Topics:

- Backpropagation
- Matrix/Linear Algebra view

CS 4644-DL / 7643-A

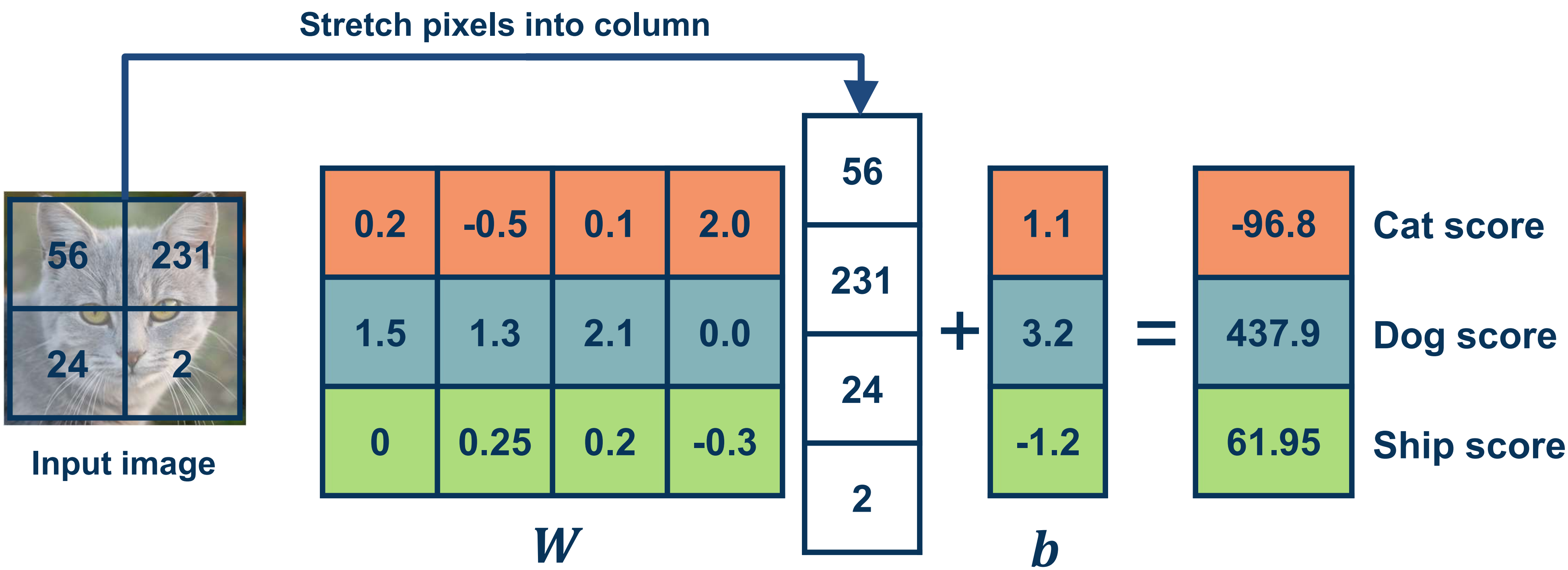
ZSOLT KIRA

- **Assignment 1 out!**
  - **Due Feb 3rd (with grace period 5th)**
  - Start now, start now, start now!
  - Start now, start now, start now!
  - Start now, start now, start now!

- Resources:
  - These lectures
  - Matrix calculus for deep learning
  - Gradients notes and MLP/ReLU Jacobian notes.
  - Assignment 1 (@67) and matrix calculus (@86), convex optimization (@89)

- Piazza: Project teaming thread
  - Project proposal overview during my OH (Thursday 3pm ET)

Example with an image with **4 pixels**, and **3 classes** (cat/dog/ship)



Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Derivatives

- We can find the steepest descent direction by computing the **derivative (gradient):**

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

- Steepest descent direction is the **negative gradient**
- **Intuitively:** Measures how the function changes as the argument a changes by a small step size
  - As step size goes to zero
- **In Machine Learning:** Want to know how the **loss function** changes **as weights** are varied
  - Can consider each parameter separately by taking **partial derivative** of loss function with respect to that parameter

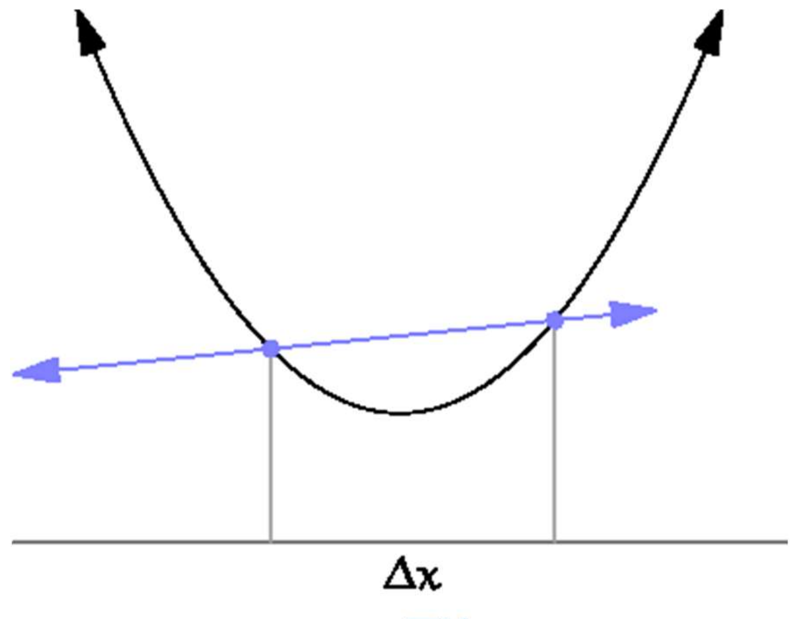
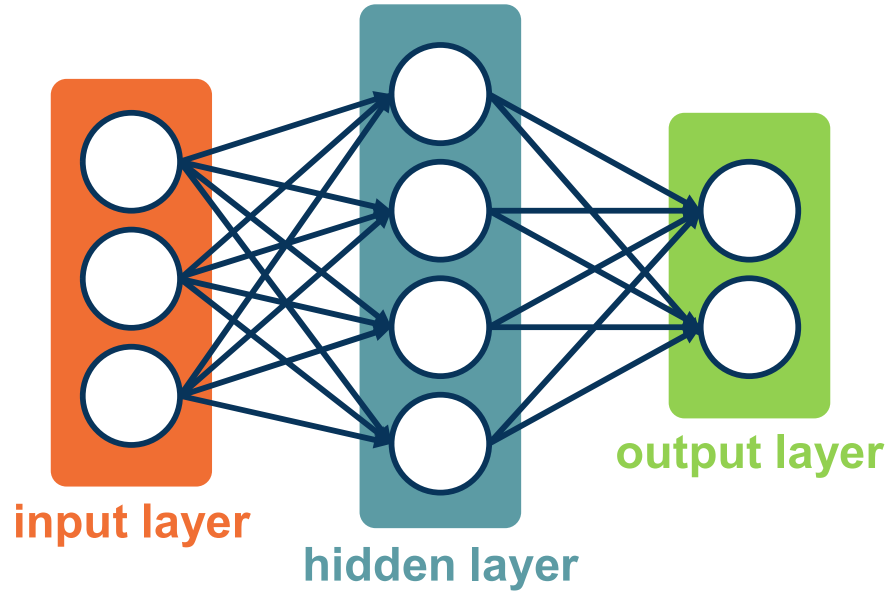


Image and equation from: https://en.wikipedia.org/wiki/Derivative#/media/File:Tangent_animation.gif

The same two-layered neural network **corresponds to adding another weight matrix**

- We will prefer the linear algebra view, but use some terminology from neural networks (& biology)



x W_1 W_2

$$=$$

$$f(x, W_1, W_2) = \sigma(W_2 \sigma(W_1 x))$$

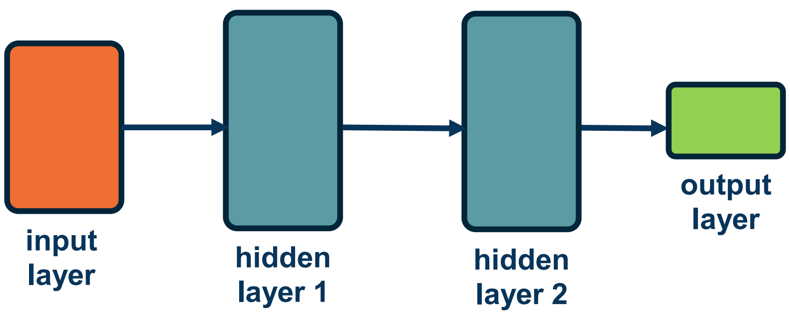
Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

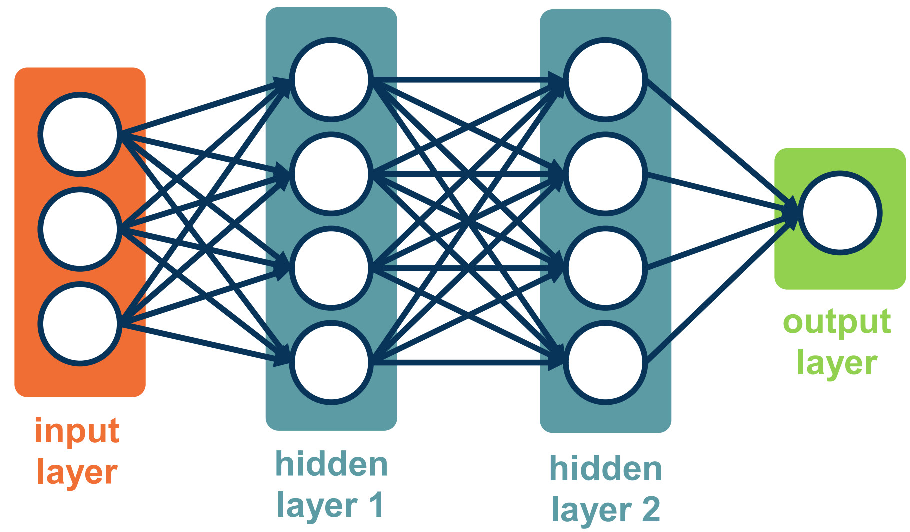
Large (deep) networks can be built by adding more and more layers

Three-layered neural networks can represent **any function**

- The number of nodes could grow unreasonably (exponential or worse) with respect to the complexity of the function

We will show them **without edges**:





$$f(x, W_1, W_2, W_3) = \sigma(W_2\sigma(W_1x))$$

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Adding More Layers!

