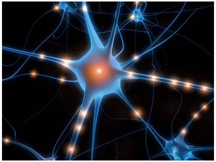
Neural Networks

clement.chatelain@insa-rouen.fr

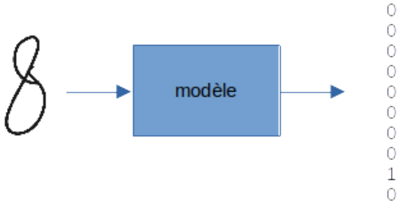
17 avril 2024

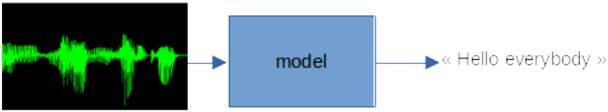


Sommaire

Introduction

Machine learning before 2010 (more or less) :





Limited range of machine learning

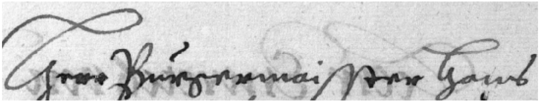
- Only dedicated to very specific tasks
- Not very impressive ...

Introduction

Machine learning since 2010 :

"Two pizzas sitting on top of a stove top oven"

"A group of young people playing a game of frisbee"



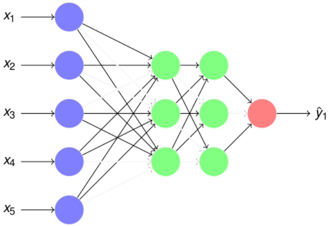
and also : Deepl, medical imaging, `https://thispersondoesnotexist.com/`, deep nostalgia, etc.

Introduction

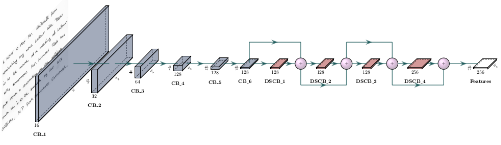
What happens in 2010 ? The emergence of “Deep learning” !

- Statistical models that benefit from 50 years of research in ML
- Deep Learning = Deep **Neural networks** = connectionist architectures
- Impressive results : some models pass the turing test !

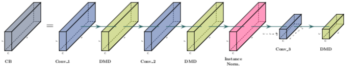
A NN model before :



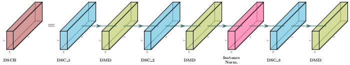
A NN model now [Coquenet2021] :



(a) FCN Encoder overview.

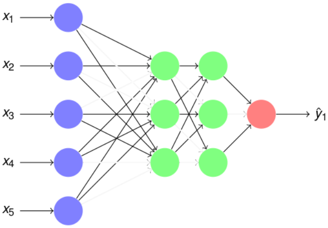


(b) Convolution Block (CB) definition.



In this Course you will learn :

- What's a neuron
- How they are connected & how they can model complex decision functions
- How they are trained (roughly)
- What's the deep learning revolution ?



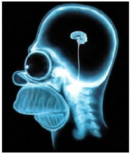
Plan

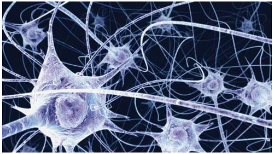
Plan

Human vs. Machine learning (1)

The best machine learning model is a human (yet ;))

- Brain made of simple entities : neurons
- Neurons are connected through synapses
- Activation (or not) of neurons output depend of its inputs
- Information propagates from inputs (senses, memory) till decisions





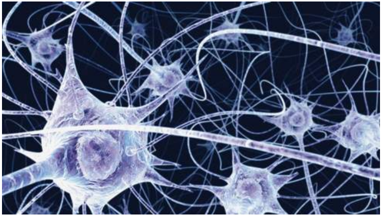
Remarks :

