JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

EN.601.482/682 Deep Learning

# Computational Graphs and Backprop

## Part I

Mathias Unberath, PhD
Assistant Professor
Dept of Computer Science
Johns Hopkins University

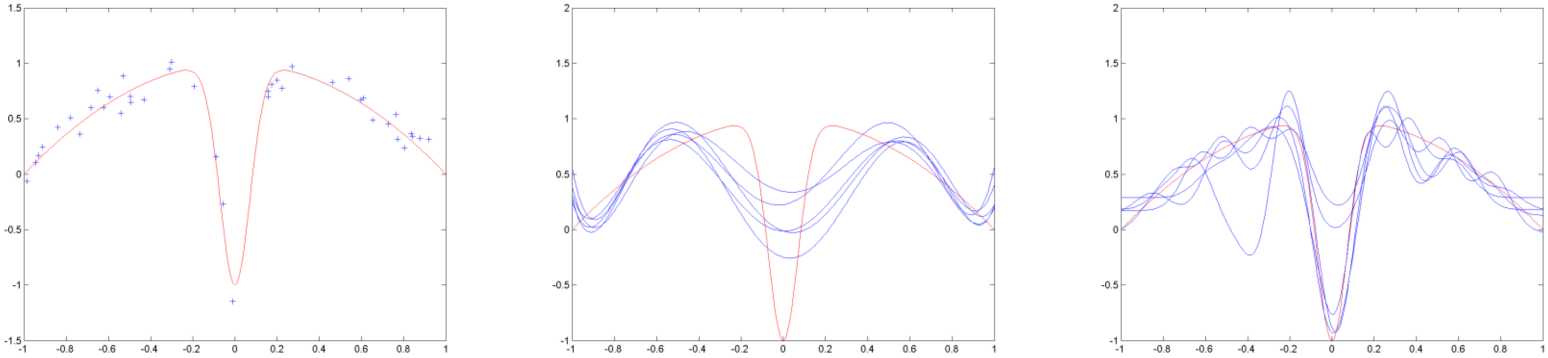
# Reminder

- **Homework assignment 1 is due today**
- **Homework assignment 2 will be released today** (due in 2 weeks)

# Reminder

- The bias variance tradeoff

$$L(W) = \underbrace{(E[\hat{y}] - y)^2}_{\text{Bias}^2} + \underbrace{E[(\hat{y} - E[\hat{y}])^2]}_{\text{Variance}} + \sigma$$



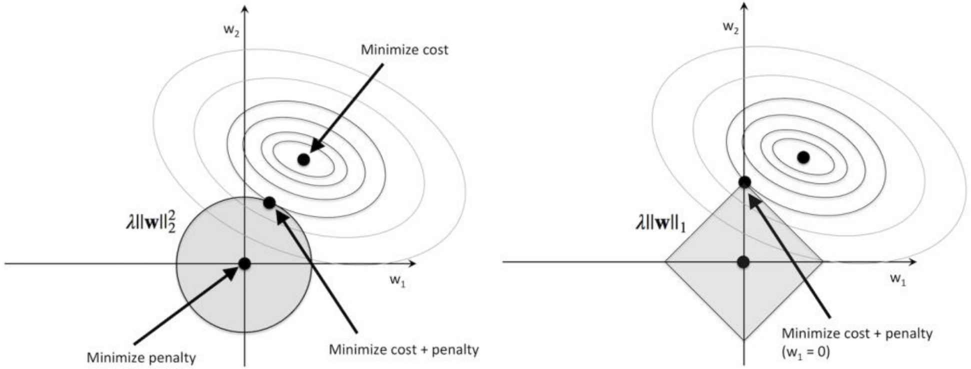
# Reminder

- The bias variance tradeoff

$$L(W) = \underbrace{\left(E[\hat{y}] - y\right)^2}_{\text{Bias}^2} + \underbrace{E[(\hat{y} - E[\hat{y}])^2]}_{\text{Variance}} + \sigma$$

- Regularization, e.g. L-norms

$$L(W) = \underbrace{\frac{1}{N} \sum_i L_i\left(f(x_i, W), y_i\right)}_{\textbf{Data fidelity}} + \underbrace{\lambda R(W)}_{\textbf{Regularization}}$$



# Reminder

- The bias variance tradeoff

$$L(W) = \underbrace{(E[\hat{y}] - y)^2}_{\text{Bias}^2} + \underbrace{E[(\hat{y} - E[\hat{y}])^2]}_{\text{Variance}} + \sigma$$

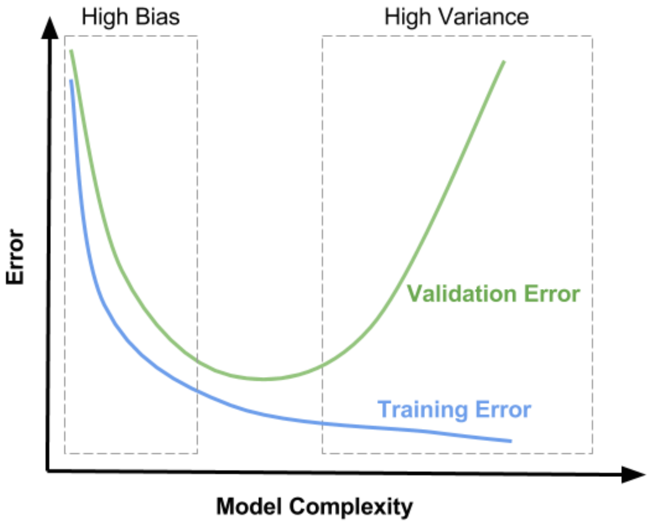
- Optimization:
  - Evaluate gradient of loss L at current estimate W

$$\nabla L(\mathbf{W}) = \left( \frac{\partial L}{\partial W_1} \quad \frac{\partial L}{\partial W_2} \quad \cdots \quad \frac{\partial L}{\partial W_n} \right)$$

  - Update W in the direction of steepest descent

$$\mathbf{W}' = \mathbf{W} - \lambda \nabla L(\mathbf{W})$$

  - When to stop?



# Reminder

- The bias variance tradeoff

$$L(W) = \underbrace{(E[\hat{y}] - y)^2}_{\text{Bias}^2} + \underbrace{E[(\hat{y} - E[\hat{y}])^2]}_{\text{Variance}} + \sigma$$

- Appropriate data curation and early stopping

| Train | Validation | Test |
|---|---|---|

→ Split data into train, validation, and test; hyperparameters chosen on validation, then evaluated on test.

# Reminder

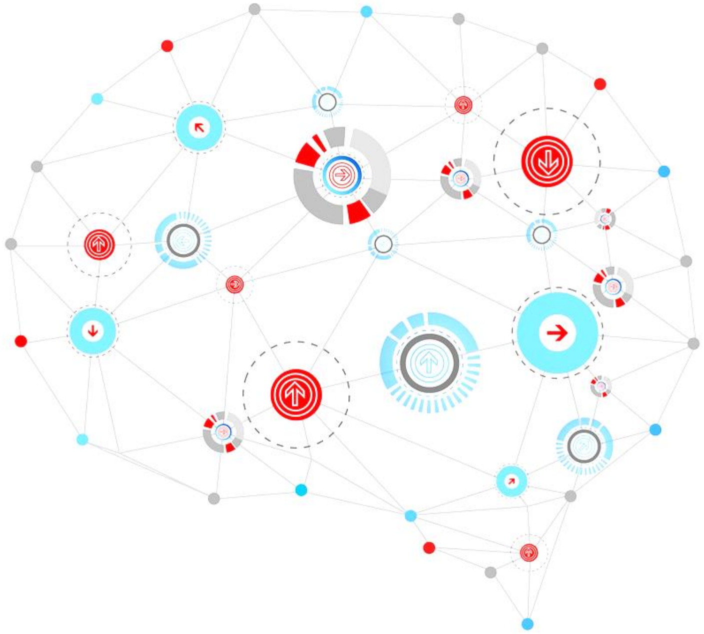
- Numerical gradient: Approximate and slow, but easy
- Analytic gradient: Exact and fast, but can get complicated with complex function expressions

