Deep Learning Architectures

Bart.Lamiroy@univ-reims.fr

January 2022

Université de Reims Champagne-Ardenne

General Goal

Understand the main families of deep artificial neural network architectures.

More specifically :

- How their structure impacts their functions
- How they are trained
- What they are used for

Outline

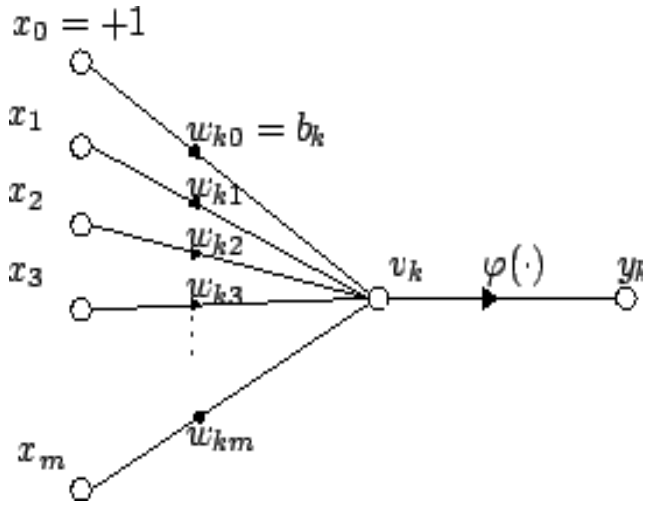
- Vocabulary and Scientific Background
- Quick recap of artificial neural networks basics
- Deep neural network categories
- Applications

Vocabulary and Scientific Background

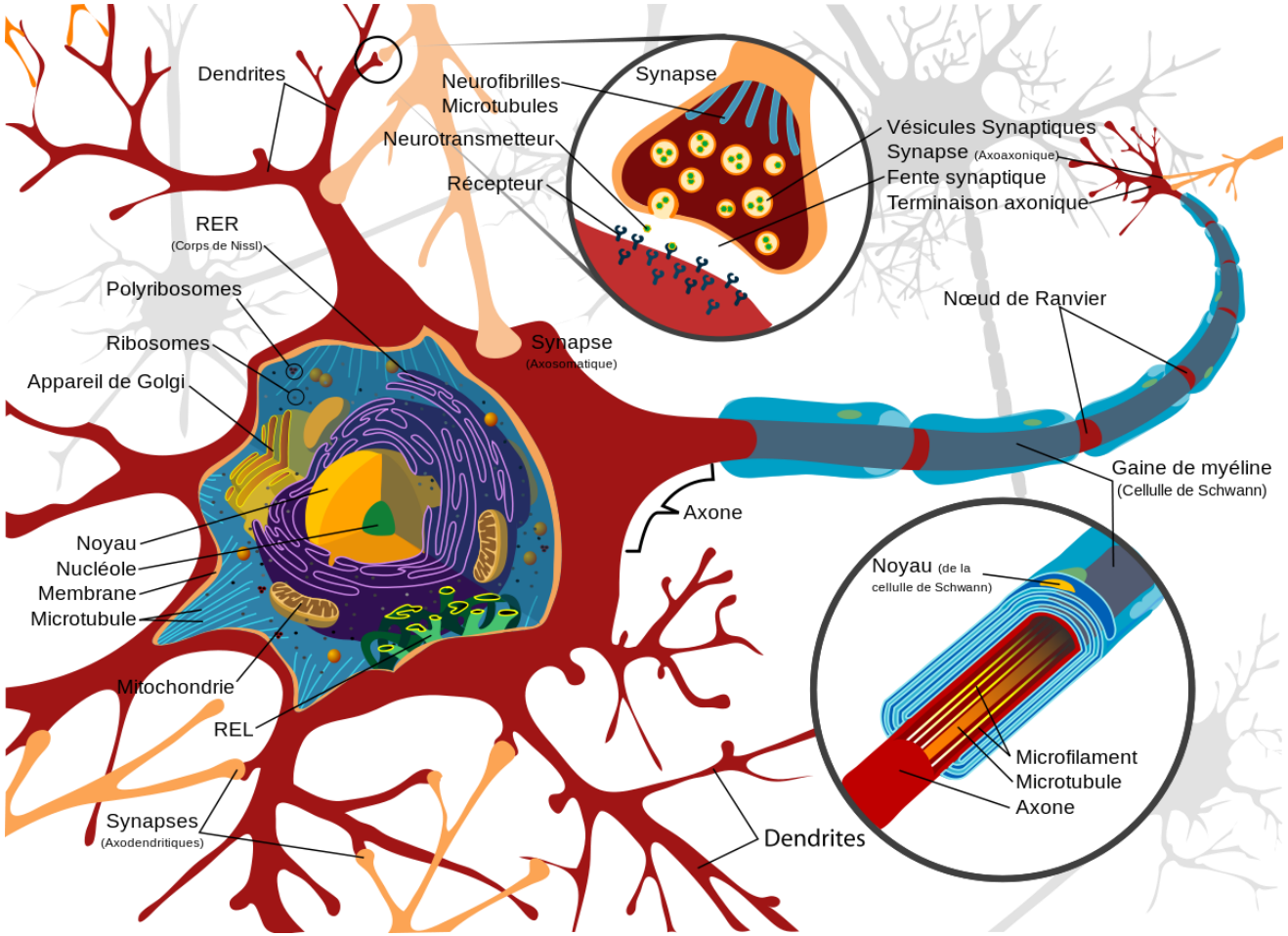
- Supervised Learning
- Self-Supervised Learning
- Non Supervised Learning

General Idea Behind Neural Networks

- Connexionism
- Transfer Function (activation function, treshhold function)
- Loss Function
- Gradient Back-Propagation
- Universal Approximation Theorem

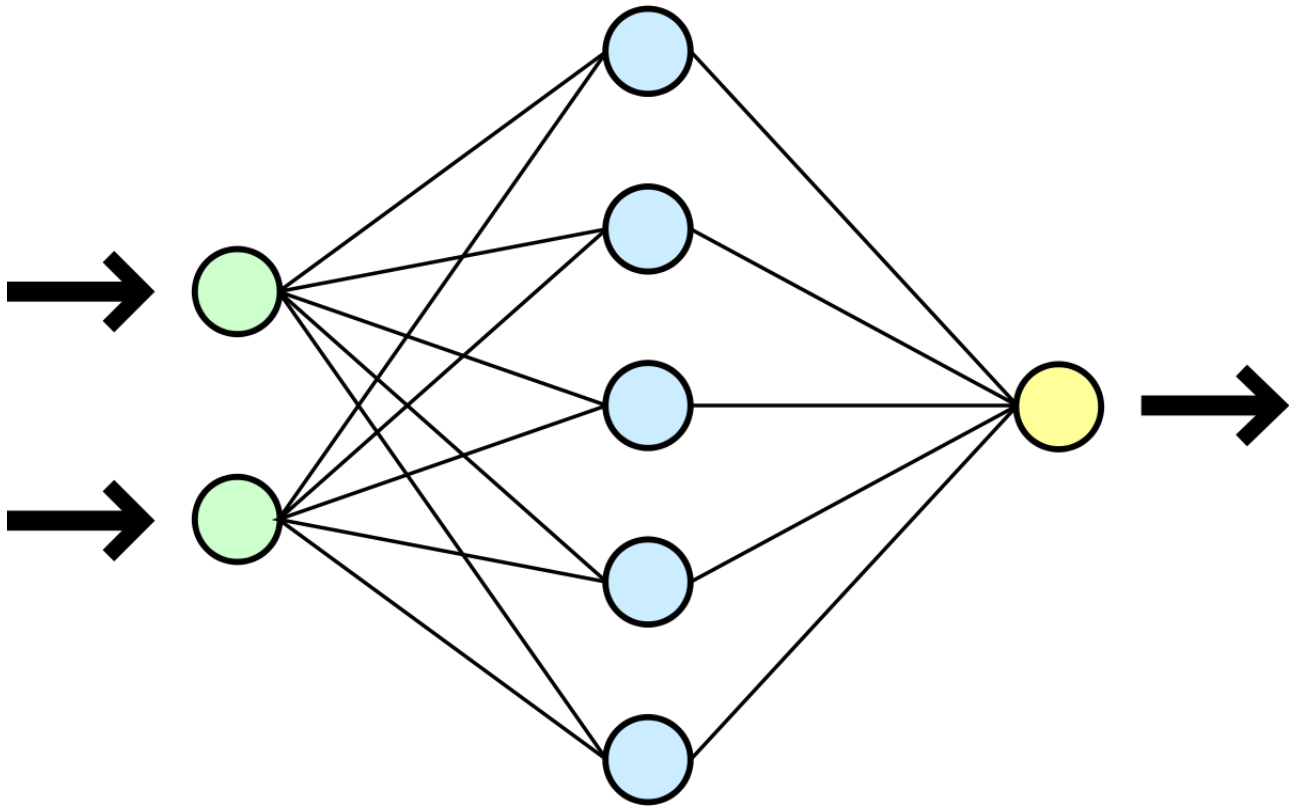


Biological Neuron and its Properties



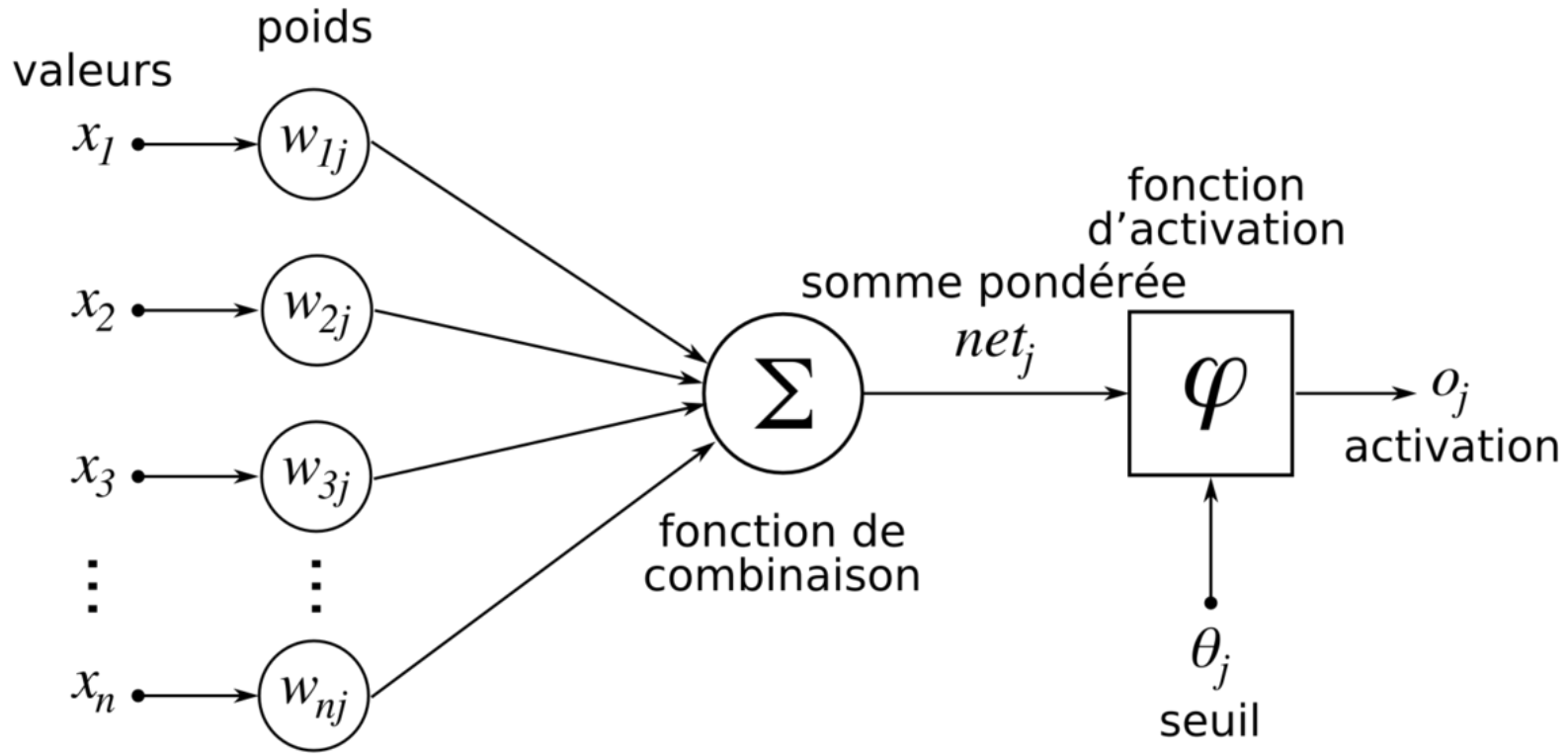
Source : Wikipedia

A Simplified Model : Oriented Acyclic Graph



Source : Wikipedia

Artificial Neuron

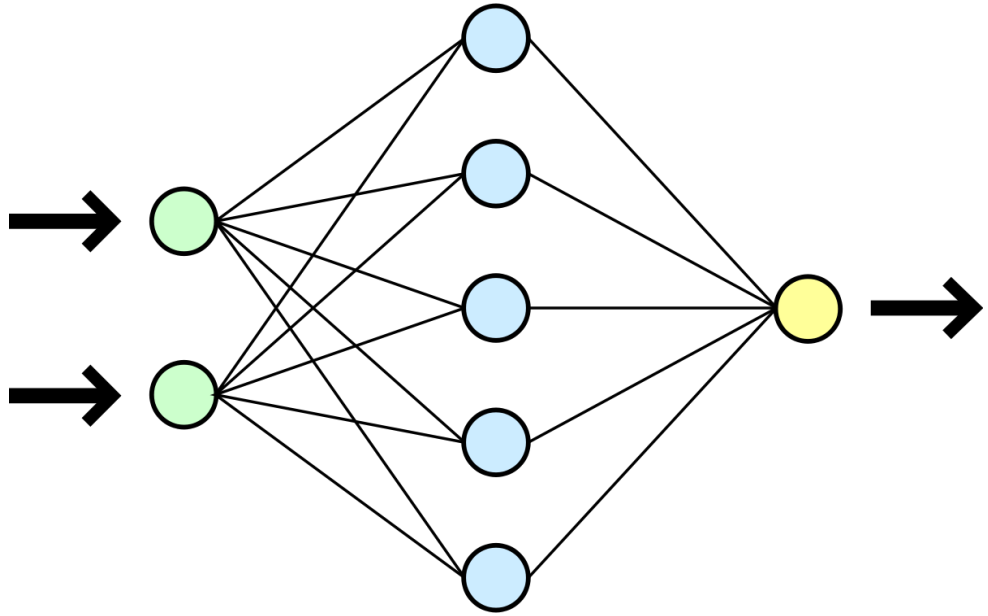


F. Rosenblatt (1958), "The perceptron: a probabilistic model for information storage and organization in the brain", repris dans J.A. Anderson & E. Rosenfeld (1988), Neurocomputing. Foundations of Research, MIT Press

Source : Wikipedia

Feed-forward

- Scalar products
- Highly Scalable through paralellization
- SIMD operations (very well suited for GPU)
- Universal approximator



Universal Approximator Theorem

- Given $\varphi()$ a continuous, strictly increasing bounded function;
- let I_m be the unit hypercube of dimension m and $C(I_m)$ the set of continuous functions over I_m;
- then for any $\varepsilon > 0$ there exists a natural number N such that for any function f in $C(I_m)$ there exists $v_i, b_i \in \mathbb{R}$ and $w_i \in \mathbb{R}^m$ such that

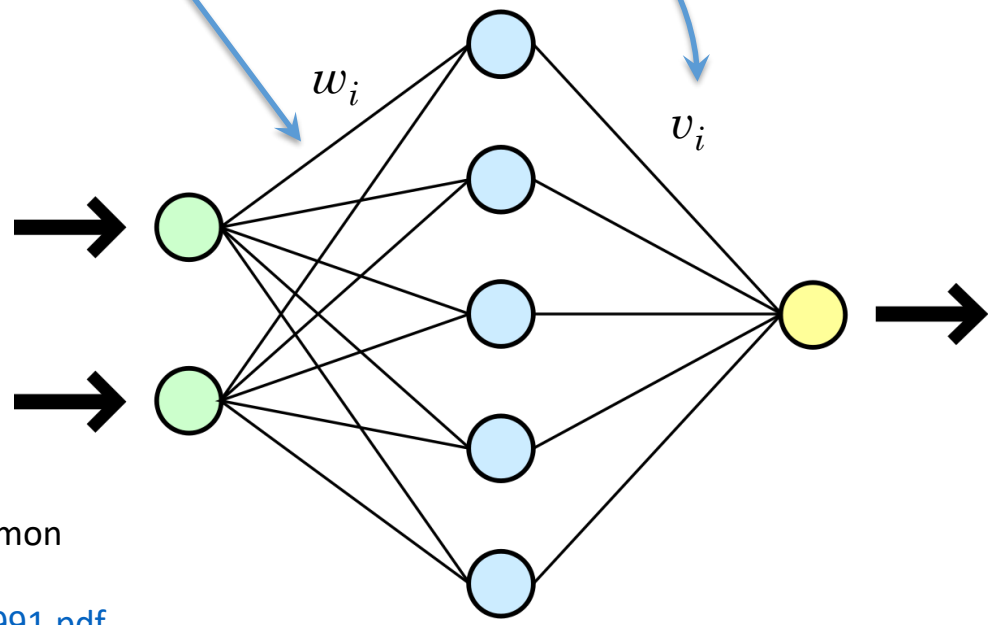
$$|F(x) - f(x)| < \varepsilon \quad \text{for any} \quad x \in I_m$$

