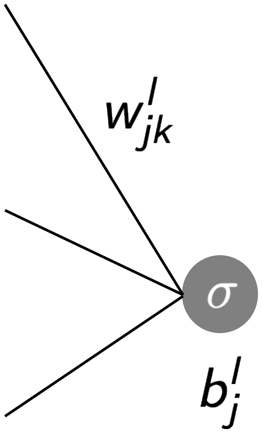
Computational MRI

Introduction to machine learning and neural networks

Friedrich-Alexander-Universität
Computational Imaging Lab

Neural networks



$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right)$$

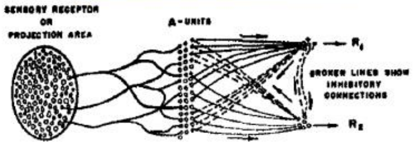


FIG. 2A. Schematic representation of connections in a simple perceptron.

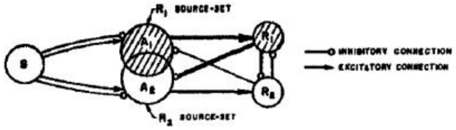
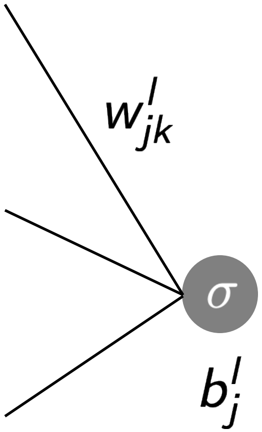


FIG. 2B. Venn diagram of the same perceptron (shading shows active sets for R_1 response).

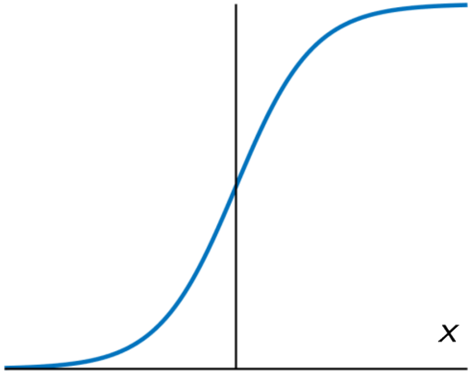
Rosenblatt Psychological review 1957

Neural networks



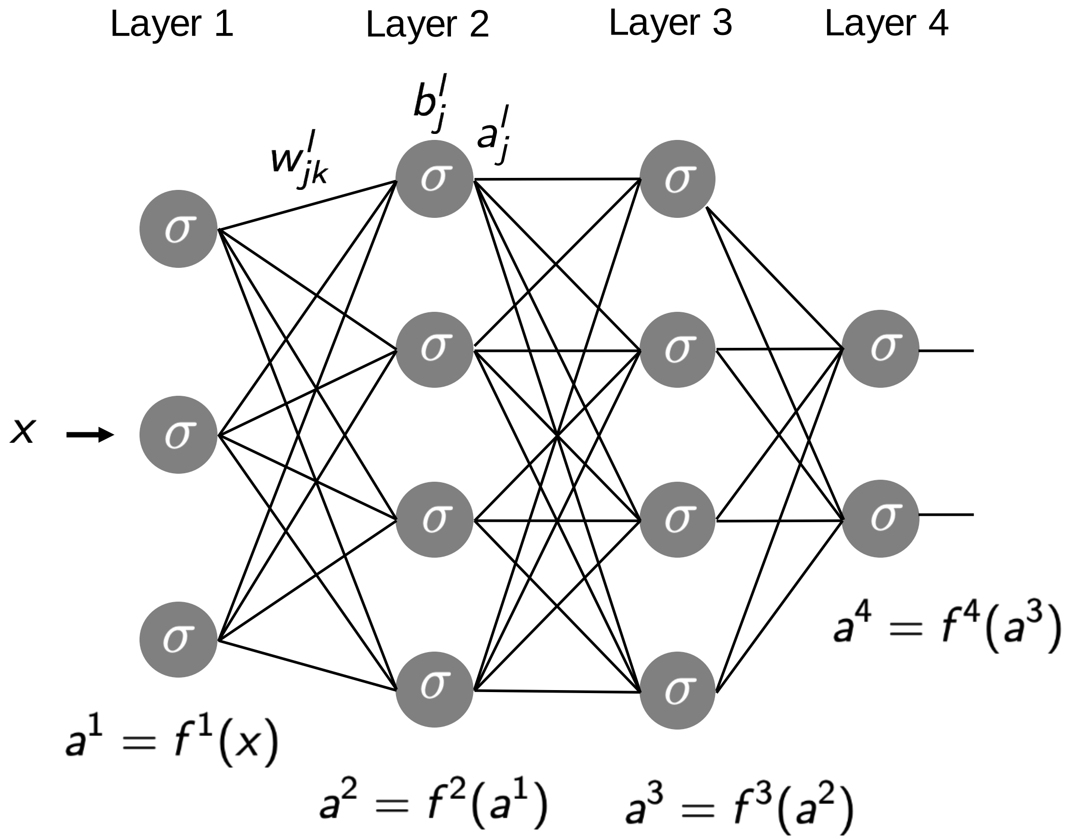
$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right)$$