**Goal for today:** Compute analytic gradients of complex expressions!

# Today’s Lecture

## Computational Graphs

## Backpropagation



Computational Graphs and Backpropagation

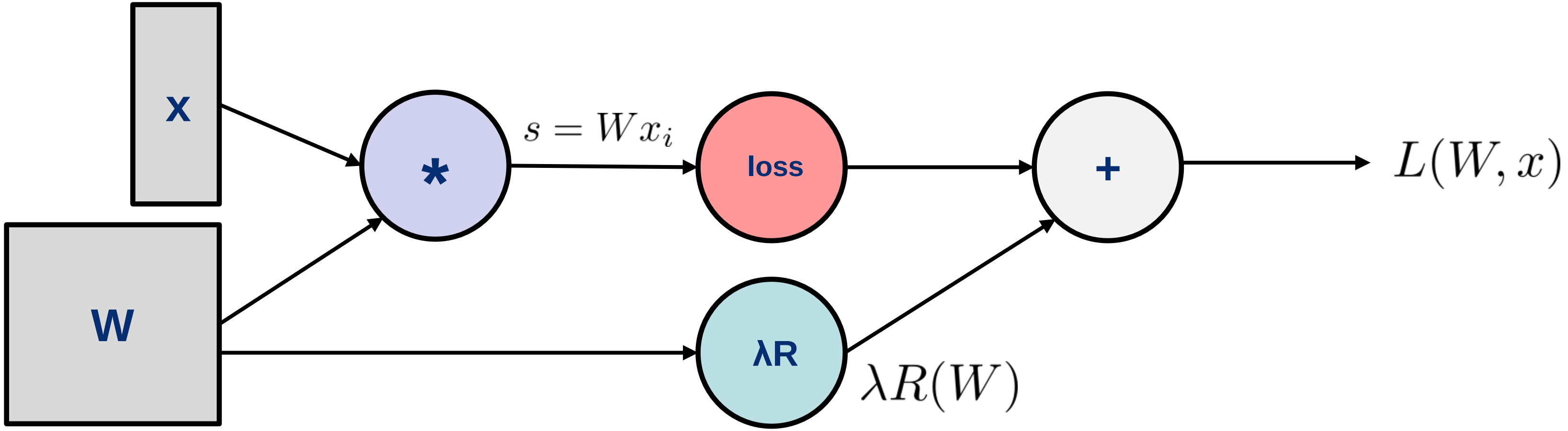
# Computational Graphs

# Computational Graphs

Score function: $s = f(x_i, W) = Wx_i$

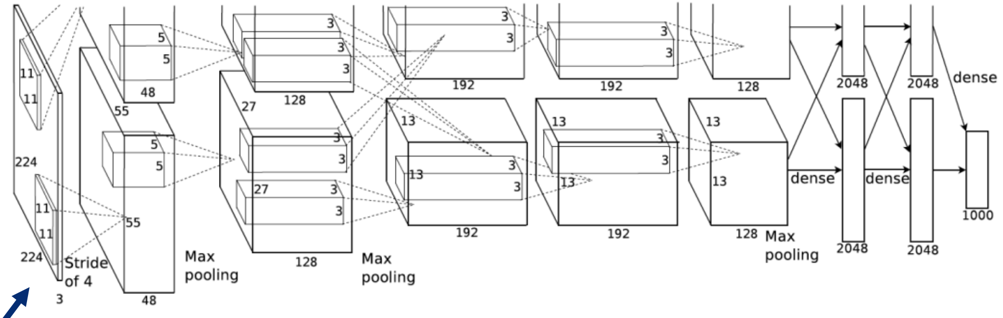
Loss function: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

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



# Computational Graphs

A convolutional neural network (AlexNet) as computational graph

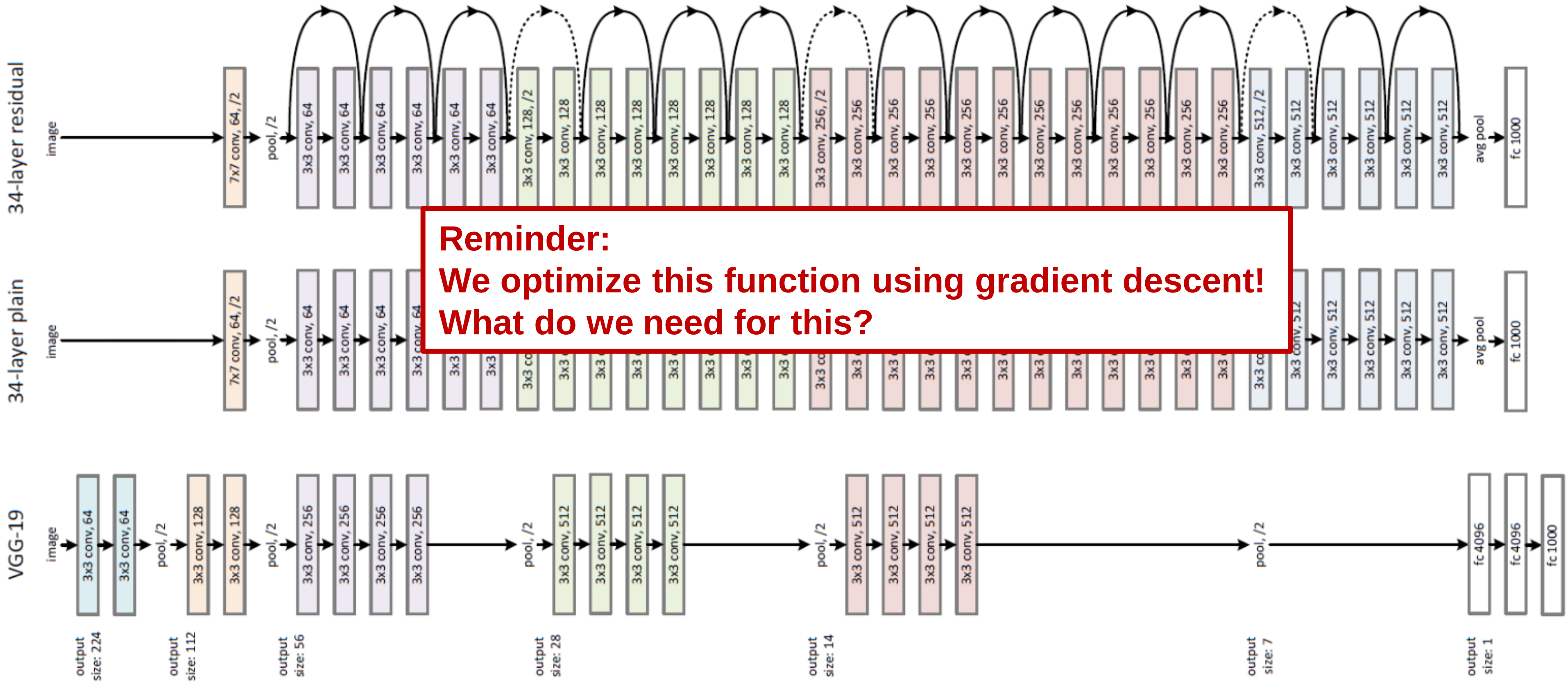



Input image

Weights

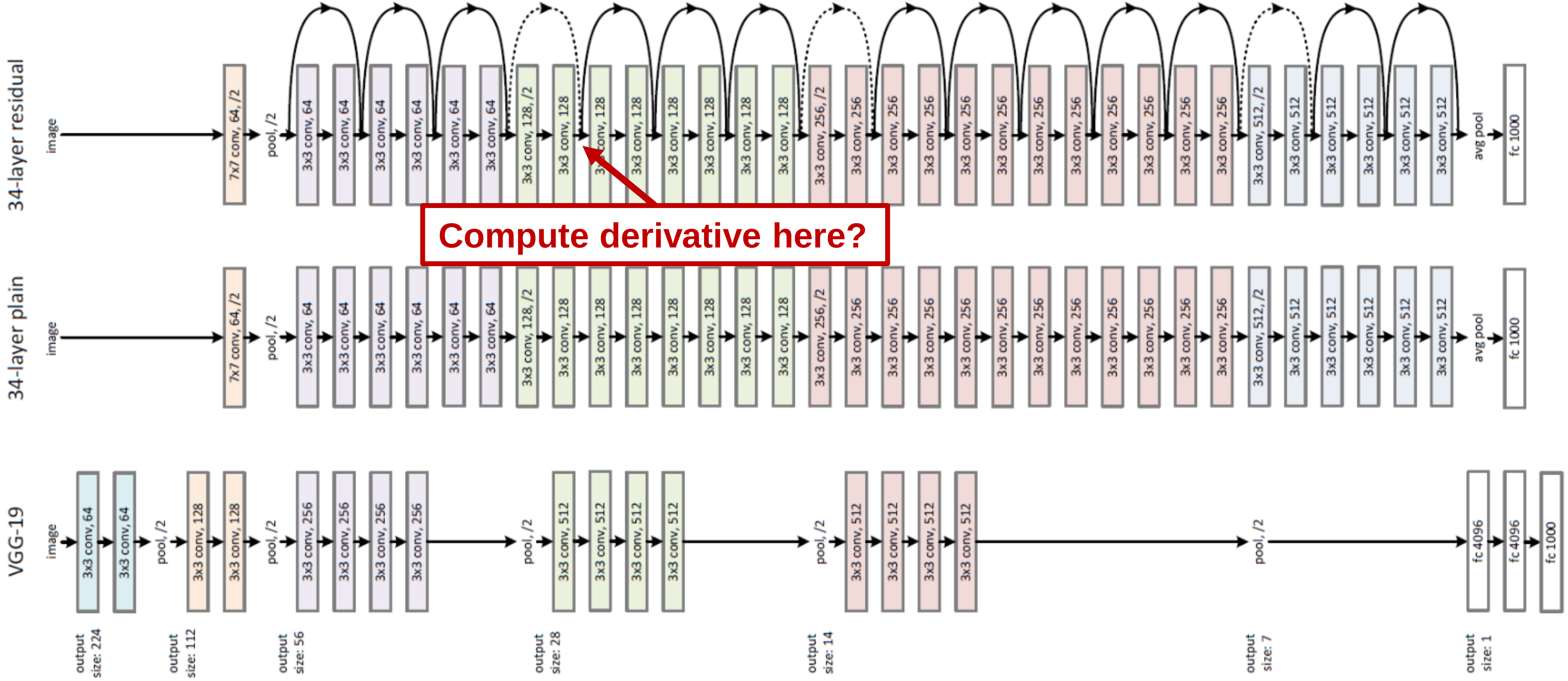
Output

# Computational Graphs




**Reminder:**
**We optimize this function using gradient descent!**
**What do we need for this?**

# Computational Graphs



# Computational Graphs

Computational graphs help us compute derivatives at arbitrary locations!

**Q: Why?**

# Computational Graphs

Computational graphs help us compute derivatives at arbitrary locations!

**Q: Why?**

**A:**

- Complex expressions are broken down into easy functions
- Forward pass: Evaluate expression
- Backward pass → We are about to find out!

