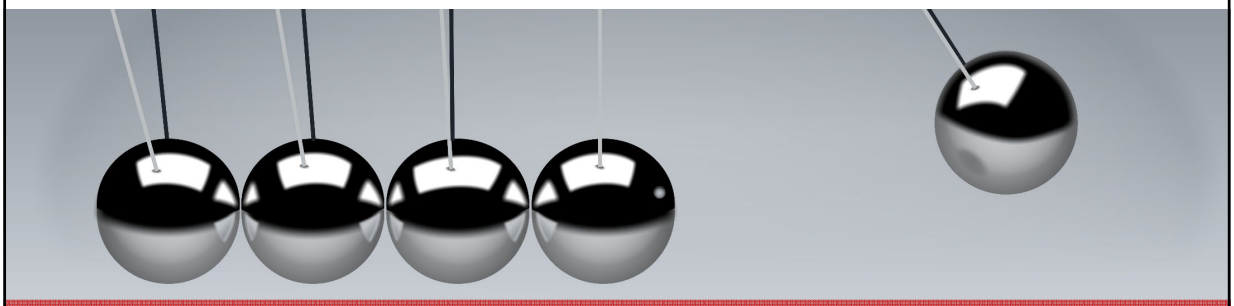


Module 4: Neural networks & backpropagation

5LSM0: Convolutional neural networks for computer vision

Fons van der Sommen

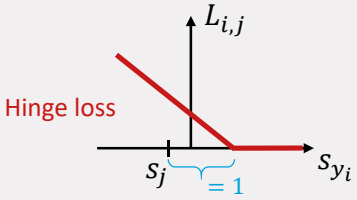
Electrical Engineering / VCA research group

Last time

Loss functions

- Way to quantify the "badness" of a certain (set of) prediction(s)
- Hinge loss:

$$L_i = \sum_{i \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



- Total loss:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

Regularization — *How much do we like this solution?*

- Cross-entropy loss:

$$L_i = -\log P(Y = k|X = x_i) = -\log\left(\frac{e^{s_k}}{\sum_j e^{s_j}}\right)$$

Softmax function — *Probabilities!*

Last time

Optimization

- Follow the slope to go downwards in the loss-landscape
- Compute gradient information to find the slope:
  - *Finite differences:* $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$ Super slow!!
- Derive an analytical solution for the gradient $\nabla_W L$

Use update function $W \leftarrow W - \eta \nabla_W L$ to update parameters

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(f(x_i, W), y_i) + \lambda \nabla_W R(W)$$

- Data sets can be large, we don't want to wait until we've seen all the data

Mini-batch gradient descent!

This time

I want to descent in that loss landscape!

➢ So how do I find the gradient of my loss function?

Backprop!

Linear classification is okish, but we want to do cooler stuff!

➢ Can we combine linear models to make more advanced models?

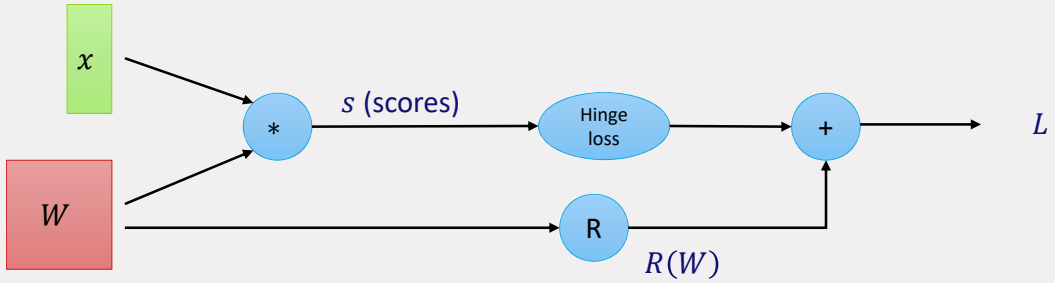
Yes! (but you need to add some special ingredients)

