CS 360: Machine Learning

Sara Mathieson, Sorelle Friedler

Spring 2024

HAVERFORD
COLLEGE

Admin

- **Lab 7** check in today
  – Should be finished with the fully connected network
- **Project proposal** due April 8 (short)
- I may receive a call

Outline for April 2

- Finish CNNs
- Neural network regularization
- Backpropagation

Outline for April 2

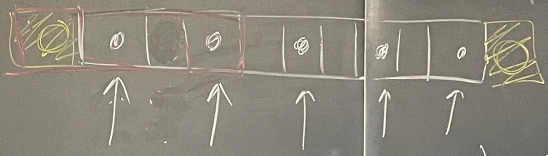
- Finish CNNs
- Neural network regularization
- Backpropagation

CNN filters

- W = input width
- F = filter size (almost always odd)
- P = padding on one side
- S = stride — how much to shift the filter each pass

typical $\Rightarrow S=1$

$W=9$ $\quad$ $P=1$



$F=3$

output size?

$\cancel{W-F+1}$

$9-3+1=7$ without padding

$$\boxed{\underbrace{W-F+2P}_{\text{new input shape}}+1}$$

$9-3+2\cdot 1+1=\boxed{9}$ with padding

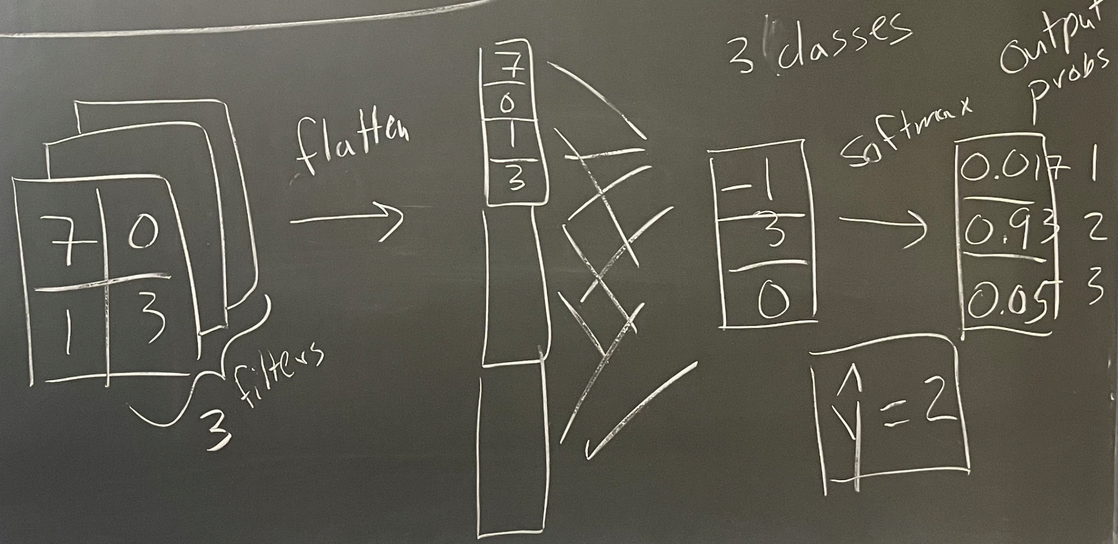
$$\boxed{\frac{W-F+2P}{S}+1}$$

formula for output shape

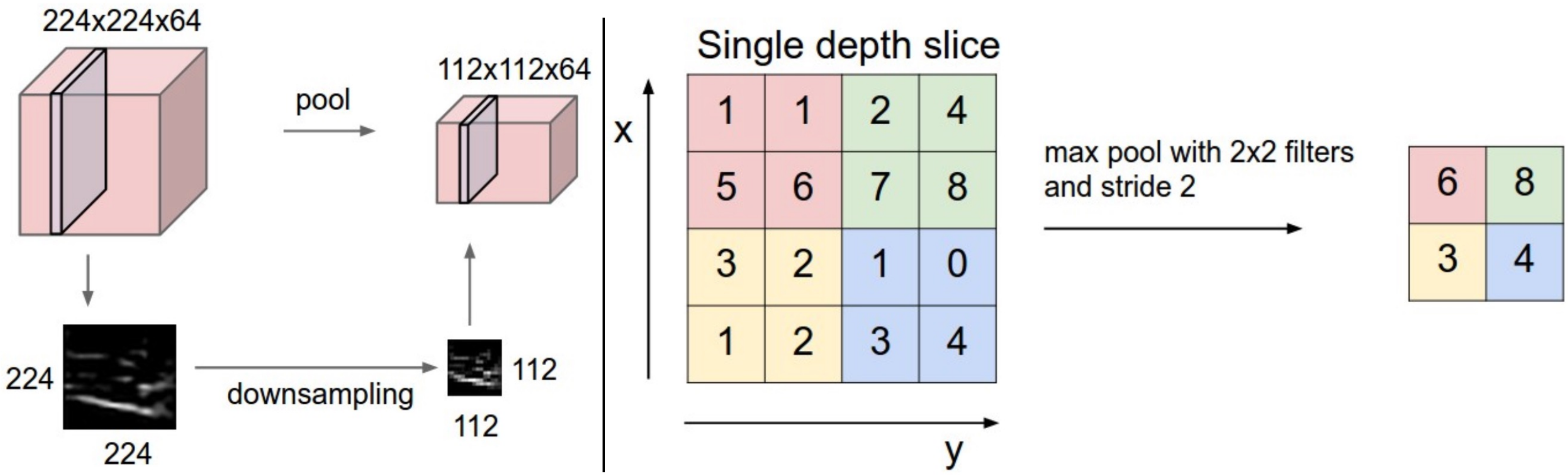
$S = 2$

$$\frac{9 - 3 + 2\cdot 1}{2} + 1 \Rightarrow \boxed{5}$$ output shape

FC layers after conv layers



Pooling



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left**: In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right**: The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Image: Stanford Course CS231n: http://cs231n.github.io/convolutional-networks/

Handout 18

(a) Which steps require parameter learning? (out of CONV, RELU, POOL, FLATTEN, FC)

(b) First layer params

(c) Second layer params

(d) Third layer params

(e) Total # params

Handout 18

(a) Which steps require parameter learning? (out of CONV, RELU, POOL, FLATTEN, FC)

CONV, FC

(b) First layer params

(c) Second layer params

(d) Third layer params

(e) Total # params

Handout 18

(a) Which steps require parameter learning? (out of CONV, RELU, POOL, FLATTEN, FC)

CONV, FC

(b) First layer params 5*5*3*20 + 20 = 1520

(c) Second layer params 3*3*20*10 + 10 = 1810

(d) Third layer params 8*8*10*10 + 10 = 6410

(e) Total # params 9740

Handout 18

(a) Which steps require parameter learning? (out of CONV, RELU, POOL, FLATTEN, FC)

CONV, FC

(b) First layer params

5*5*3*20 + 20 = 1520

(c) Second layer params

3*3*20*10 + 10 = 1810

(d) Third layer params

8*8*10*10 + 10 = 6410

(e) Total # params

9740

If we had a FC with p_1=100 and p_2=50, we would have 312,860 params to learn (check this after class). CNN is much better!

