Convolutional Neural Networks II

CS 4391 Introduction Computer Vision

Instructor Yu Xiang

The University of Texas at Dallas

Some slides of this lecture are courtesy Stanford CS231n

Fully Connected Layer

- What is the drawback of only using fully connected layers?

$$y = Wx$$

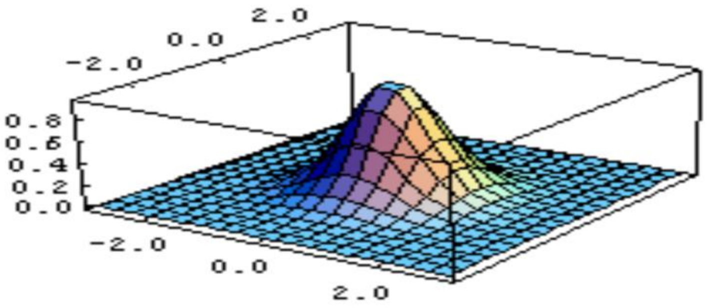
- Consider an image with 640 x 480
  - x is with dimension 307,200
  - The weight matrix of the fully connect layer is too large

Convolutional Layers

- Consist of convolutional filters
- Share weights among different image locations

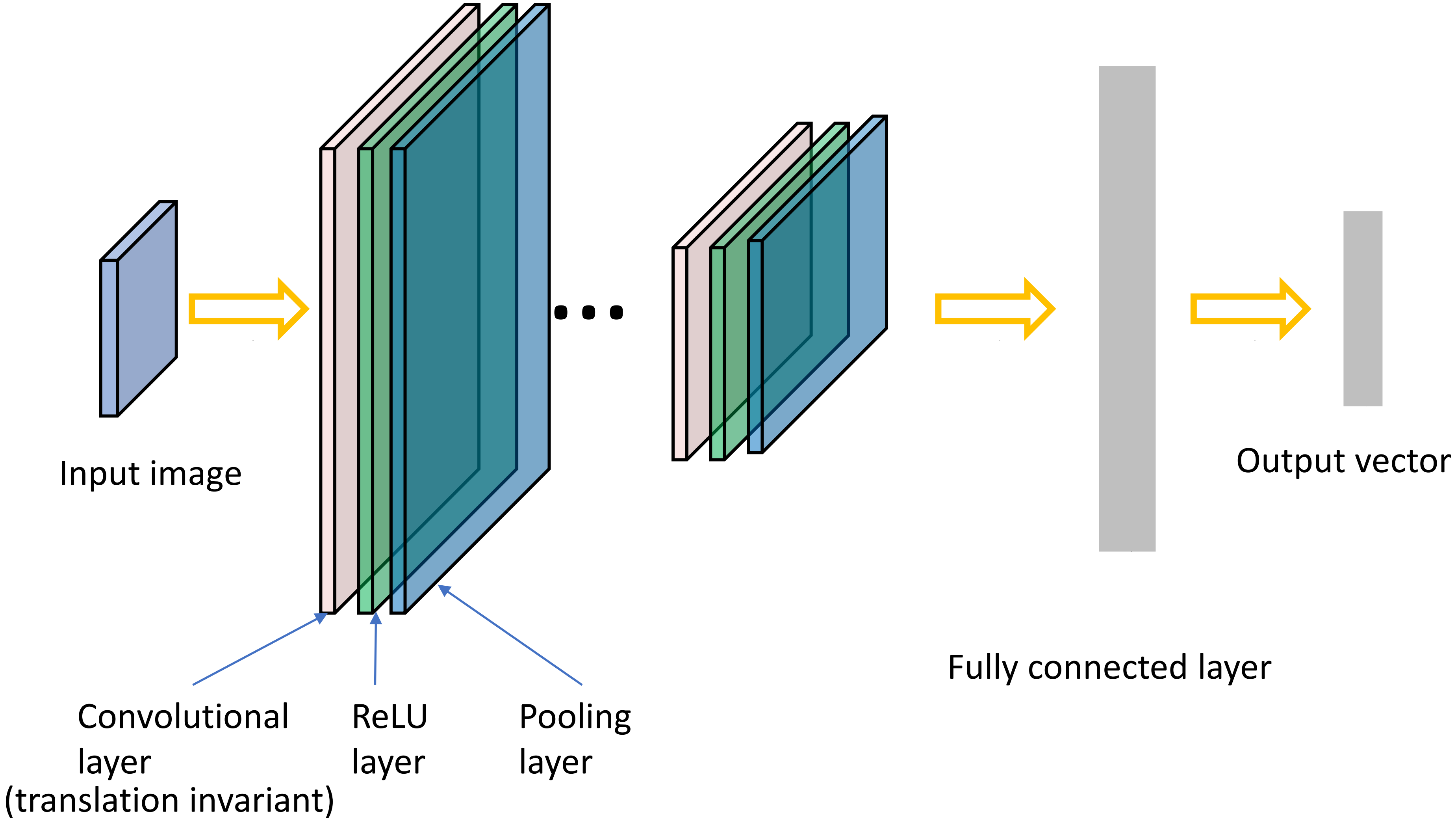
$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian Filter

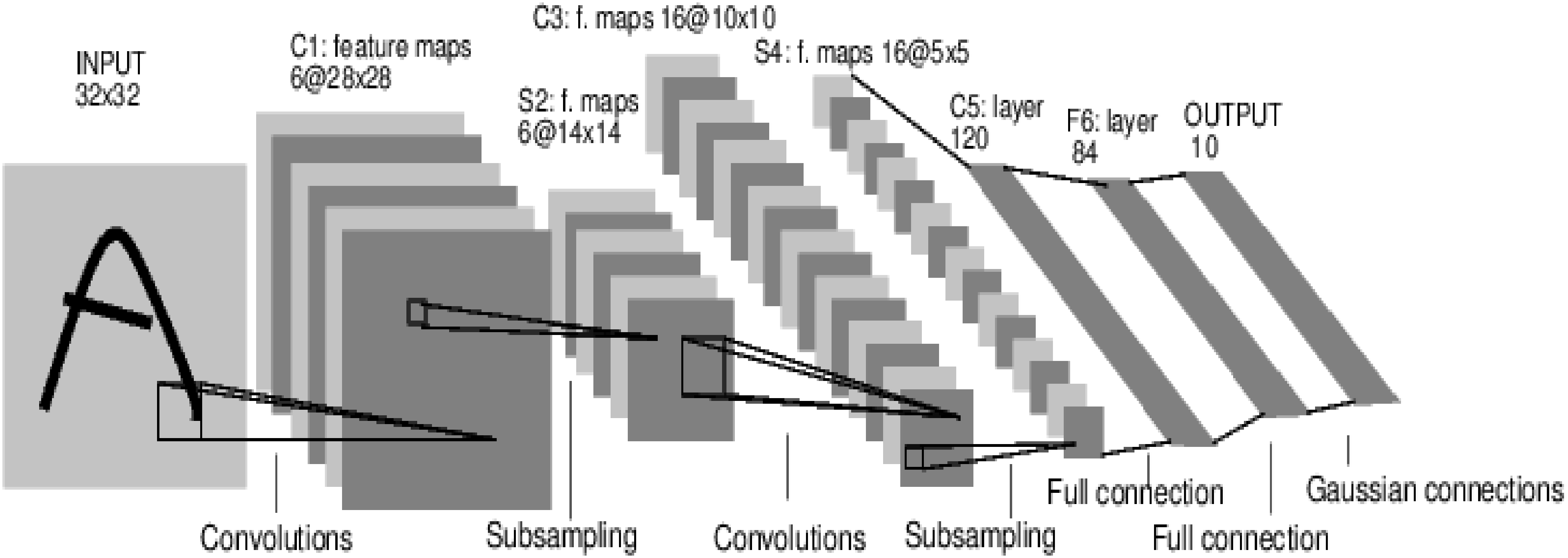


Learn the weights!

Convolutional Neural Networks



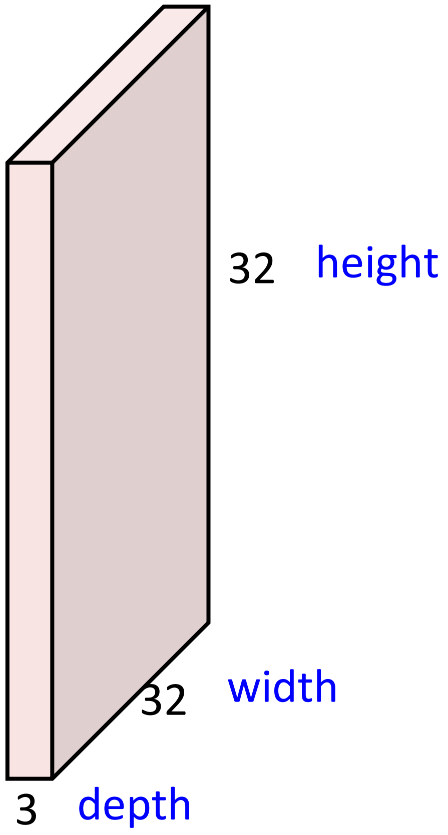
Convolutional Neural Networks



[LeNet-5, LeCun 1980]

