

Indian Institute of Information Technology, Allahabad



Introduction to Machine Learning Neural Networks

By

Dr. Shiv Ram Dubey

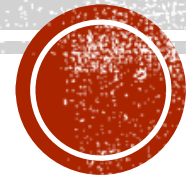
Associate Professor

Computer Vision and Biometrics Lab

Department of Information Technology

Indian Institute of Information Technology, Allahabad

Web: <https://profile.iita.ac.in/srdubey/>

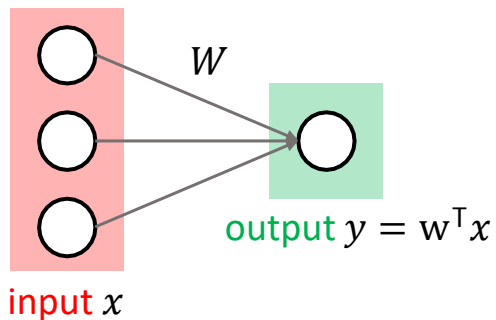


DISCLAIMER

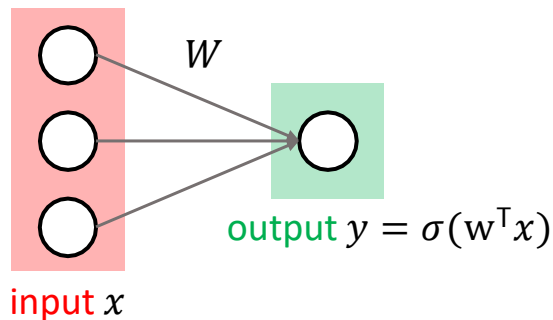
The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

So far in this class: A Unifying View

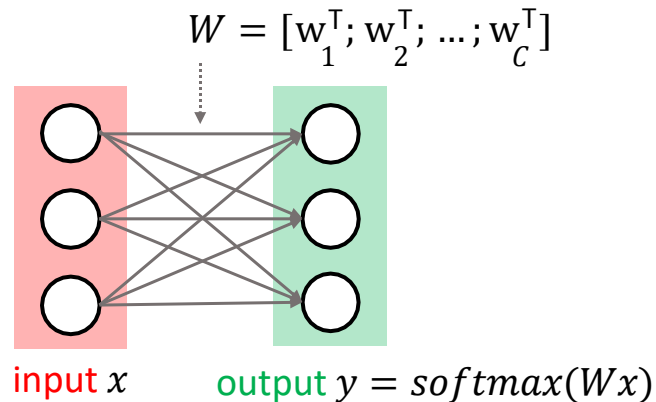
Linear Regression



Binary Classification



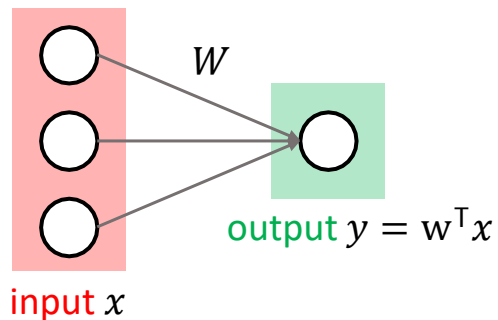
Multi-Class Classification



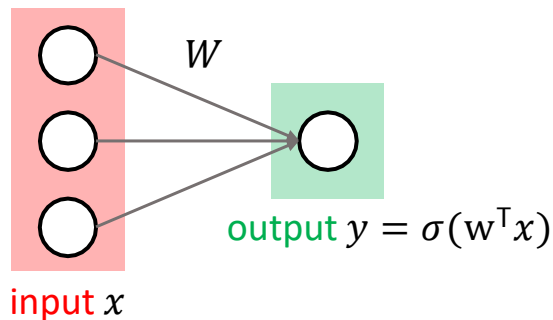
So far in this class: A Unifying View

Linear transformation of input features followed by an activation function

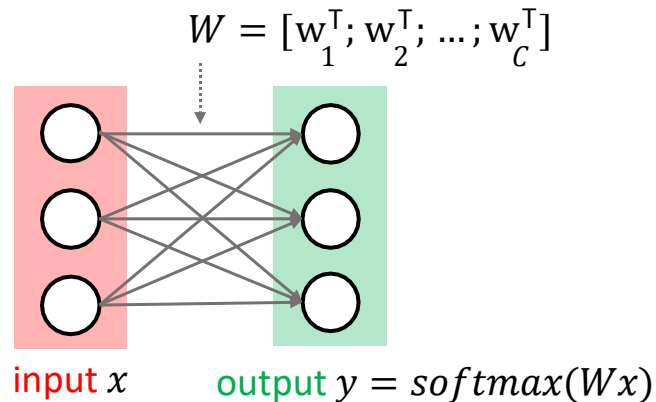
Linear Regression



Binary Classification



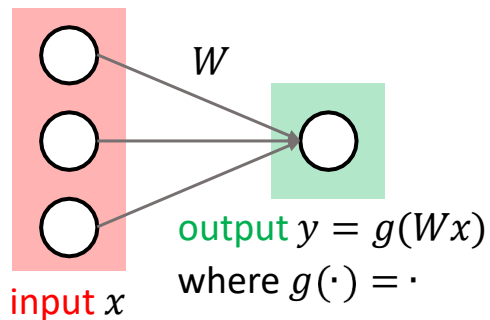
Multi-Class Classification



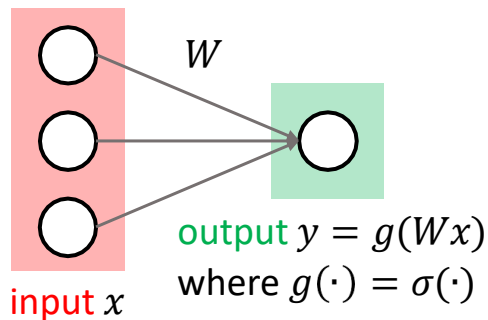
So far in this class: A Unifying View

Linear transformation of input features (Wx) followed by an activation function $g(\cdot)$

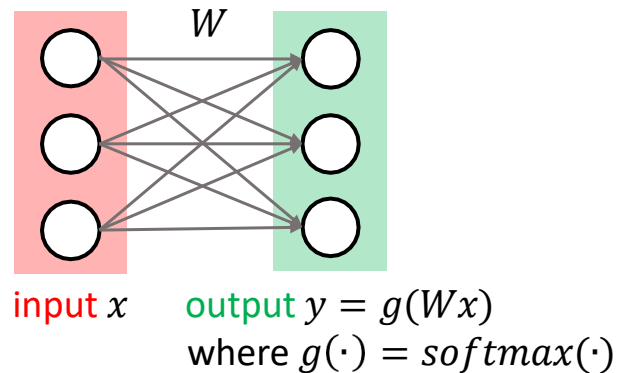
Linear Regression



Binary Classification



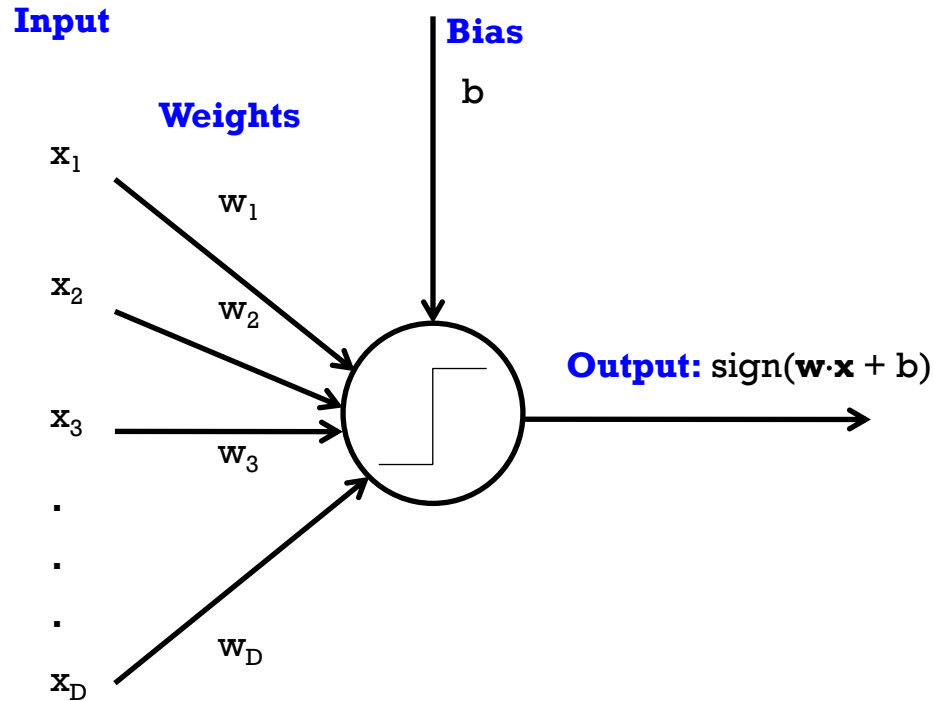
Multi-Class Classification



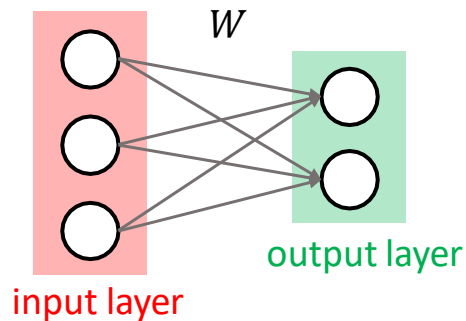
$W \in \mathbb{R}^{C \times D}$ where D is the input dimension and C is the output dimension.

Perceptron

- Supervised learning of binary classifier



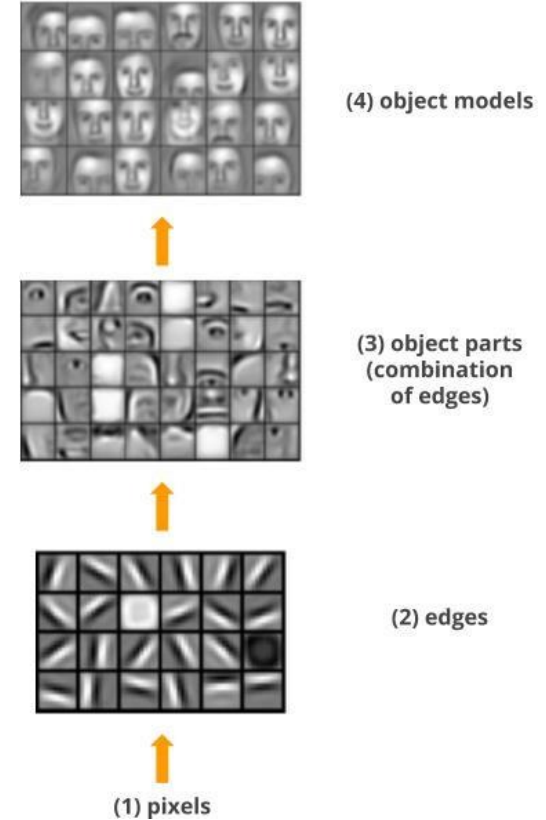
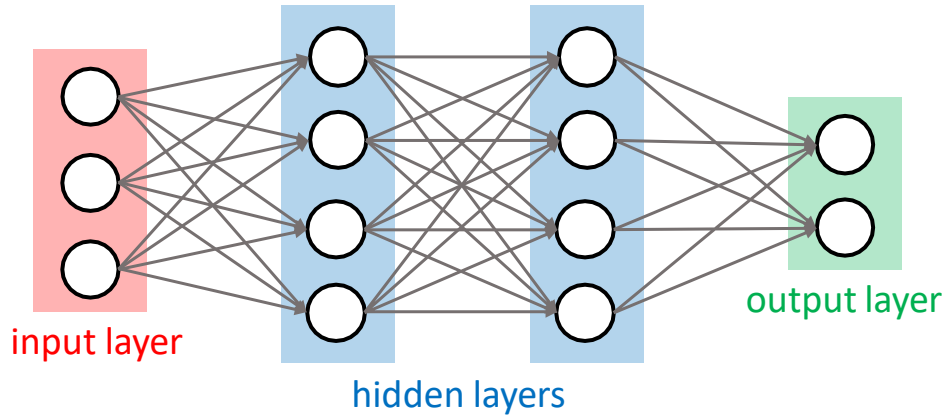
A Unifying View: Single-Layer Neural Network



- Challenge: it needs “good” input features
- Most ML work before focused on hand designing features

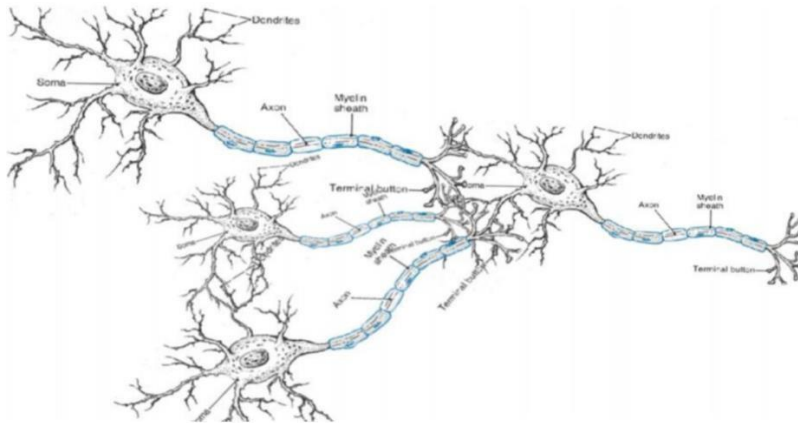
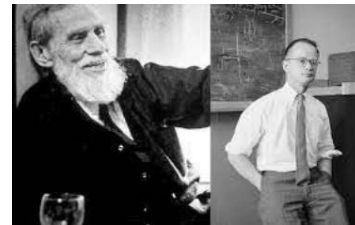
The “Promise” of Deep Neural Networks

- **Representation Learning:** automatically learn good features for tasks
- **Deep Learning:** learn multiple levels of representation at increasing levels of complexity



Inspired by Simplified Models of Brain Neurons

- **1943:** Perceptron model (McCulloch & Pitts)
 - Intended as theoretical model of biological neurons



“Dark Ages”

- **1969:** Perceptrons cannot learn XOR (Minsky & Papert)
 - Highly controversial
- **1998:** Convolutional neural networks for MNIST (Lecun)
 - Human-level performance on handwritten digit recognition
- **1997:** Long Short-term Memory Networks (Hochreiter and Schmidhuber)

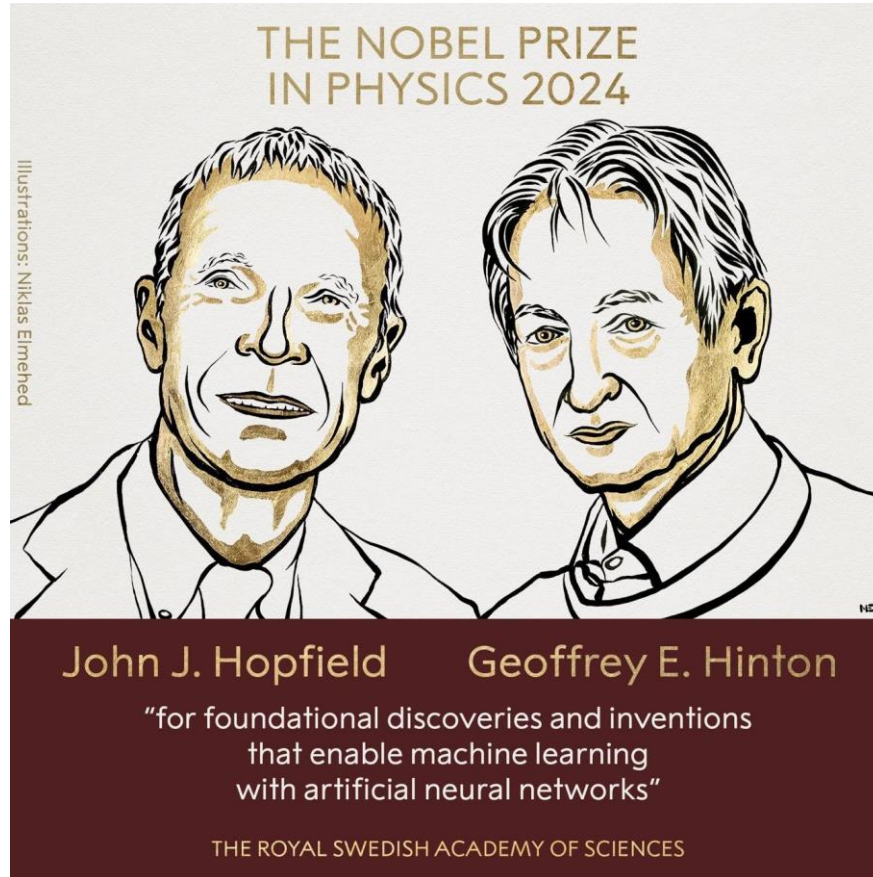


Extremely
Similar Today

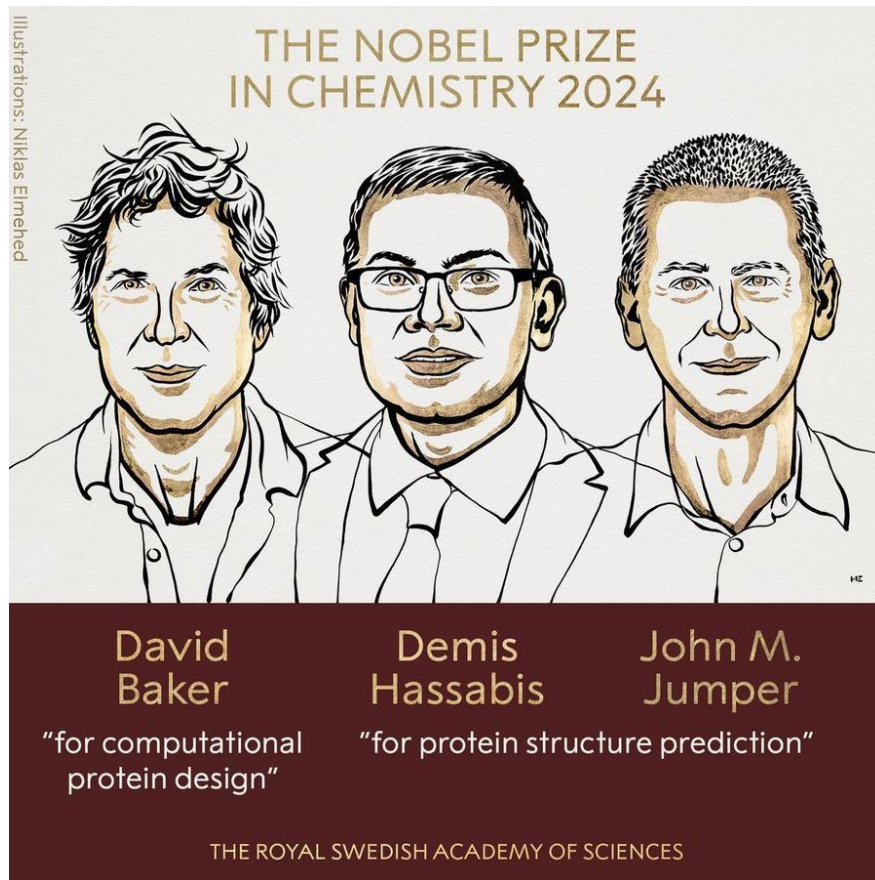
2012 - NOW

- **2012:** ImageNet breakthrough (Krizhevsky, Sutskever, & Hinton)
 - Reduced error on image classification by 50%
- **2017:** Transformer architecture (Vaswani et al.)
- **2018:** Turing award (Bengio, Hinton, & Lecun)

The Nobel Prize in Physics 2024



The Nobel Prize in Chemistry 2024



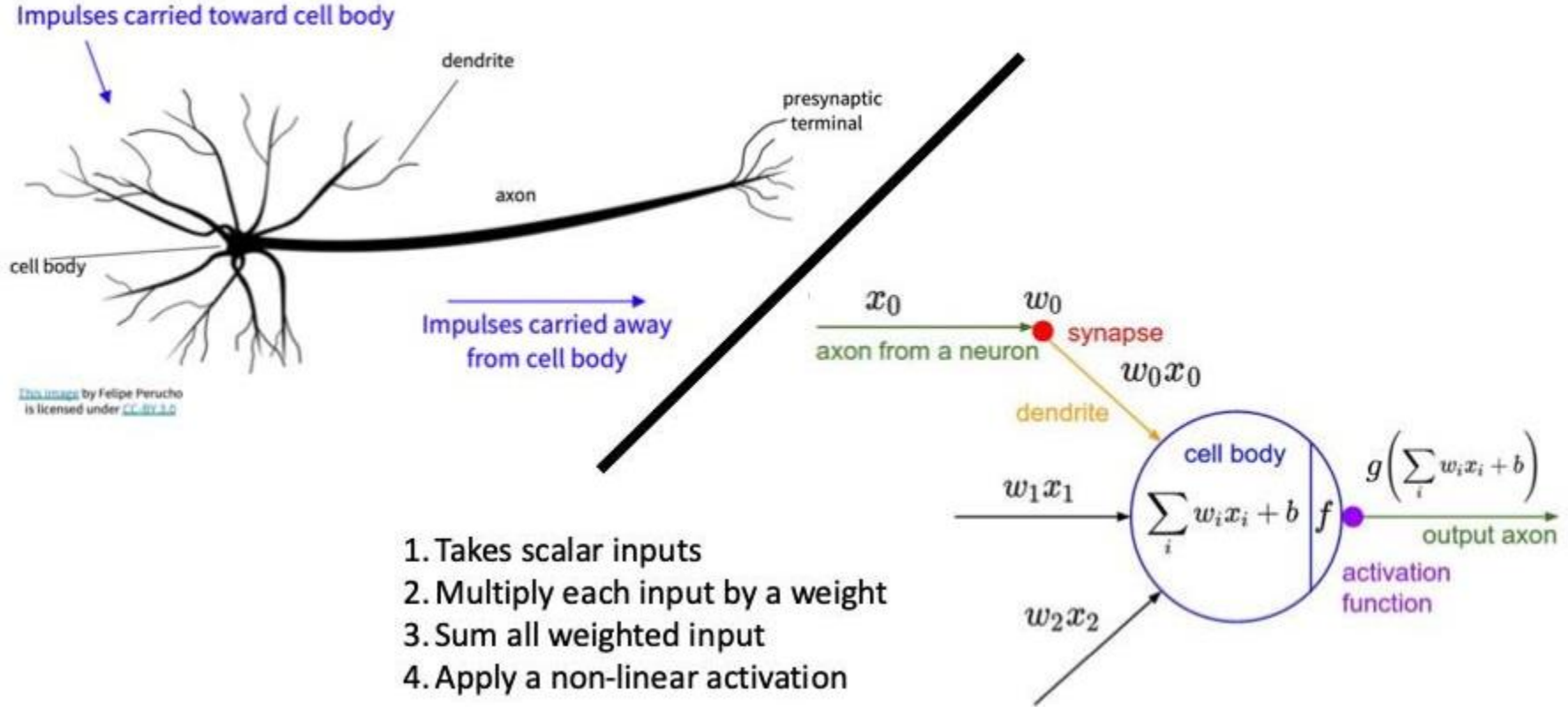
2012 - NOW

- **2012:** ImageNet breakthrough (Krizhevsky, Sutskever, & Hinton)
 - Reduced error on image classification by 50%
- **2017:** Transformer architecture (Vaswani et al.)
- **2018:** Turing award (Bengio, Hinton, & Lecun)
- **2024:** The Nobel Prize in Physics & Chemistry
- **Generative AI & LLMs... To be continued?**

Why Wasn't it Working Before?

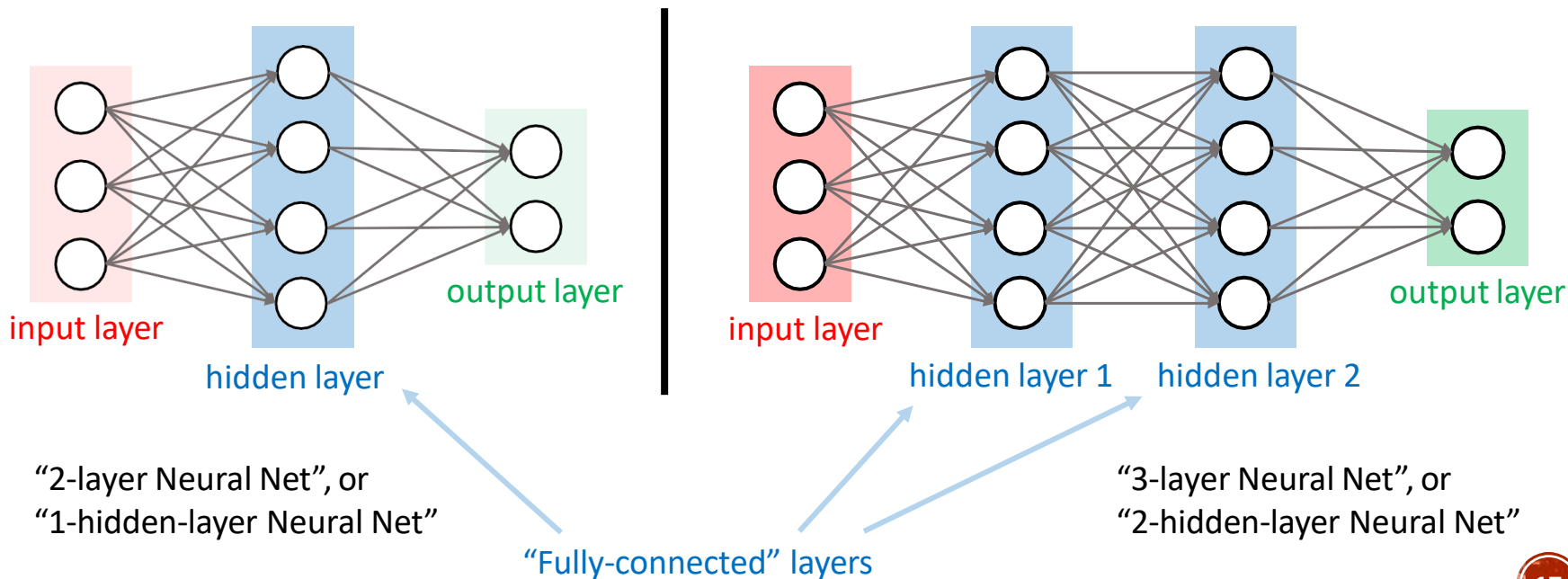
- Small Datasets & Less Capable Hardware
 - Machine translation needs millions of sentences to see improvements
- Missing bag of tricks for optimization
 - Regularization like Batch Normalization and Dropout
- Some domain specific tricks & architectures
 - Word embedding for NLP

Simplified Brain Modeling: Loose Inspiration

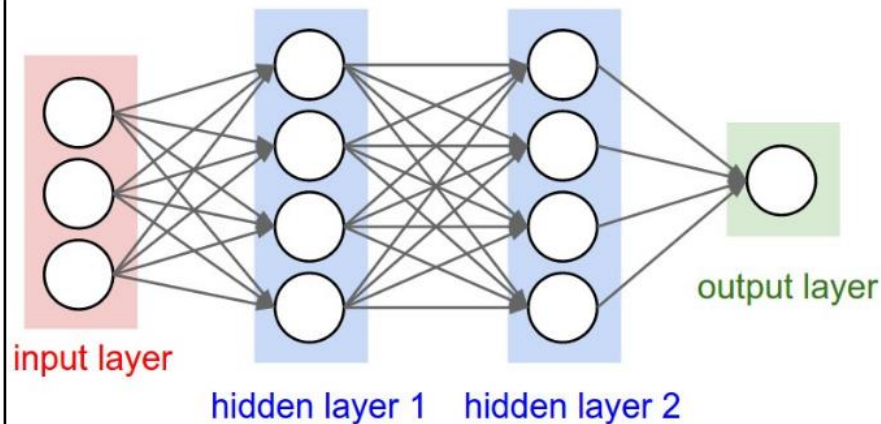
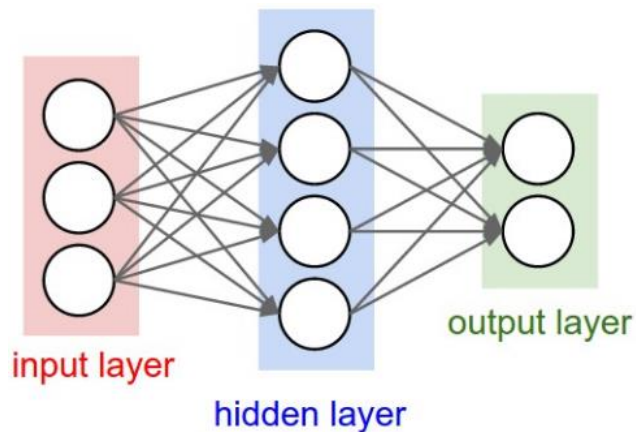


Feed-Forward Neural Networks

- Signals move in one direction – forward – with no cycles or loops.
- Also called Multi-Layer Perceptrons (MLP)



SIZING NEURAL NETWORKS

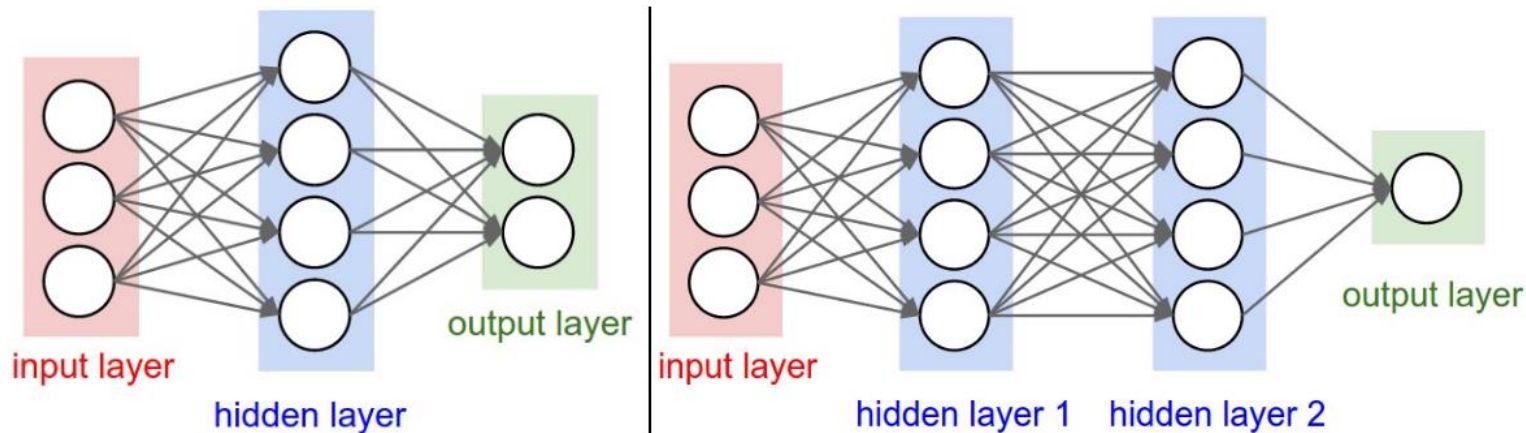


First network (left):

No. of neurons (not counting the inputs):

No. of learnable parameters:

Sizing Neural Networks

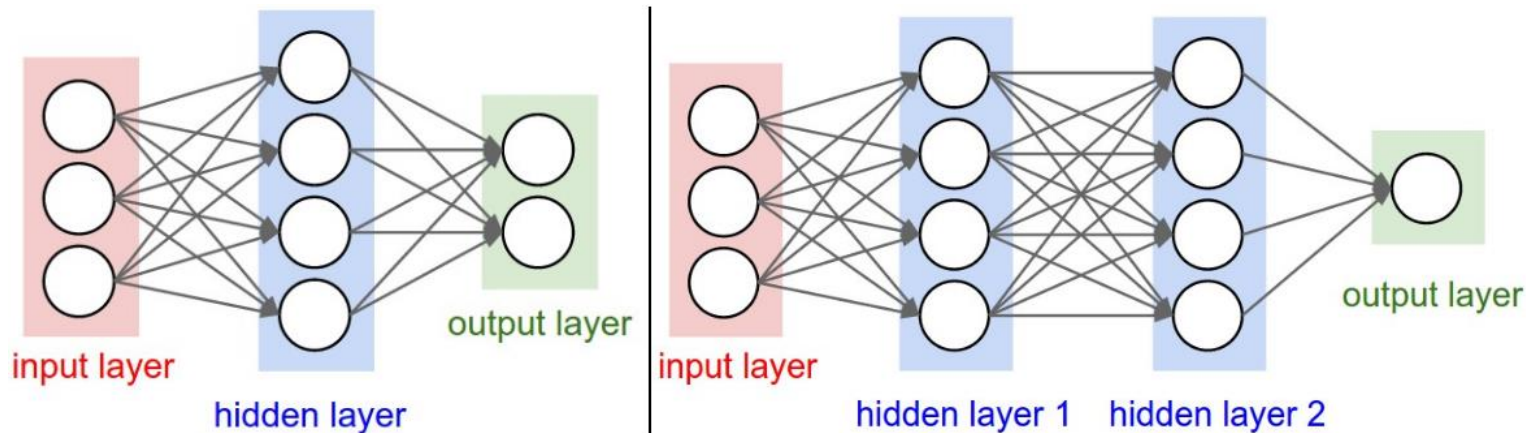


First network (left):

No. of neurons (not counting the inputs): $4 + 2 = 6$

No. of learnable parameters:

Sizing Neural Networks

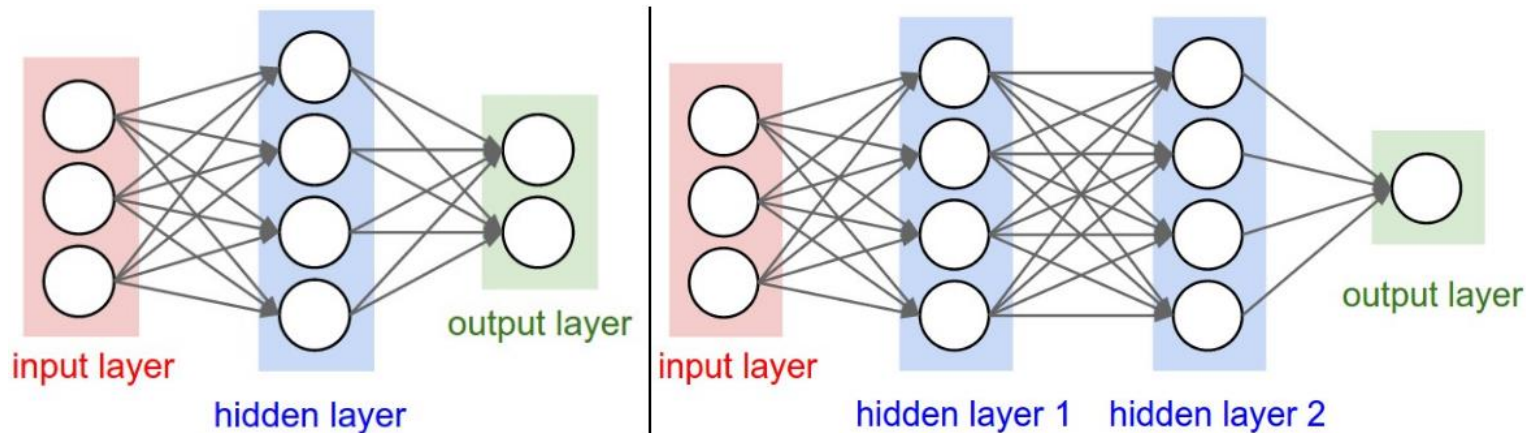


First network (left):

No. of neurons (not counting the inputs): $4 + 2 = 6$

No. of learnable parameters: $[3 \times 4] + [4 \times 2] = 20$ weights +
 $4 + 2 = 6$ biases = 26.

Sizing Neural Networks



First network (left):

No. of neurons (not counting the inputs): $4 + 2 = 6$

No. of learnable parameters: $[3 \times 4] + [4 \times 2] = 20$ weights +
 $4 + 2 = 6$ biases = 26.

Second network (right):

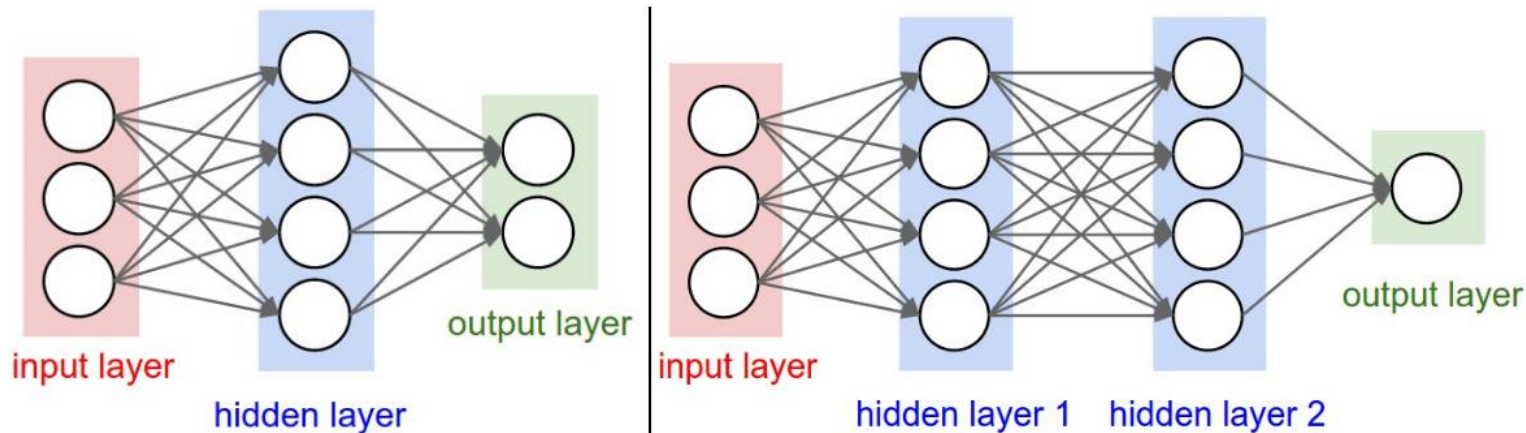
No. of neurons (not counting the inputs):

No. of learnable parameters:

Source:

<http://cs231n.github.io>

Sizing Neural Networks



First network (left):

No. of neurons (not counting the inputs): $4 + 2 = 6$

No. of learnable parameters: $[3 \times 4] + [4 \times 2] = 20$ weights +
 $4 + 2 = 6$ biases = 26.

Second network (right):

No. of neurons (not counting the inputs): $4 + 4 + 1 = 9$

No. of learnable parameters: $[3 \times 4] + [4 \times 4] + [4 \times 1] = 32$ weights +
 $4 + 4 + 1 = 9$ biases = 41.

Source:

<http://cs231n.github.io>

Matrix Notation

1 layer Neural Net: $y = W_1 x$

2 layer Neural Net: $y = W_2 g(W_1 x)$

3 layer Neural Net: $y = W_3 g(W_2 g(W_1 x))$

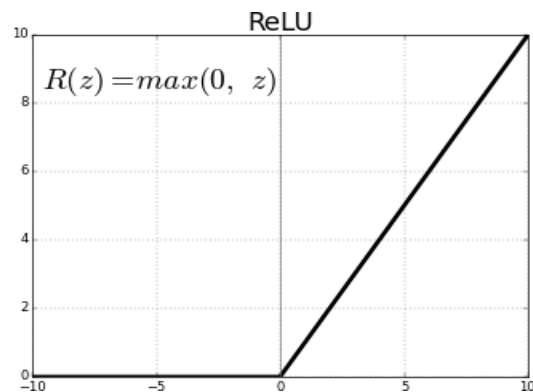
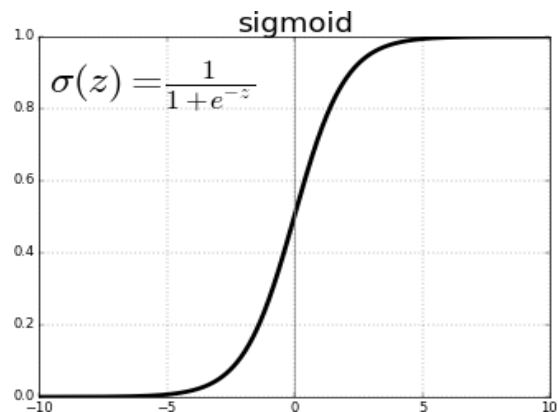
$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H_1 \times D}, W_2 \in \mathbb{R}^{H_2 \times H_1}, W_3 \in \mathbb{R}^{H_3 \times H_2}$$

g is a non-linear activation
function for hidden layers

(In practice we will usually add a learnable bias at each layer as well)

Non-Linearity g

- Sigmoid activation function:
 - Outputs values between 0 and 1
 - Probability of neuron firing/activated
- ReLU (Rectified Linear Unit):
 - Efficient computation
 - Doesn't saturate
 - Most commonly used today



Why Non-Linearity?

Q: What if we try to build a neural network without one?

Why Non-Linearity?

Q: What if we try to build a neural network without one?

2-layer Neural Net: $y = W_2 g(W_1 x)$ $\longrightarrow$ $y = W_2 W_1 x$

3-layer Neural Net: $y = W_3 g(W_2 g(W_1 x))$ $\longrightarrow$ $y = W_3 W_2 W_1 x$

Why Non-Linearity?

Q: What if we try to build a neural network without one?

2-layer Neural Net: $y = W_2 g(W_1 x)$ $\longrightarrow$ $y = W_2 W_1 x$

3-layer Neural Net: $y = W_3 g(W_2 g(W_1 x))$ $\longrightarrow$ $y = W_3 W_2 W_1 x$

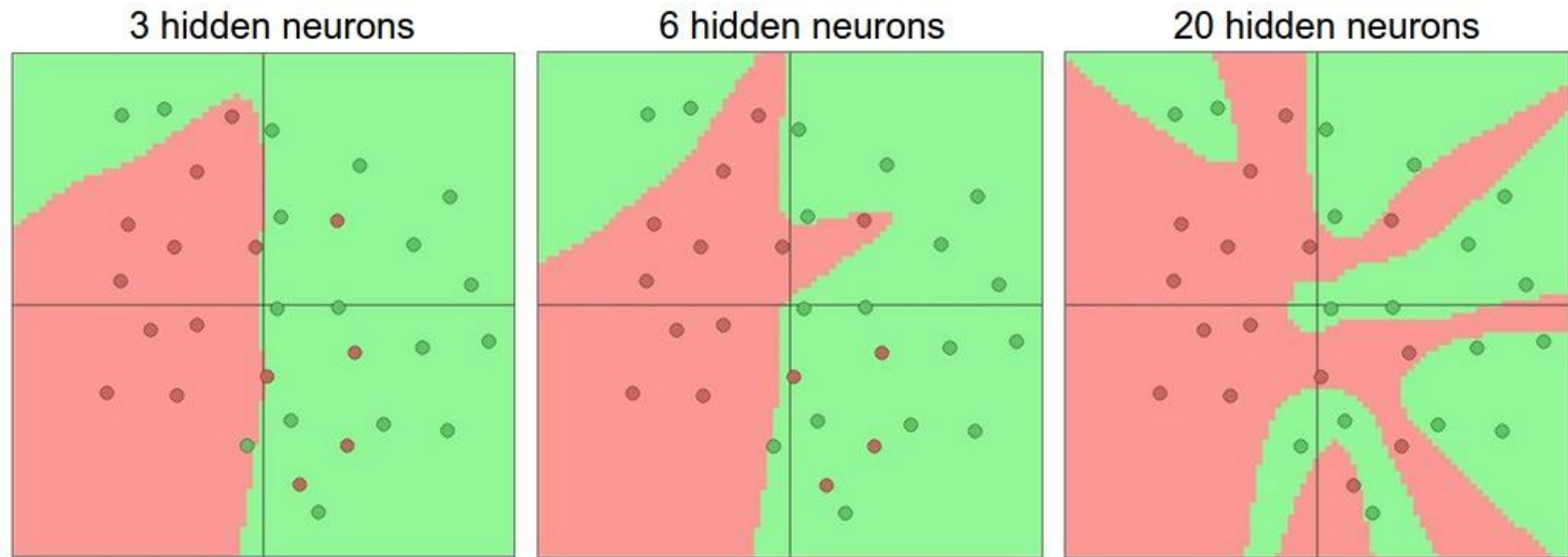
A: We would end up with linear classifier!

Non-Linearities are important for learning features/representations with increasing levels of complexity

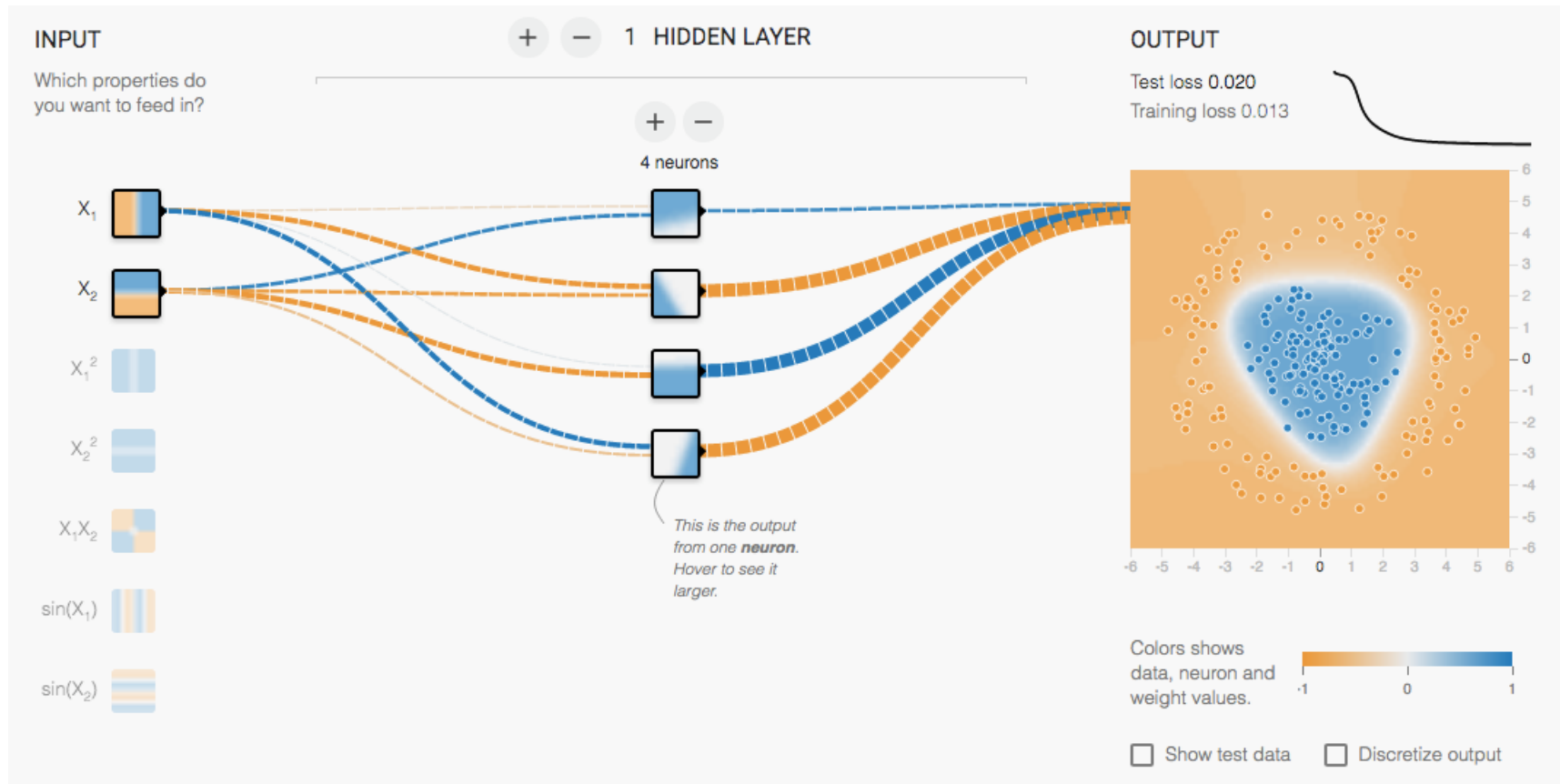
Model Capacity

- Capacity of a feed-forward neural network is affected by both:
 - **Depth**: number of hidden layers
 - **Width**: number of neurons in each hidden layer
- **More neurons = more capacity**

Model Capacity



Multi-Layer network demo



Loss Functions

- Same as single-layer models (i.e., linear and logistic regression)

- Regression:

- MSE loss: $\mathcal{L}_{\text{MSE}}(y, y^*) = (y - y^*)^2$

- Classification:

- Binary cross entropy for binary classification:

$$\mathcal{L}(y, y^*) = -y^* \log y - (1 - y^*) \log (1 - y)$$

- Cross entropy for multi-class classification:

$$\mathcal{L}(y, y^*) = - \sum_{i=1}^C y_i^* \log y_i$$

Optimization

Solve for $\theta^* = \operatorname{argmin}_{\theta} L(\hat{y}, y)$

Q: Don't I have to optimize differently for different $L(\cdot)$?

A: No, just use gradient descent. It is the most general optimization approach we know.

Q: But what if $L(\cdot)$ is non-convex in W ?

A: It almost surely is. Do gradient descent anyway. Just make sure everything is differentiable.

Computing Gradients

- You could write down the full function and calculate the gradients for all the weights manually.
 - It takes a lot of time and paper
 - Change loss function (e.g., add L2/L1) → Need to compute from scratch again
- Better Idea:
 - Use computational graph and chain rule of gradient calculation

Backpropagation

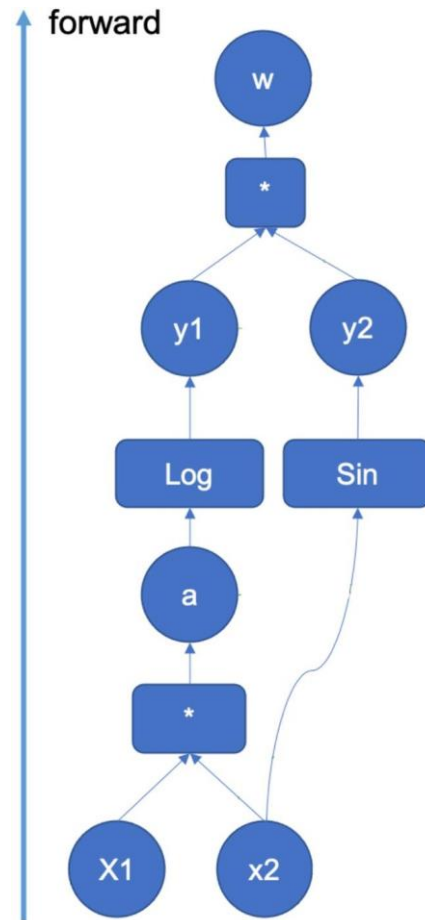
- It's taking **derivatives and applying chain rule!**
- We will **re-use derivatives computed for higher layers** in computing derivatives for lower layers so as to minimize computation
- Good news is that **modern automatic differentiation tools did all for you!**
 - Implementing backprop by hand is like programming in assembly languages.

$$\frac{\partial F(G(x))}{\partial x} = \frac{\partial F(G(x))}{\partial G} * \frac{\partial G(x)}{\partial x}$$



Computational Graph

- Break down function computation:
 - Data (input, output, and intermediate)
 - Operators (e.g., addition, multiplication)
- Consider the following function:
$$w = \log(x_1 x_2) \sin(x_2)$$



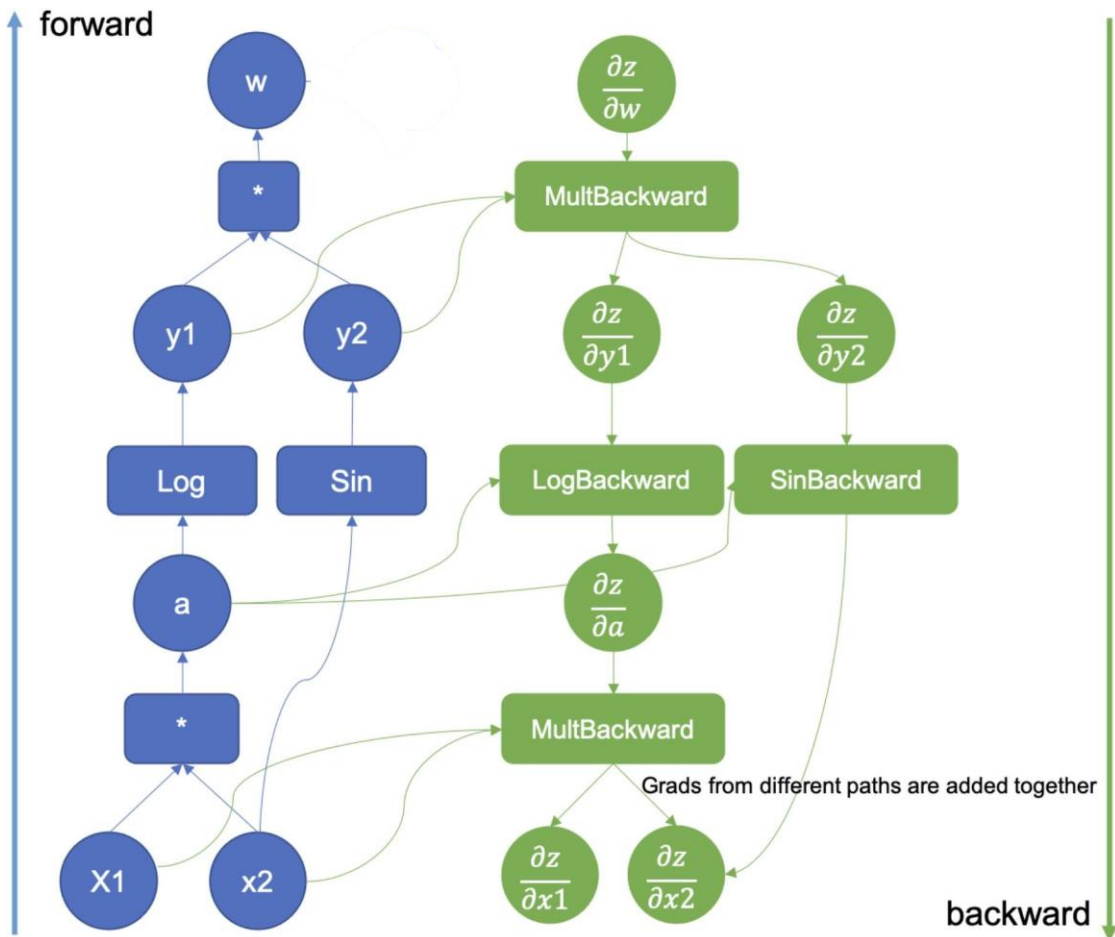
Full Graph

Requirement for each operator:

Forward: compute their output function given input.

Backward: compute the gradient of their output w.r.t. each input.

If all operators can do forward & backward computation, we can compute the derivative of the output with respect to any input, procedurally.



Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = \quad \frac{\partial f}{\partial y} =$$

Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$

Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$

$$f(x, y) = xy \quad \rightarrow \quad \frac{\partial f}{\partial x} = \quad \frac{\partial f}{\partial y} =$$

Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$

$$f(x, y) = xy \quad \rightarrow \quad \frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x$$

Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$

$$f(x, y) = xy \quad \rightarrow \quad \frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x$$

$$f(x, y) = \max(x, y) \quad \rightarrow \quad \frac{\partial f}{\partial x} = \quad \frac{\partial f}{\partial y} =$$

Interpretation of the Gradient

Interpretation. Derivatives indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$

$$f(x, y) = xy \quad \rightarrow \quad \frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x$$

$$f(x, y) = \max(x, y) \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1(x \geq y) \quad \frac{\partial f}{\partial y} = 1(y \geq x)$$

Compound Expressions with Chain Rule

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

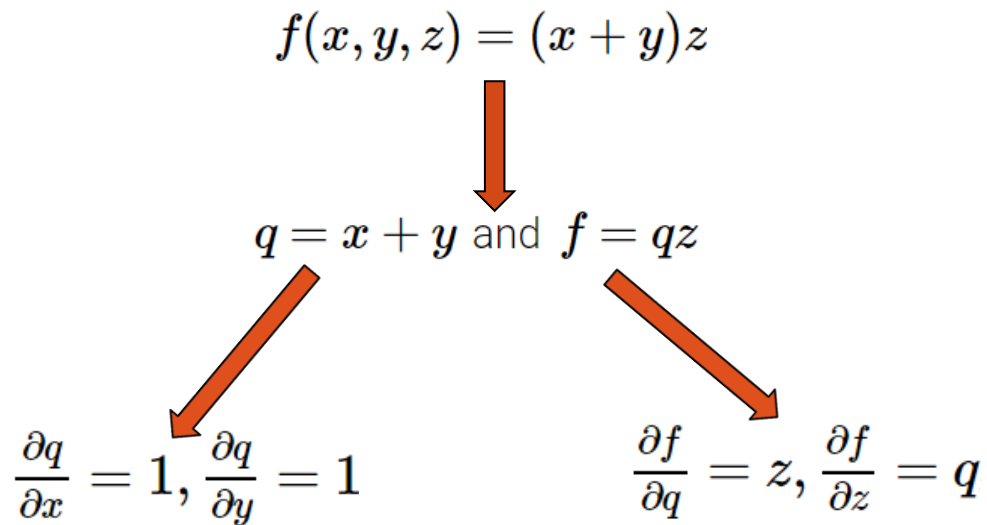
Compound Expressions with Chain Rule

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

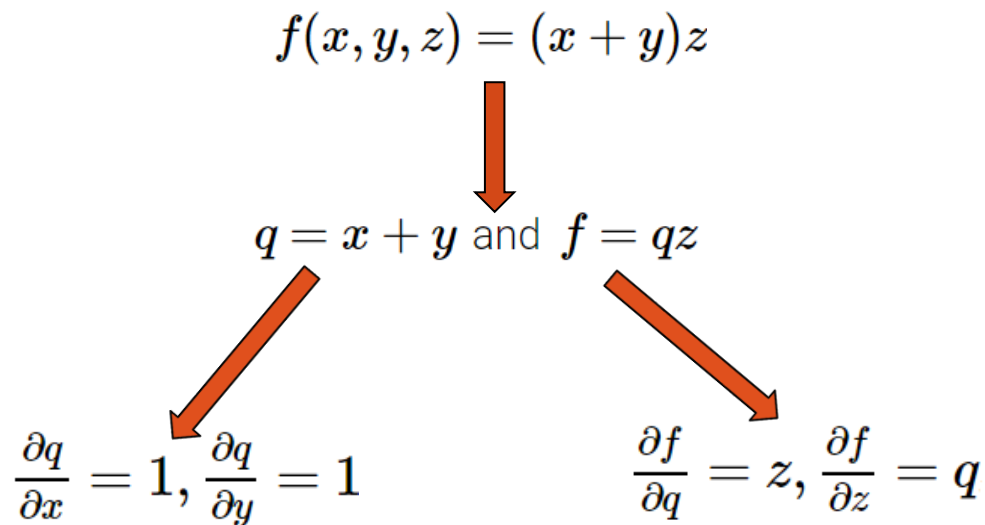


$$q = x + y \text{ and } f = qz$$

Compound Expressions with Chain Rule



Compound Expressions with Chain Rule



Chain rule: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$

Compound Expressions with Chain Rule

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

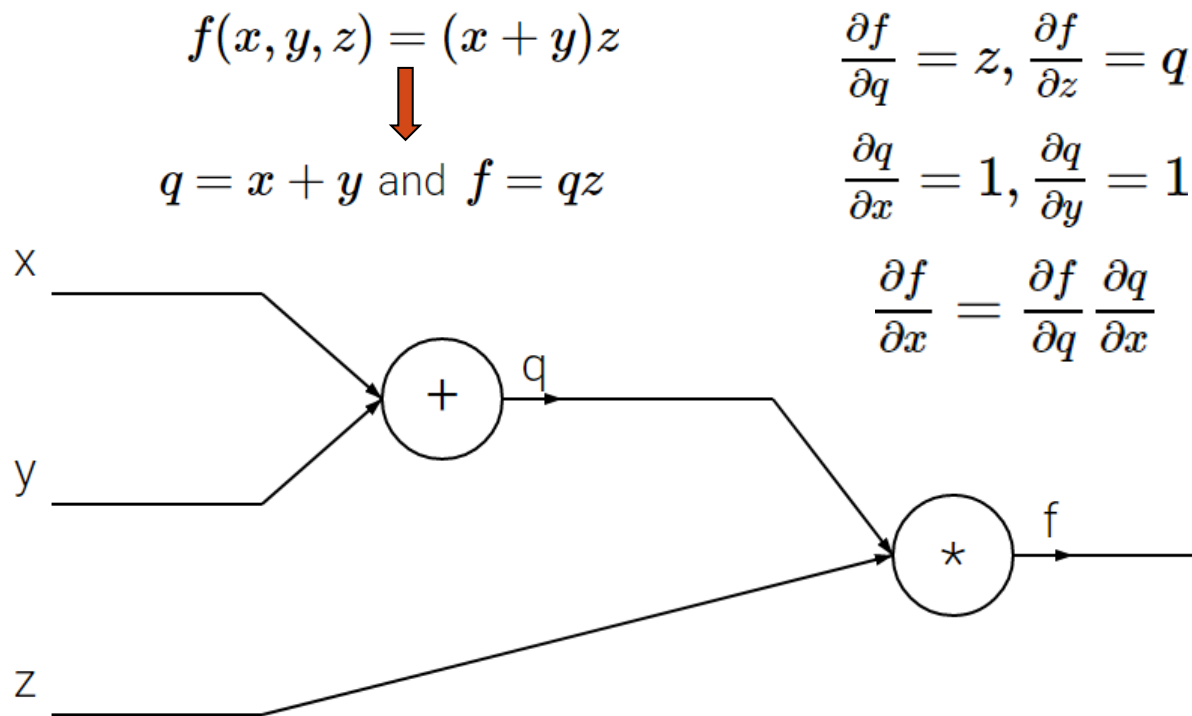
$$q = x + y \text{ and } f = qz$$

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

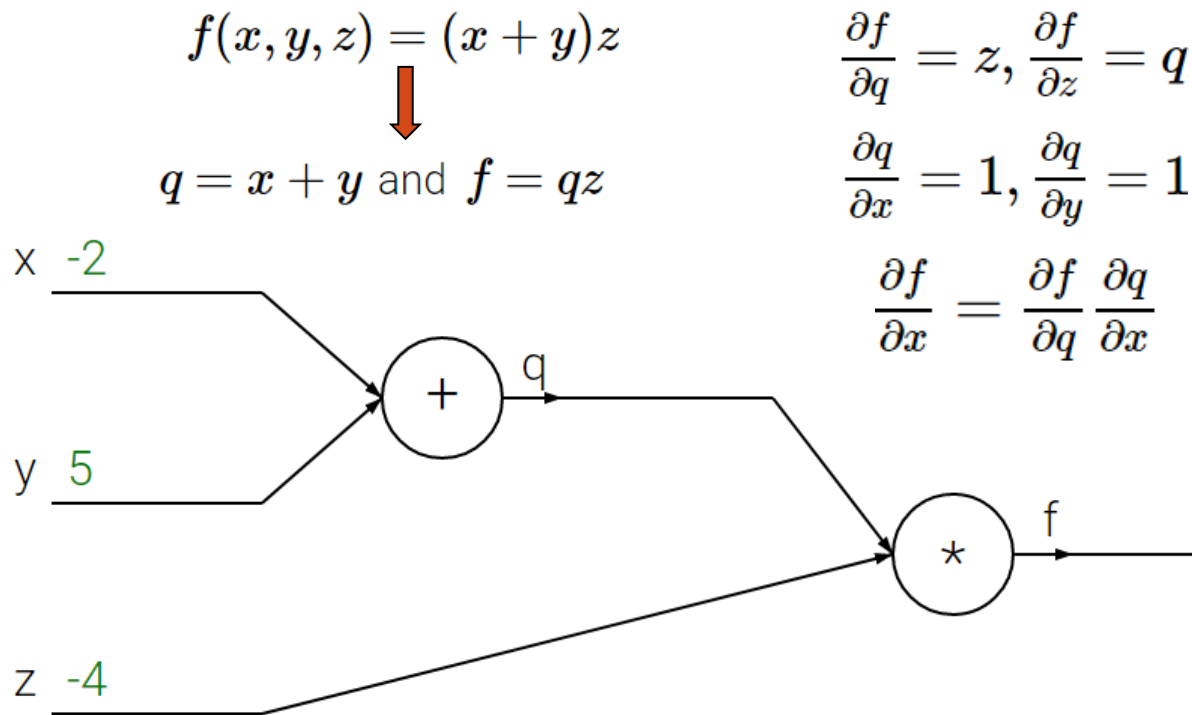
$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

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

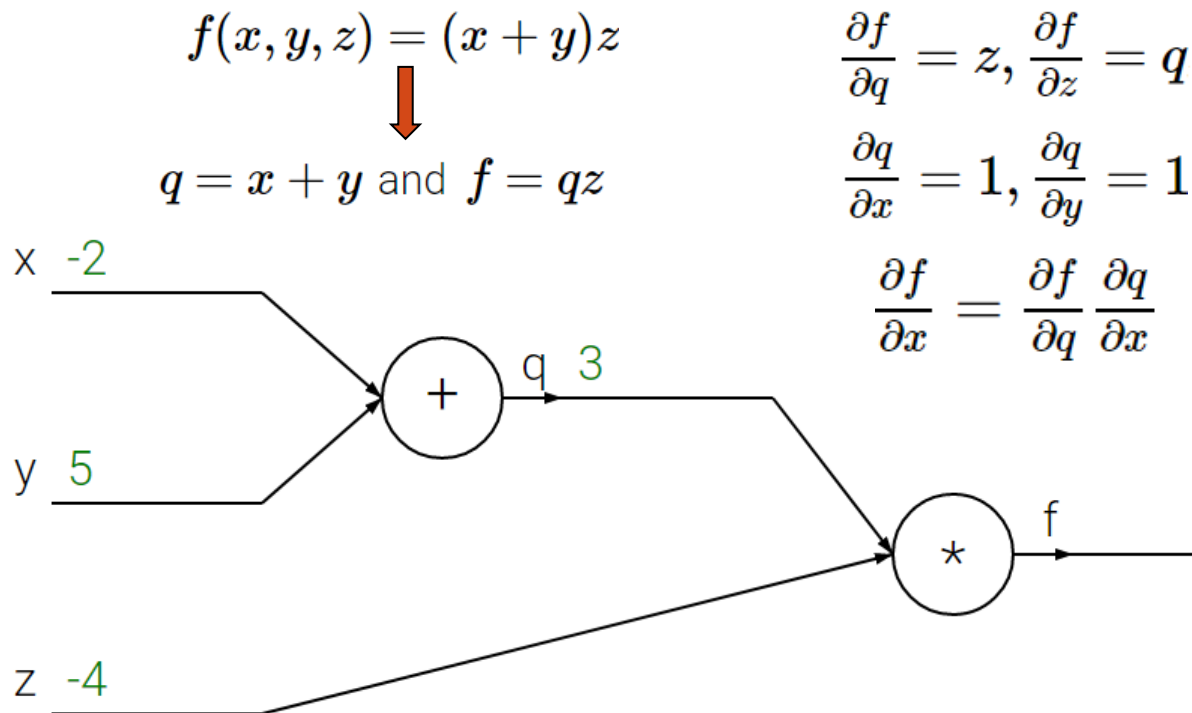
Compound Expressions with Chain Rule



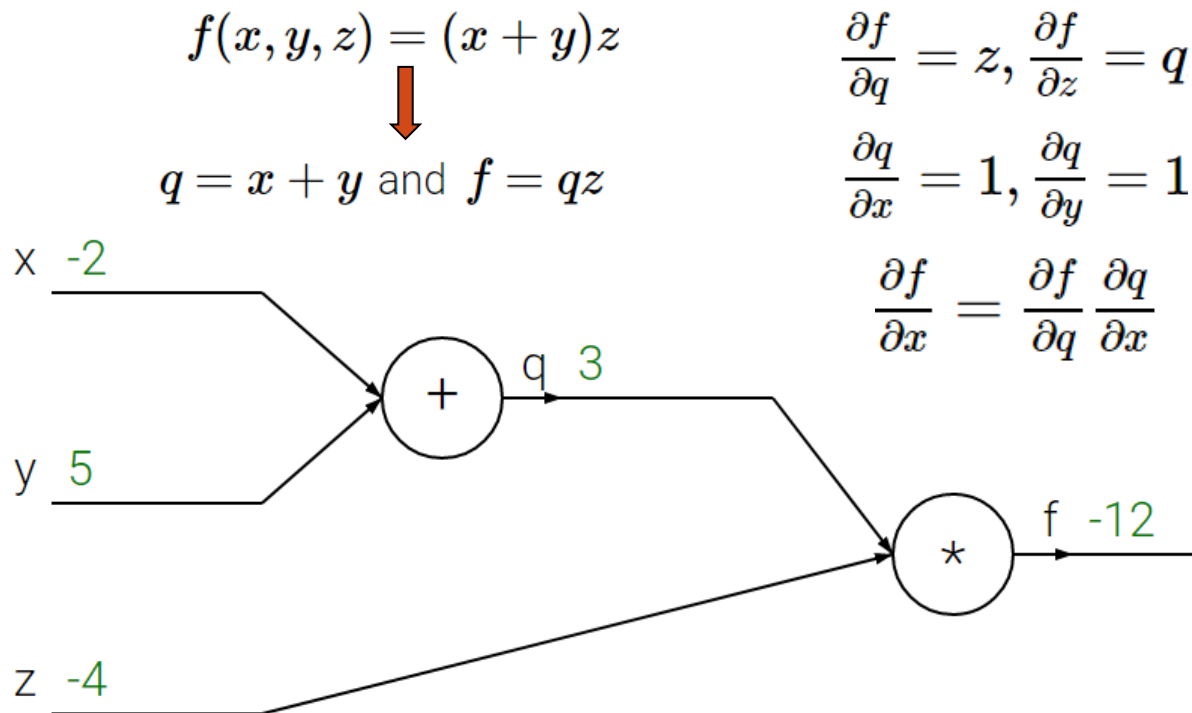
Compound Expressions with Chain Rule



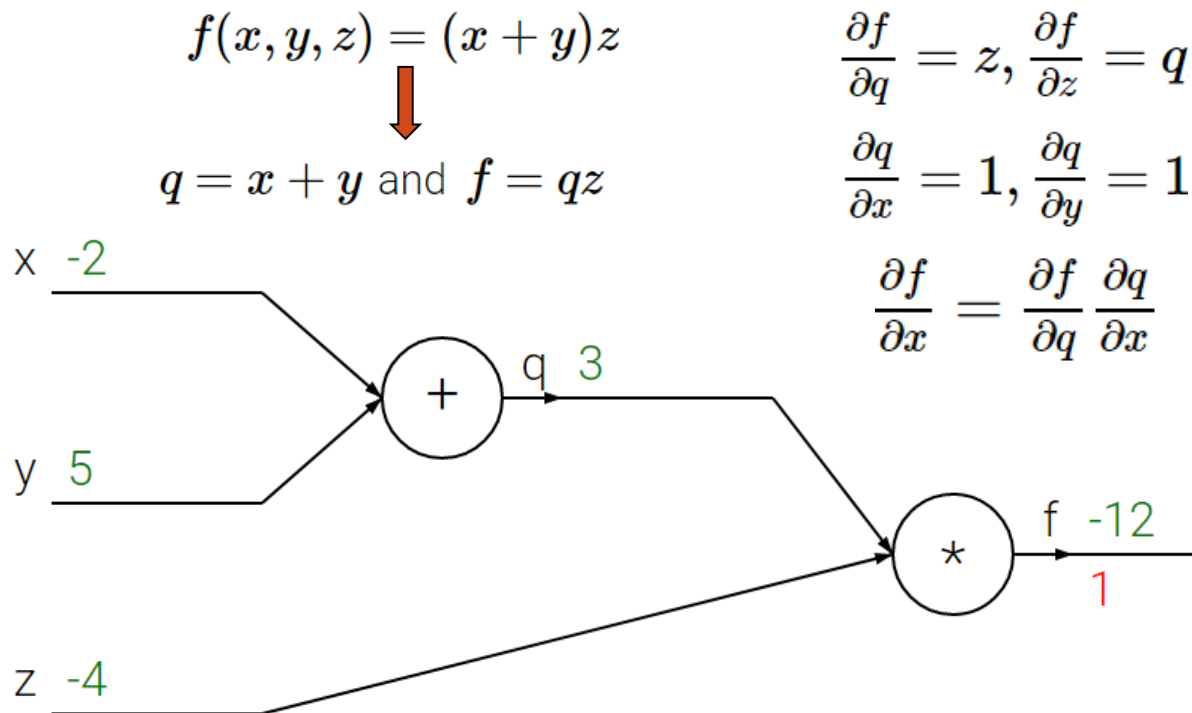
Compound Expressions with Chain Rule



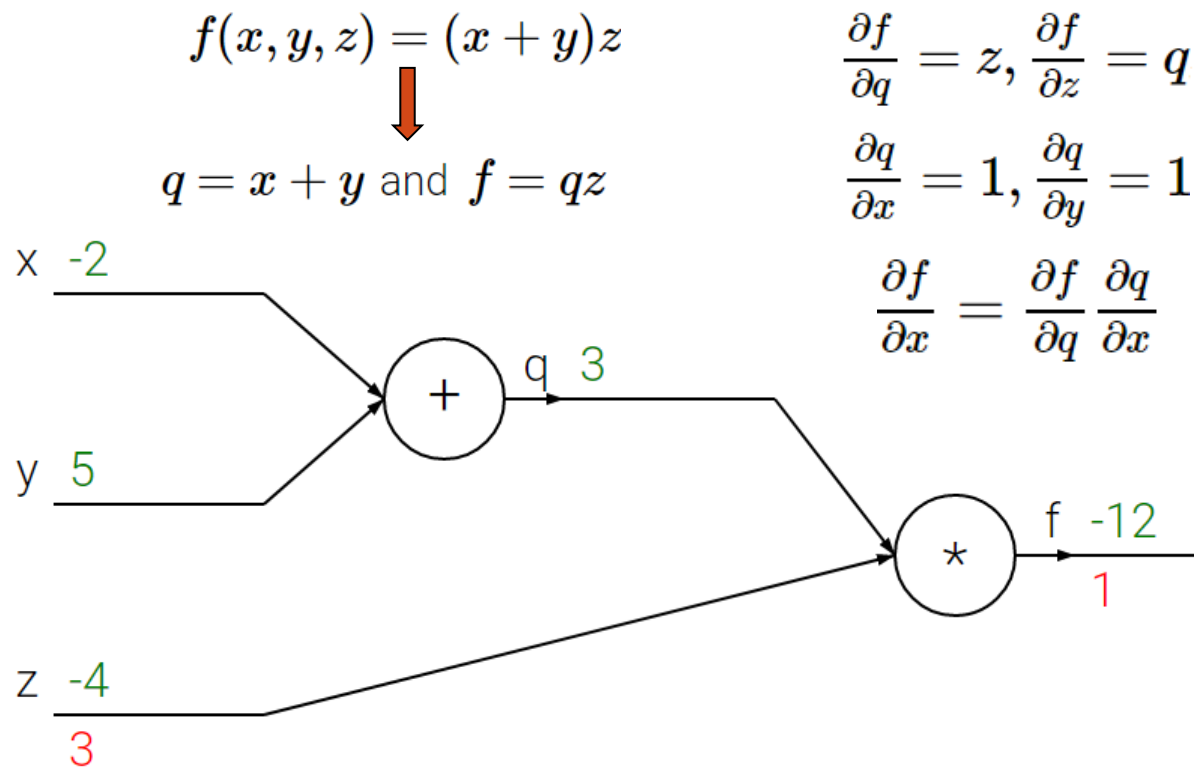
Compound Expressions with Chain Rule



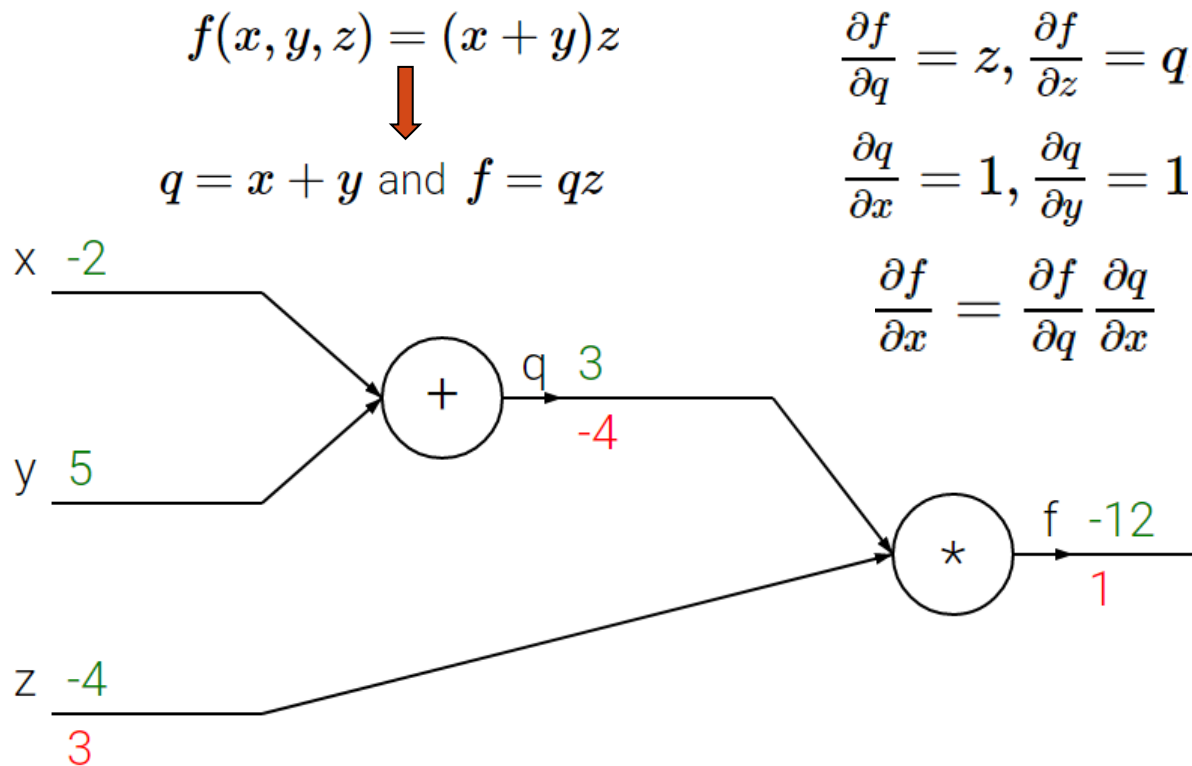
Compound Expressions with Chain Rule



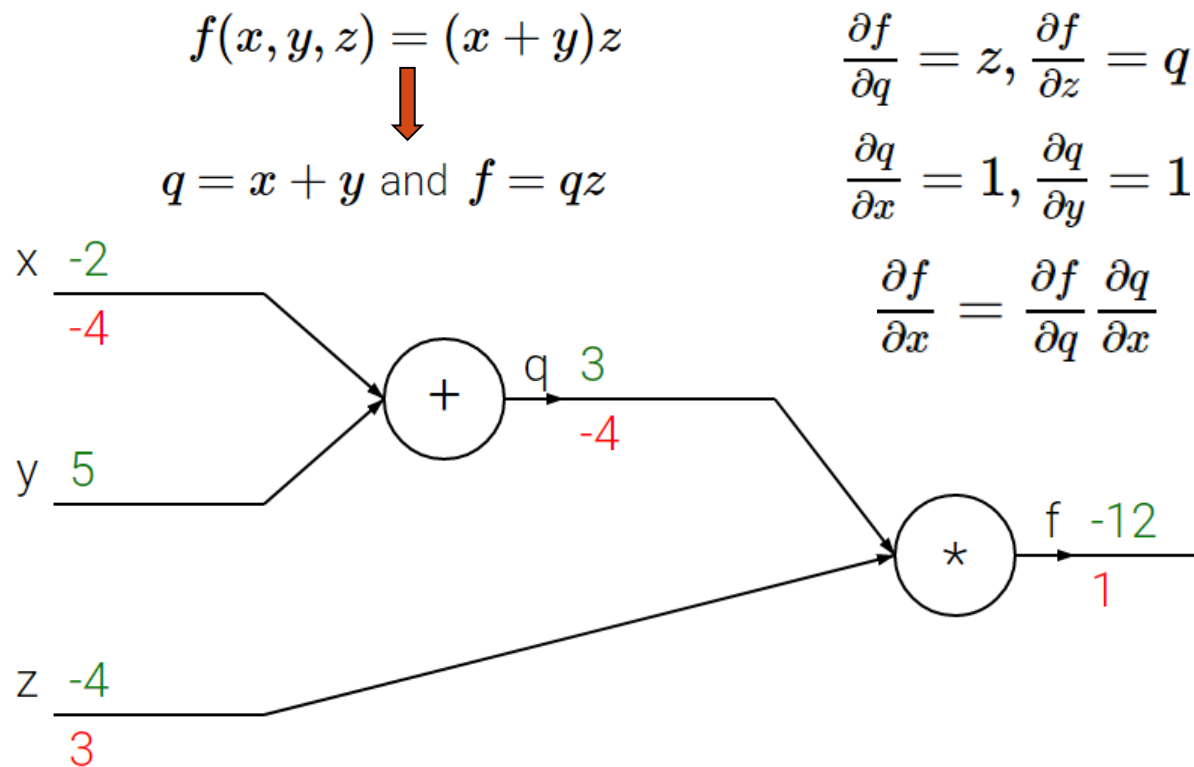
Compound Expressions with Chain Rule



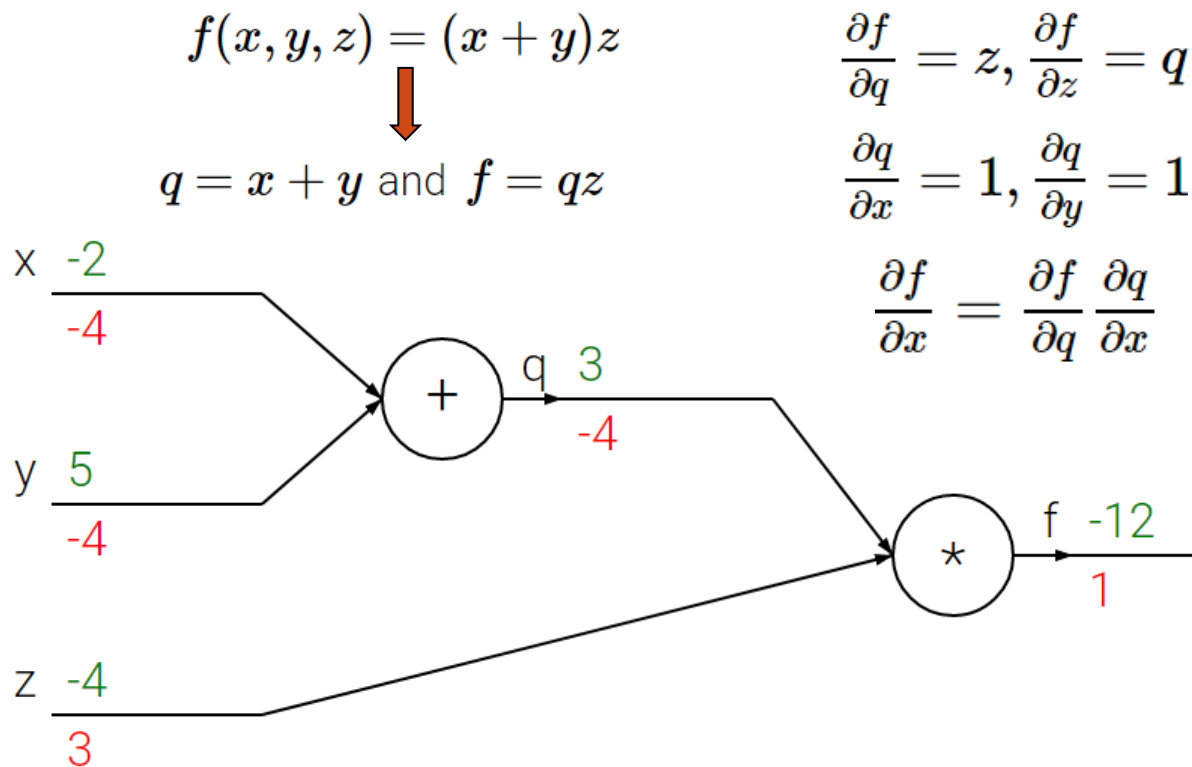
Compound Expressions with Chain Rule



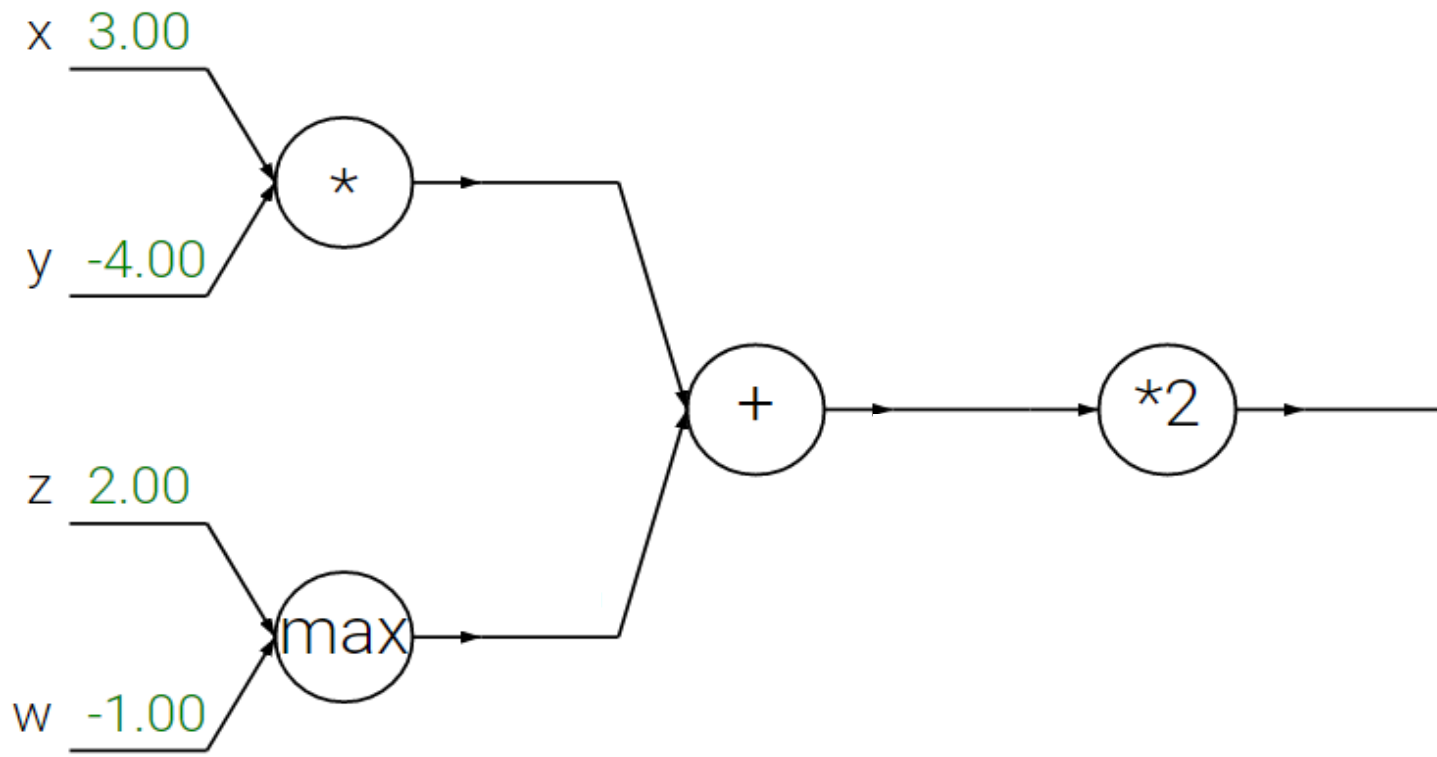
Compound Expressions with Chain Rule



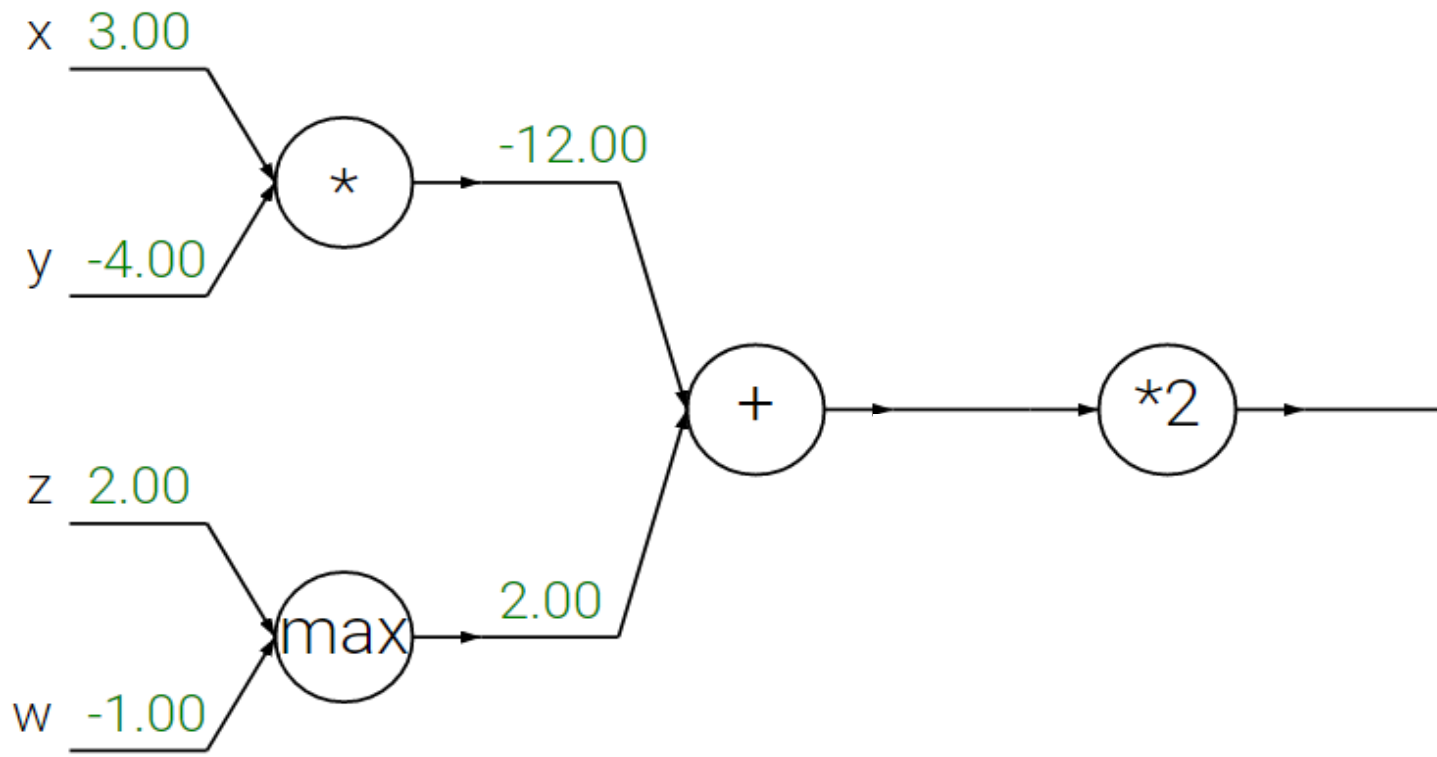
Compound Expressions with Chain Rule



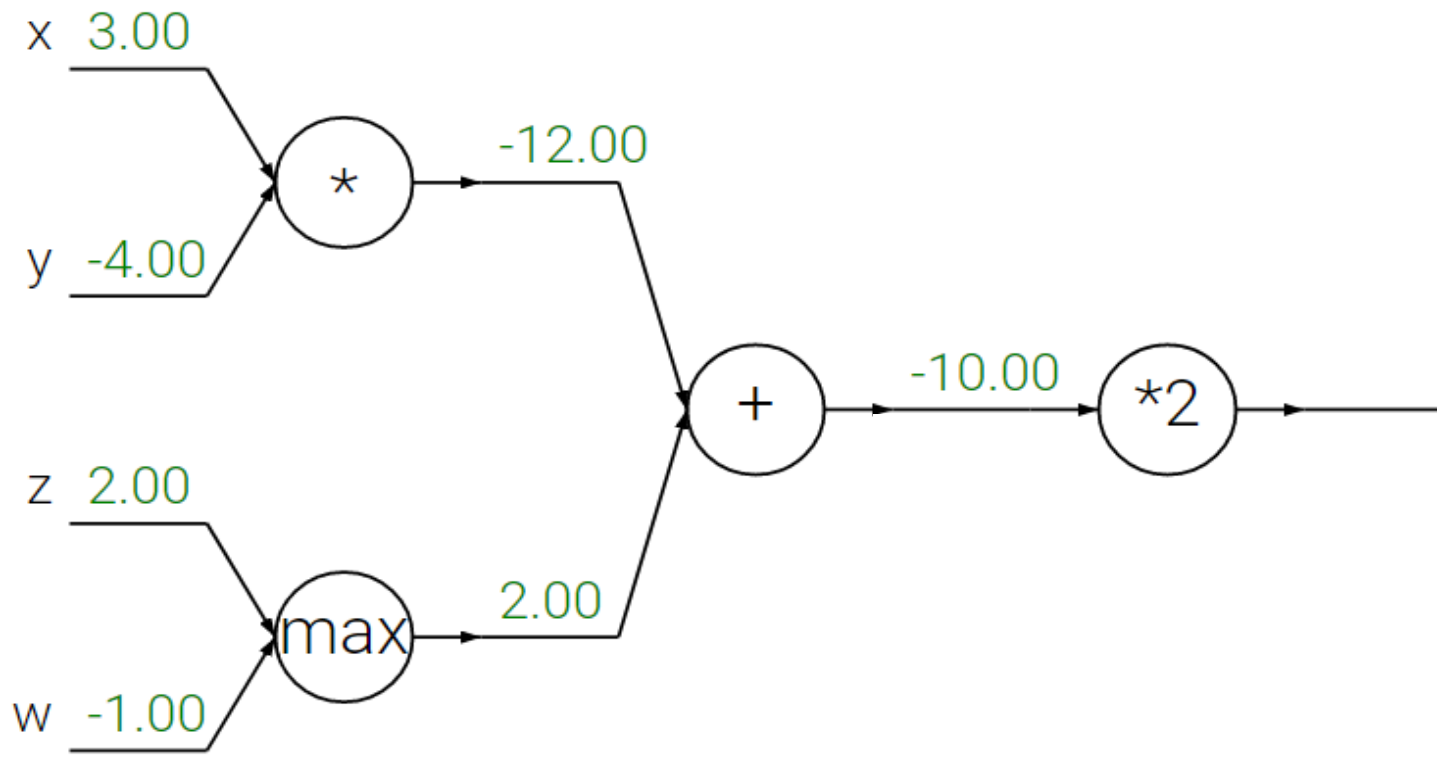
Forward and Backward Pass



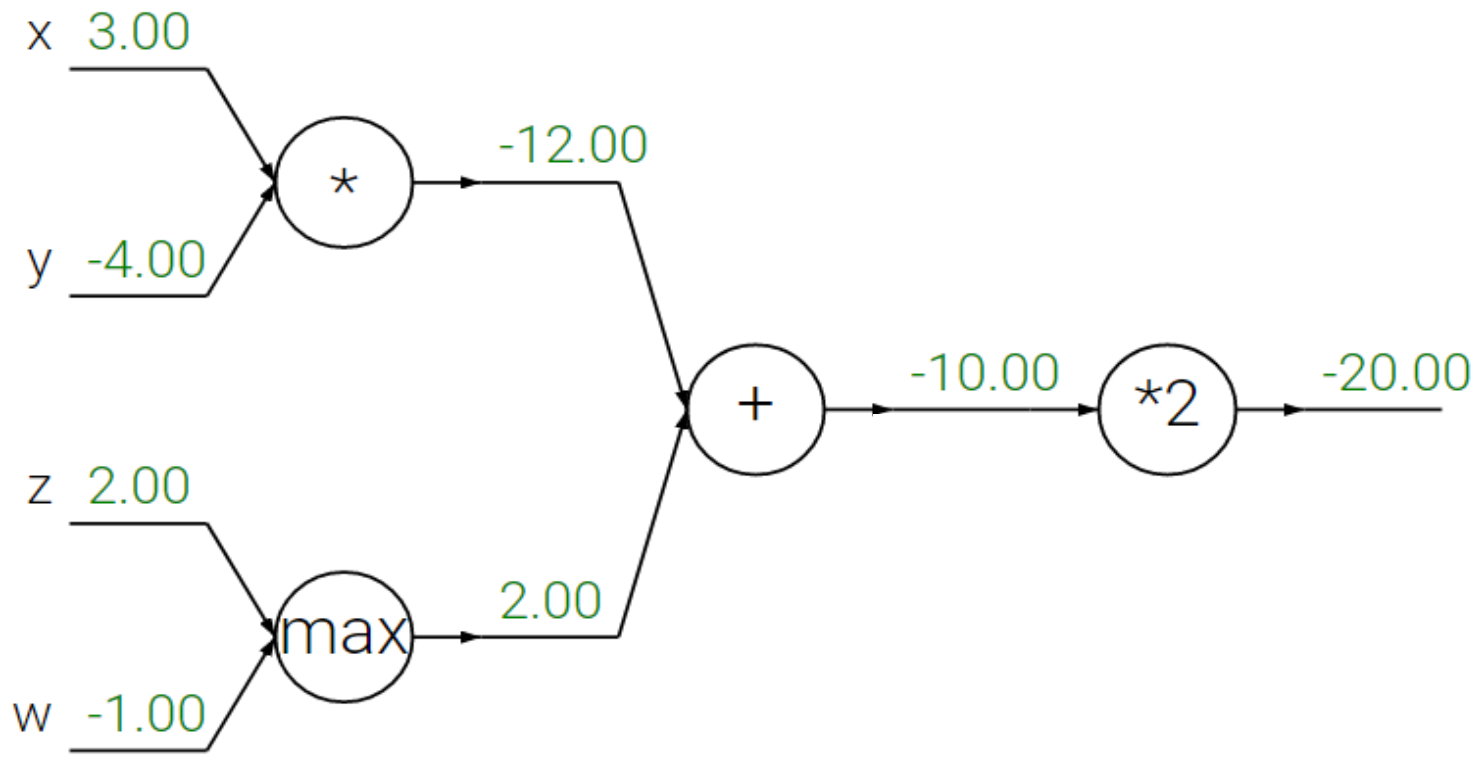
Forward and Backward Pass



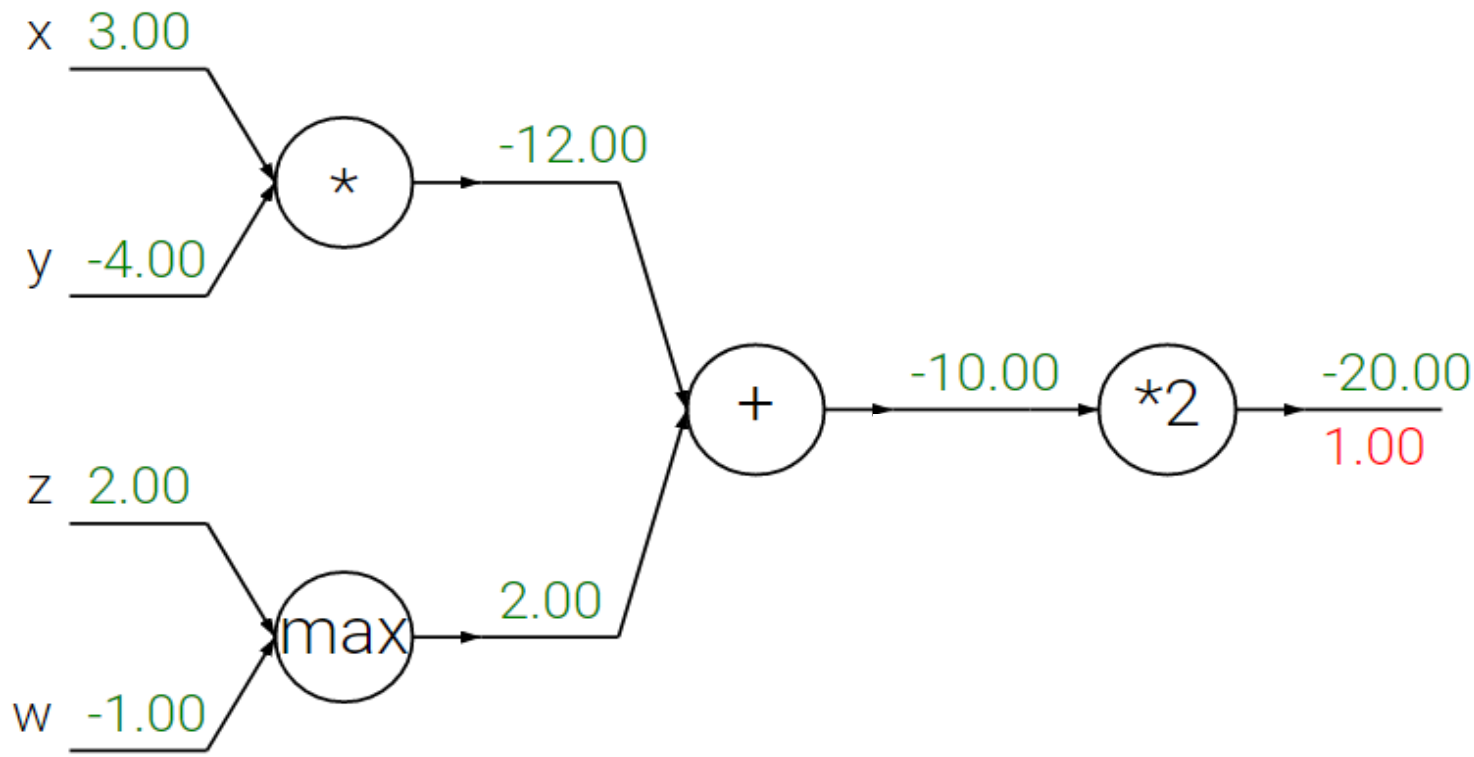
Forward and Backward Pass



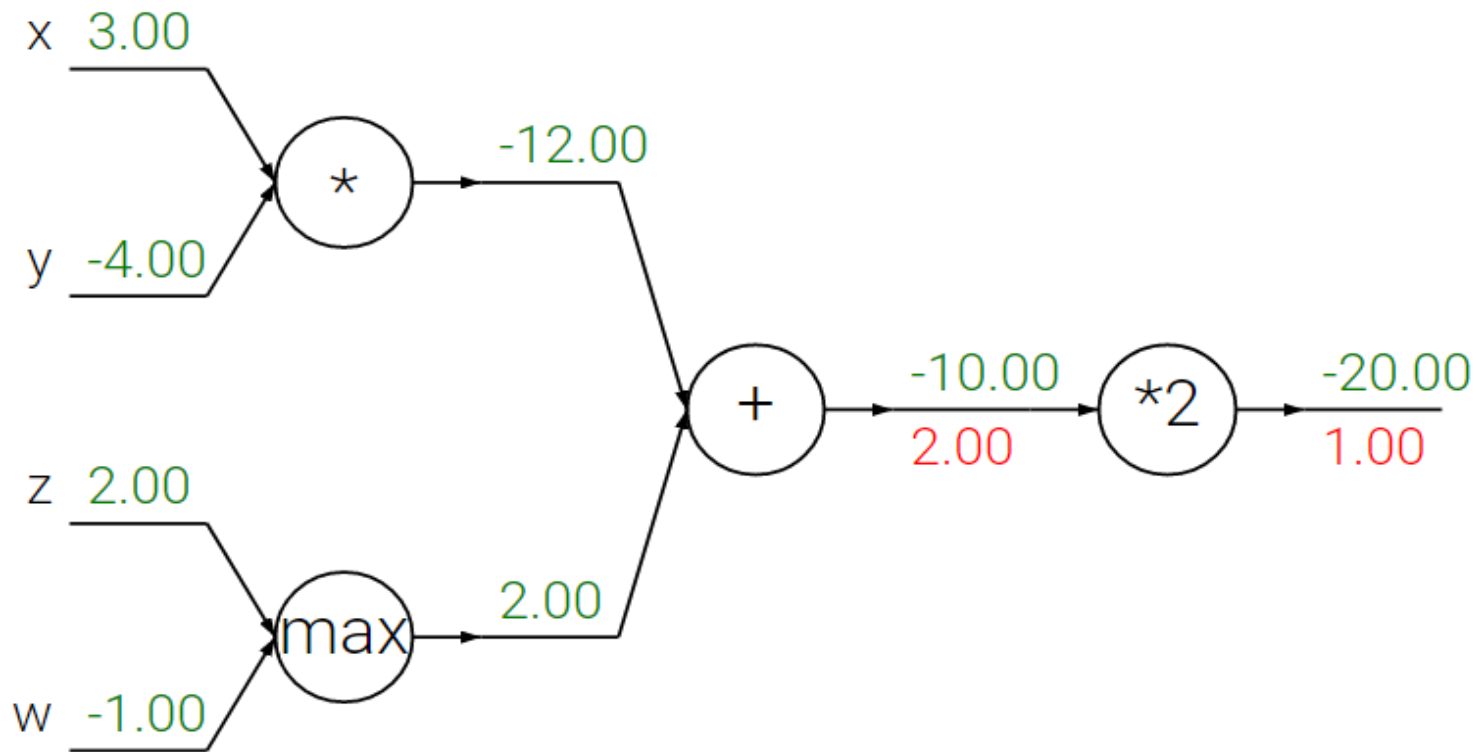
Forward and Backward Pass



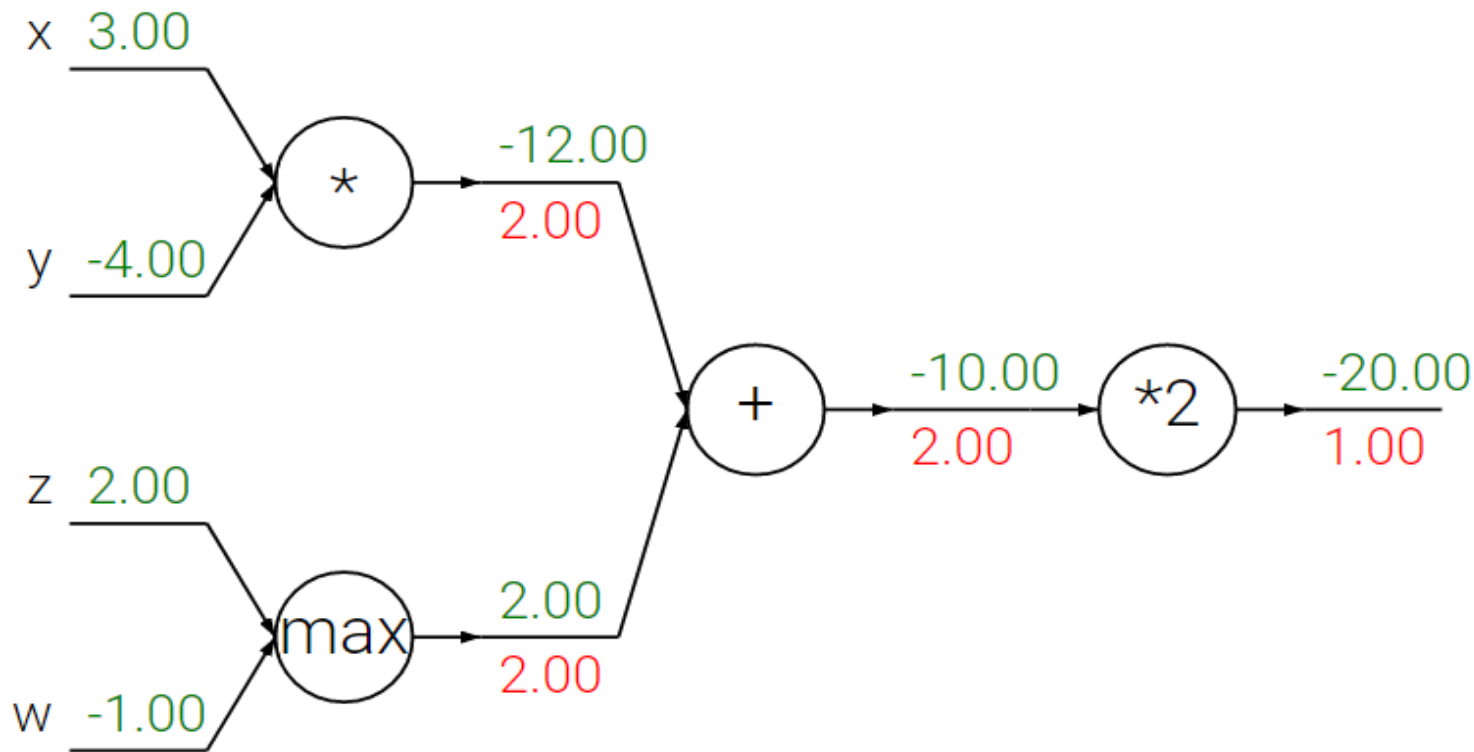
Forward and Backward Pass



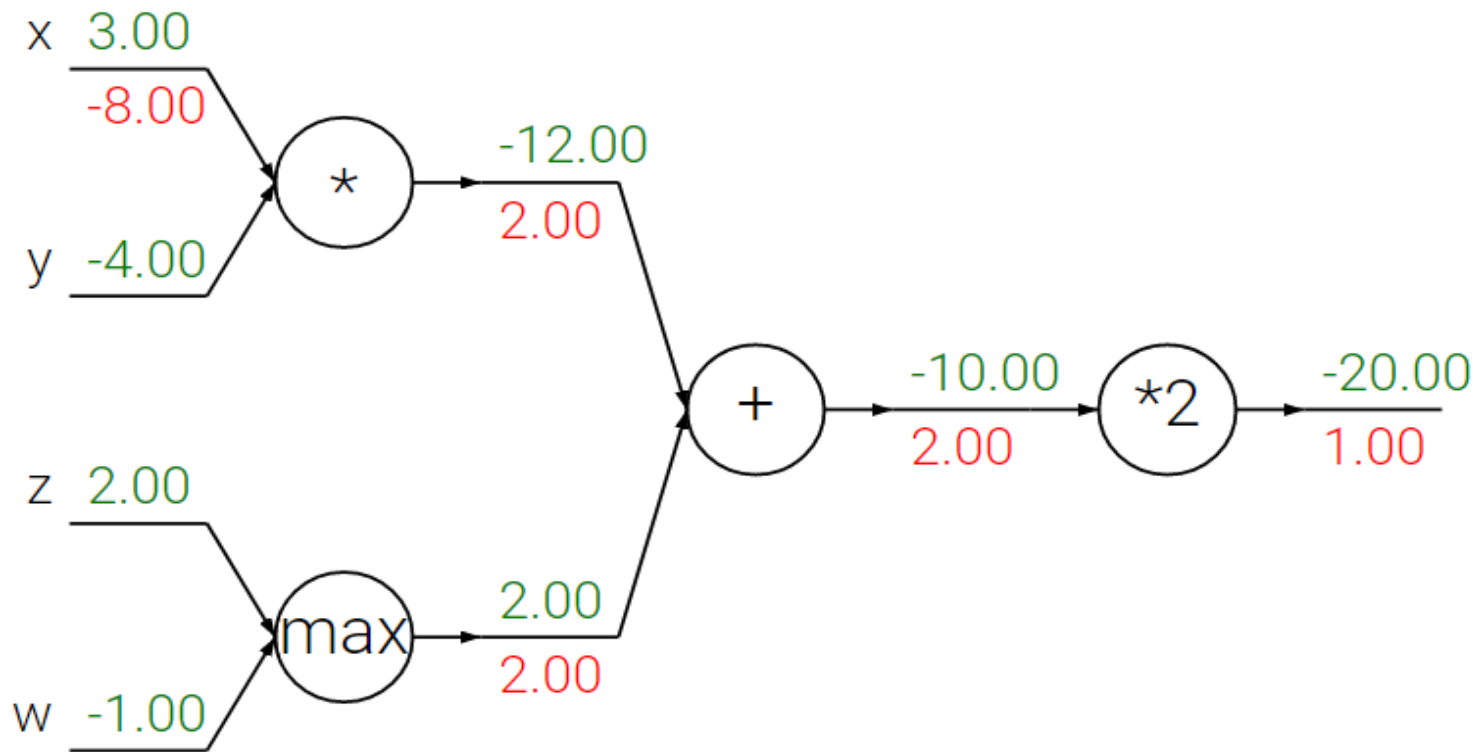
Forward and Backward Pass



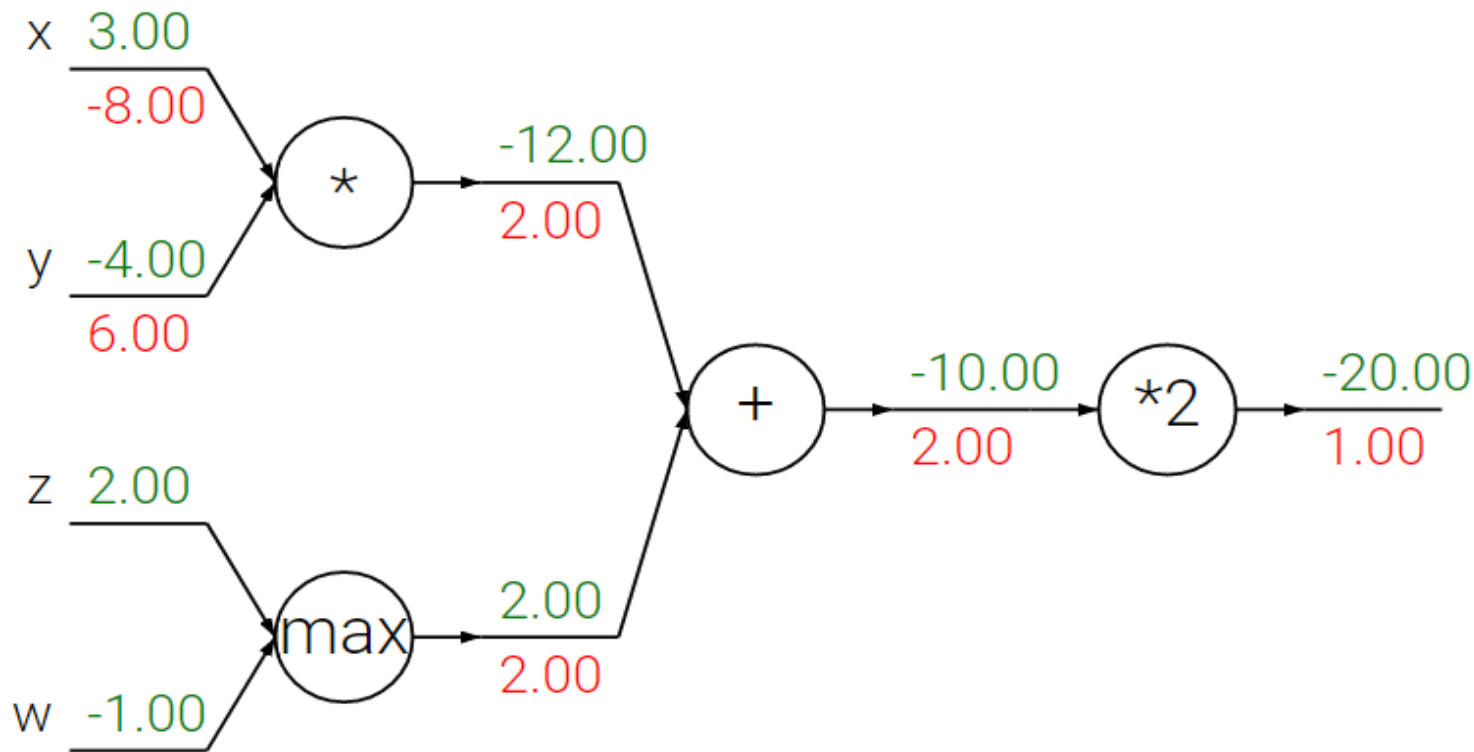
Forward and Backward Pass



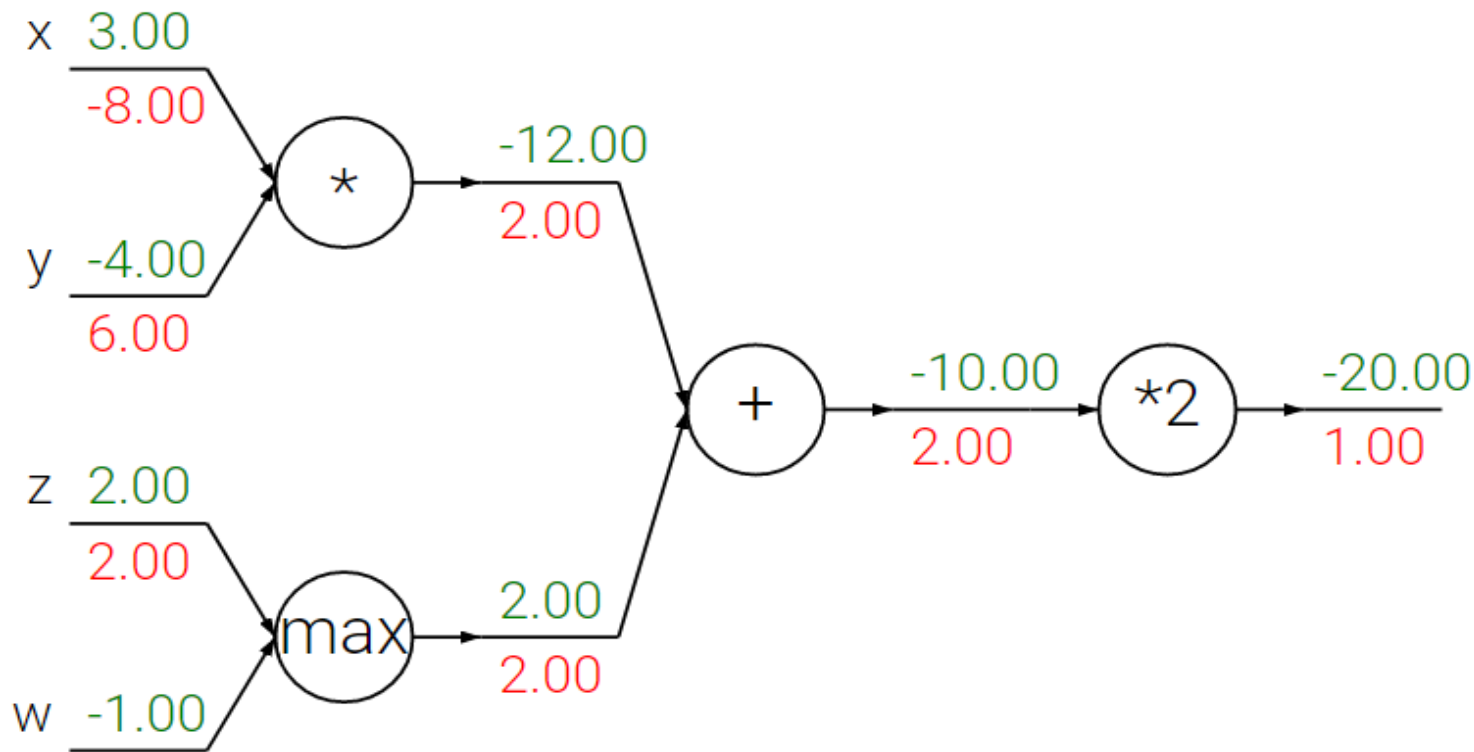
Forward and Backward Pass



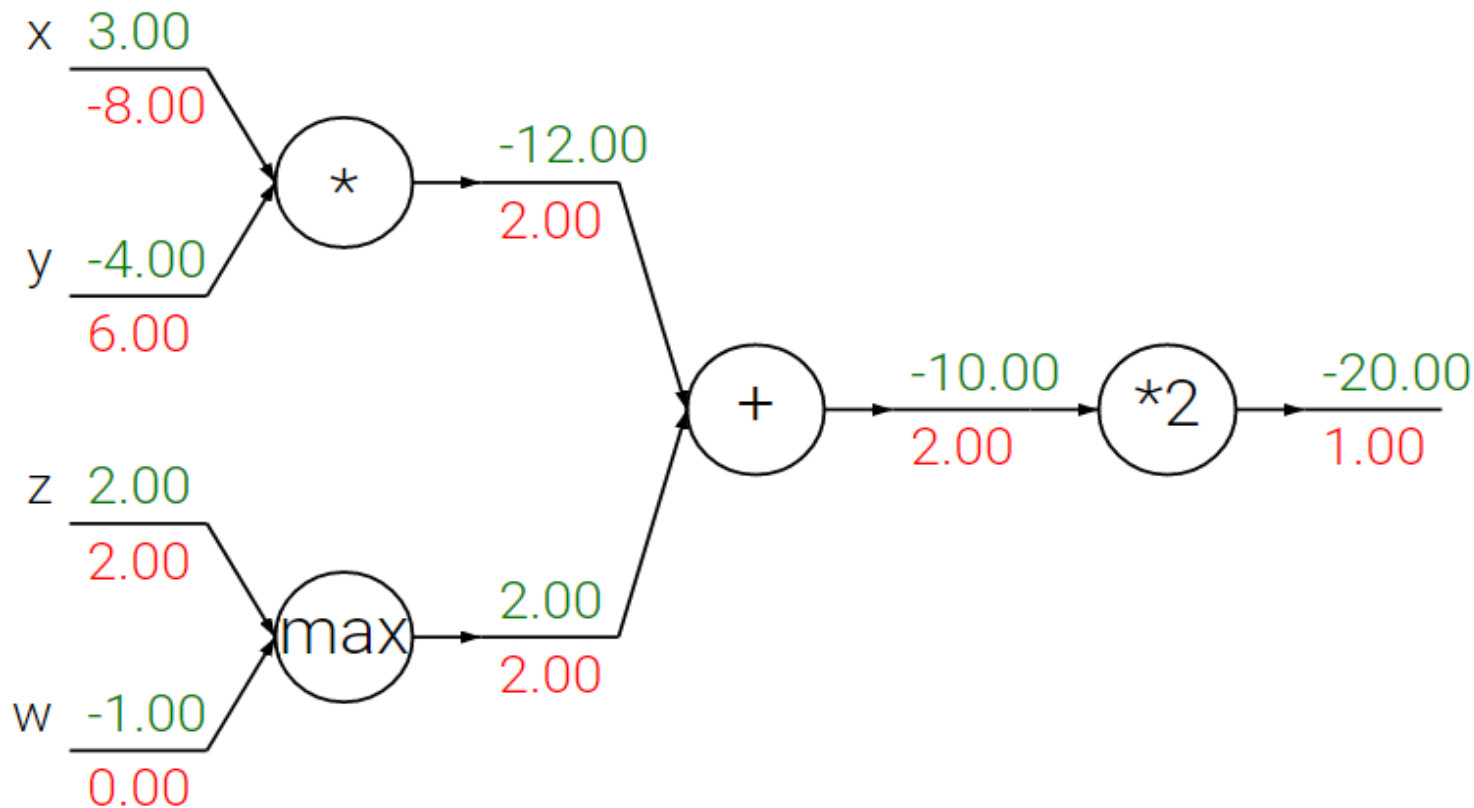
Forward and Backward Pass



Forward and Backward Pass



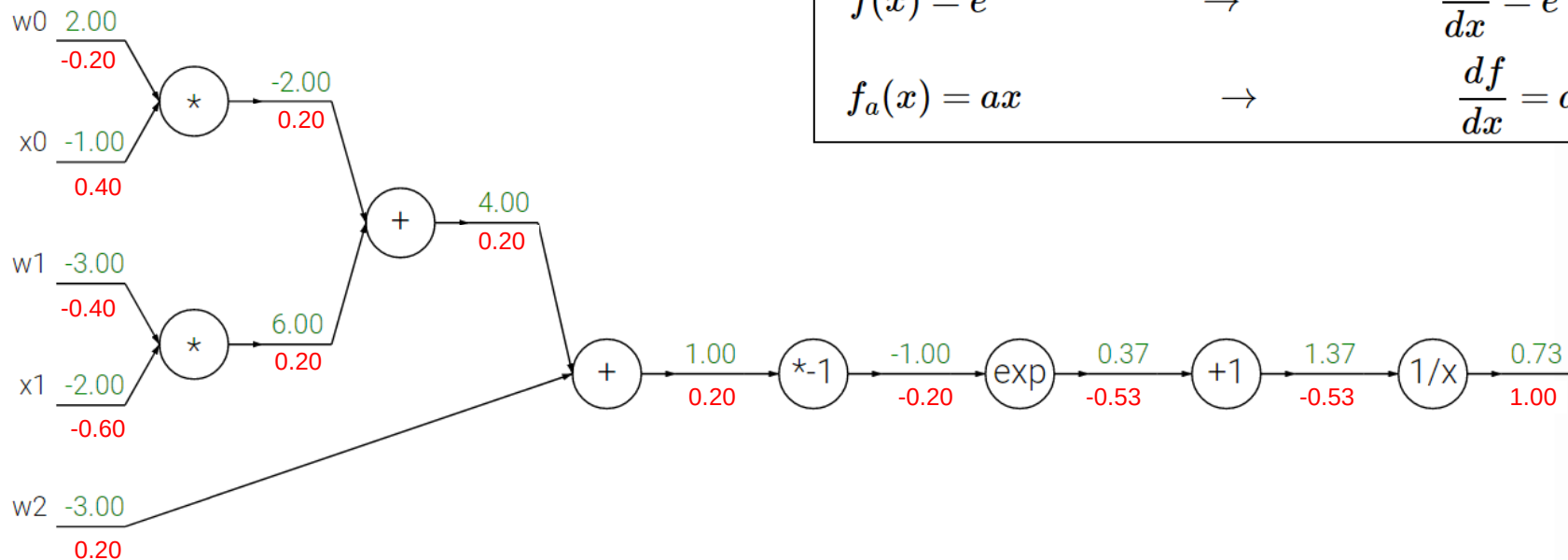
Forward and Backward Pass



Sigmoid Example

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

$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$
$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$



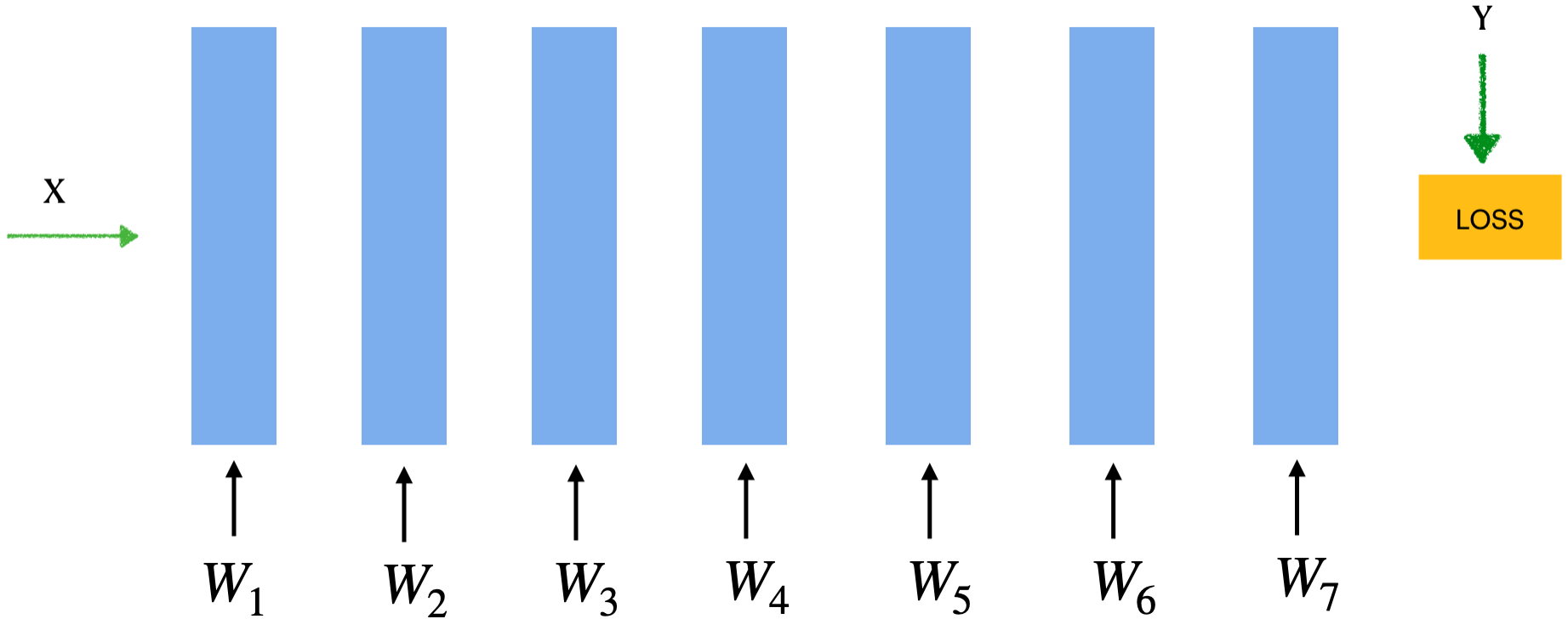
Backpropagation in General Computational Graph

- Forward propagation: visit nodes in topological sort order
 - Compute value of node given predecessors
- Backward propagation:
 - Initialize output gradient as 1
 - Visit nodes in reverse order and compute gradient wrt each node using gradient wrt successors

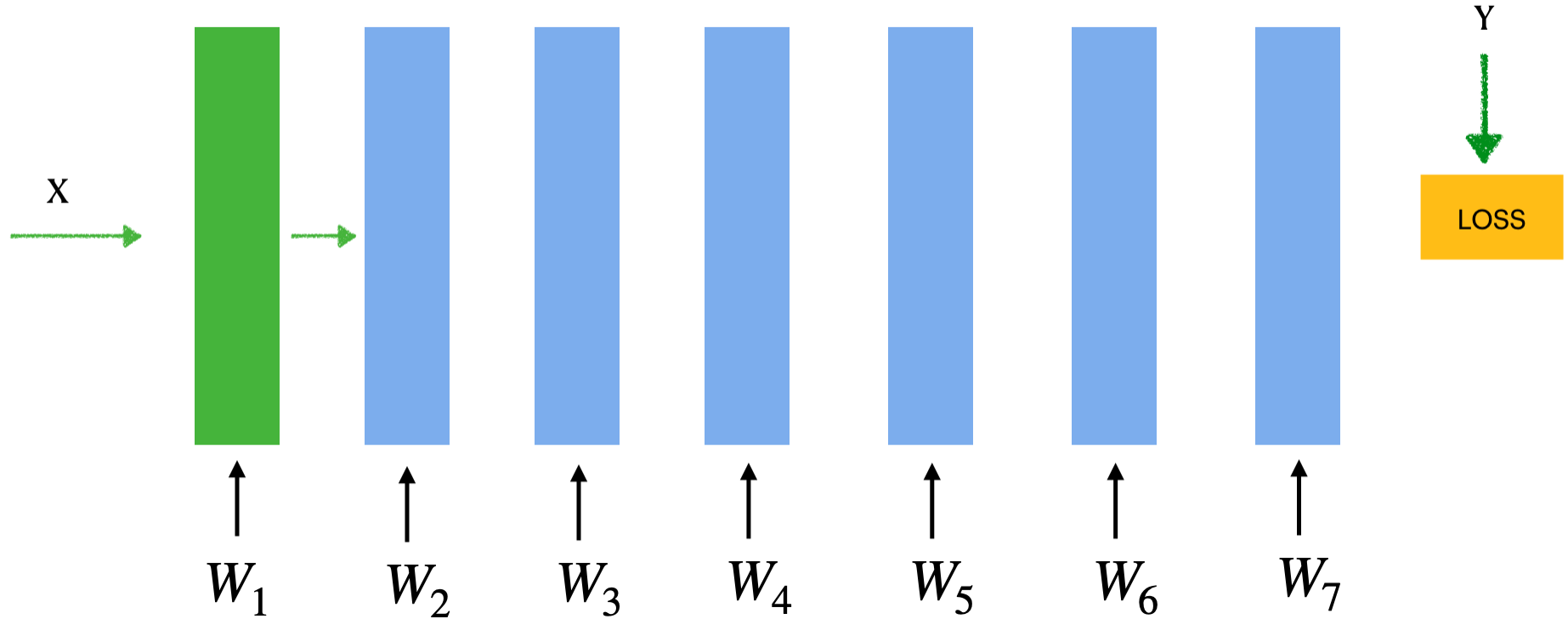
$$\frac{\partial L}{\partial x} = \sum_{i=1}^n \frac{\partial \mathcal{L}}{\partial y_i} \frac{\partial y_i}{\partial x}$$

$$\{y_1, \dots, y_n\} = \text{successors of } x$$

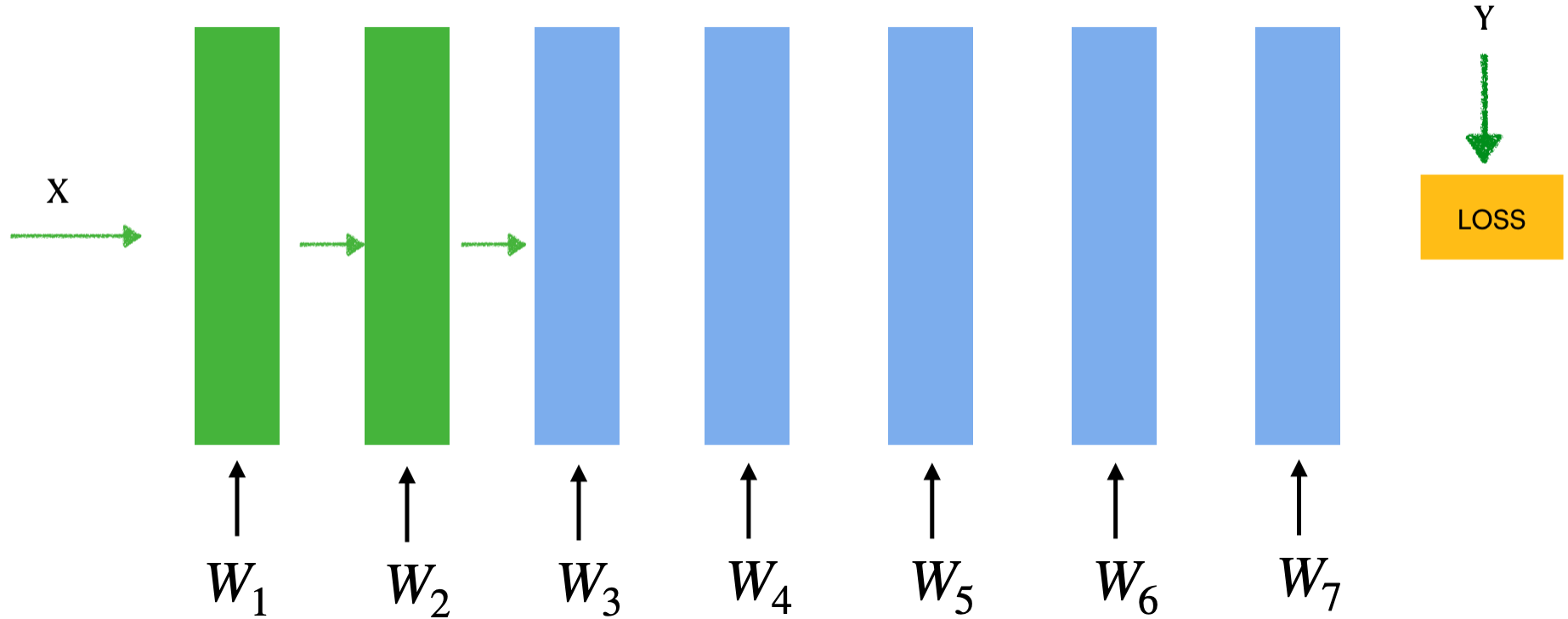
Forward Computation



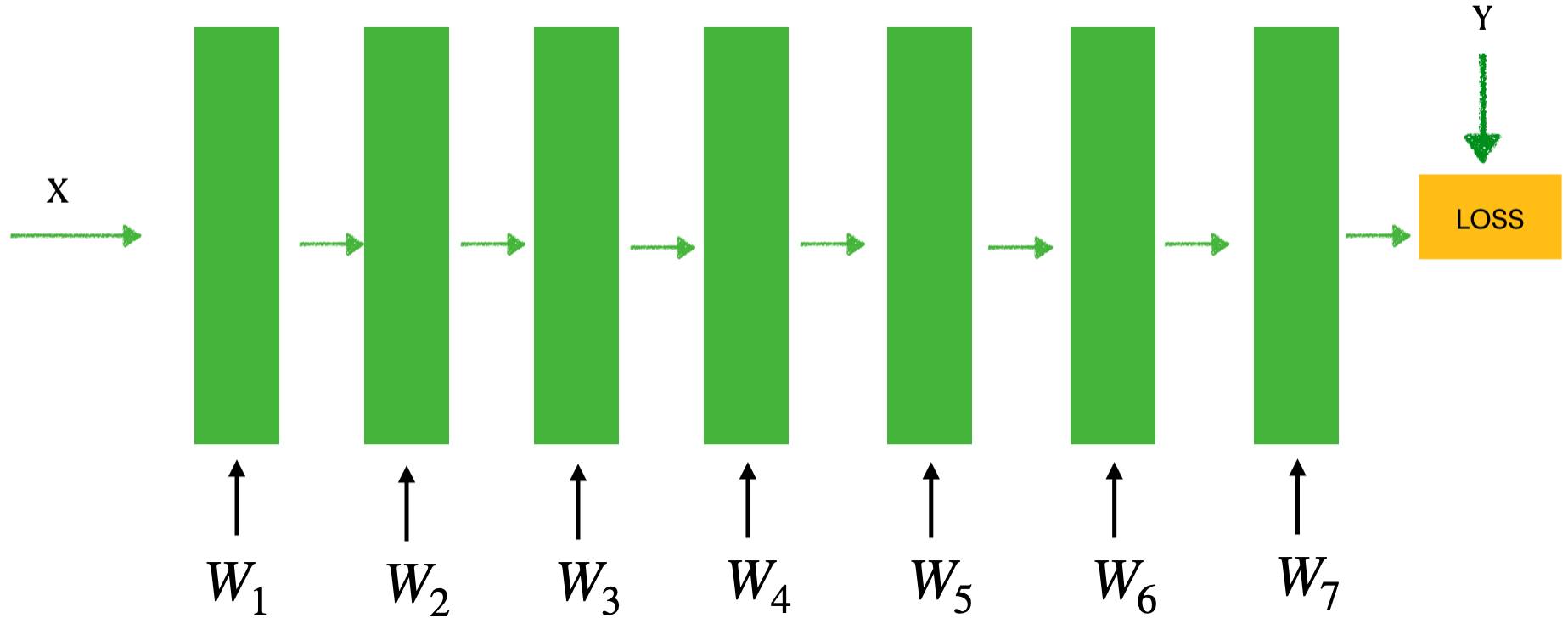
Forward Computation



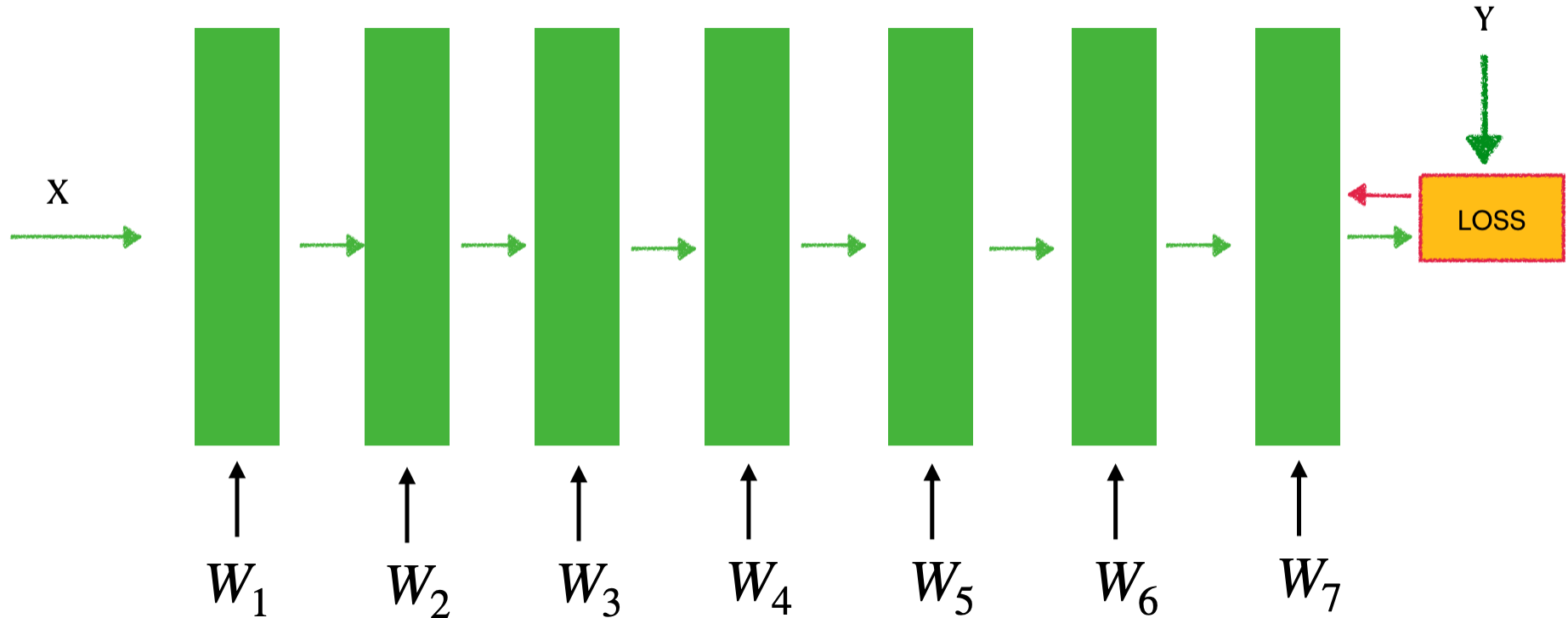
Forward Computation



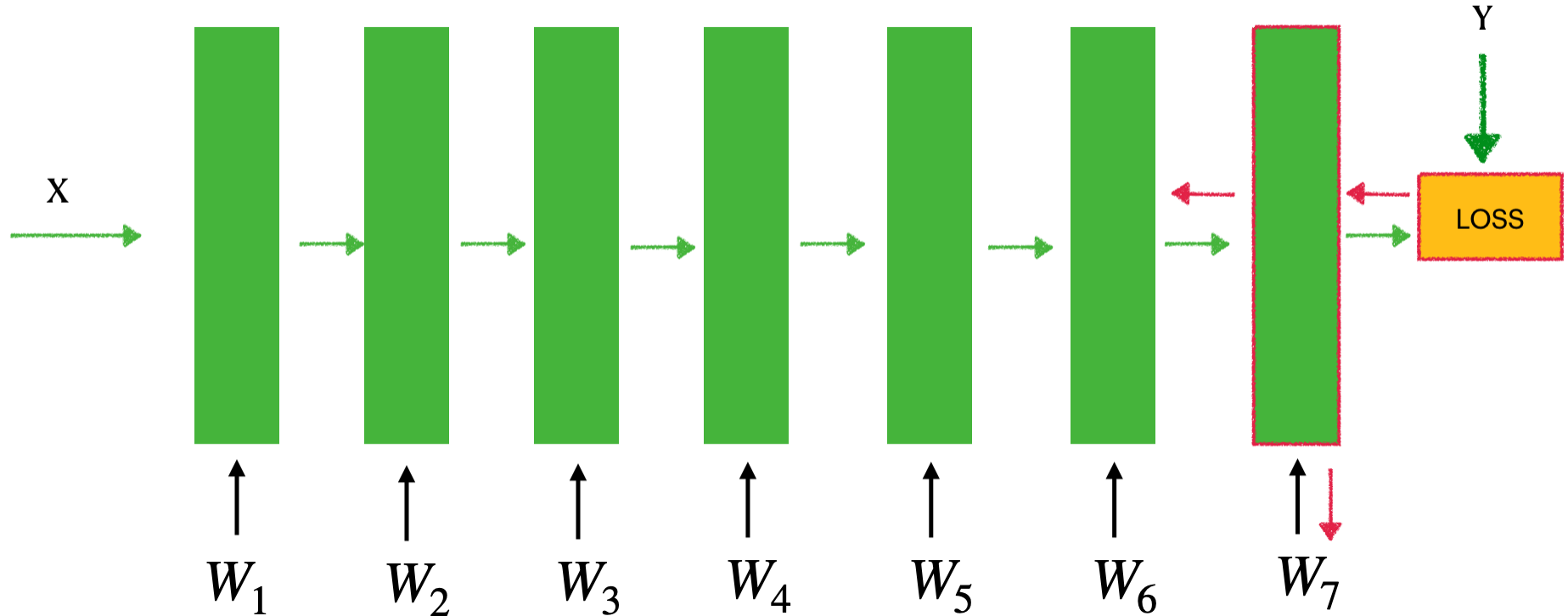
Forward Computation



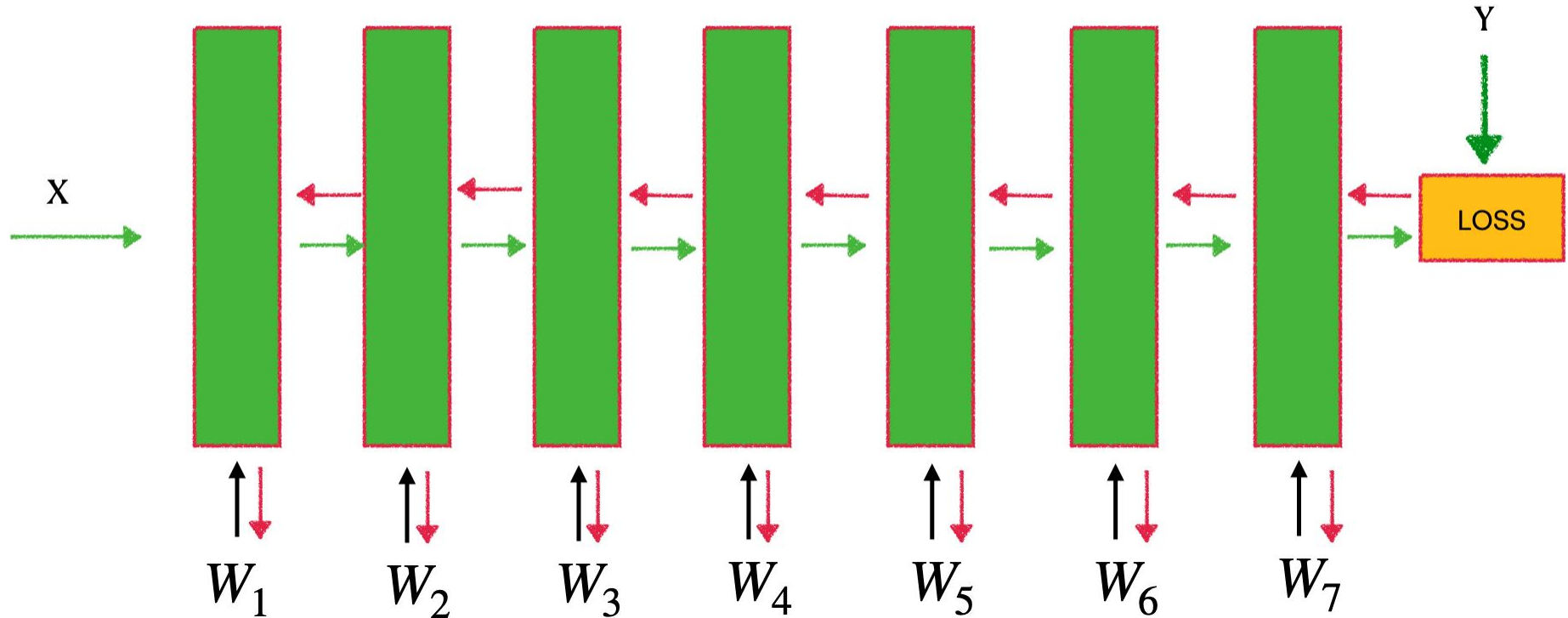
Backward Computation



Backward Computation

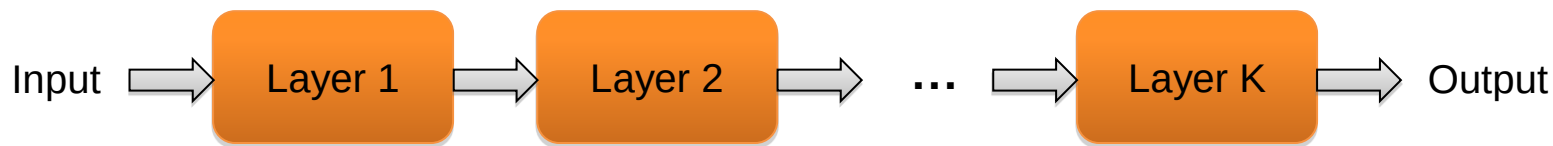


Backward Computation



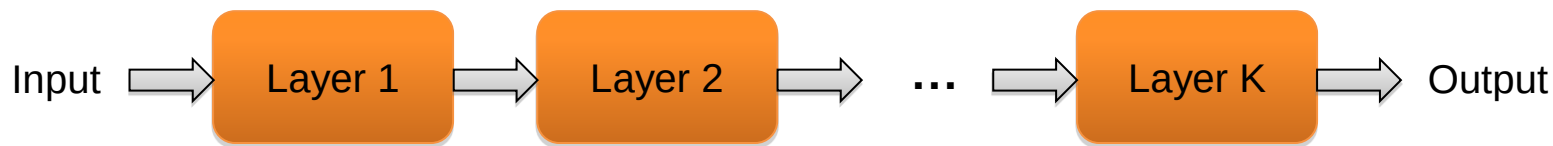
Recall: Multi-layer Neural Networks

- The function computed by the network is a composition of the functions computed by individual layers (e.g., linear layers and nonlinearities):

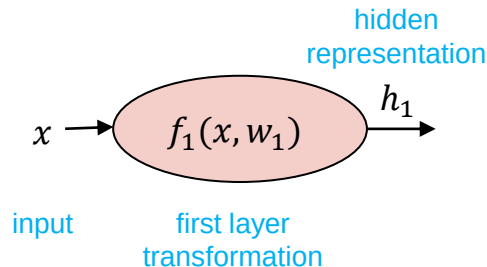


Recall: Multi-layer Neural Networks

- The function computed by the network is a composition of the functions computed by individual layers (e.g., linear layers and nonlinearities):

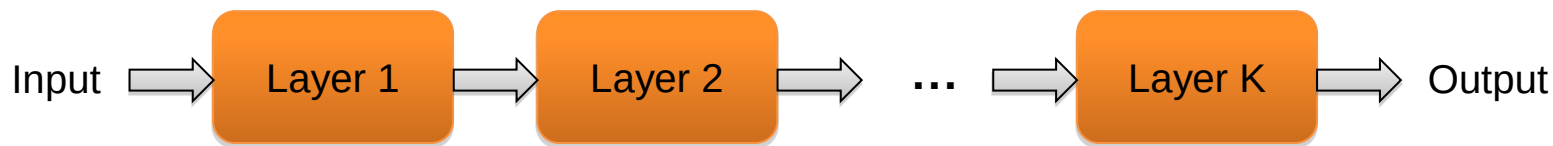


- More precisely:

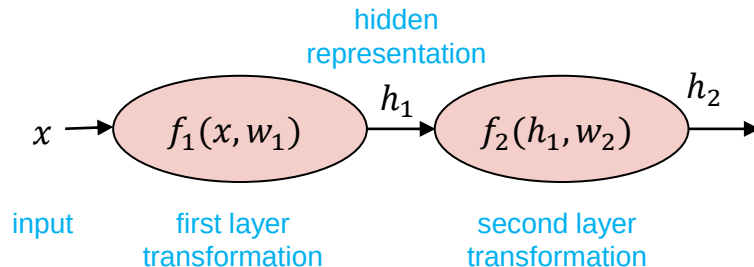


Recall: Multi-layer Neural Networks

- The function computed by the network is a composition of the functions computed by individual layers (e.g., linear layers and nonlinearities):

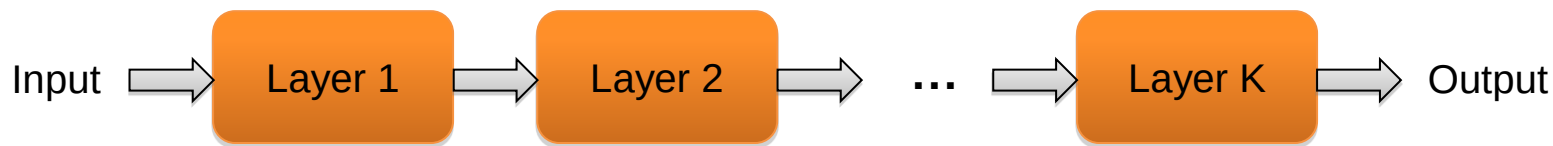


- More precisely:

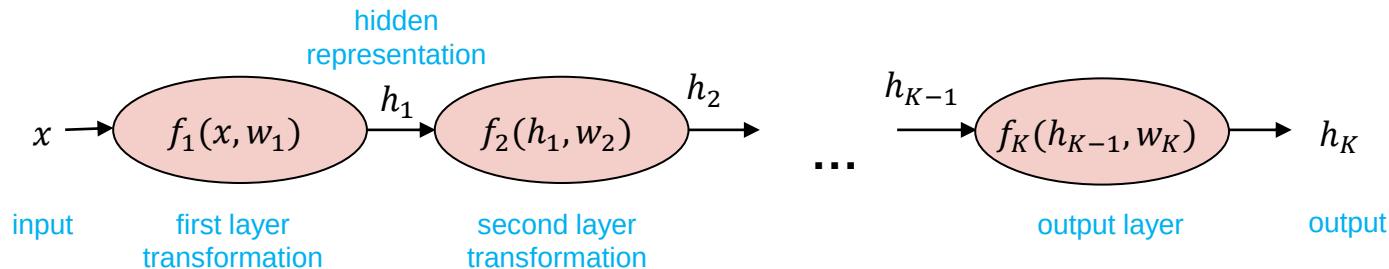


Recall: Multi-layer Neural Networks

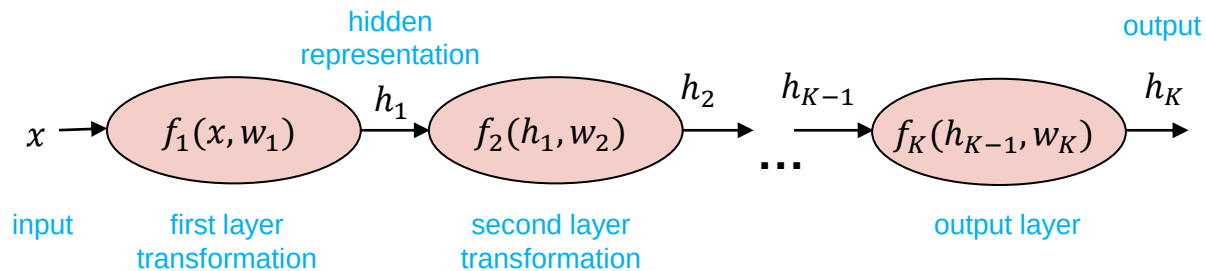
- The function computed by the network is a composition of the functions computed by individual layers (e.g., linear layers and nonlinearities):



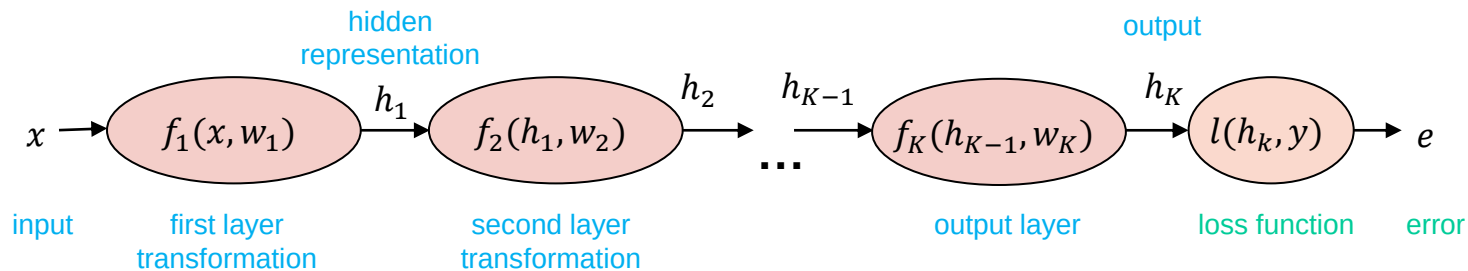
- More precisely:



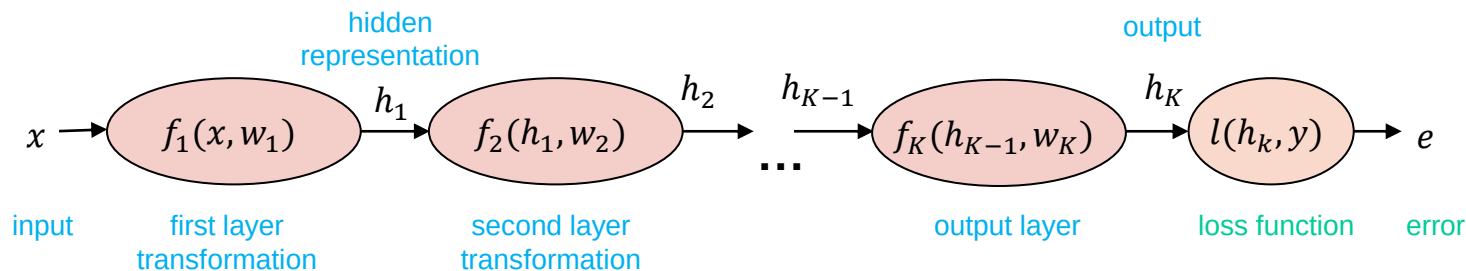
Training a Multi-layer Network



Training a Multi-layer Network

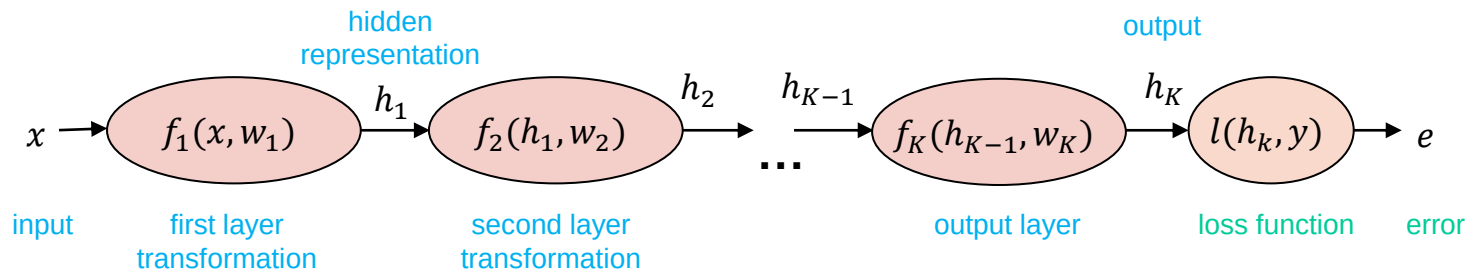


Training a Multi-layer Network



- What is the SGD update for the parameters w_k of the k th layer?

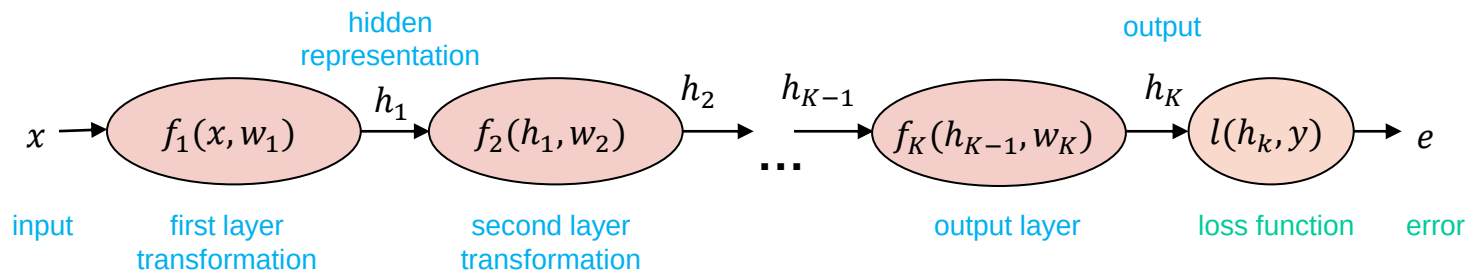
Training a Multi-layer Network



- What is the SGD update for the parameters w_k of the k th layer?

$$w_k \leftarrow w_k - \eta \frac{\partial e}{\partial w_k}$$

Training a Multi-layer Network



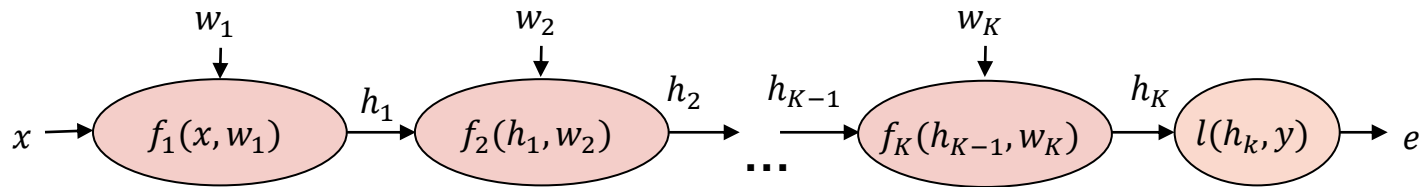
- What is the SGD update for the parameters w_k of the k th layer?

$$w_k \leftarrow w_k - \eta \frac{\partial e}{\partial w_k}$$

- To train the network, we need to find the

gradient of the error w.r.t. the parameters of each layer, $\frac{\partial e}{\partial w_k}$

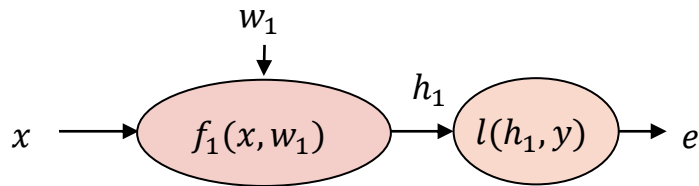
Computation Graph



Chain Rule

Let's start with $k = 1$

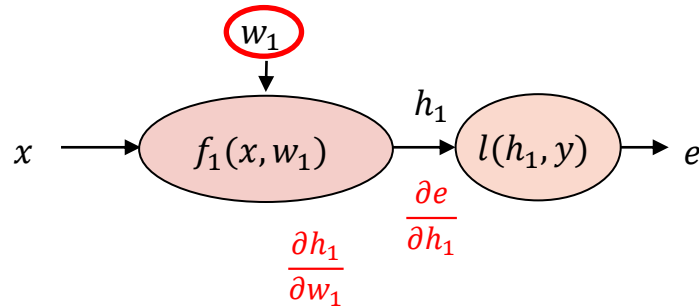
$$e = l(f_1(x, w_1), y)$$



Chain Rule

Let's start with $k = 1$

$$e = l(f_1(x, w_1), y)$$

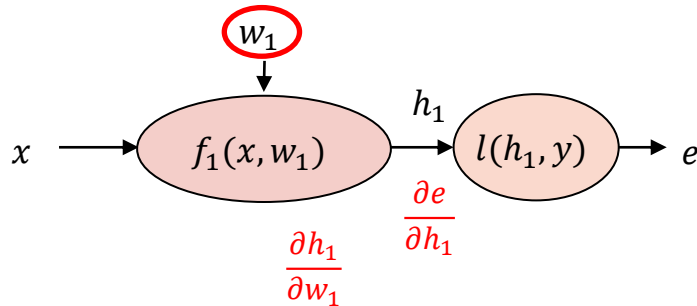


Chain Rule

Let's start with $k = 1$

$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$



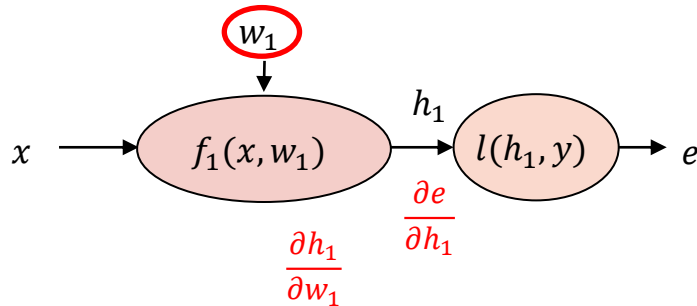
Chain Rule

Let's start with $k = 1$

$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Example: $e = (y - w_1^T x)^2$



Chain Rule

Let's start with $k = 1$

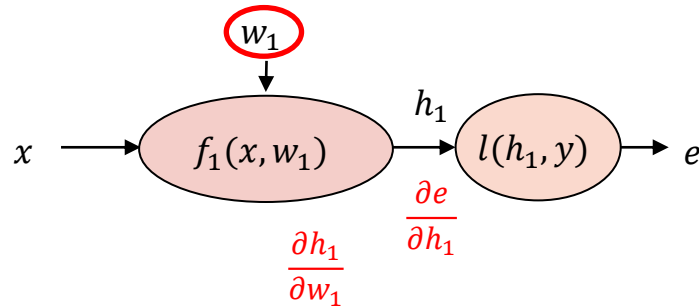
$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Example: $e = (y - w_1^T x)^2$

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$e = l(h_1, y) = (y - h_1)^2$$



Chain Rule

Let's start with $k = 1$

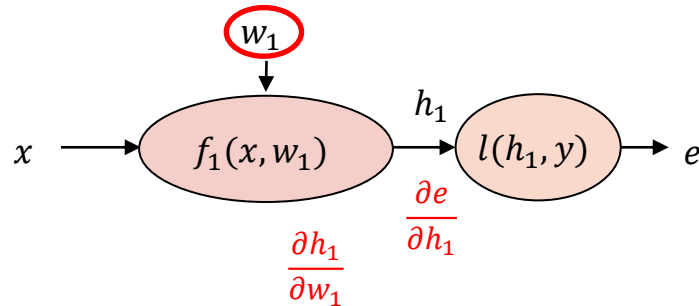
$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Example: $e = (y - w_1^T x)^2$

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$e = l(h_1, y) = (y - h_1)^2$$



$$\frac{\partial h_1}{\partial w_1} = \text{??????}$$

$$\frac{\partial e}{\partial h_1} = \text{??????}$$

Chain Rule

Let's start with $k = 1$

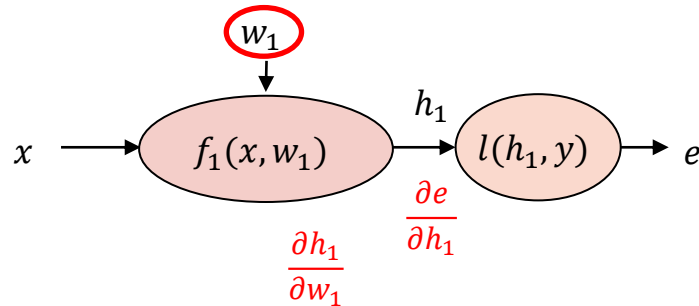
$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Example: $e = (y - w_1^T x)^2$

$$h_1 = f_1(x, w_1) = w_1^T x$$

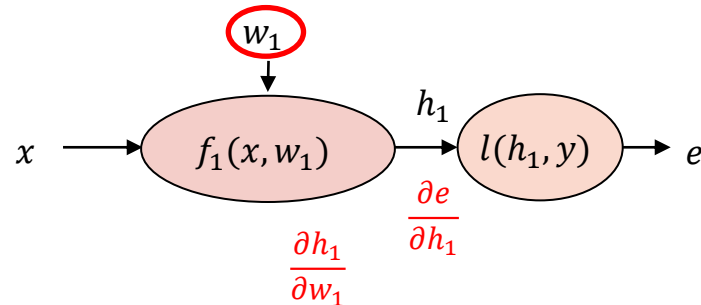
$$e = l(h_1, y) = (y - h_1)^2$$



$$\begin{aligned} \frac{\partial h_1}{\partial w_1} &= x \\ \frac{\partial e}{\partial h_1} &= -2(y - h_1) = -2(y - w_1^T x) \end{aligned}$$

Chain Rule

Let's start with $k = 1$



$$e = l(f_1(x, w_1), y)$$

$$\frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Example: $e = (y - w_1^T x)^2$

$$h_1 = f_1(x, w_1) = w_1^T x$$

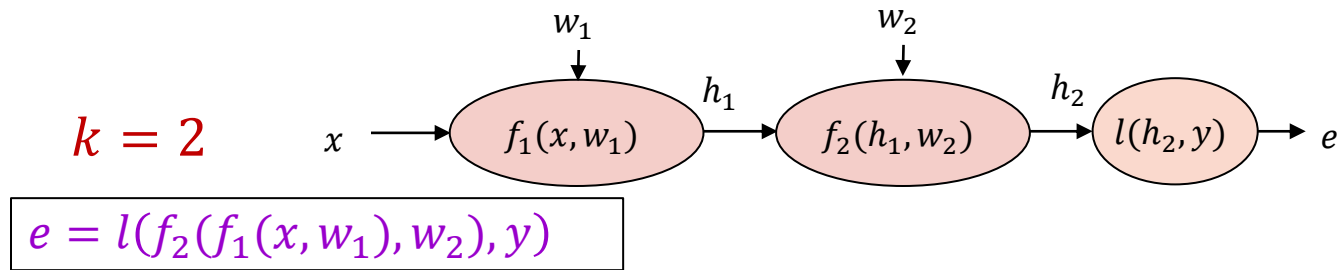
$$e = l(h_1, y) = (y - h_1)^2$$

$$\frac{\partial h_1}{\partial w_1} = x$$

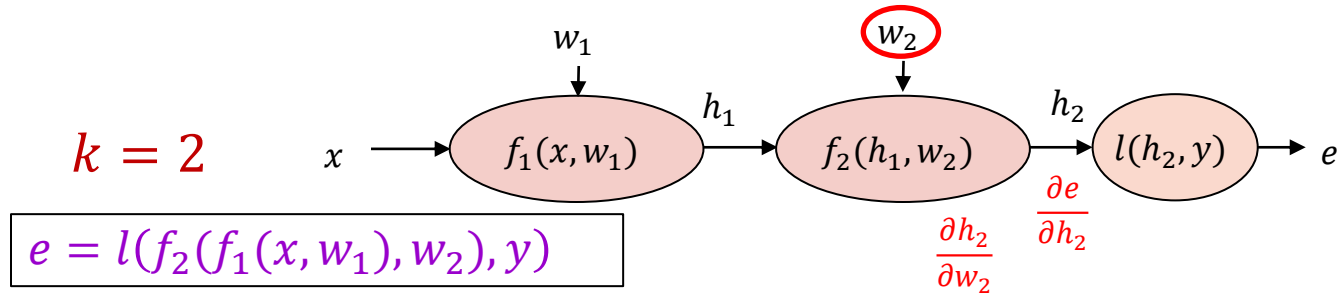
$$\frac{\partial e}{\partial h_1} = -2(y - h_1) = -2(y - w_1^T x)$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1} = -2x(y - w_1^T x)$$

Chain Rule

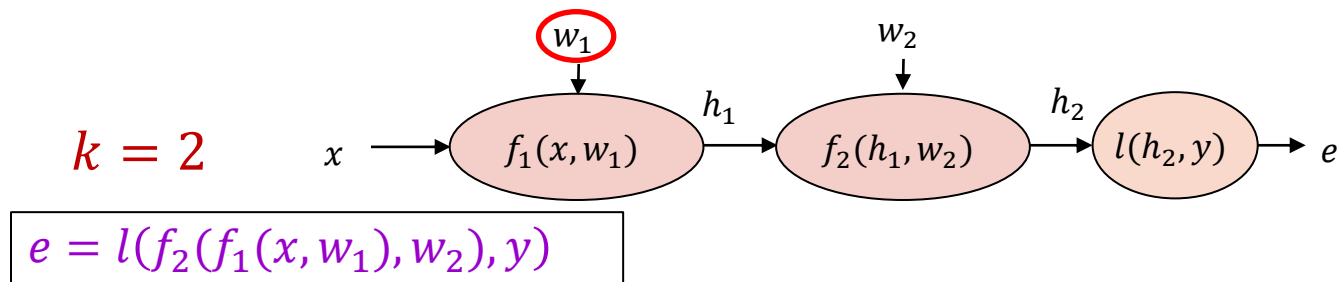


Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

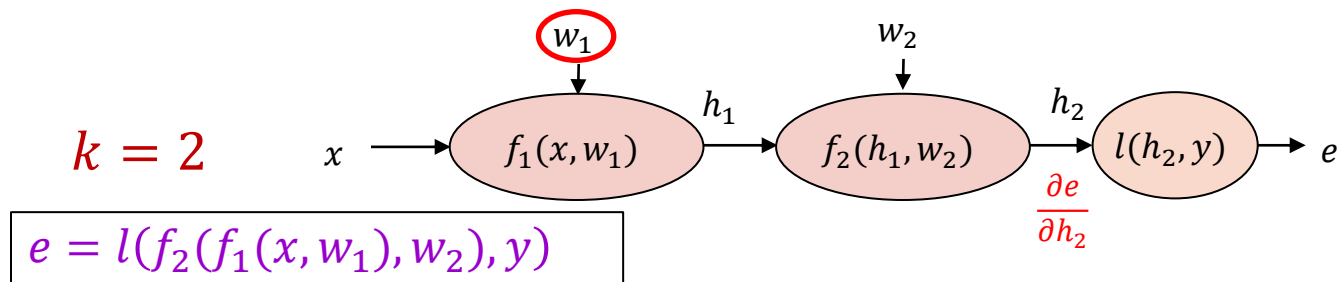
Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} =$$

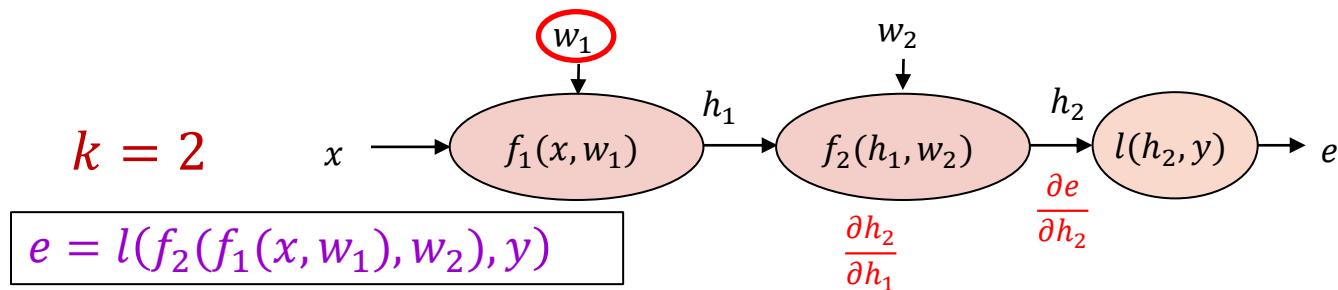
Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2}$$

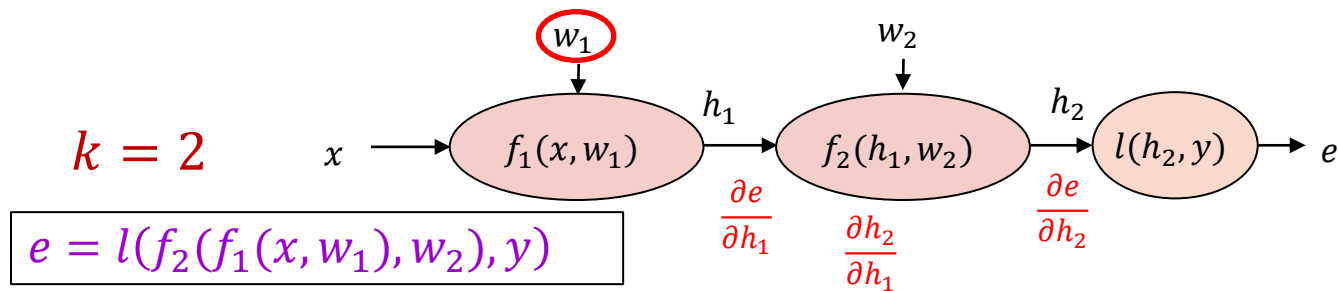
Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}$$

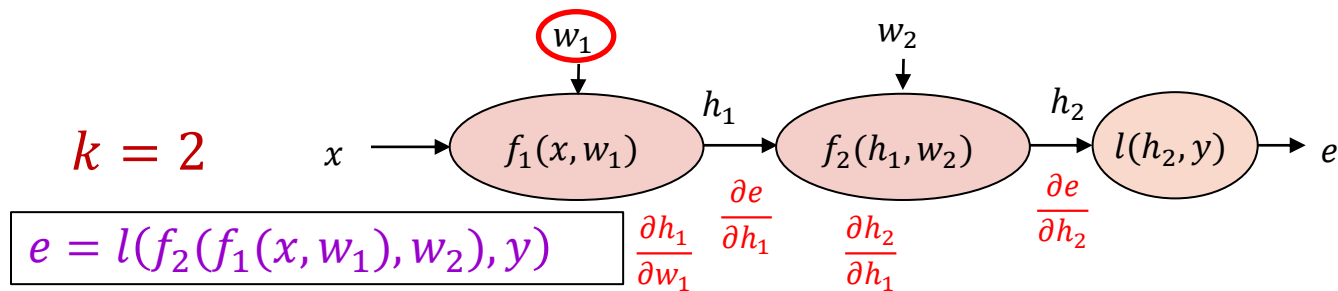
Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}}$$

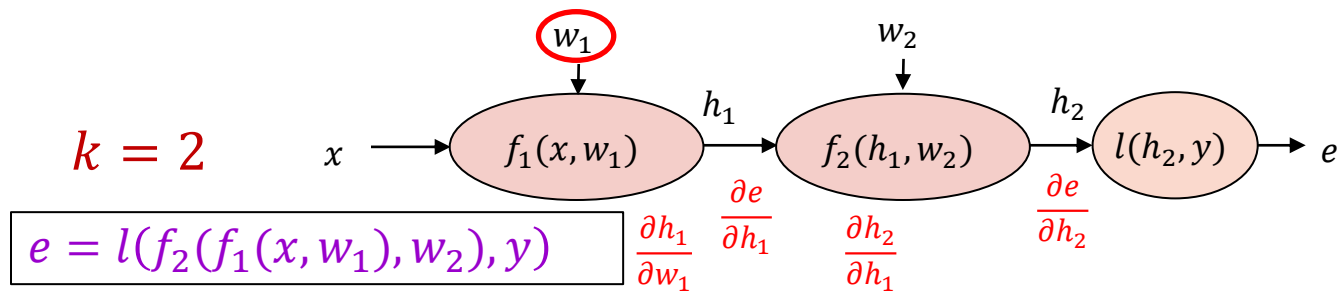
Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial w_1}$$

Chain Rule

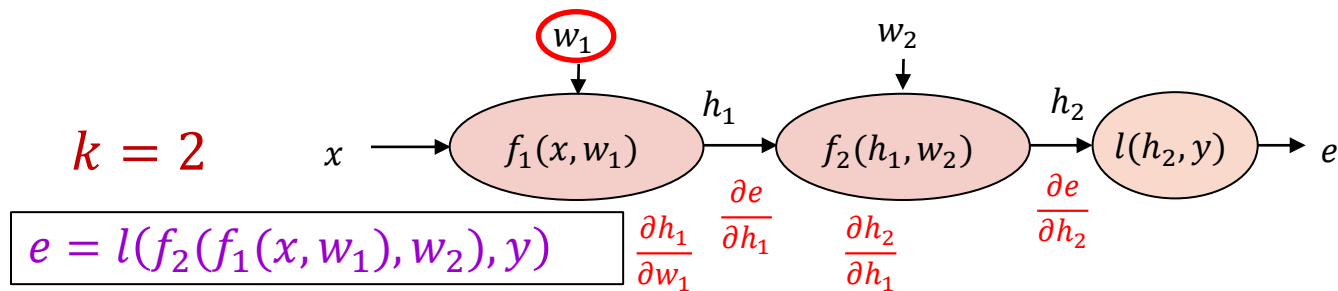


$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial w_1}$$

Example: $e = -\log(\sigma(w_1^T x))$ (assume $y = 1$)

Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial w_1}$$

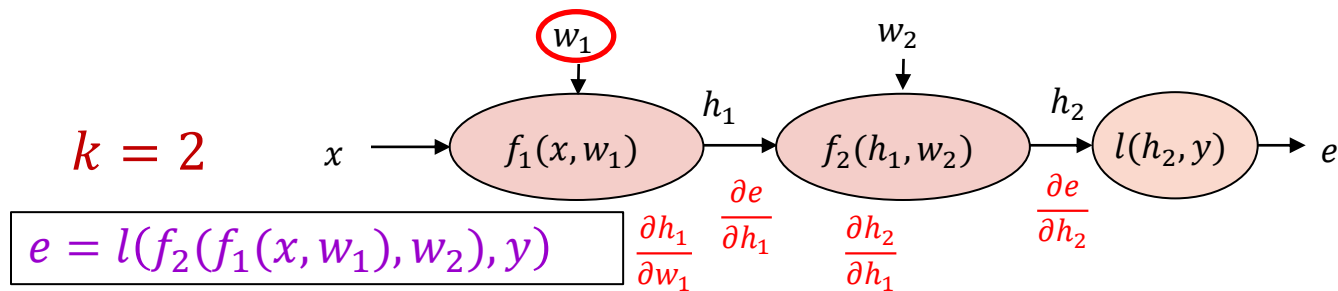
Example: $e = -\log(\sigma(w_1^T x))$ (assume $y = 1$)

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$h_2 = f_2(h_1) = \sigma(h_1)$$

$$e = l(h_2, 1) = -\log(h_2)$$

Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial w_1}$$

Example: $e = -\log(\sigma(w_1^T x))$ (assume $y = 1$)

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$h_2 = f_2(h_1) = \sigma(h_1)$$

$$e = l(h_2, 1) = -\log(h_2)$$

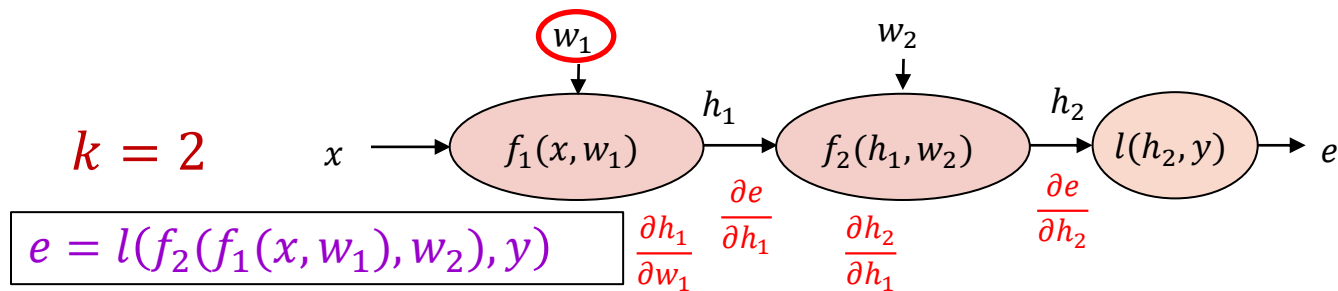
$$\frac{\partial e}{\partial h_2} =$$

$$\frac{\partial h_1}{\partial w_1} =$$

$$\frac{\partial h_2}{\partial h_1} =$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w_1} =$$

Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial w_1}$$

Example: $e = -\log(\sigma(w_1^T x))$ (assume $y = 1$)

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$h_2 = f_2(h_1) = \sigma(h_1)$$

$$e = l(h_2, 1) = -\log(h_2)$$

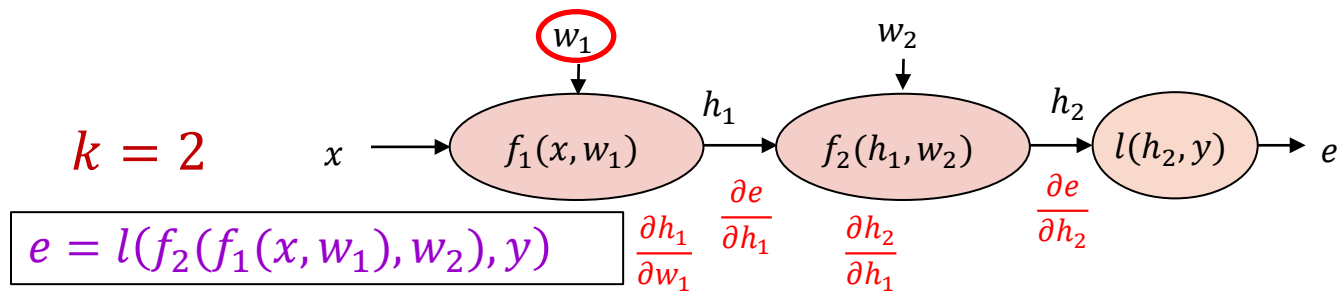
$$\frac{\partial e}{\partial h_2} = -\frac{1}{h_2}$$

$$\frac{\partial h_1}{\partial w_1} = x$$

$$\frac{\partial h_2}{\partial h_1} = \sigma'(h_1) = \sigma(h_1)(1 - \sigma(h_1))$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w_1} =$$

Chain Rule



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial w_1} = \boxed{\frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial w_1}$$

Example: $e = -\log(\sigma(w_1^T x))$ (assume $y = 1$)

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$h_2 = f_2(h_1) = \sigma(h_1)$$

$$e = l(h_2, 1) = -\log(h_2)$$

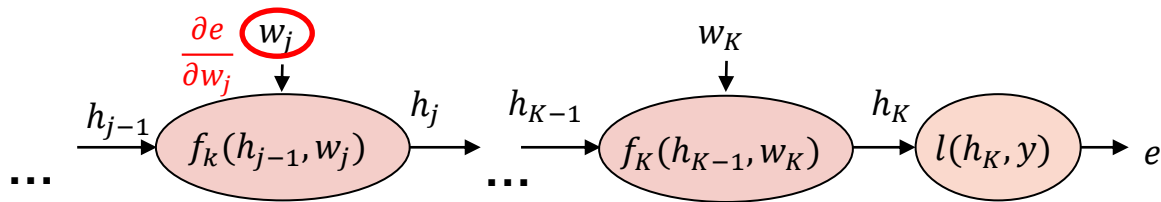
$$\frac{\partial e}{\partial h_2} = -\frac{1}{h_2}$$

$$\frac{\partial h_1}{\partial w_1} = x$$

$$\frac{\partial h_2}{\partial h_1} = \sigma'(h_1) = \sigma(h_1)(1 - \sigma(h_1))$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w_1} = -\frac{1}{\sigma(w_1^T x)} \sigma(w_1^T x) (1 - \sigma(w_1^T x)) x = -\sigma(-w_1^T x) x$$

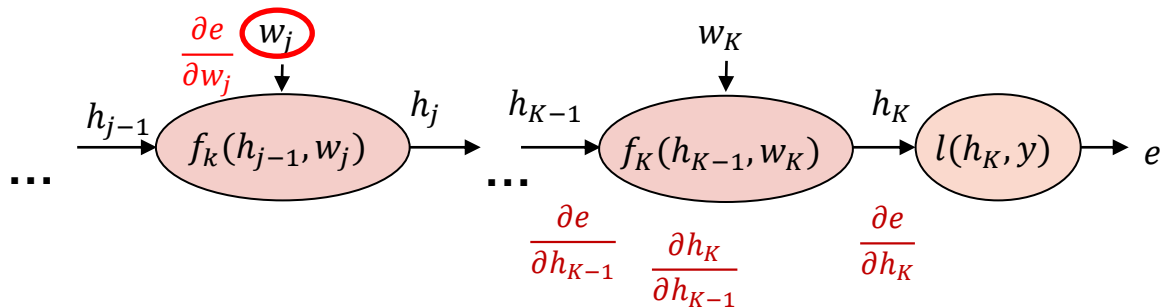
Chain Rule



General case:

$$\frac{\partial e}{\partial w_j} = \frac{\partial e}{\partial h_K} \frac{\partial h_K}{\partial h_{K-1}} \cdots \frac{\partial h_{j+1}}{\partial h_j} \frac{\partial h_j}{\partial w_j}$$

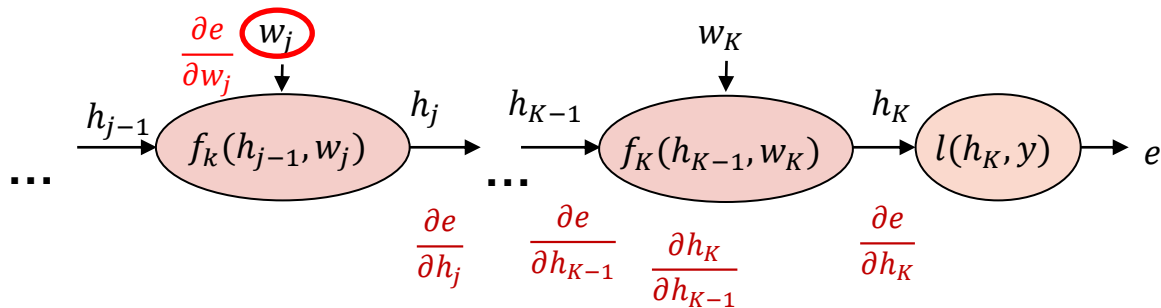
Chain Rule



General case:

$$\frac{\partial e}{\partial w_j} = \frac{\partial e}{\partial h_K} \frac{\partial h_K}{\partial h_{K-1}} \cdots \frac{\partial h_{j+1}}{\partial h_j} \frac{\partial h_j}{\partial w_j}$$

Chain Rule



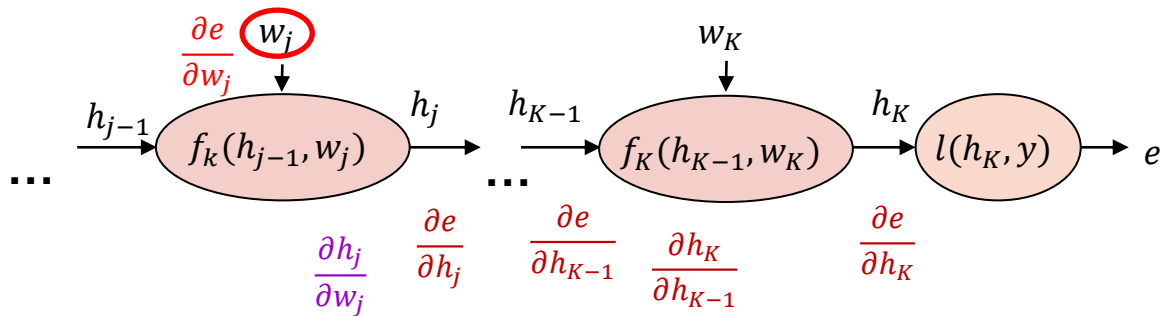
General case:

$$\frac{\partial e}{\partial w_j} = \boxed{\frac{\partial e}{\partial h_K} \frac{\partial h_K}{\partial h_{K-1}} \cdots \frac{\partial h_{j+1}}{\partial h_j}} \frac{\partial h_j}{\partial w_j}$$

Upstream gradient

$$\frac{\partial e}{\partial h_j}$$

Chain Rule



General case:

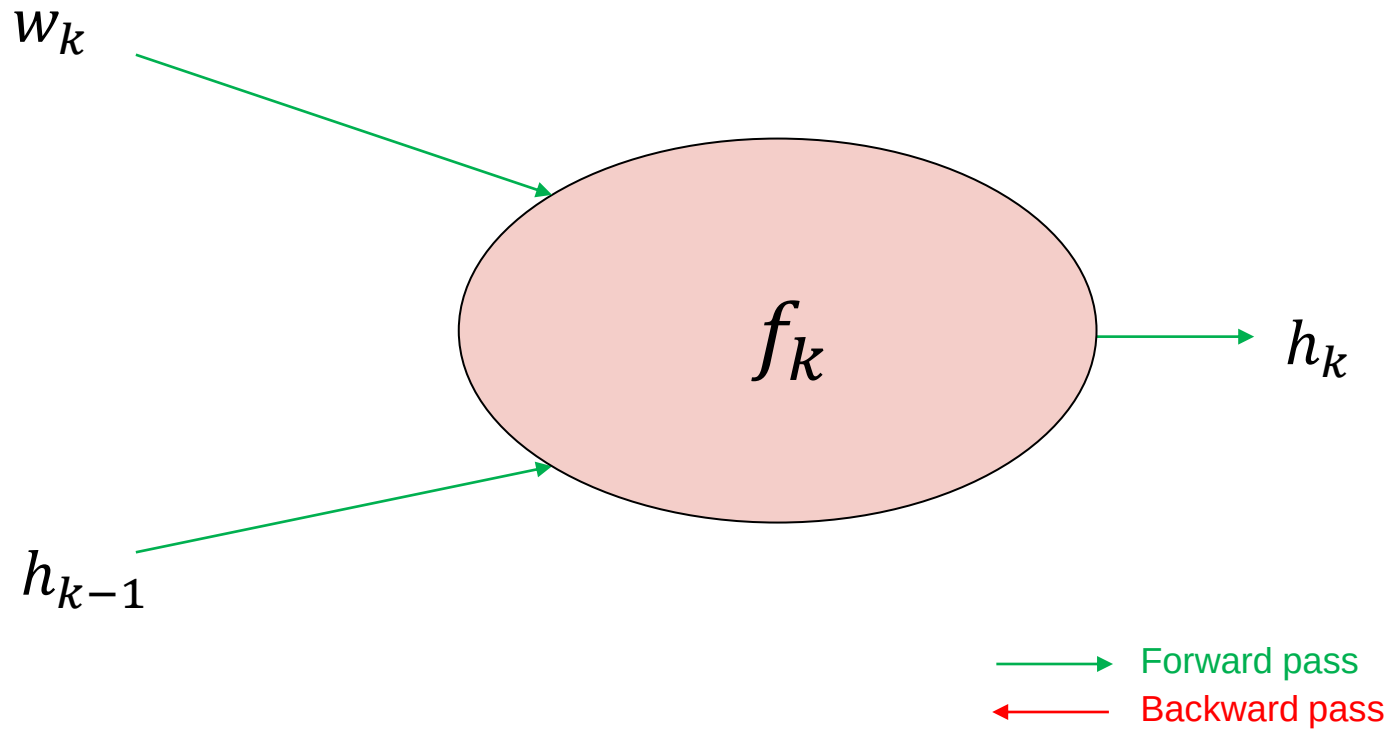
$$\frac{\partial e}{\partial w_j} = \boxed{\frac{\partial e}{\partial h_K} \frac{\partial h_K}{\partial h_{K-1}} \cdots \frac{\partial h_{j+1}}{\partial h_j}} \frac{\partial h_j}{\partial w_j}$$

Upstream gradient

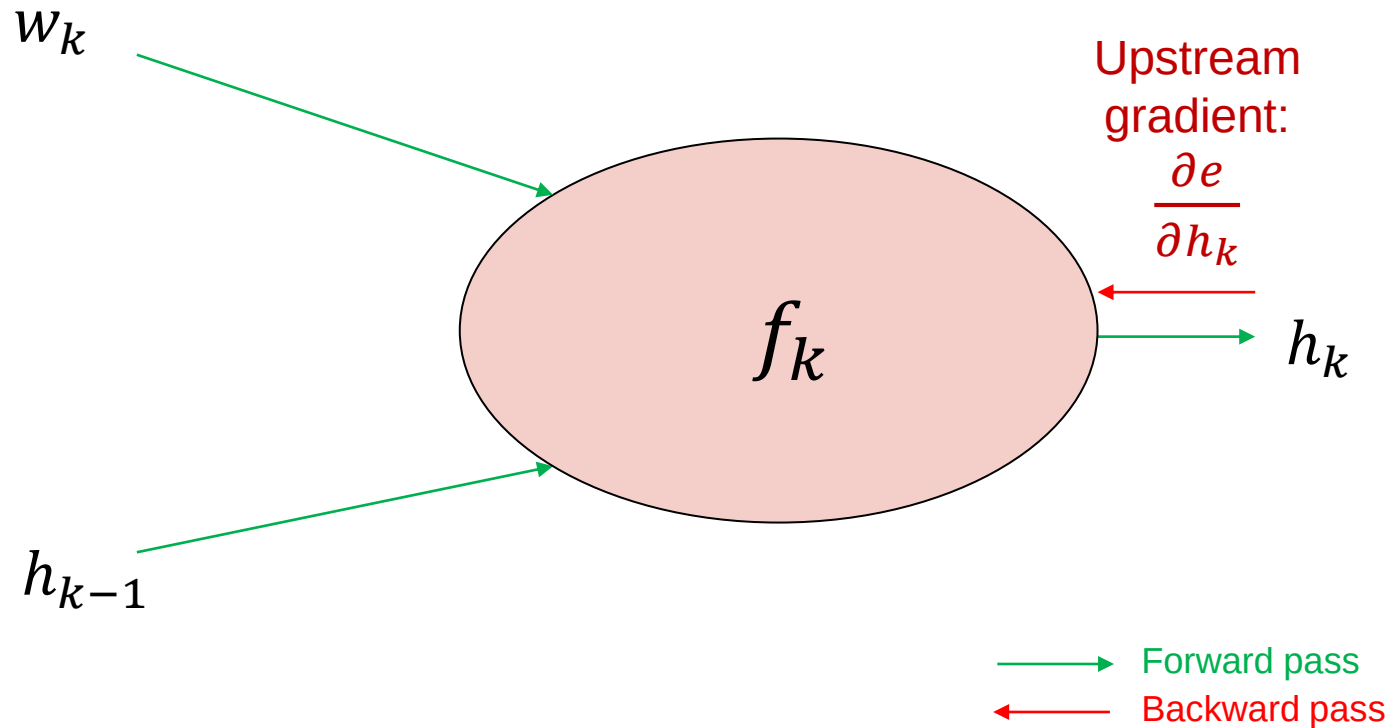
$$\frac{\partial e}{\partial h_j}$$

Local
gradient

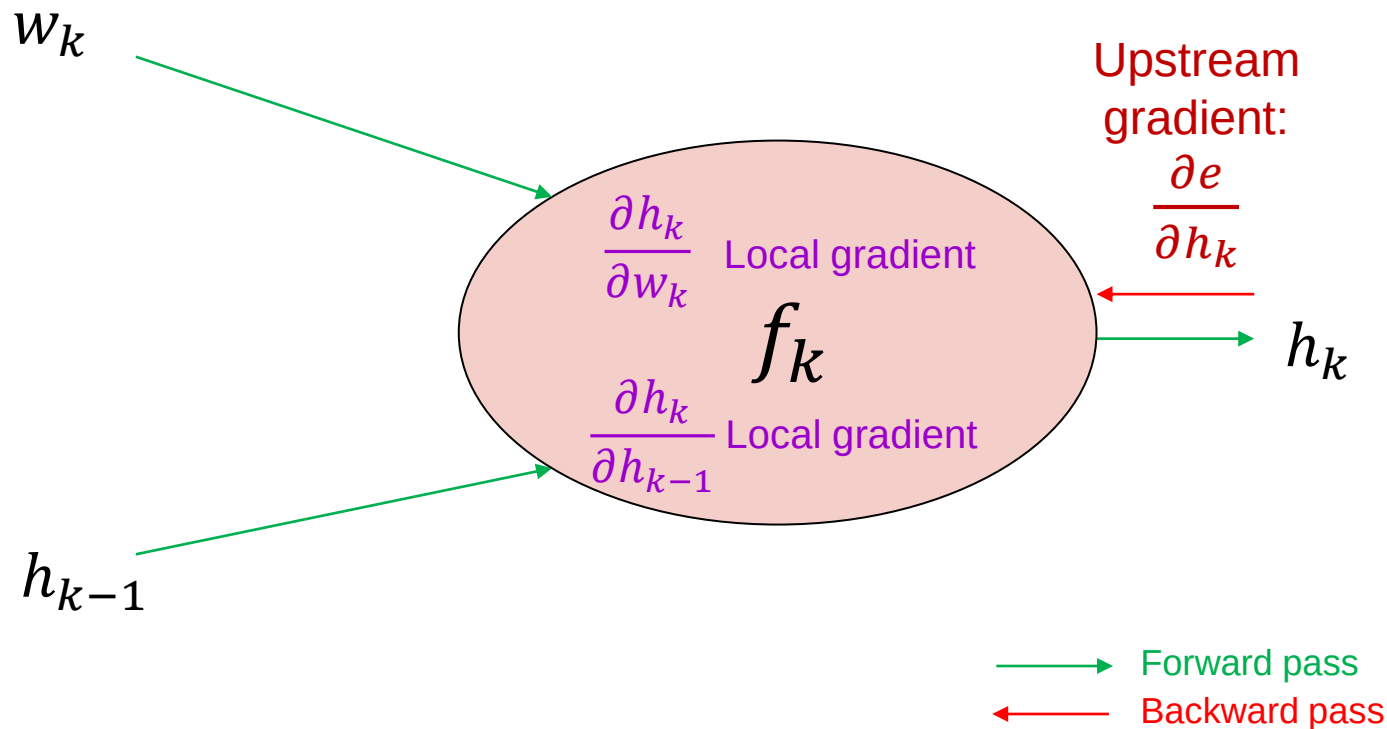
Backpropagation Summary



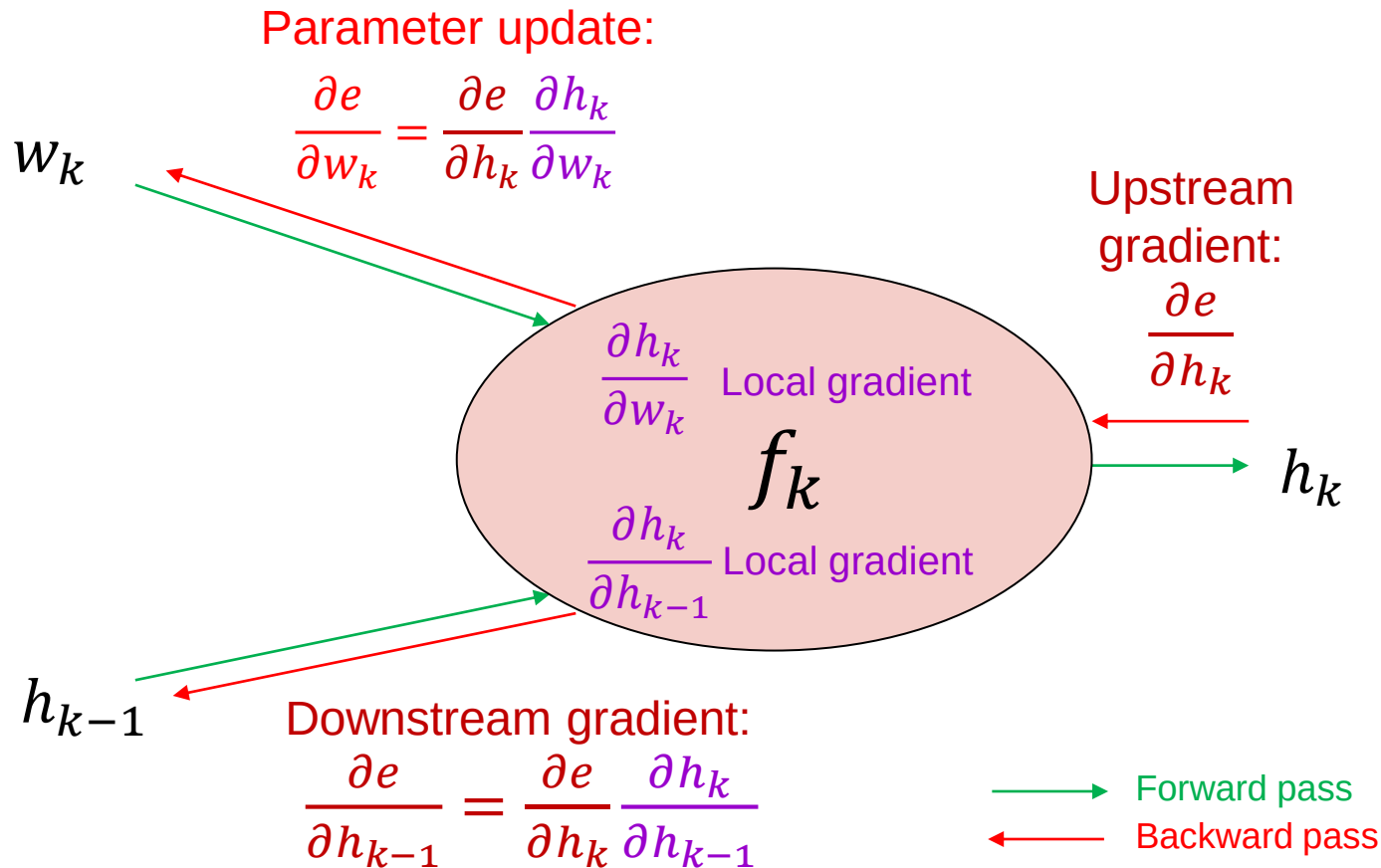
Backpropagation Summary



Backpropagation Summary

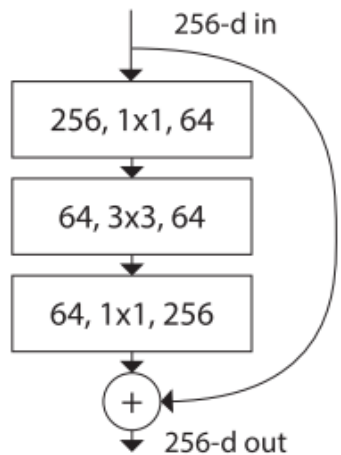


Backpropagation Summary

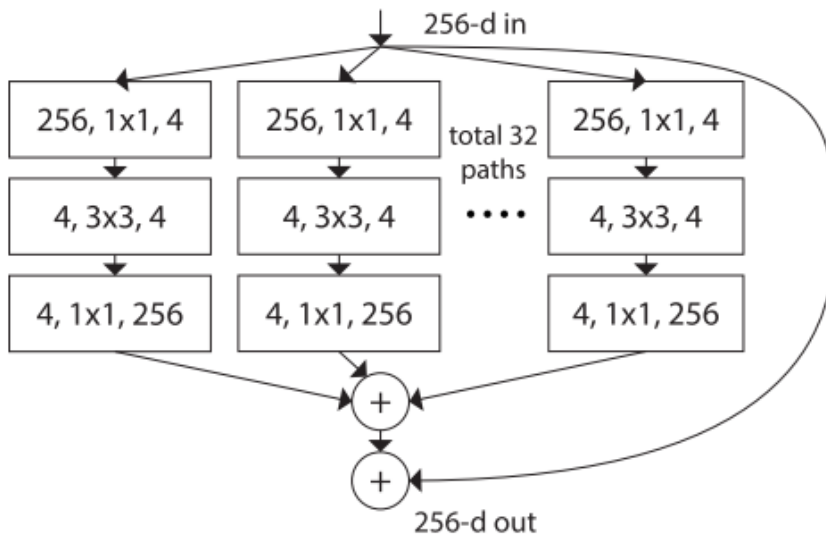


What about more general computation graphs?

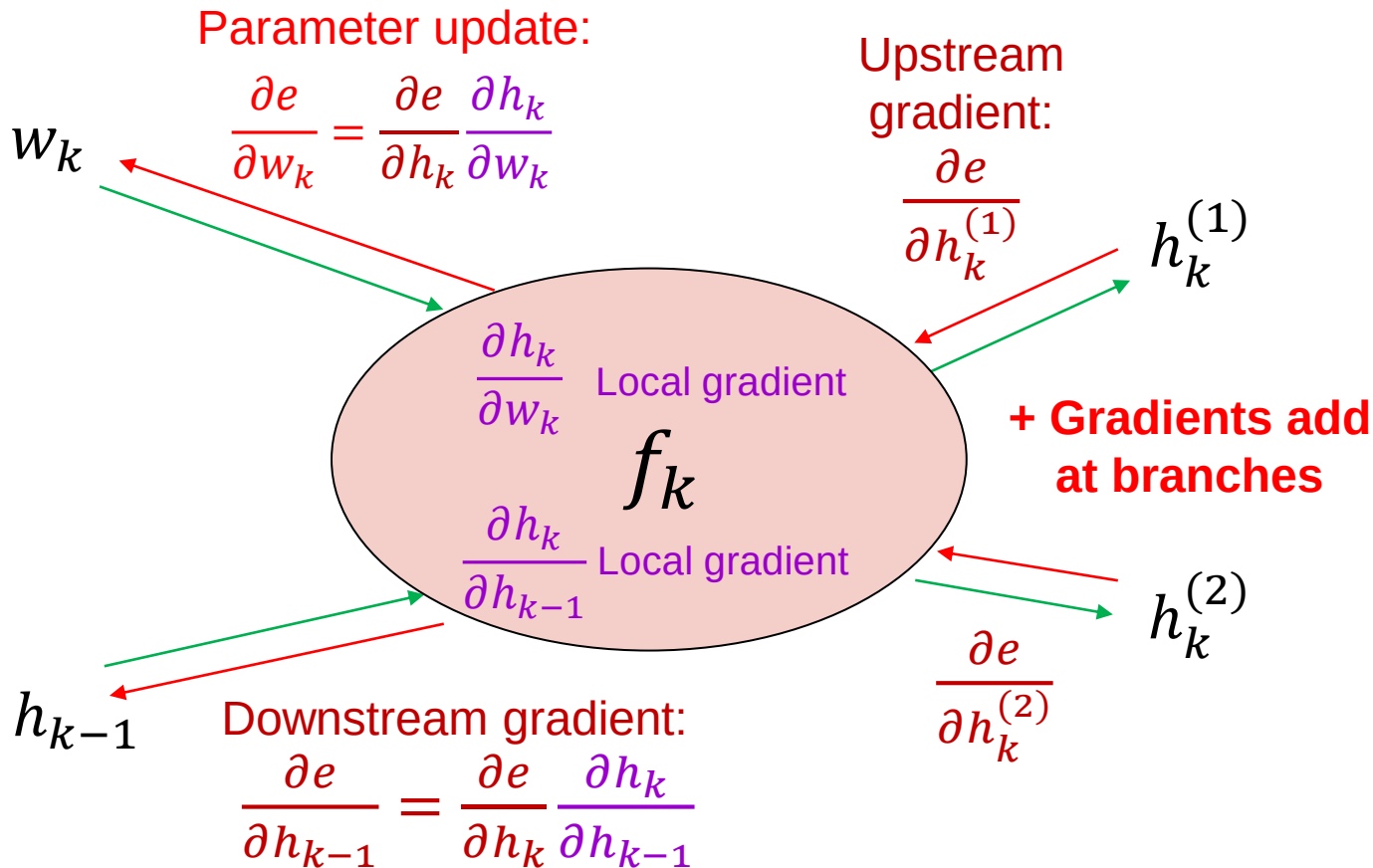
ResNet



ResNeXt



What about more general computation graphs?



Common Frameworks Do this Automatically

All you have to do is define the forward pass, in terms of known operators.



<https://www.tensorflow.org/>



<https://pytorch.org/>

Acknowledgement

Most of the slides are based on the ML course of:

- Mingmin Zhao and Jiatao Gu, University of Pennsylvania

THANK YOU