Workshop "Robotics & AI"

Pasteur Paris University

Jean-Luc.Charles@ENSAM.EU

June 2022

Arts et Métiers Institute of Technology

poppy STATION

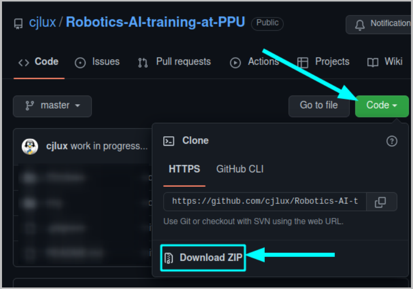
Welcome to the "Robotic & AI with Python" workshop

A three days workshop to get familiarised with...

- Scientific Python programming
- Robotics programming with Python
- Machine learning with Python
  - Images Classification
  - Objects Detection in images (possibly)

Welcome to the "Robotic & AI with Python" workshop

Get the zip file on github.com/cjlux/Robotics-AI-training-at-PPU:


cjlux / Robotics-AI-training-at-PPU
Public
Notification
Code
Issues
Pull requests
Actions
Projects
Wiki
master
Go to file
Code
cjlux work in progress...
Clone
HTTPS
GitHub CLI
https://github.com/cjlux/Robotics-AI-t
Use Git or checkout with SVN using the web URL.
Download ZIP

Extract the `Robotics-AI-training-at-PPU-master` directory in a convenient place...

Wake up your Python skills...

A programming language

- Proposed in the 90s by *Guido van Rossum* who chose the name Python in tribute to the *Monty Python* serie...
- Powerful, compact, visual, **interpreted**
- Full object oriented
- Multi plateforms: GNU/Linux, Mac OS X, Windows... and more!
- Free: distributed under the PSF(*Python Software Foundation*) licence

Wake up your Python skills...

We will use two types of IDE (*Integrated Development Environment*):

for editing native Python files <*.py>

- idlex: the simplest IDE in the world !
  one editor window & one interpreter window
- Visual Studio Code (a.k.a VSCode or simply "code"):
  a complete & powerfull free IDE by MicroSoft
- ...

for editing notebook files <*.ipynb>

- Jupyter notebook: Python cells within a web browser
- Jupyter lab: the same, plus some goodies (disk tree navigator panel...)
- ...

Wake up your Python skills...

We will use a **Python Virtual Environment** (PVE) for this workshop:

Benefits of a Python Virtual Environment

- Encapsulation in a dedicated and persistent environment.
- Specific versions of Python and all the needed modules.
- Independence from other Python installation(s) likely to coexist on the same machine.
- Independence from computer updates.
- Can be created, deleted, re-created... easily without impacting other Python installations.
- Simply based on a dedicated disk tree.

Wake up your Python skills...

Main themes of the Python training session

- Install a Python Virtual Environment for the workshop (possibly prepared in advance...)
- IDE installation & configuration: idlex, jupyter notebook, jupyter lab...
- Get familiar with the main Python object types, key words, useful modules...
- Make some calculus with the numpy module.
- Plot data with the matplotlib module.

Start the Python training session with **idlex**...

⤳ All the course material is in the directory `1-Python_training`...

Begin with `Create a Python Virtual Environnement.pdf`

Laptop under Windows

1. Install `miniconda3` and the `(pyml)` PVE.
2. Create the short-cut `(pyml)-idlex` on your desktop.
3. Doucle-clic on the icon `(pyml)-idlex` to start `idlex`

Laptop under Mac or Ubuntu

1. Install `miniconda3` and the `(pyml)` PVE.
2. Open a terminal, activate the `(pyml)` PVE and type `idlex`

you are ready to start the interactive Python training session with me !

Start the Python training session with **jupyter lab**...

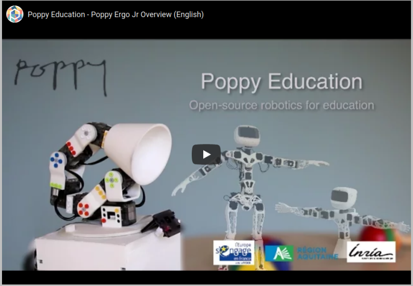
⤳ All the course material is in the directory `1-Python_training`...

