

FACULTY OF MATHEMATICS,  
PHYSICS AND INFORMATICS  
Comenius University  
Bratislava

Neural Networks for Computer Vision

# Lecture 4: Fully Connected NNs and Backpropagation

Ing. Viktor Kocur, PhD., doc. RNDr. Zuzana Černeková, PhD.

5.10.2023

# Contents



- Neural networks
- Activation functions
- Forward pass of NNs
- Biological inspiration
- Calculating gradients in NNs
- Backpropagation
- Backpropagation - vectorized

# Acknowledgment



The majority of slides are directly adopted from slides for CS231n<sup>1</sup> course at Stanford University!

<sup>1</sup>Fei-Fei Li, Ranjay Krishna, and Danfei Xu. *Stanford CS231n lecture slides*. <http://cs231n.stanford.edu/slides/>

# Recap - I



So far we have:

- A linear function:  $s = f(x, W) = W\mathbf{x}$
- Hinge loss:  $L_i = \sum_{j \neq i} \max(0, s_j - s_{y_i} + 1)$
- With regularization  $L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda \sum_{i,j} W_{i,j}^2$

How do we find the best  $W$ ?

## Recap - II



Finding the best  $W$  is an optimization task! We can use gradient descent!

$$W_{i,j}^{n+1} = W_{i,j}^n - \eta \frac{\partial L}{\partial w_i} \quad (1)$$

## Recap - III



Partial derivative and gradient definition:

$$\frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i, \dots, x_n)}{h} \quad (2)$$

$$\text{grad}(f) = \nabla f = \frac{\partial f}{\partial \mathbf{x}} = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right) \quad (3)$$

## Recap - IV



- Numerical gradient -  $\frac{\partial f}{\partial x_i} \approx \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i - h, \dots, x_n)}{2h}$ 
  - ▶ Slow
  - ▶ Approximate
  - ▶ Easy to compute once  $f$  is implemented



■ Numerical gradient -  $\frac{\partial f}{\partial x_i} \approx \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i - h, \dots, x_n)}{2h}$

- ▶ Slow
- ▶ Approximate
- ▶ Easy to compute once  $f$  is implemented

■ Analytical gradient

- ▶ Fast
- ▶ Accurate
- ▶ Requires more involved implementation



■ Numerical gradient -  $\frac{\partial f}{\partial x_i} \approx \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i - h, \dots, x_n)}{2h}$

- ▶ Slow
- ▶ Approximate
- ▶ Easy to compute once  $f$  is implemented

■ Analytical gradient

- ▶ Fast
- ▶ Accurate
- ▶ Requires more involved implementation

In practice we derive analytical gradient, but we check our implementation by using numerical approximations!

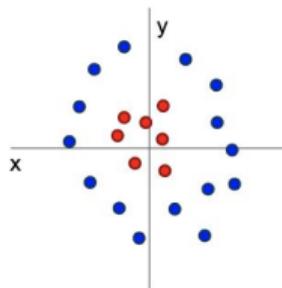


Problem: Linear Classifiers are not very powerful

Visual Viewpoint



Geometric Viewpoint



Linear classifiers learn  
one template per class

Linear classifiers  
can only draw linear  
decision boundaries

# Pixel features



$$f(x) = Wx$$

Class  
scores



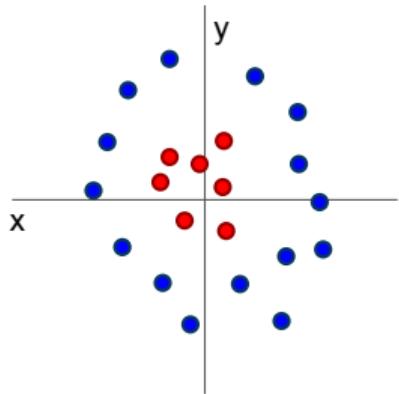
# Image features



Feature Representation

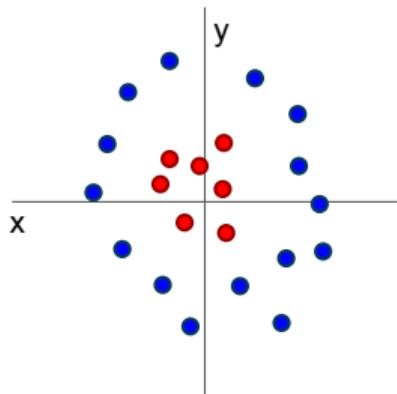
Class scores

# Image features - motivation



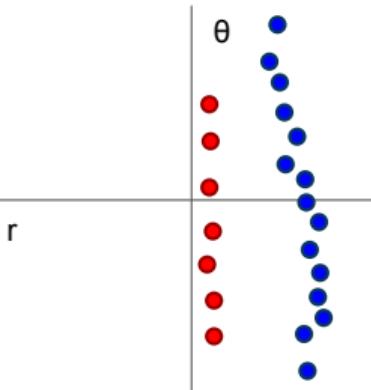
Cannot separate red  
and blue points with  
linear classifier

# Image features - motivation



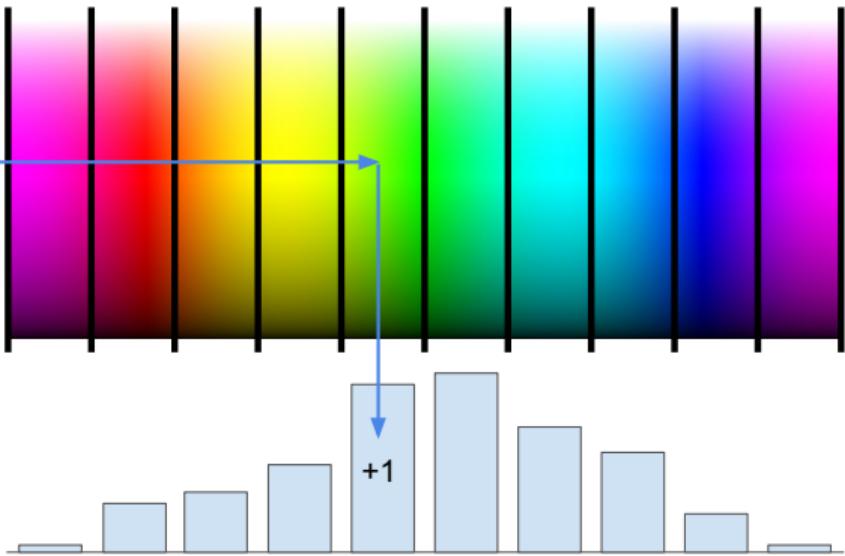
Cannot separate red  
and blue points with  
linear classifier

$$f(x, y) = (r(x, y), \theta(x, y))$$



After applying feature  
transform, points can  
be separated by linear  
classifier

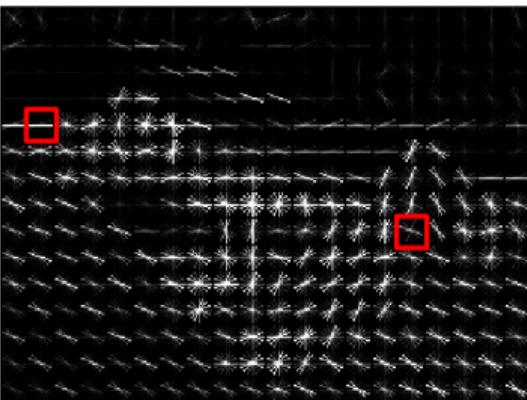
# Image features - color histogram



# Image features - SIFT, HOG



Divide image into 8x8 pixel regions  
Within each region quantize edge  
direction into 9 bins



Example: 320x240 image gets divided  
into 40x30 bins; in each bin there are  
9 numbers so feature vector has  
 $30 \times 40 \times 9 = 10,800$  numbers

Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection." In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893

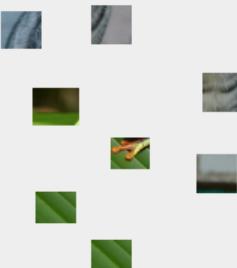
# Image features - bag of words



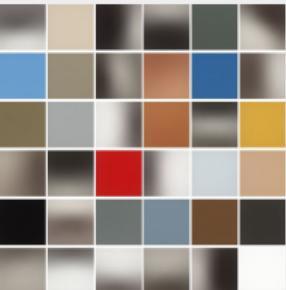
## Step 1: Build codebook



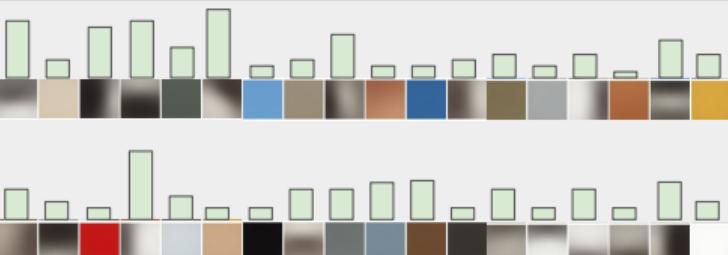
Extract random patches



Cluster patches to  
form "codebook"  
of "visual words"

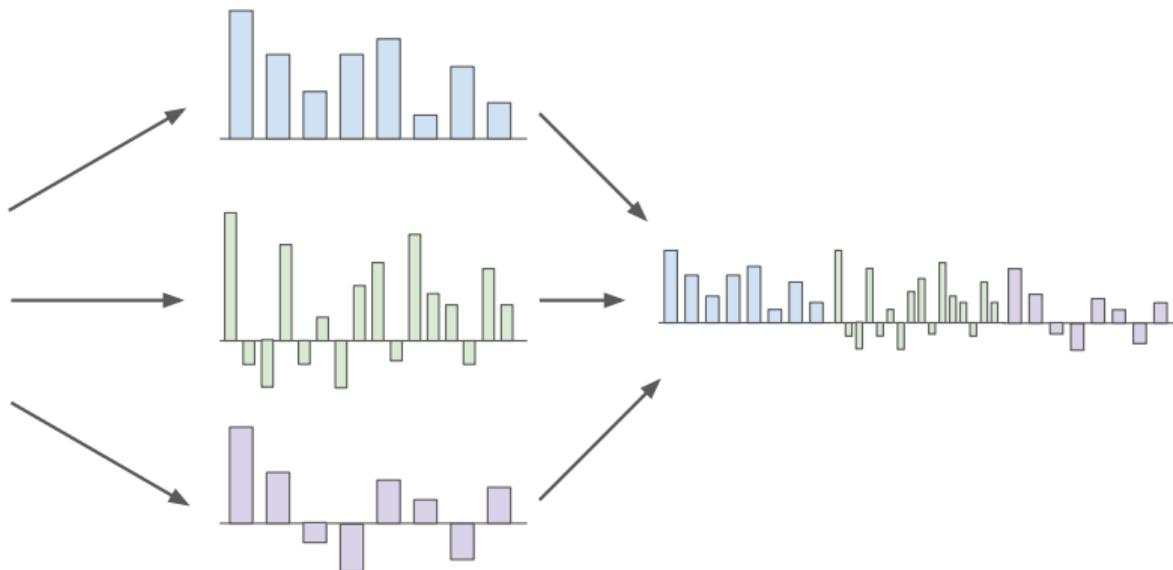


## Step 2: Encode images

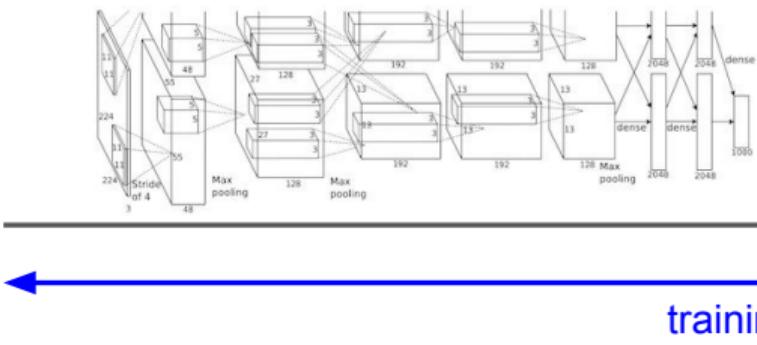
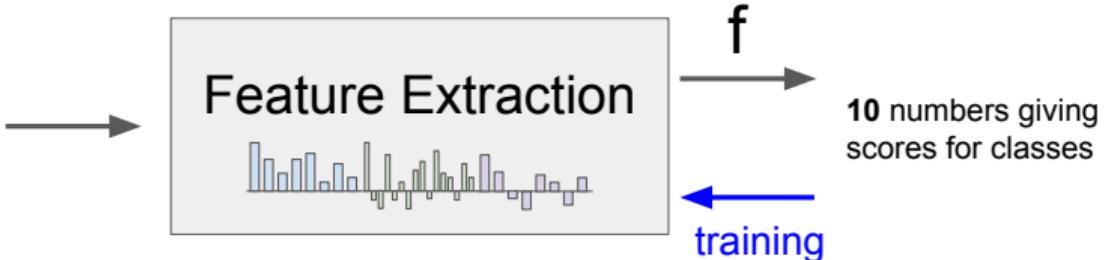


Fel-Fei and Perona, "A bayesian hierarchical model for learning natural scene categories", CVPR 2005

# Image features



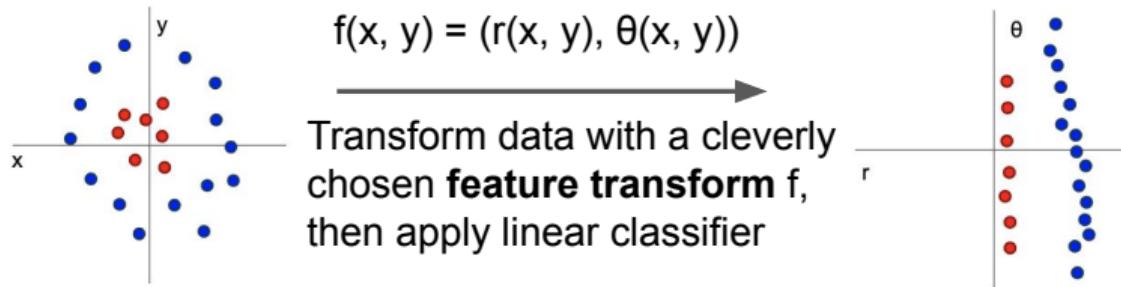
# Engineered features vs CNNs



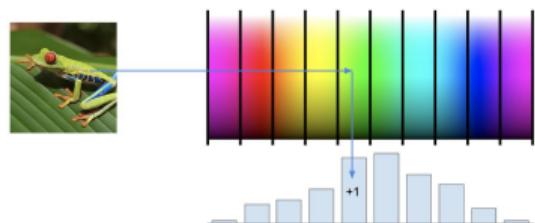
Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105



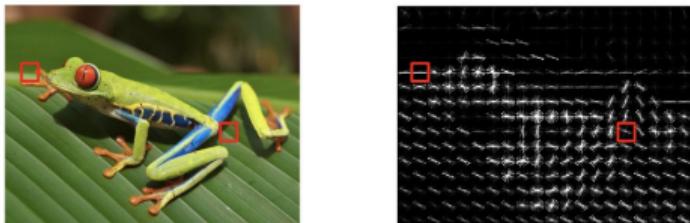
# Feature transformation



Color Histogram



Histogram of Oriented Gradients (HoG)



Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105

# Linear classifier vs NN



## ■ Linear classifier $f(x) = Wx$

- ▶  $x \in \mathbb{R}^D$
- ▶  $W \in \mathbb{R}^{C \times D}$

# Linear classifier vs NN



- Linear classifier  $f(x) = Wx$ 
  - ▶  $x \in \mathbb{R}^D$
  - ▶  $W \in \mathbb{R}^{C \times D}$
- 2-layer NN:  $f(x) = W_2 \max(0, W_1 x)$ 
  - ▶  $x \in \mathbb{R}^D$
  - ▶  $W_1 \in \mathbb{R}^{H \times D}$
  - ▶  $W_2 \in \mathbb{R}^{C \times H}$

# Linear classifier vs NN



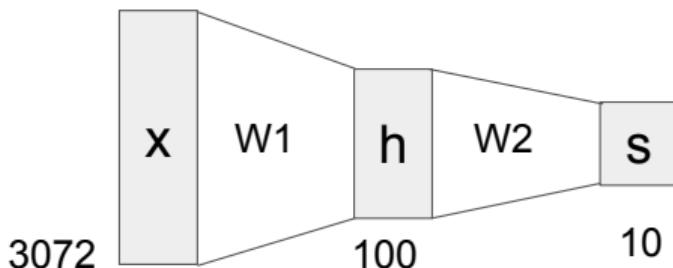
- Linear classifier  $f(x) = Wx$ 
  - ▶  $x \in \mathbb{R}^D$
  - ▶  $W \in \mathbb{R}^{C \times D}$
- 2-layer NN:  $f(x) = W_2 \max(0, W_1 x)$ 
  - ▶  $x \in \mathbb{R}^D$
  - ▶  $W_1 \in \mathbb{R}^{H \times D}$
  - ▶  $W_2 \in \mathbb{R}^{C \times H}$
- 3-layer NN:  $f(x) = W_3 \max(0, W_2 \max(0, W_1 x))$ 
  - ▶  $x \in \mathbb{R}^D$
  - ▶  $W_1 \in \mathbb{R}^{H_1 \times D}$
  - ▶  $W_2 \in \mathbb{R}^{H_2 \times H_1}$
  - ▶  $W_3 \in \mathbb{R}^{C \times H_2}$

# Hierarchical computation



(Before) Linear score function:  $f = Wx$

(Now) 2-layer Neural Network  $f = W_2 \max(0, W_1 x)$



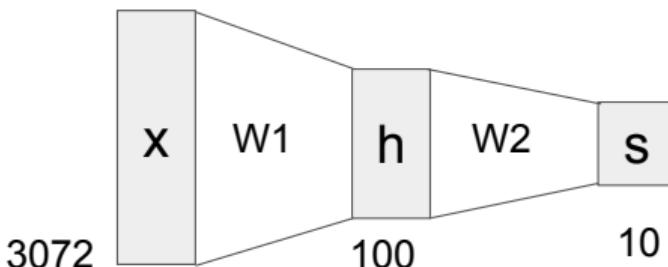
$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

# Better representation



**(Before)** Linear score function:  $f = Wx$

**(Now)** 2-layer Neural Network  $f = W_2 \max(0, W_1 x)$



Learn 100 templates instead of 10.

Share templates between classes

# Why do we need max?



- Linear classifier  $f(x) = Wx$
- 2-layer NN:  $f(x) = W_2 \max(0, W_1 x)$



# Why do we need max?

- Linear classifier  $f(x) = Wx$
- 2-layer NN:  $f(x) = W_2 \max(0, W_1 x)$

**Why do we need max?**



# Why do we need max?

- Linear classifier  $f(x) = Wx$
- 2-layer NN:  $f(x) = W_2 \max(0, W_1 x)$

## Why do we need max?

If we did not use max  $\rightarrow f(x) = W_2 W_1 x = \hat{W}x$ .



# Why do we need max?

- Linear classifier  $f(x) = Wx$
- 2-layer NN:  $f(x) = W_2 \max(0, W_1 x)$

## Why do we need max?

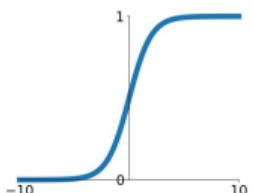
If we did not use max  $\rightarrow f(x) = W_2 W_1 x = \hat{W}x$ . This is just linear classifier!

