Machine Learning for Image Analysis

Lecture 1: Neural Networks - Theory and Image Classification

Dagmar Kainmueller, 16.04.2024

Course Outline

- **Introduction to Image Analysis**
- **Basics: Neural Networks**
- Convolutional Neural Networks
- Transformers
- Model Interpretability
- Self-supervised Learning
- Generative Models (GANs, Diffusion)

Requirements

- Linear Regression, Logistic Regression
  - Hypothesis function
  - Loss function (cross-entropy, mean squared error)
  - Regularization: Weight decay
- Gradient Descent
- Basic linear algebra and multi-variate calculus

Introduction to Image Analysis

Image Classification

Assign one class label to an image

{dog, cat, truck, plane, …}

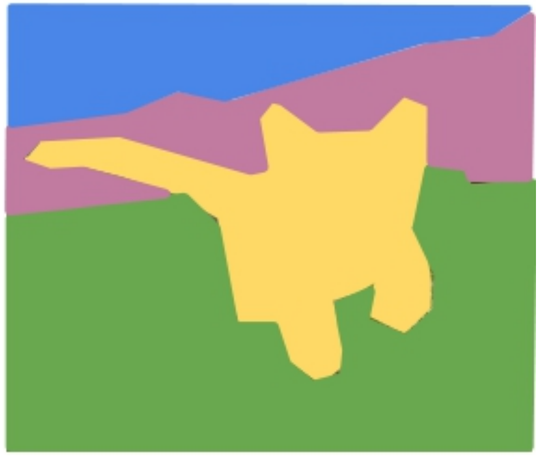
⟼ CAT

- No spatial extent of the output
- What if localized information is sought in an image? → Pixel-wise tasks

Image Segmentation: Semantic Segmentation

Assign one class label to each pixel in an image

{dog, cat, truck, plane, …}

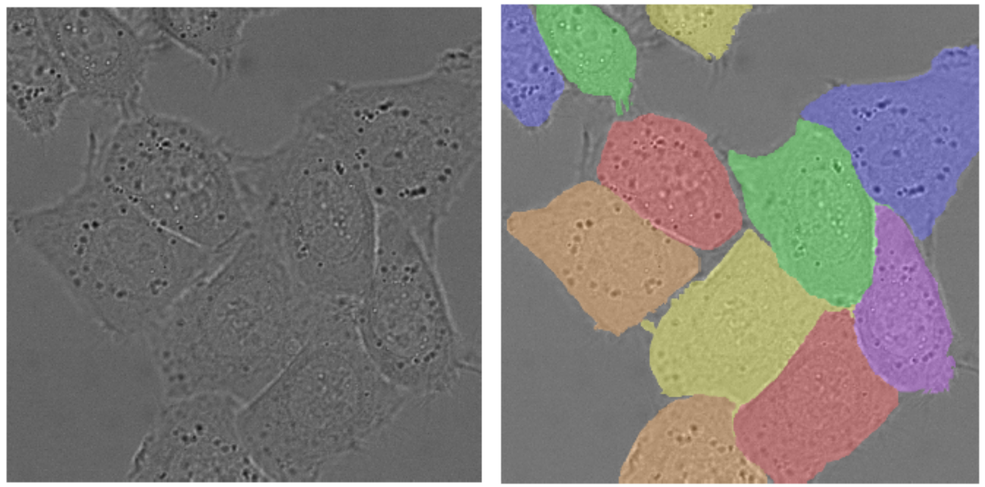
{cat, grass, tree, sky}


{cow, grass, tree, sky}

- Does not distinguish individual object instances

Instance Segmentation

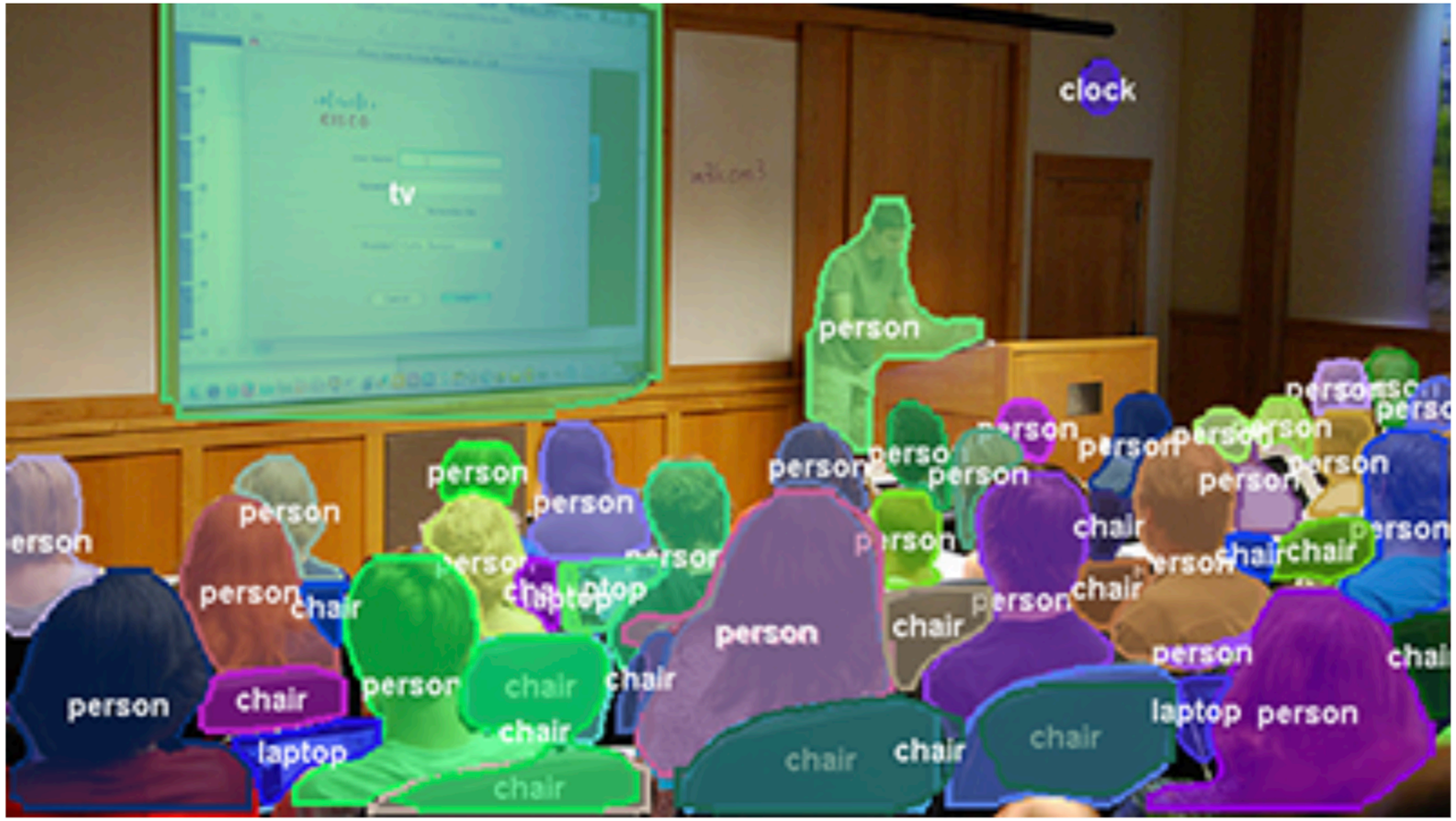
Assign one instance label to each pixel in an image



https://segment-anything.com/demo#

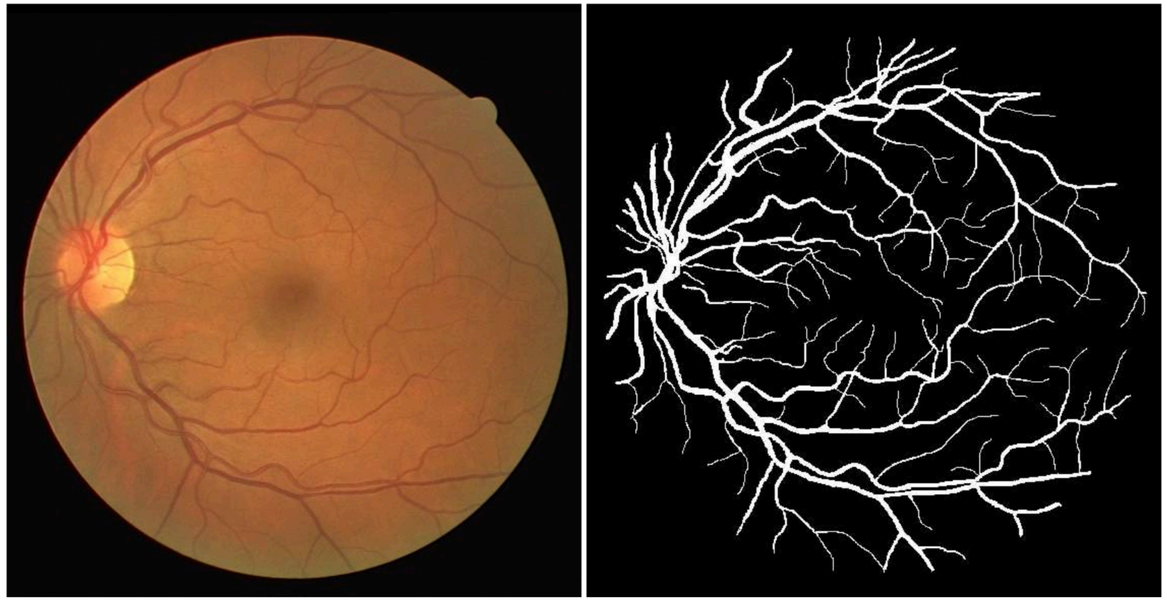
Semantic Instance Segmentation (aka Panoptic Segmentation)

Assign one class and instance label to each pixel in an image



Some more segmentation examples …

Foreground/Background Segmentation

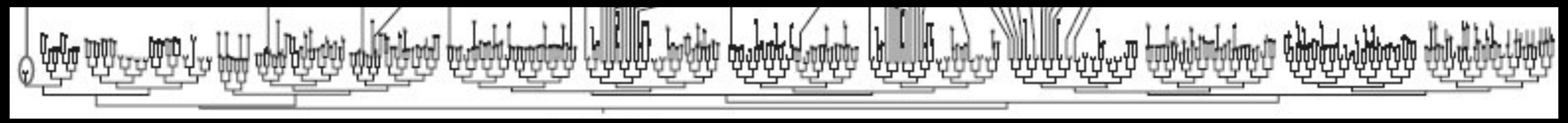


https://github.com/CVxTz/medical_image_segmentation

Semantic Parts Segmentation (aka Object Parsing)

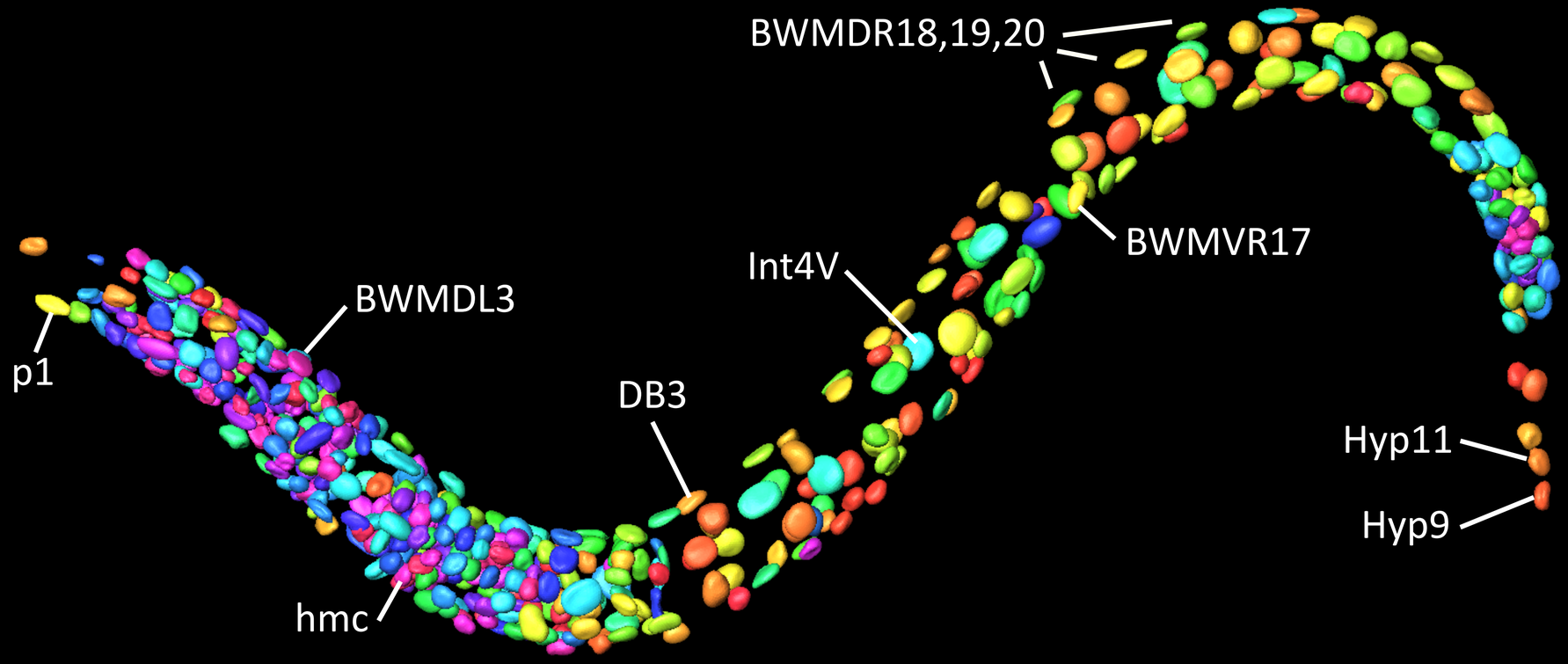
https://www.arxiv-vanity.com/papers/1611.06612/

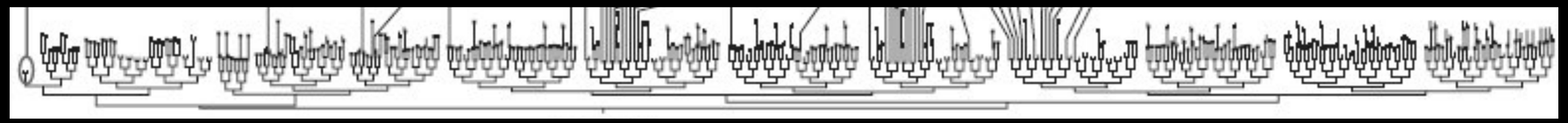
Some more segmentation examples ...