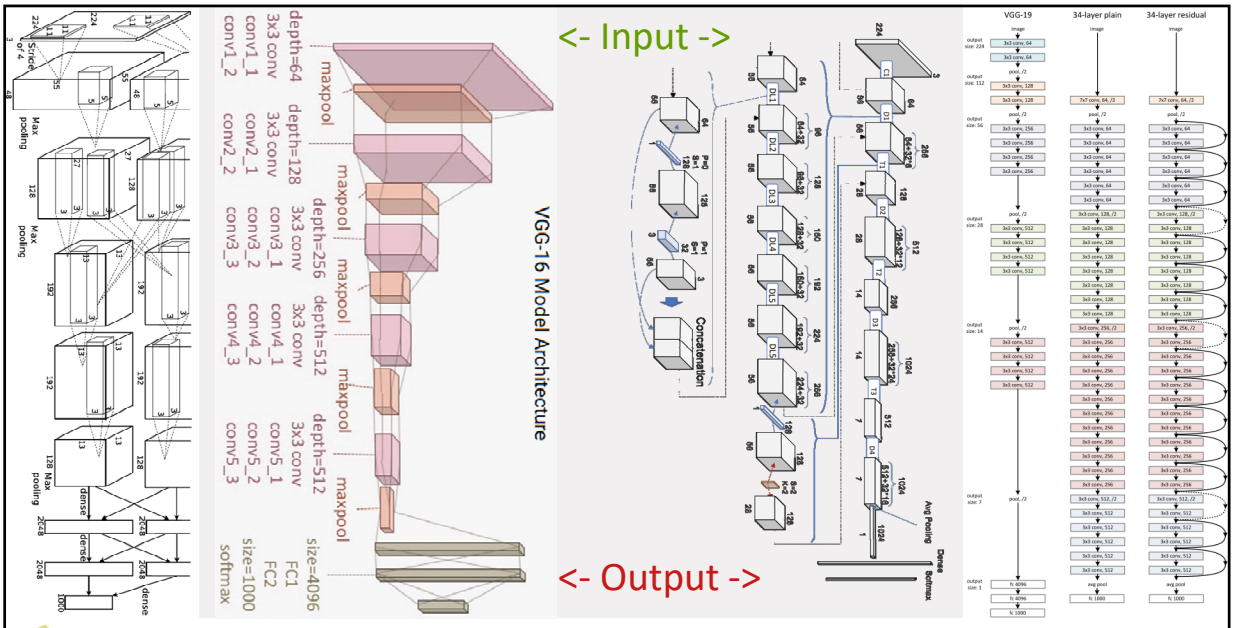
➢ Is it really like a human brain? :D

NO!:D

Computational graphs

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} \qquad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$





Images from www.towardsdatascience.com

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Backpropagation

Backpropagation

The backpropagation of errors

- Concept goes back to the early 60ies, but fully appreciated after paper by Rumelhart, Hinton and Williams (1986)
- Simple and efficient procedure to evaluate the gradient of the loss function
  - *Or any differentiable function for that matter…*
- Very useful to train neural networks (computational graphs)
- Basically: find out how each parameter in a (machine learning) model affects the loss
  - *Perfect to determine how to change the weights of a neural network in order to minimize the loss!*

Chapter 6.5 of deeplearningbook.org

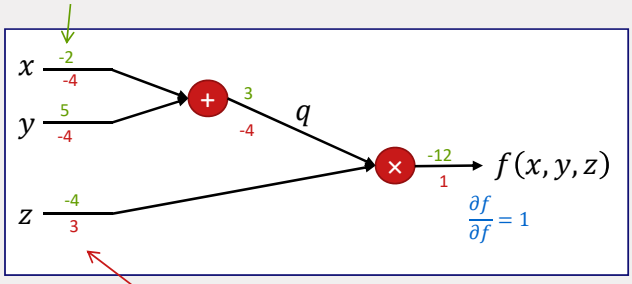
*Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". Nature. **323** (6088): 533–536*

Backpropagation

Example*

- Given: $f(x, y, z) = (x + y)z$
- We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Inputs above the lines



Gradients below the lines

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

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

Chain rule

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

$\frac{\partial f}{\partial q} = -4$

$\frac{\partial f}{\partial z} = 3$

$\frac{\partial f}{\partial y} = -4$

$\frac{\partial f}{\partial x} = -4$

*Example from cs231n lecture 4 slides 12-23

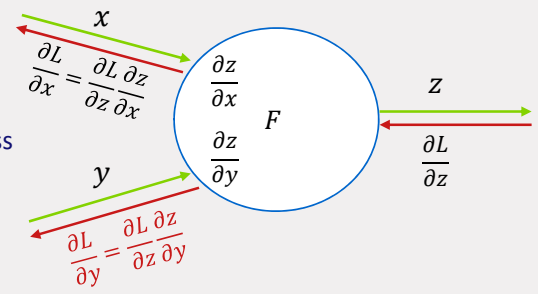
Backpropagation

Forward pass Compute network prediction and loss

Backward pass Backpropage the error (loss) to find the influence of all the weights

Local gradient information

- Each node only considers the inputs and output
- Two things a gate can compute on forward pass
  - *(1) Output z, (2) gradient w.r.t. all the inputs*
- Receive influence on final loss during backward pass
- Compute effect of inputs on final loss by means of the chain rule
- Gates communicate the influence on the final loss



Backpropagation

Another example*:

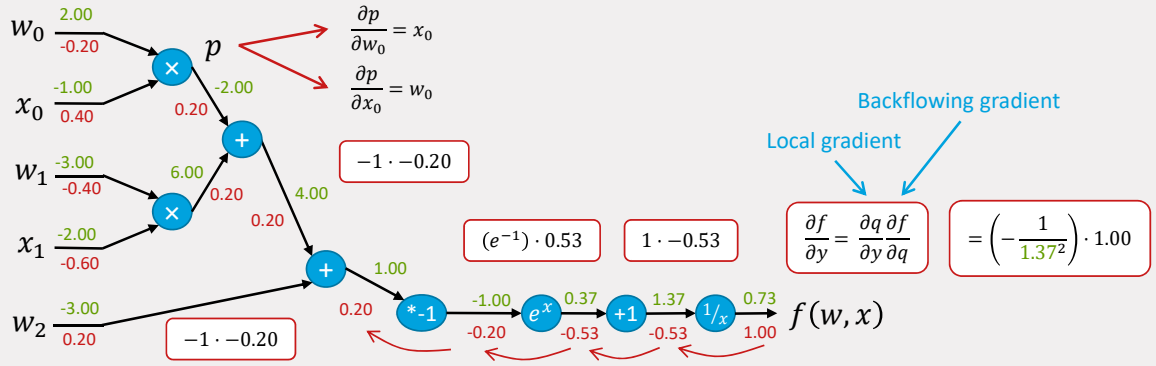
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$f(x) = c + x \rightarrow \frac{df}{dx} = 1$

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -\frac{1}{x^2}$



*Example from cs231n lecture 4 slides 12-23

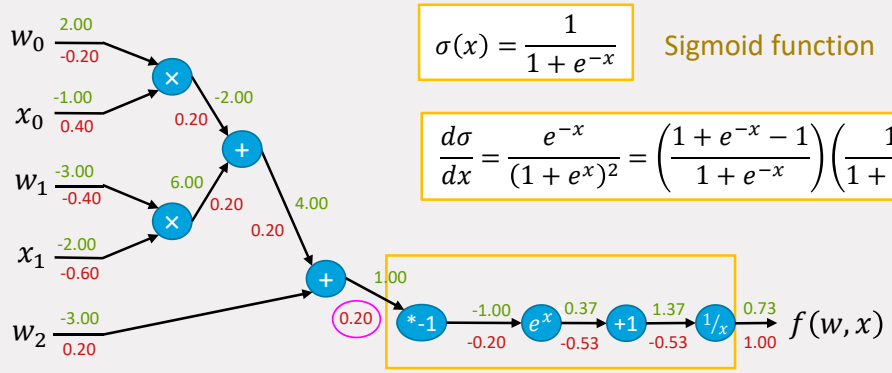
Backpropagation

Another example*:

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

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

$$\frac{d\sigma}{dx} = \frac{e^{-x}}{(1+e^x)^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



$$\frac{d\sigma}{dx}\frac{df}{d\sigma} = \frac{0.73}{1-0.73} \cdot 1 = 0.2$$

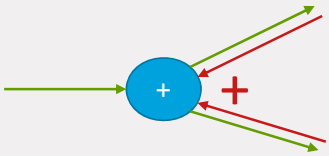
*Example from cs231n lecture 4 slides 12-23

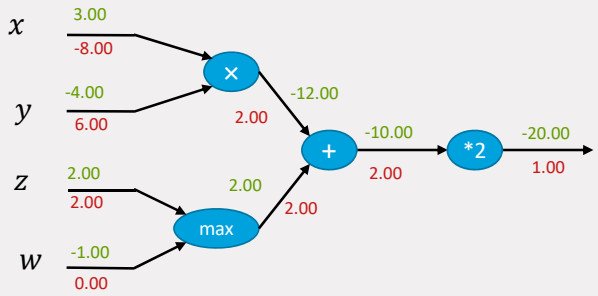
Backpropagation

Patterns in backward flow

- Add gate: gradient distributor
- Max gate: gradient router
- Multiply gate: …

Gradients add at braches



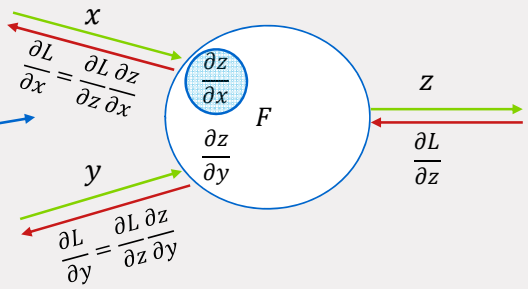


Backpropagation

From scalars to vectors…

$x, y, z \rightarrow$ **vectors**

Jacobian matrix:
Derivative of each element of z w.r.t. each element of x



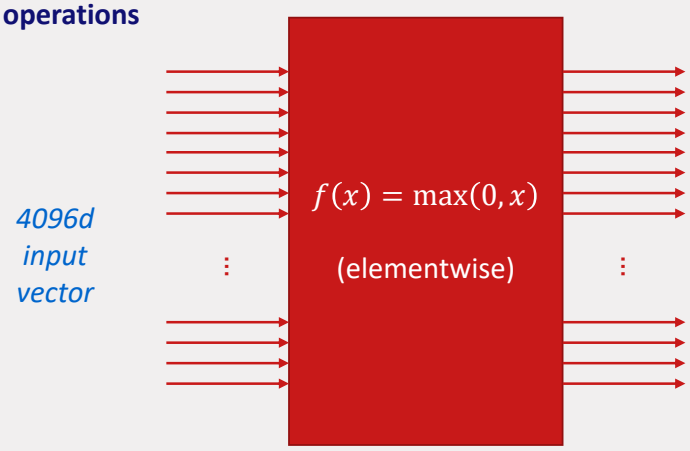
Backpropagation

Q1: What would be the size of our Jaccobian? $\frac{\partial f}{\partial x}$

Q2: Structure of Jaccobian matrix?