$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

The Perceptron

Universal approximator formula

$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$



K. Hornik (1991), "Approximation Capabilities of Muitilayer Feedforward Networks", Neural Networks, Vol. 4, pp. 251-257, Pergamon Press, 1991.
https://web.njit.edu/~Eusman/courses/cs677_spring21/hornik-nn-1991.pdf

The Perceptron

$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

Challenge : determine parameters $v_i, b_i \in \mathbb{R} \quad w_i \in \mathbb{R}^m$

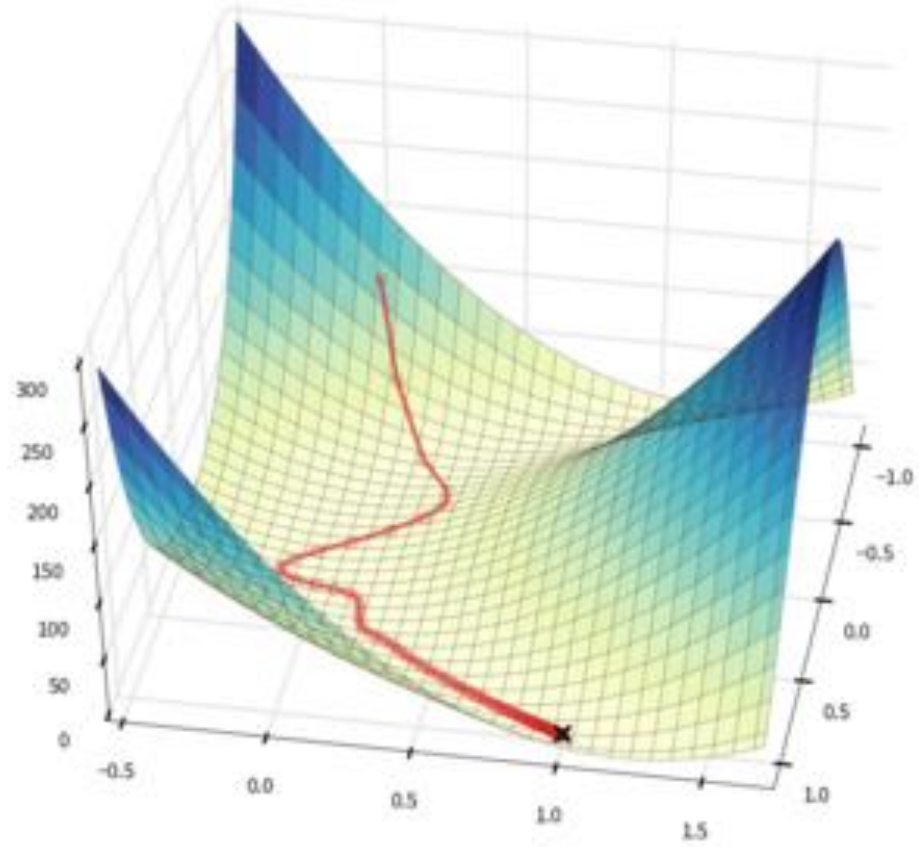
Solution : Machine Learning approach by expressing the problem as a *gradient descent minimization*.

Caveat : N is known to exist but cannot be computed.

Apprentissage et calcul de gradient

- Couples (entrée $\boldsymbol{x}$, sortie attendue $\boldsymbol{F(x)}$)
- Présenter l'entrée et observer la sortie
- La sortie est une combinaison de fonctions linéaires et la fonction d'activation, paramétrées par les poids des « synapes »
- Il est possible de reformuler le paramétrage du réseau comme une minimisation de l'erreur par descente du gradient sur les poids $\boldsymbol{b_i}$, $\boldsymbol{v_i}$, $\boldsymbol{w_i}$.

Gradient Descent



Source : https://mrmint.fr

Gradient computation

$$E = \frac{1}{2} \sum_{i=1}^{p} \|\mathbf{o}_i - \mathbf{t}_i\|^2.$$

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_\ell}\right).$$

Each weight is updated using the increment

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \quad \text{for } i = 1, \ldots, \ell,$$

Gradient computation

Observed Error (Loss Function)

Expected Output

Observed Output

$$E = \frac{1}{2} \sum_{i=1}^{p} |\mathbf{o}_i - \mathbf{t}_i|^2.$$

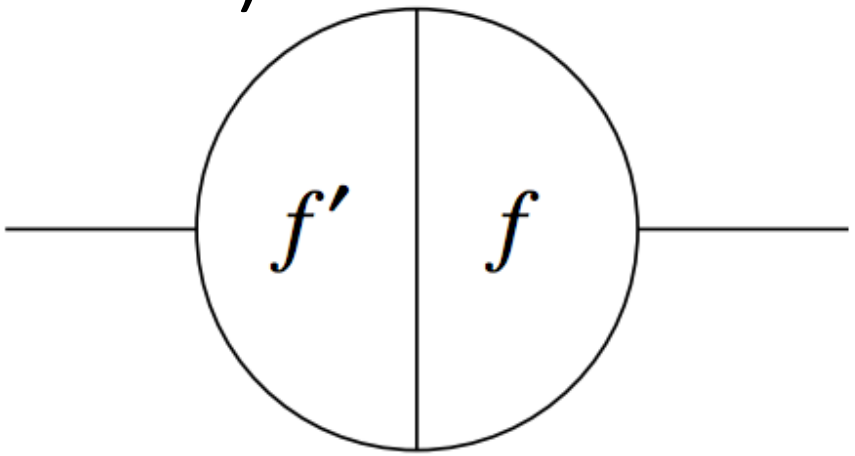
$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_\ell} \right).$$

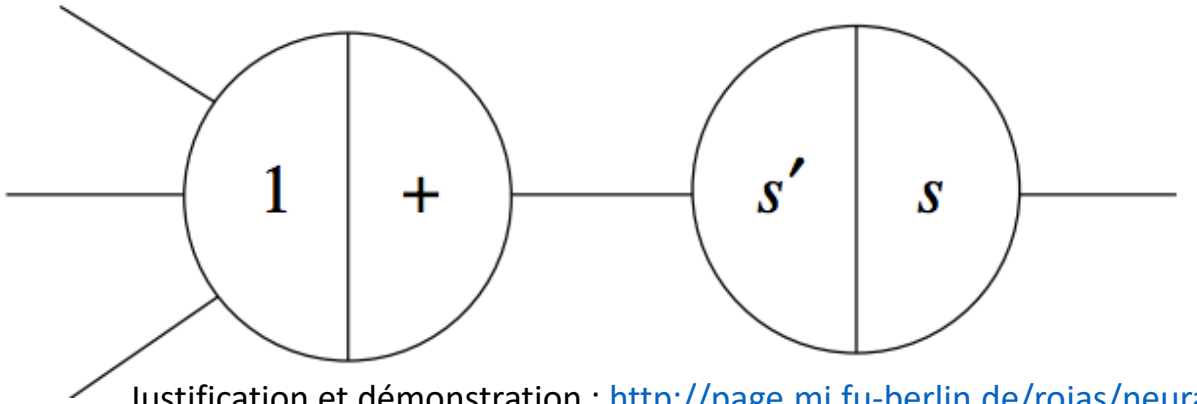
Network Weights

Each weight is updated using the increment

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \quad \text{for } i = 1, \ldots, \ell,$$

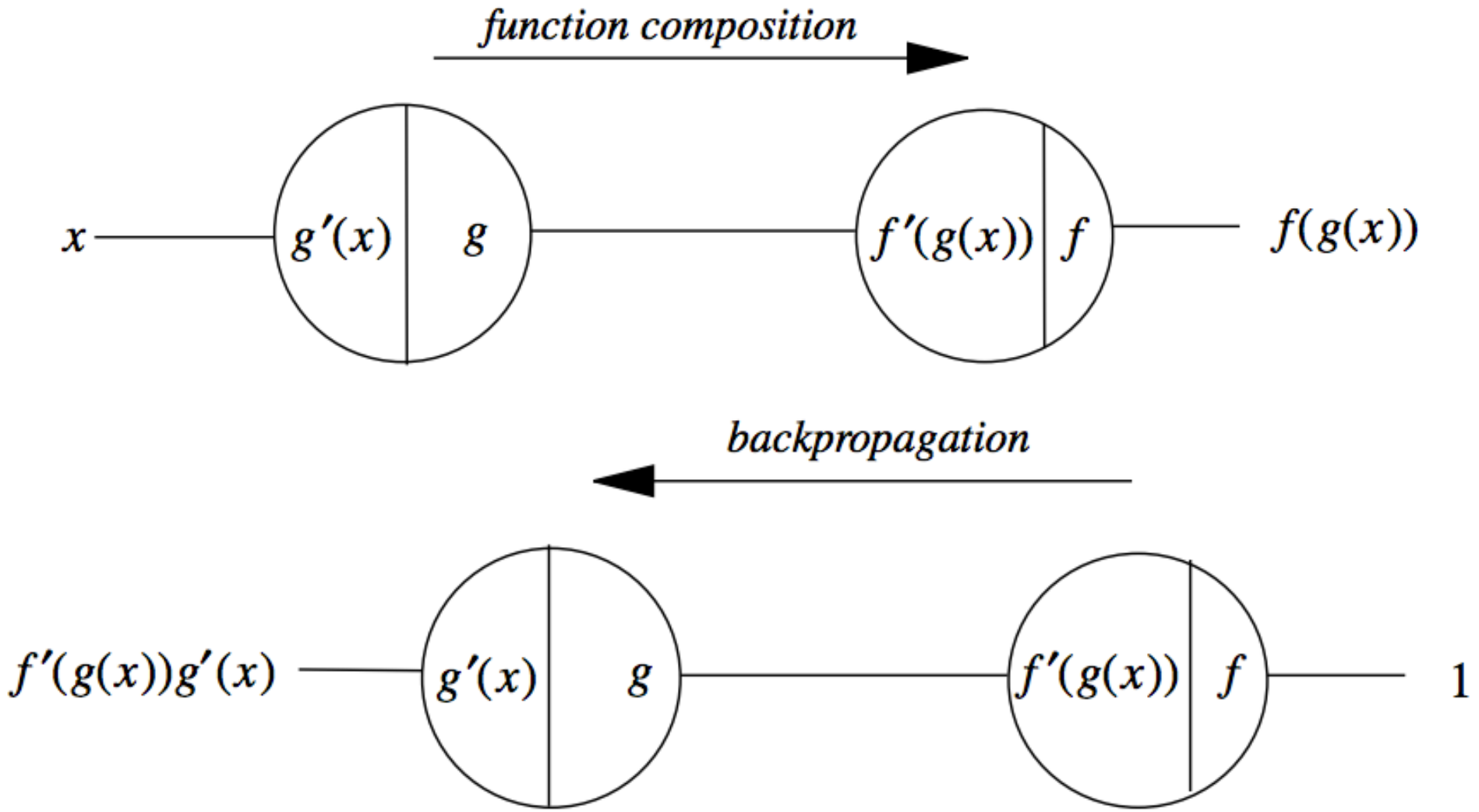
Gradient Backpropagation Algorithm (single variable case)





Justification et démonstration : http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf

Backpropagation Algorithm



Result

It is straightforward to prove that

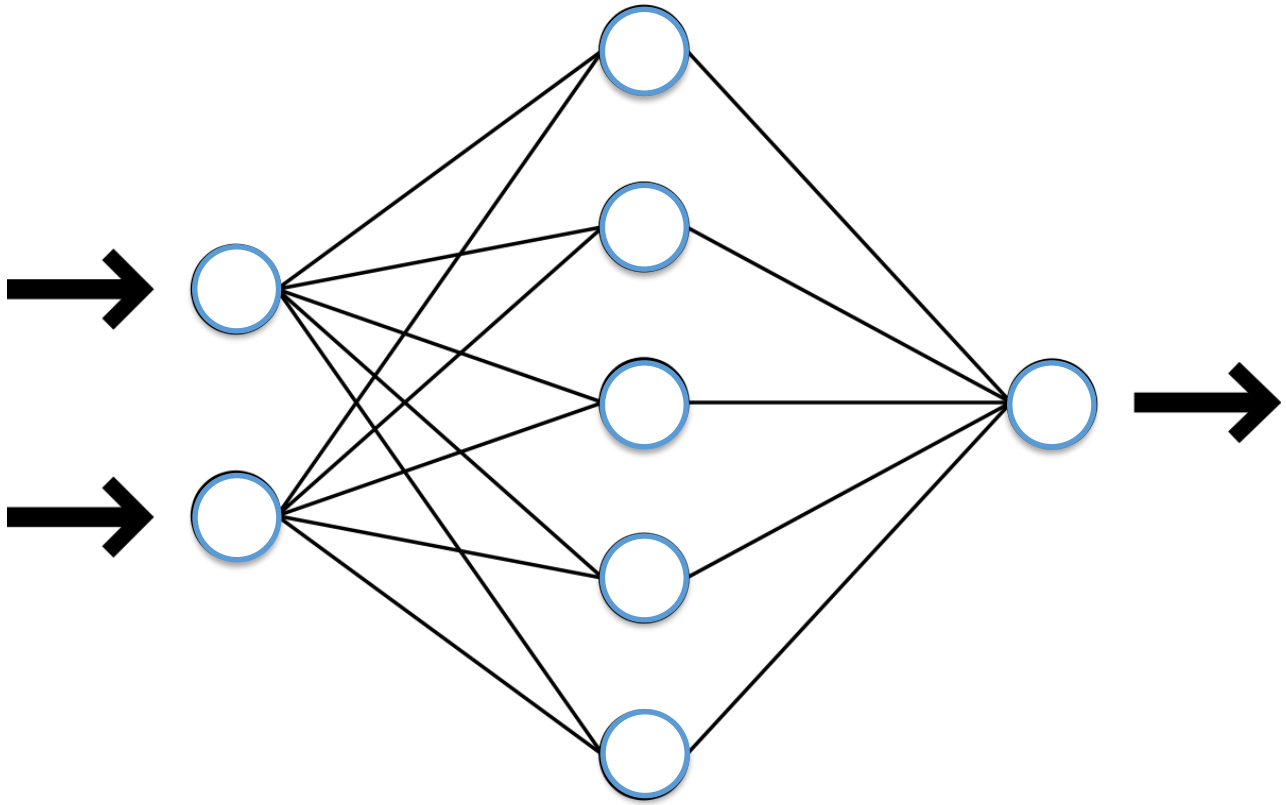
$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j . \qquad \Delta w_{ij} = -\gamma o_i \delta_j .$$

where δ_j is the loss function value backpropagated to node j and where o_i is the computed value at node i (feed-forward).

