CSC 578
Neural Networks and Deep Learning

6-2. Convolutional Neural Networks (2)

What is an Image?

Image

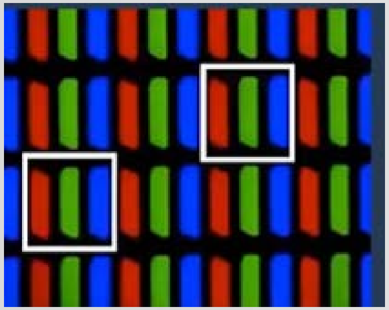
An image is matrix that specifies the color of various pixels in terms of the amount of red, green and blue components.

Every pixel is assigned an RGB value, each component being a value between 0 to 255. These components are mixed together to form a unique color value.

E.g. Blue is rgb(0,0,255) and Black is rgb(0,0,0)

Displaying an Image

Any display of an image consists of an arrangement of red, green, and blue dots. A set of one dot of each color form a pixel



What is a Kernel Filter ?

Kernel

A kernel is a square matrix that specifies spatial weights of various pixels in an image.

Different image processing techniques use different kernels.

1	1	1
1	1	1
1	1	1

3*3 Mean Kernel

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

5*5 Gaussian Kernel

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

3*3 Sobel Kernel

What is Convolution?

BIG
CONCEPT

Convolution

Convolution of a matrix involves laying a matrix over another and then calculating the weighted sum of all pixel values.

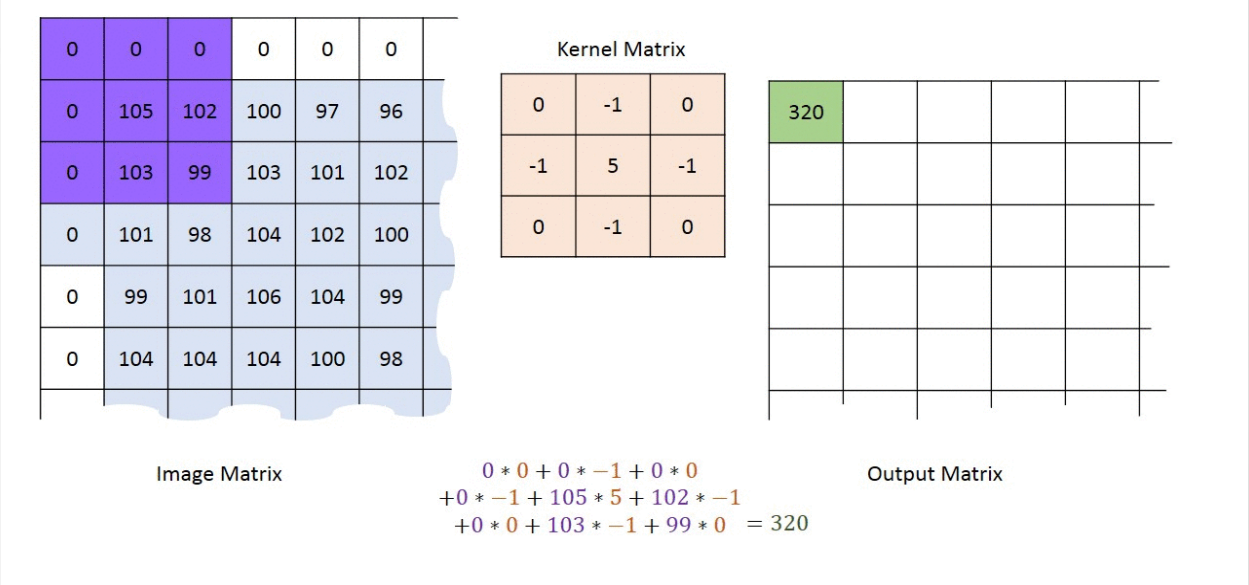


Image Matrix

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

Output Matrix

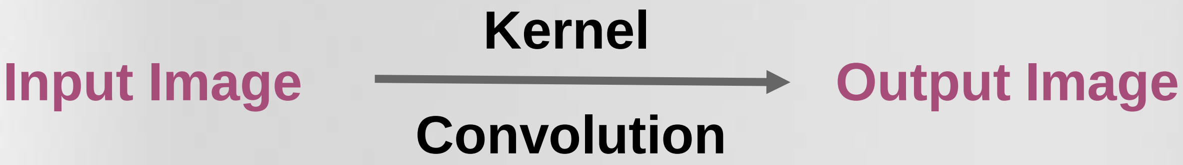
320				

$$\begin{aligned}&0*0+0*-1+0*0\\&+0*-1+105*5+102*-1\\&+0*0+103*-1+99*0 \quad = 320\end{aligned}$$

Convolution with horizontal and vertical strides = 1

Image Processing

How to process an image?



