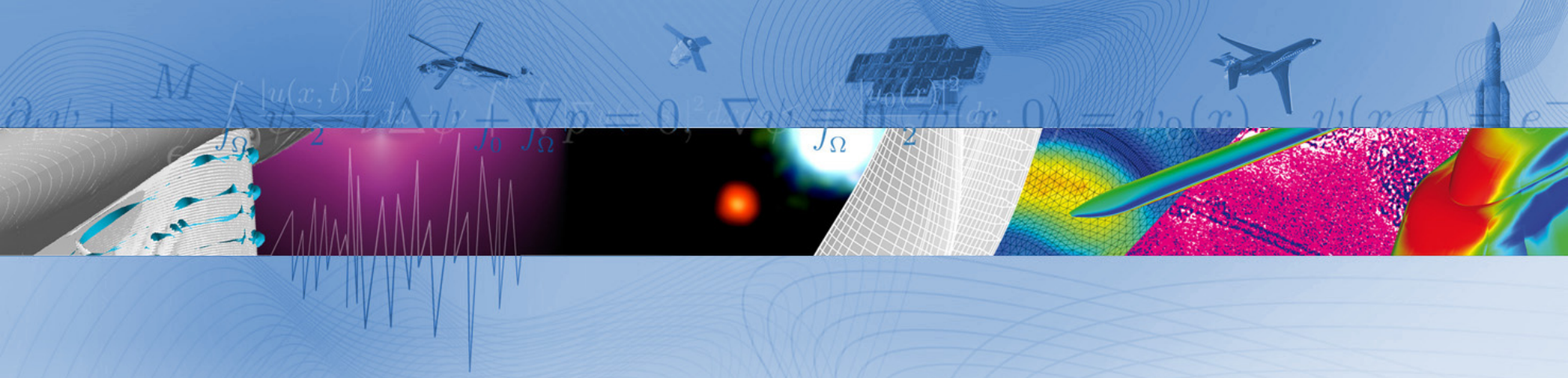


Machine Learning and Pattern Recognition

Deep Learning

Alexandre BOULCH

ONERA
THE FRENCH AEROSPACE LAB

Introduction

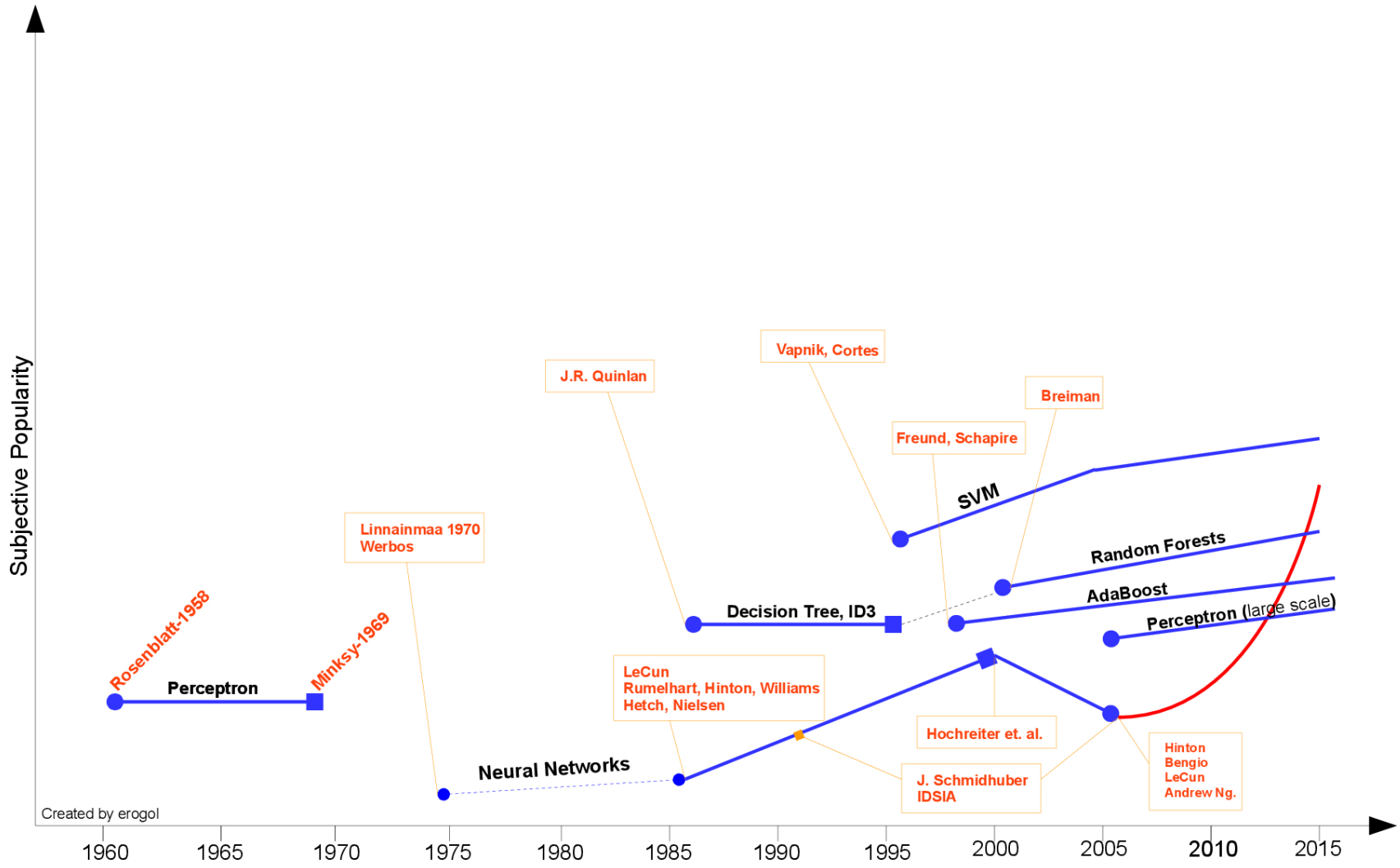
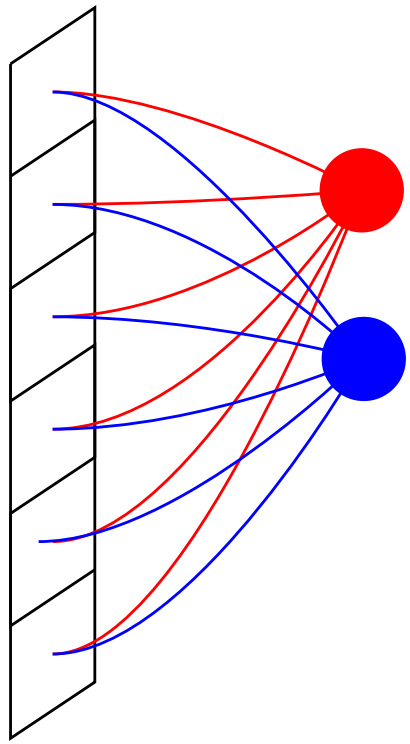
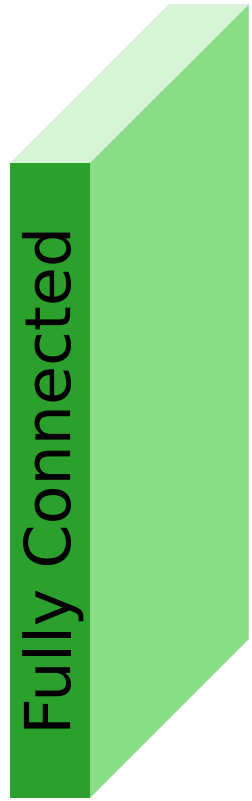


Table of contents

Fully connected

- Also called Linear layer or perceptron
- A neuron is connected to every input.
- High number of parameters ($|inputs| * |outputs|$)



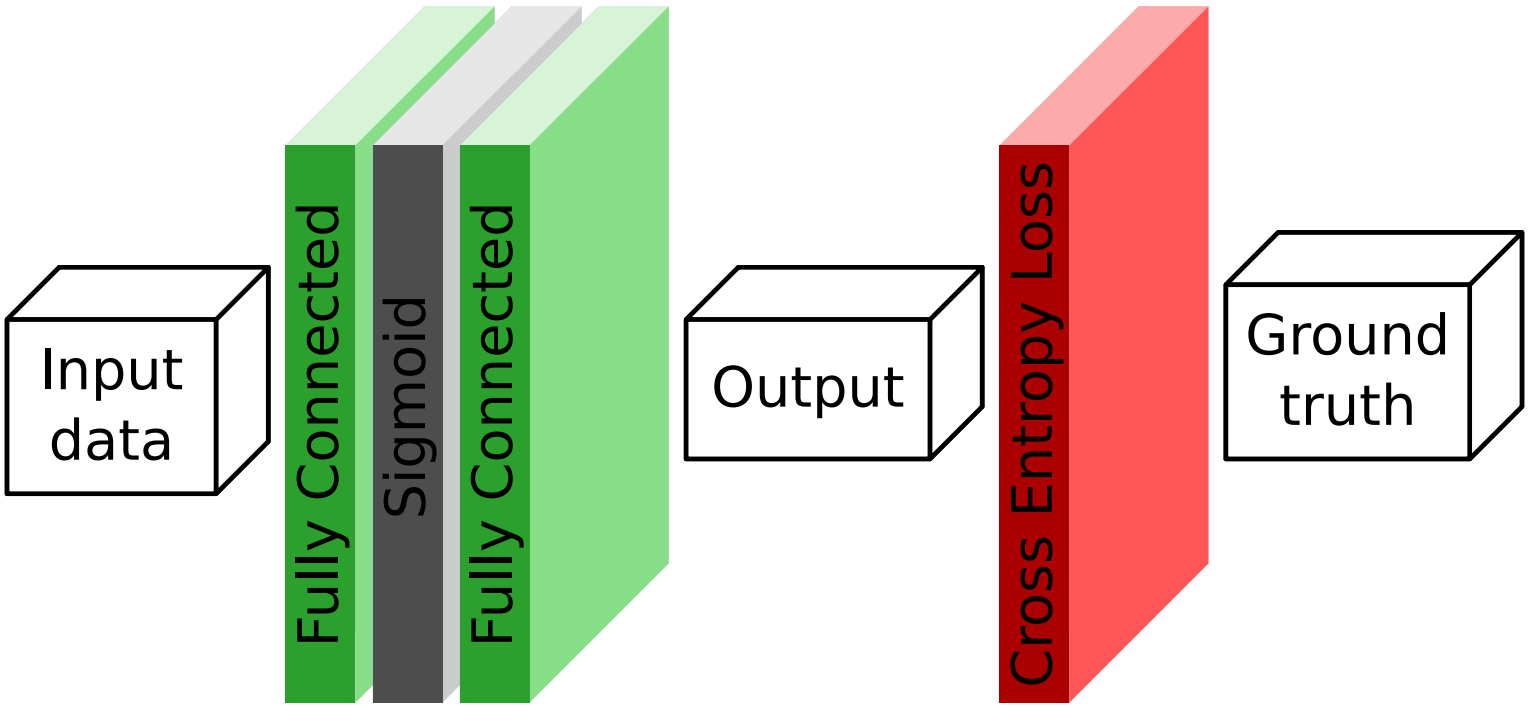


Multi layer perceptron

ONERA
THE FRENCH AEROSPACE LAB

Multi Layer Perceptron

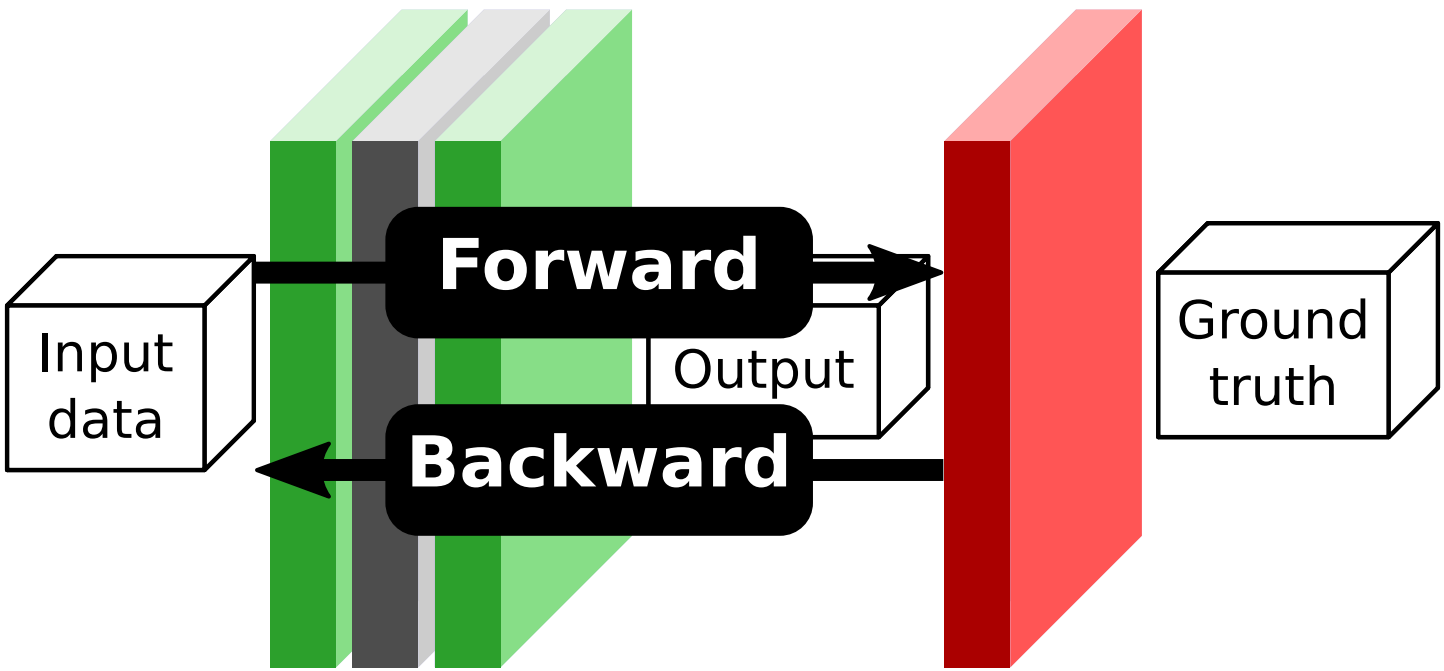
A stack of linear layers and activation functions (e.g. sigmoid).