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



Rosenblatt Psychological review 1957

Neural networks



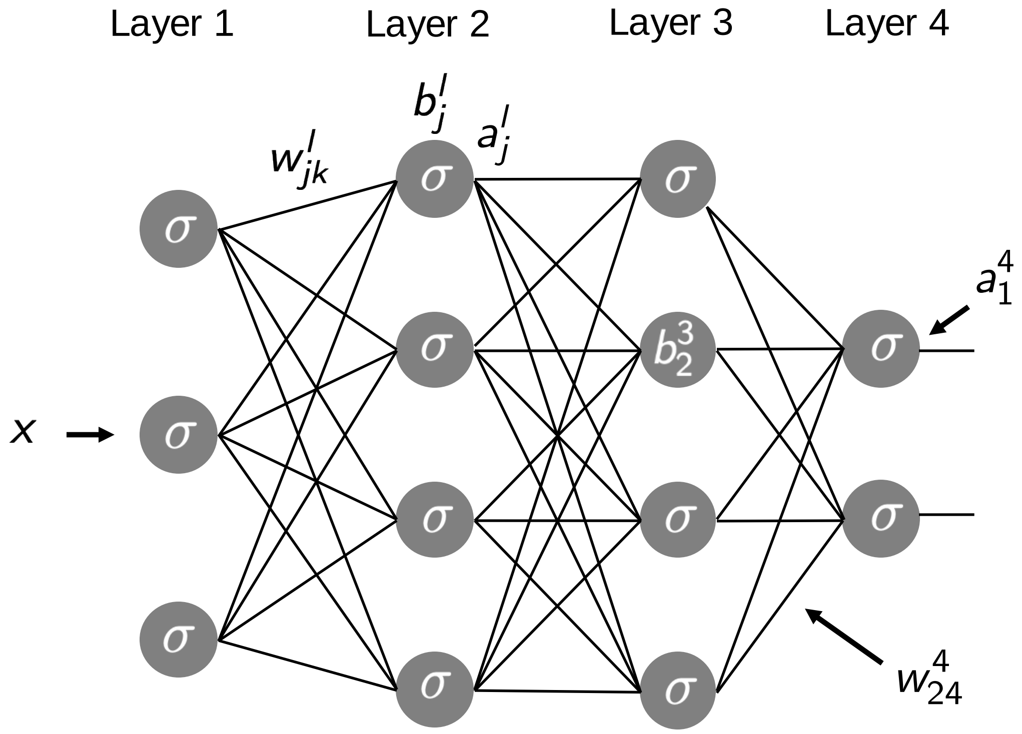
Large number of model parameters

High descriptive capacity

$$a^4 = F(x) = f^4(f^3(f^2(f^1(x))))$$

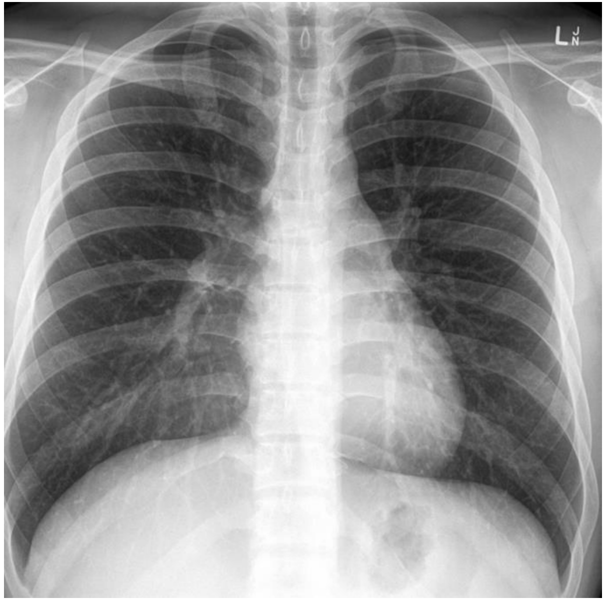
Cybenko: Math Control Sig Sys 1989

Neural networks: Notation

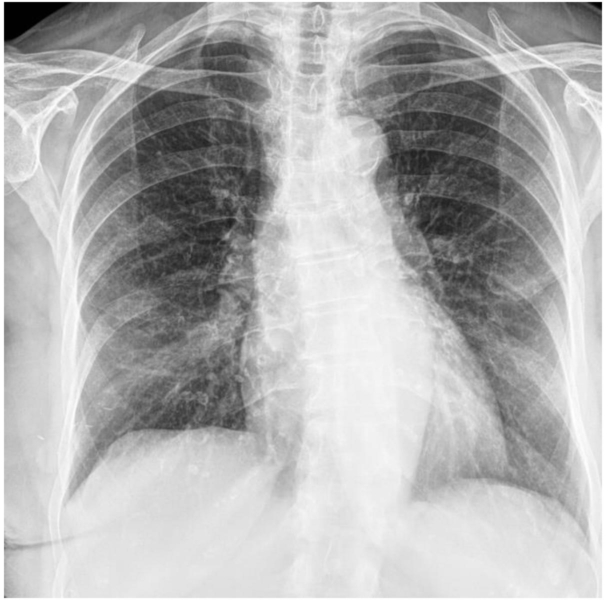


ER chest X-Ray diagnosistic classification

Normal



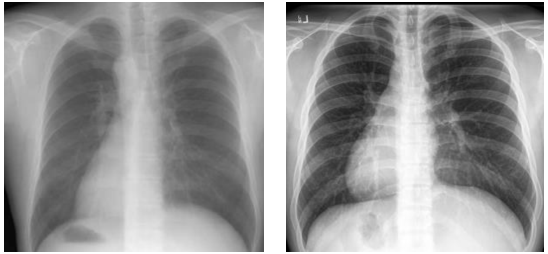
COVID-19



Radiopedia.org

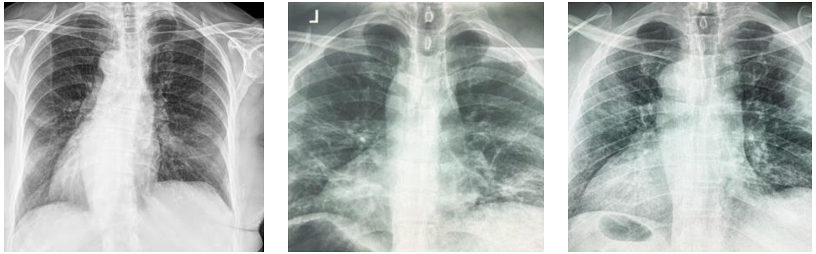
Chest X-Ray data set

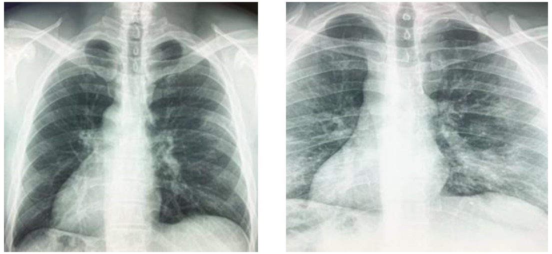
Normal



$$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

COVID-19





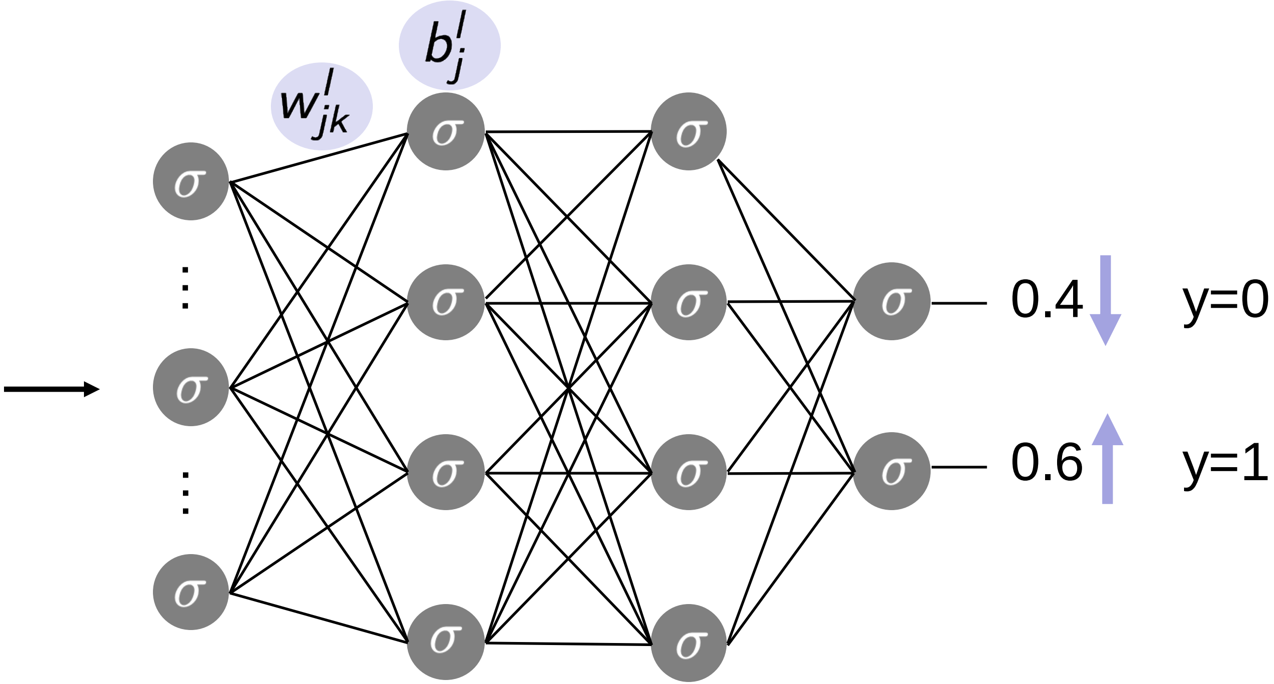
$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\{(x_1, y_1), \ldots (x_N, y_N)\}$$