Input Image

Kernel

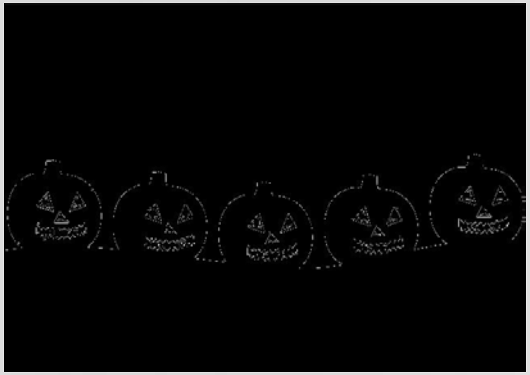
Convolution

Output Image

RGB to Grayscale

Edge detection

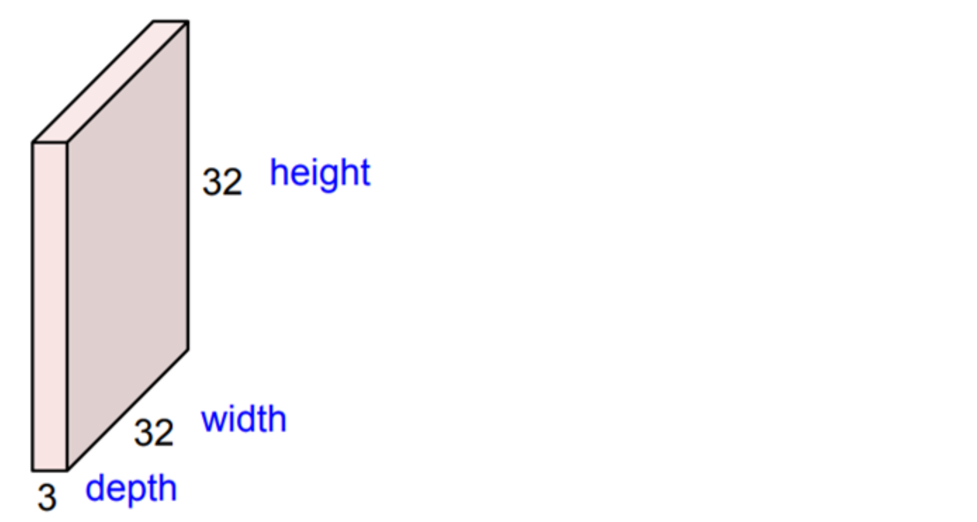
-1	0	1
-2	0	2
-1	0	1



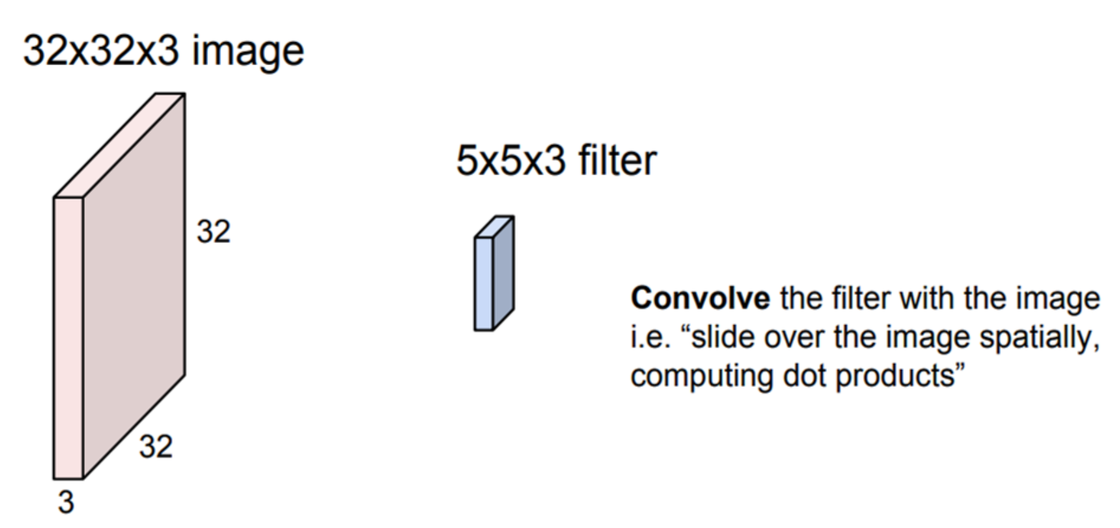
Convolution Neural Network

Convolution Layer

32*32*3 image -> Preserve spatial structure

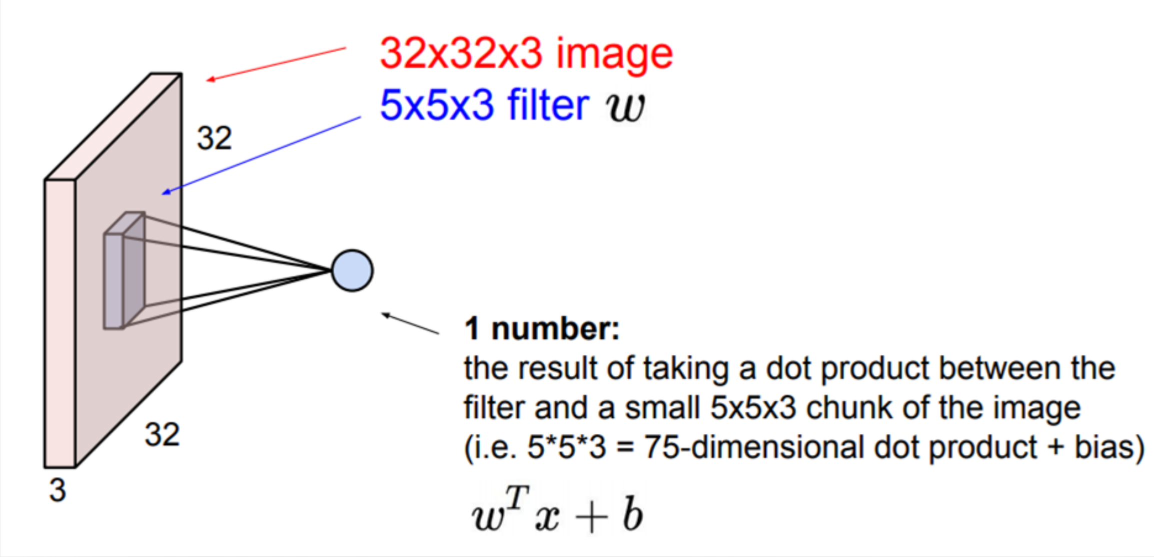


Convolution Layer



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

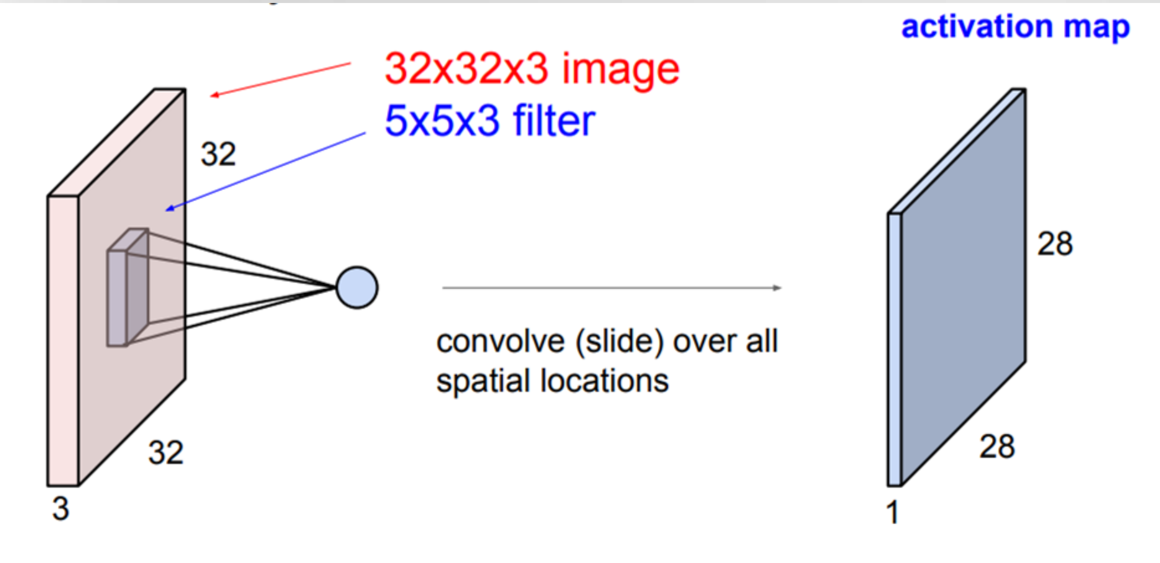
Convolution Layer



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)

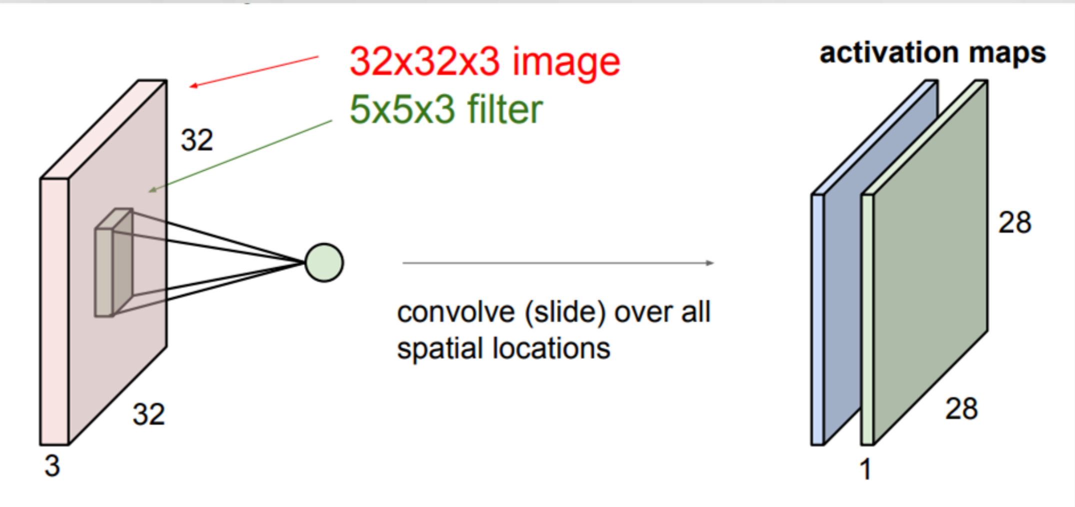
$$w^T x + b$$

Convolution Layer



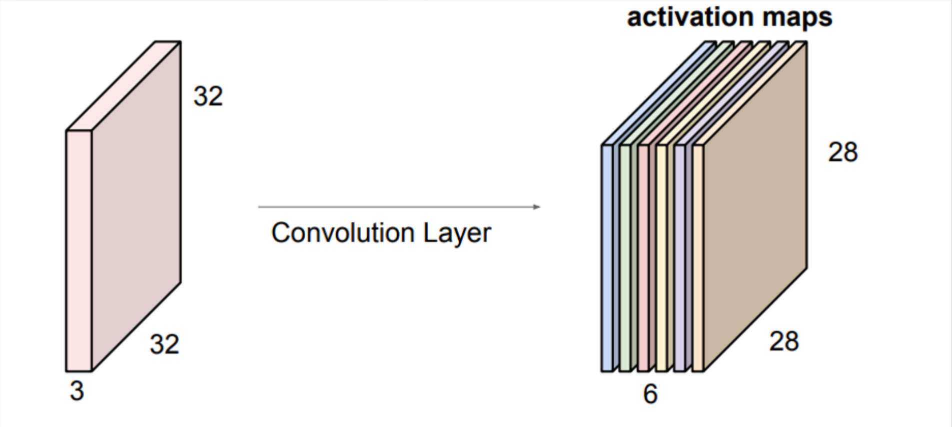
Convolution Layer

Consider a second, green filter



Convolution Layer

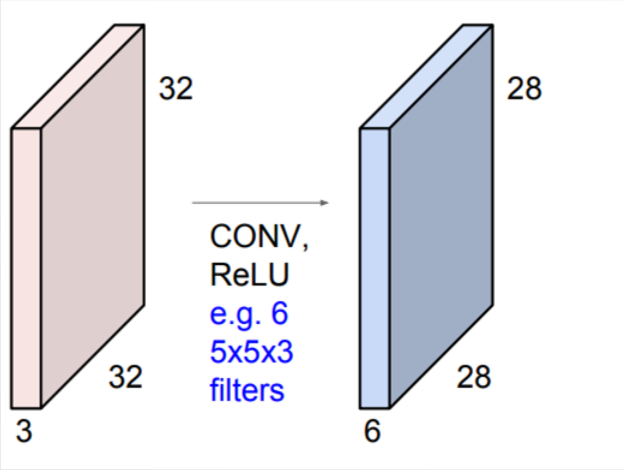
E.g. if we had 6 5*5 filters, we'll get 6 separate activation maps



We stack themup to get a "new image" of size 28*28*6

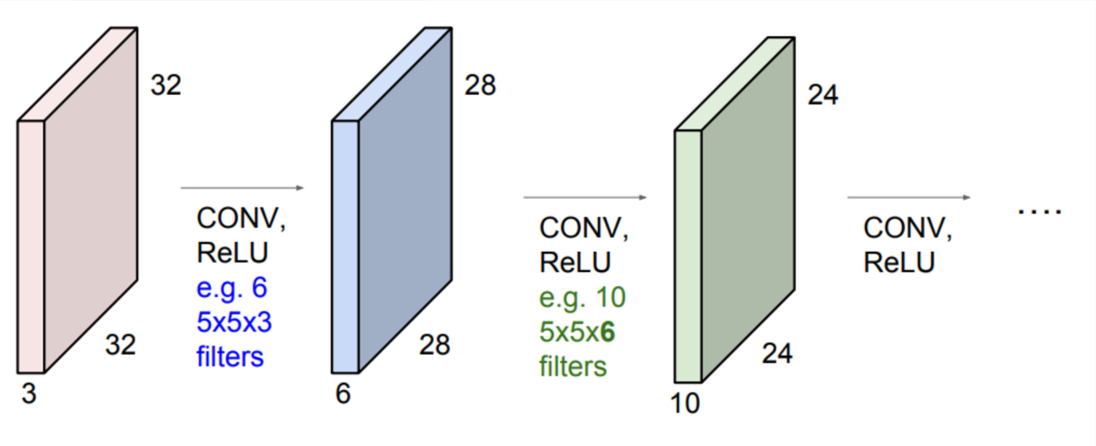
Convolution Layer

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



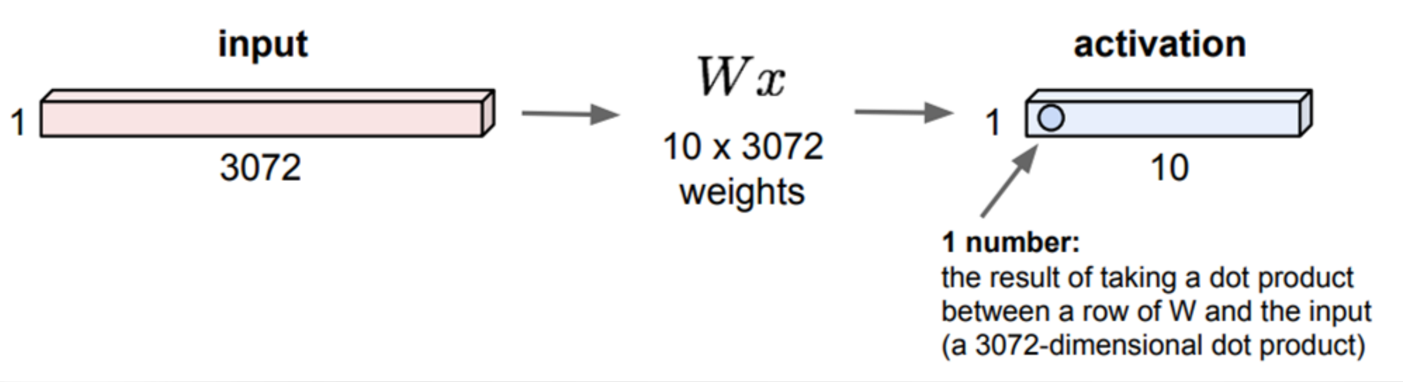
Convolution Layer

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



Fully Connected Layer

32*32*3 image -> stretch to 3072*1


input
1
3072
Wx
10 x 3072
weights
activation
1
10
1 number:
the result of taking a dot product
between a row of W and the input
(a 3072-dimensional dot product)