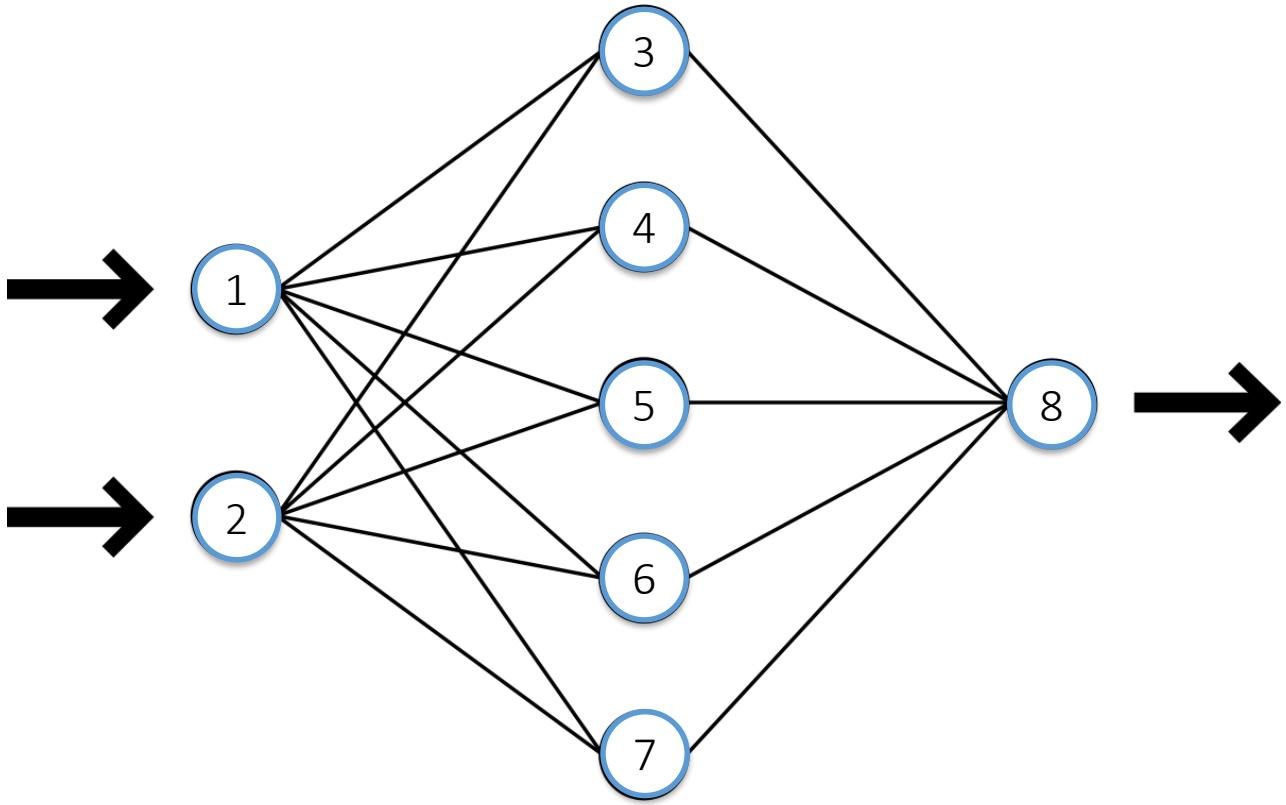
Algorithme d'apprentissage supervisée

- Recueillir "suffisamment" de données de type ($\boldsymbol{x}$, $\boldsymbol{F(x)}$)
- Initialiser les poids $\boldsymbol{w_{ij}}$ du réseau de neurones à des valeurs aléatoires
- De façon itérative :
  - Exécuter le réseau en *feed-forward* avec une valeur $\boldsymbol{x_k}$ et mémoriser tous les états $\boldsymbol{o_i}$ des neurones intermédiaires.
  - Calculer l'erreur $\boldsymbol{E}$ entre la sortie du réseau et la valeur $\boldsymbol{F(x)}$ attendue
  - Rétropropager l'erreur à travers le réseau et mémoriser les valeurs de $\boldsymbol{\delta_j}$
  - Ajuster les poids du réseau
  - Continuer jusqu'à convergence $\Delta w_{ij} = -\gamma o_i \delta_j .$

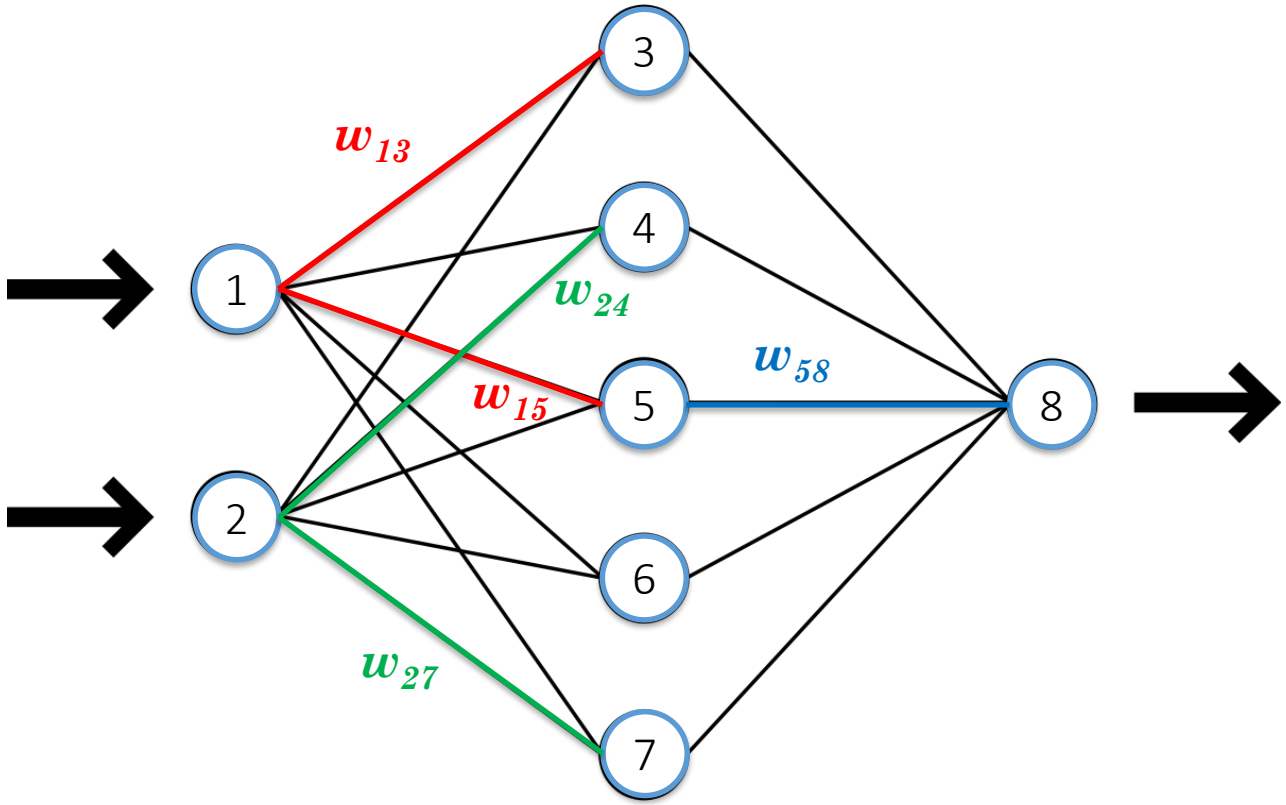
Feed-Forward Phase



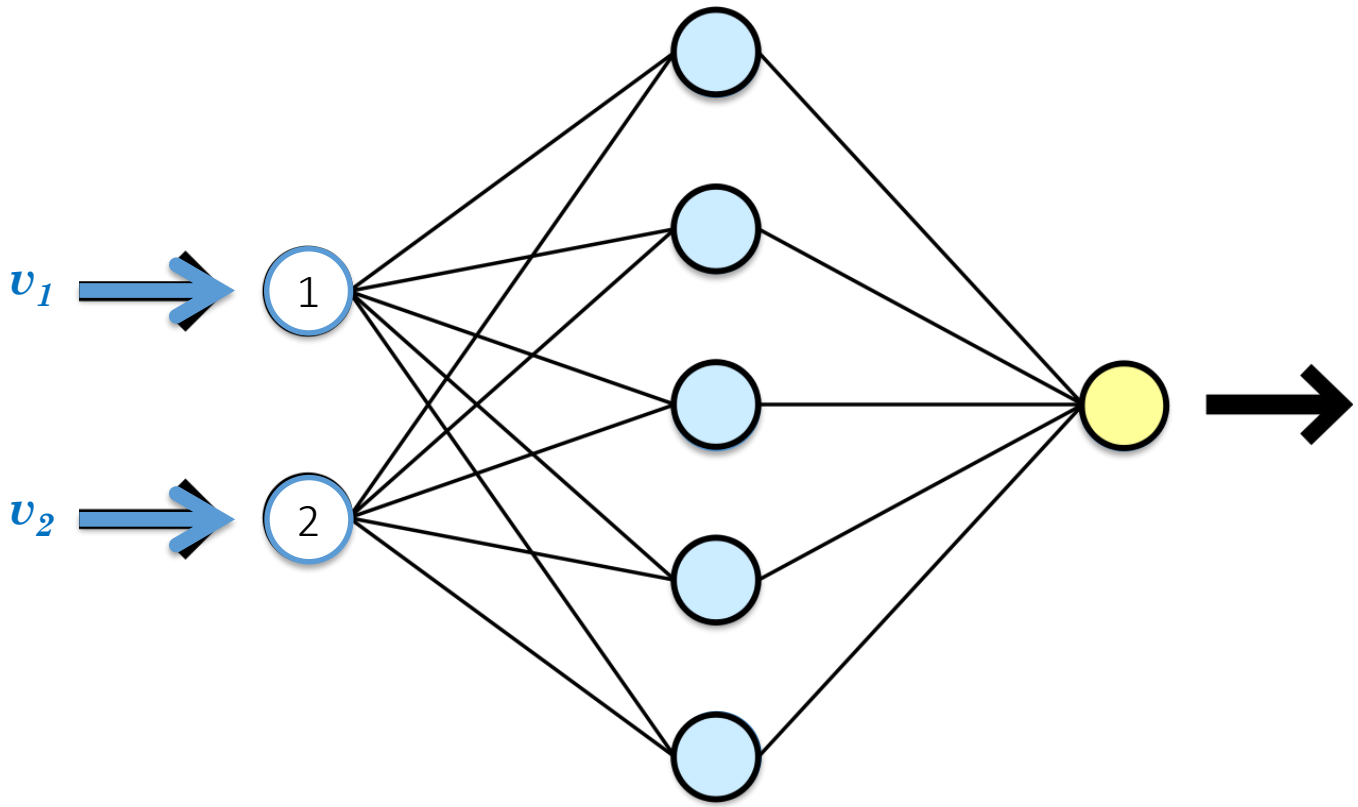
Feed-Forward Phase (notations)



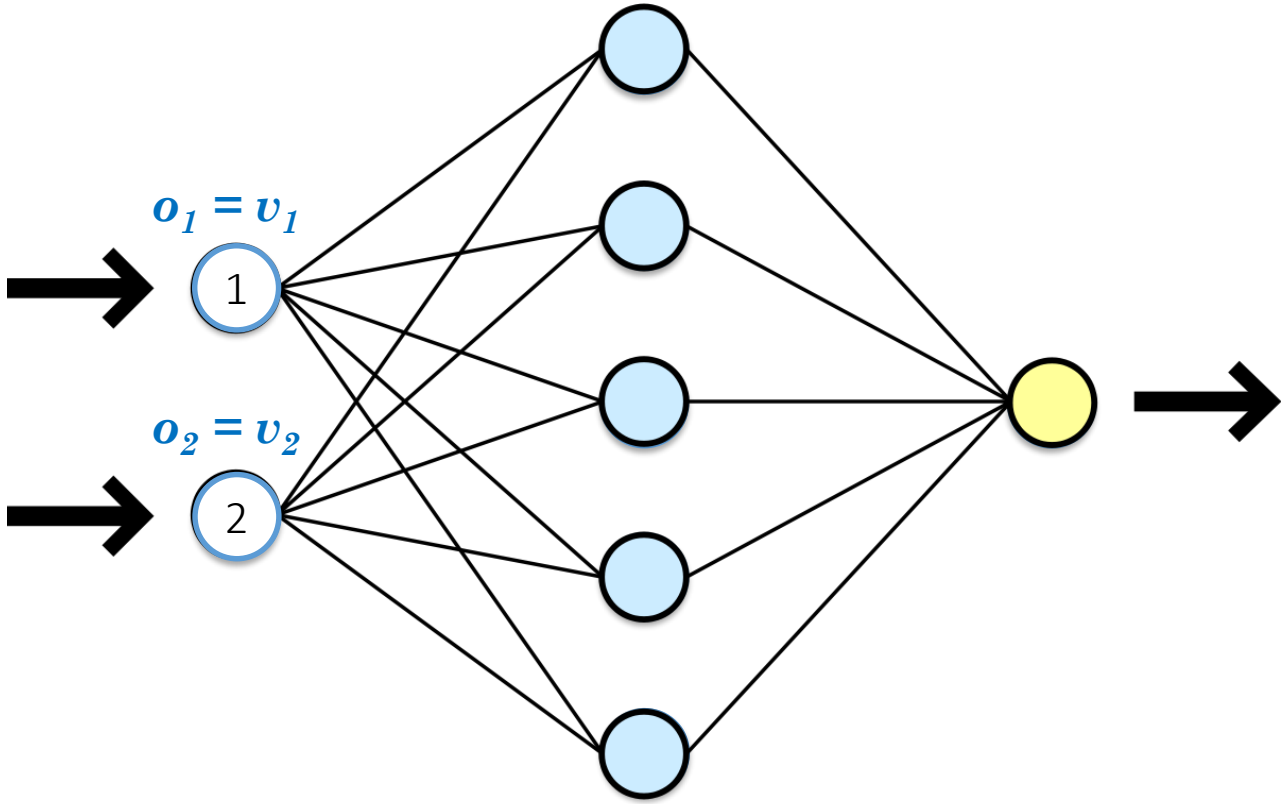
Feed-Forward Phase (notations)



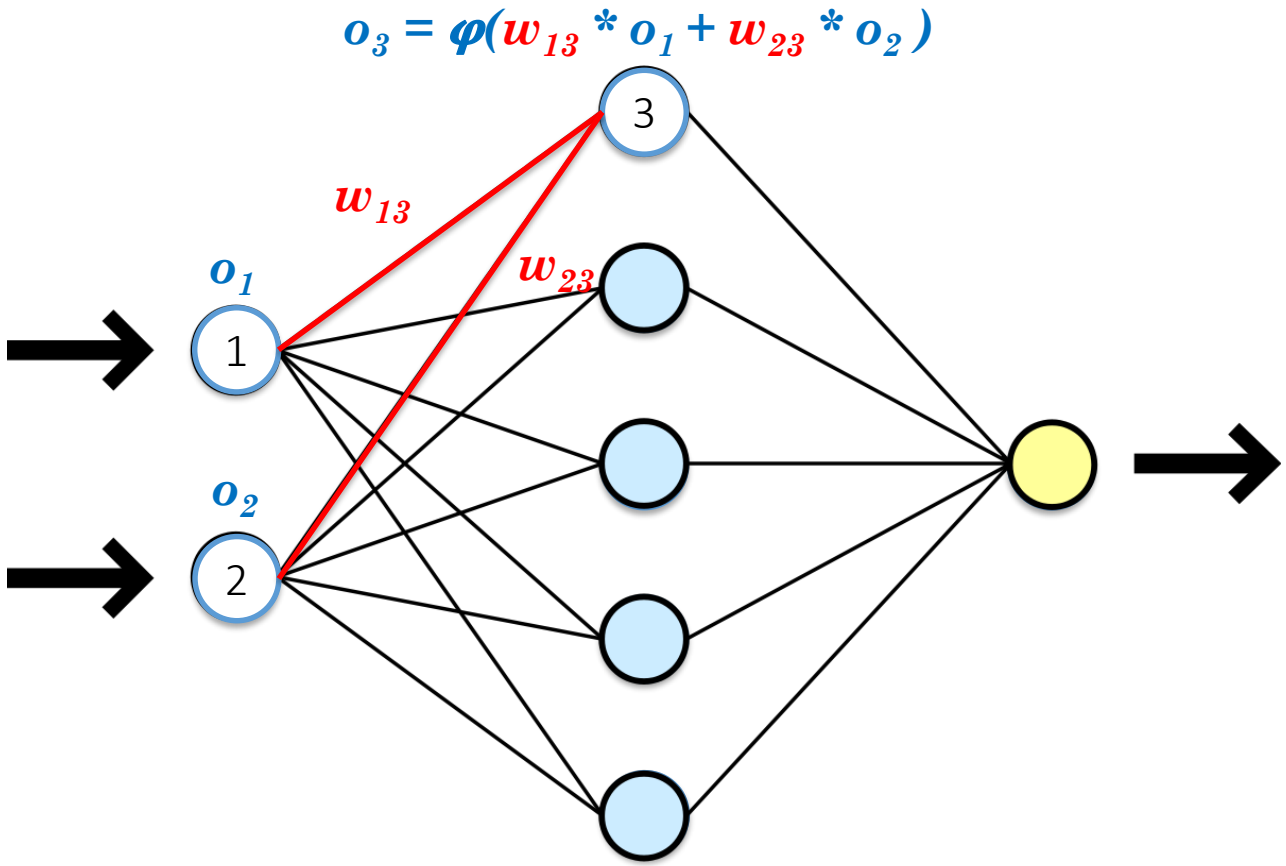
Feed-Forward Phase (Step 0)