Vectorized operations

- Example:



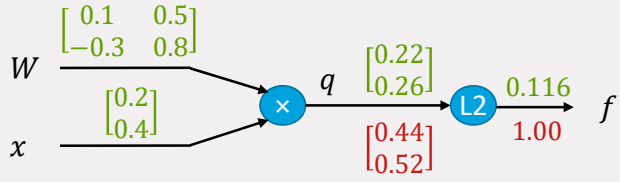
Backpropagation

$\mathbf{1}_{k=i}$: *indicator function*

Vectorized operations

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

$W \in \mathbb{R}^{n \times n}$ $x \in \mathbb{R}^n$



$f(q) = \|q\|^2 = q_1^2 + q_2^2 + \cdots + q_n^2$

$$q = W \cdot x = \begin{bmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{bmatrix}$$

$\frac{\partial f}{\partial q_i} = 2q_i$ $\nabla_q f = 2q$

$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} \cdot x_j$

$$\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \sum_k (2q_k)(\mathbf{1}_{k=i} \cdot x_j) = 2q_i x_j$$

$\nabla_W f = 2q \cdot x^T$

**Example from cs231n lecture 4 slides 58-74*

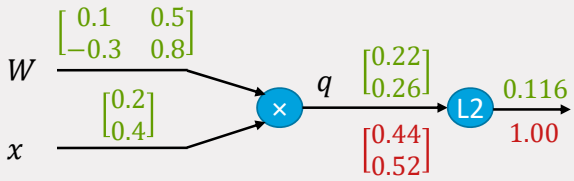
Backpropagation

$\mathbf{1}_{k=i}$: *indicator function*

Vectorized operations

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

$W \in \mathbb{R}^{n \times n}$ $\quad$ $x \in \mathbb{R}^n$



$$f(q) = \|q\|^2 = q_1^2 + q_2^2 + \cdots + q_n^2$$

$$q = W \cdot x = \begin{bmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{bmatrix}$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i} \qquad \frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}$$

$$\boxed{\nabla_x f = 2W^T \cdot q}$$

**Example from cs231n lecture 4 slides 58-74*

So far…

Neural networks will be large

- Impractical to write down an explicit formulation of the gradient by hand

Backpropagation recursively applies the chain rule to find the influence of all the parameters and inputs on the final loss

- Forward pass:
  - *Compute the outputs of all the gates to finally compute the network output and the resulting loss.*
  - *Also compute the local gradients and store them for use during backward pass.*
- Backward pass:
  - *Get the upstream gradient (from a specific gate to the output)*
  - *Multiply the upstream gradient with the local gradient, and pass it on to all parent nodes*

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Neural networks

Neural networks

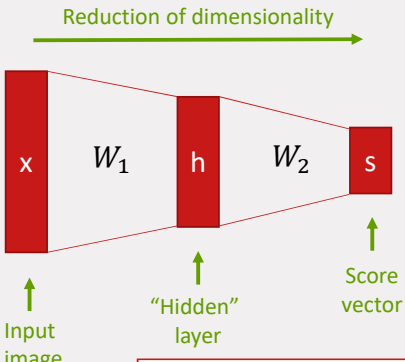
Without the brain stuff…

- Linear score function $f = Wx$
- ➢ 2-layer Neural network $F = W_2 \max(0, W_1 x)$
  - *Or 3-layer..* $F = W_3 \max(0, W_2 \max(0, W_1 x))$