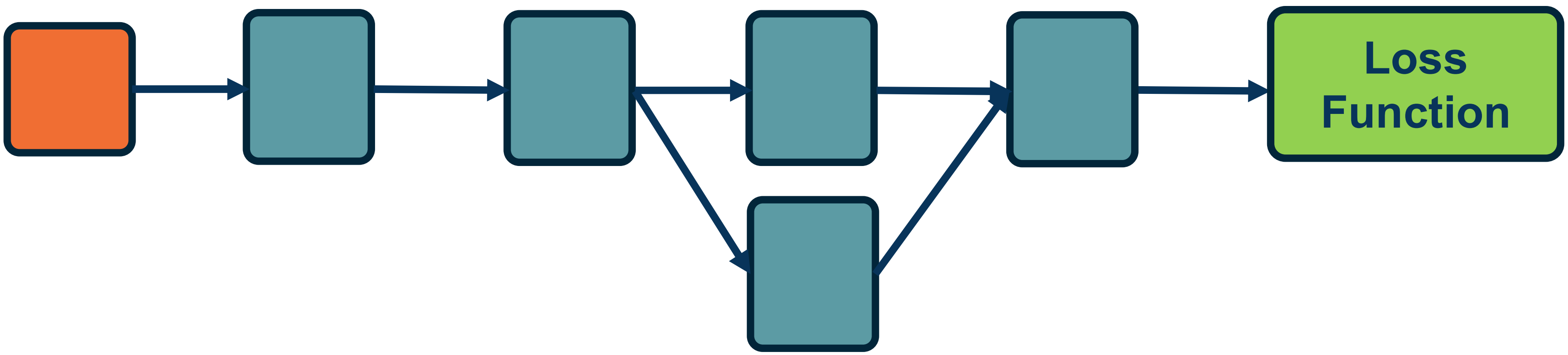
Functions can be made **arbitrarily complex** (subject to memory and computational limits), e.g.:

$$f(x, W) = \sigma(W_5\sigma(W_4\sigma(W_3\sigma(W_2\sigma(W_1x))$$

We can use **any type of differentiable function (layer)** we want!

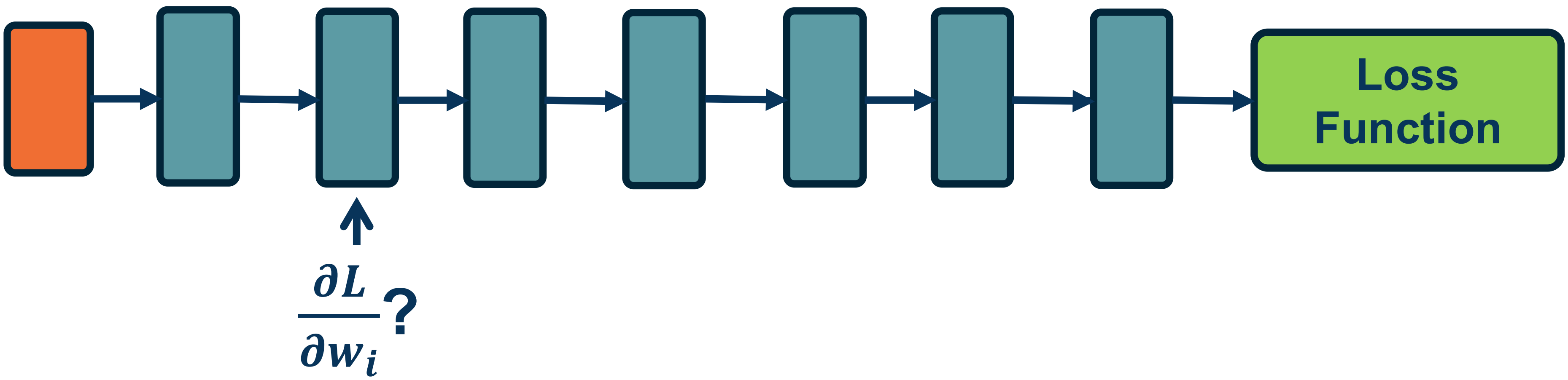
- At the end, **add the loss function**

Composition can have **some structure**



Adding Even More Layers

- We are learning **complex models** with significant amount of parameters (millions or billions)
- How do we compute the gradients of the **loss** (at the end) with respect to **internal** parameters?
- Intuitively, want to understand how **small changes** in weight deep inside **are propagated** to affect the **loss function** at the end



Computing Gradients in Complex Function

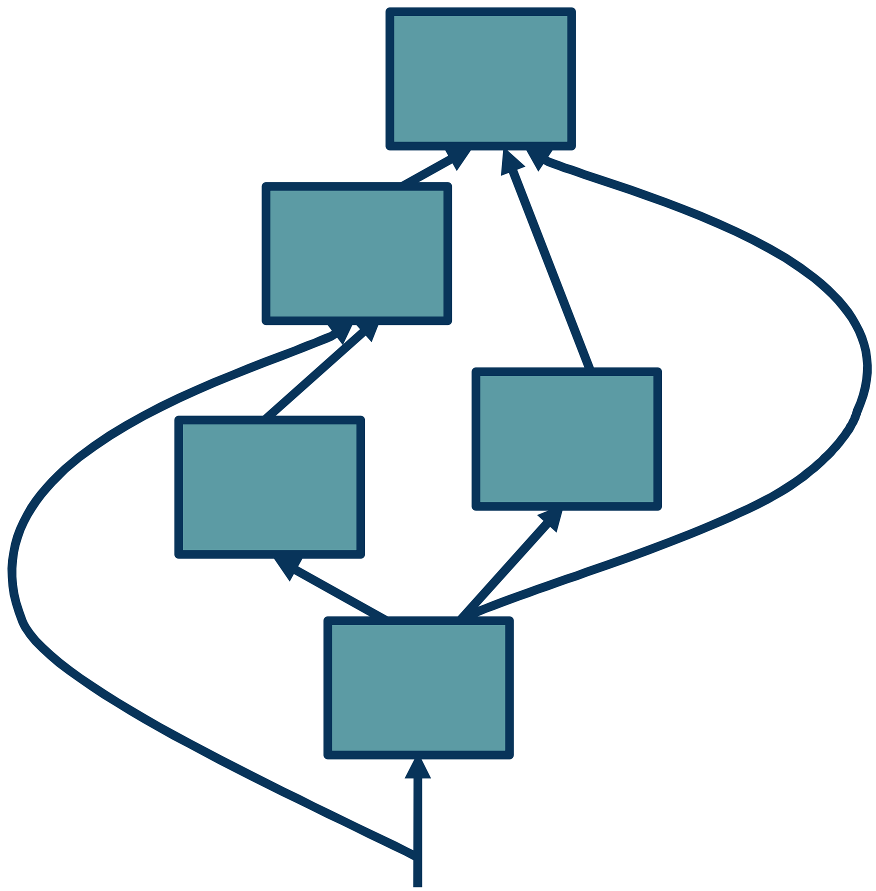
Georgia Tech

To develop a general algorithm for this, we will view the function as a **computation graph**

Graph can be any **directed acyclic graph (DAG)**

- Modules must be differentiable to support gradient computations for gradient descent

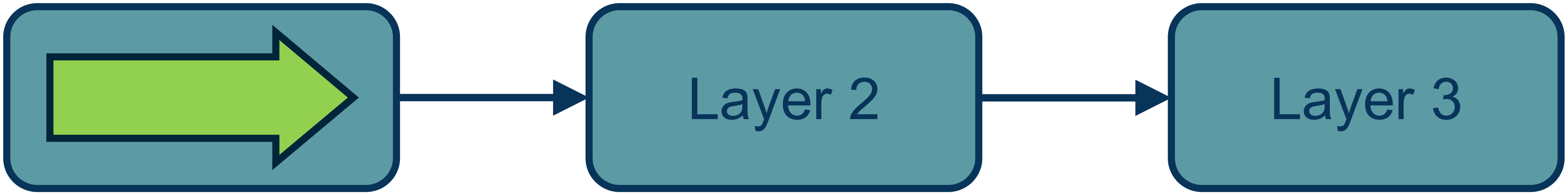
A **training algorithm** will then process this graph, **one module at a time**



Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

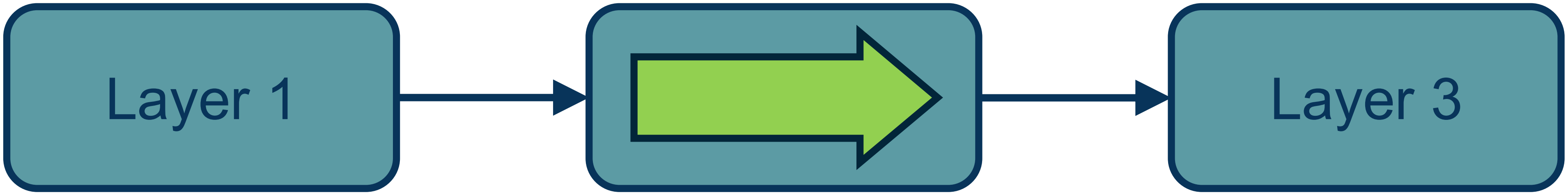
A General Framework

Step 1: Compute Loss on Mini-Batch: **Forward Pass**



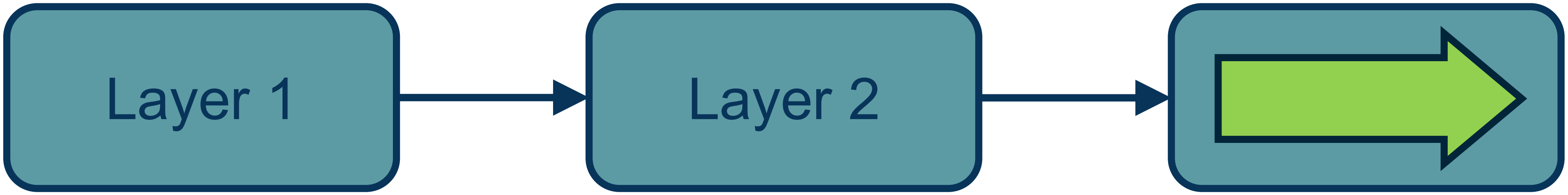
Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

Step 1: Compute Loss on Mini-Batch: **Forward Pass**



Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

Step 1: Compute Loss on Mini-Batch: **Forward Pass**



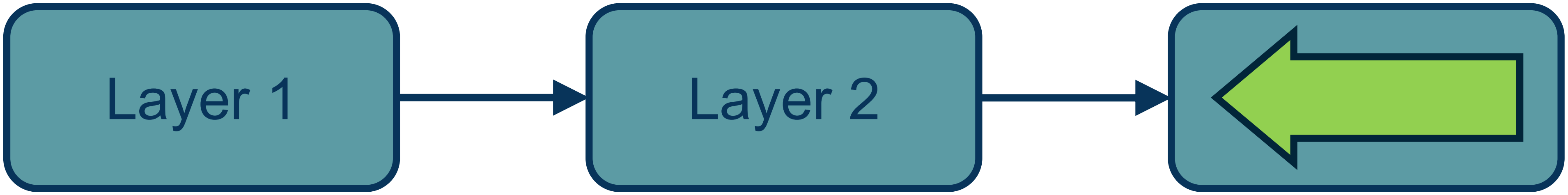
Note that we must store the **intermediate outputs of all layers**!