Radiopedia.org
Weinstock JUCM 2020

Neural network training

COVID-19



Change weights and biases to bring output closer to target

Neural network training: Cost function

$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

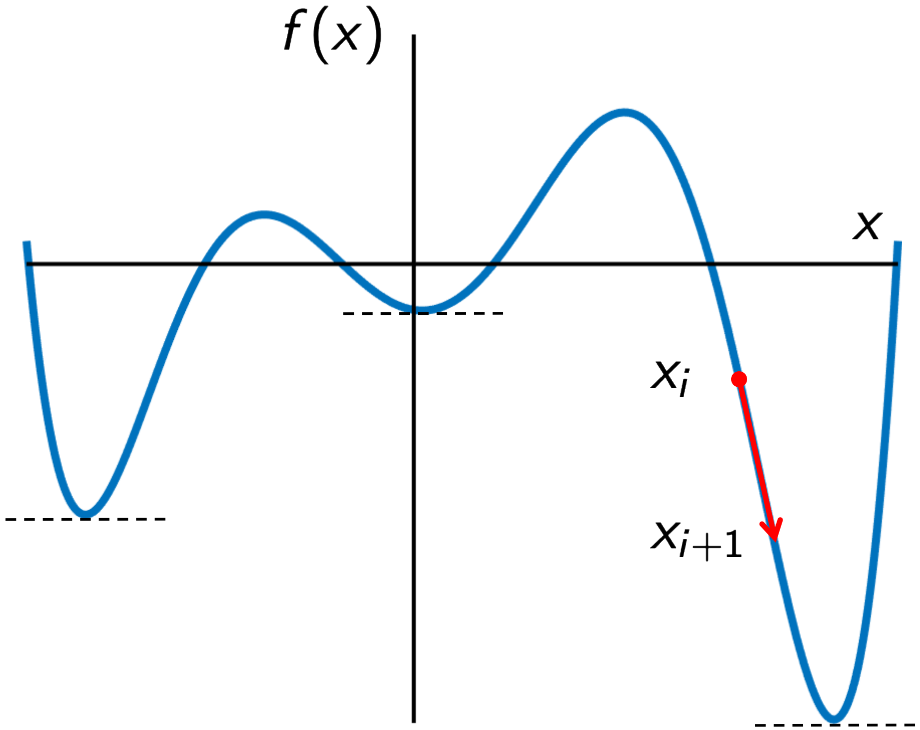
$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

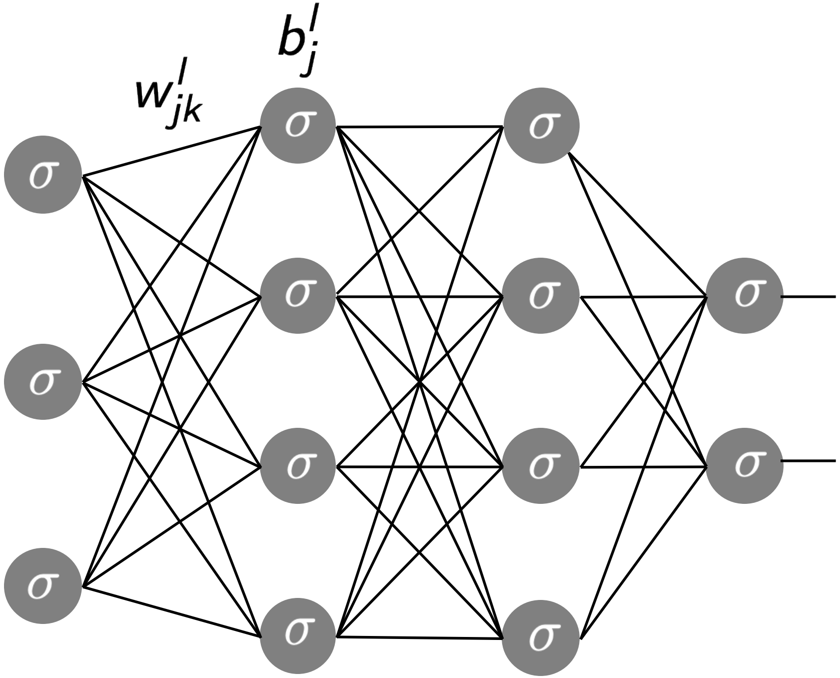
$$C(w, b) = \frac{1}{2N} \sum_{x=1}^{N} ||y_x - a_x||_2^2$$

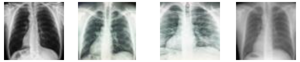
Find minimum of function: Gradient descent



$$x_{i+1} = x_i - \alpha \frac{\partial f(x_i)}{\partial x_i}$$

Neural network training: Gradient descent



$$\min_{w,b} C(w, b)$$  $\cdots$

$$\tilde{w}_{jk}^l = w_{jk}^l - \alpha \frac{\partial C}{\partial w_{jk}^l}$$

$$\tilde{b}_j^l = b_j^l - \alpha \frac{\partial C}{\partial b_j^l}$$

$\longrightarrow$ Chain rule

Backpropagation

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

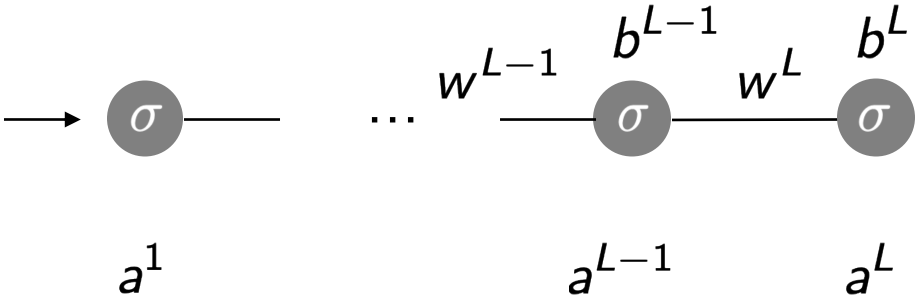
Rumelhart (1986)

$$\frac{\partial C}{\partial w^l_{jk}} \qquad \frac{\partial C}{\partial b^l_j}$$

→ Efficient recursion to calculate these partial derivatives

Backpropagation: Forward pass

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

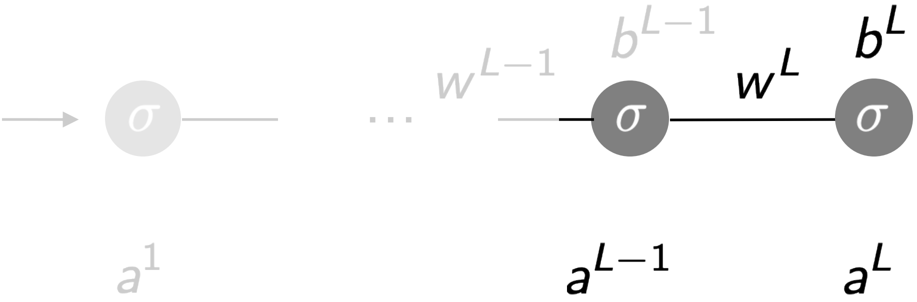


$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Backward pass

$$a^L = \sigma \underbrace{\left(w^L a^{L-1} + b^L\right)}_{z^L}$$

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Backward pass

$$
\begin{tikzcd}
w^L \arrow[dr, dash] & a^{L-1} \arrow[d, dash] & b^L \arrow[dl, dash] \\
& z^L \arrow[d, dash] & \\
& a^L \arrow[d, dash] & \\
& C_0 &
\end{tikzcd}
$$

$$z^L = w^L a^{L-1} + b^L$$

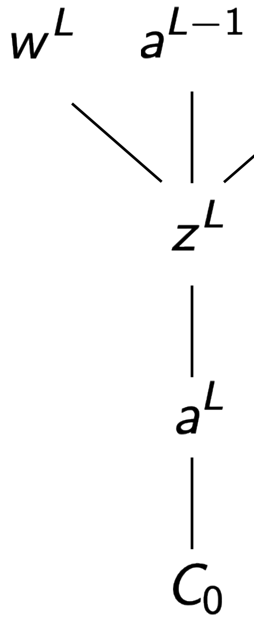
$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Partial derivatives

$$\frac{\partial C_0}{\partial w^L} = ?$$

$$\frac{\partial C_0}{\partial b^L} = ?$$



$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Weights

$$\frac{\partial C_0}{\partial w^L}$$

$$\begin{tikzcd}[arrows=dash]
w^L \arrow[dr] & a^{L-1} \arrow[d] & b^L \arrow[dl] \\
& z^L \arrow[d] & \\
& a^L \arrow[d] & \\
& C_0 &
\end{tikzcd}$$

$$z^L = w^L a^{L-1} + b^L$$

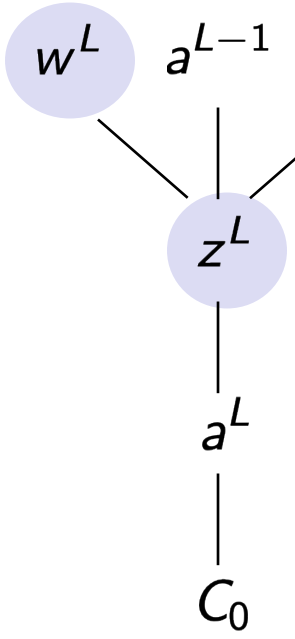
$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Weights

