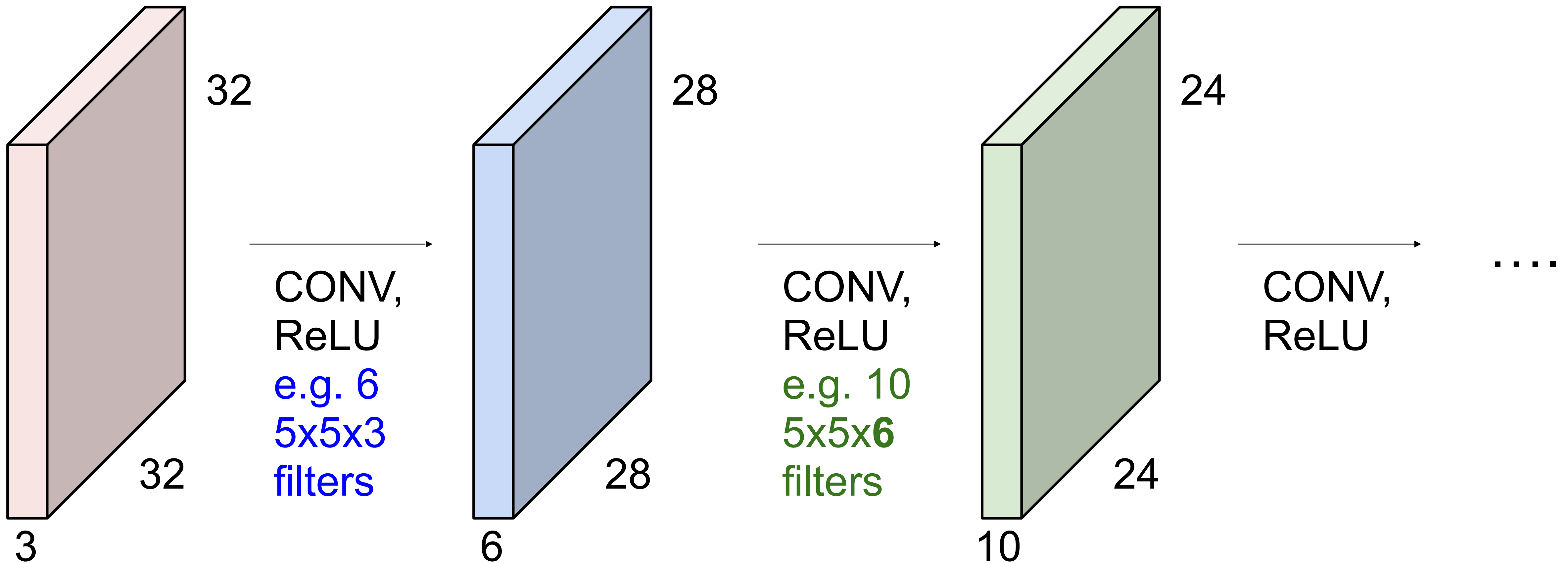
$F = 5 \Rightarrow$  zero pad with 2

$F = 7 \Rightarrow$  zero pad with 3



## Remember back to...

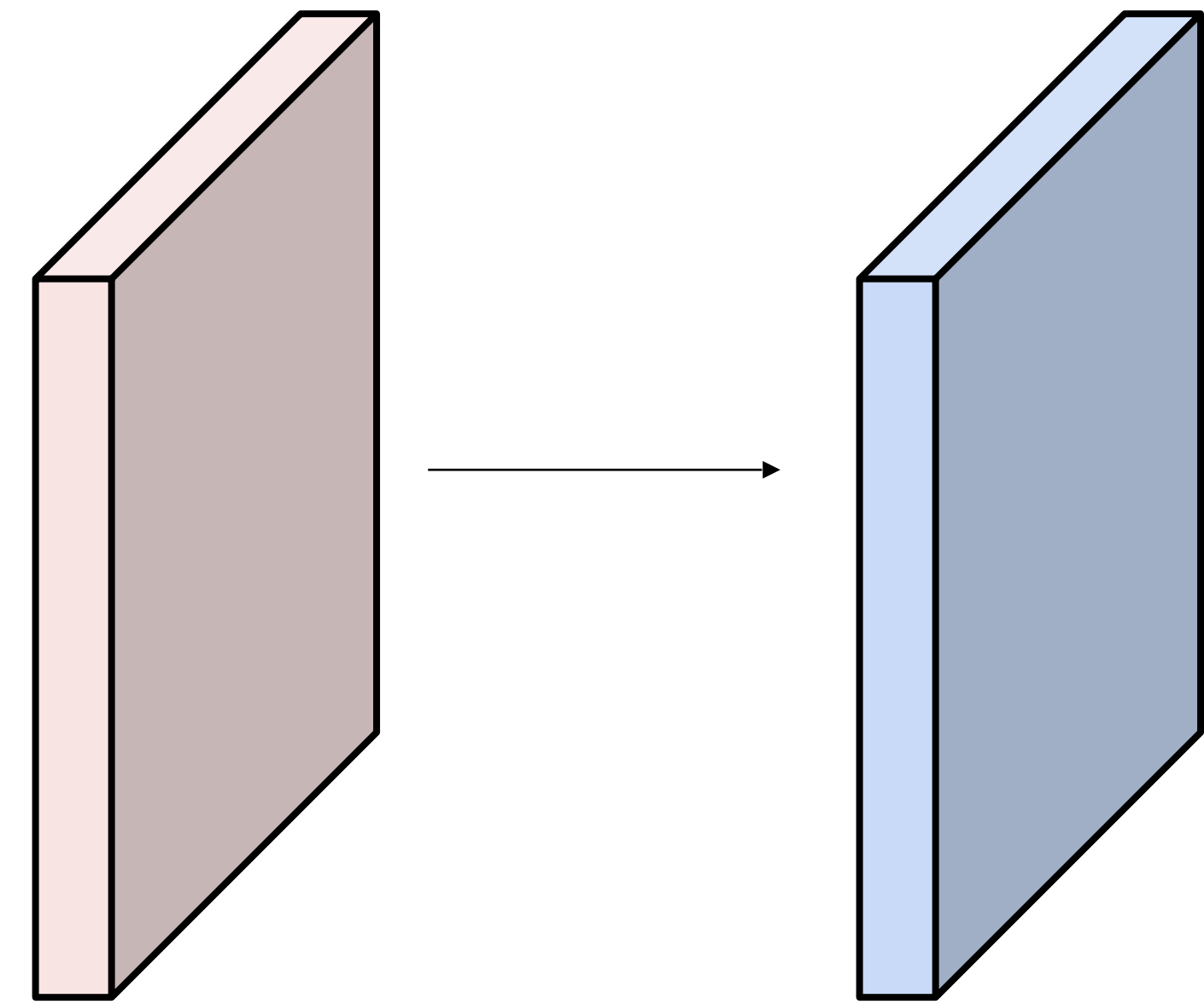
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32  $\rightarrow$  28  $\rightarrow$  24 ...). Shrinking too fast is not good, doesn't work well.



# Example

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

Output volume size: ?