# Activation function



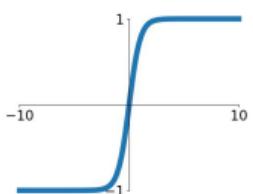
## Sigmoid

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



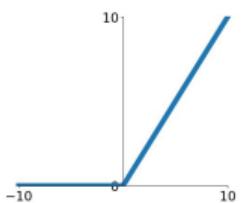
## tanh

$$\tanh(x)$$



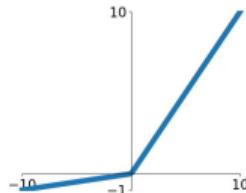
## ReLU

$$\max(0, x)$$



## Leaky ReLU

$$\max(0.1x, x)$$

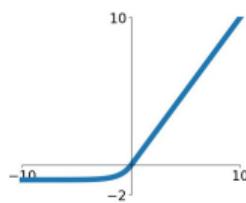


## Maxout

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

## ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

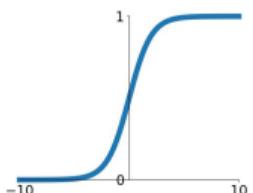


# Activation function



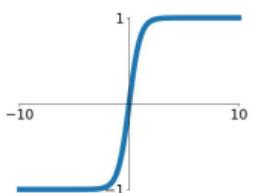
## Sigmoid

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



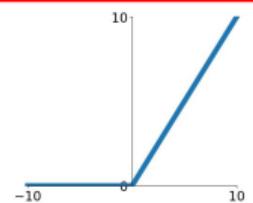
## tanh

$$\tanh(x)$$



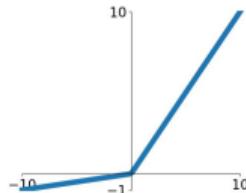
## ReLU

$$\max(0, x)$$



## Leaky ReLU

$$\max(0.1x, x)$$

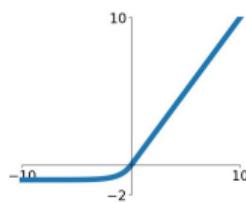


## Maxout

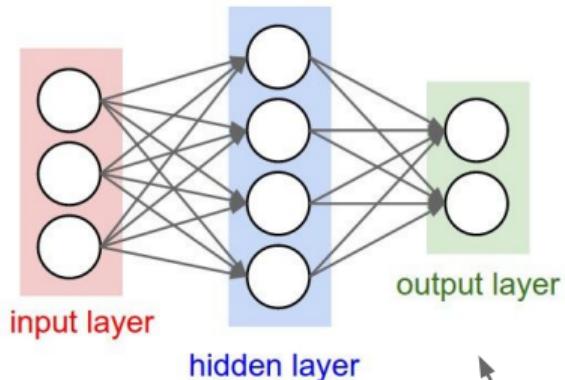
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

## ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

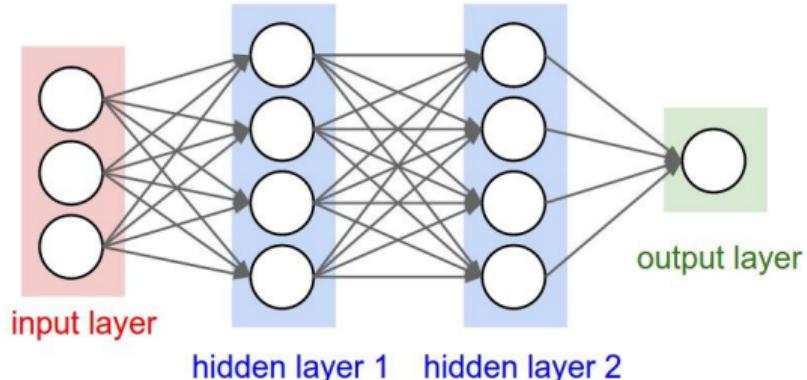


# Architectures



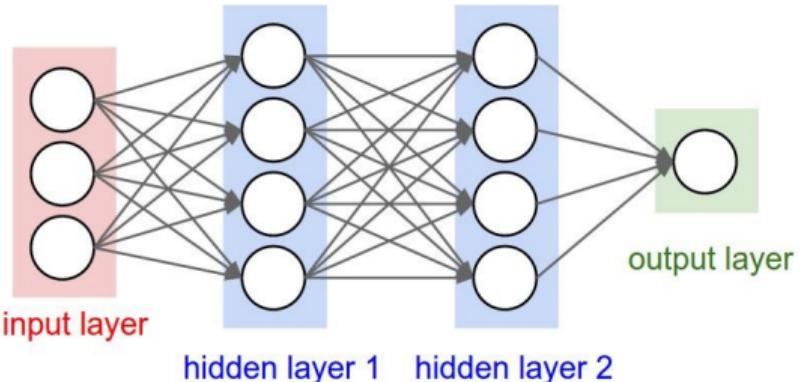
“2-layer Neural Net”, or  
“1-hidden-layer Neural Net”

**“Fully-connected” layers**



“3-layer Neural Net”, or  
“2-hidden-layer Neural Net”

# Implementation example



```
# forward-pass of a 3-layer neural network:  
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)  
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)  
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)  
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)  
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

# Implementation example - training



```
1 import numpy as np
2 from numpy.random import randn
3
4 N, D_in, H, D_out = 64, 1000, 100, 10
5 x, y = randn(N, D_in), randn(N, D_out)
6 w1, w2 = randn(D_in, H), randn(H, D_out)
7
8 for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14    grad_y_pred = 2.0 * (y_pred - y)
15    grad_w2 = h.T.dot(grad_y_pred)
16    grad_h = grad_y_pred.dot(w2.T)
17    grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19    w1 -= 1e-4 * grad_w1
20    w2 -= 1e-4 * grad_w2
```



# Implementation example - training

```
1 import numpy as np
2 from numpy.random import randn
3
4 N, D_in, H, D_out = 64, 1000, 100, 10
5 x, y = randn(N, D_in), randn(N, D_out)
6 w1, w2 = randn(D_in, H), randn(H, D_out)
7
8 for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14    grad_y_pred = 2.0 * (y_pred - y)
15    grad_w2 = h.T.dot(grad_y_pred)
16    grad_h = grad_y_pred.dot(w2.T)
17    grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19    w1 -= 1e-4 * grad_w1
20    w2 -= 1e-4 * grad_w2
```

Define the network

# Implementation example - training



```
1 import numpy as np
2 from numpy.random import randn
3
4 N, D_in, H, D_out = 64, 1000, 100, 10
5 x, y = randn(N, D_in), randn(N, D_out)
6 w1, w2 = randn(D_in, H), randn(H, D_out)
7
8 for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14 grad_y_pred = 2.0 * (y_pred - y)
15 grad_w2 = h.T.dot(grad_y_pred)
16 grad_h = grad_y_pred.dot(w2.T)
17 grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19 w1 -= 1e-4 * grad_w1
20 w2 -= 1e-4 * grad_w2
```

Define the network

Forward pass



# Implementation example - training

```
1 import numpy as np
2 from numpy.random import randn
3
4 N, D_in, H, D_out = 64, 1000, 100, 10
5 x, y = randn(N, D_in), randn(N, D_out)
6 w1, w2 = randn(D_in, H), randn(H, D_out)
7
8 for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14    grad_y_pred = 2.0 * (y_pred - y)
15    grad_w2 = h.T.dot(grad_y_pred)
16    grad_h = grad_y_pred.dot(w2.T)
17    grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19    w1 -= 1e-4 * grad_w1
20    w2 -= 1e-4 * grad_w2
```

Define the network

Forward pass

Calculate the analytical gradients



# Implementation example - training