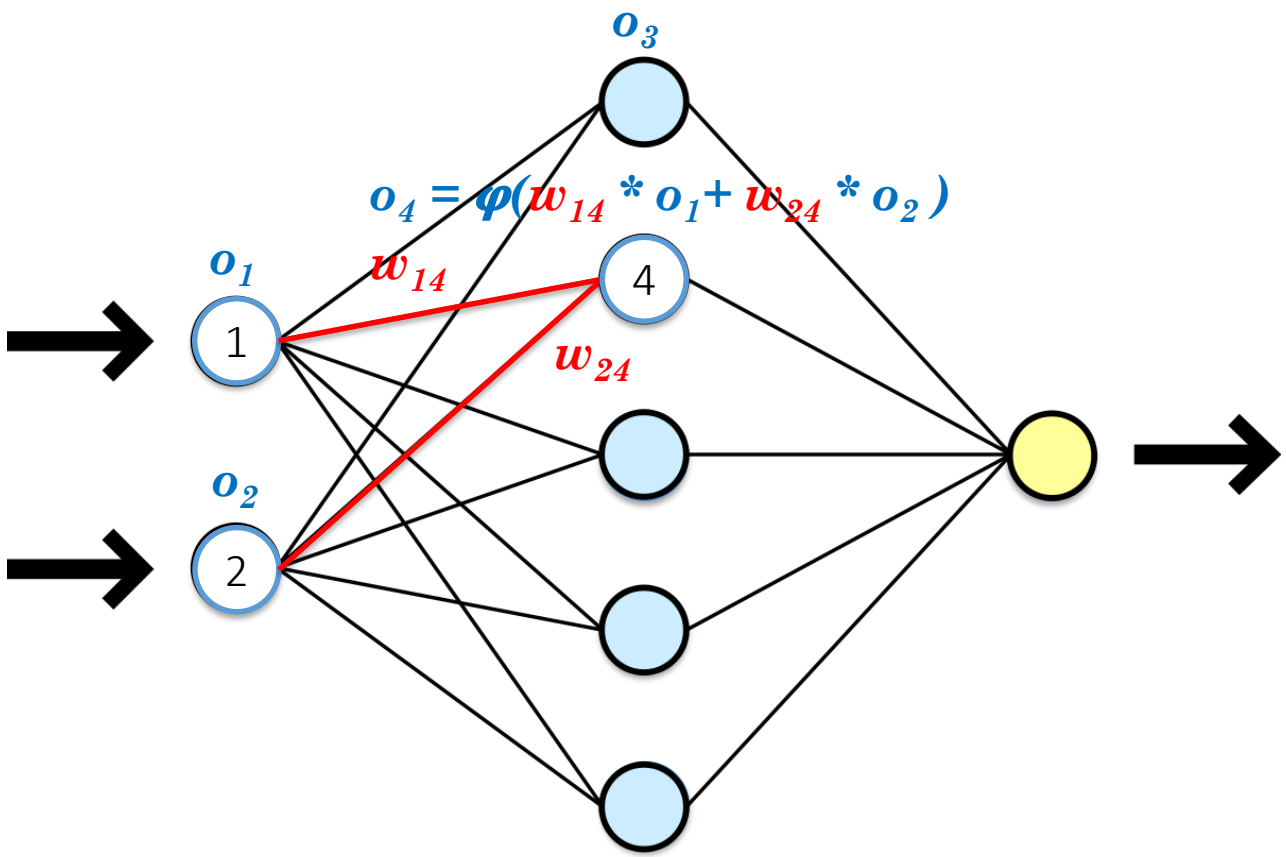
Feed-Forward Phase (Step 1)



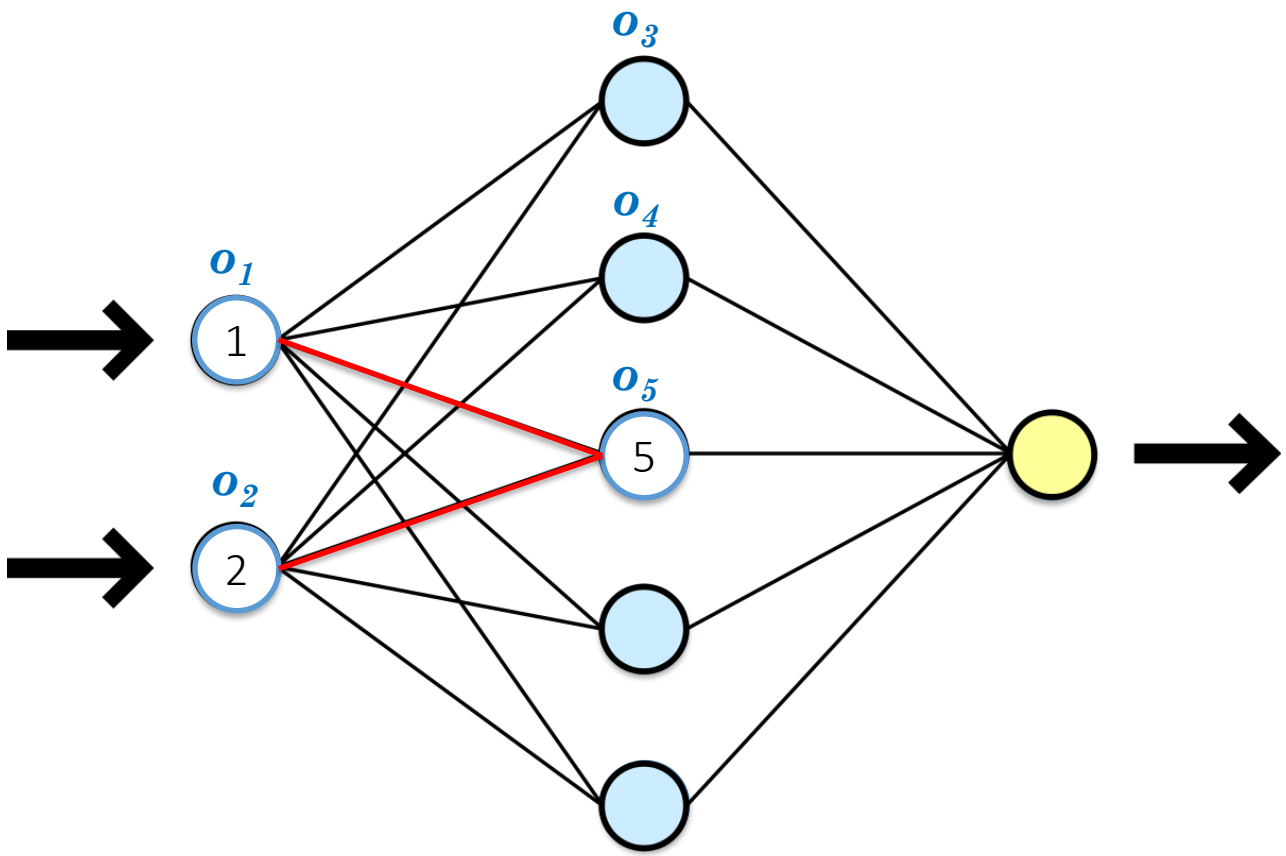
Feed-Forward Phase (Step 2)



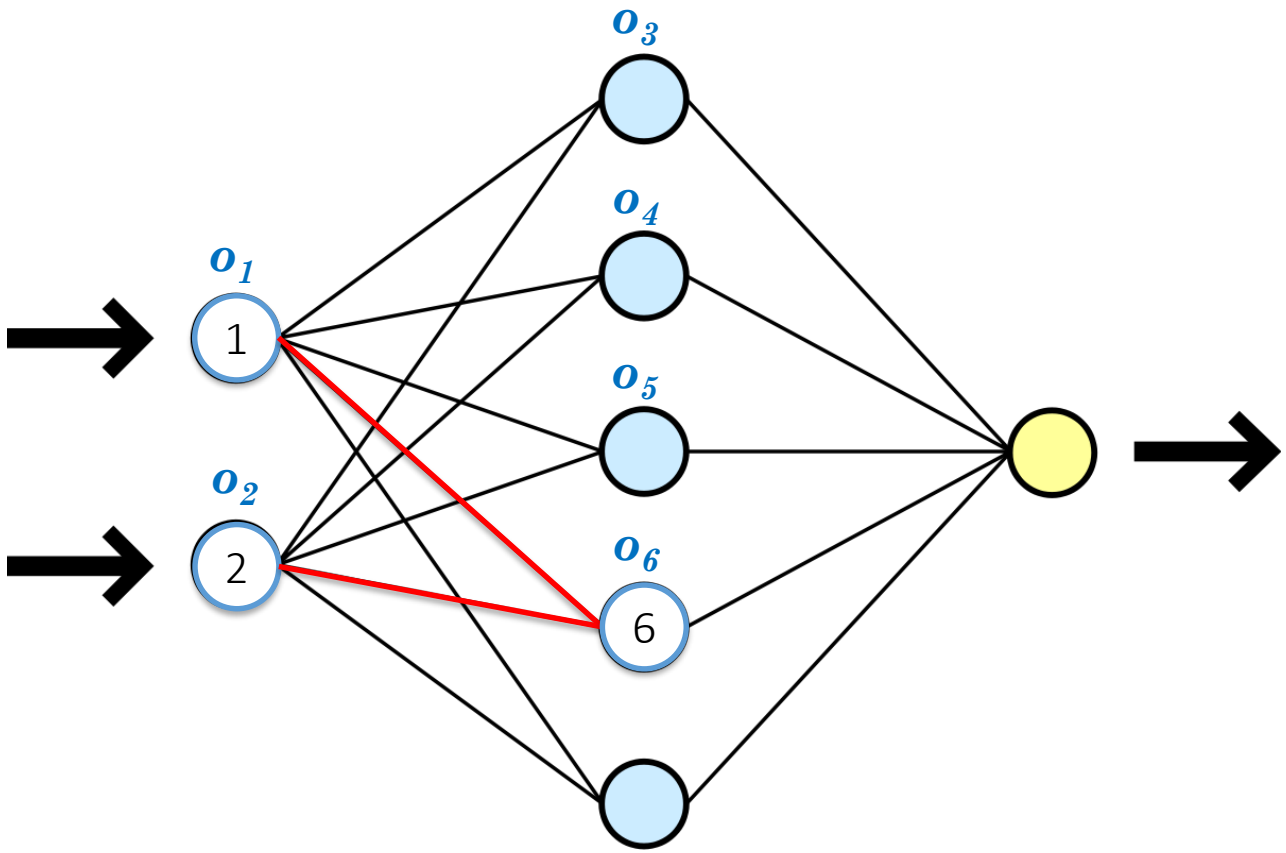
Feed-Forward Phase (Step 2)



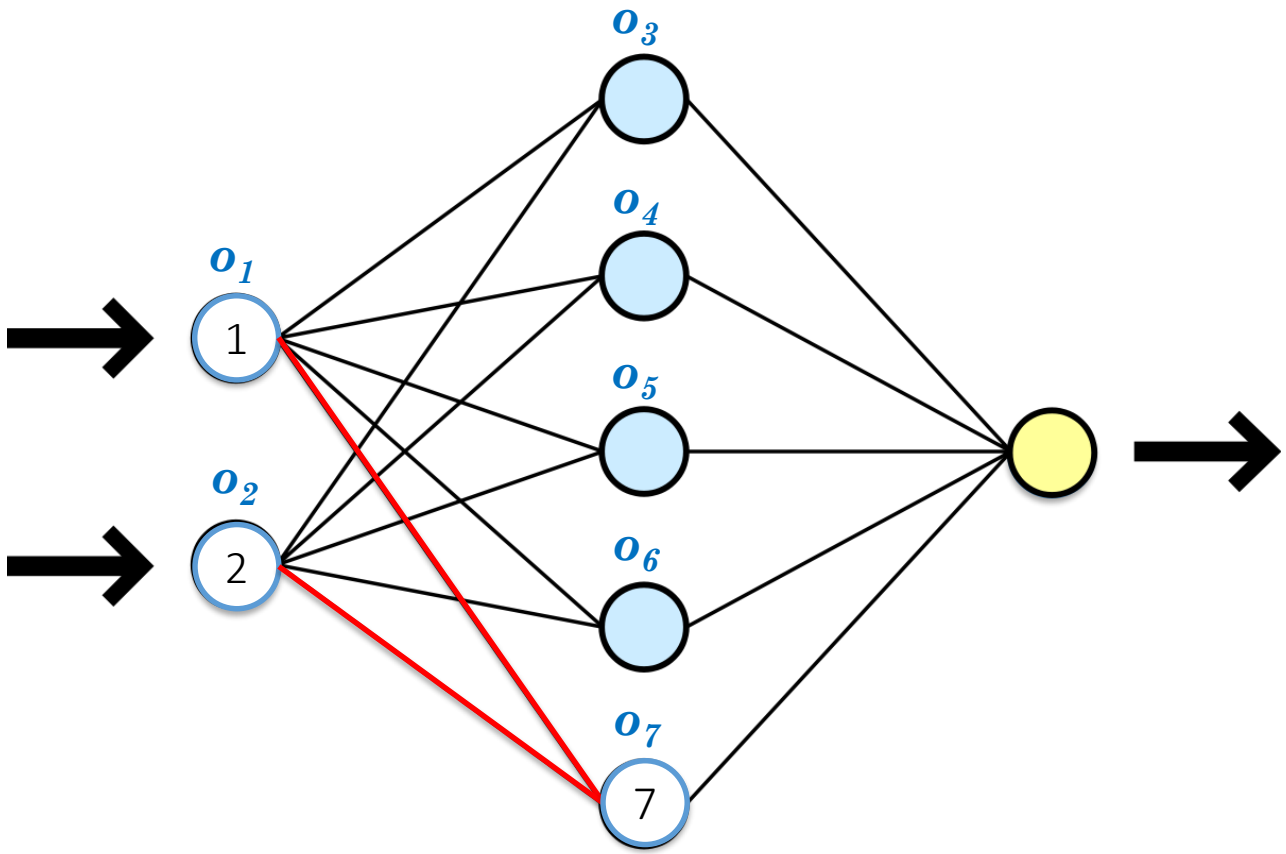
Feed-Forward Phase (Step 2)



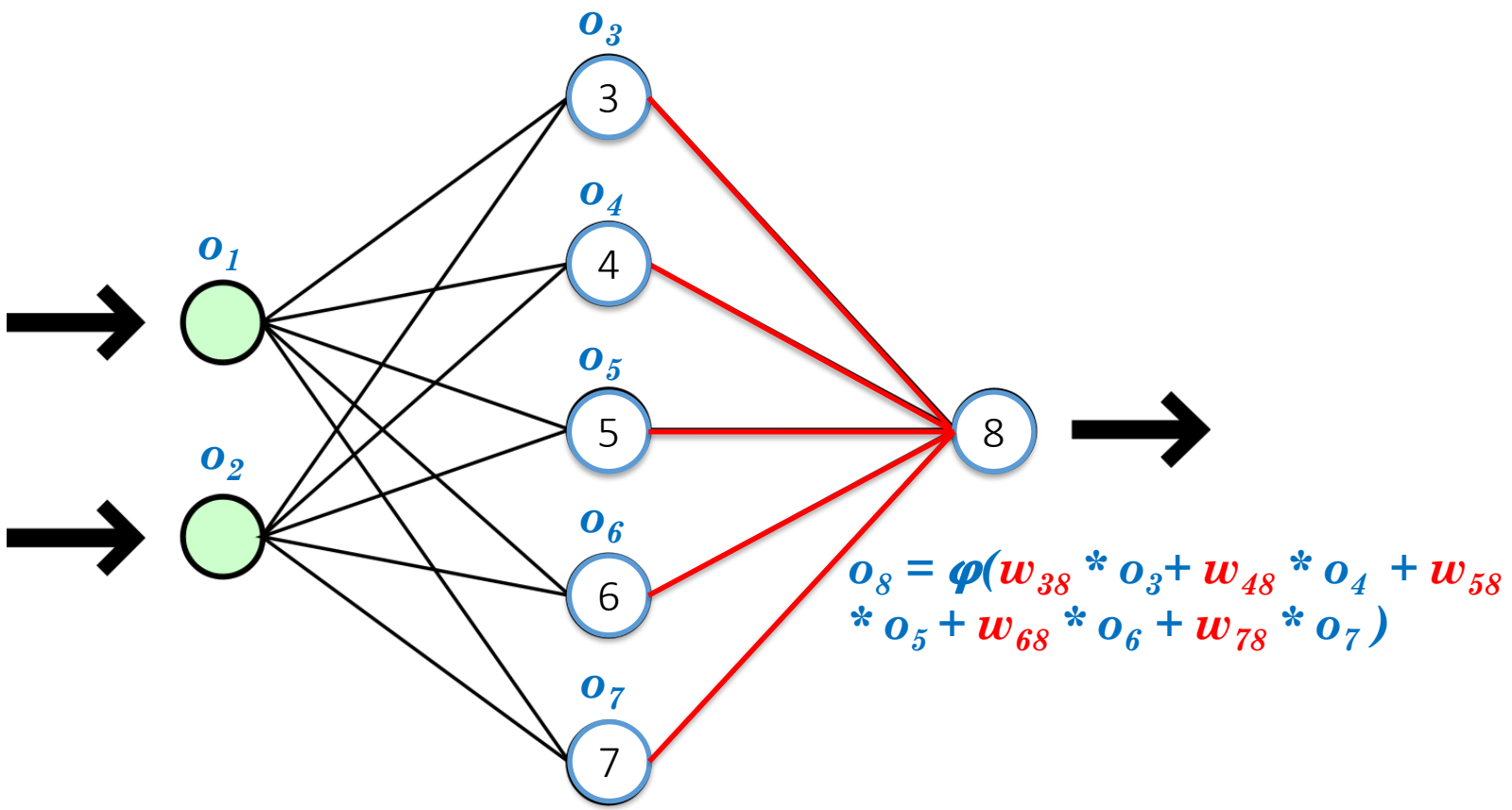
Feed-Forward Phase (Step 2)