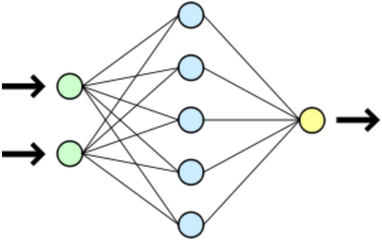
- Combining many simple entities can model very complex functions
- Knowledge = **connections** between neurons, not neurons themself
- $\rightarrow$ learning = set the good connections between neurons

Human vs. Machine learning (2)

Neural networks are simplified mathematical modelization of the human brain, where :

- Biological neurons = **formal neurons**
- Synapses = (weighted) connections
- Learning = optimization. Of what ? **the connections = weights**
- Reward = Loss





Formal neuron [McCulloch & Pitts, 1943]

Formal neuron

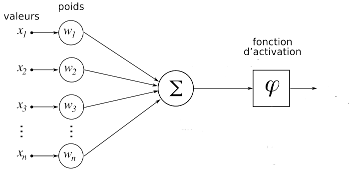
Given E inputs x_e, and an output y :

- **Combine information** : a weighted sum α of x_e is computed :

$$\alpha = \sum_{e=0}^{E} w_e x_e \quad \text{with} \quad x_0 = 1$$

- **Output activation** : apply an activation function φ on α :

$$y = \varphi(\alpha) = \varphi\left(\sum_{e=0}^{E} w_e x_e\right)$$

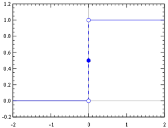
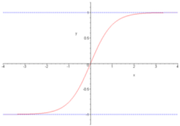
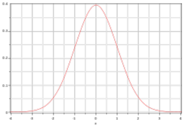
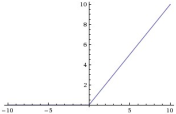


A perceptron

Formal neuron [McCulloch & Pitts, 1943]

Aim of activation functions :

Modelize the output activation when the weighted sum is above/around a certain value.

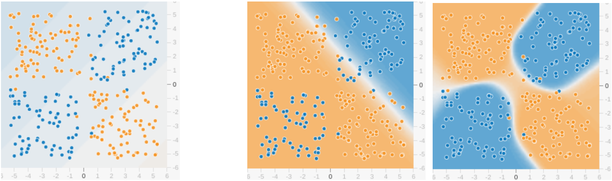
heaviside - tanh - gaussian - ReLU

Remarks :

- The “certain value” can be set (learned !) thanks to the bias w_0
- φ should be differentiable ($\rightarrow$ ~~heaviside~~)
- Often chosen non linear

Formal neuron [McCulloch & Pitts, 1943]

Let us consider a classification problem $\{orange, blue\}$ with 2 inputs x_1, x_2.
Once the weights learned, linear/non linear activations brings decisions such as :



- OK for a simple decision (2 outputs) in a small space (2 inputs)
- What for a natural scene image classification (512×512 inputs, 1000 outputs) ?
- **We need more neurons to build more complex functions ! !**

How to connect neurons ? Topology.

Plan

Topologies : many ways to arrange neurons and connections

topologies

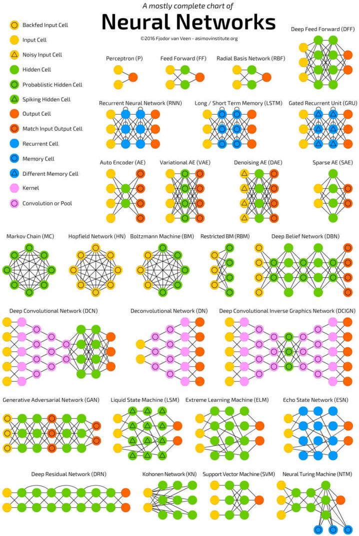
- Layered / Random / Totally interconnected
- Feedforward / recurrent
- Shared parameters
- ...

(and also different kinds of neurons : classical/LSTM/...)