```
1 import numpy as np
2 from numpy.random import randn
3
4 N, D_in, H, D_out = 64, 1000, 100, 10
5 x, y = randn(N, D_in), randn(N, D_out)
6 w1, w2 = randn(D_in, H), randn(H, D_out)
7
8 for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14    grad_y_pred = 2.0 * (y_pred - y)
15    grad_w2 = h.T.dot(grad_y_pred)
16    grad_h = grad_y_pred.dot(w2.T)
17    grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19    w1 -= 1e-4 * grad_w1
20    w2 -= 1e-4 * grad_w2
```

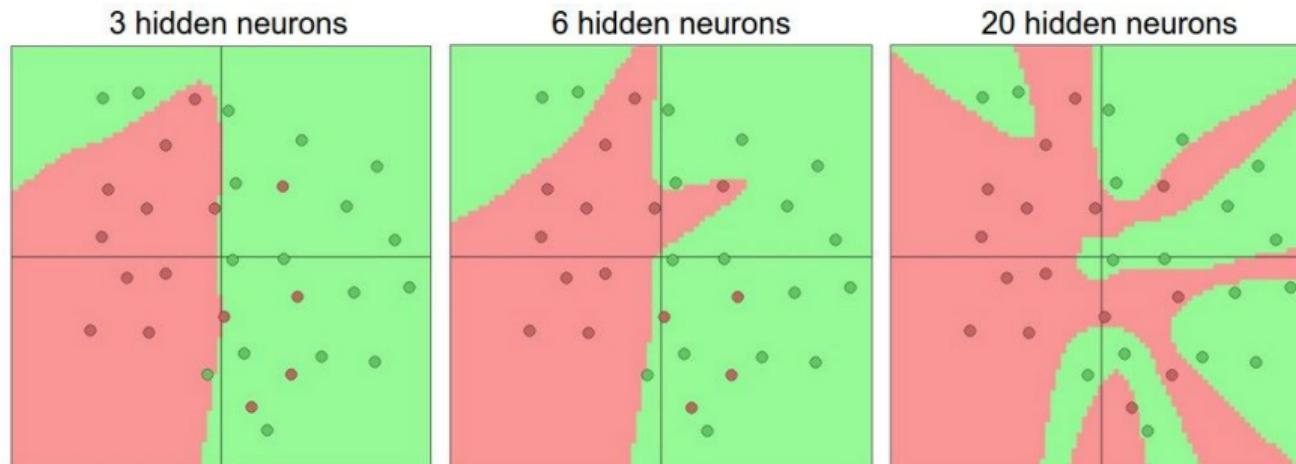
Define the network

Forward pass

Calculate the analytical gradients

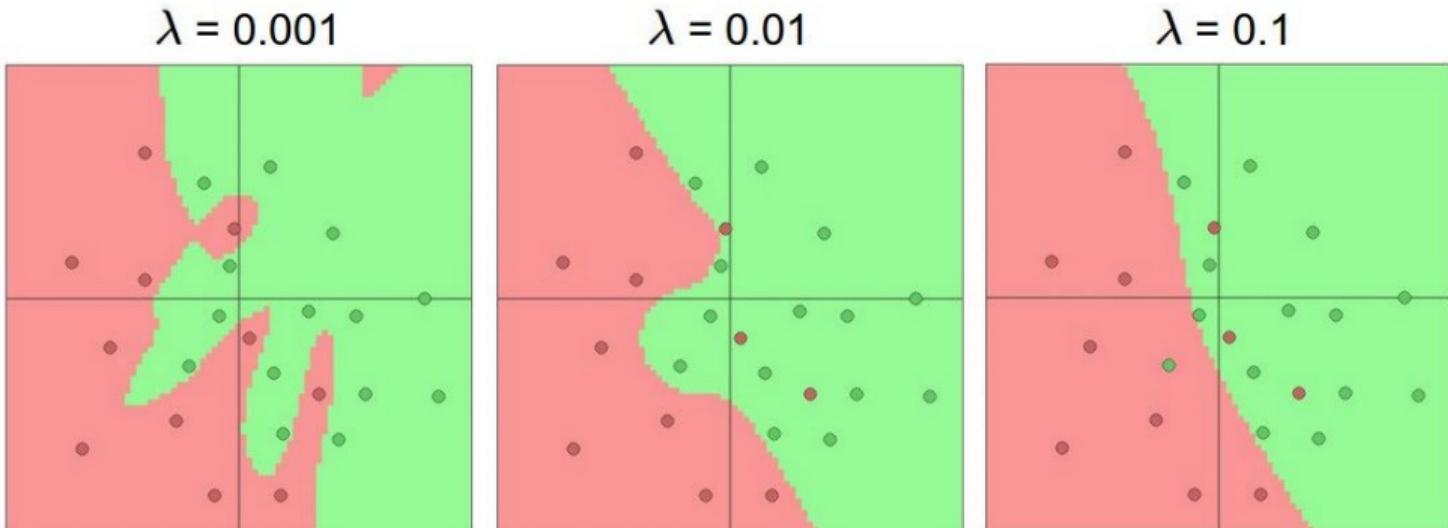
Gradient descent

# More neurons - more complexity



↑  
more neurons = more capacity

# Do not limit neurons as regularization!

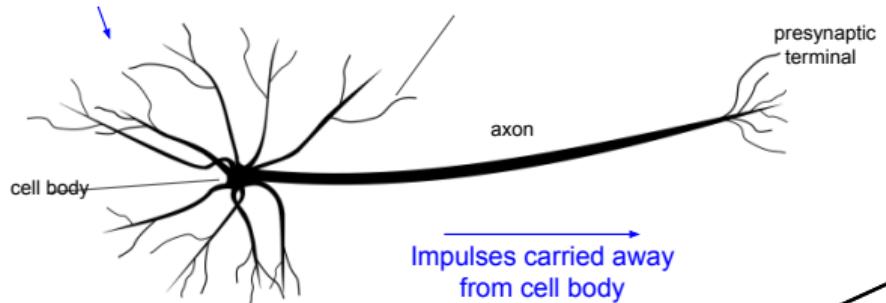


(Web demo with ConvNetJS:

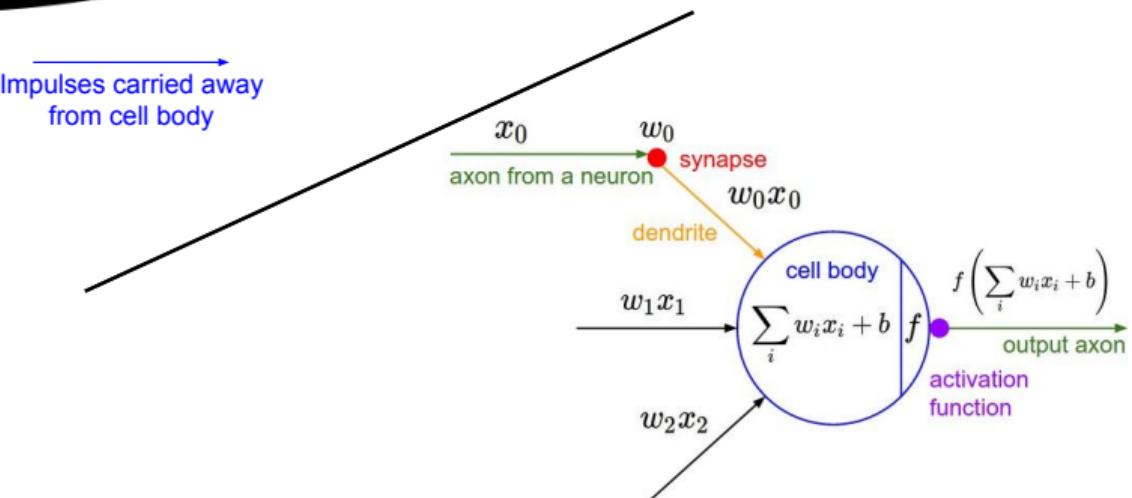
<http://cs.stanford.edu/people/karpathy/convnetjs/demo>

$$L(W) = \frac{1}{N} \sum_i^N L_i(f(x_i, W), y_i) + \lambda R(W)$$

# Biological neurons



This image by Felipe Perucco  
is licensed under [CC-BY 3.0](#)



# Neural network training



$$s = f(x; W_1, W_2) = W_2 \max(0, W_1 x) \quad \text{Nonlinear score function}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM Loss on predictions}$$

$$R(W) = \sum_k W_k^2 \quad \text{Regularization}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(W_1) + \lambda R(W_2) \quad \text{Total loss: data loss + regularization}$$

# Gradient computation



$$s = f(x; W_1, W_2) = W_2 \max(0, W_1 x) \quad \text{Nonlinear score function}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM Loss on predictions}$$

$$R(W) = \sum_k W_k^2 \quad \text{Regularization}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(W_1) + \lambda R(W_2) \quad \text{Total loss: data loss + regularization}$$

If we can compute  $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}$  then we can learn  $W_1$  and  $W_2$

# Doing it by hand is tedious



$$s = f(x; W) = Wx$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda \sum_k W_k^2$$

$$= \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2$$

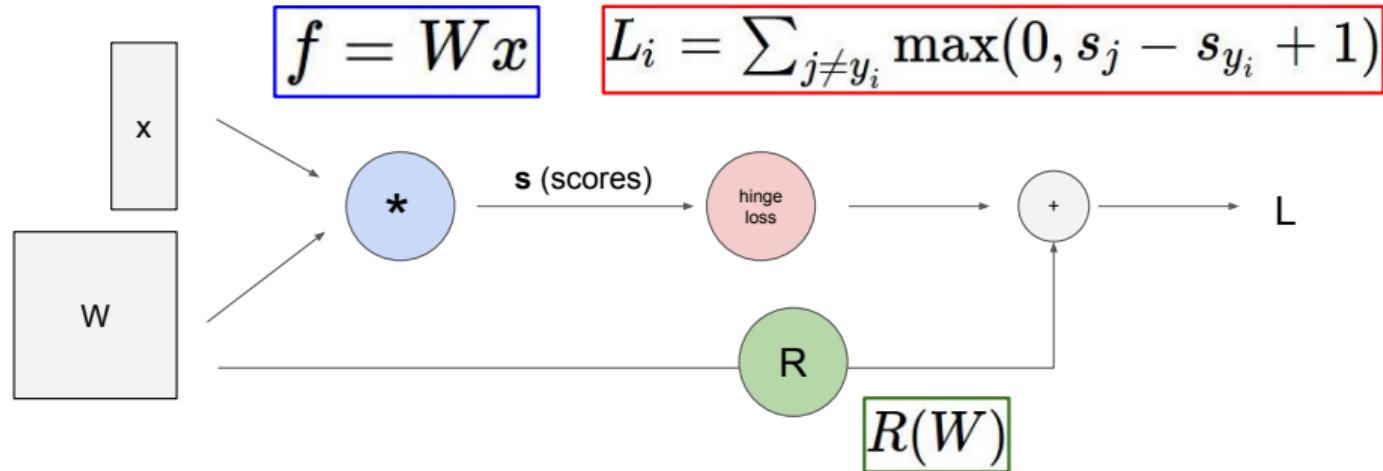
$$\nabla_W L = \nabla_W \left( \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2 \right)$$

**Problem:** Very tedious: Lots of matrix calculus, need lots of paper

**Problem:** What if we want to change loss? E.g. use softmax instead of SVM? Need to re-derive from scratch =(

**Problem:** Not feasible for very complex models!

# Computational graphs + backpropagation



# Backpropagation



Backpropagation: a simple example

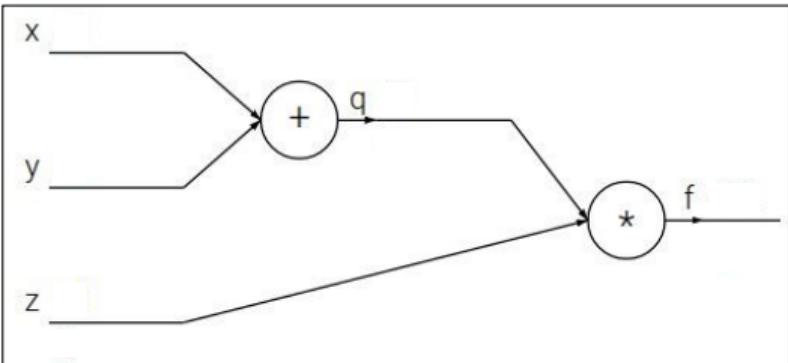
$$f(x, y, z) = (x + y)z$$

# Backpropagation



Backpropagation: a simple example

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



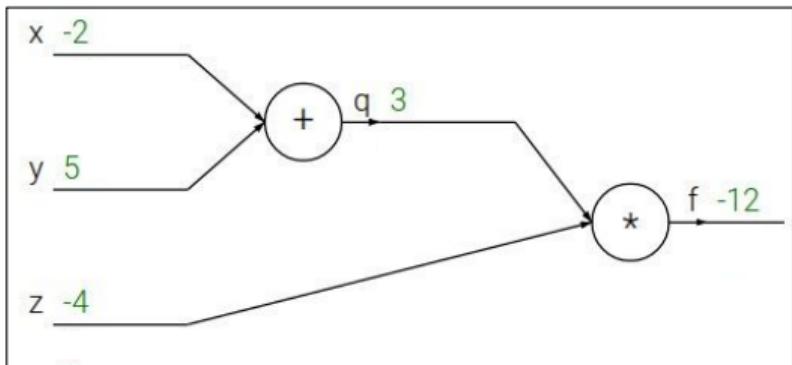
# Backpropagation



Backpropagation: a simple example

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

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



# Backpropagation

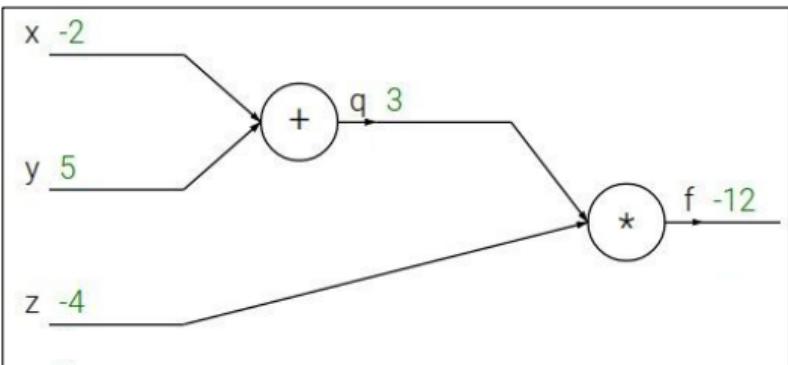


Backpropagation: a simple example

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

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

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



# Backpropagation



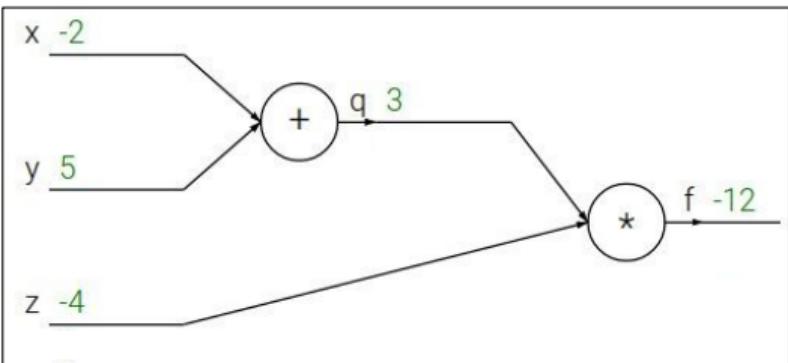
Backpropagation: a simple example

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

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

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

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



# Backpropagation



Backpropagation: a simple example

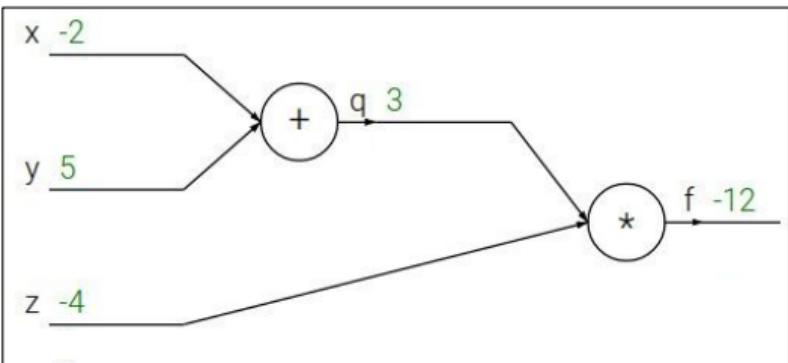
$$f(x, y, z) = (x + y)z$$

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

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

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

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



# Backpropagation



Backpropagation: a simple example

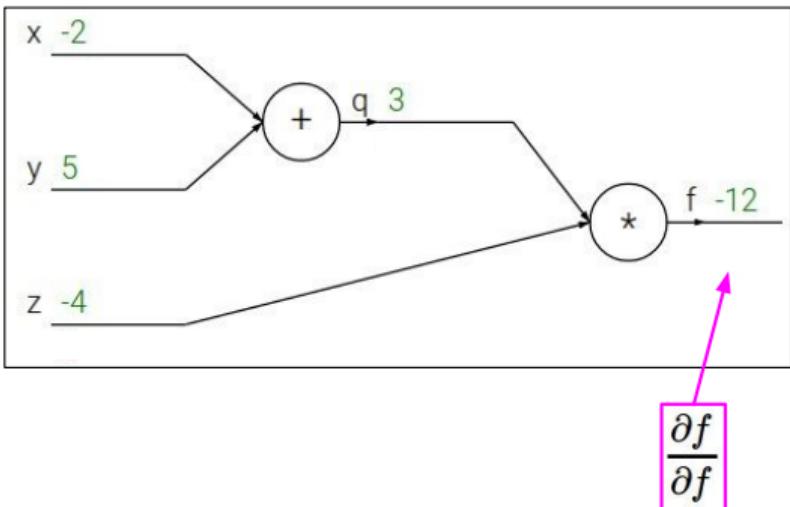
$$f(x, y, z) = (x + y)z$$

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

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

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

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



# Backpropagation



Backpropagation: a simple example