# Example

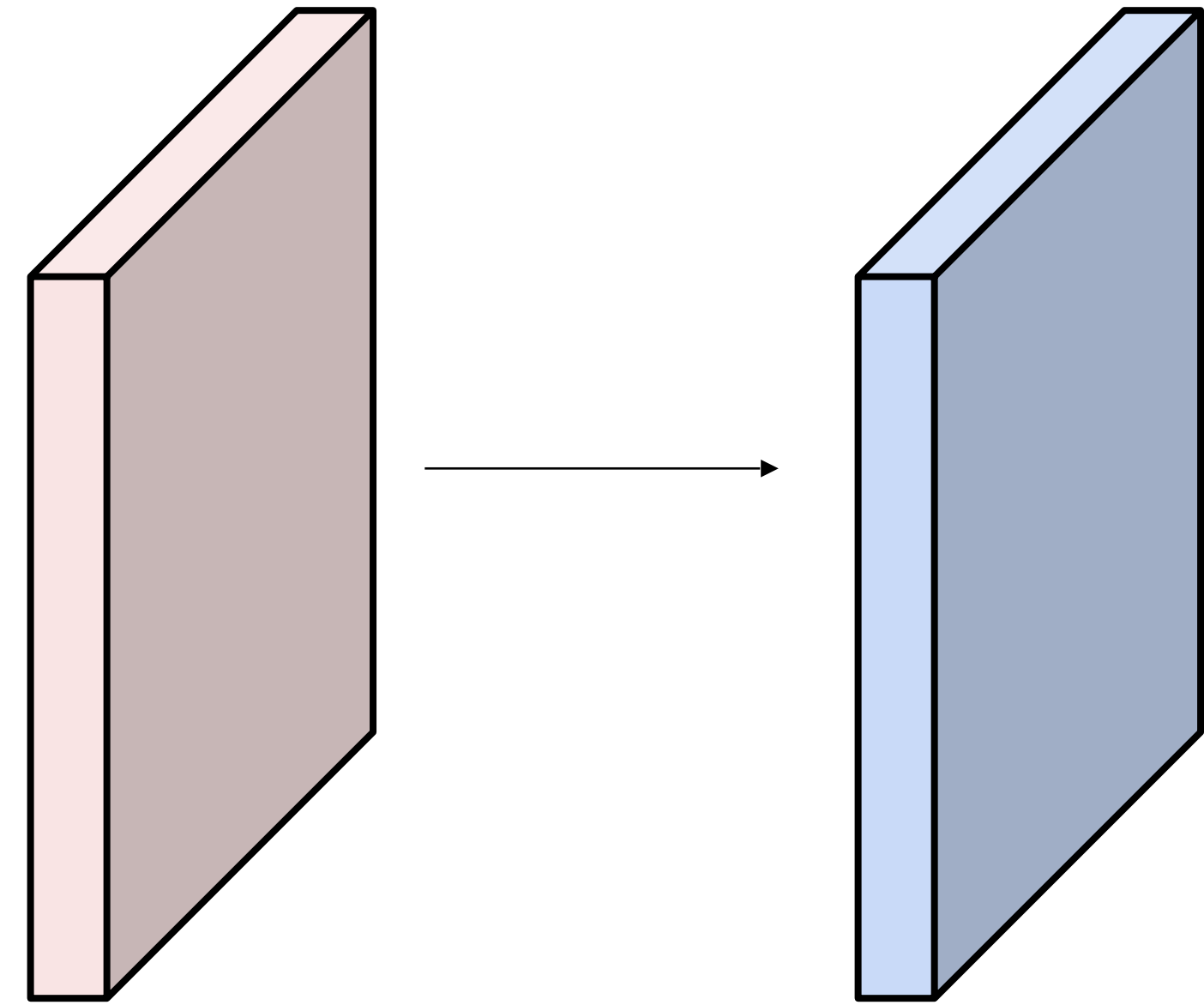
Input volume: **32x32x3**

**10** **5x5** filters with stride **1**, pad **2**

Output volume size:

$(32 + 2 * 2 - 5) / 1 + 1 = 32$  spatially, so

**32x32x10**

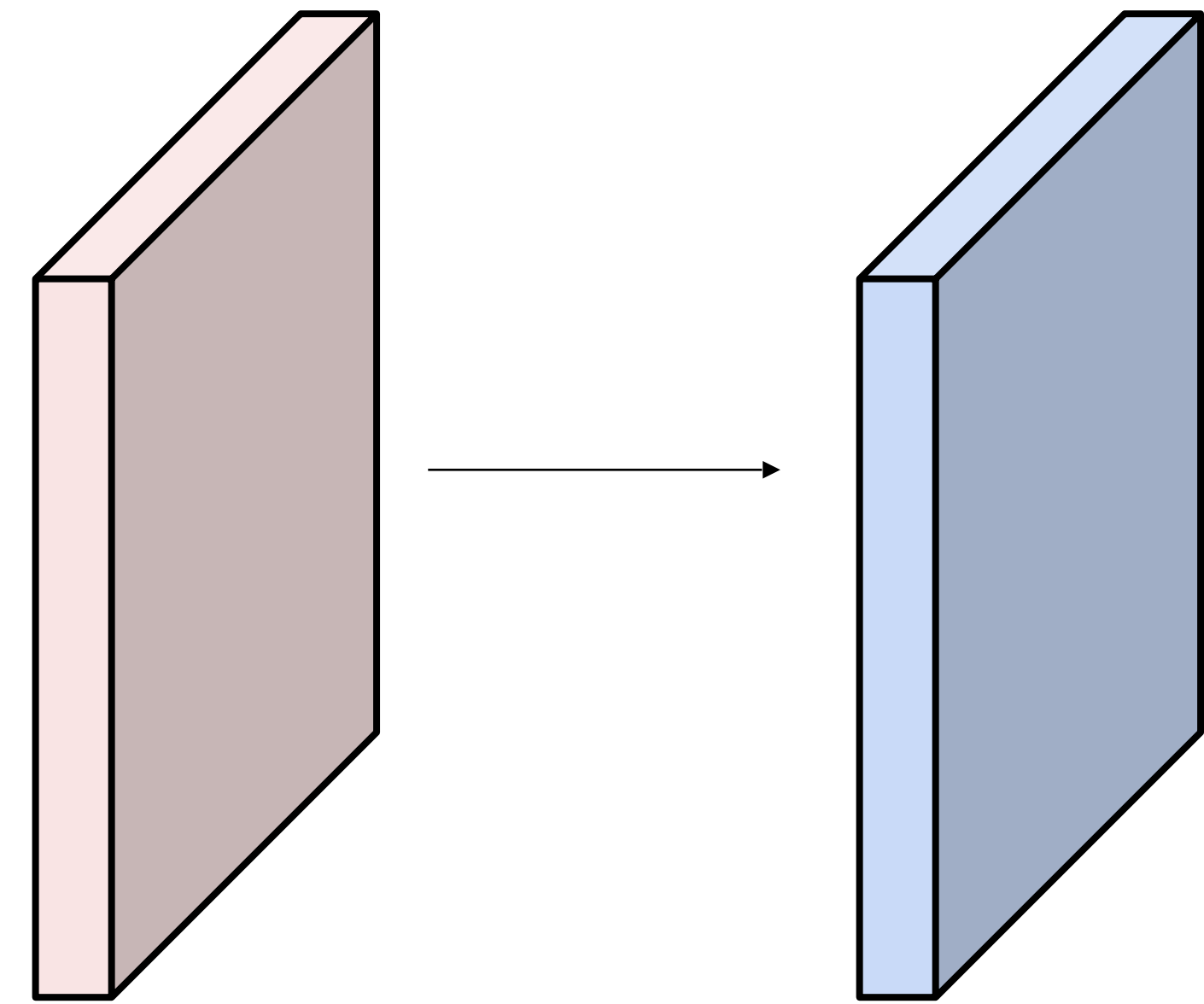


# Example

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?