Linear operation

Non-linearity

- Linear operation followed by a non-linearity

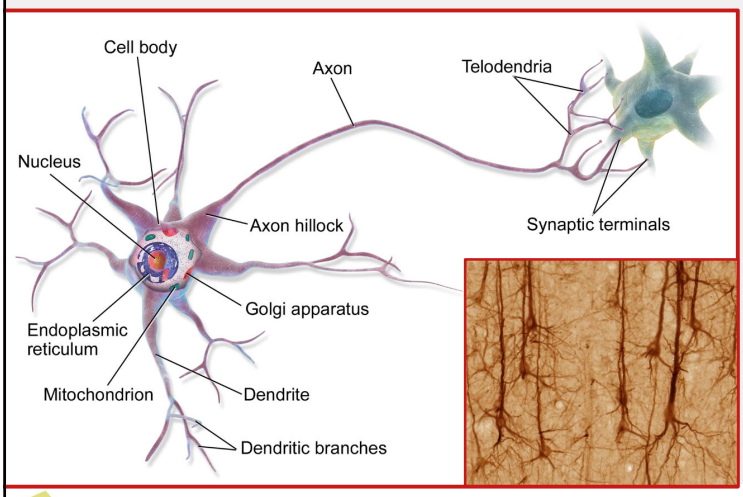


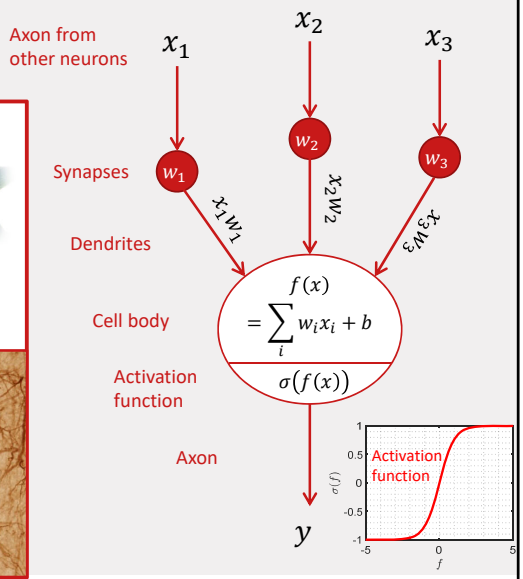
Q1: What happens in the hidden state here?

Q2: What would happen if we remove the non-linearity?

TU/e

Neural networks





*Images from Wikipedia

Neural networks

Don't take this brain comparison all too seriously…

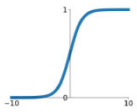
Biological neurons are much more complicated

- Many different types of neurons → in Artificial Neural Networks (ANNs) they're all the same
- The dendrites themselves can perform non-linear computations → in ANN's just connections
- Synapses are complex non-linear dynamical systems → in ANN's just single weights
- Communication within the brain much more complex than just firing rates (activation scores)
- Neuron firing behavior affected by e.g. hormones → in ANN's always the same behavior
- …

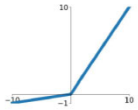
Neural networks

Many different activation functions!

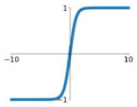
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

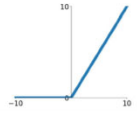


tanh
$\tanh(x)$

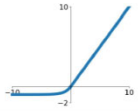


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



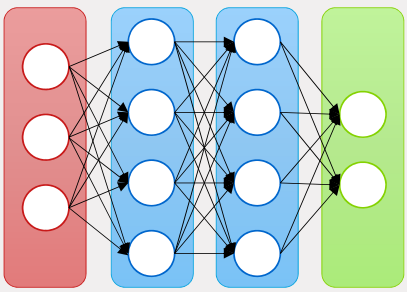
ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



More on this next module!

*From cs231n lecture 4 slide 96

Neural networks



- Three-layer neural network
  - *Count the number of layers that do computations*
  - *Number of layers minus input layer*
- Each connection has a certain weight
- Fully-connected layers
- Each layer can be evaluated using matrix multiplication

Neural network

DEMO TIME

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html
- https://playground.tensorflow.org/
- Neural networks map to higher dimensional space in which data is linearly separable

Q: Other methods exploiting such an approach?

Summary

Backpropagation efficiently finds the gradient of the loss function

- Use computational graphs and the chain rule
  - *Forward pass:* *Compute final loss and local gradients*
  - *Backward pass:* *Propagate the loss through the network to find the influence of all the weights*

Neural networks

- Layer-wise arrangement of artificial neurons (linear classifiers followed by a non-linearity)
- Biological comparison sounds cool but has only limited validity

Next time

Convolutional neural networks (CNNs)!

- Organize a hierarchical filtering structure as a neural network
- We can learn our filter kernels using backprop!

What can you do with CNNs?

What are they made of?

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

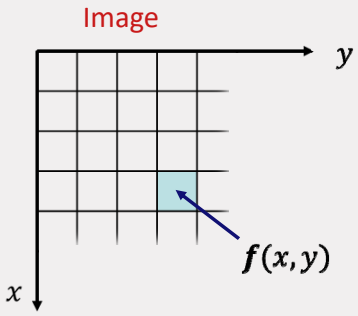
Encore: Convolution (recap?)

