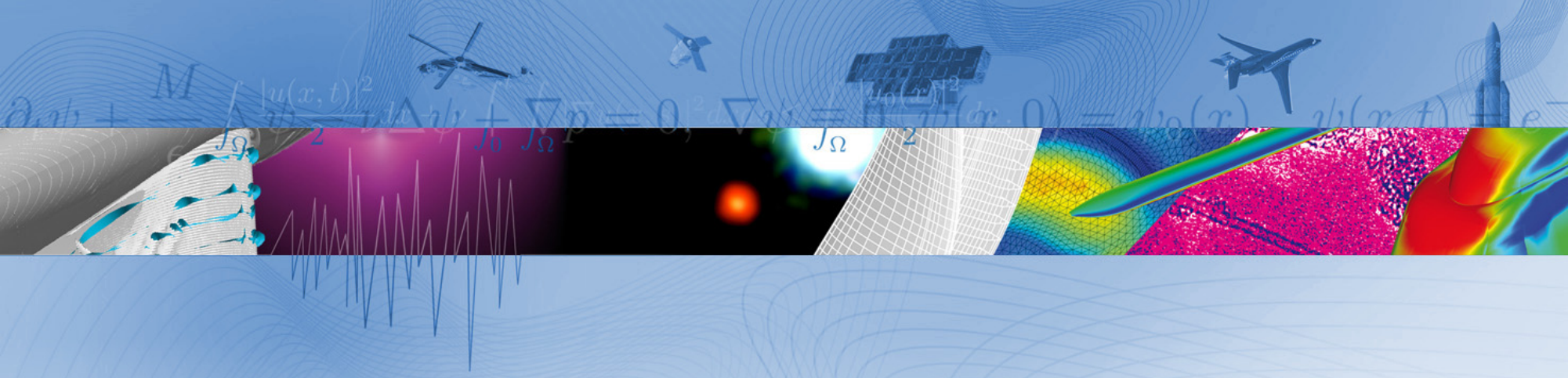


Reconnaissance des formes

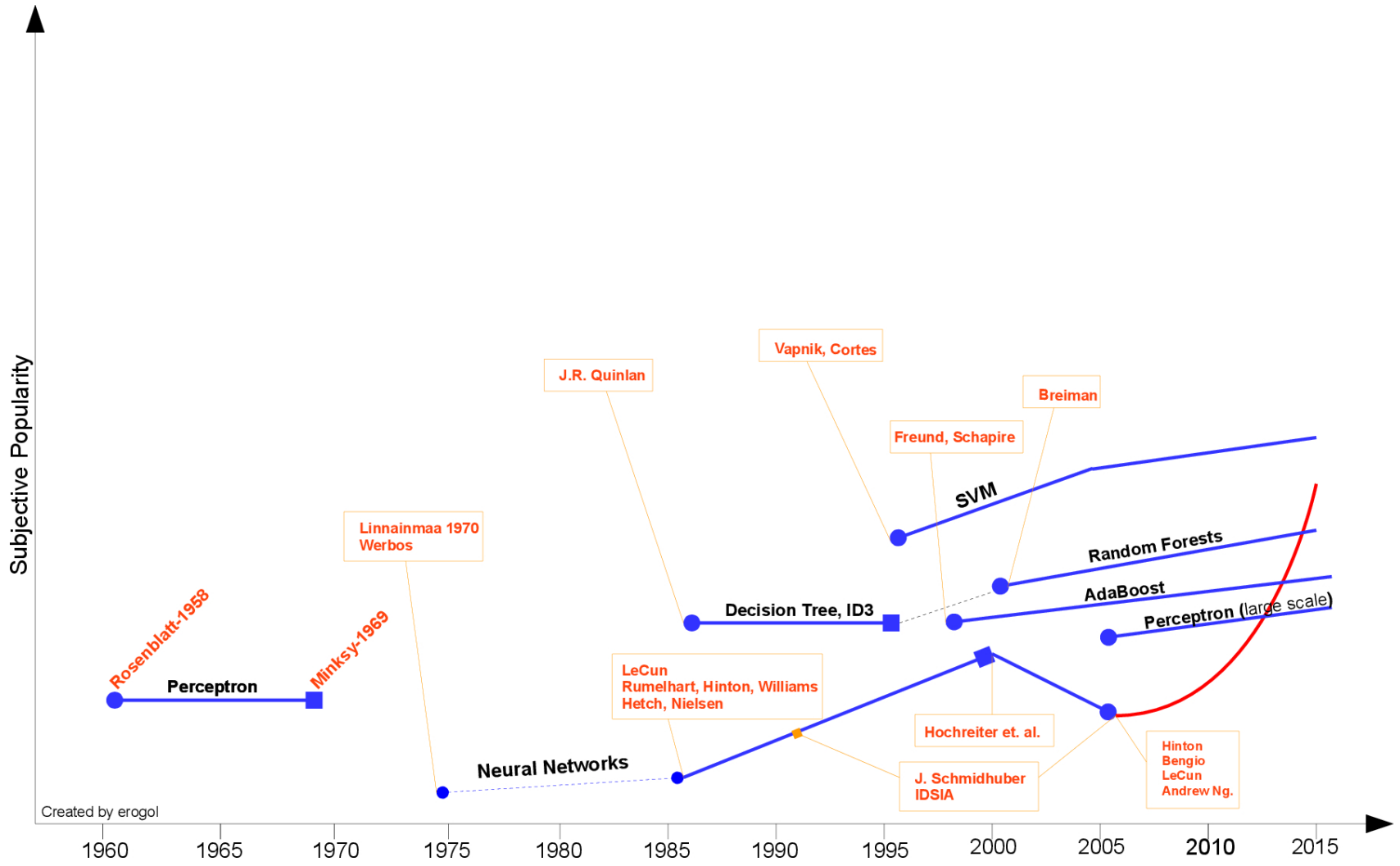
Deep Learning

Alexandre BOULCH

ONERA
THE FRENCH AEROSPACE LAB

Plan de la séance

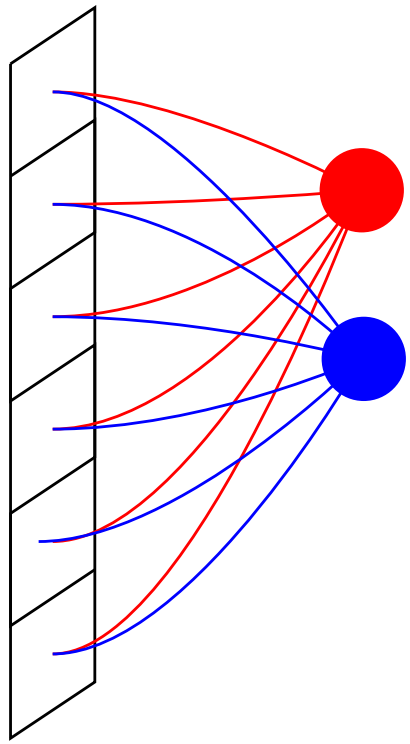
Introduction



ONERA
THE FRENCH AEROSPACE LAB

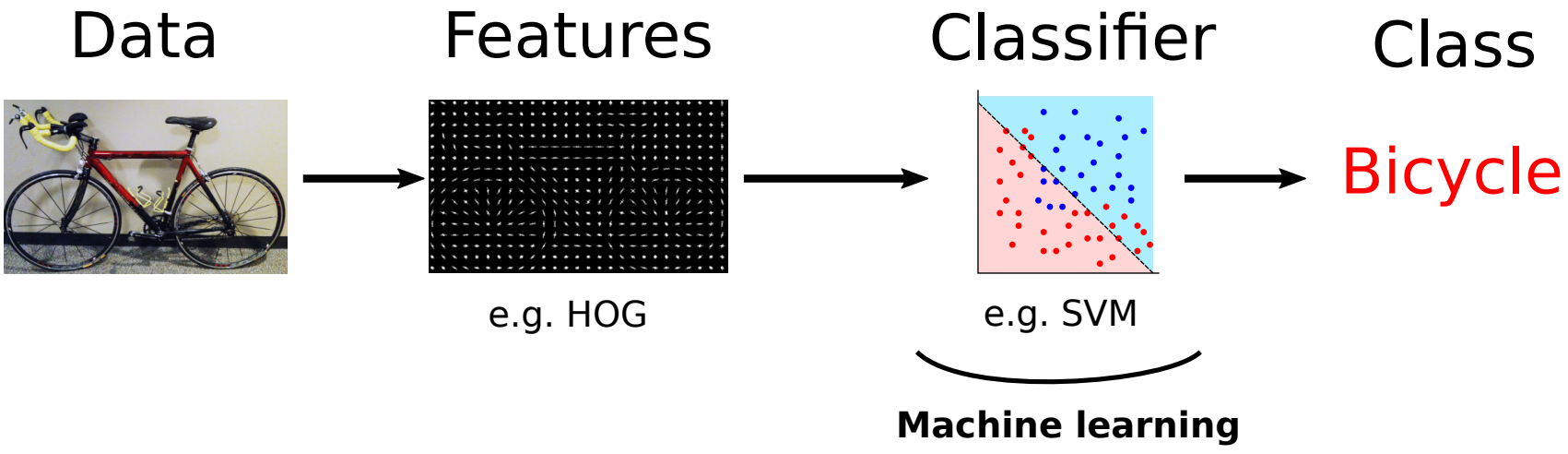
Fully connected

- Perceptron
- A neuron is connected to every input.
- High number of parameters ($|inputs| * |outputs|$)