# Example

Input volume: **32x32x3**

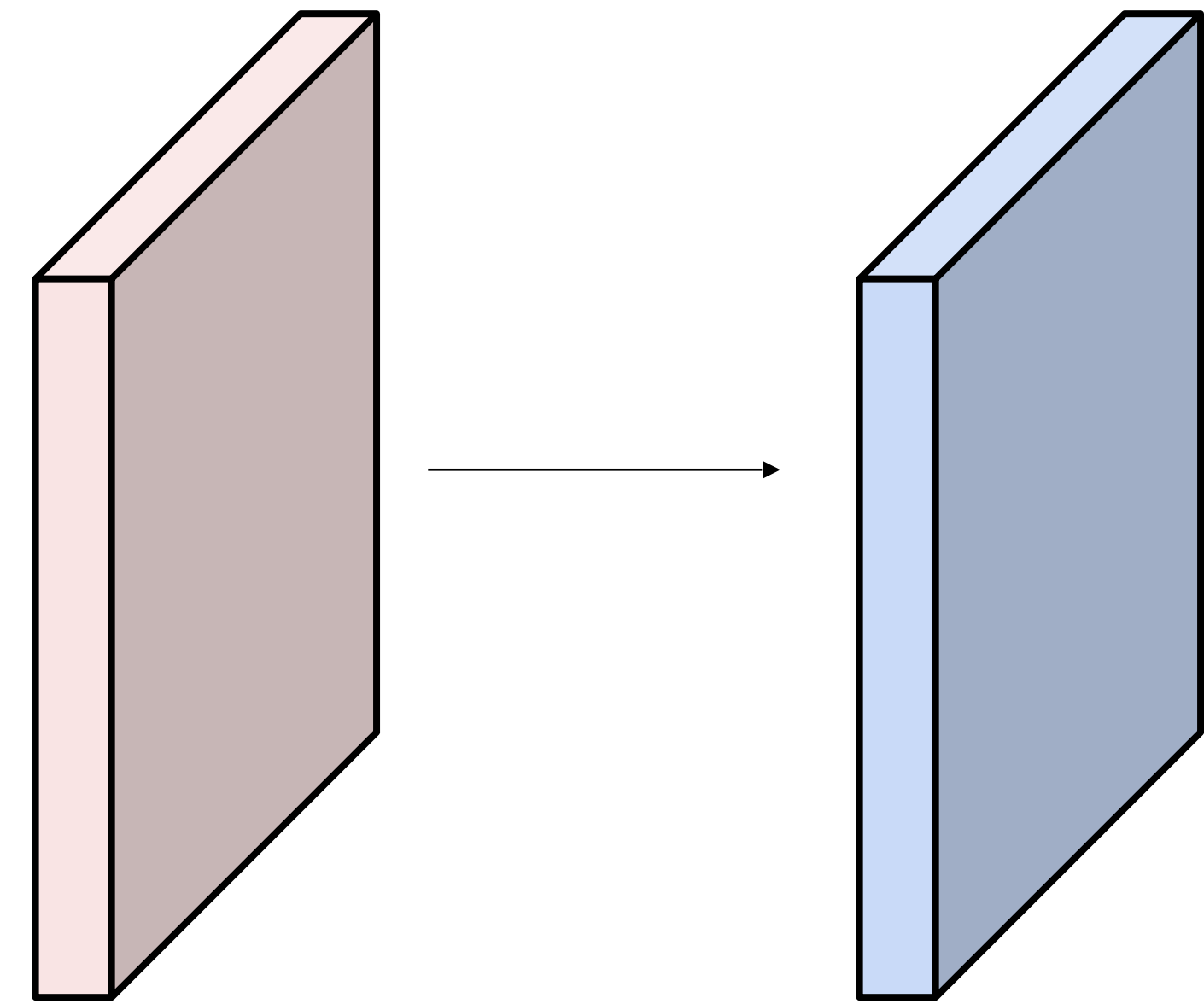
**10** **5x5** filters with stride 1, pad 2

Number of parameters in this layer?

each filter has  $5*5*3 + 1 = 76$  params

(+1 for bias)

=>  $76*10 = 760$



**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- 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 then offset by  $d$ -th bias.



## Common settings:

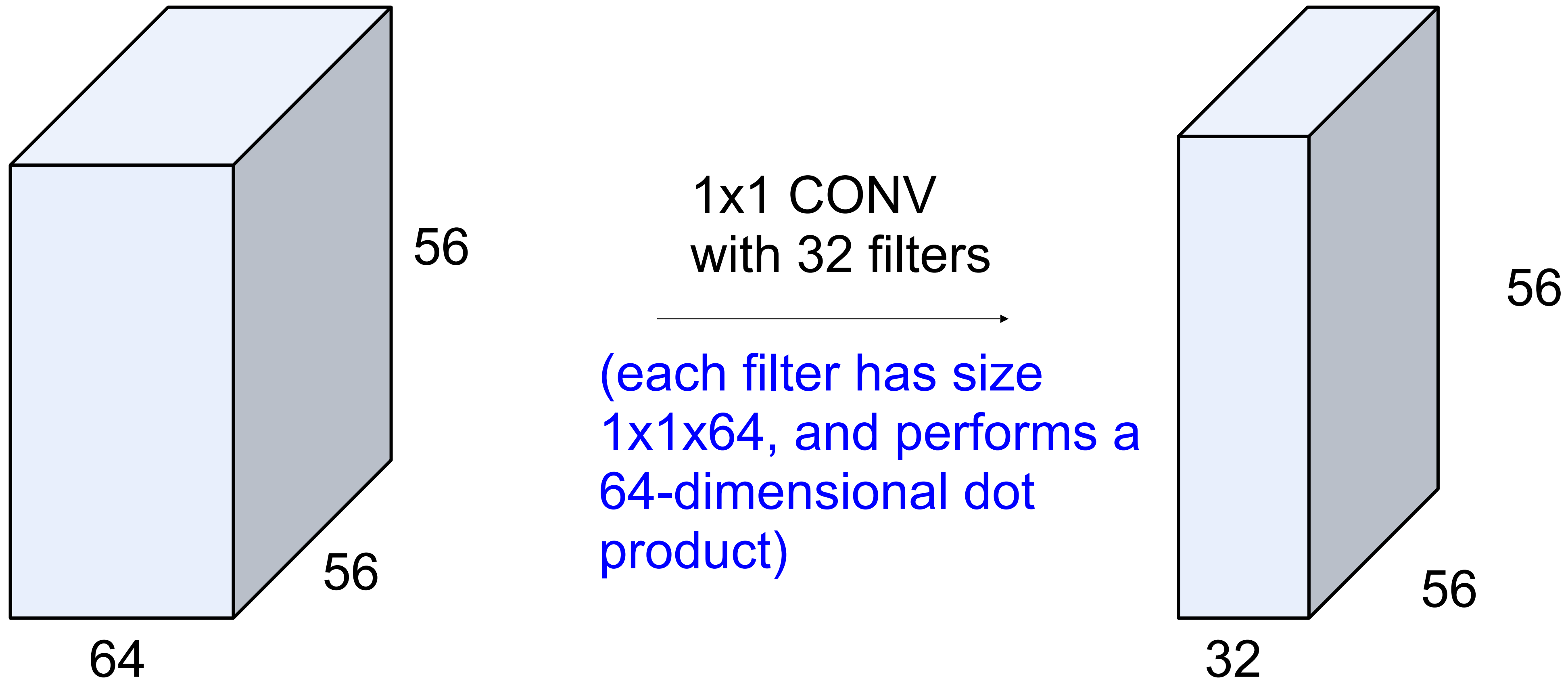
**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- 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 then offset by  $d$ -th bias.