Compute Graphs and Backpropagation

# Backpropagation

# A Simple Example

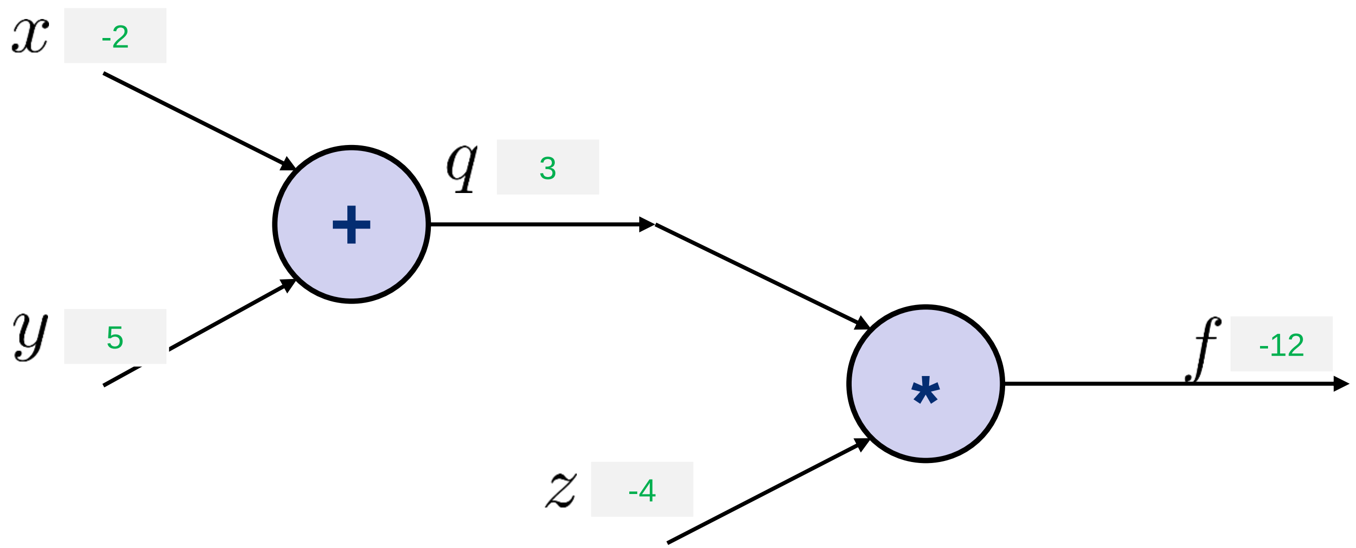
$$f(x, y, z) = (x + y) \cdot z$$

Tasks:

1. Set-up computational graph
2. Compute forward pass
3. Find derivatives $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

# Setting Up the Computation Graph & Forward Pass

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

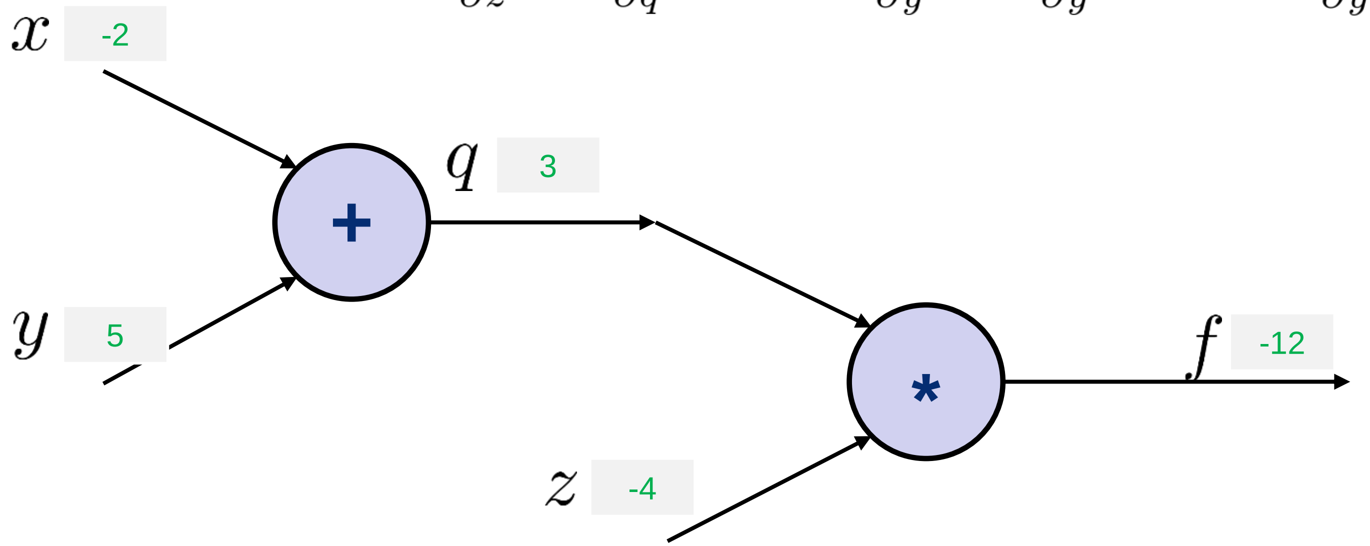


# Finding Local Derivatives

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

$$\frac{\partial f}{\partial q} = \frac{\partial q \cdot z}{\partial q} = z \quad \Bigg| \quad \frac{\partial q}{\partial x} = \frac{\partial x+y}{\partial x} = 1 \quad \Bigg| \quad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial z} = \frac{\partial q \cdot z}{\partial q} = q \quad \Bigg| \quad \frac{\partial q}{\partial y} = \frac{\partial x+y}{\partial y} = 1 \quad \Bigg| \quad \frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

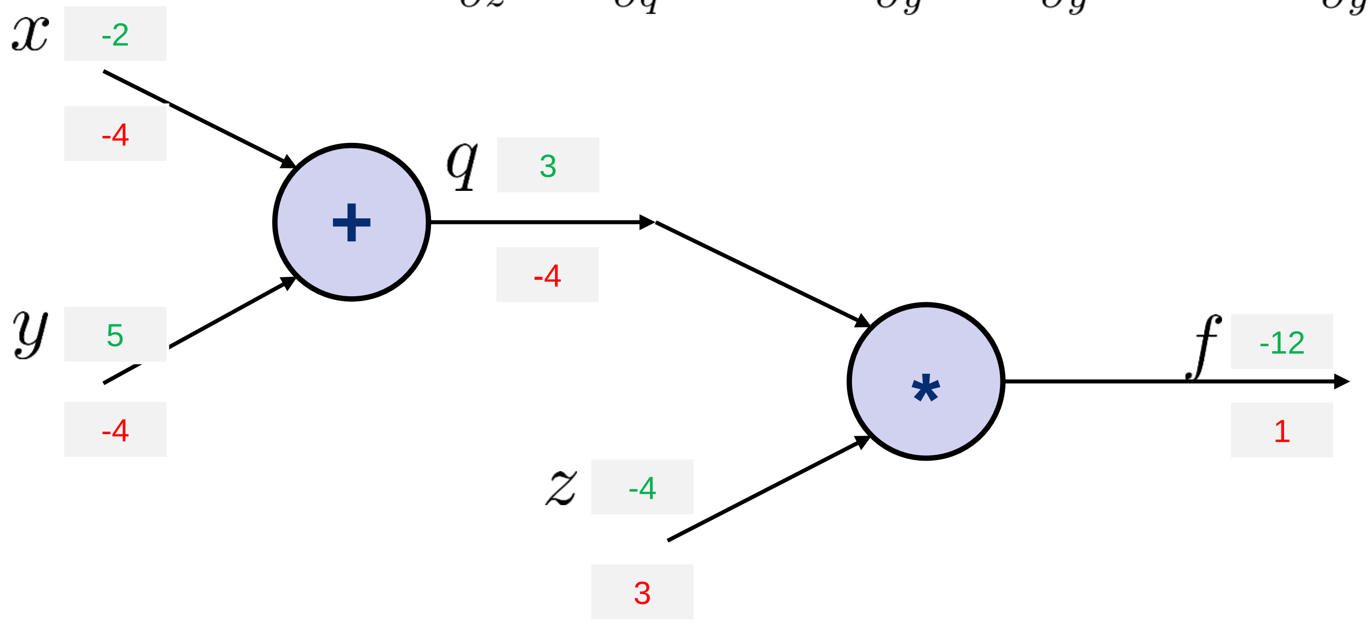


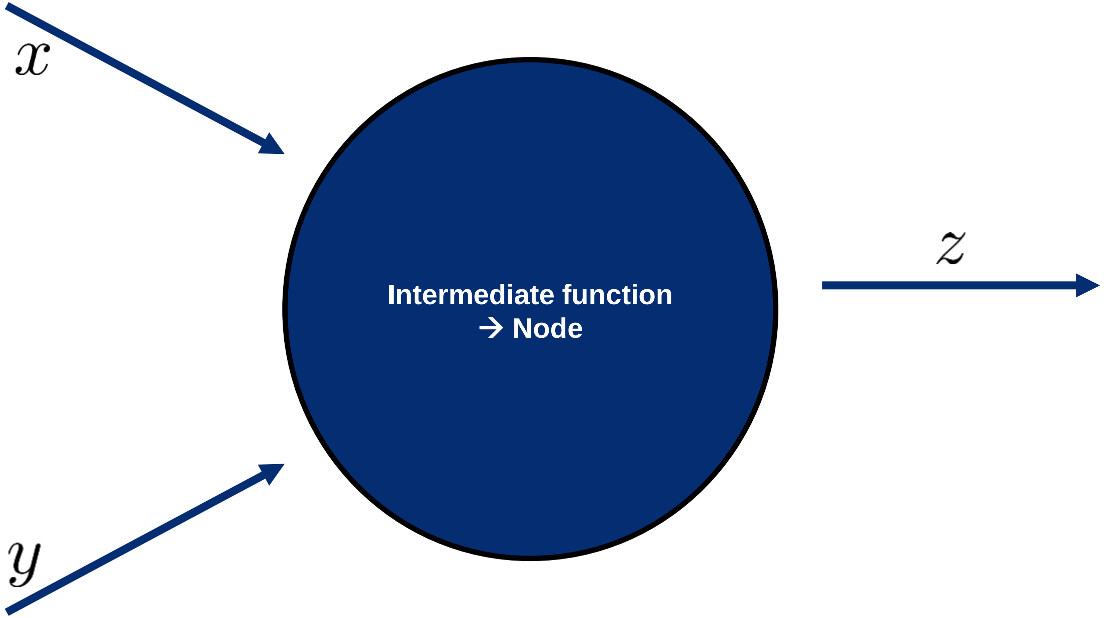
# Computing Backward Pass

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

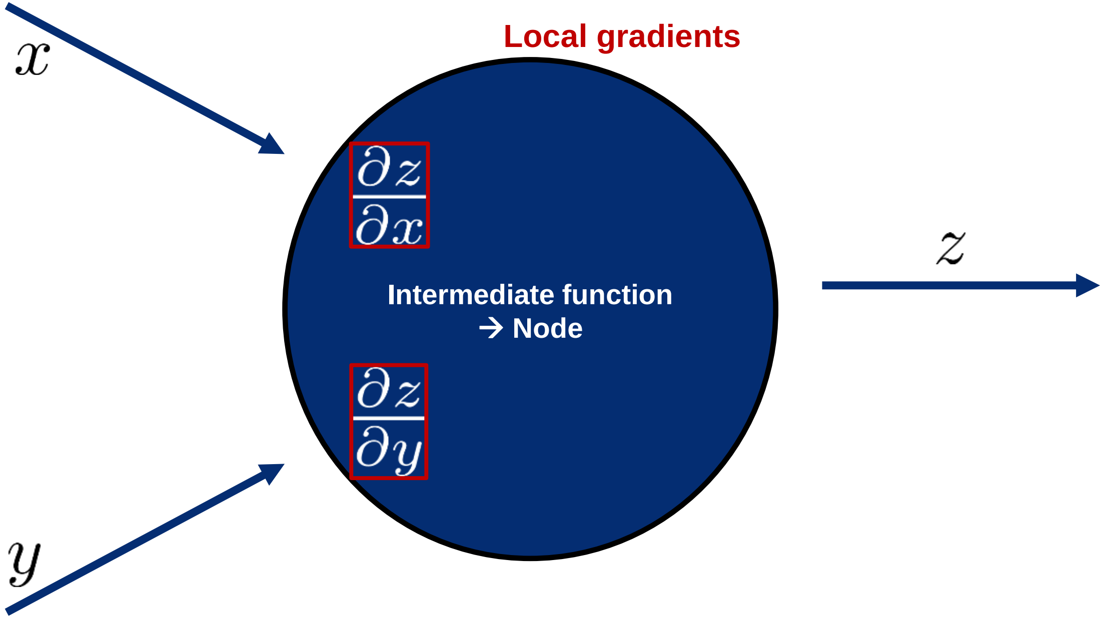
$$\frac{\partial f}{\partial q} = \frac{\partial q \cdot z}{\partial q} = z \quad \Bigg| \quad \frac{\partial q}{\partial x} = \frac{\partial x + y}{\partial x} = 1 \quad \Bigg| \quad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial z} = \frac{\partial q \cdot z}{\partial q} = q \quad \Bigg| \quad \frac{\partial q}{\partial y} = \frac{\partial x + y}{\partial y} = 1 \quad \Bigg| \quad \frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

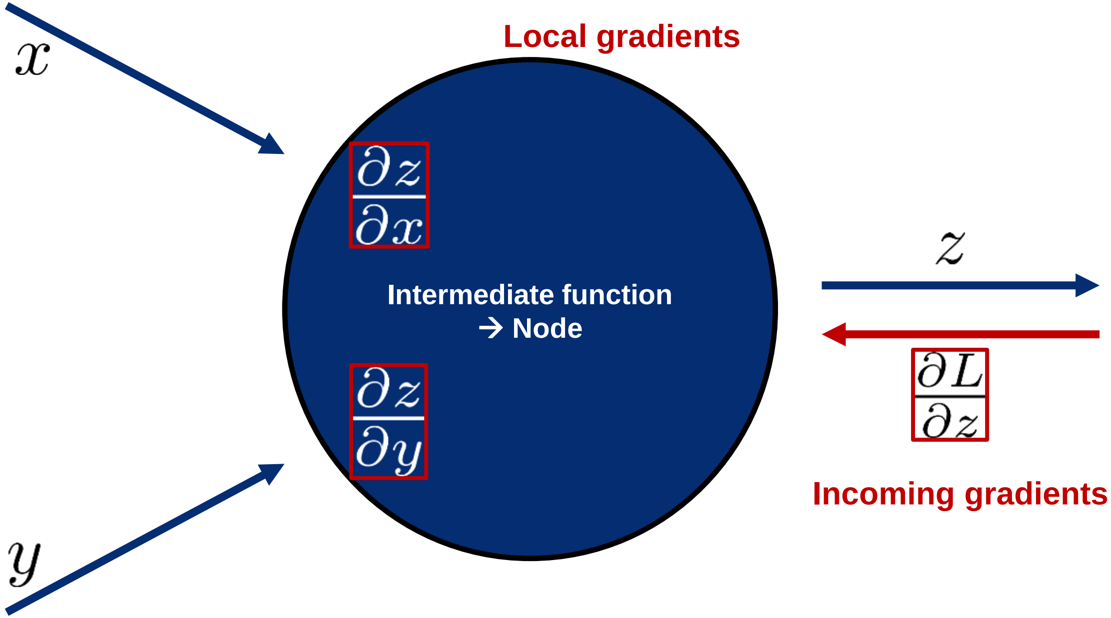




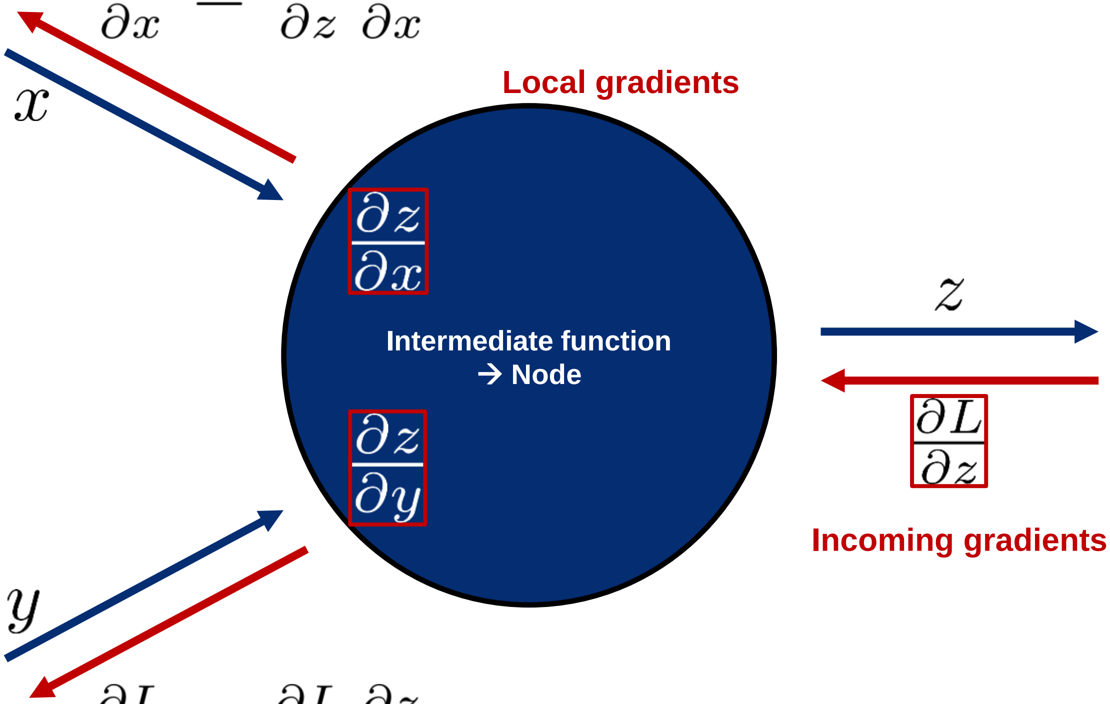
$x$
$y$
Intermediate function
→ Node
$z$



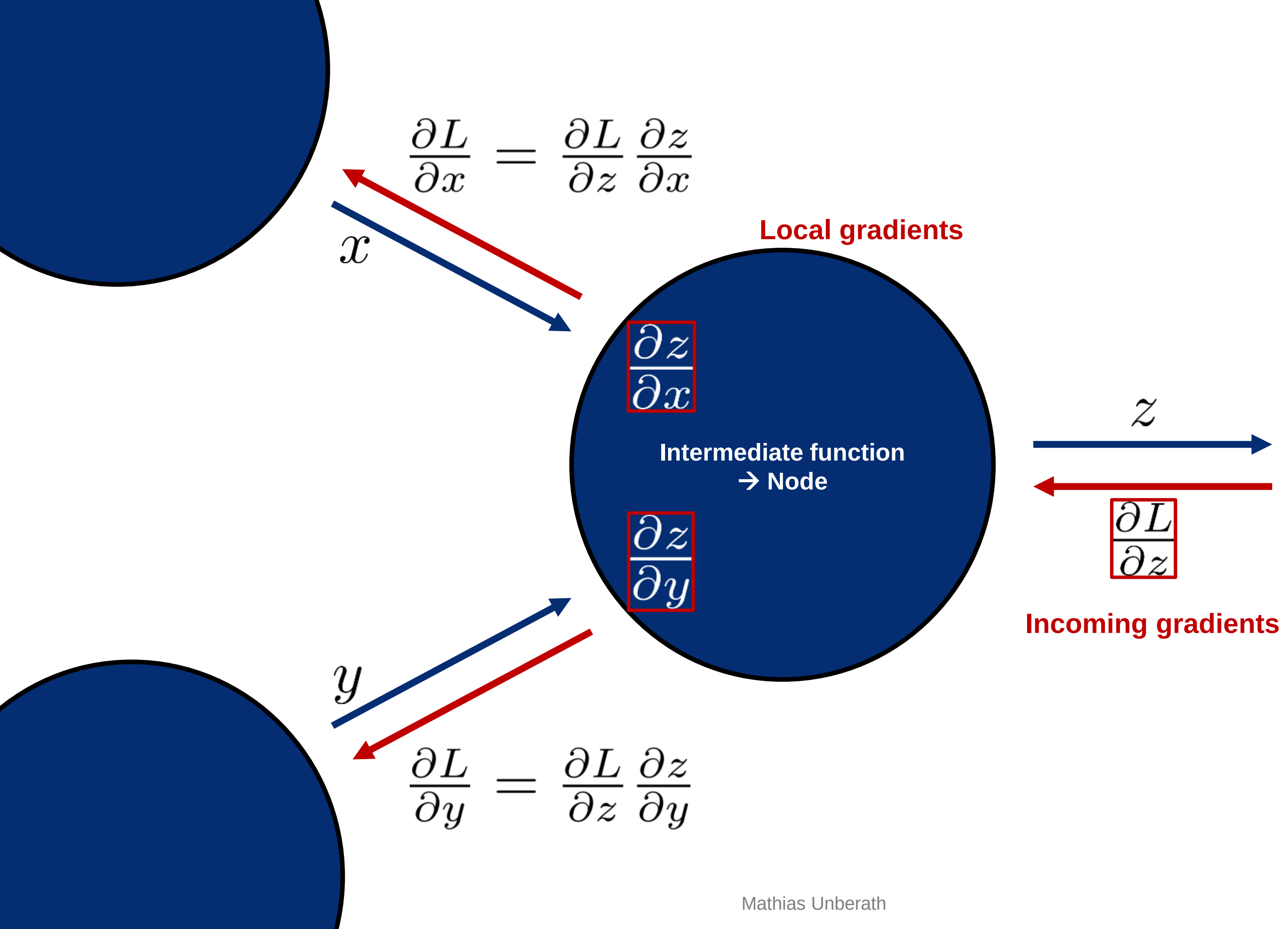
$x$
Local gradients
$\frac{\partial z}{\partial x}$
Intermediate function
→ Node
$\frac{\partial z}{\partial y}$
$z$
$y$


$x$
Local gradients
$\frac{\partial z}{\partial x}$
$z$
Intermediate function
→ Node
$\frac{\partial L}{\partial z}$
$\frac{\partial z}{\partial y}$
Incoming gradients
$y$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$




$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$$



$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
$x$
Local gradients
$\frac{\partial z}{\partial x}$
Intermediate function
→ Node
$\frac{\partial z}{\partial y}$
$z$
$\frac{\partial L}{\partial z}$
Incoming gradients
$y$
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$

# A More Complicated Example

$$f(w,x) = \frac{1}{1+e^{-(w_0 x + w_1)}}$$

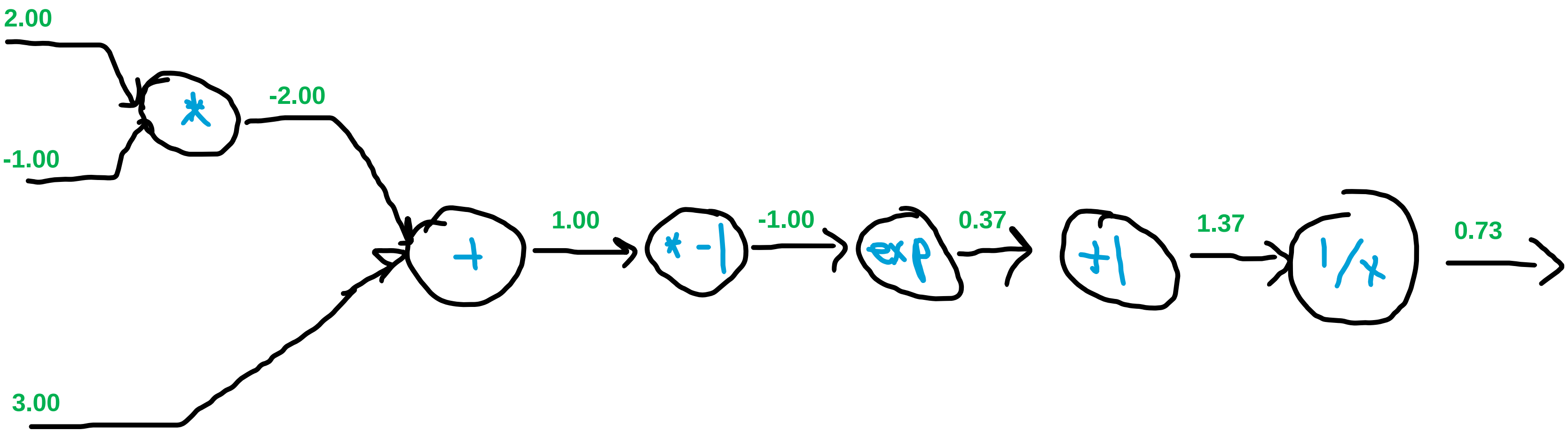
Tasks:

1. Set-up computational graph
2. Compute forward pass
3. Find derivatives $\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial x}$

# A More Complicated Example

$$f(w,x) = \frac{1}{1+e^{-(w_0 x + w_1)}}$$

## Forward pass



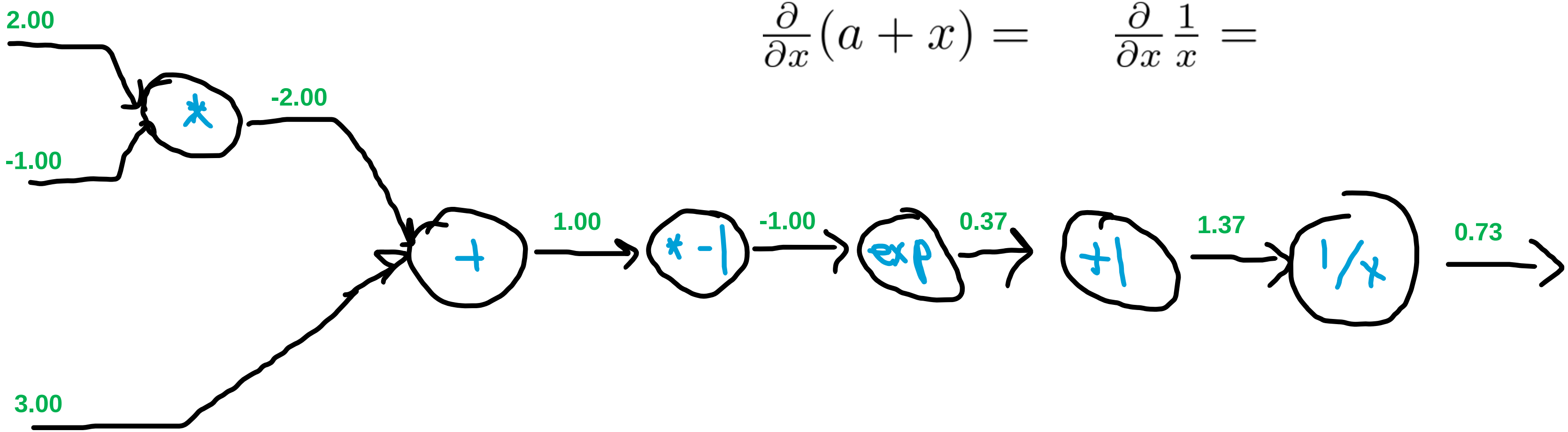
# A More Complicated Example

$$f(w,x) = \frac{1}{1+e^{-(w_0 x + w_1)}}$$

## Intermediate functions

$$\frac{\partial}{\partial x} a \cdot x = \qquad \frac{\partial}{\partial x} e^x =$$

$$\frac{\partial}{\partial x}(a+x) = \qquad \frac{\partial}{\partial x}\frac{1}{x} =$$



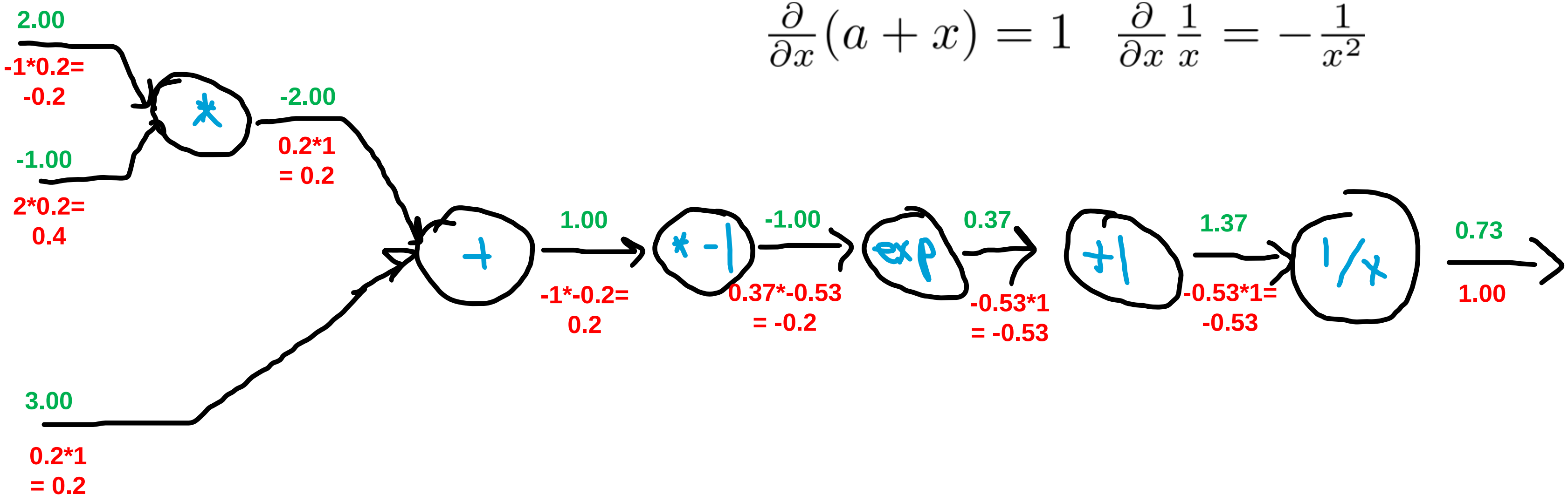
# A More Complicated Example

$$f(w,x) = \frac{1}{1+e^{-(w_0 x + w_1)}}$$

## Forward pass

$$\frac{\partial}{\partial x} a \cdot x = a \qquad \frac{\partial}{\partial x} e^x = e^x$$

$$\frac{\partial}{\partial x}(a+x) = 1 \qquad \frac{\partial}{\partial x}\frac{1}{x} = -\frac{1}{x^2}$$

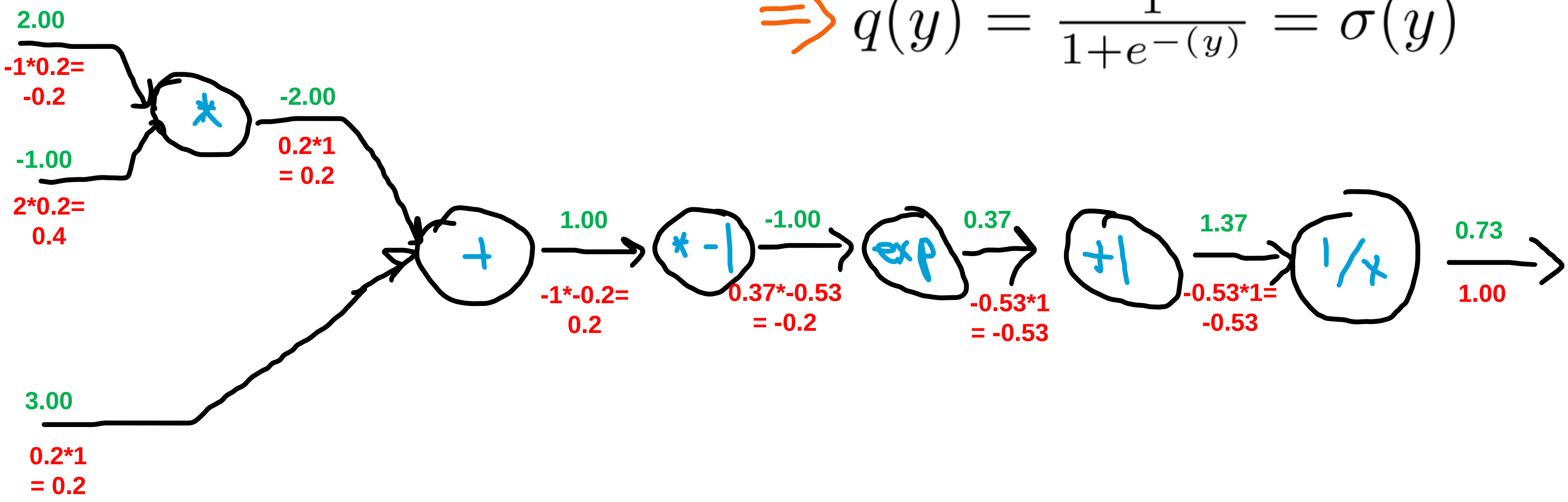


# A More Complicated Example

$$f(w,x) = \frac{1}{1+e^{-\underbrace{(w_0 x + w_1)}_{y}}}$$

## Forward pass

$$\Rightarrow q(y) = \frac{1}{1+e^{-(y)}} = \sigma(y)$$

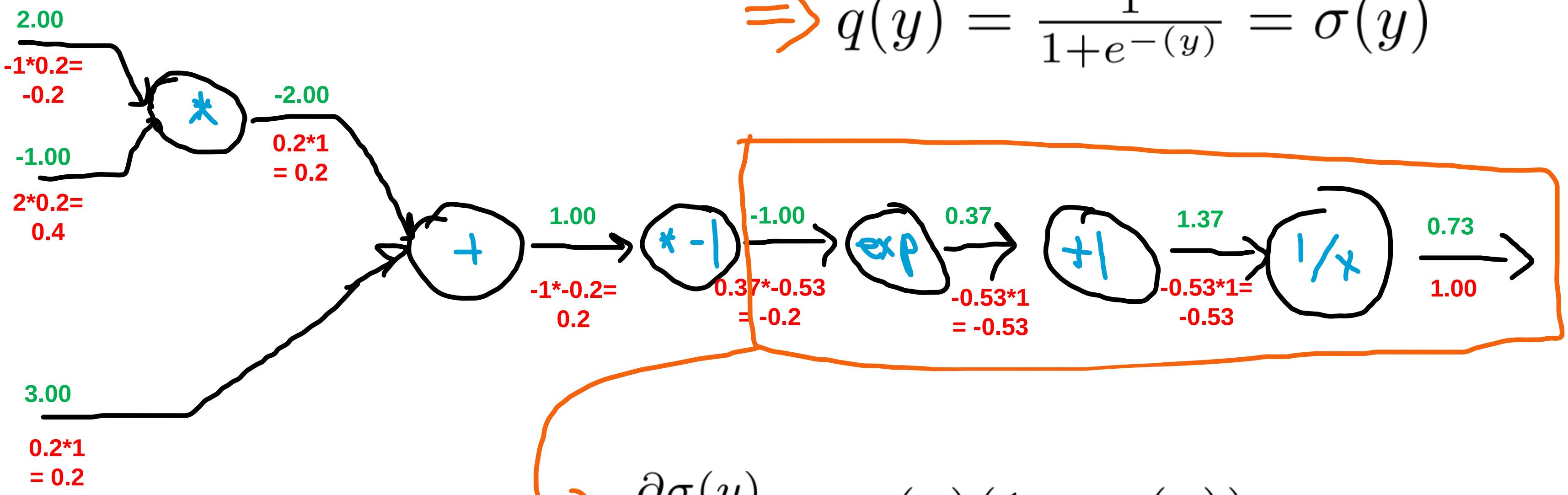


# A More Complicated Example

$$f(w,x) = \frac{1}{1+e^{-(\underbrace{w_0 x + w_1}_{y})}}$$

## Forward pass

$$\Rightarrow q(y) = \frac{1}{1+e^{-(y)}} = \sigma(y)$$




$$\frac{\partial \sigma(y)}{y} = \sigma(y)(1-\sigma(y))$$

# Patterns in Flow

**Add** node

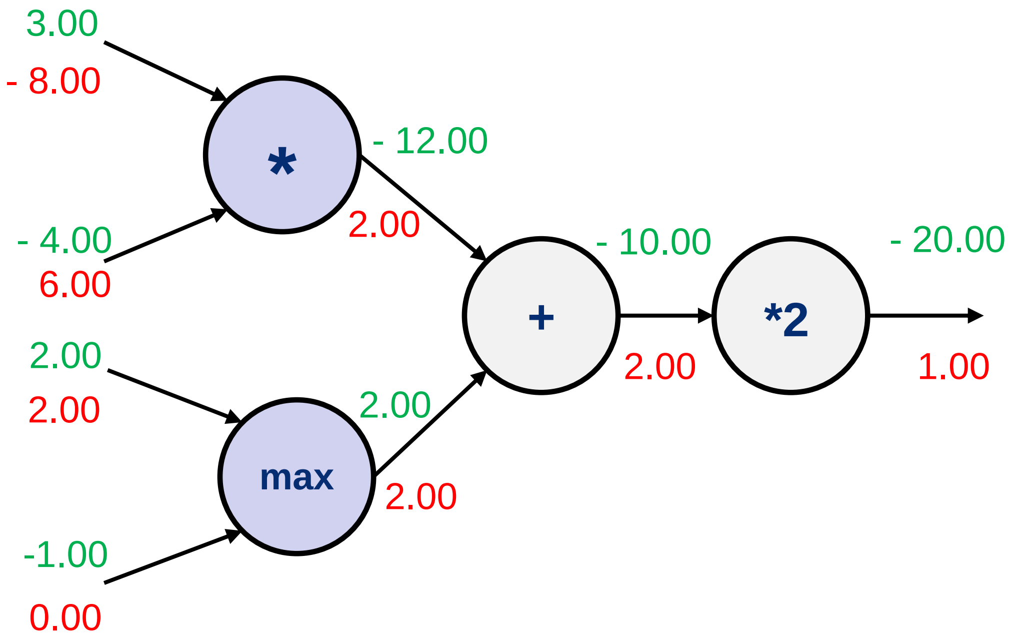
→ ?

**Max** node

→ ?

**Mul** node

→ ?

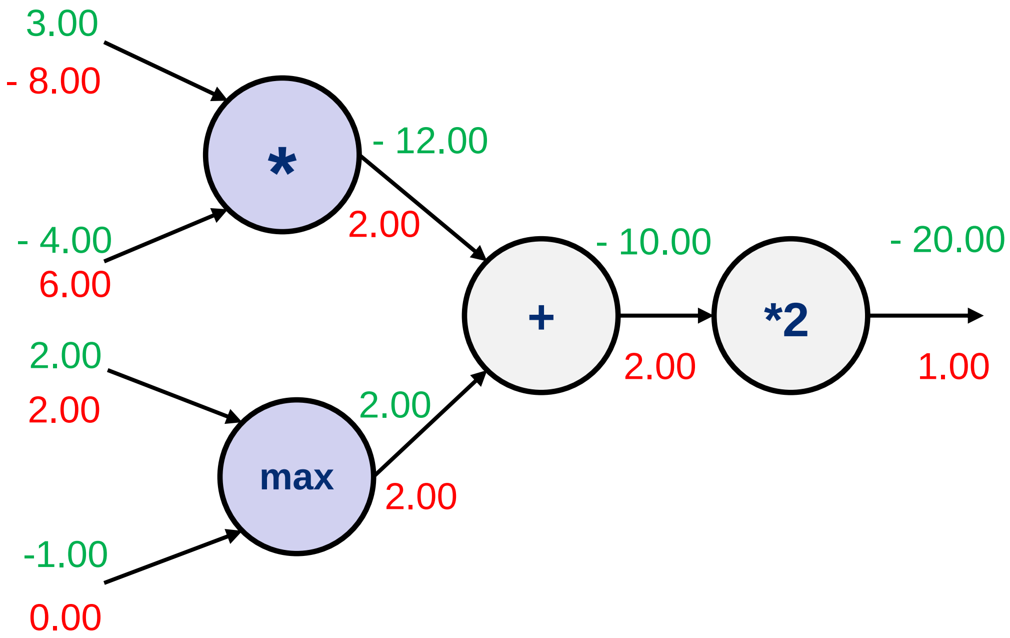


# Patterns in Flow

**Add** node
→ Distributes gradient

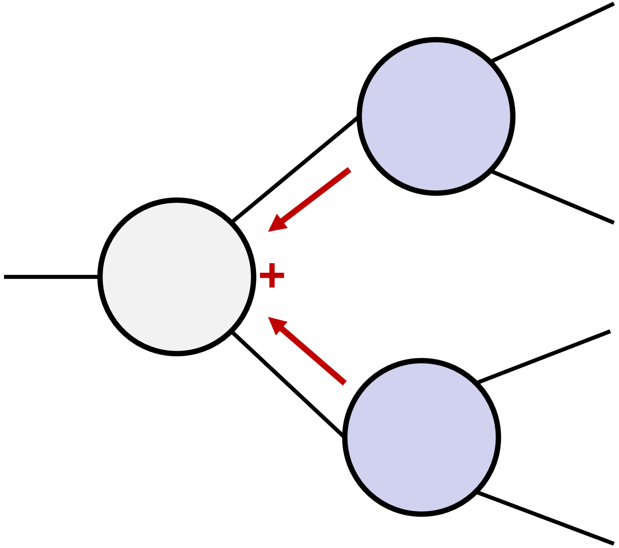
**Max** node
→ Routes gradient

**Mul** node
→ Switches gradient



# Patterns in Flow

Gradients add at branches



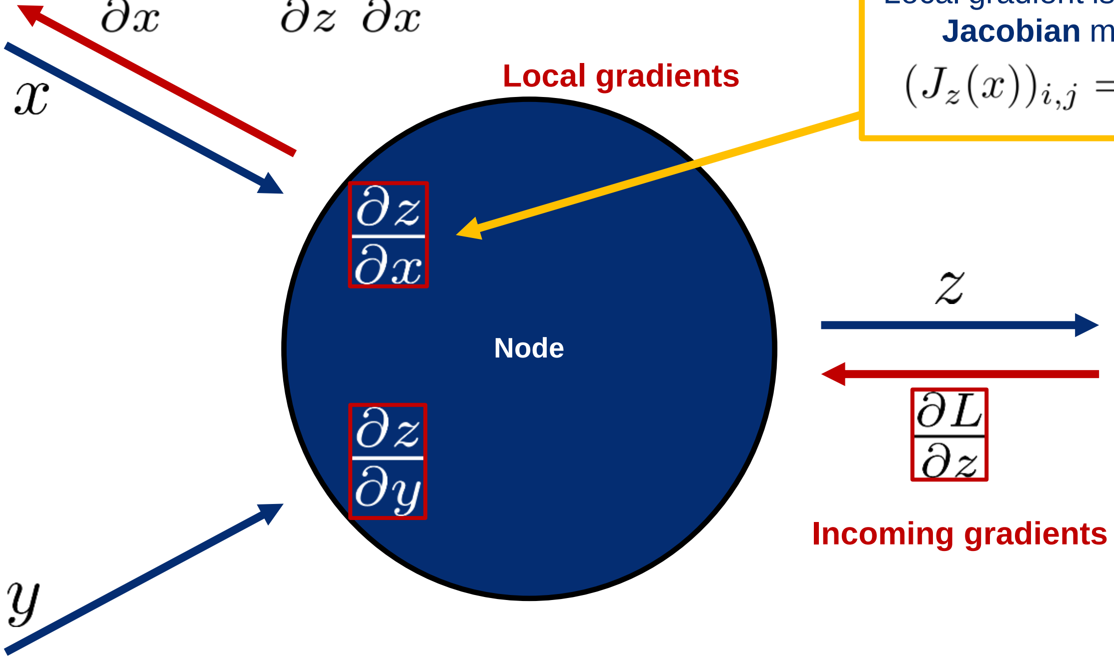
# Vector-valued Functions

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

Vector-valued function:
→ x,y,z are vectors!

Local gradient is now the **Jacobian** matrix.

$$(J_z(x))_{i,j} = \frac{\partial z_i}{\partial x_j}$$



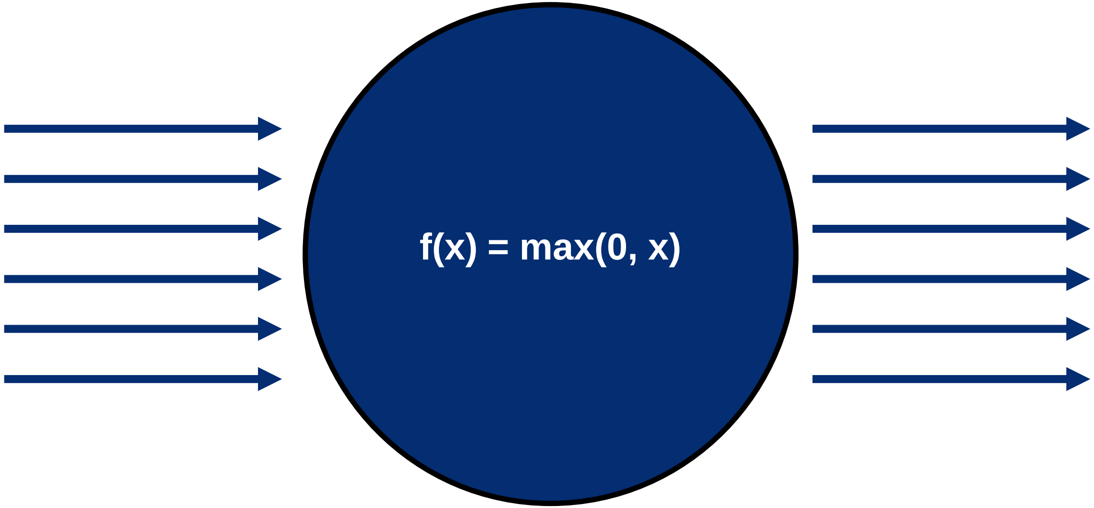
# Vector-valued Functions

Upstream

$$\frac{\partial L}{\partial x} = \boxed{\frac{\partial L}{\partial f}}\boxed{\frac{\partial f}{\partial x}}$$

Jacobian

**Input vector:**
**4096-dim**




**Output vector:**
**4096-dim**

**Q: Size of the Jacobian?**

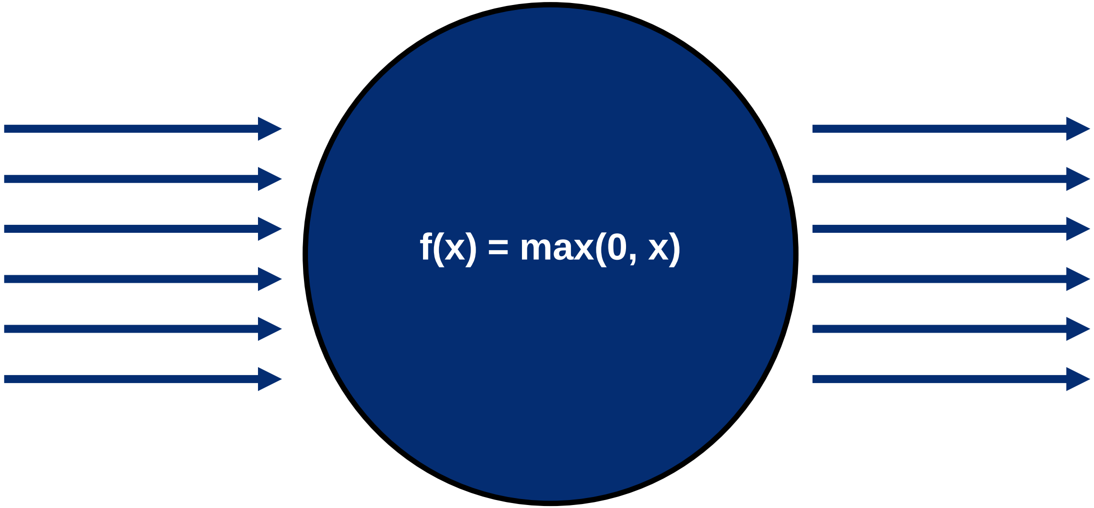
# Vector-valued Functions

Upstream

$$\frac{\partial L}{\partial x} = \boxed{\frac{\partial L}{\partial f}}\boxed{\frac{\partial f}{\partial x}}$$

Jacobian

**Input vector:**
**4096-dim**




**Output vector:**
**4096-dim**

**Q: Size of the Jacobian?**
**A: 4096 x 4096**

Processing of minibatches of size 100:
Jacobian would be 409,600 x 409,600.
This is ~ 625 GB.
→ J is diagonal, so that's good =)

# A Vectorized Example

$$f(W, x) : \mathbb{R}^{n \times n} \times \mathbb{R}^n \mapsto \mathbb{R}$$

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

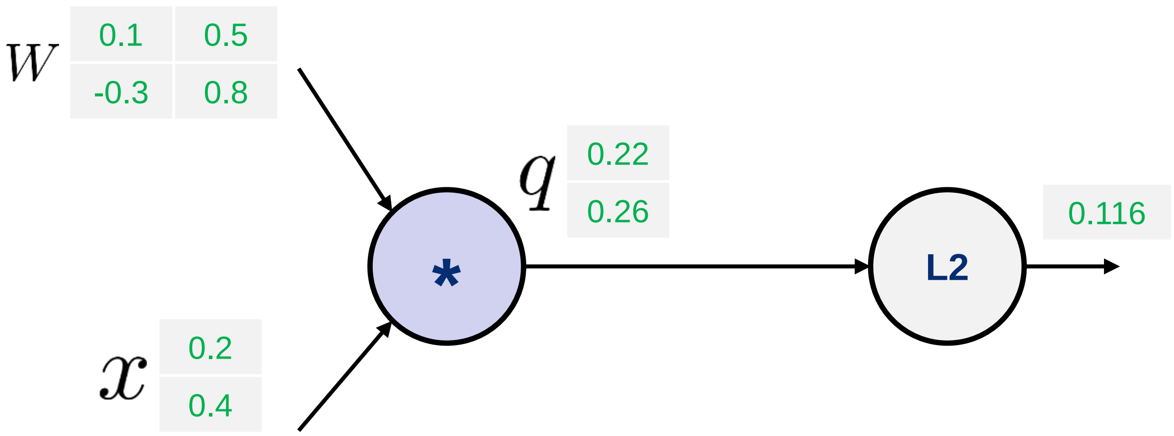
Tasks:

1. Set-up computational graph
2. Define appropriate intermediate functions
3. Find derivatives

# A Vectorized Example

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

## Forward pass



# A Vectorized Example

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

## Some gradient computations

This is zero almost always, unless l == i

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{else} \end{cases}$$

$$q_k = \sum_i W_{ki} \cdot x_i$$

$$\frac{\partial q_k}{\partial x_i} = \frac{\partial}{\partial x_i} \left( \sum_l W_{kl} \cdot x_l \right) = \sum_l W_{kl} \boxed{\frac{\partial x_l}{\partial x_i}} = \sum_l W_{kl} \boxed{\delta(l - i)} = W_{ki}$$

$$\frac{\partial q_k}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \left( \sum_l W_{kl} \cdot x_l \right) = \sum_l \boxed{\frac{\partial W_{kl}}{\partial W_{ij}}} x_l = \sum_l \delta(i - k)\delta(l - j) x_l = \delta(i - k) x_j$$

$$\frac{\partial f}{\partial q_k} = \frac{\partial}{\partial q_k} \left( \sum_l q_l^2 \right) = \sum_l \frac{\partial q_l^2}{\partial q_k} = 2 q_k$$

# A Vectorized Example

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

## Some gradient computations

$$q_k = \sum_i W_{ki} \cdot x_i$$

$$\frac{\partial q_k}{\partial x_i} = W_{ki}$$

$$\frac{\partial q_k}{\partial W_{ij}} = \delta(i-k) x_j$$

$$\boxed{\frac{\partial f(q(x))}{\partial x_i} = \frac{\partial f}{\partial (q_1, \ldots, q_m)} \frac{\partial (q_1, \ldots, q_m)}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i}}$$

Can be proved via total derivative, not relevant here.

$$\boxed{\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \sum_k 2 q_k W_{ki}}$$

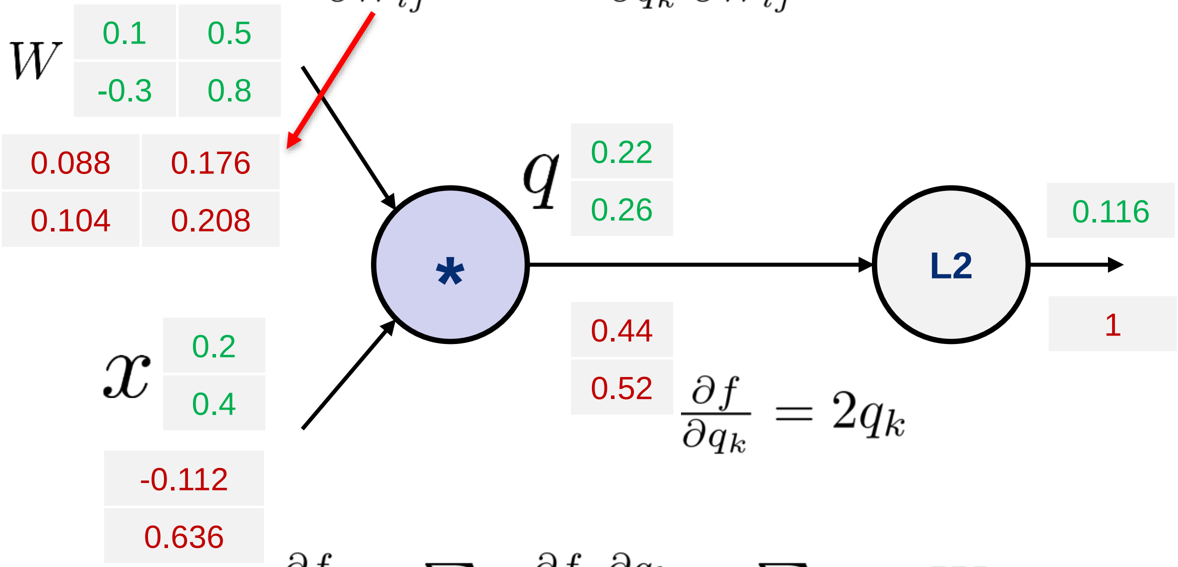
$$\boxed{\frac{\partial f}{\partial W_{ij}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{ij}} = \sum_k 2 q_k \delta(i-k) x_j = 2 q_i x_j}$$

# A Vectorized Example

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

## Forward pass

$$\frac{\partial f}{\partial W_{ij}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{ij}} = 2 q_i x_j$$




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

$$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \sum_k 2 q_k W_{ki}$$

# Summary

- **Computational graphs**
  Split complex formulas into easy components

- **Backpropagation**
  Recursive application of chain rule yields analytic gradients

- **Implementation**
  Nodes of the computational graph implement *forward*() / *backward*()

# Part II of this Lecture is Available Online!

- More explanations
- Matrix examples
- …

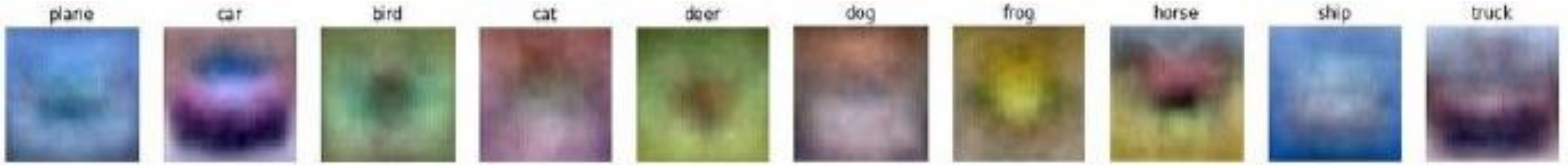
Part II will be uploaded to Piazza for self-review at home.

And now finally: **Towards Neural Networks and Deep Learning**

# Towards Neural Networks

**Until now**: Linear score function $f(x, W) = Wx$

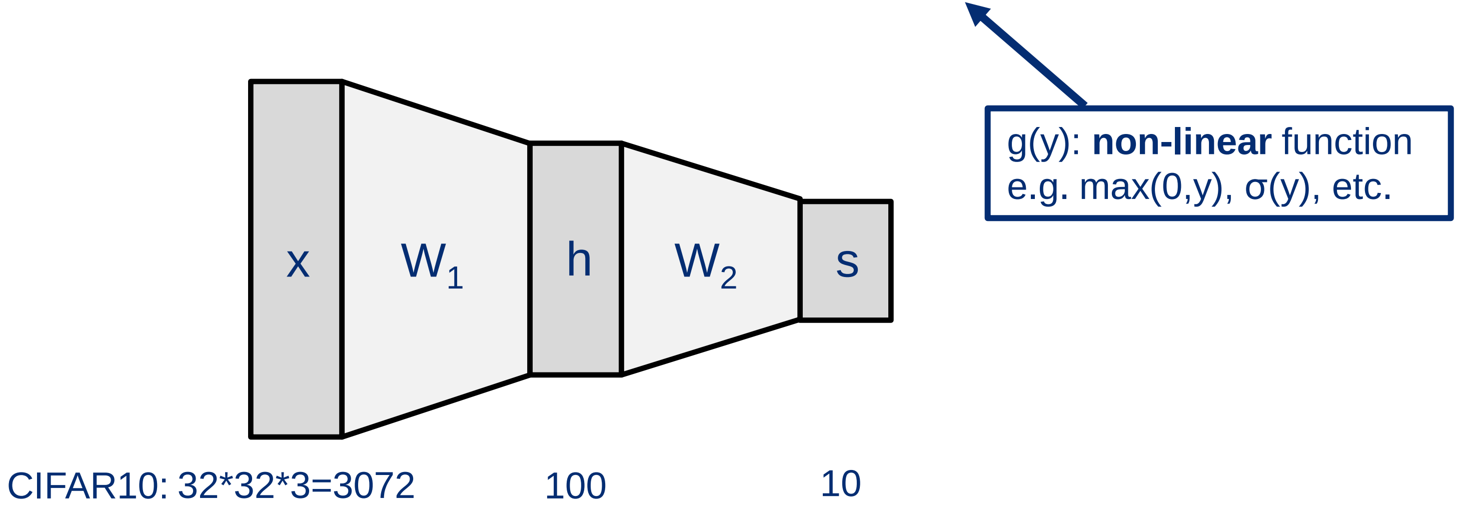
f(x,W) → **class scores**

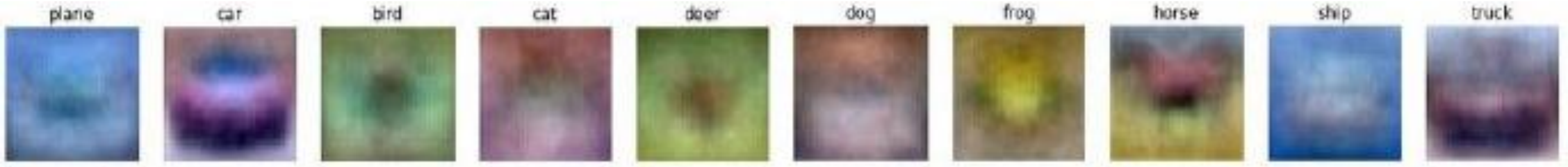


# Towards Neural Networks

**Until now**: Linear score function $f(x, W) = Wx$

**From now**: Layered score functions $f(x, \{W_i\}) = W_2(g(W_1 x))$

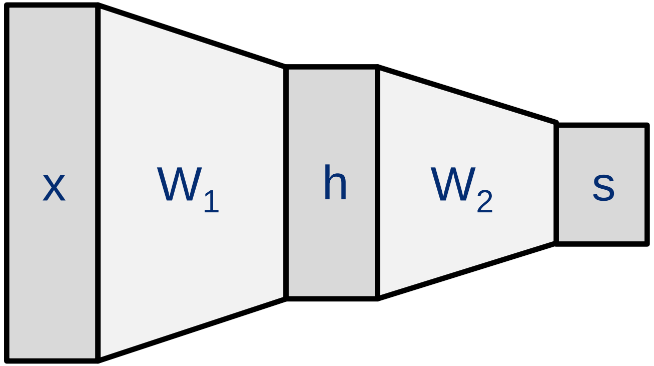


# Towards Neural Networks

**Until now**: Linear score function $f(x, W) = Wx$

**From now**: Layered score functions $f(x, \{W_i\}) = W_2(g(W_1 x))$




→ Can be many more layers!

Compute Graphs and Backpropagation

# Questions?