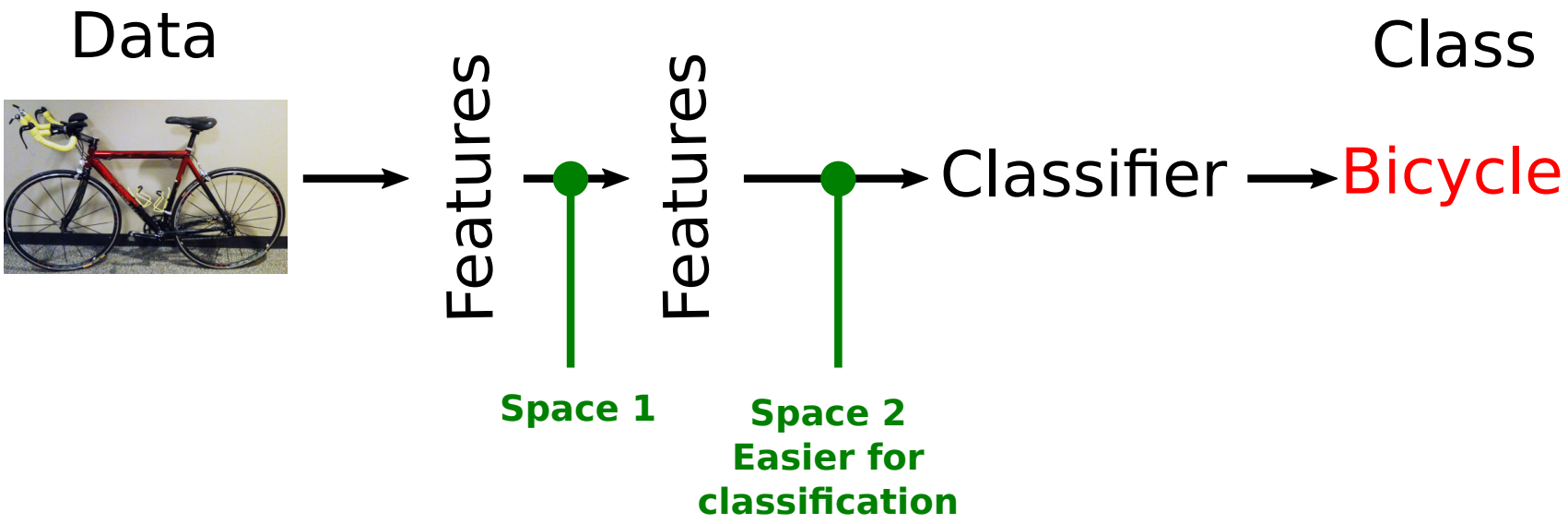


Plan de la séance

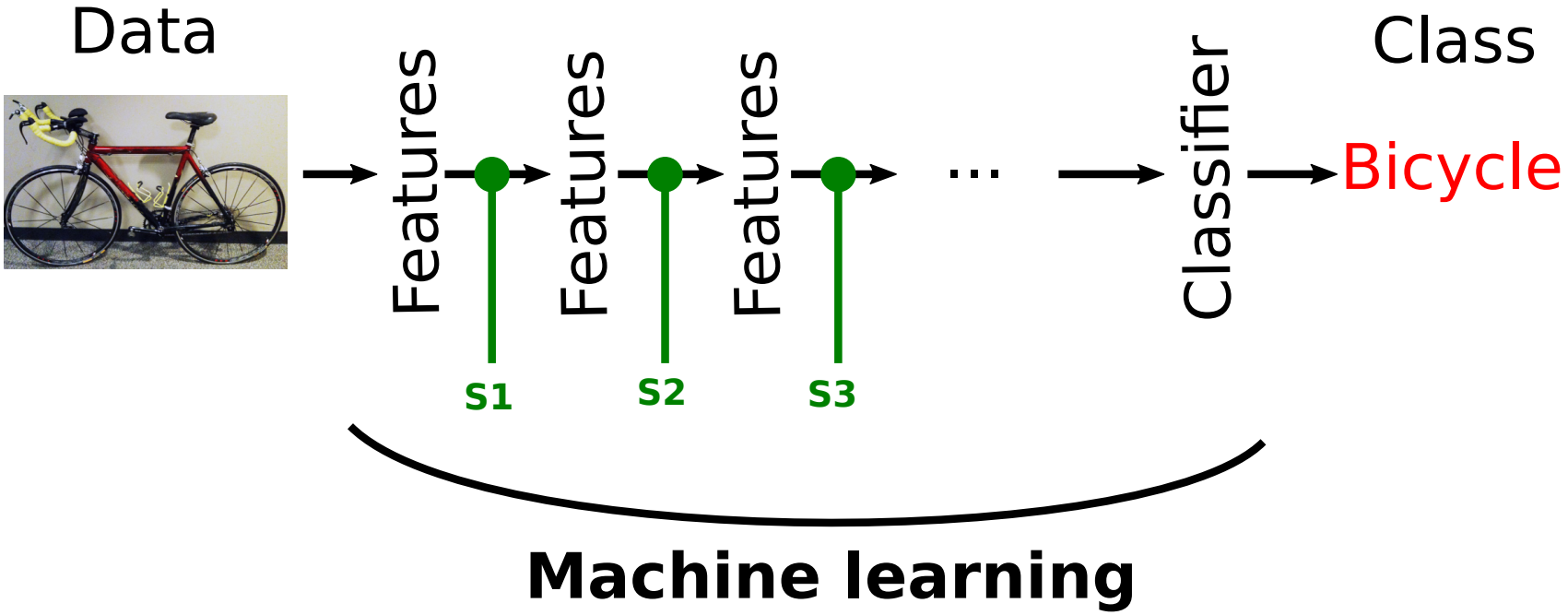
Deep learning concept



Deep learning concept



Deep learning concept



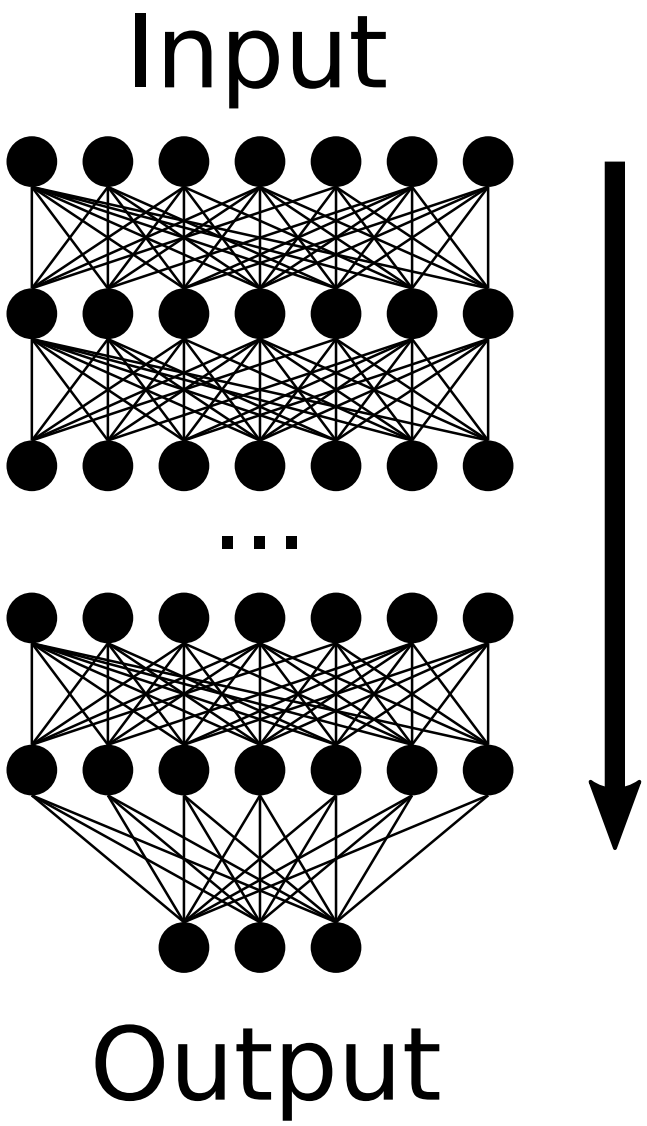
Concept

Deep learning is about learning representations.

Multi-layer perceptron

MLP de plus en plus profonds (avant 2005)

- Très grosses difficultés d'optimisation
- Convergence difficile
- Peu de données
- Entraînement très long (Abandon progressif au profit des SVMs)
  - Simples à utiliser
  - Preuves de convergence
  - Rapides



First convolutional networks

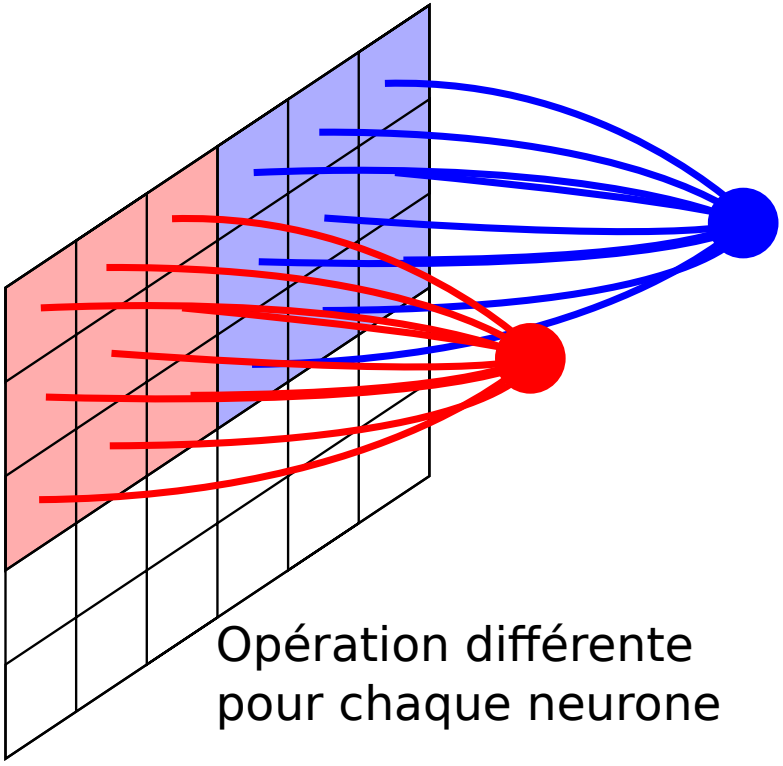
Idea

What is important with images ?