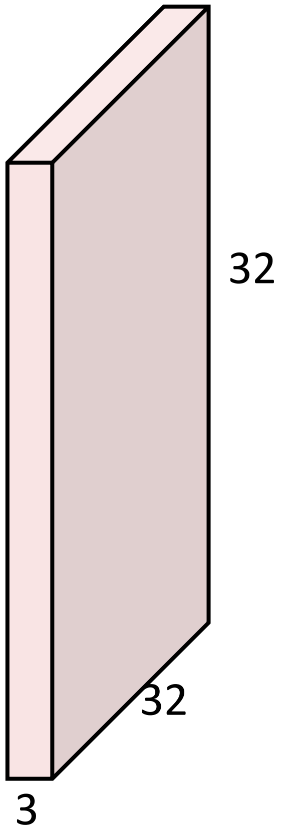
Convolutional Layer

32x32x3 image

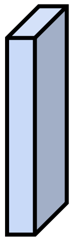


Convolutional Layer

32x32x3 image



5x5x3 filter

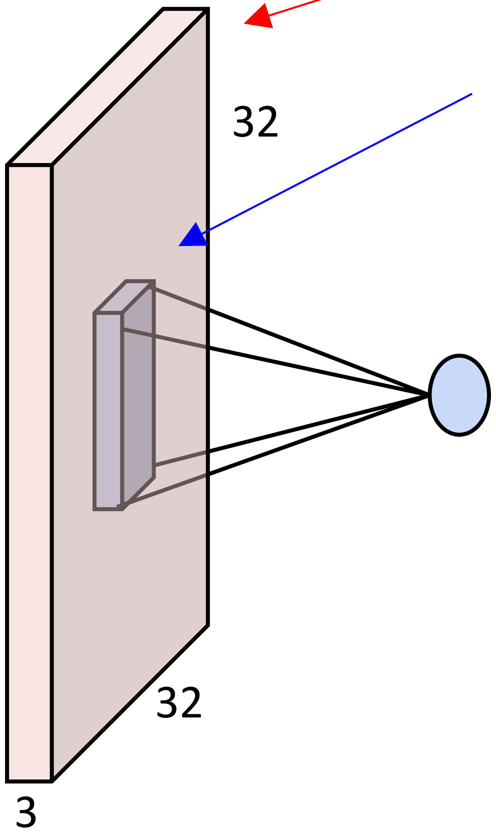


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolutional Layer

32x32x3 image

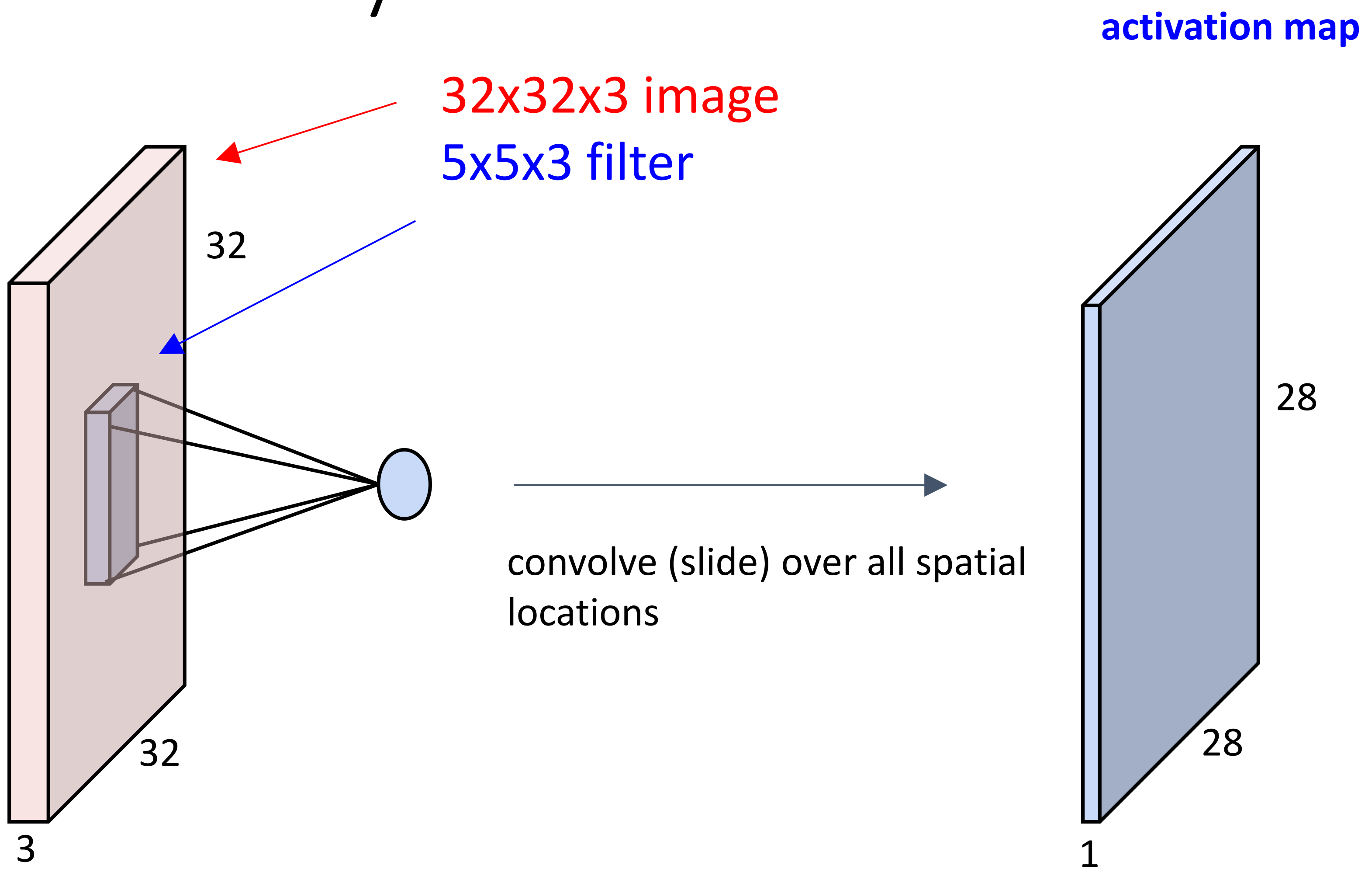
5x5x3 filter w



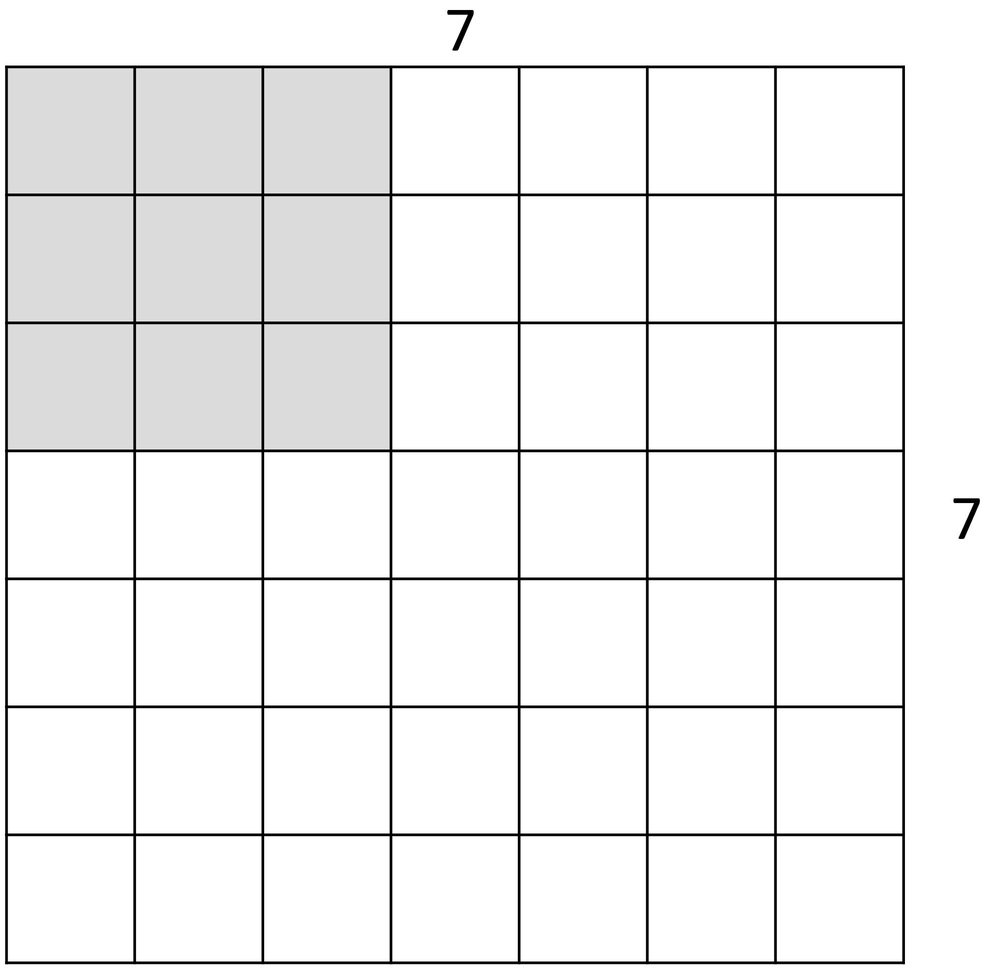
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolutional Layer



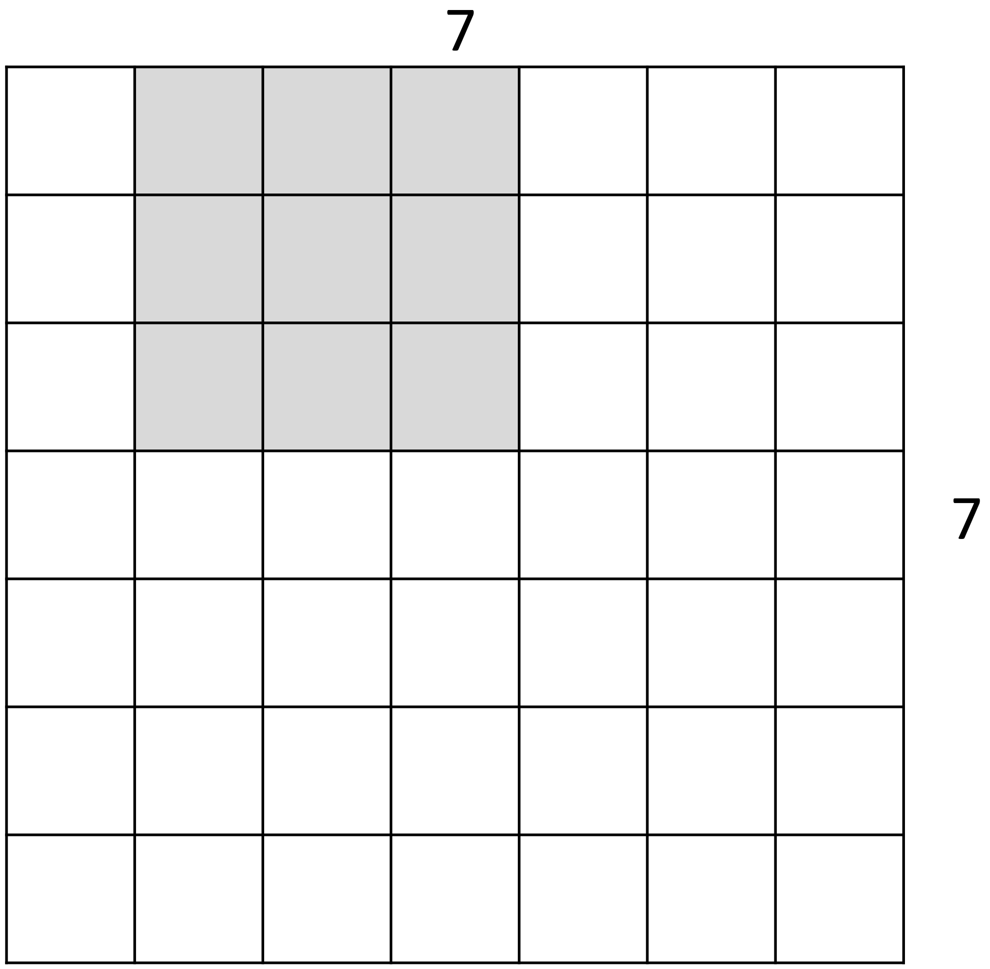
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

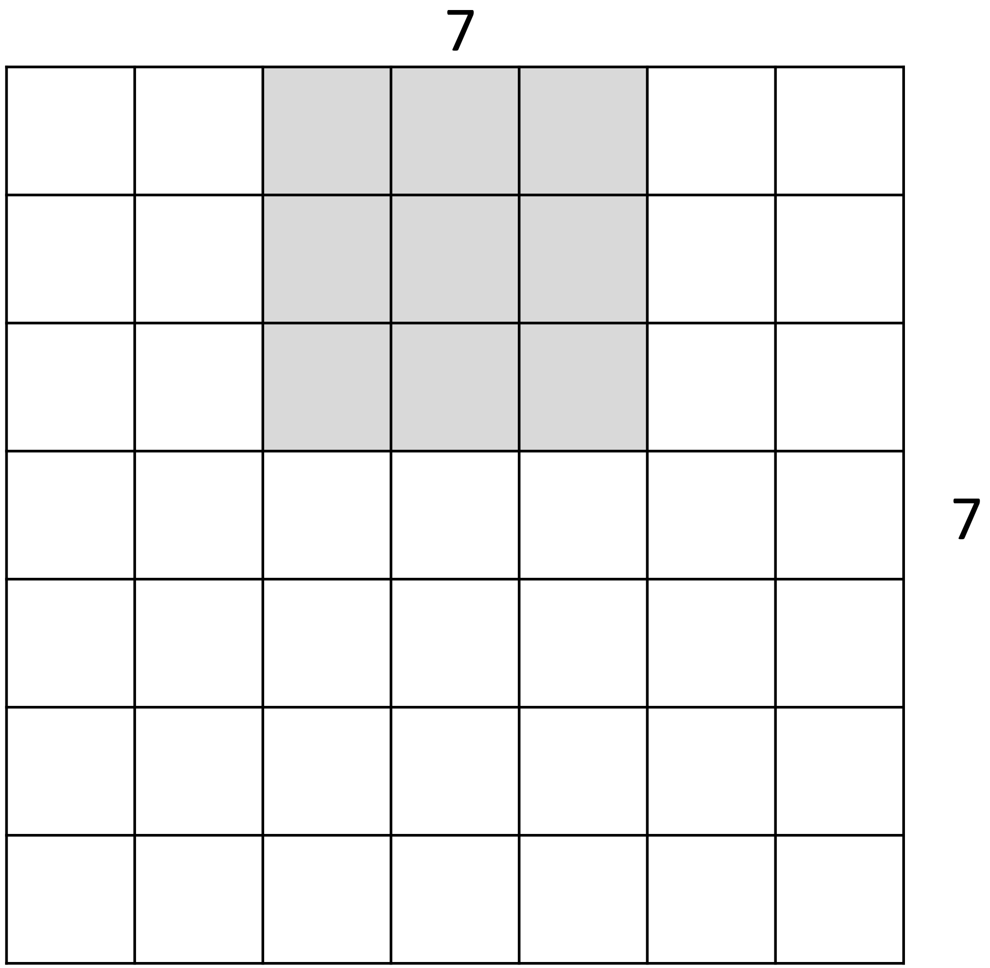
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