Optimization

Forward-Backward algorithm

- **Forward**: iteratively compute the output of each layer from data to prediction
- **Backward**: iteratively compute the derivative of the parameters starting from the loss



Optimization

ONERA
THE FRENCH AEROSPACE LAB

Forward-Backward algorithm

- **Forward**: iteratively compute the output of each layer from data to prediction
- **Backward**: iteratively compute the derivative of the parameters starting from the loss

Chain rule

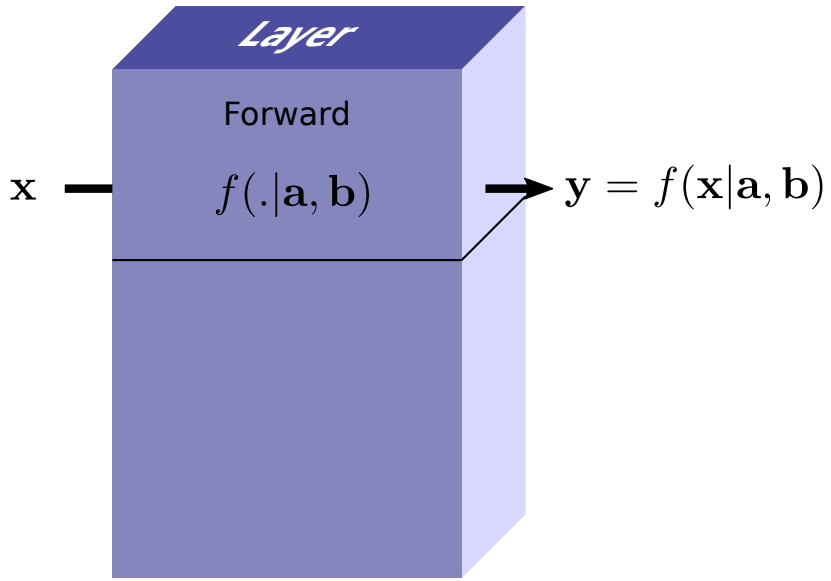
Easy differntiation is obtain by derivation of function composition:

$$\frac{\partial f(a = g(x, y, z), b, c)}{\partial x} = \frac{\partial g(x, y, z)}{\partial x} \frac{\partial f(a = g(x, y, z), b, c)}{\partial a}$$

Optimization: differentiation

Algorithm

- Forward pass

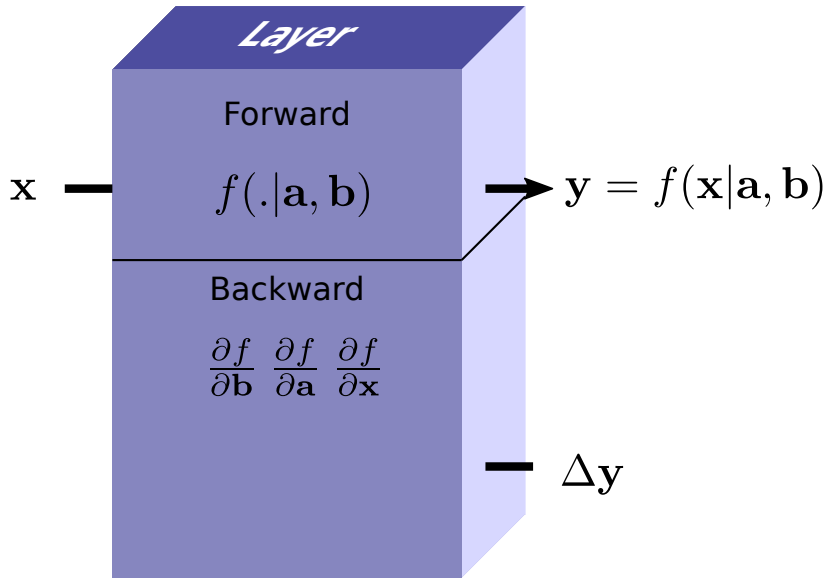


Optimization: differentiation

ONERA
THE FRENCH AEROSPACE LAB

Algorithm

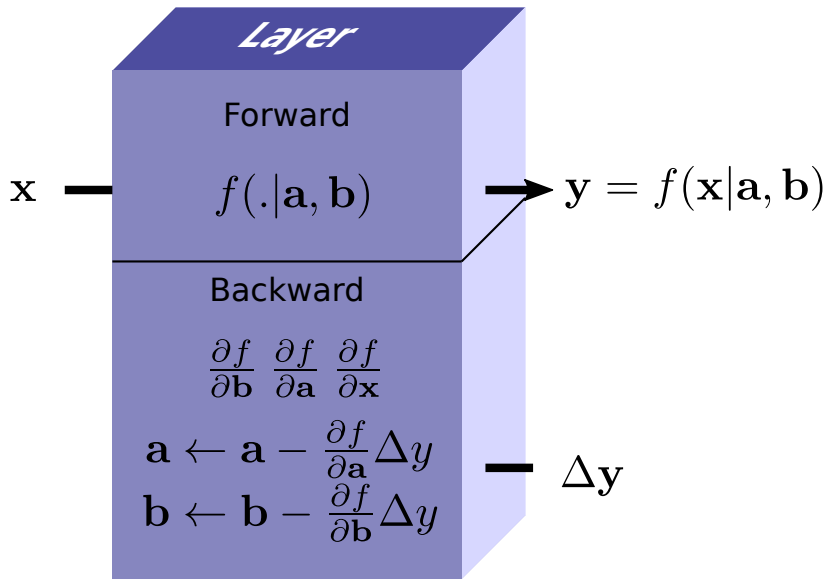
- Forward pass
- Get the output differential ($\Delta\mathbf{y}$)
- Compute partial derivatives for parameters and input



Optimization: differentiation

Algorithm

- Forward pass
- Get the output differential ($\Delta \mathbf{y}$)
- Compute partial derivatives for parameters and input
- Update the parameters



Optimization: differentiation

ONERA
THE FRENCH AEROSPACE LAB

Algorithm

- Forward pass
- Get the output differential ($\Delta\mathbf{y}$)
- Compute partial derivatives for parameters and input
- Update the parameters
- Compute differential for input

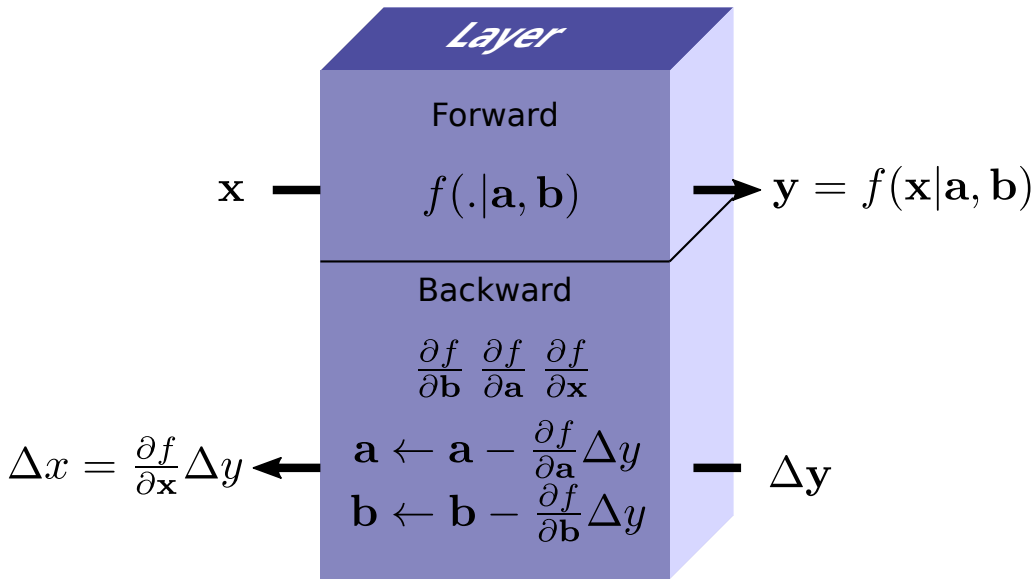
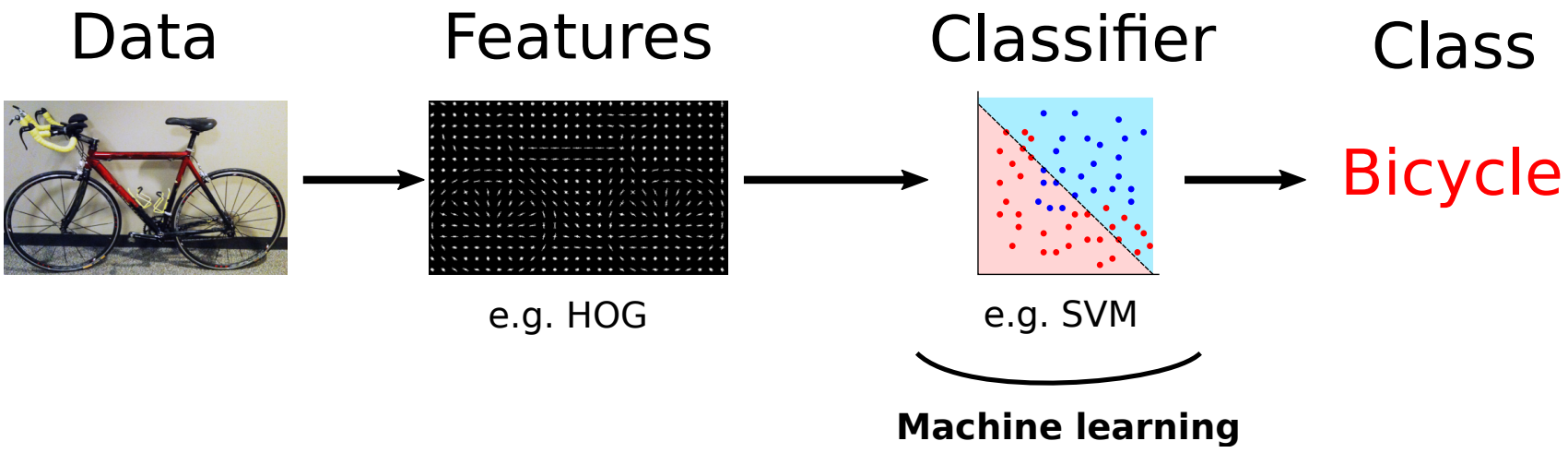


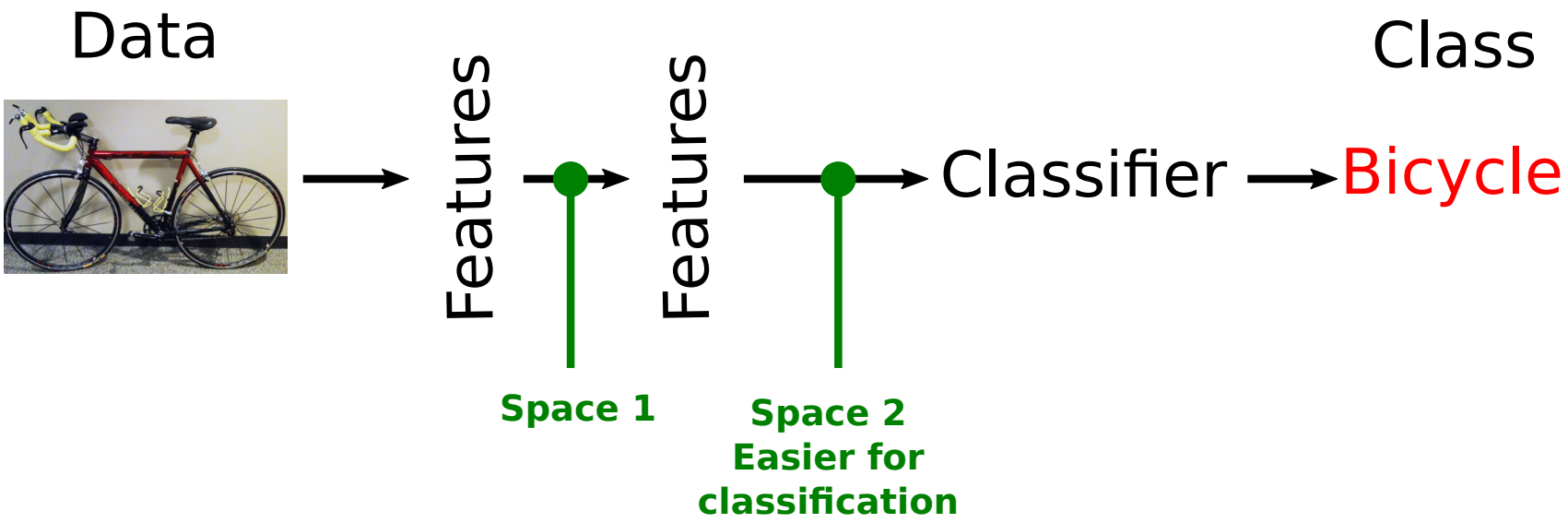
Table of contents

Table of contents

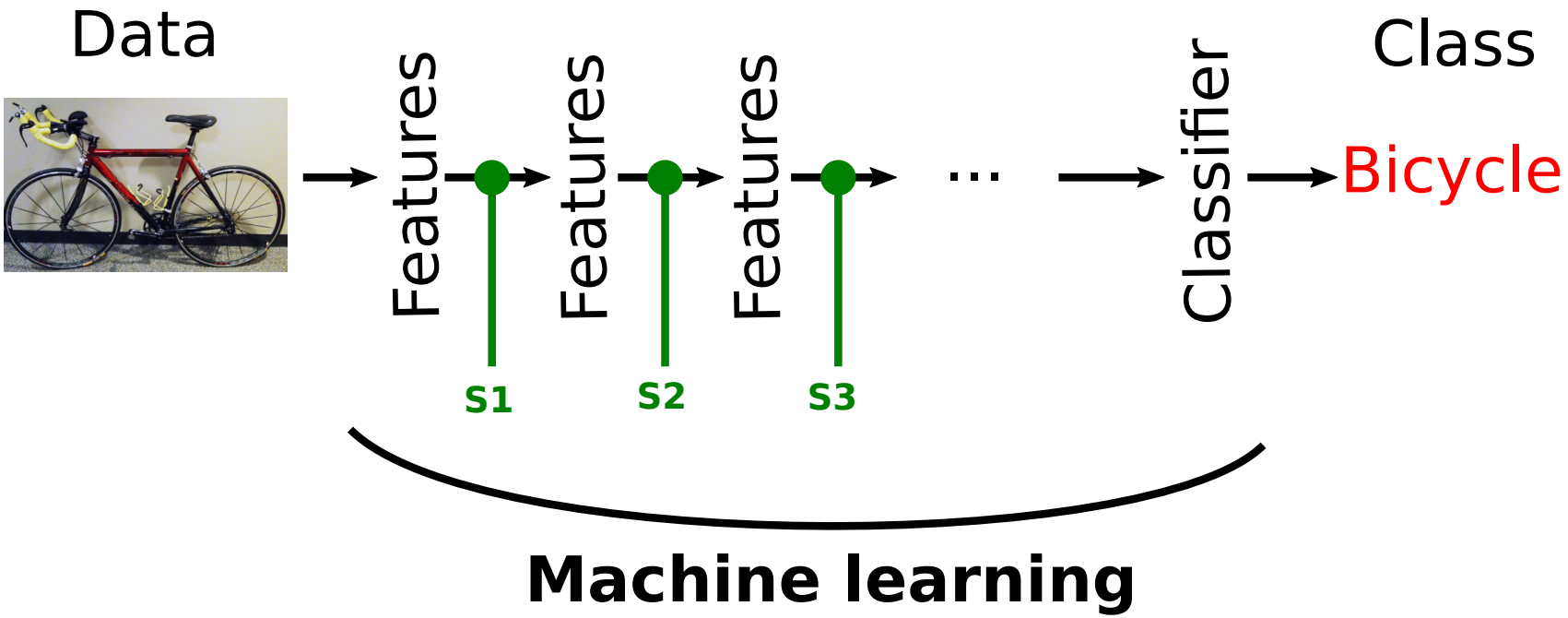
Deep learning concept



Deep learning concept



Deep learning concept



Concept

Deep learning is about learning representations.