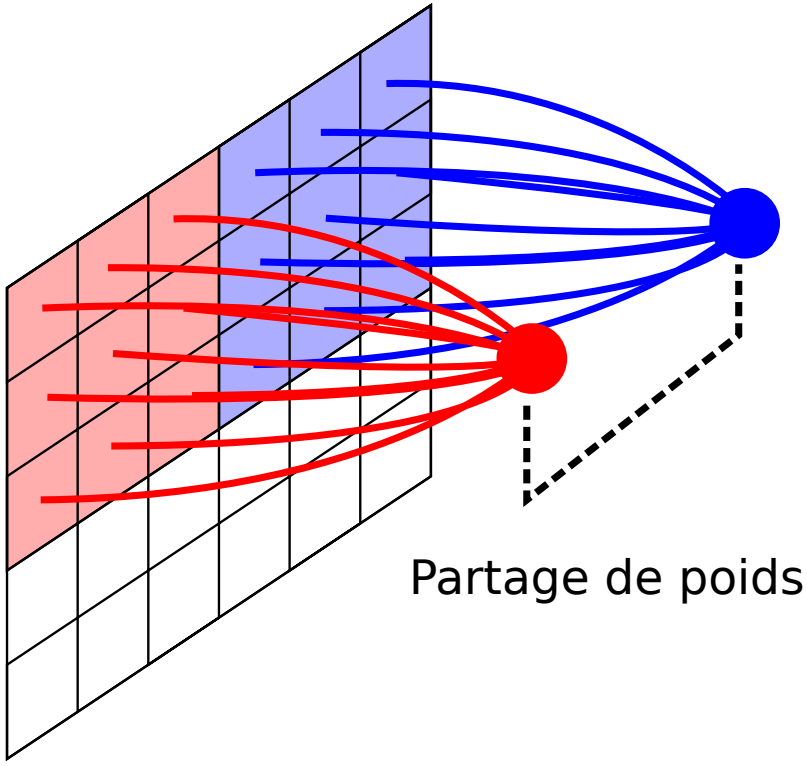
1. Neighborhood (objects have limited sizes)
2. Interest par can be everywhere (same operations everywhere)

Convolutions

Convolutions are useful for structured data.

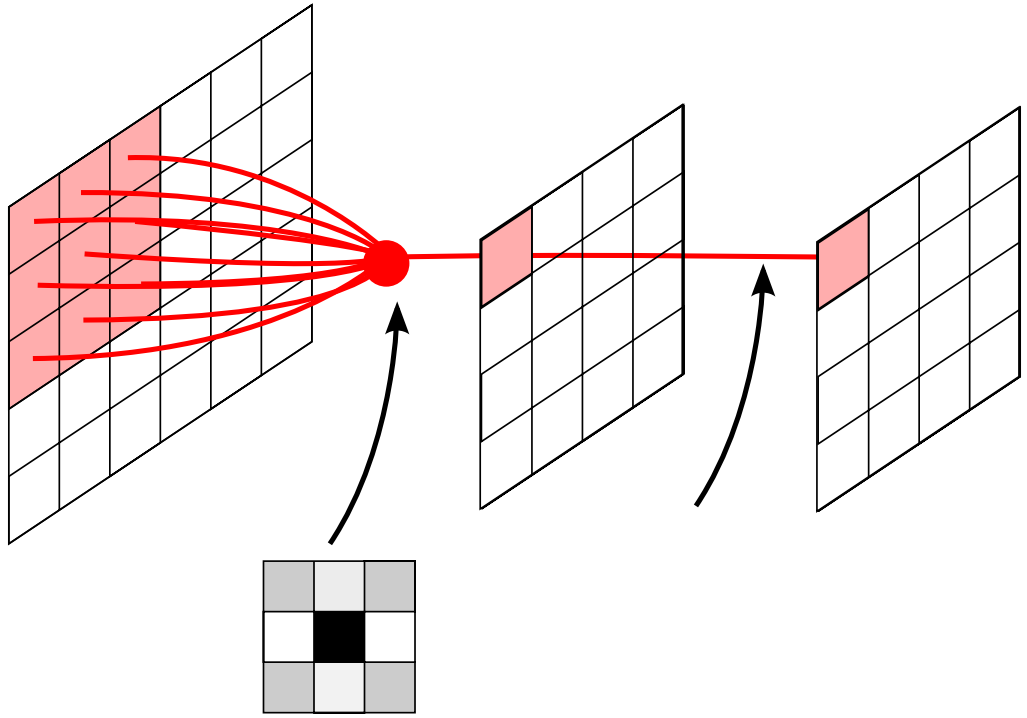


Opération différente
pour chaque neurone



Partage de poids

Convolutions



Forward

$$x^l_{i,j} = \sum_a \sum_b \omega_{a,b} y^l_{i+a,j+b} \quad (1)$$

$$y^{l+1}_{i,j} = \sigma x^l_{i,j} \quad (2)$$

Convolution

Backward weight update

$$\frac{\partial E}{\partial \omega_{a,b}} = \sum_i \sum_j \frac{\partial E}{\partial x^l_{i,j}} \frac{x^l_{i,j}}{\omega_{a,b}} = \sum_i \sum_j \frac{\partial E}{\partial x^l_{i,j}} y^{l-1}_{i,j} \tag{3}$$

And

$$\frac{\partial E}{\partial x^l_{i,j}} = \frac{\partial E}{\partial y^l_{i,j}} \frac{\partial y^l_{i,j}}{\partial x^l_{i,j}} = \frac{\partial E}{\partial y^l_{i,j}} \sigma'(x^l_{i,j}) \tag{4}$$

Finally

$$\frac{\partial E}{\partial \omega_{a,b}} = \sum_i \sum_j \frac{\partial E}{\partial y^l_{i,j}} \sigma'(x^l_{i,j}) y^{l-1}_{i,j} \tag{5}$$

Convolution

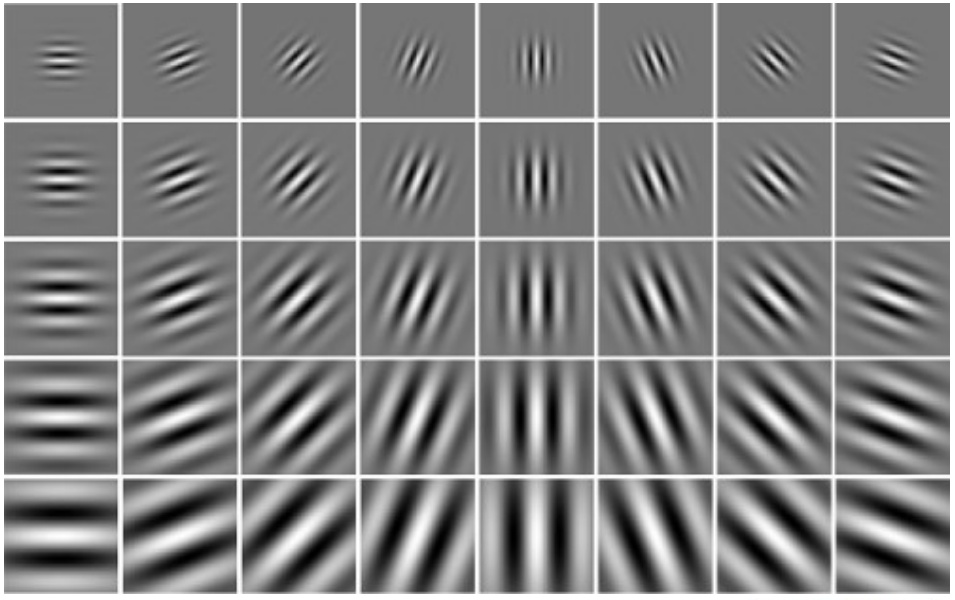
Backward gradient propagation

$$\frac{\partial E}{\partial y_{i,j}^{l-1}} = \sum_a \sum_b \frac{\partial E}{\partial x_{i-a,j-b}^{l}} \frac{\partial x_{i-a,j-b}^{l}}{y_{i,j}^{l-1}} \tag{6}$$

$$= \sum_a \sum_b \frac{\partial E}{\partial x_{i-a,j-b}^{l}} \omega_{a,b} \tag{7}$$

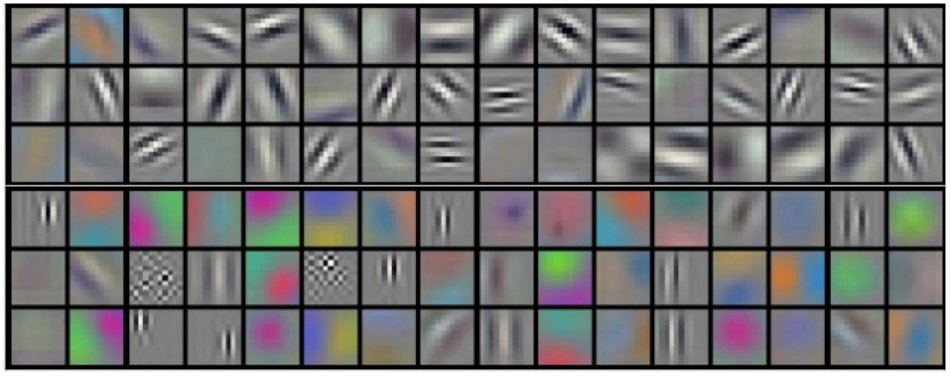
Convolution

What do convolutions learn ?



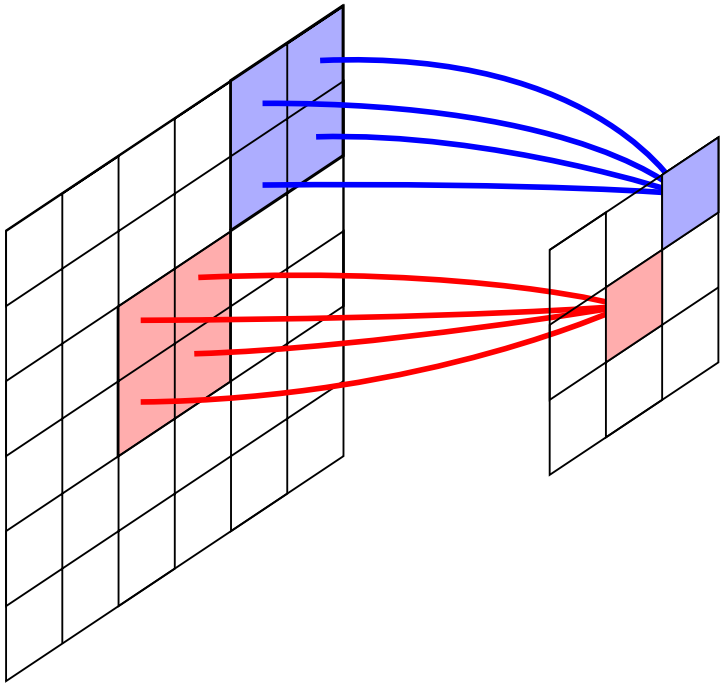
Gabor filters.

First layer of AlexNet.