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

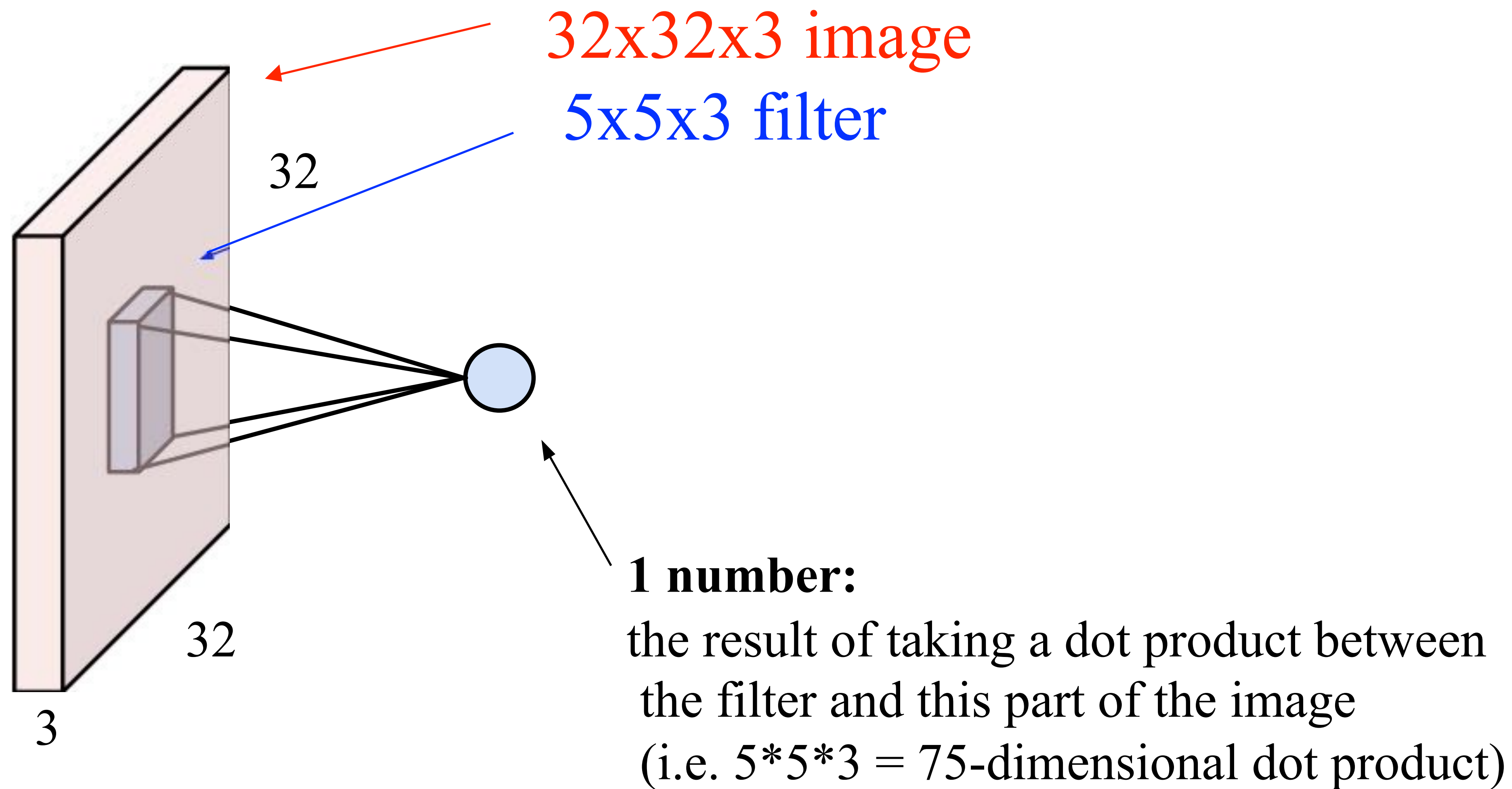
- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ?$  (whatever fits)
- $F = 1, S = 1, P = 0$

(btw, 1x1 convolution layers make perfect sense)

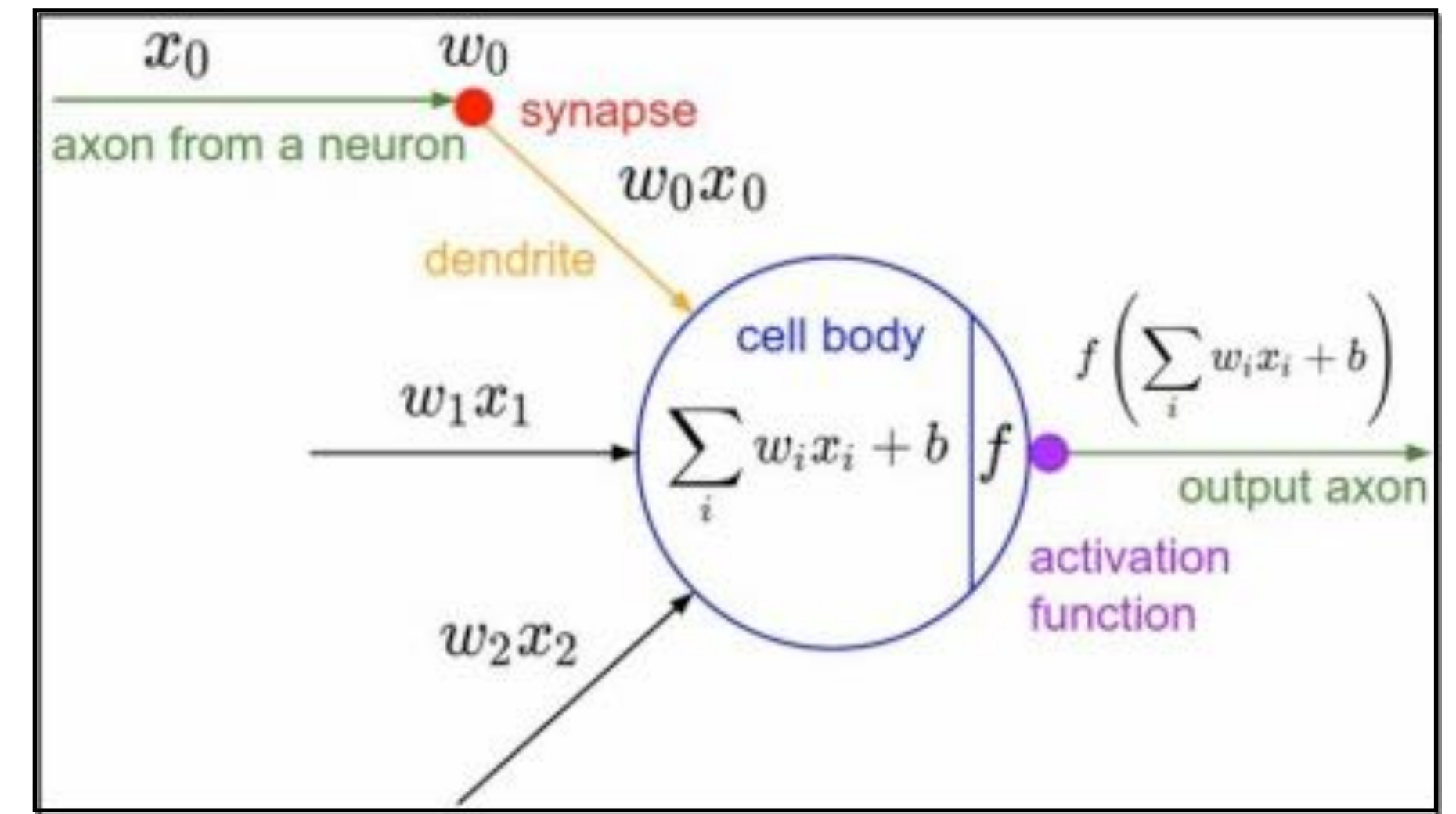
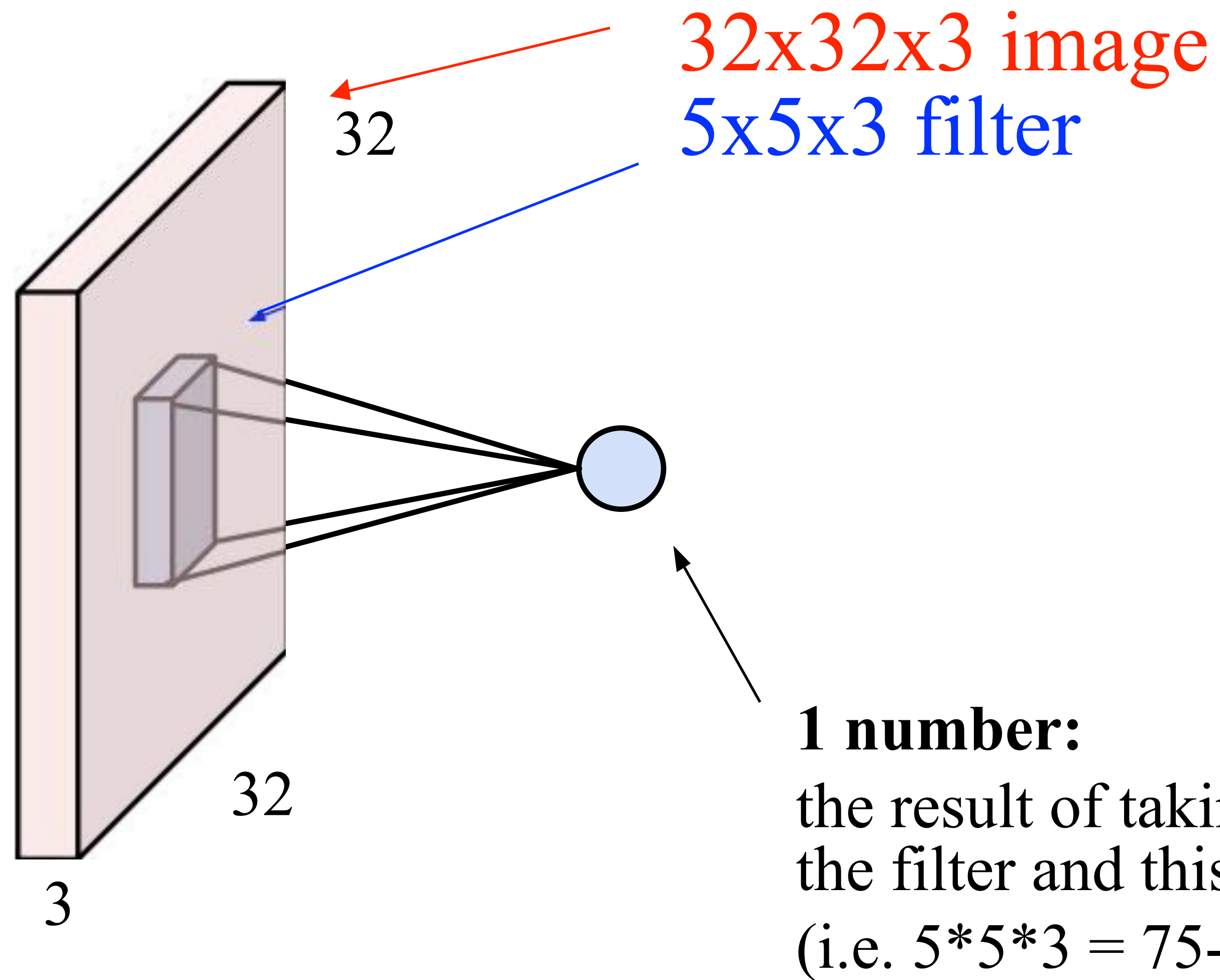