- This is because we will need them to **compute the gradients** (the gradient equations will have terms with the output values in them)

Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

Step 1: Compute Loss on Mini-Batch: **Forward Pass**

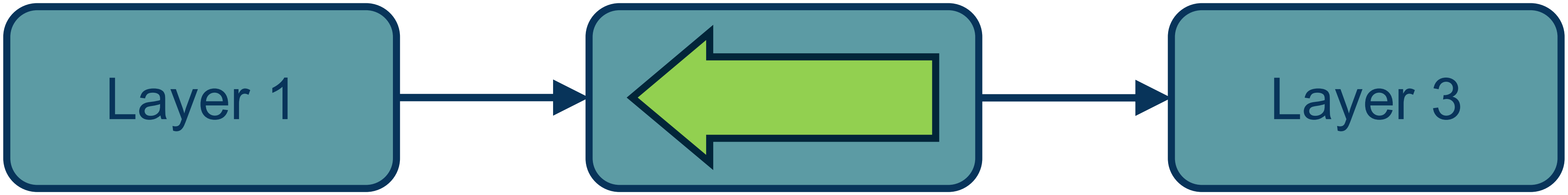
Step 2: Compute Gradients wrt parameters: **Backward Pass**



Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

Step 1: Compute Loss on Mini-Batch: **Forward Pass**

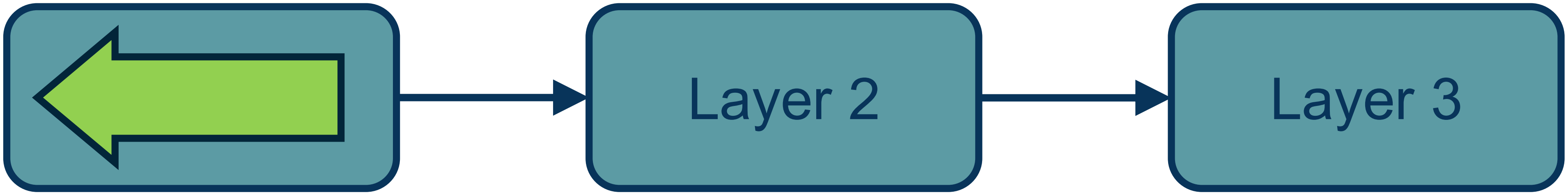
Step 2: Compute Gradients wrt parameters: **Backward Pass**



Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

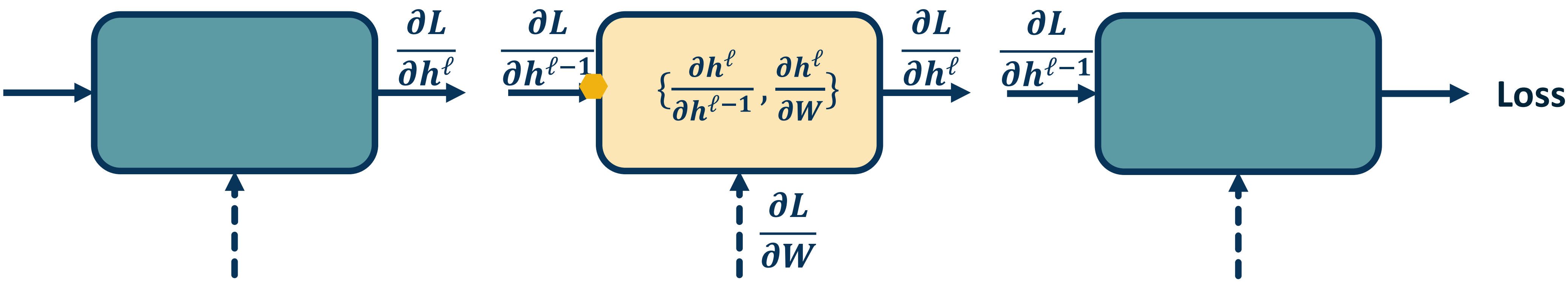
Step 1: Compute Loss on Mini-Batch: **Forward Pass**

Step 2: Compute Gradients wrt parameters: **Backward Pass**



Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

- We want to compute: $\left\{\frac{\partial L}{\partial h^{\ell-1}}, \frac{\partial L}{\partial W}\right\}$



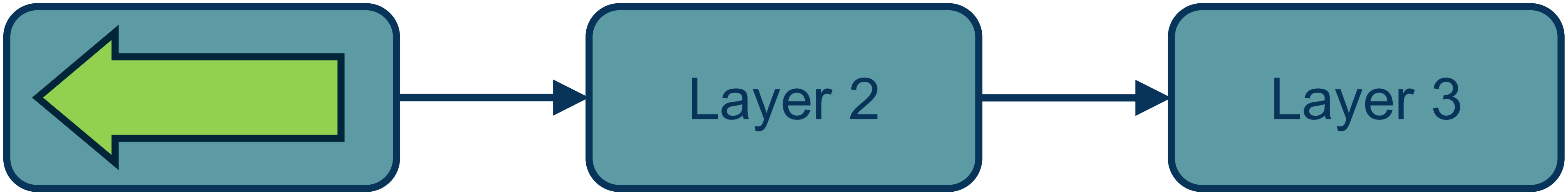
- We will use the *chain rule* to do this:

Chain Rule: $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$

Step 1: Compute Loss on Mini-Batch: **Forward Pass**

Step 2: Compute Gradients wrt parameters: **Backward Pass**

Step 3: Use **gradient** to update **all parameters** at the end



$$w_i = w_i - \alpha \frac{\partial L}{\partial w_i}$$

Backpropagation is the application of gradient descent to a computation graph via the chain rule!

Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

- We can compute **local gradients**: $\{\frac{\partial \boldsymbol{h}^{\ell}}{\partial \boldsymbol{h}^{\ell-1}}, \frac{\partial \boldsymbol{h}^{\ell}}{\partial \boldsymbol{W}}\}$

- This is just the **derivative of our function** with respect to its parameters and inputs!

Example: If $\boldsymbol{h}^{\ell} = \boldsymbol{W}\boldsymbol{h}^{\ell-1}$

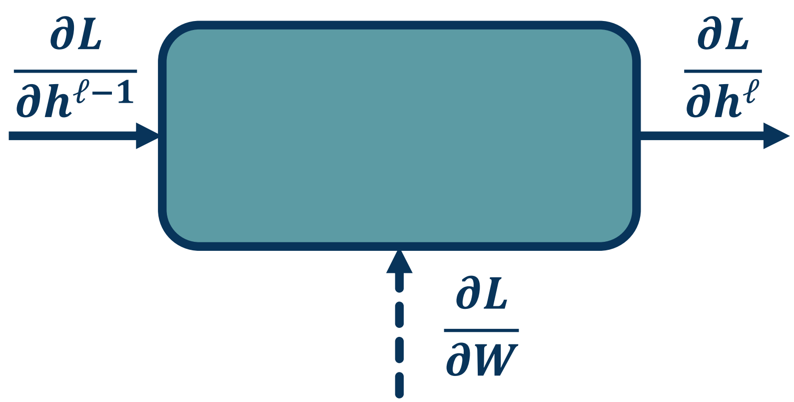
then $\frac{\partial \boldsymbol{h}^{\ell}}{\partial \boldsymbol{h}^{\ell-1}} = \boldsymbol{W}$

and $\frac{\partial \boldsymbol{h}^{\ell}}{\partial \boldsymbol{w}_i} = \boldsymbol{h}^{\ell-1,T}$ (a sparse matrix with $\boldsymbol{h}^{\ell-1,T}$ in the *i*-th row)

Computing the Gradients of Loss

- We will use the **chain rule** to compute: $\{\frac{\partial L}{\partial h^{\ell-1}}, \frac{\partial L}{\partial W}\}$
- **Gradient of loss w.r.t. inputs:** $\frac{\partial L}{\partial h^{\ell-1}} = \frac{\partial L}{\partial h^{\ell}} \frac{\partial h^{\ell}}{\partial h^{\ell-1}}$
- **Gradient of loss w.r.t. weights:** $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h^{\ell}} \frac{\partial h^{\ell}}{\partial W}$

Given by upstream module **(upstream gradient)**



Adapted from figure by Marc'Aurelio Ranzato, Yann LeCun

Backpropagation: a simple example

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

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

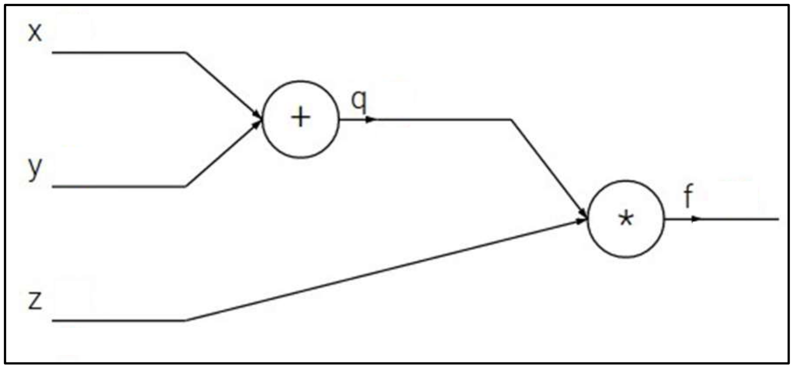


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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

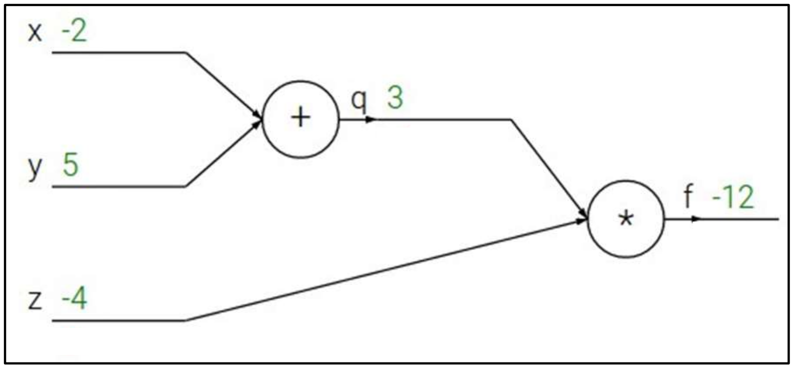
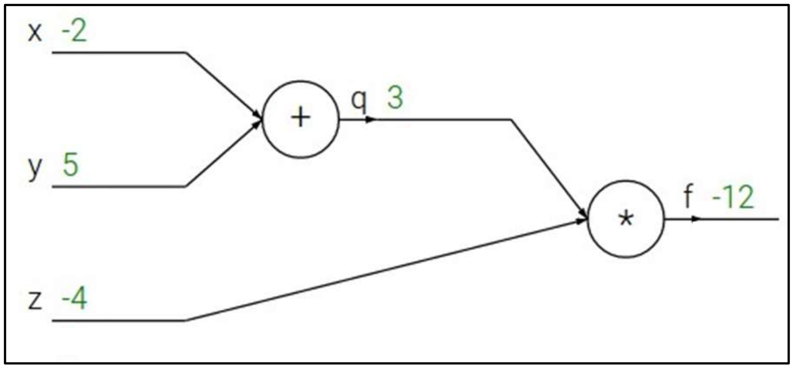