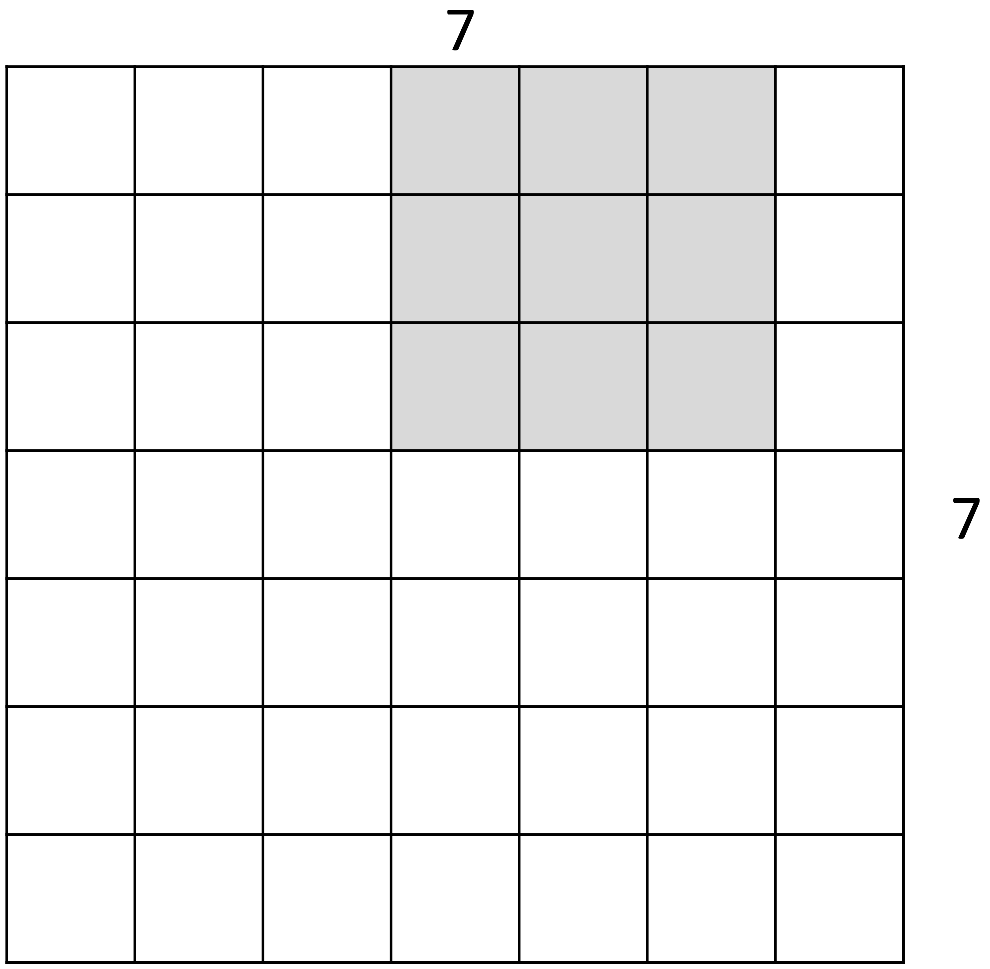
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

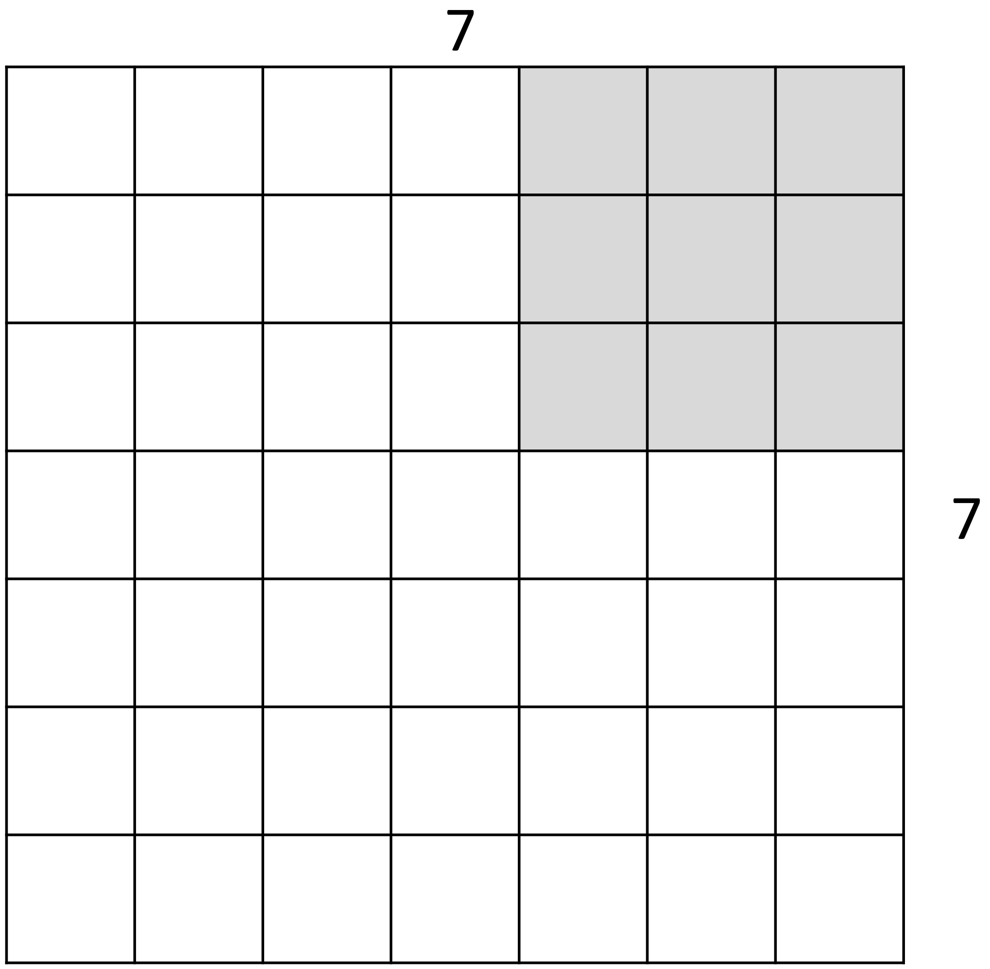
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

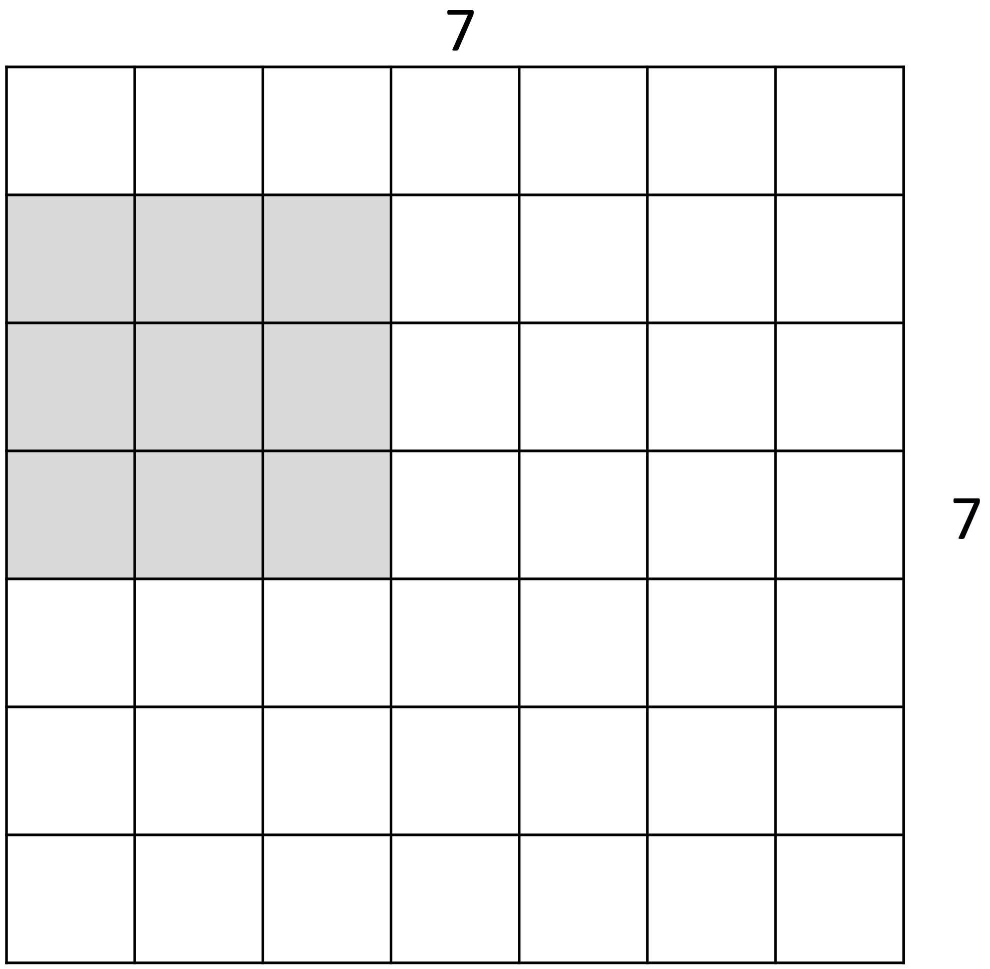
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

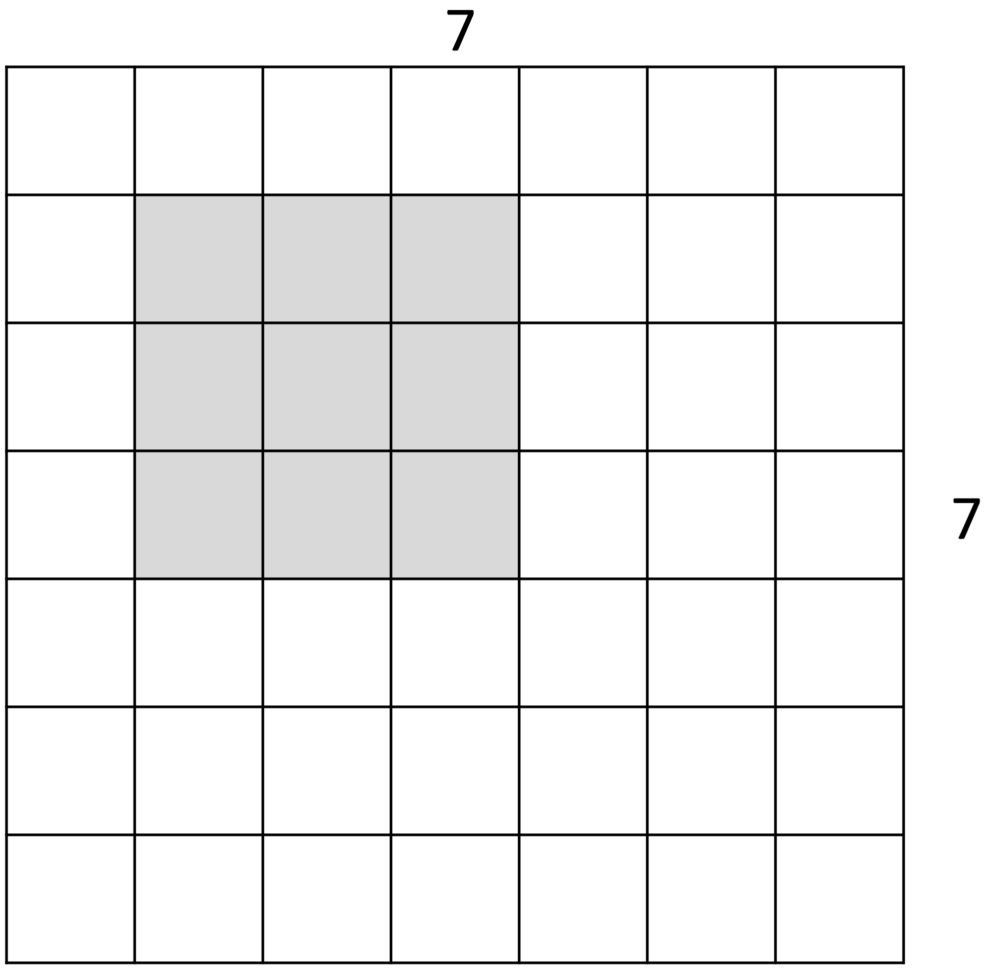
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

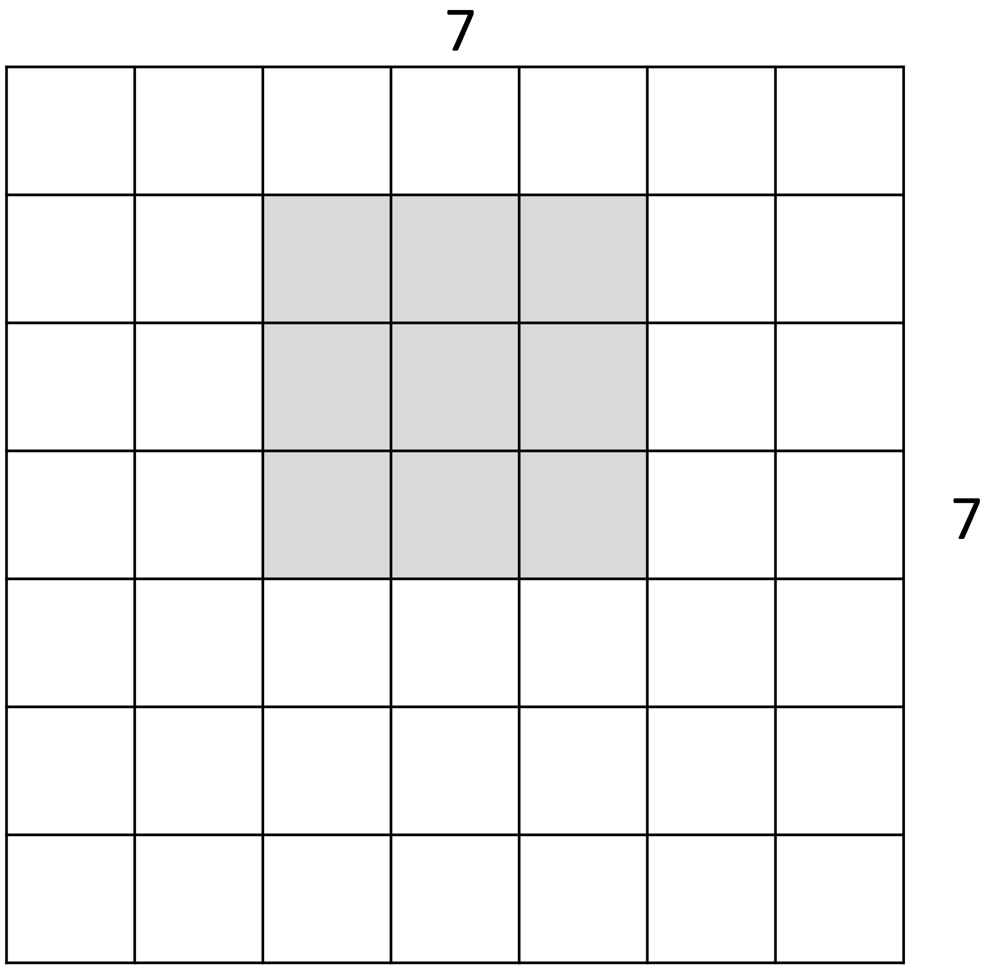
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