Work to start

1. (Windows) ⤳ Open an `Anaconda prompt` window
   (Mac & Linux) ⤳ Open a terminal
2. Use the command `cd` (*change directory*) to go into the `Robotics-AI-training-at-PPU-master` directory.
3. Activate the `(pyml)` PVE.
4. Type the command `jupyter lab` to get Jupyter in a tab of your web browser...

you are ready to start the the jupyter lab self_training session :
see notebooks `BasicPythonTraining-1.ipynb` and `BasicPythonTraining-2.ipynb`...

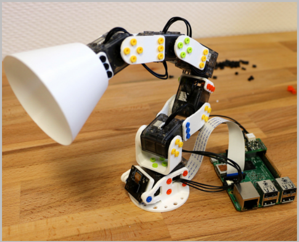
Open source Robotics with Poppy Project


Poppy Education - Poppy Ergo Jr Overview (English)
poppy
Poppy Education
Open-source robotics for education
Région Aquitaine
Inria

- **Poppy Ergo Jr** : a small and low cost 6-degree-of-freedom robot arm, easy to build and modify.
- The robot is **Open Source software and hardware.**
- Supports multiple programming modes:
  - Visual programming: with snap and scratch for programming initiation in schools
  - textual (object oriented) programming with Python.

Open source Robotics with Poppy Project

- Poppy Ergo Jr is controled by a Raspberry Pi 3 micro-computer that uses a SD card as a disk to boot a Linux-based operating system.
- The robot software is build upon the Pypot Python module.
- The robot is made of 6 XL-320 servomotors from Robotis.

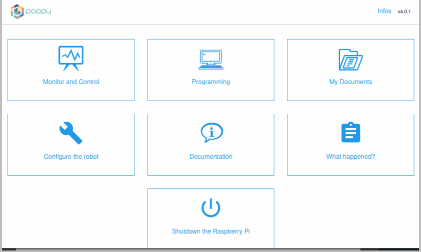


- Each servomotor embeds an electronic board that receives commands (position, speed, torque...) and communicate with other servos.
- You can chain up several servomotors and command them all from one end of the chain: each motor will pass the orders to the next one.
- By default the RPi3 of the Robot emits a WiFi acces point : you can connect your computer to this WiFi to communicate with the robot.

Open source Robotics with Poppy Project

Work to do for the Robotics session:

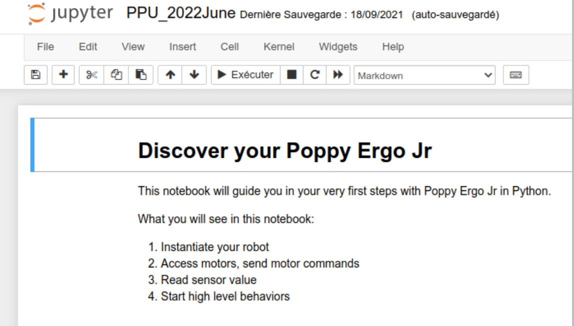
- Locate the number **n** (1 to 6) written on the base of the robot
- Power the RPi3: the green LED will blink for a while...
- Wait for the WiFi SSID `Poppy-Hotspot-`**n** to appear...
- Select the `Poppy-Hotspot-`**n** and connect your laptop with the key `poppyproject`
- Launch a web browser (preferably Chrome or Chromium) and open the URL `poppy.local`:
  you get the Poppy home page



Open source Robotics with Poppy Project

Work to do for the Robotics session:

- Clic on `My Document` : you get the jupyter notebook main page
- Open the directory `Python notebooks`
- Open the notebook `PPU_2022June.ipynb`...follow the guide...



jupyter
PPU_2022June Dernière Sauvegarde : 18/09/2021 (auto-sauvegardé)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Exécuter
Markdown

Discover your Poppy Ergo Jr

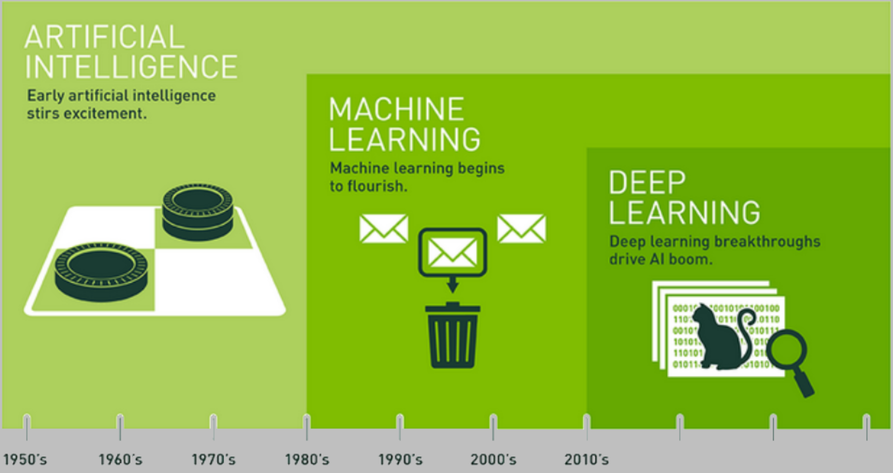
This notebook will guide you in your very first steps with Poppy Ergo Jr in Python.