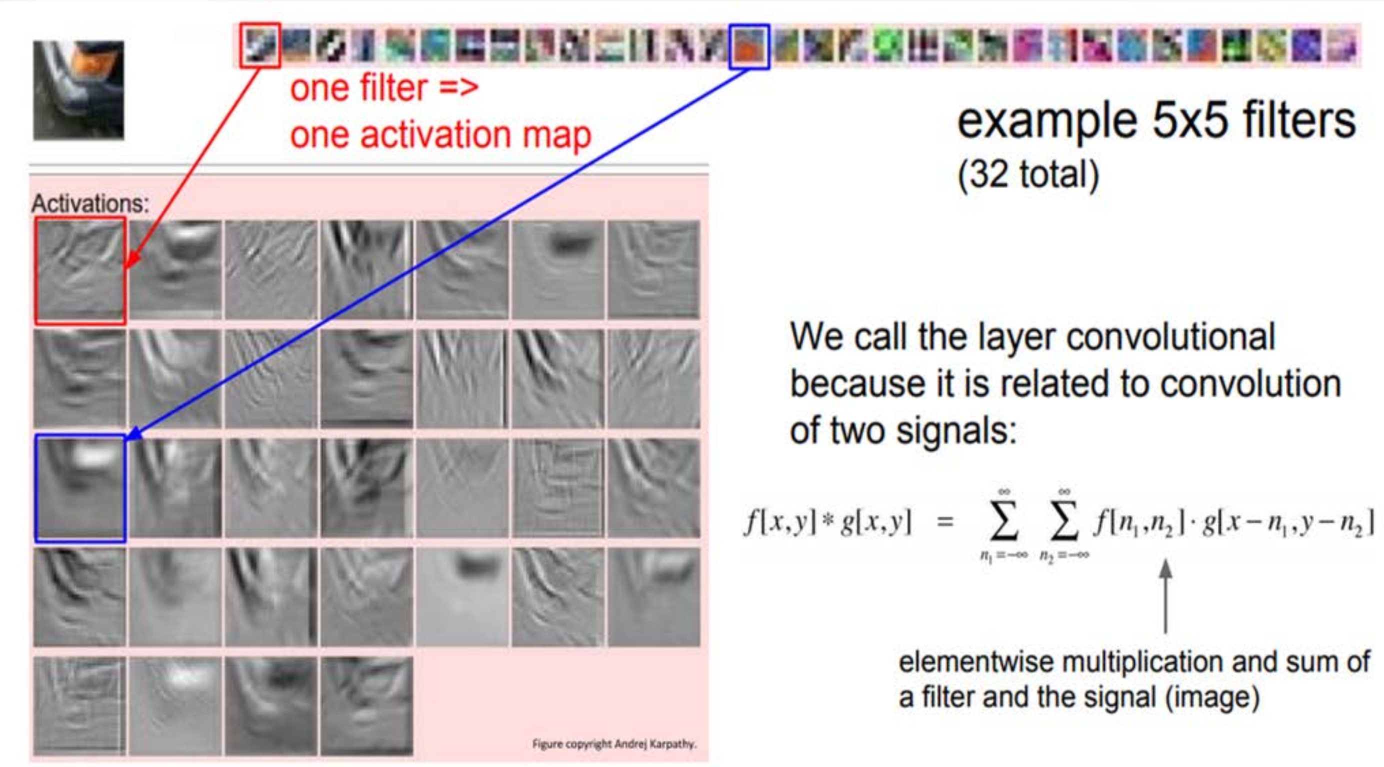


Figure copyright Andrej Karpathy.

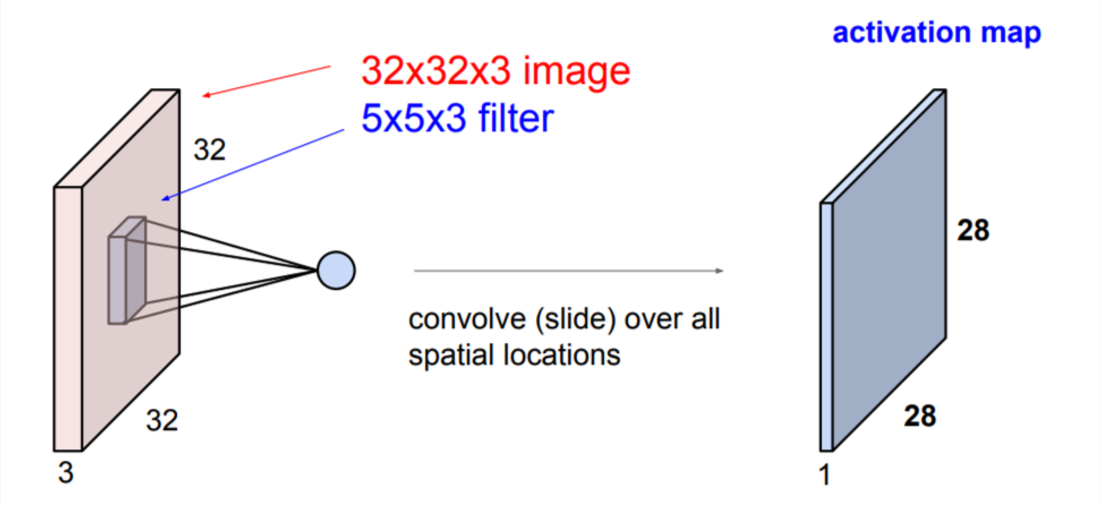
example 5x5 filters
(32 total)

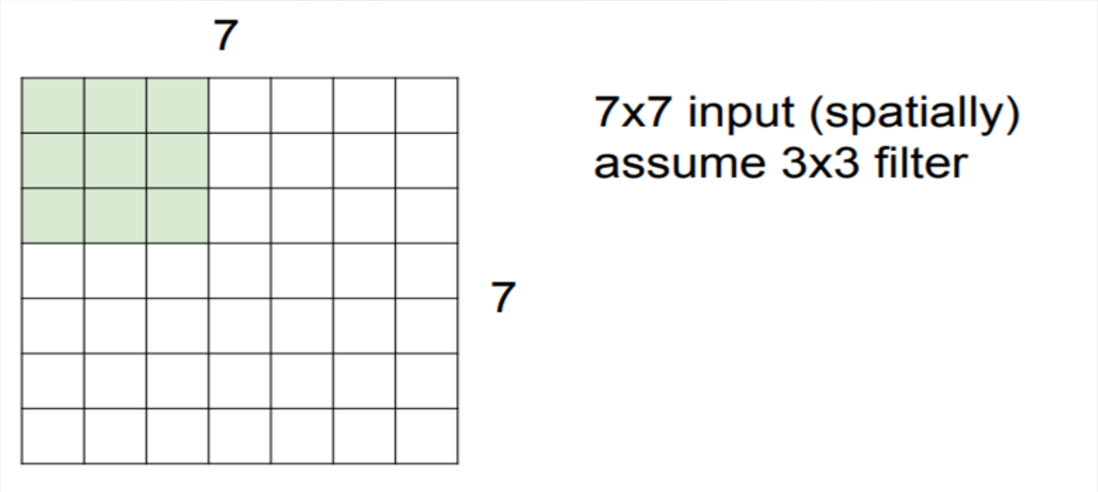
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

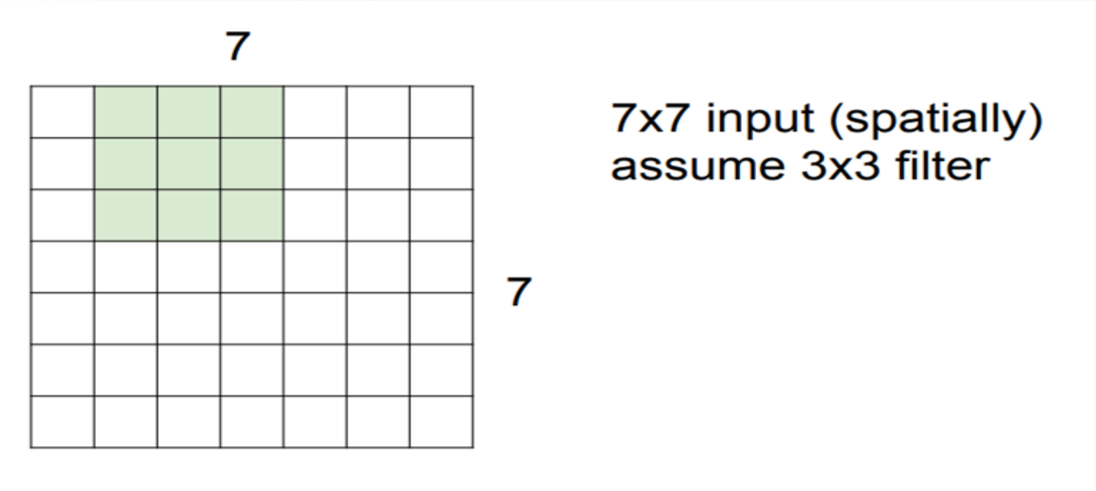
elementwise multiplication and sum of a filter and the signal (image)

A closer look at spatial dimension (1)

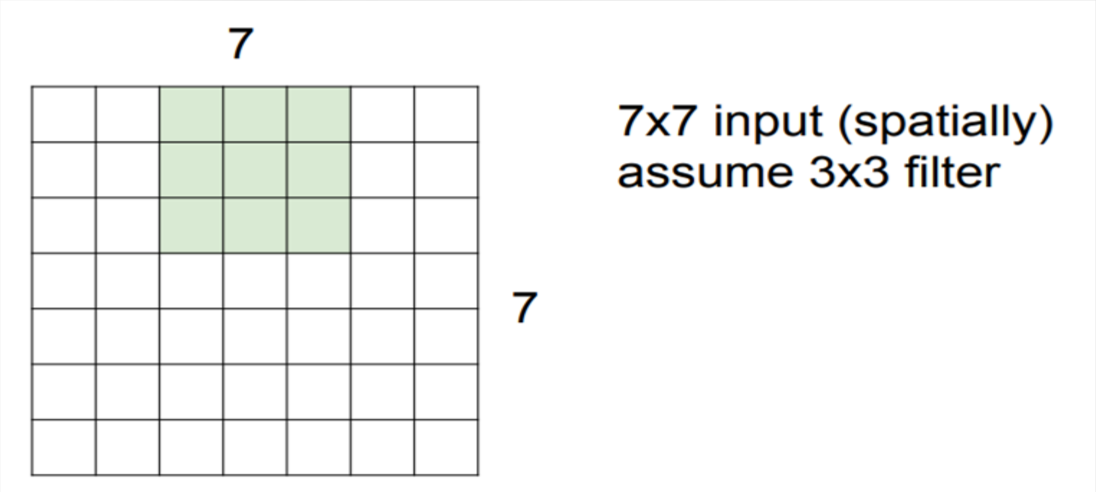




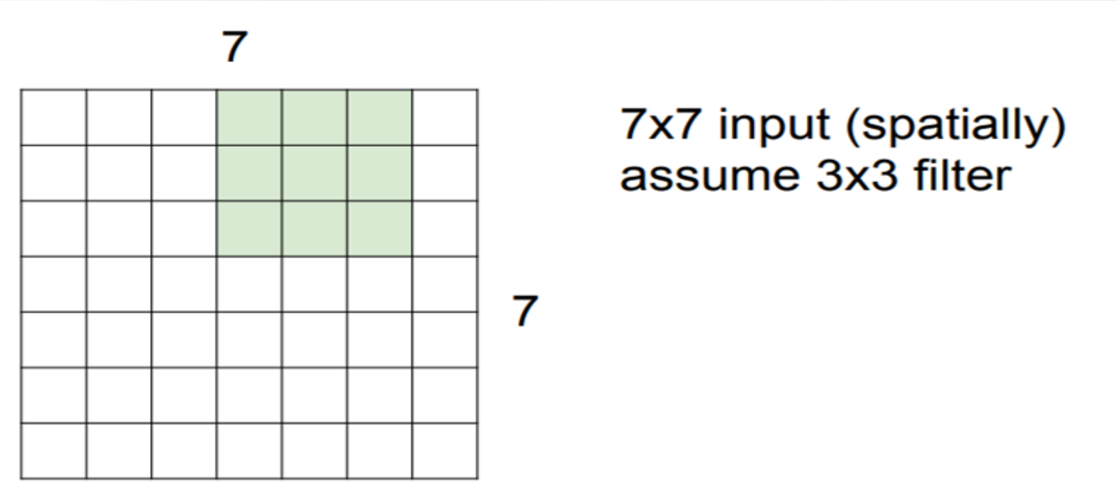
7x7 input (spatially)
assume 3x3 filter



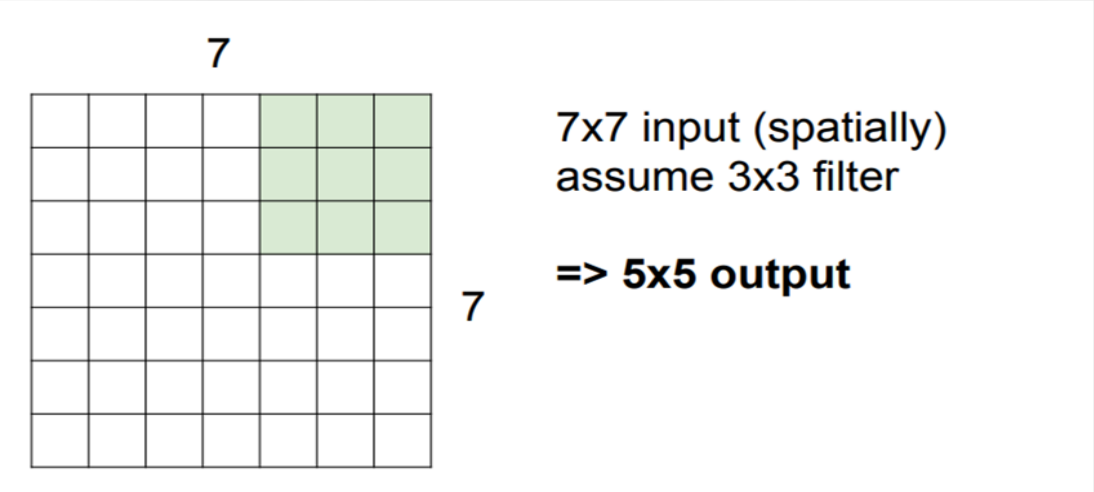
7x7 input (spatially)
assume 3x3 filter



7x7 input (spatially)
assume 3x3 filter



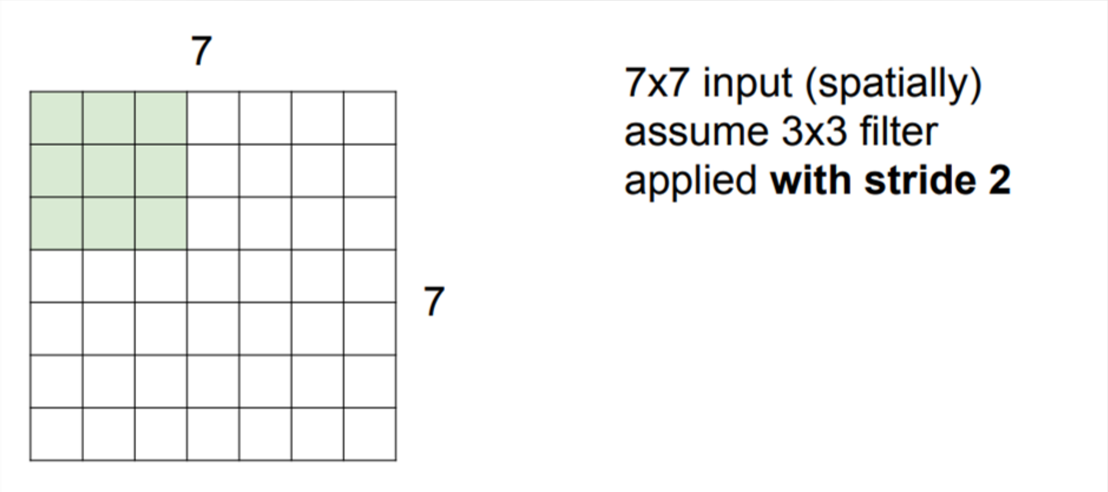
7x7 input (spatially)
assume 3x3 filter



7x7 input (spatially)
assume 3x3 filter

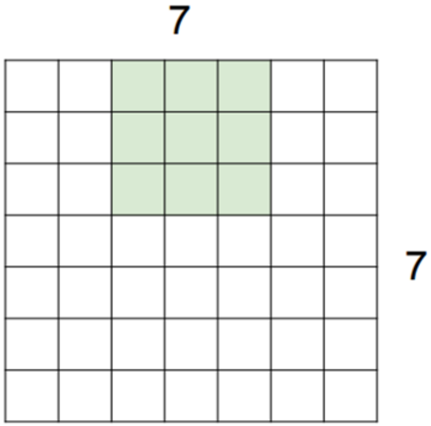
=> 5x5 output

A closer look at spatial dimension

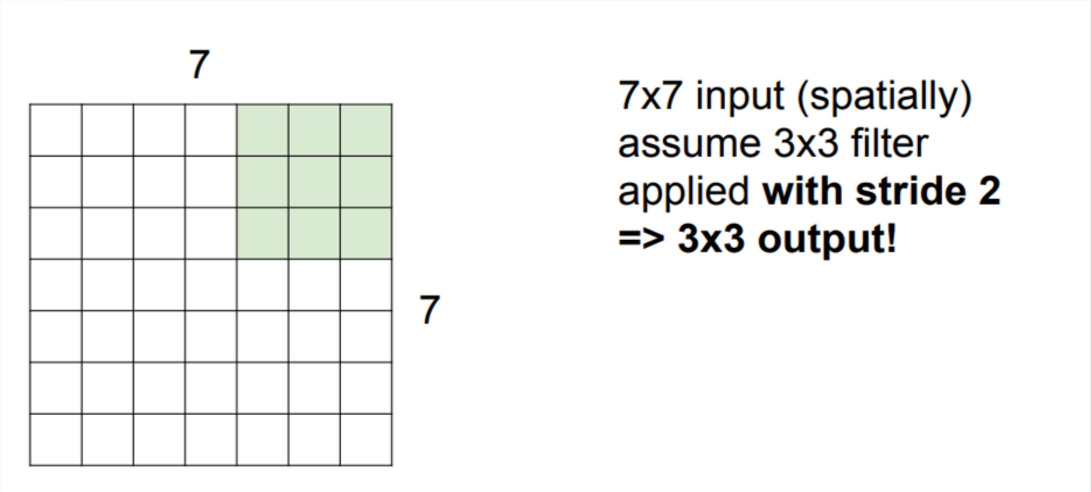


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimension (2)

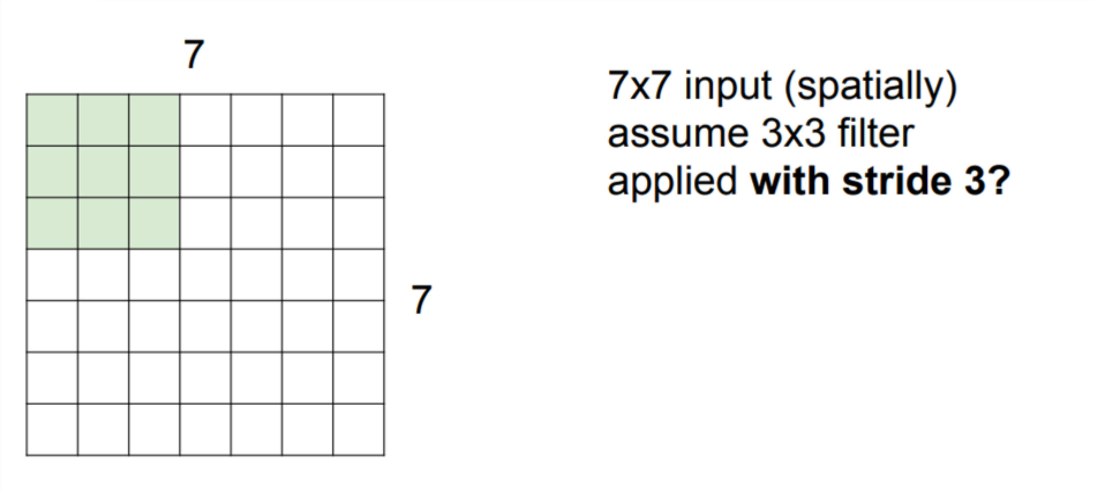


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

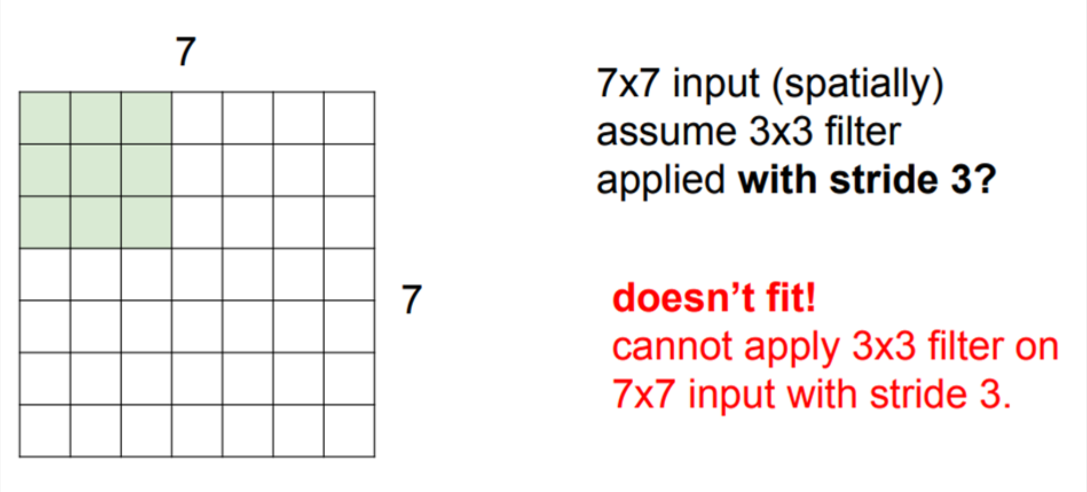


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimension (3)



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



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

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

Constraints on Output Size

For the input volume size (W), the receptive field size of the Conv Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border, the size of the output is

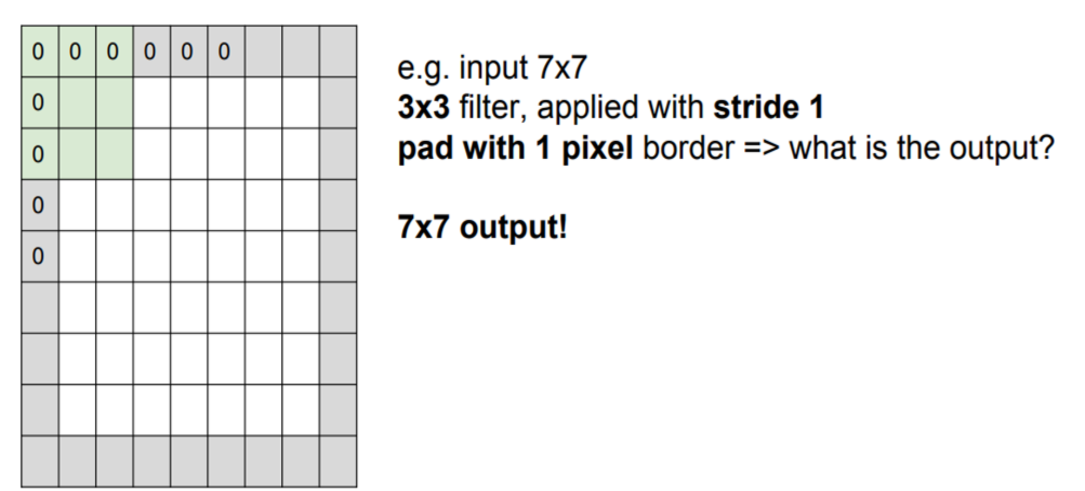
$$(W-F+2P)/S+1$$

Examples:

- **For a 7x7 input and a 3x3 filter with stride 1 and pad 0, we would get a 5x5 output.**
- **With the same input and filter but with stride 2, we would get a 3x3 output.**

Use of zero-padding

- To avoid shrinking output
- To throw away information from edges

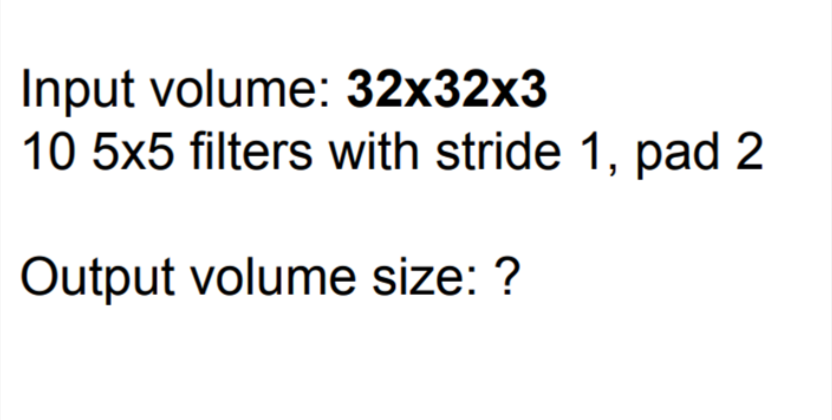


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

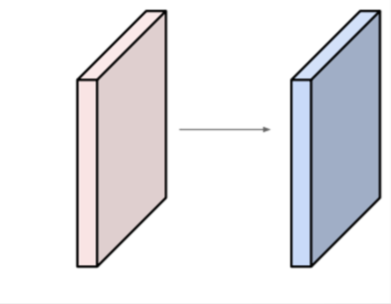
Setting zero padding to be P=(F−1)/2 when the stride S=1 ensures that the input volume and output volume will have the same size spatially.

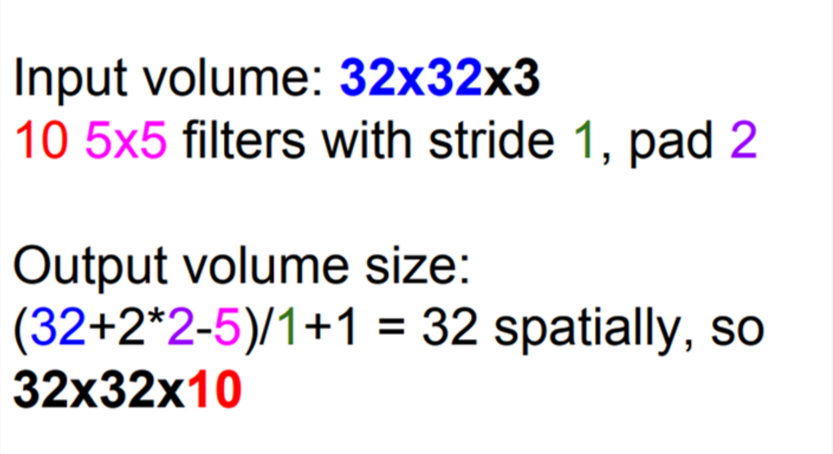
Exercise (1)



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

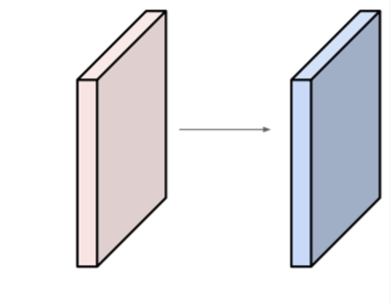
Output volume size: ?



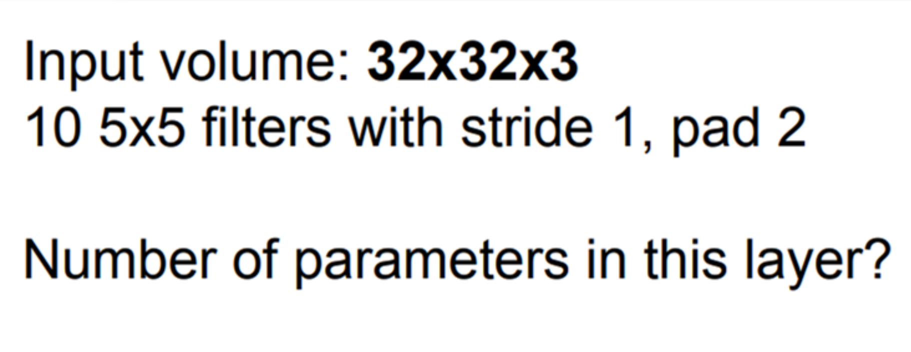


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

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

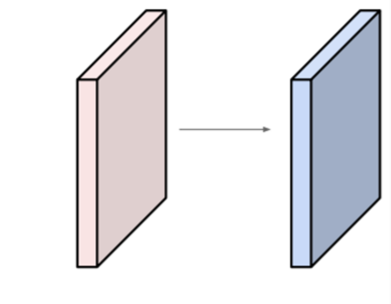


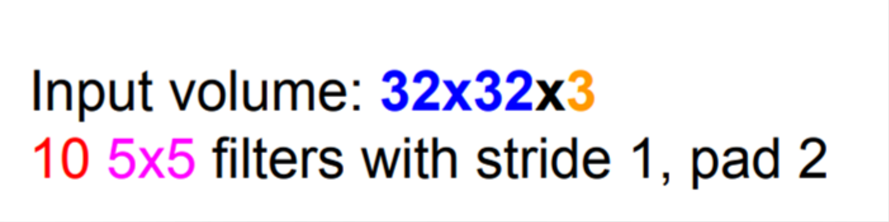
Exercise (2)



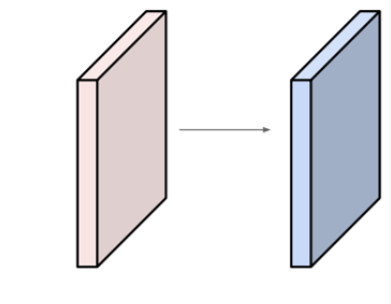
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

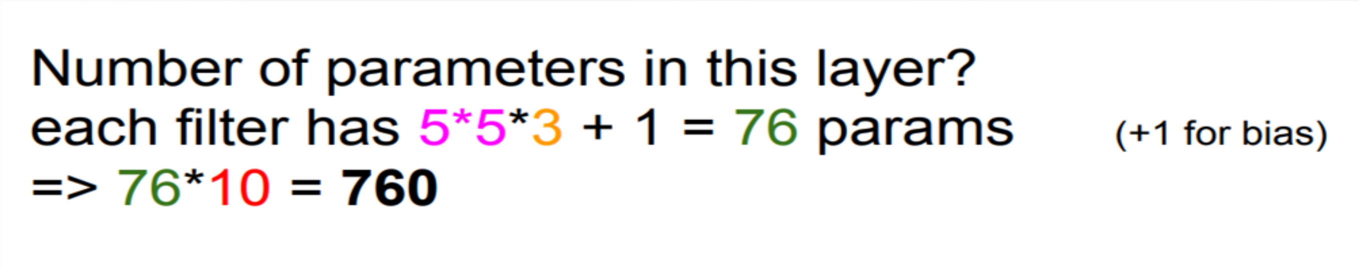
Number of parameters in this layer?





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



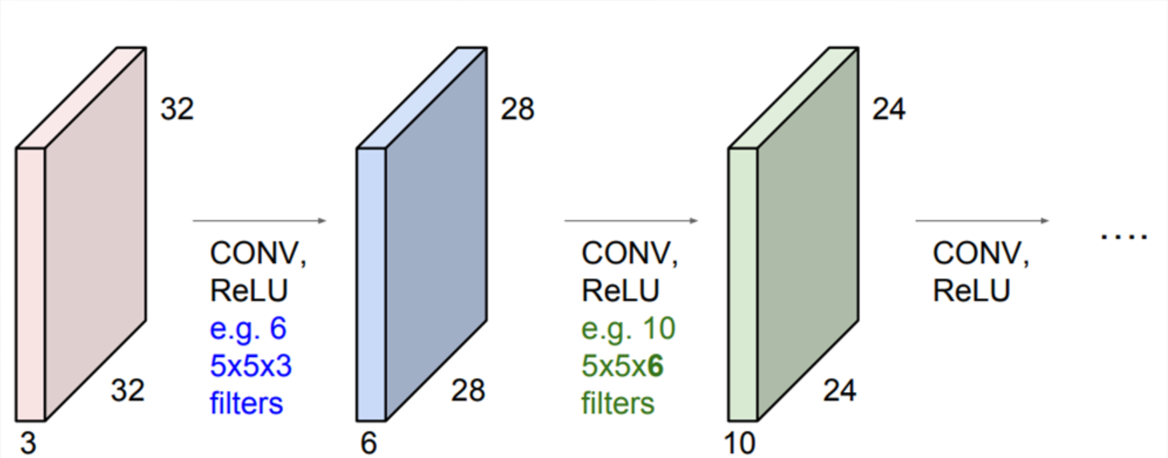


Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Reference: http://cs231n.github.io/convolutional-networks/ for explanation and animation on the DEPTH of input volume.

CAUTION:

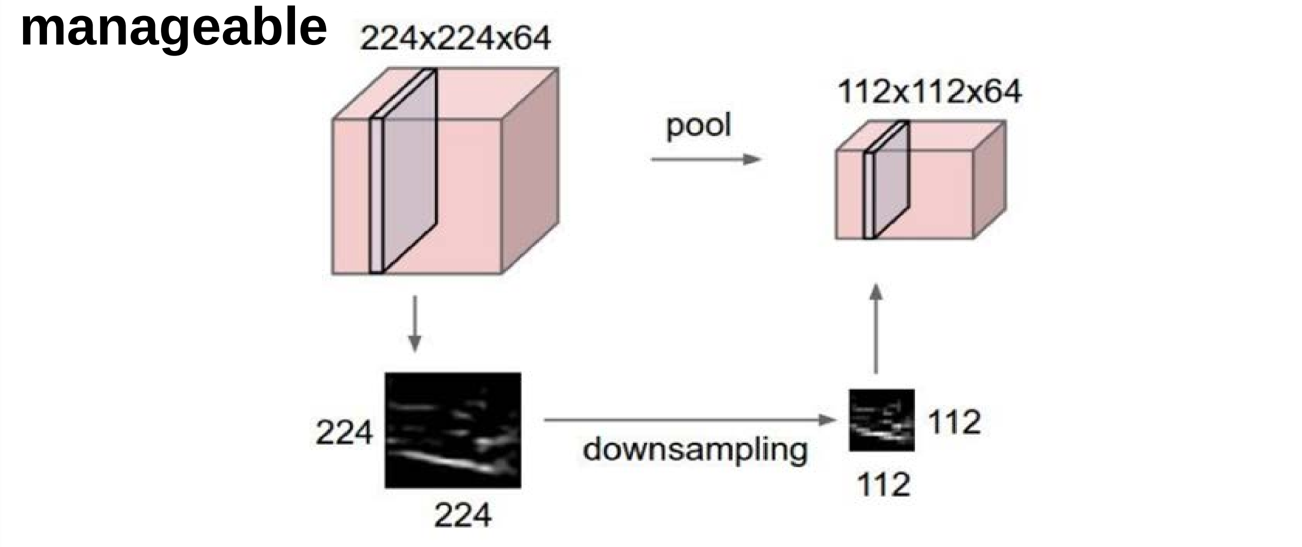
E.g. 32*32 input convolved repeatedly with 5*5 filters shrinks volumes spatially.



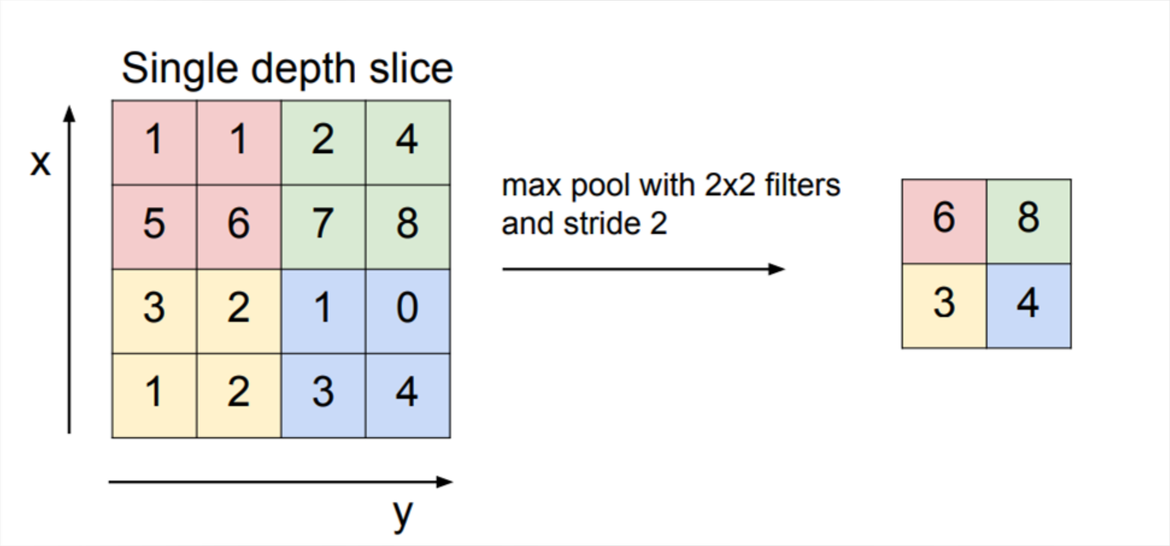
BUT shrinking too fast is not good, because feature maps lose information as they get deeper.

Pooling Layer

Makes the representations smaller and more manageable



Max Pooling



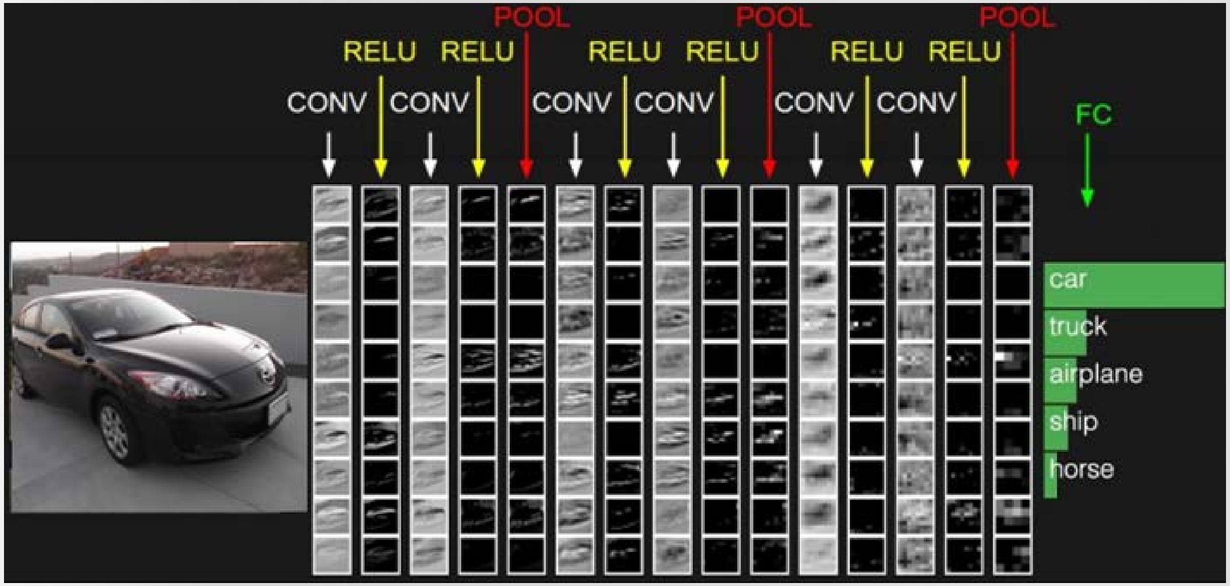
Single depth slice

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

max pool with 2x2 filters and stride 2

6	8
3	4

Intervening Activation Layer



CNN Architectures

AlexNet

[Krizhevsky et al. 2012]

Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

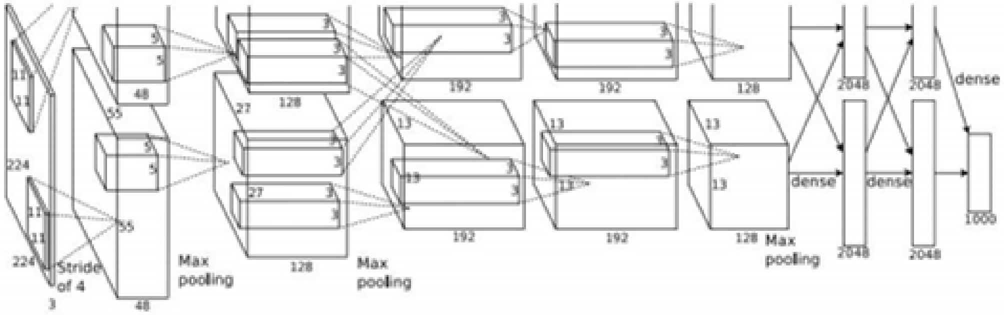
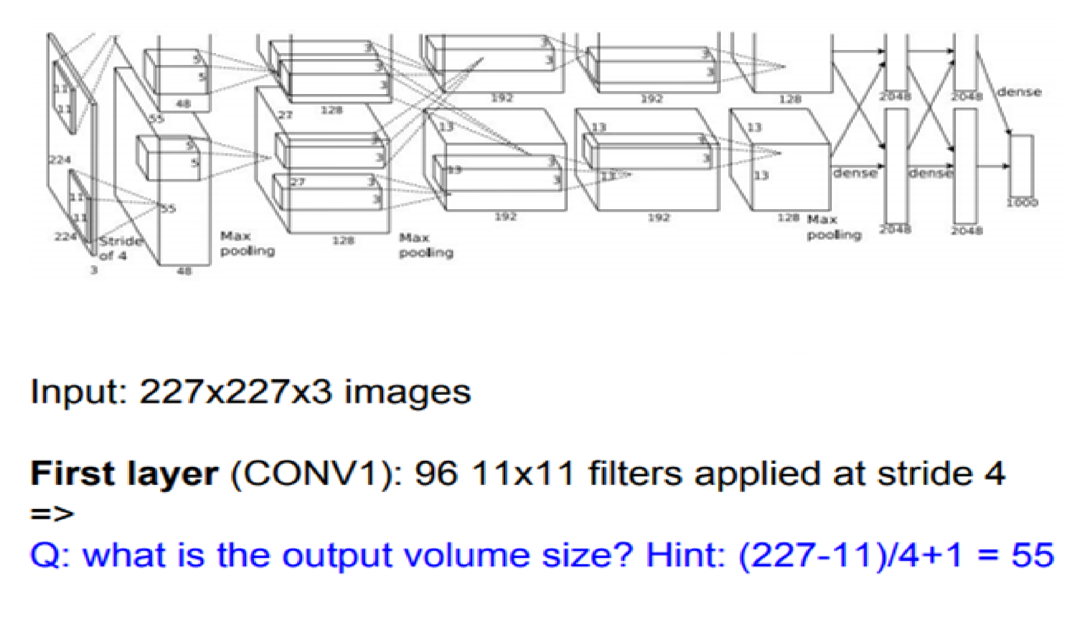


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

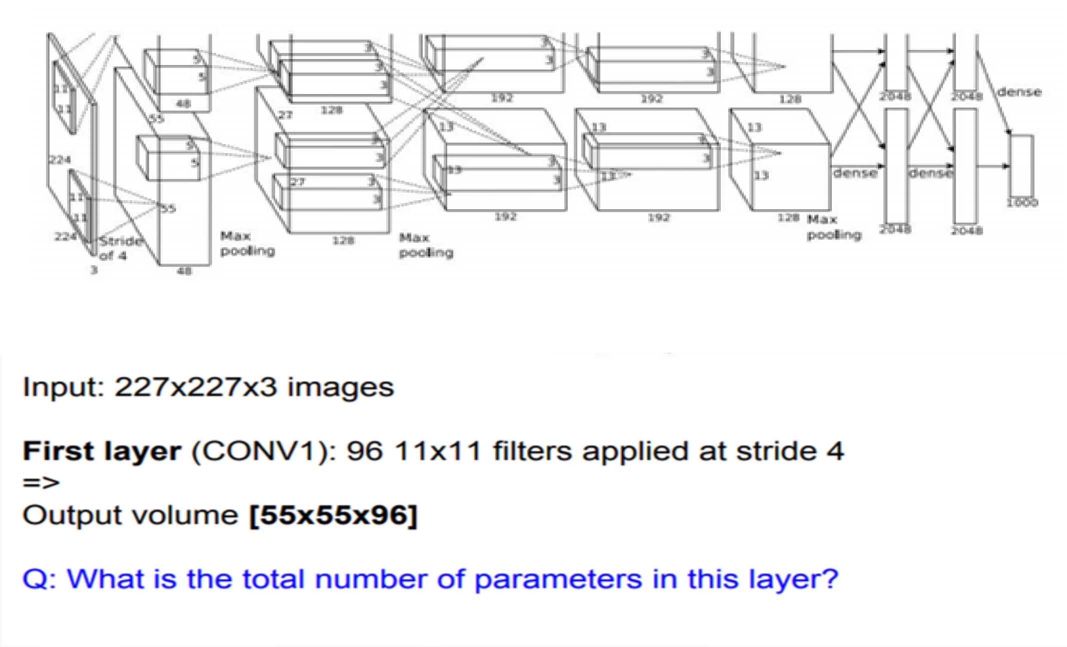
AlexNet



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Q: what is the output volume size? Hint: (227-11)/4+1 = 55

AlexNet

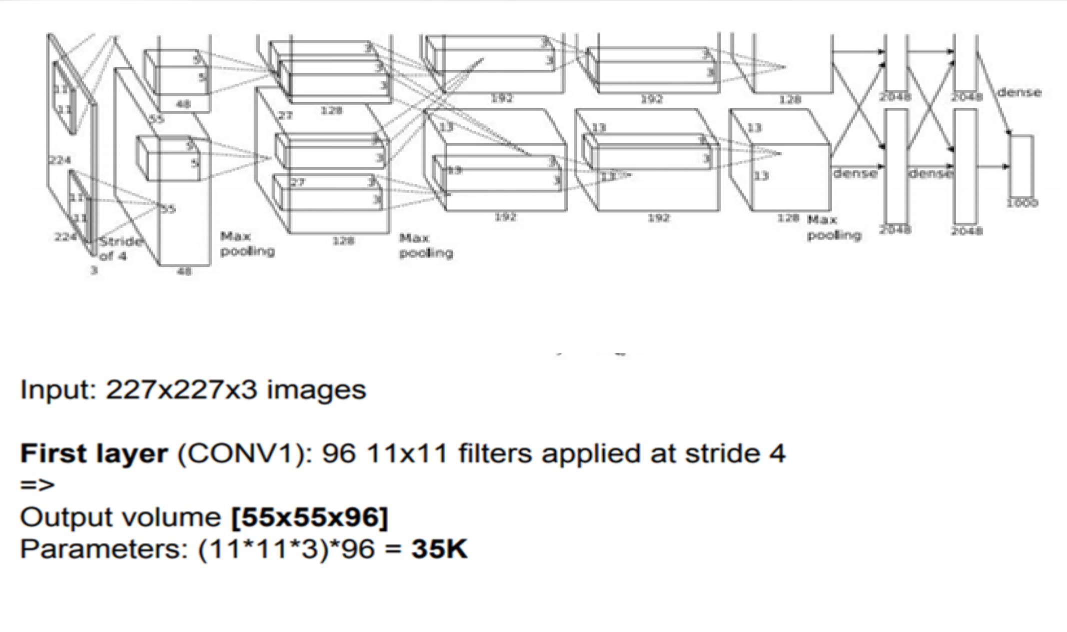


Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

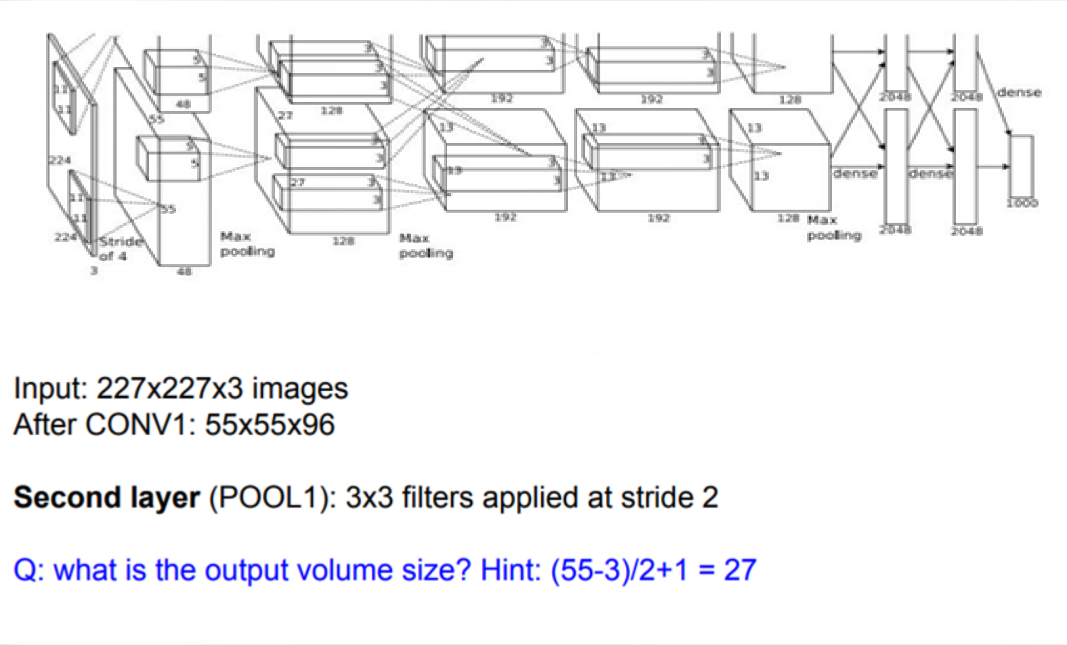
AlexNet



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

AlexNet

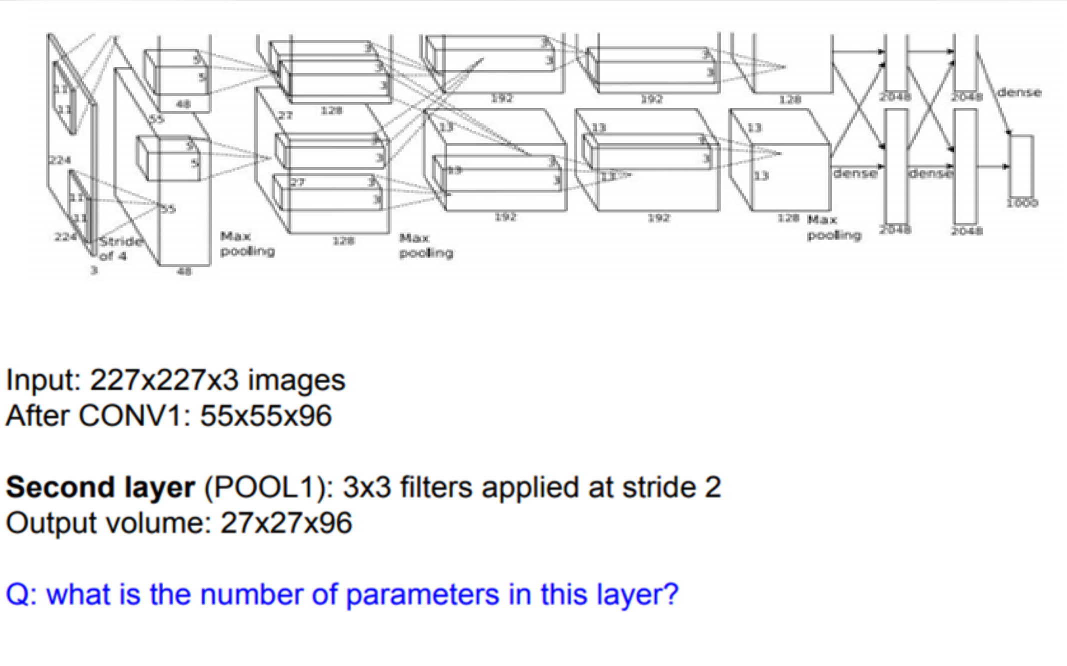


Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

AlexNet

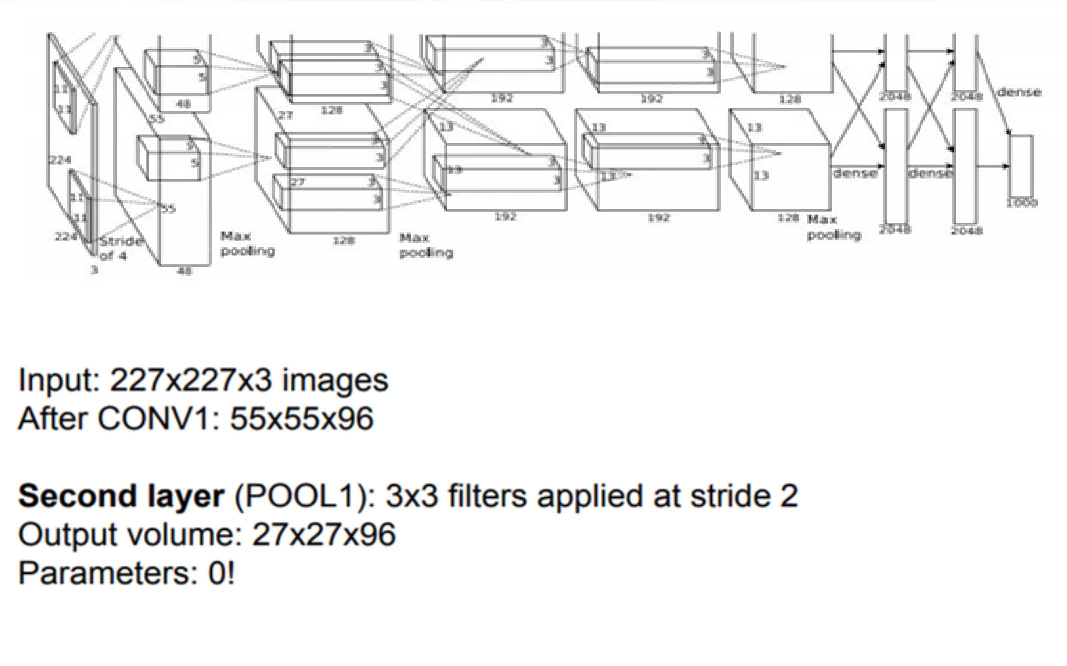


Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

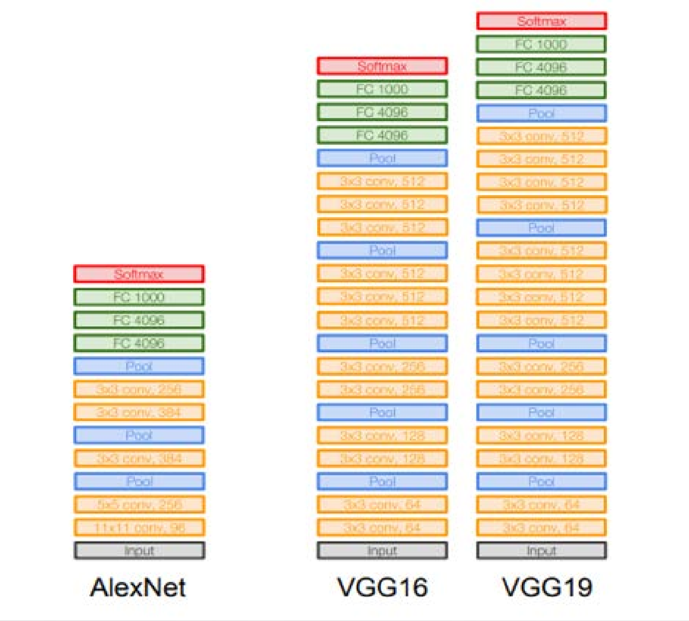
AlexNet



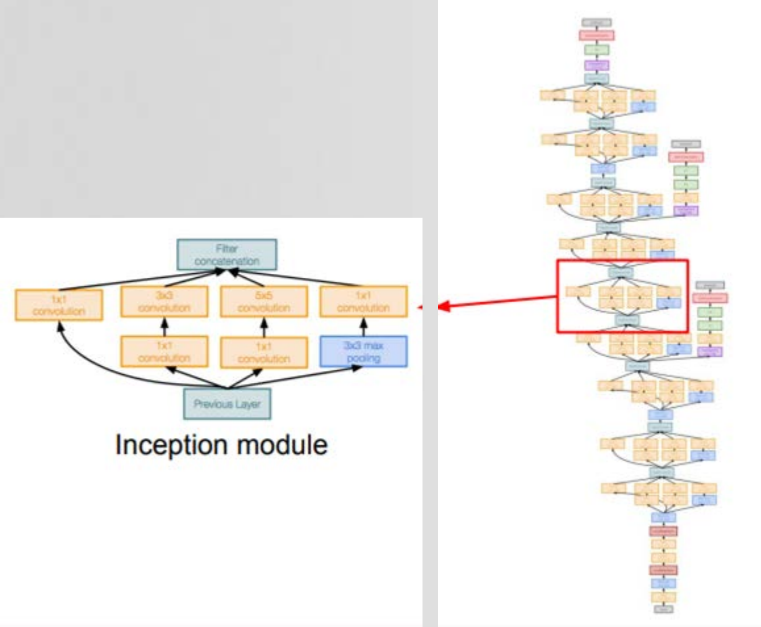
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

VGGNet


Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 256
3x3 conv, 384
Pool
3x3 conv, 384
Pool
5x5 conv, 256
11x11 conv, 96
Input
AlexNet
Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
VGG16
Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
VGG19

GoogleNet


Filter concatenation
1x1 convolution
3x3 convolution
5x5 convolution
1x1 convolution
1x1 convolution
1x1 convolution
3x3 max pooling
Previous Layer
Inception module