# The brain/neuron view of CONV Layer



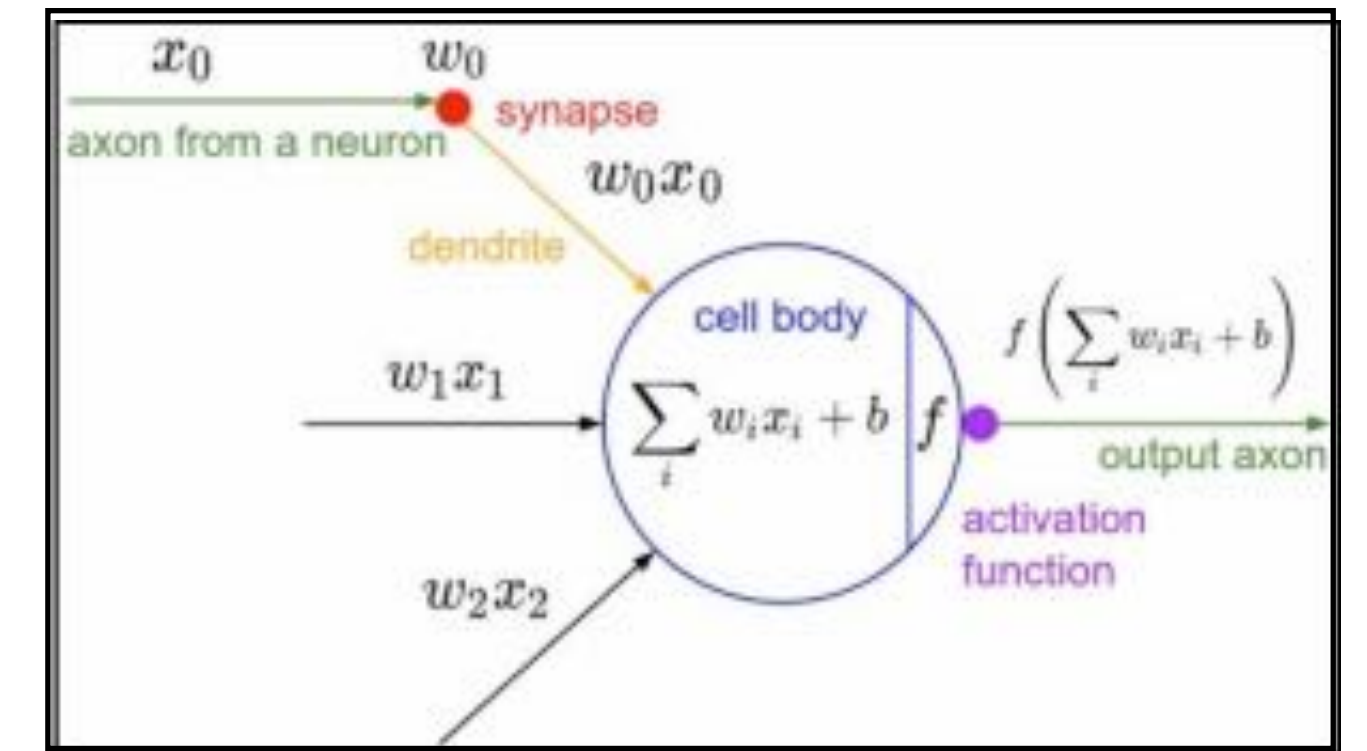
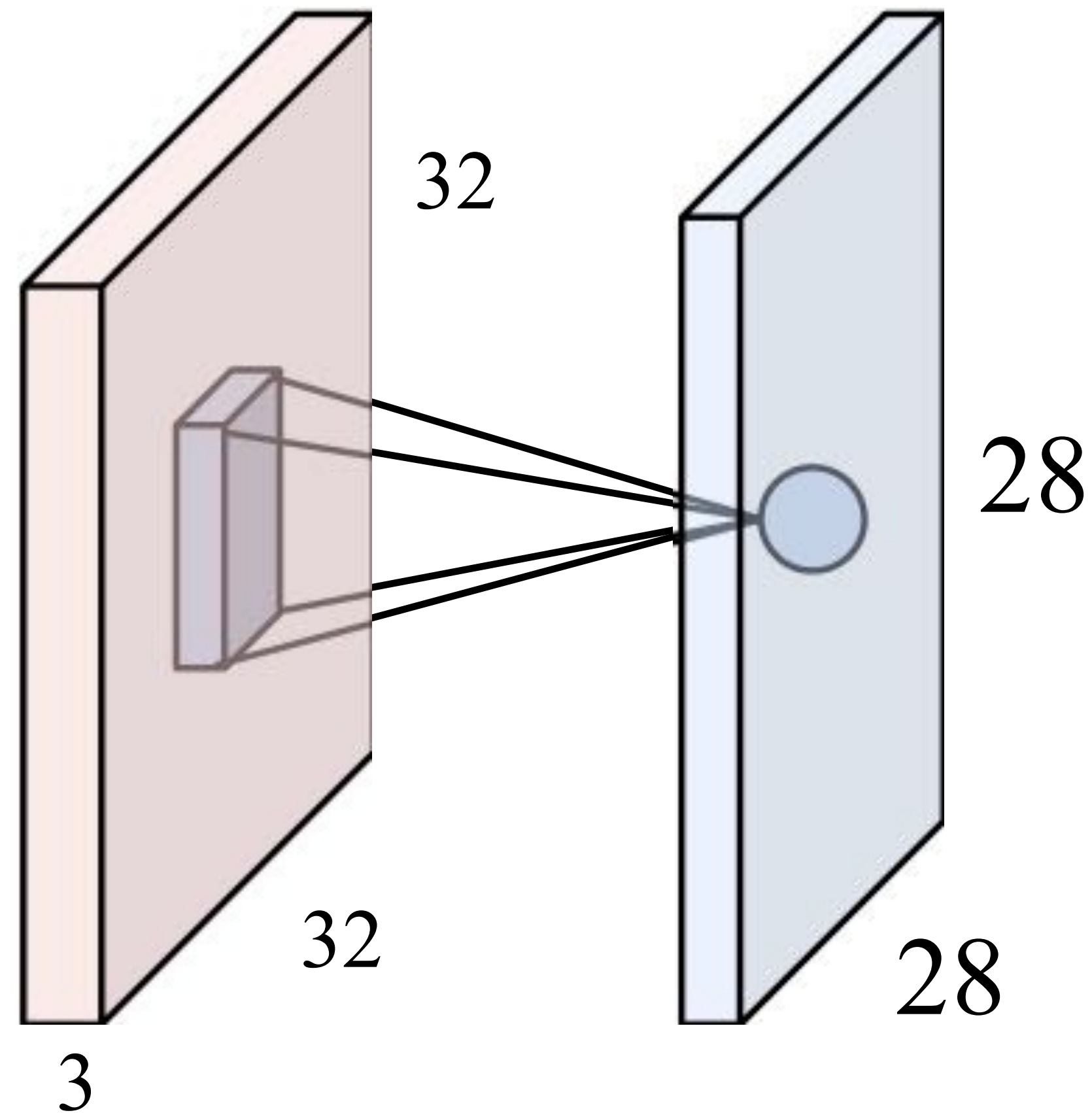
# The brain/neuron view of CONV Layer