Handout 19, #1

(a) W=10, F=7, P=3, S=3

(b) Draw padding

(c) Shade units where cross-correlation is performed

Handout 19, #1

(a) W=10, F=7, P=3, S=3

$(10-7+6)/3 + 1 = 4$ (output size)

(b) Draw padding

(c) Shade units where cross-correlation is performed

Handout 19, #1

(a) W=10, F=7, P=3, S=3

(10-7+6)/3 + 1 = 4 (output size)

(b) Draw padding

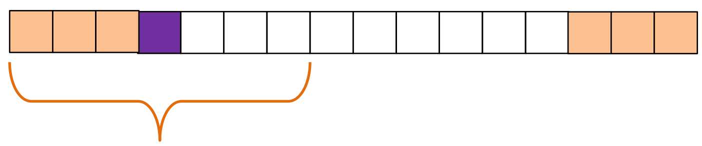
(c) Shade units where cross-correlation is performed

Handout 19, #1

(a) W=10, F=7, P=3, S=3

(10-7+6)/3 + 1 = 4 (output size)

(b) Draw padding



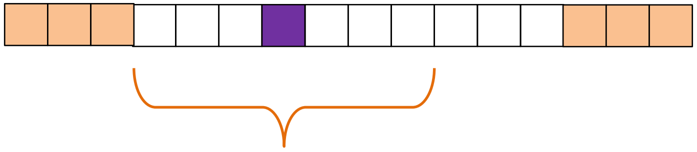
(c) Shade units where cross-correlation is performed

Handout 19, #1

(a) W=10, F=7, P=3, S=3

(10-7+6)/3 + 1 = 4 (output size)

(b) Draw padding



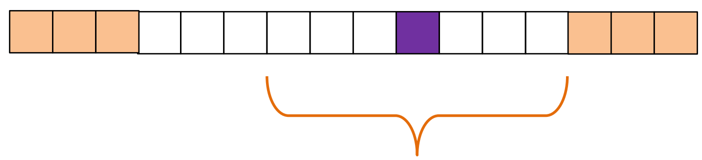
(c) Shade units where cross-correlation is performed

Handout 19, #1

(a) W=10, F=7, P=3, S=3

(10-7+6)/3 + 1 = 4 (output size)

(b) Draw padding



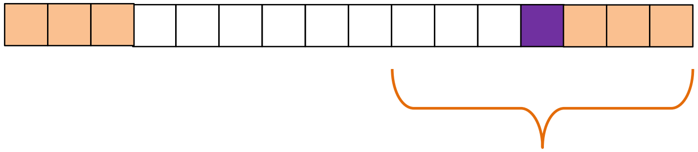
(c) Shade units where cross-correlation is performed

Handout 19, #1

(a) W=10, F=7, P=3, S=3

(10-7+6)/3 + 1 = 4 (output size)

(b) Draw padding



(c) Shade units where cross-correlation is performed

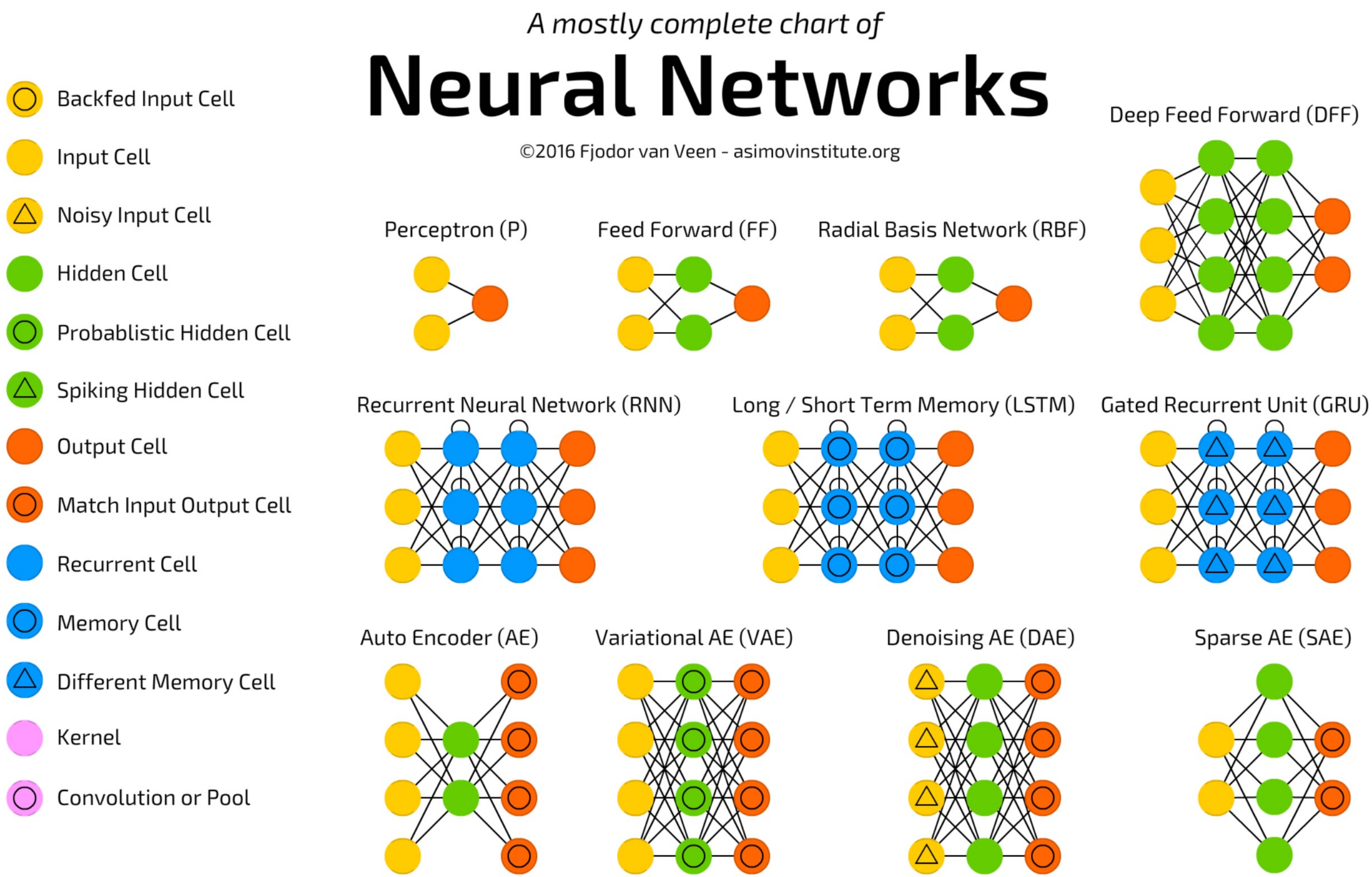


Image from Neural Network Zoo: http://www.asimovinstitute.org/neural-network-zoo/