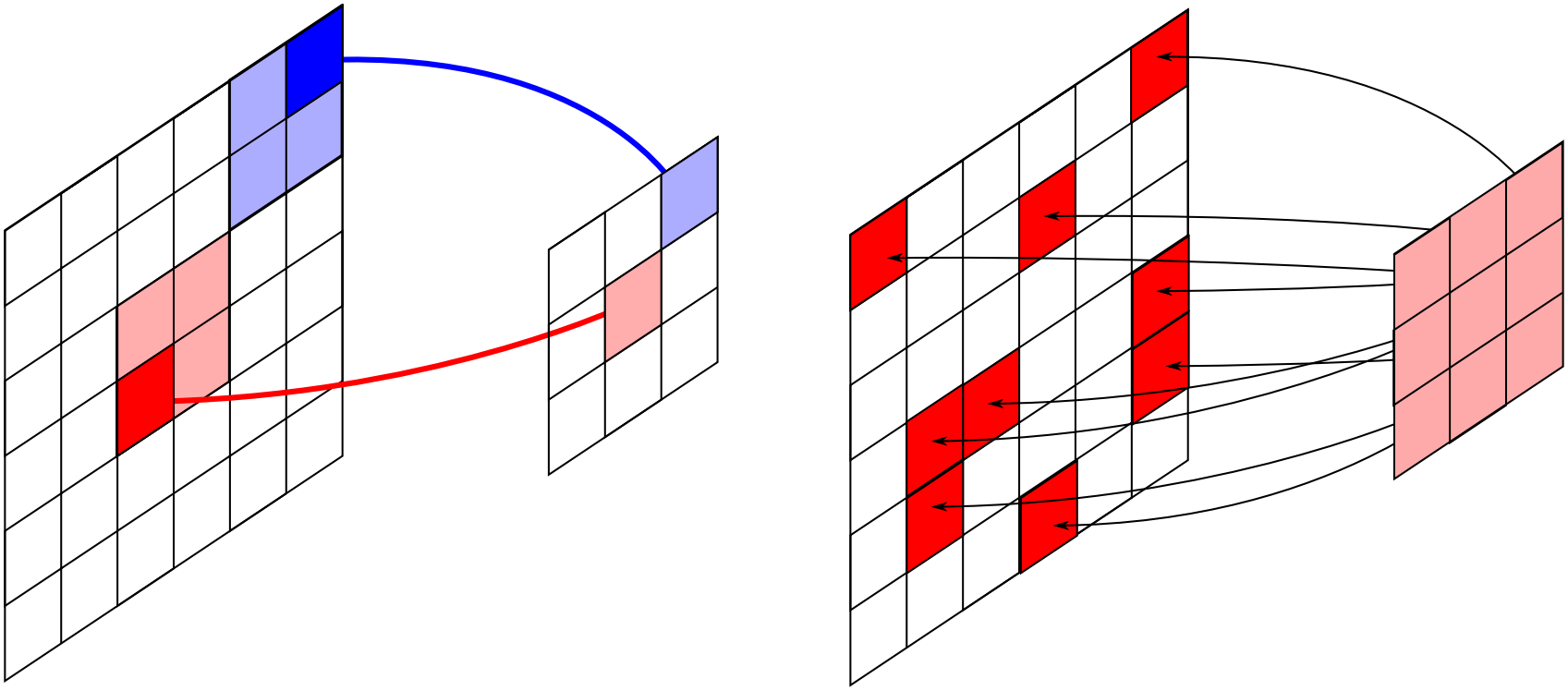


Max Pooling

- dimension reduction
- translation invariability



Max Pooling

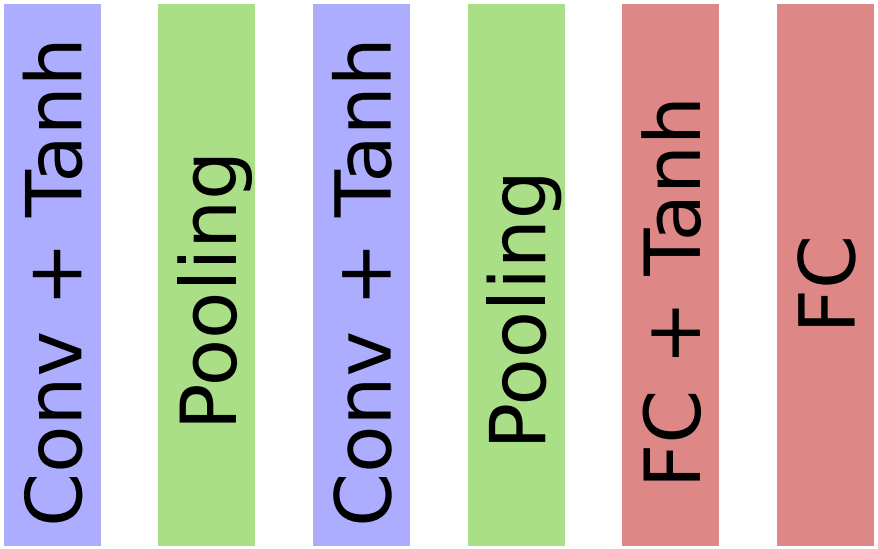


Forward : max signal

Backward : Gradient transmission to max signal origin, zero otherwise

LeNet (1990)

LeNet (1990)
Images 28x28



Issues

Issues

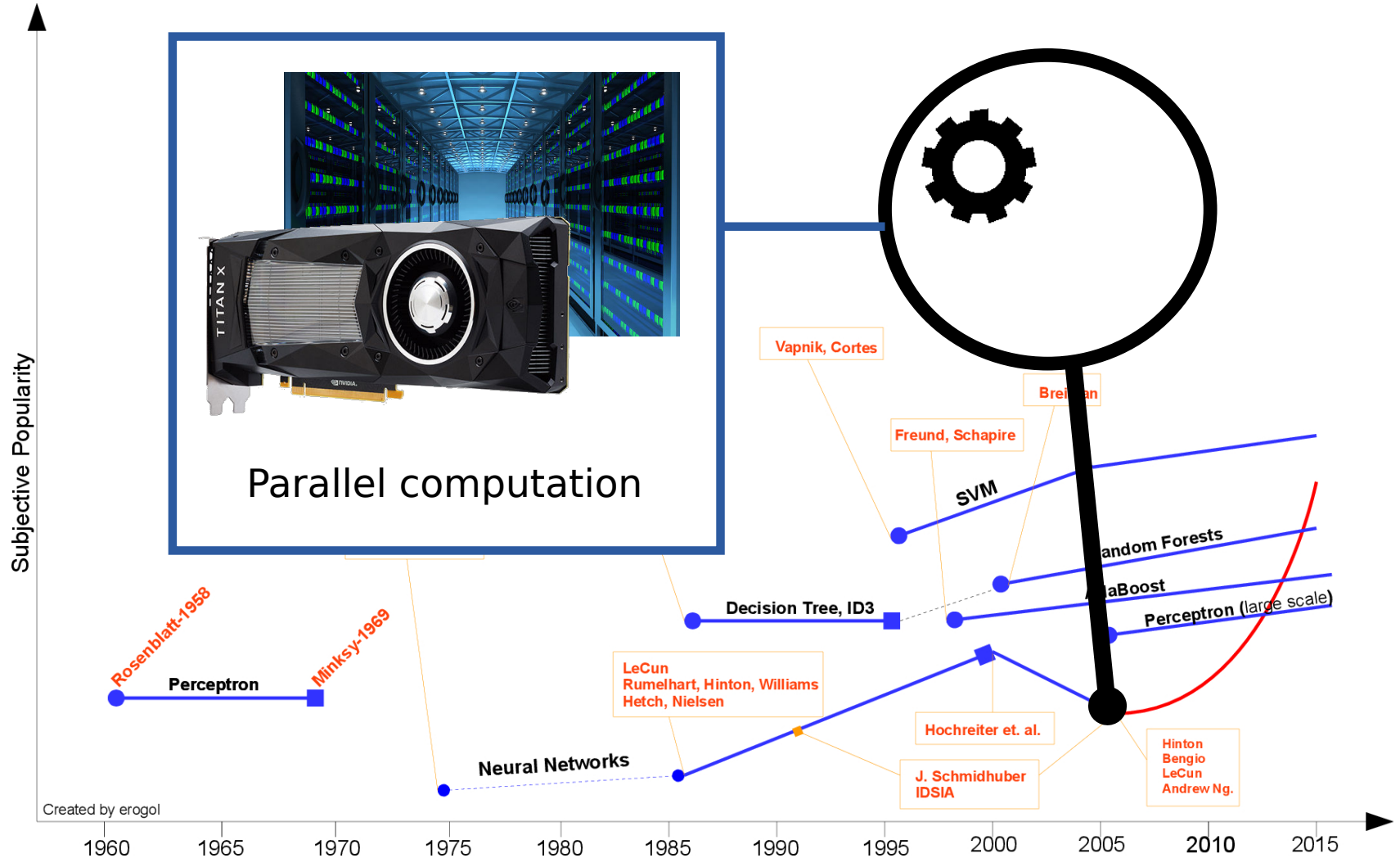
- Learning speed
- Exploding or vanishing gradients
- Overfitting
- Local minima