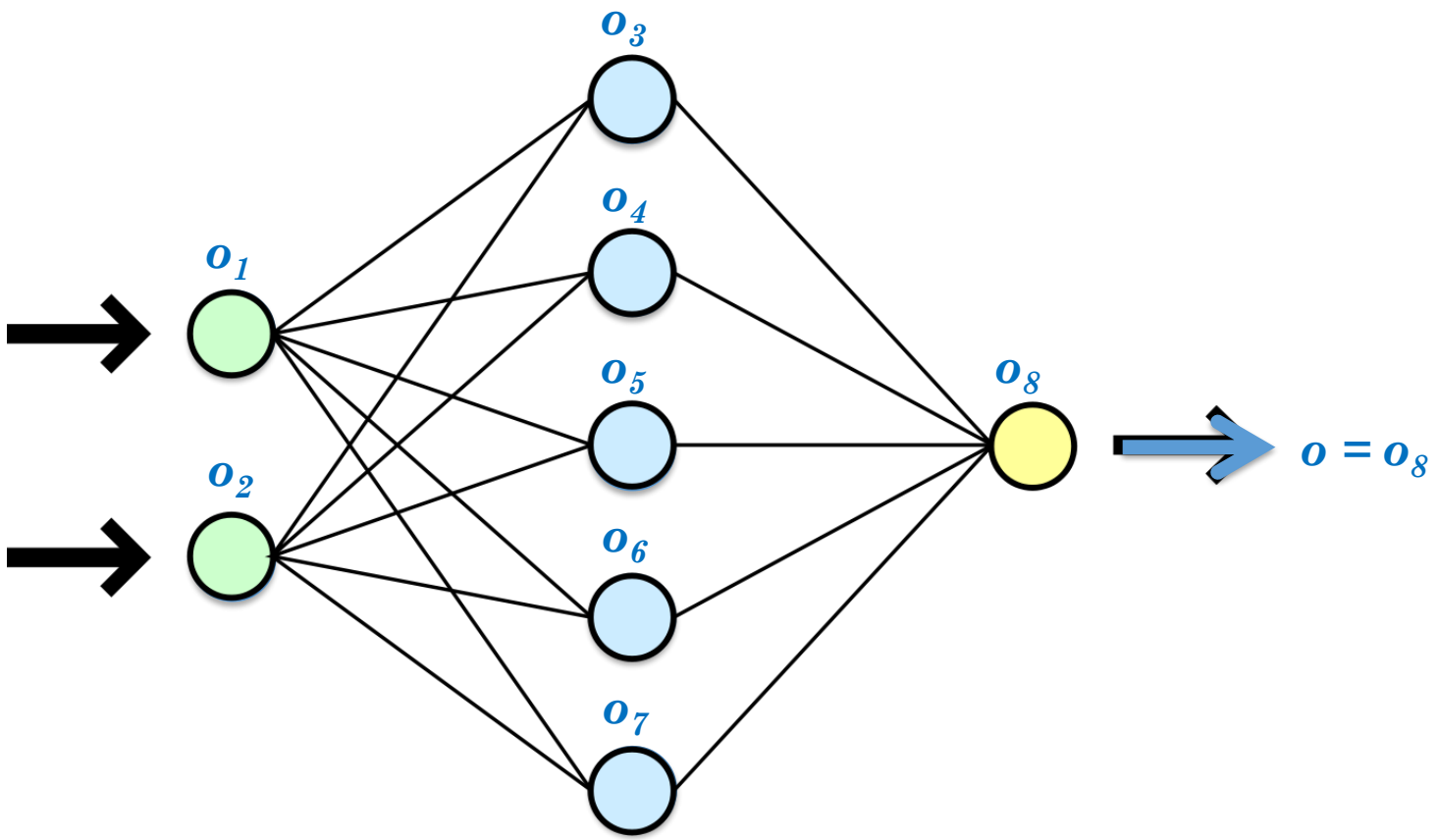
Feed-Forward Phase (Step 2)



Feed-Forward Phase (Step 3)



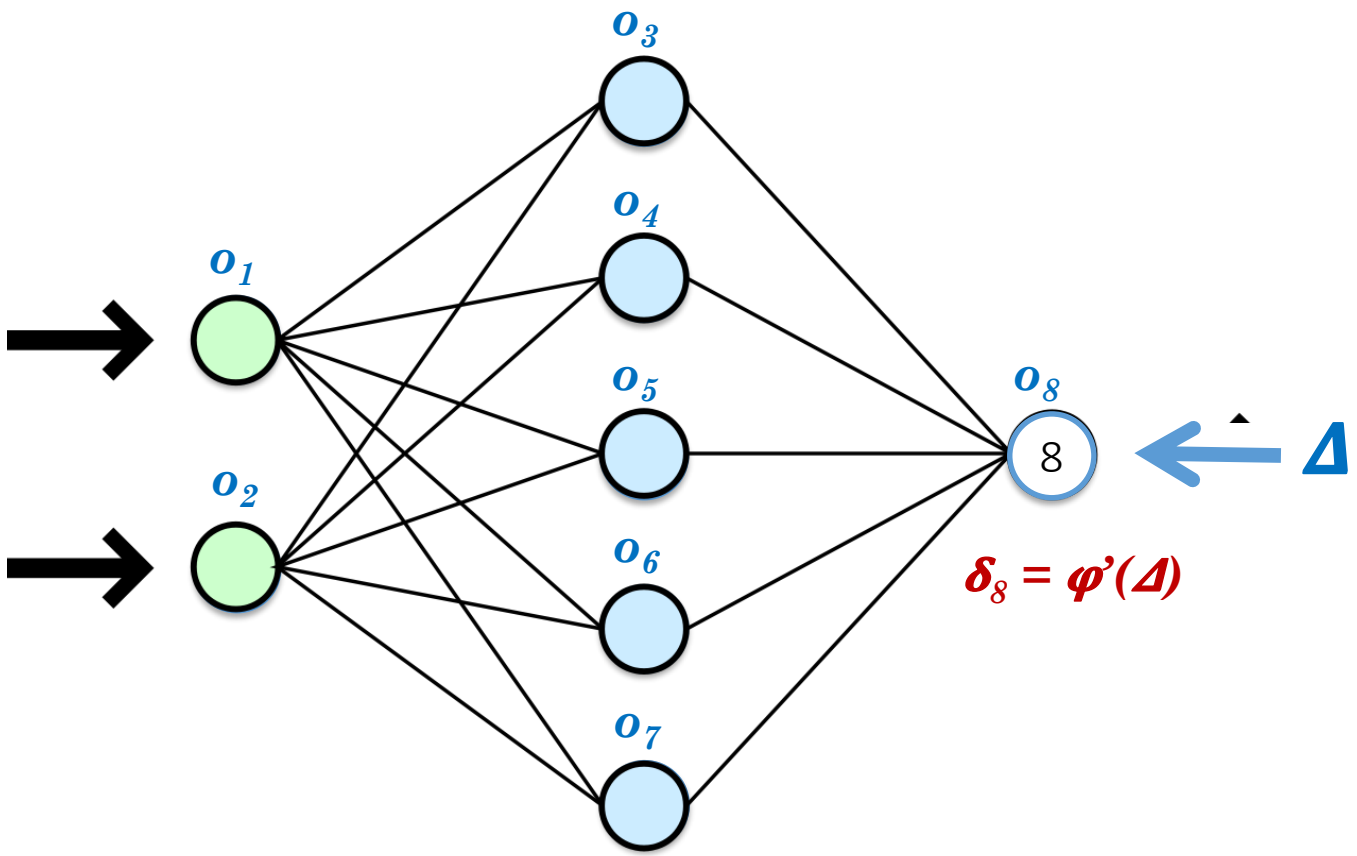
End of *Feed-Forward* Phase



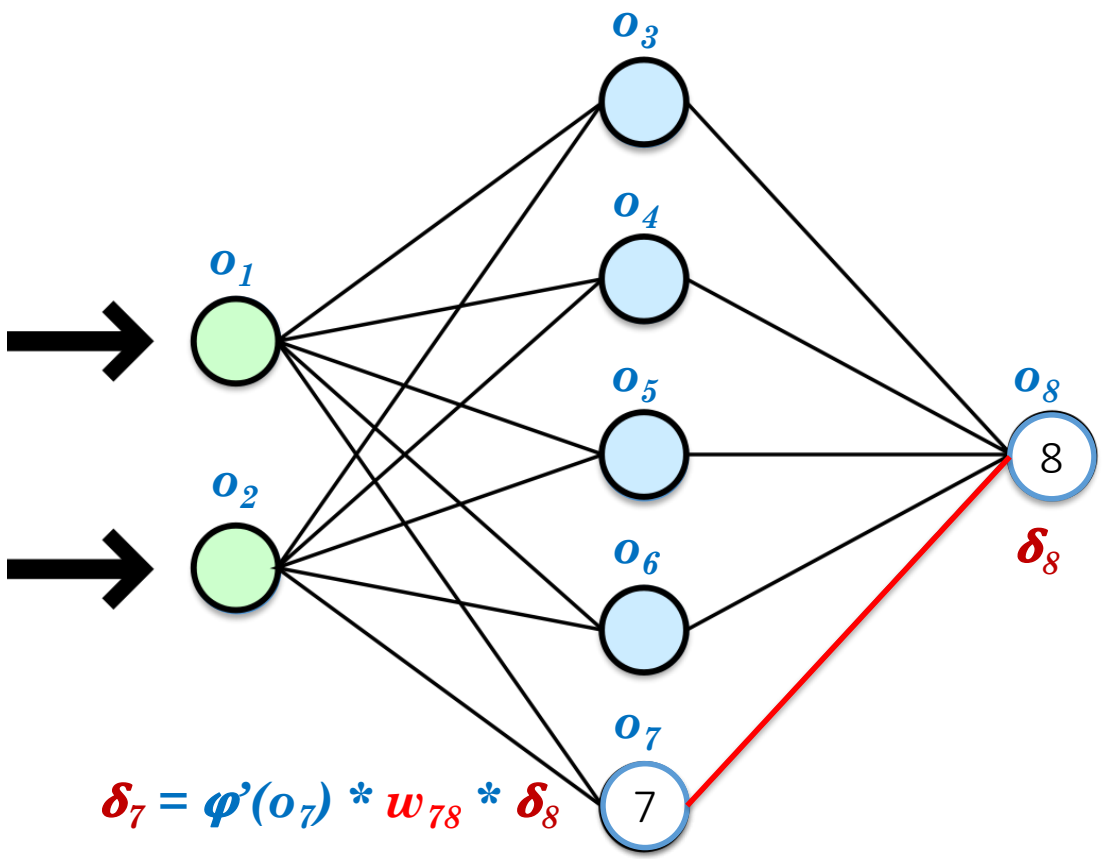
Analyse de l’erreur observée

- o est le résultat obtenu à partir de l’entrée $[v1, v2]$
- Le résultat attentu était t
- On calcule donc $\Delta = |o - t|$