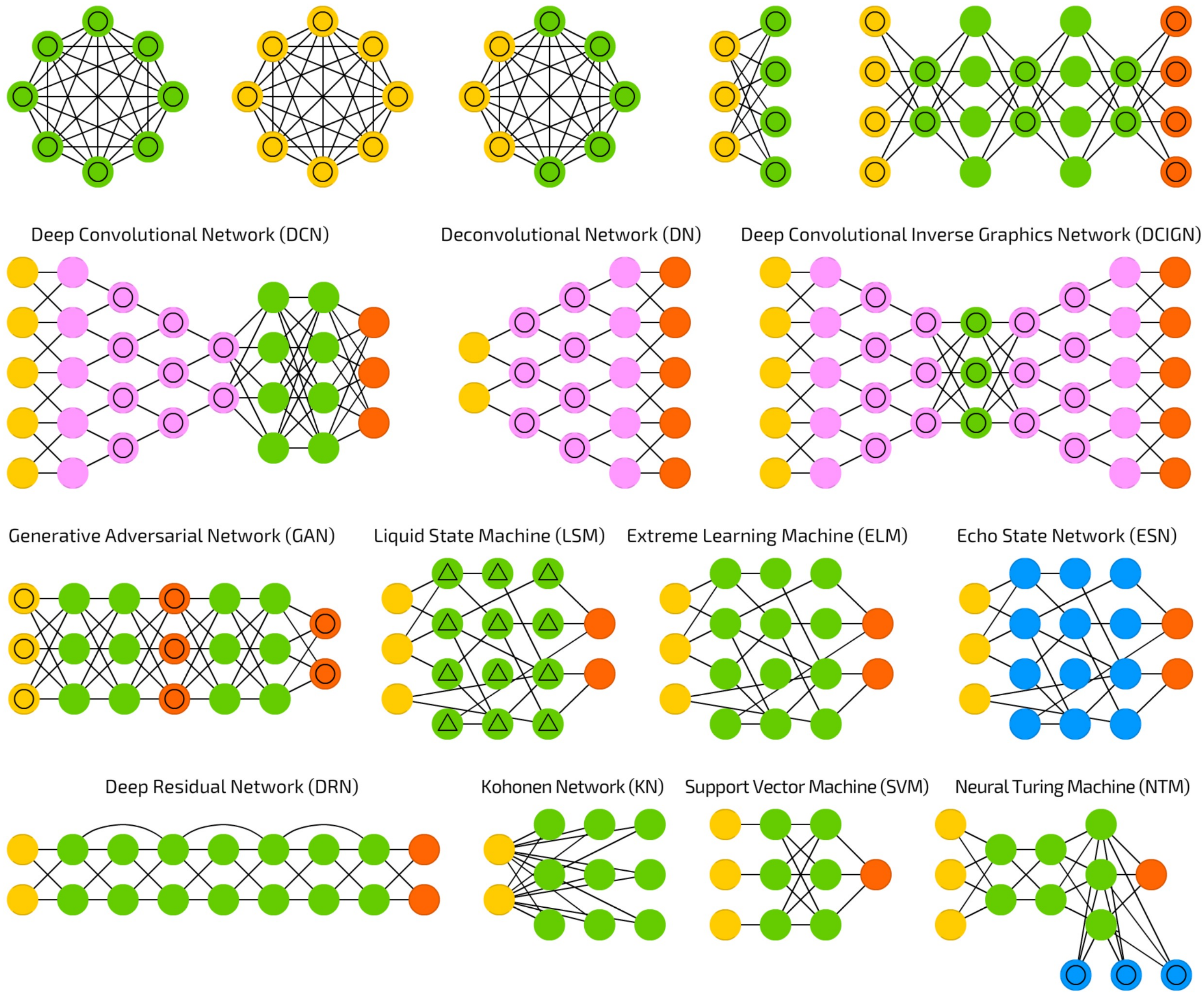


Image from Neural Network Zoo: http://www.asimovinstitute.org/neural-network-zoo/

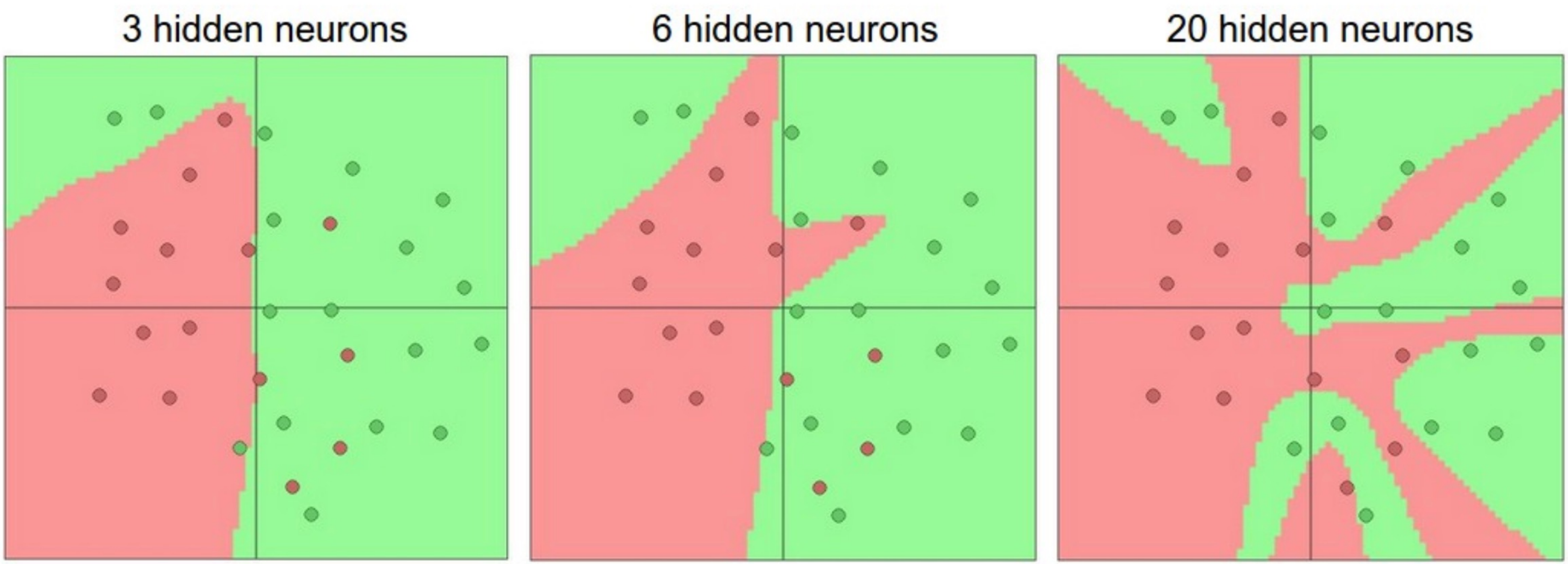
Outline for April 2

- Finish CNNs
- Neural network regularization
- Backpropagation

Weight initialization

- We still have to initialize the pre-training
- All 0's initialization is bad! Causes nodes to compute the same outputs, so then the weights go through the same updates during gradient descent
- Need asymmetry! => usually use small random values