Sulston & Horvitz, *Dev. Biol.* 56 (1),1977

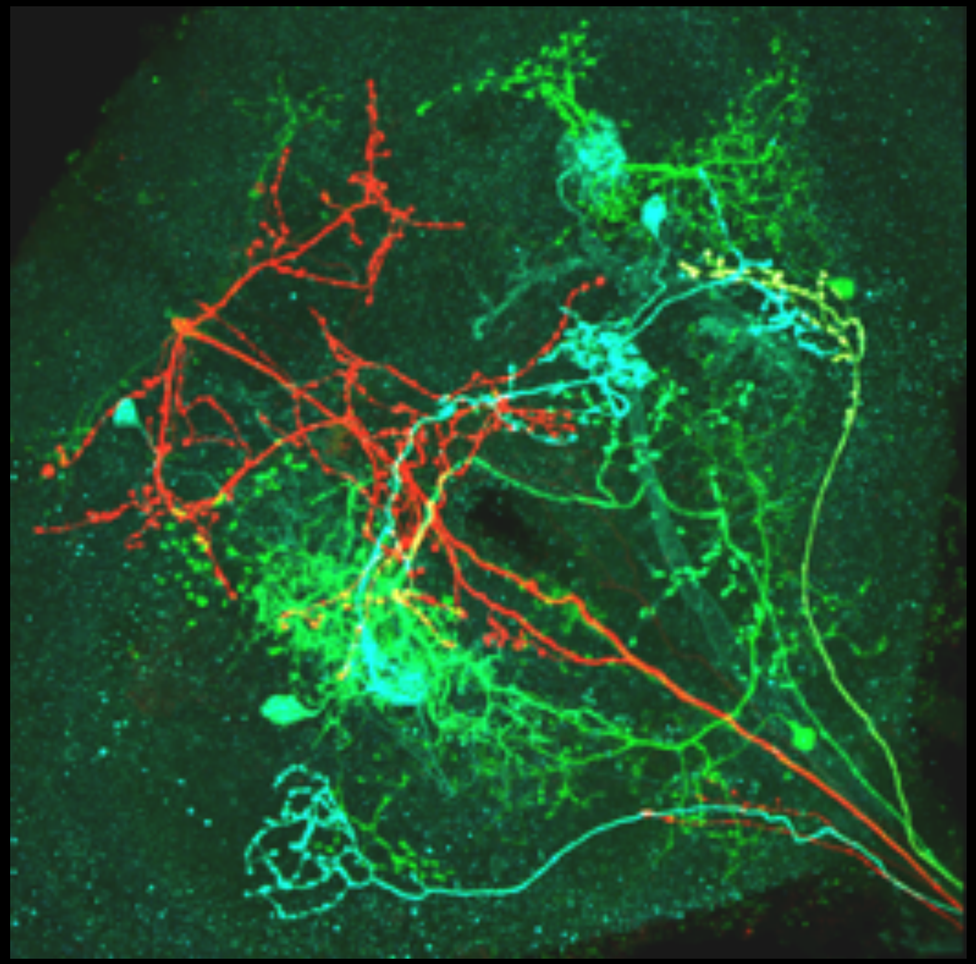
Some more segmentation examples …





Sulston & Horvitz, *Dev. Biol.* 56 (1),1977

Some more segmentation examples ...



FlyLight Project, HHMI Janelia

Other Image-related Learning Tasks

- Object tracking in video
- Regression tasks on images:
  image-wise, pixel-wise
- Image Generation
  from other image, from text,
  from set of conditions, …
- Image Captioning

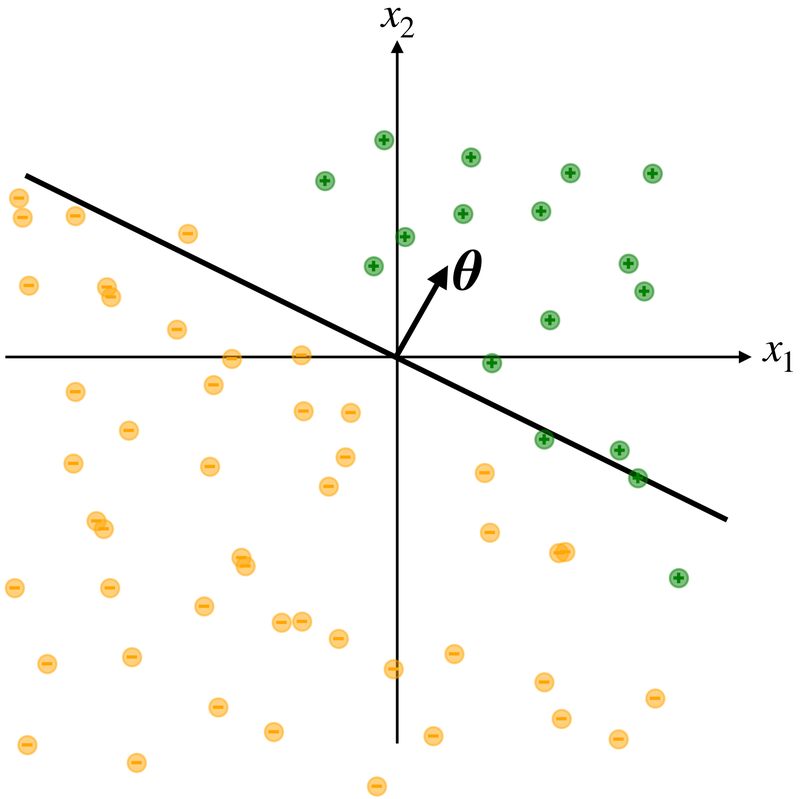
https://stablediffusionweb.com/

Basics: Neural Networks

- Multi-layer Perceptrons
- Backpropagation
- … on Images

The Perceptron [McCulloch and Pitts, 1943]

The Perceptron

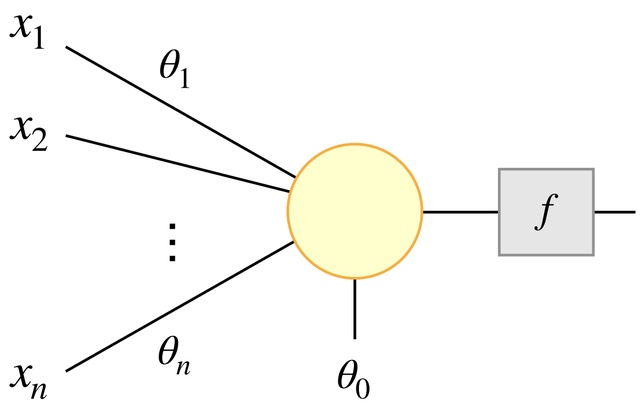


Algorithm : The Perceptron Learning Algorithm

$P \leftarrow$ samples with label 1
$N \leftarrow$ samples with label -1
Initialize $\boldsymbol{\theta}$ randomly
while !convergence **do**
  pick random sample $\mathbf{x}$
  if $\mathbf{x} \in P$ and $\boldsymbol{\theta}^T \mathbf{x} < 0$ **then**
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \mathbf{x}$
  end if
  if $\mathbf{x} \in N$ and $\boldsymbol{\theta}^T \mathbf{x} \geq 0$ **then**
    $\boldsymbol{\theta} = \boldsymbol{\theta} - \mathbf{x}$
  end if
end while
// the algorithm converges when all samples are classified correctly

The Perceptron

Linear classifier, building block of neural networks



Output: $h_\theta(x) = f\left([\theta_1, \ldots, \theta_n] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \theta_0\right)$

Input: $\tilde{\boldsymbol{x}} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$

Label: $y = \pm 1$

Weights: $\tilde{\boldsymbol{\theta}} = [\theta_1, \ldots, \theta_n]^T$

Bias: θ_0

Activation func: 1) $f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$

equivalent

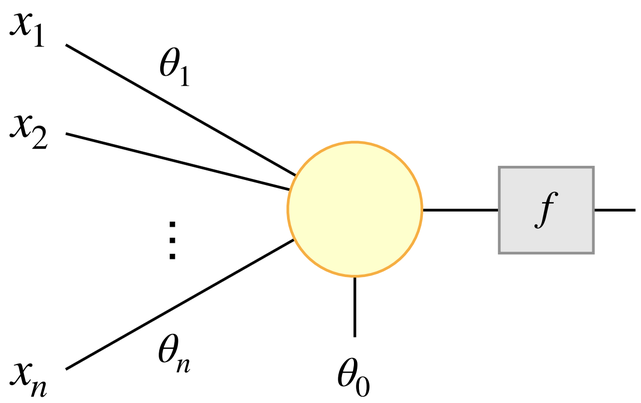
2) $f(x) = \tanh(x)$

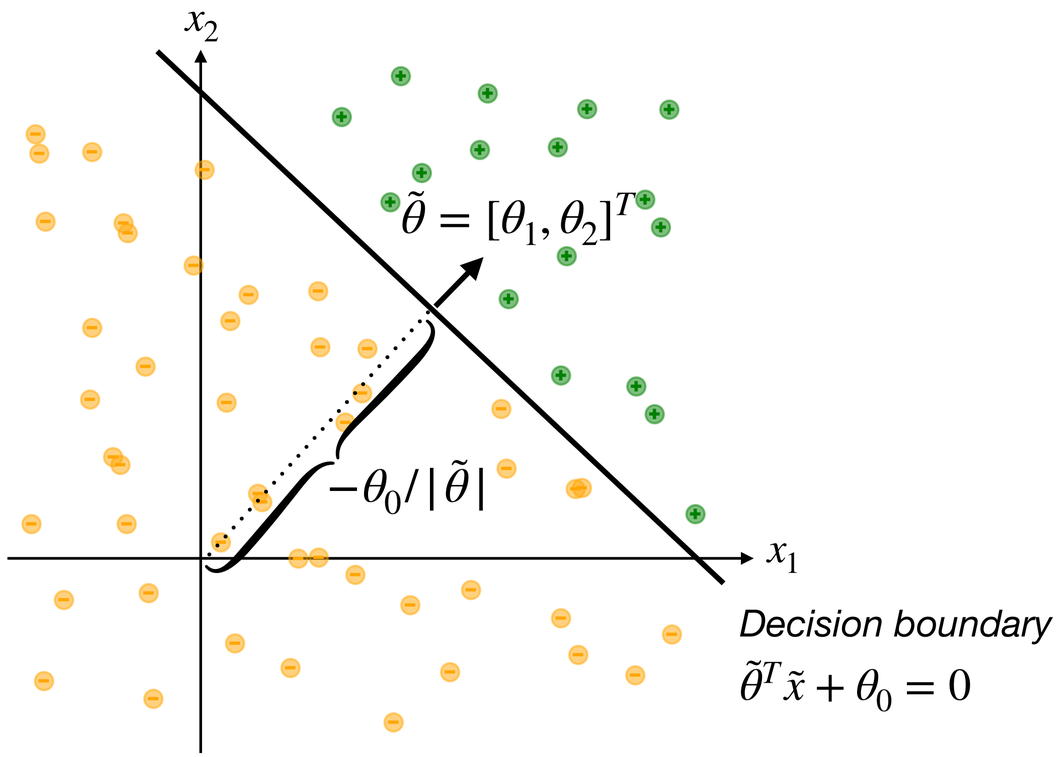
3) $f(x) = \dfrac{1}{1 + e^{-x}}$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