For feature extraction

Convolution for feature extraction

Linear filtering

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$



$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-1)$	$w(1,0)$	$w(1,1)$

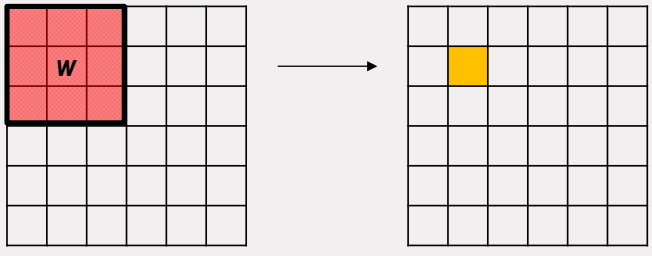
Filter kernel

TU/e

Convolution for feature extraction

Linear filtering

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

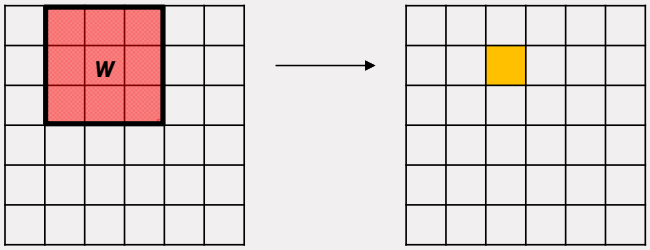


TU/e

Convolution for feature extraction

Linear filtering

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

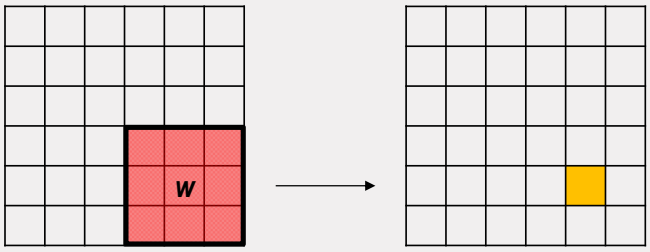


TU/e

Convolution for feature extraction

Linear filtering

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$



TU/e

Convolution for feature extraction

Linear filtering

Flip mask w.r.t. signal

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Image f

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Kernel w

1	2	3
4	5	6
7	8	9

Zero-padded image

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Convolution result

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	2	3	0	0	0
0	0	0	4	5	6	0	0	0
0	0	0	7	8	9	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

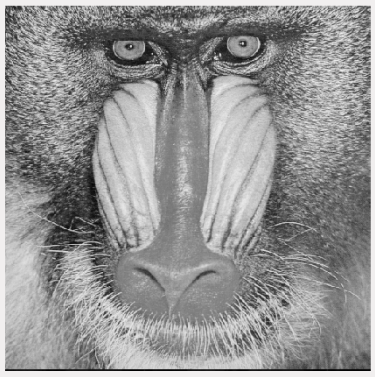
Cropped result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