What you will see in this notebook:

1. Instantiate your robot
2. Access motors, send motor commands
3. Read sensor value
4. Start high level behaviors

Machine learning with Python

Historical aspects...


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

(crédit : developer.nvidia.com/deep-learning)

Artificial Intelligence ?

Artificial Intelligence [1]: remains an ambiguous term with multiple definitions

- *"...the science of making computers do things that require intelligence when done by humans."* Alan Turing, 1940
- *"the field of study that gives computers the ability to learn without being explicitly programmed."* Arthur Samuel, 1960
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."* Tom Mitchell, 1997
- Notion of *intelligent agent* or *rational agent*
  "...agent that acts in such a way as to reach the best solution or, in an uncertain environment, the best predictable solution." Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015

[1] first used in 1956 by John McCarthy, researcher at Stanford during the Dartmouth conference

Artfificial Intelligences?

Strong AI

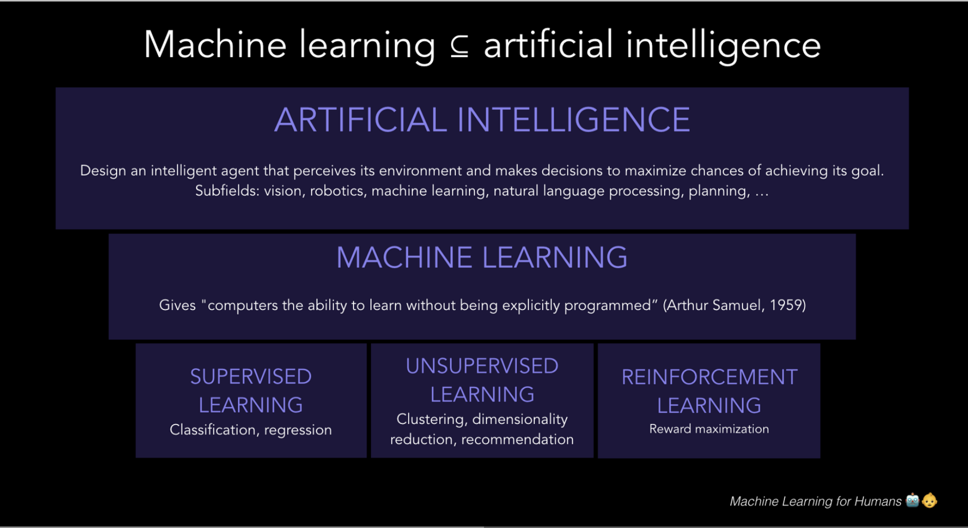
- Aims to design systems that think exactly like humans.
- May help explain how humans think...
- We’re still far away... do we really want to go that far?

Weak AI

- Aims to design systems that can “behave” like humans.
- Tells us nothing about how humans think.
- We’re already there... We use it every day!
  facial recognition, voice recognition, anti-spam, translation...

Machine Learning and AI

Page from medium.com/machine-learning-for-humans/...



Machine learning ⊆ artificial intelligence

ARTIFICIAL INTELLIGENCE

Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goal.
Subfields: vision, robotics, machine learning, natural language processing, planning, …

MACHINE LEARNING

Gives "computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)

SUPERVISED LEARNING
Classification, regression

UNSUPERVISED LEARNING
Clustering, dimensionality reduction, recommendation

REINFORCEMENT LEARNING
Reward maximization

Machine Learning for Humans 🤖👶

Machine Learning and AI

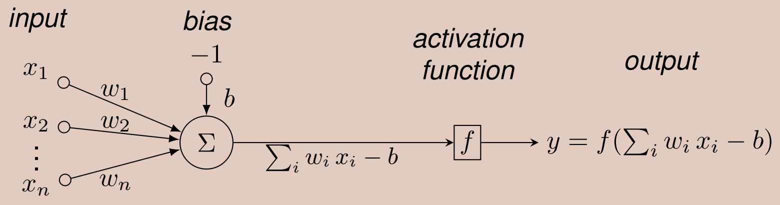
Several approaches can be used to design *Machine Learning* algorithms:

- Genetic programming
- Bayesian inference
- Fuzzy logic
- Neural Networks
- ...