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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



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

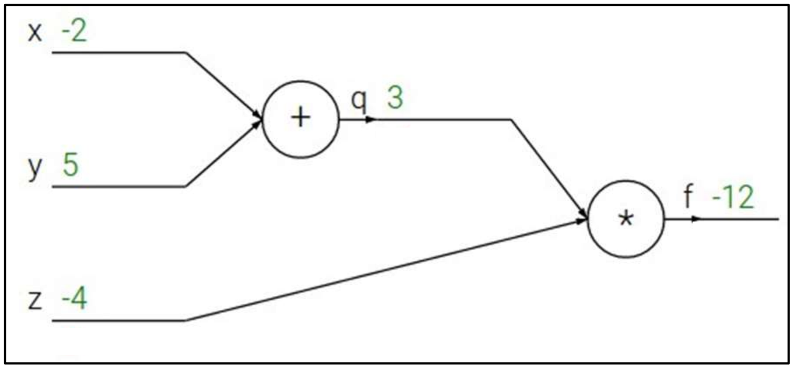
Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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

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



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

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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

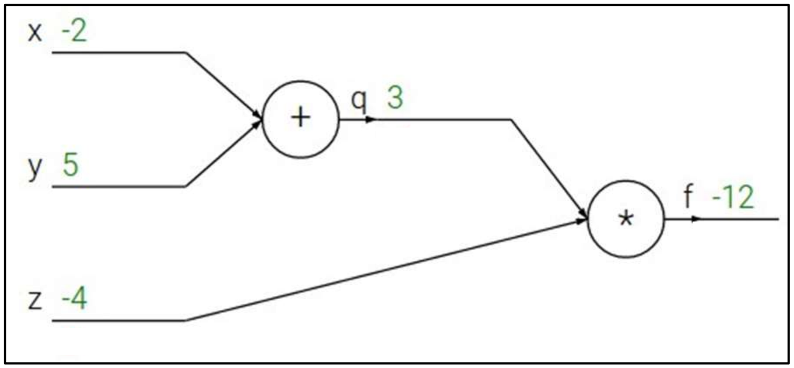


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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

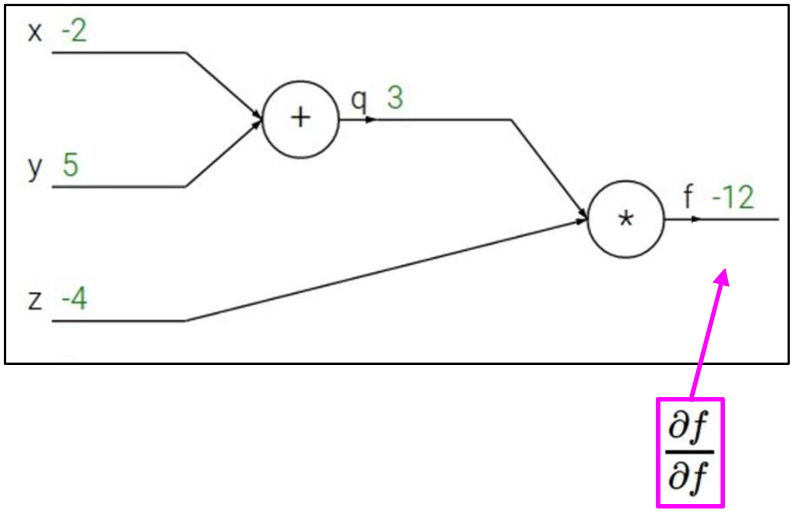


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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

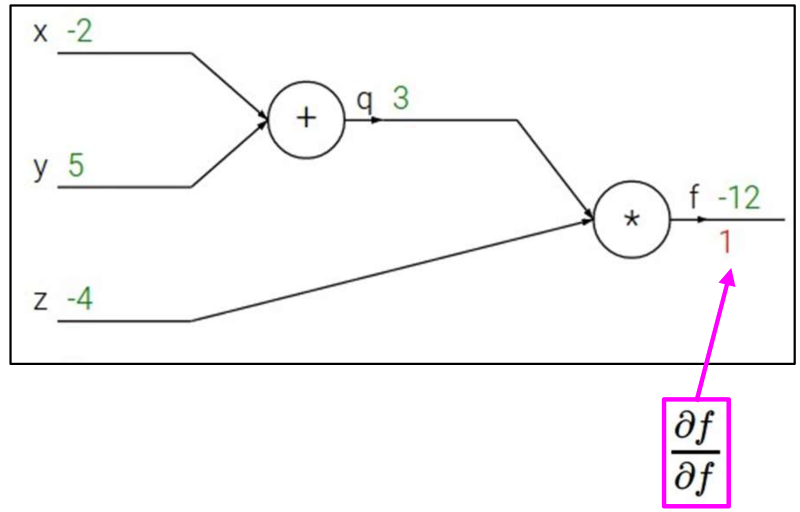


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

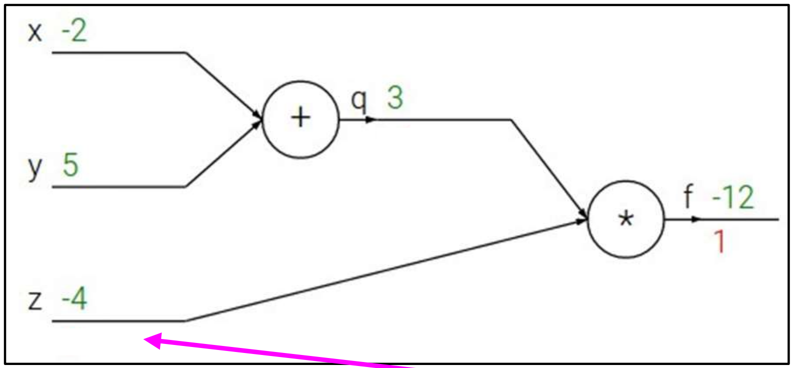
$$f(x, y, z) = (x + y)z$$

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



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

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

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

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

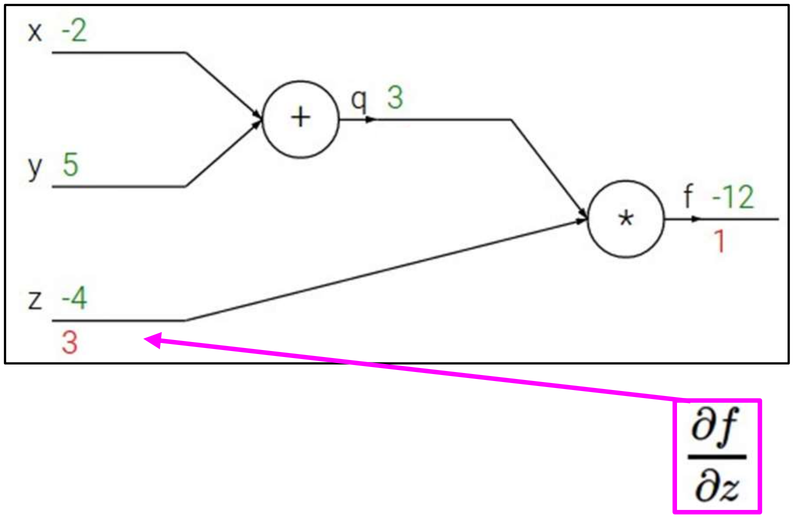


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

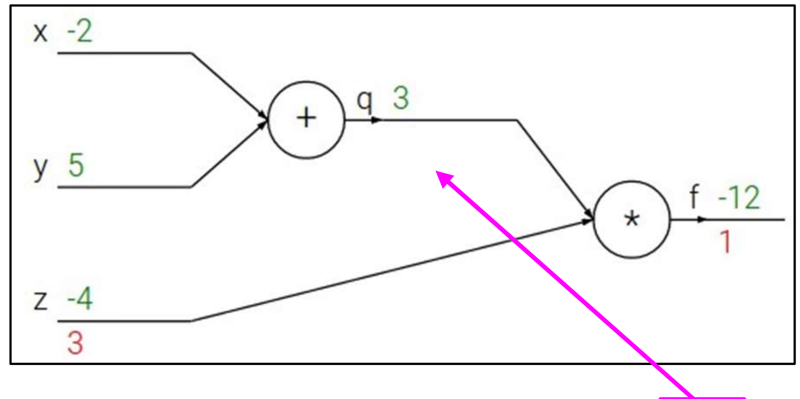
$$f(x, y, z) = (x + y)z$$

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



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

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

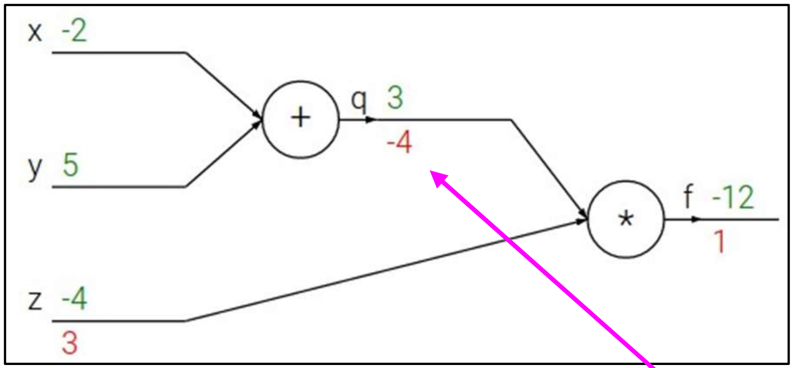
$$f(x, y, z) = (x + y)z$$

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



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

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

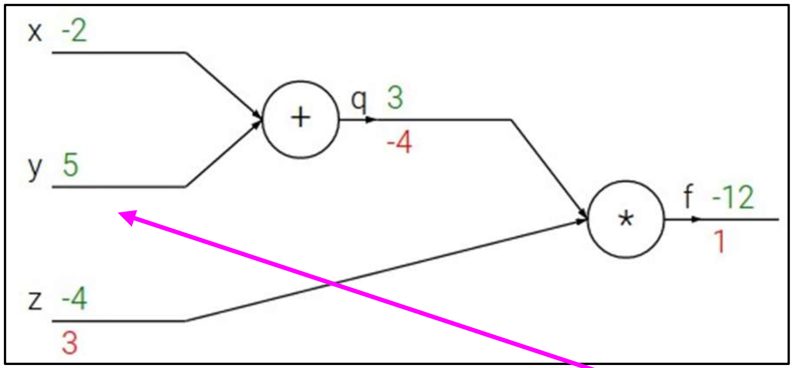
$$f(x, y, z) = (x + y)z$$

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



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

Chain rule:

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

Upstream gradient

Local gradient

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

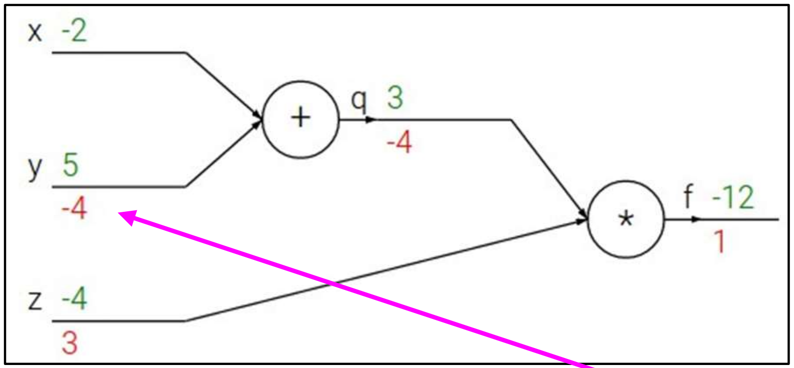
$$f(x, y, z) = (x + y)z$$

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



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

Chain rule:

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

Upstream gradient
Local gradient

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

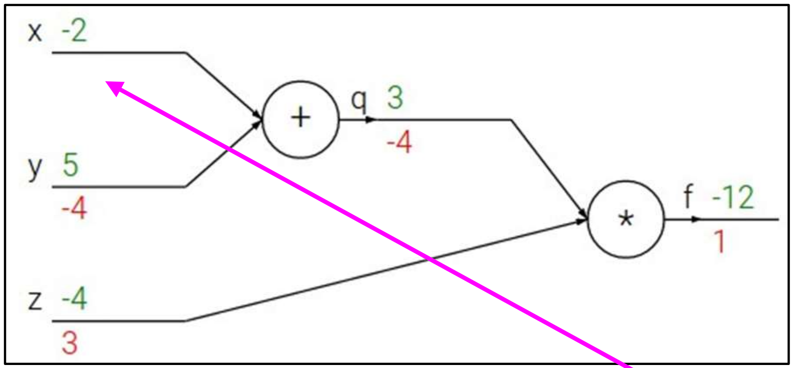
$$f(x, y, z) = (x + y)z$$

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



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

Chain rule:

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

Upstream gradient

Local gradient

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

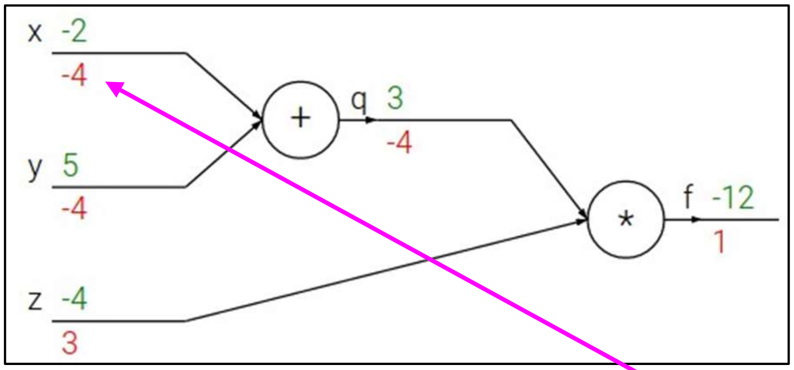
$$f(x, y, z) = (x + y)z$$

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



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

Chain rule:

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

Upstream gradient

Local gradient

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

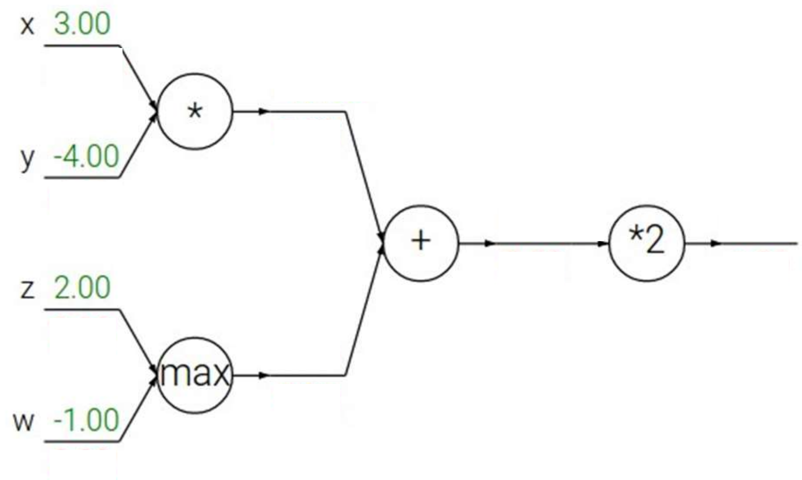


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Backpropagation: a simple example

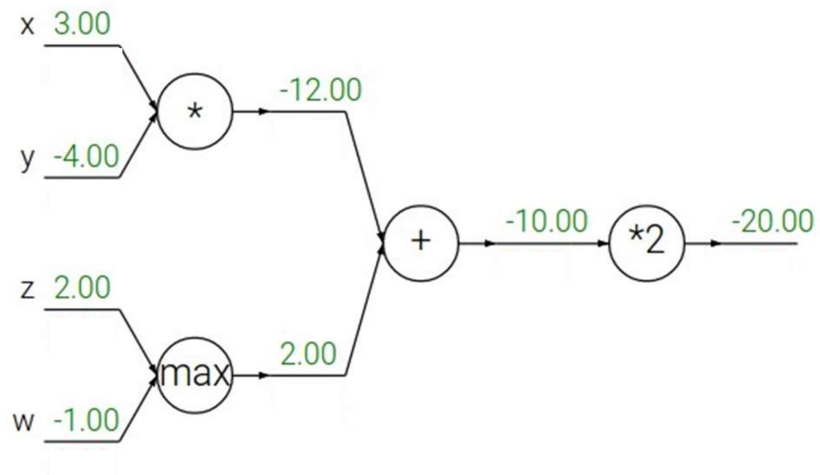


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

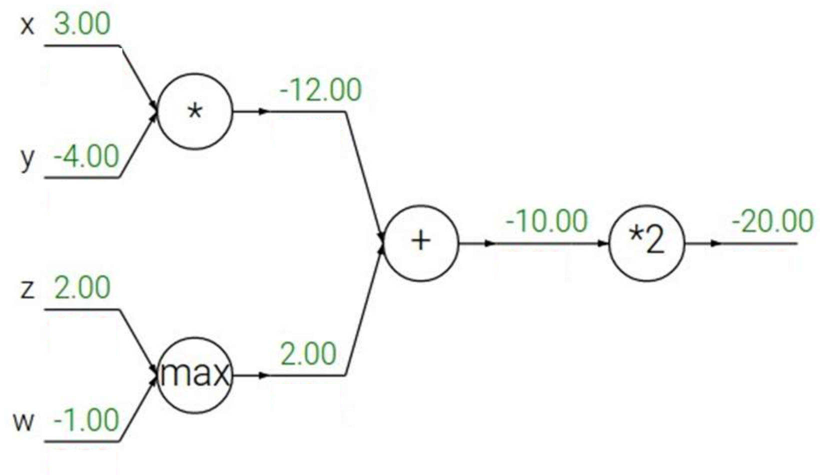


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

Q: What is an **add** gate?

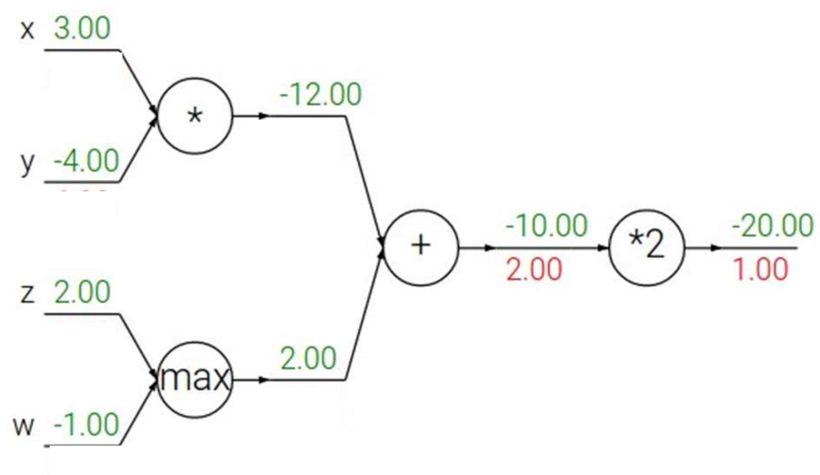


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

add gate: gradient distributor

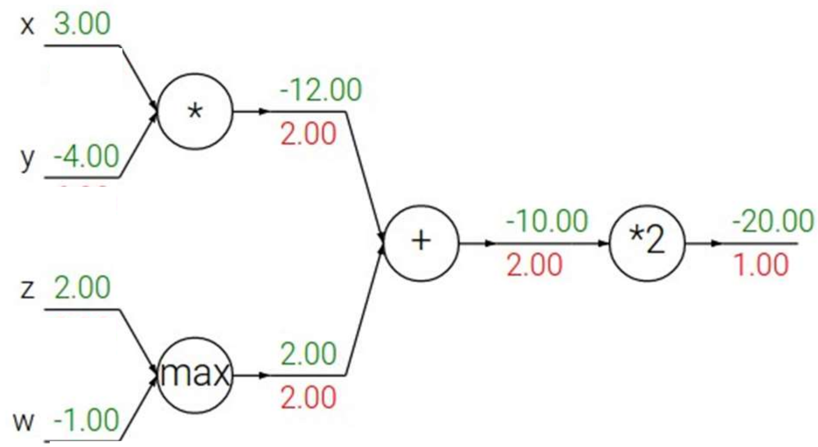


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

add gate: gradient distributor

Q: What is a **max** gate?