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial z^L}{\partial w^L} = a^{L-1}$$



$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

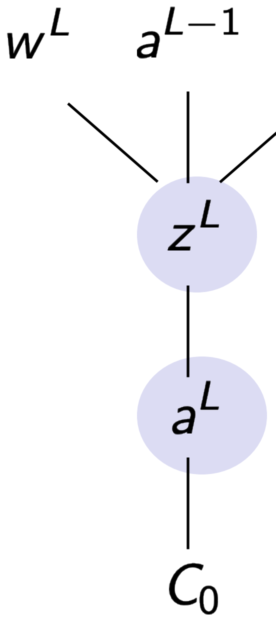
$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Weights

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}$$

$$\frac{\partial z^L}{\partial w^L} = a^{L-1}$$

$$\frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$



$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

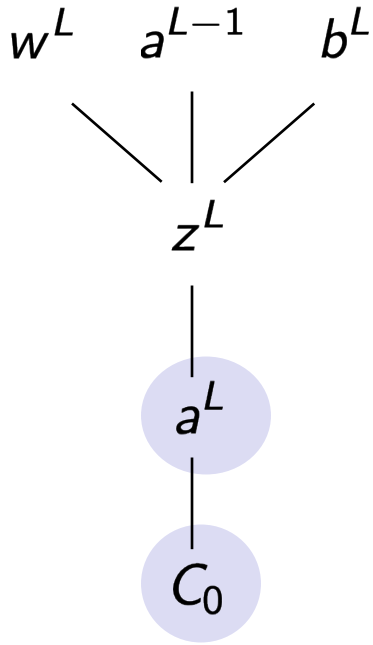
Backpropagation: Weights

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L}$$

$$\frac{\partial z^L}{\partial w^L} = a^{L-1}$$

$$\frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$

$$\frac{\partial C_0}{\partial a^L} = a^L - y$$



$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Weights

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = a^{L-1}\sigma'(z^L)(a^L - y)$$

$$\frac{\partial z^L}{\partial w^L} = a^{L-1}$$

$$\frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$

$$\frac{\partial C_0}{\partial a^L} = a^L - y$$

$$\begin{tikzcd} w^L \arrow[dr, dash] & a^{L-1} \arrow[d, dash] & b^L \arrow[dl, dash] \\ & z^L \arrow[d, dash] & \\ & a^L \arrow[d, dash] & \\ & C_0 & \end{tikzcd}$$

$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Bias

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = \sigma'(z^L)(a^L - y)$$

$w^L \quad a^{L-1} \quad b^L$

z^L

$$z^L = w^L a^{L-1} + b^L$$

$$\frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$

a^L

$$a^L = \sigma\left(z^L\right)$$

$$\frac{\partial C_0}{\partial a^L} = a^L - y$$

C_0

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Bias

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial z^L}{\partial b^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = \sigma'(z^L)(a^L - y)$$

$$\begin{tikzcd}
w^L \arrow[dr, dash] & a^{L-1} \arrow[d, dash] & b^L \arrow[dl, dash] \\
& z^L \arrow[d, dash] & \\
& a^L \arrow[d, dash] & \\
& C_0 &
\end{tikzcd}$$

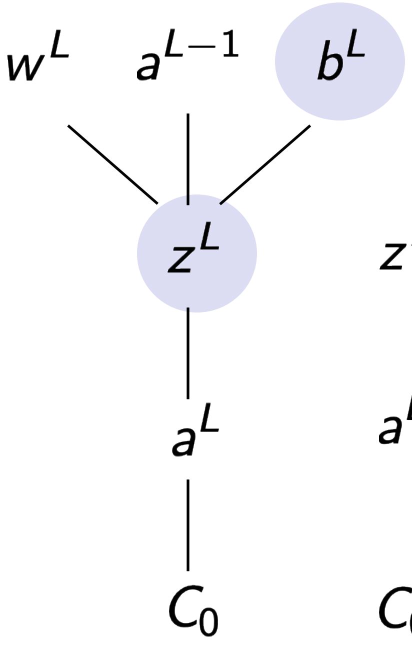
$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Bias

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial z^L}{\partial b^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = 1\sigma'(z^L)(a^L - y)$$



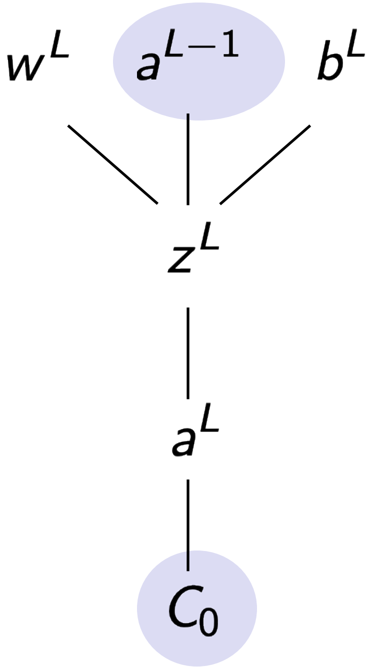
$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Activation of previous layer

$$\frac{\partial C_0}{\partial a^{L-1}} = \frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = \sigma'(z^L)(a^L - y)$$



$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Activation of previous layer

$$\frac{\partial C_0}{\partial a^{L-1}} = \frac{\partial z^L}{\partial a^{L-1}} \frac{\partial a^L}{\partial z^L} \frac{\partial C_0}{\partial a^L} = \qquad \sigma'(z^L)(a^L - y)$$

$$\begin{tikzcd}
w^L \arrow[dr, dash] & a^{L-1} \arrow[d, dash] & b^L \arrow[dl, dash] \\
& z^L \arrow[d, dash] & \\
& a^L \arrow[d, dash] & \\
& C_0 &
\end{tikzcd}$$

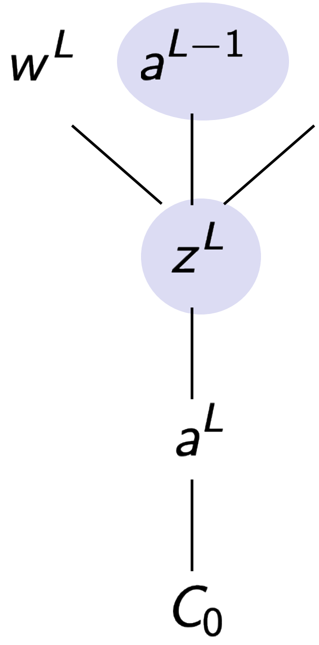
$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

Backpropagation: Activation of previous layer

$$\frac{\partial C_0}{\partial a^{L-1}} = \frac{\partial z^L}{\partial a^{L-1}} \frac{\partial a^L}{\partial z^L} \frac{\partial C_0}{\partial a^L} = w^L \sigma'(z^L)(a^L - y)$$