Multi-layer perceptron

ONERA
THE FRENCH AEROSPACE LAB

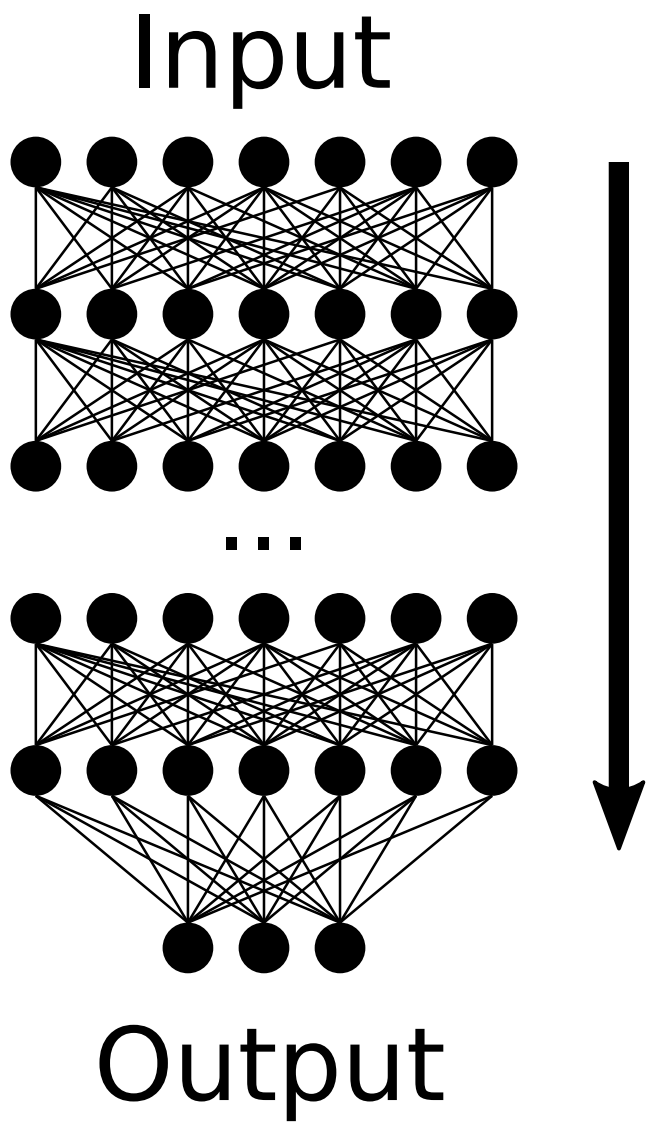
MLP becomes larger and deeper (before 1990)

- Difficult convergence
- Few data
- Very long training

$\longrightarrow$ Progessive loss of interest

SVM

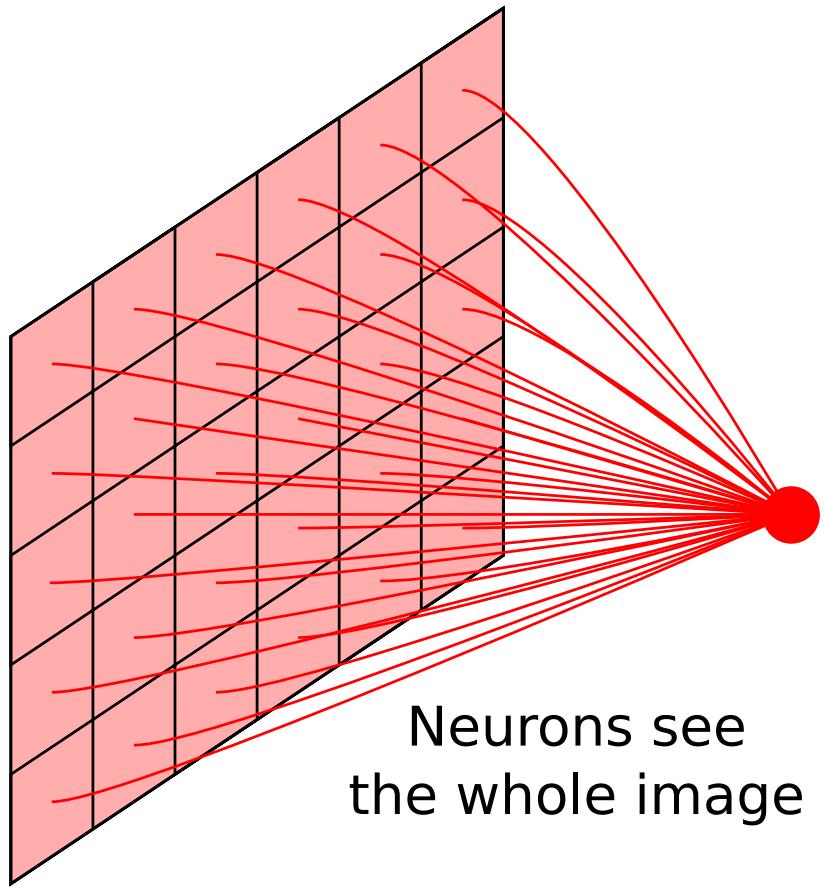
- Simpe to use
- Convergence proof
- Fast



Convolutions

Using a linear layer ?

- Lot of weights ! (at least one per pixel !)

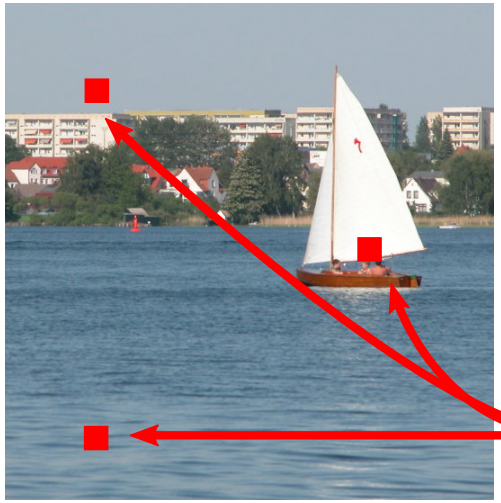


https://www.flickr.com/photos/nathelbiya/17050330

Convolutions

Using a linear layer ?

- Lot of weights ! (at least one per pixel !)
- Is it interesting to look at relations in the whole image ?

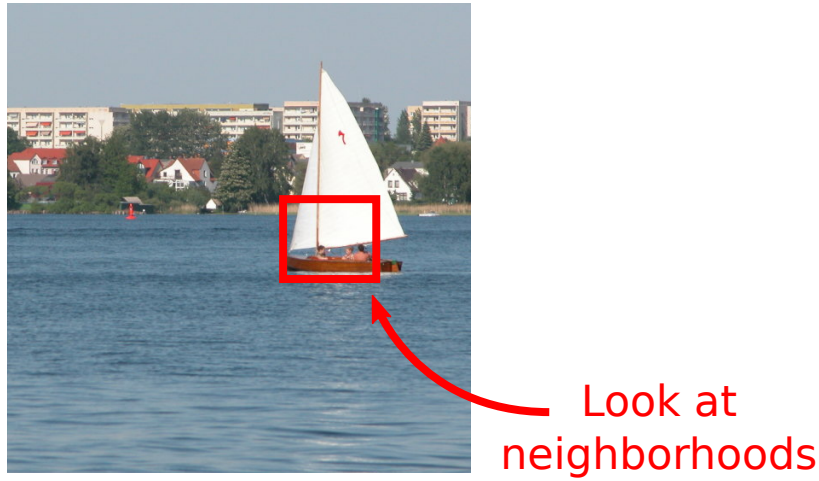


https://www.flickr.com/photos/nathelbiya/17050330

Convolutions

Idea ?

- Look at small neighborhoods (where the objects are)

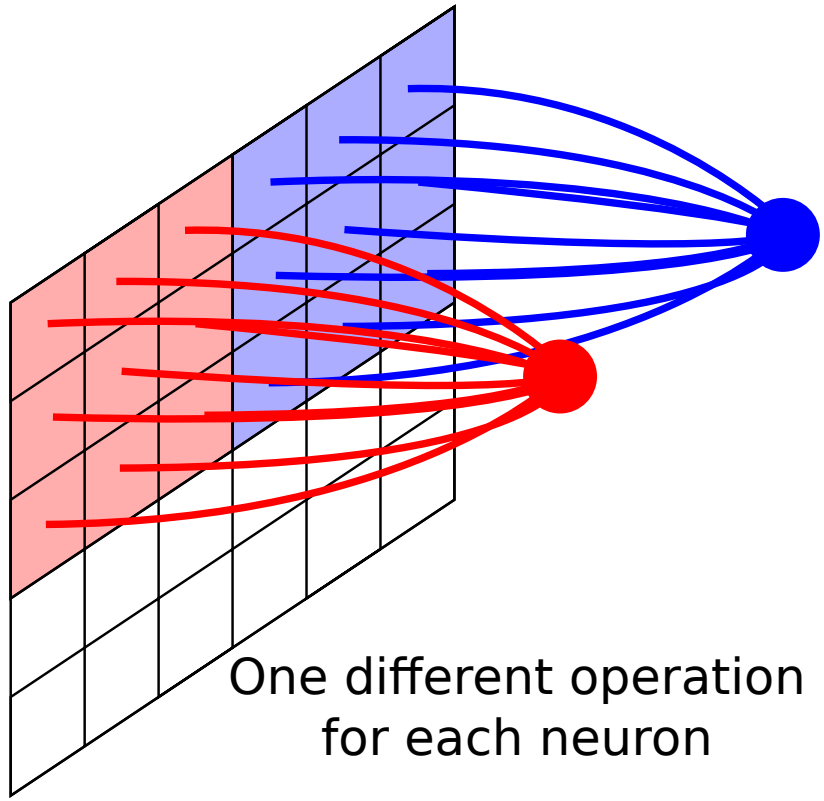


https://www.flickr.com/photos/nathelbiya/17050330

Convolutions

Idea ?

- Look at small neighborhoods (where the objects are)
- Create neurons that take a patch input



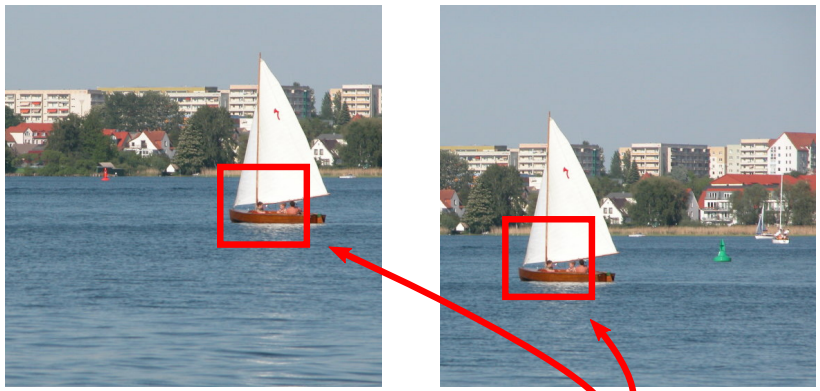
https://www.flickr.com/photos/nathelbiya/17050330

Convolutions

ONERA
THE FRENCH AEROSPACE LAB

Problem ?

- Translation of the object must lead to same behaviour of the neurons



Same behaviour for the same pattern at different location

https://www.flickr.com/photos/nathelbiya/17050330

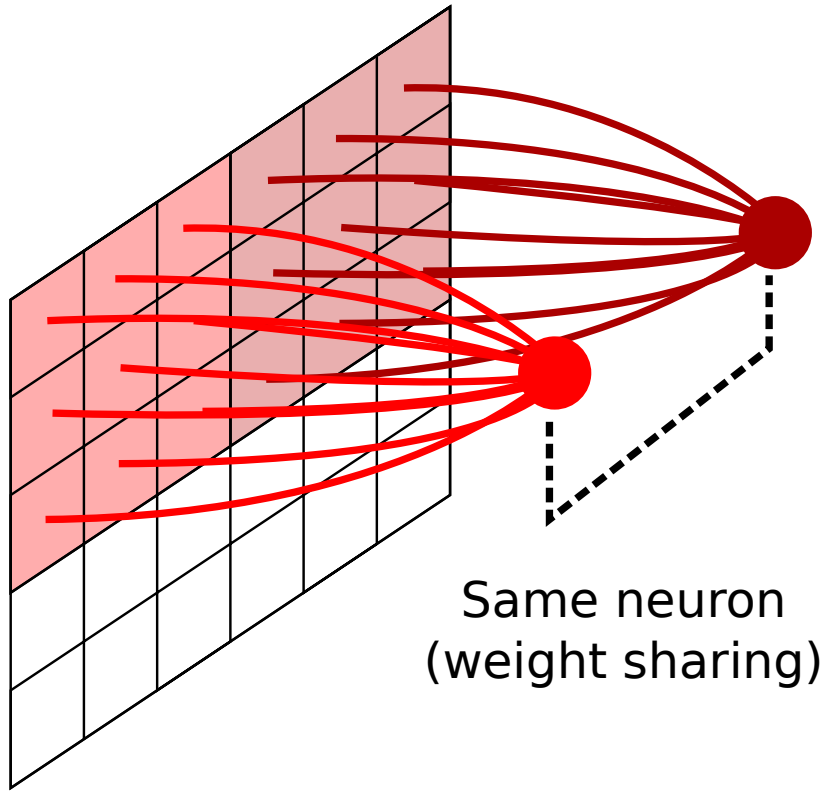
Convolutions

Problem ?

- Translation of the object must lead to same behaviour of the neurons

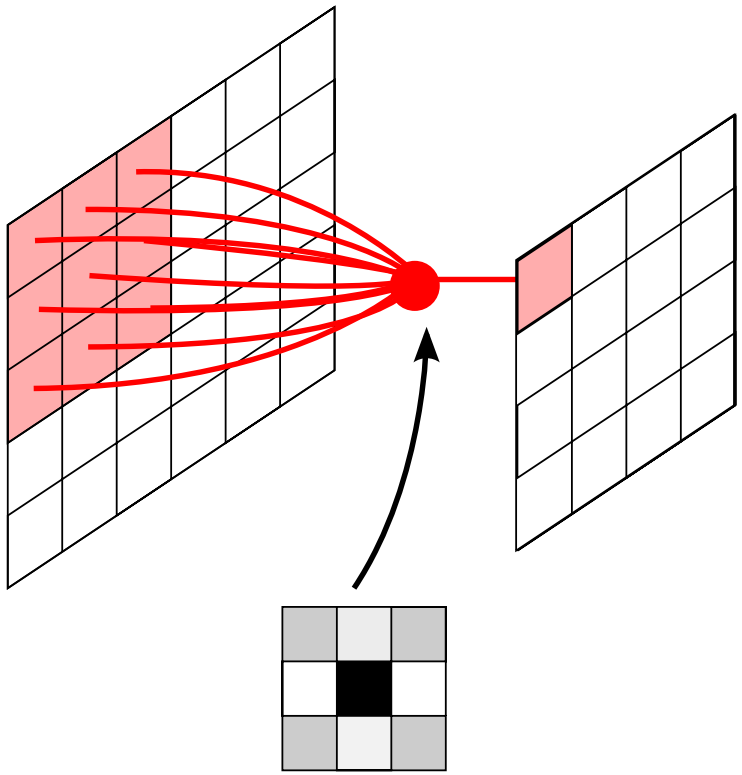
Solution ?