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



# The brain/neuron view of CONV Layer

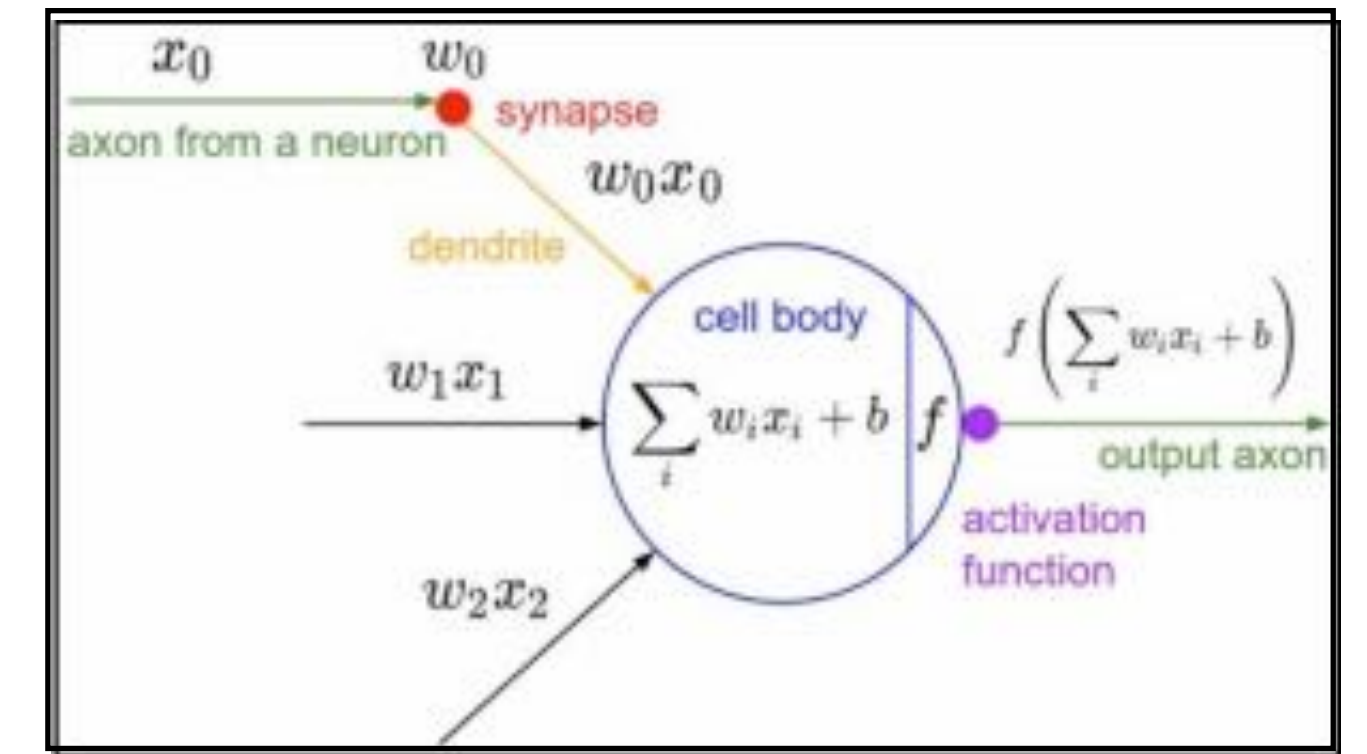
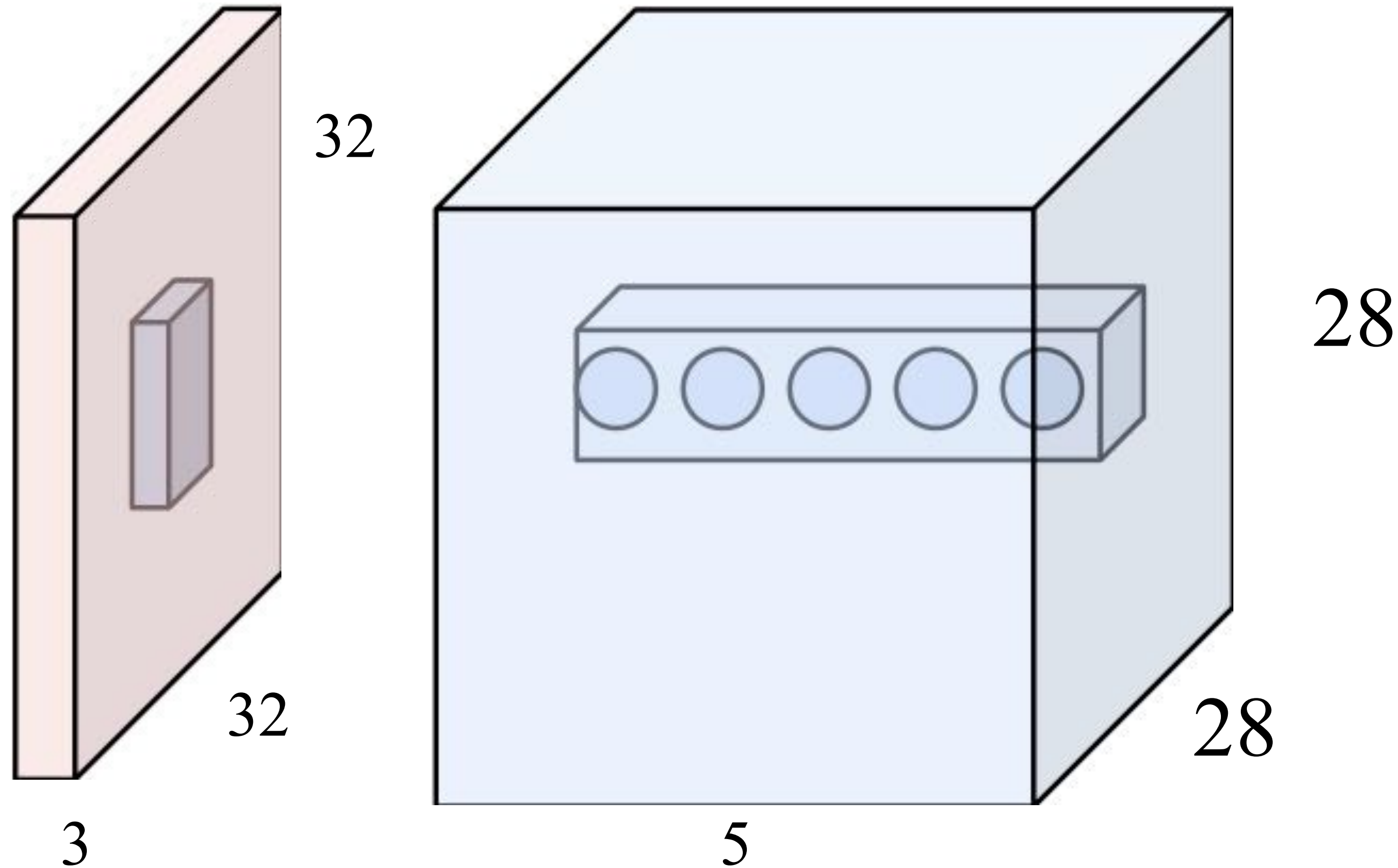