Limitations

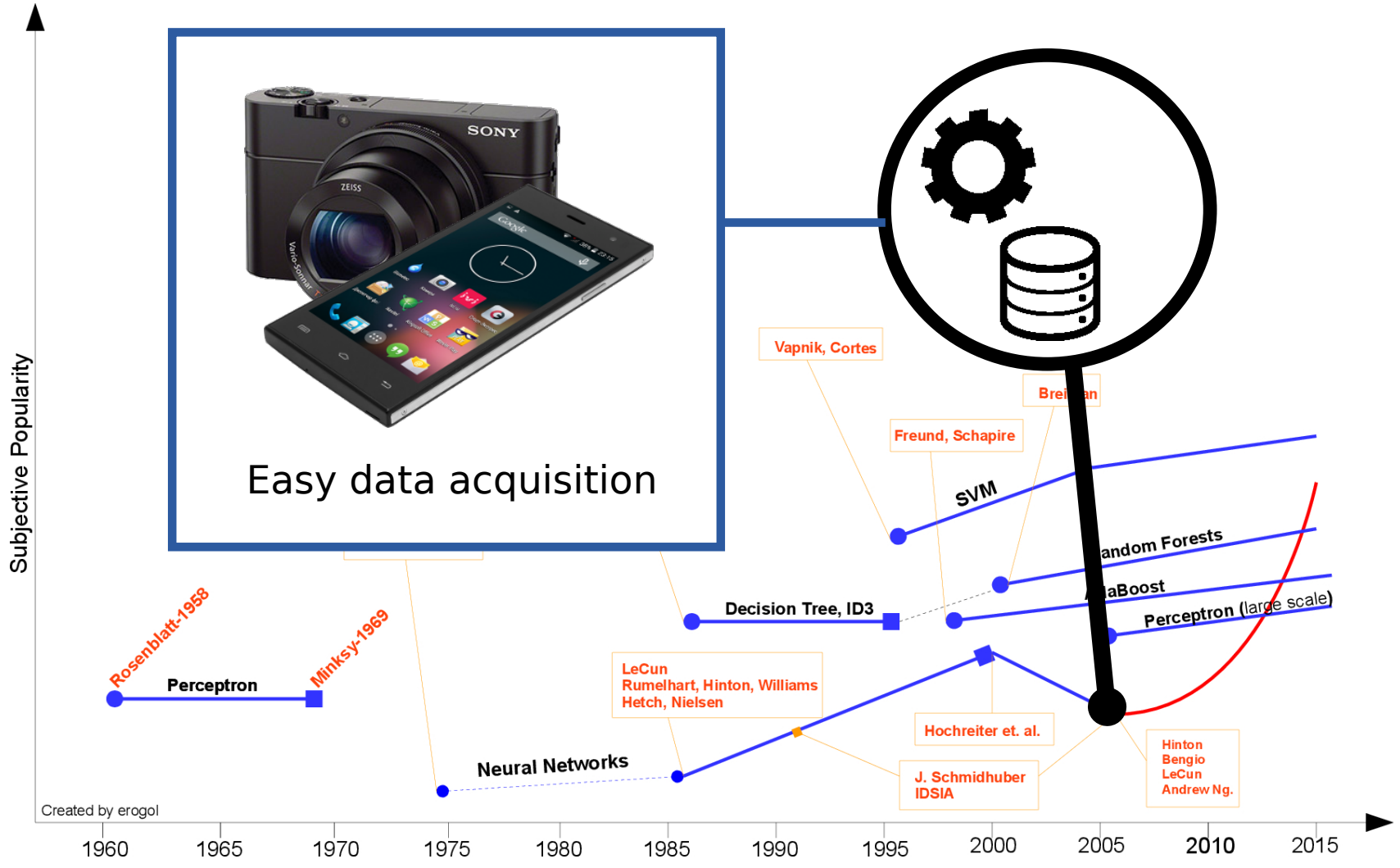
- Architecture
- Initialization
- Computing power
- Data
- Optimization

Plan de la séance

Massively data driven approaches

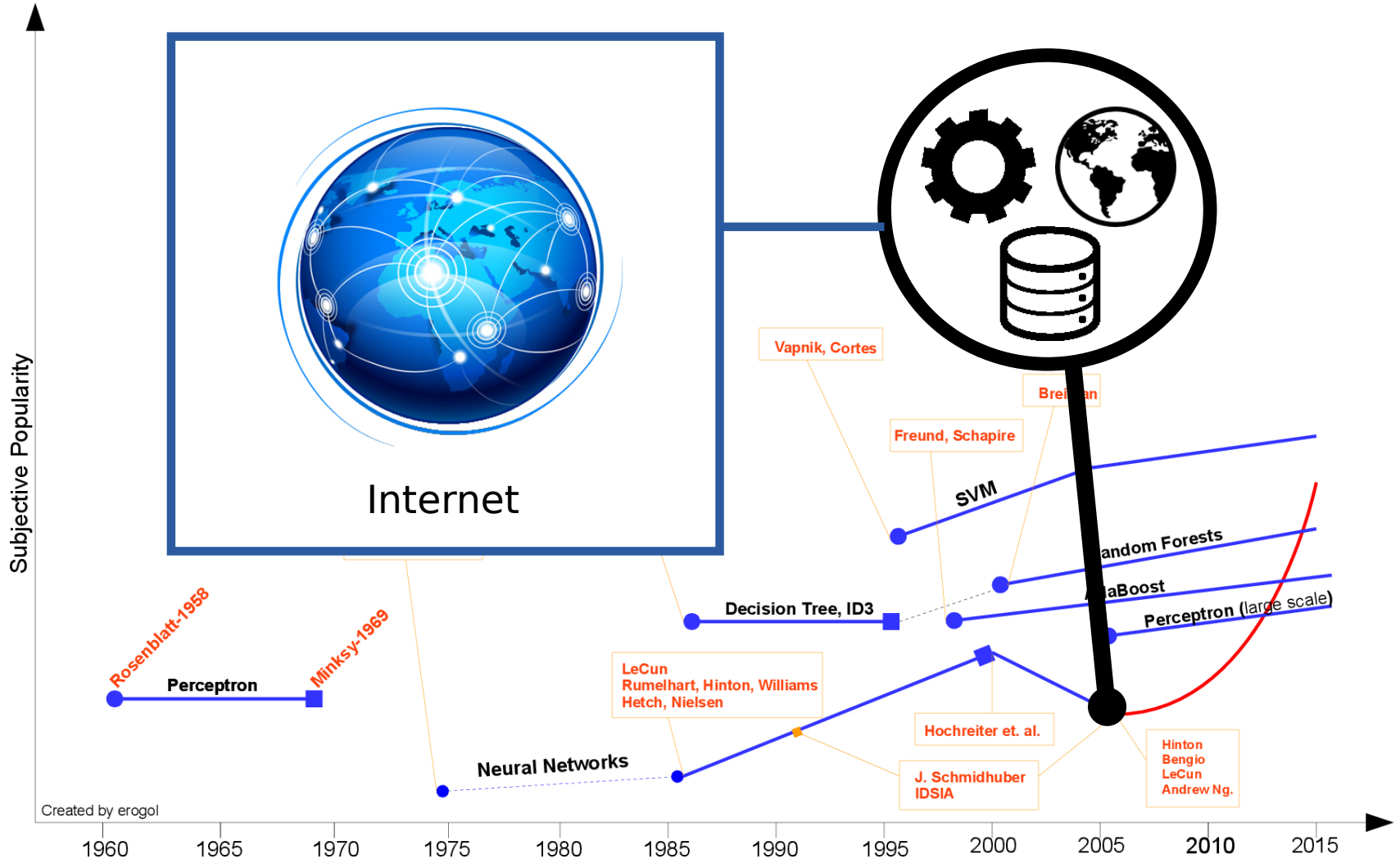


Massively data driven approaches



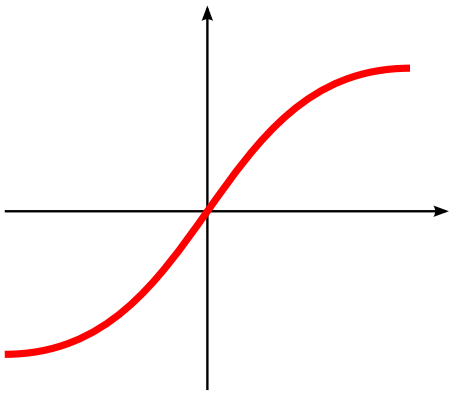
ONERA
THE FRENCH AEROSPACE LAB

Massively data driven approaches

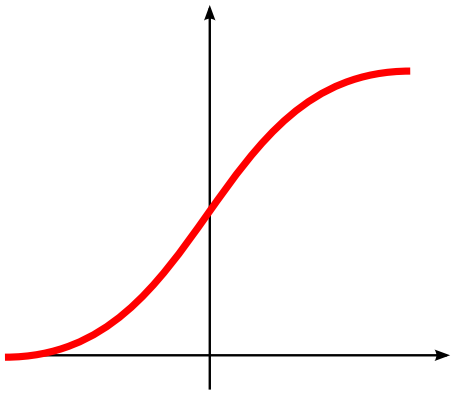


ONERA
THE FRENCH AEROSPACE LAB

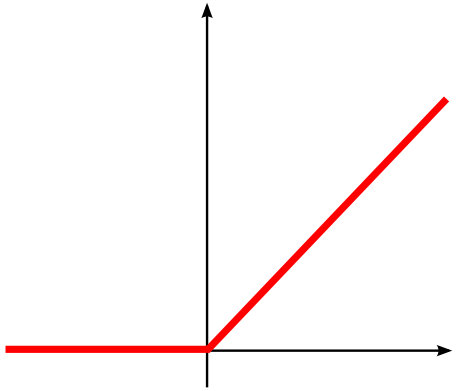
Activations



Tangente hypebolique



Sigmoïde

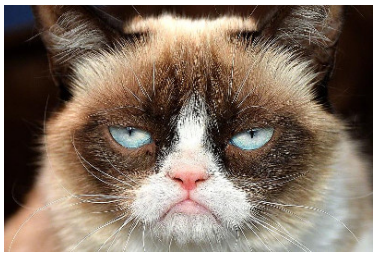


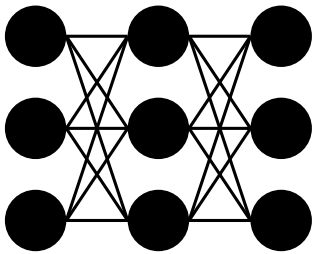
Rectified Linear Unit (ReLU)

Rectified linear unit

1. Faster gradient computation
2. Similar convergence

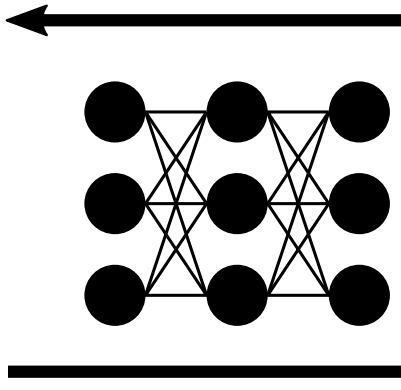
Mini-batches





Loss image 1

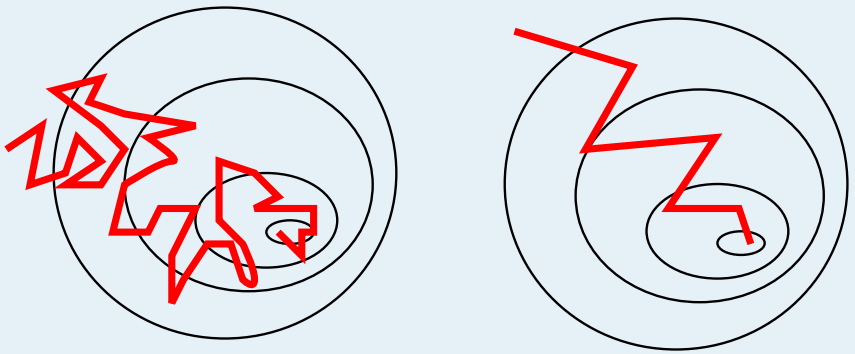
Mini-batches



Backward with mean of losses

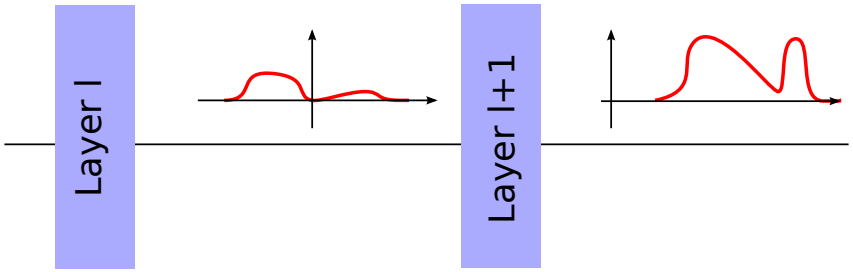
Loss image 1
Loss image 2
Loss image 3
Loss image 4
...