The following deals only with **Artificial Neural Network**.

Artificial neuron

The computer model of the artificial neuron



A **artificial neuron**:

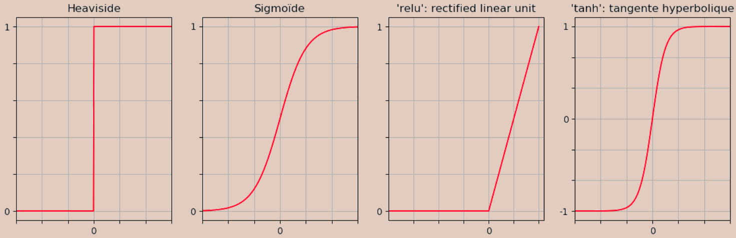
- receives the input data $(x_i)_{i=1..n}$ affected by the **weights** $(w_i)_{i=1..n}$ (*weights*)
- calculates the **weighted sum** of its entries minus the bias $\sum_i w_i\, x_i - b$
- outputs a **activation** $f(\sum_i w_i\, x_i - b)$, computed with an activation function f (generally non-linear).

Artificial neuron

The activation function of a neuron:

- introduces a non-linear behavior,
- sets the range of the neuron output, for example $[-1,1]$, $[0,1]$ or even $[0,\infty[$.

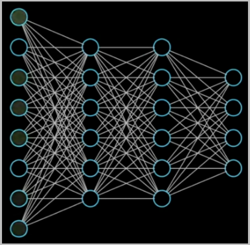
Examples of often used activation functions



- The bias b sets the activation threshold of the neuron.

Neural networks studied

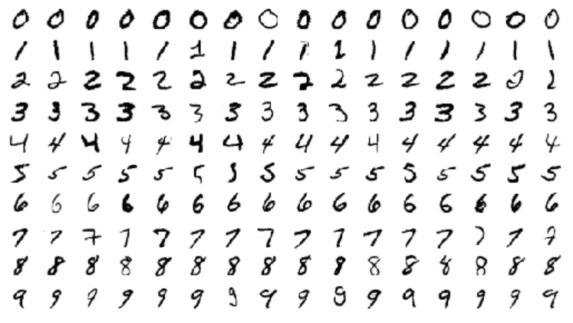
- Neural networks are more or less complex assemblies of artificial neurons.



- Two architectures are studied for image classification:
  - The **Dense Neura Network** (DNN), simple, generalist, can provide fairly good score.
  - The more complex **Convolutional Neural Network** (CNN), specialized in image processing, up to a score of 99%.

Data used to train networks

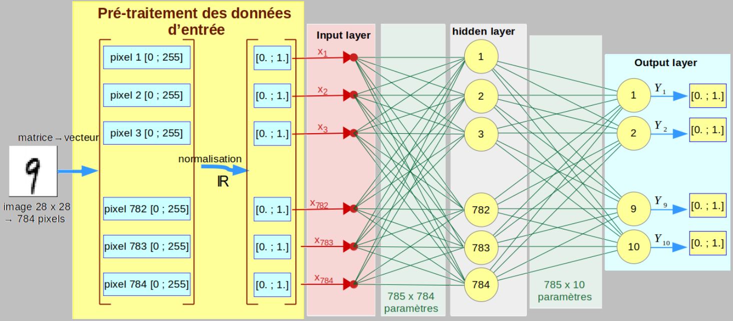
- MNIST: bank of 70000 **labeled images**



- grayscale images 28 × 28 pixels.
- 60000 training images and 10000 test images.

1 - Dense Neural Network

Each matrix 28 $\times$ 28 $\rightsquigarrow$ normalized vector of 784 components `float` $\in [0; 1]$.



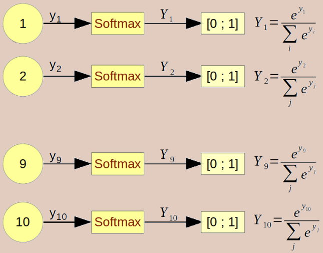
Structure of the network:

- An *Input layer* sets the size of network inputs to 784 values.
  It has no neurons.
- A *Hidden layer* of 784 neurons (we could have more, or less...), receives the input data. It is connected to the next layer.
- An *Output layer* of 10 neurons (1 neuron for each digit to be recognized).

1 - Dense Neural Network

- In the intermediate layers the activation function *relu* often favors the learning of the network [2] algorithm.
- Classification (last layer) uses the *softmax* function:

Activation function *softmax*


1
y_1
Softmax
Y_1
[0 ; 1]
$Y_1 = \frac{e^{y_1}}{\sum_i e^{y_i}}$
2
y_2
Softmax
Y_2
[0 ; 1]
$Y_2 = \frac{e^{y_2}}{\sum_j e^{y_j}}$
9
y_9
Softmax
Y_9
[0 ; 1]
$Y_9 = \frac{e^{y_9}}{\sum_j e^{y_j}}$
10
y_{10}
Softmax
Y_{10}
[0 ; 1]
$Y_{10} = \frac{e^{y_{10}}}{\sum_j e^{y_j}}$

- The activation of neuron k is $Y_k = e^{y_k} / \sum_i e^{y_i}$ with $y_k = \sum_i \omega_i x_i - b$ calculated by the neuron k.
- The outputs of the neurons are interpreted as probabilities in the interval [0,1].

The neuron with the greatest probability (activation) gives the response of the network by its associated label.

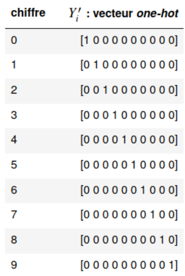
[2] avoids the *vanishing gradient* that appears in the *back propagation*

1 - Dense Neural Network

One-hot encoding of labels

Purpose: to put the image labels in the format of the network output

- Image labels: **integers** from 0 to 9.
- Network output: **vector of 10 `float`** in the interval [0,1] calculated by the *softmax* functions of the 10 output neurons.
- *one-hot* coding of an ordered collection of N unique elements:



chiffre	Y_i' : vecteur *one-hot*
0	[1 0 0 0 0 0 0 0 0 0]
1	[0 1 0 0 0 0 0 0 0 0]
2	[0 0 1 0 0 0 0 0 0 0]
3	[0 0 0 1 0 0 0 0 0 0]
4	[0 0 0 0 1 0 0 0 0 0]
5	[0 0 0 0 0 1 0 0 0 0]
6	[0 0 0 0 0 0 1 0 0 0]
7	[0 0 0 0 0 0 0 1 0 0]
8	[0 0 0 0 0 0 0 0 1 0]
9	[0 0 0 0 0 0 0 0 0 1]

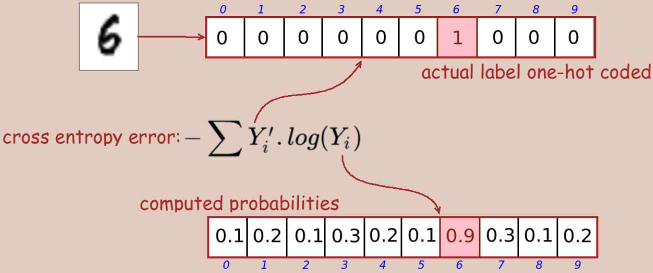
- each element is coded by a vector of N null components except one,
- the *i th* element $\leadsto$ vector with a 1 for *i th* component.

The *one-hot* encoding of labels '0' to '9' results in a 10-component vector, like the one computed by the neural network.

1 - Dense Neural Network

Error function: *Cross entropy error*

- An image processed by the network $\rightsquigarrow$ vector Y of 10 `float` to compare to the *hot-one* encoding Y' of the label of the image.
- We use the error (or loss) function *cross entropy* adapted to the coding *one-hot*: $e(Y, Y') = -\sum_i Y_i^{'} \; log(Y_i)$



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

actual label one-hot coded

cross entropy error: $-\sum Y_i'.log(Y_i)$

computed probabilities

0.1	0.2	0.1	0.3	0.2	0.1	0.9	0.3	0.1	0.2
0	1	2	3	4	5	6	7	8	9

1 - Dense Neural Network

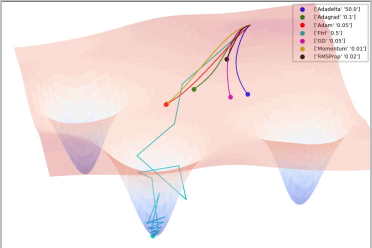
Optimization and *Back Propagation*

- During the learning phase an optimization algorithm calculates the gradient of the loss function relative to the network weights.
- The *Back Propagation* algorithm **modifies** the weights of the network layer by layer thanks to the gradient of the loss function, iterating from the last layer to the first layer.
- Examples of optimization algorithm used:
  - *Gradient Descent* (GD)
  - *Stochastic Gradient Descent* (SGD)
  - *Adam* (enhanced version of gradient descent)...