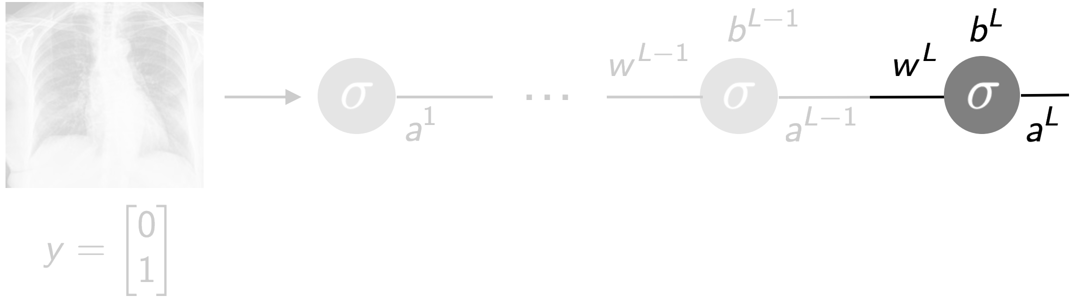
$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

$$C_0 = \frac{1}{2}(y - a^L)^2$$

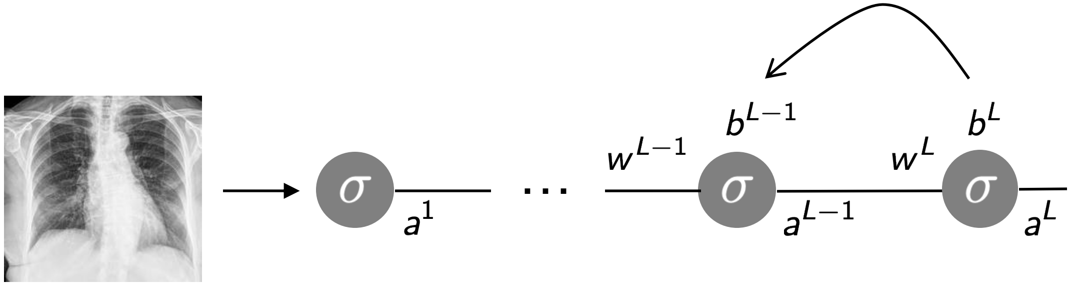
Backpropagation

$$\frac{\partial C_0}{\partial w^L} = a^{L-1}\sigma'(z^L)\frac{\partial C_0}{\partial a^L} \qquad \frac{\partial C_0}{\partial b^L} = \sigma'(z^L)\frac{\partial C_0}{\partial a^L} \qquad \frac{\partial C_0}{\partial a^L} = a^L - y$$



Backpropagation

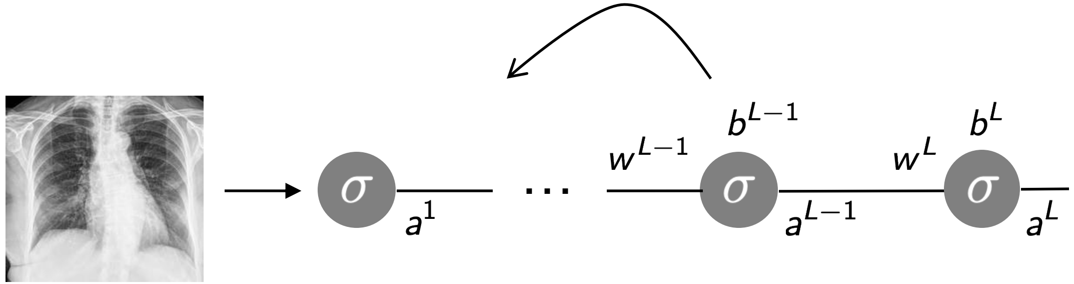
$$\frac{\partial C_0}{\partial w^{L-1}} = a^{L-2}\sigma'(z^{L-1})\frac{\partial C_0}{\partial a^{L-1}} \qquad \frac{\partial C_0}{\partial b^{L-1}} = \sigma'(z^{L-1})\frac{\partial C_0}{\partial a^{L-1}} \qquad \frac{\partial C_0}{\partial a^{L-1}} = w^L\sigma'(z^L)\frac{\partial C_0}{\partial a^L}$$



$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Backpropagation

$$\frac{\partial C_0}{\partial w^l} = a^{l-1}\sigma'(z^l)\frac{\partial C_0}{\partial a^l} \qquad \frac{\partial C_0}{\partial b^l} = \sigma'(z^l)\frac{\partial C_0}{\partial a^l} \qquad \frac{\partial C_0}{\partial a^l} = w^{l+1}\sigma'(z^{l+1})\frac{\partial C_0}{\partial a^{l+1}}$$

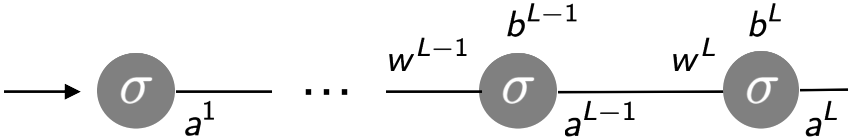


$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

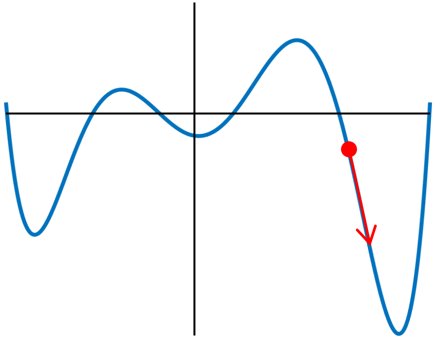
Plug partial derivatives into gradient descent

$$\frac{\partial C_0}{\partial w^l} = a^{l-1}\sigma'(z^l)\frac{\partial C_0}{\partial a^l} \qquad \frac{\partial C_0}{\partial b^l} = \sigma'(z^l)\frac{\partial C_0}{\partial a^l} \qquad \frac{\partial C_0}{\partial a^l} = w^{l+1}\sigma'(z^{l+1})\frac{\partial C_0}{\partial a^{l+1}}$$

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



$$\min_{w,b} C(w, b)$$



Gradient descent

Loop over training examples

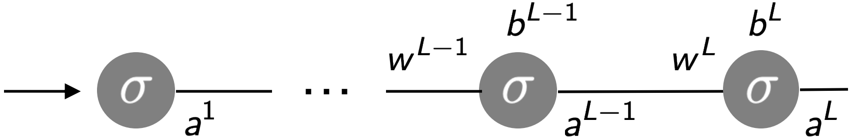
$$\frac{\partial C_0}{\partial w^l} = a^{l-1}\sigma'(z^l)\frac{\partial C_0}{\partial a^l}$$

$$\frac{\partial C_0}{\partial b^l} = \sigma'(z^l)\frac{\partial C_0}{\partial a^l}$$

$$\frac{\partial C_0}{\partial a^l} = w^{l+1}\sigma'(z^{l+1})\frac{\partial C_0}{\partial a^{l+1}}$$

$$\min_{w,b} C(w, b)$$

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

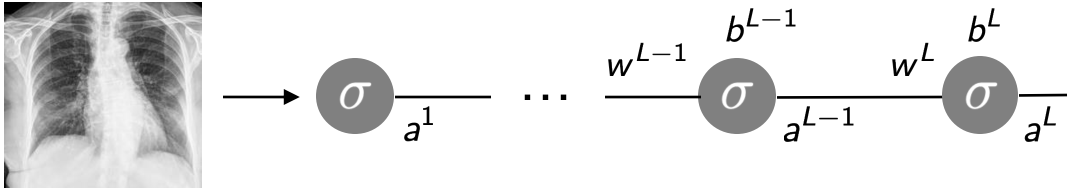
$\cdots$

$$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Backpropagation: Efficiency and insights

$$\frac{\partial C_0}{\partial w^l} = a^{l-1}\sigma'(z^l)\frac{\partial C_0}{\partial a^l} \qquad \frac{\partial C_0}{\partial b^l} = \sigma'(z^l)\frac{\partial C_0}{\partial a^l} \qquad \frac{\partial C_0}{\partial a^l} = w^{l+1}\sigma'(z^{l+1})\frac{\partial C_0}{\partial a^{l+1}}$$