- Use the same neuron (i.e. all the neurons of the layer share weights)



https://www.flickr.com/photos/nathelbiya/17050330

Convolutions



Forward

Let (i, j) be the coordinates in the input map.
(k, l) be the size of the patch (size of the kernel, usually $k = l$)
Then:

$$y_{i,j} = \sum_k \sum_l w_{k,l} x_{i+k,j+l} + b$$

Convolution

ONERA
THE FRENCH AEROSPACE LAB

Backward weight update

$$\frac{\partial y_{i,j}}{\partial w_{k,l}} = x_{i+k,j+l} \tag{1}$$

Let y be the output map and Δy be the gradient coming back:

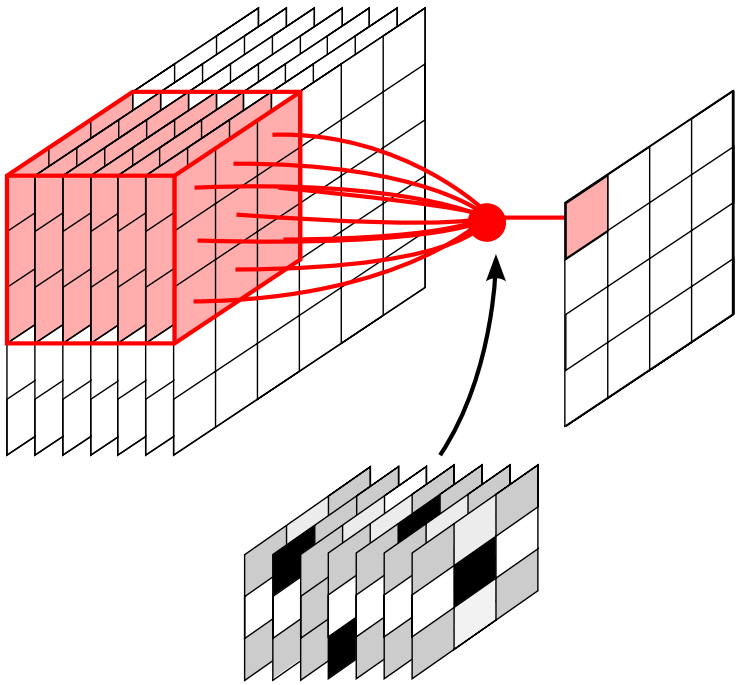
$$\frac{\partial y}{\partial w_{k,l}} = \sum_i \sum_j x_{i+k,j+l} \tag{2}$$

Finally, the update rule:

$$w_{k,l} \leftarrow w_{k,l} - \alpha \frac{\partial y}{\partial w_{k,l}} \Delta y \tag{3}$$

$$\leftarrow \sum_i \sum_j x_{i+k,j+l} \Delta y_{i,j} \tag{4}$$

Convolutions



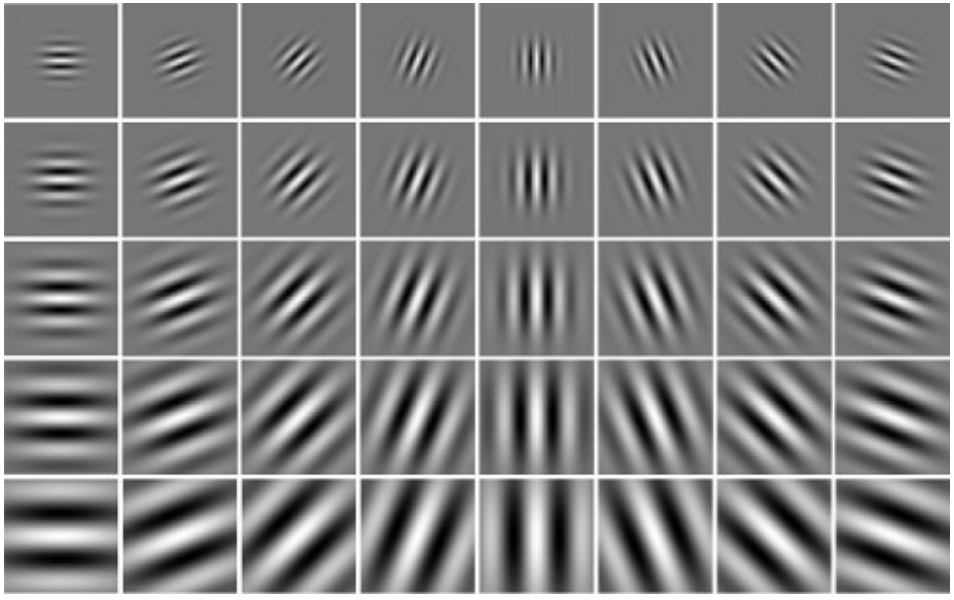
Forward

Same with term to term multiplication:

$$y_{i,j} = \sum_a \sum_b \mathbf{w}_{a,b} \mathbf{x}_{i+a,j+b}$$

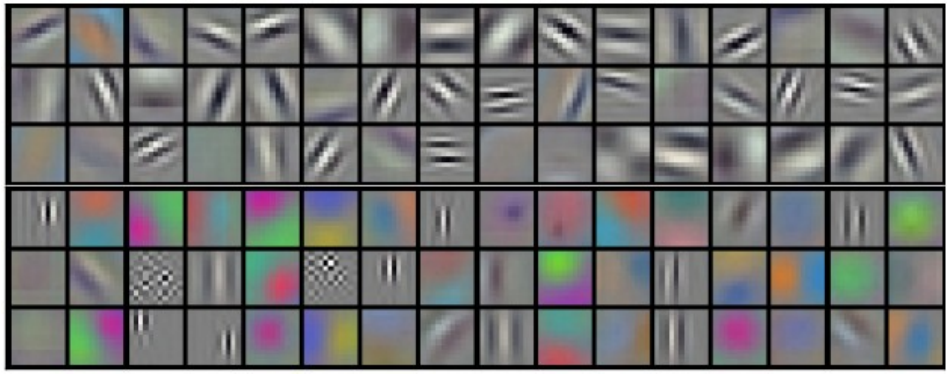
Convolution

What do convolutions learn ?



Gabor filters.

First layer of AlexNet.



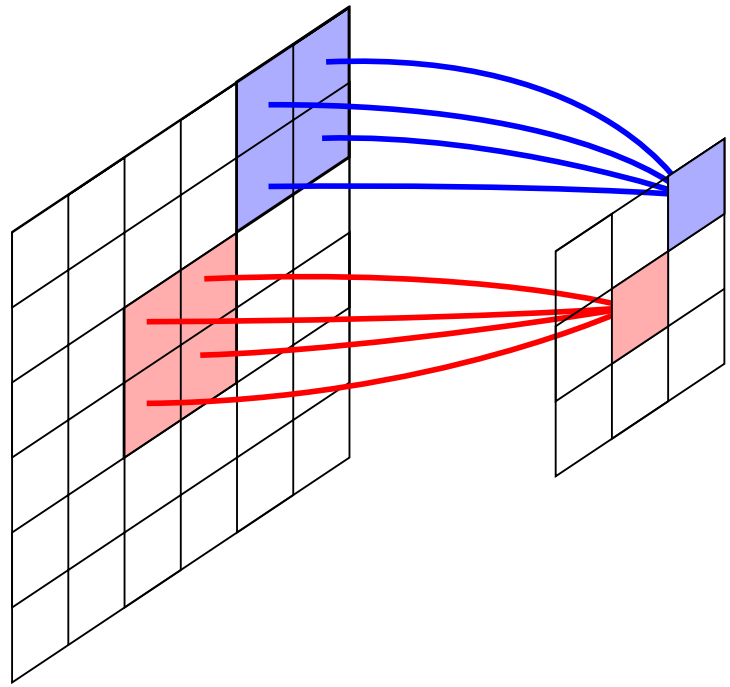
Dimension reduction

With the previous convolution, the output dimension is the same as the output dimension.
For classification: only one label, need for **dimension reduction**.

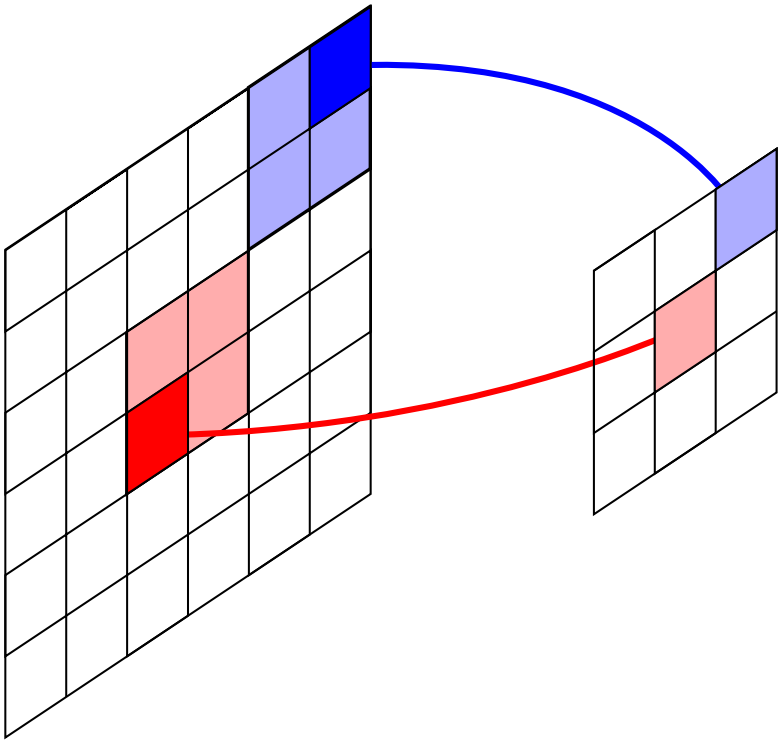
- convolution stride: do not look at all the pixels of the input (one every two, one every three...)
- **Max Pooling**

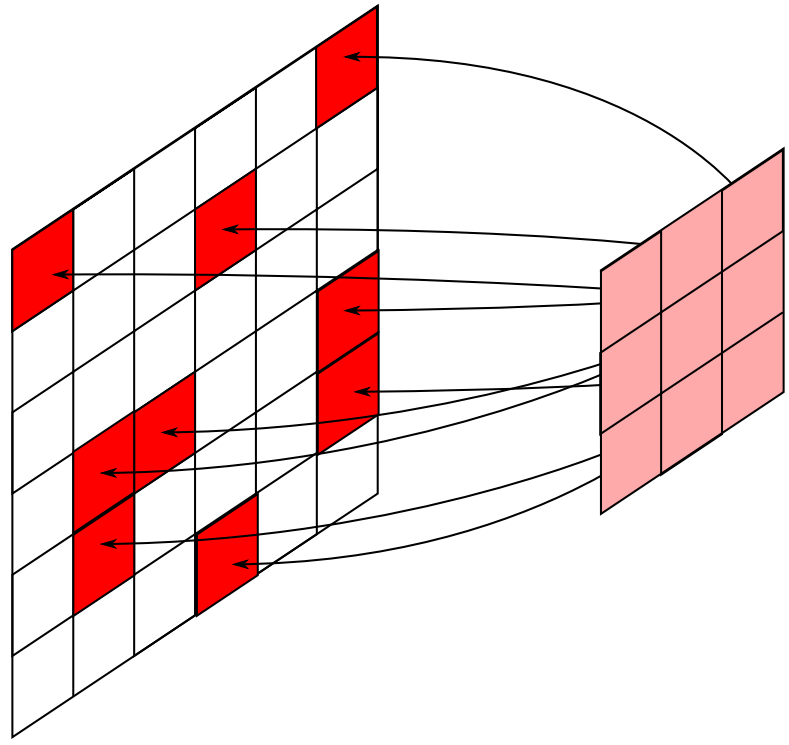
Max Pooling

- dimension reduction
- relative translation invariability



Max Pooling





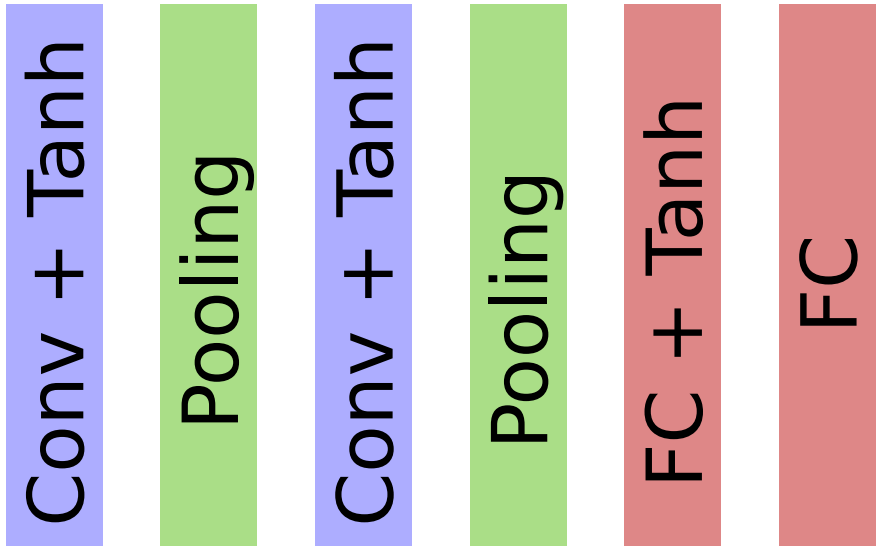
Forward: max signal

Backward: Gradient transmission to max signal origin, zero otherwise

LeNet (1990)

ONERA
THE FRENCH AEROSPACE LAB

LeNet (1990)
Images 28x28



Very good results on digits recognition !

Issues

ONERA
THE FRENCH AEROSPACE LAB

Issues

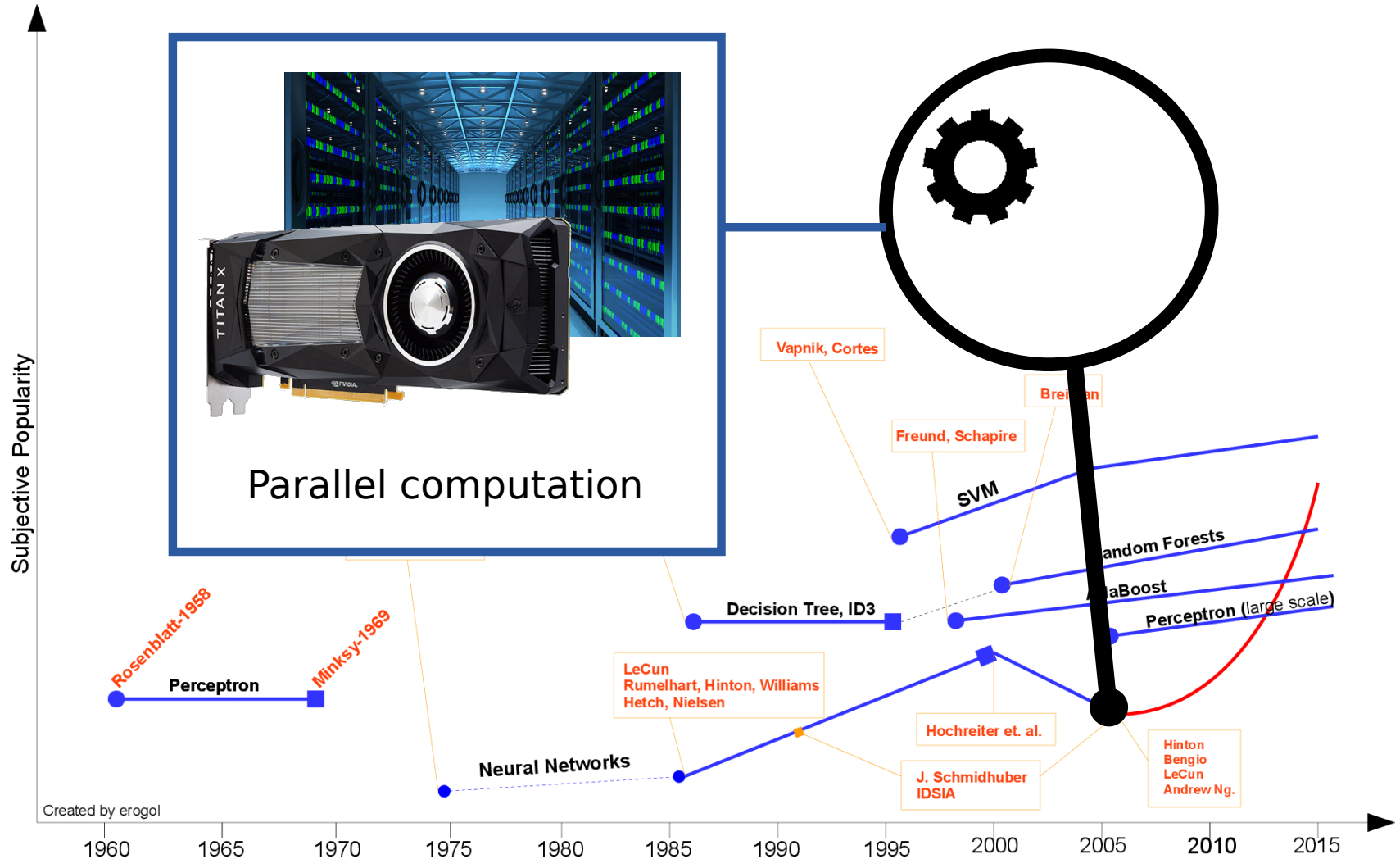
- Learning speed
- Exploding or vanishing gradients
- Overfitting
- Local minima