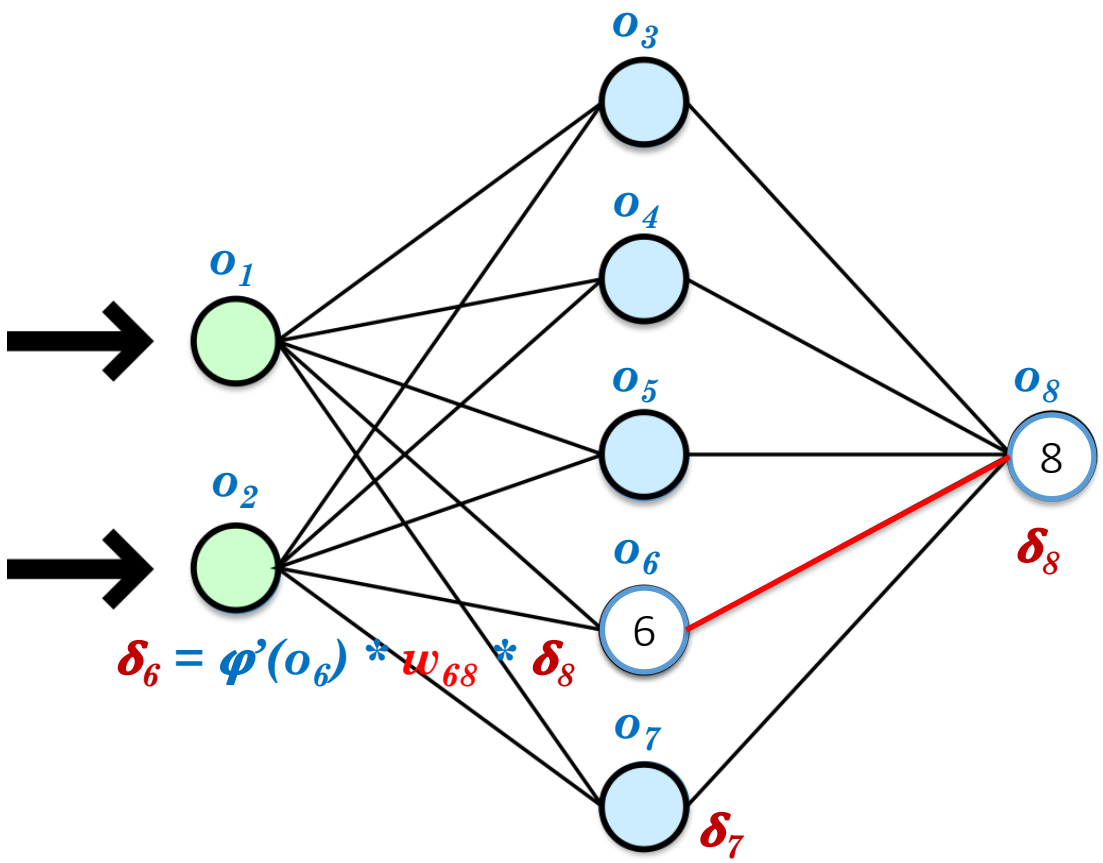
Phase de rétropropagation



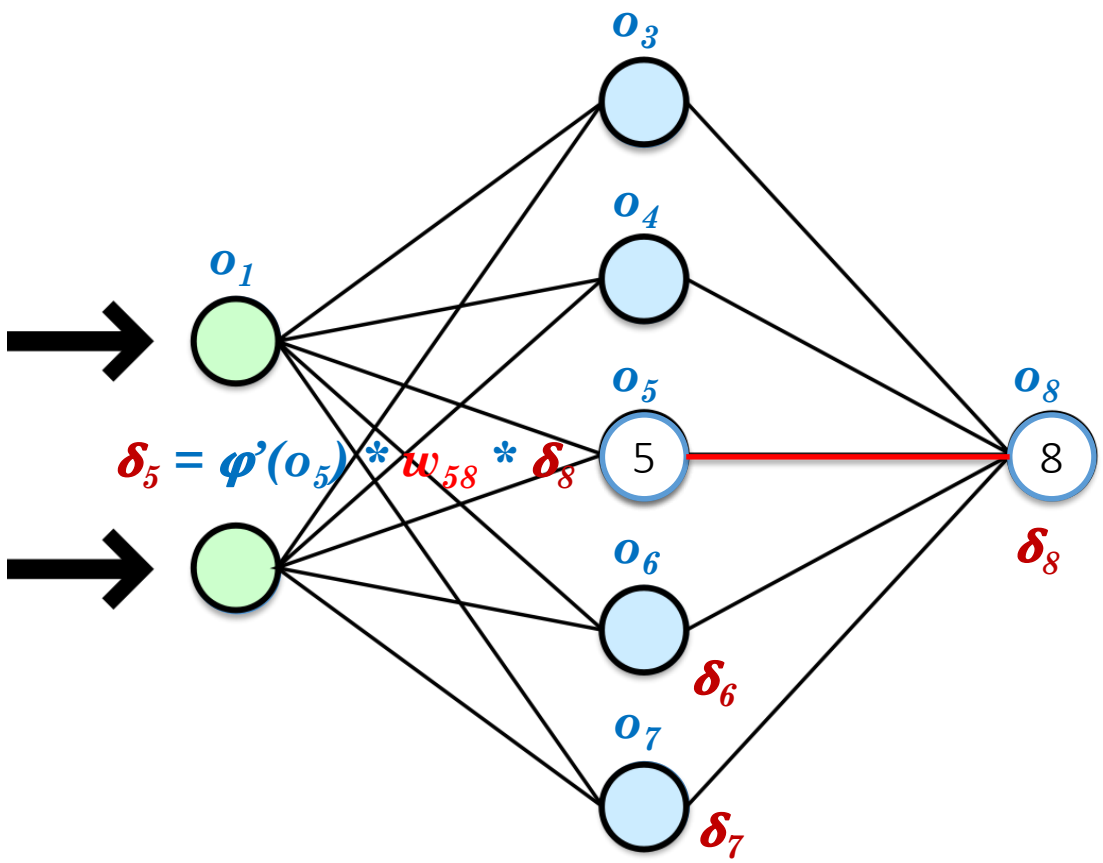
Phase de rétropropagation



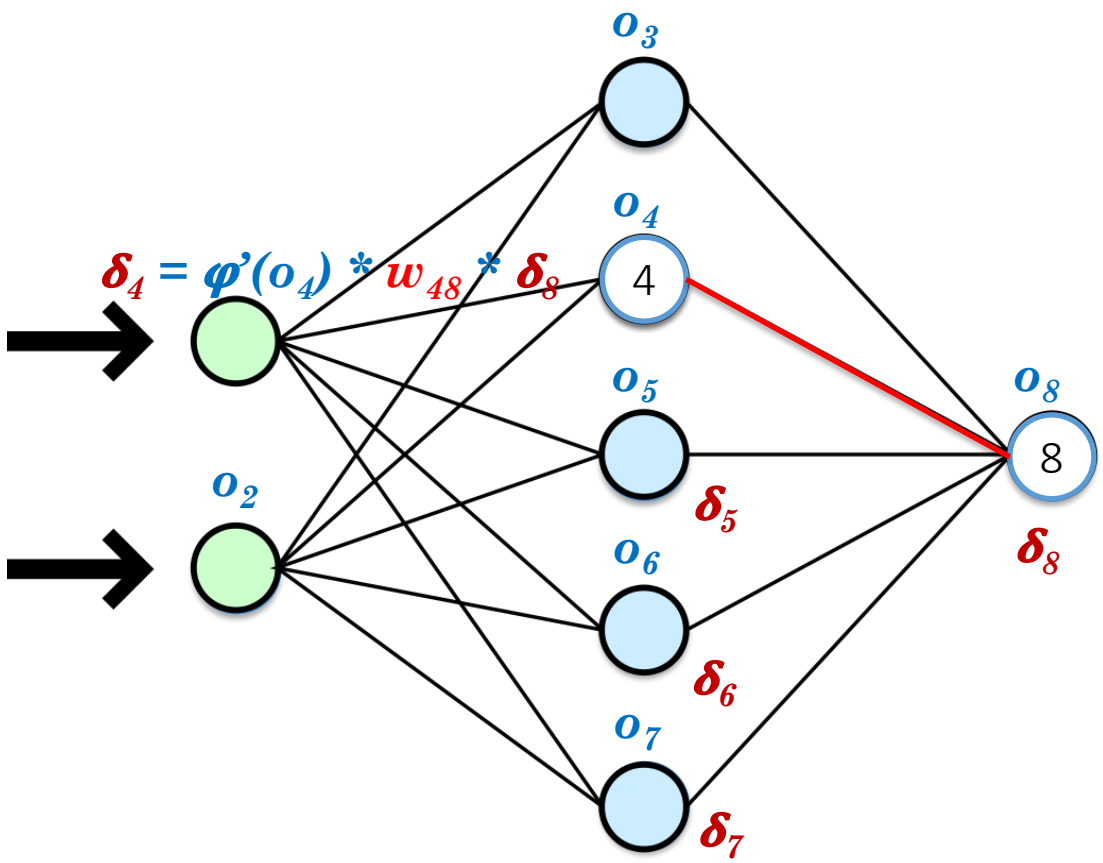
Phase de rétropropagation



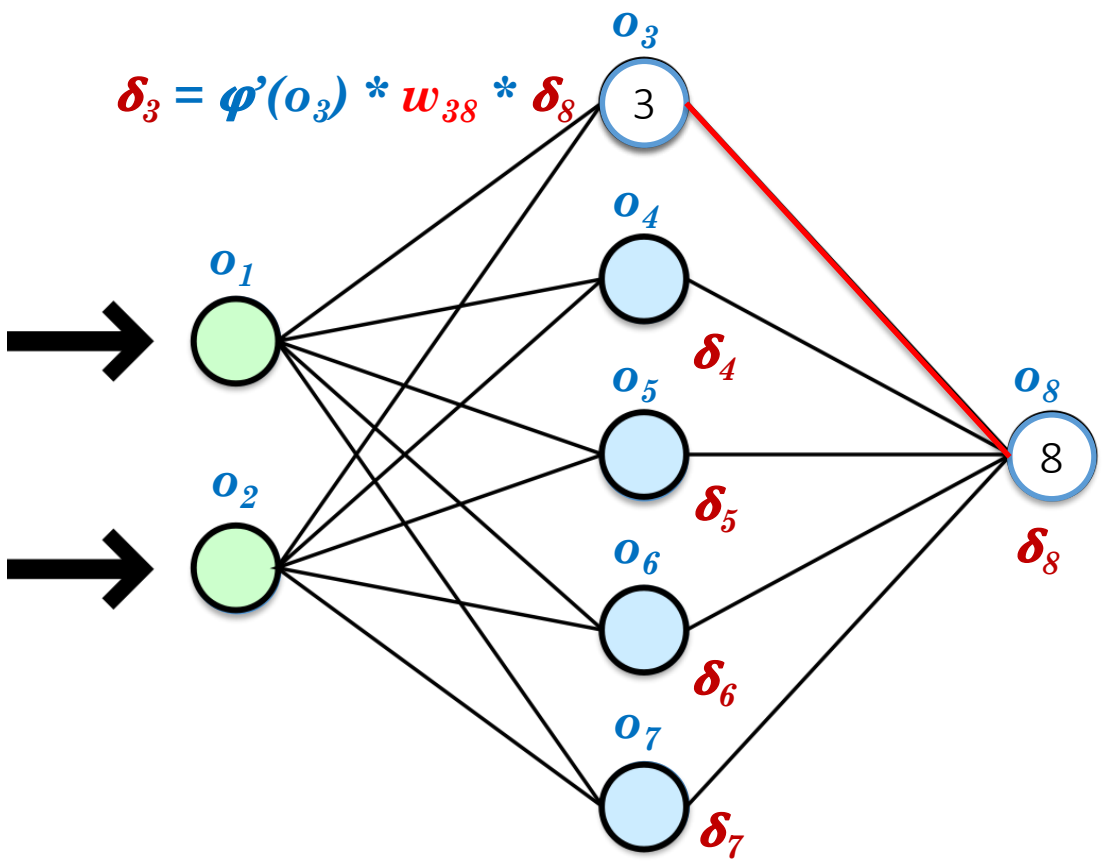
Phase de rétropropagation



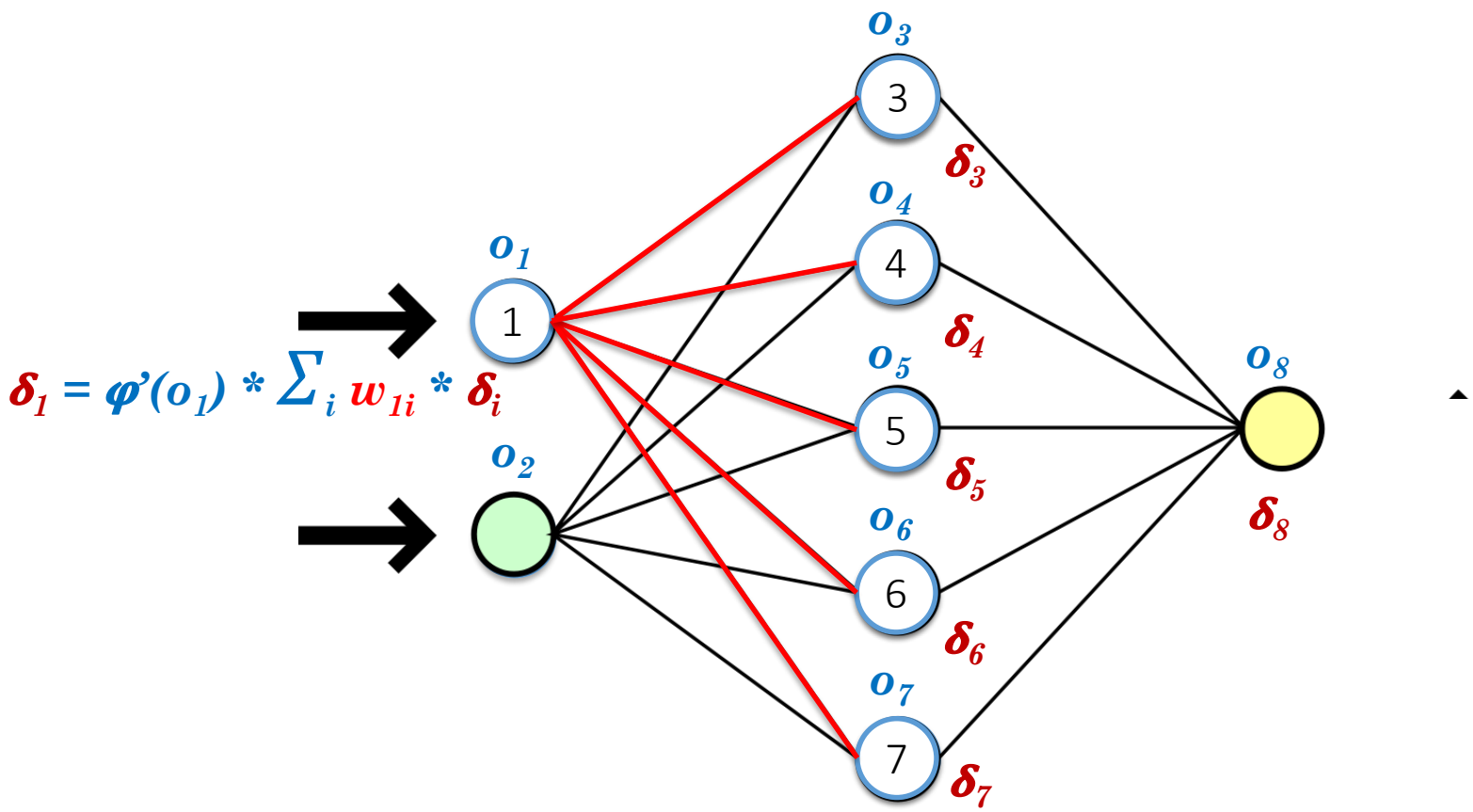
Phase de rétropropagation



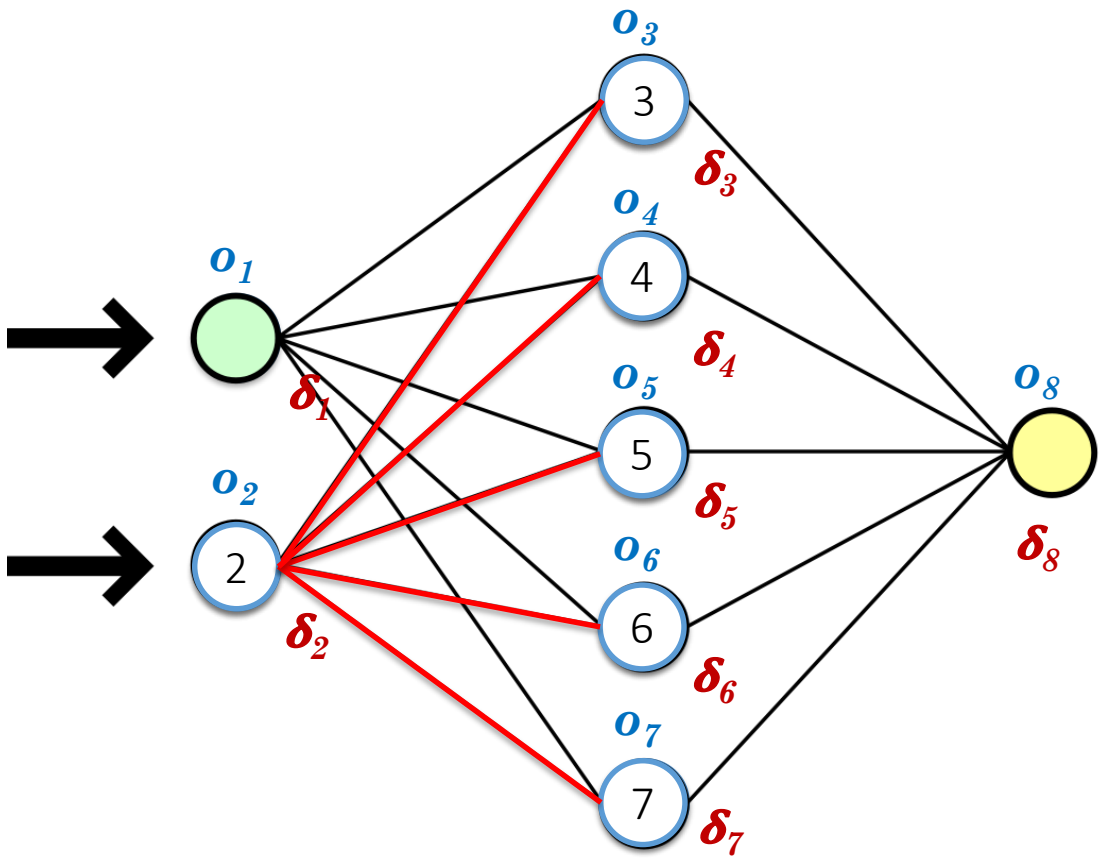
Phase de rétropropagation



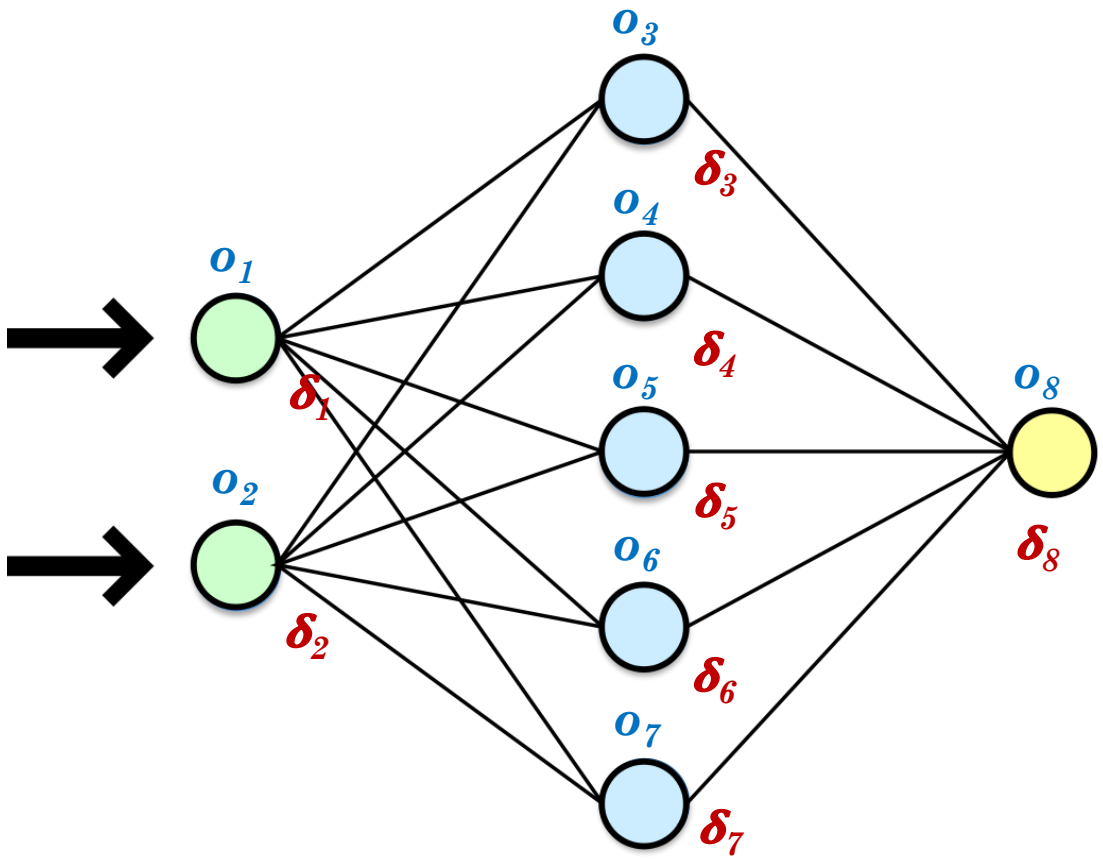
Phase de rétropropagation



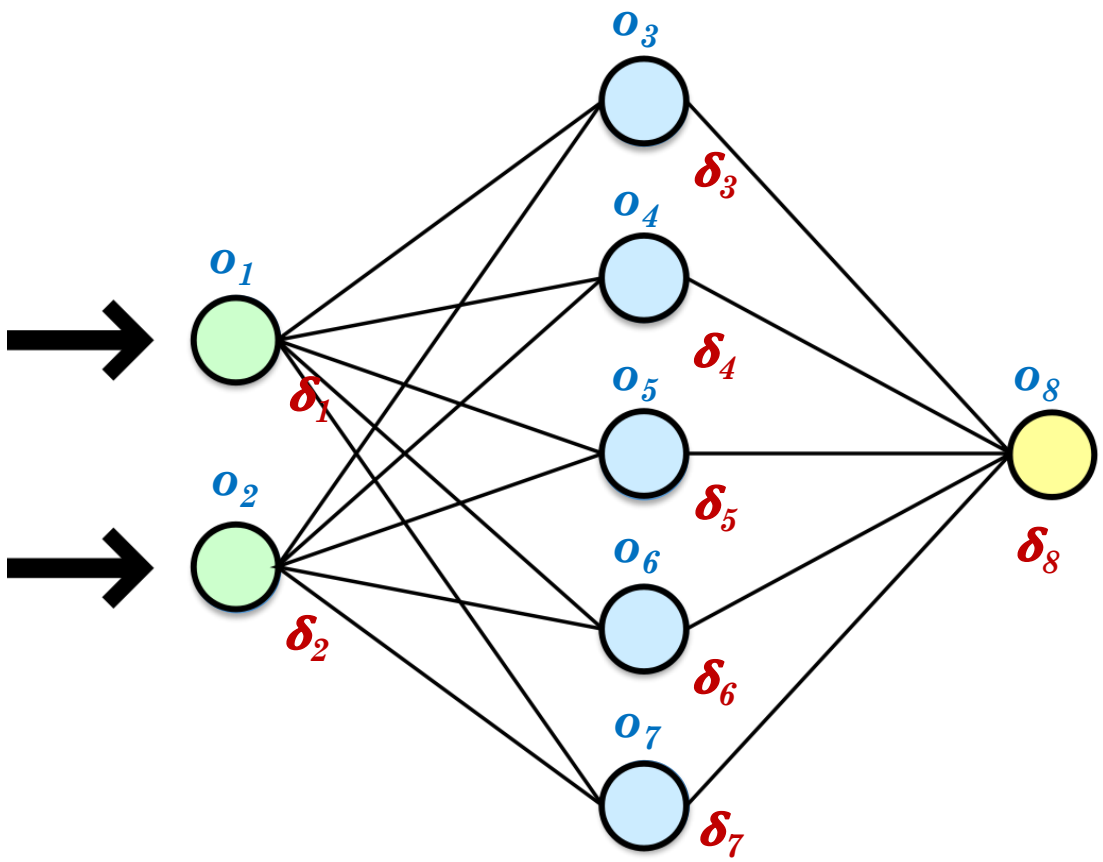
Phase de rétropropagation



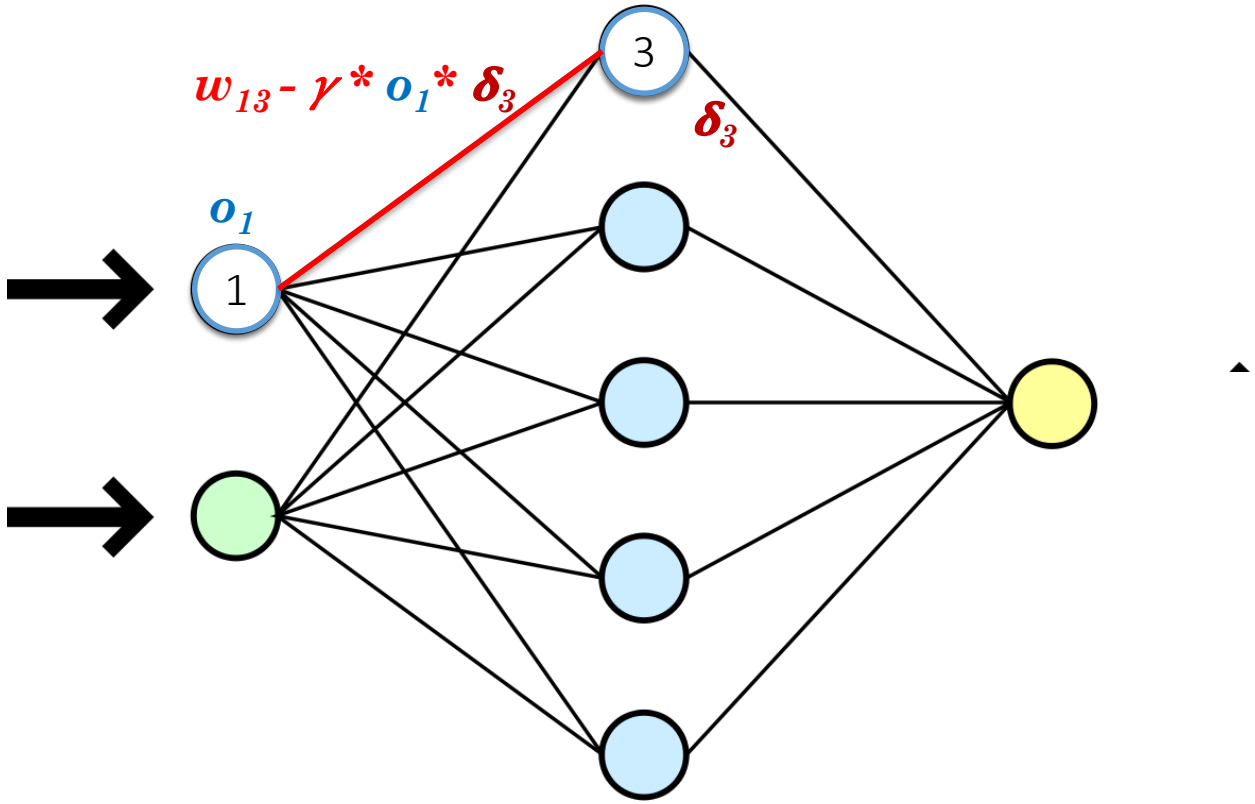
Résultat de la rétropropagation



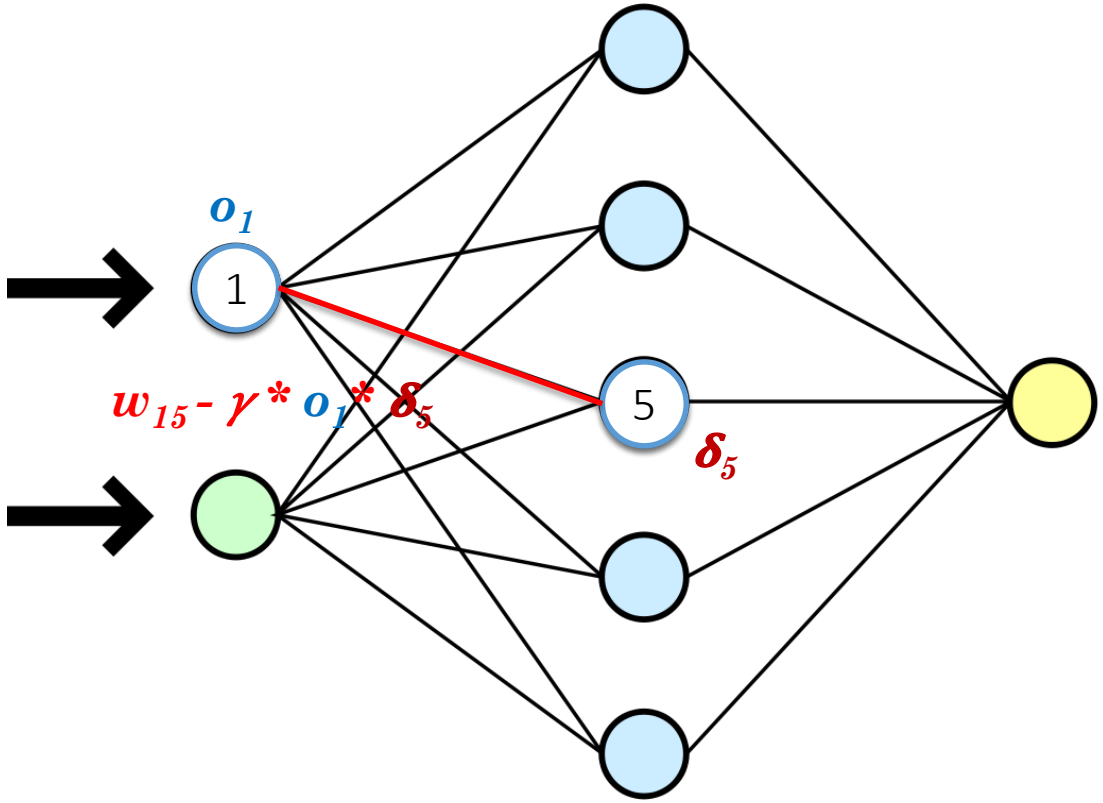
Mise à jour des poids



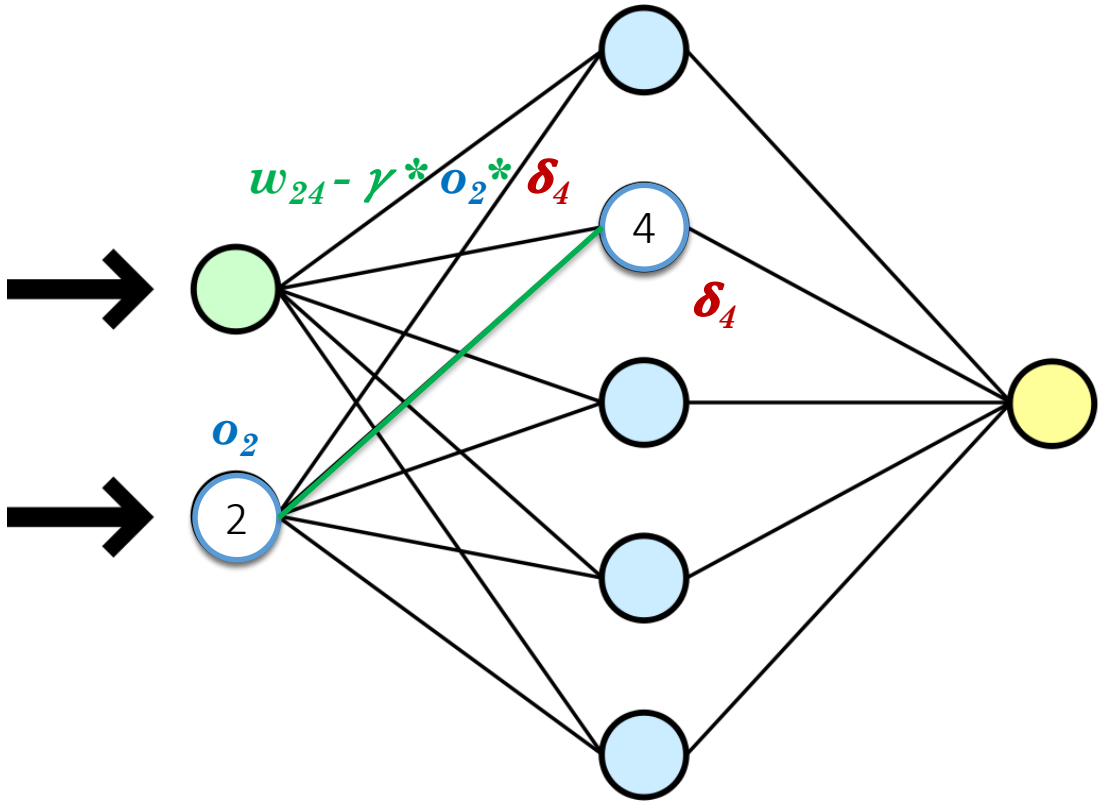
Mise à jour des poids



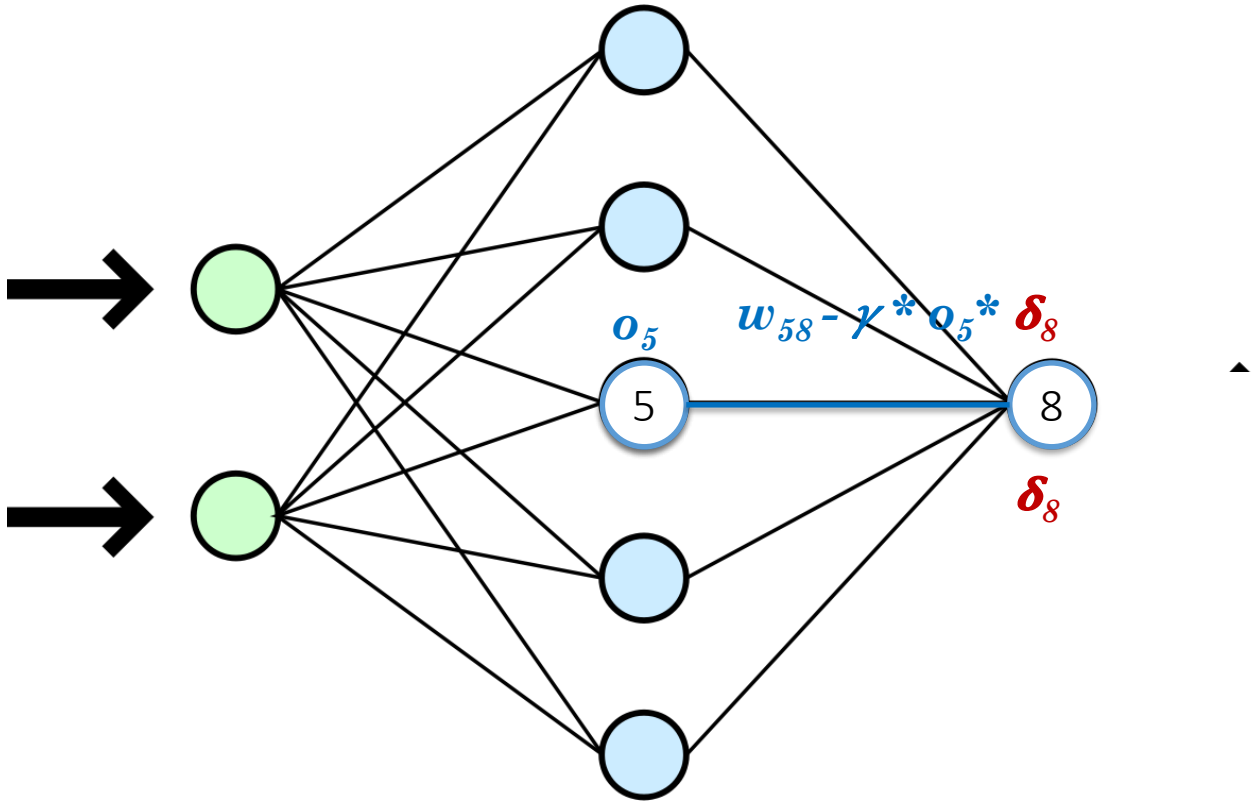
Mise à jour des poids