Here we will explore basic architecture :

- Layered, Feedforward

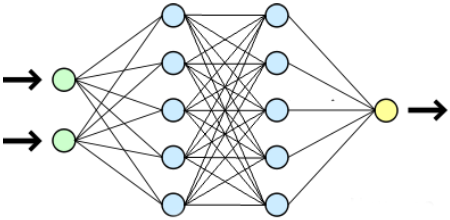
source :
https://www.asimovinstitute.org/author/fjodorvanveen/



Feedforward networks

Neurons are organized into layers

- Only connections between consecutive layers
- « feedforward » : information from input to output



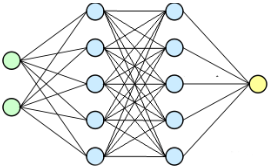
The connectionnist idea :
successive application of simple function may lead to complex functions.

- Efficient learning algorithm : gradient backpropagation
- May include different kinds of layer : Dense, Convolutional, pooling

Feedforward networks : Dense Layers

AKA Linear layers

- Each neuron is fed with outputs from **all** neurons from previous layer
- Can be used with any activation function
- “Legacy” layers



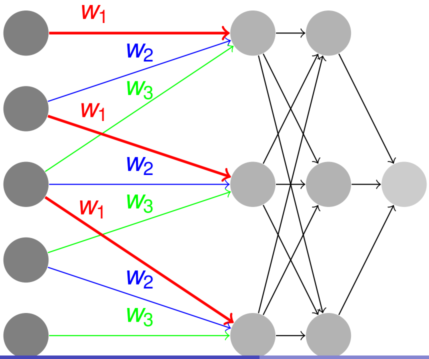
Remarks

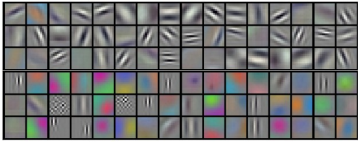
- Contain a lot of parameters (“Dense”)
- Last layer of a NN is generally a dense layer
  - Either for classif. with Softmax activation
  - Either for regression with Identity

Feedforward networks : Convolutional layers

Convolutional layers

- Each neuron is fed with outputs from **a subset** of neurons
- Each input subset changes, but the parameters subset is shared
- Convolutions through kernels = the subset of parameters
- Main specificities : light, **extract features**



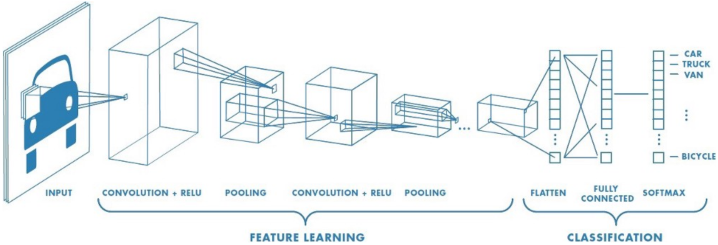


CNN filters
[Krizhevsky et al.]

Feedforward networks : Conv. neural nets (CNN)

Typical architecture :

- A bunch of convolutional layers, with several kernels per layer
- Often copled with pooling layers to concentrate the information
- Flatten
- Dense layer for decision purpose



Nice online demonstration :
`https://www.cs.ryerson.ca/~aharley/vis/conv/`

Plan

Training : problem statement

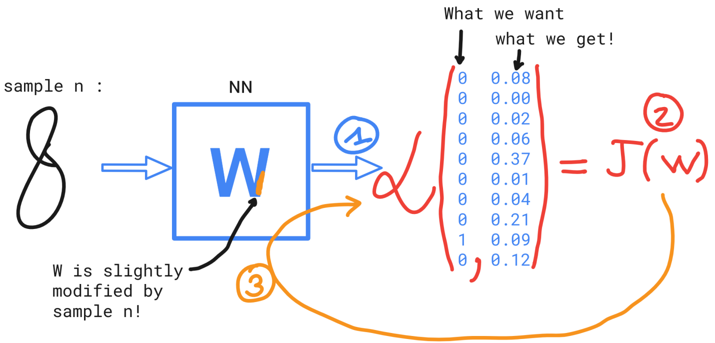
Recall : we aim at finding f, i.e. :

- Learn the parameters = connections between neurons W ...
- ... using N samples from the train set ...
- ... for optimizing a Loss $\mathcal{L}(\mathbf{W})$ **that must be differentiable**.

