lecture 6

Dong-Geol Choi
Hanbat Nat'l Univ.

Remind

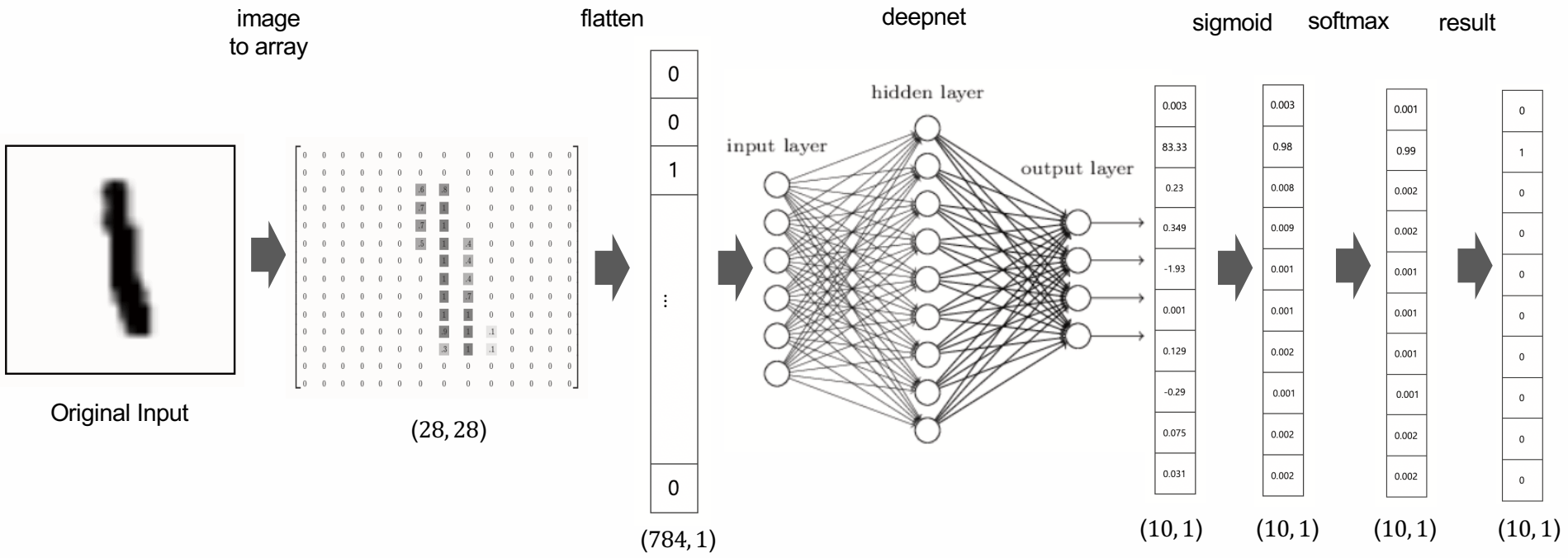
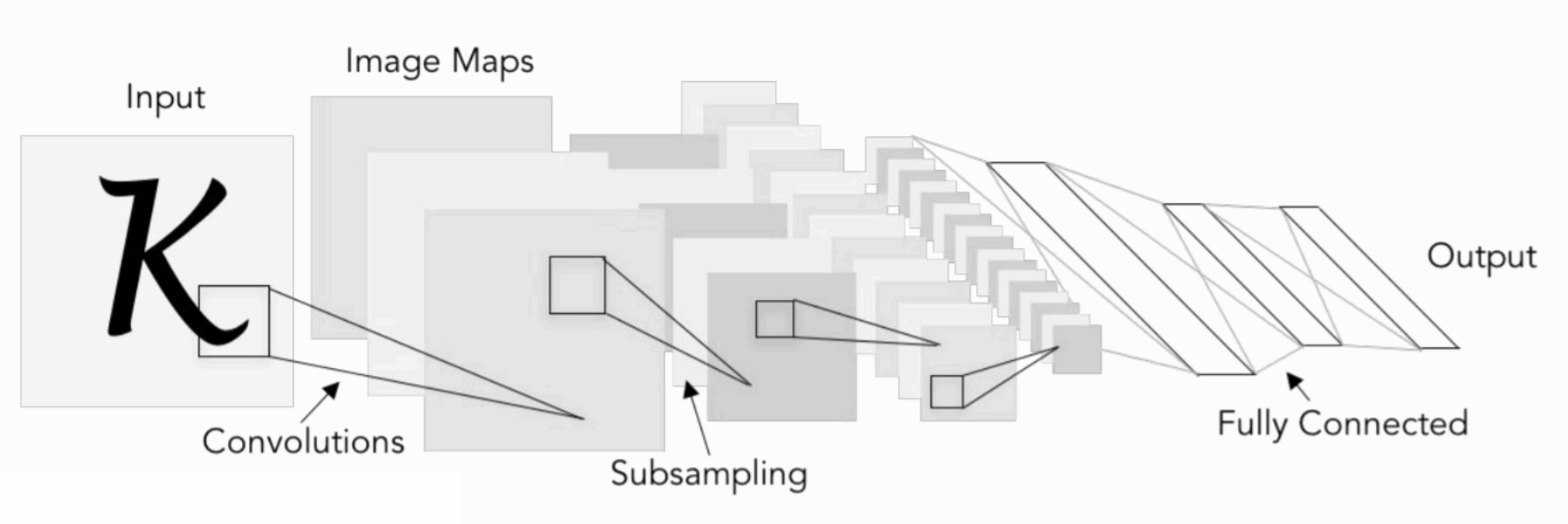


image
to array
flatten
deepnet
sigmoid
softmax
result
Original Input
(28, 28)
0
0
1
0
(784, 1)
input layer
hidden layer
output layer
0.003
83.33
0.23
0.349
-1.93
0.001
0.129
-0.29
0.075
0.031
(10, 1)
0.003
0.98
0.008
0.009
0.001
0.001
0.002
0.001
0.002
0.002
(10, 1)
0.001
0.99
0.002
0.002
0.001
0.001
0.001
0.001
0.002
0.002
(10, 1)
0
1
0
0
0
0
0
0
0
0
(10, 1)

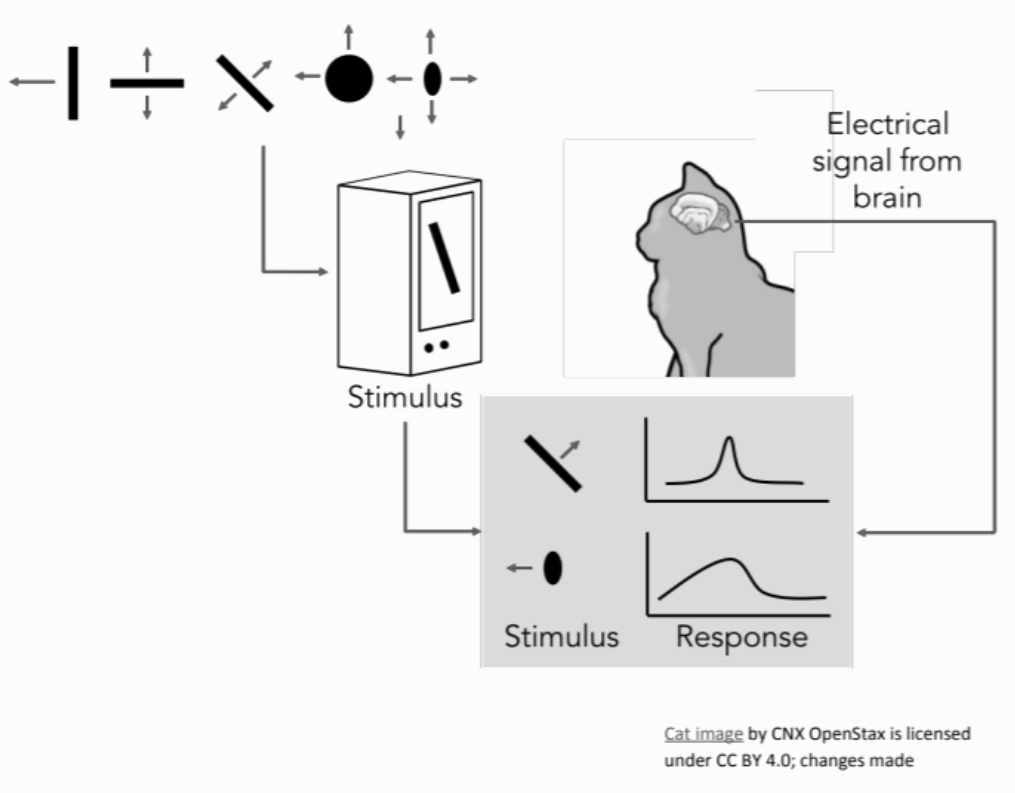
Convolutional Neural Networks



LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Hubel and Wiesel Cat Experiment

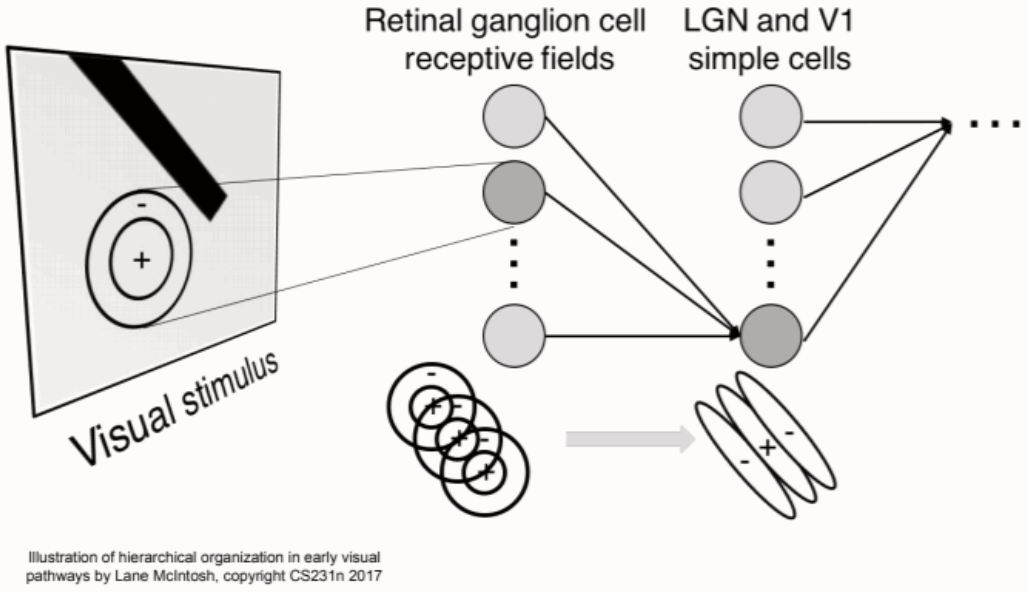
Hubel and Wiesel Cat Experiment : Results



This experiment found that neurons responded to things like oriented **edges** and **shapes**.

Experiments have found that neurons have a **hierarchy**.

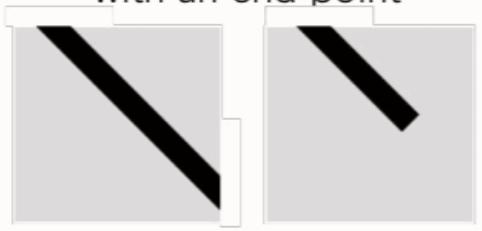
Hubel and Wiesel Cat Experiment : Results



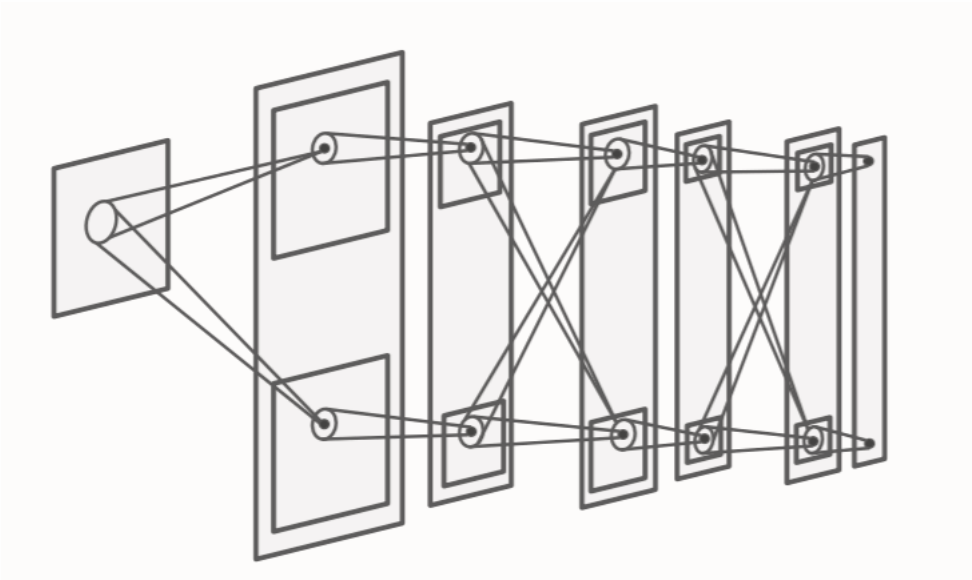
Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