Limitations

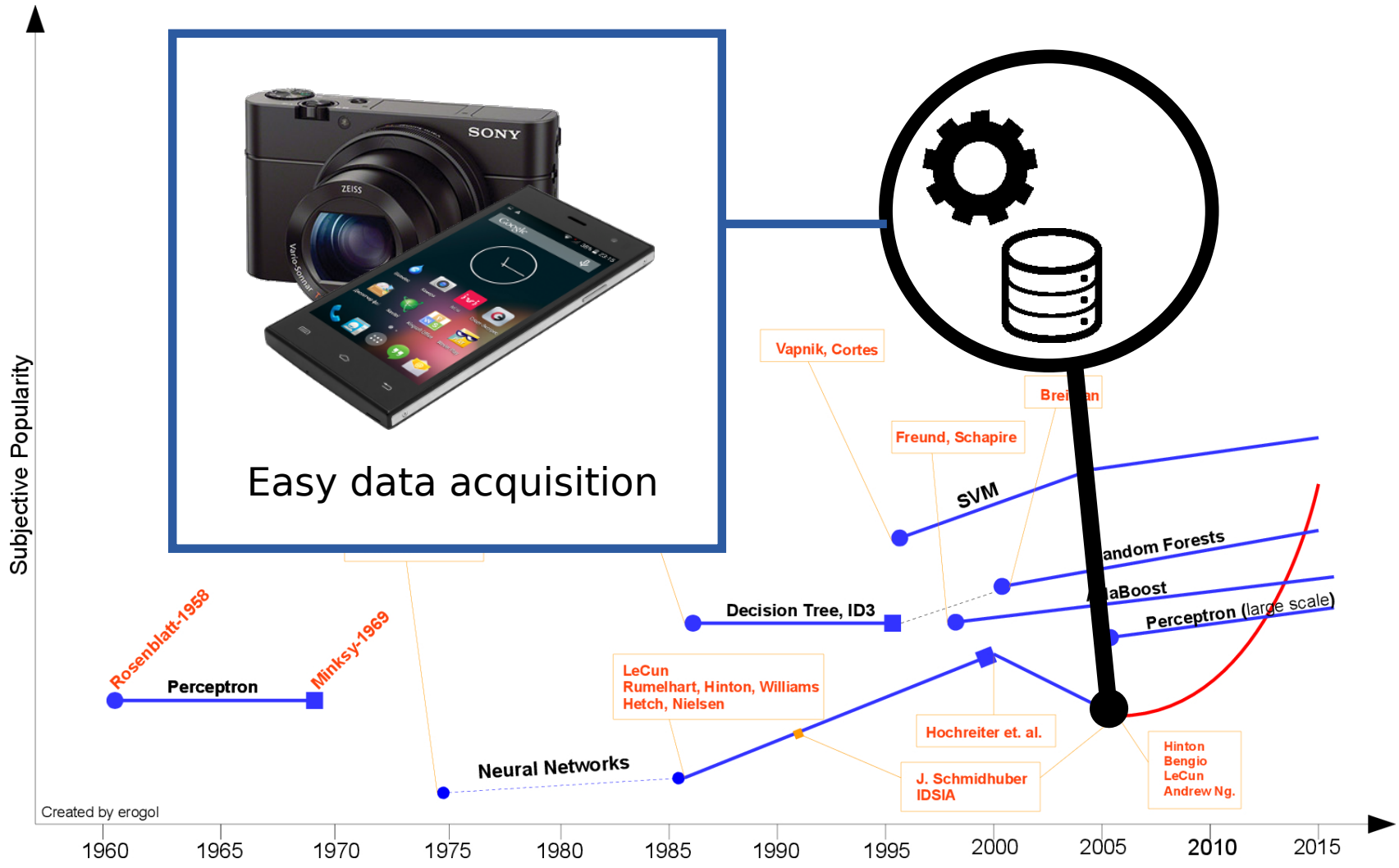
- Architecture
- Initialization
- Computing power
- Data
- Optimization

Table of contents

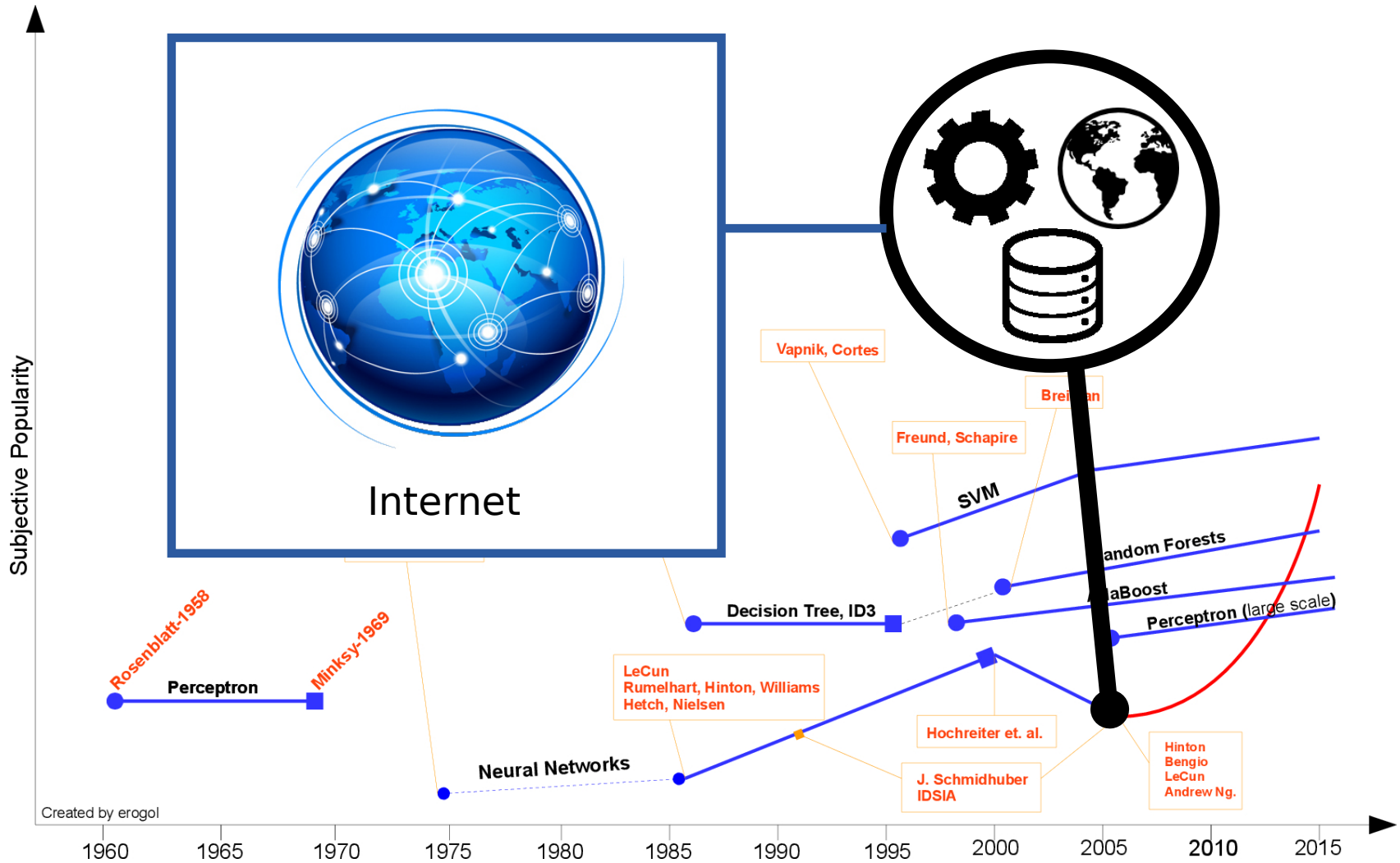
Massively data driven approaches



Massively data driven approaches



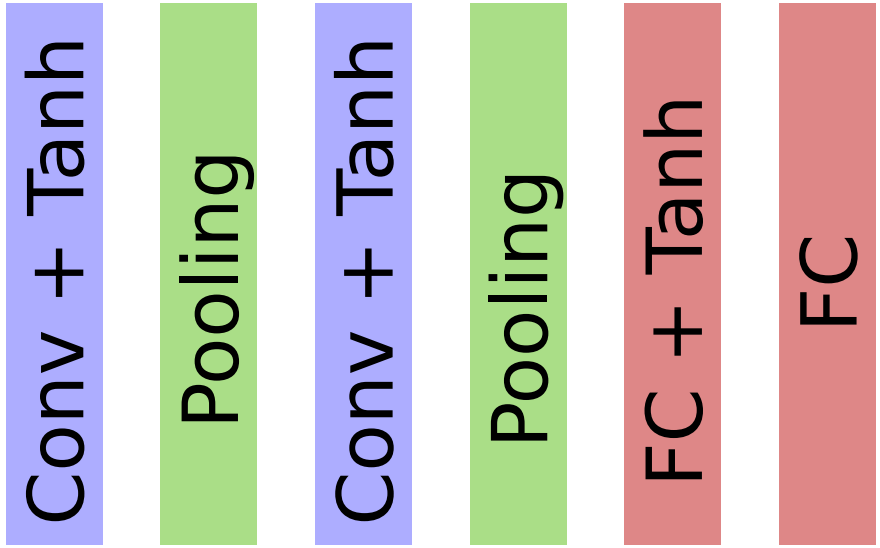
Massively data driven approaches



LeNet (1990)

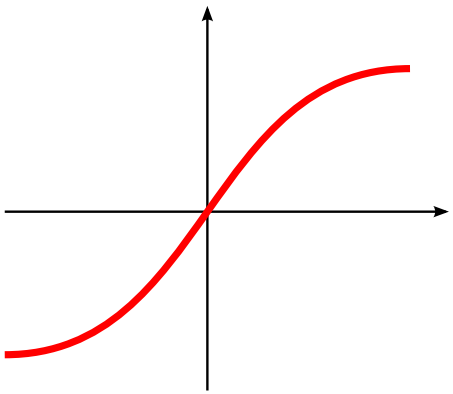
ONERA
THE FRENCH AEROSPACE LAB

LeNet (1990)
Images 28x28

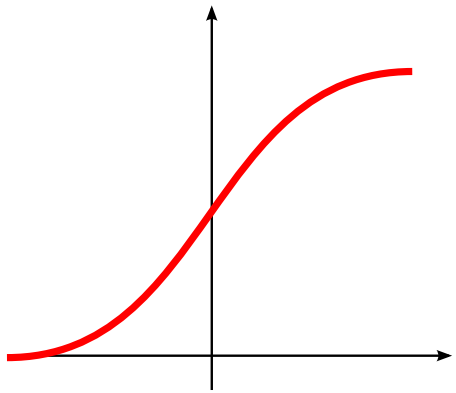


After each linear transformation (linear layer or convolution layer): a non linear layer (activation) otherwise **linear + linear = linear** (impossible to learn non linear decision).

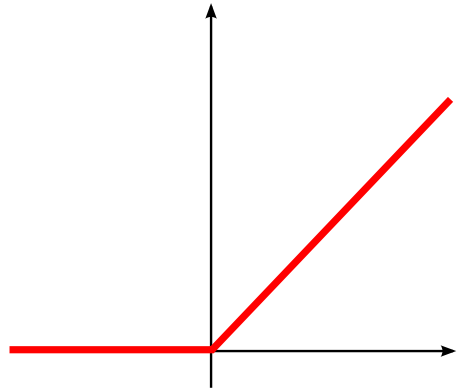
Activations



Tangente hypebolique



Sigmoïde

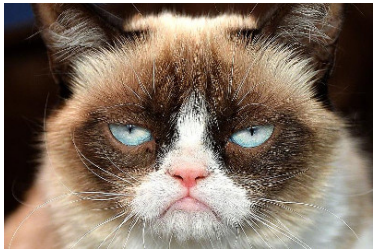


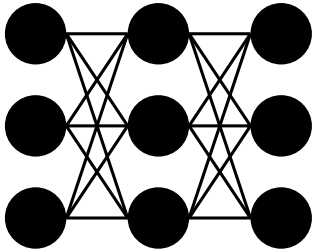
Rectified Linear Unit (ReLU)

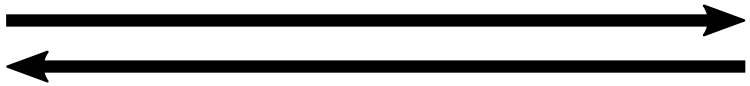
Rectified linear unit

1. Faster gradient computation
2. Similar convergence

Mini-batches

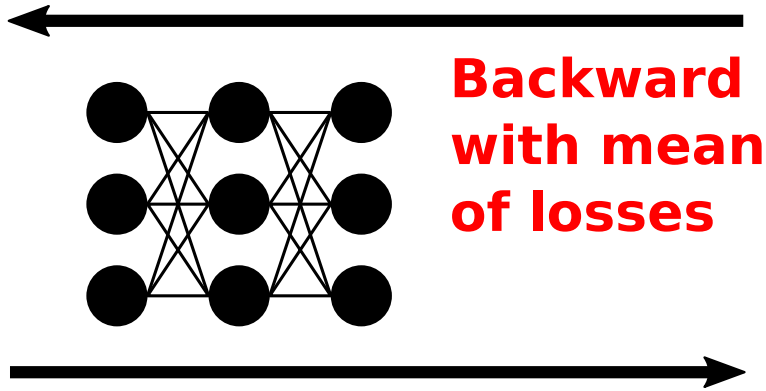






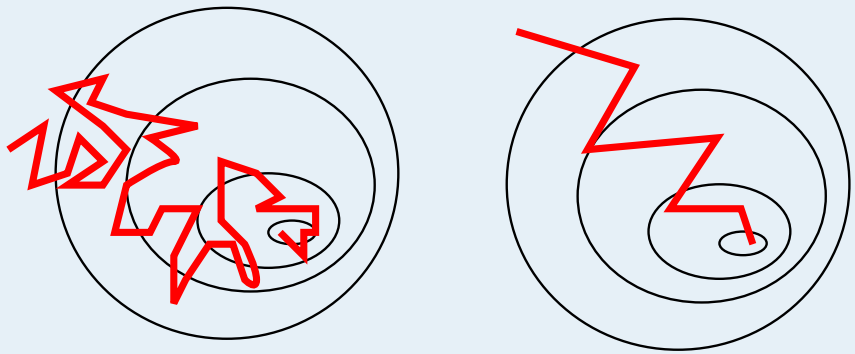
Loss image 1

Mini-batches



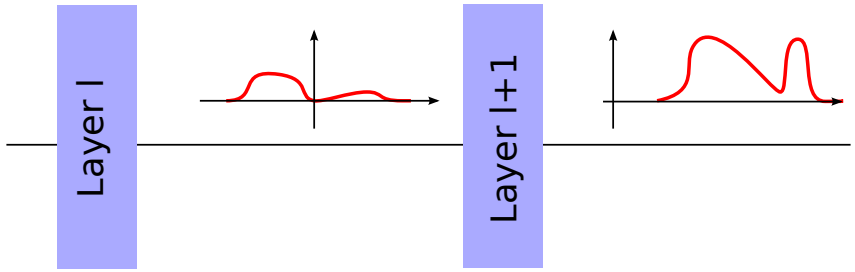
Loss image 1
Loss image 2
Loss image 3
Loss image 4
...