Gradient smoothing



Smoother gradient converges faster.

BatchNorm

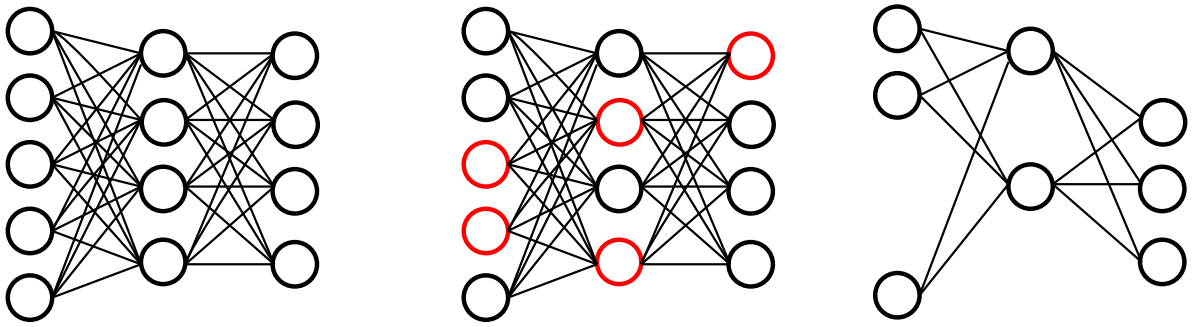


Changes in the signal dynamic make the model more difficult to optimize : exponential or vanishing gradients.
Objective : control the signal distribution :

$$y^{l*} = \frac{y^l - \mu}{\sqrt{\sigma^2 + \epsilon}}\gamma + \beta \tag{8}$$

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

Dropout



Training

- Shutdown neurons with probability $1 - p$
- Equivalent to training multiple sub-networks

Inference

- Use all neurons, with p ponderation

Dropout give robustness to overfitting.

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

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

See neural network class

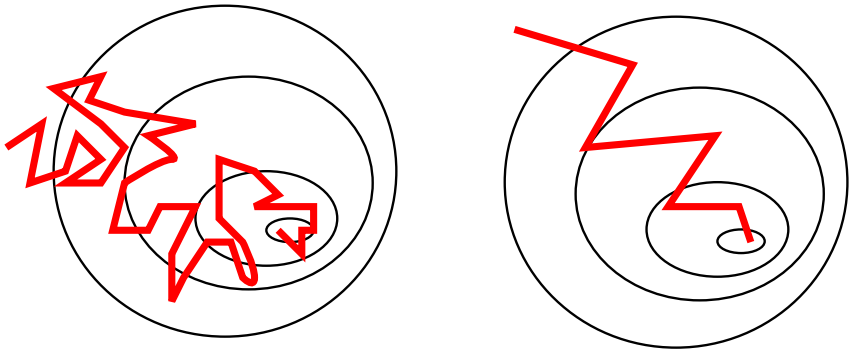
Update rule : $w_{t+1} = w_t + \alpha \Delta w$ (learning rate α)

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)**

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

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

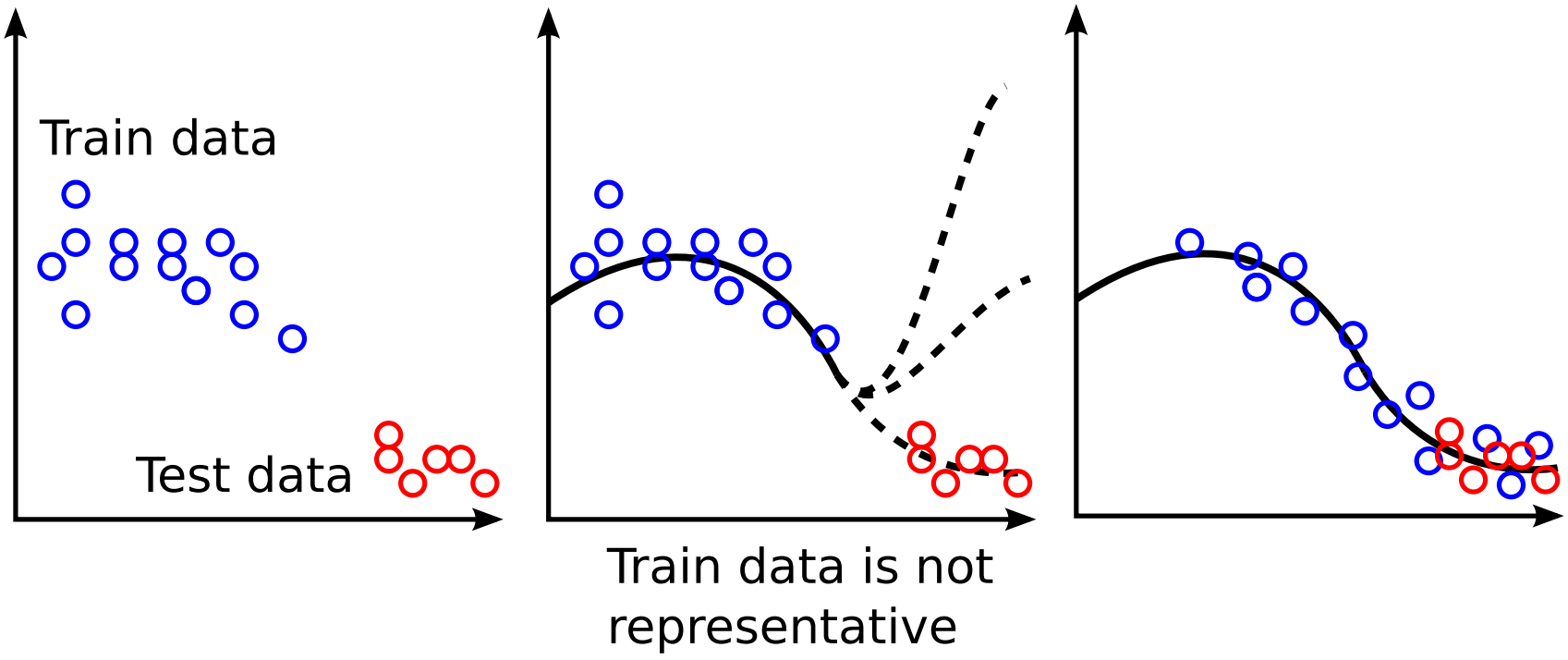
Plan de la séance

Data

- Representative
- Data augmentation
- Data normalization

Data

Train data must be representative of the problem



Data

Data normalization

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

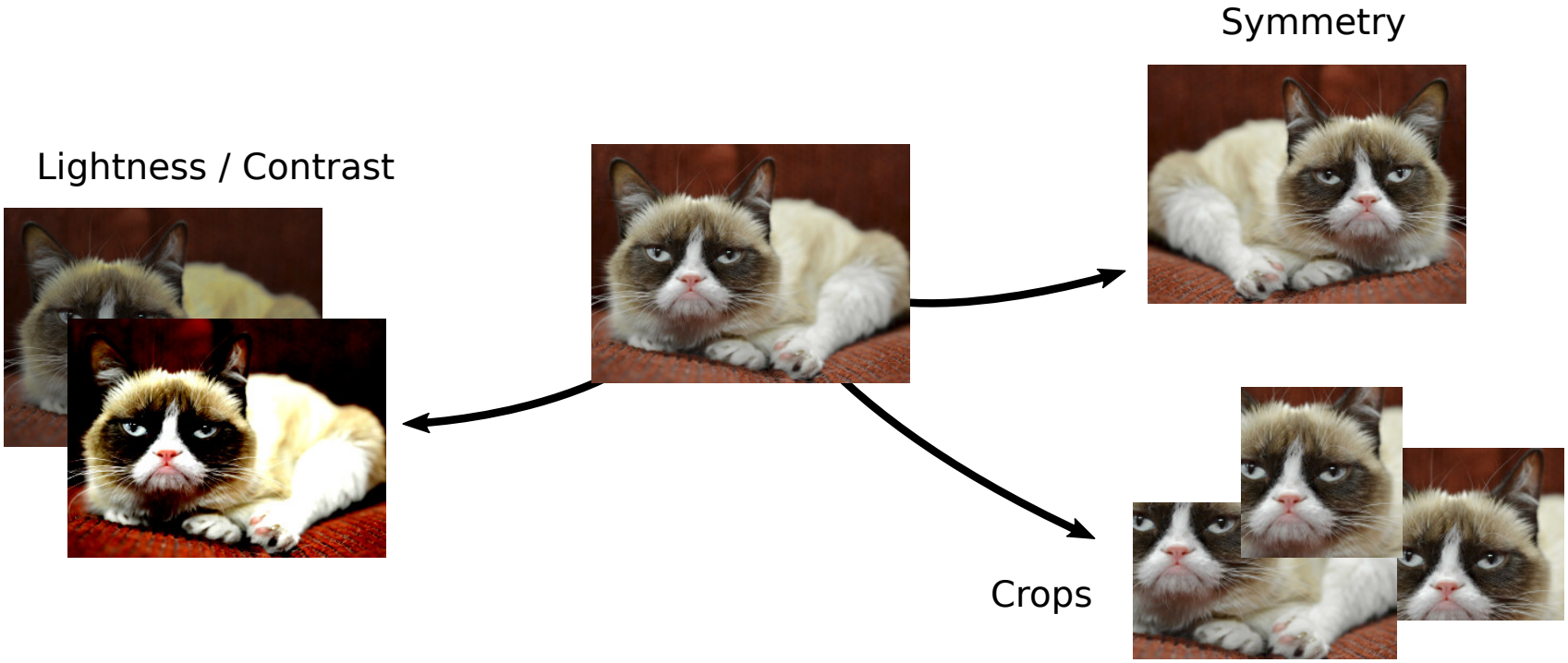
$$\hat{I} = \frac{I - \mu}{\sigma}$$

Data

Data augmentation

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

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



ONERA
THE FRENCH AEROSPACE LAB

Problems and partial solutions

Problems

- Small amount of data
- Low computational power

Solutions ?