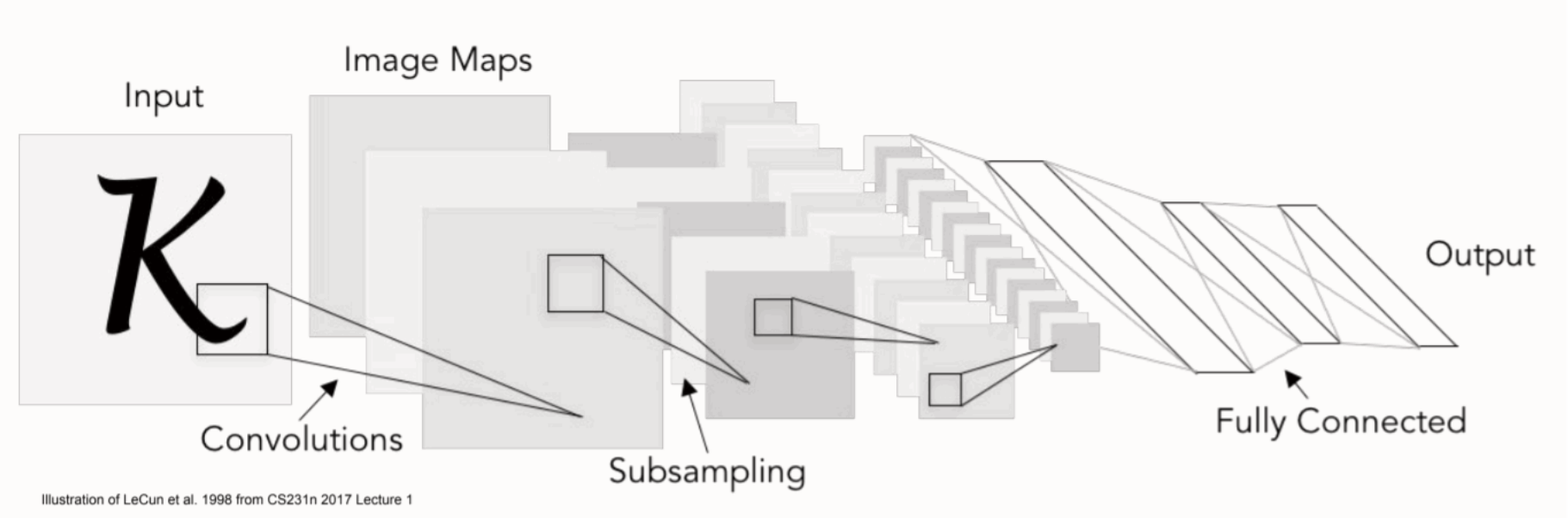
Neocognitron [Fukushima 1980]



The first NN to use the idea of simple / complex cells.

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

Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998]



In 1998, Yann LeCun first trained NN Apply backprob and gradient-based learning.

It works well for recognizing postal code numbers and is widely used for real postal code recognition in postal services.

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

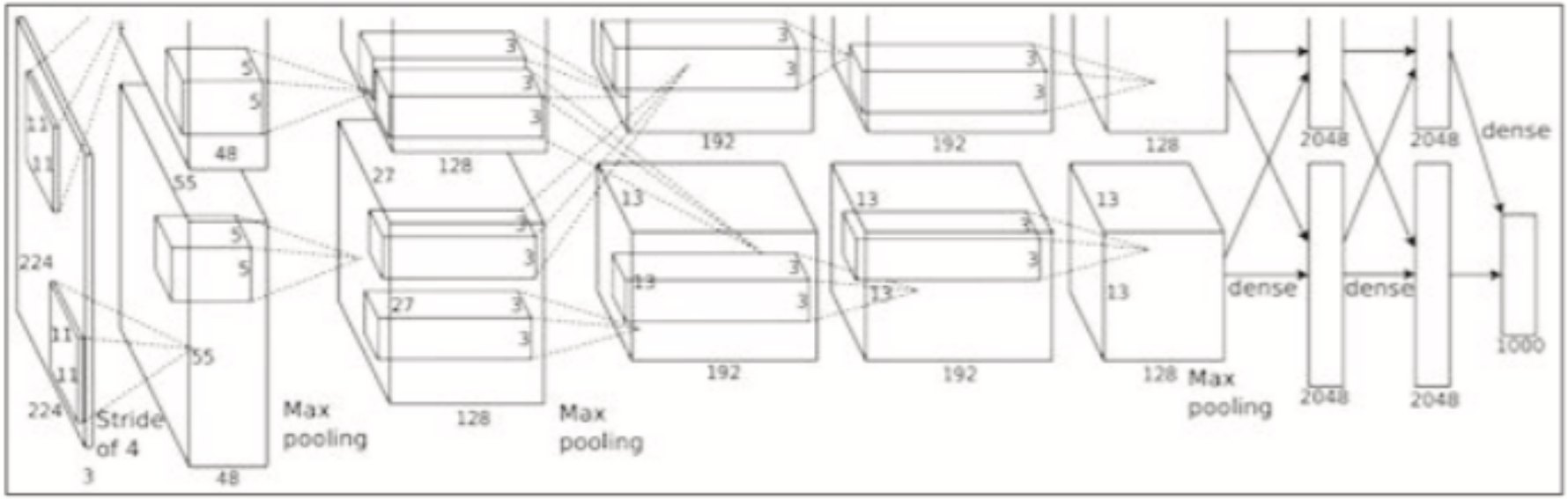
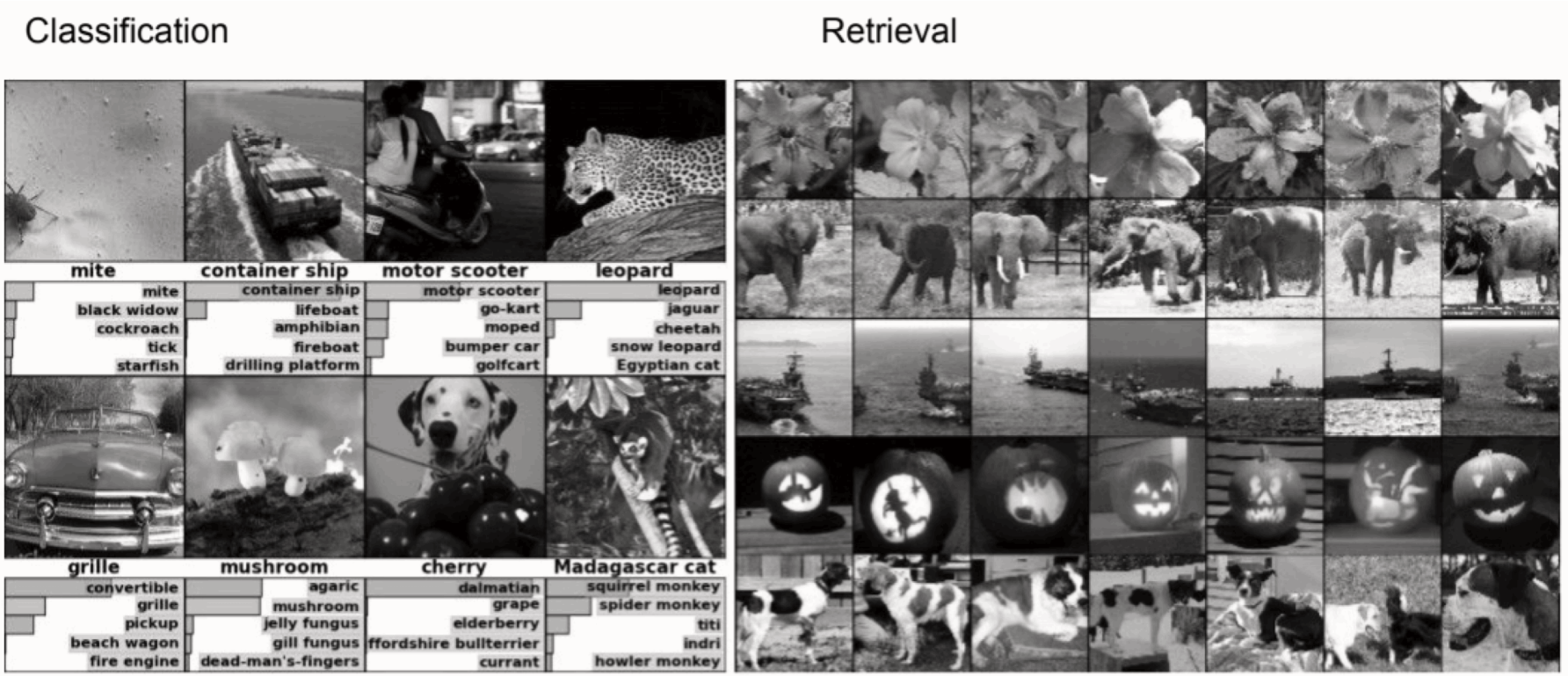


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

AlexNet

The most important point is that now we have a **big data** is there and we can take advantage of the **GPU**.

ConvNets are everywhere : Classification, Retrieval



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

ConvNets are everywhere : Detection, Segmentation

Detection

Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, 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]

ConvNets are everywhere : Video

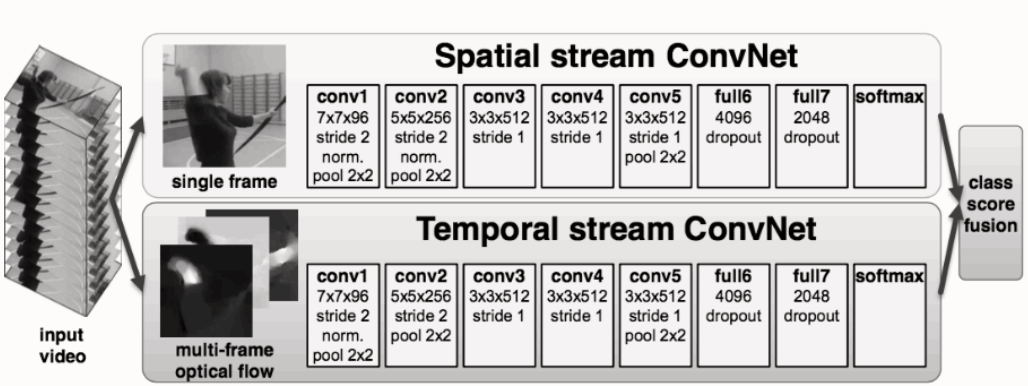
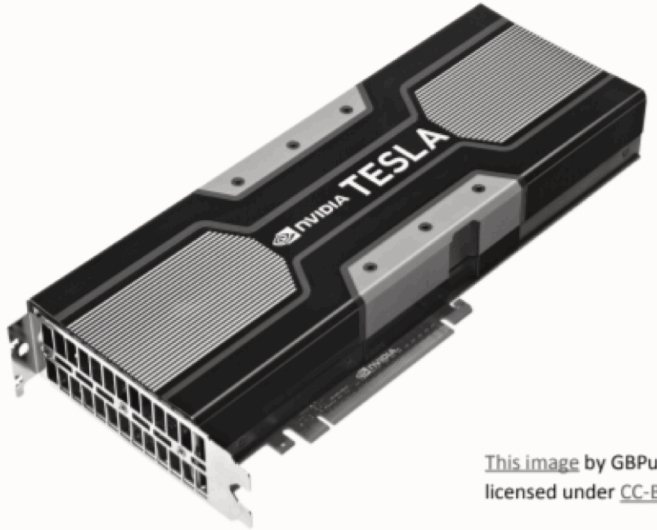