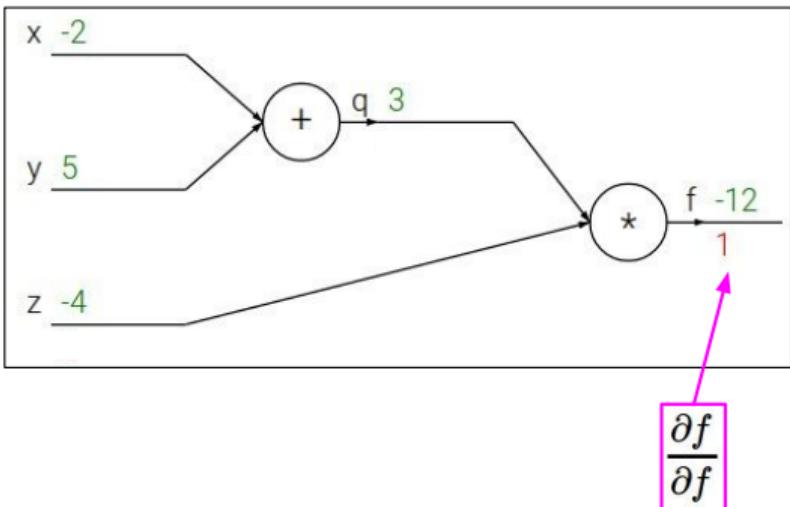
$$f(x, y, z) = (x + y)z$$

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

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

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

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



# Backpropagation



Backpropagation: a simple example

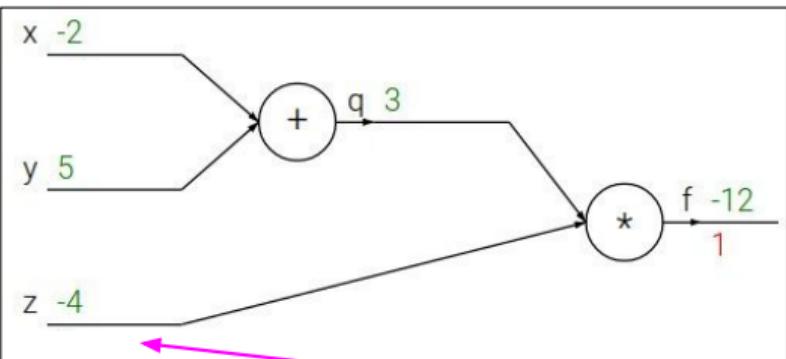
$$f(x, y, z) = (x + y)z$$

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

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

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

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



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

# Backpropagation



Backpropagation: a simple example

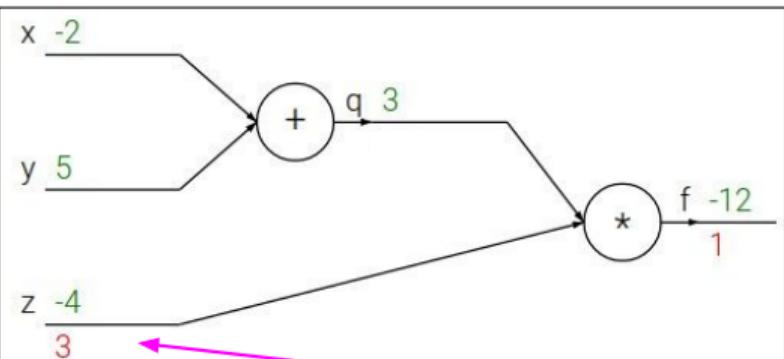
$$f(x, y, z) = (x + y)z$$

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

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

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

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



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

# Backpropagation



Backpropagation: a simple example

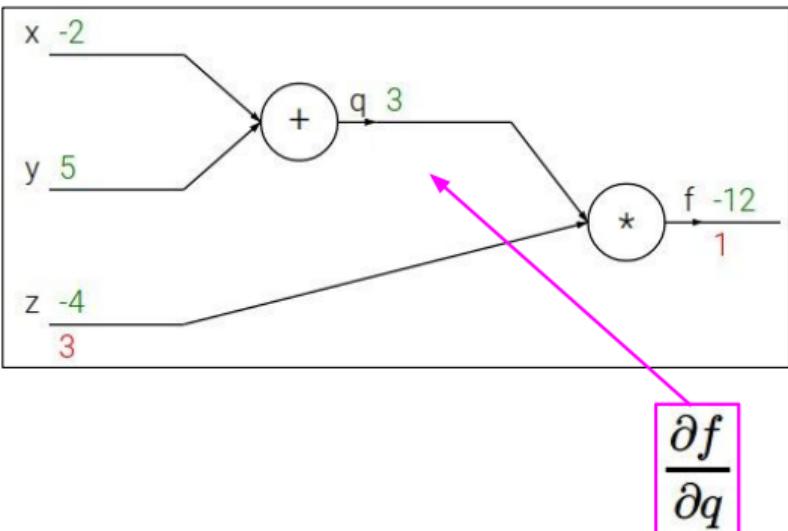
$$f(x, y, z) = (x + y)z$$

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

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

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

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



# Backpropagation



Backpropagation: a simple example

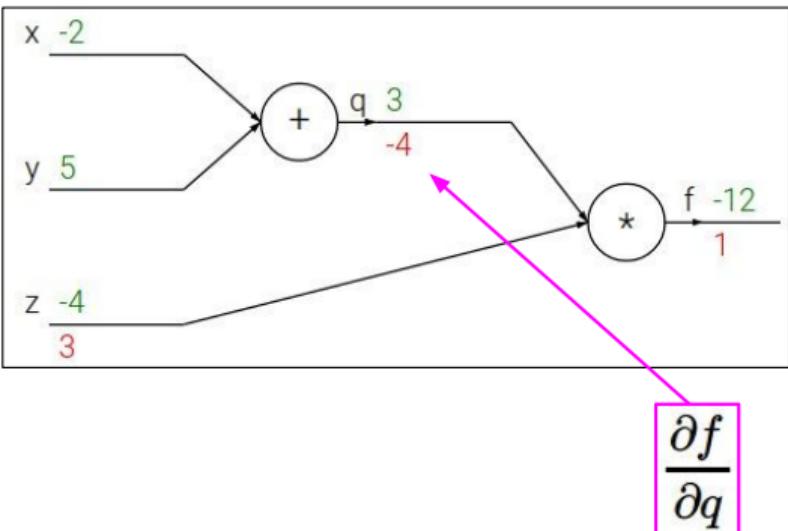
$$f(x, y, z) = (x + y)z$$

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

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

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

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



# Backpropagation



Backpropagation: a simple example

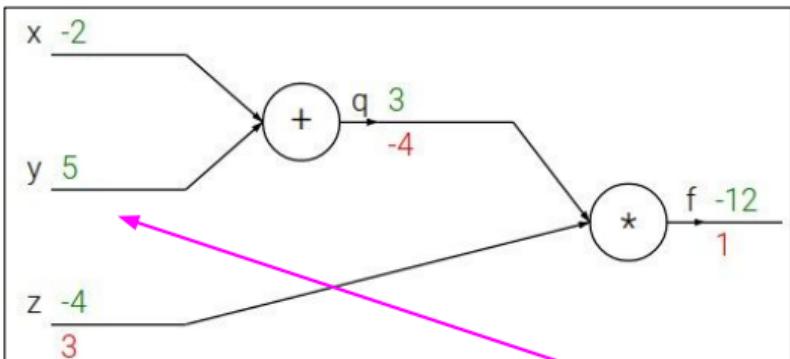
$$f(x, y, z) = (x + y)z$$

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

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

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

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



Chain rule:

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

Upstream  
gradient

Local  
gradient

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

# Backpropagation



Backpropagation: a simple example

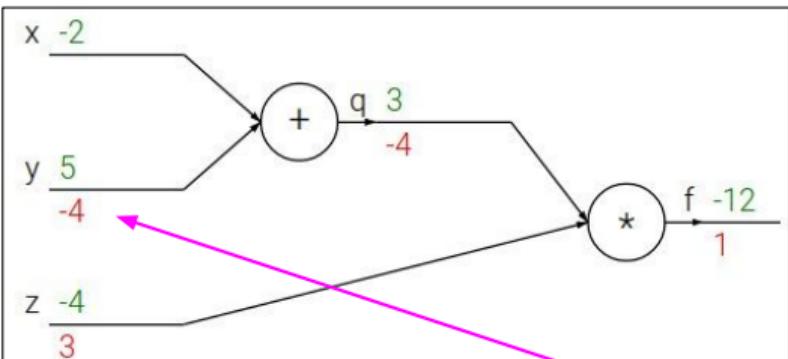
$$f(x, y, z) = (x + y)z$$

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

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

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

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



Chain rule:

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

Upstream  
gradient      Local  
gradient

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

# Backpropagation



Backpropagation: a simple example

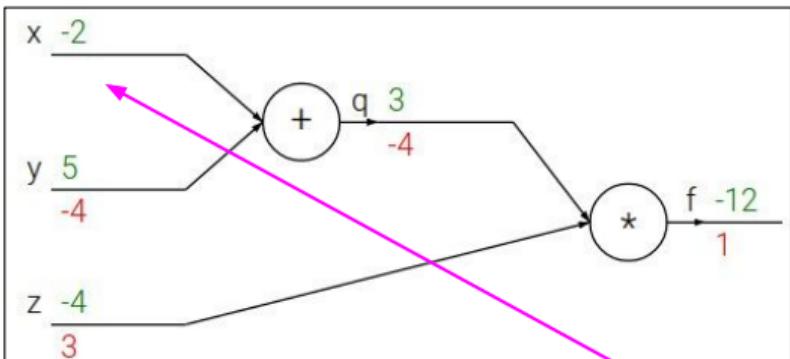
$$f(x, y, z) = (x + y)z$$

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

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

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

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



Chain rule:

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

Upstream gradient      Local gradient

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

# Backpropagation



Backpropagation: a simple example

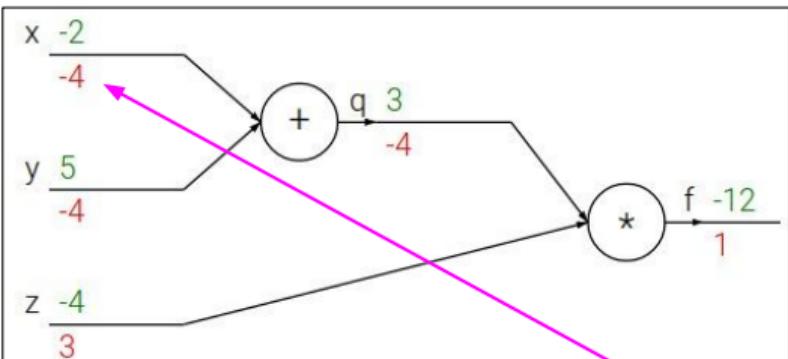
$$f(x, y, z) = (x + y)z$$

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

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

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

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



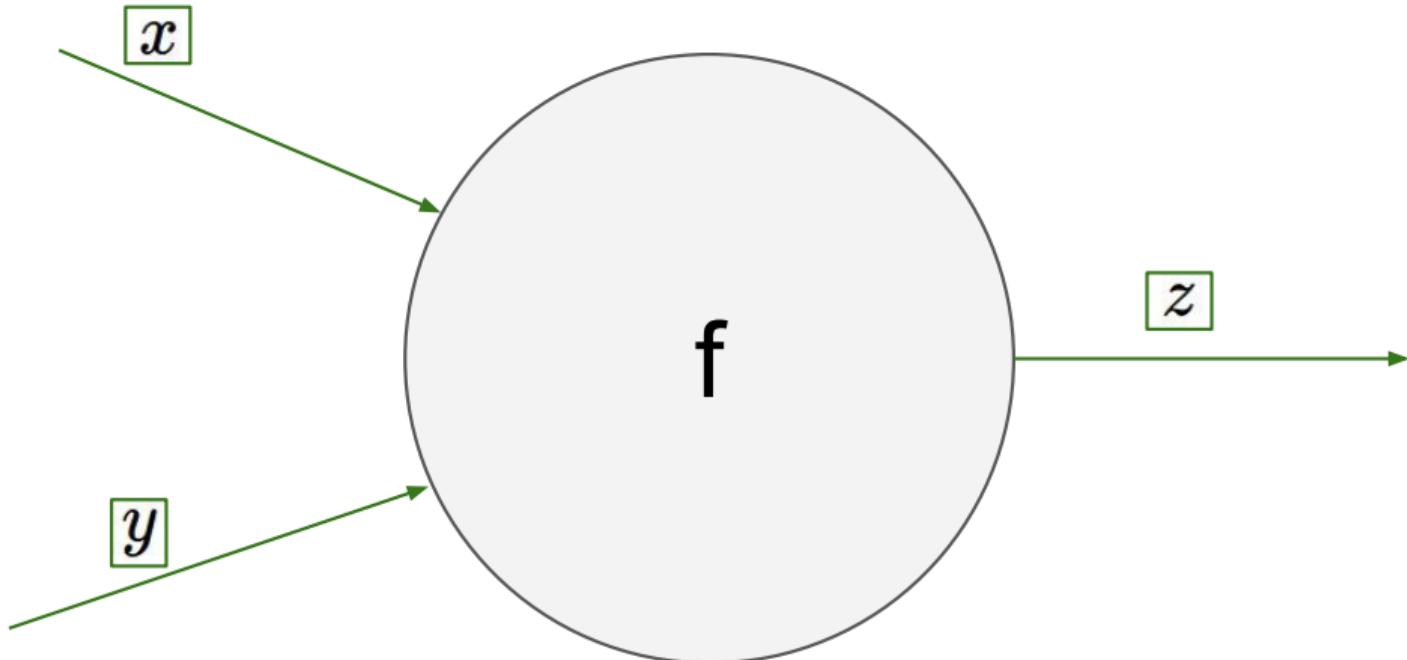
Chain rule:

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

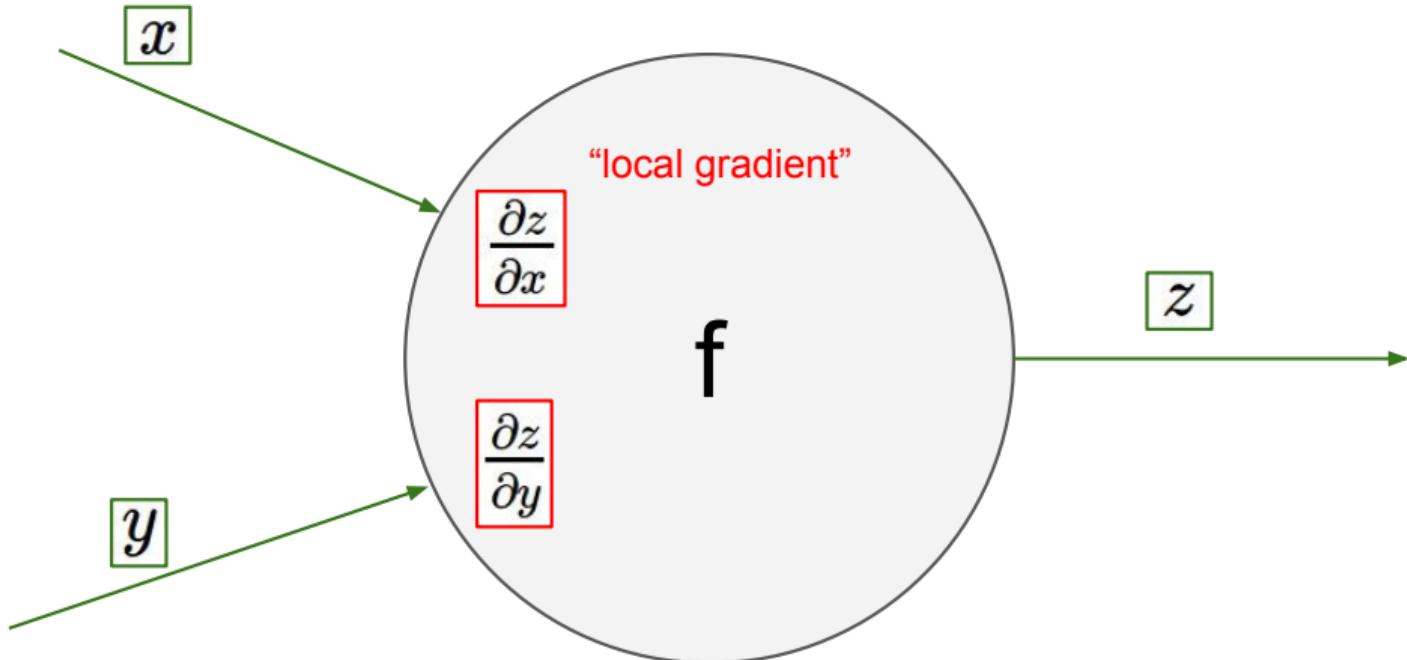
Upstream gradient      Local gradient

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

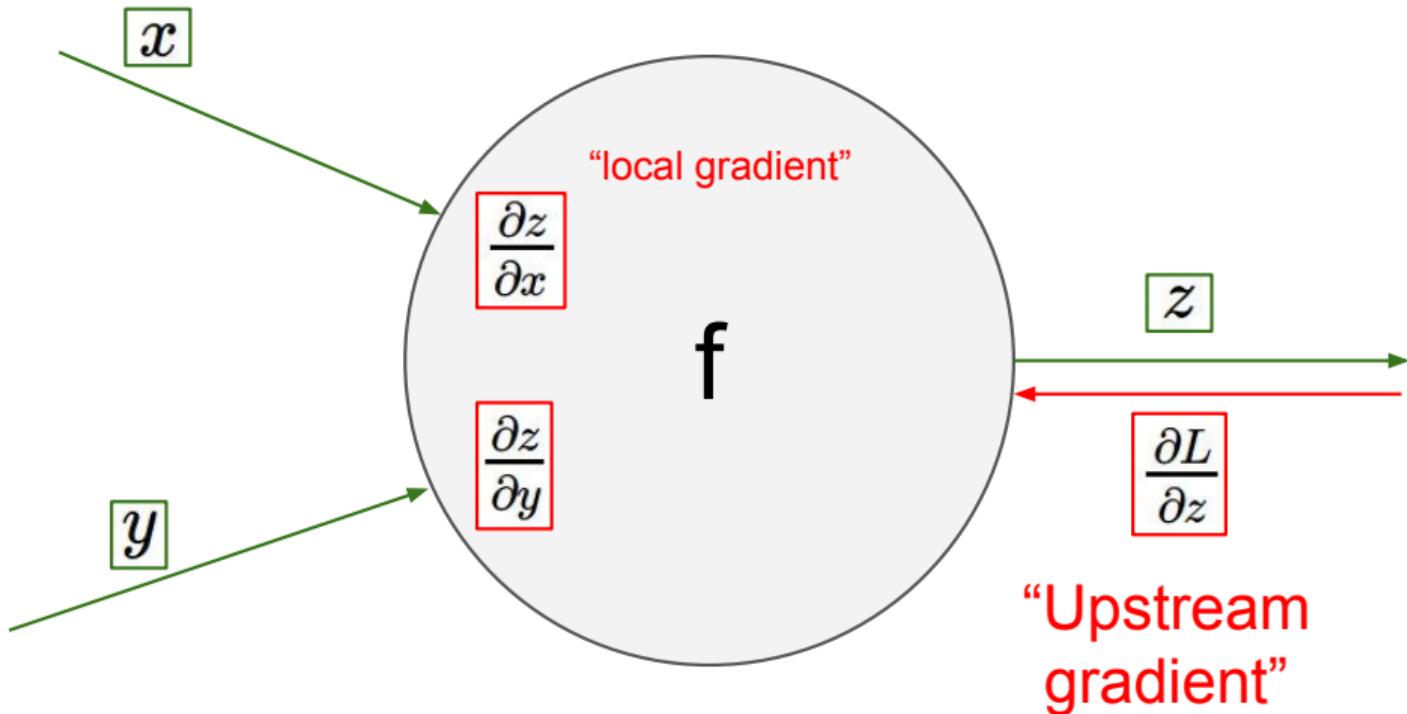
# Node in computational graph



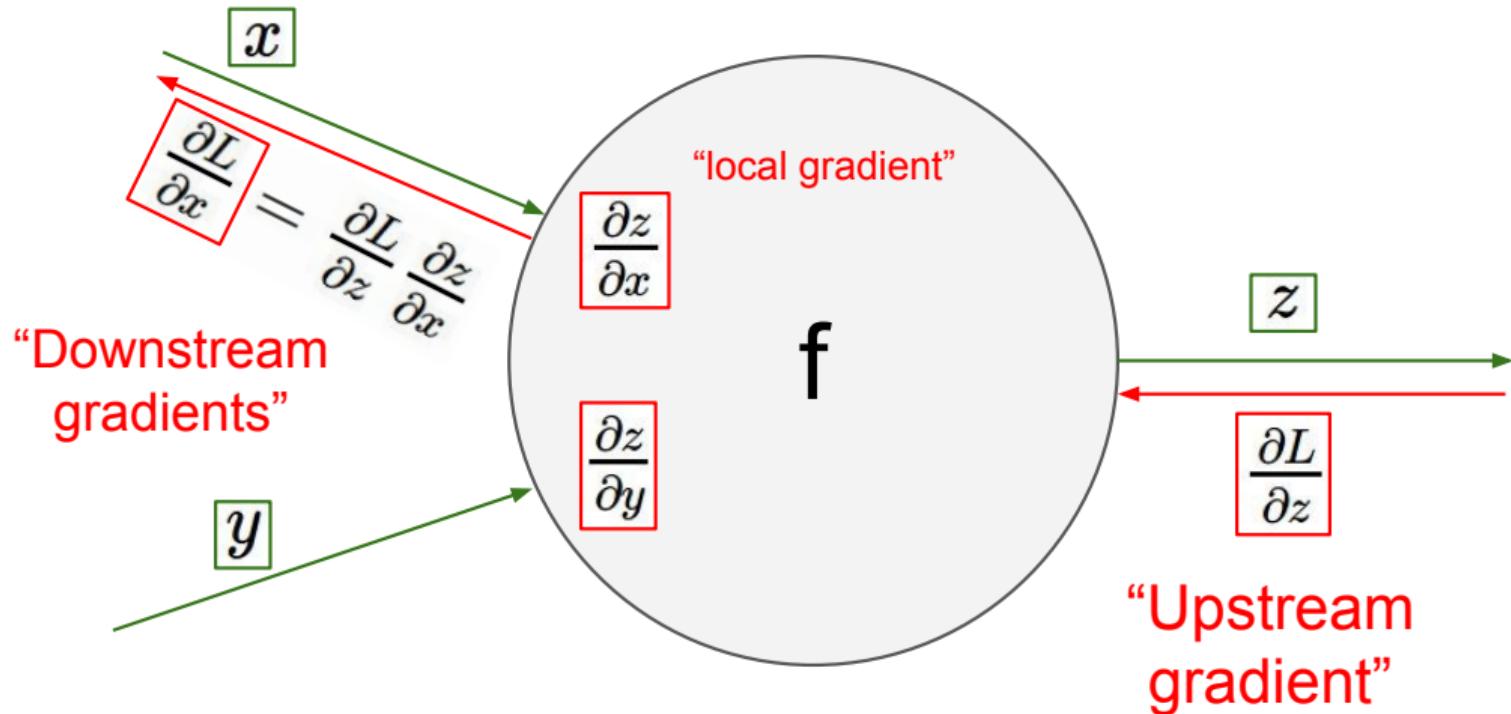
# Node in computational graph



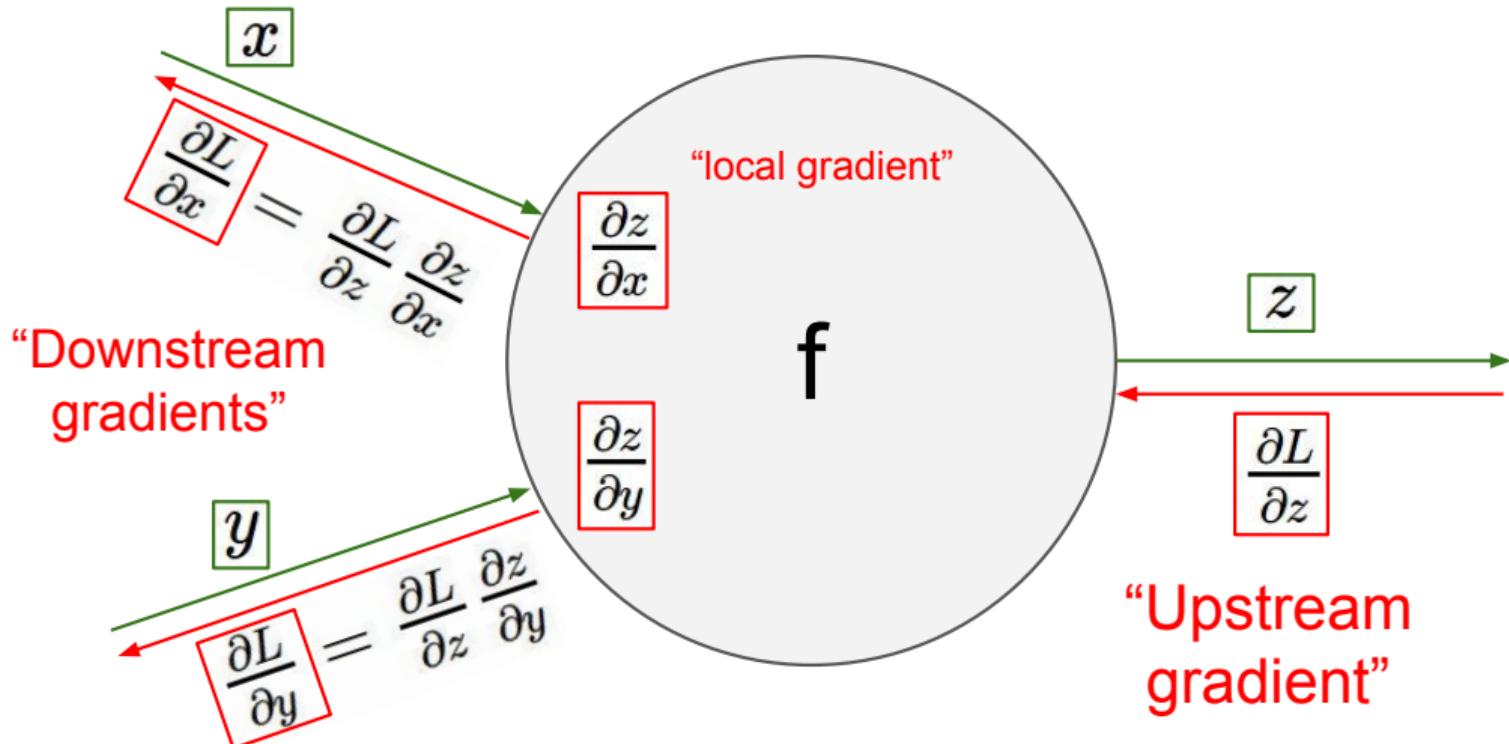
# Node in computational graph



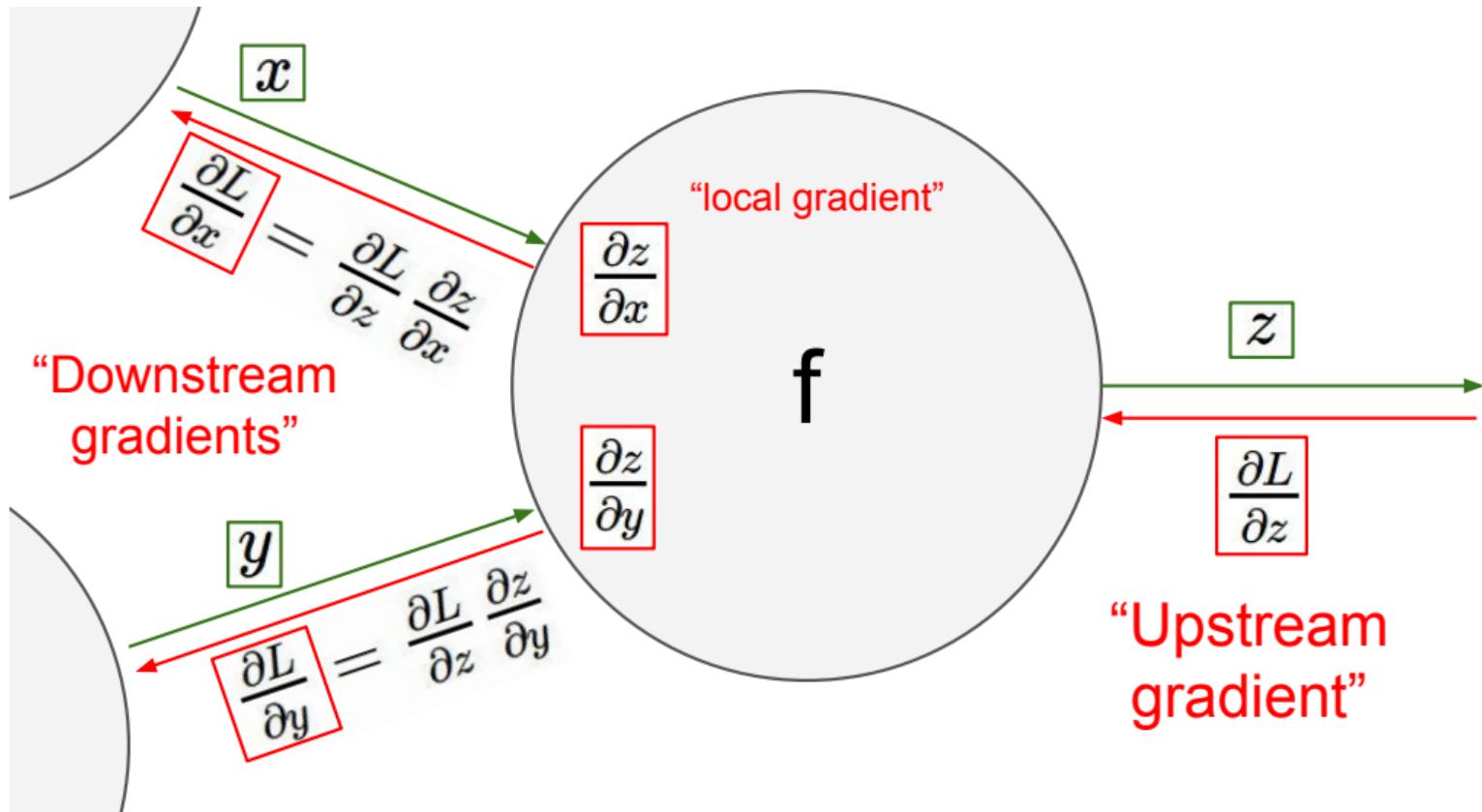
# Node in computational graph



# Node in computational graph



# Node in computational graph

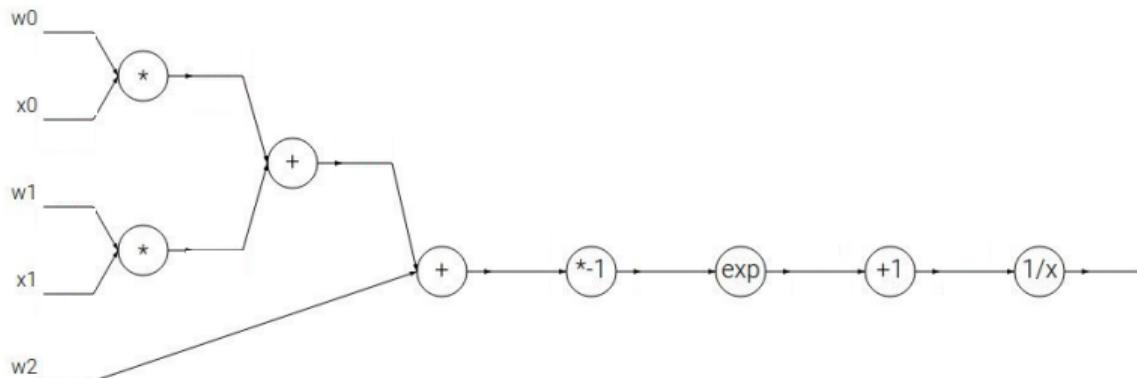


# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

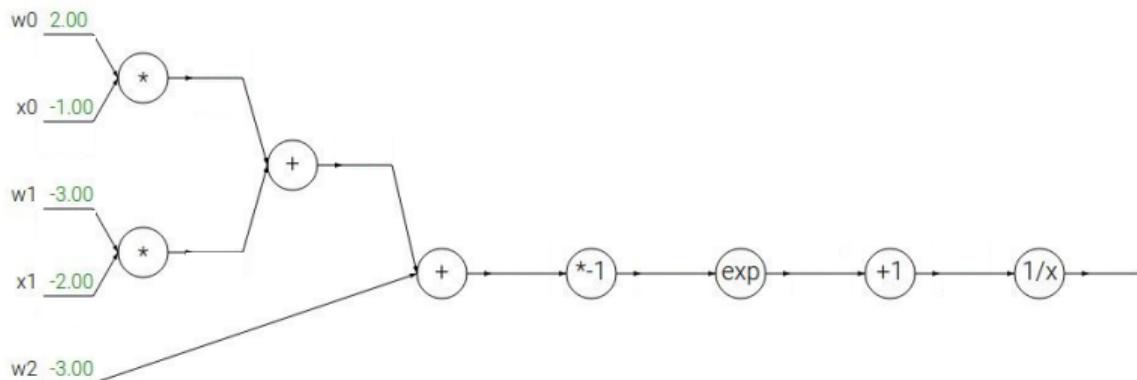


# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

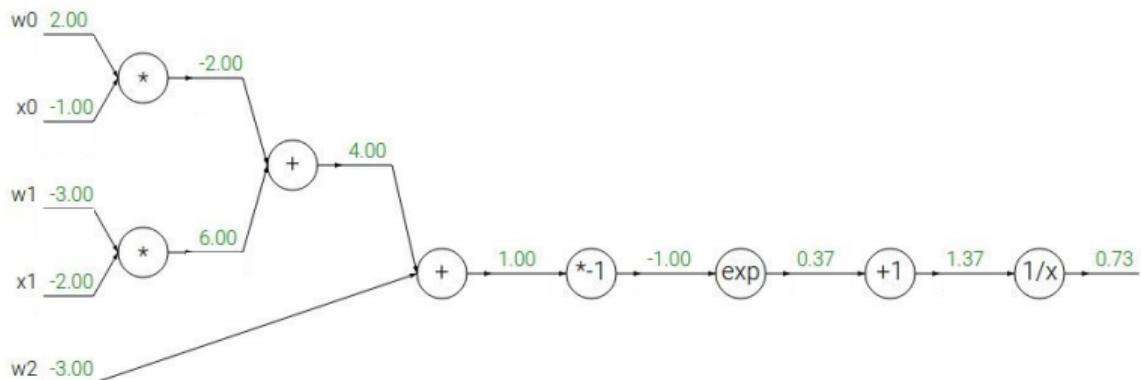


# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

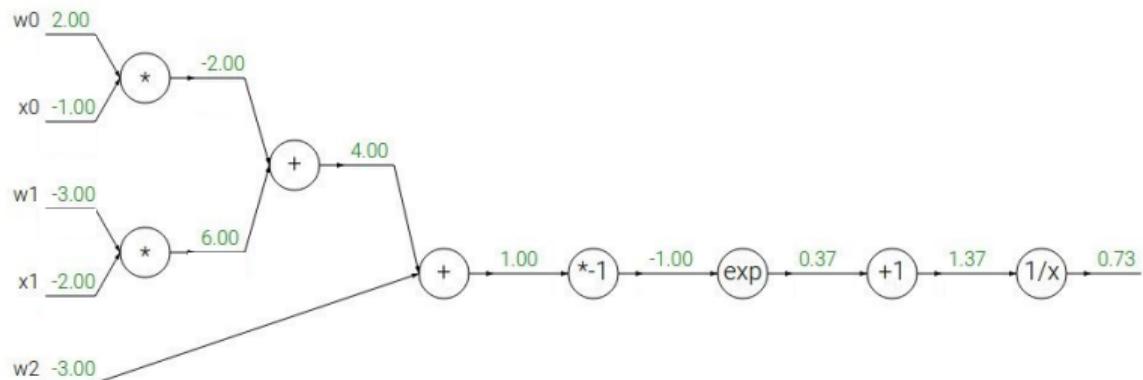


# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

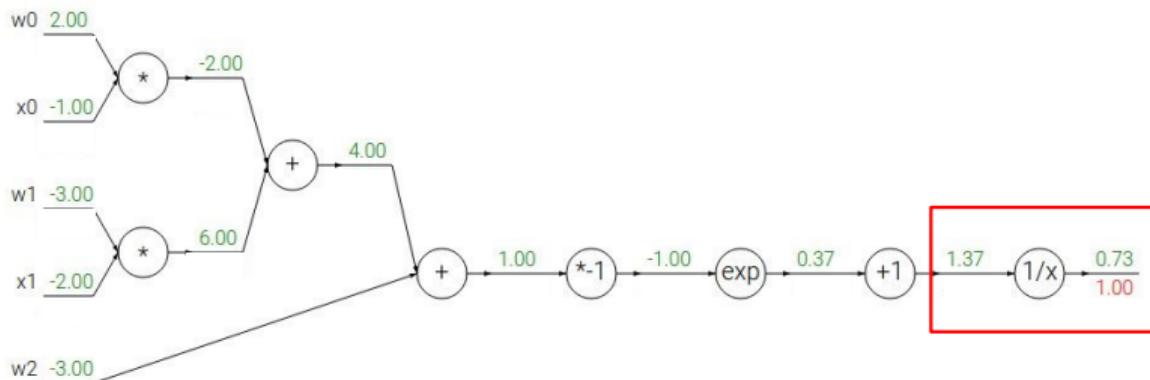
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = -1/x^2$$

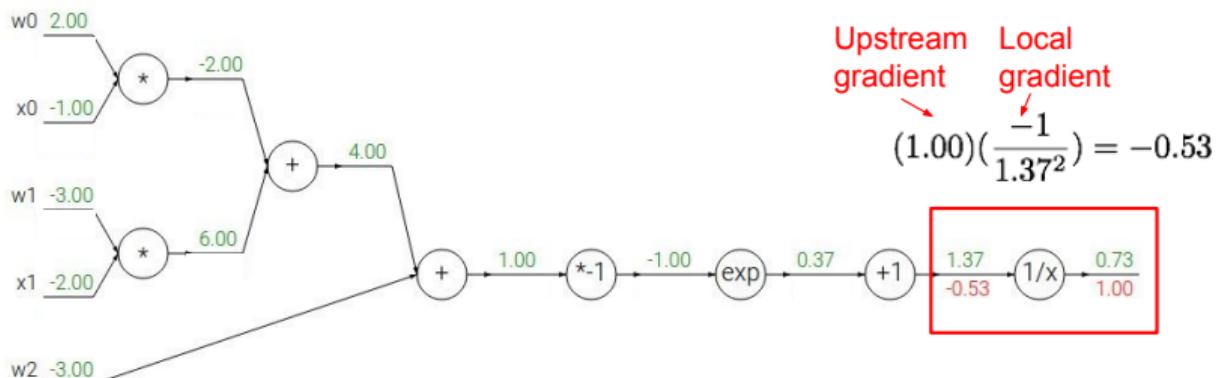
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

$\rightarrow$

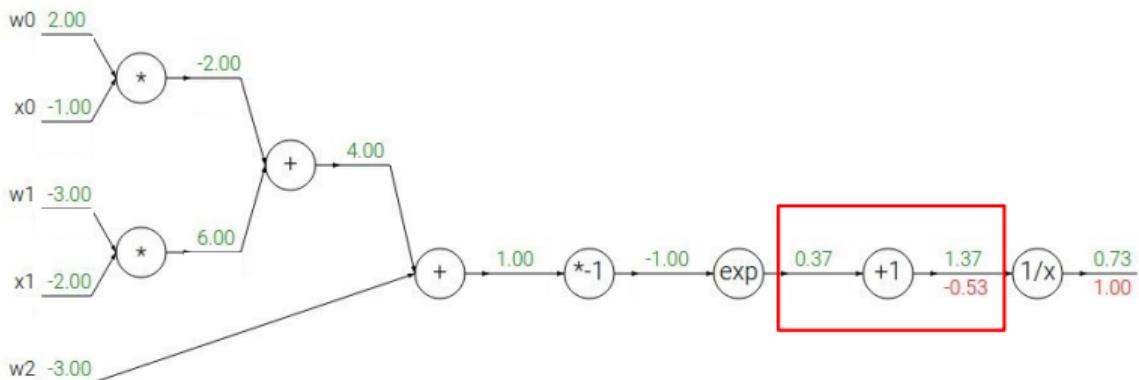
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

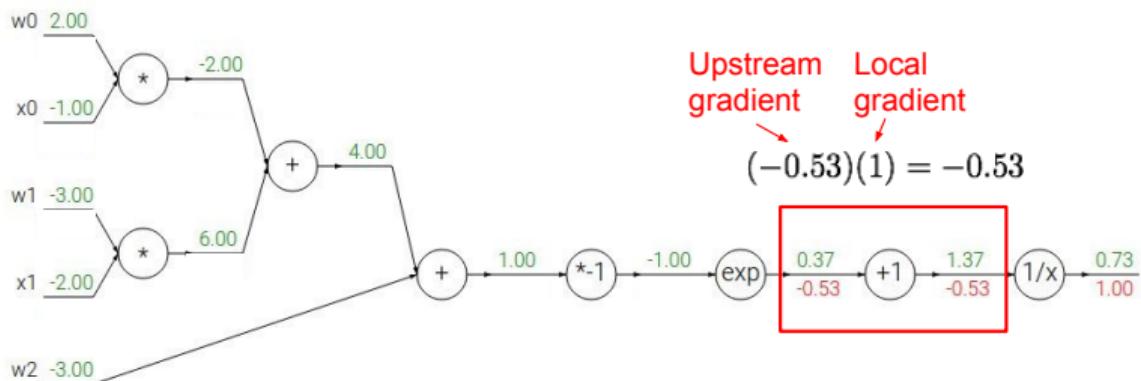
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

$\rightarrow$

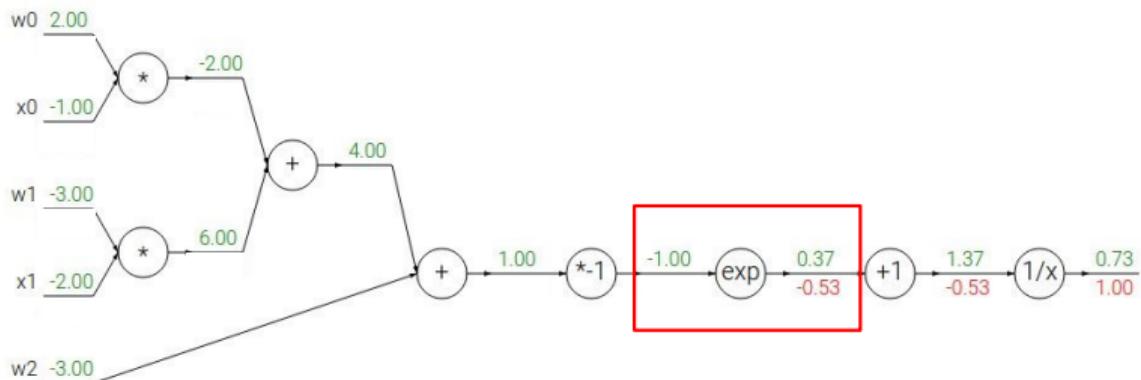
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

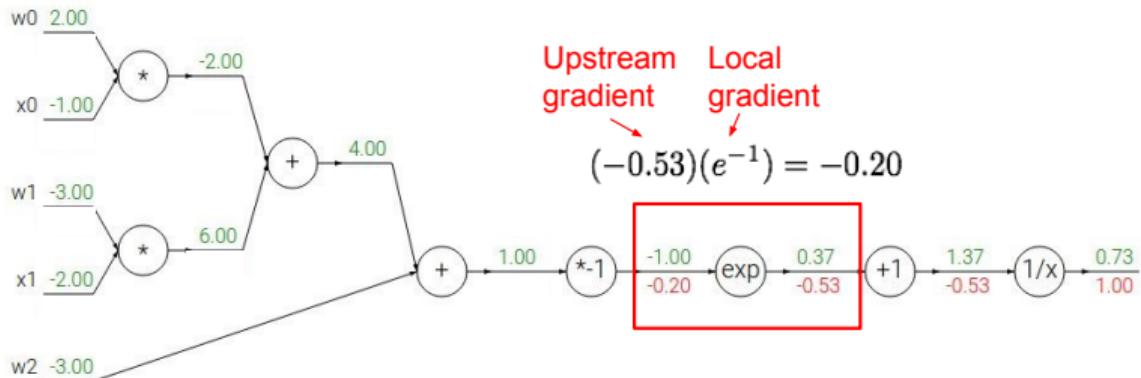
$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

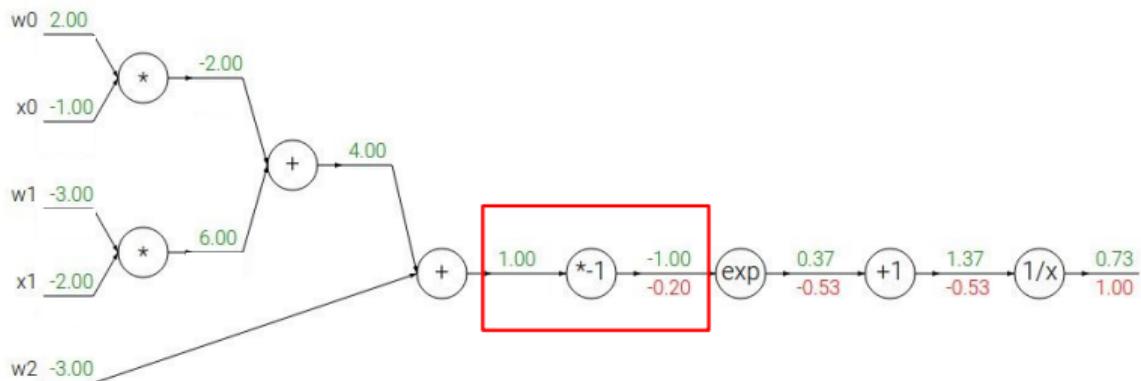
$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = -1/x^2$$

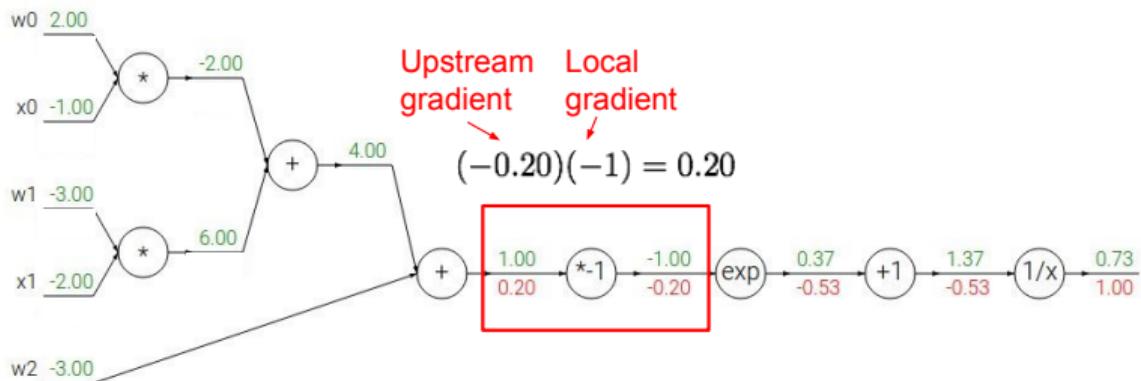
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

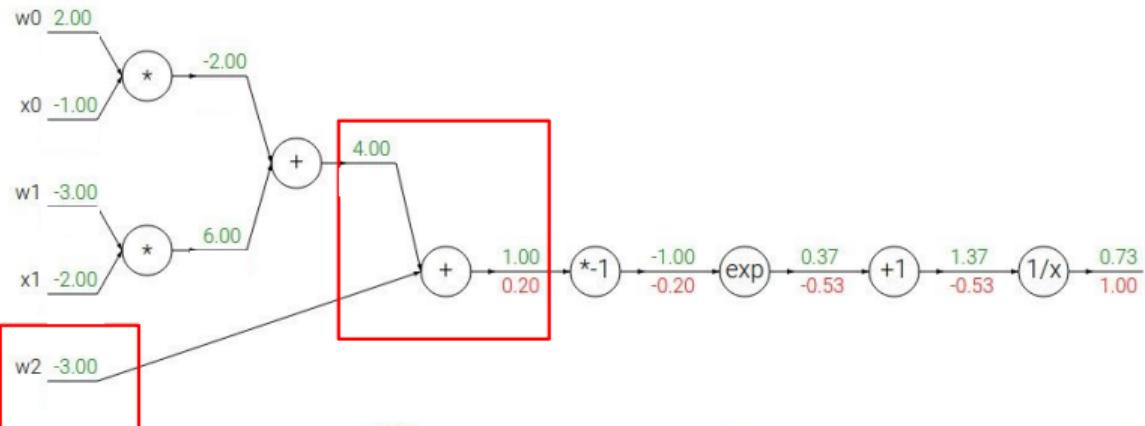
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f_c(x) = c + x$$

→

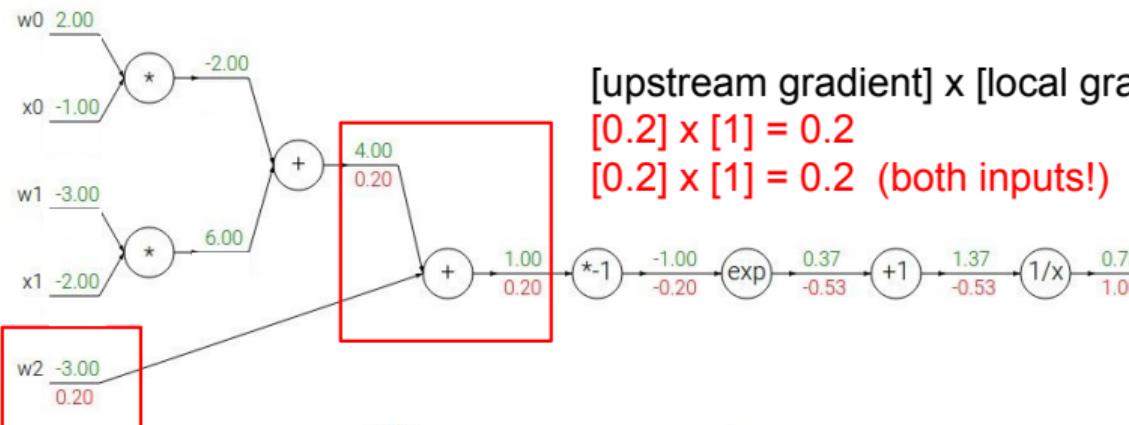
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

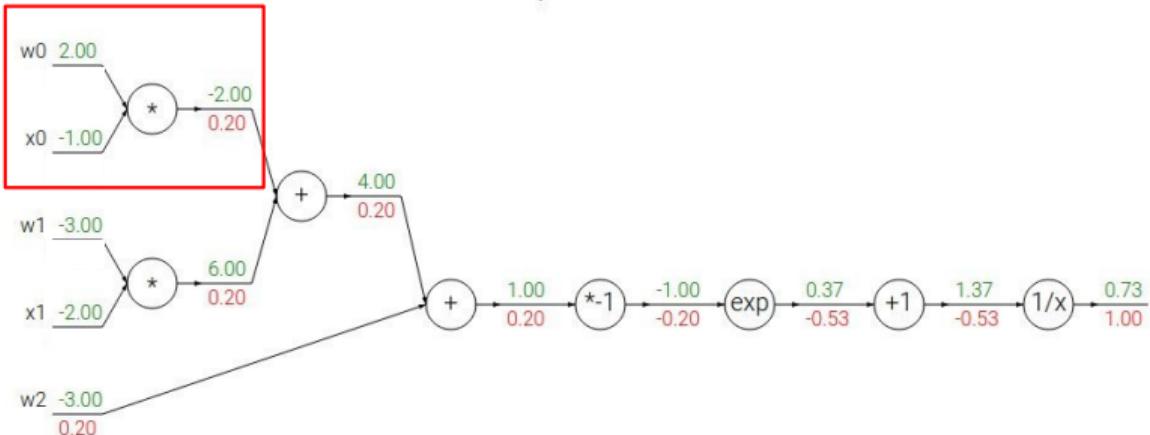
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

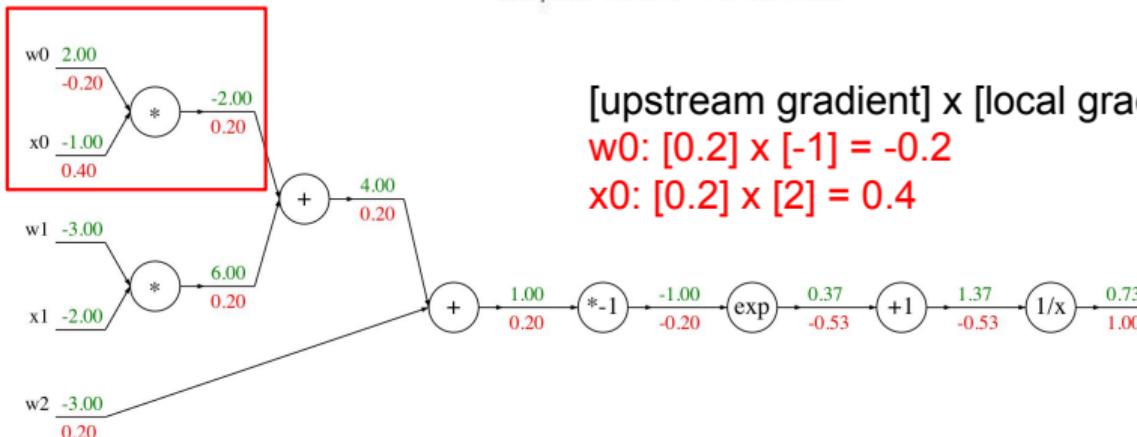
$$\frac{df}{dx} = 1$$

# Backpropagation



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



[upstream gradient] x [local gradient]

$$w_0: [0.2] \times [-1] = -0.2$$

$$x_0: [0.2] \times [2] = 0.4$$

$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

$\rightarrow$

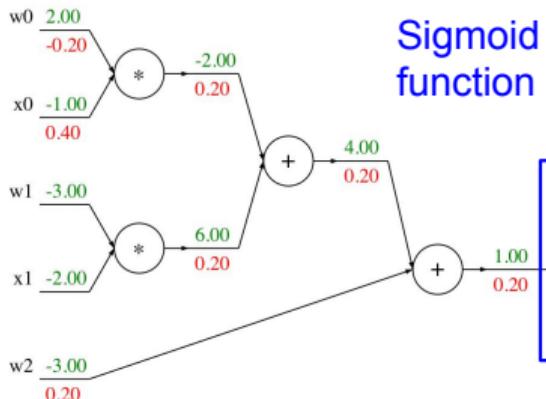
$$\frac{df}{dx} = 1$$

# Backpropagation



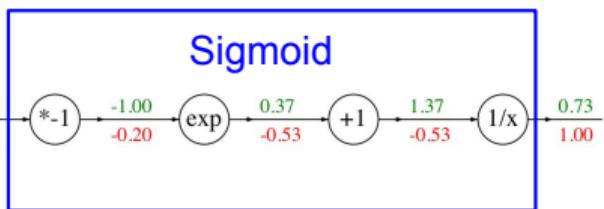
Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Sigmoid  
function

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



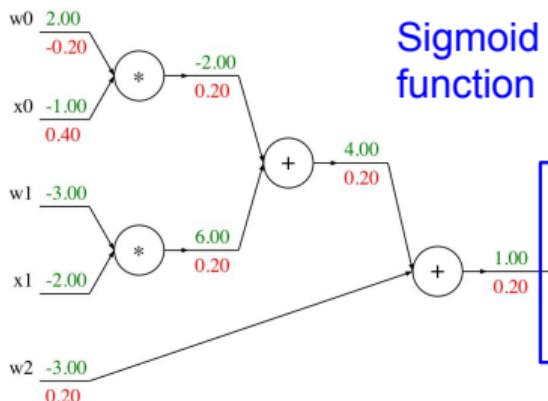
Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!

# Backpropagation



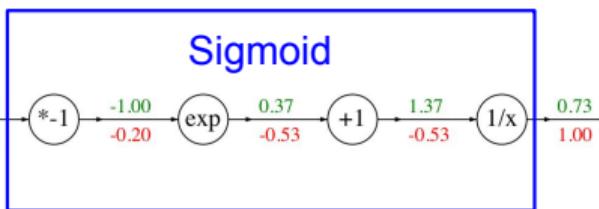
Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Sigmoid  
function

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



Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!

Sigmoid local gradient:

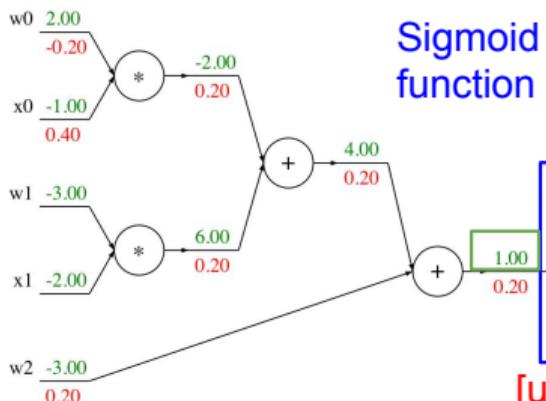
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

# Backpropagation

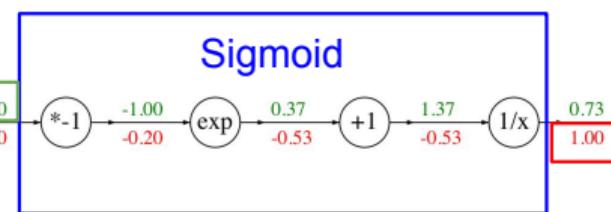


Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



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



Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!

Sigmoid local gradient:

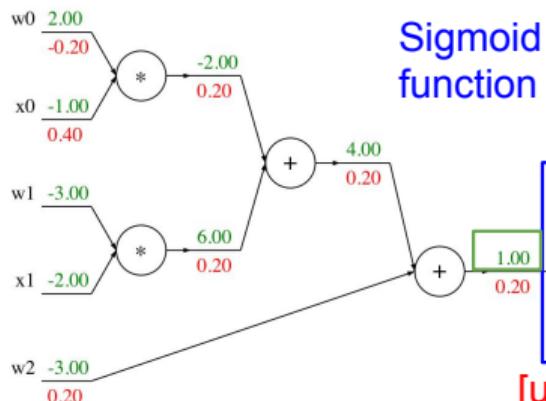
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

# Backpropagation



Another example:

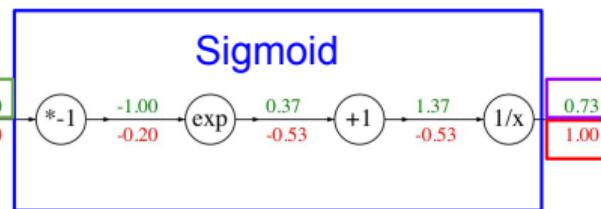
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Sigmoid  
function

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

Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!



[upstream gradient]  $\times$  [local gradient]  
 $[1.00] \times [(1 - 0.73)(0.73)] = 0.2$

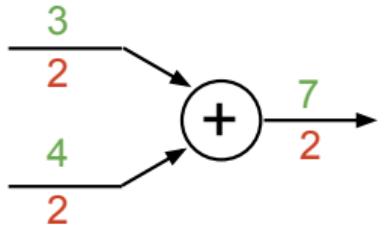
Sigmoid local  
gradient:

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

# Nodes as gates



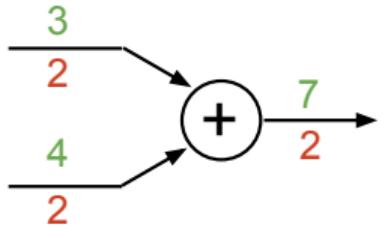
**add gate:** gradient distributor



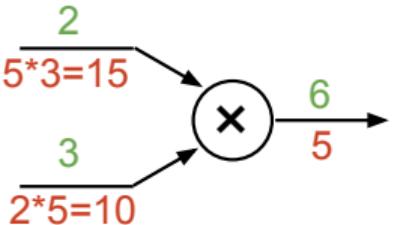
# Nodes as gates



**add** gate: gradient distributor



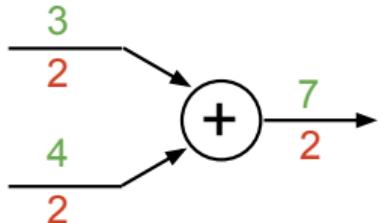
**mul** gate: “swap multiplier”



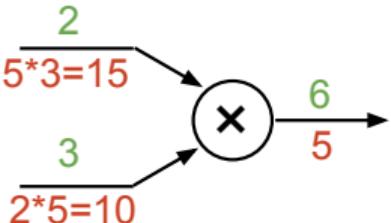
# Nodes as gates



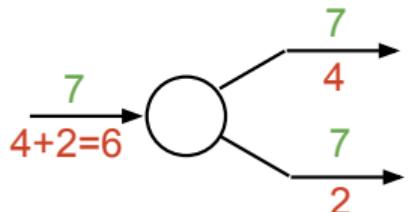
**add** gate: gradient distributor



**mul** gate: “swap multiplier”



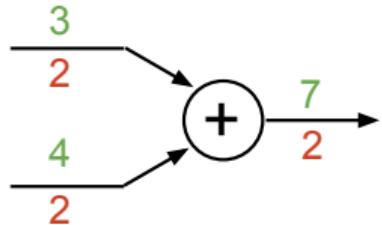
**copy** gate: gradient adder



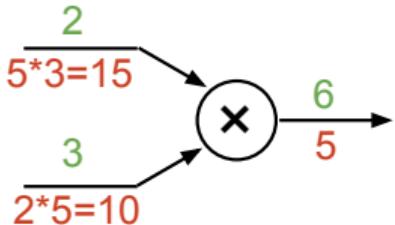
# Nodes as gates



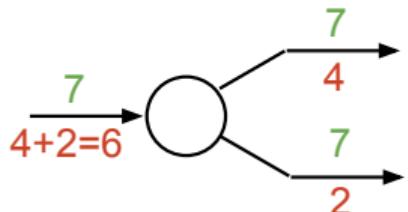
**add** gate: gradient distributor



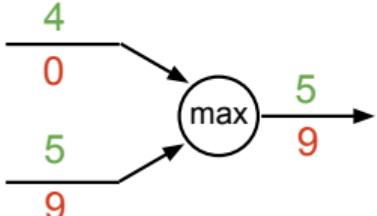
**mul** gate: “swap multiplier”



**copy** gate: gradient adder



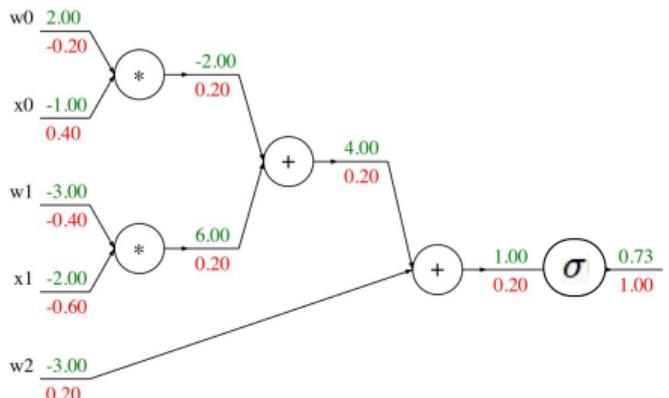
**max** gate: gradient router



# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

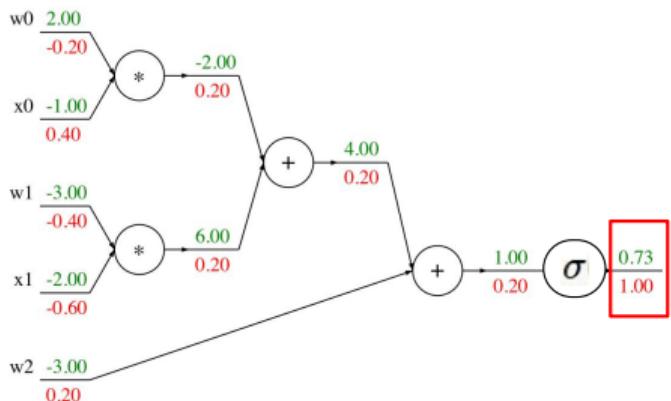
Backward pass:  
Compute grads