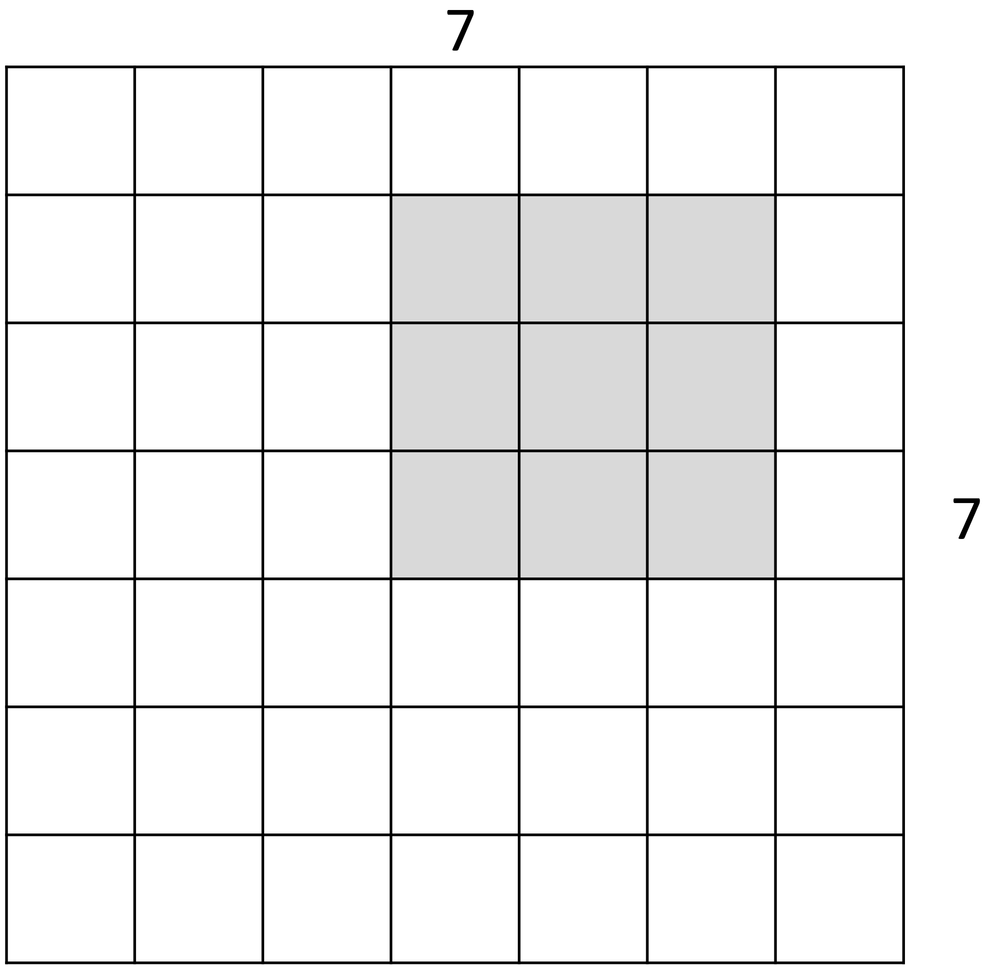
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

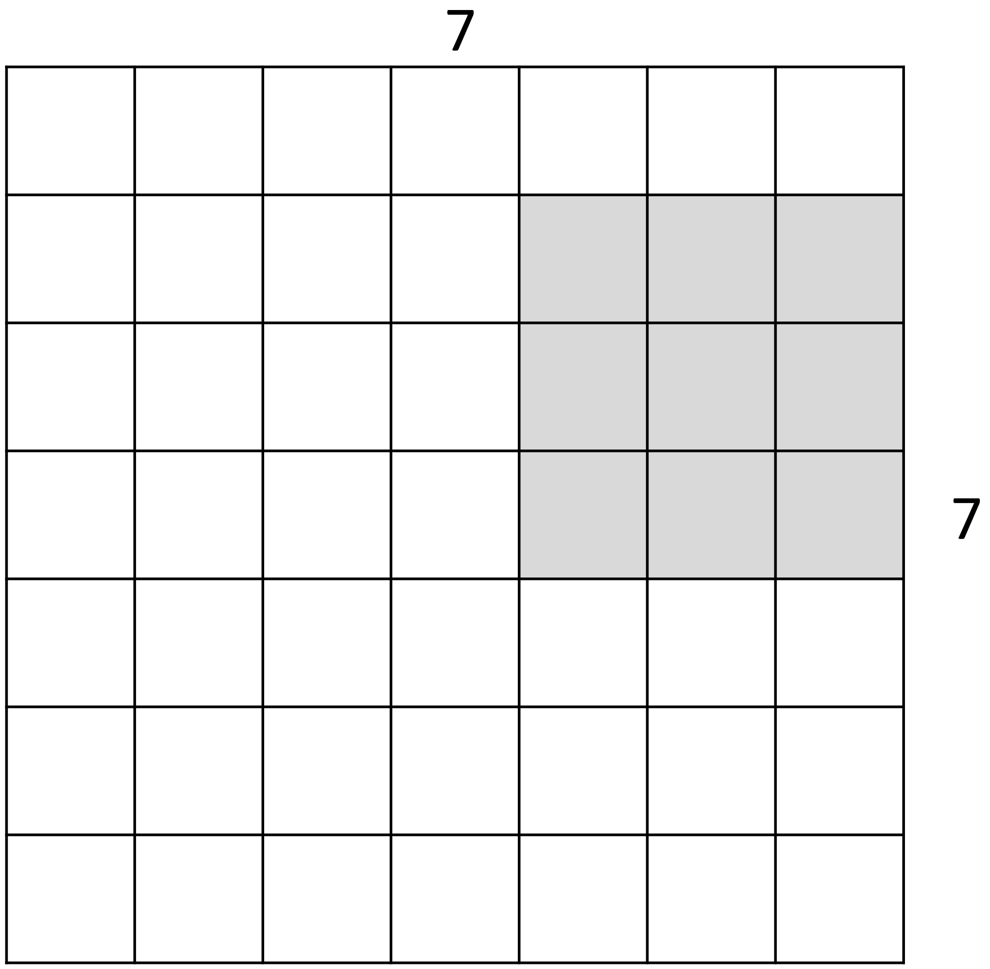
Convolutional Layer



A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

Convolutional Layer

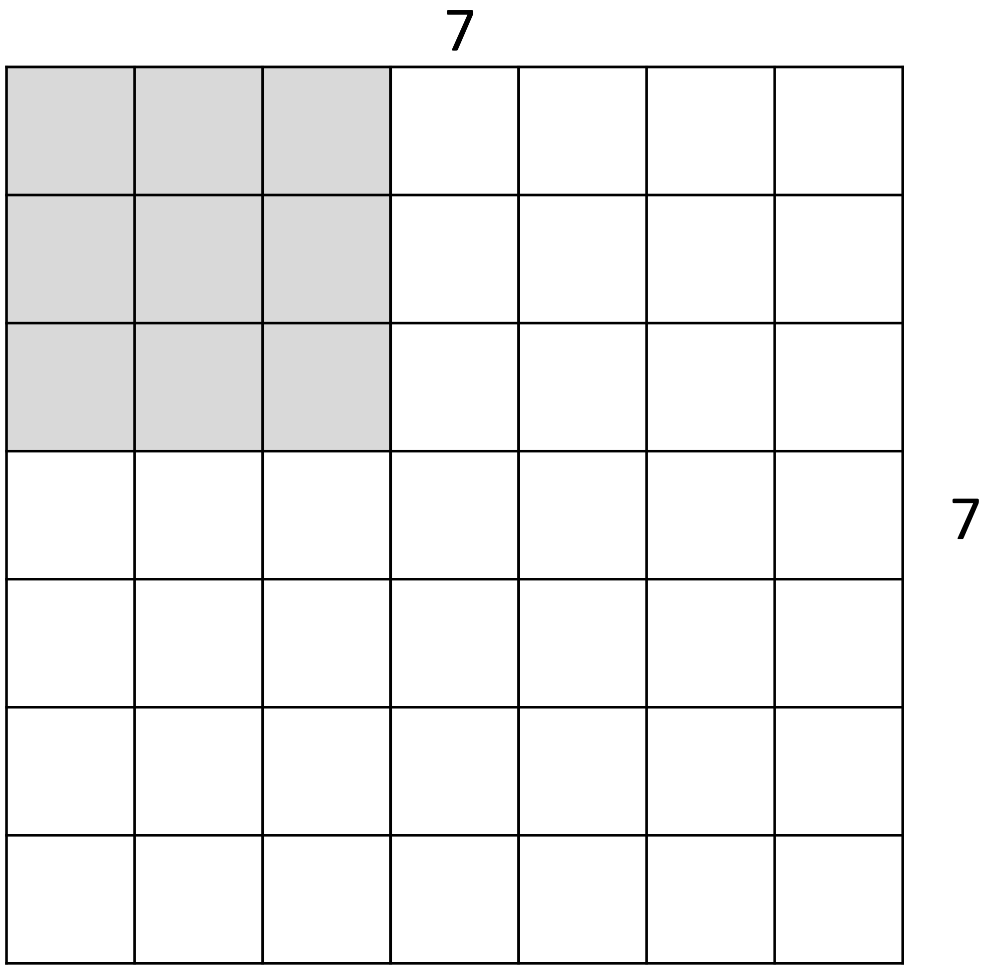


A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

=> 5x5 output

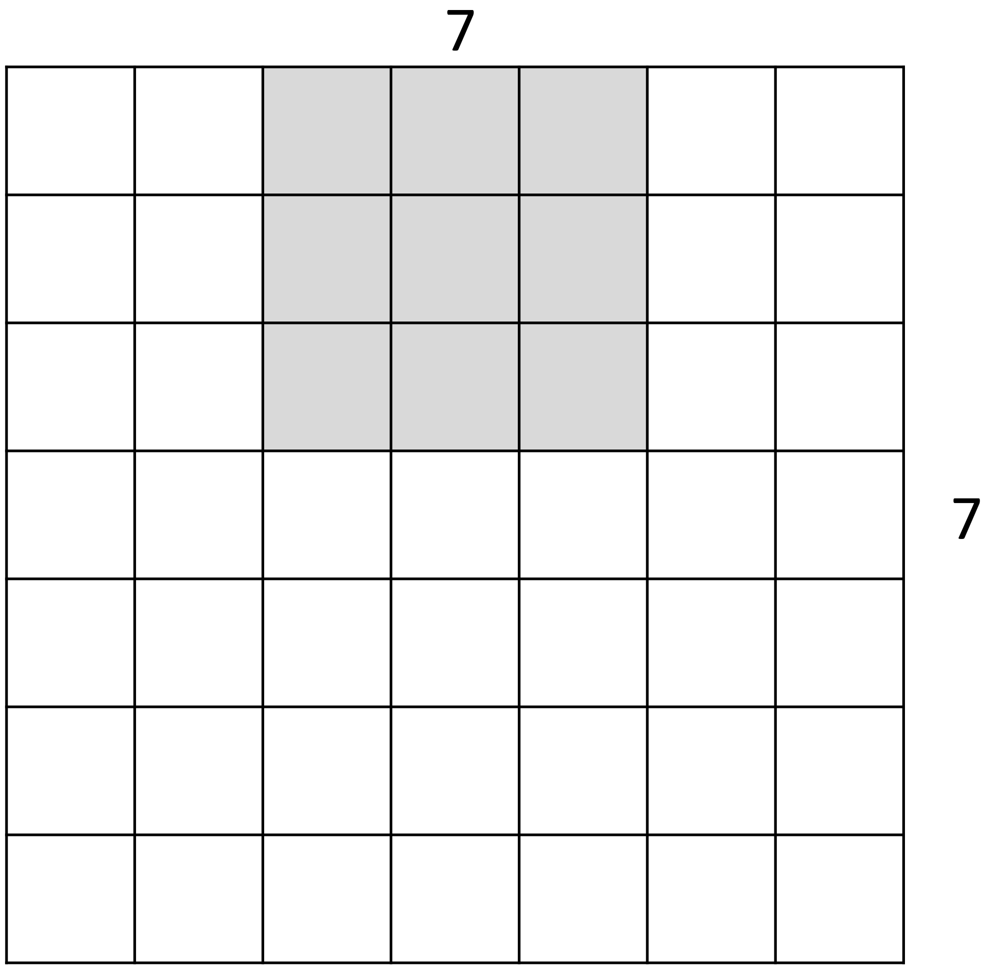
Convolutional Layer



A closer look at spatial dimensions:

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

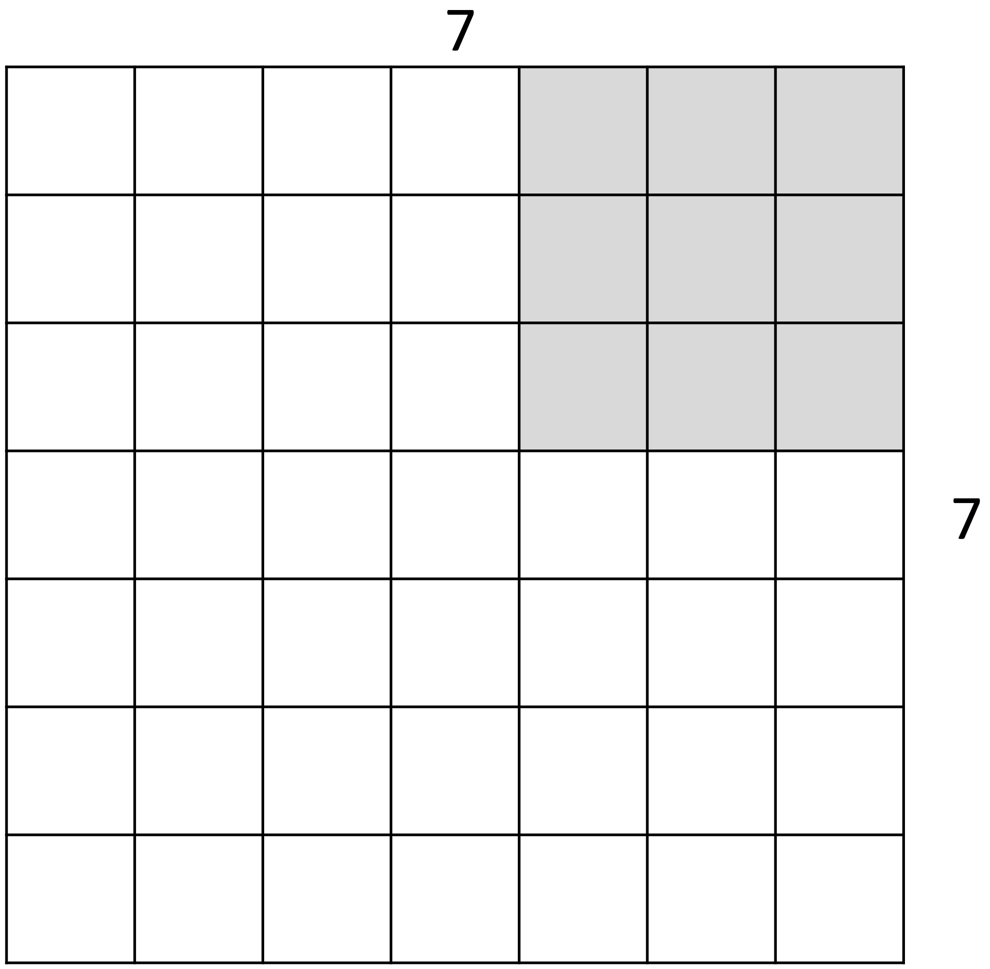
Convolutional Layer



A closer look at spatial dimensions:

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

Convolutional Layer

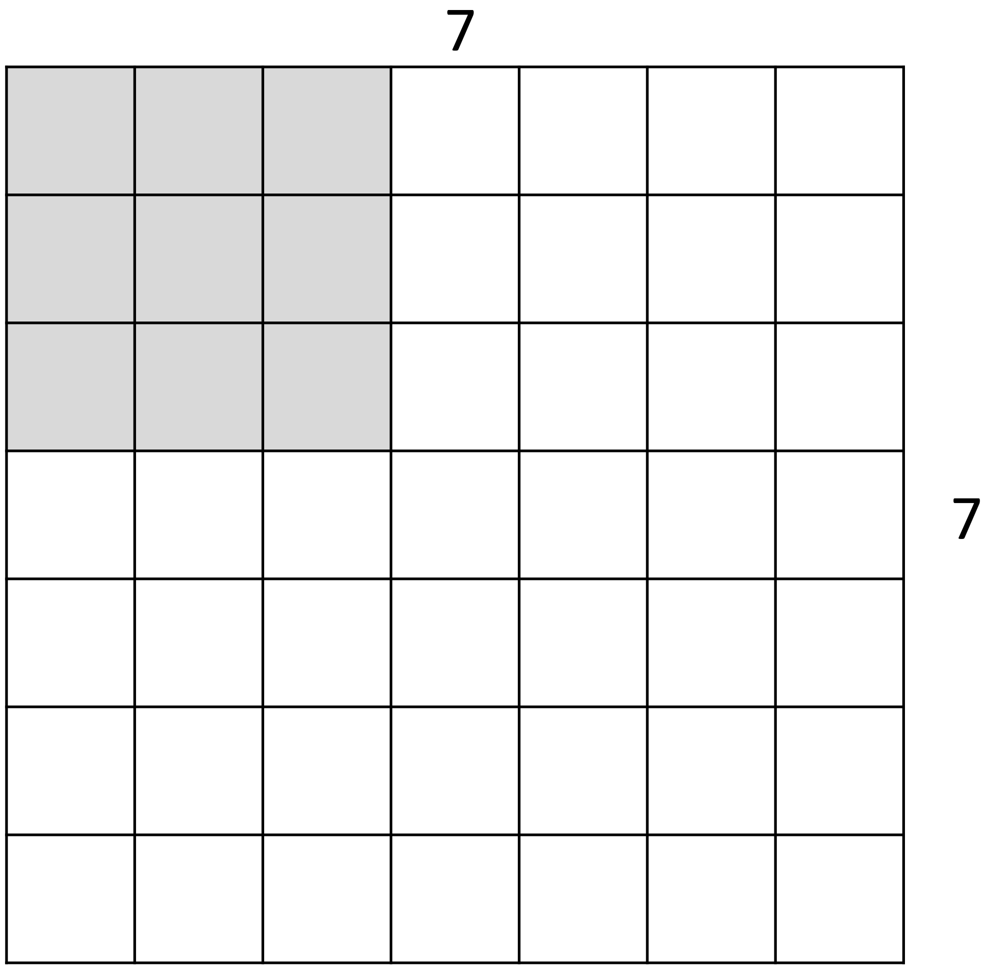


A closer look at spatial dimensions:

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

=> 3x3 output!

Convolutional Layer

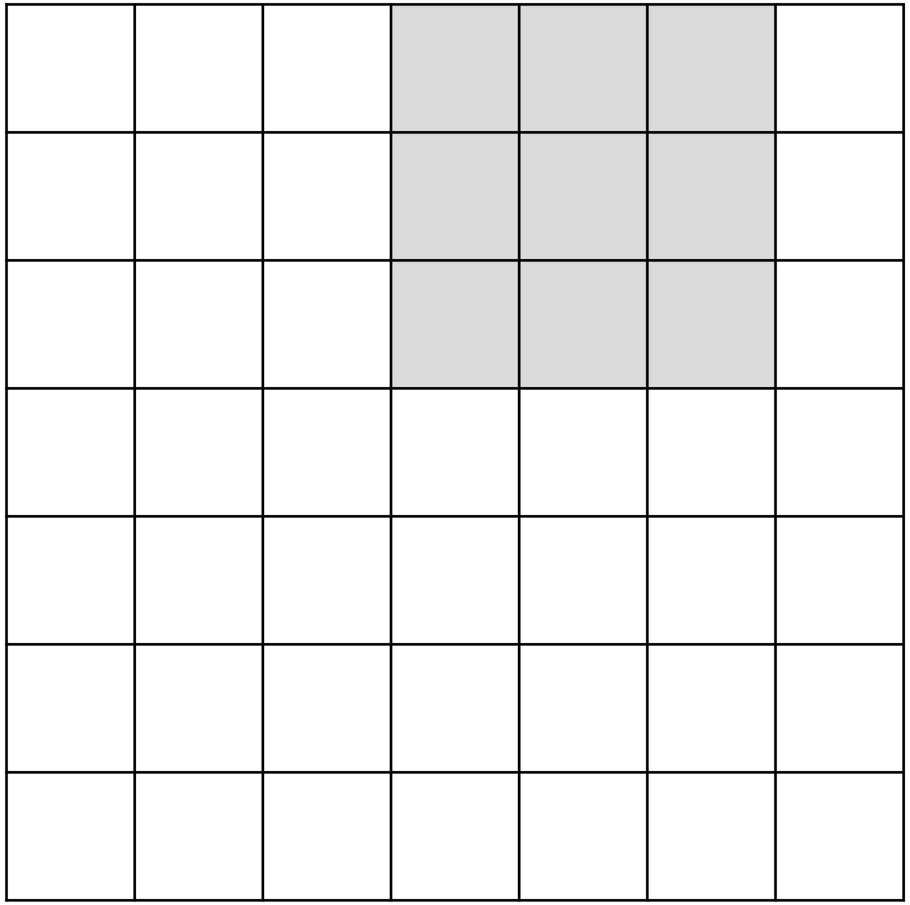


A closer look at spatial dimensions:

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

Convolutional Layer

7



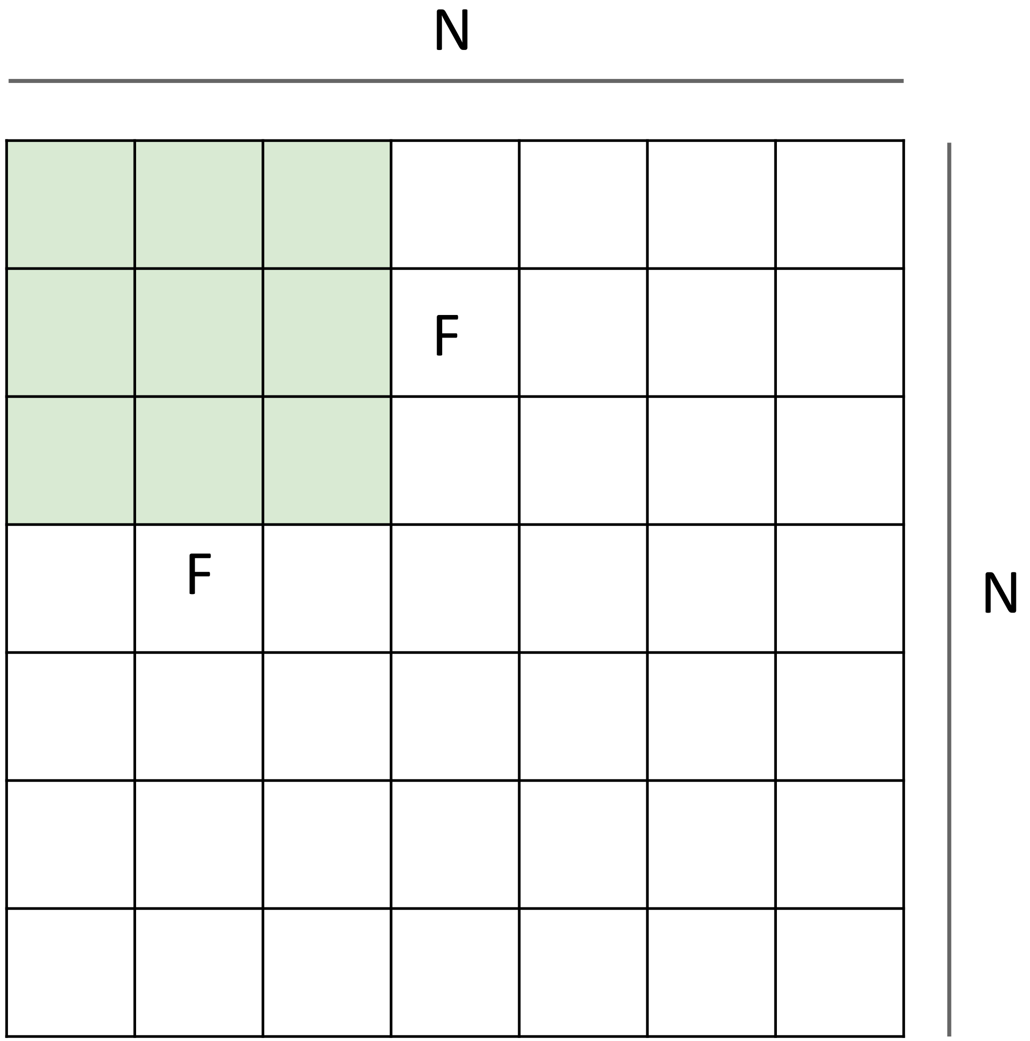
7

A closer look at spatial dimensions:

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

doesn't fit!
cannot apply 3x3 filter on 7x7 input with stride 3.