- Use classical approaches (Perceptron, SVM, ...)
- Use pre-trained deep features and learn classifier (**TP**)

Plan de la séance

Image classification - LeNet vs VGG

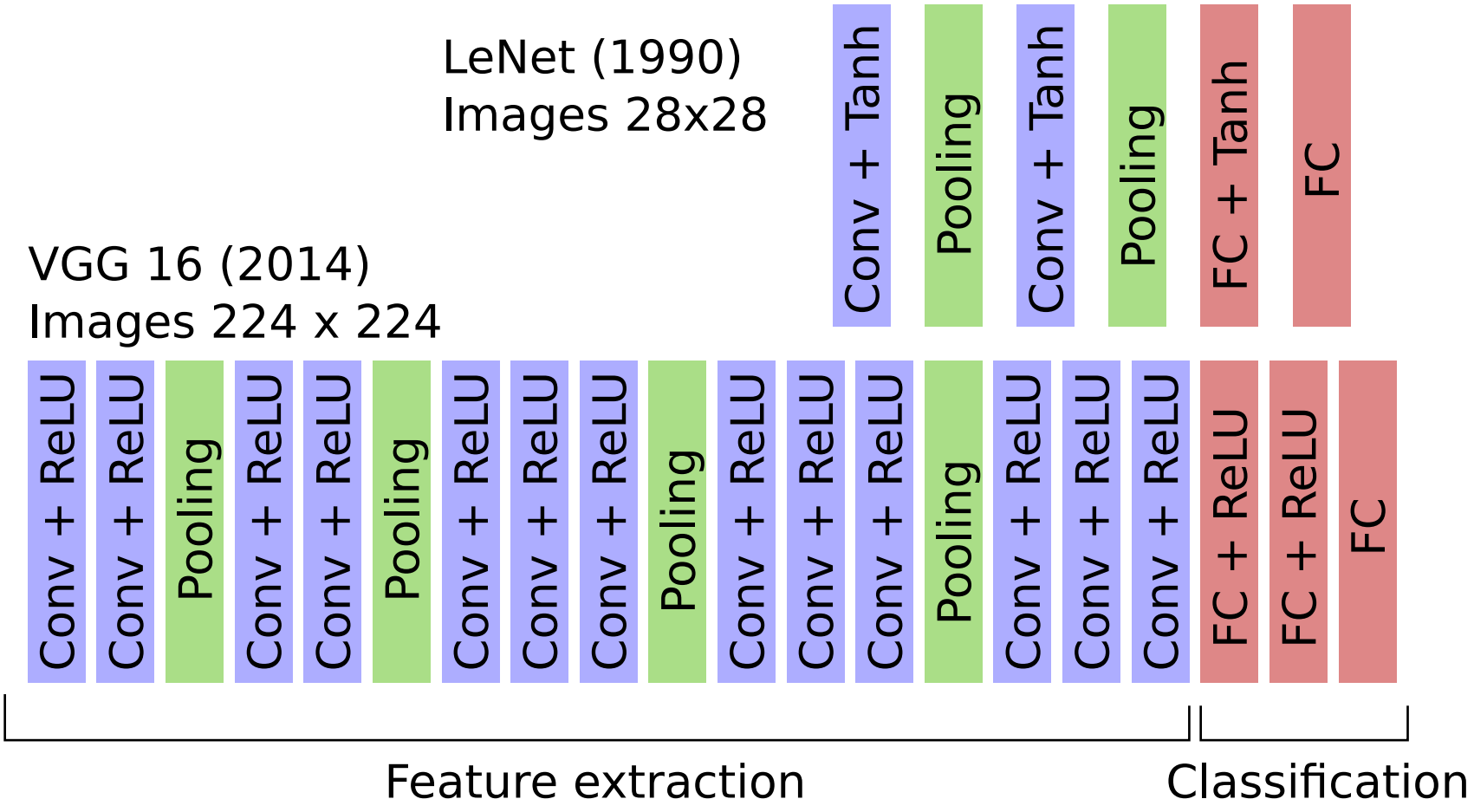
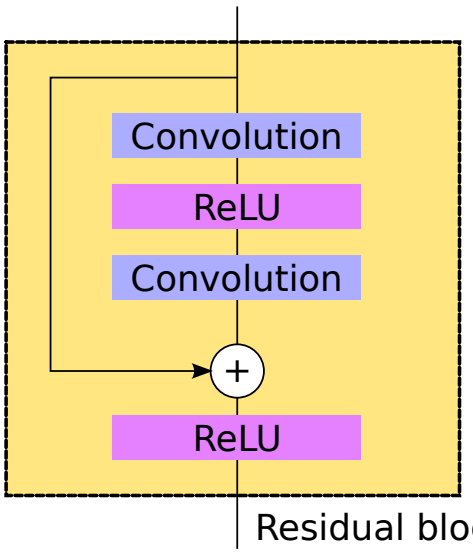


Image classification - ResNet



Convolution
Pooling
Res. block
Res. block
Res. block
Res Block /2
Res. block
Res. block
Res. block
Res Block /2
...

...
Res. block
Res. block
Res. block
Res. block
Res. block
Res Block /2
Res. block
Res. block
Av .Pooling
FC

ONERA
THE FRENCH AEROSPACE LAB

Semantic segmentation

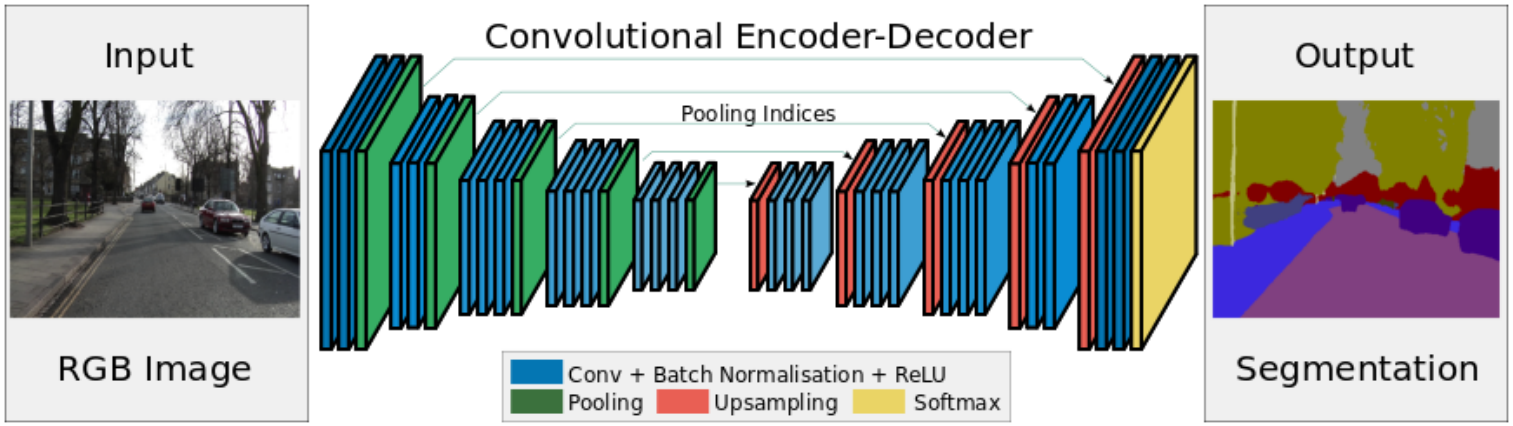
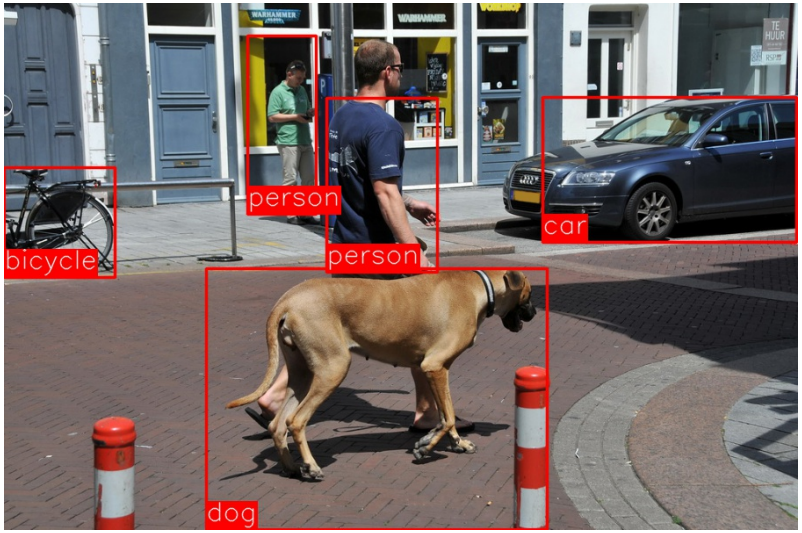
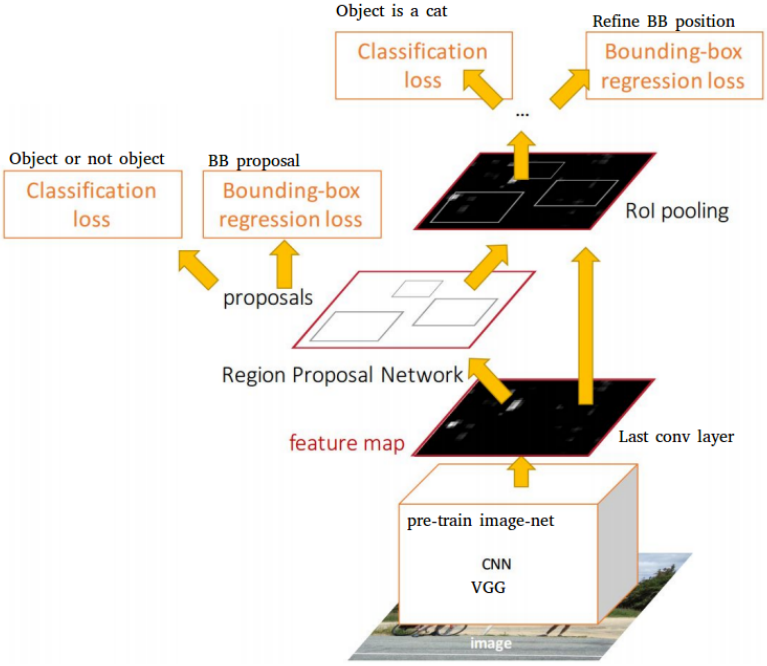