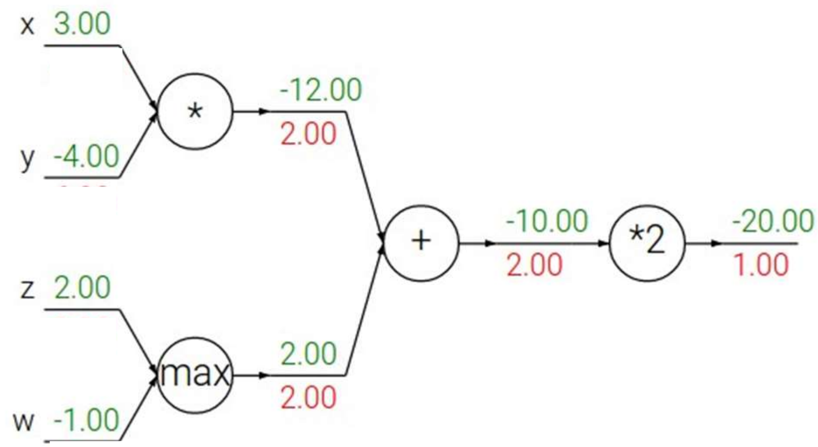


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

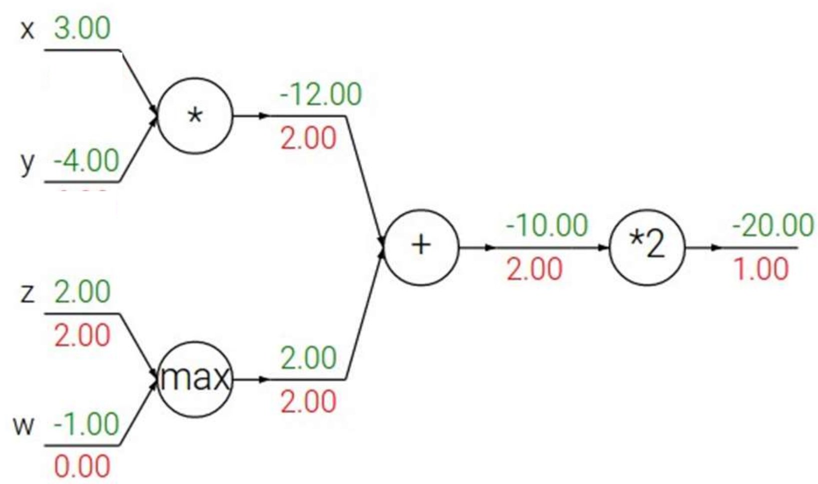


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

Q: What is a **mul** gate?

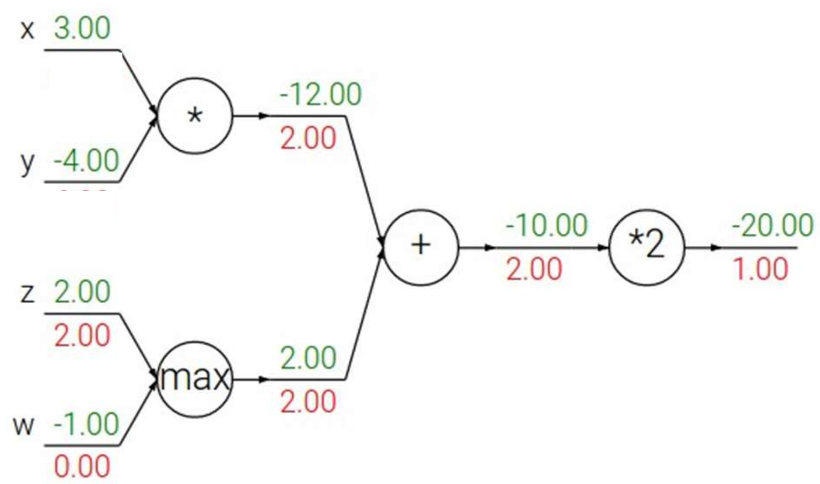


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

mul gate: gradient switcher

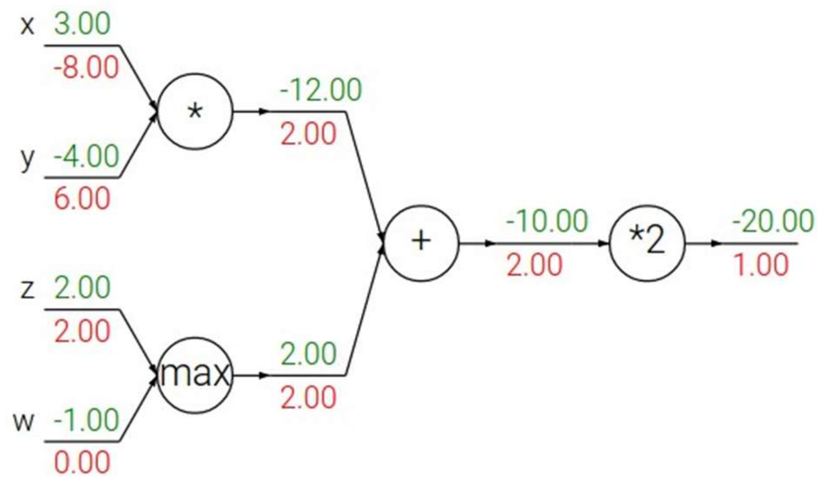
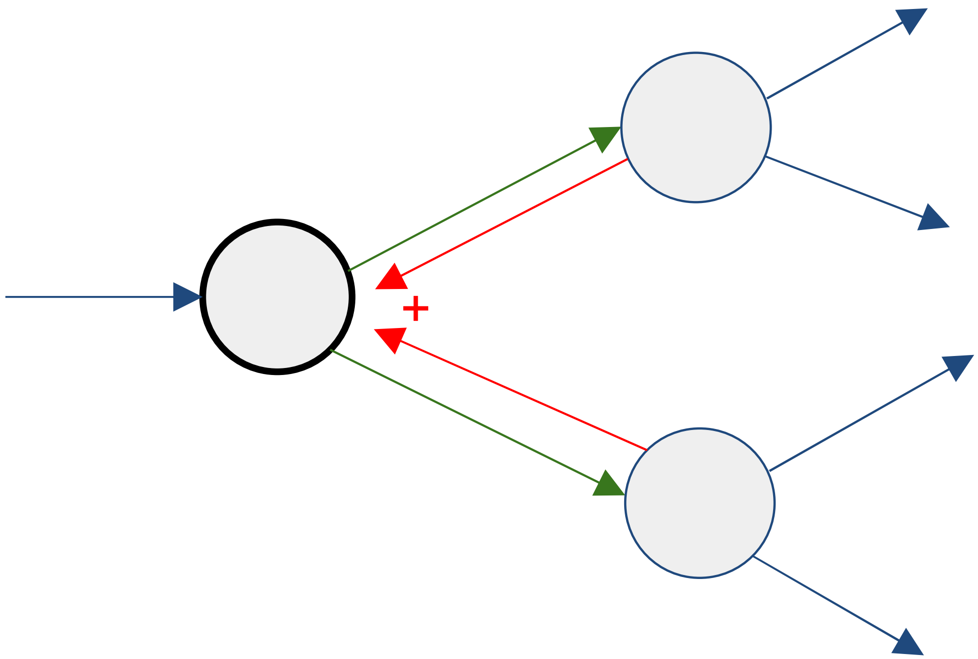
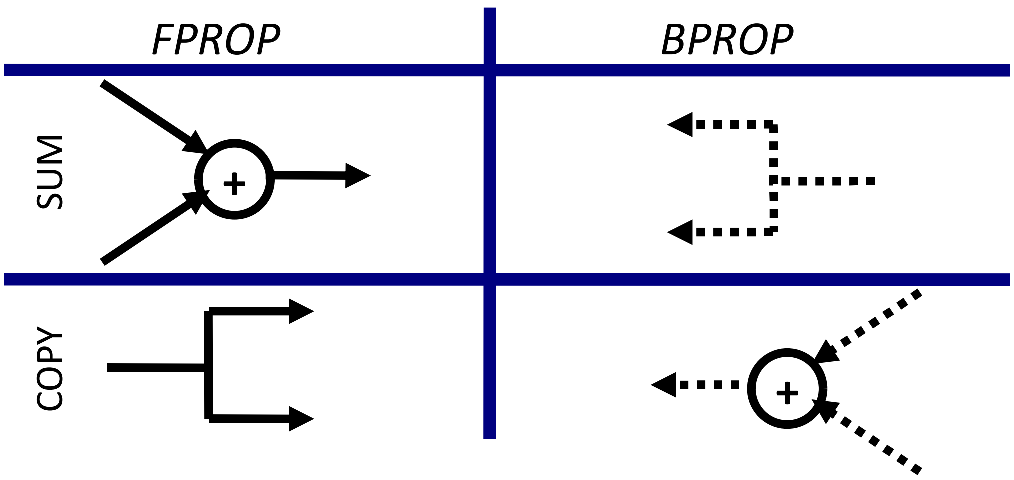


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Gradients add at branches



Duality in Fprop and Bprop



- Neural networks involves composing simple functions into a **computation graph**

- Optimization (updating weights) of this graph is through backpropagation
  - Recursive algorithm: Gradient descent (partial derivatives) plus chain rule

- Remaining questions:
  - How does this work with vectors, matrices, tensors?
    - Across a composed function?
  - How can we implement this algorithmically to make these calculations automatic? **Automatic Differentiation**

Summary

Georgia Tech

Linear Algebra View: Vector and Matrix Sizes

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} & b1 \\ w_{21} & w_{22} & \cdots & w_{2m} & b2 \\ w_{31} & w_{32} & \cdots & w_{3m} & b3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ 1 \end{bmatrix}$$

$\boldsymbol{W}$ $\boldsymbol{x}$

Sizes: $[c \times (d+1)]$ $[(d+1) \times 1]$

Where c is number of classes

d is dimensionality of input

Closer Look at a Linear Classifier

Conventions:

- Size of derivatives for scalars, vectors, and matrices:
  Assume we have scalar $\boldsymbol{s} \in \mathbb{R}^1$, vector $\boldsymbol{v} \in \mathbb{R}^m$, i.e. $\boldsymbol{v} = [v_1, v_2, \ldots, v_m]^T$
  and matrix $\boldsymbol{M} \in \mathbb{R}^{k \times \ell}$

	$\boldsymbol{S}$	$\boldsymbol{V}$	$\boldsymbol{M}$
$\boldsymbol{S}$	$\frac{\partial s_1}{\partial s_2}$	$\frac{\partial s}{\partial v}$	$\frac{\partial s}{\partial M}$
$\boldsymbol{V}$	$\frac{\partial v}{\partial s}$	$\frac{\partial v_1}{\partial v_2}$	Tensors
$\boldsymbol{M}$	$\frac{\partial M}{\partial s}$	Tensors	Tensors

Conventions:

- Size of derivatives for scalars, vectors, and matrices:
  Assume we have scalar $\boldsymbol{s} \in \mathbb{R}^{1}$, vector $\boldsymbol{v} \in \mathbb{R}^{m}$, i.e. $\boldsymbol{v} = [v_1, v_2, \dots, v_m]^T$ and matrix $\boldsymbol{M} \in \mathbb{R}^{k \times \ell}$

- What is the size of $\frac{\partial v}{\partial s}$? $\mathbb{R}^{m \times 1}$ (column vector of size m)

$$\begin{bmatrix} \frac{\partial v_1}{\partial s} \\ \frac{\partial v_2}{\partial s} \\ \vdots \\ \frac{\partial v_m}{\partial s} \end{bmatrix}$$

- What is the size of $\frac{\partial s}{\partial v}$? $\mathbb{R}^{1 \times m}$ (row vector of size m)

$$\begin{bmatrix} \frac{\partial s}{\partial v_1} & \frac{\partial s}{\partial v_1} & \cdots & \frac{\partial s}{\partial v_m} \end{bmatrix}$$

Conventions:

- What is the size of $\frac{\partial v^1}{\partial v^2}$? A matrix:

$$\begin{array}{c} \text{Col } j \\ \text{Row } i \quad \begin{bmatrix} \frac{\partial v_1^1}{\partial v_1^2} & \cdots & \cdots & \cdots & \cdots & \\ \cdots & & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial v_i^1}{\partial v_1^2} & \cdots & \frac{\partial v_i^1}{\partial v_j^2} & \cdots & \frac{\partial v_i^1}{\partial v_{m_2}^2} & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \end{bmatrix}_{m_1 \times m_2} \end{array}$$

- This matrix of partial derivatives is called a **Jacobian**

(Note this is slightly different convention than on Wikipedia). Also, computationally other conventions are used.