TU/e

Convolution for feature extraction

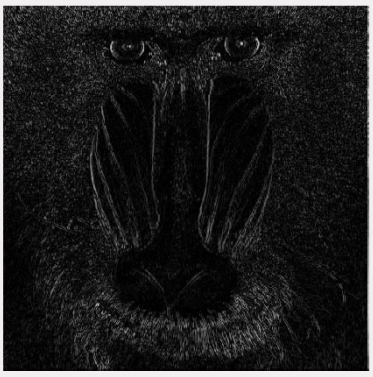
Example kernels



Convolution

Kernel w =

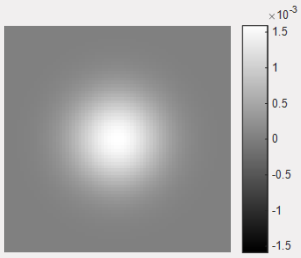
1	0	-1
1	0	-1
1	0	-1

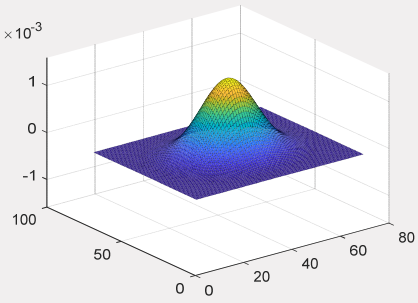


TU/e

Convolution for feature extraction

Example kernels

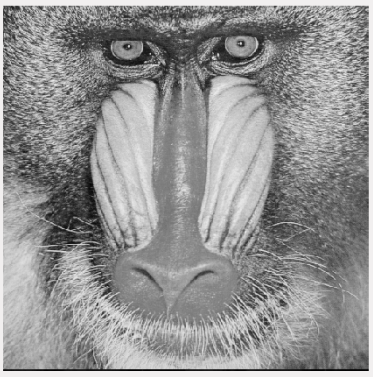




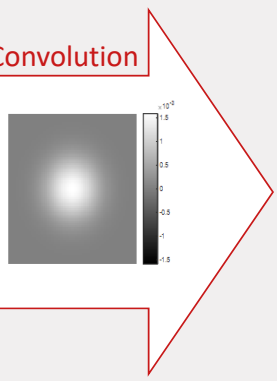
Low-pass filter

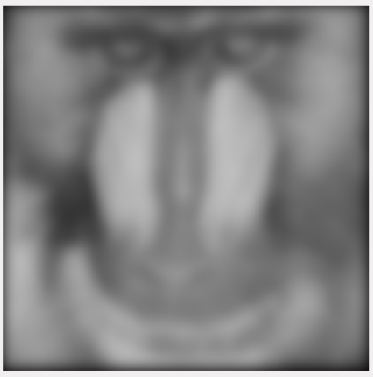
Convolution for feature extraction

Example kernels



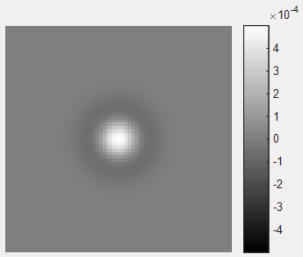
Convolution

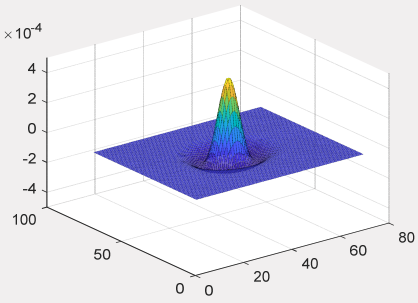




Convolution for feature extraction

Example kernels

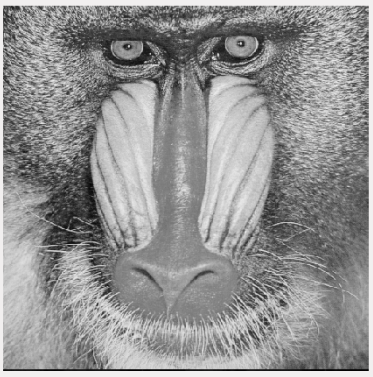




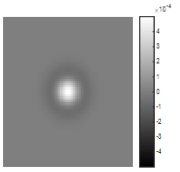
High-pass filter

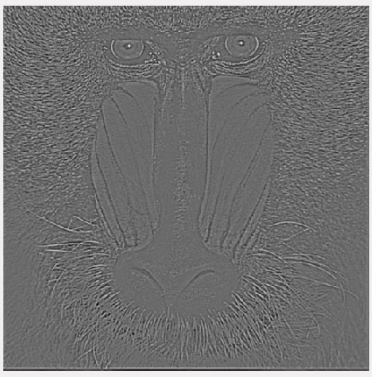
Convolution for feature extraction

Example kernels



Convolution

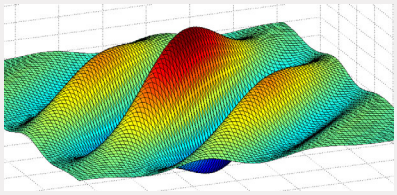


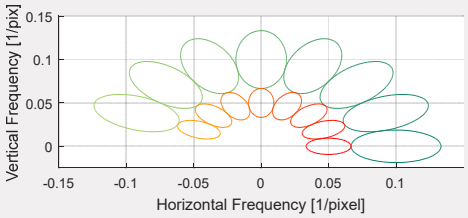


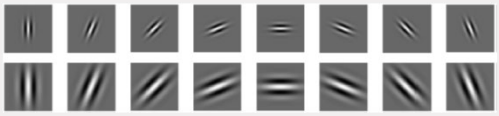
Convolution for feature extraction

Example kernels

Pass-band filters