Figure 1: **Two-stream architecture for video classification.**

ConvNets are everywhere : self-driving cars

Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

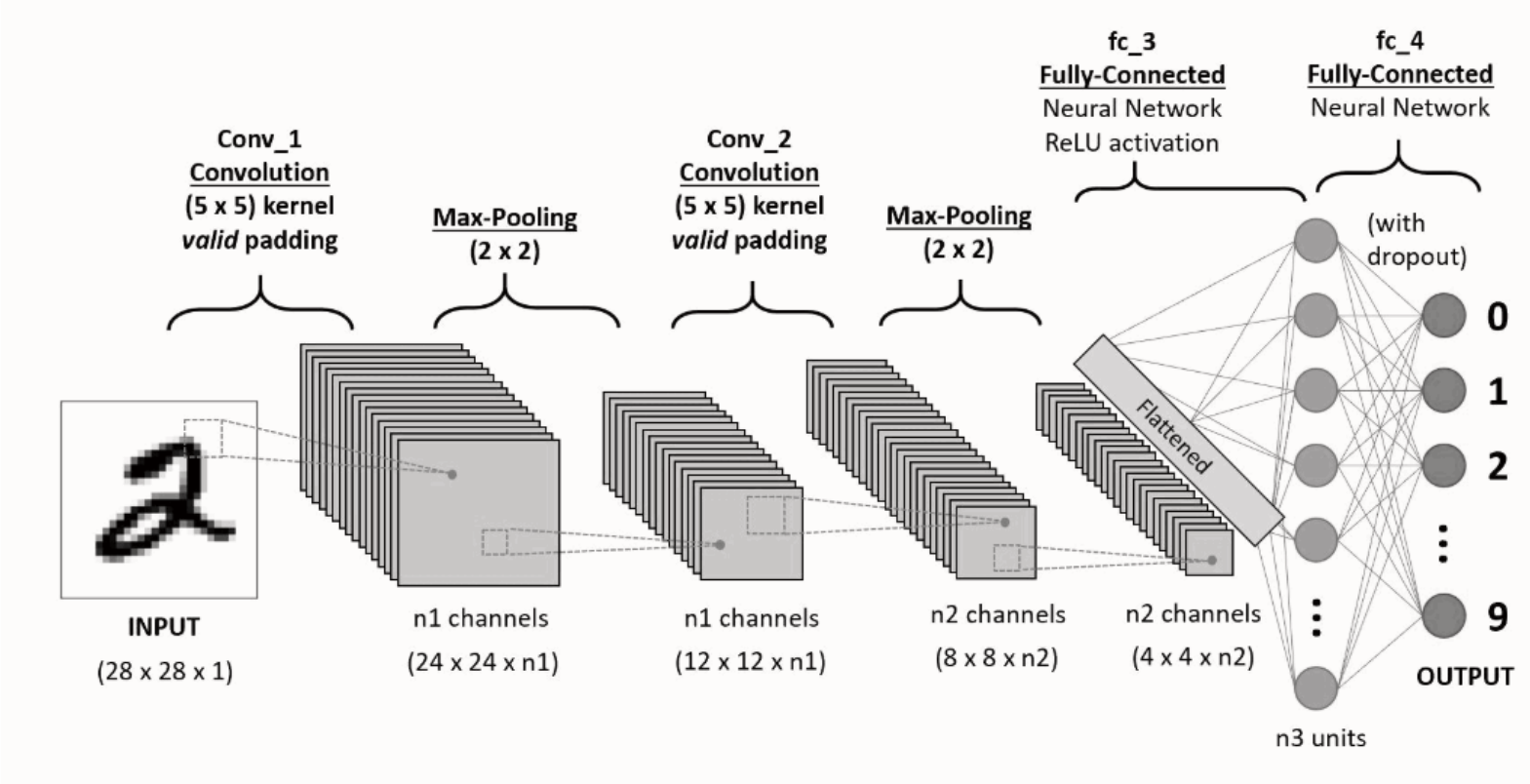


NVIDIA Tesla line

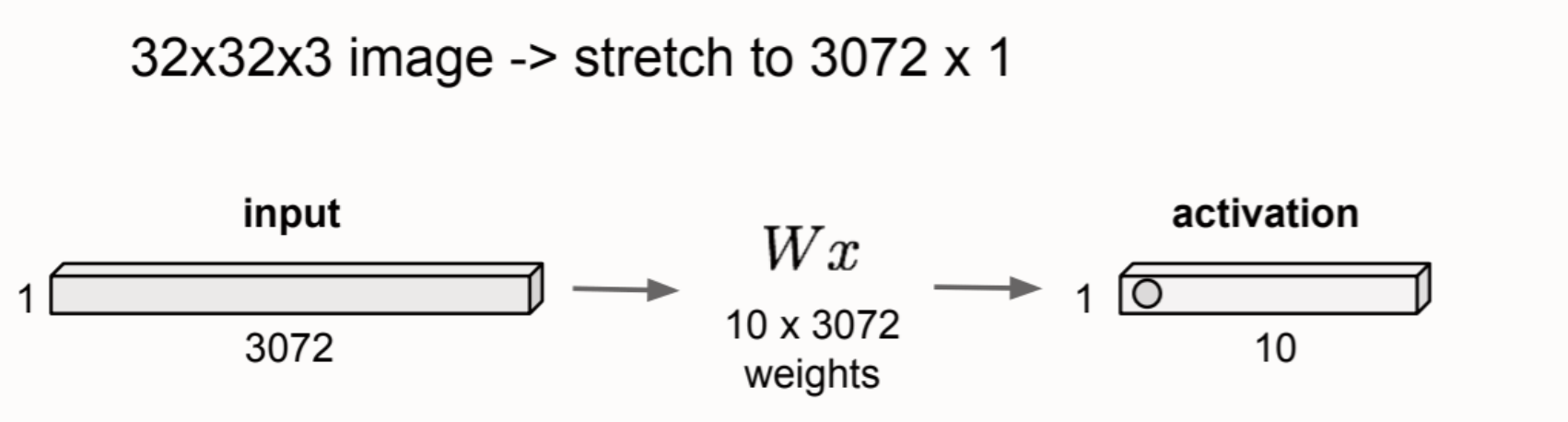
(these are the GPUs on rye01.stanford.edu)

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

Convolutional Neural Networks



Fully Connected Layer

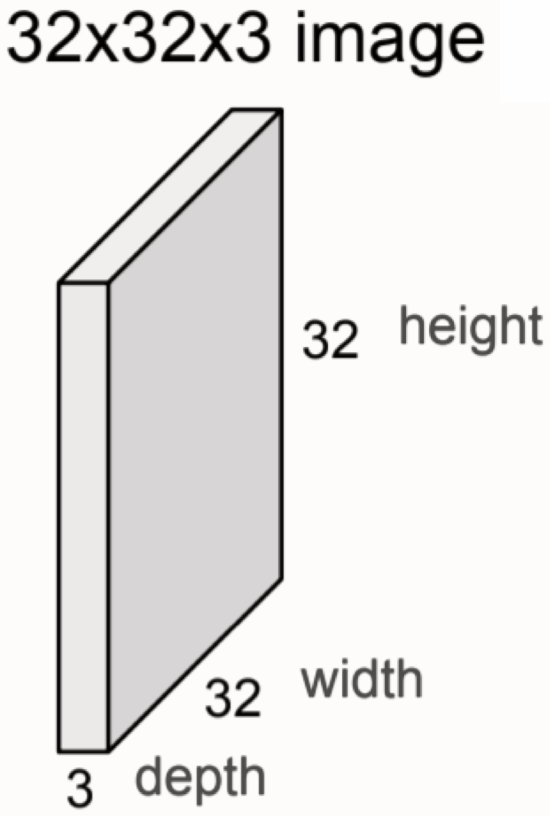


Fully Connected Layer **loses spatial information**.

The Fully Connected layer does **not work if the image size is different.**

Since the Fully Connected layer has **too many parameters**, it is difficult to stack several.

Convolution Layer

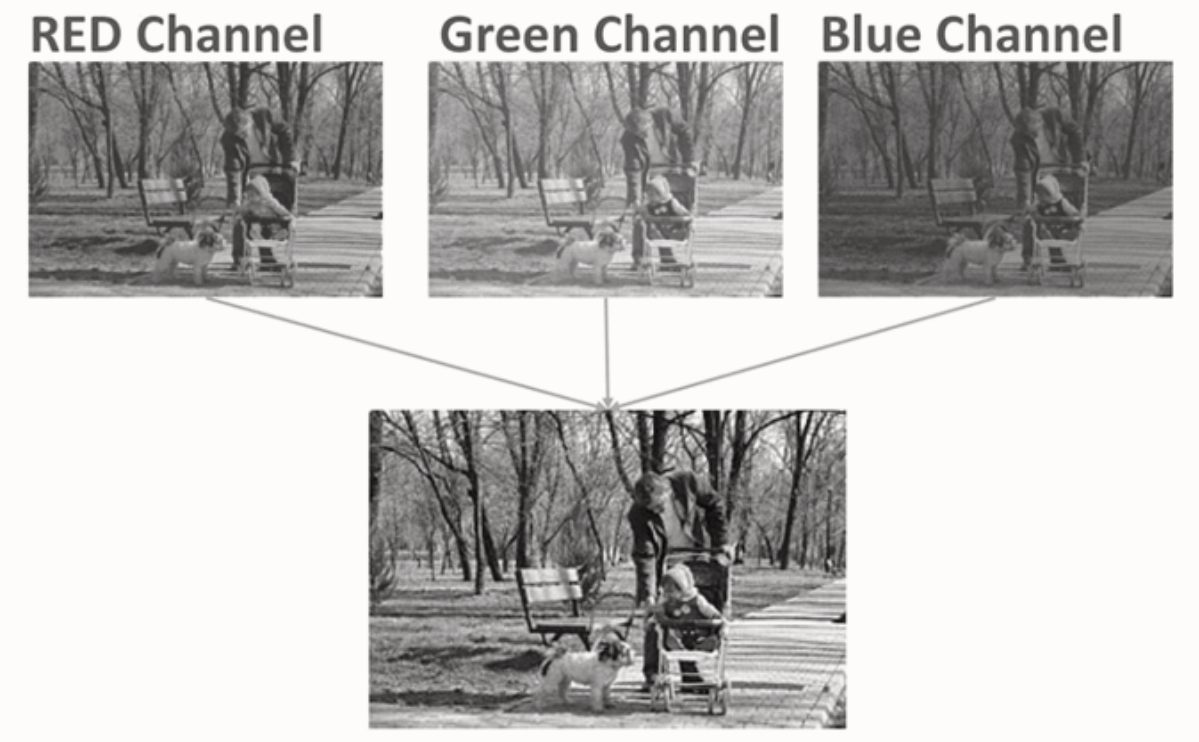


Keep spatial information.

It works **regardless of the size of the input image.**

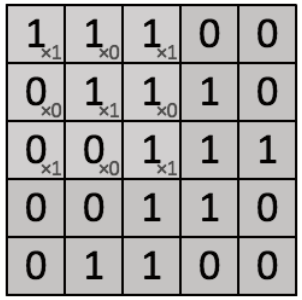
There are **fewer parameters** than FC layer.

What is Depth?

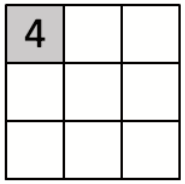


Depth = RED Channel + Green Channel + Blue Channel
3 = 1 + 1 + 1

Filter? Filtering?