Gradient smoothing



Smoother gradient converges faster.

BatchNorm



Changes in the signal dynamic make the model more difficult to optimize: exponential or vanishing gradients.
Objective: control the signal distribution:

$$y^{l*} = \frac{y^l - \mu}{\sqrt{\sigma^2 + \epsilon}}\gamma + \beta \tag{5}$$

γ and β are learnt, μ and σ are computed (mean and standard deviation).
Learning is faster (iteration number) but slower (statistics computation).

Weight initialization

Weights have great influence on convergence speed. They are randomly initialized.

- too small weigths: vanishing signal
- to high: exploding signal

Objective

Conservation of signal properties.

$$Var(Y) = Var(X)$$

Weight initialization

ONERA
THE FRENCH AEROSPACE LAB

Xavier initialization

$X \in \mathbb{R}^n$, weights W and output $Y \in \mathbb{R}$

$$Y = W_1X_1 + W_2X_2 + \cdots + W_nX_n$$

X_i and W_i independent:

$$Var(W_iX_i) = E[X_i]^2 Var(W_i) + Var(X_i)Var(W_i) + Var(X_i)E[W_i]^2$$

$E[X_i] = 0$ and $E[W_i] = 0$:

$$Var(W_iX_i) = Var(X_i)Var(W_i)$$

finally $Var(Y) = nVar(X_i)Var(W_i)$ and we chose

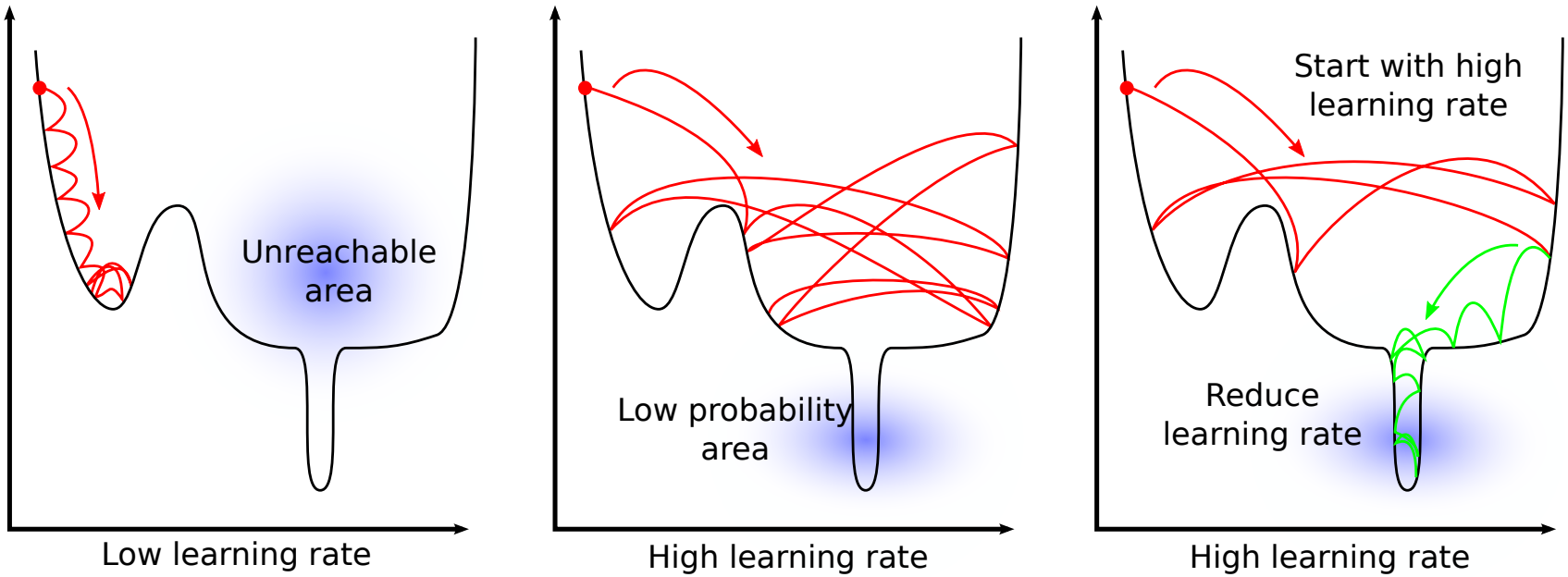
$$Var(W_i) = \frac{1}{n}$$

Optimization - Stochastic Gradient Descent

ONERA
THE FRENCH AEROSPACE LAB

See neural network class

Update rule: $w_{t+1} = w_t + \alpha \Delta w$ (learning rate α)



Optimization - Stochastic Gradient Descent

ONERA
THE FRENCH AEROSPACE LAB

See neural network class

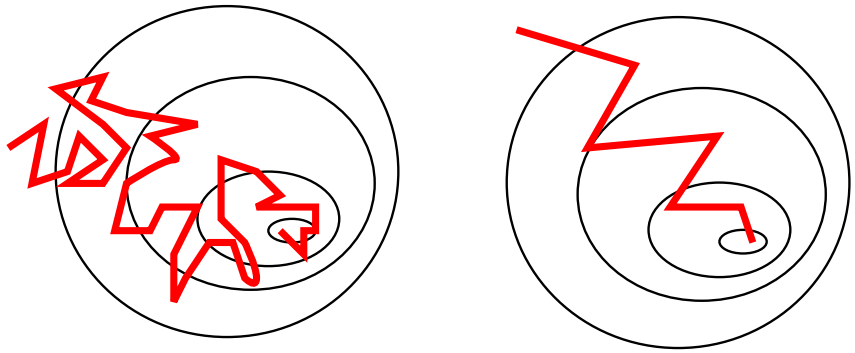
Update rule: $w_{t+1} = w_t + \alpha \Delta w$ (learning rate α)

Learning rate variation

- **Step decrease**
- Exponential decrease
- ...

Optimization - SGD with Momentum

Same idea as mini batch: smooth gradient in the good direction



Momentum

Use previous gradient to ponderate the direction of the new gradient.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \alpha \Delta w \\ w_t &= w_{t-1} - v_t \end{aligned}$$

γ is the momentum.

Optimization - Adaptative methods

- **Adagrad**

$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{G_{t,i} + \epsilon}} \Delta w_i$$

with $G_{t,i}$ sum of gradients of w_i

- **Adam (adaptive momentum estimation)**

$$\begin{aligned} m_t &= \frac{\beta_1}{1-\beta_1} m_{t-1} + \Delta w_t \\ v_t &= \frac{\beta_2}{1-\beta_2} v_{t-1} + \Delta w_t^2 \end{aligned}$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} m_t$$

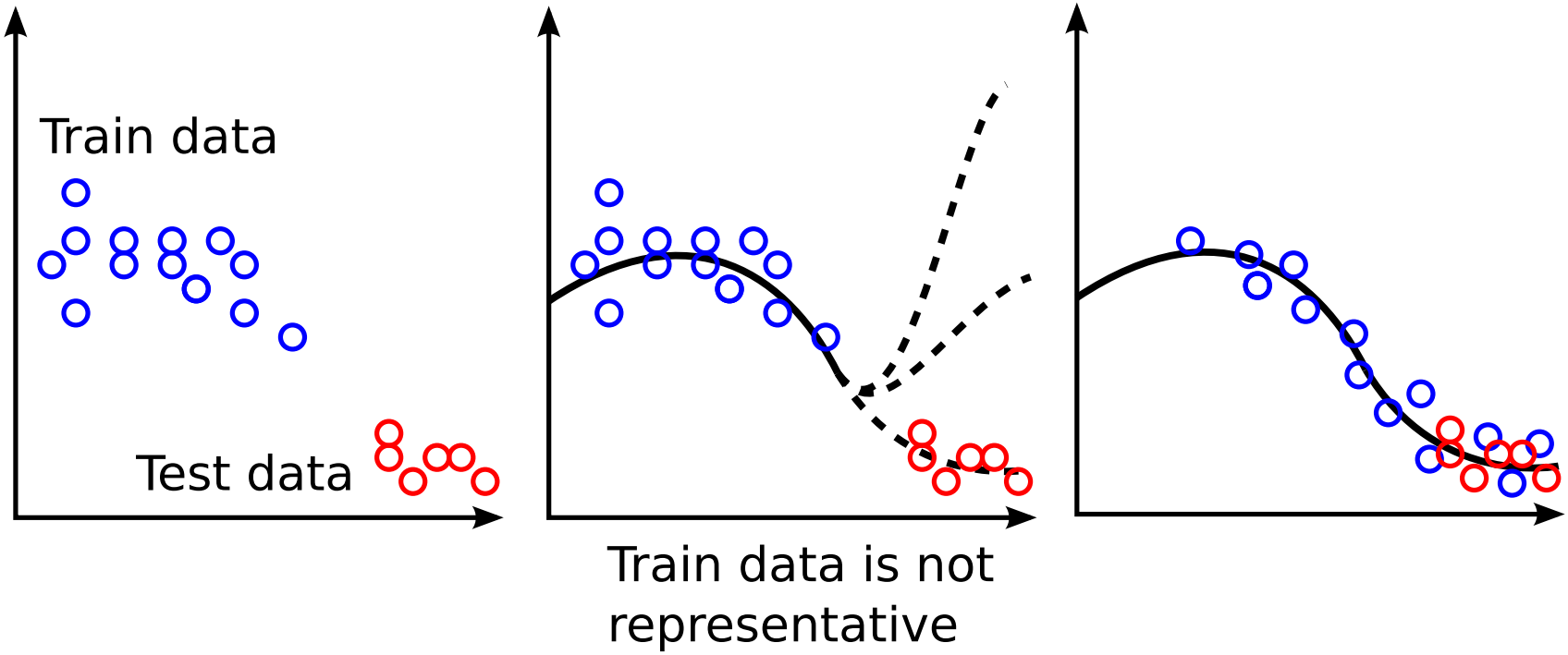
Table of contents

Data

- Representative
- Data augmentation
- Data normalization

Data

Train data must be representative of the problem



Data

ONERA
THE FRENCH AEROSPACE LAB

Data normalization

- Compute mean μ and standard deviation σ on the train set.
- Normalize input I (train and test):

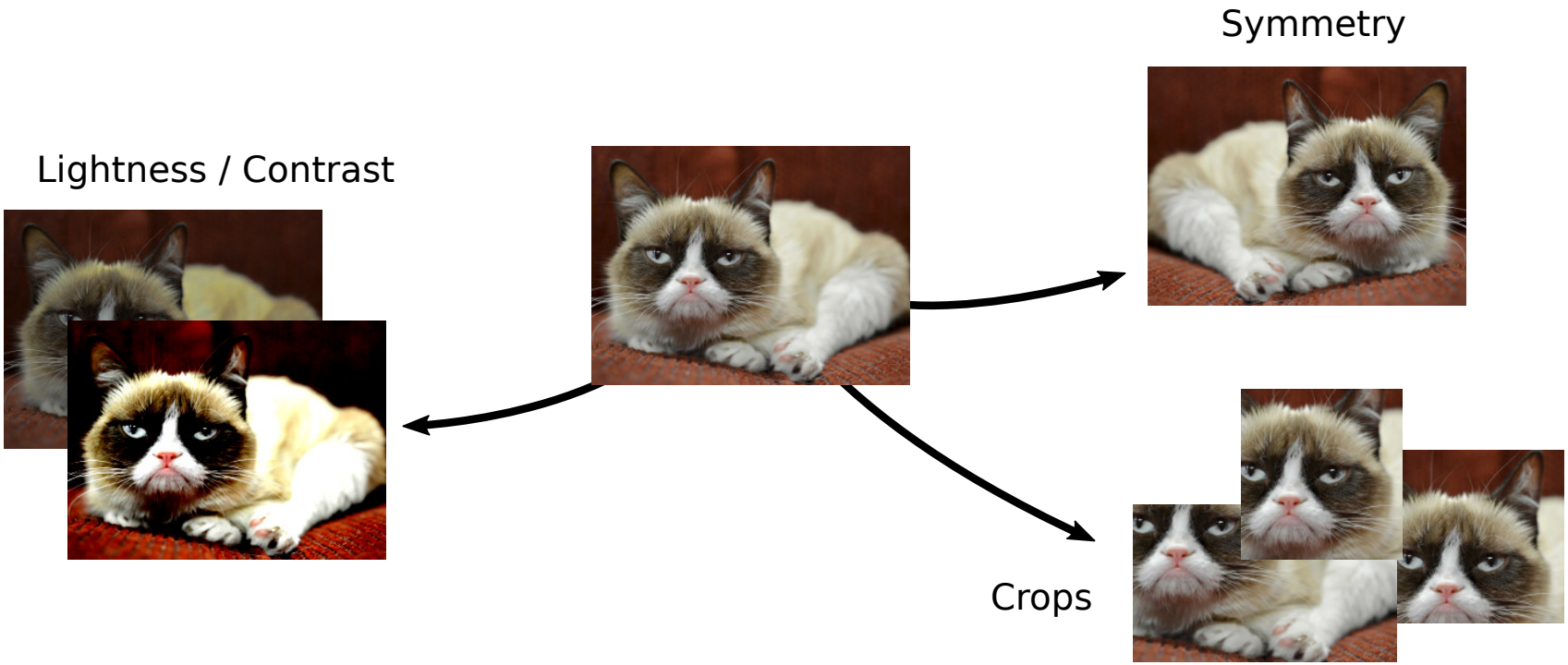
$$\hat{I} = \frac{I - \mu}{\sigma}$$

Data

Data augmentation

Random variations of input parameters (images: lightness, contrast ...)

- train on a more representative set
- avoid learning on unwanted features



Problems and partial solutions

ONERA
THE FRENCH AEROSPACE LAB

Problems

- Small amount of data
- Low computational power

Solutions ?

- Use classical approaches (Perceptron, SVM, . . .)