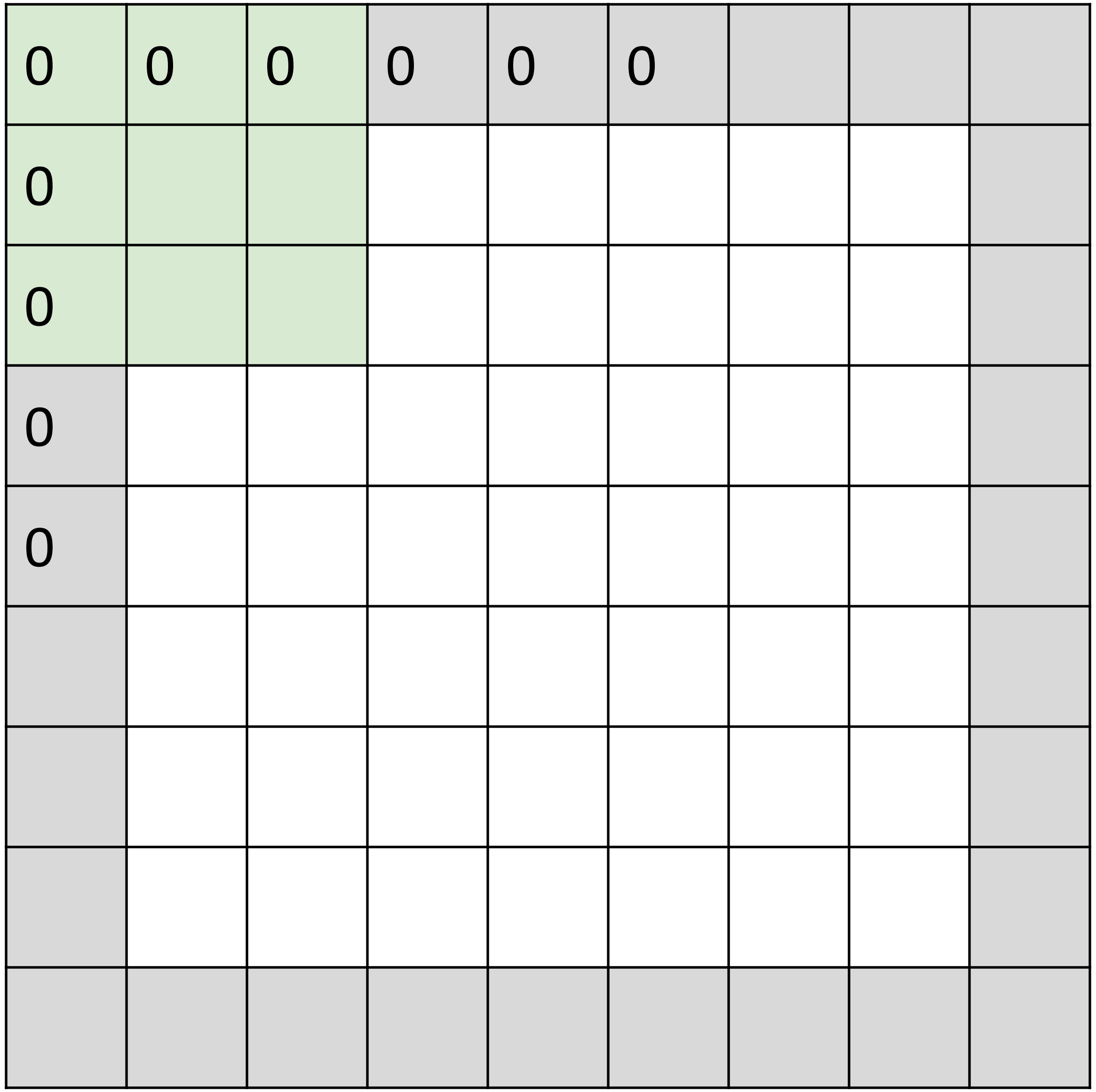
Convolutional Layer



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

Convolutional Layer

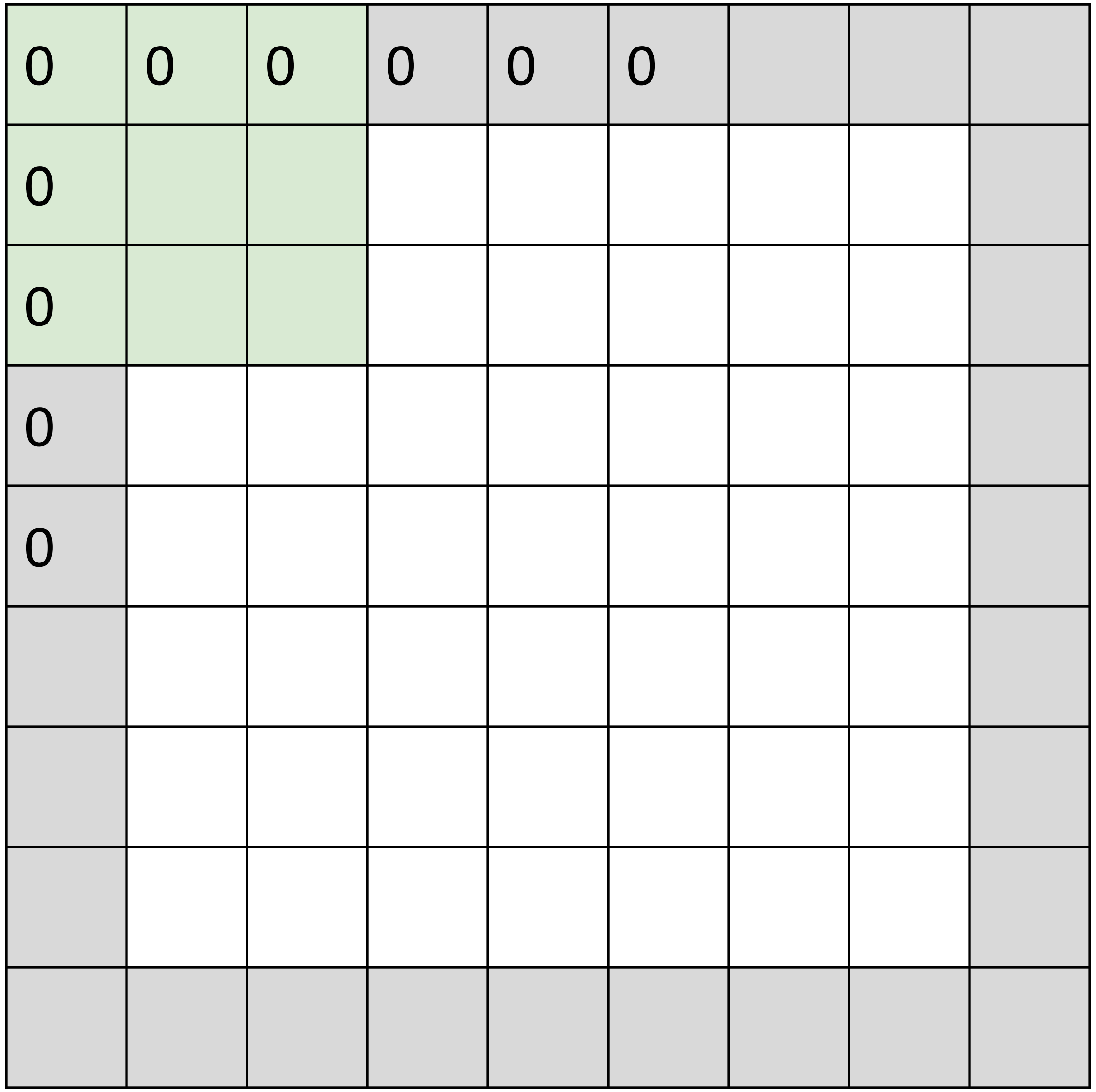


In practice: Common to zero pad the border

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

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

Convolutional Layer

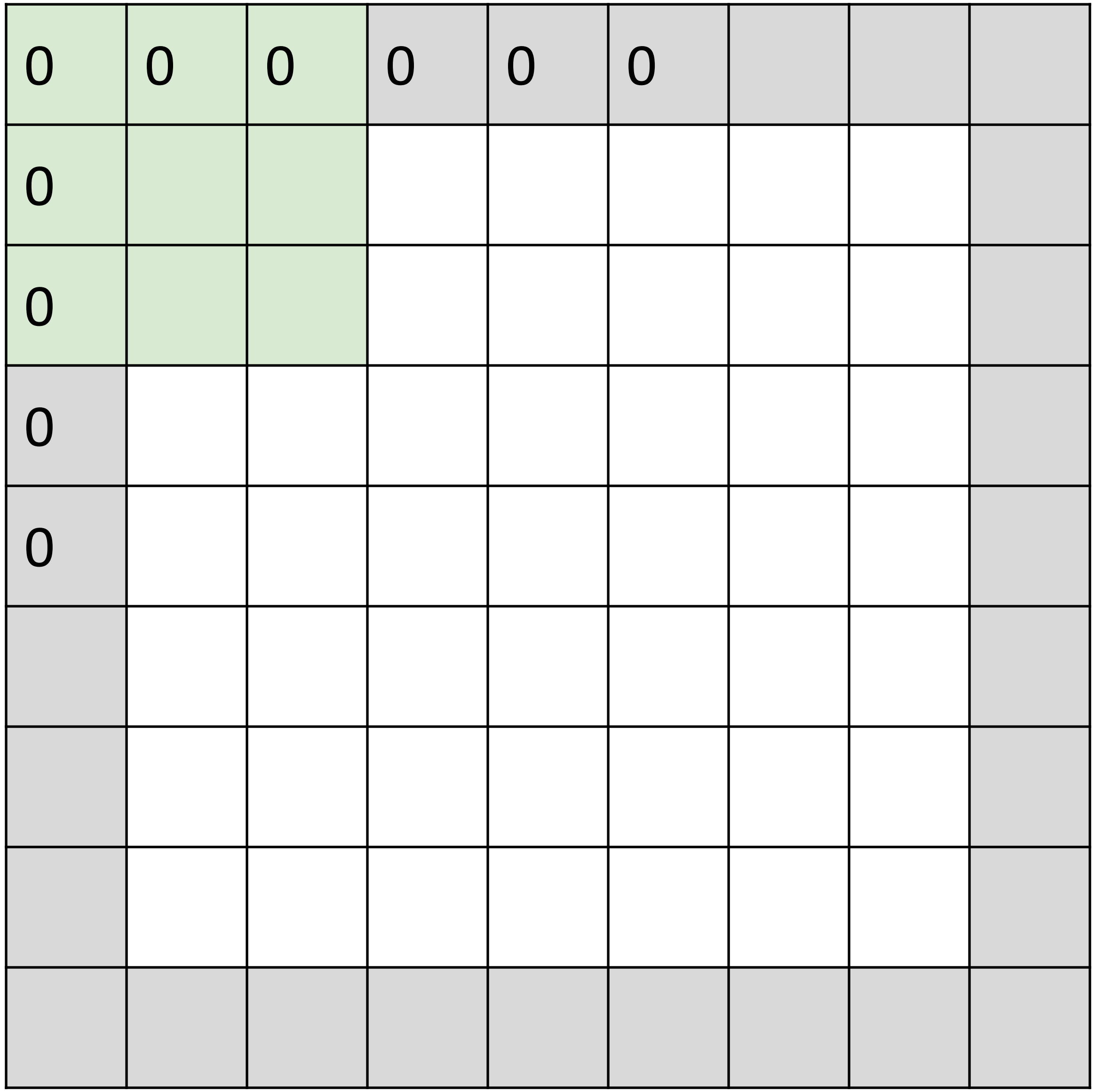


In practice: Common to zero pad the border

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

7x7 output!