```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

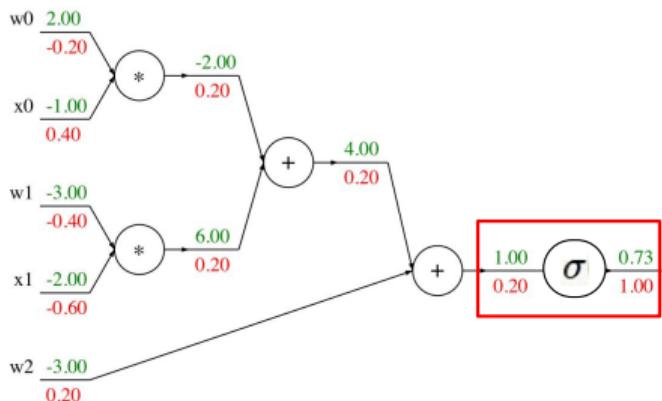
Base case

```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

Sigmoid

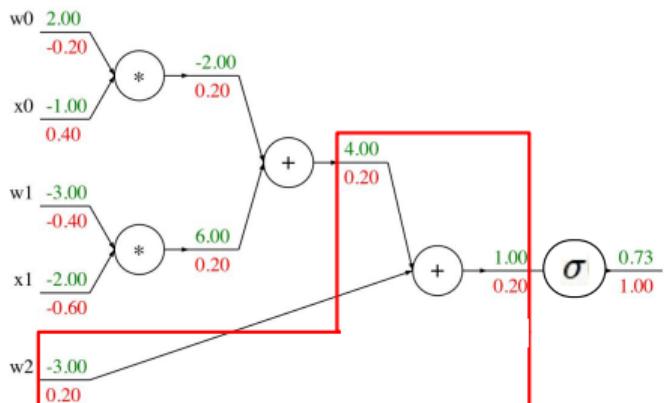
```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