Losses to optimize :

- Mean Square Error $\mathcal{L}(\mathbf{W}) = \sum_{n=1}^{N} \left(y(n) - f(x(n))^2\right)$
- Categorical Cross Entropy (for classification)
- etc.

Neural Networks : Learning Scheme

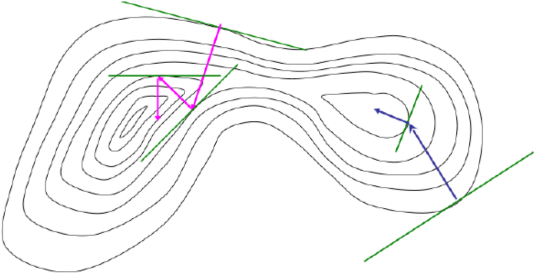


1. Forward sample through a randomly initialized **W**
2. Compute Loss between “what we get” and “what we want” (the target)
3. Modify (slightly) **W** so that “what we get” going closer to “what we want”

How to modify **W** ? Gradient Descent !

Gradient Descent principles

Hidden values of Loss $\mathcal{L}(\mathbf{W})$ in a two parameters space $(\mathbf{W}_1, \mathbf{W}_2)$:



Principles of gradient descent for tuning $\mathbf{W}$:

1. Start with a random $\mathbf{W}_0$
2. The variation of $\mathcal{L}$ around $\mathbf{W}_t$ provides the descent direction
3. Let's make a small step in this direction
4. While $\mathcal{L}(\mathbf{W})$ decrease, start again from (2)

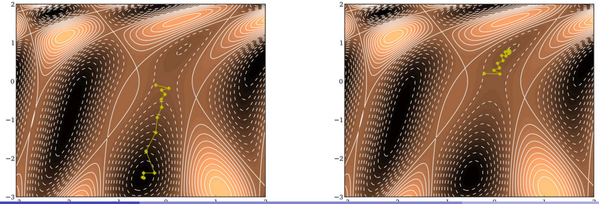
Gradient Descent : the algorithm

Iterative gradient descent for tuning $\mathbf{W}$

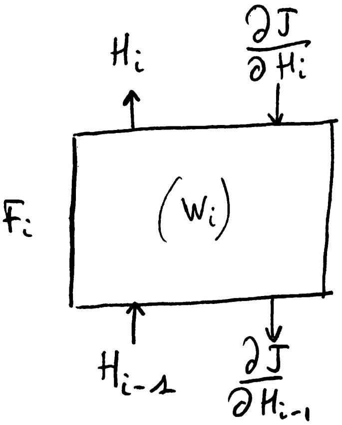
1. Start with a random $\mathbf{W}_0$
2. Compute the good direction $\frac{\mathrm{d}\mathcal{L}(\mathbf{W})}{\mathrm{d}\mathbf{W}}\Big|_{\mathbf{W}_t}$
3. Apply

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta \left.\frac{\mathrm{d}\mathcal{L}(\mathbf{W})}{\mathrm{d}\mathbf{W}}\right|_{\mathbf{W}_t}$$

   where η is the step size (not too small, nor too big ...)
4. While $\mathcal{L}$ decrease, start again from (2)



Gradient descent applied to NN (Y. Lecun [1])



Propagation

foreach sample
    for $i = 1$ to L
        $H_i = F_i(H_{i-1})$
        // either Dense, activation, etc.
    endfor
end foreach

