Introduction to Deep Learning

Presentation and history of Deep Learning

J. Rynkiewicz

Université Paris 1

2022

2012 : The Big Bang of Deep Learning

- Artificial neural network models have been around for a long time (over 60 years), but they fell into disuse in the early 2000s.
- In 2012, at the largest computer vision conference a deep neural network crushed all other techniques.
- See (in french) Dominique Cardon, Jean-Philippe Cointet et Antoine Mazières, « La revanche des neurones. L'invention des machines inductives et la controverse de l'intelligence artificielle », Réseaux 2018/5 (n° 211), p. 173-220.
- Since then, Deep Learning is the state of the art in many fields.

Deep learning

Several families of deep neural networks are currently used in practice :

- Convolutional networks are the state of the art for image classification.
- Recurrent networks that can model time series. They have long been used for natural language processing (speech recognition, translation, etc...).
- The "transformers" which are the state of the art for natural language processing.

This course aims to study these families of deep neural networks. There are other interesting models but they are not studied in this course :

- Adversarial generative models (for image synthesis).
- Learning by reinforcement (AlphaGo, AlphaZero, etc...)

Learning algorithm

We must predict to which class a random variable belongs Y, from the value of explanatory data X.

$X =$  $\Longrightarrow$ $Y =$ “cat”

- The parameters of the neural network are estimated on a training dataset. It must make few classification errors on this set.
- The model must "generalize" well : It must make few errors on new data.
- The difference between the learning error and the generalization error is called "overfitting".

The plague of large dimension

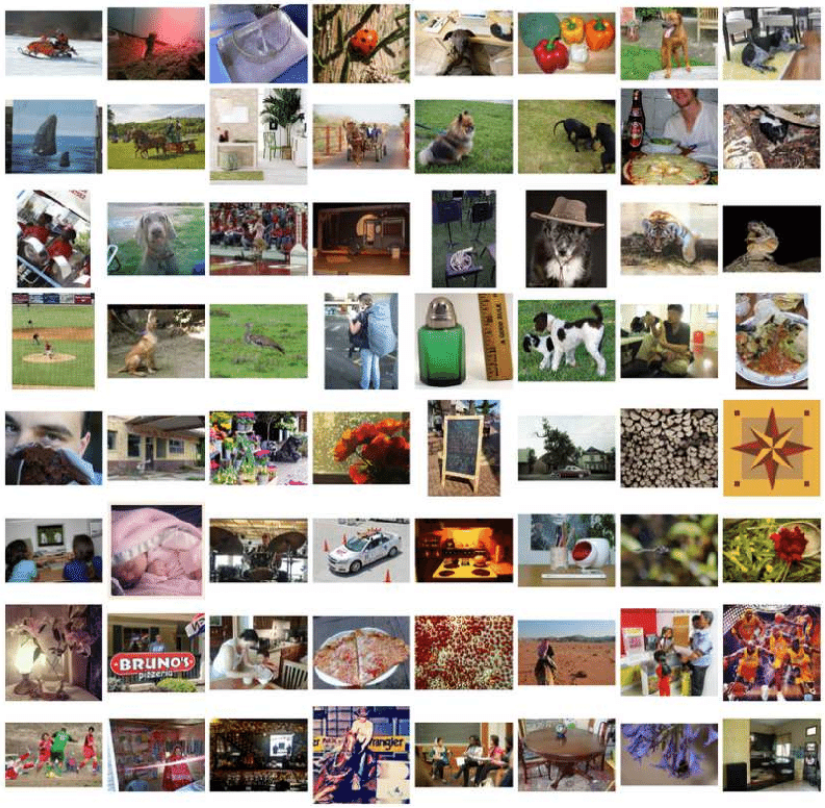
- The classification function $x \mapsto f(x)$ can be approximated with examples by local interpolation :

$$\left\{ \begin{pmatrix} x_1 \\ f(x_1) \end{pmatrix}, \dots, \begin{pmatrix} x_n \\ f(x_n) \end{pmatrix} \right\}$$

  So, if x is close to x_i, we can guess that $f(x)$ will be close to $f(x_i)$.
- For images, the d dimension of x ranges from a few thousand to a few hundred thousand.
- To cover $[0,1]^d$ with balls of radius 0.1, you need 10^d points, but there are less than 10^{100} atoms in the universe...
- The Euclidean distance between two different images of cats is very large.
- The model must discover other regularities than proximity of Euclidean distance.