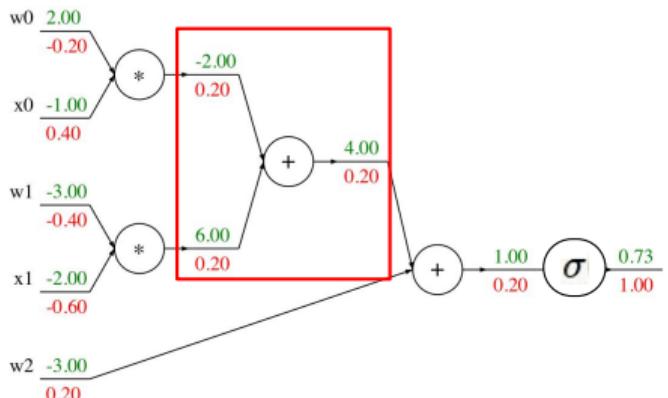
Add gate

```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

Add gate

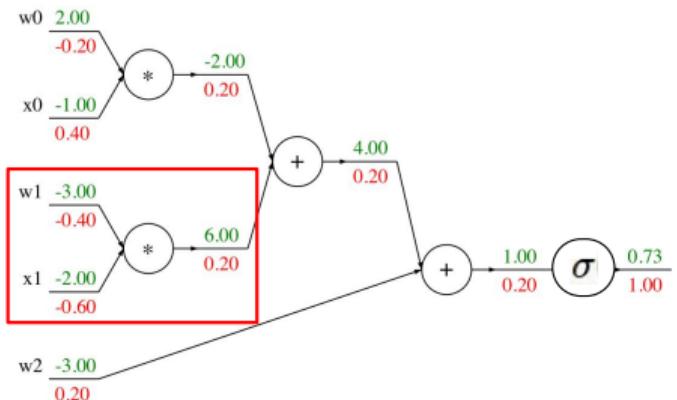
```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

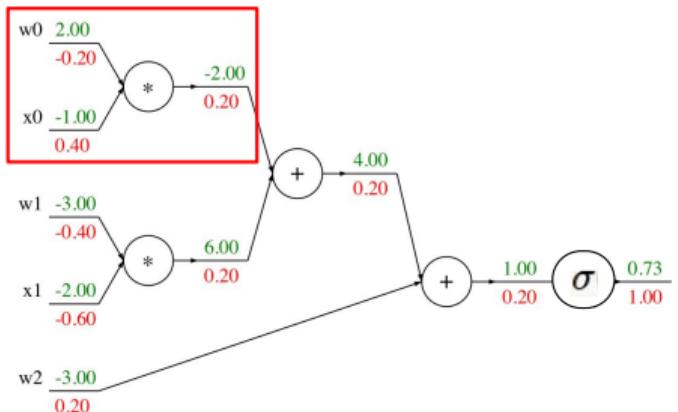
```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

Multiply gate

# Backprop code



## Backprop Implementation: “Flat” code



Forward pass:  
Compute output

```
def f(w0, x0, w1, x1, w2):  
    s0 = w0 * x0  
    s1 = w1 * x1  
    s2 = s0 + s1  
    s3 = s2 + w2  
    L = sigmoid(s3)
```

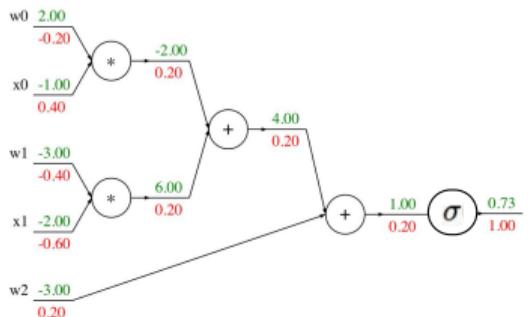
```
grad_L = 1.0  
grad_s3 = grad_L * (1 - L) * L  
grad_w2 = grad_s3  
grad_s2 = grad_s3  
grad_s0 = grad_s2  
grad_s1 = grad_s2  
grad_w1 = grad_s1 * x1  
grad_x1 = grad_s1 * w1  
grad_w0 = grad_s0 * x0  
grad_x0 = grad_s0 * w0
```

Multiply gate

# General code structure

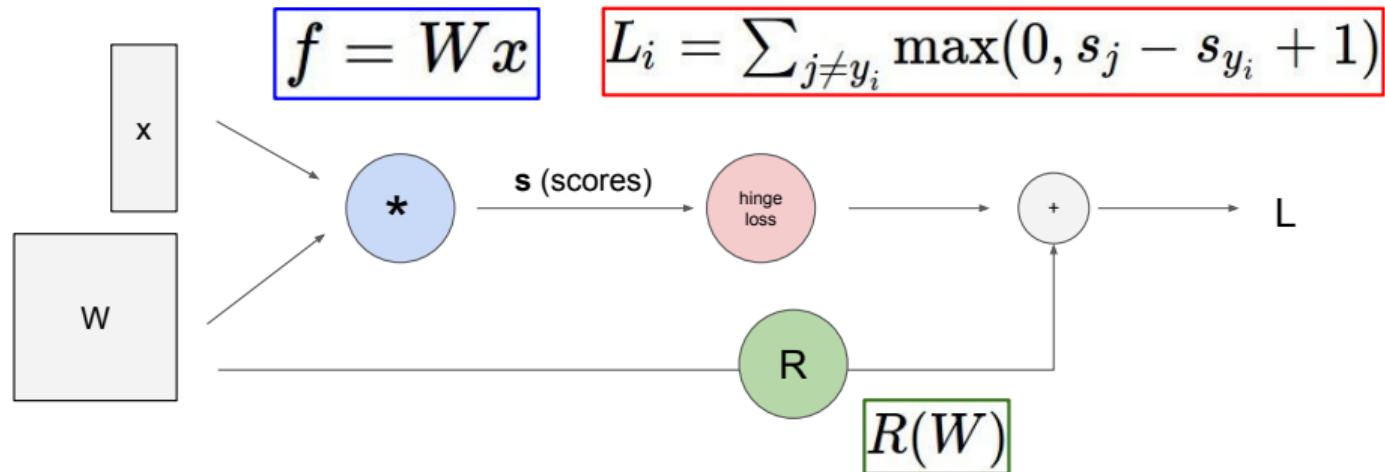


Graph (or Net) object (*rough pseudo code*)