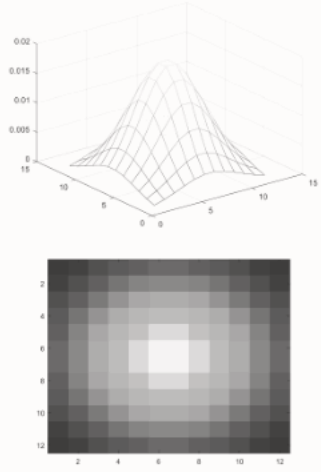


Image



Convolved Feature

Image

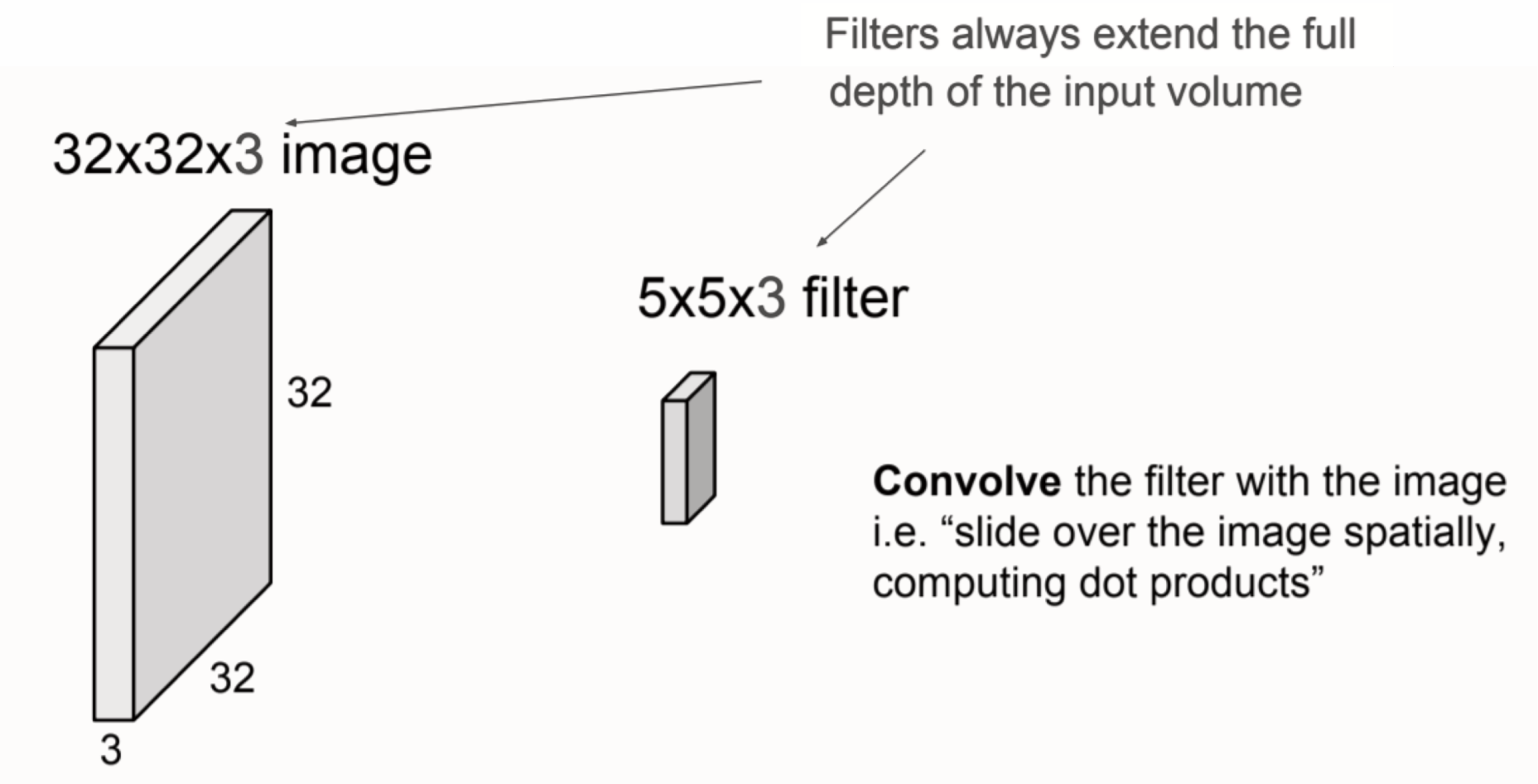


Kernel

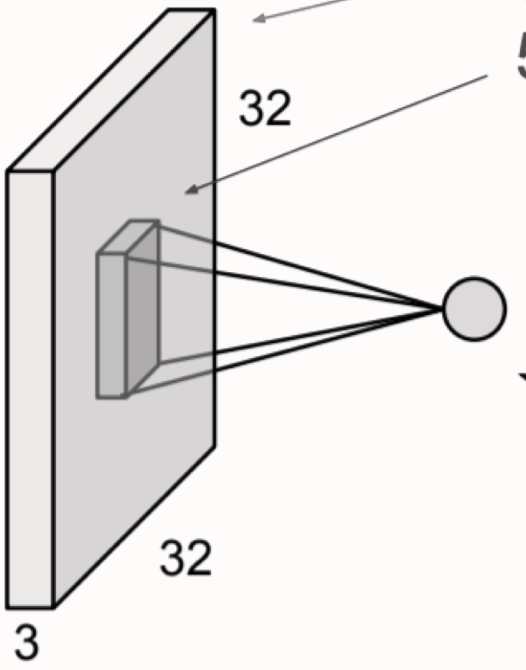
Output

Filtering in an image is a process of **extracting a desired component.**

Convolution Layer (1)



Convolution Layer (2)



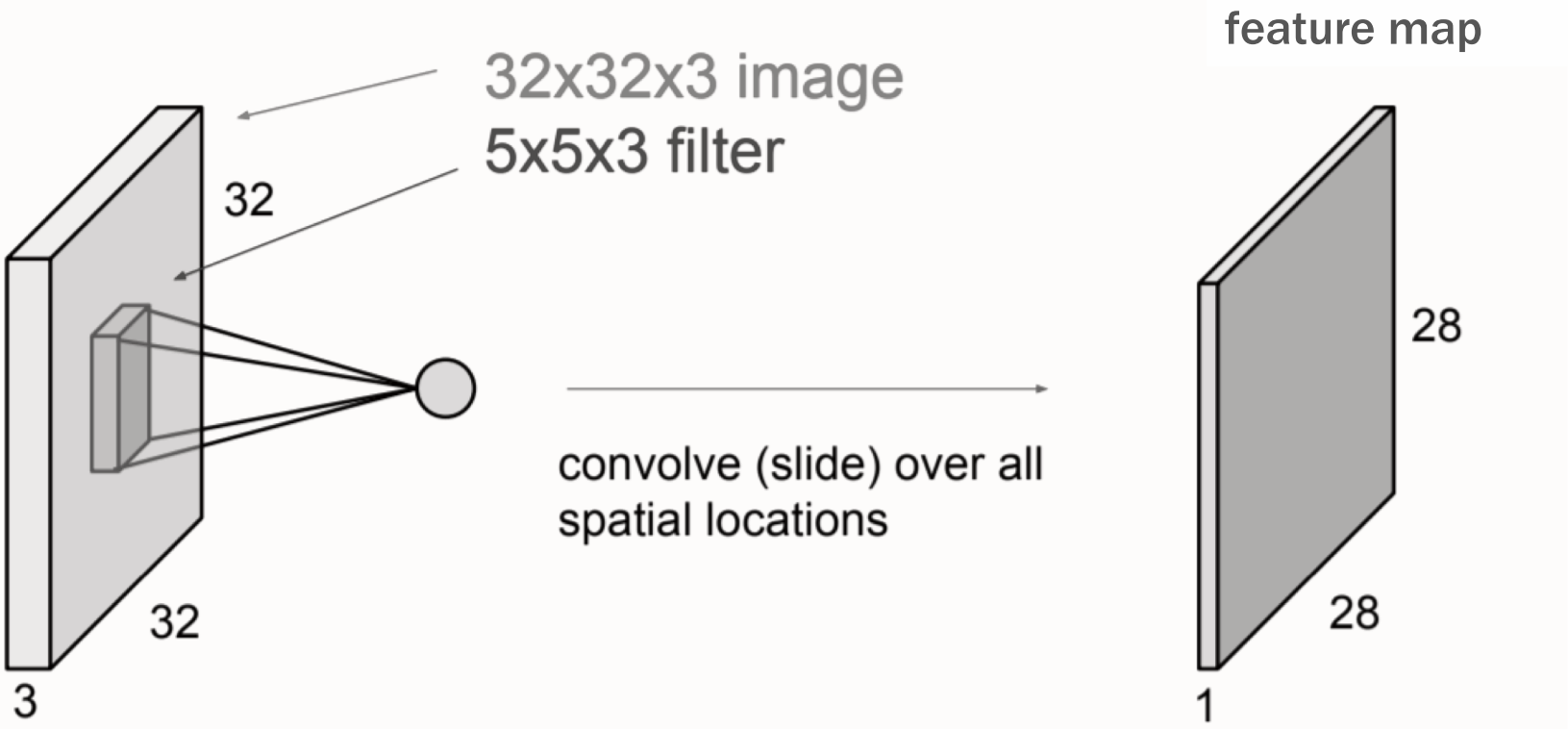
32x32x3 image

5x5x3 filter w

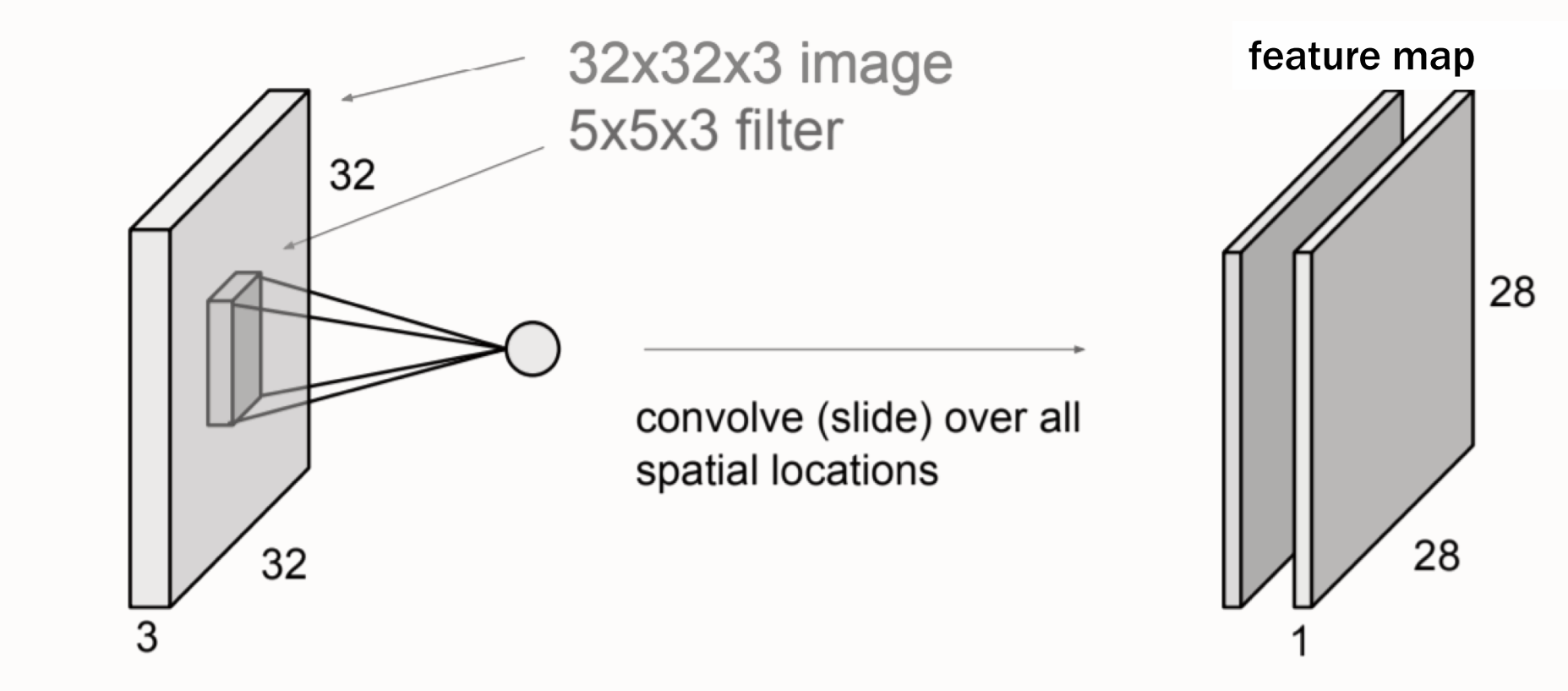
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

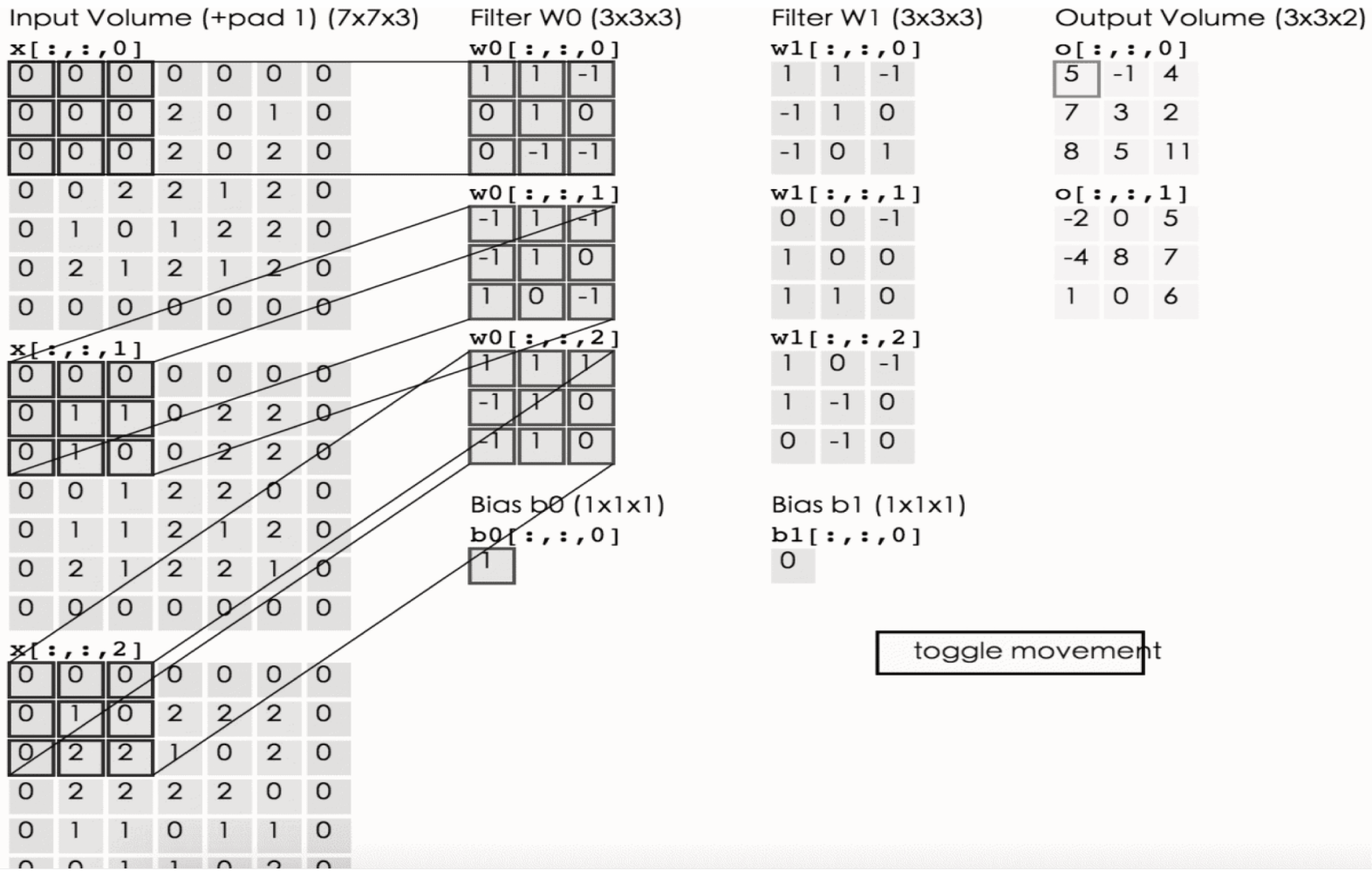
Convolution Layer (3)