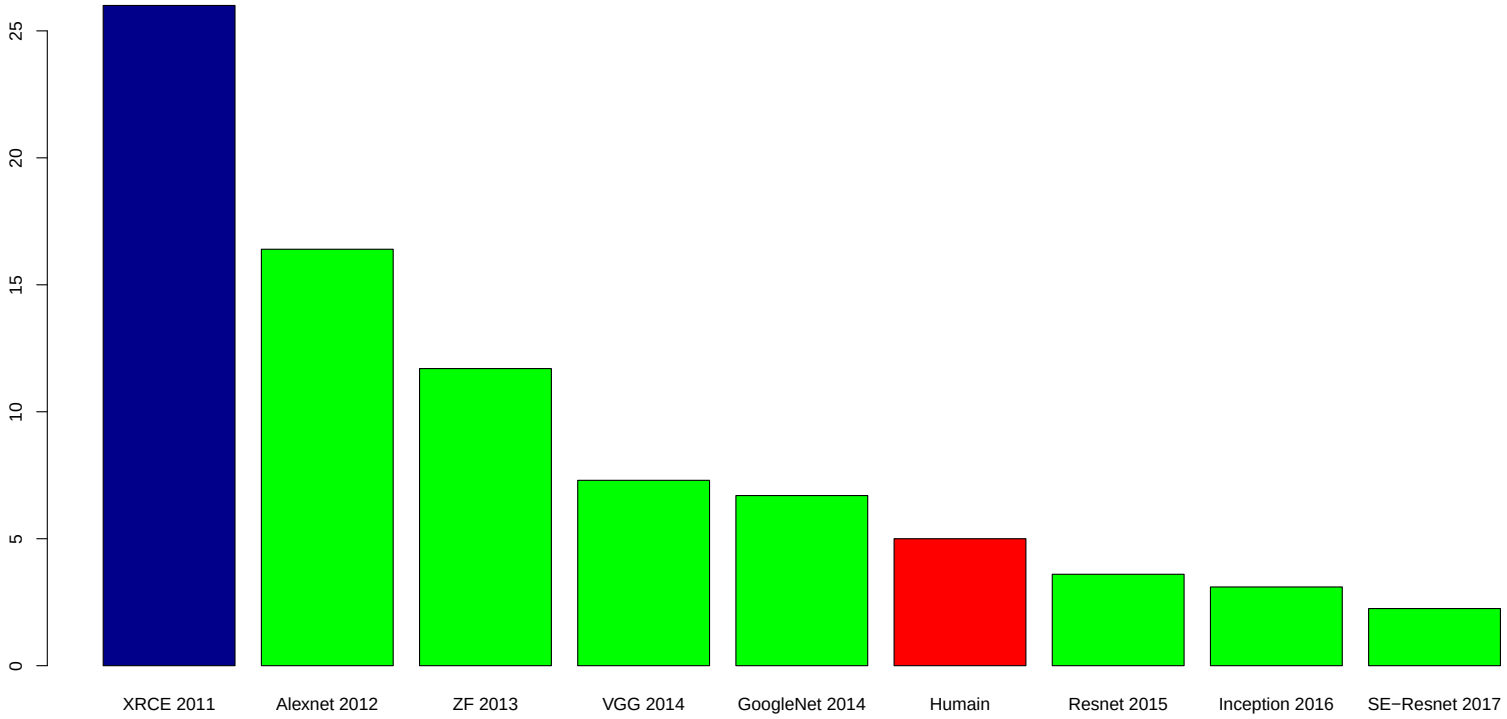
Imagenet competition

- Image classification competition on a database with $1,4$ million images and 1000 classes.