The Perceptron

Linear classifier, building block of neural networks





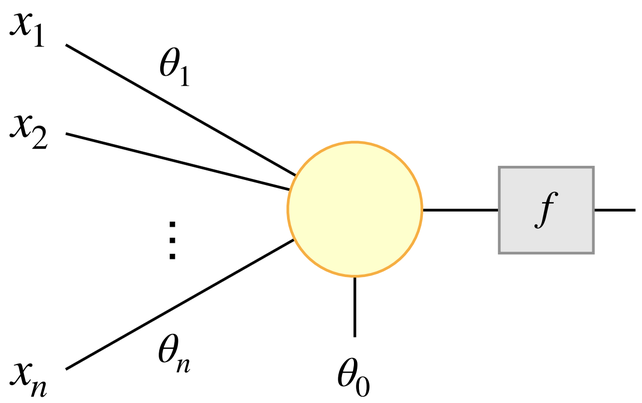
Decision boundary

$\tilde{\theta}^T \tilde{x} + \theta_0 = 0$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

The Perceptron

Linear classifier, building block of neural networks



Recap: Concise notation in **homogeneous coordinates**

Output: $h_\theta(x) = f([\theta_1, \ldots, \theta_n] \cdot [x_1, \ldots, x_n]^T + \theta_0) =: f(\tilde{\theta}^T \tilde{x} + \theta_0)$

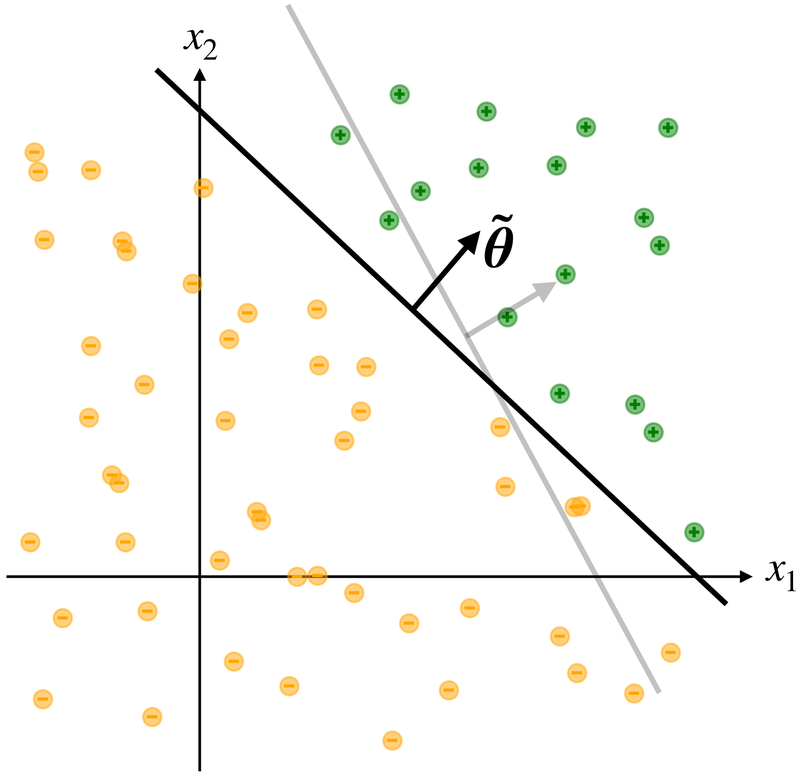
$= f([\theta_0, \theta_1, \ldots, \theta_n] \cdot [1, x_1, \ldots, x_n]^T)$

$=: f(\boldsymbol{\theta}^T \boldsymbol{x})$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

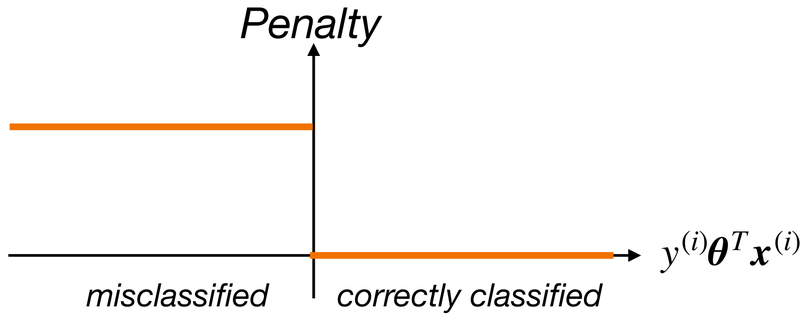
Loss function (aka cost function)

Goal: Find an optimal solution



Heaviside function

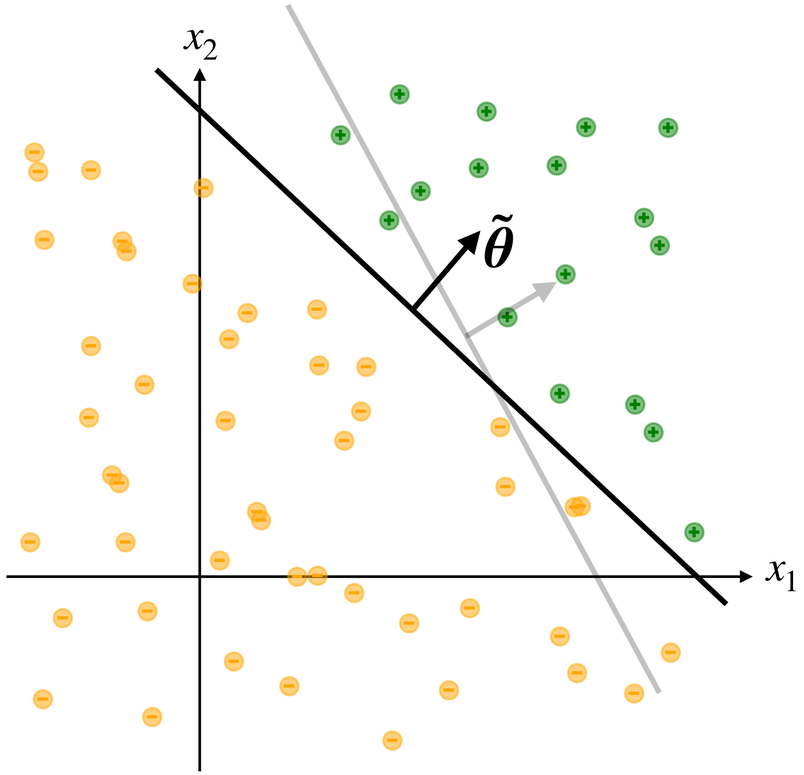
A) The 0-1 loss: $\varepsilon(\boldsymbol{\theta}) = \sum_{i=1}^{m} H(-y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})$



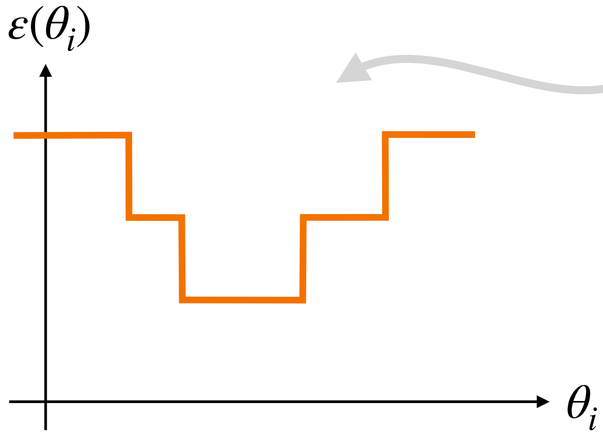
Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Loss function

Goal: Find an optimal solution



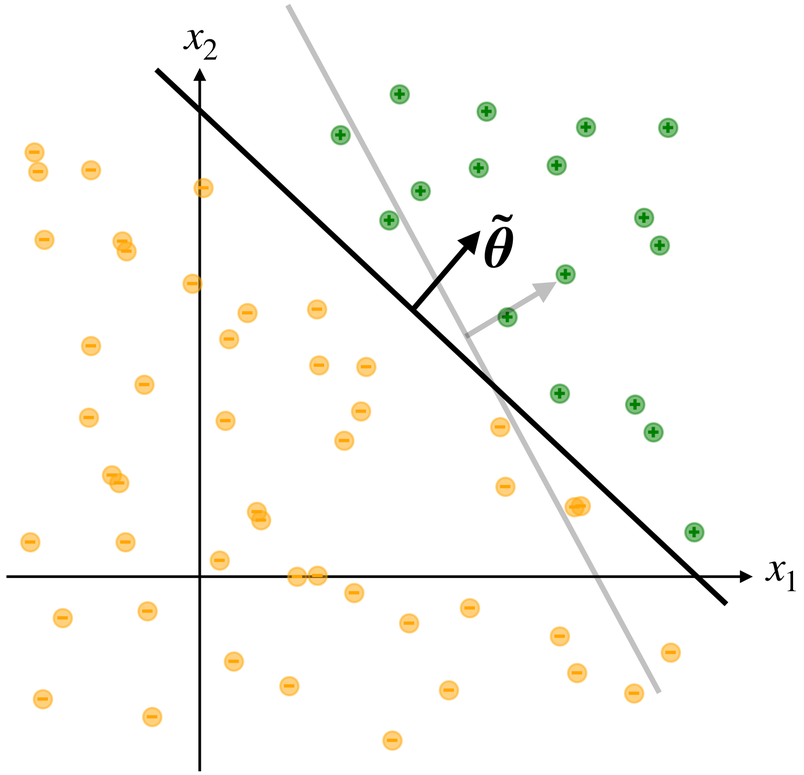
A) The 0-1 loss: $\varepsilon(\boldsymbol{\theta}) = \sum_{i=1}^{m} H(-y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})$



Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

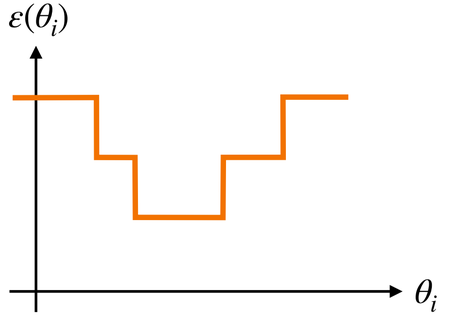
Loss function

Goal: Find an optimal solution



A) The 0-1 loss: $\varepsilon(\boldsymbol{\theta}) = \sum_{i=1}^{m} H(-y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})$

- We can't perform gradient descent
- Not convex
- Ambiguity of "optimal" solution



Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

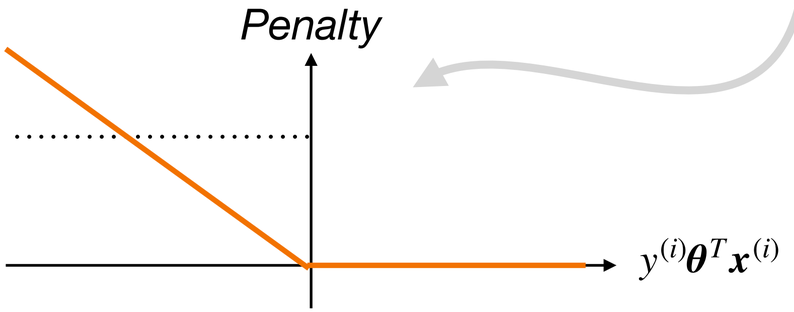
Loss function

Goal: Find an optimal solution

Algorithm : The Perceptron Learning Algorithm

$P \leftarrow$ samples with label 1
$N \leftarrow$ samples with label 0
Initialize $\boldsymbol{\theta}$ randomly
while !convergence **do**
  pick random sample $\mathbf{x}$
  if $\mathbf{x} \in P$ and $\boldsymbol{\theta}^T \mathbf{x} < 0$ **then**
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \mathbf{x}$
  end if
  if $\mathbf{x} \in N$ and $\boldsymbol{\theta}^T \mathbf{x} \geq 0$ **then**
    $\boldsymbol{\theta} = \boldsymbol{\theta} - \mathbf{x}$
  end if
end while
// the algorithm converges when all samples are classified correctly

B) The hinge loss: $\varepsilon(\boldsymbol{\theta}) = \sum_{i=1}^{m} \max(-y^{(i)} \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}, 0)$