Table of contents

Image classification - LeNet vs VGG

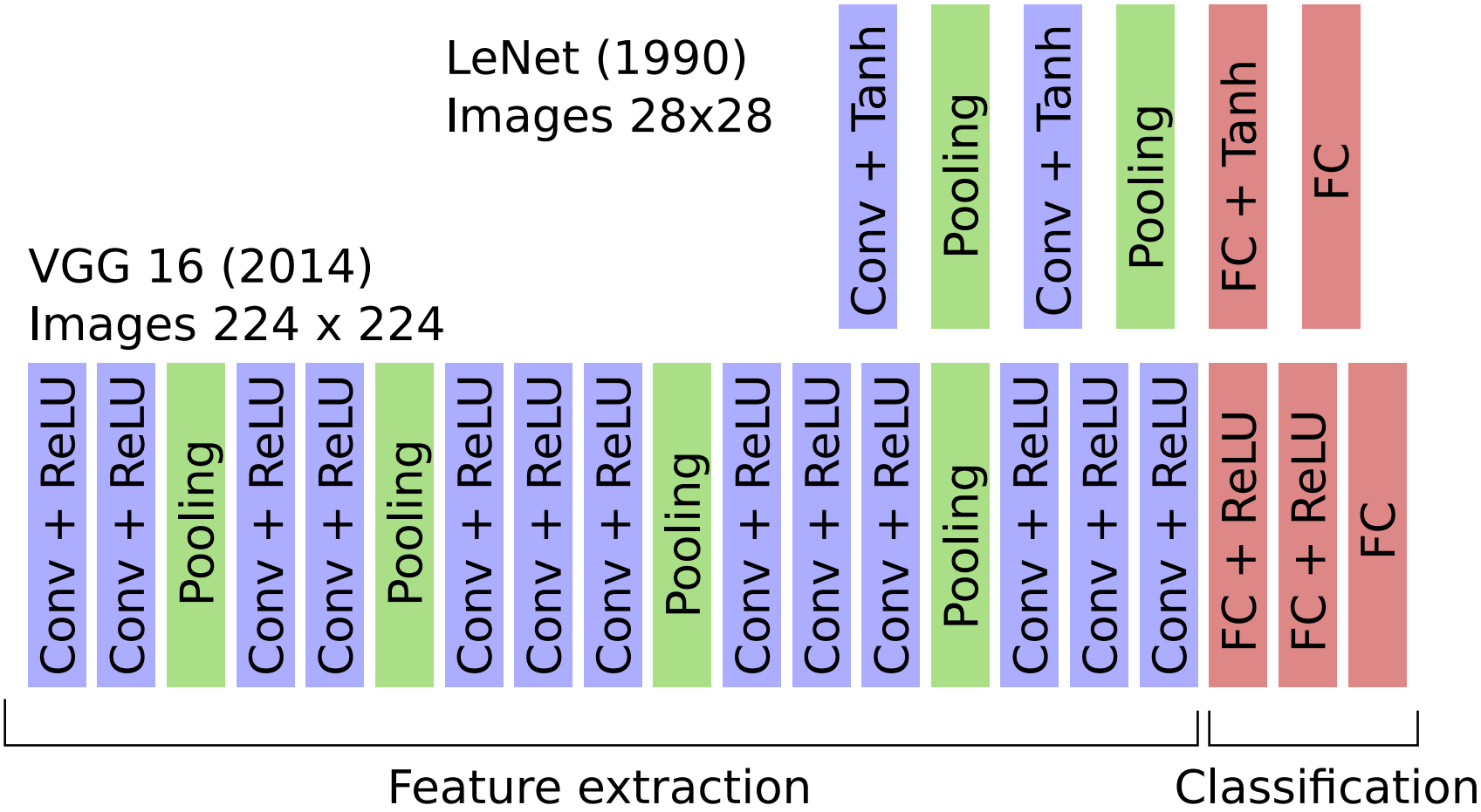
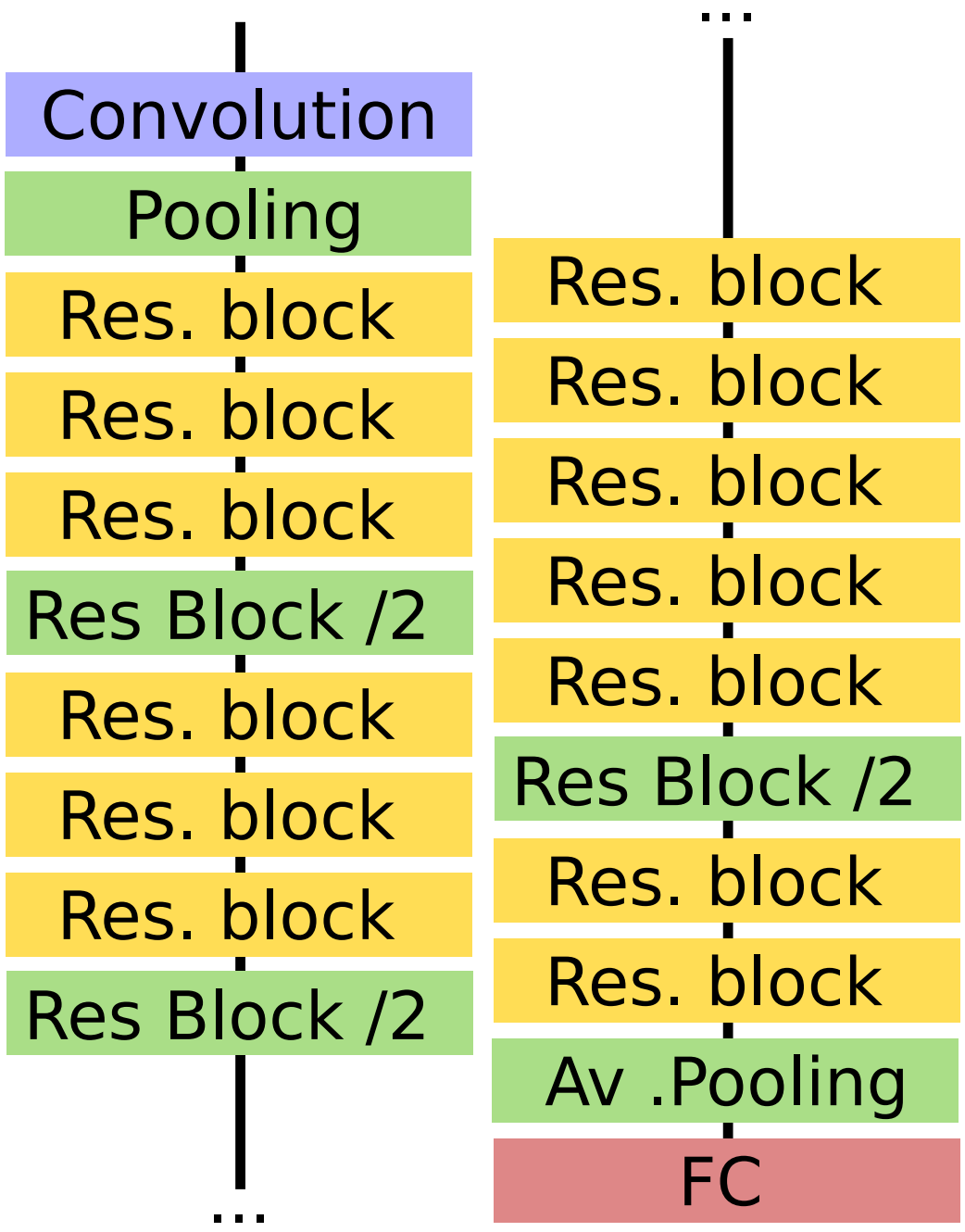
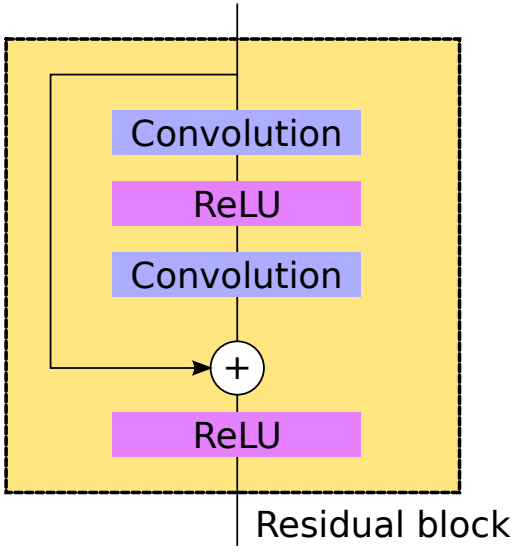


Image classification - ResNet





Semantic segmentation

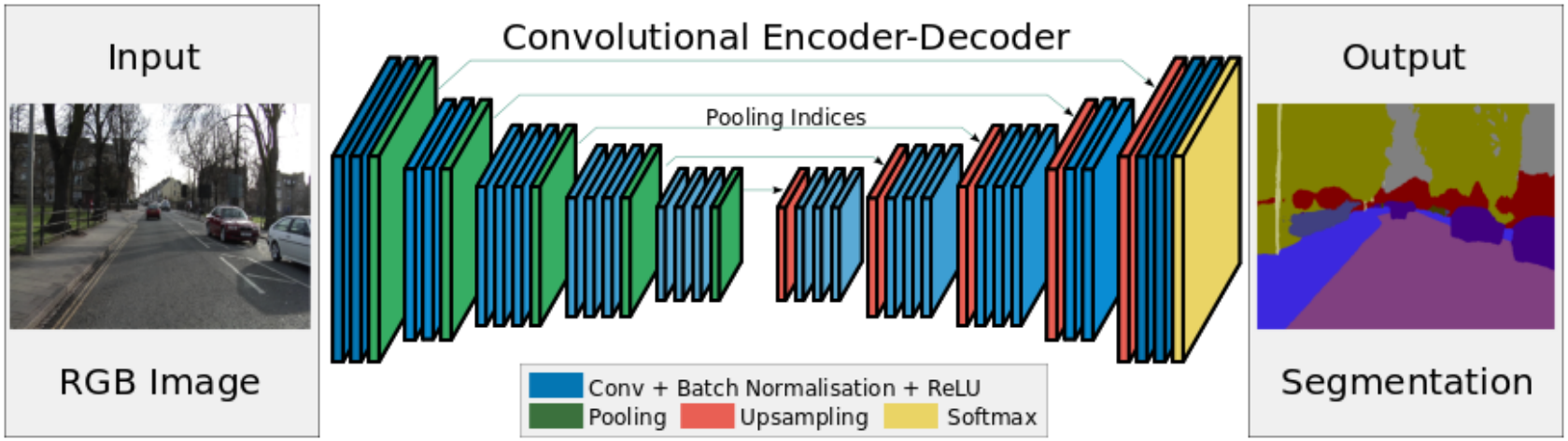
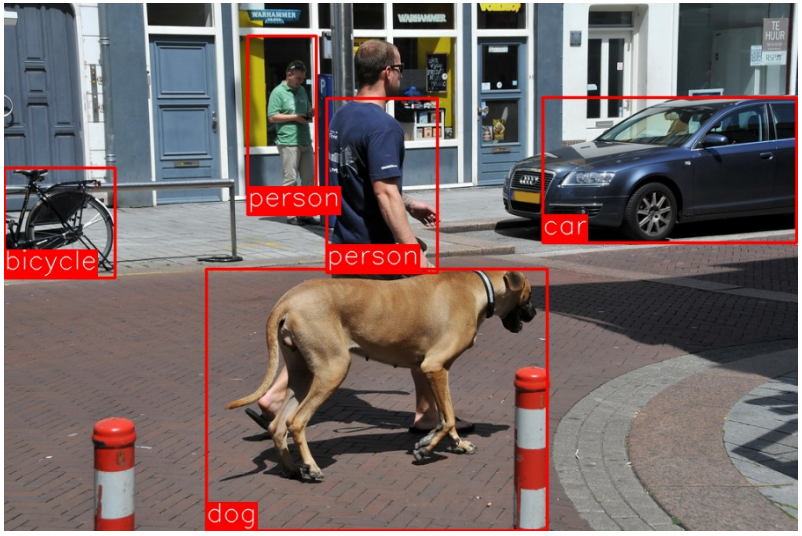
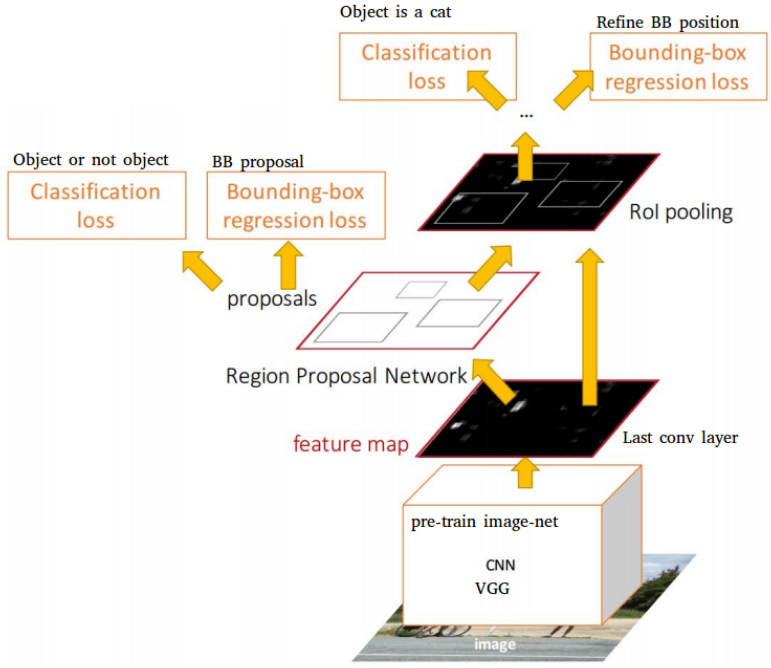


Image : http://mi.eng.cam.ac.uk/projects/segnet/

Detection


bicycle
person
person
car
dog

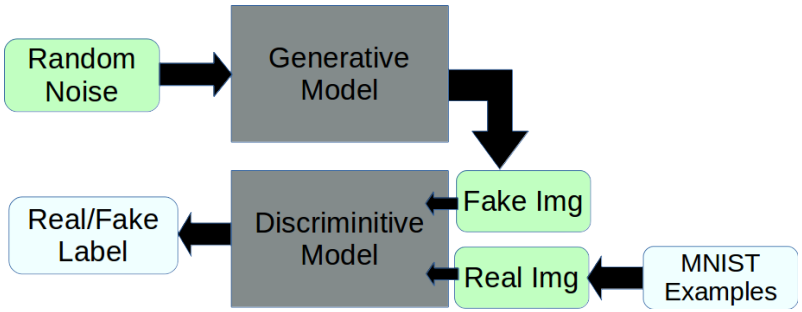

Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
...
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
Last conv layer
pre-train image-net
CNN
VGG
image

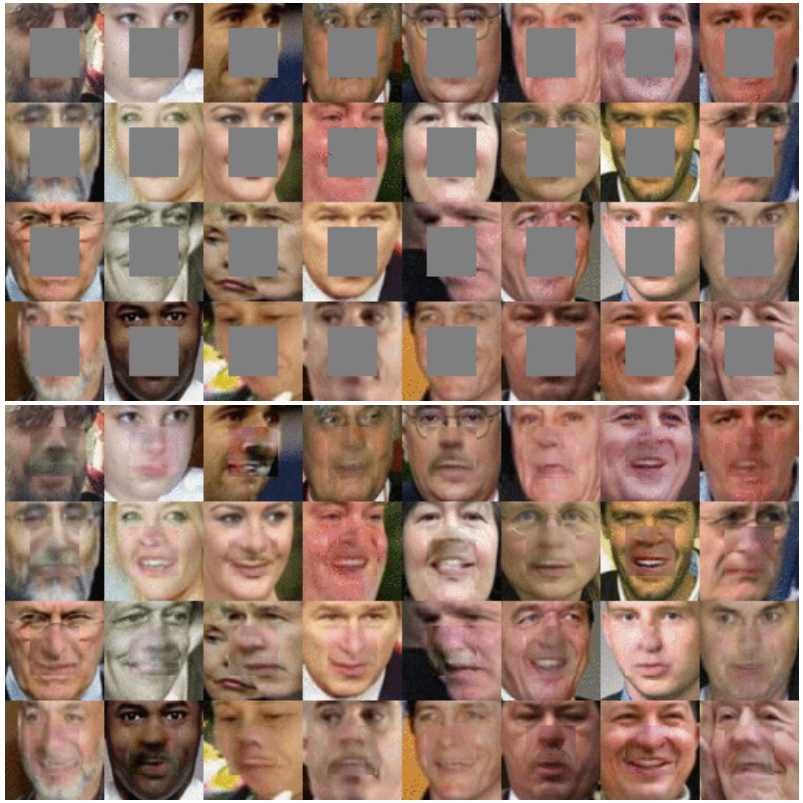
Data generation

ONERA
THE FRENCH AEROSPACE LAB

Generative Adversarial Networks

2 networks: one for data generation, one trying to discriminate real and generated data.