Conventions:

- What is the size of $\frac{\partial s}{\partial M}$? A matrix:

$$\begin{bmatrix} \frac{\partial s}{\partial m_{[1,1]}} & \cdots & \cdots & \cdots & \cdots & \\ \cdots & & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \frac{\partial s}{\partial m_{[i,j]}} & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \end{bmatrix}$$

(Note this is slightly different convention than on Wikipedia). Also, computationally other conventions are used.

Example 1:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x \\ x^2 \end{bmatrix} \qquad \frac{\partial \boldsymbol{y}}{\partial x} = \begin{bmatrix} 1 \\ 2x \end{bmatrix}$$

Example 2:

$$y = \boldsymbol{w}^T \boldsymbol{x} = \sum_k w_k x_k$$

$$\frac{\partial y}{\partial \boldsymbol{x}} = \left[\frac{\partial y}{\partial x_1}, \dots, \frac{\partial y}{\partial x_m}\right]$$

$$= [w_1, \dots, w_m]$$ because $$\frac{\partial(\sum_k w_k x_k)}{\partial x_i} = w_i$$

$$= \boldsymbol{w}^T$$

Example 3:

$$y = Wx \qquad \frac{\partial y}{\partial x} = W$$

Col j

Row i

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \cdots & \cdots & \cdots & \\ \cdots & & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \frac{\partial y_i}{\partial x_j} & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & w_{ij} & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \end{bmatrix}$$

$$y_i = \sum_j w_{ij} x_j$$

Example 4:

$$\frac{\partial (wAw)}{\partial w} = 2w^T A$$ (assuming A is symmetric)

- What is the size of $\frac{\partial L}{\partial W}$?
  - Remember that loss is a **scalar** and W is a matrix:

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} & b1 \\ w_{21} & w_{22} & \cdots & w_{2m} & b2 \\ w_{31} & w_{32} & \cdots & w_{3m} & b3 \end{bmatrix}$$

$$W$$

Jacobian is also a matrix:

$$\begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \cdots & \frac{\partial L}{\partial w_{1m}} & \frac{\partial L}{\partial b_1} \\ \frac{\partial L}{\partial w_{21}} & \cdots & \cdots & \frac{\partial L}{\partial w_{2m}} & \frac{\partial L}{\partial b_2} \\ \cdots & \cdots & \cdots & \frac{\partial L}{\partial w_{3m}} & \frac{\partial L}{\partial b_3} \end{bmatrix}$$

Jacobians of Batches

Batches of data are **matrices** or **tensors** (multi-dimensional matrices)

Examples:

- Each instance is a vector of size m, our batch is of size $[\boldsymbol{B} \times \boldsymbol{m}]$
- Each instance is a matrix (e.g. grayscale image) of size $\boldsymbol{W} \times \boldsymbol{H}$, our batch is $[\boldsymbol{B} \times \boldsymbol{W} \times \boldsymbol{H}]$
- Each instance is a multi-channel matrix (e.g. color image with R,B,G channels) of size $\boldsymbol{C} \times \boldsymbol{W} \times \boldsymbol{H}$, our batch is $[\boldsymbol{B} \times \boldsymbol{C} \times \boldsymbol{W} \times \boldsymbol{H}]$

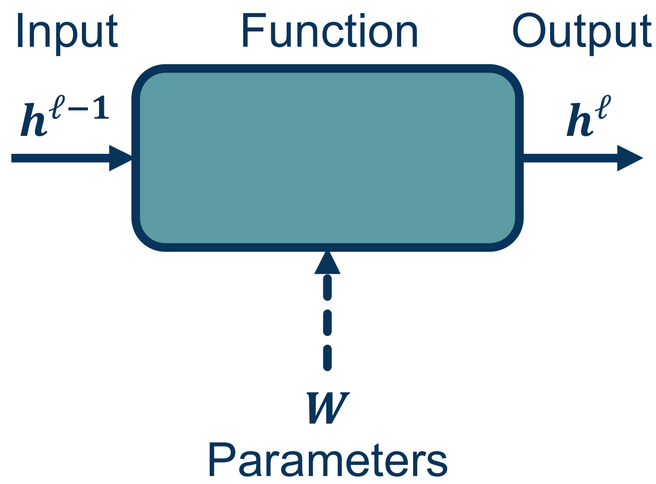
Jacobians become tensors which is complicated

- Instead, flatten input to a vector and get a vector of derivatives!
- This can also be done for partial derivatives between two vectors, two matrices, or two tensors

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}$$

Flatten

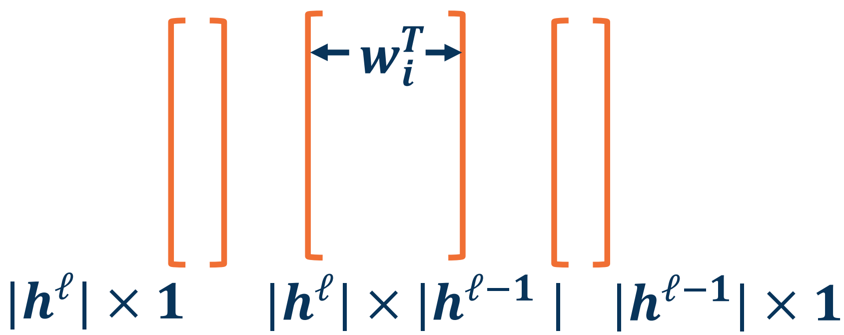
$$\begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{21} \\ x_{22} \\ \vdots \\ x_{n1} \\ \vdots \\ x_{nn} \end{bmatrix}$$



Define:

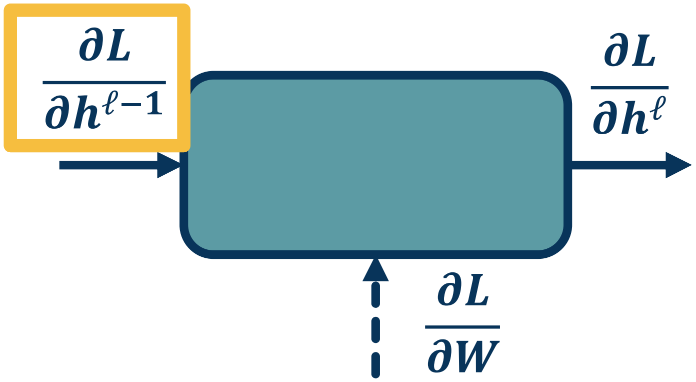
$$h_i^{\ell} = w_i^T h^{\ell-1}$$

$$h^{\ell} = W h^{\ell-1}$$



Fully Connected (FC) Layer: Forward Function

Georgia Tech



$$h^{\ell} = W h^{\ell-1}$$

$$\frac{\partial h^{\ell}}{\partial h^{\ell-1}} = W$$

Define:

$$h_i^{\ell} = w_i^T h^{\ell-1}$$

$$\frac{\partial L}{\partial h^{\ell-1}} = \frac{\partial L}{\partial h^{\ell}} \frac{\partial h^{\ell}}{\partial h^{\ell-1}}$$

$$1 \times |h^{\ell-1}| \quad 1 \times |h^{\ell}| \quad |h^{\ell}| \times |h^{\ell-1}|$$

Fully Connected (FC) Layer

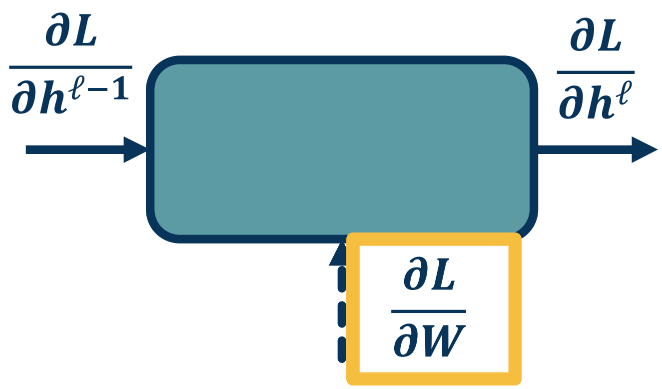
$$h^{\ell} = W h^{\ell-1}$$

$$\frac{\partial h^{\ell}}{\partial h^{\ell-1}} = W$$

Define:

$$h_i^{\ell} = w_i^T h^{\ell-1}$$

$$\frac{\partial h_i^{\ell}}{\partial w_i^T} = h^{(\ell-1),T}$$



Note doing this on full W matrix would result in Jacobian tensor!