Convolutional Layer

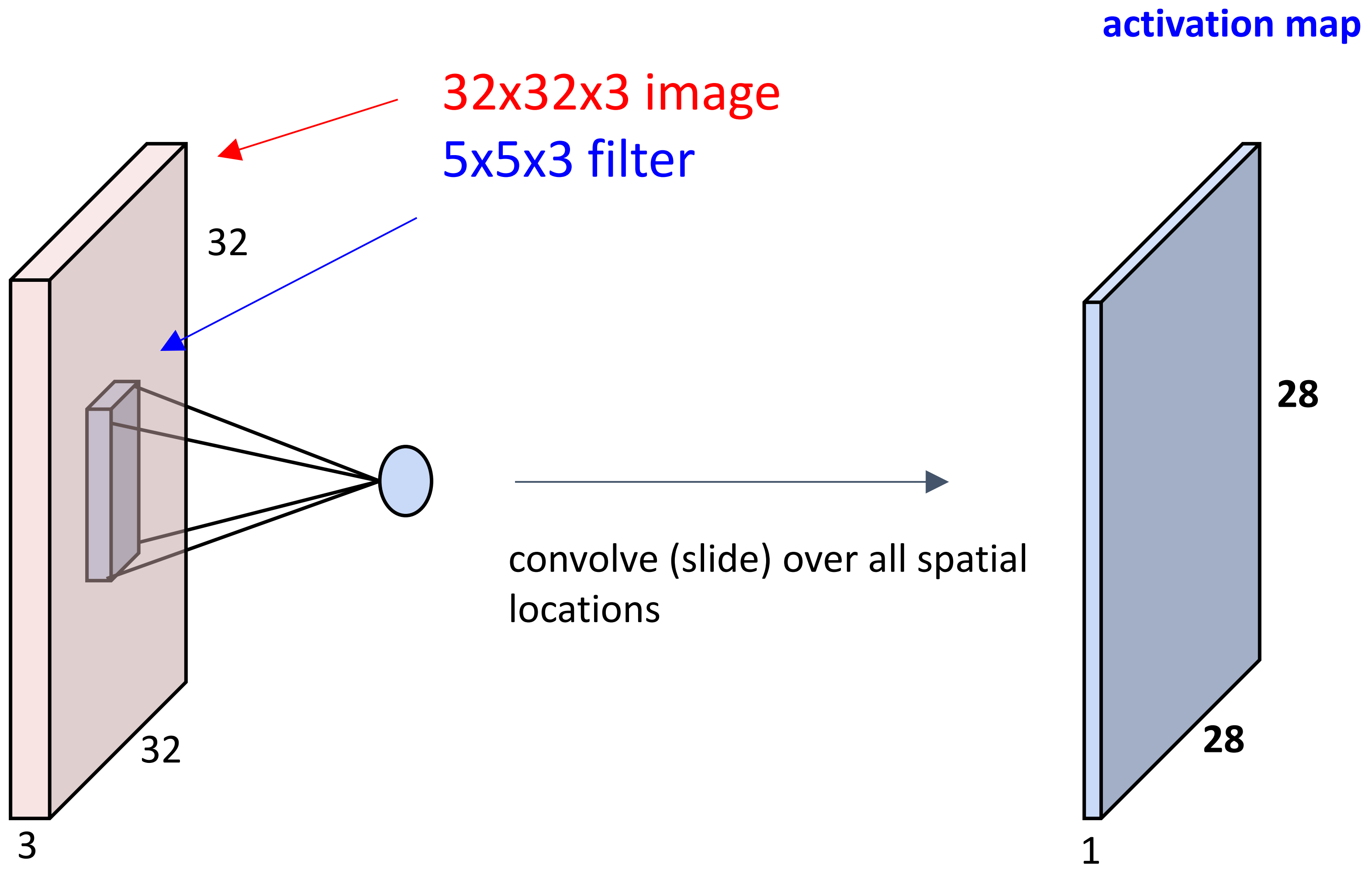


In practice: Common to zero pad the border

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)

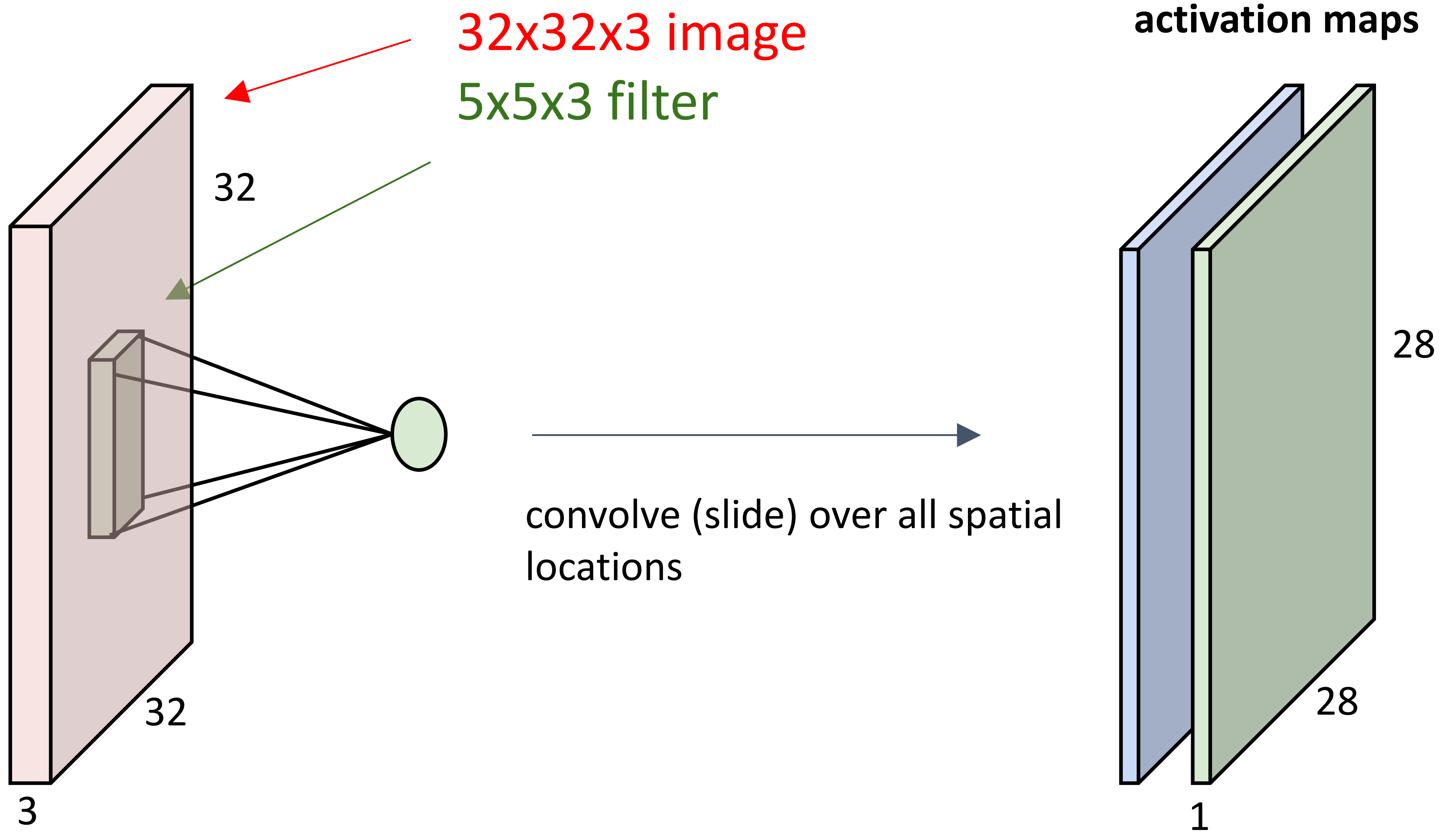
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Convolutional Layer

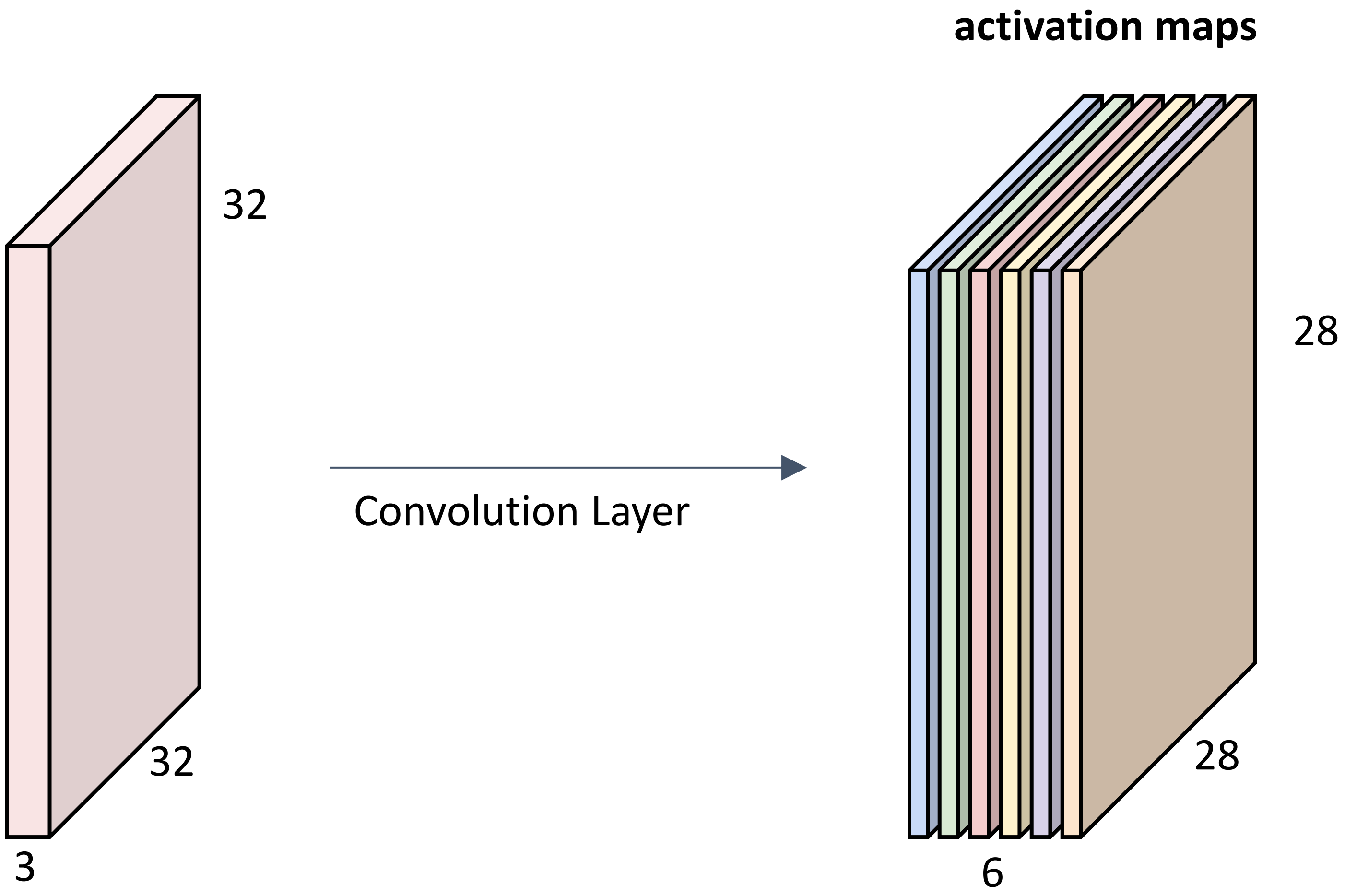


Convolutional Layer

consider a second, green filter



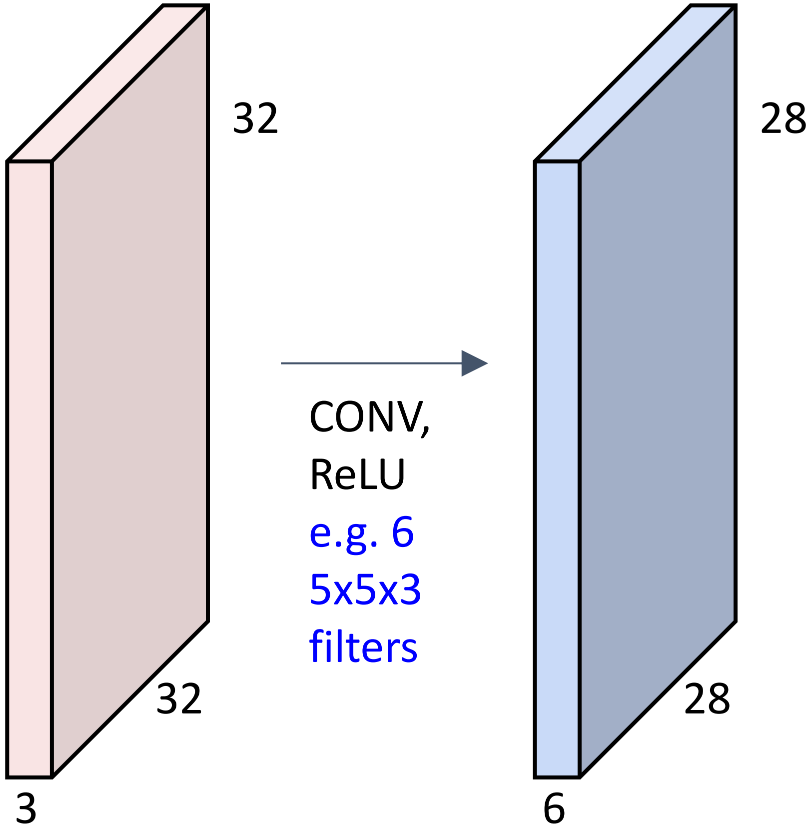
Convolutional Layer



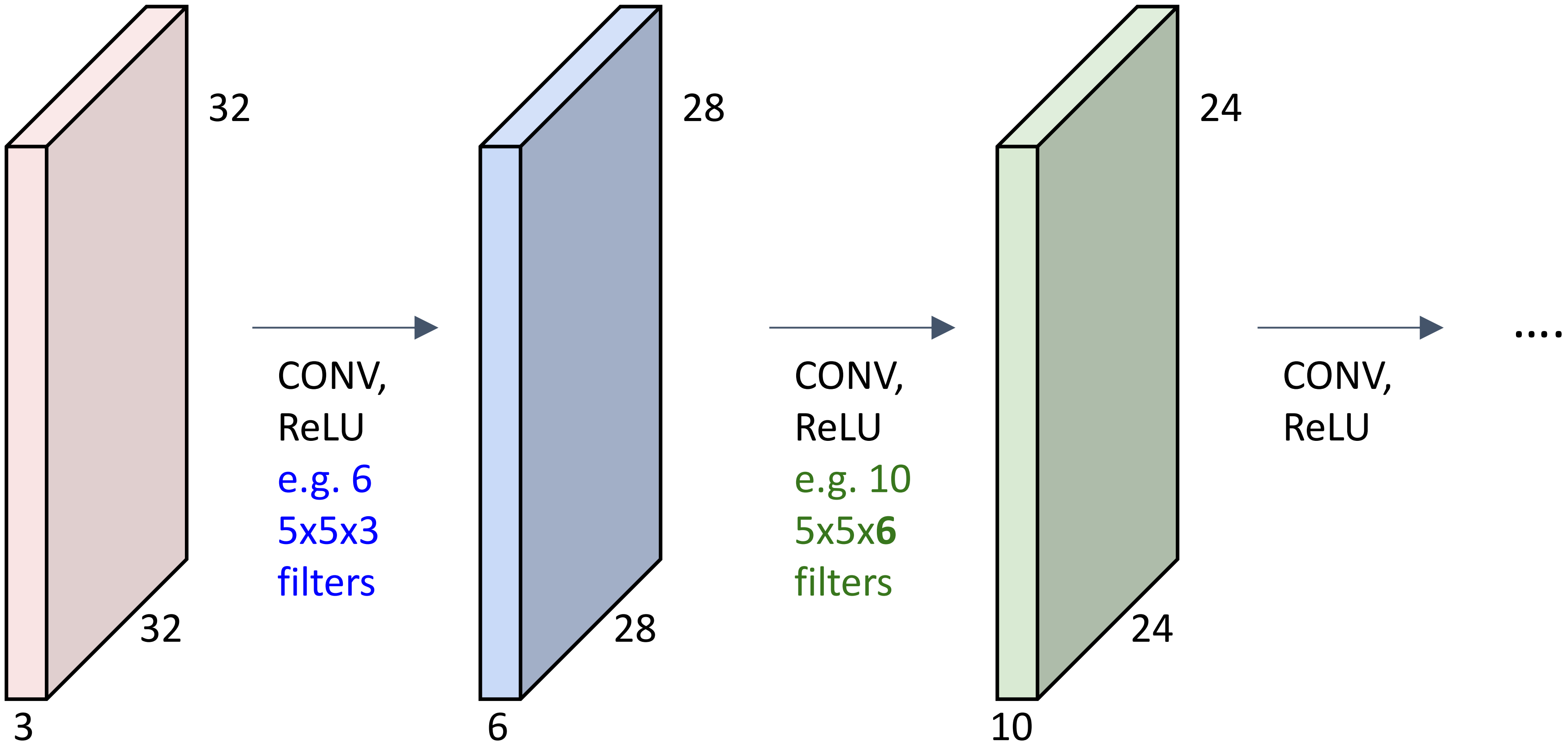
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps

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

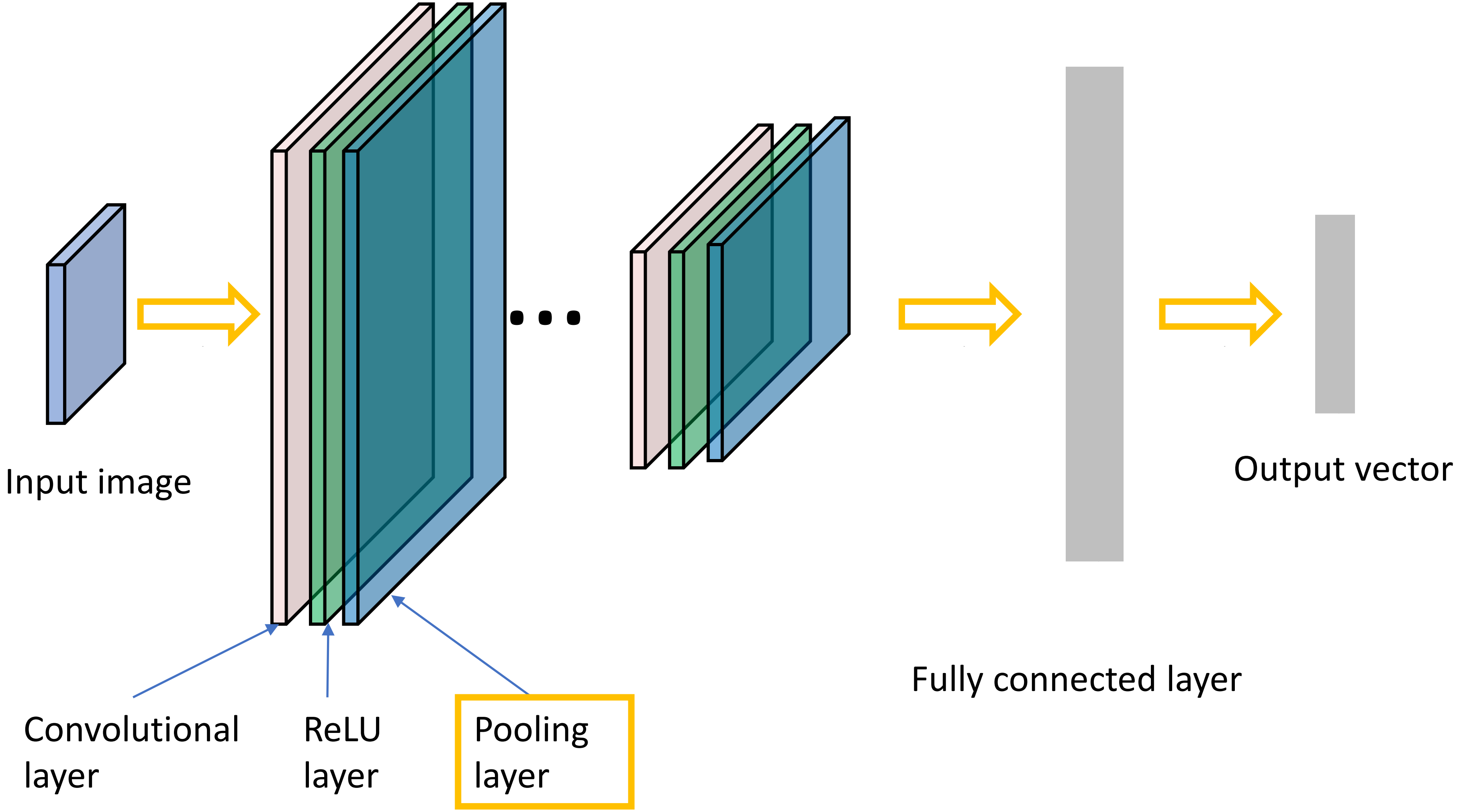
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



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

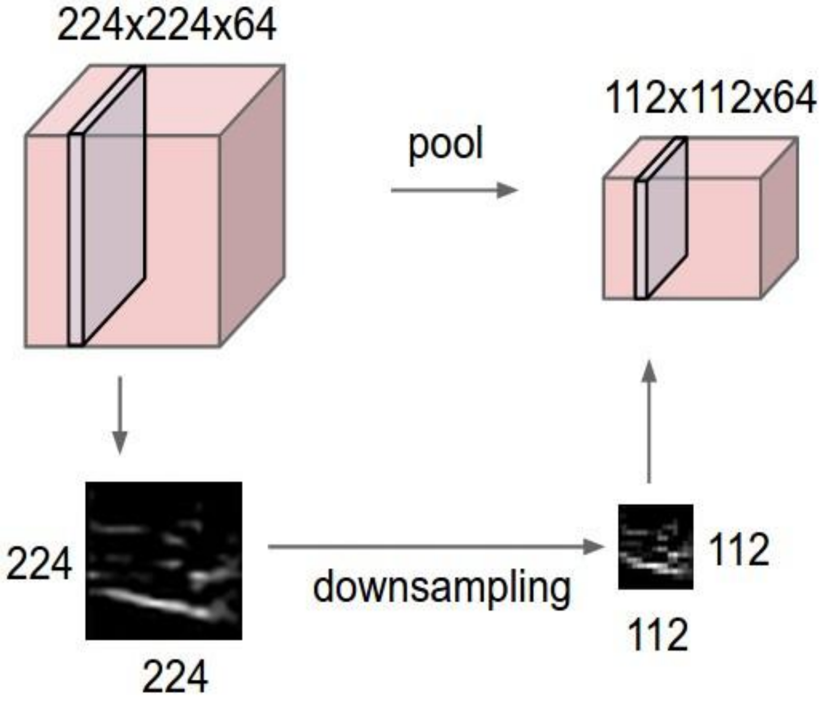


Convolutional Neural Networks



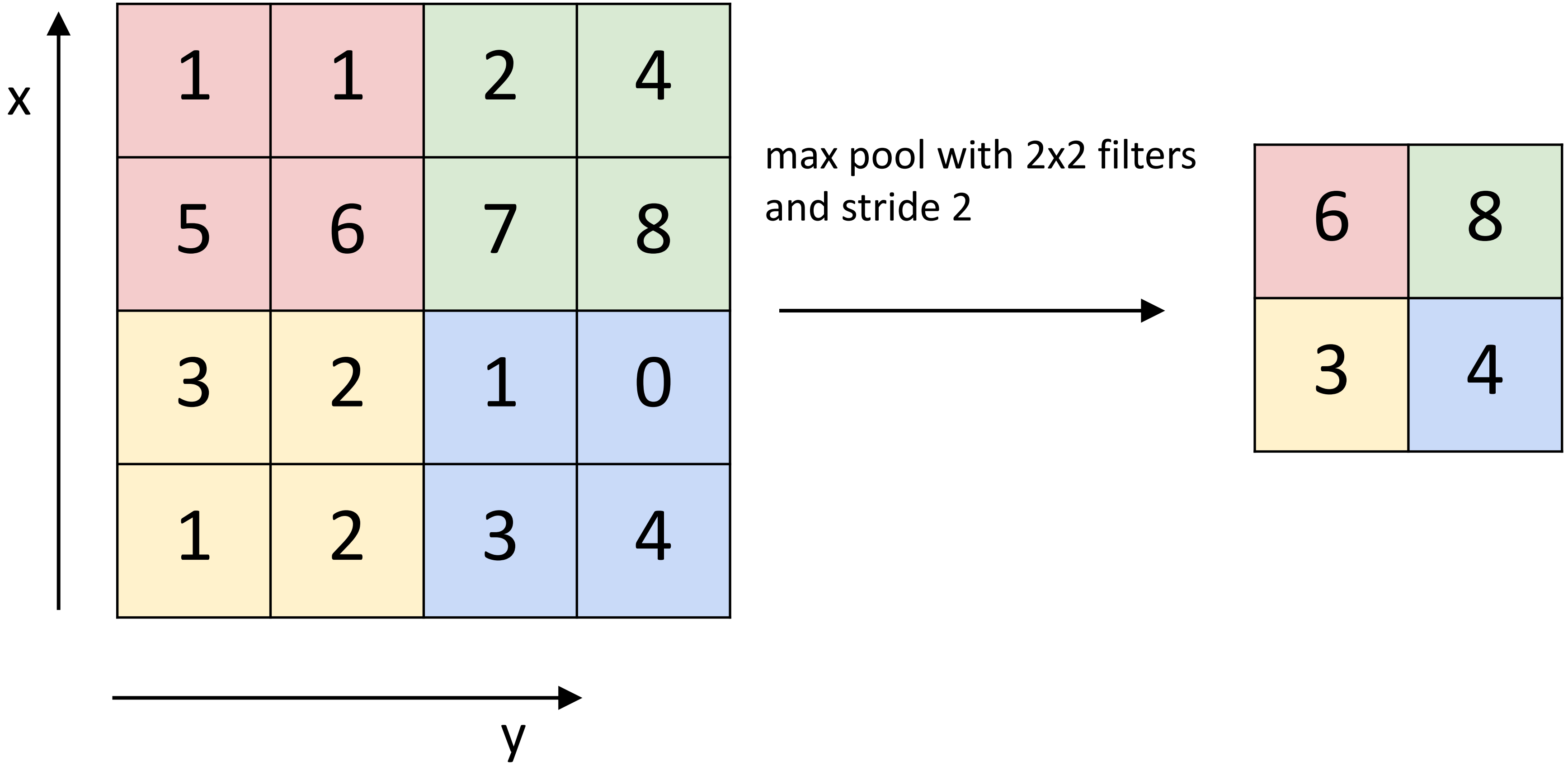
Pooling Layer

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



Max Pooling Layer

Single depth slice



My Python Setup

Further Reading

- Stanford CS231n, lecture 5, Convolutional Neural Networks
  http://cs231n.stanford.edu/schedule.html
- Deep learning with PyTorch
  https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
- AlexNet (2012):
  https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html
- Vgg16 (2014): https://arxiv.org/abs/1409.1556
- GoogleNet (2014): https://arxiv.org/abs/1409.4842
- ResNet (2015): https://arxiv.org/abs/1512.03385