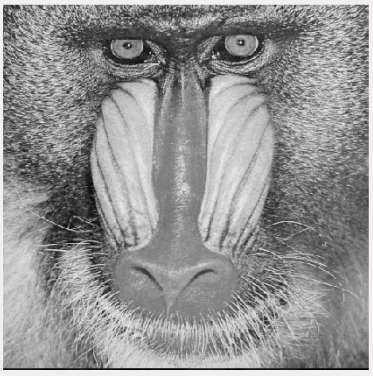




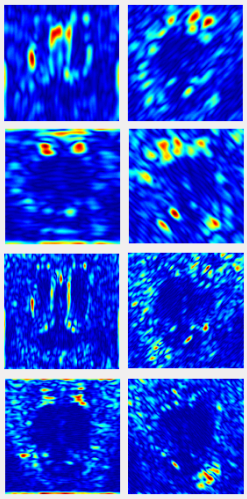


Convolution for feature extraction

Example kernels

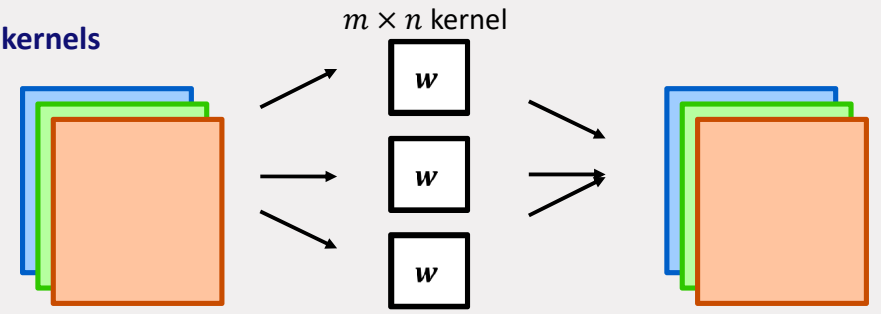


Convolution



Convolution for feature extraction

Color kernels



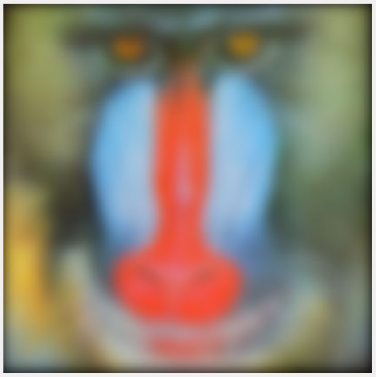
Input image

Three independent Convolutions with the same kernel

Filtered image

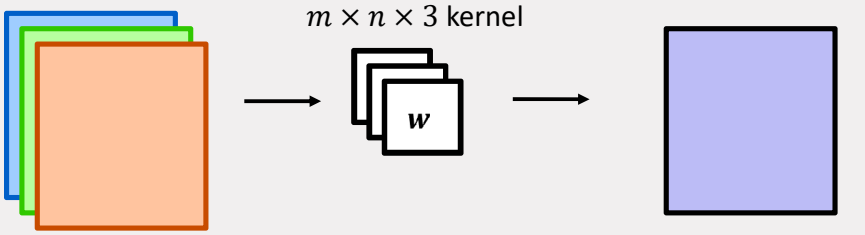
Convolution for feature extraction

Color kernels



Convolution for feature extraction

Color kernels



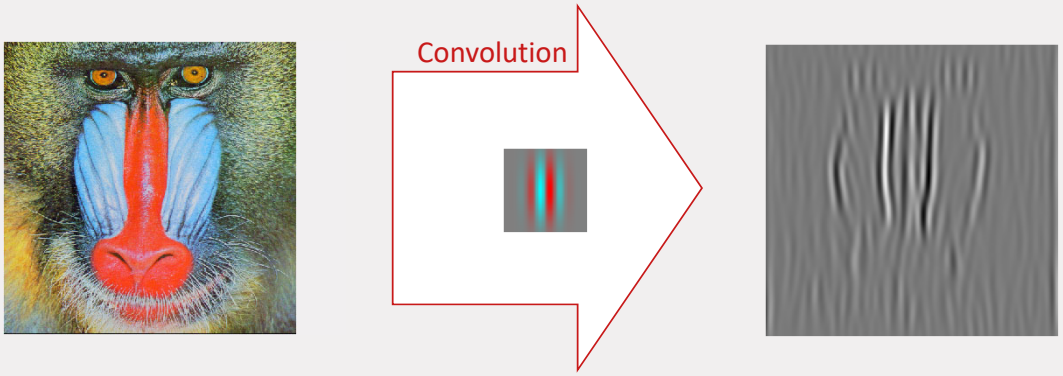
Input image

One "color" convolution

Filtered image

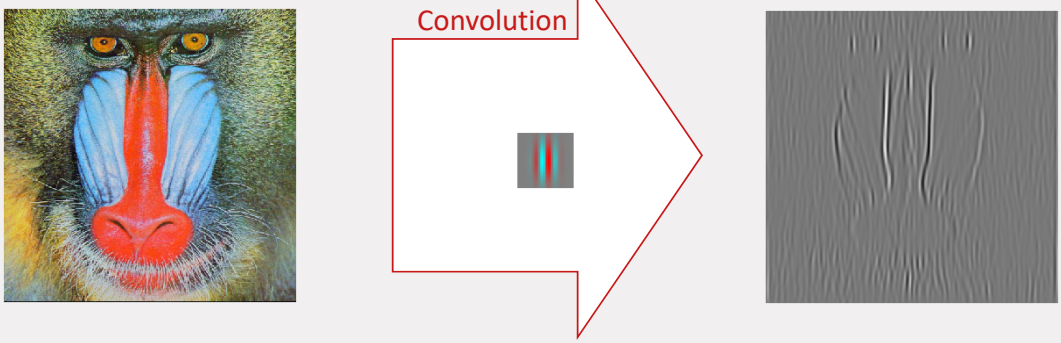
Convolution for feature extraction

Single filters to find specific color changes



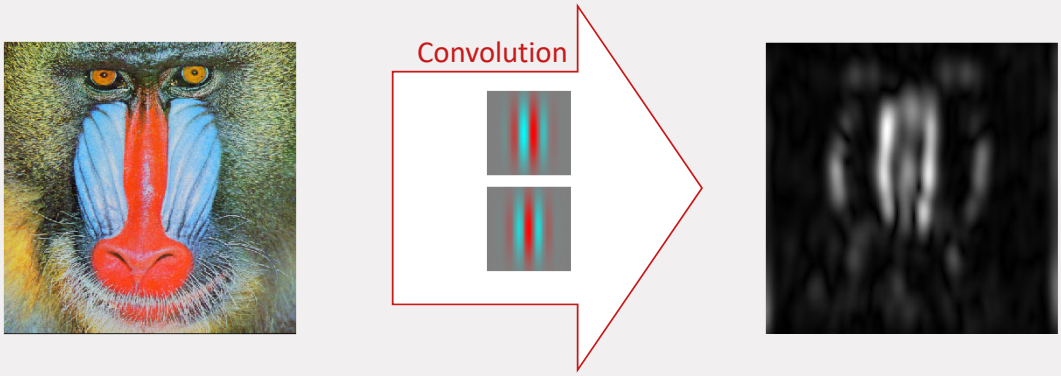
Convolution for feature extraction

Single filters to find specific color changes



Convolution for feature extraction

Quadrature pairs to find specific frequency content



Convolution for feature extraction

Quadrature pairs to find specific frequency content