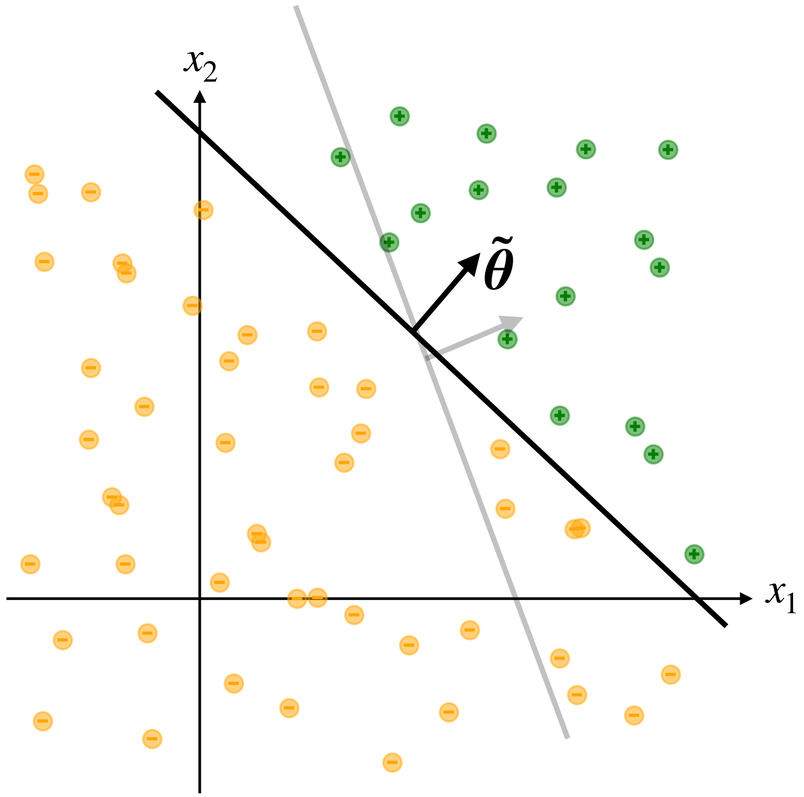
- We **can** perform gradient descent

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \lambda \frac{\partial \varepsilon(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta}^t + \lambda \sum_{i\text{: misclassified}} y^{(i)} \boldsymbol{x}^{(i)}$$

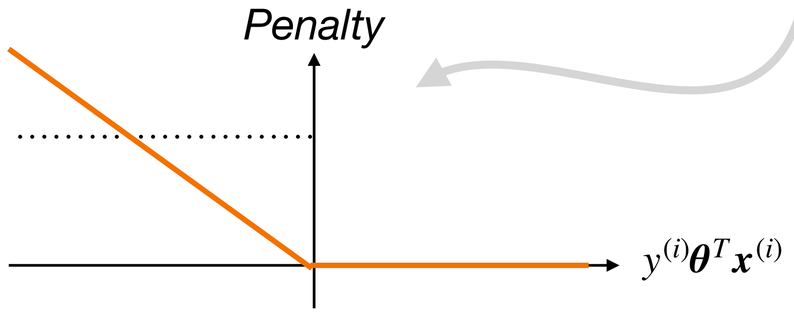
Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Loss function

Goal: Find an optimal solution



B) The hinge loss: $\varepsilon(\boldsymbol{\theta}) = \sum_{i=1}^{m} \max(-y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}, 0)$

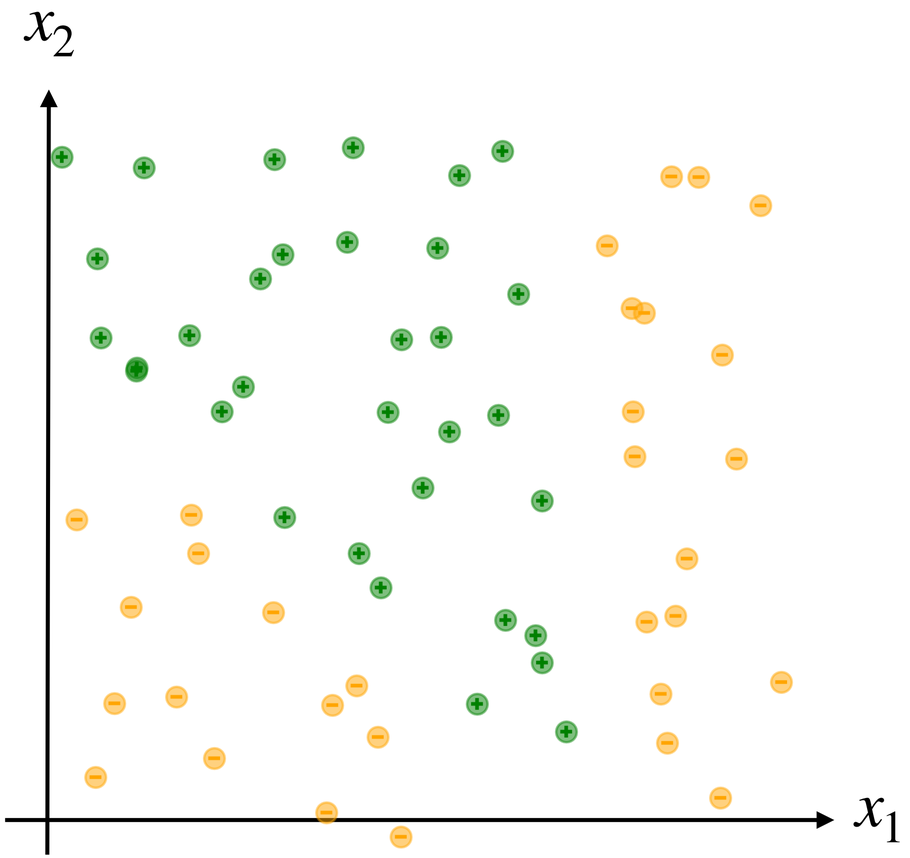


- We **can** perform gradient descent
- But: not convex, ambiguity of “optimal” solution

=> Logistic Regression :)

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

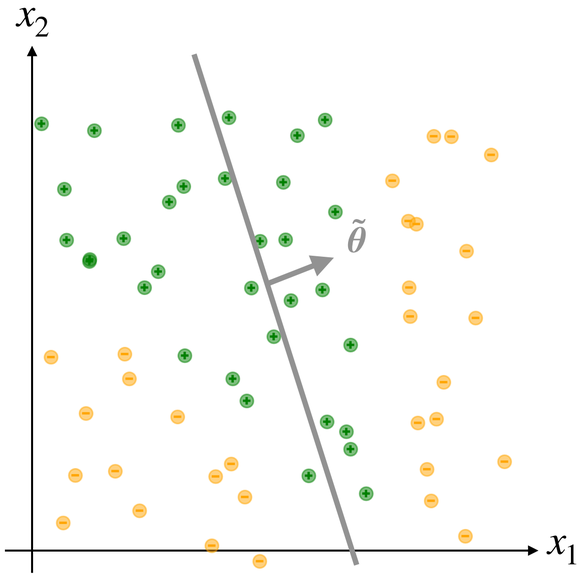
*What if the problem is **not** linearly separable?*

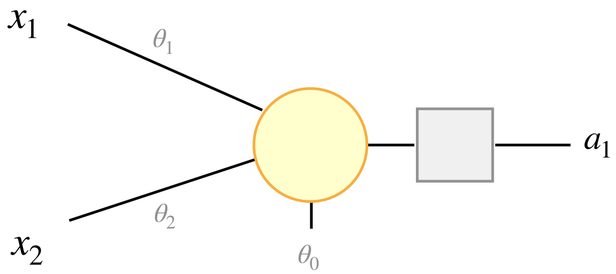


Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Multi-layer perceptrons (MLPs) / Feed-forward neural networks