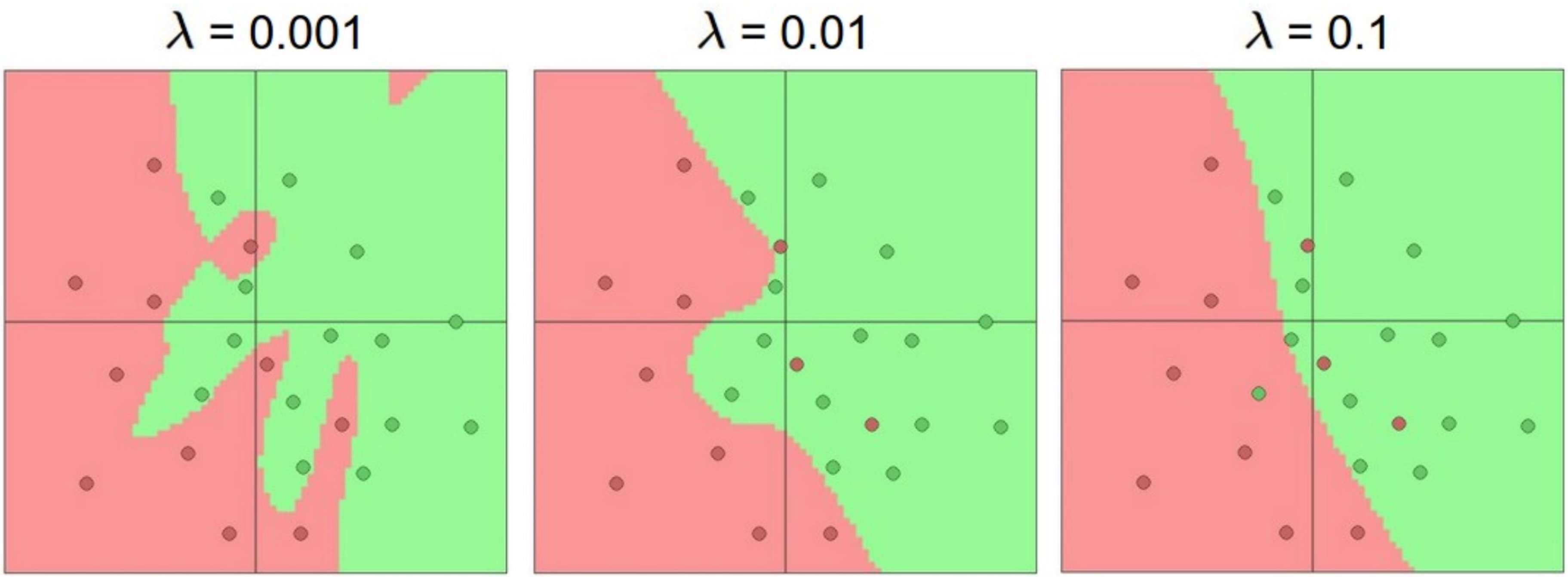
More hidden units can contribute to overfitting



Larger Neural Networks can represent more complicated functions. The data are shown as circles colored by their class, and the decision regions by a trained neural network are shown underneath. You can play with these examples in this ConvNetsJS demo.

Image from: http://cs231n.github.io/neural-networks-1/

However! It is always better to use a more expressive network and regularize in other ways

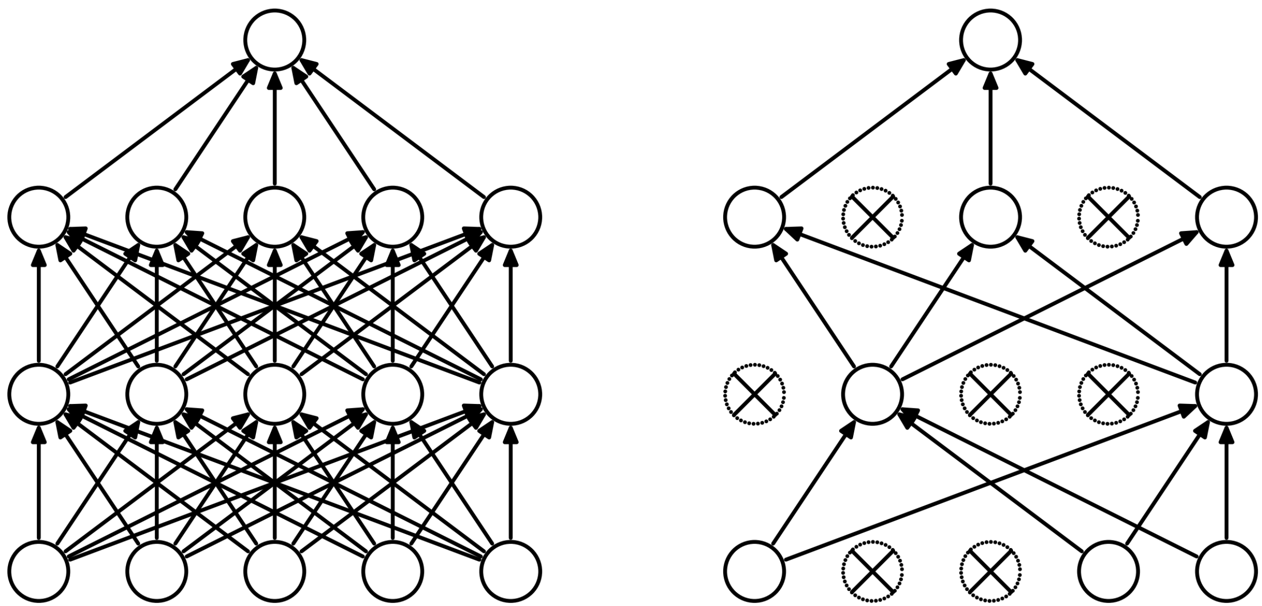


The effects of regularization strength: Each neural network above has 20 hidden neurons, but changing the regularization strength makes its final decision regions smoother with a higher regularization. You can play with these examples in this ConvNetsJS demo.

Image from: http://cs231n.github.io/neural-networks-1/

One regularization approach: dropout

- Idea: keep a neuron active with some probability p, otherwise, do not send its output forward to the next layer



(a) Standard Neural Net (b) After applying dropout.

Image and more information: "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"
http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Outline for April 2

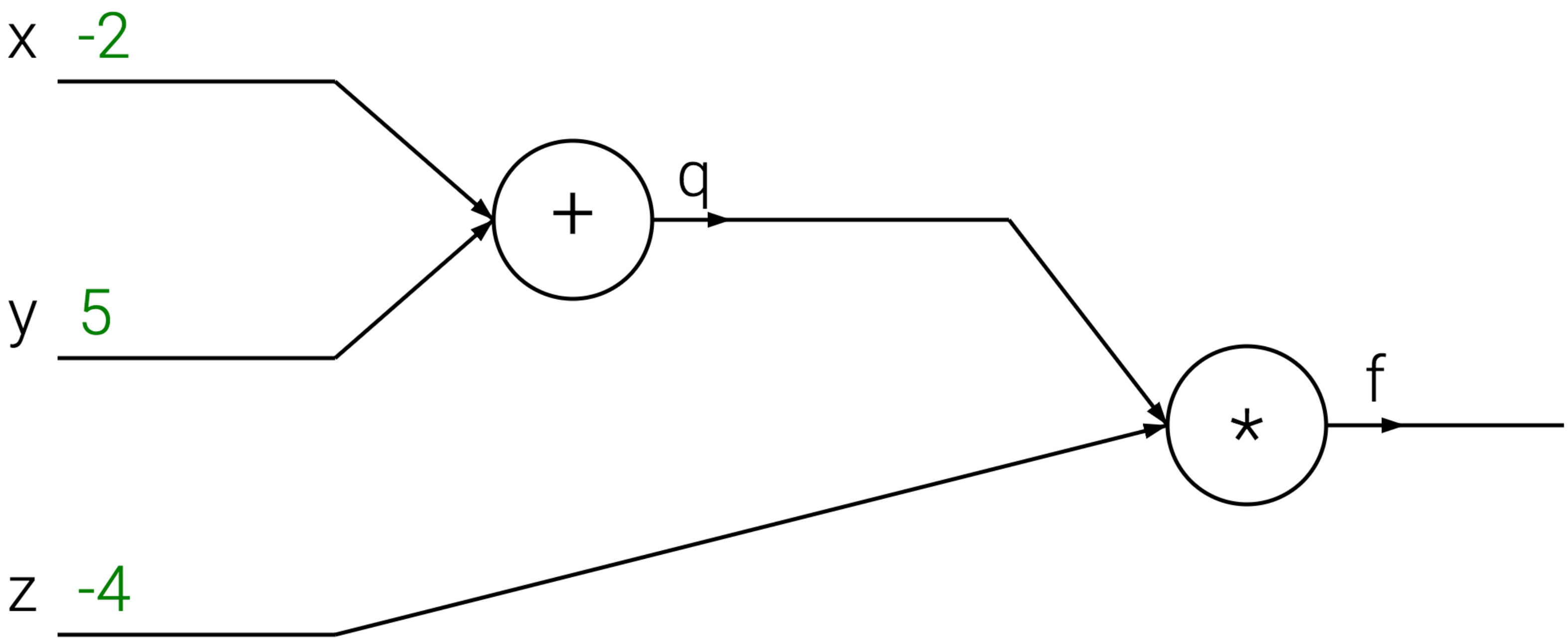
- Finish CNNs
- Neural network regularization
- Backpropagation