Gradient Backpropagation

foreach sample
    for $i = L$ downto 1
        $\frac{\partial J}{\partial H_{i-1}} \leftarrow \frac{\partial J}{\partial H_i} \times \frac{\partial H_i}{\partial H_{i-1}}$
        $\frac{\partial J}{\partial W_i} \leftarrow \frac{\partial J}{\partial H_i} \times \frac{\partial H_i}{\partial W_i}$ // if necessary
    endfor
end foreach

1. https://www.college-de-france.fr/site/yann-lecun/course-2016-02-12-14h30.htm

Conclusion on feedforward NN

Resume

- **Universal approximation theorem** : "A unique non-linear layer is enoughto estimate any function f, provided we have an infinite number of neurons and an infinite number of example [Lippman 87]

Solution : Let's stack more layers !

☺Complex decision boundaries
☺High level data representation
☺Backpropagation theoretically OK
☹But practically fails to reach low layers due to **vanishing gradient** issues

$\rightarrow$ Deep learning !

Plan

Plan

What is deep learning ?

Deep means :

- Many layers (3 ... 100 ...), leading to big models : Millions of parameters
- Deep neural network example : VGG16 (2014, 138M parameters)
- Impressive results !

Training deep neural networks is hard :

- Need for expert to design the architecture and control learning
- Need for big datasets and big computers (GPU)
- Need for fighting against **vanishing gradient**

Vanishing gradient

- Saturated Neurons, gradient $\rightarrow 0$
- Inefficient backpropagation

Fighting against the vanishing gradient

Many theoretical/practical solutions

- Pretraining
- Extensive use of Convolution (few parameters)
- ReLU (provides better dynamic to the gradient)
- Regularization : Dropout, Batch normalization, Tikhonov
- Transfer Learning
- Feeling, experience & hyperparameter tuning

Not described in this short introduction to DL ☹

Need for big data ?

Rather **annotated** big data : ImageNet [Krizhevsky 2012]

- $>$ 14M images, 1000+ classes (objects, animals, natural scenes, etc.)
- RGB Images $512 * 512$

Object classification

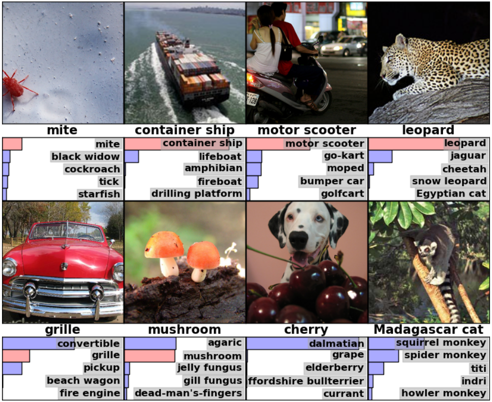
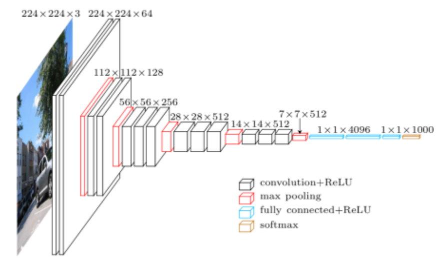
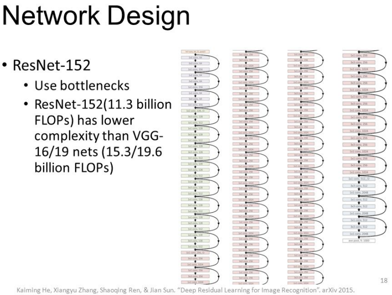


Figure – ImageNet [Krizhevsky 2012]

Famous architectures with high performance ImageNet

VGG16, VGG19, AlexNet, GoogleNet, Inception, ResNet ($L > 150$!), etc.