But it is *sparse* – each output only affected by corresponding weight row

$$\frac{\partial L}{\partial w_i^T} = \frac{\partial L}{\partial h^{\ell}} \frac{\partial h^{\ell}}{\partial w_i^T}$$

$$\begin{bmatrix} \quad \end{bmatrix} \begin{bmatrix} \quad \end{bmatrix} \begin{bmatrix} \leftarrow & 0 & \rightarrow \\ \leftarrow & \frac{\partial h_i^{\ell}}{\partial w_i^T} & \rightarrow \\ \leftarrow & 0 & \rightarrow \end{bmatrix}$$

$$1 \times |h^{\ell-1}| \quad 1 \times |h^{\ell}| \quad |h^{\ell}| \times |h^{\ell-1}|$$

Fully Connected (FC) Layer

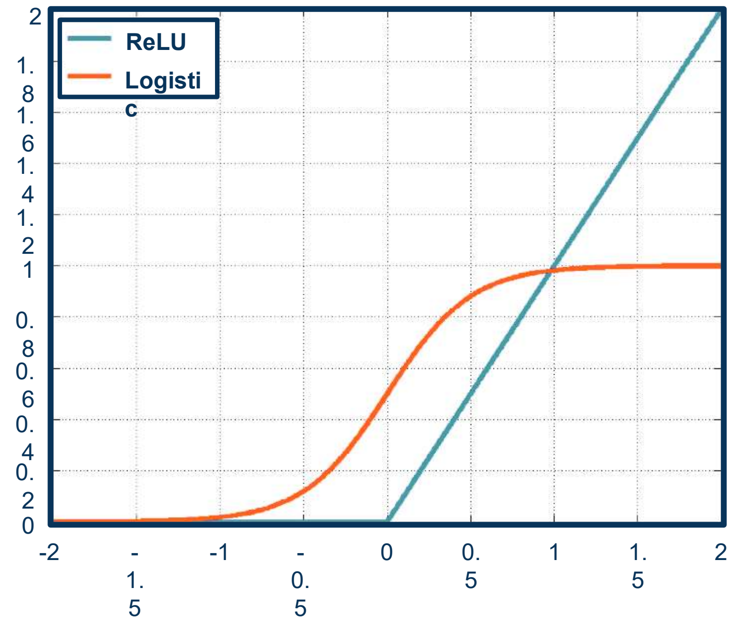
Rectified Linear Unit (ReLU)

We can employ **any differentiable (or piecewise differentiable) function**

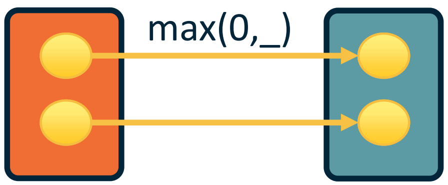
A common choice is the **Rectified Linear Unit**

- Provides non-linearity but better gradient flow than sigmoid
- Performed **element-wise**

How many parameters for this layer?



$$h^{\ell} = \max(0, h^{\ell-1})$$



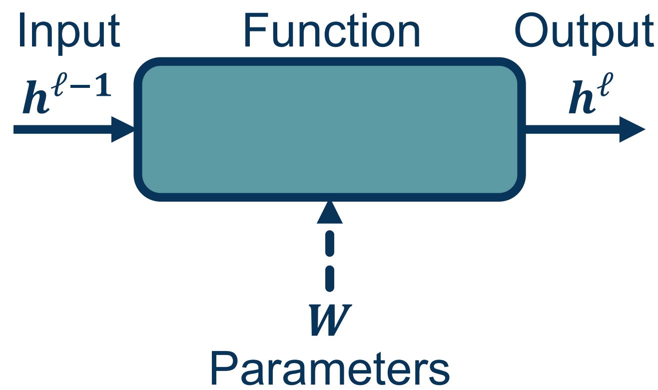
Jacobian of ReLU

Full Jacobian of ReLU layer is **large** (output dim x input dim)

- But again it is **sparse**
- Only **diagonal values non-zero** because it is element-wise
- An output value affected only by **corresponding input value**

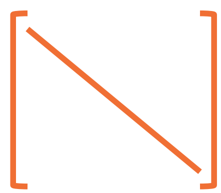
Max function **funnels gradients through selected max**

- Gradient will be **zero** if input **<= 0**



Forward: $h^{\ell} = \max(0, h^{\ell-1})$

Backward: $\frac{\partial L}{\partial h^{\ell-1}} = \frac{\partial L}{\partial h^{\ell}} \frac{\partial h^{\ell}}{\partial h^{\ell-1}}$

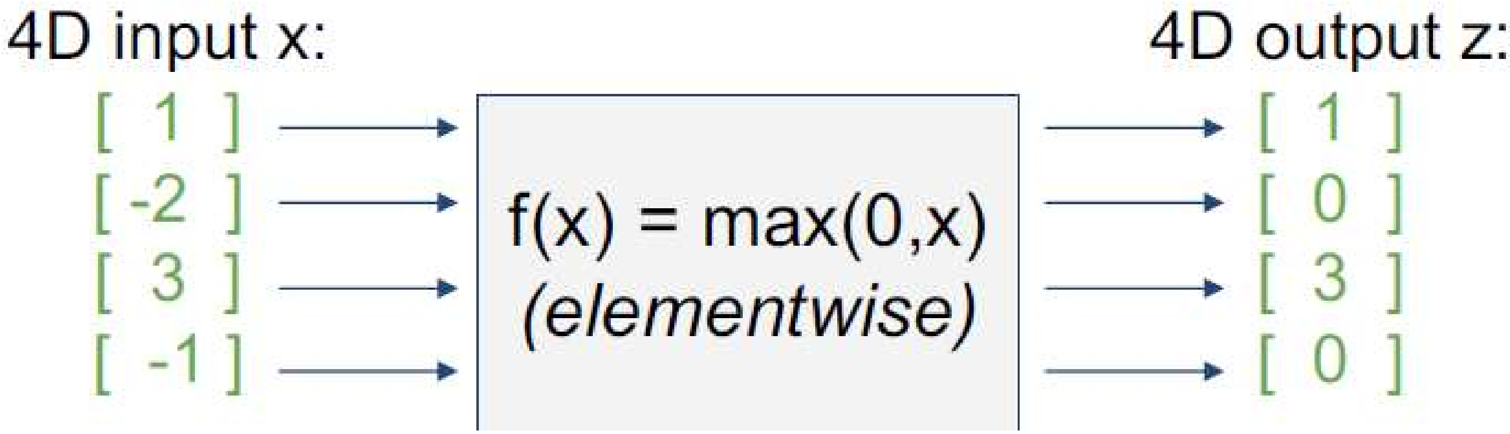


$|h^{\ell} \times h^{\ell-1}|$

For diagonal

$$\frac{\partial h^{\ell}}{\partial h^{\ell-1}} = \begin{cases} 1 & if\ h^{\ell-1} > 0 \\ 0 & otherwise \end{cases}$$

Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



What does $\frac{\partial z}{\partial x}$ look like?

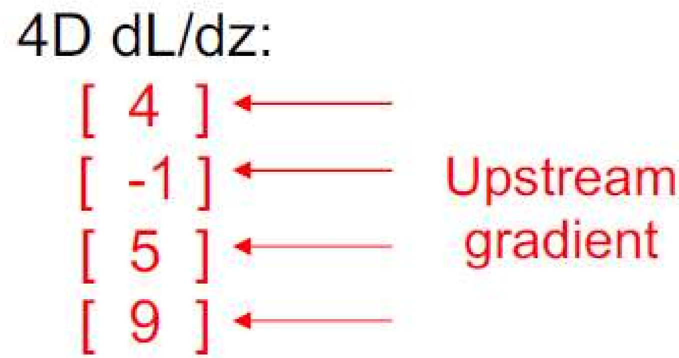
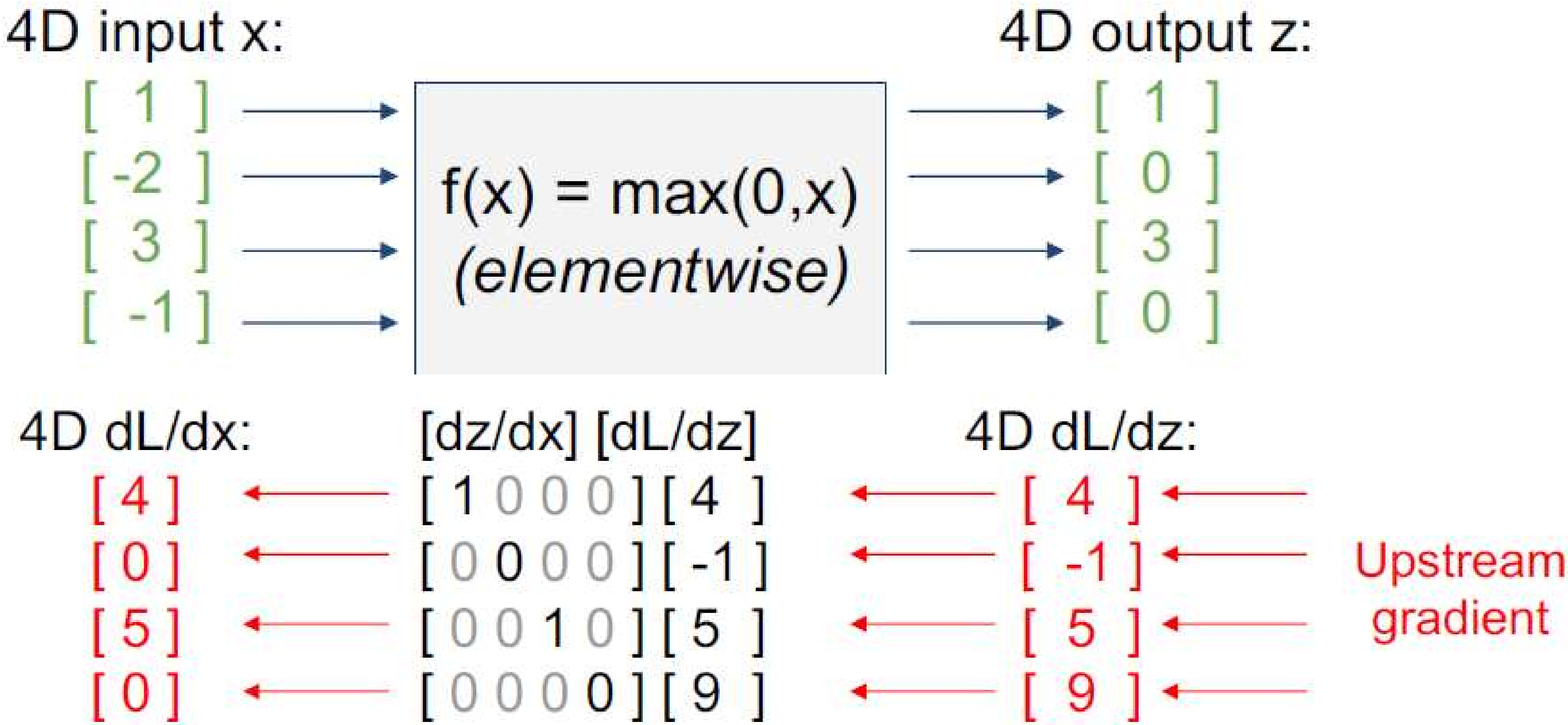


Figure adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



For element-wise ops, jacobian is **sparse**: off-diagonal entries always zero!
Never **explicitly** form Jacobian -- instead use elementwise multiplication

Summary

- Neural networks involves composing simple functions into a **computation graph**

- Optimization (updating weights) of this graph is through backpropagation
  - Recursive algorithm: Gradient descent (partial derivatives) plus chain rule

- Remaining questions:
  - How does this work with vectors, matrices, tensors?
    - Across a composed function? **Next Time!**
  - How can we implement this algorithmically to make these calculations automatic? **Automatic Differentiation**