Convolution Layer (4)

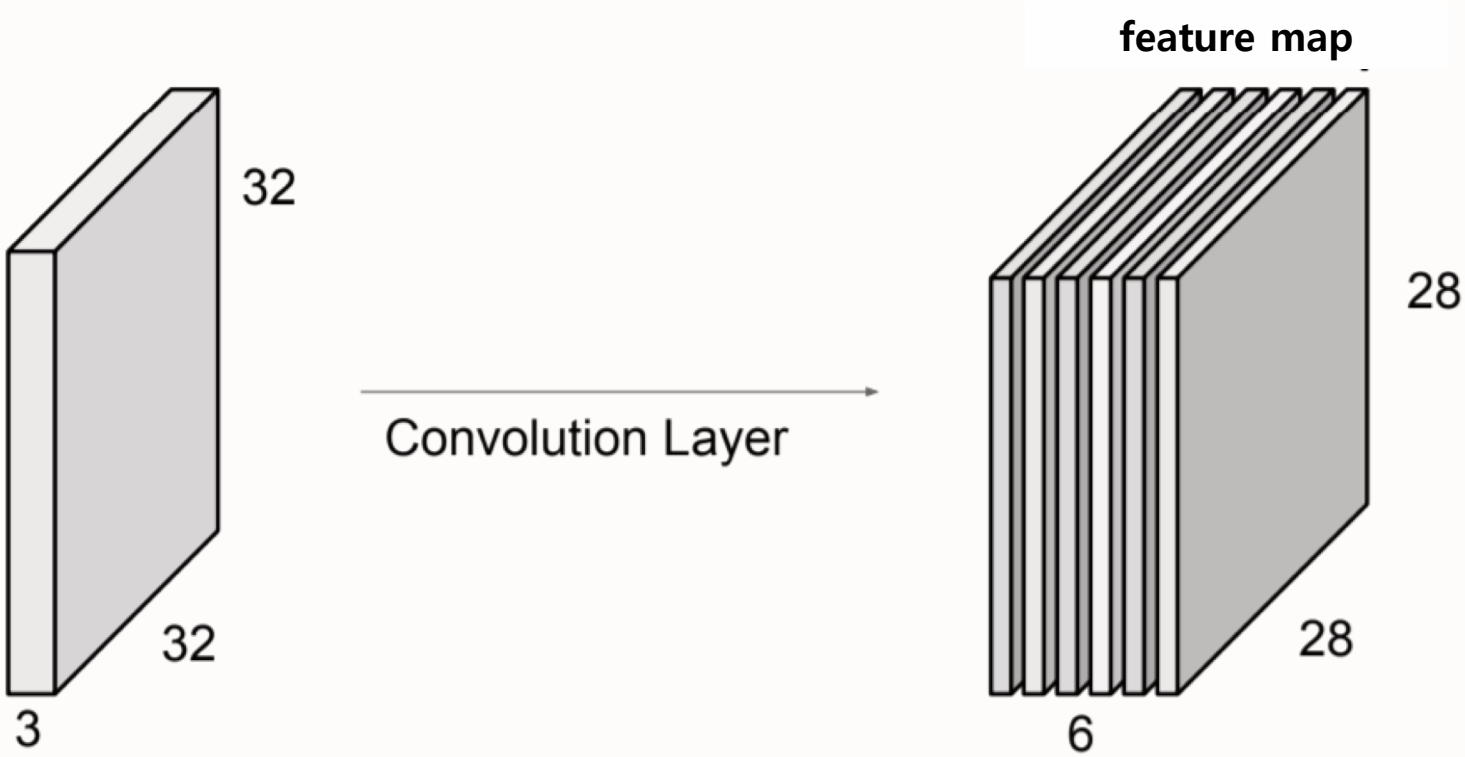


Convolution Layer (5)



Convolution Layer (6)

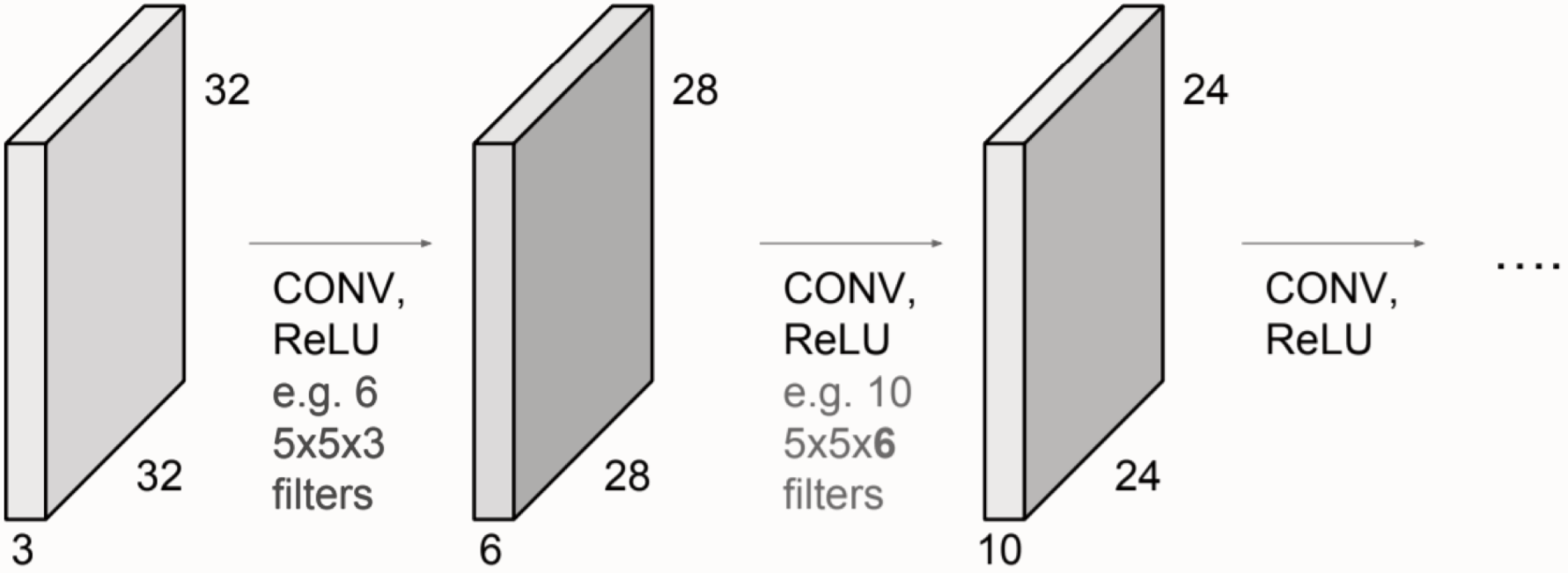
For example, if we had 6 5x5 filters, we'll get 6 separate **feature map**



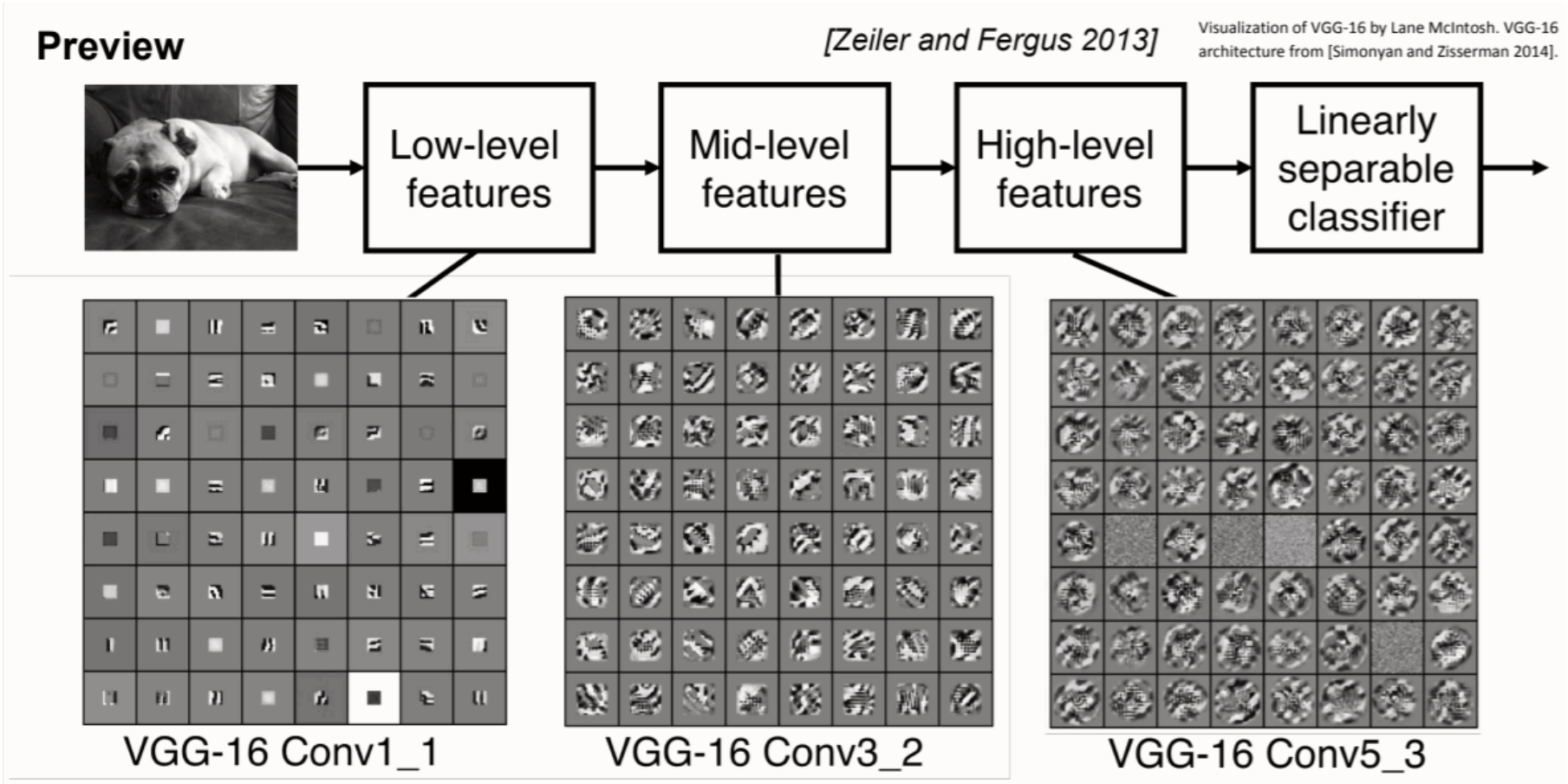
We stack these up to get a "new image" of size 28x28x6!