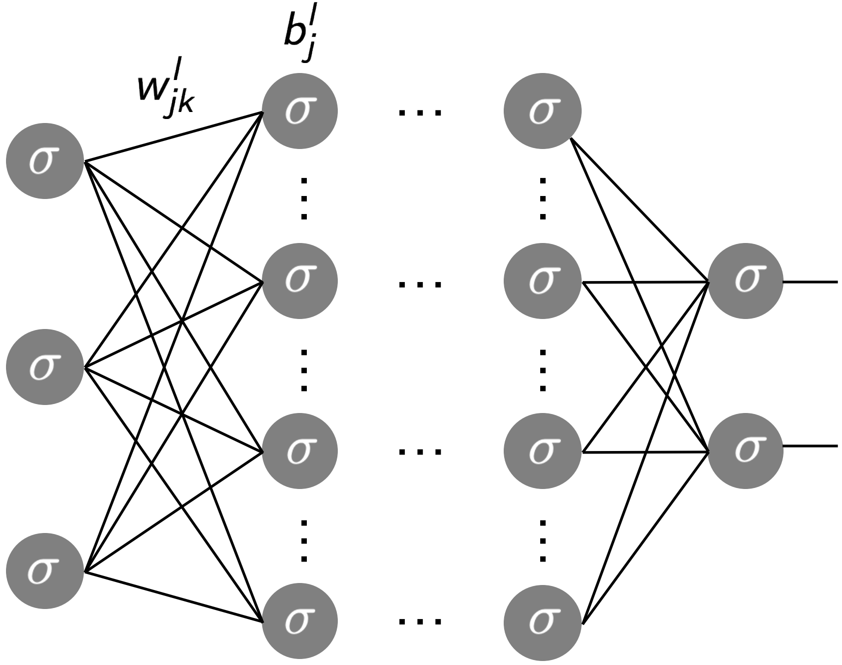


$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Each computation involves just two layers
- Avoid recomputing identical expressions in the chain rule
- Gradients provide insight into what determines speed of learning

Backpropagation: General formulation

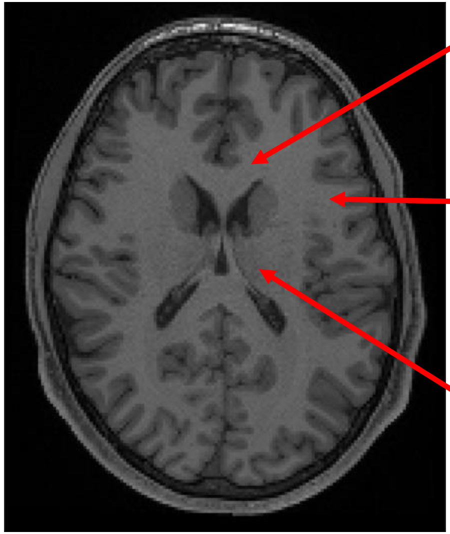
$$\frac{\partial C_0}{\partial w_{jk}^l} = a_k^{l-1}\sigma'(z_j^l)\frac{\partial C_0}{\partial a_j^l} \qquad \frac{\partial C_0}{\partial b_j^l} = \sigma'(z_j^l)\frac{\partial C_0}{\partial a_j^l} \qquad \frac{\partial C_0}{\partial a_j^l} = \sum_j w_{jk}^{l+1}\sigma'(z_j^{l+1})\frac{\partial C_0}{\partial a_j^{l+1}}$$



Exercise example 1:
Classification of brain tissue from DTI data

Classification of brain tissue from HCP DTI data

T1w

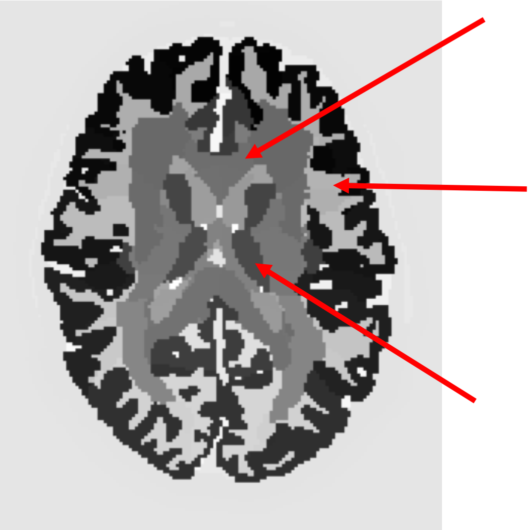


Genu of corpus callusum
(highly aligned WM)

Subcortical WM

Thalamus (GM)

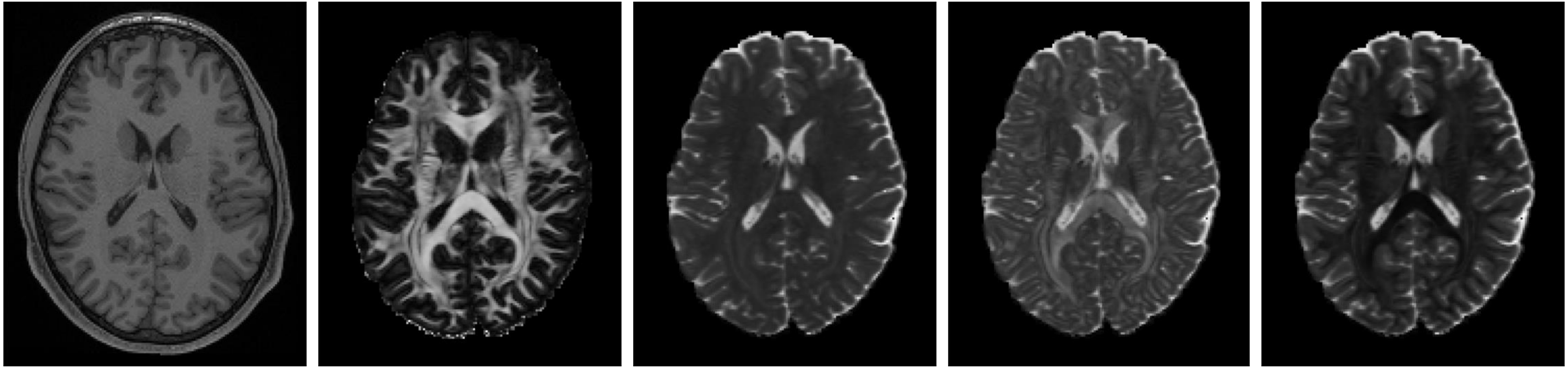
Segmentation



https://www.humanconnectome.org/

Classification of brain tissue from HCP DTI data

T1w | FA | MD $(\lambda_1 + \lambda_2 + \lambda_3)/3$ | AD (λ_1) | RD $(\lambda_2 + \lambda_3)/2$