An activation map is a 28x28 sheet of neuron outputs:

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)

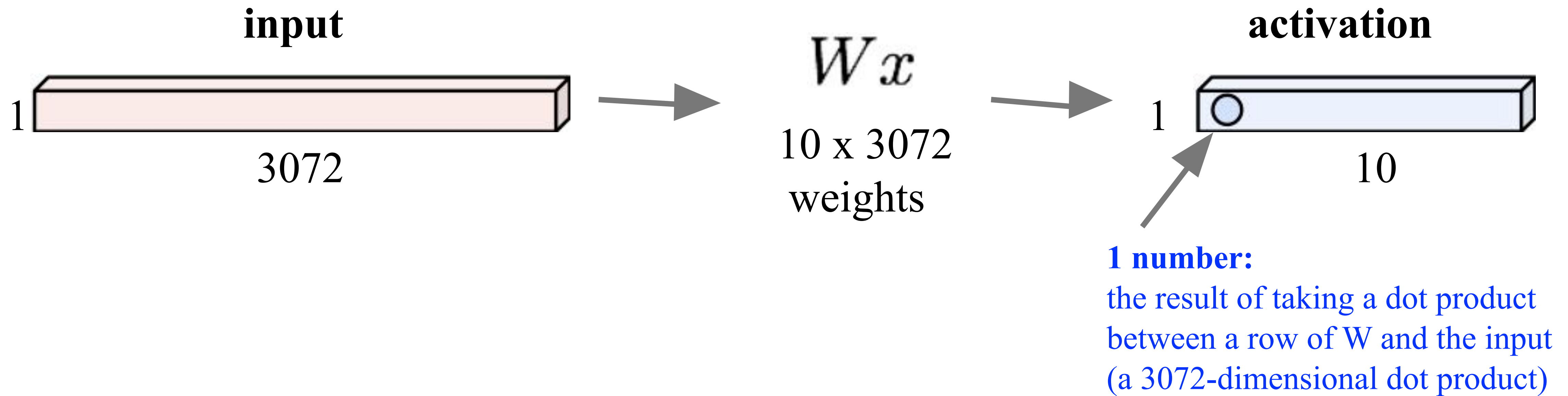
There will be 5 different  
neurons all looking at the same  
region in the input volume



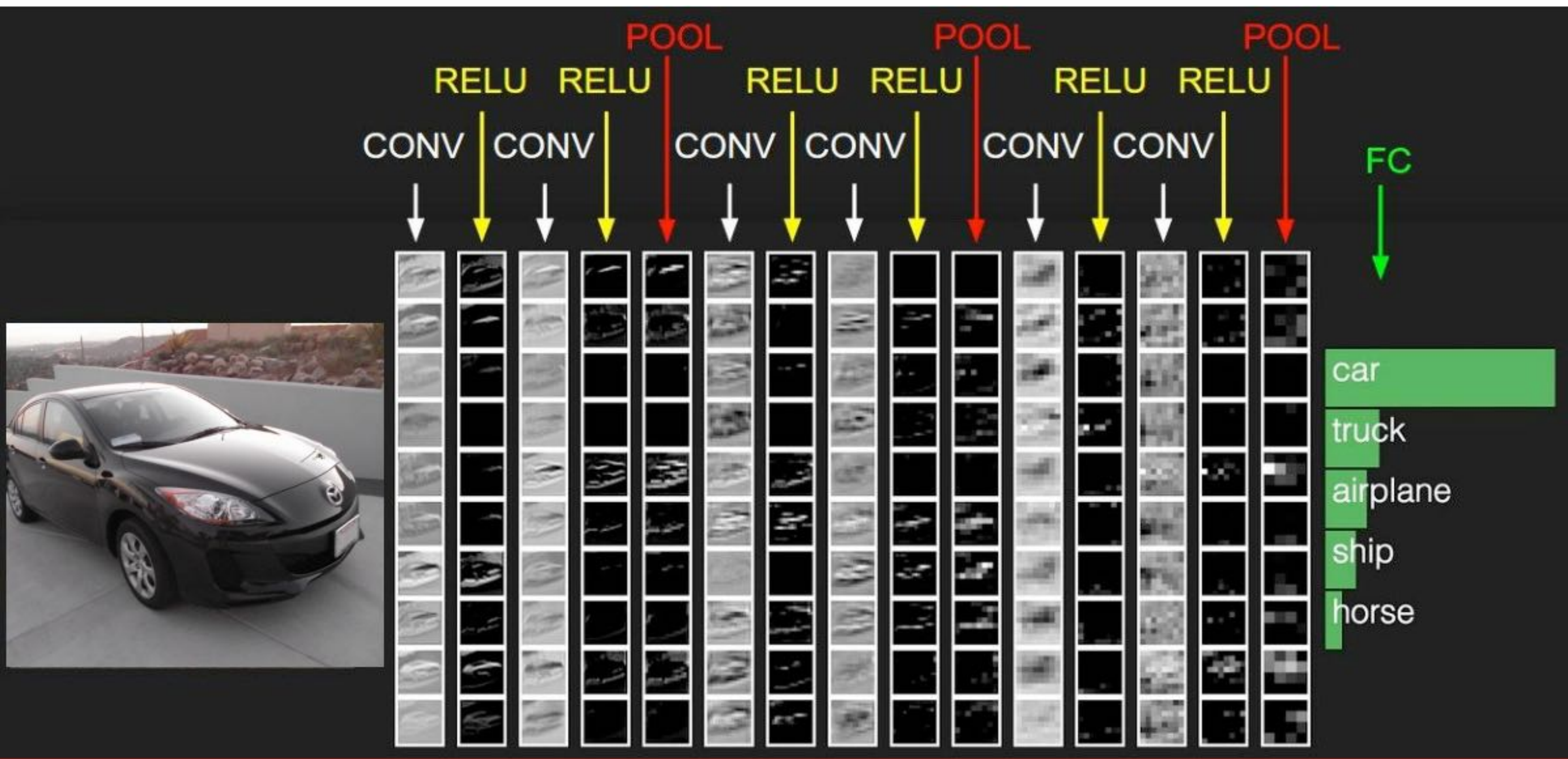
# Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron  
looks at the full  
input volume