Convolution Layer (7)

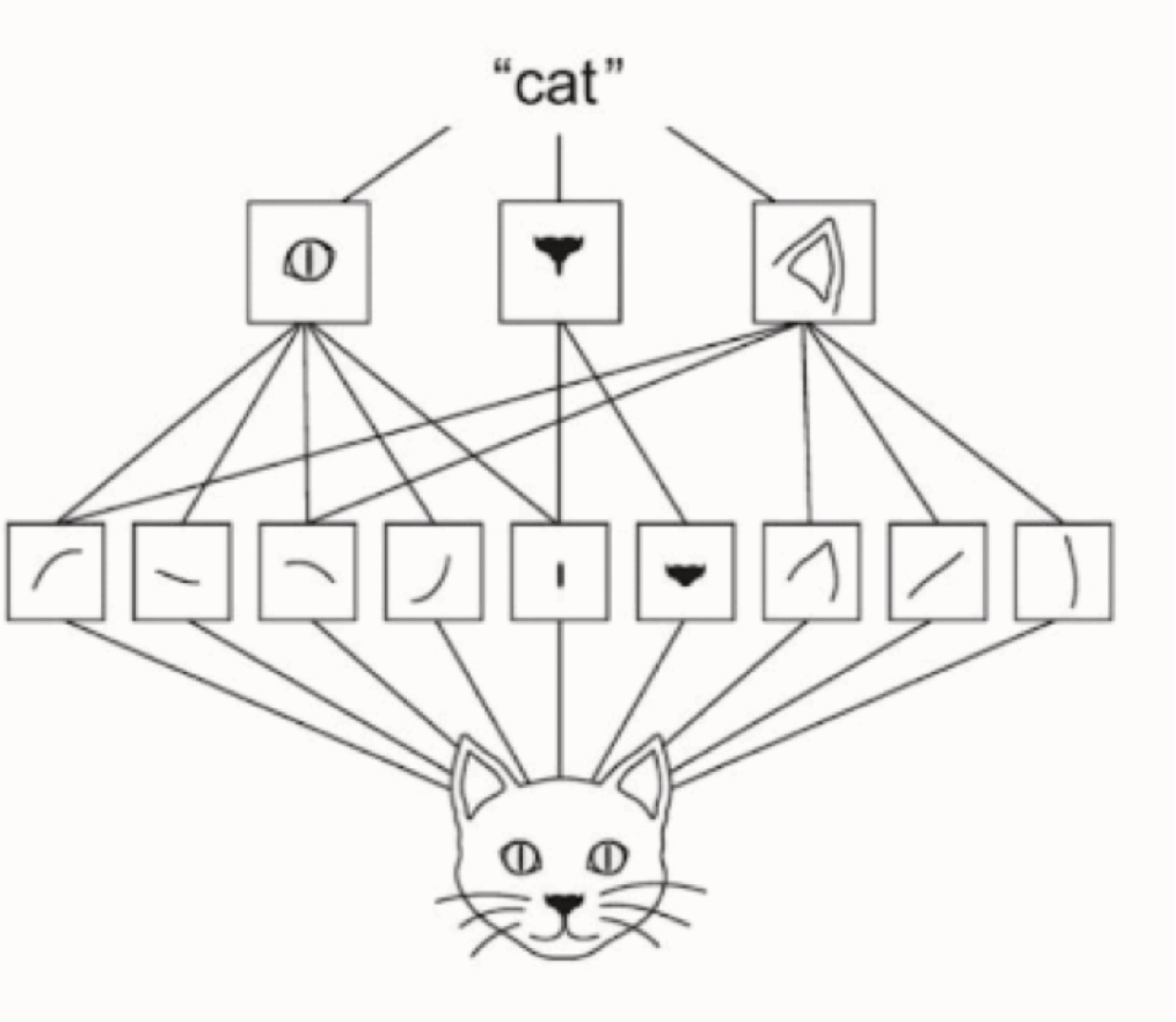
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Convolution Layer (8)

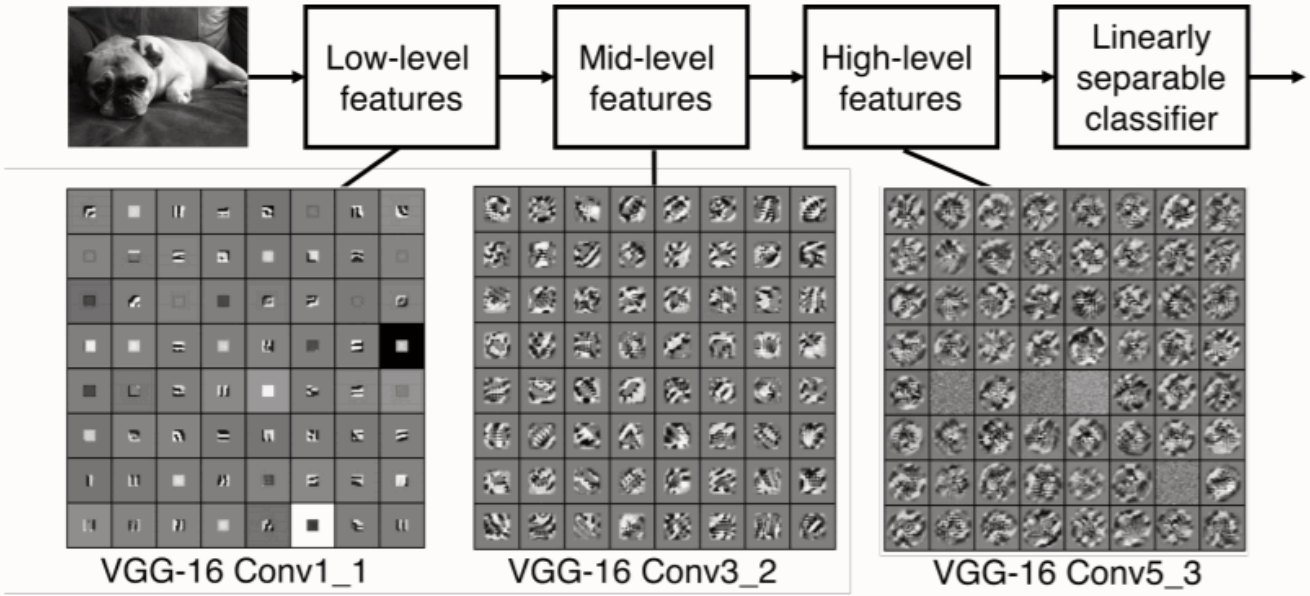


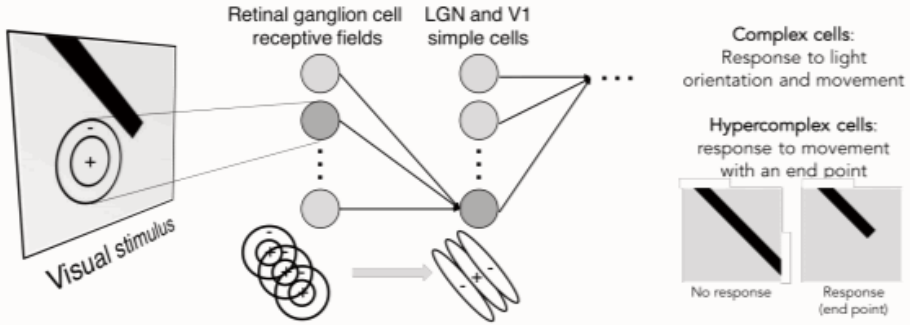
Convolution Layer (9)



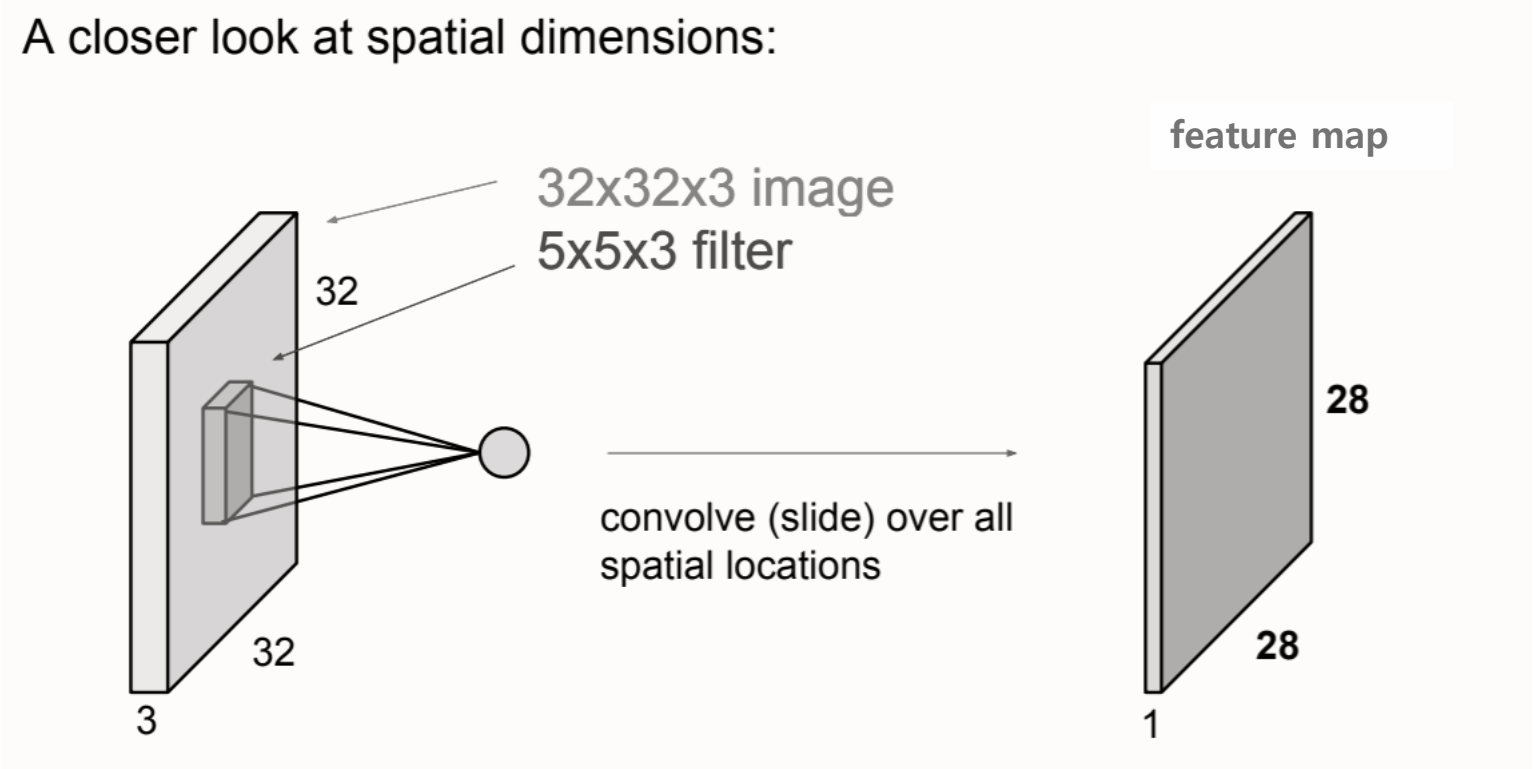
Convolution Layer (10)

Preview



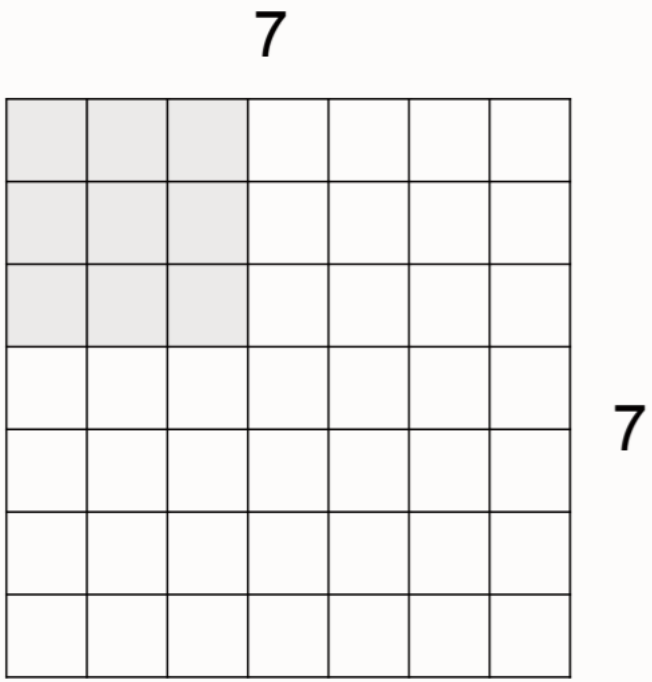


Convolution Layer (11)



Convolution Layer (12)

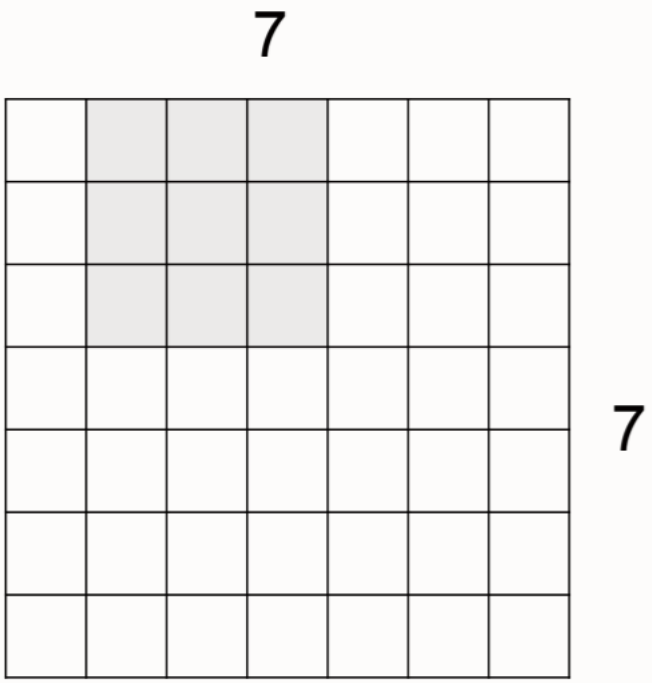
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolution Layer (13)

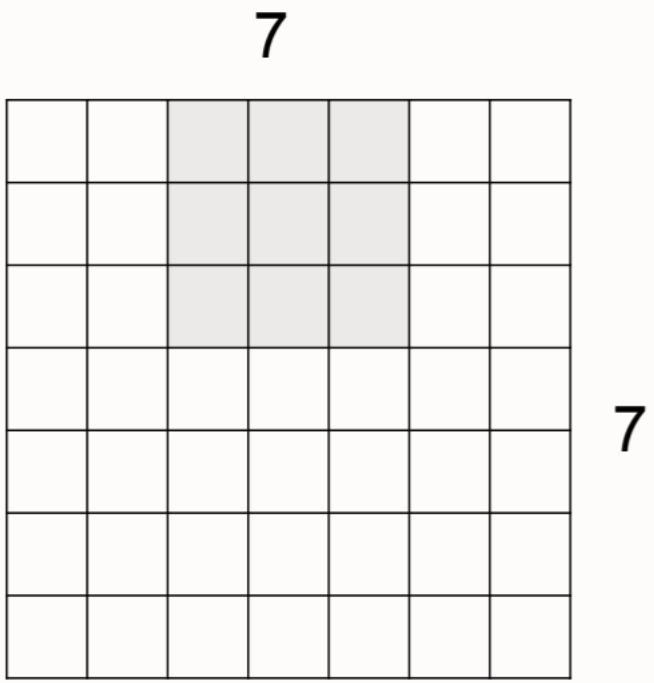
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolution Layer (14)

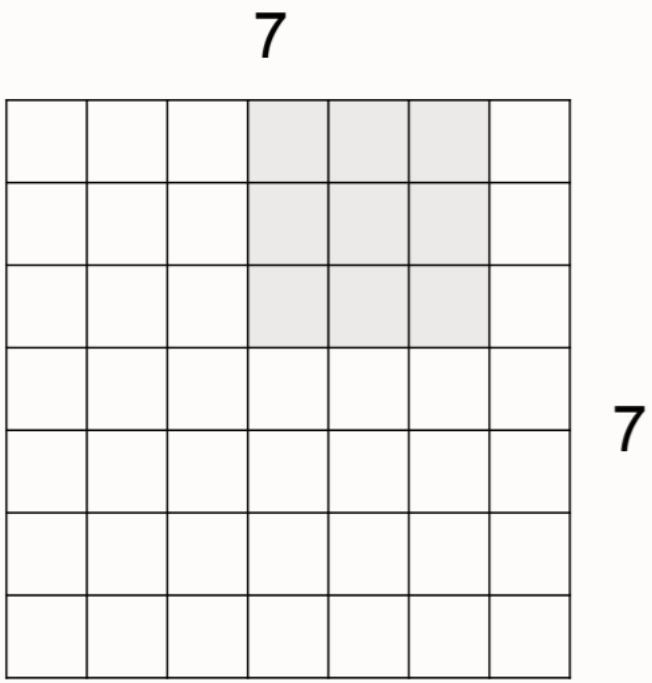
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolution Layer (15)

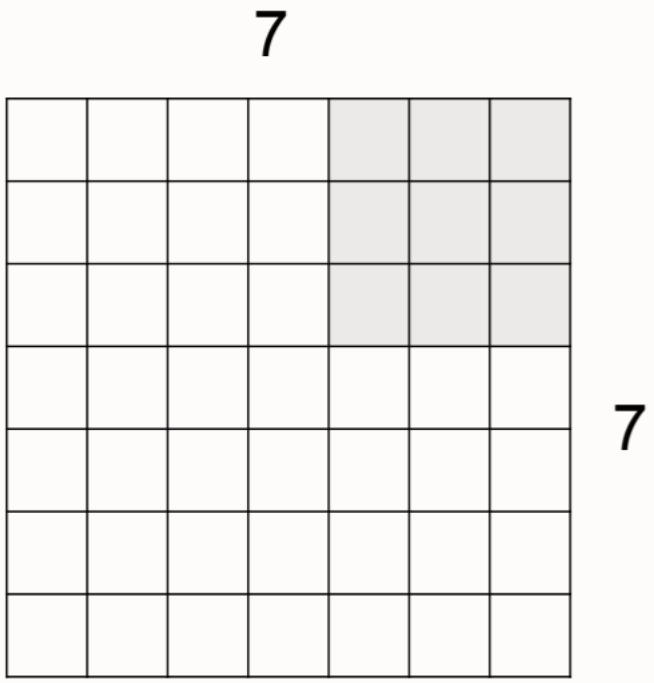
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolution Layer (16)

A closer look at spatial dimensions:

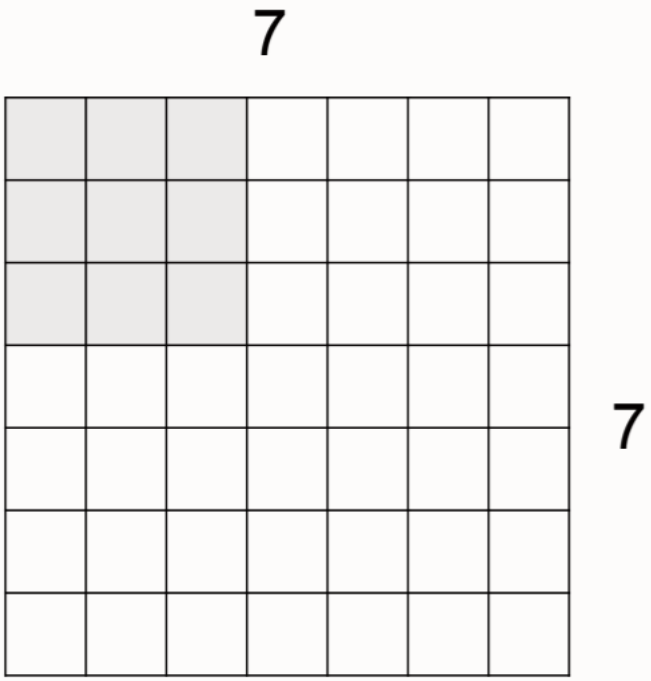


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Convolution Layer (17)

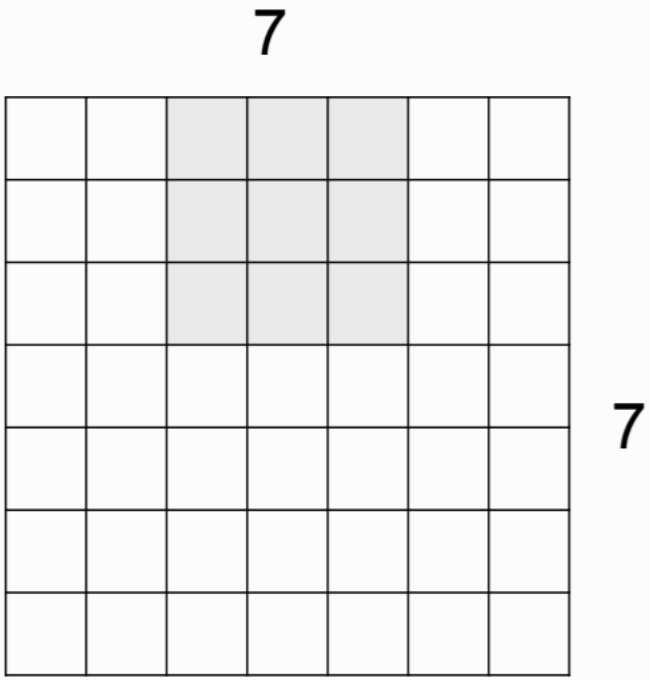
A closer look at spatial dimensions:



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

Convolution Layer (18)

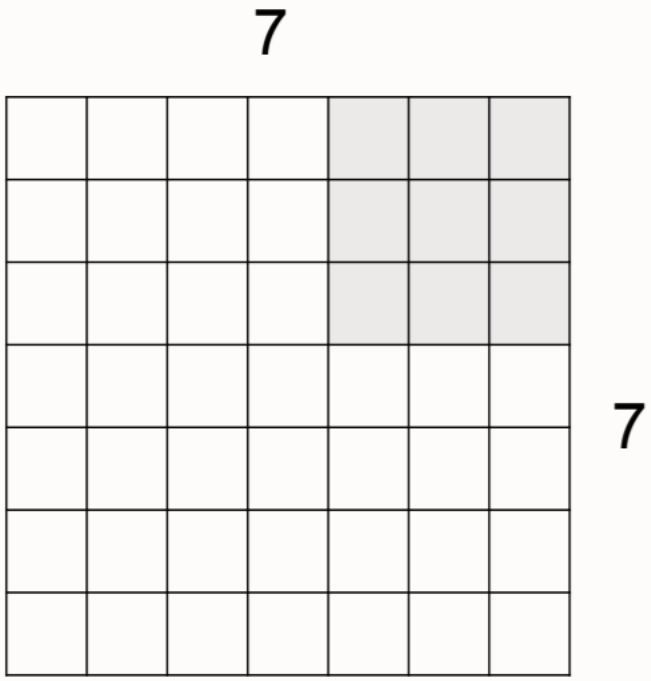
A closer look at spatial dimensions:



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

Convolution Layer (19)

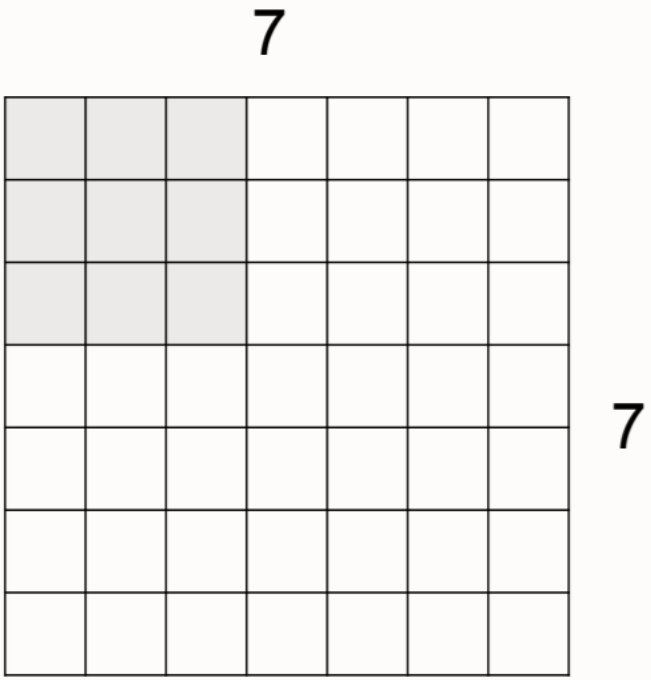
A closer look at spatial dimensions:



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

Convolution Layer (20)

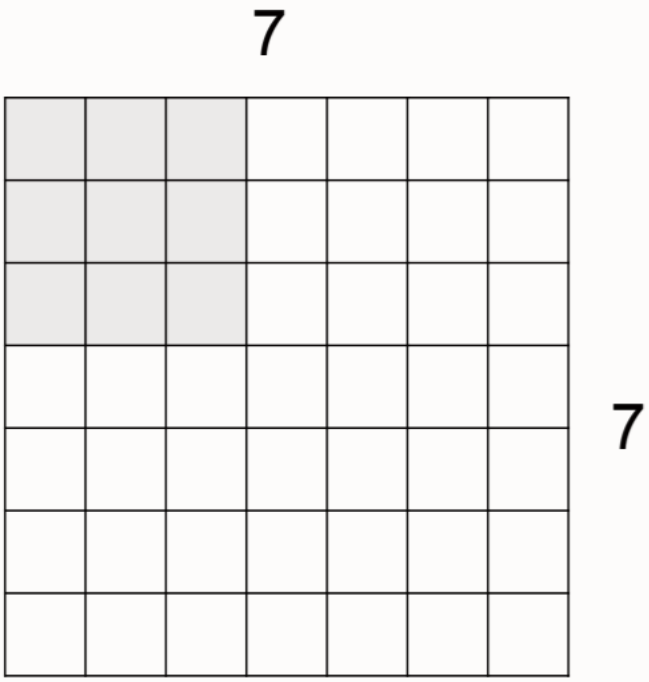
A closer look at spatial dimensions:



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

Convolution Layer (21)

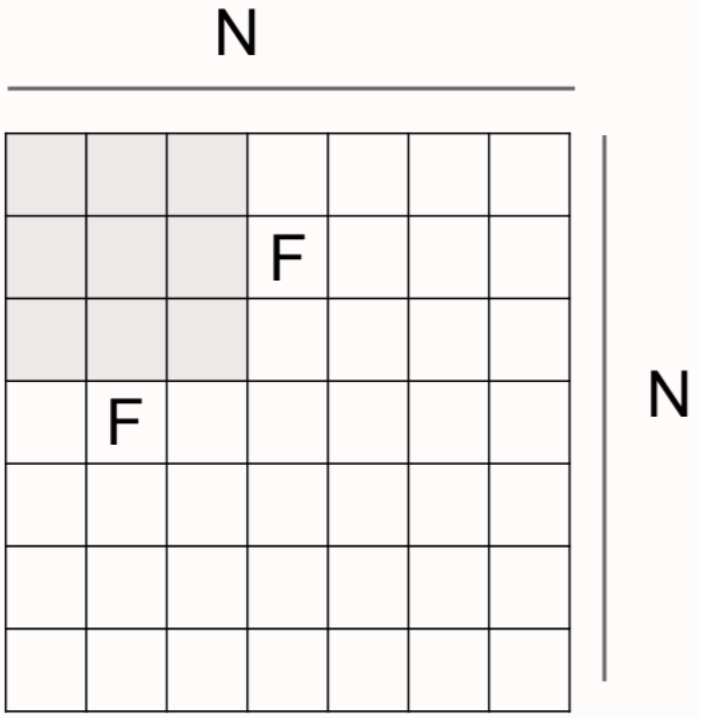
A closer look at spatial dimensions:



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 (22)

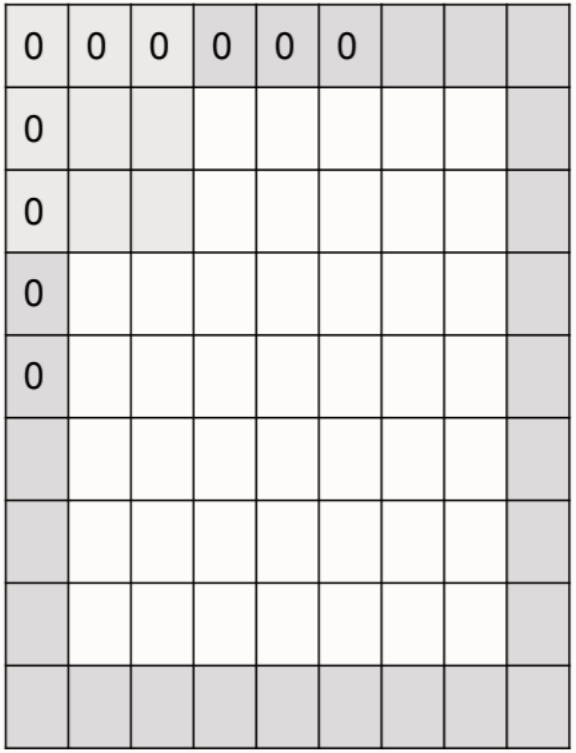


Output size:
(N - F) / stride + 1

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 = 2.33 :\

Convolution Layer (23)

In practice: Common to zero pad the border

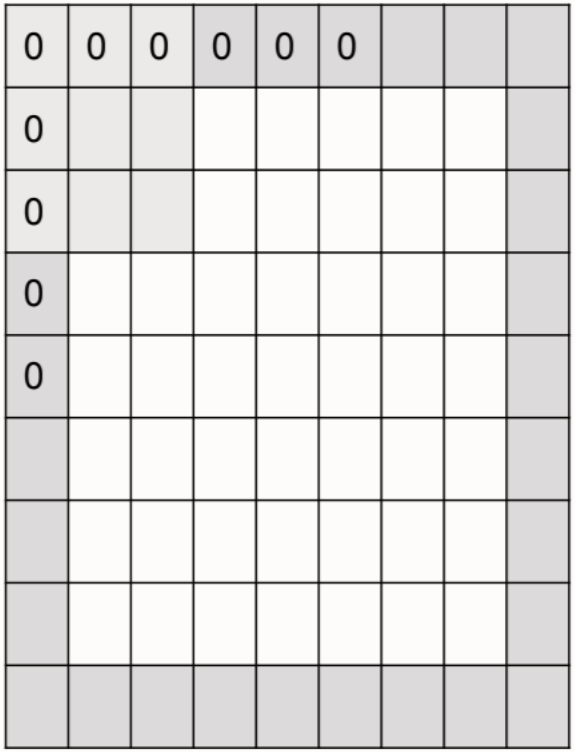


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

Convolution Layer (24)

In practice: Common to zero pad the border

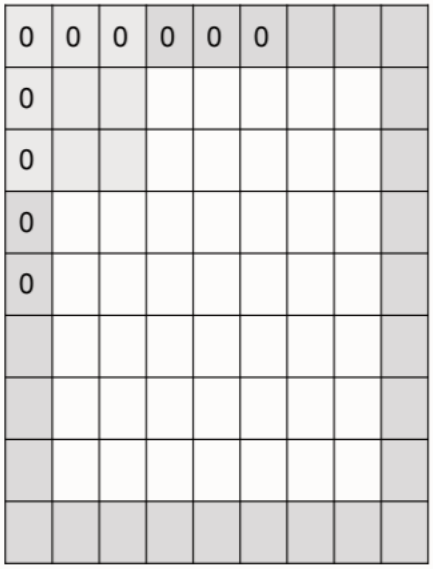


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Convolution Layer (25)

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?

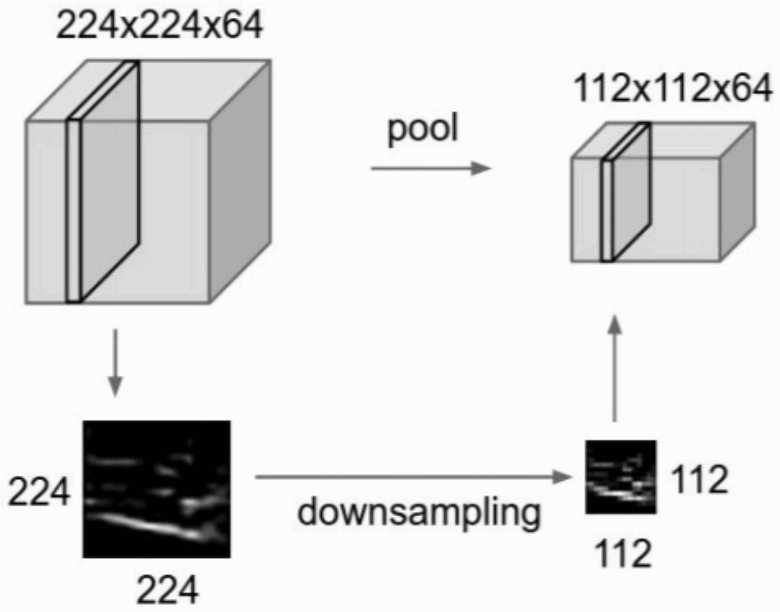
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

By using zero padding, you can **maintain the size of the image**.

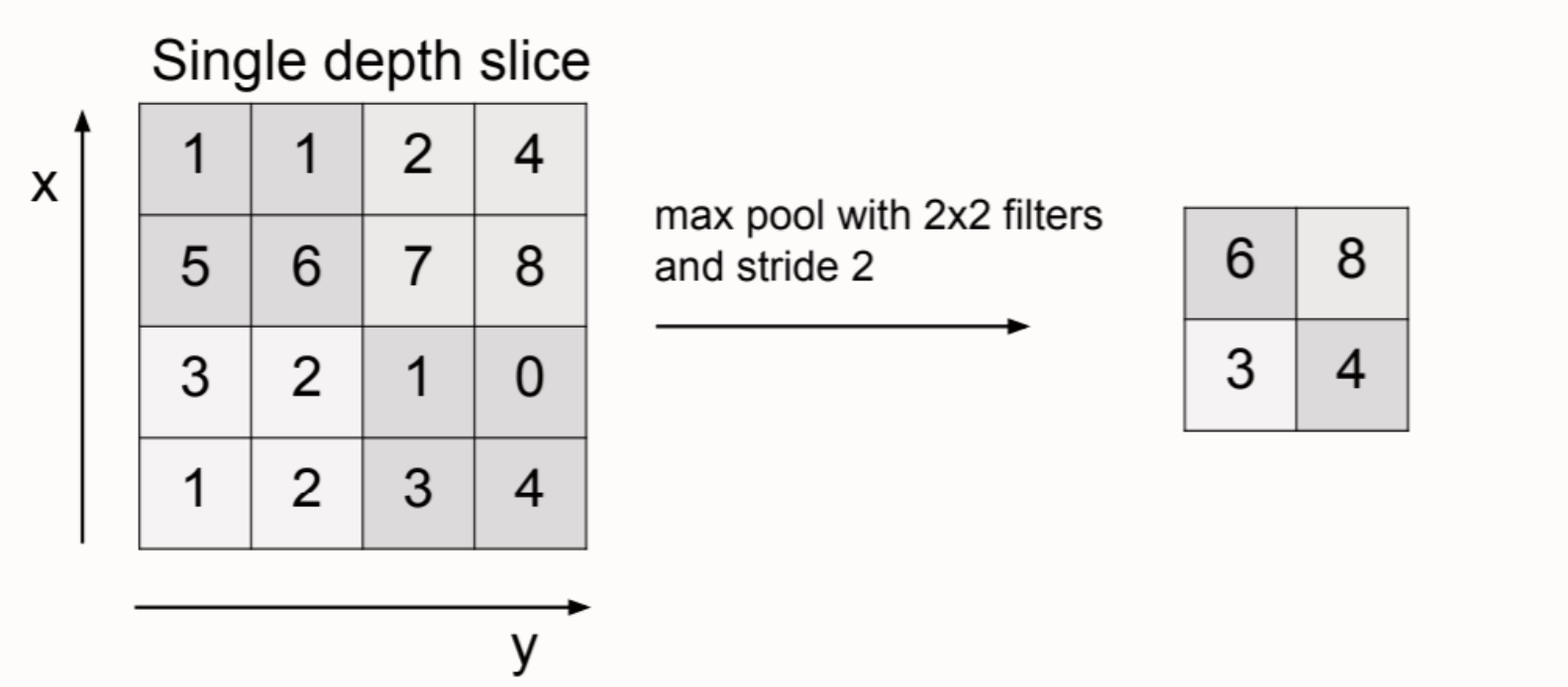
The filter allows you to see the feature map in **more detail.**

Pooling Layer

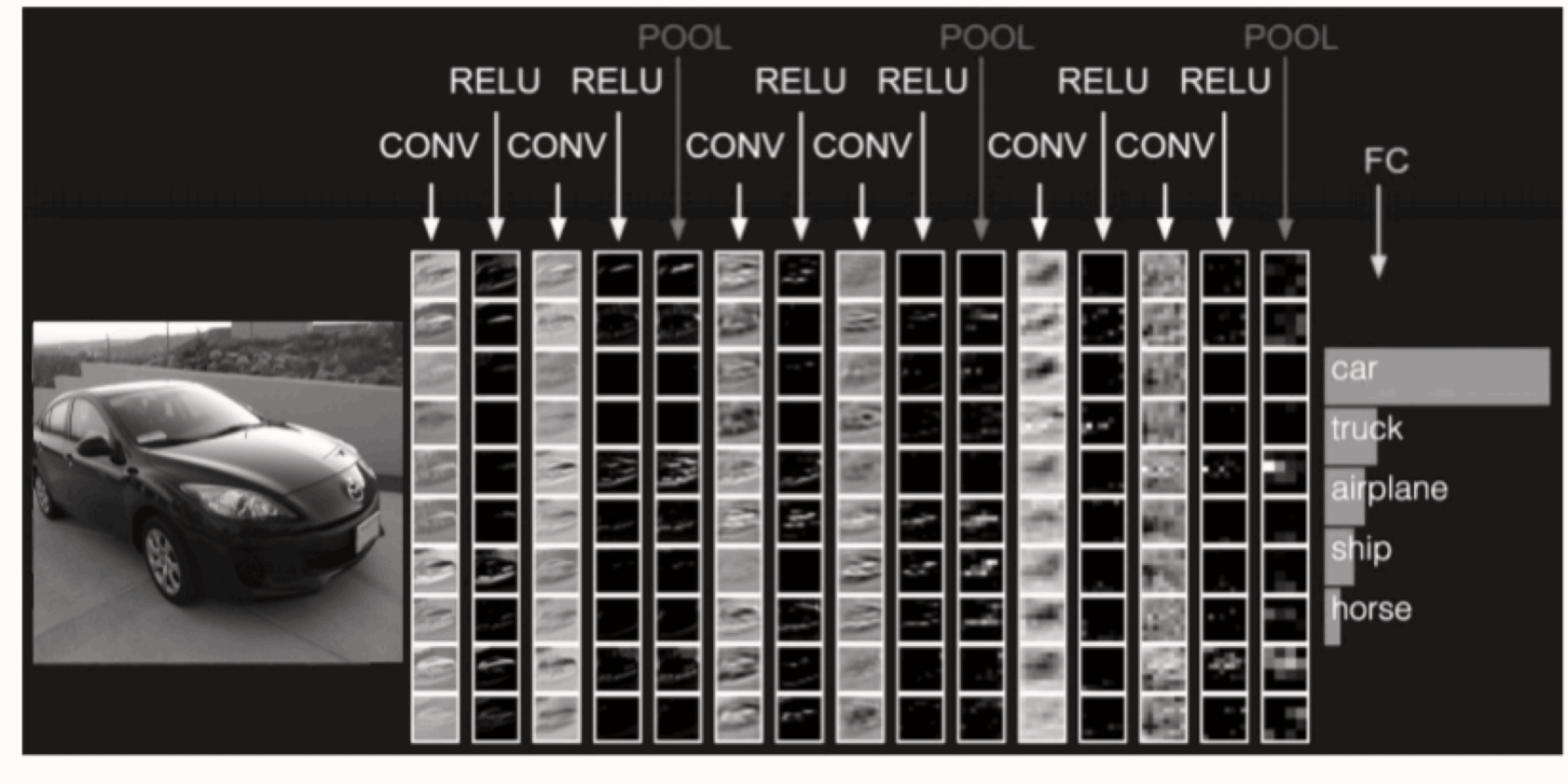
- makes the representations smaller and more manageable
- operates over each activation map independently:



Pooling Layer



Summary



Today's generic NN architecture.