https://www.humanconnectome.org/

Classification of brain tissue from HCP DTI data

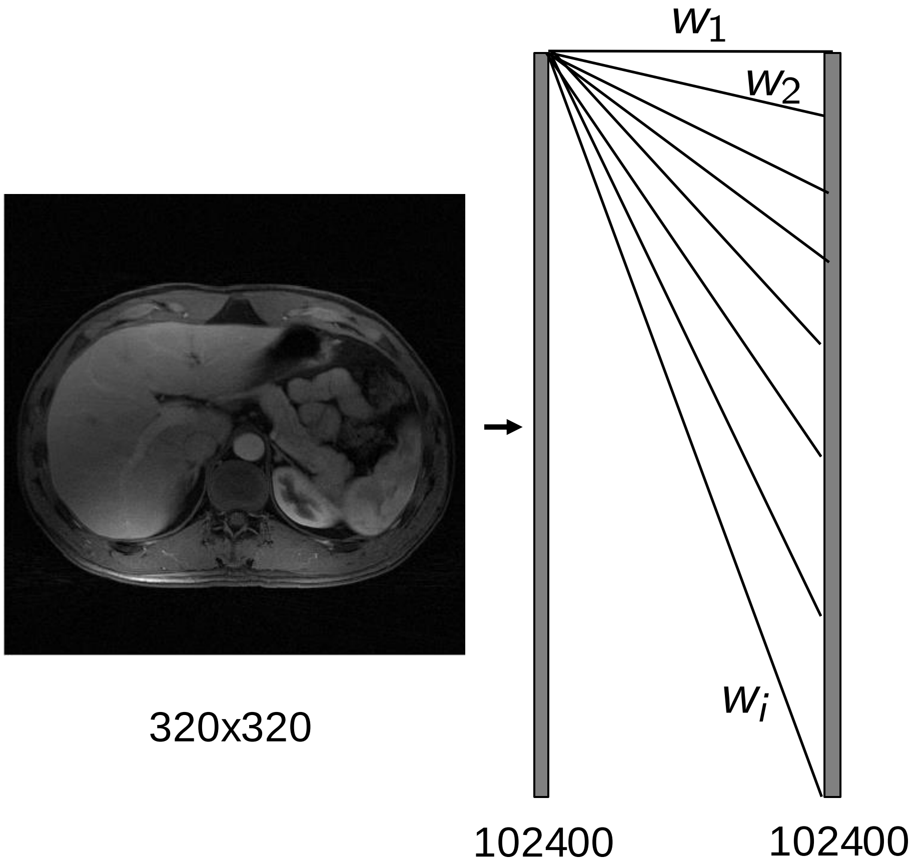
$$\{(x_1, y_1), \ldots (x_N, y_N)\}$$

$$x_i = [T1w_i, FA_i, MD_i, AD_i, RD_i]$$

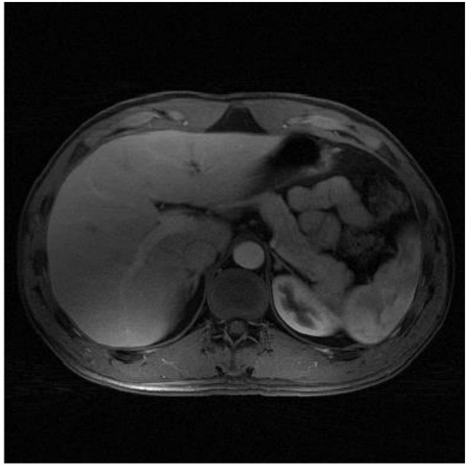
T1w (a.u.)	FA (-)	MD $\left(\frac{\mu m^2}{ms}\right)$	AD $\left(\frac{\mu m^2}{ms}\right)$	RD $\left(\frac{\mu m^2}{ms}\right)$	Class	Class label
898	0.22	1.066592	1.33	0.94	Thalamus	1
1007	0.68	0.39	0.72	0.22	CC	2
867	0.38	0.58	0.82	0.45	Cortical WM	3
…	…	...	...	...	…	…

Exercise example 2:
Classification of image quality of accelerated reconstructions with convolutional Neural Networks (CNNs)

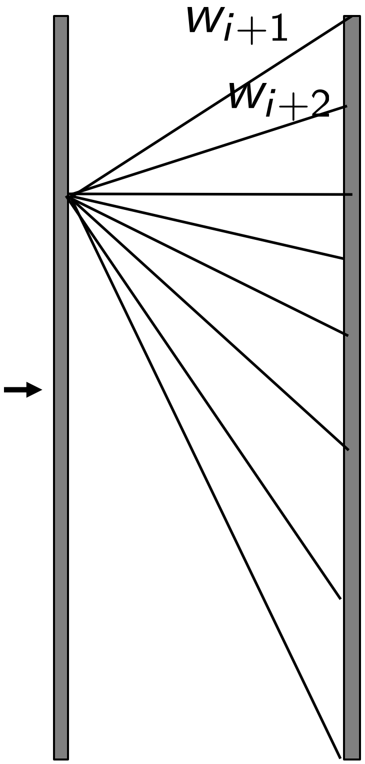
Fully connected Neural Networks



Fully connected Neural Networks



320x320



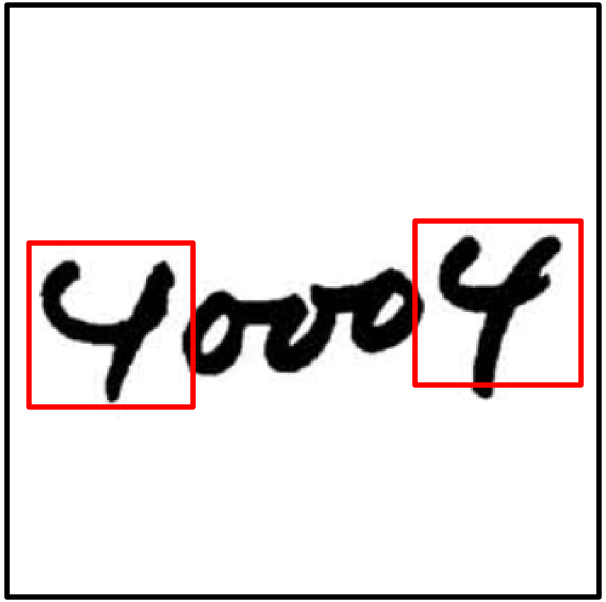
102400 102400

Number of parameters:
$102400*102400 \approx 1.05*10^{10}$

More efficient use of parameters!

Convolutional neural networks

Figure 1: Examples of original zipcodes from the testing set.



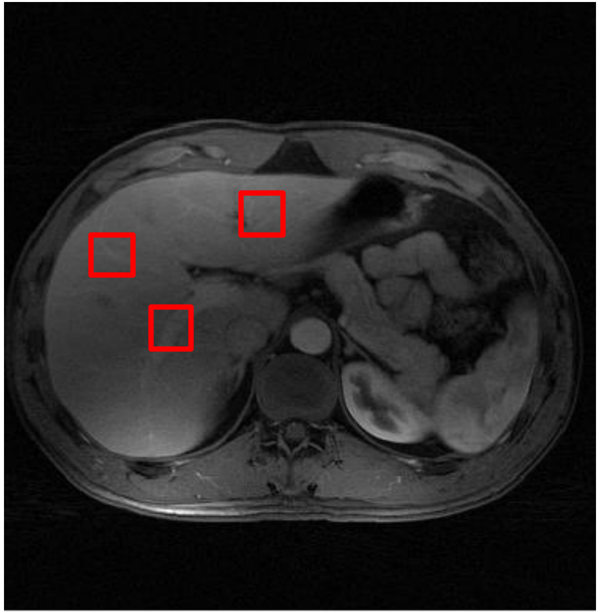
Local connectivity

Share parameters

Lecun: NIPS 1989

Convolutional neural networks

Figure 1: Examples of original zipcodes from the testing set.