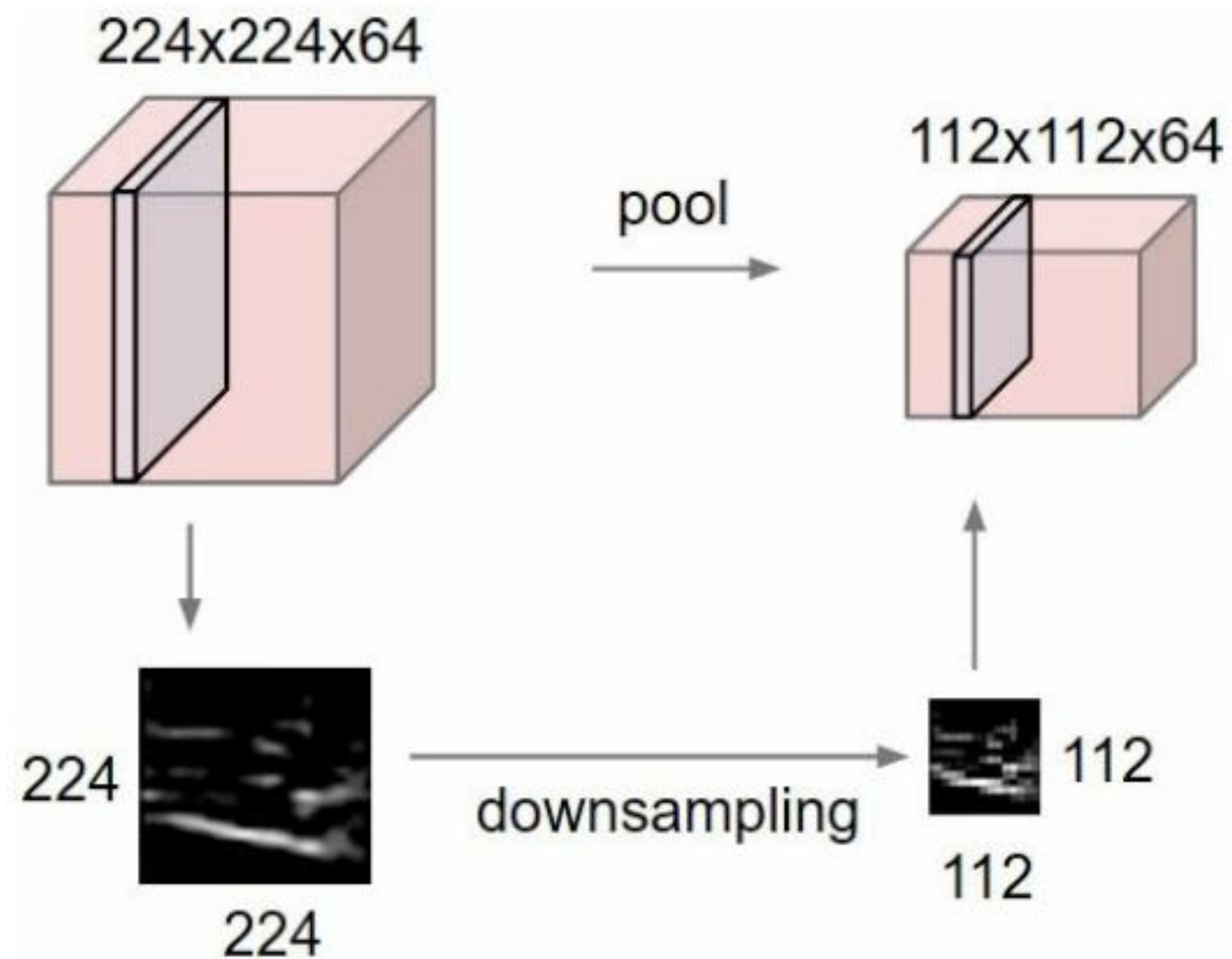
two more layers to go: POOL/FC





# Pooling layer

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



# MAX POOLING

Single depth slice

x ▲

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

→ y

max pool with 2x2 filters  
and stride 2



6	8
3	4



- 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:

$$F = 2, S = 2$$

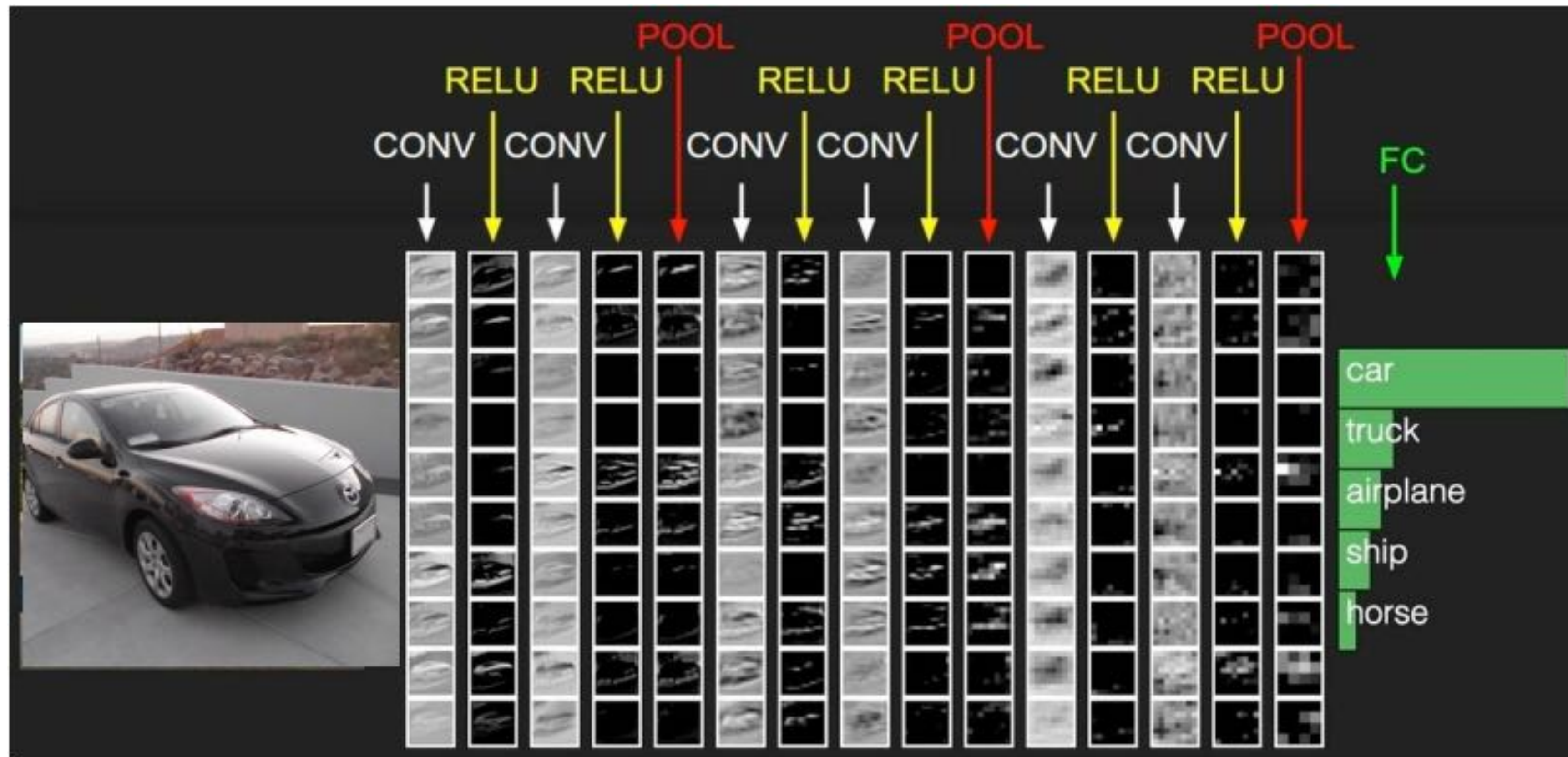
$$F = 3, S = 2$$

- 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



# Fully Connected Layer (FC layer)

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



# 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 ~5, M is large,  $0 \leq K \leq 2$ .
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm