Reading Group on Deep Learning: Session 3

Introduction to Convolutional Neural Networks

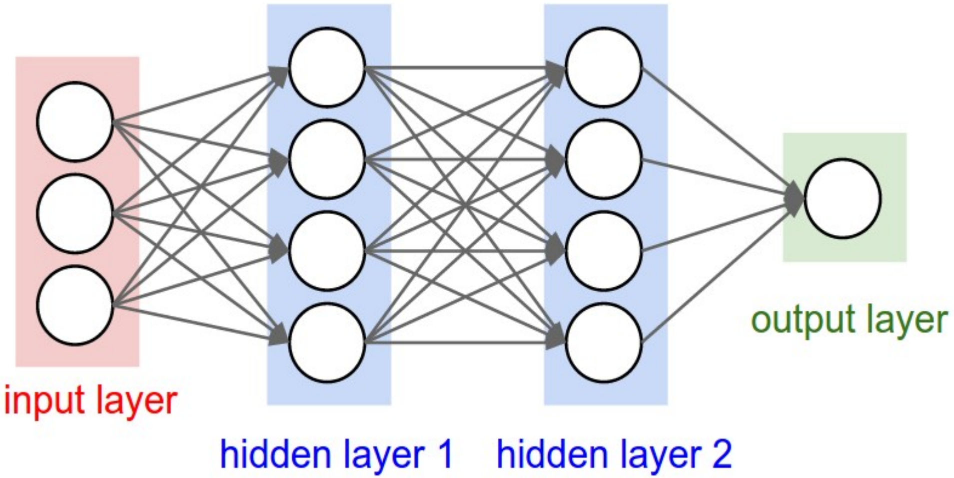
Vicky Kalogeiton

1 July 2016

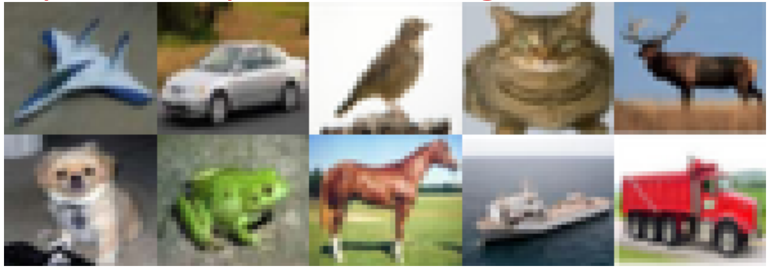
What are CNNs ?

CNN = Neural Network with a convolution operation
instead of matrix multiplication
in at least one of the layers

Neural Networks



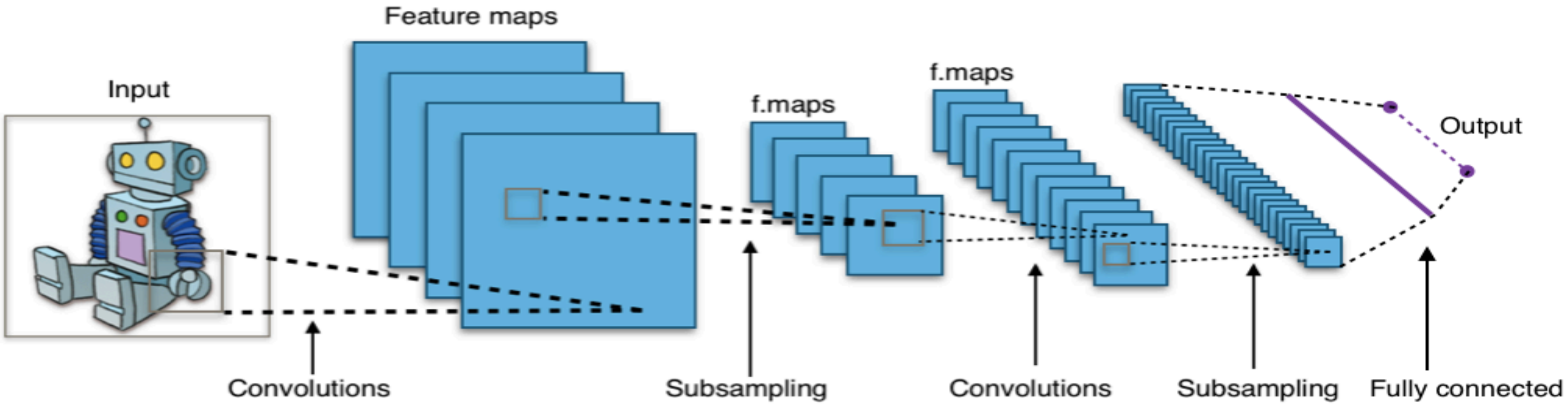
Input example : one image



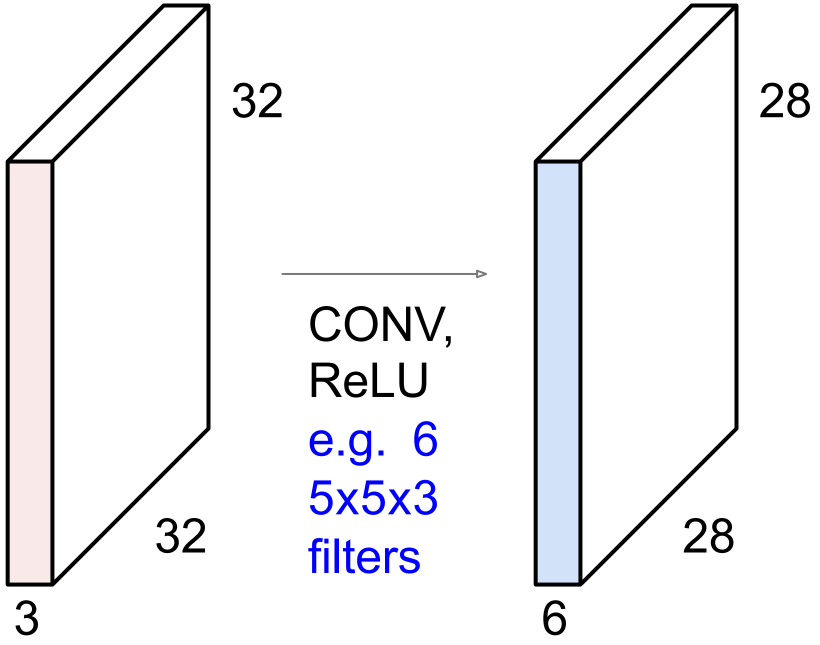
Output example : one class

airplane	dog
automobile	frog
bird	horse
cat	ship
deer	truck

A typical CNN architecture

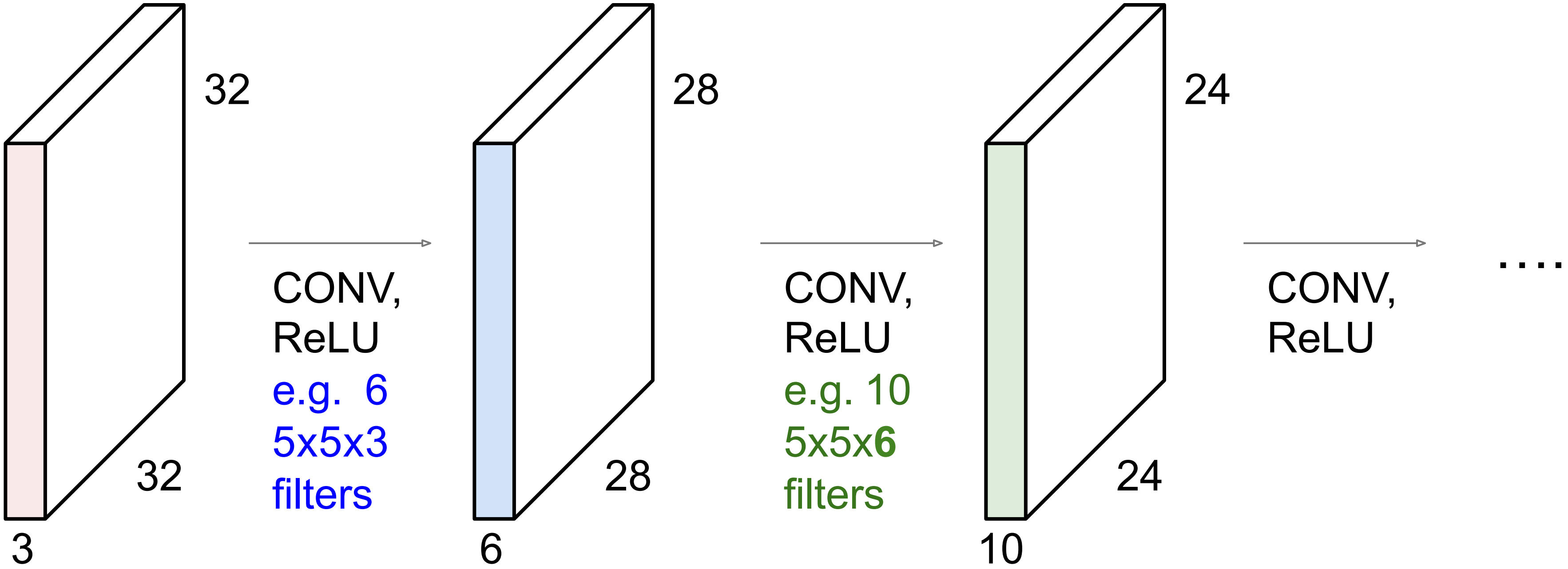


Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



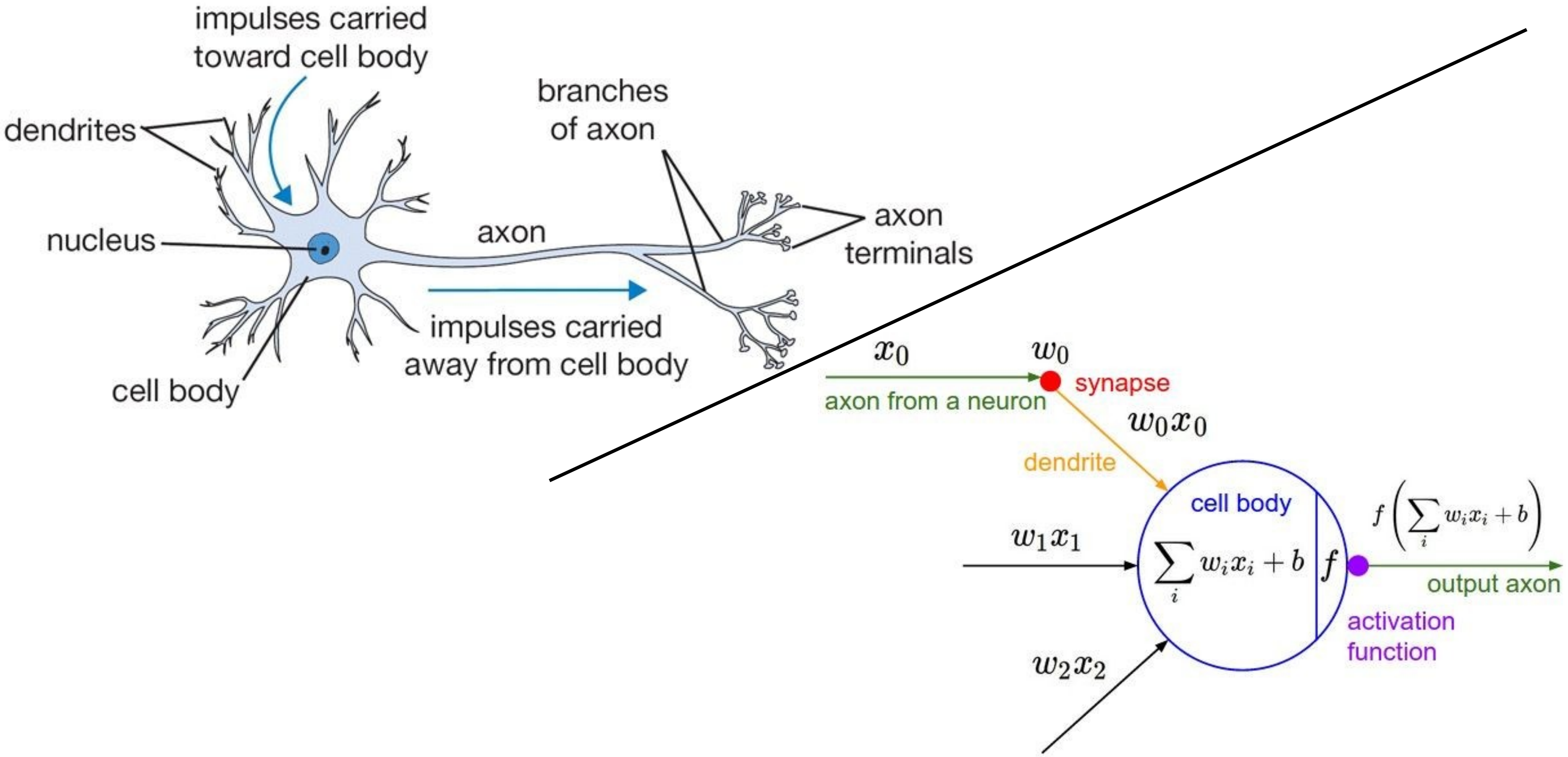
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

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



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

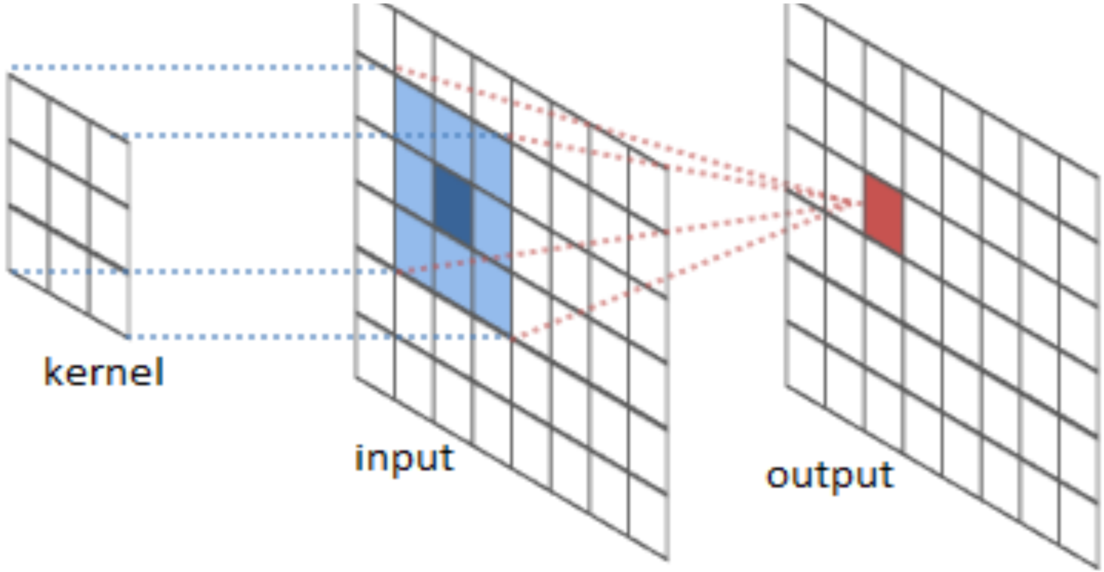
Biological neuron & mathematical model



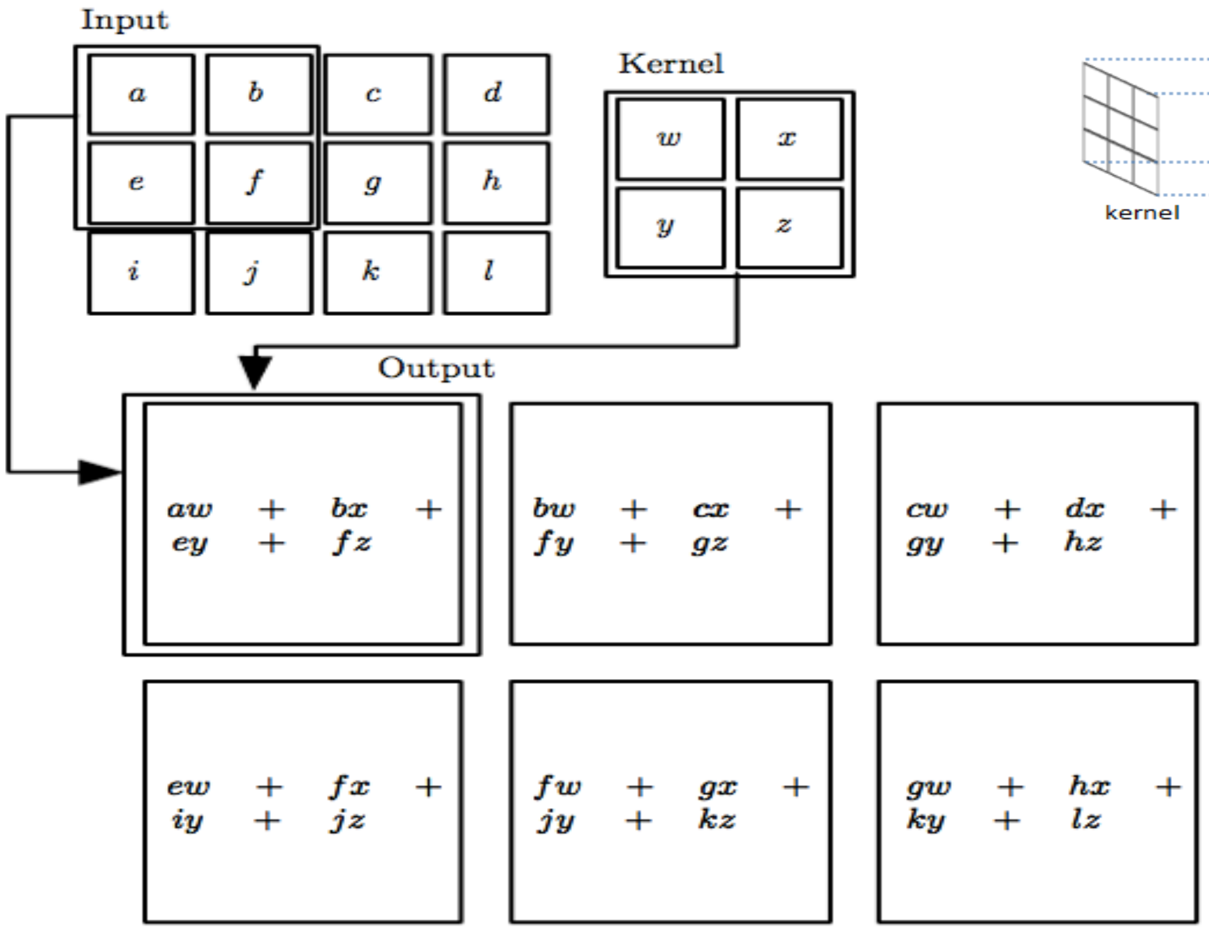
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

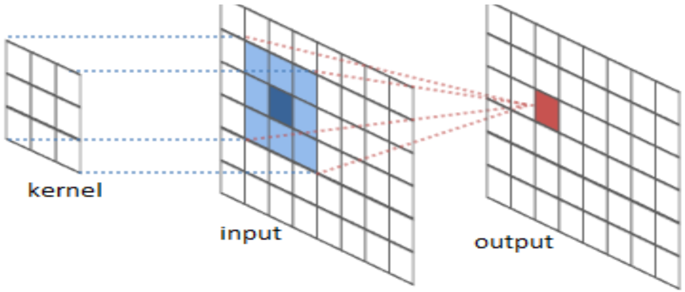
Convolution

The convolution operation



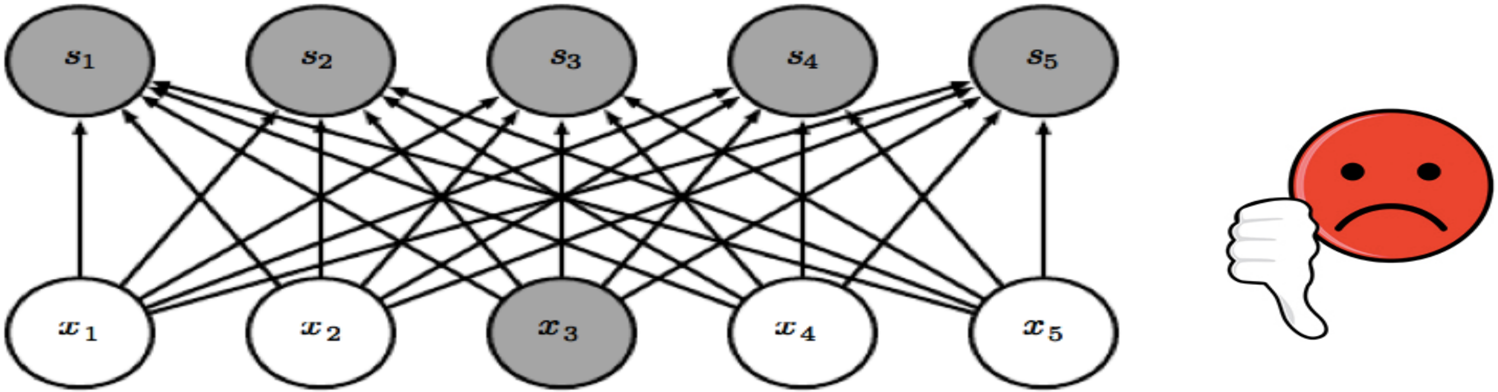
The convolution operation

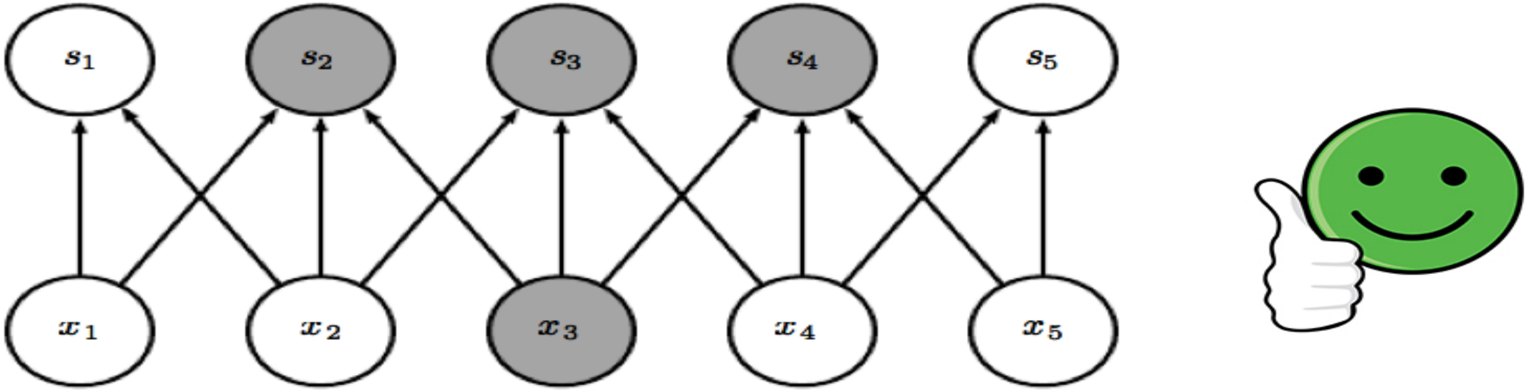




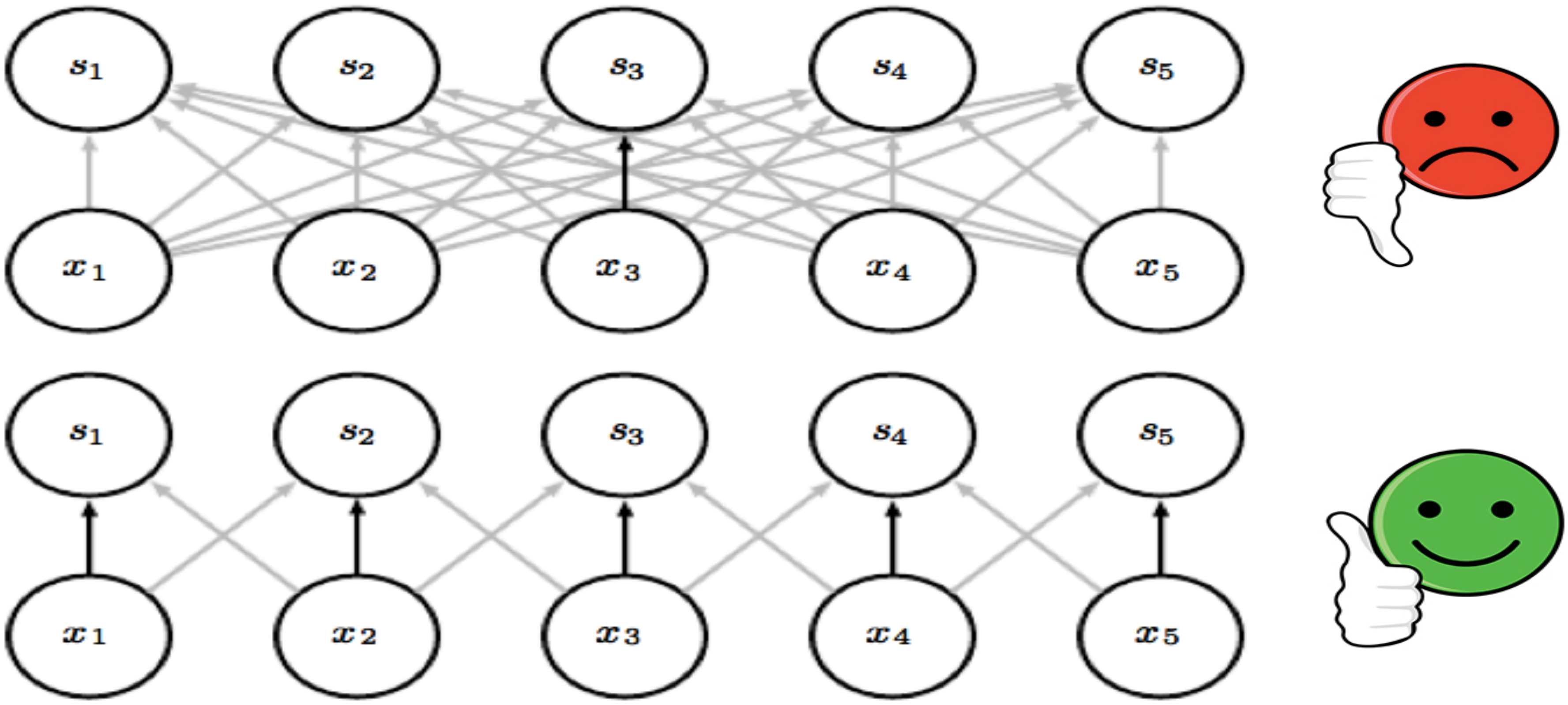
3 reasons why convolution is cool

Reason 1 : Sparse Connectivity





Reason 2 : Parameter sharing



Reason 3 : Equivariant Representations

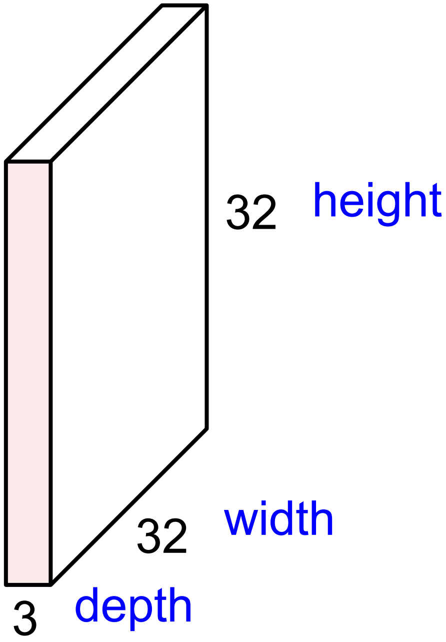
When the input changes -> output changes in the same way

Eg. Let I be a function giving images brightness at integer coordinates
Let g be a function mapping one image function to another image function,
such that $I' = g(I)$ is the image function with $I'(x,y) = I(x - 1,y)$.
This shifts every pixel of I one unit to the right.
If we apply this transformation to I, then apply convolution,
the result will be the same as if we applied convolution to I',
then applied the transformation g to the output.

Convolution Layers

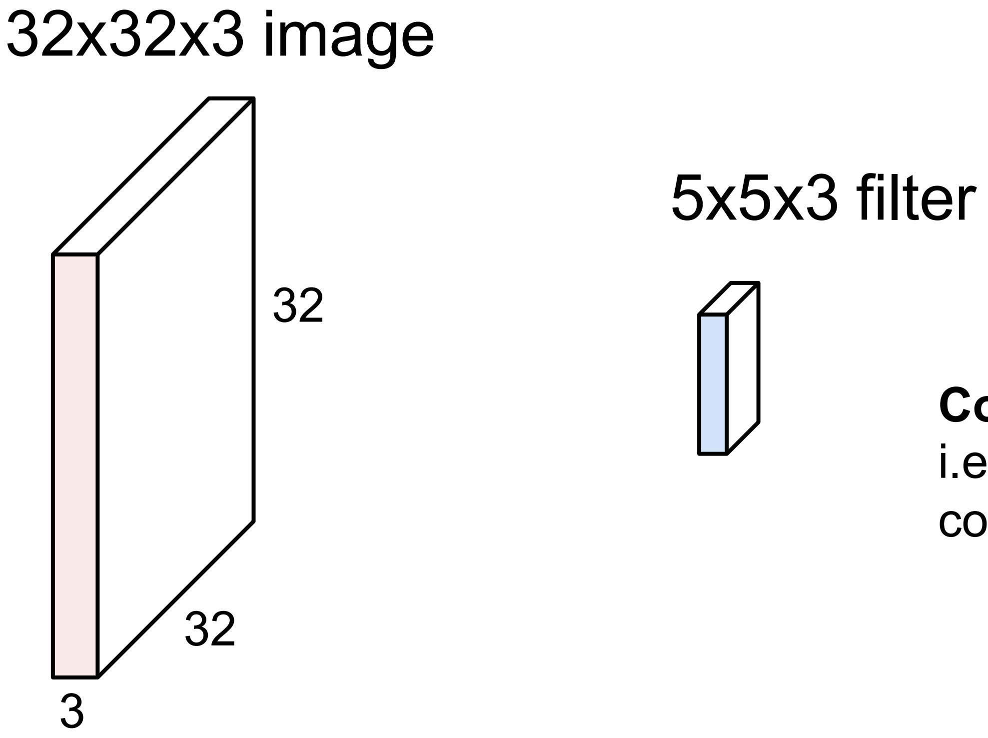
Convolution Layer

32x32x3 image



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer



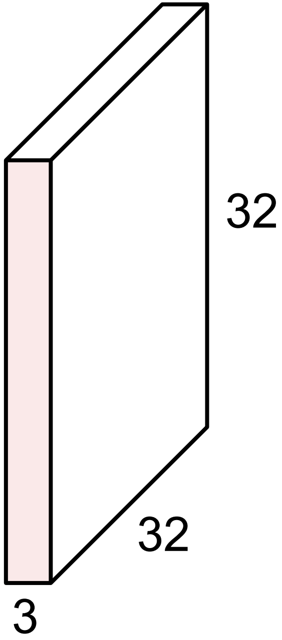
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

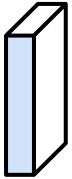
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



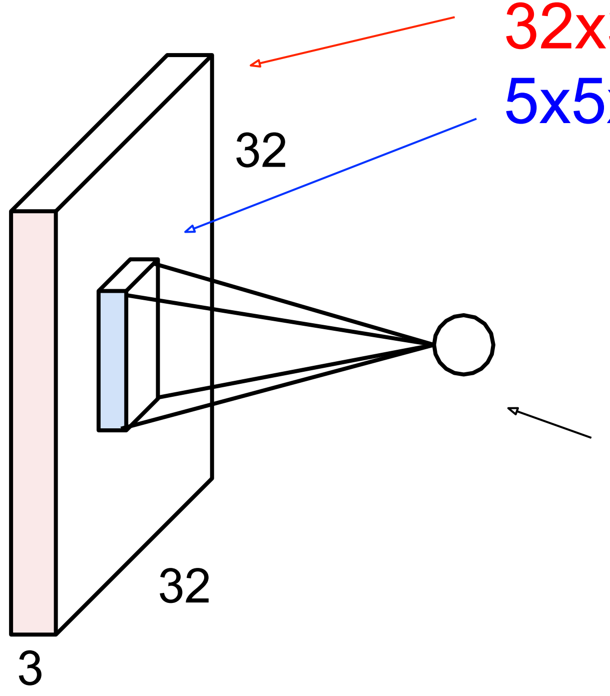
5x5x3 filter



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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

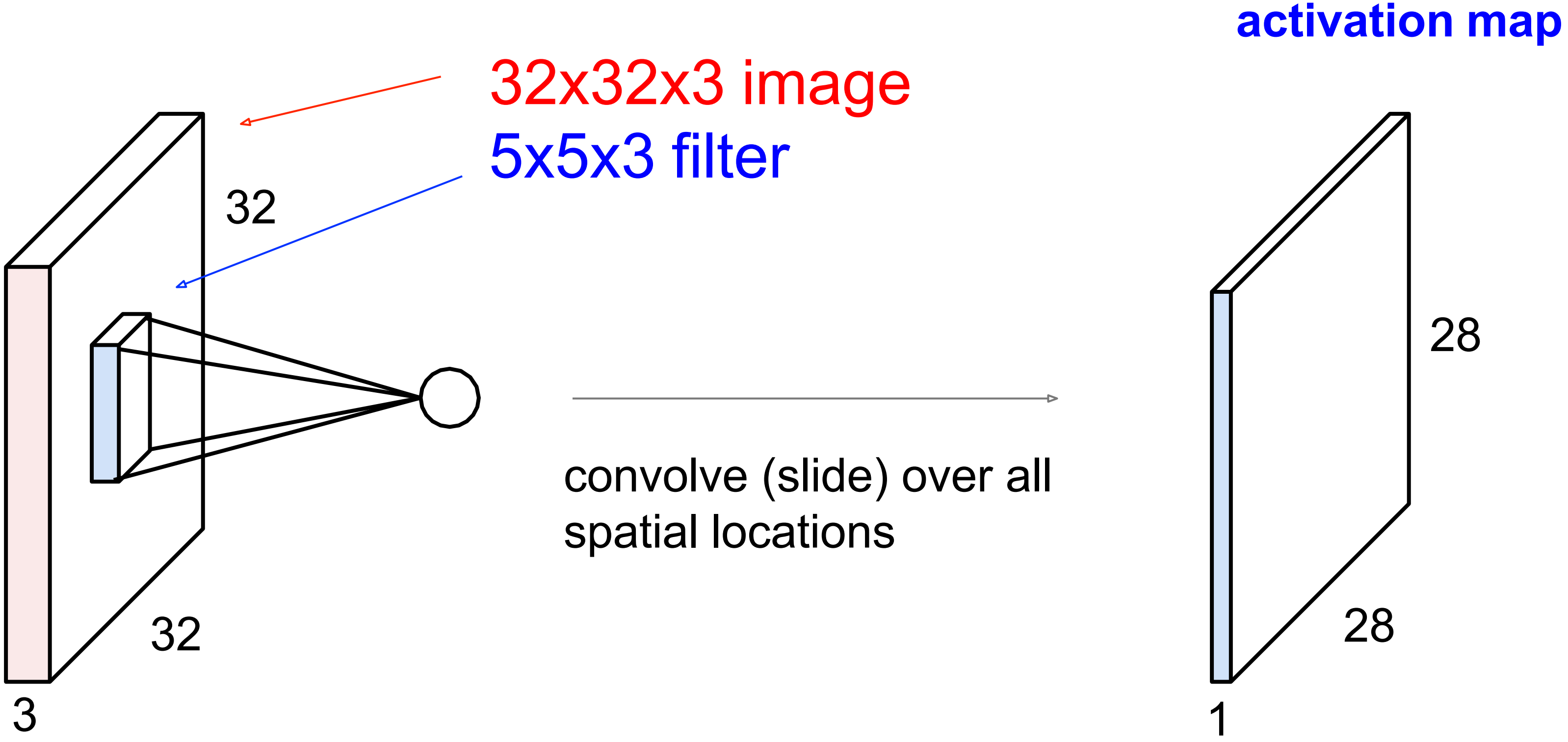


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer



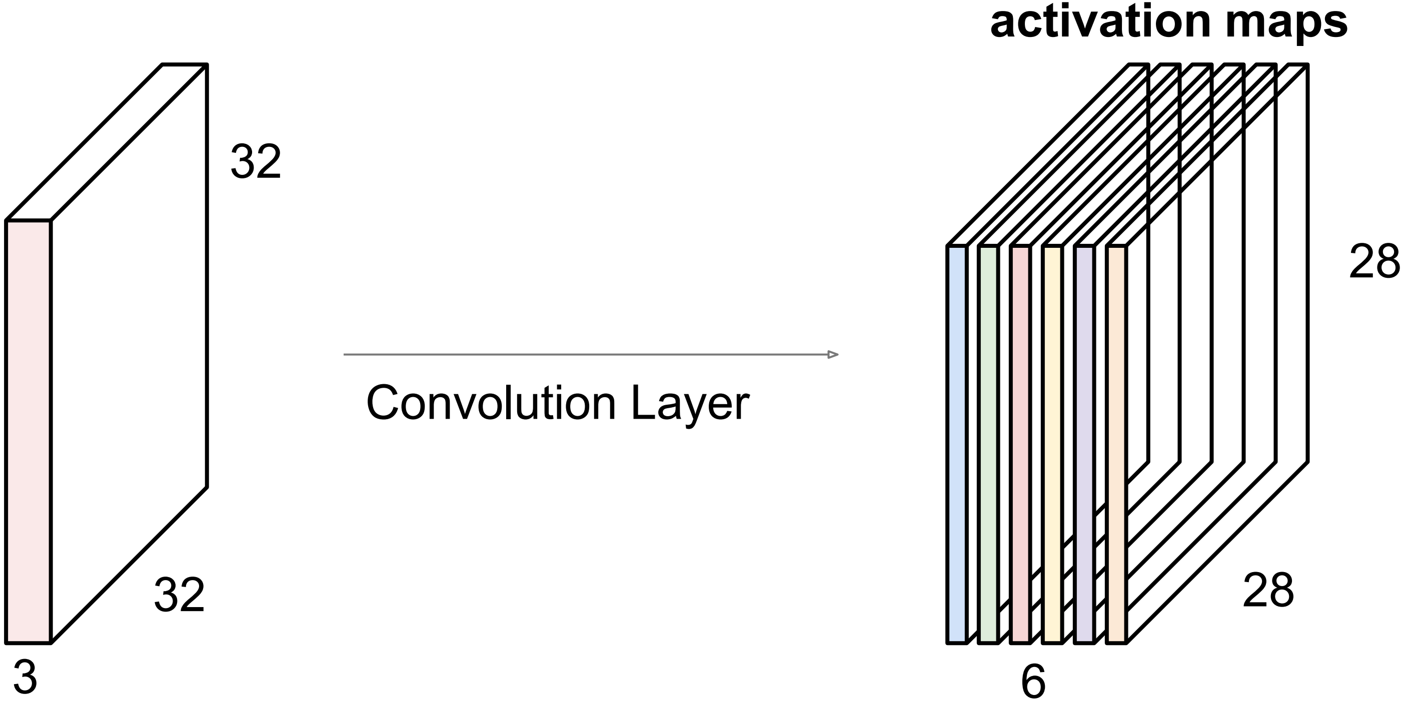
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

consider a second, green filter

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

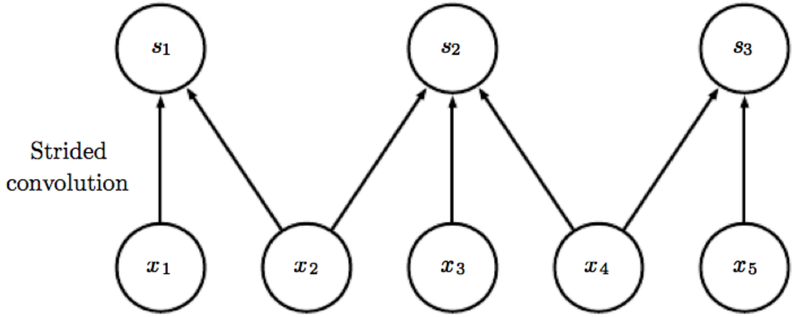
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

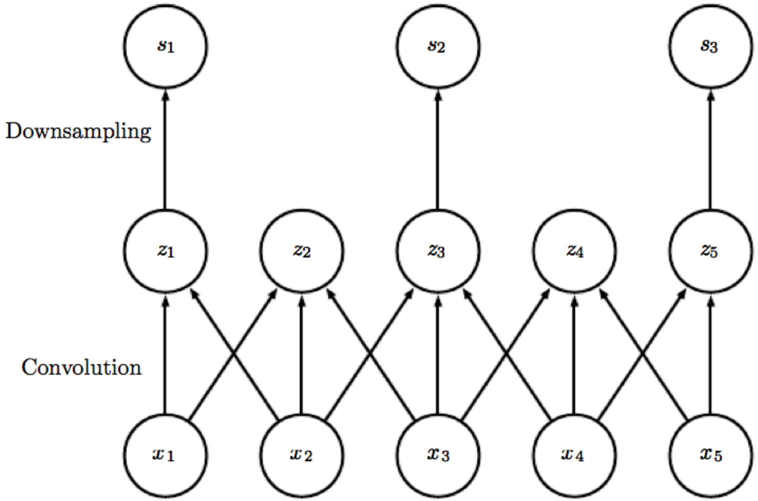


We stack these up to get a "new image" of size 28x28x6!

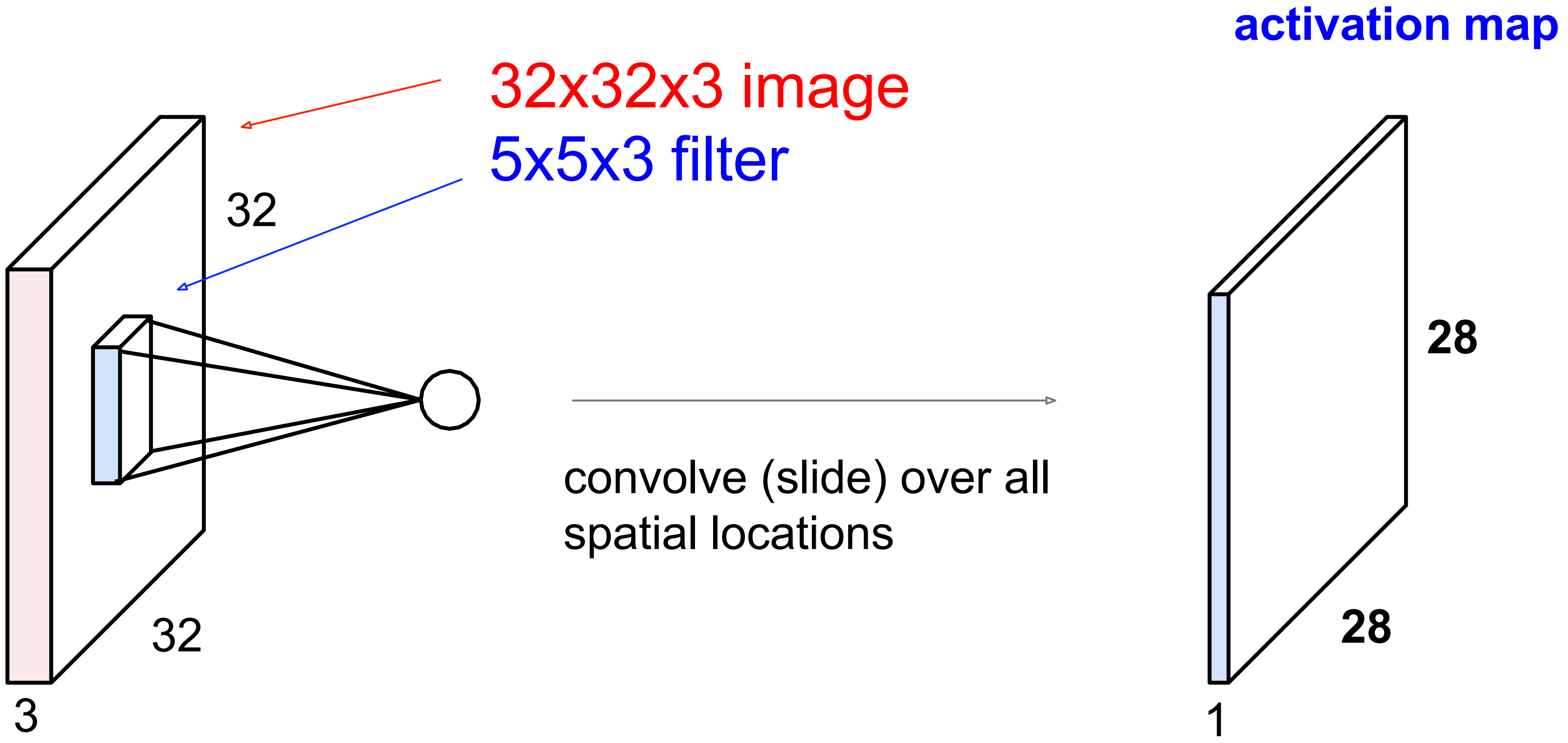
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Stride


s_1
s_2
s_3
Strided convolution
x_1
x_2
x_3
x_4
x_5


s_1
s_2
s_3
Downsampling
z_1
z_2
z_3
z_4
z_5
Convolution
x_1
x_2
x_3
x_4
x_5

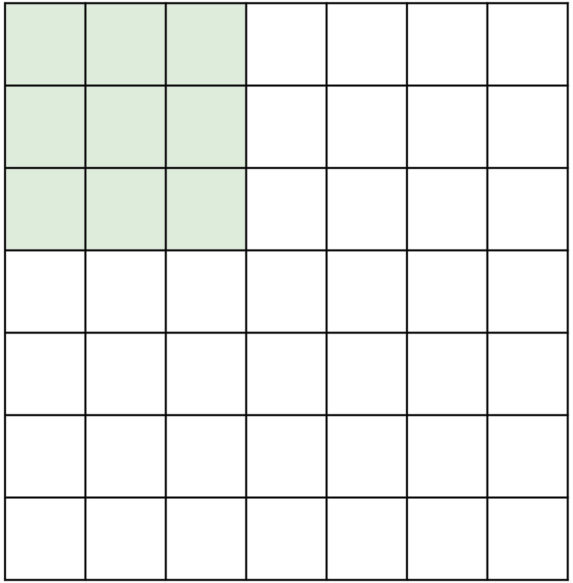
A closer look at spatial dimensions:



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

A closer look at spatial dimensions:

7

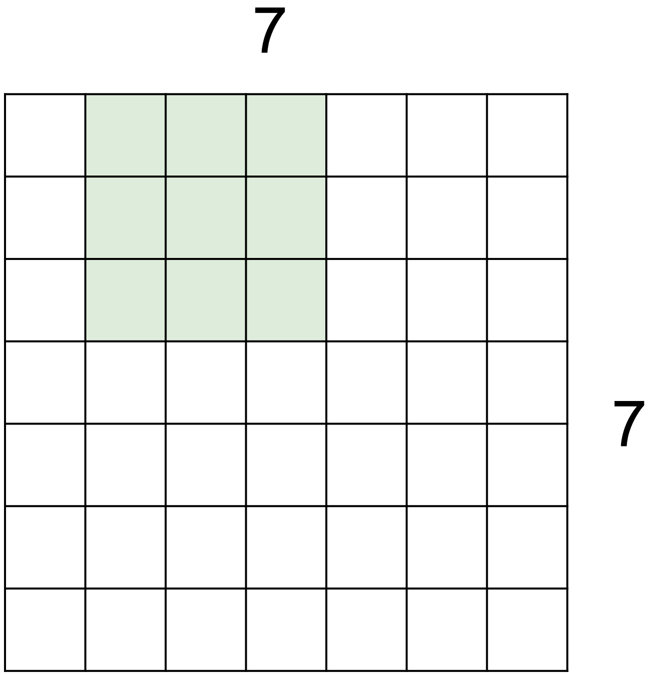


7

7x7 input (spatially)
assume 3x3 filter

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

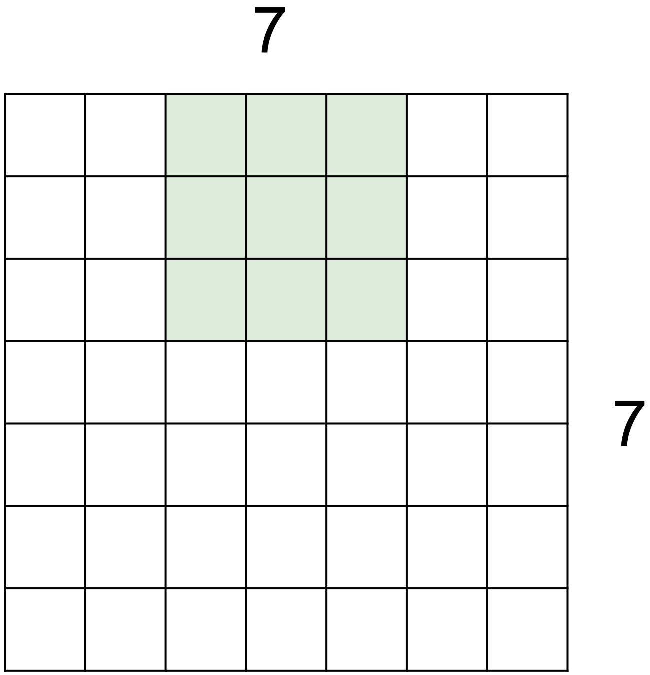
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

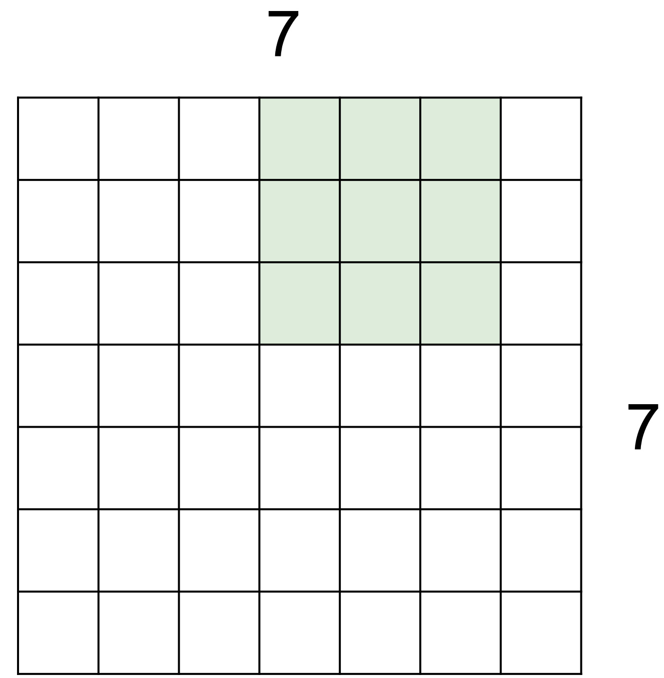
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

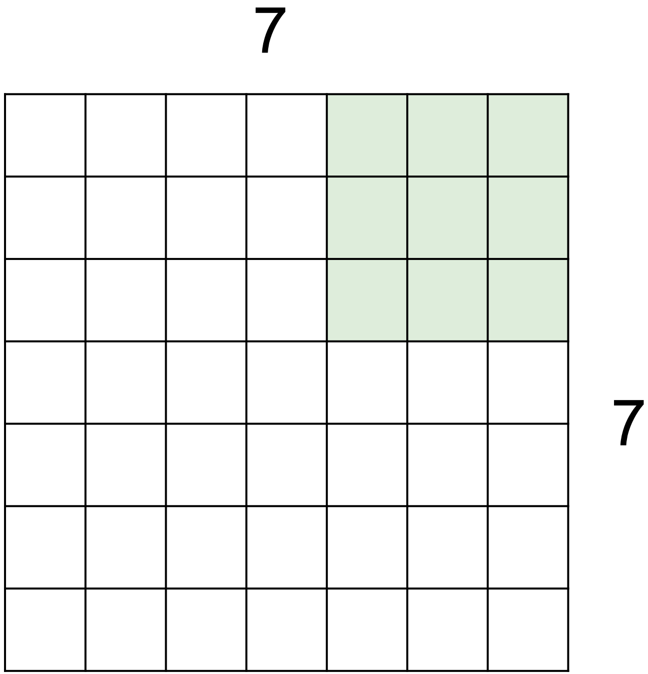
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

A closer look at spatial dimensions:

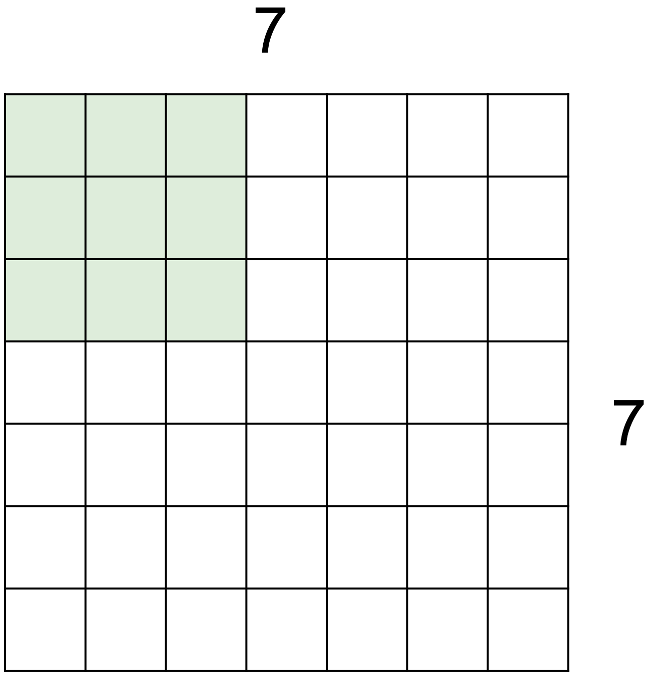


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

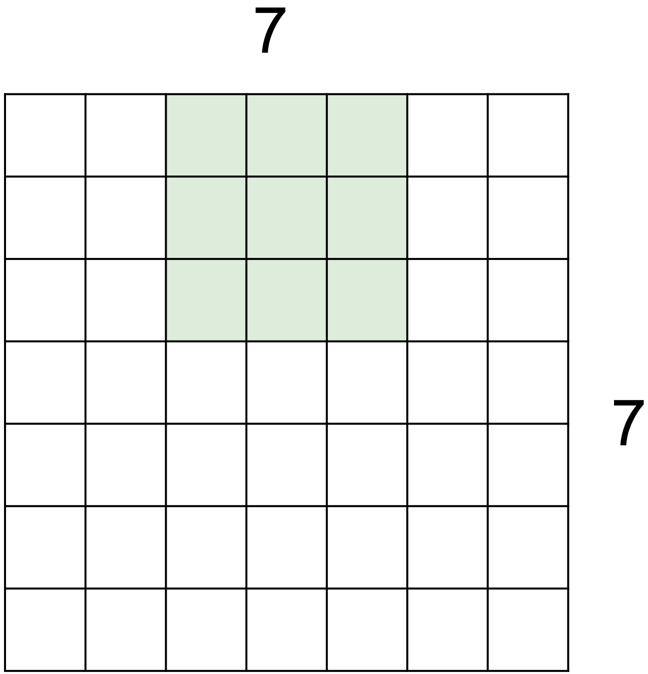
A closer look at spatial dimensions:



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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

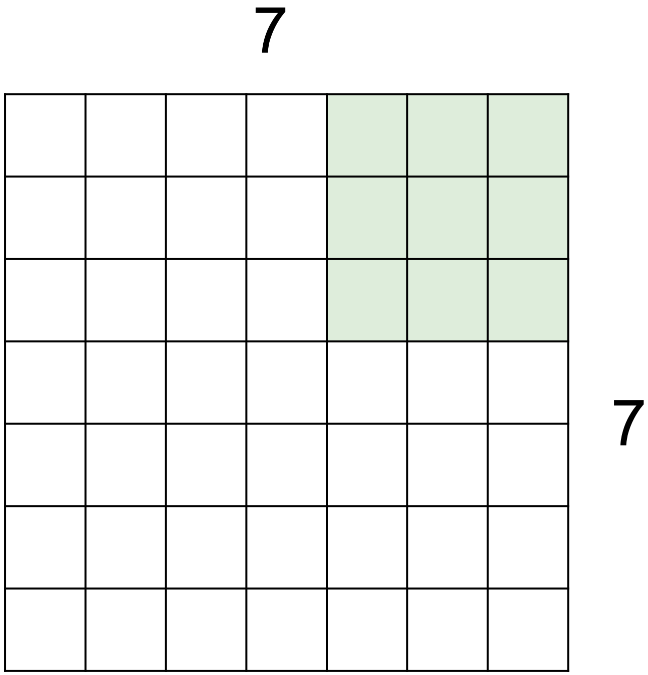
A closer look at spatial dimensions:



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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

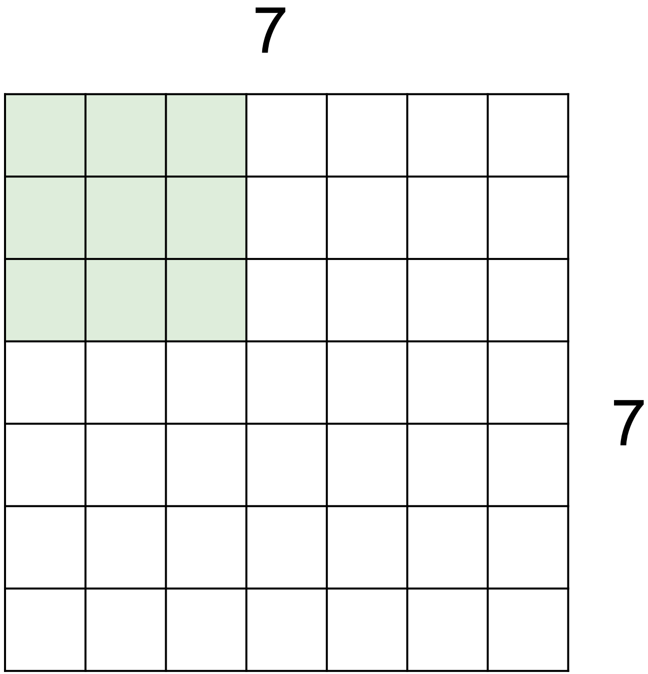
A closer look at spatial dimensions:



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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

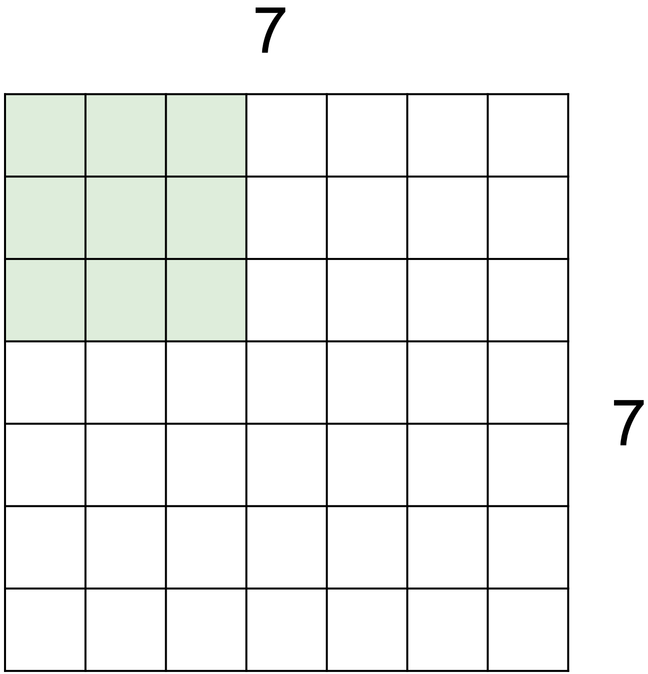
A closer look at spatial dimensions:



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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

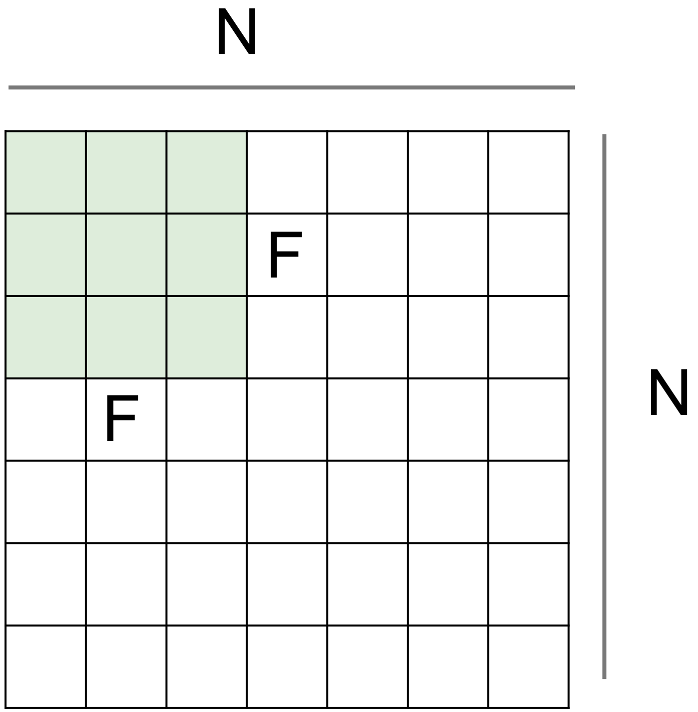
A closer look at spatial dimensions:



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

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

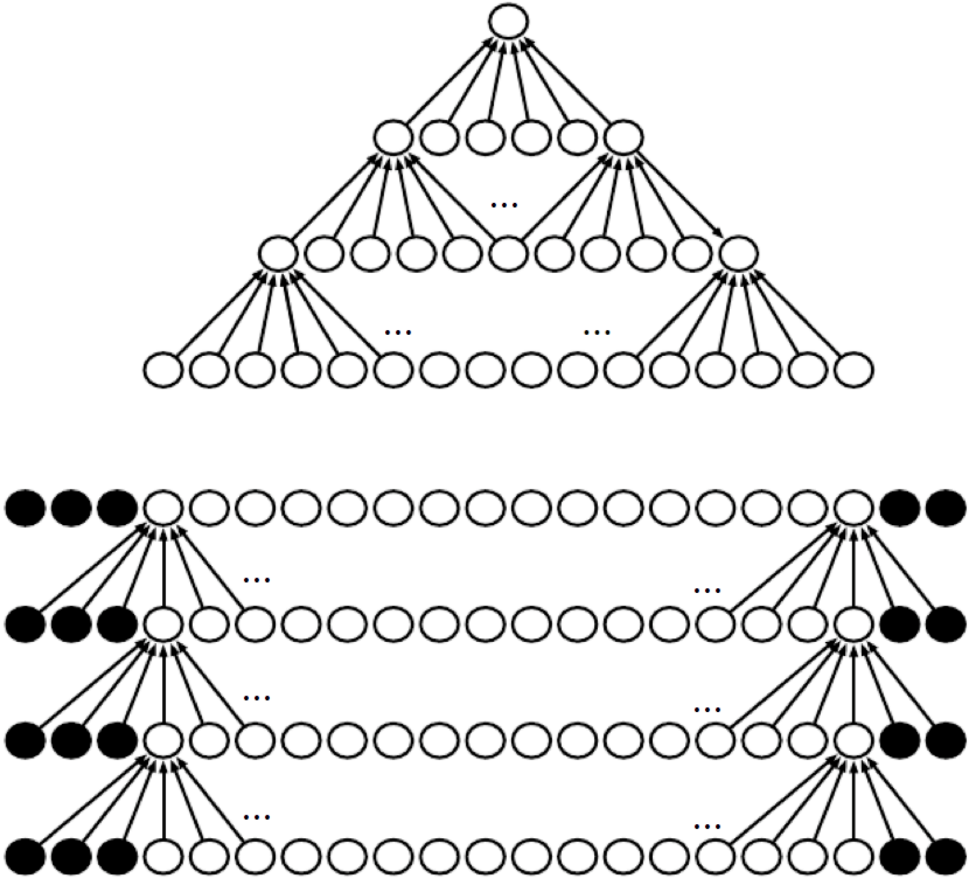


Output size:
(N - F) / stride + 1

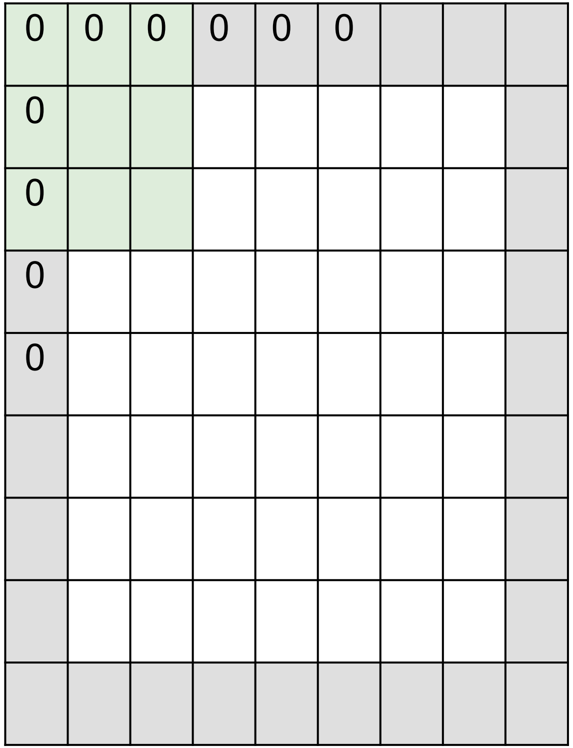
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Zero-Padding



Zero-Padding: common to the border

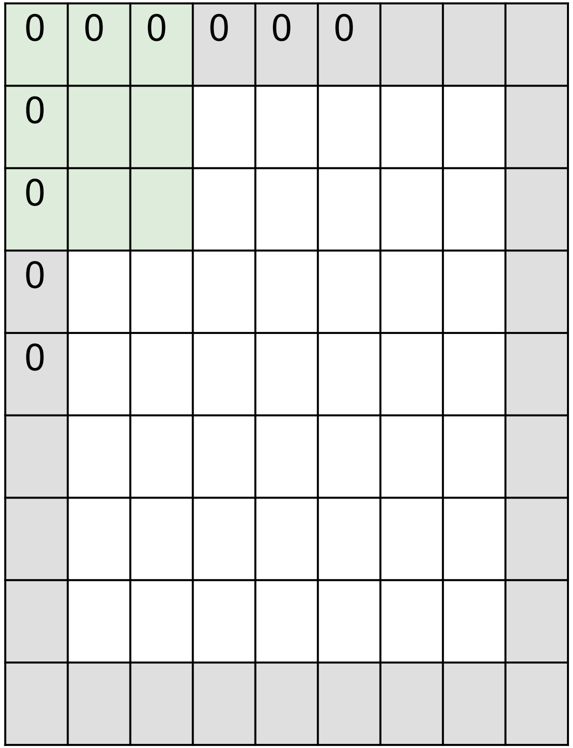


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Zero-Padding: common to the border

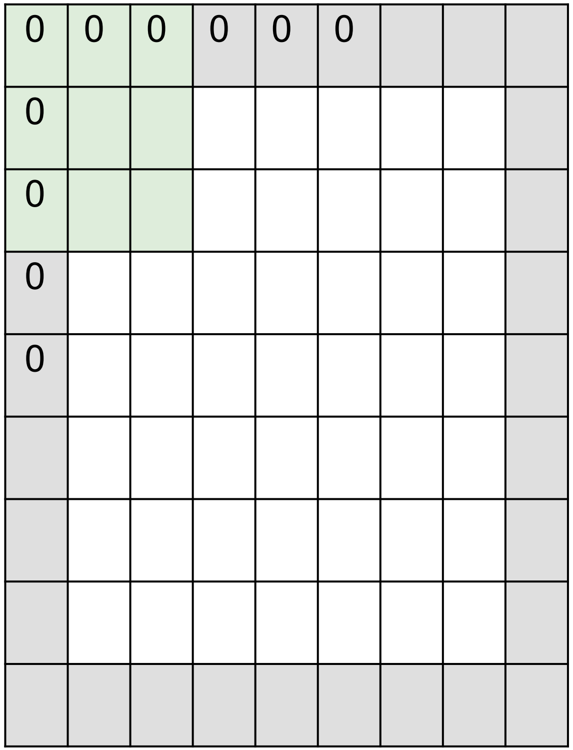


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Zero-Padding: common to the border

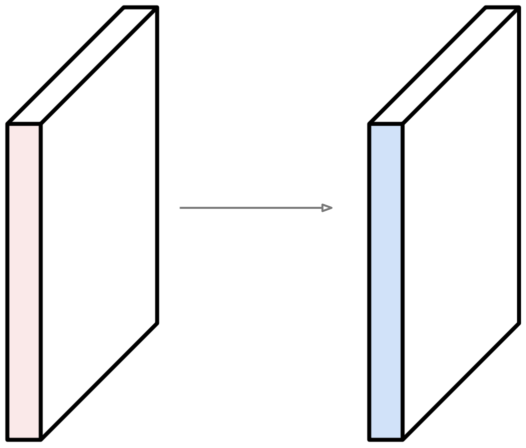


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Examples time:



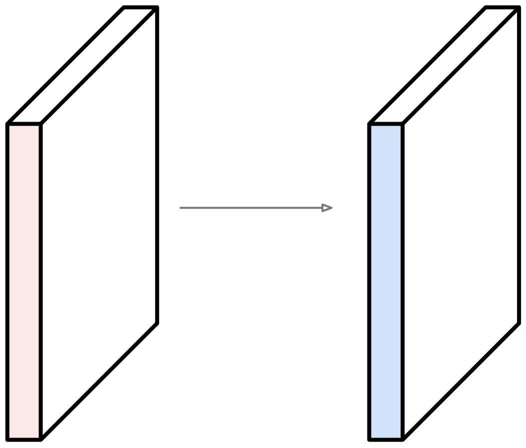
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size: ?

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

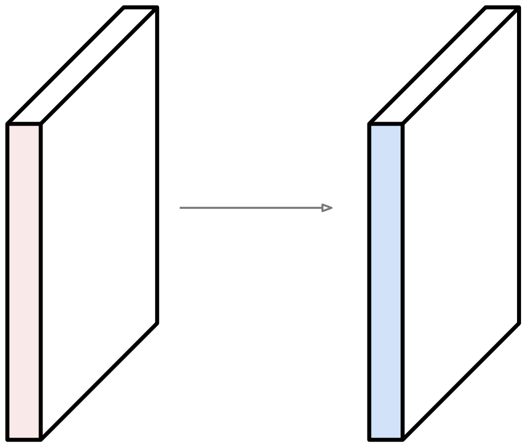
(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

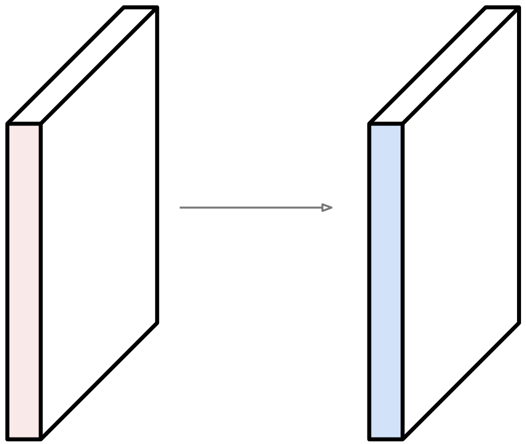
Examples time:

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



Number of parameters in this layer?

Examples time:



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**

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Summary

- 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.

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

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

Summary. To summarize, the Conv Layer:

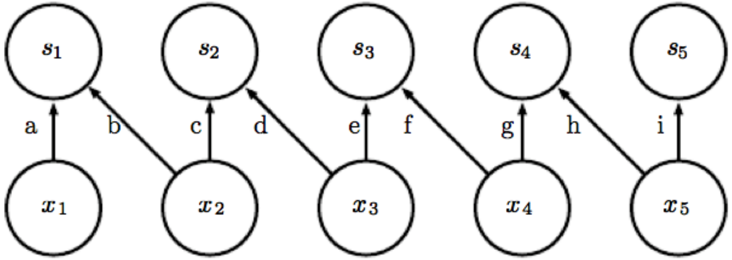
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (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.

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

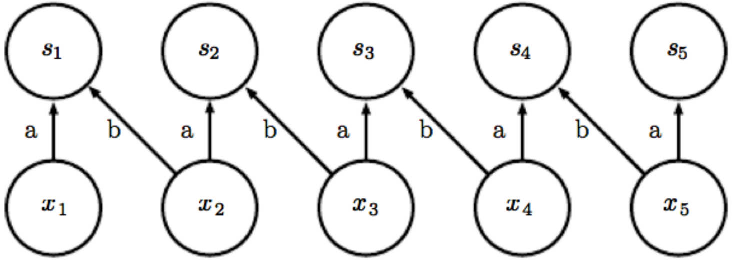
Local connectivity & tiled convolution

Local connectivity

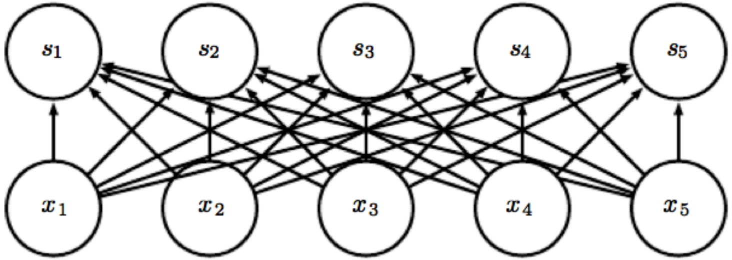
Locally connected layer



Convolutional layer

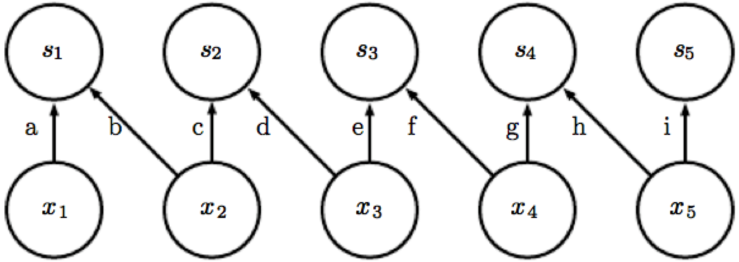


Fully connected layer

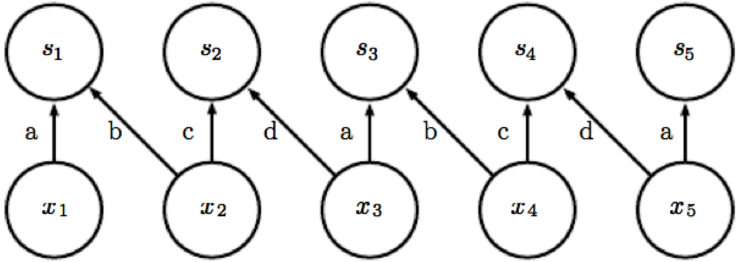


Tiled convolution

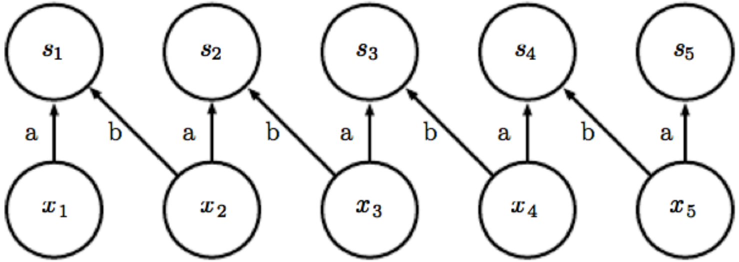
Locally connected layer



Tiled convolution

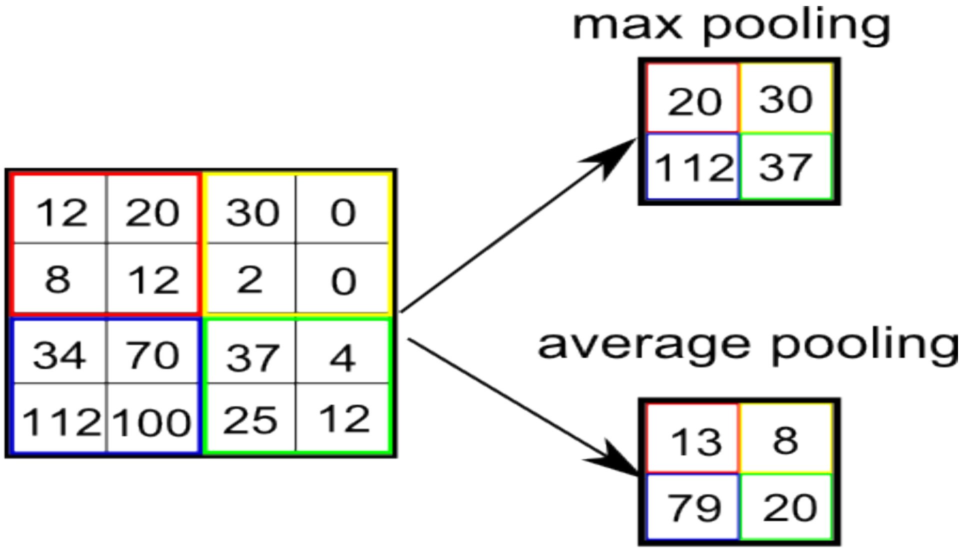


Convolutional layer



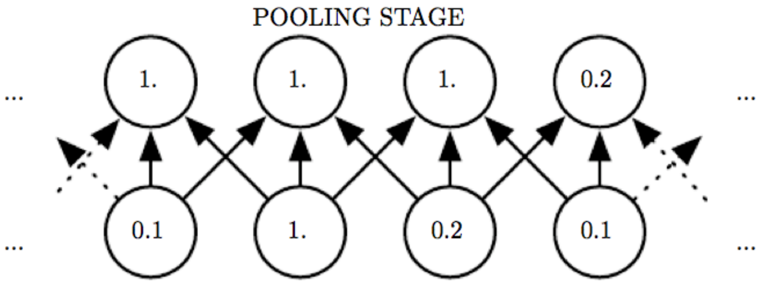
Pooling

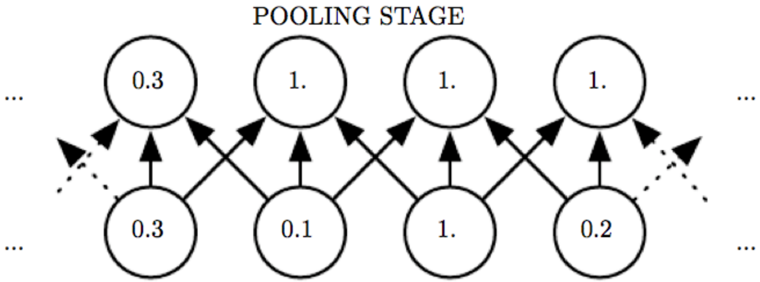
Pooling



Effect = invariance to small translations of the input

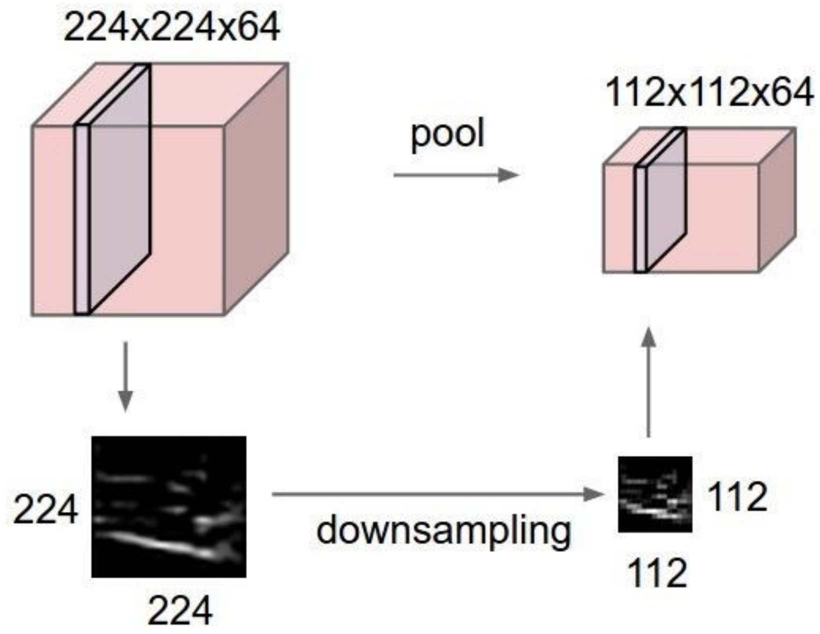
Pooling





Pooling

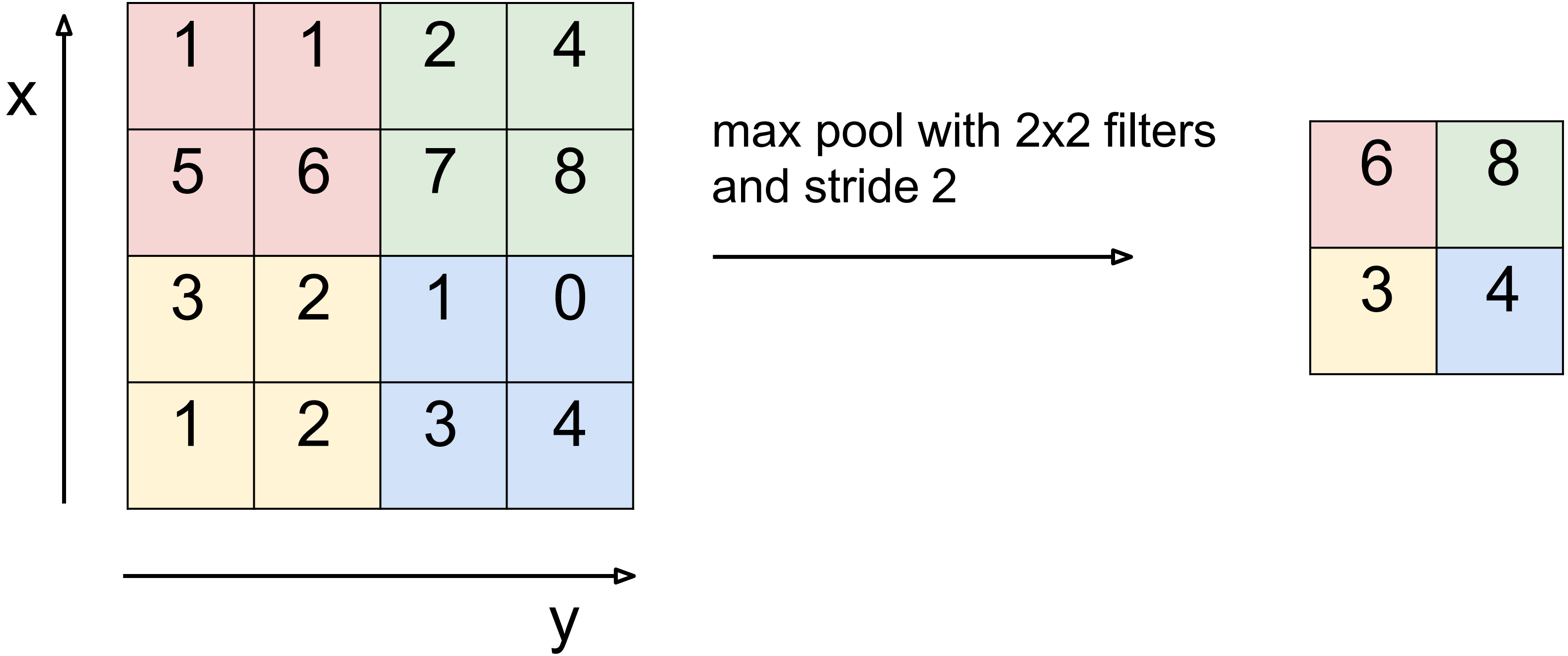
- makes the representations smaller and more manageable
- operates over each activation map independently



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Max Pooling

Single depth slice



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Summary

- 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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

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 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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

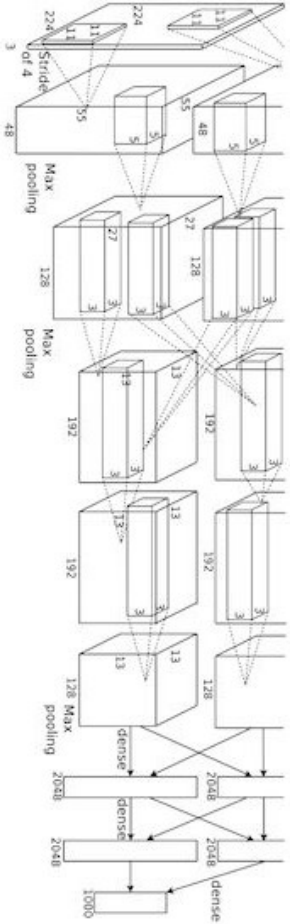
Back propagation

Convolutional Network (AlexNet)

input image

weights

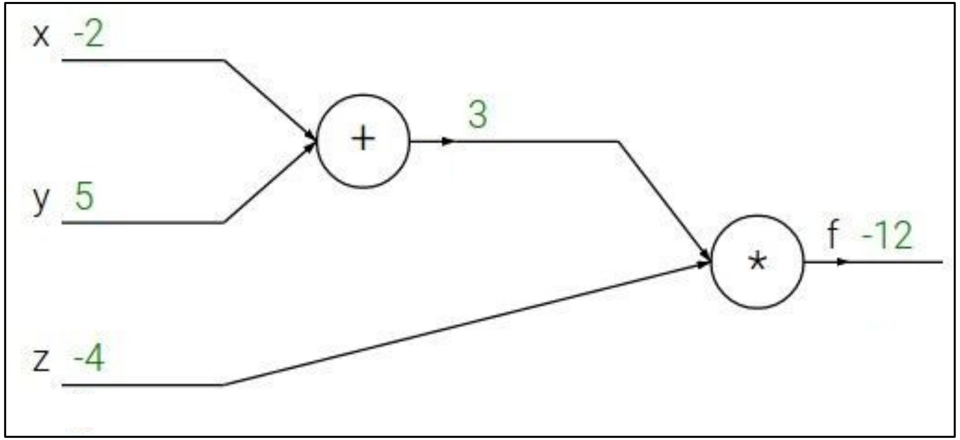
loss



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x, y, z) = (x + y)z$$

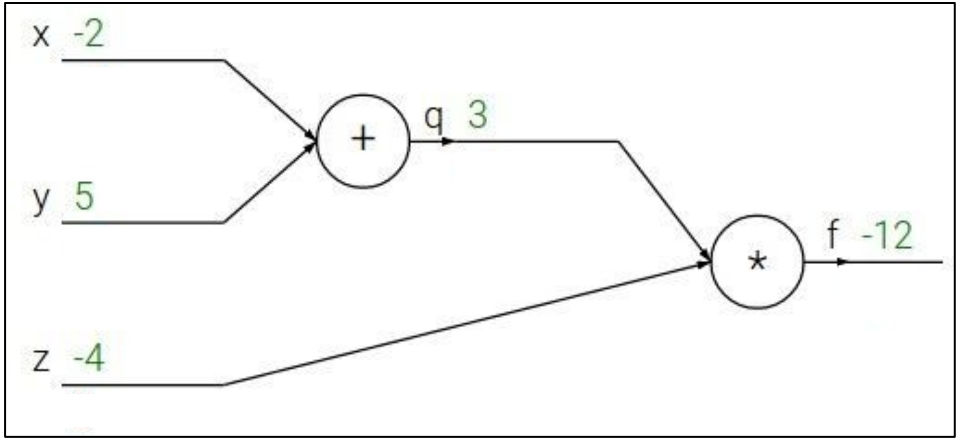
e.g. x = -2, y = 5, z = -4



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

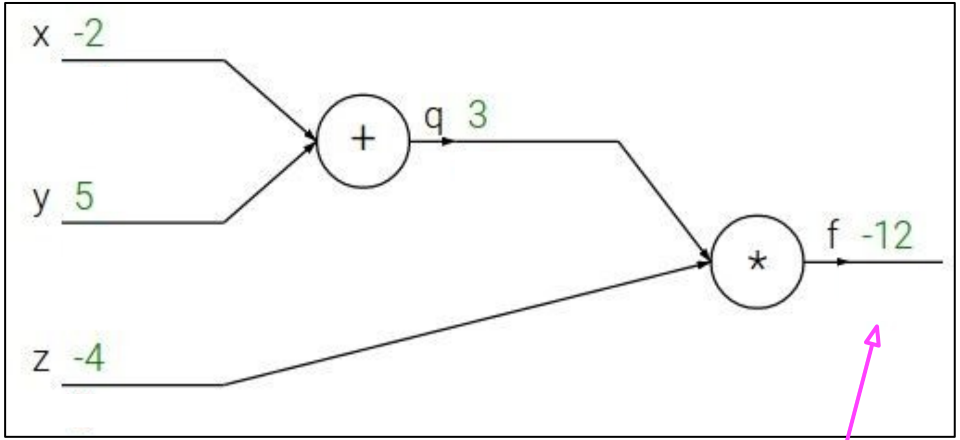
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial f}$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

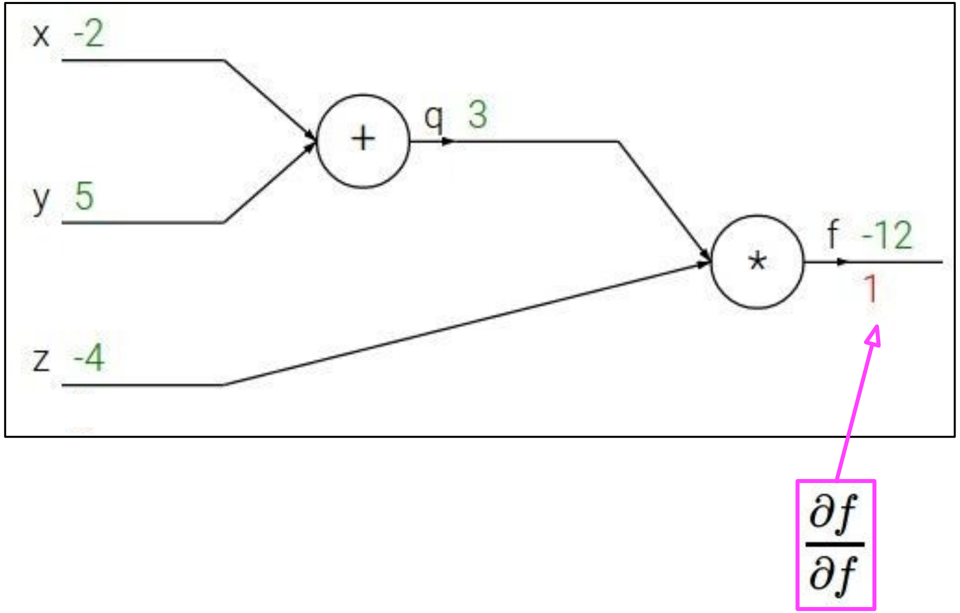
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

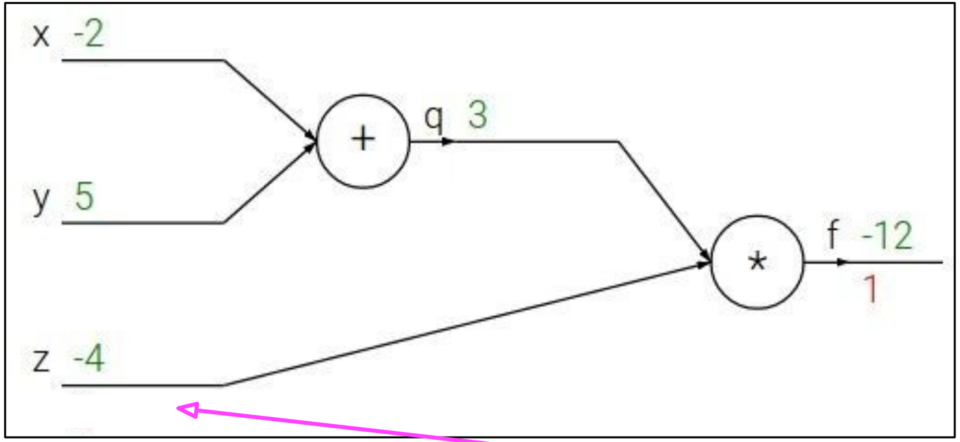
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial z}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

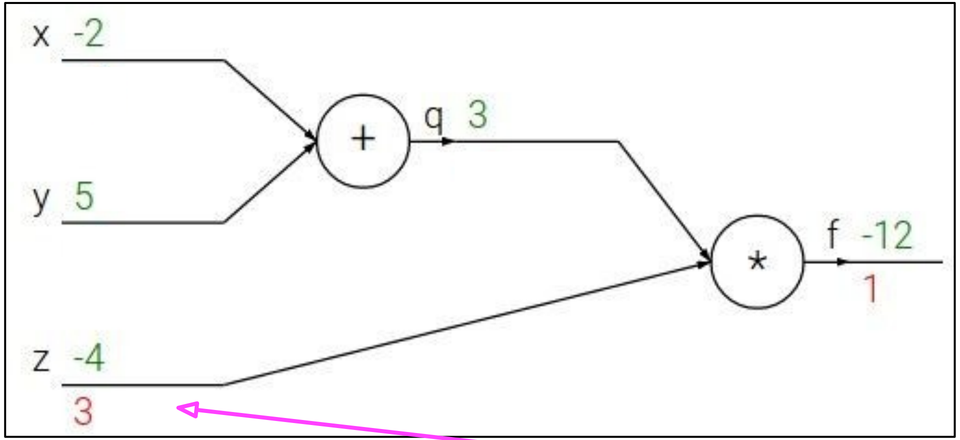
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial z}$

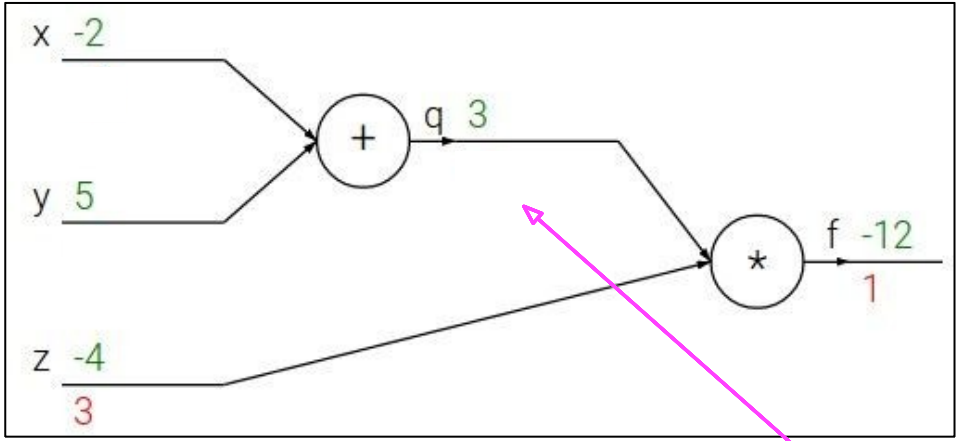
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



$$\frac{\partial f}{\partial q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

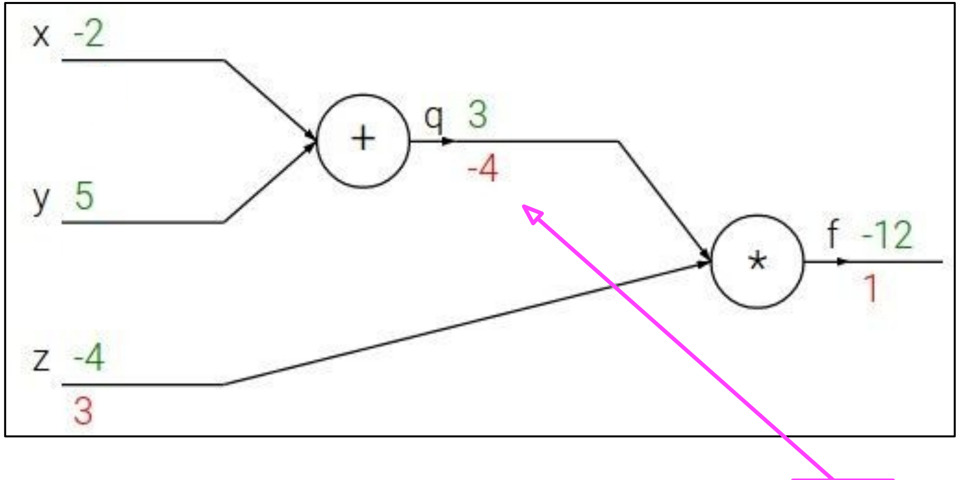
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

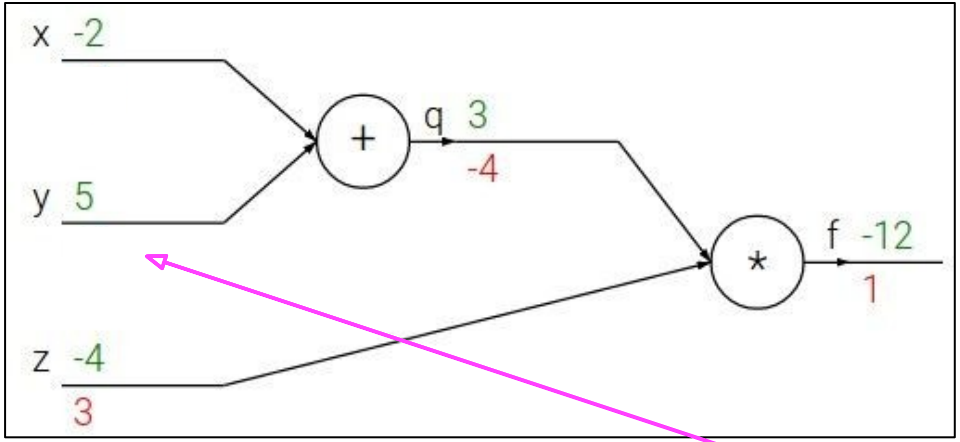
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



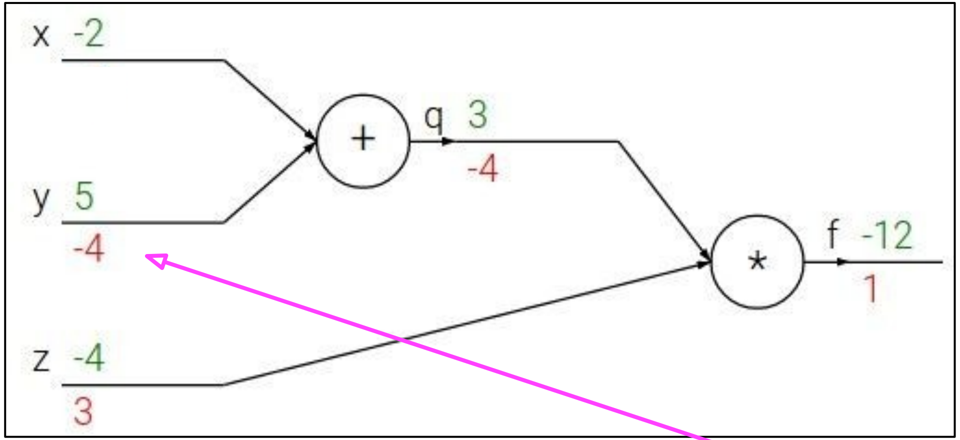
$$\frac{\partial f}{\partial y}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial y}$$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

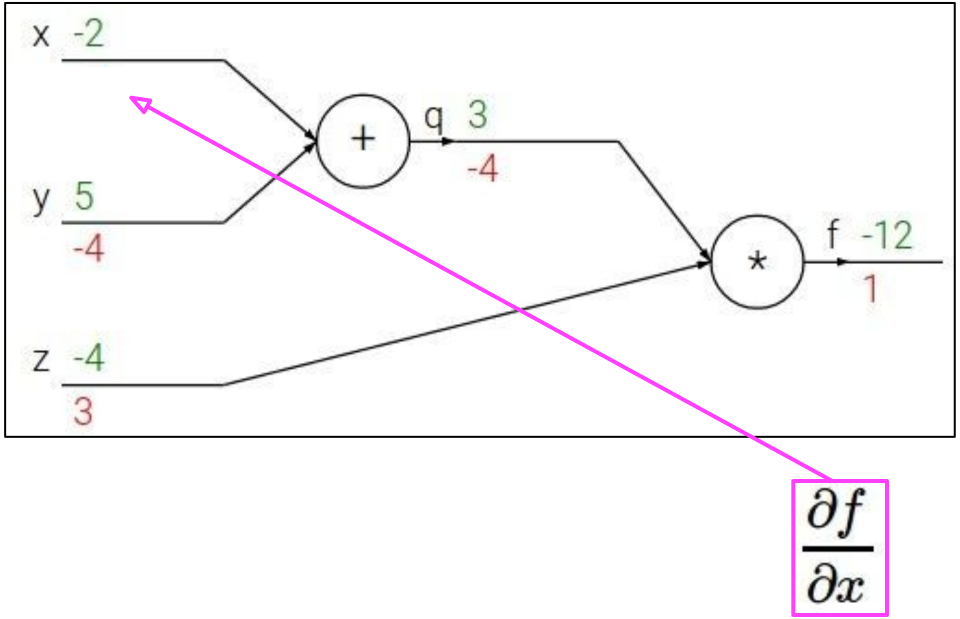
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

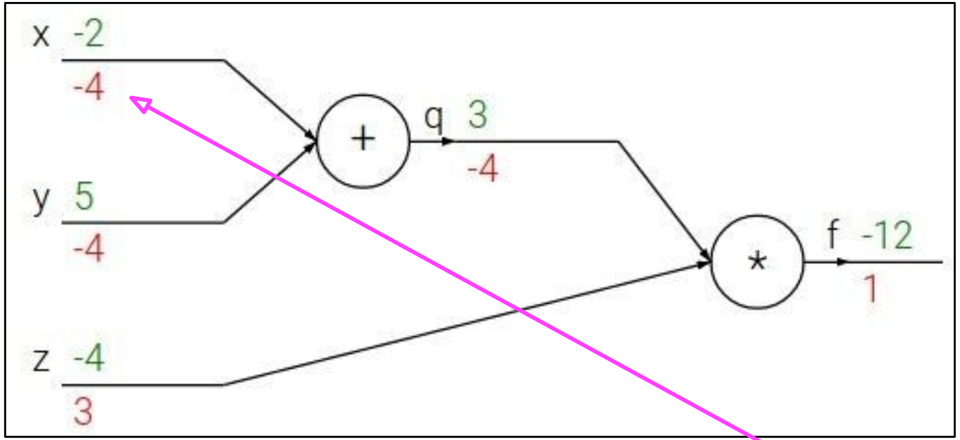
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

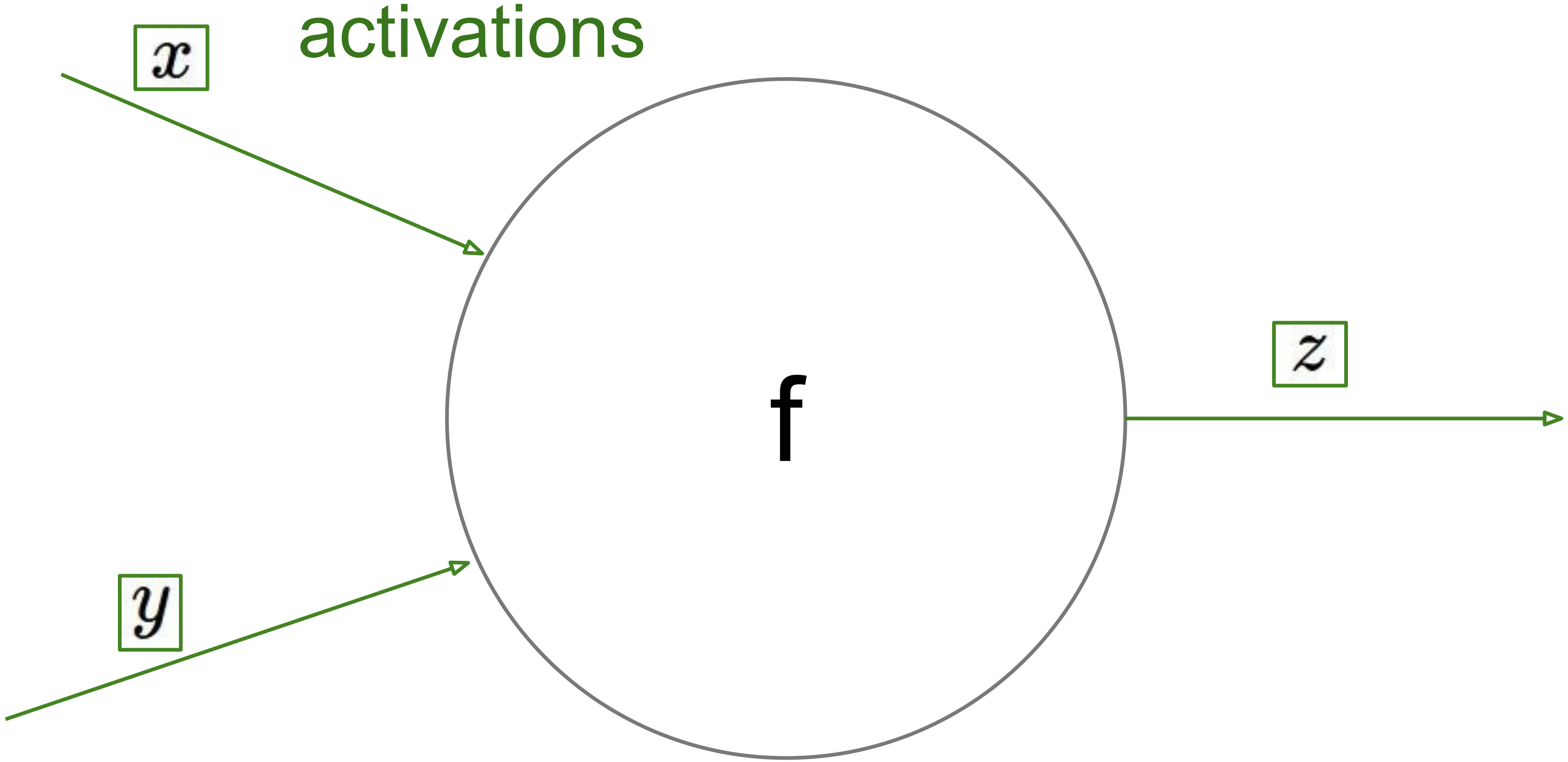
$$\frac{\partial f}{\partial x}$$

Chain rule:

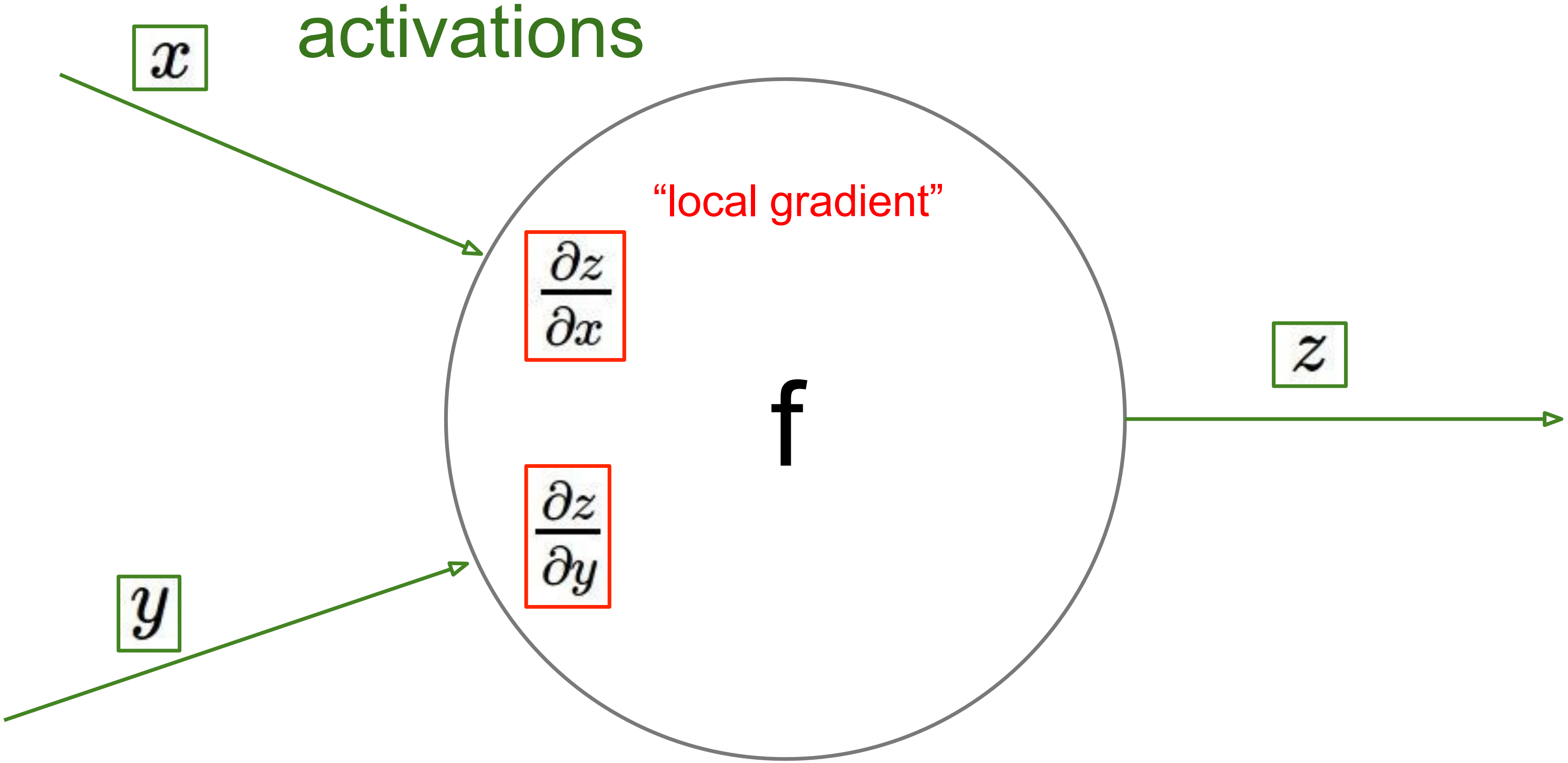
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

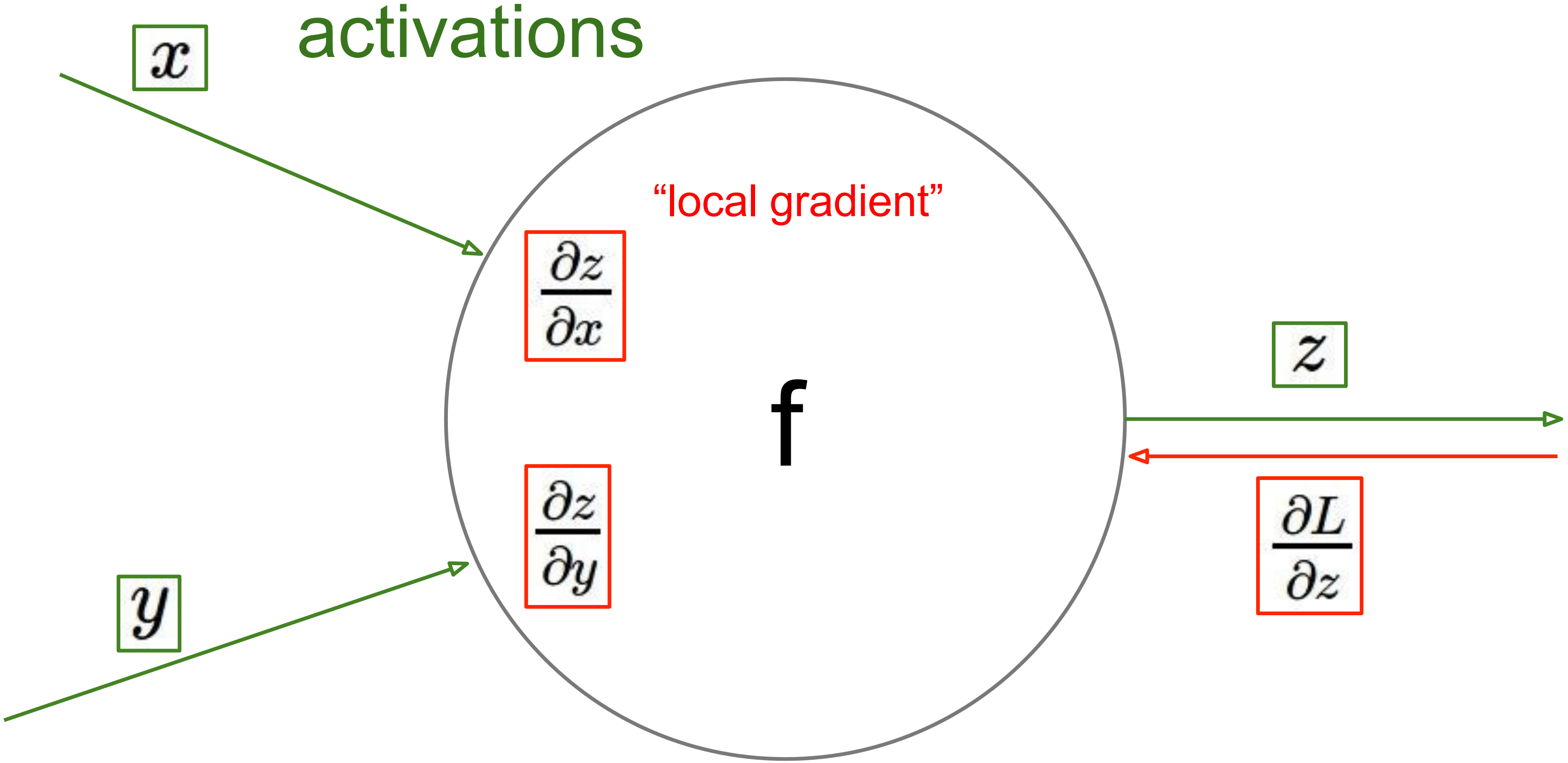
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson



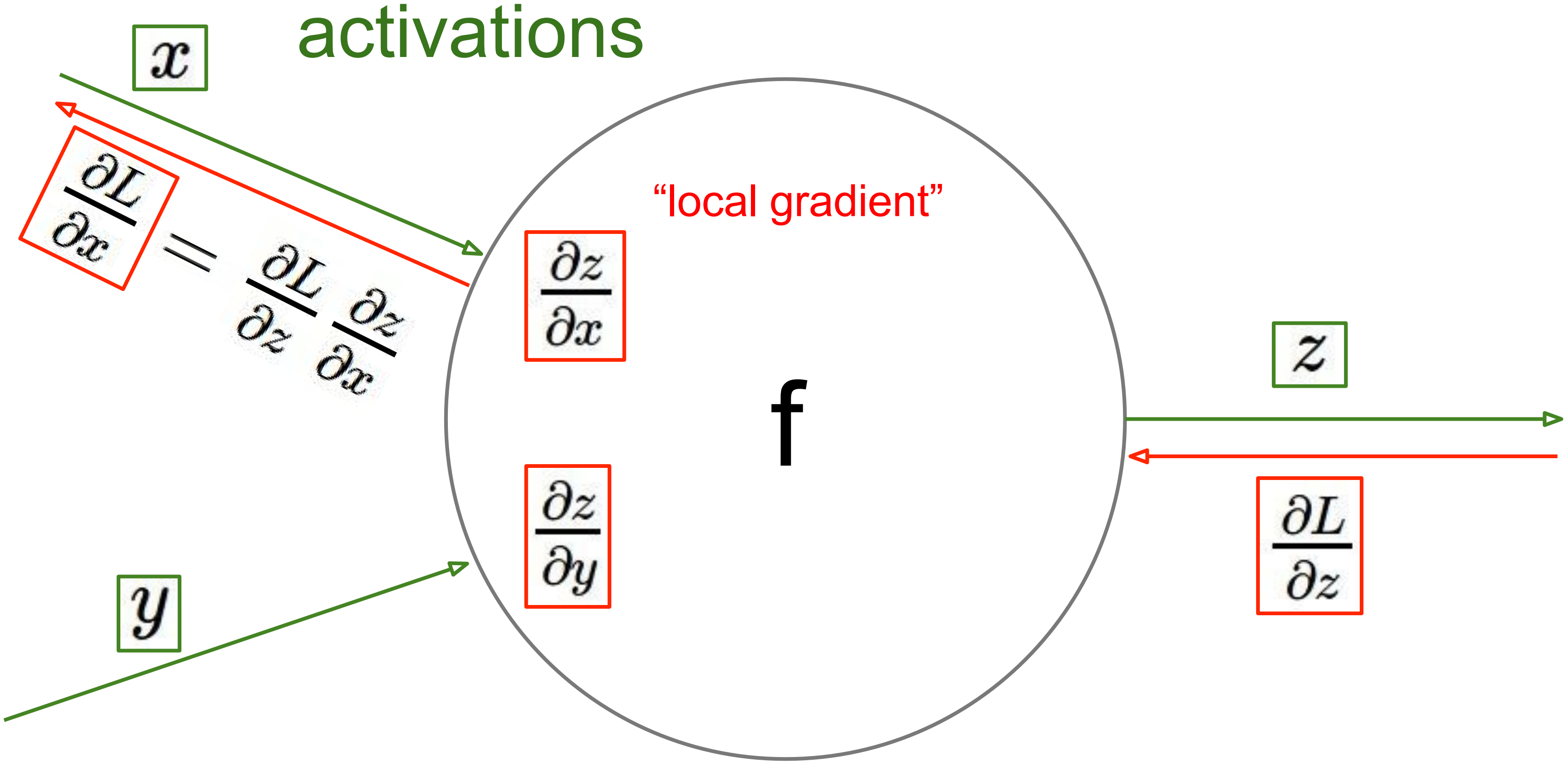
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson



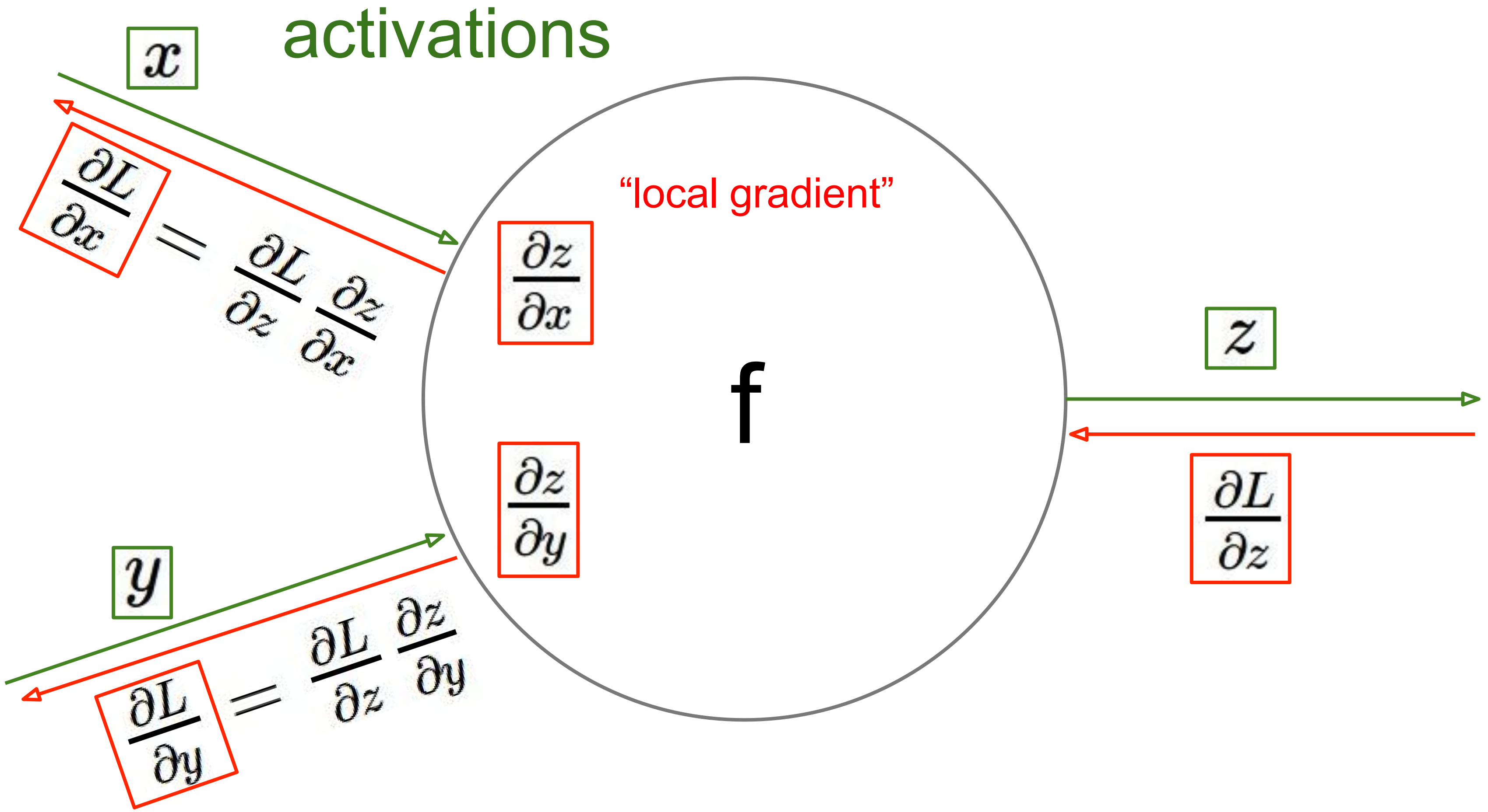
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson



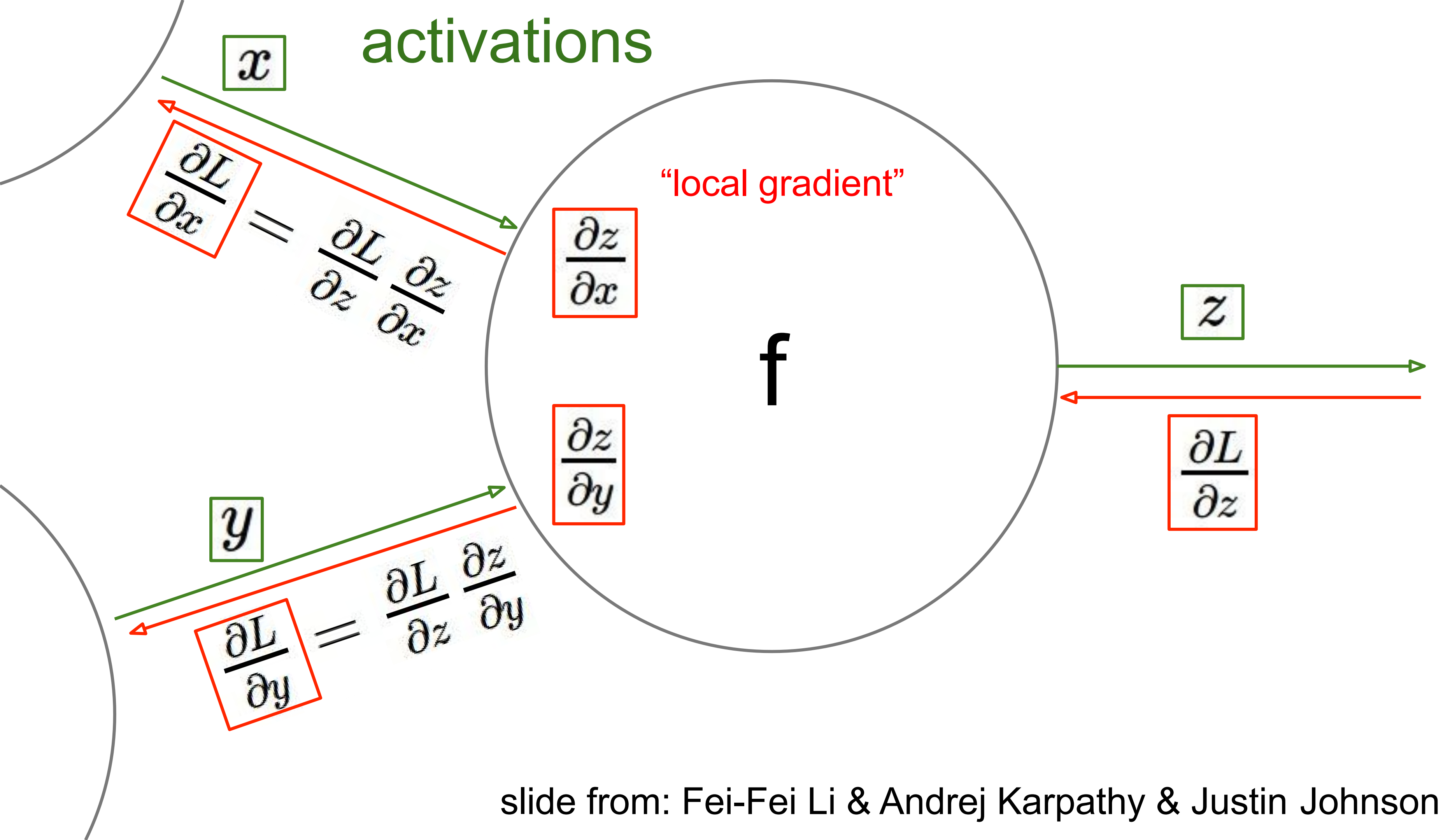
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson



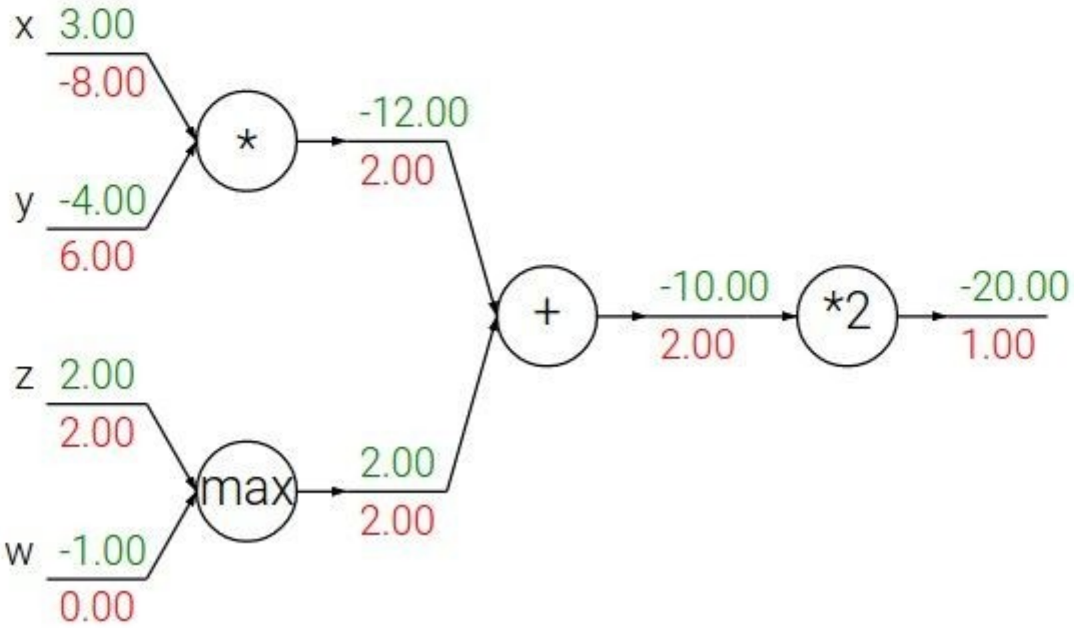
slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Patterns in backward flow

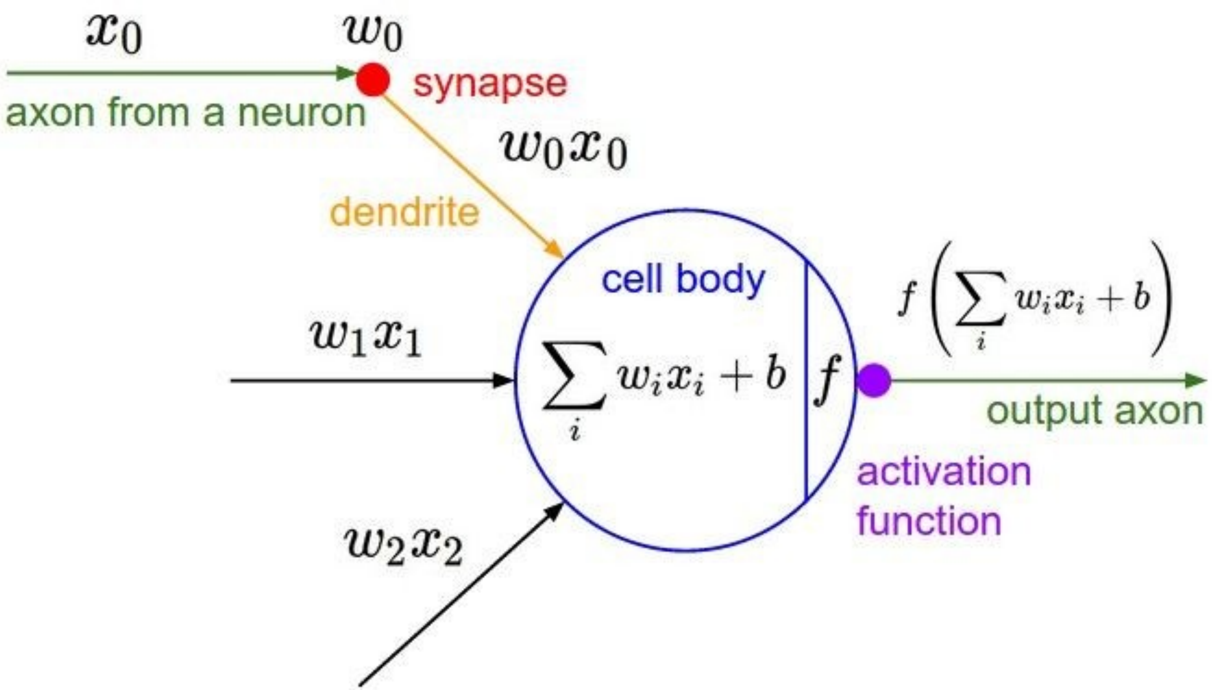
add gate: gradient distributor
max gate: gradient router
mul gate: gradient… “switcher”?



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation function

Activation Functions

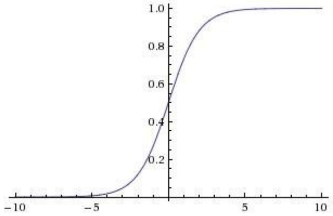


slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

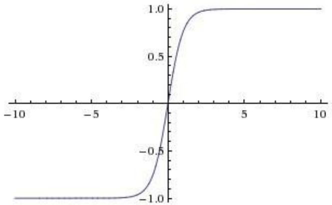
Activation Functions

Sigmoid

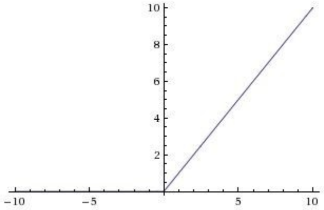
$$\sigma(x) = 1/(1+e^{-x})$$



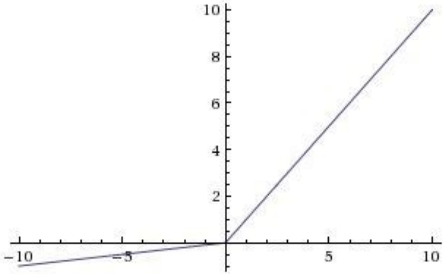
tanh tanh(x)



ReLU max(0,x)



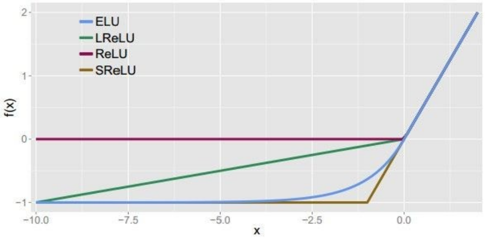
Leaky ReLU



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

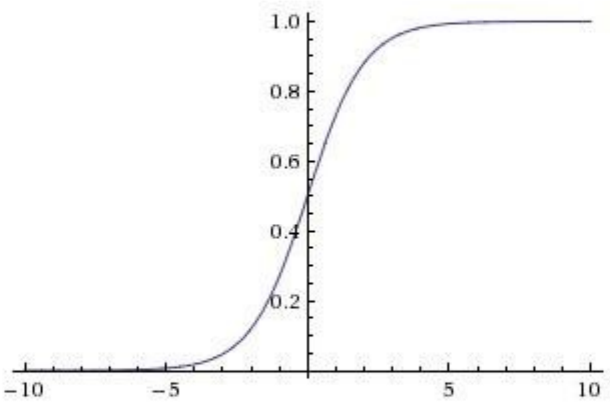
ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\ (\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



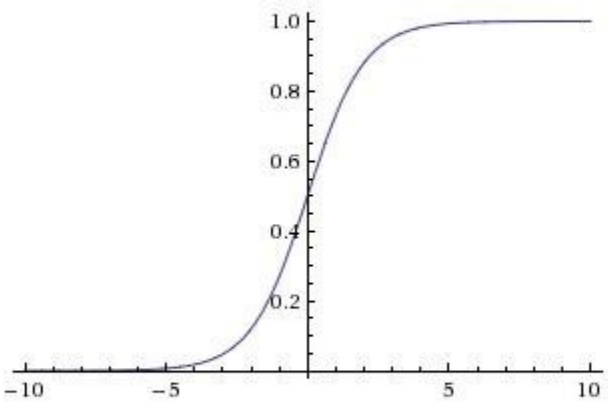
Sigmoid

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

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



Sigmoid

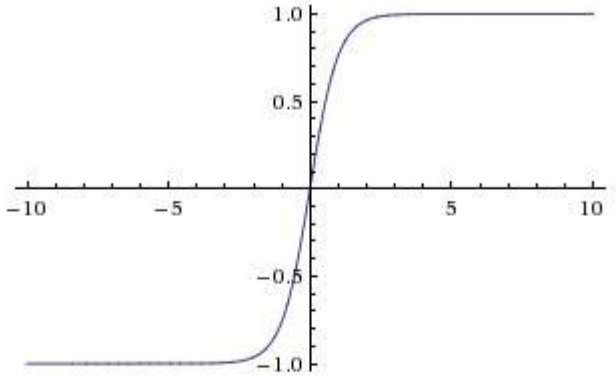
$$\sigma(x) = 1/(1 + e^{-x})$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



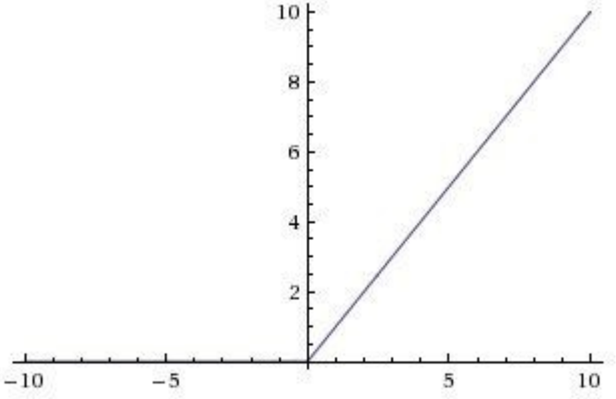
tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

[LeCun et al., 1991]

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



ReLU
(Rectified Linear Unit)

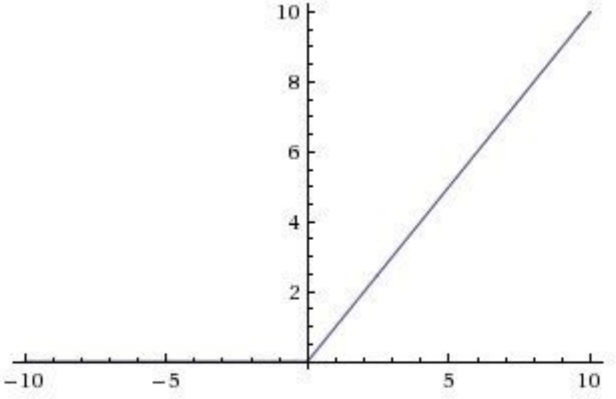
Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

[Krizhevsky et al., 2012]

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



ReLU
(Rectified Linear Unit)

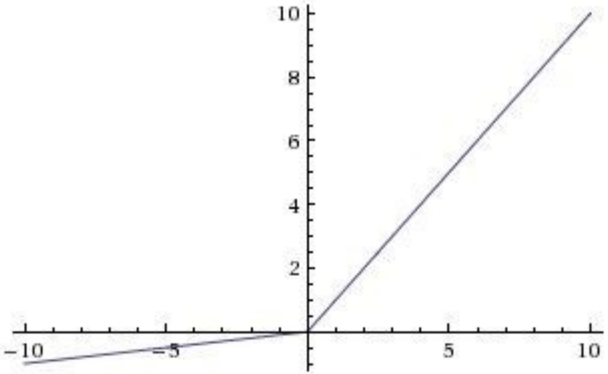
Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

- Not zero-centered output
- ReLU units can "die"

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Activation Functions



Leaky ReLU

$$f(x) = \max(0.01x, x)$$

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not “die”.**

[Mass et al., 2013] [He et al., 2015]

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

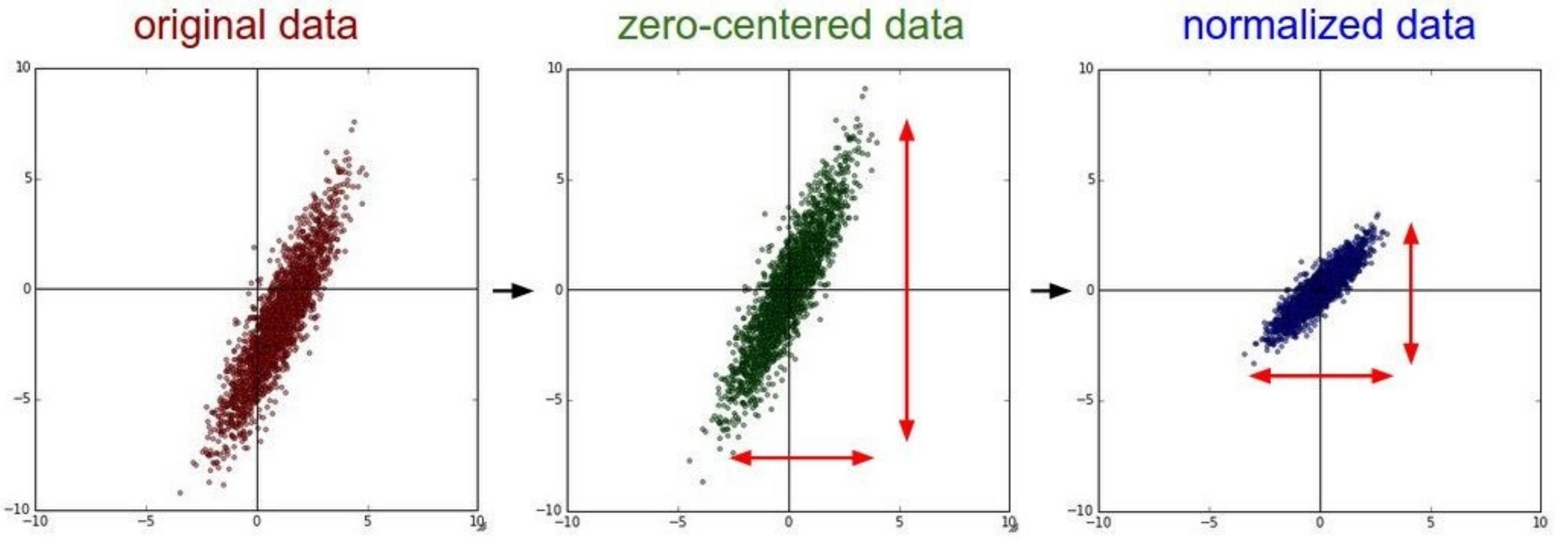
In practice

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU
- Try out tanh but don't expect much
- Don't use sigmoid

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

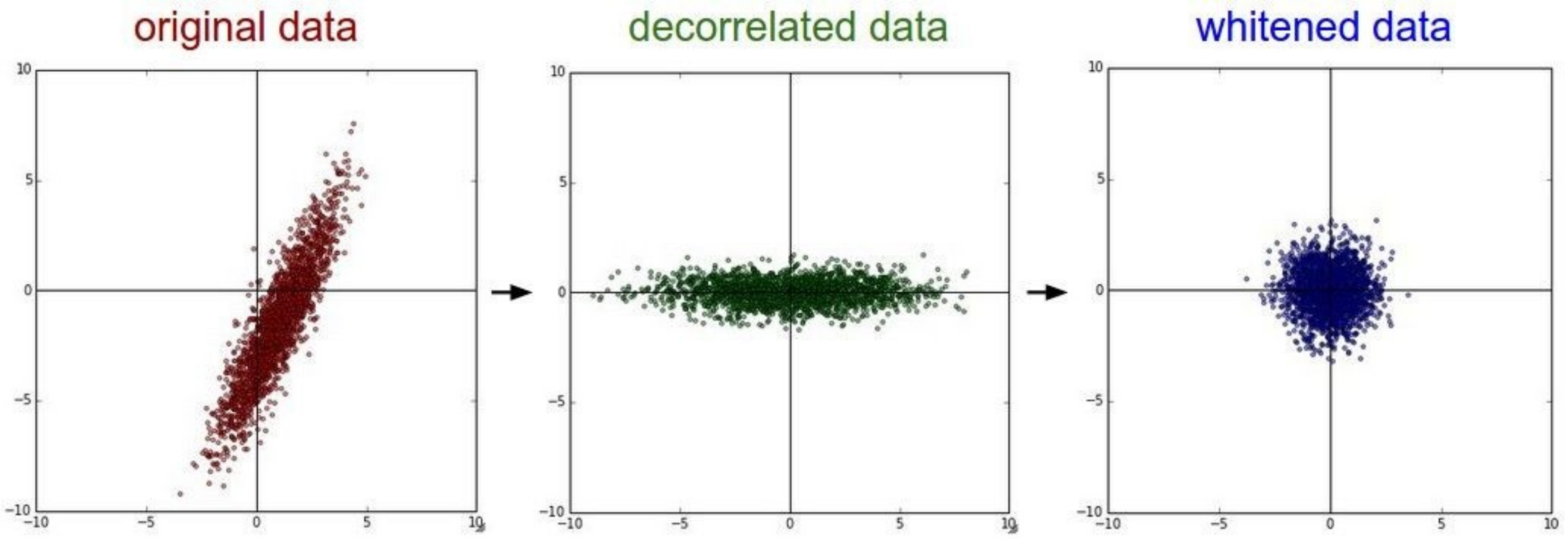
Preprocessing data

Preprocessing data



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Preprocessing data



slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

In practice: for images

e.g. consider CIFAR-10 example with [32,32,3] images

- Subtract the mean image (e.g. AlexNet)
  (mean image = [32,32,3] array)
- Subtract per-channel mean (e.g. VGGNet)
  (mean along each channel = 3 numbers)

Not common to normalize variance, to do PCA or whitening

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Weights initialization

Weights initialization

- If the weights in a network start too small,
  then the signal shrinks as it passes through each layer until it's too tiny to be useful.
- If the weights in a network start too large,
  then the signal grows as it passes through each layer until it's too massive to be useful.

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Weights initialization

- All zero initialization

- Small random numbers

- Draw weights from a Gaussian distribution
  with standard deviation of sqrt(2/n),
  where n is the number of outputs to the neuron

Batch normalization

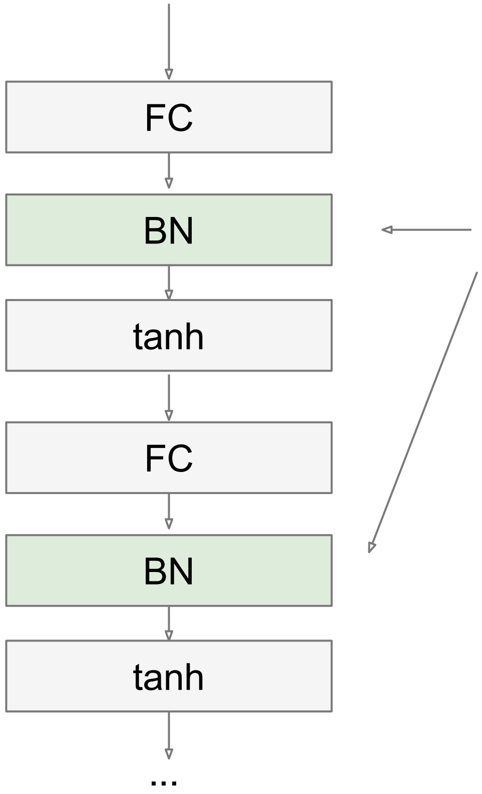
Batch normalization

Initialization of NNs by explicitly forcing the activations throughout the network to take on a unit Gaussian distribution at the beginning of the training.

Normalization is a simple differentiable operation

[Ioffe and Szegedy, 2015]

Batch normalization



Usually inserted after Fully Connected and/or Convolutional layers, and before nonlinearity.

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Batch normalization

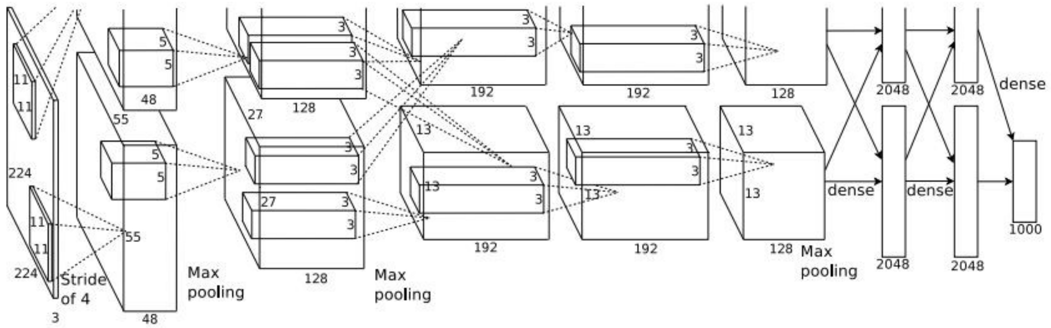
- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization in a funny way, and slightly reduces the need for dropout

Thank you for your attention

AlexNet example

Case Study: AlexNet

[Krizhevsky et al. 2012]



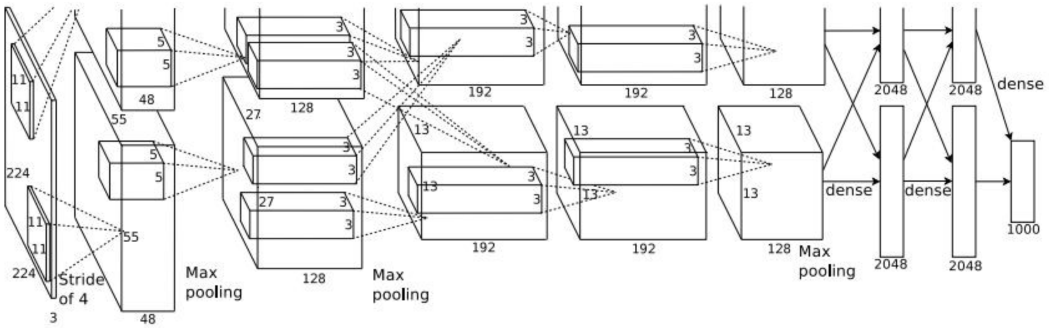
Input: 227x227x3 images

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

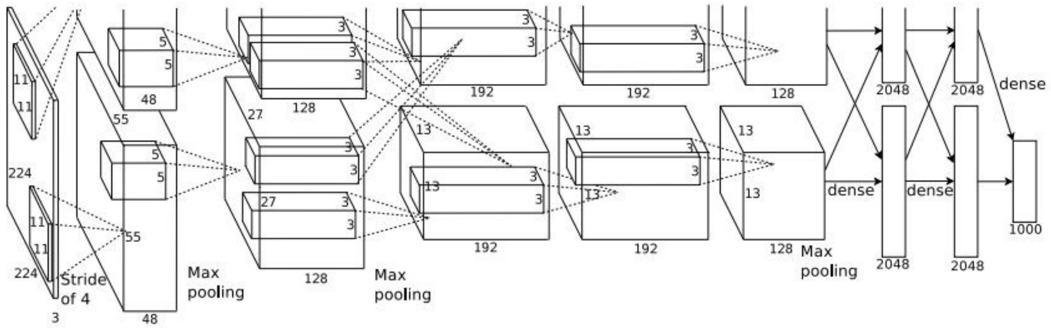
First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

[Krizhevsky et al. 2012]



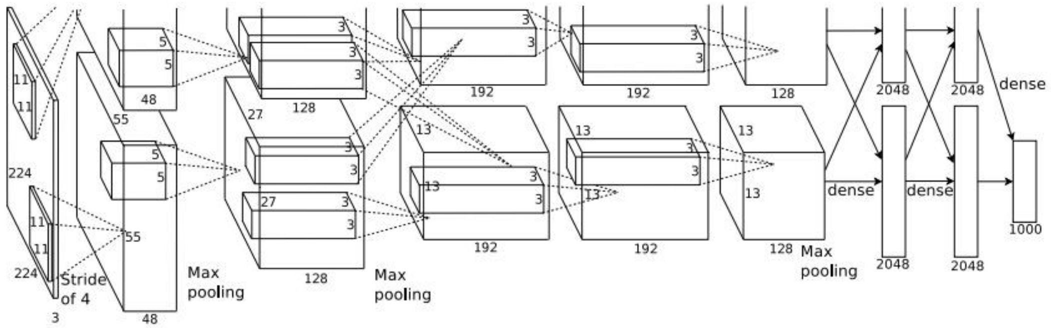
Input: 227x227x3 images

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96

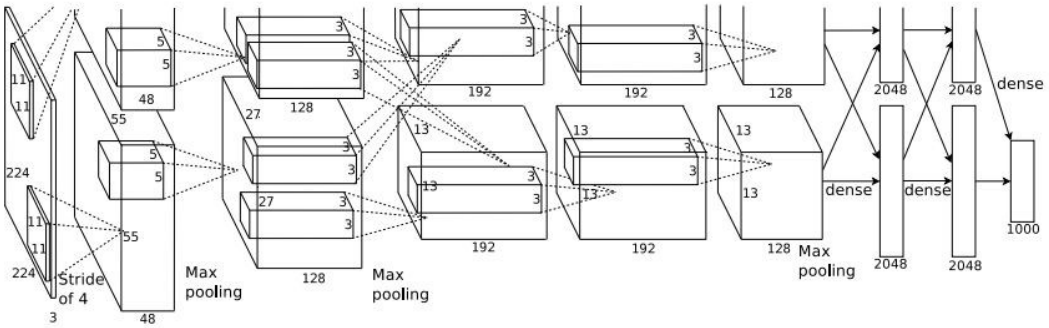
Second layer (POOL1): 3x3 filters applied at stride 2

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96

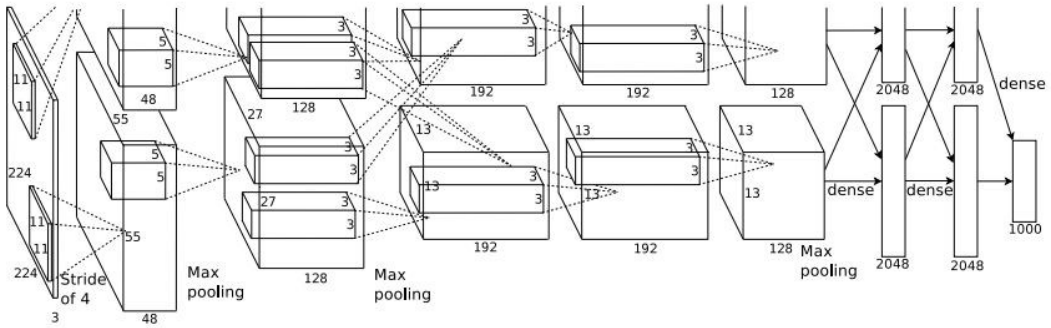
Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

[Krizhevsky et al. 2012]



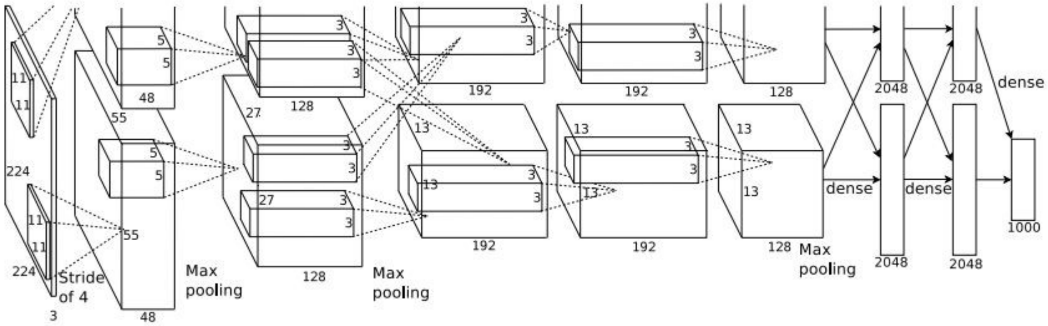
Input: 227x227x3 images
After CONV1: 55x55x96

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

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

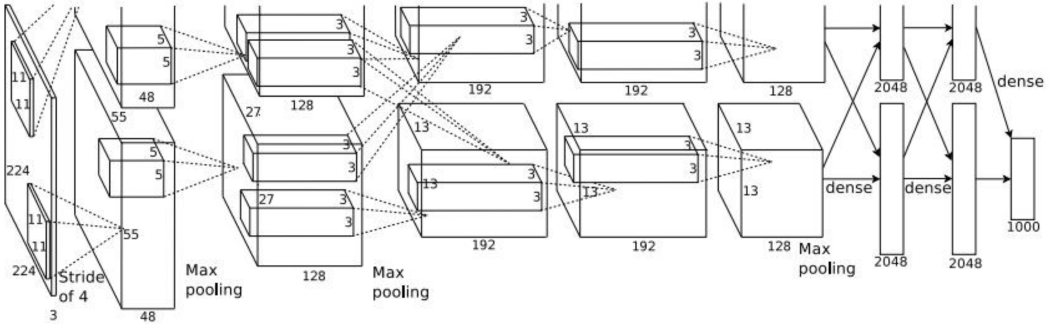
[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96
...

Case Study: AlexNet

[Krizhevsky et al. 2012]

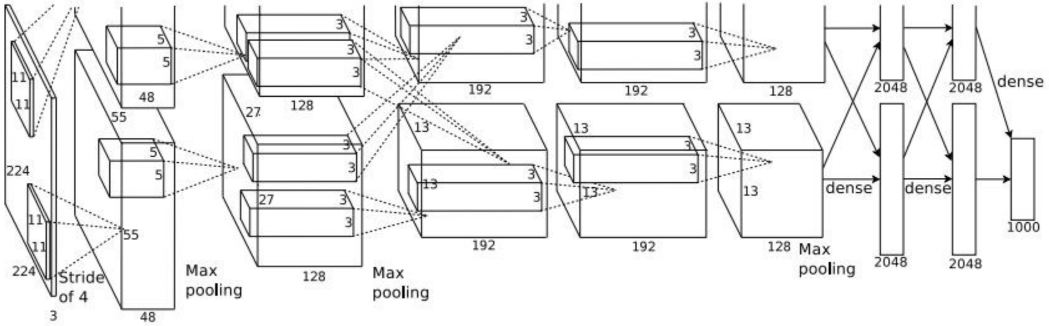


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
-first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
-Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

slide from: Fei-Fei Li & Andrej Karpathy & Justin Johnson