Mise à jour des poids



Mise à jour des poids



Conclusions hâtives

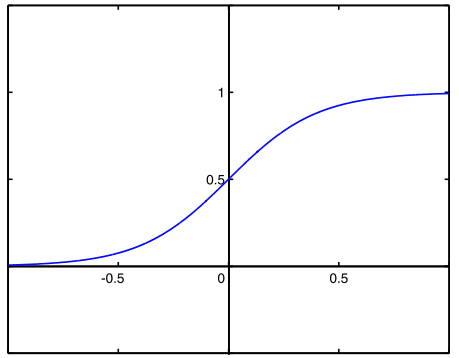
- Paradigme générique de “programmation” de fonctions
- Pas besoin de modèle, uniquement de données

- Généralisation à des architectures de réseaux plus complexes
- Problème d’évanouissement du gradient
- Problème des choix des hyper-paramètres du modèle
- Problème du biais des données choisies

Zoom on Transfer Functions

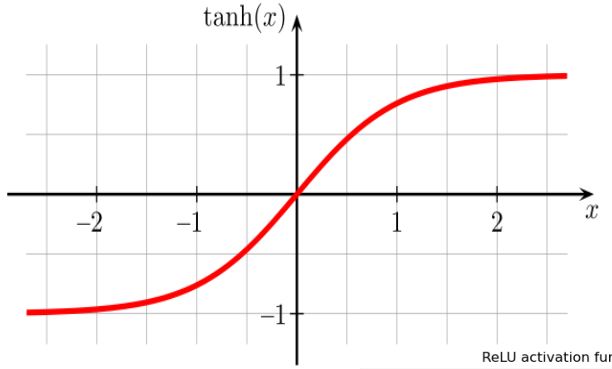
- Sigmoid

$$f(x) = \frac{1}{1+e^{-x}} \; ; \; f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x)\big(1 - f(x)\big)$$



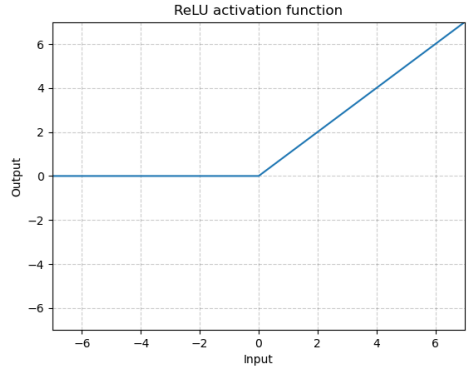
- Tanh

$$f(x) = \frac{1 - e^{1-2x}}{1 + e^{1-2x}} \; ; f'(x) = 1 - f^2(x)$$



- ReLU

$$f(x) = \max(0, x) \; ; f'(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise} \end{cases}$$