Multi-Layer Perceptrons

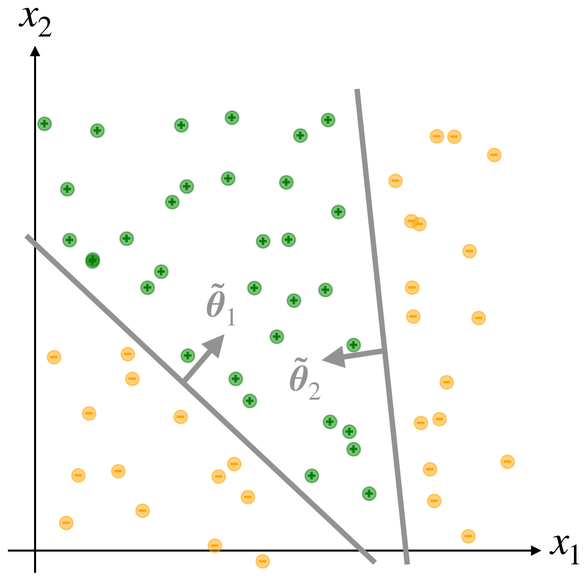


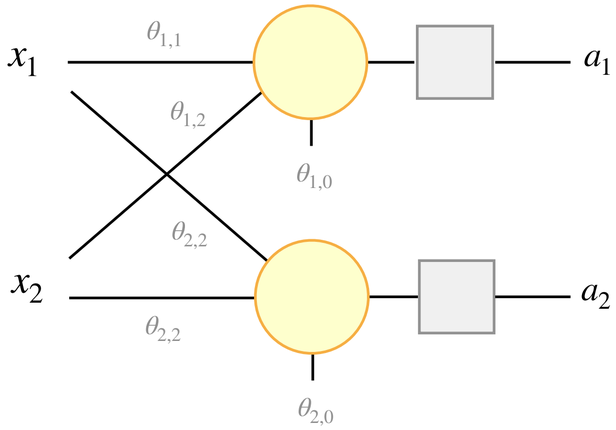


Not linearly separable

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

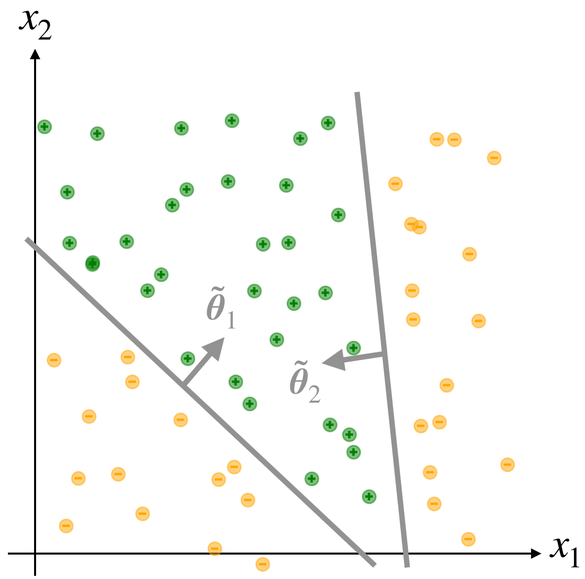
Multi-Layer Perceptrons

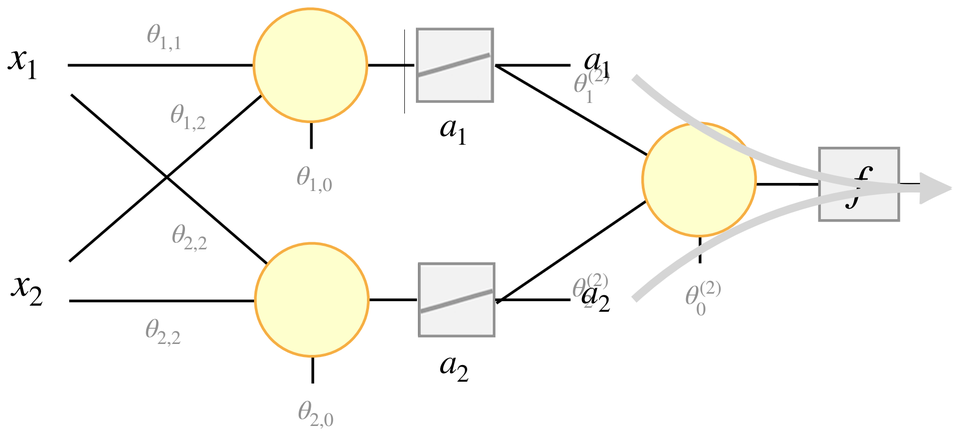


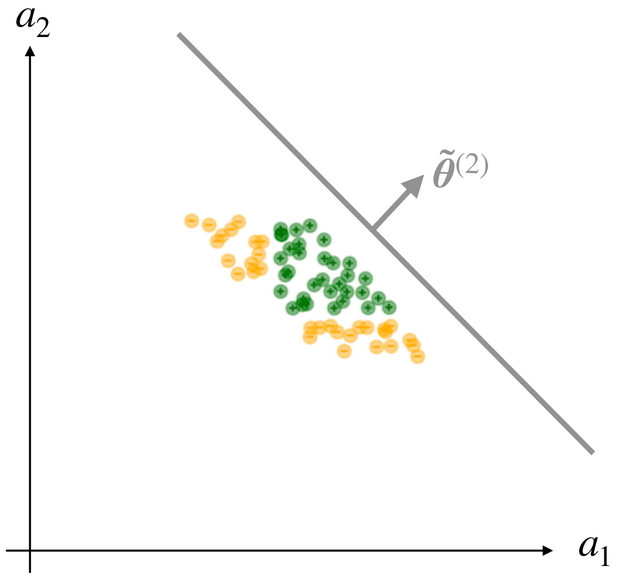


Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Multi-Layer Perceptrons







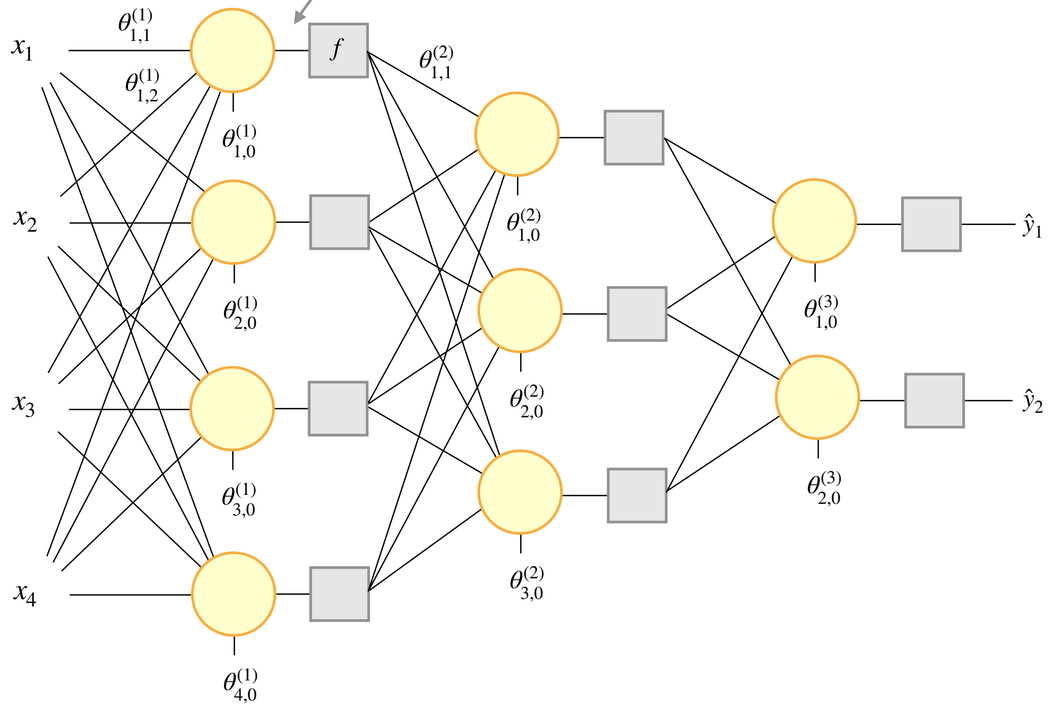
Why non-linear activation function?

NN with linear activation func ≙ Single layer NN

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Multi-Layer Perceptrons

General notation



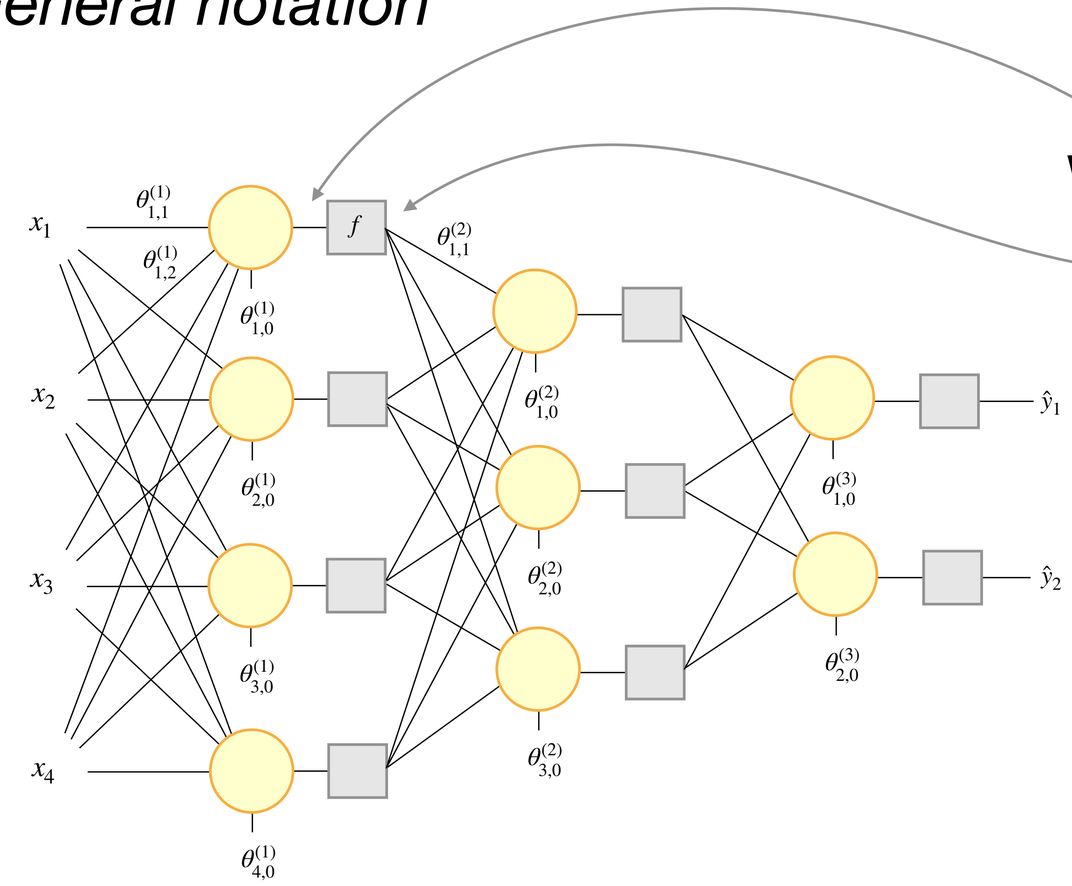
Weights to first neuron

Weighted input: $z^{(1)} = \Theta^{(1)} x = \begin{bmatrix} \theta^{(1)}_{1,0} & \theta^{(1)}_{1,1} & \dots & \theta^{(1)}_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta^{(1)}_{k,0} & \theta^{(1)}_{k,1} & \dots & \theta^{(1)}_{k,n} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Multi-Layer Perceptrons

General notation



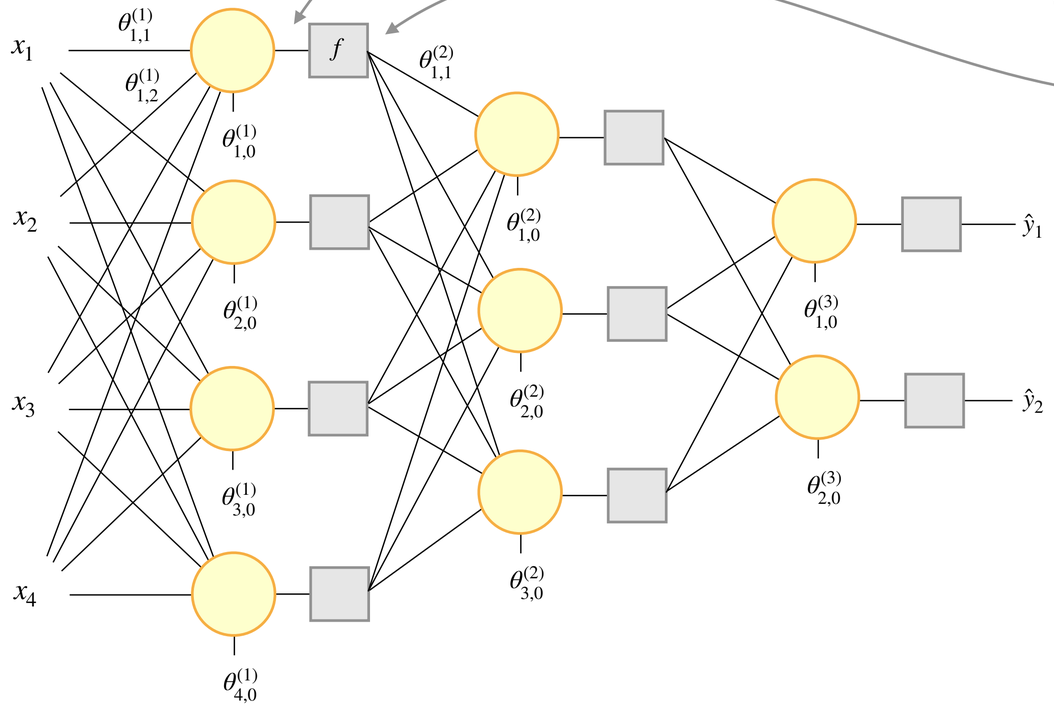
Weighted input: $z^{(1)} = \Theta^{(1)} x$

Activation: $a^{(1)} = \sigma(z^{(1)})$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Multi-Layer Perceptrons

General notation



Weighted input: $z^{(1)} = \Theta^{(1)} x$

Activation: $a^{(1)} = f^{(1)}(z^{(1)})$

General notation for layer l:

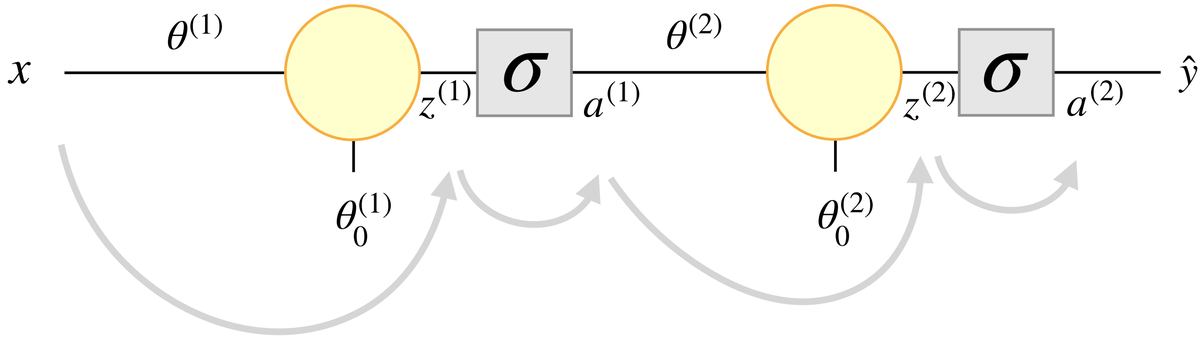
Weighted input: $z^{(l)} = \Theta^{(l)} a^{(l-1)}$

Activation: $a^{(l)} = f^{(l)}(z^{(l)})$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

The Idea of Backpropagation

for 1D Neural Network



1) Feedforward pass

$$z^{(1)} = \theta^{(1)}x + \theta_0^{(1)}$$
$$a^{(1)} = \sigma(z^{(1)})$$
$$z^{(2)} = \theta^{(2)}a^{(1)} + \theta_0^{(2)}$$
$$\hat{y} = a^{(2)} = \sigma(z^{(2)})$$

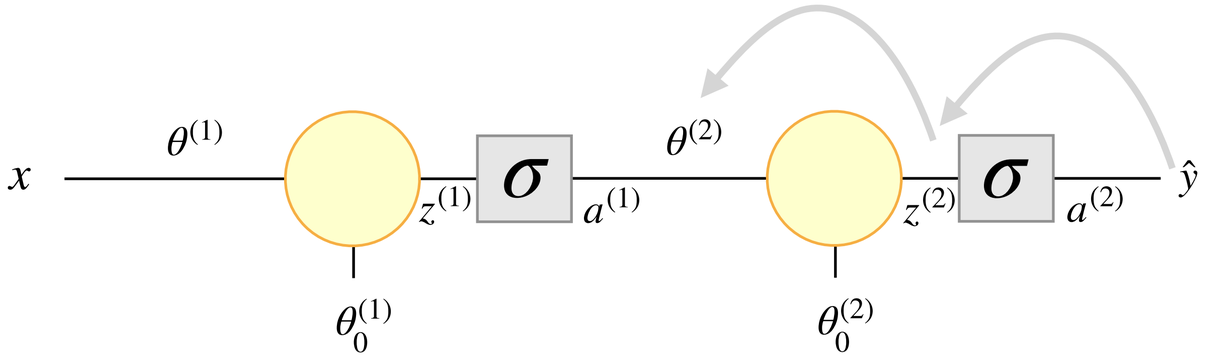
2) Compute loss

$$\varepsilon(\hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

The Idea of Backpropagation

for 1D Neural Network



1) Feedforward pass

$$z^{(1)} = \theta^{(1)}x + \theta_0^{(1)}$$
$$a^{(1)} = \sigma(z^{(1)})$$
$$z^{(2)} = \theta^{(2)}a^{(1)} + \theta_0^{(2)}$$
$$\hat{y} = a^{(2)} = \sigma(z^{(2)})$$

2) Compute loss

$$\varepsilon(\hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

3) Backpropagate loss gradient

$$\frac{\partial L}{\partial \theta^{(2)}} = ? \qquad L(\theta^{(2)}) = \epsilon \circ \sigma \circ l(\theta^{(2)})$$

$$\frac{\partial L}{\partial \theta^{(2)}} = \frac{\partial \varepsilon}{\partial a^{(2)}} \cdot \frac{\partial \sigma}{\partial z^{(2)}} \cdot \frac{\partial l}{\partial \theta^{(2)}}$$

Goal:

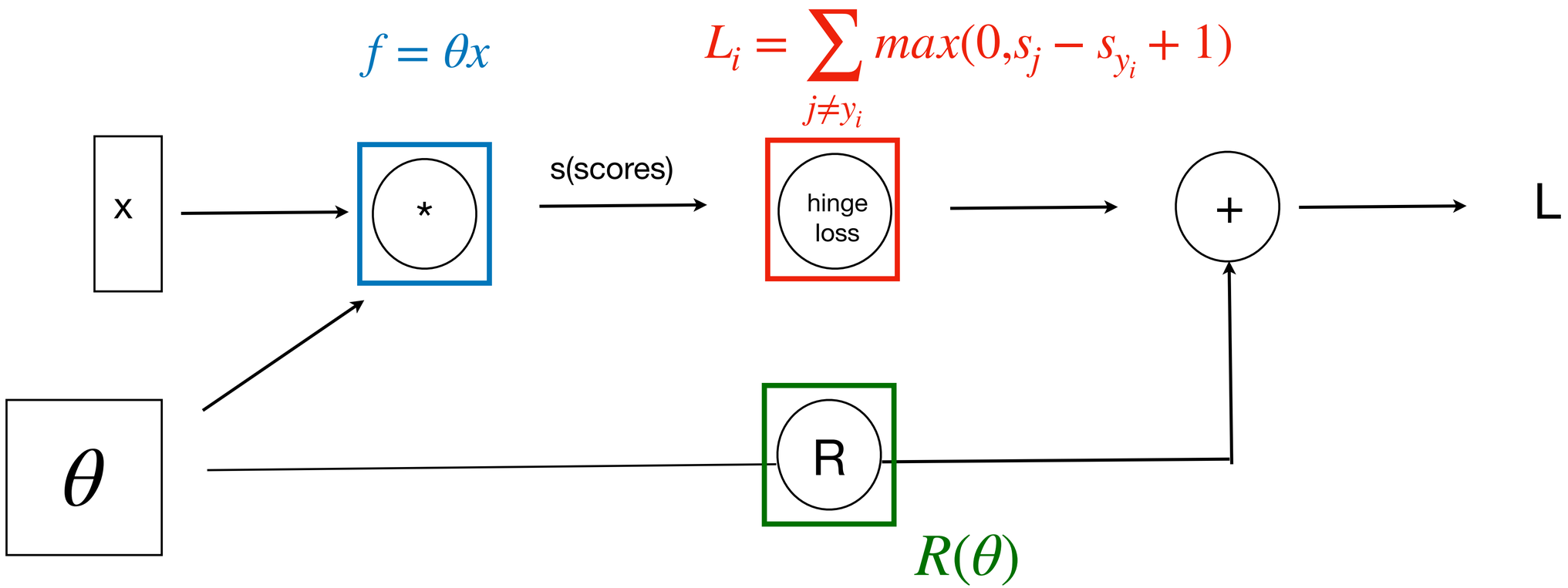
Gradient Descent

$$\theta^{t+1} = \theta^t - \lambda \frac{\partial \varepsilon(\theta)}{\partial \theta}$$

Credit: Fred Hamprecht, University of Heidelberg, Pattern Recognition Class (2012)

Backpropagation

Computational graphs



Convolutional network

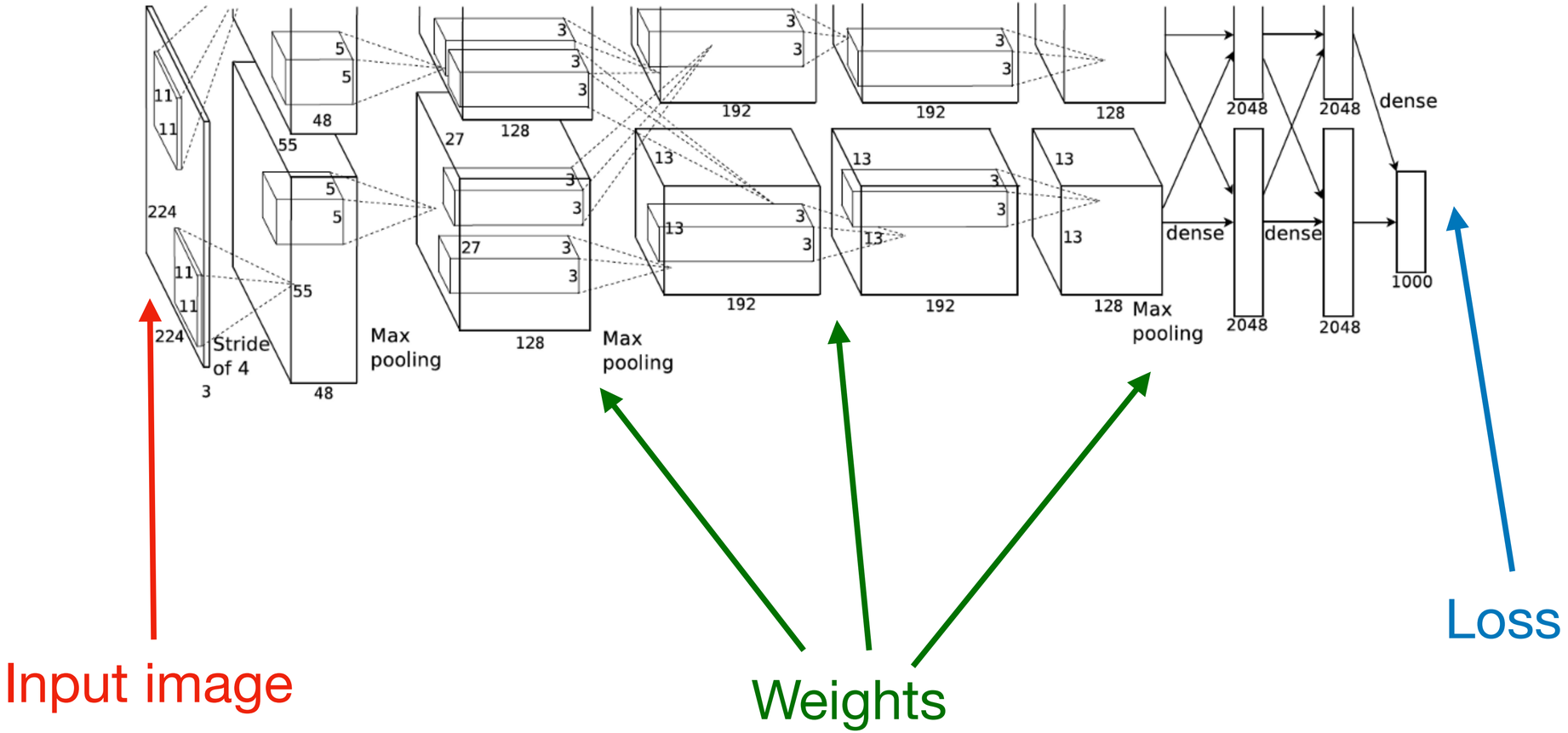


Figure copyright Alex Krizhevsky,Ilya Sutskever and Geoffrey Hinton, 2012

Credits to FeiFei Li & Justin Johnson & Serena Yeung (Stanford CNNs for Visual Recognition, 2017, cs231)

Neural Turing machine

Input image

Loss

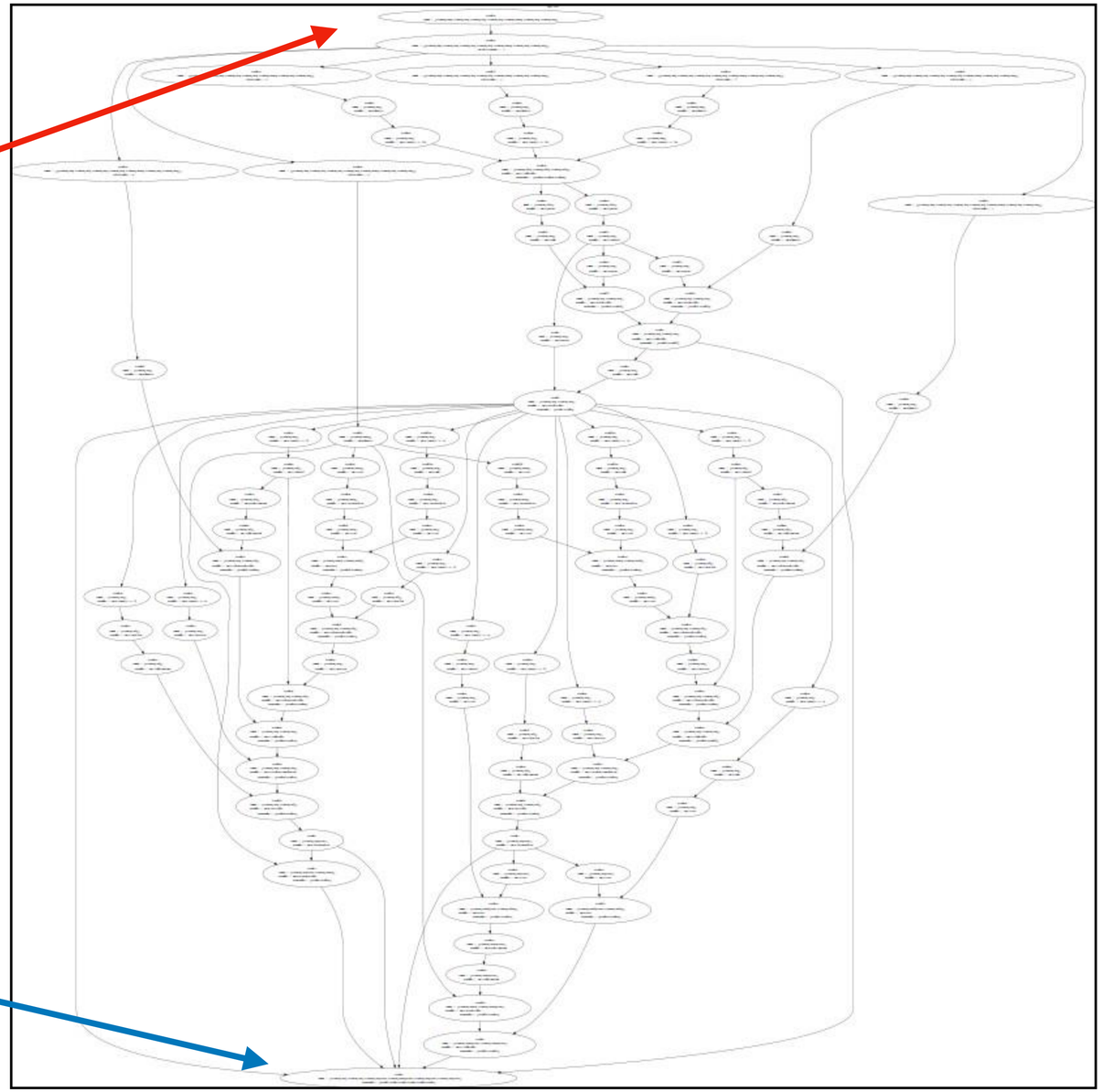


Figure: Andrej Karpathy, Twitter post

Credits to FeiFei Li & Justin Johnson & Serena Yeung (Stanford CNNs for Visual Recognition, 2017, cs231)

Neural Turing machine

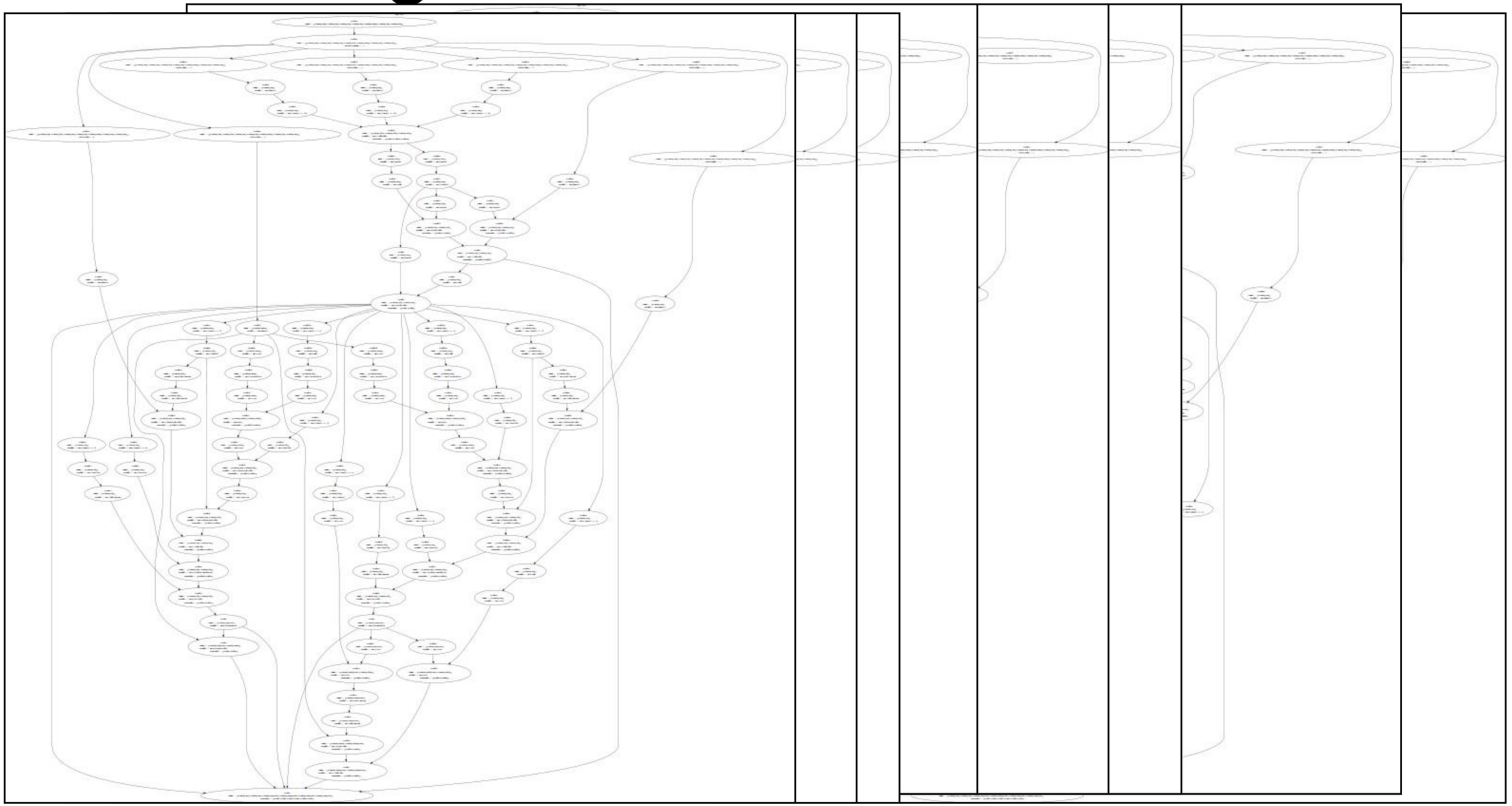


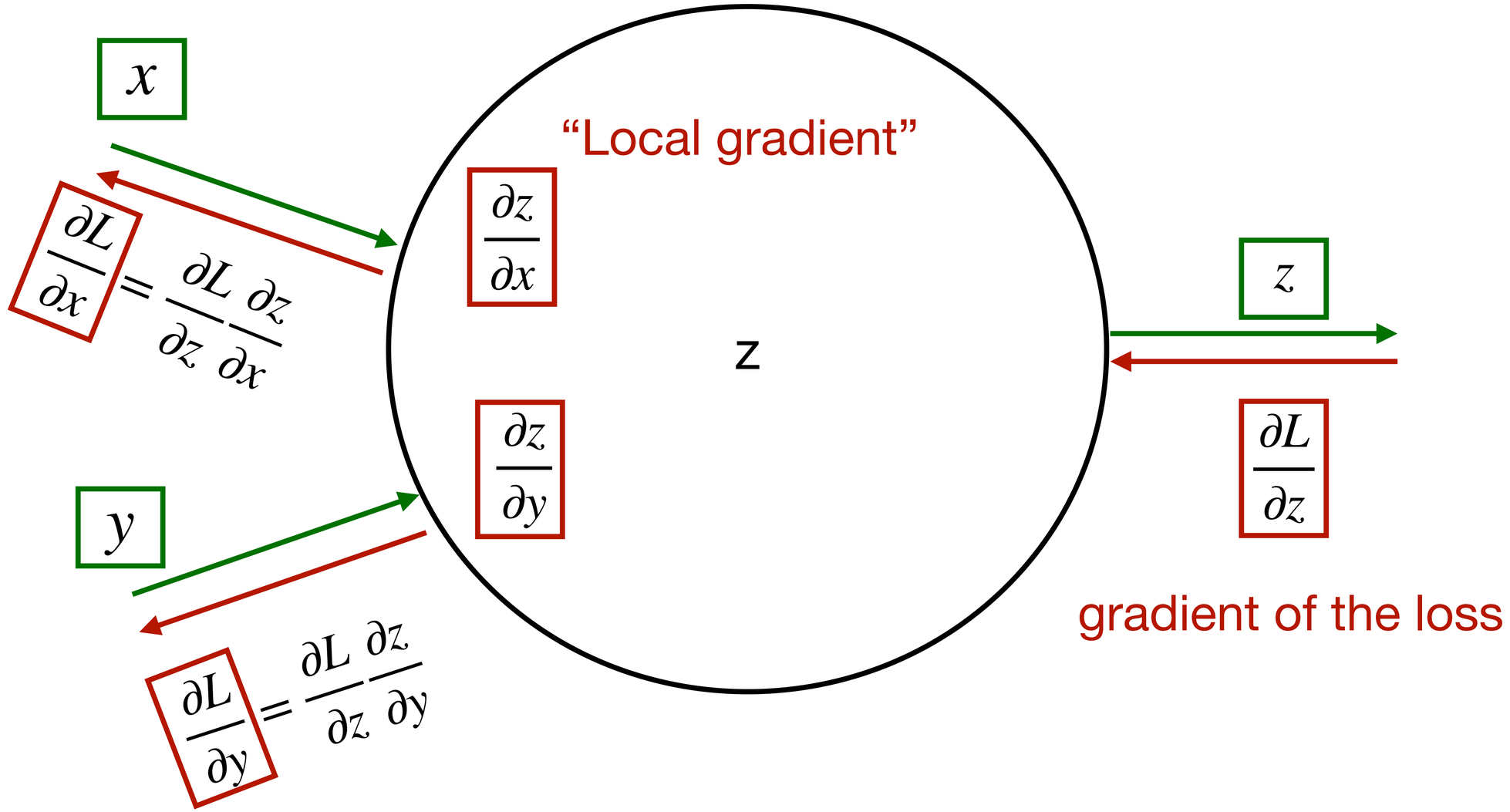
Figure: Andrej Karpathy, Twitter post

Credits to FeiFei Li & Justin Johnson & Serena Yeung (Stanford CNNs for Visual Recognition, 2017, cs231)

Backpropagation

Chain rule

$$\frac{\partial L(z(x))}{\partial x} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial x}$$



Credits to FeiFei Li & Justin Johnson & Serena Yeung (Stanford CNNs for Visual Recognition, 2017, cs231)