```
class ComputationalGraph(object):  
    ...  
    def forward(inputs):  
        # 1. [pass inputs to input gates...]  
        # 2. forward the computational graph:  
        for gate in self.graph.nodes_topologically_sorted():  
            gate.forward()  
        return loss # the final gate in the graph outputs the loss  
    def backward():  
        for gate in reversed(self.graph.nodes_topologically_sorted()):  
            gate.backward() # little piece of backprop (chain rule applied)  
        return inputs_gradients
```

# Code example - PyTorch



# Vector derivatives



## Scalar to Scalar

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If  $x$  changes by a small amount, how much will  $y$  change?

# Vector derivatives



## Scalar to Scalar

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If  $x$  changes by a small amount, how much will  $y$  change?

## Vector to Scalar

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is **Gradient**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left( \frac{\partial y}{\partial x} \right)_n = \frac{\partial y}{\partial x_n}$$

For each element of  $x$ , if it changes by a small amount then how much will  $y$  change?

# Vector derivatives



## Scalar to Scalar

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If  $x$  changes by a small amount, how much will  $y$  change?

## Vector to Scalar

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is **Gradient**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left( \frac{\partial y}{\partial x} \right)_n = \frac{\partial y}{\partial x_n}$$

For each element of  $x$ , if it changes by a small amount then how much will  $y$  change?

## Vector to Vector

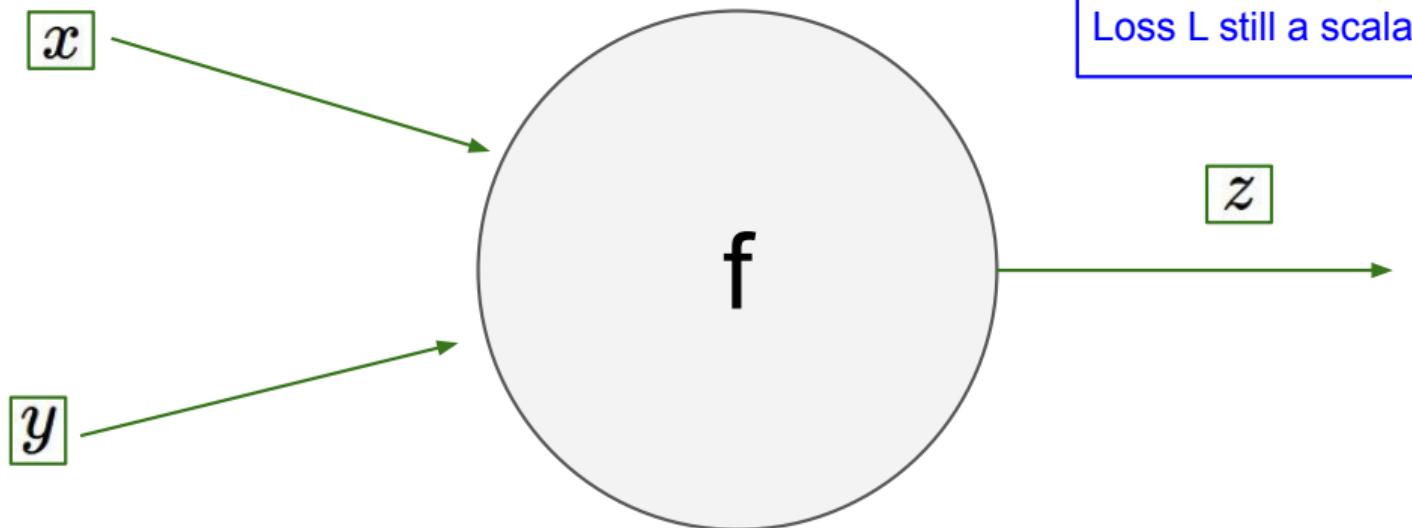
$$x \in \mathbb{R}^N, y \in \mathbb{R}^M$$

Derivative is **Jacobian**:

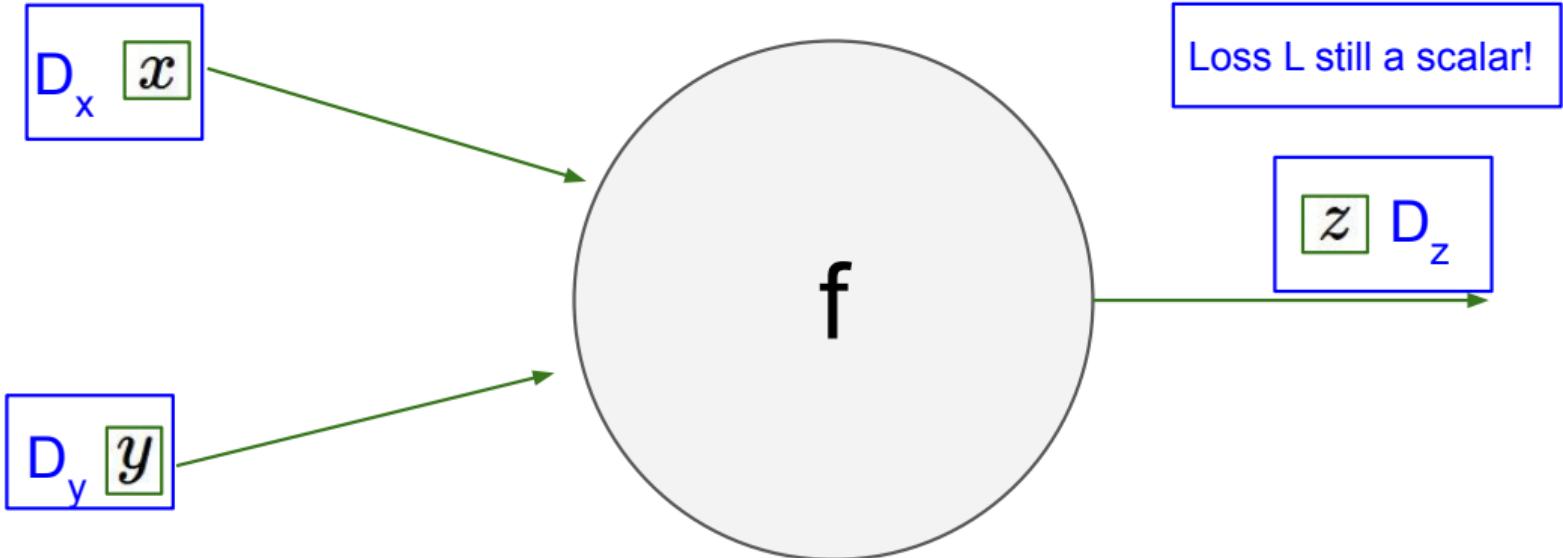
$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \quad \left( \frac{\partial y}{\partial x} \right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

For each element of  $x$ , if it changes by a small amount then how much will each element of  $y$  change?

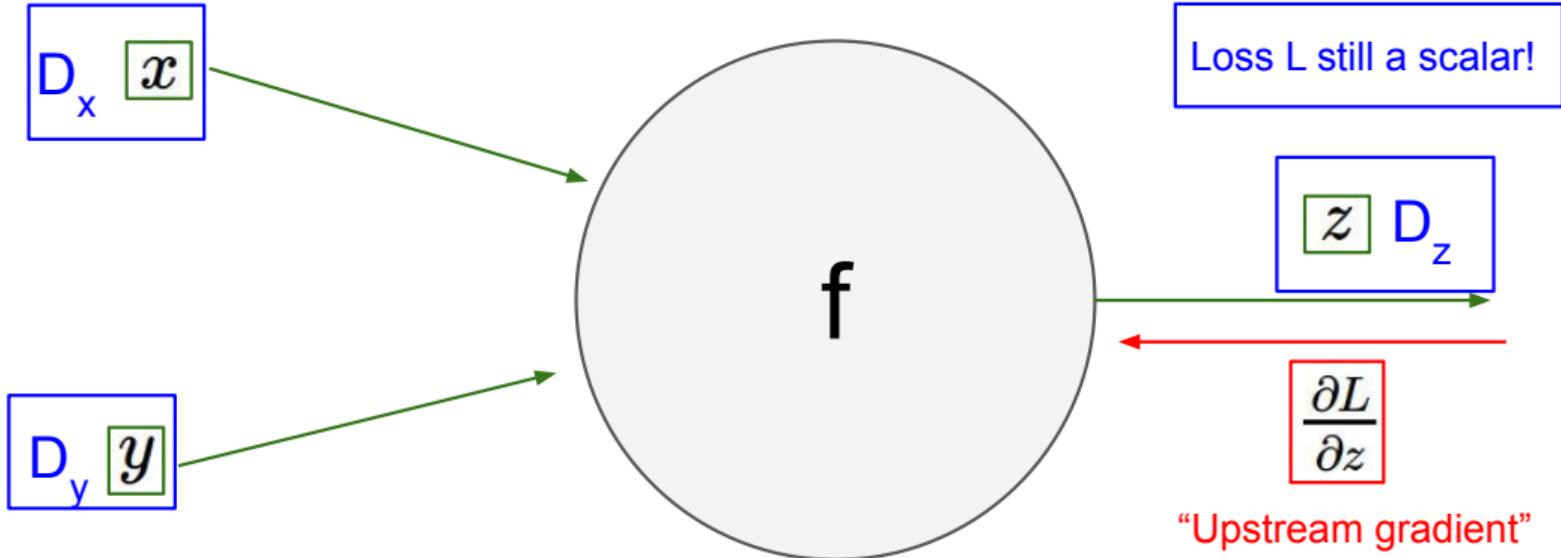
# Vector backprop



# Vector backprop

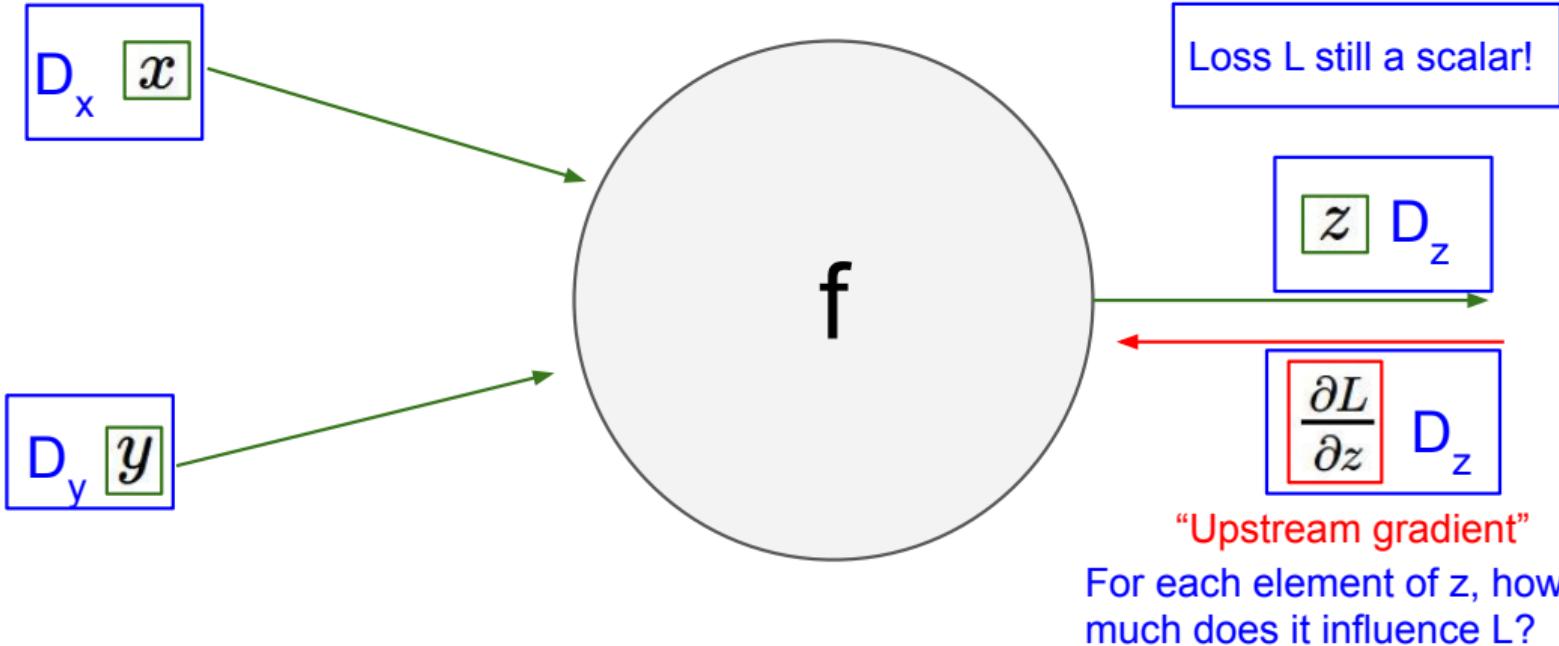


# Vector backprop

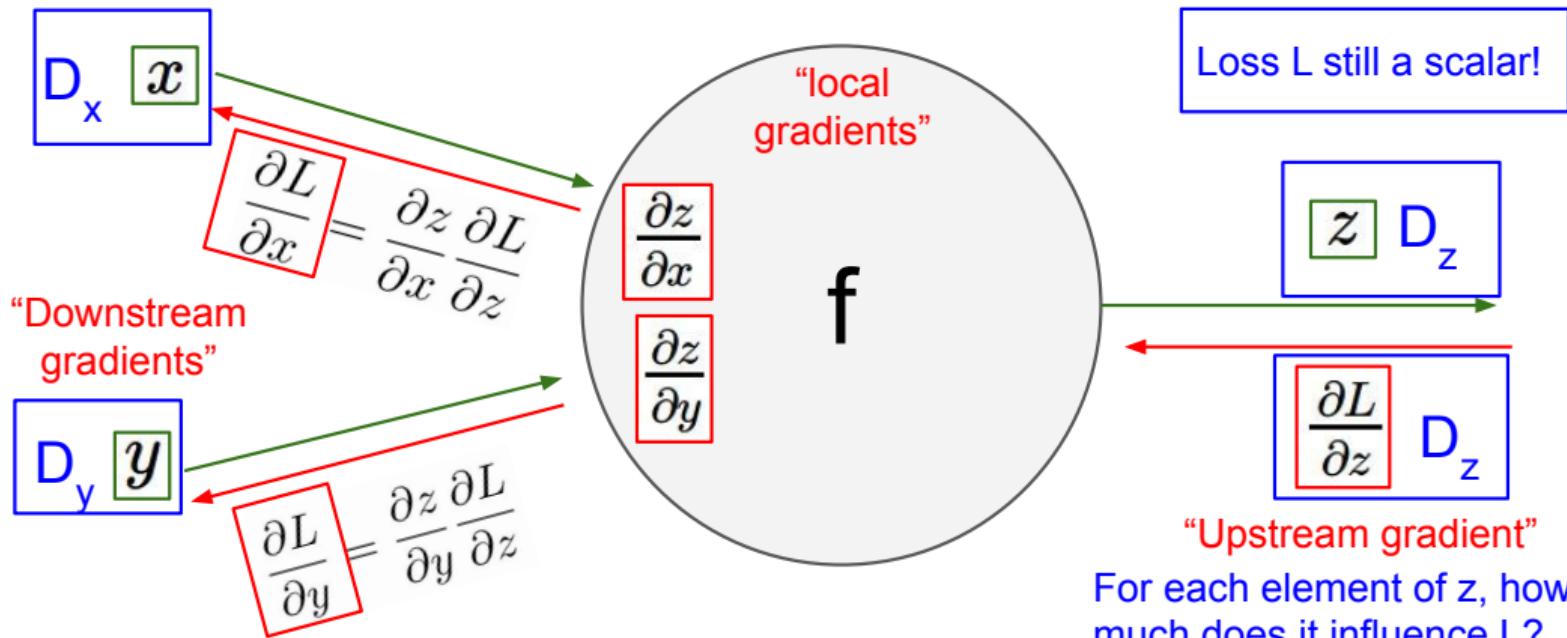


“Upstream gradient”

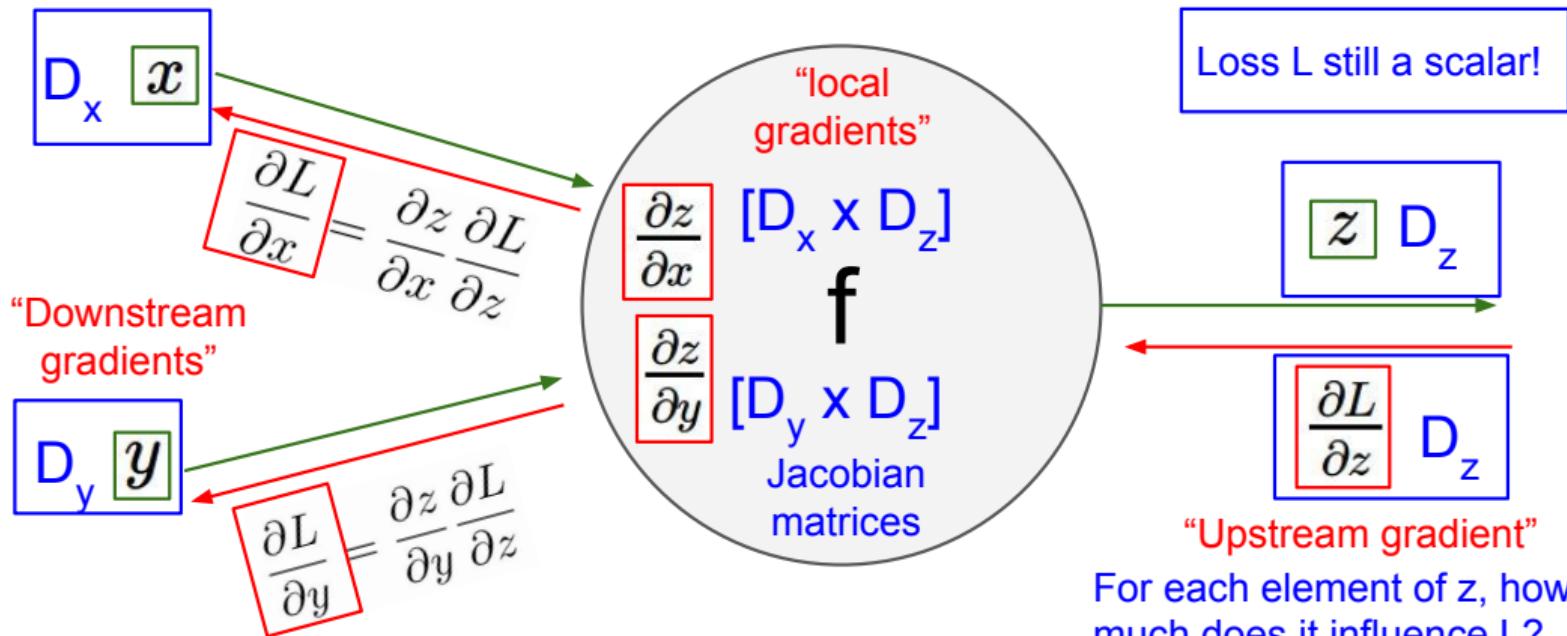
# Vector backprop



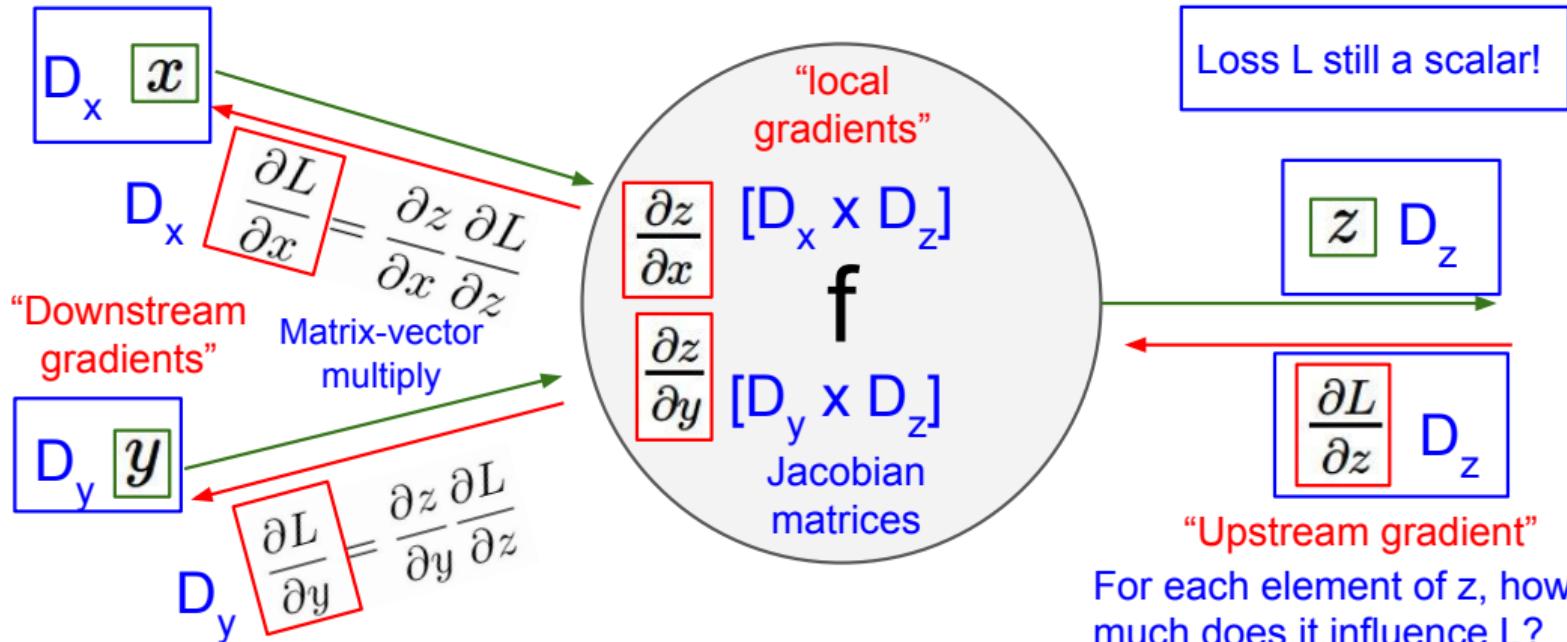
# Vector backprop



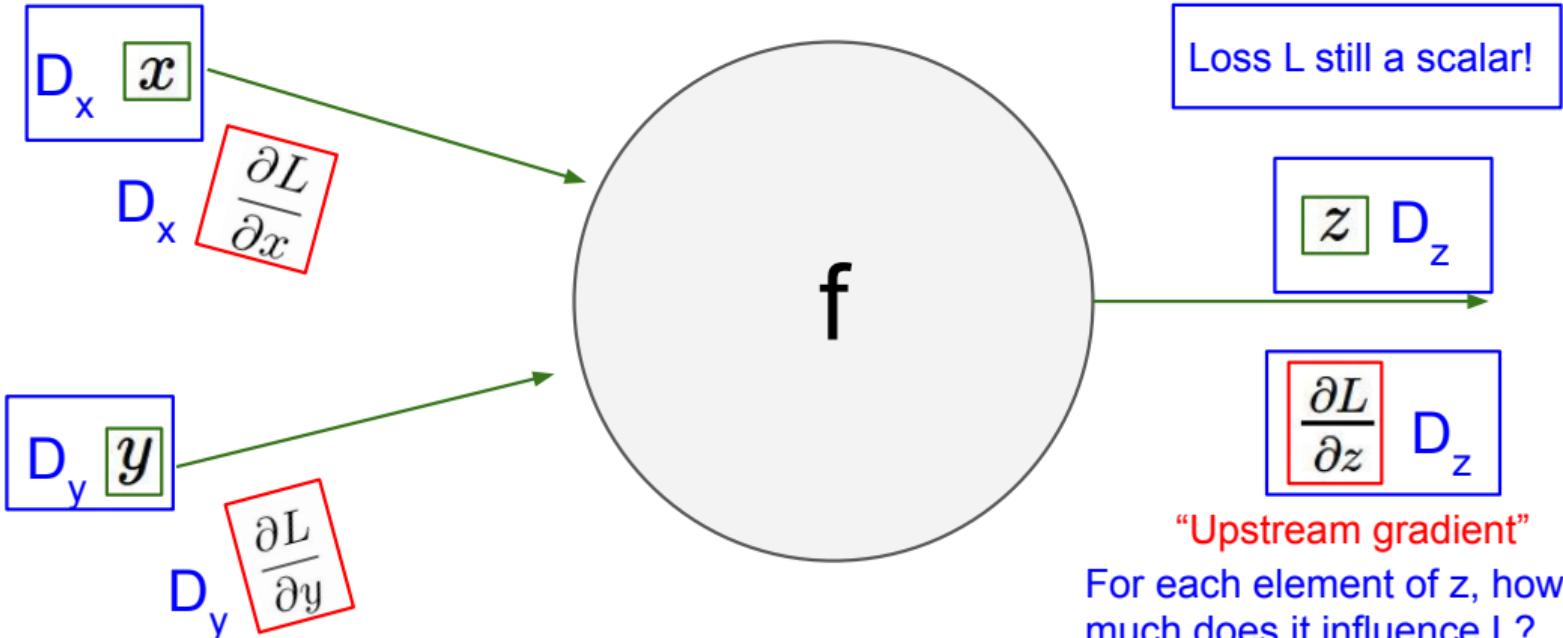
# Vector backprop



# Vector backprop



# Vector backprop



# Vector backprop



4D input x:

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix} \longrightarrow$$

$f(x) = \max(0, x)$   
*(elementwise)*

4D output z:

$$\longrightarrow \begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

# Vector backprop



4D input x:

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix}$$

$$f(x) = \max(0, x)$$

(elementwise)

4D output z:

$$\begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

4D  $dL/dz$ :

$$\begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

Upstream  
gradient

# Vector backprop



4D input x:

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix}$$

$$f(x) = \max(0, x)$$

(elementwise)

4D output z:

$$\begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

Jacobian  $dz/dx$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

4D  $dL/dz$ :

$$\begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

Upstream  
gradient

# Vector backprop



4D input x:

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix}$$

$$f(x) = \max(0, x)$$

(elementwise)

4D output z:

$$\begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

[dz/dx] [dL/dz]

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} [4]$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} [-1]$$
$$\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} [5]$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} [9]$$

4D dL/dz:

$$\begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

Upstream  
gradient

# Vector backprop



4D input x:

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix}$$

$$f(x) = \max(0, x)$$

(elementwise)

4D output z:

$$\begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

4D  $dL/dx$ :

$$\begin{bmatrix} 4 \\ 0 \\ 5 \\ 0 \end{bmatrix}$$

$[dz/dx]$   $[dL/dz]$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

4D  $dL/dz$ :

$$\begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

Upstream  
gradient

# Vector backprop



Jacobian is **sparse**:  
off-diagonal entries  
always zero! Never  
**explicitly** form  
Jacobian -- instead  
use **implicit**  
multiplication

4D input  $x$ :

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix}$$

$$f(x) = \max(0, x)$$

(elementwise)

4D output  $z$ :

$$\begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

4D  $dL/dx$ :

$$\begin{bmatrix} 4 \\ 0 \\ 5 \\ 0 \end{bmatrix}$$

$$[dz/dx] [dL/dz]$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

4D  $dL/dz$ :

$$\begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix}$$

Upstream  
gradient

# Vector backprop



Jacobian is **sparse**:  
off-diagonal entries  
always zero! Never  
**explicitly** form  
Jacobian -- instead  
use **implicit**  
multiplication

4D input x:

$$\begin{bmatrix} 1 \\ -2 \\ 3 \\ -1 \end{bmatrix}$$

$$f(x) = \max(0, x)$$

(elementwise)

4D output z:

$$\begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$$

4D  $dL/dx$ :

$$\begin{bmatrix} 4 \\ 0 \\ 5 \\ 0 \end{bmatrix} \leftarrow$$

$$\left( \frac{\partial L}{\partial x} \right)_i = \begin{cases} \left( \frac{\partial L}{\partial z} \right)_i & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

[ $dz/dx$ ] [ $dL/dz$ ]

4D  $dL/dz$ :

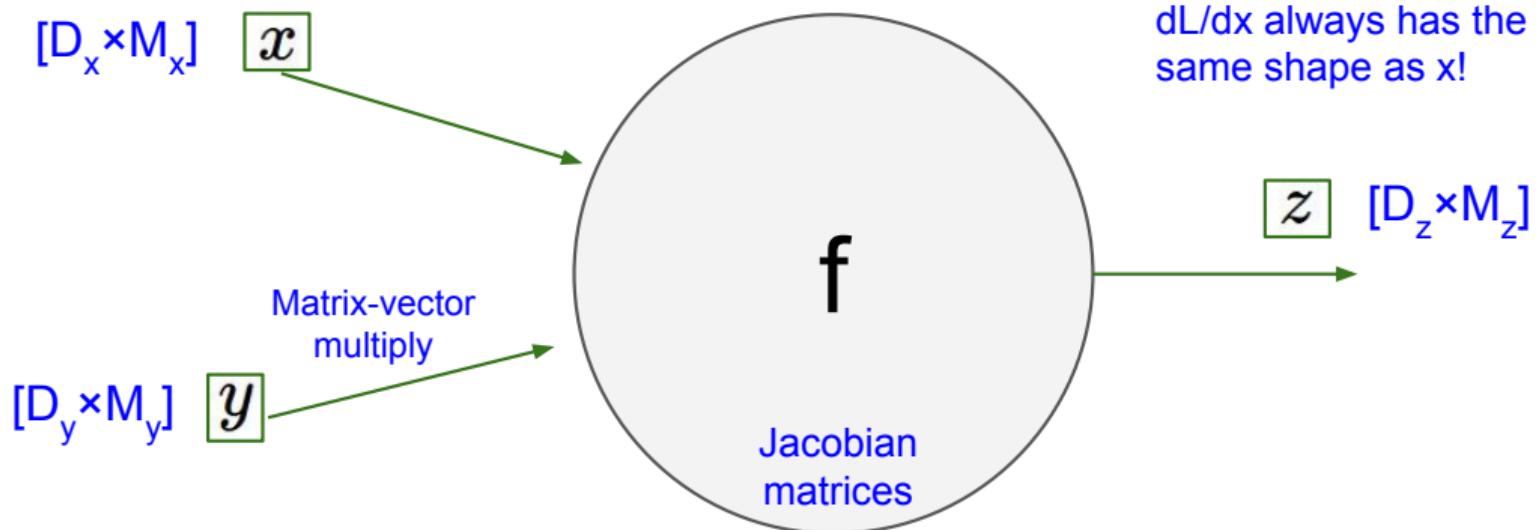
$$\begin{bmatrix} 4 \\ -1 \\ 5 \\ 9 \end{bmatrix} \leftarrow$$

Upstream  
gradient

# Matrix backprop



## Backprop with Matrices (or Tensors)



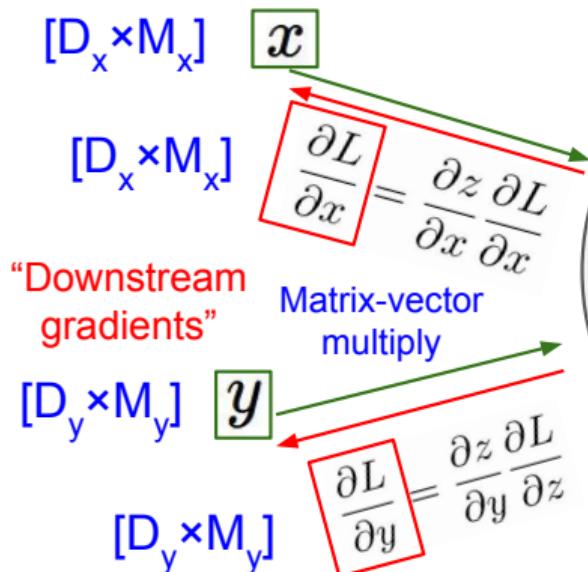
Loss  $L$  still a scalar!

$dL/dx$  always has the same shape as  $x$ !

# Matrix backprop



## Backprop with Matrices (or Tensors)

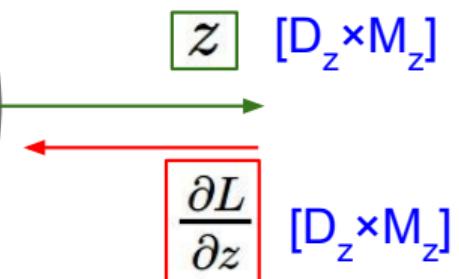


$f$

Jacobian  
matrices

Loss  $L$  still a scalar!

$dL/dx$  always has the same shape as  $x$ !

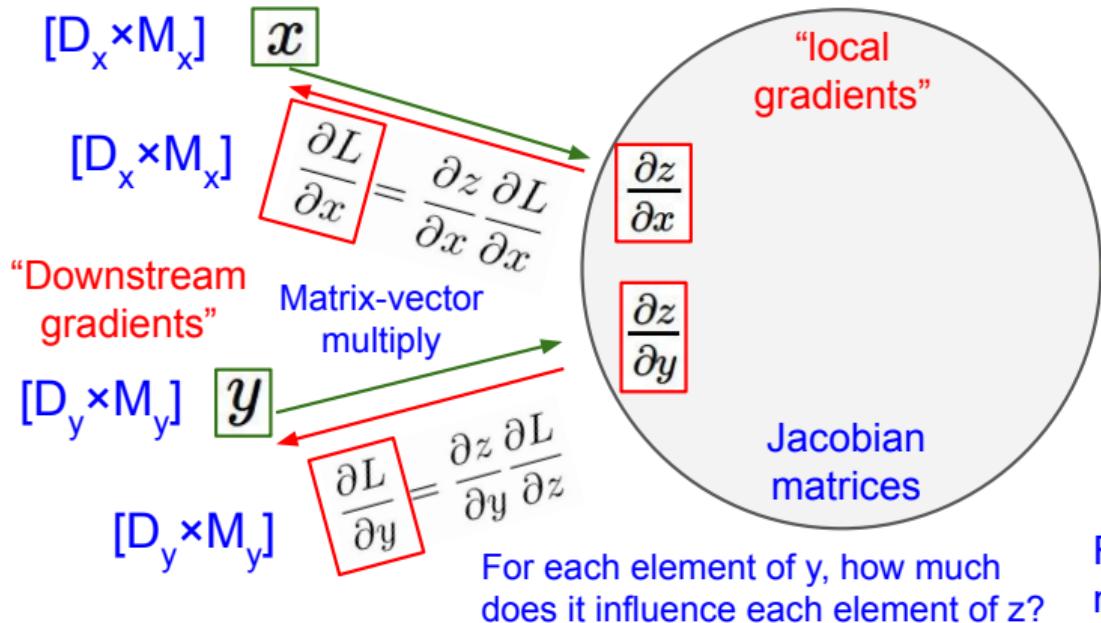


“Upstream gradient”  
For each element of  $z$ , how much does it influence  $L$ ?

# Matrix backprop

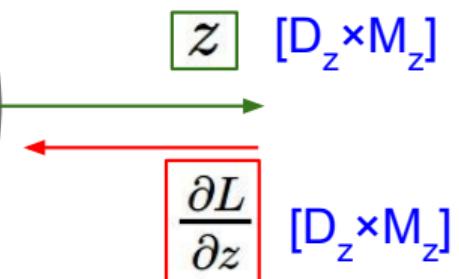


## Backprop with Matrices (or Tensors)



Loss  $L$  still a scalar!

$dL/dx$  always has the same shape as  $x$ !



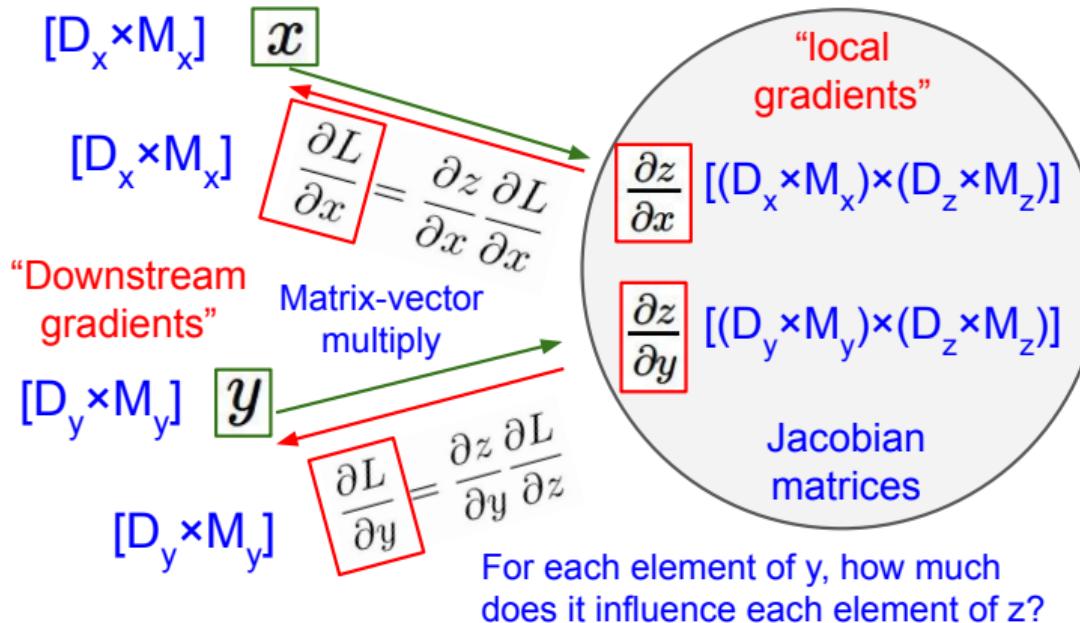
"Upstream gradient"

For each element of  $z$ , how much does it influence  $L$ ?

# Matrix backprop

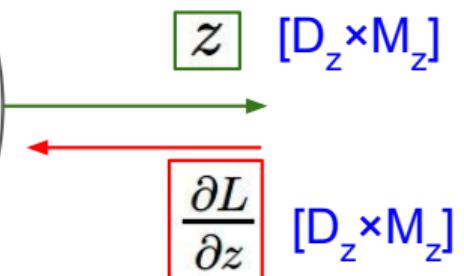


## Backprop with Matrices (or Tensors)



Loss  $L$  still a scalar!

$dL/dx$  always has the same shape as  $x$ !



# Matrix backprop



## Backprop with Matrices

x: [N×D]

$$\begin{bmatrix} 2 & 1 & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

### Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

### Jacobians:

dy/dx: [(N×D)×(N×M)]

dy/dw: [(D×M)×(N×M)]

y: [N×M]

$$\begin{bmatrix} 13 & 9 & -2 & -6 \\ 5 & 2 & 17 & 1 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

For a neural net we may have

N=64, D=M=4096

Each Jacobian takes 256 GB of memory!

Must work with them implicitly!

# Matrix backprop



## Backprop with Matrices

x: [N×D]

$$\begin{bmatrix} 2 & \boxed{1} & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

### Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

**Q:** What parts of y  
are affected by one  
element of x?

y: [N×M]

$$\begin{bmatrix} 13 & 9 & -2 & -6 \\ 5 & 2 & 17 & 1 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

# Matrix backprop



## Backprop with Matrices

x: [N×D]

$$\begin{bmatrix} 2 & \boxed{1} & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

### Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

**Q:** What parts of y  
are affected by one  
element of x?

**A:**  $x_{n,d}$  affects the  
whole row  $y_n$ .

$$\frac{\partial L}{\partial x_{n,d}} = \sum_m \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}}$$

y: [N×M]

$$\begin{bmatrix} 13 & 9 & -2 & -6 \\ 5 & 2 & 17 & 1 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

# Matrix backprop



## Backprop with Matrices

x: [N×D]

$$\begin{bmatrix} 2 & \boxed{1} & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

### Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

**Q:** What parts of y  
are affected by one  
element of x?

**A:**  $x_{n,d}$  affects the  
whole row  $y_{n,:}$

y: [N×M]

$$\begin{bmatrix} \textcolor{green}{13} & 9 & \boxed{-2} & -6 \\ 5 & 2 & 17 & 1 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} \textcolor{red}{2} & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

**Q:** How much  
does  $x_{n,d}$   
affect  $y_{n,m}$ ?

$$\frac{\partial L}{\partial x_{n,d}} = \sum_m \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}}$$

# Matrix backprop



## Backprop with Matrices

x: [N×D]

$$\begin{bmatrix} 2 & \boxed{1} & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & \boxed{3} & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

### Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

**Q:** What parts of y  
are affected by one  
element of x?

**A:**  $x_{n,d}$  affects the  
whole row  $y_{n,:}$

**Q:** How much  
does  $x_{n,d}$   
affect  $y_{n,m}$ ?

**A:**  $w_{d,m}$

y: [N×M]

$$\begin{bmatrix} 13 & 9 & \boxed{-2} & -6 \\ 5 & 2 & 17 & 1 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

$$\frac{\partial L}{\partial x_{n,d}} = \sum_m \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}} = \sum_m \frac{\partial L}{\partial y_{n,m}} w_{d,m}$$

# Matrix backprop



## Backprop with Matrices

$$\begin{array}{l} x: [N \times D] \\ \left[ \begin{array}{ccc} 2 & 1 & -3 \\ -3 & 4 & 2 \end{array} \right] \\ w: [D \times M] \\ \left[ \begin{array}{cccc} 3 & 2 & 1 & -1 \\ 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & -2 \end{array} \right] \\ [N \times D] \quad [N \times M] \quad [M \times D] \end{array}$$

Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

$$y: [N \times M]$$

13	9	-2	-6
5	2	17	1

$$dL/dy: [N \times M]$$

2	3	-3	9
-8	1	4	6

**Q:** What parts of  $y$  are affected by one element of  $x$ ?

**A:**  $x_{n,d}$  affects the whole row  $y_{n,:}$

**Q:** How much does  $x_{n,d}$  affect  $y_{n,m}$ ?

**A:**  $w_{d,m}$

$$\frac{\partial L}{\partial x} = \left( \frac{\partial L}{\partial y} \right) w^T$$

$$\frac{\partial L}{\partial x_{n,d}} = \sum_m \frac{\partial L}{\partial y_{n,m}} \frac{\partial y_{n,m}}{\partial x_{n,d}} = \sum_m \frac{\partial L}{\partial y_{n,m}} w_{d,m}$$

# Matrix backprop



## Backprop with Matrices

x: [N×D]

$$\begin{bmatrix} 2 & \boxed{1} & -3 \\ -3 & 4 & 2 \end{bmatrix}$$

w: [D×M]

$$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & \boxed{3} & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$$

[N×D] [N×M] [M×D]

$$\frac{\partial L}{\partial x} = \left( \frac{\partial L}{\partial y} \right) w^T$$

### Matrix Multiply

$$y_{n,m} = \sum_d x_{n,d} w_{d,m}$$

By similar logic:

[D×M] [D×N] [N×M]

$$\frac{\partial L}{\partial w} = x^T \left( \frac{\partial L}{\partial y} \right)$$

y: [N×M]

$$\begin{bmatrix} 13 & 9 & \boxed{-2} & -6 \\ 5 & 2 & 17 & 1 \end{bmatrix}$$

dL/dy: [N×M]

$$\begin{bmatrix} 2 & 3 & -3 & 9 \\ -8 & 1 & 4 & 6 \end{bmatrix}$$

These formulas are easy to remember: they are the only way to make shapes match up!

## Vector backprop - example



A vectorized example:  $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^n (W \cdot x)_i^2$

## Vector backprop - example



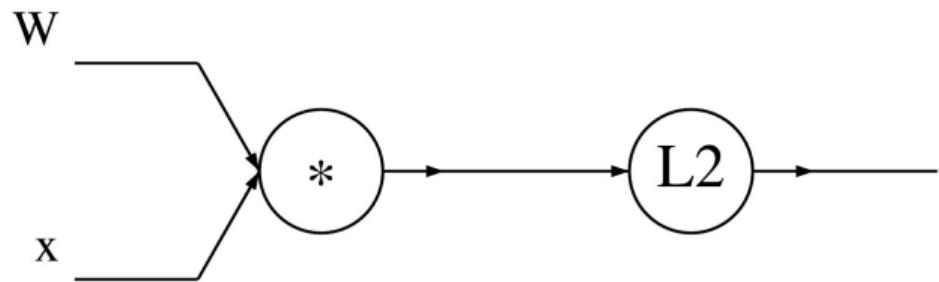
A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{matrix} \downarrow & \downarrow \\ \in \mathbb{R}^n & \in \mathbb{R}^{n \times n} \end{matrix}$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

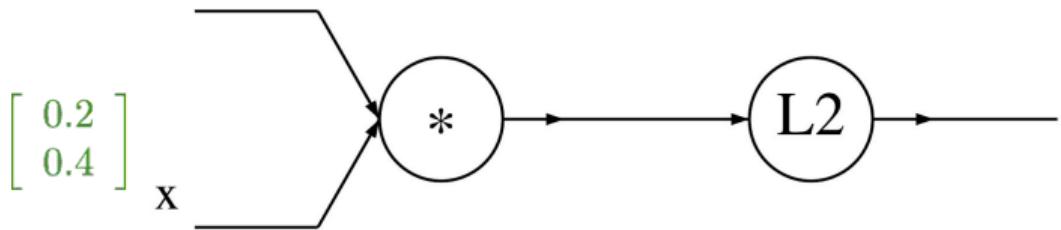


# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix}_W$$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

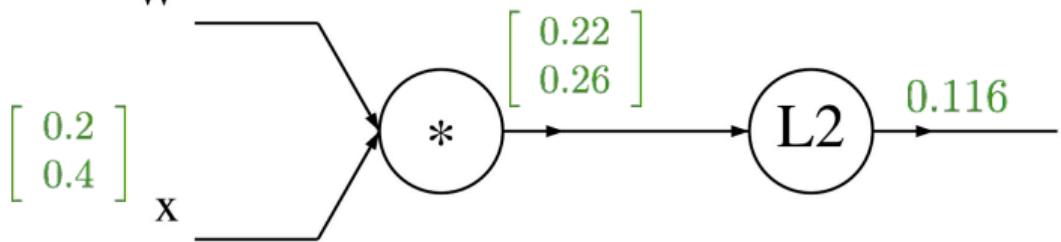
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix}_W$$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

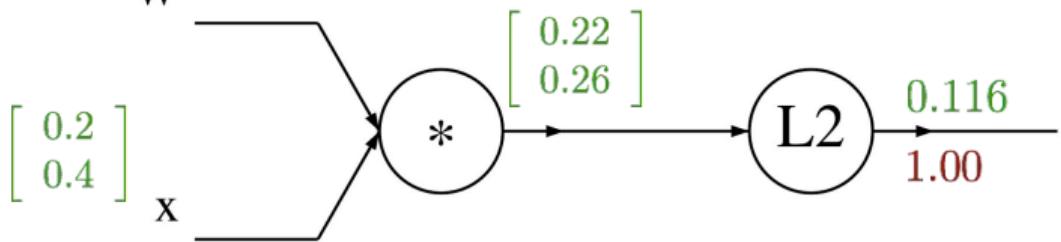
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix} W$$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

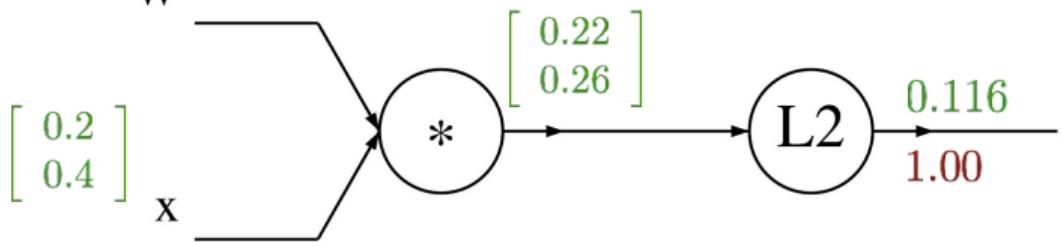
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix} W$$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

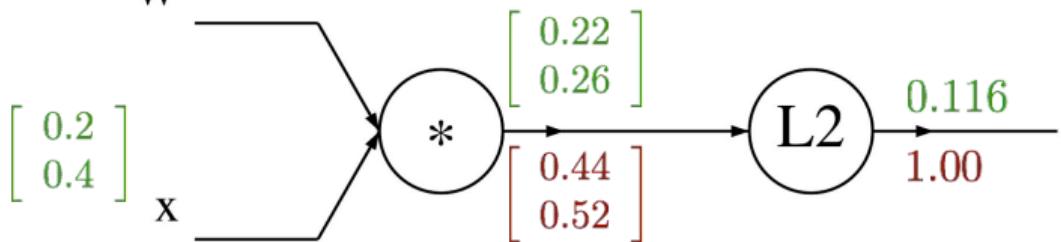
$$\boxed{\nabla_q f = 2q}$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix} W$$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

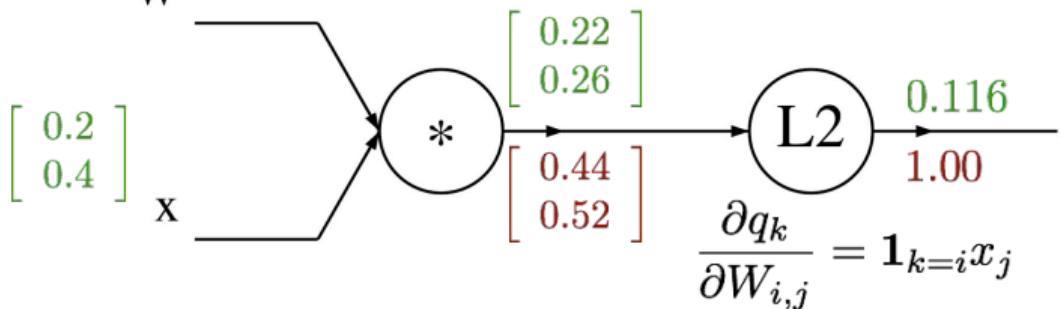
$$\boxed{\nabla_q f = 2q}$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix}_W$$



$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

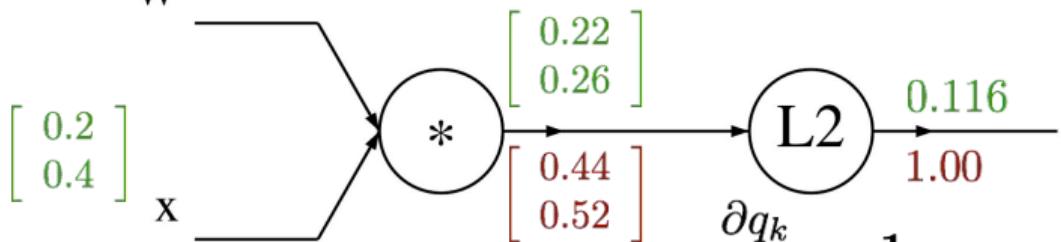
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix}_W$$



$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

$$\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}}$$

$$= \sum_k (2q_k)(\mathbf{1}_{k=i} x_j)$$

$$= 2q_i x_j$$

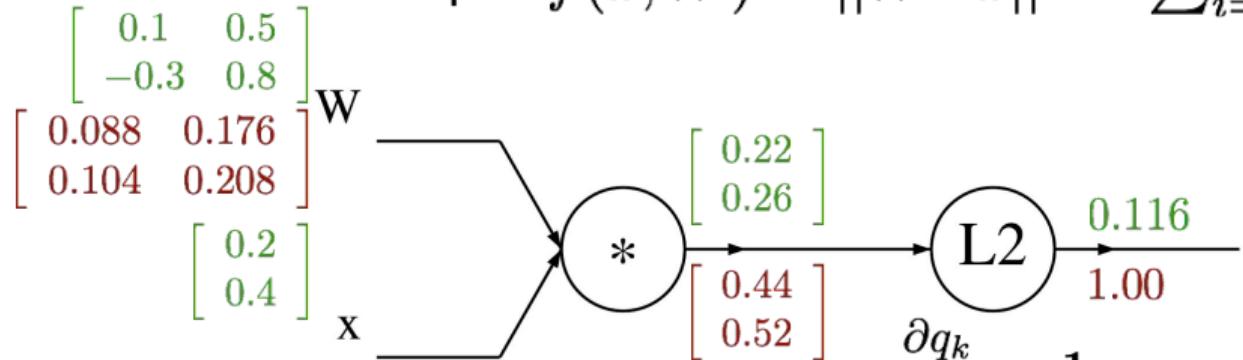
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

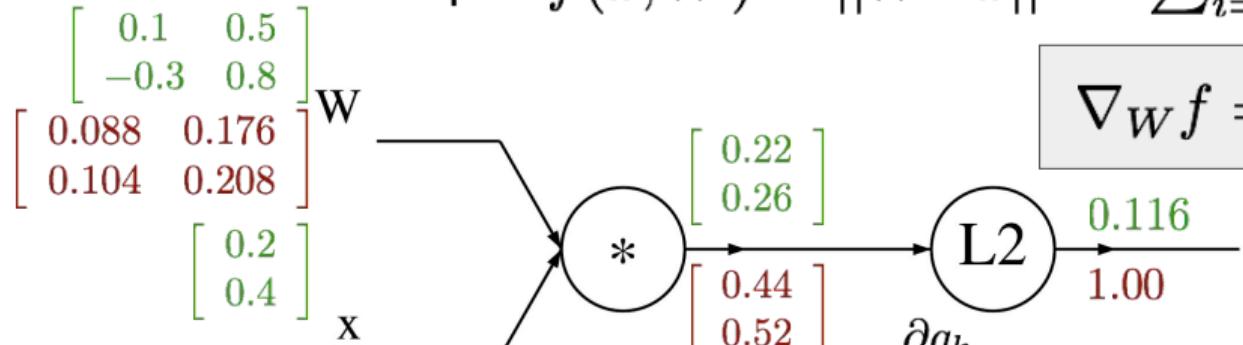
$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i}x_j$$

$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k (2q_k)(\mathbf{1}_{k=i}x_j) \\ &= 2q_i x_j \end{aligned}$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k (2q_k)(\mathbf{1}_{k=i} x_j) \\ &= 2q_i x_j \end{aligned}$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \\ 0.088 & 0.176 \\ 0.104 & 0.208 \end{bmatrix}^W \quad \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}^x$$

$$* \quad \begin{bmatrix} 0.22 \\ 0.26 \\ 0.44 \\ 0.52 \end{bmatrix}$$

$$\nabla_W f = 2q \cdot x^T$$

Always check: The gradient with respect to a variable should have the same shape as the variable

$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k (2q_k)(\mathbf{1}_{k=i} x_j) \\ &= 2q_i x_j \end{aligned}$$

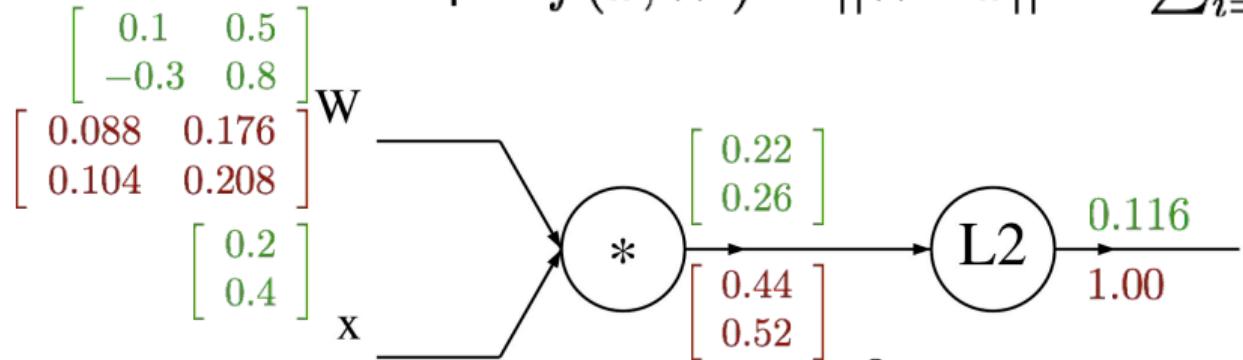
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$



$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

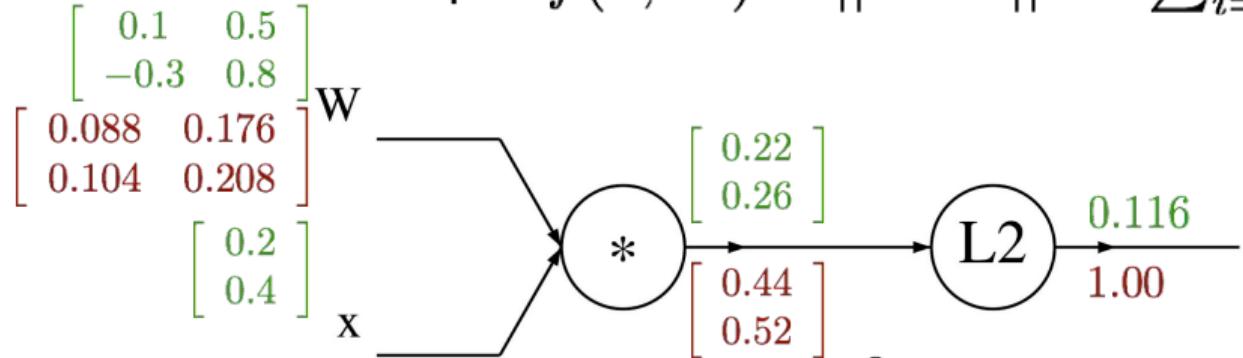
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

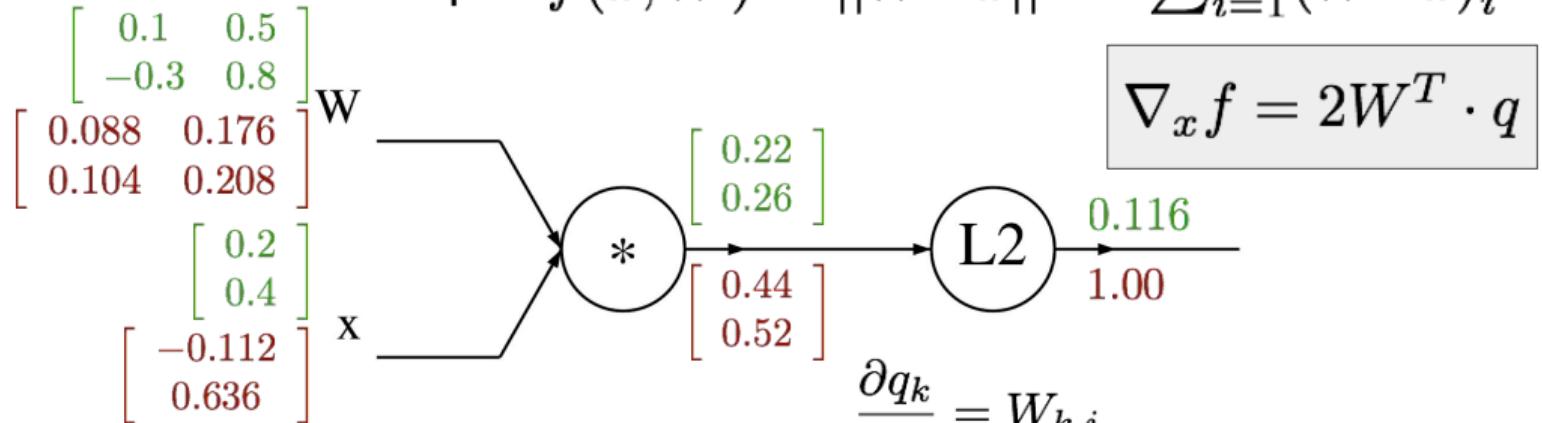
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{aligned}\frac{\partial q_k}{\partial x_i} &= W_{k,i} \\ \frac{\partial f}{\partial x_i} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} \\ &= \sum_k 2q_k W_{k,i}\end{aligned}$$

# Vector backprop - example



A vectorized example:  $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$



$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{aligned}\frac{\partial q_k}{\partial x_i} &= W_{k,i} \\ \frac{\partial f}{\partial x_i} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} \\ &= \sum_k 2q_k W_{k,i}\end{aligned}$$

# Vector backprop - another example



$$z_1 = XW_1$$

$$h_1 = \text{ReLU}(z_1)$$

$$\hat{y} = h_1 W_2$$

$$L = \|\hat{y}\|_2^2$$

$$\frac{\partial L}{\partial W_2} = ?$$

$$\frac{\partial L}{\partial W_1} = ?$$