The module tf.keras.optimizers offers Python implementation of several optimization algorithms.

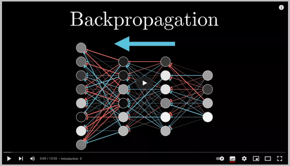
1 - Dense Neural Network

Visualization of gradient descent algorithm iterations for an ultra-simple loss function with only 2 variables:



(source: github.com/Jaewan-Yun/optimizer-visualization)

back propagation algorithm explanation video:



1 - Dense Neural Network

Implementation for the workshop

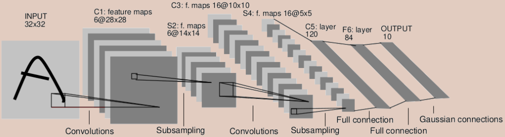
- The three *notebooks* `ML1_MNIST.ipynb`, `ML2_DNN.ipynb` and `ML3_DNNipynb` target the skill:
  - load and pre-process MNIST images,
  - build a **dense** neural network,
  - train the network to recognize MNIST images,
  - evaluate and operate the trained network.
- The Python modules used to create and train the neural networks are tensorflow and keras.
- Scores obtained with dense networks can reach 98% success in the most favorable cases.

2 - Convolutional Neural Network

To significantly improve the success score, it is necessary to switch to networks specialized in image processing: **convolutional neural networks** (RNC), or *Convolutional Neural Network (CNN)*.

Implementation for the workshop

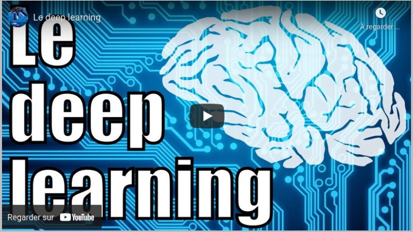
- The *notebook* `ML4_CNN.ipynb` targets the skills:
  - build a **convolutional** neural network inspired by the **LeNet5** network (one of the first RNCs proposed by Yann LeCun *et al.* in the 90s),



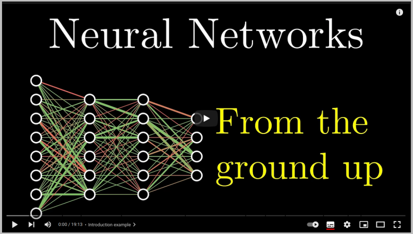
Yann Lecun *et al.*, 1998, "Gradient-based learning applied to document recognition", Proceedings of the IEEE. 86 (11)

  - train the network to recognize MNIST images,
  - evaluate and operate the trained network.

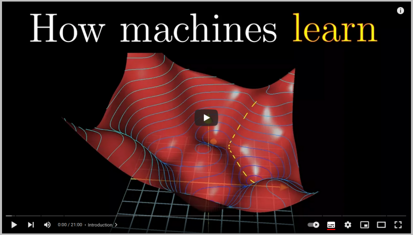
Vidéographie



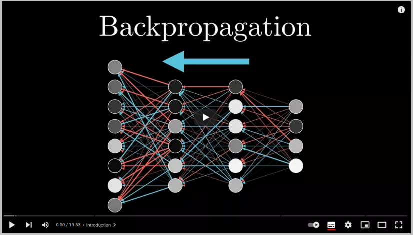
1/ Local: "Le deep learning - YouTube.webm"



2/ local : "But what is a neural network.webm"



3/ Local: "Gradient descent how neural networks learn.webm"



4/ Local: "What is backpropagation really doing .webm"

Biliographie

[1] *Intelligence Artificielle*, 3e édition, PEARSON Education, 2010, ISBN : 2-7440–7455–4, aima.cs.berkeley.edu

[2] *What is artificial intelligence (AI), and what is the difference between general AI and narrow AI?*, Kris Hammond, 2015 www.computerworld.com/article/2906336/what-is-artificial-intelligence.html

[3] *Stanford Encyclopedia of Philosophy*, plato.stanford.edu/entries/artificial-intelligence

[4] *Deep Learning.*, Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016), MIT Pres, ISBN 9780262035613