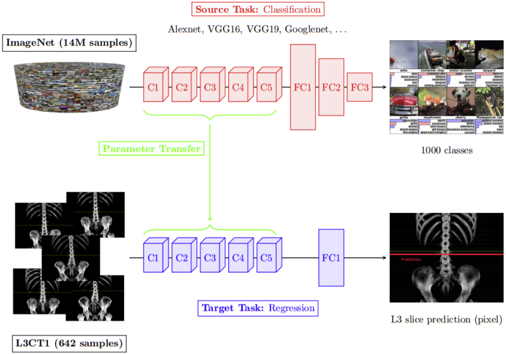


performance : error around 15 % in 2021 using EfficientNet, NFNet, etc.

Transfer learning : what if we only have few examples ?

Hack a classical net pretrained on a big dataset

- Drop the last decision layer, replace by a randomly initialized dense layer that suits your problem
- Finetune the whole architecture on your (small) dataset.
- Here we are, most of time it works !



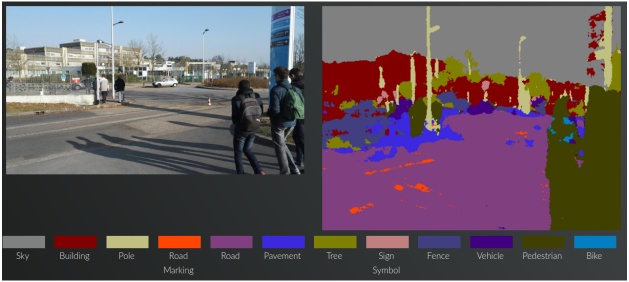
[Belharbi2017]

Plan

Semantic Segmentation (1)

Pixel labeling

- Many CV tasks : medical imaging, road safety, document, etc.
- Difficult problem : high dimension, structured output, often few pixel-labeled examples, etc.

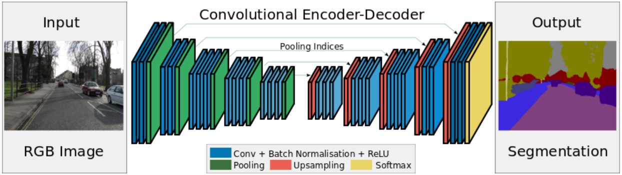


Now very efficiently achieved by neural networks since 2015 !

- Fully Convolutional Networks (FCN)
- Light, need few examples, easy to train ☺

Semantic Segmentation (2)

SegNet (démo sur `http://mi.eng.cam.ac.uk/projects/segnet/`)



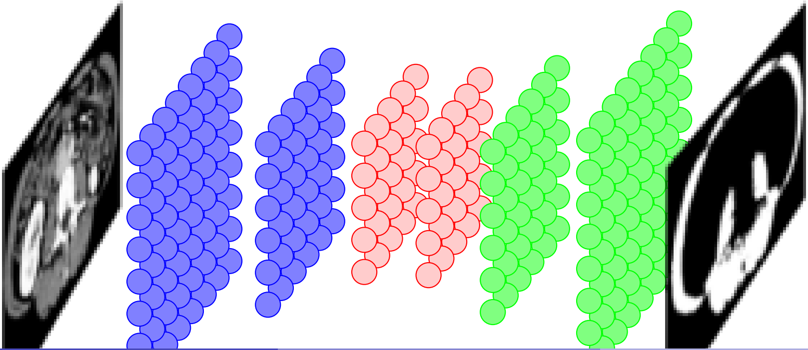
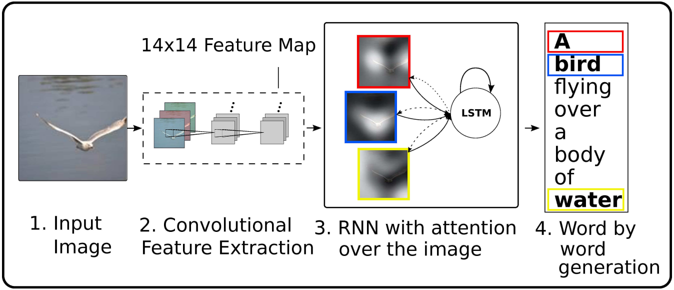


Image Captioning (1)

results from the Google system

Image Captioning (2)



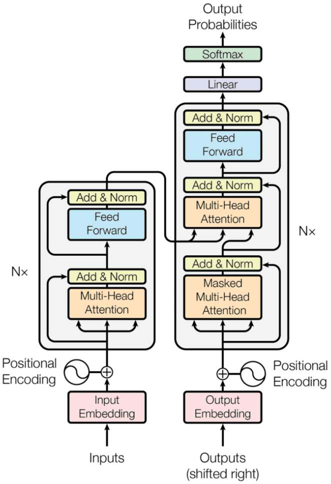
[Xu 2015]

CNN/LSTM generative architecture

- CNN layer extract features (pretrained on imageNet)
- LSTM layers generate the output signal (also pretrained)
- Backprop on the Coco dataset for linking both architectures

Transformer architecture

"Attention Is All You Need", A. Vaswani, N. Shazeer, N.Parmar et. al Neurips 2017

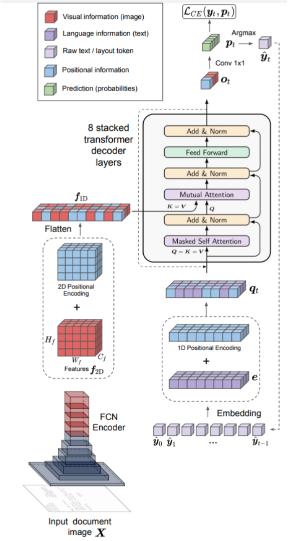


Main architecture for any sequence problem

- encoder = transformer layer
- decoder = transformer layer
- based on multihead attention and linear layers
- along with positional encoding

On the use of transformers

- Encoder can be replaced by another encoder (DAN)
- Encoder alone can be used (BERT)
- Decoder alone can be used (GPT)



Plan

In practice

∃ many neural networks libraries

- TensorFlow (Google, python) `https://www.tensorflow.org`
- Keras (Google, python) `https://keras.io/` (+ Theano ou TF)
- pytorch (Facebook, python) `http://pybrain.org/` (basé sur torch)
- ...

Keras code example for vgg16 : `code/vgg16.py`

Bibliography

Interesting links :

- Yann Lecun's course at collège de France (in french)
  https://www.college-de-france.fr/site/yann-lecun/
- Youtube channel of H. Larochelle : http://tinyurl.com/lpkvjm4
- "Deep Learning" book, Ian Goodfellow, Yoshua Bengio and Aaron Courville, MIT Press
  https://mitpress.mit.edu/books/deep-learning

See also ITI's department courses :

- EC "Machine Learning" (ITI4.1)
- EC "Deep Learning" (ITI4.2)
- EC "Advanced Machine Learning" (ITI5.1)
- EC "Advanced Deep Learning" (ITI5.1)

Bibliography

- F. Rosenblatt. Principles of Neurodynamics. New York : Spartan, 1962.
- C.M. Bishop. Neural networks for pattern recognition, Oxford : Oxford University Press, 1995.
- D.E. Rumelhart, G.E. Hinton and R.J. Williams. Learning internal representations by error propagation. Parallel Distributed Processing Explorations in the Microstructure of Cognition. MIT Press, Bradford Books, vol. 1, pp. 318-362, 1986.
- S. Belharbi, C. Chatelain, R. Hérault, S. Adam, S. Thureau, M. Chastan, and R. Modzelewski, ”Spotting L3 slice in CT scans using deep convolutional network and transfer learning”, Computers in Biology and Medicine, vol. 87, pp. 95-103, 2017.
- Soufiane Belharbi, Clément Chatelain, Romain Hérault, Sébastien Adam : Input/Output Deep Architecture for Structured Output Problems. CoRR abs/1504.07550 (2015)