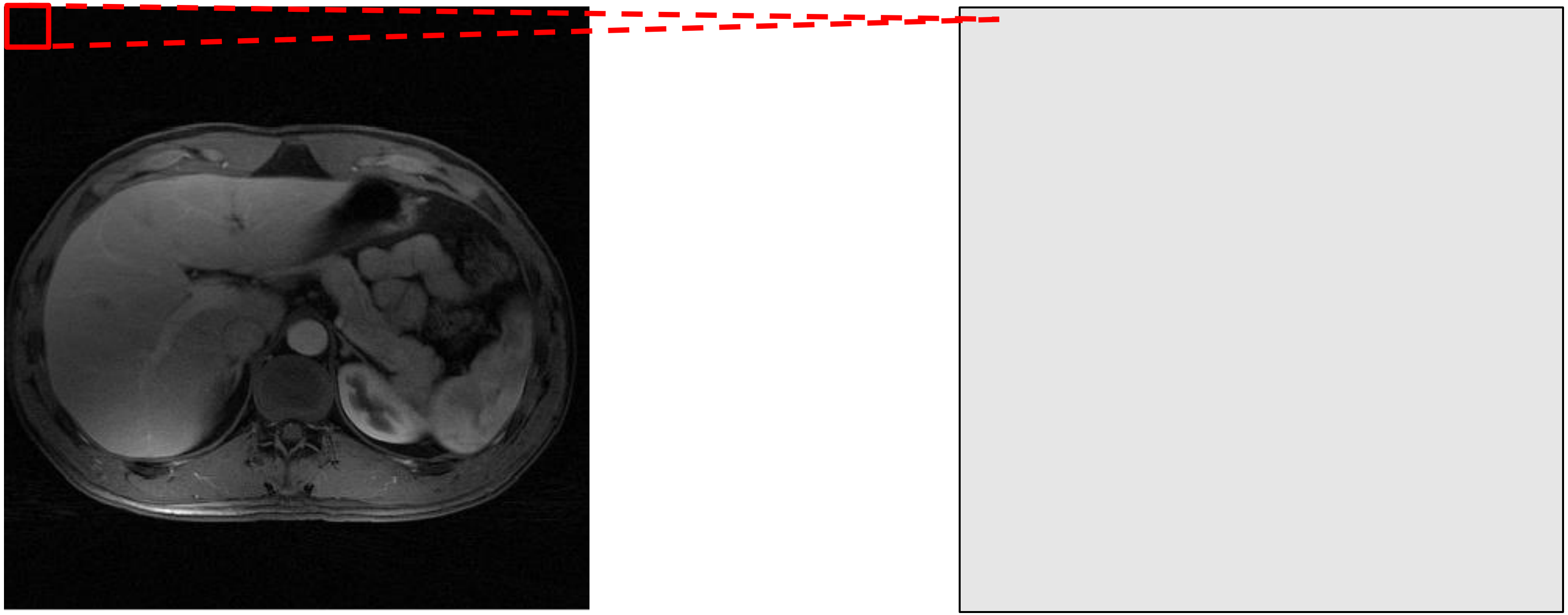
Local connectivity

Share parameters

Lecun: NIPS 1989

Convolutional layers

$$w^T x + b$$

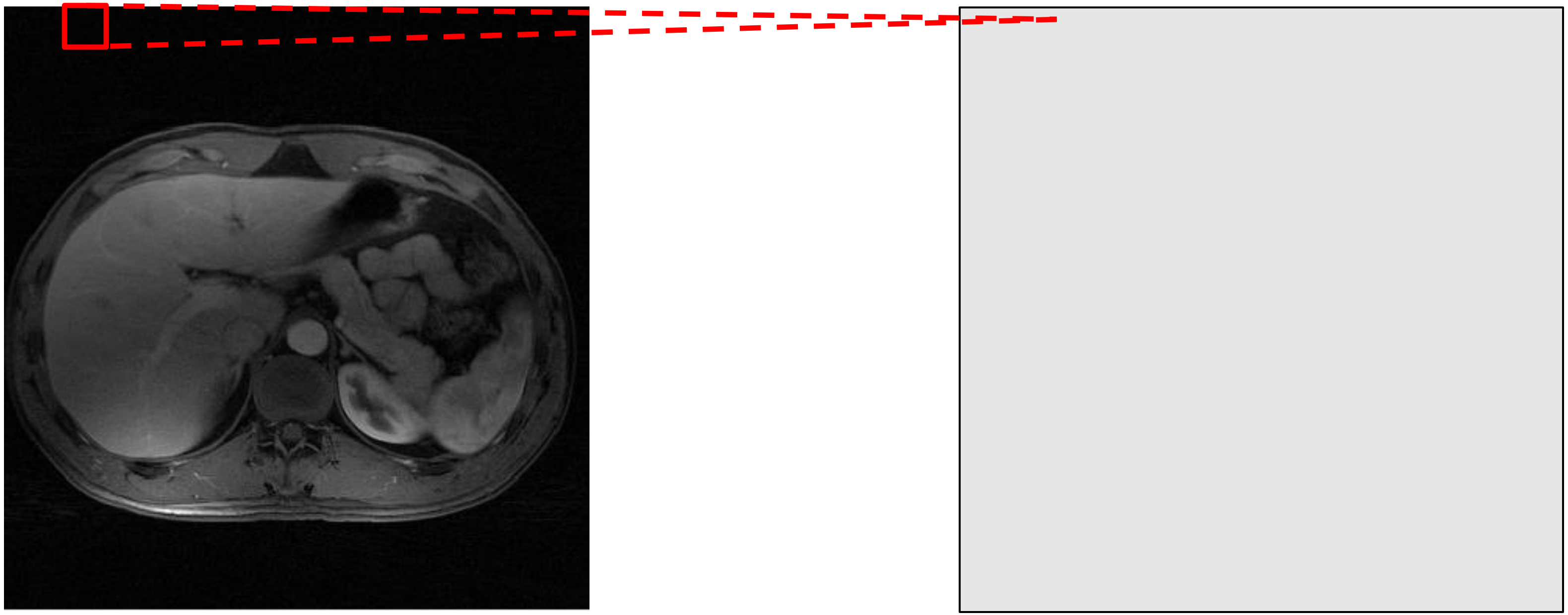


320x320 image
3x3 filter w

Lecun: NIPS 1989

Convolutional layers

$$w^T x + b$$



320x320 image
3x3 filter w

Model parameters: 3*3+1 = 10

Lecun: NIPS 1989

Example 2: Classification of image quality of accelerated reconstructions

Fully sampled vs 4 times PI-CS accelerated

Fully sampled reference

PI-CS R=4

Knoll, MRM (2011)

Fully sampled vs 4 times PI-CS accelerated

Fully sampled reference

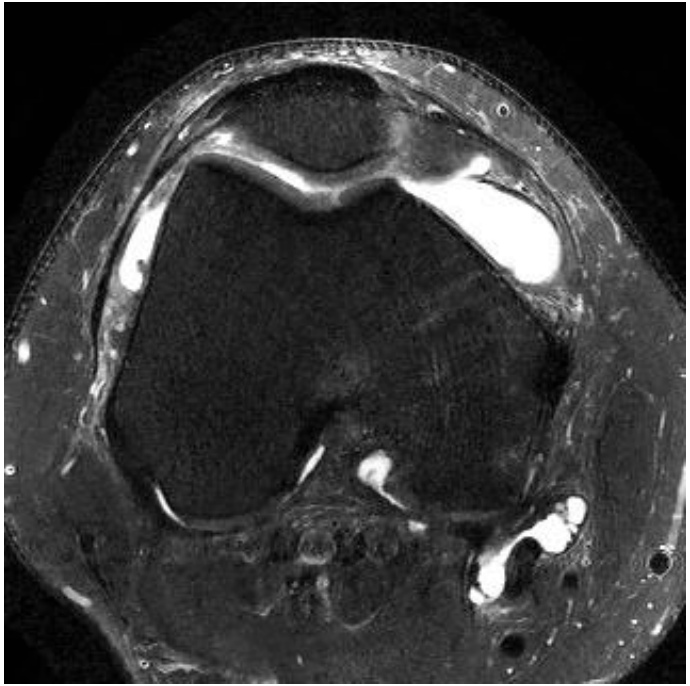
PI-CS R=4

Knoll, MRM (2011)

Fully sampled vs 4 times PI-CS accelerated

Fully sampled reference

PI-CS R=4



Knoll, MRM (2011)

Fully sampled vs 4 times PI-CS accelerated

Fully sampled reference

PI-CS R=4

Knoll, MRM (2011)

Summary

- Short recap of neural networks
- Training neural networks with gradient descent
- Backpropagation: Efficient implementation of chain rule
- Exercise: PyTorch examples for MLPs, CNNs