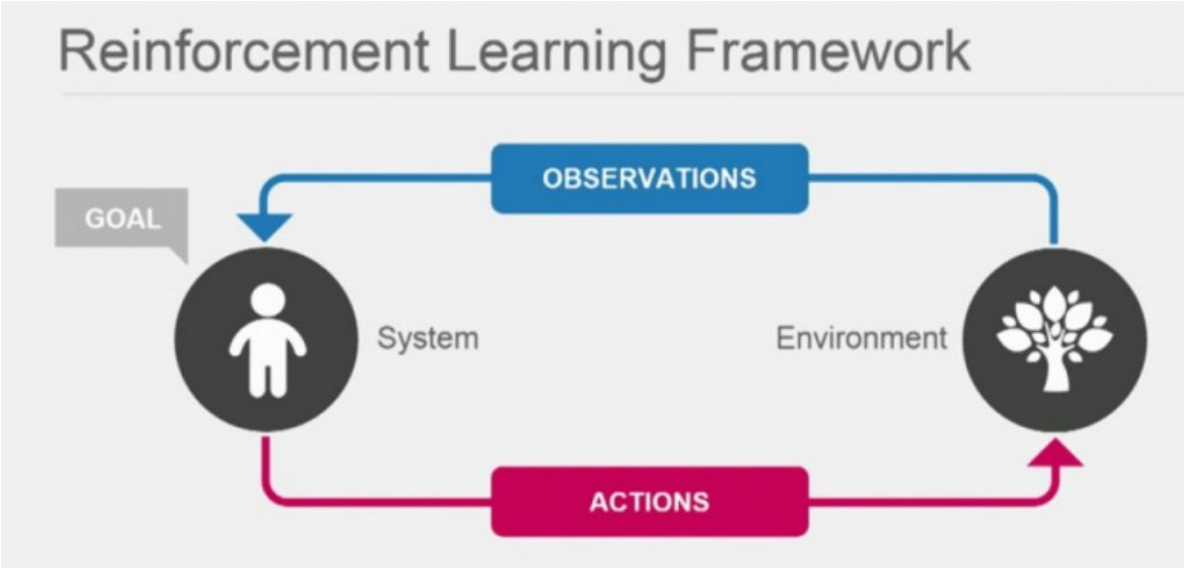




Reinforcement learning

Try and error learning

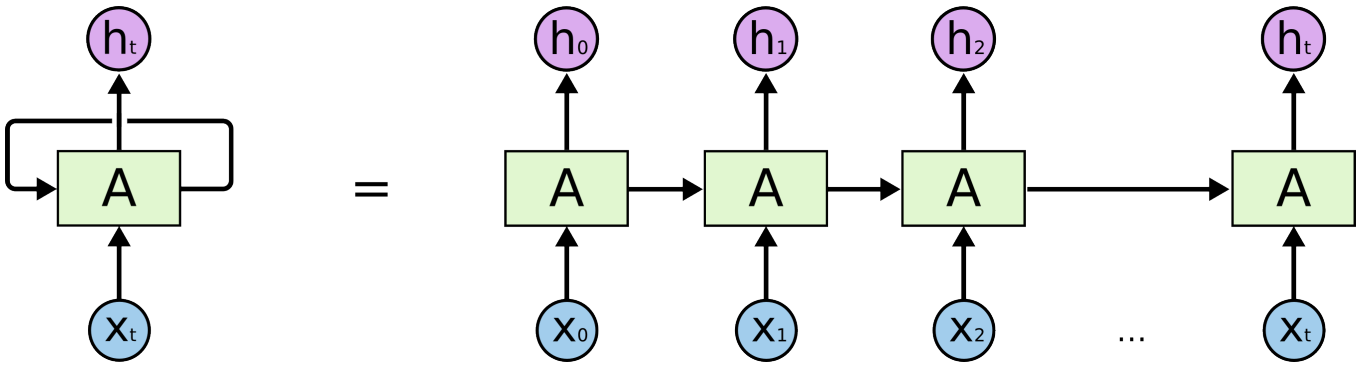
Reward and Penalty.



1

[1] Image http://visteon.bg/2017/03/02/machine-learning-algorithms-autonomous-cars/

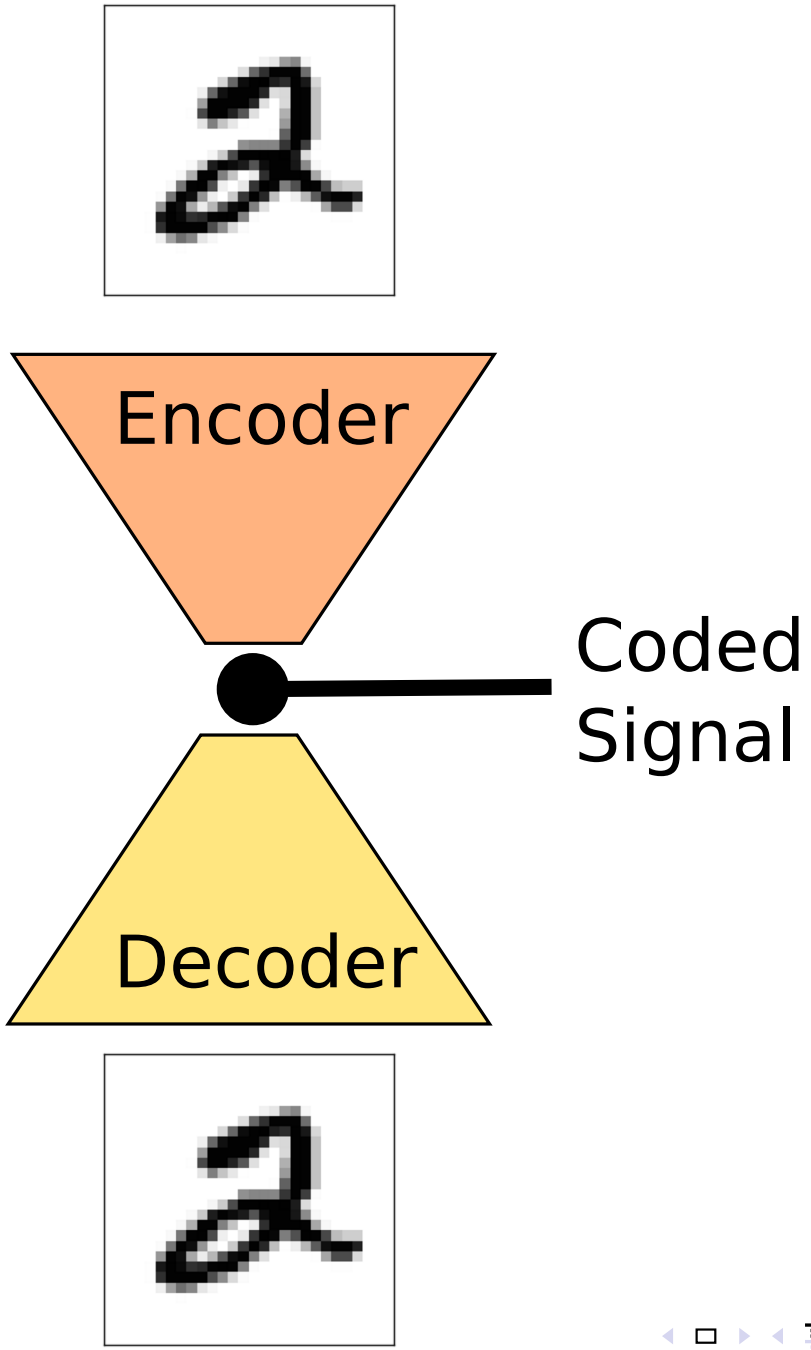
Recurrent neural networks



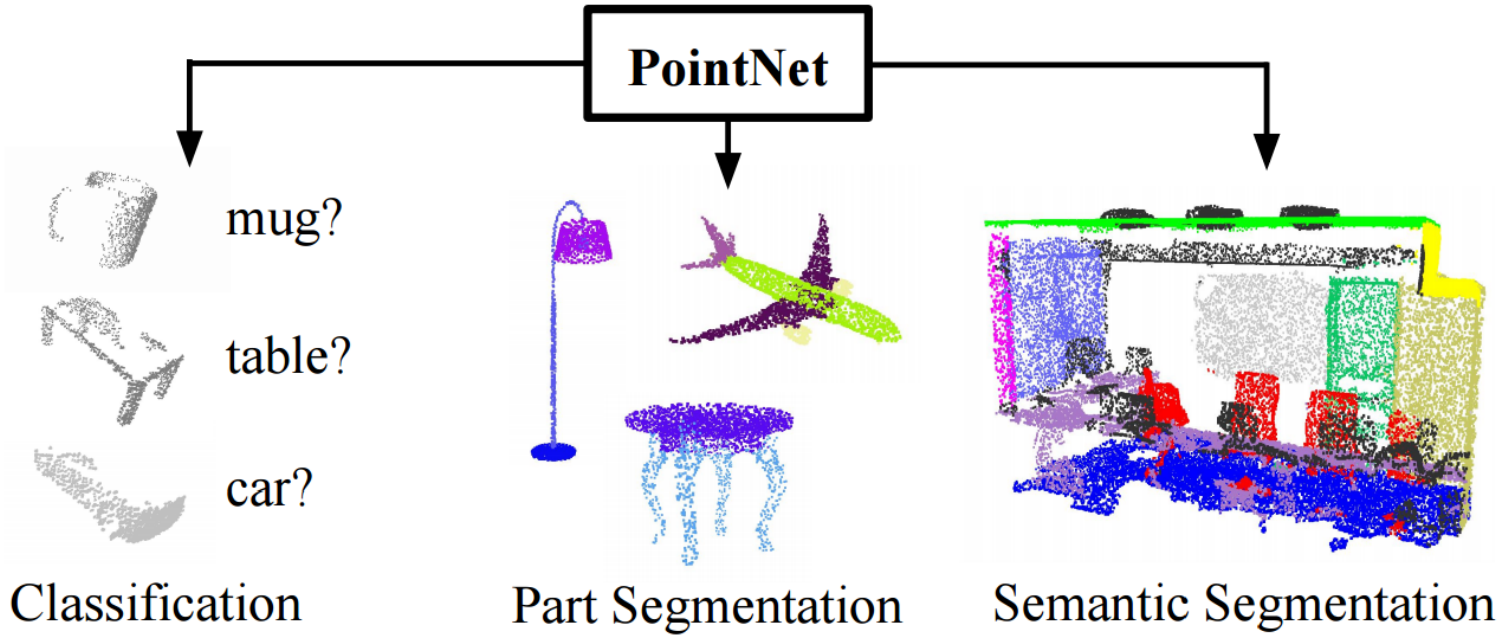
Dealing with sequences

Apply the same operation on each data and use a memroy to propagate information in time.

Autoencoders



3D data



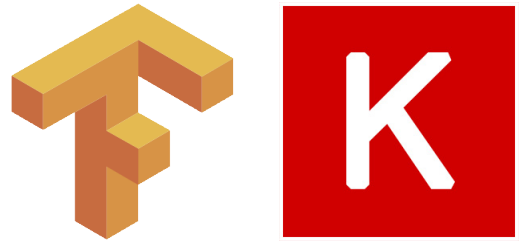
- Use voxels (Similar to images)
- Take Snapshots of the point cloud (SnapNet)
- Deal directly with point clouds (PointNet, PointNet++ ...)

https://github.com/charlesq34/pointnet

Table of contents

Frameworks

Torch



Tensorflow and Keras

Principaux frameworks Python

Pytorch and Tensorflow: CPU and GPU, good maintenance, tutorials.
Keras run on top of Tensorflow (and Theano, CNTK...). It provides a high level API.

Conclusion: some open problems

- Learn physics law ?
  (gravity...)

https://www.cos-mag.com/

Conclusion: some open problems

- Learn physics law ? (gravity...)
- Do it unsupervided ?

https://www.cos-mag.com/

Conclusion: some open problems

- Learn physics law ? (gravity...)
- Do it unsupervided ?
- Include physical knowledge in the machine learning ?

https://www.cos-mag.com/

Conclusion: some open problems

- Learn physics law ? (gravity...)
- Do it unsupervided ?
- Include physical knowledge in the machine learning ?
- Go from recognition to real reasoning ?

https://www.cos-mag.com/

Conclusion

ONERA
THE FRENCH AEROSPACE LAB

- Fast overview
- Few applications
- Practical work: create a CNN for image classification

Projects

Starting next week: projects.

- group of 2
- 3 subjects

For next week

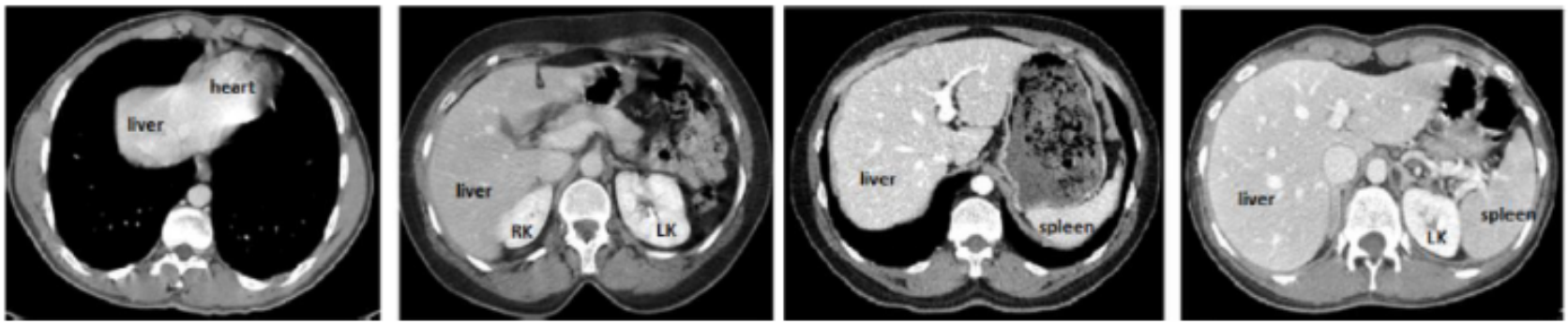
- form working pair
- choose subject

Project 1: dehazing

URL

https://competitions.codalab.org/competitions/21163

Project 2: medical images segmentation



URL `https://chaos.grand-challenge.org`

Project 3: trafic sign recognition

ONERA
THE FRENCH AEROSPACE LAB

- Classification
- 40+ classes
- 50000 images

URL `http://benchmark.ini.rub.de/?section=gtsrb&subsection=news`

Project Evaluation

Code (notebook) + **Report** (can be in the notebook)

Evaluation criteria

- Project execution (code)
- Justification of algorithmic choices
- Methodology
- Results analysis

Mid-course evaluation

Next class: **1h evaluation**

- All notes and course material
- **Exept for Neural networks and deep learning**
  $\longrightarrow$ knowledge and understanding test