Image : http://mi.eng.cam.ac.uk/projects/segnet/

Detection



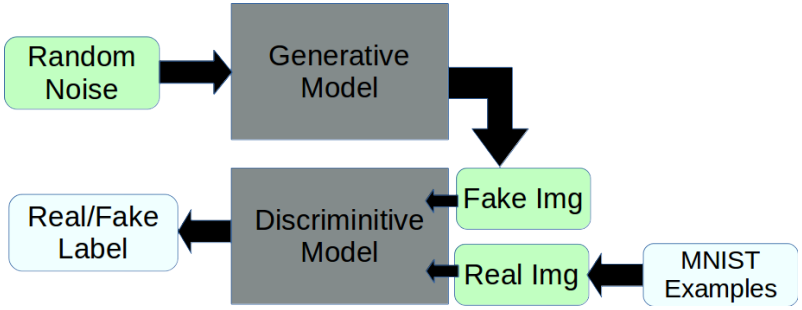


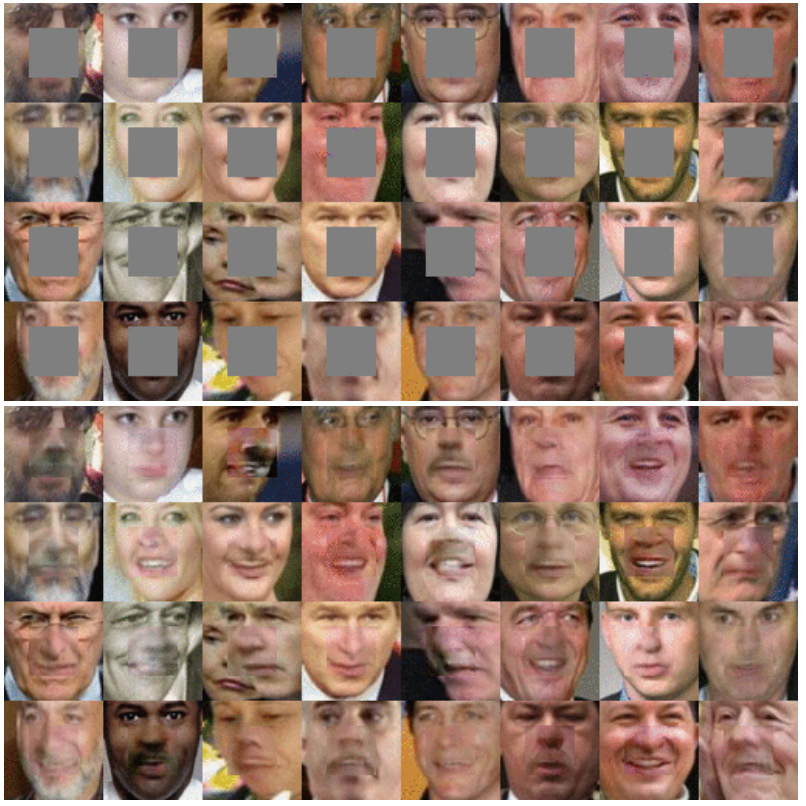
ONERA
THE FRENCH AEROSPACE LAB

Data generation

Generative Adversarial Networks

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

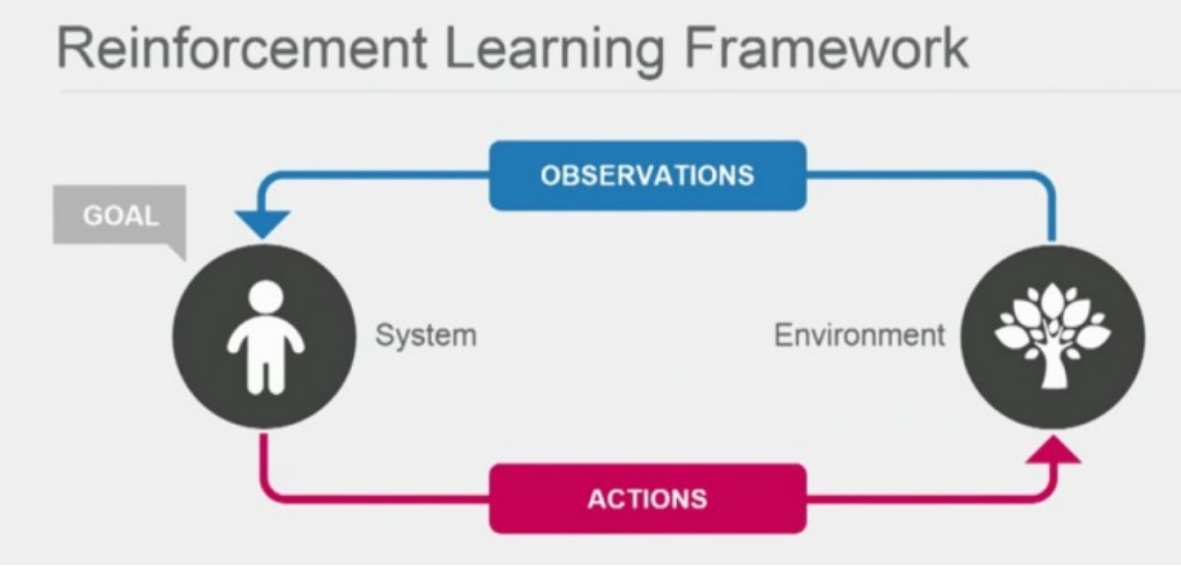




Reinforcement learning

Try and error learning

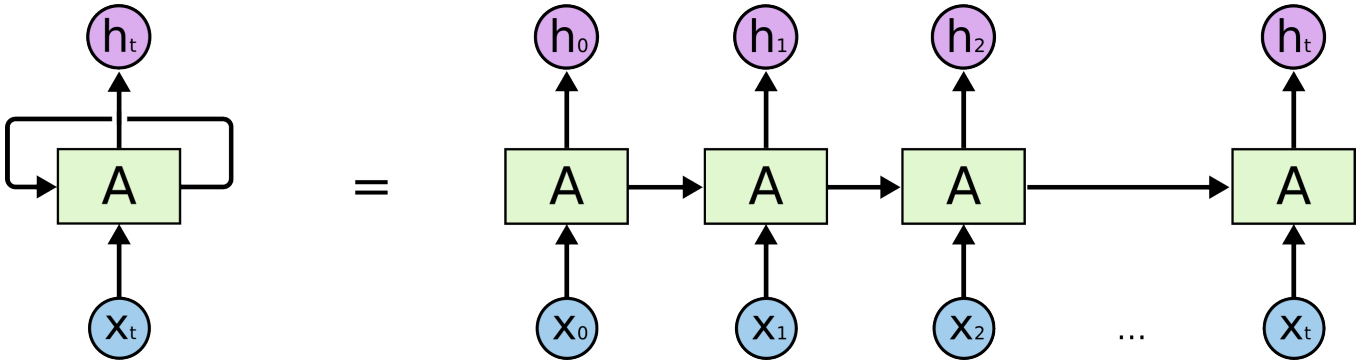
Reward and Penalty.



1

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

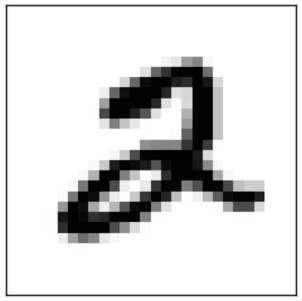
Recurrent neural networks

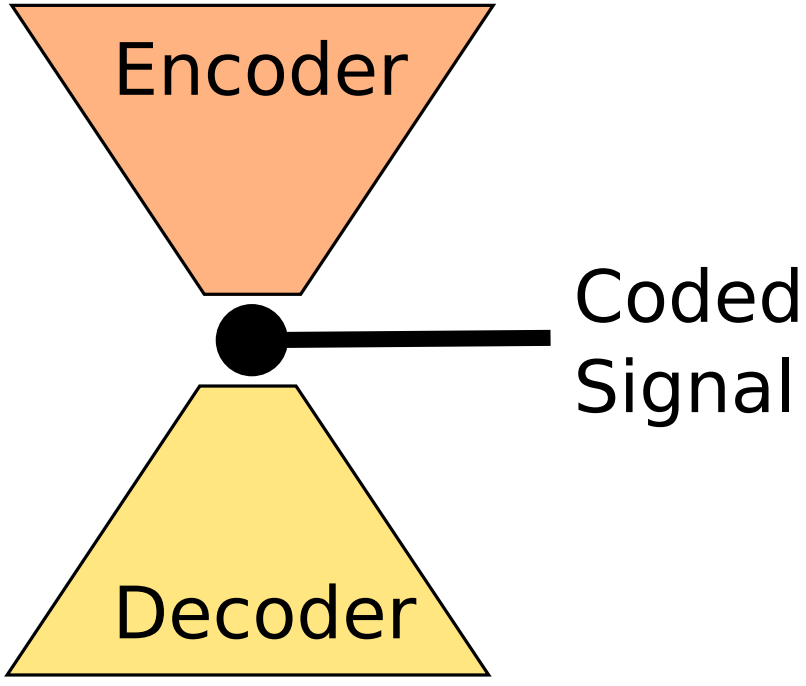


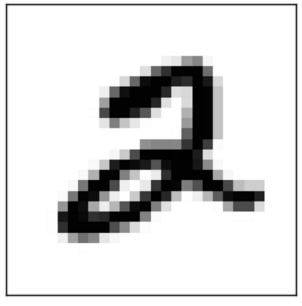
Dealing with sequences

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

Autoencoders







Plan de la séance

Frameworks

PYTORCH

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

- Fast overview
- Few applications
- Practical work : finetuning and small recurrent neural networks