Backpropagation

- *High-level goal:* we want to know how the output depends on the input
- *Issue:* network is very complicated and overall gradient may be difficult to compute
- *Idea:* use the chain rule to compute local gradients throughout the network
- *Takeaway:* nodes can know about their value and local gradient without knowing about the network they are imbedded in

Backpropagation: Example

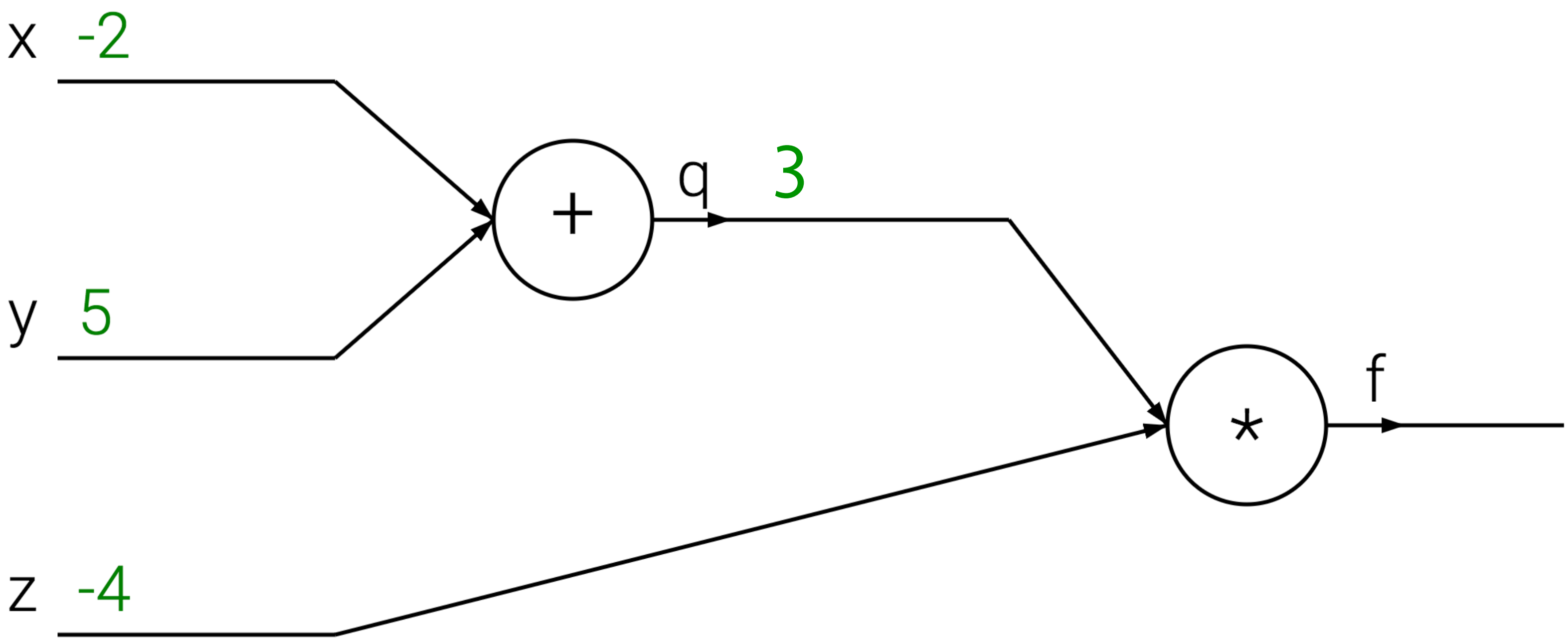
Forward pass: compute values



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

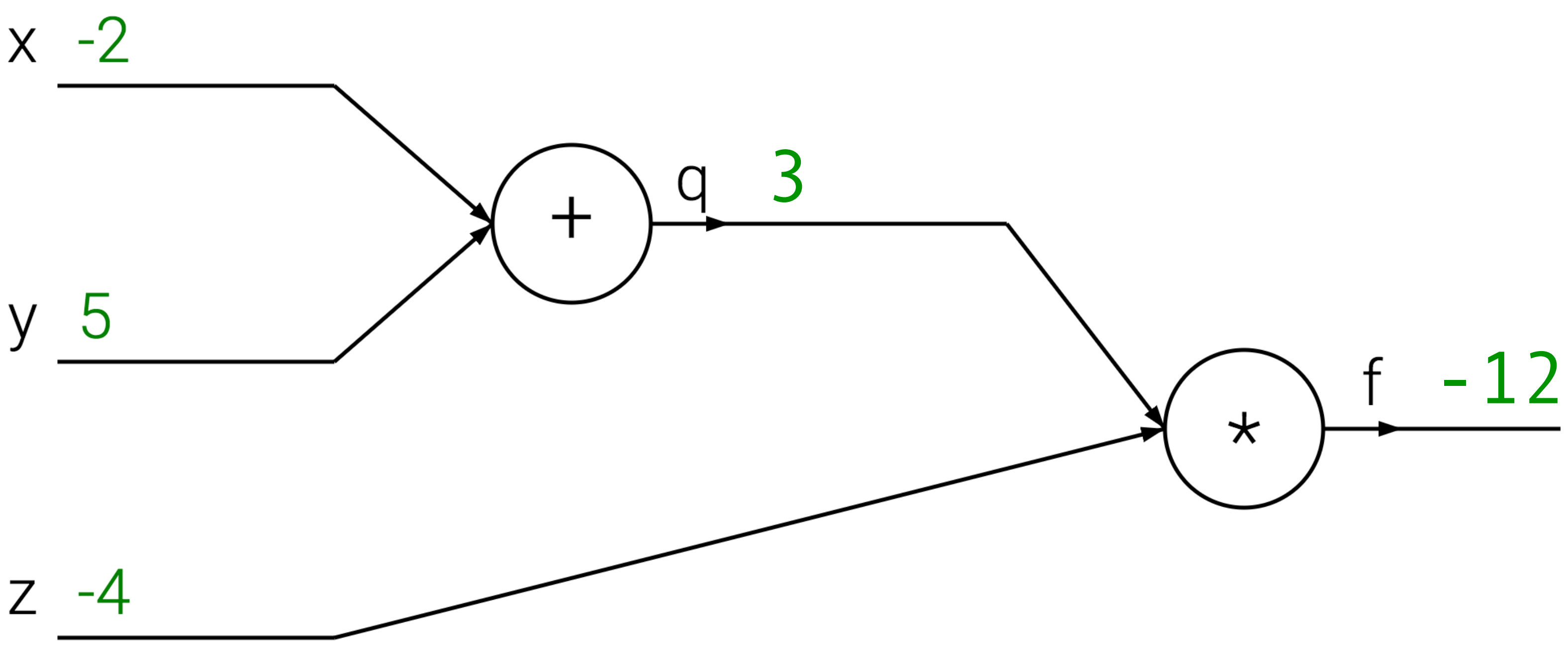
Forward pass: compute values



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

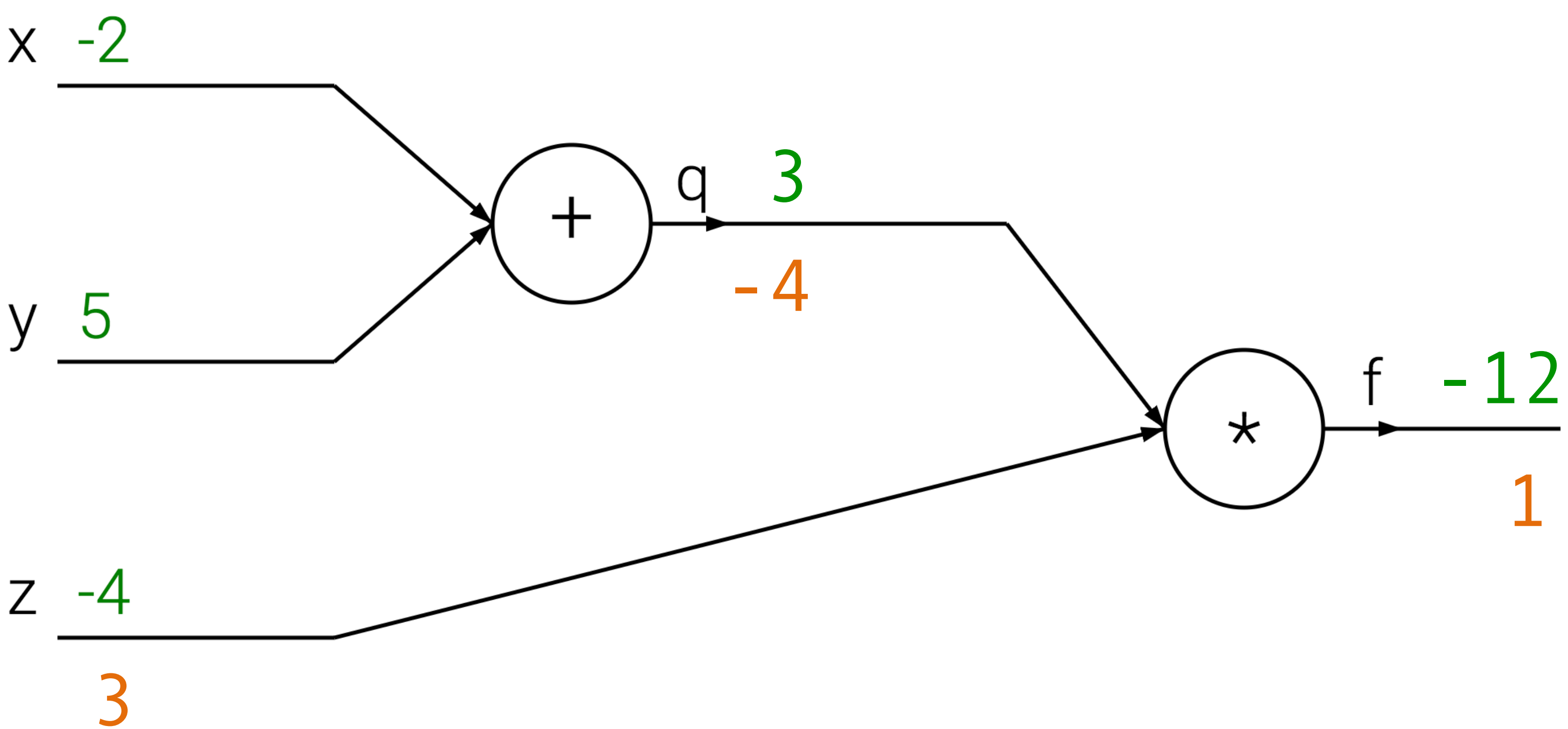
Forward pass: compute values



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