Deep learning and Imagenet

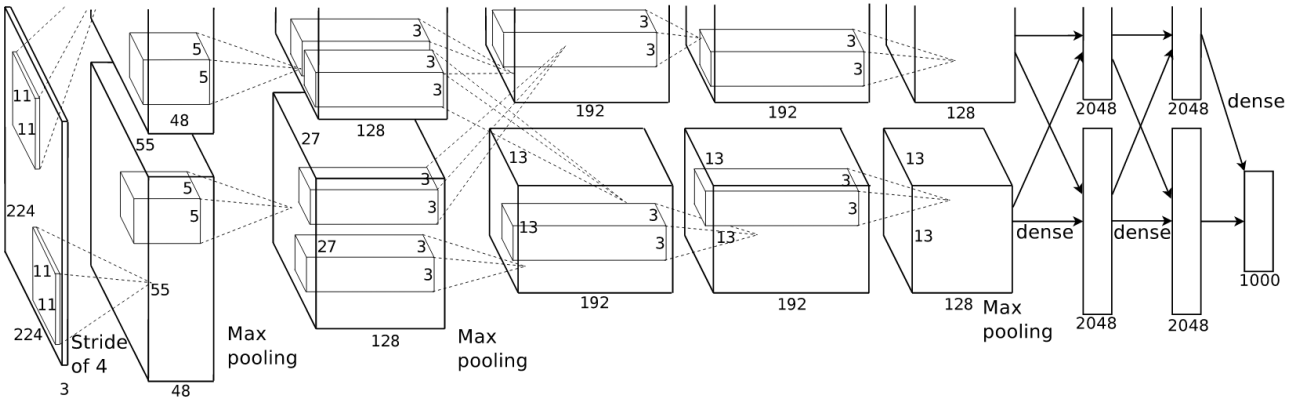
- Classification error rate on Imagenet (top 5).
- In blue : before Deep-Learning, in green : Models from Deep Learning, in red : human being.



Alexnet (2012)

It opened the door to deep learning models for image recognition.

- This network starts with convolutional layers and max-pooling.
- It ends with dense layers.
- It has been optimized using graphics cards with massively parallel computing algorithms.



Formalization of supervised classification (1)

It is a generalization of models to predict categorical data (logistic regression,...).

- We seek to predict the probability of a categorical variable Y knowing an explanatory variable X : $P(Y|X)$.
- We recall that, for $k \in \{0, \cdots, K-1\}$, the generalized linear model is written :

$$P_\theta(Y = k|X) = \frac{\exp(X^T \beta_k)}{1 + \sum_{l=1}^{K-1} \exp(X^T \beta_l)}$$

  with , $\theta = (\beta_1, \cdots, \beta_{K-1})$.
- For the Imagenet competition, Y is in $\{1, \cdots, 1000\}$ and X is in $\mathbb{R}^{3\times 256\times 256}$. The dimension of the parameter of the generalized linear model would thus be $999 \times 256 \times 256 \times 3 \simeq 200 \times 10^6$.

Formalization of supervised classification (2)

For a neural network, we replace the linear function $X^T\beta_k$ by a non-linear function $F_{\beta_k}(X)$.

- A neural network $f_\theta(.)$ seeks to estimate $f_\theta(X) = P_\theta(Y|X) \simeq P(Y|X)$ with

$$P_\theta(Y = k|X) = \frac{\exp(f_\theta^k(X))}{\sum_{l=1}^K \exp(f_\theta^l(X))}$$

  Here, $f_\theta(X) = (f_\theta^1(X), \cdots, f_\theta^K(X))$ is a neural network with an output of dimension K.
- In the case of a neural network, θ is also called the set of weights. These weights will be optimized, numerically, to maximize the likelihood of the model using a gradient descent.
- We will estimate $\widehat{\theta}_n = \arg\min - \sum_{t=1}^n \log P_\theta(y_i|x_i)$ with learning data $\left(\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \cdots, \begin{pmatrix} x_n \\ y_n \end{pmatrix}\right)$.
- As the θ parameter is of very large dimension (several millions), there is a risk of overfit the training data. The model will therefore be evaluated on completely new data, the test data : $\left(\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix}, \cdots, \begin{pmatrix} x_{n+T} \\ y_{n+T} \end{pmatrix}\right)$.