Backpropagation

A simple example

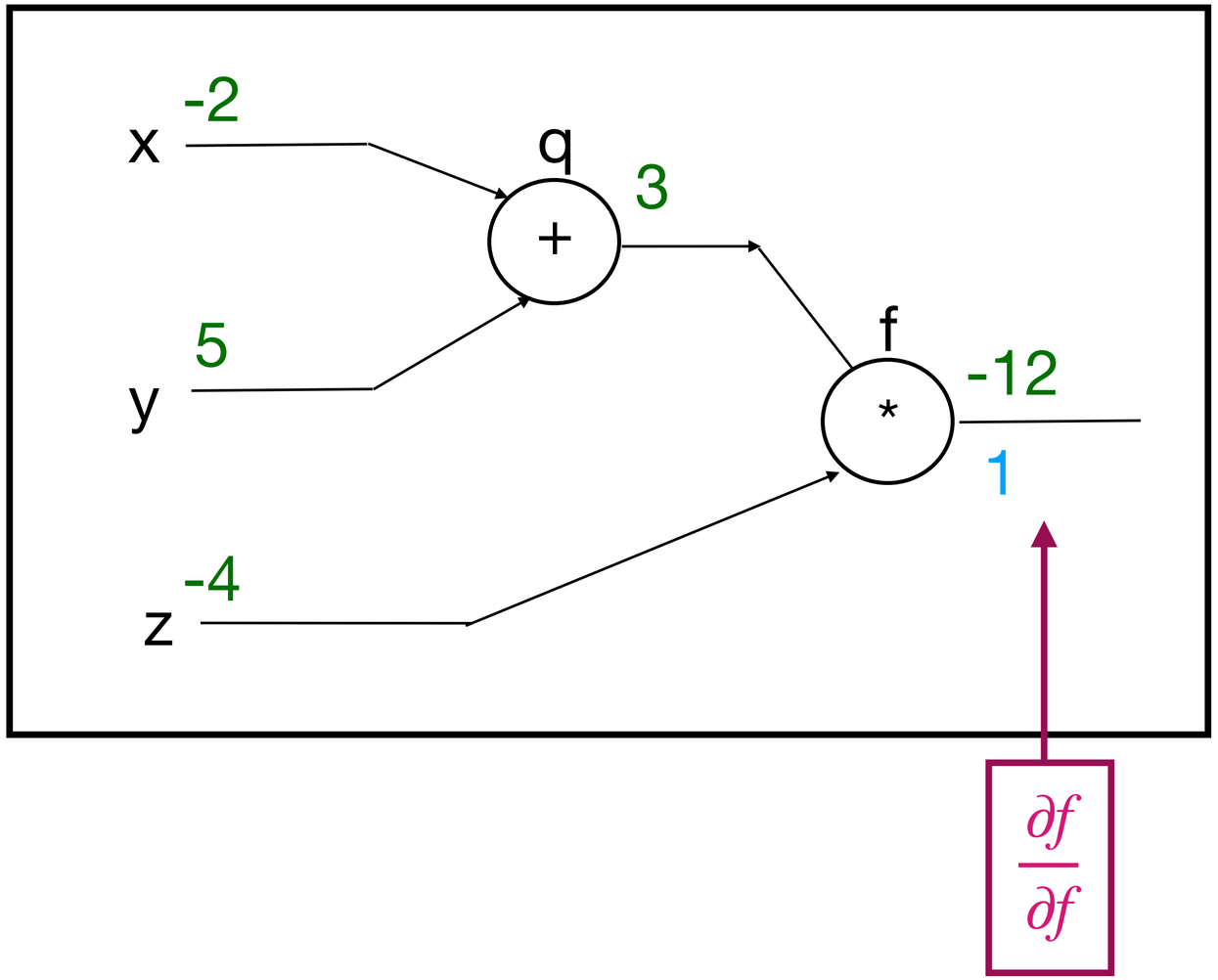
$$f(x, y, z) = (x + y)z$$

Let $x = -2, y = 5, z = -4$

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

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

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



Backpropagation

A simple example

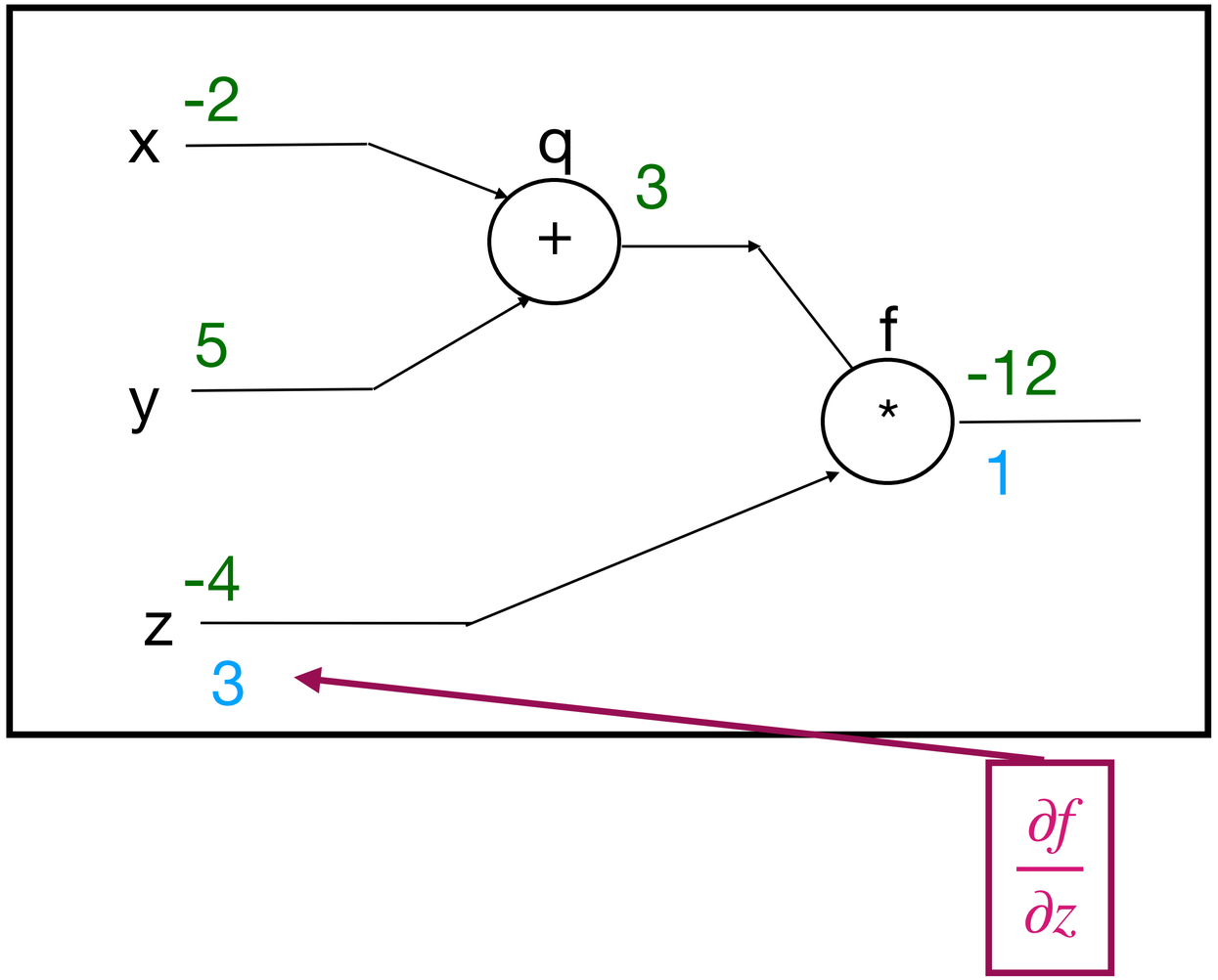
$$f(x, y, z) = (x + y)z$$

Let $x = -2, y = 5, z = -4$

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

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

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



Backpropagation

A simple example

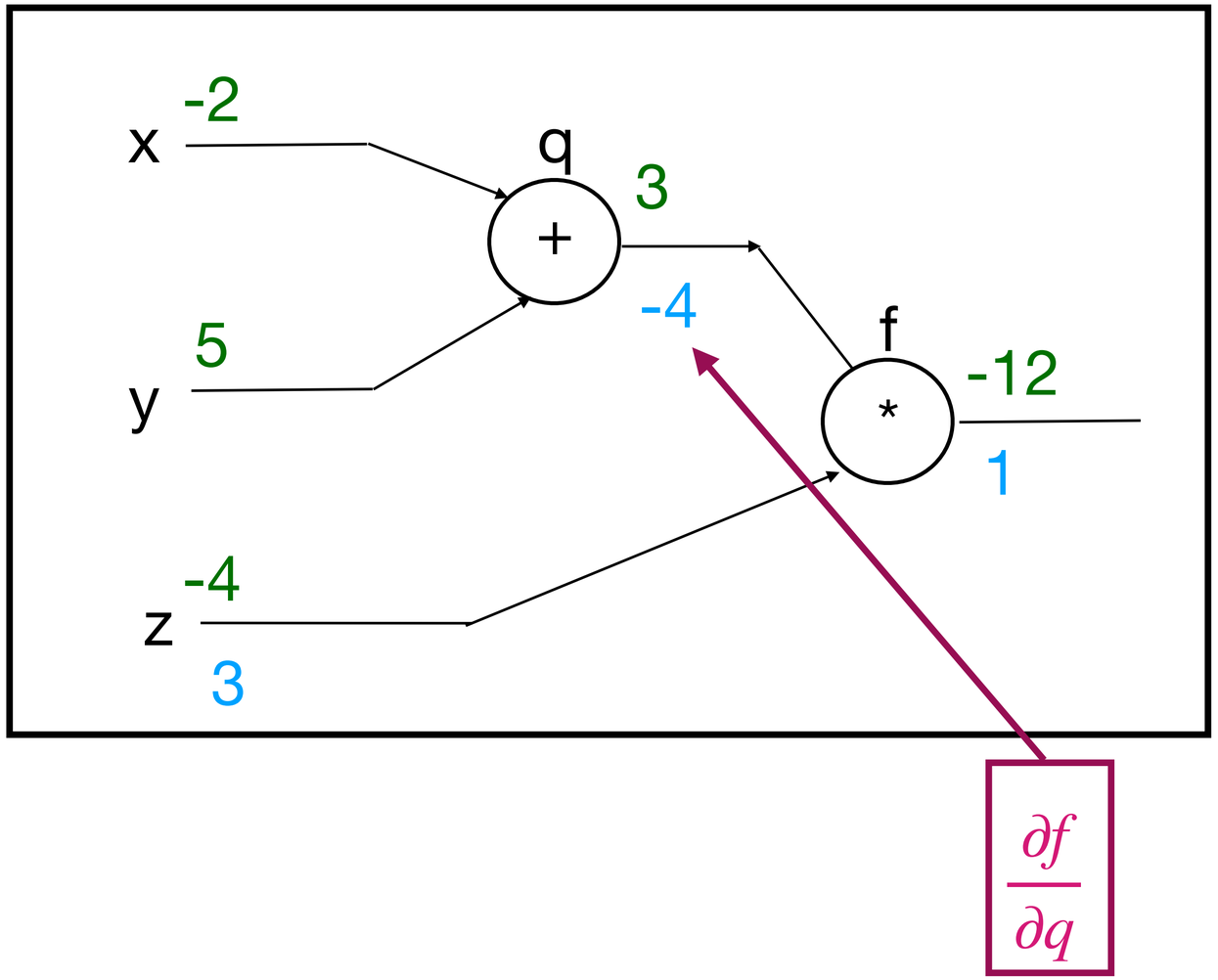
$$f(x, y, z) = (x + y)z$$

Let $x = -2, y = 5, z = -4$

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

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

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



Backpropagation

A simple example

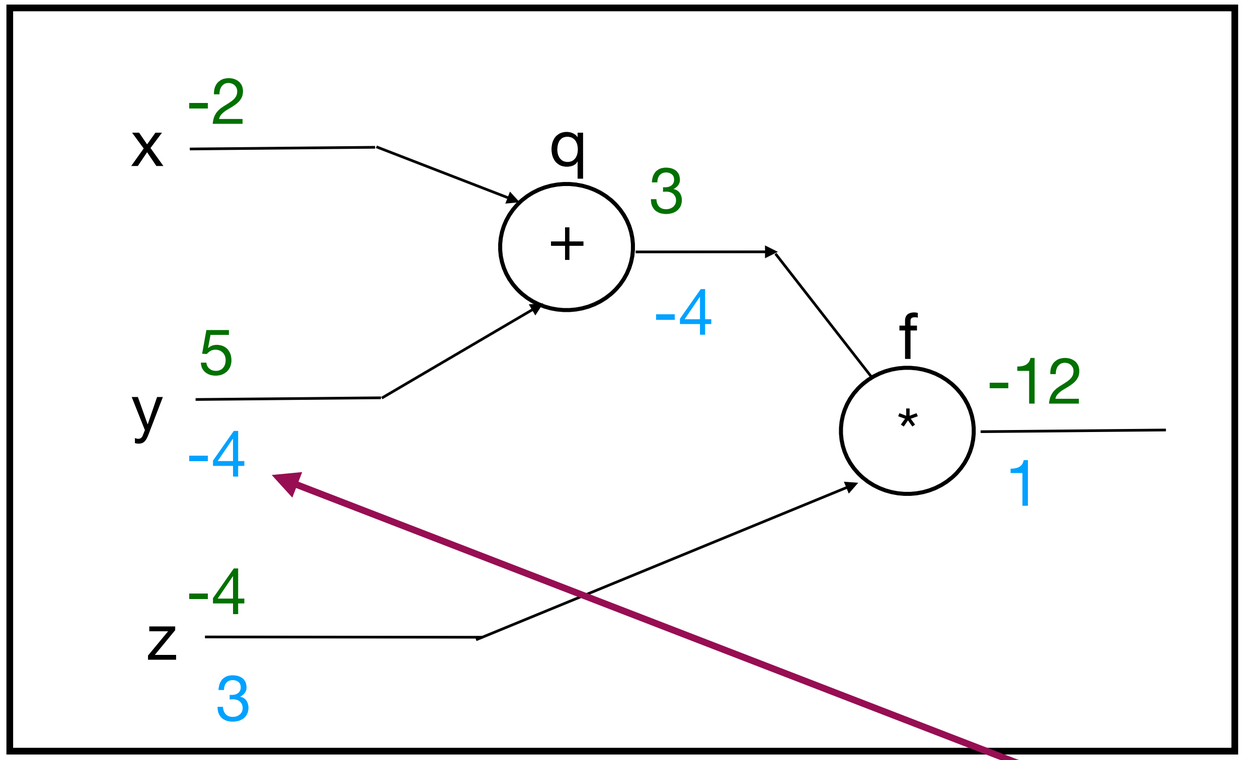
$$f(x, y, z) = (x + y)z$$

Let $x = -2, y = 5, z = -4$

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

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

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



Chain Rule:

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

Backpropagation

A simple example

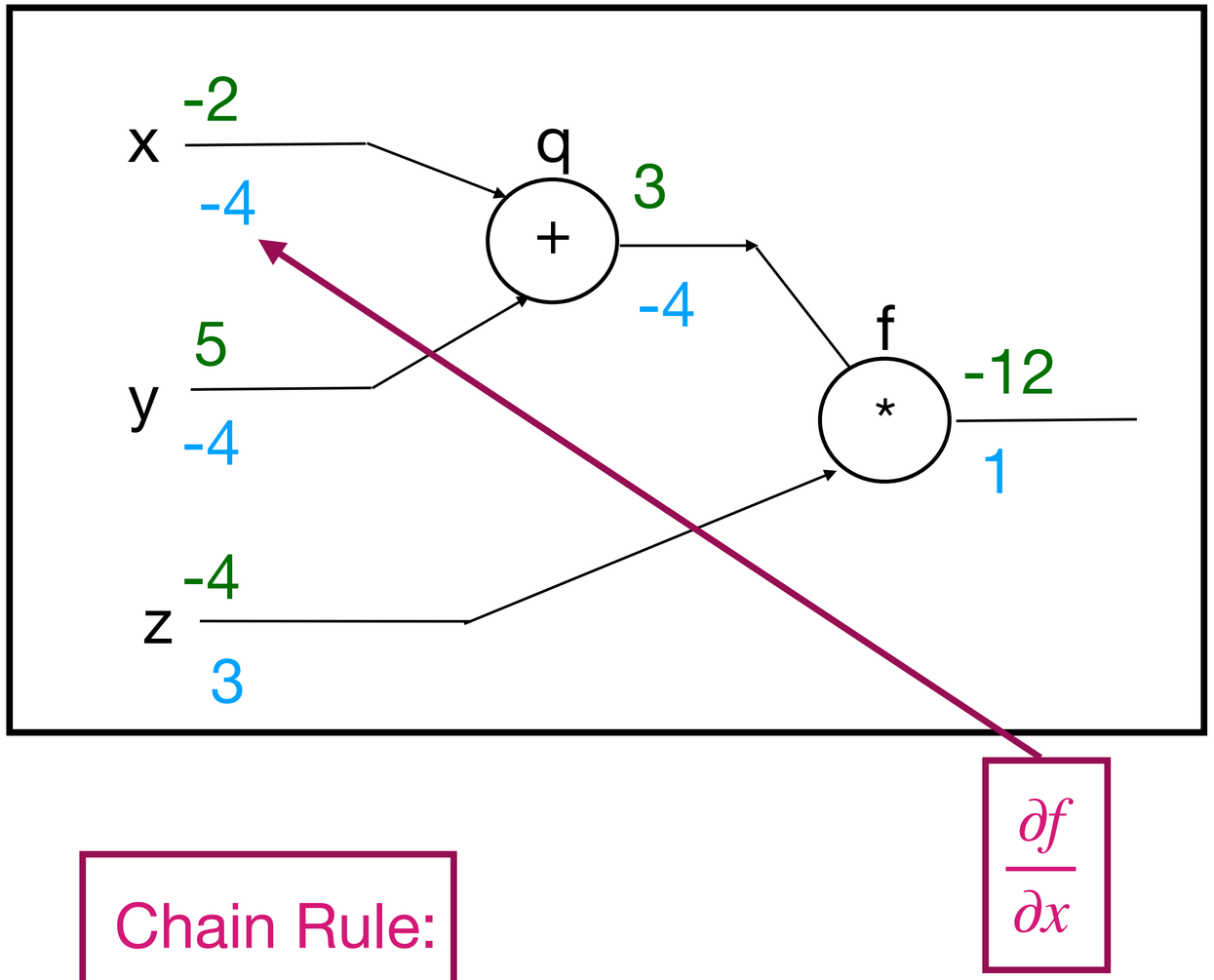
$$f(x, y, z) = (x + y)z$$

Let $x = -2, y = 5, z = -4$

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

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

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



Chain Rule:

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

Neural Networks

- Multi-layer Perceptrons
- Backpropagation

Neural Networks

- Multi-layer Perceptrons
- Backpropagation
- … on Images
- Loss functions for more than two classes
- Better activation functions
- Better gradient descent

Course Outline

- Introduction to Image Analysis
- **Basics: Neural Networks**
- **Convolutional Neural Networks**
- Transformers
- Model Interpretability
- Self-supervised Learning
- Generative Models (GANs, Diffusion)