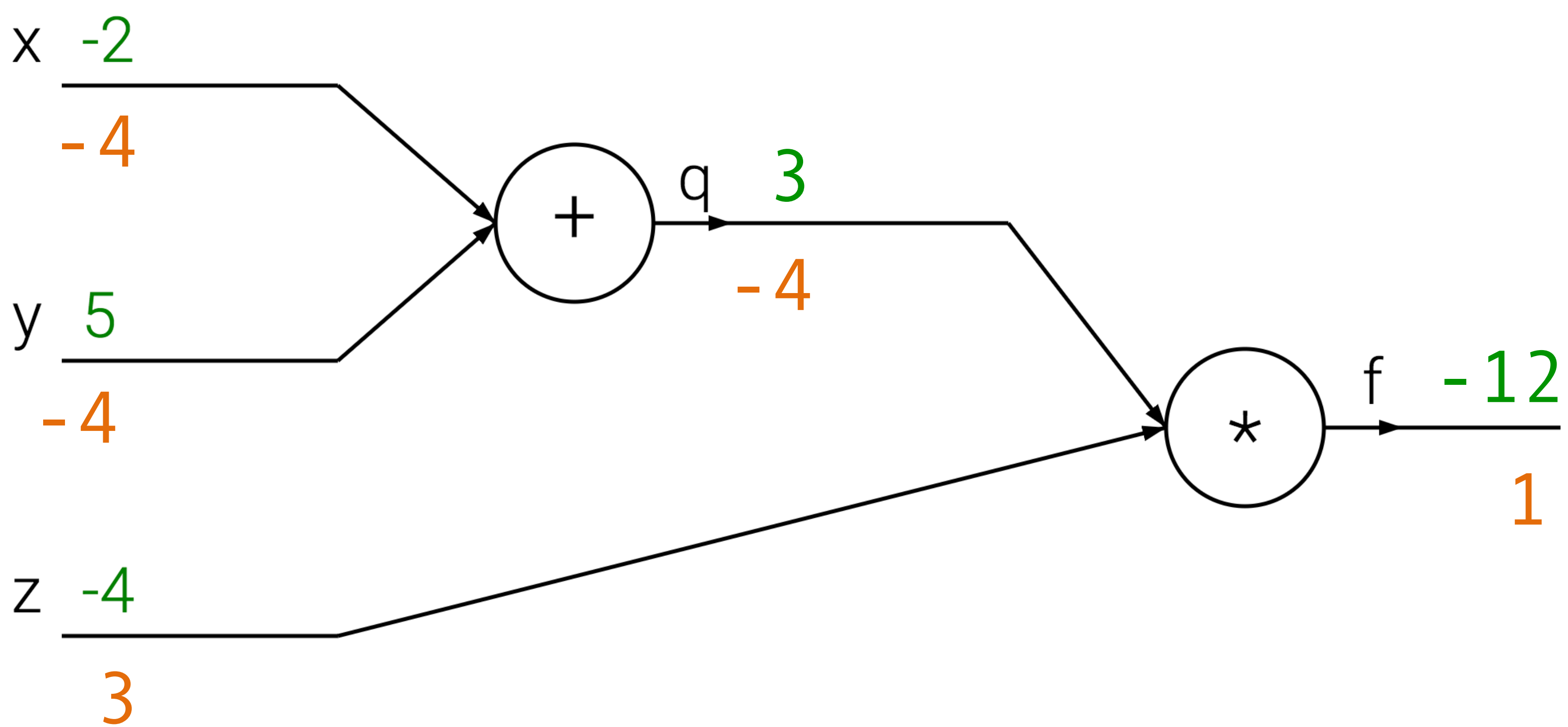
Backward pass: compute local gradients



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

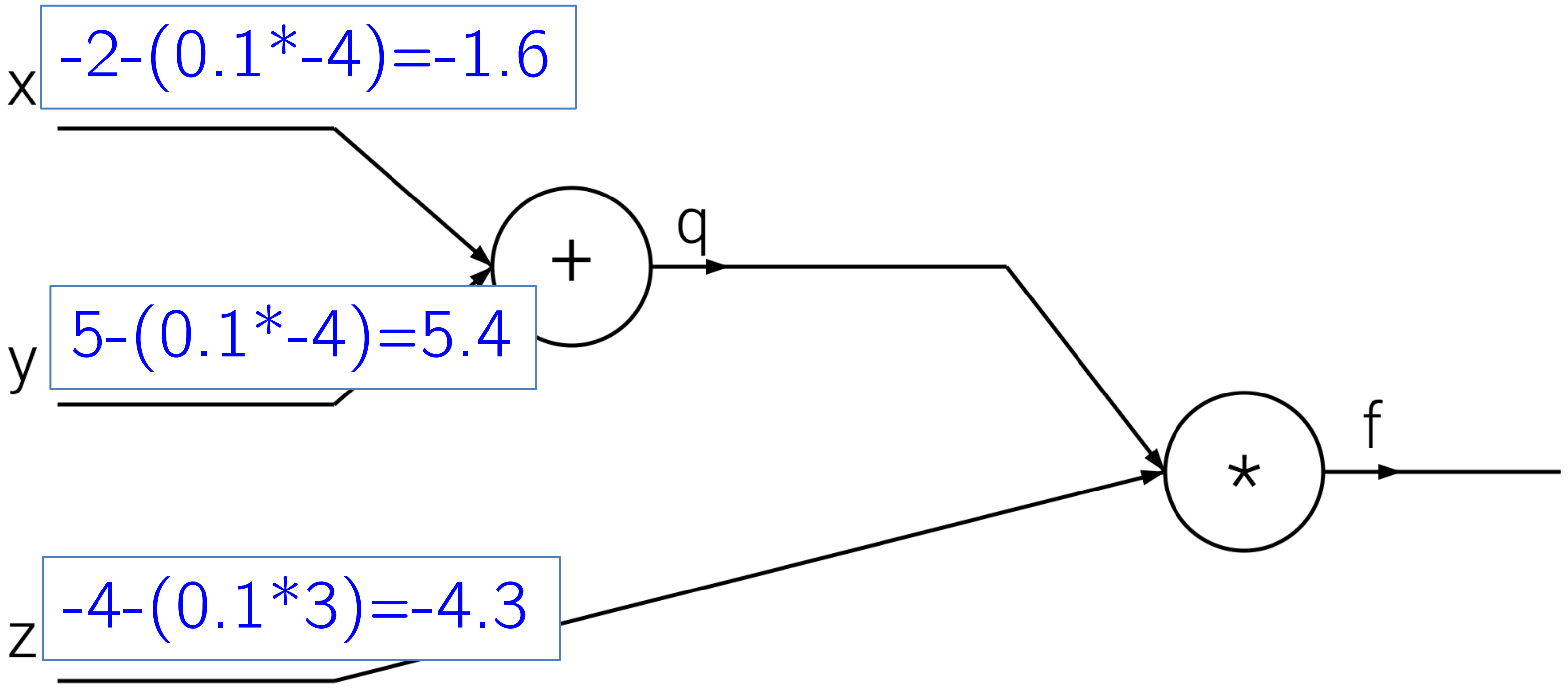
Backward pass: compute local gradients



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

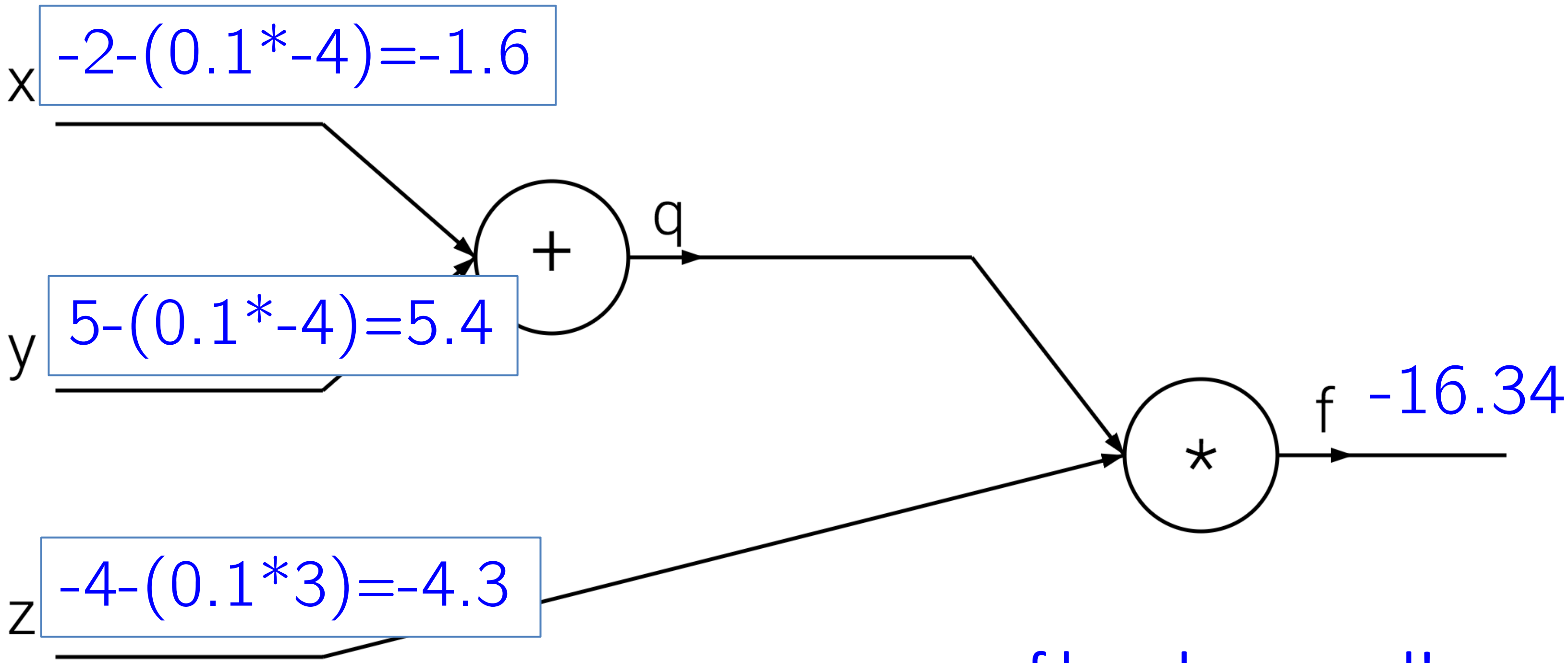
Now if we wanted to minimize f => opposite direction of gradient