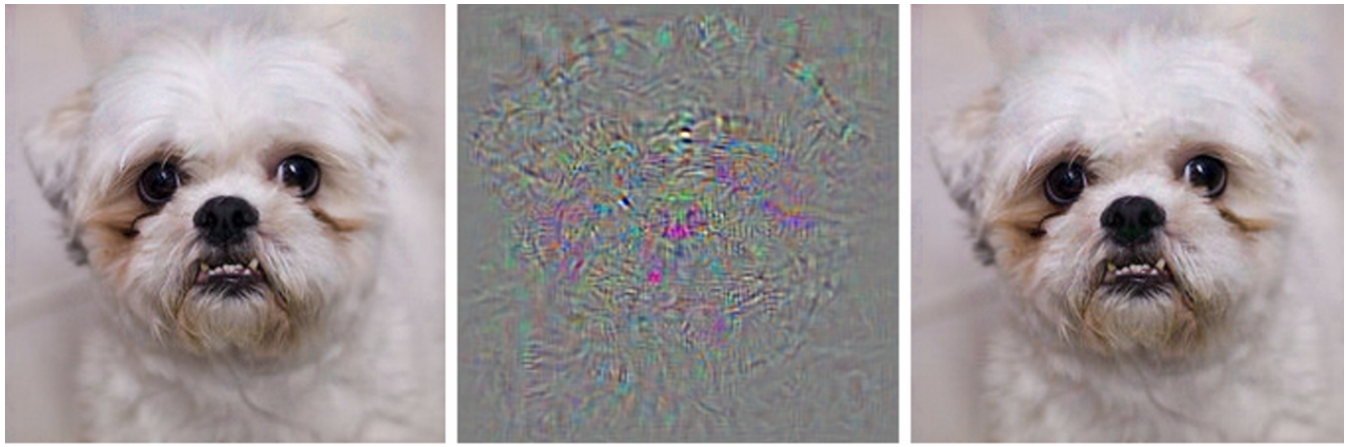
Differences with classical statistical models

The classical framework provides little information about deep networks.

- The number of parameters (the number of weights) can be larger than the number of data available for training. In this case, the asymptotic theorems are not necessarily justified.
- To avoid to overfit the training dataset will often be split into a training set to estimate the parameters $\widehat{\theta}_n$ and a validation set to know when to stop weights optimization.
- To have good results, deep networks must be very well structured (convolutions of good size, short-cut connections, pooling functions...).
- The optimization is done using the stochastic gradient algorithm, on mini-batches.

Fragility of Deep Learning models

Even if these models can be very efficient, it is rather difficult to have confidence limits for their predictions :

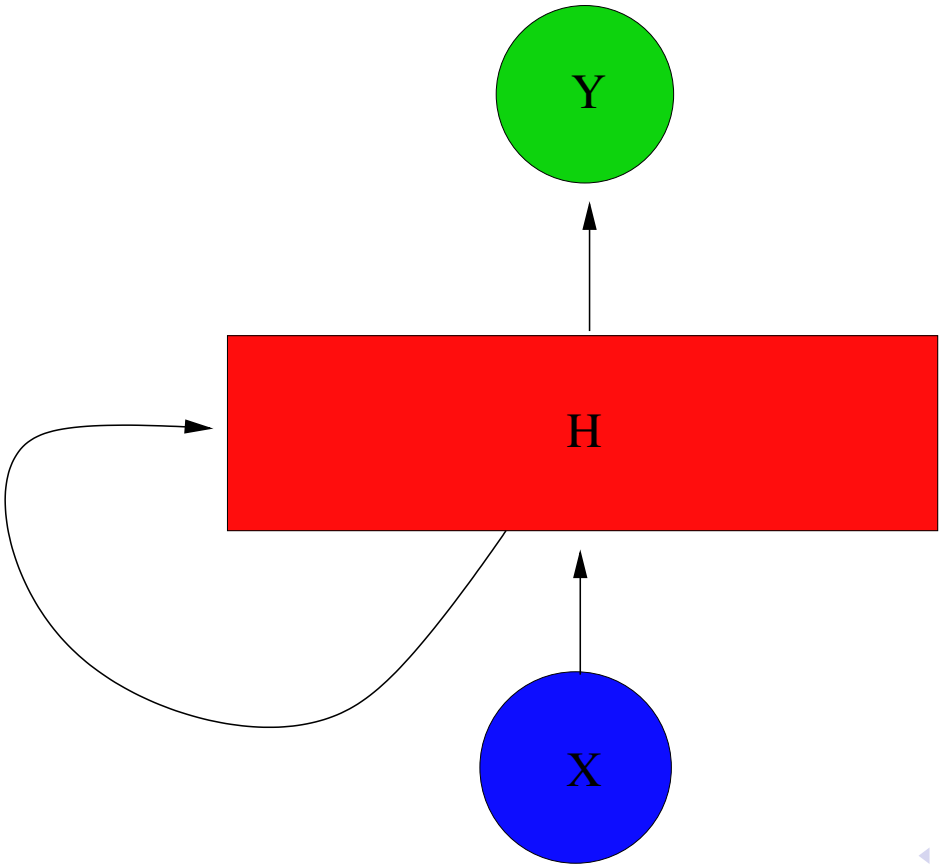


dog +noise ostrich

Recurrent neural networks

These neural networks have been used for time series modeling. They have been supplanted by transformers for natural language processing. Their principle is to take into account the state of the hidden layer at the previous time to influence the hidden layer at the present time :

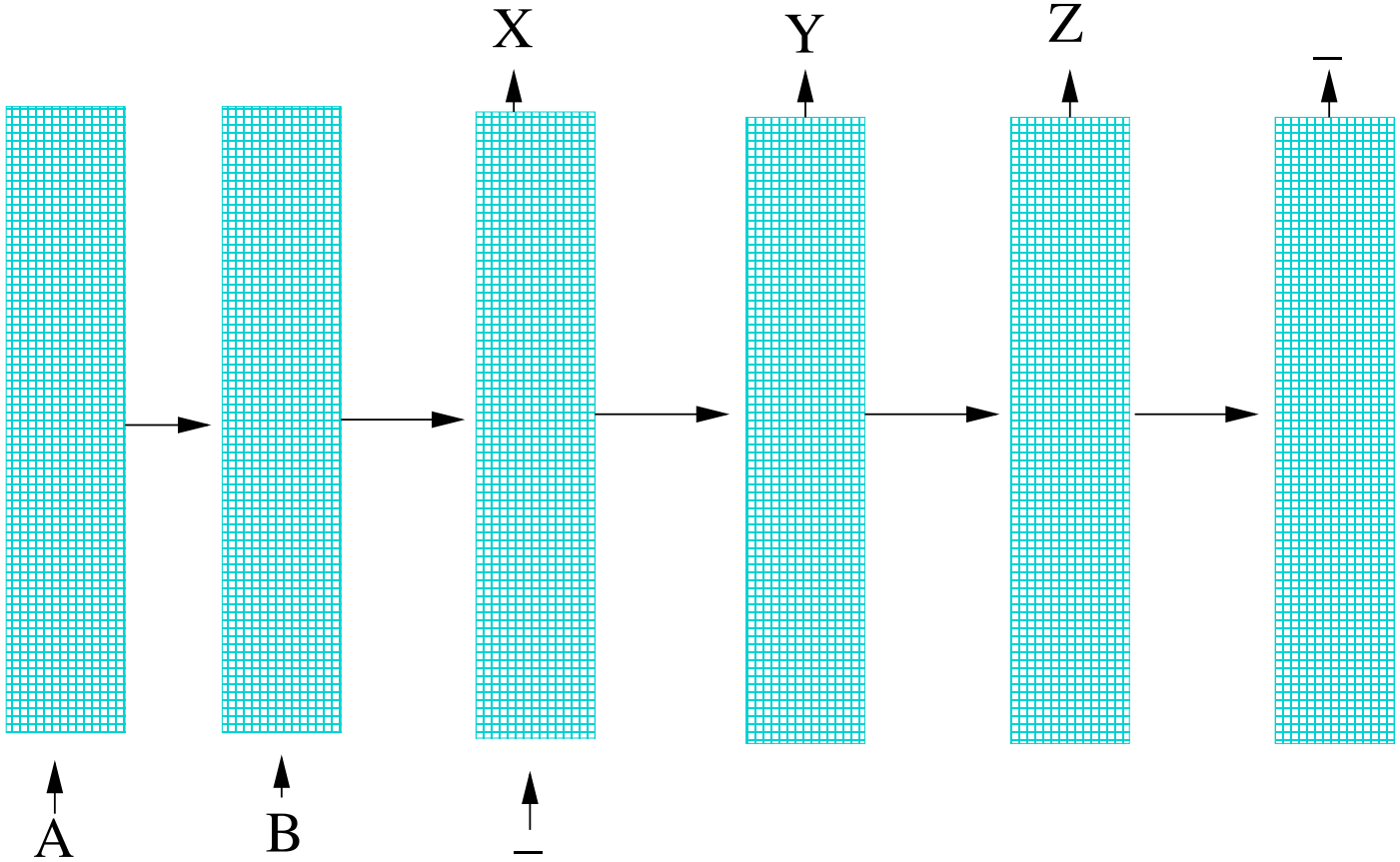
$$\begin{cases} \hat{Y}_t = f_\theta(h_t) \\ h_t = g_\theta(X_t, h_{t-1}) \end{cases}$$



Properties of recurrent neural networks

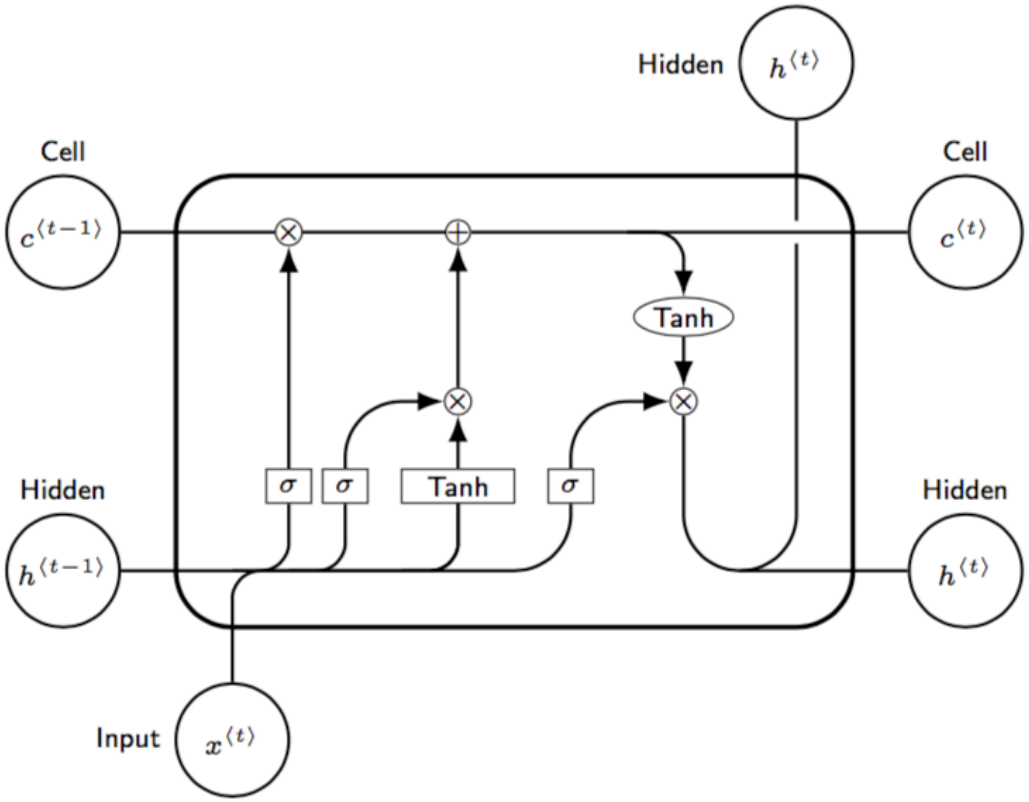
Recurrent networks are particularly suitable for time series :

- They can memorize past observations.
- The inputs and outputs of these networks can be of variable length.



Long Short Time Memory (LSTM) model

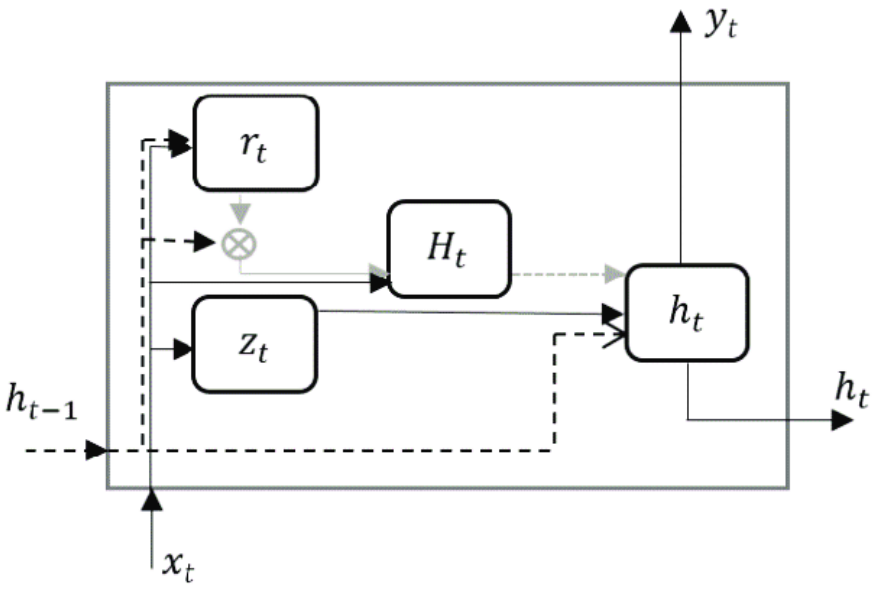
Certainly one of the most used deep models for time series is the LSTM model. This one allows the network to learn the depth of the useful memory for



a given task.

Gated Recurrent Unit (GRU) model

It is a simpler model than the LSTM model and has comparable performance



to the LSTM.

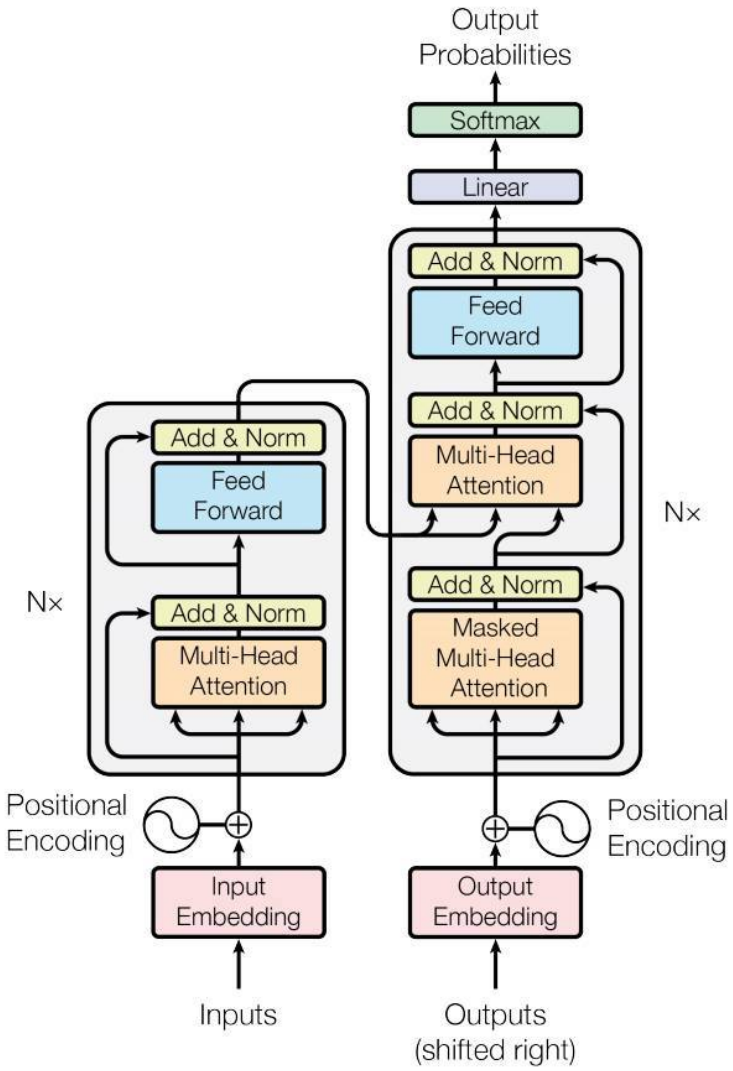
The transformers

These neural networks are the state of the art for natural language processing (Chatbot, sentiment analysis, translation, etc.). Here is an example of the translation from French to English of an excerpt from Wikipedia by "Google trad" :

- Un réseau de neurones récurrents est un réseau de neurones artificiels présentant des connexions récurrentes. Un réseau de neurones récurrents est constitué d'unités (neurones) interconnectés interagissant non-linéairement et pour lequel il existe au moins un cycle dans la structure. Les unités sont reliées par des arcs (synapses) qui possèdent un poids. La sortie d'un neurone est une combinaison non linéaire de ses entrées.
- A network of recurrent neurons is a network of artificial neurons with recurrent connections. A network of recurrent neurons consists of interconnected units (neurons) interacting non-linearly and for which there is at least one cycle in the structure. The units are connected by arches (synapses) that have a weight. The output of a neuron is a nonlinear combination of its inputs.

Principle of Transformers

These networks have attention modules that take into account the context of words.



Pytorch

Pytorch is a free Python library for building Deep Learning models.

- This python library comes from the research teams of Facebook.
- It performs optimized tensor calculations both on the CPU and on the graphics card (GPU).
- It is dynamic, at any time you can test a small part of the program (like R).
- It allows to exchange matrices and data tables easily with other Python libraries (numpy, etc...)
- The calculation of gradients for optimization is automated and implicit.
- The biggest difficulty in learning this library is to understand that all calculations must be able to run in a massively parallel way.

Tensorflow (Keras)

Tensorflow is also a free Python library for building Deep Learning models.

- This python library comes from Google's research teams.
- It performs optimized tensor calculations both on the CPU and on the graphics card (GPU) or TPU.
- It is static, so you have to run the whole program to see the results, which makes it more difficult to debug than Pytorch.
- It allows to exchange matrices and data tables easily with other Python libraries (numpy, etc...)
- The computation of gradients for optimization is automated and implicit. We can use its Keras "overlay" which makes it easier to program the models.
- Keras is simple and efficient for standard models but it is less flexible than Pytorch if you want to make more exotic models.

References

- Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016.
  - There is a French translation.
  - Associated website : https ://www.deeplearningbook.org/
- Courses at the Collège de France :
  - Yann Lecun : https ://www.college-de-france.fr/site/yann-lecun/course-2015-2016.htm
  - Stéphane Mallat : https ://www.college-de-france.fr/site/stephane-mallat/course.htm
- Pytorch : https ://pytorch.org/
  - Here you can download the program and install it on your computer.
  - Find the documentation.
  - Find tutorials.
- Tensorflow : https ://www.tensorflow.org/
  - Here you can download the program and install it on your computer.
  - Find the documentation.
  - Find tutorials.