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

Now if we wanted to minimize f => opposite direction of gradient



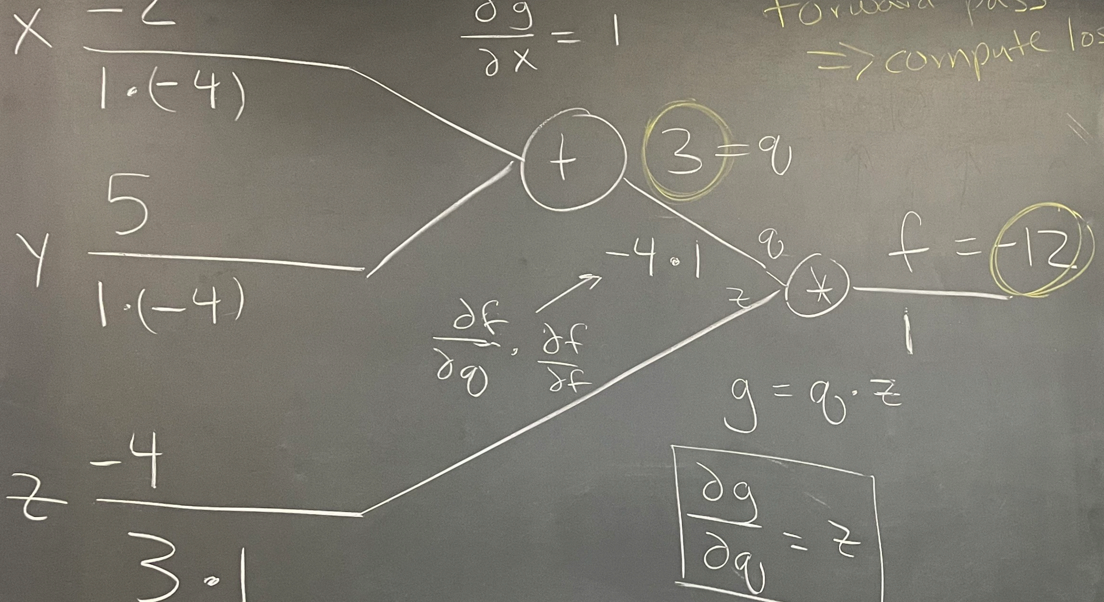
f has decreased!

Example from:
http://cs231n.github.io/optimization-2/

$g = \cancel{x} + y$

$\frac{\partial g}{\partial x} = 1$

forward pass
$\Rightarrow$ compute loss



$g = q \cdot z$

$\boxed{\frac{\partial g}{\partial q} = z}$

$\frac{\partial g}{dz} = q$

backward pass
below lines

above lines

$f(x,y,z) = \underbrace{(x+y)}_{q} z$

$f = q z$

partial derivative of f with respect to x

$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial x}$ chain rule

$= z \cdot 1$

mini

Minimize f

$\boxed{\text{update trainable variables}}$

$x \leftarrow x - \alpha \frac{\partial f}{\partial x}$

$-2 - 0.1(-4)$

$\boxed{x \leftarrow -1.6}$

$y \leftarrow y - \alpha \frac{\partial f}{\partial y}$

$y \leftarrow 5 - 0.1(-4)$

$\boxed{y \leftarrow 5.4}$

$z \leftarrow -4 - 0.1(3)$

$\boxed{z \leftarrow -4.3}$

$f(x,y,z) = (-1.6 + 5.4)(-4.3)$

$= \boxed{-16.34}$

lower than -12

☆

$\boxed{\#4 \text{ and } \#5}$

(#2 on your own)

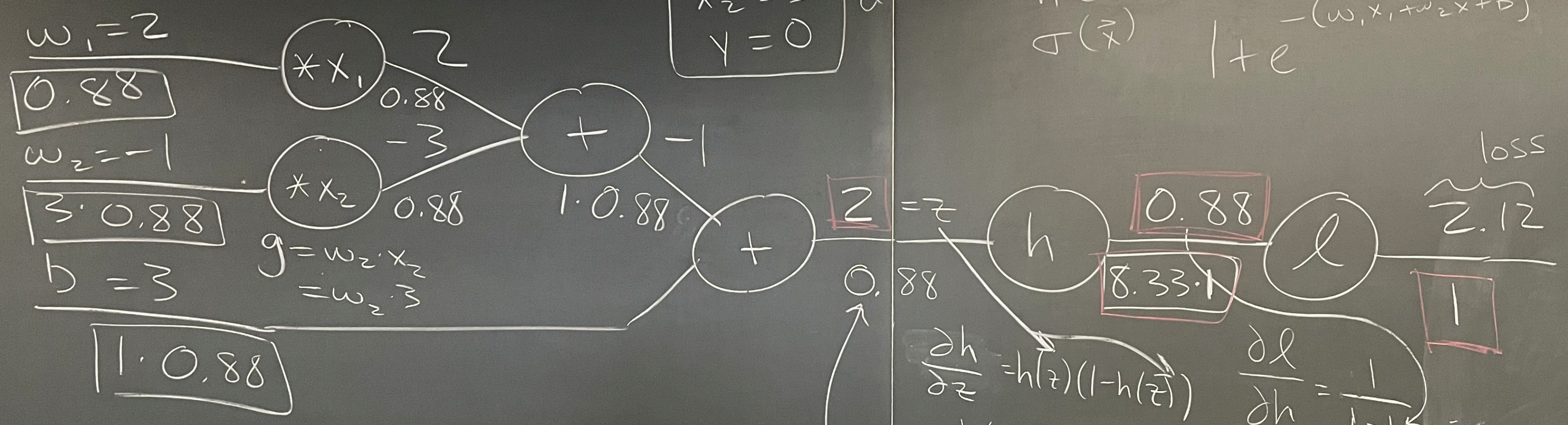
③ logistic regression

$x_1 = 1$
$x_2 = 3$
$y = 0$

data

$$h(\vec{x}) = \frac{1}{1+e^{-(w_1x_1+w_2x+b)}} = \frac{1}{1+e^{-2}}$$

$$\sigma(\vec{x})$$



$$\frac{\partial h}{\partial z} = h(z)(1-h(z))$$

$$= h(2)(1-h(2))$$

$$h(2)(\cancel{1-h(2)})\cdot \cancel{8.33}$$

$$\frac{\partial l}{\partial h} = \frac{1}{1-h(2)}$$

$$\frac{\partial l}{\partial h} = \frac{1}{1-0.88}$$

$$\ell(h) = \cancel{-y \log h} - (1-y)\log(1-h)$$

$$\ell_0(h) = -\log(1-h)$$

$$= \boxed{\frac{1}{1-h}}$$

4

$$\underset{-\frac{3}{4}}{\overset{0}{\longrightarrow}} \; \sigma \; \underset{-3}{\overset{\frac{1}{2}}{\longrightarrow}}$$

$\sigma(0) = \dfrac{1}{1+e^{-0}} = \dfrac{1}{2}$

$$\sigma'(z) = \sigma(z)\left(1 - \sigma(z)\right)$$
$$= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

5

$$\underset{0 \cdot -3}{\overset{-1}{\longrightarrow}} \; \text{ReLU} \; \underset{-3}{\overset{0}{\longrightarrow}}$$

$$= \boxed{0}$$