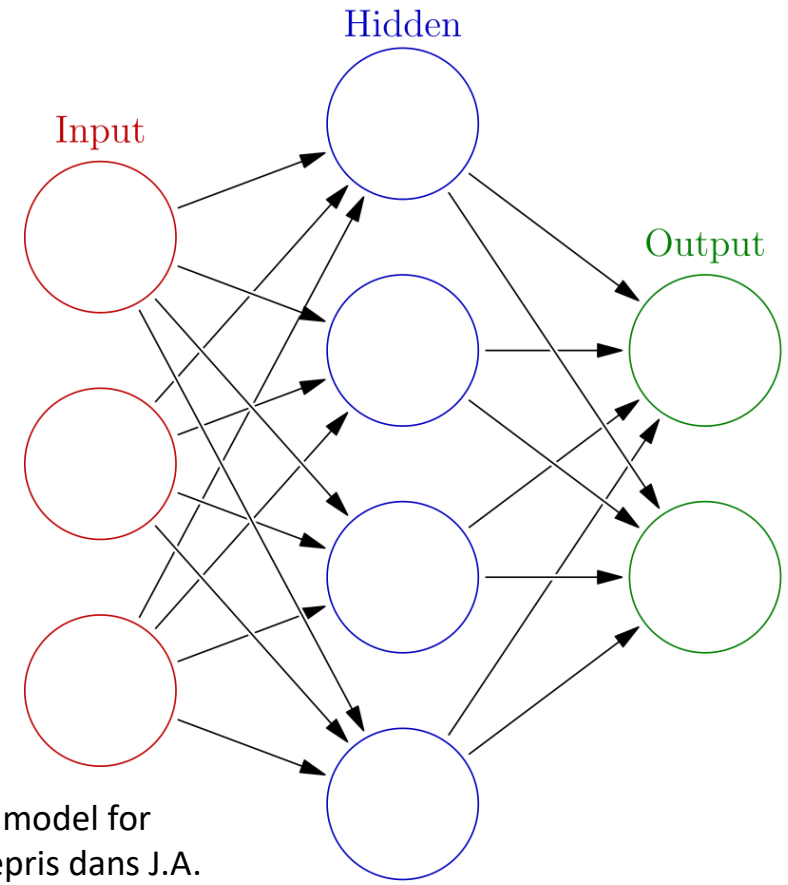
Vanishing Gradient problem

- Saturating neurons
- Zero derivative
- Backpropagation blocked

Networks

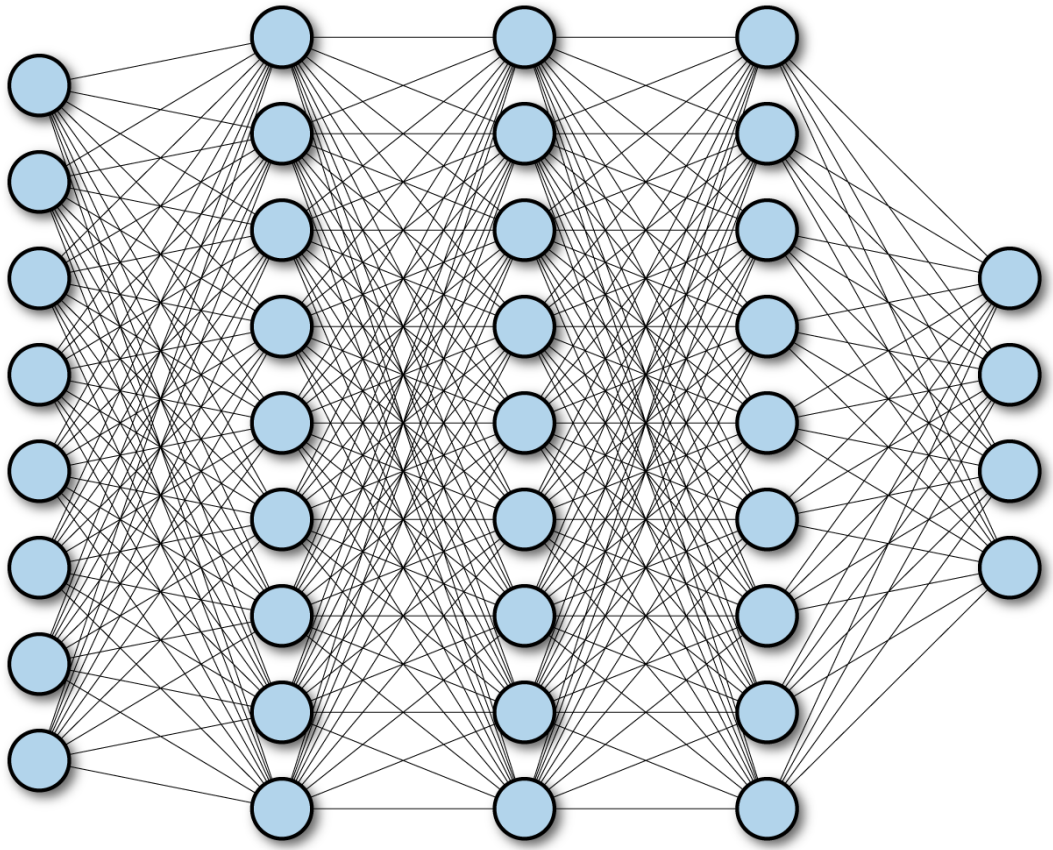
- Hidden Layer Perceptron
- FCNN – Fully Connected Neural Network
- CNN – Convolutional Neural Network
- RNN – Recurrent Neural Network
  - R-CNN – Recurrent Convolutional Neural Network
  - LSTM – Long Short-Term Memory
  - GRU – Gated Recurrent Unit
- AutoEncoder
- Transformer
- GAN – Generative Adversial Network

Hidden Layer Perceptron



F. Rosenblatt (1958), "The perceptron: a probabilistic model for information storage and organization in the brain", repris dans J.A. Anderson & E. Rosenfeld (1988), Neurocomputing. Foundations of Research, MIT Press

FCNN – Fully Connected Neural Network

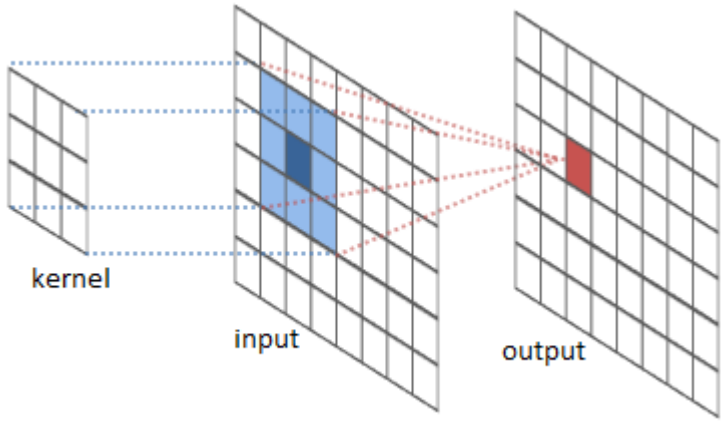


https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/assets/tfdl_0402.png

CNN – Convolutional Neural Network

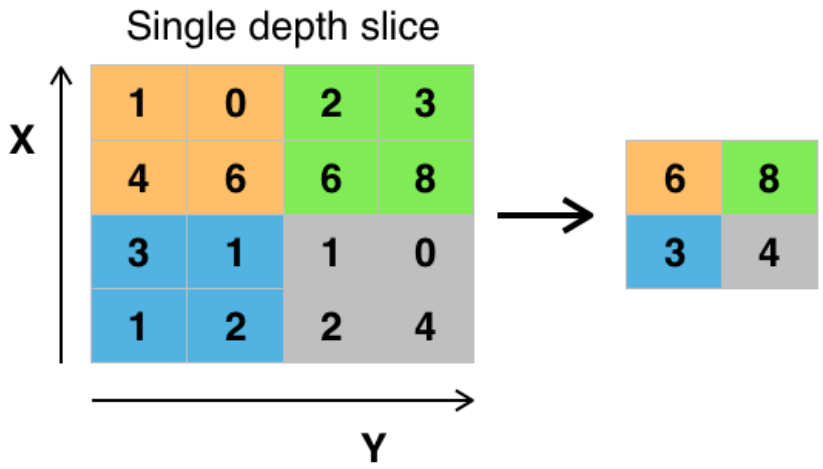
- Convolution
- Pooling (*subsampling*)
- Aggregation

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$



https://mathinfo.alwaysdata.net/wp-content/uploads/2016/11/RiverTrain-ImageConvDiagram.png

Max Pooling



By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45673581

CNN – Convolutional Neural Network

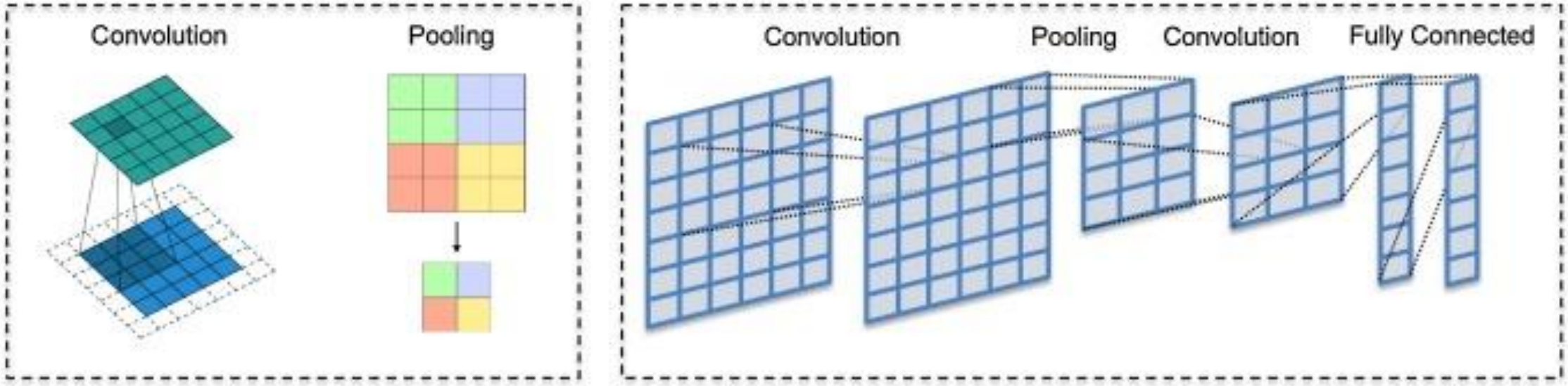
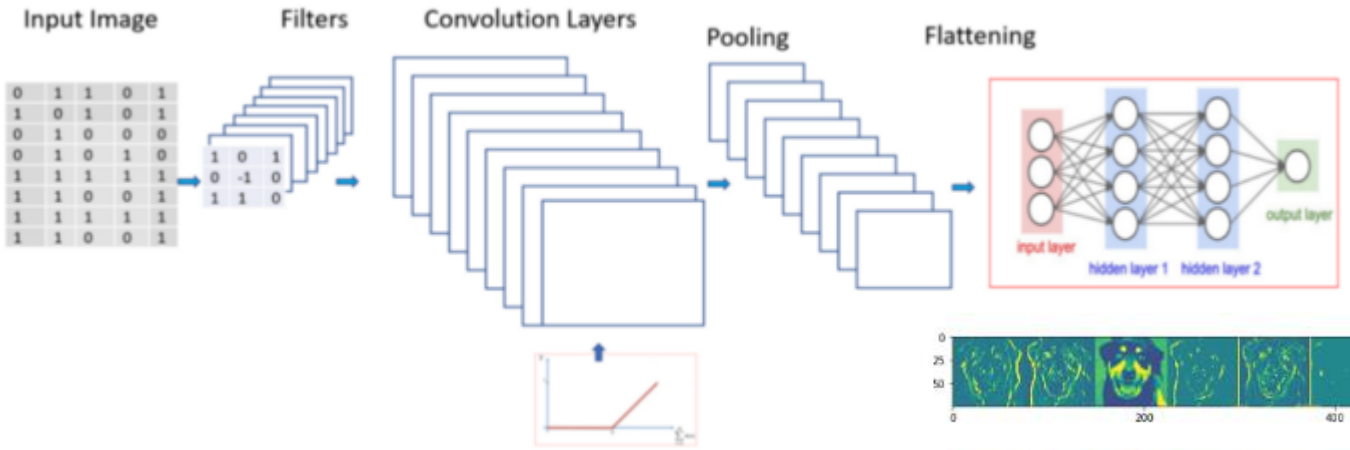
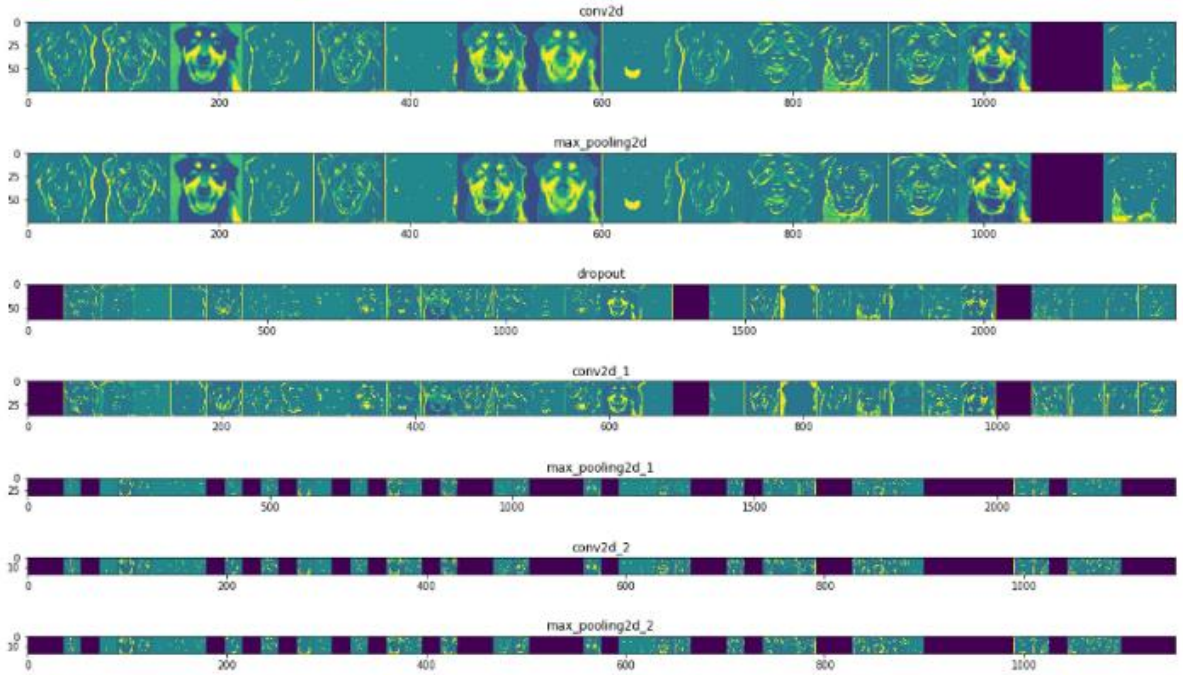


Figure 6 from Andreas Maier, Christopher Syben. Tobias Lasser. Christian Riess. "A gentle introduction to deep learning in medical image processing". Zeitschrift für Medizinische Physik Volume 29, Issue 2, May 2019, Pages 86-101.
https://www.sciencedirect.com/science/article/pii/S093938891830120X#fig0010

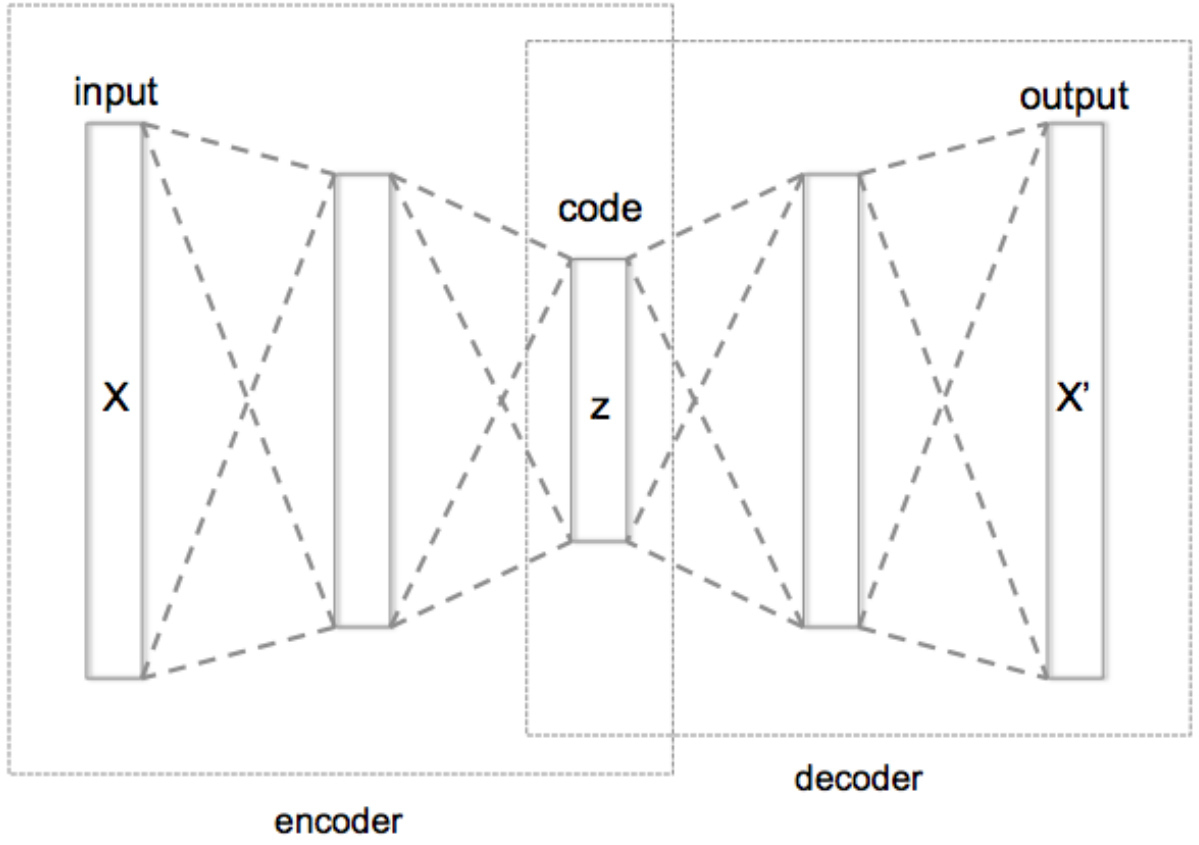
CNN end-to-end





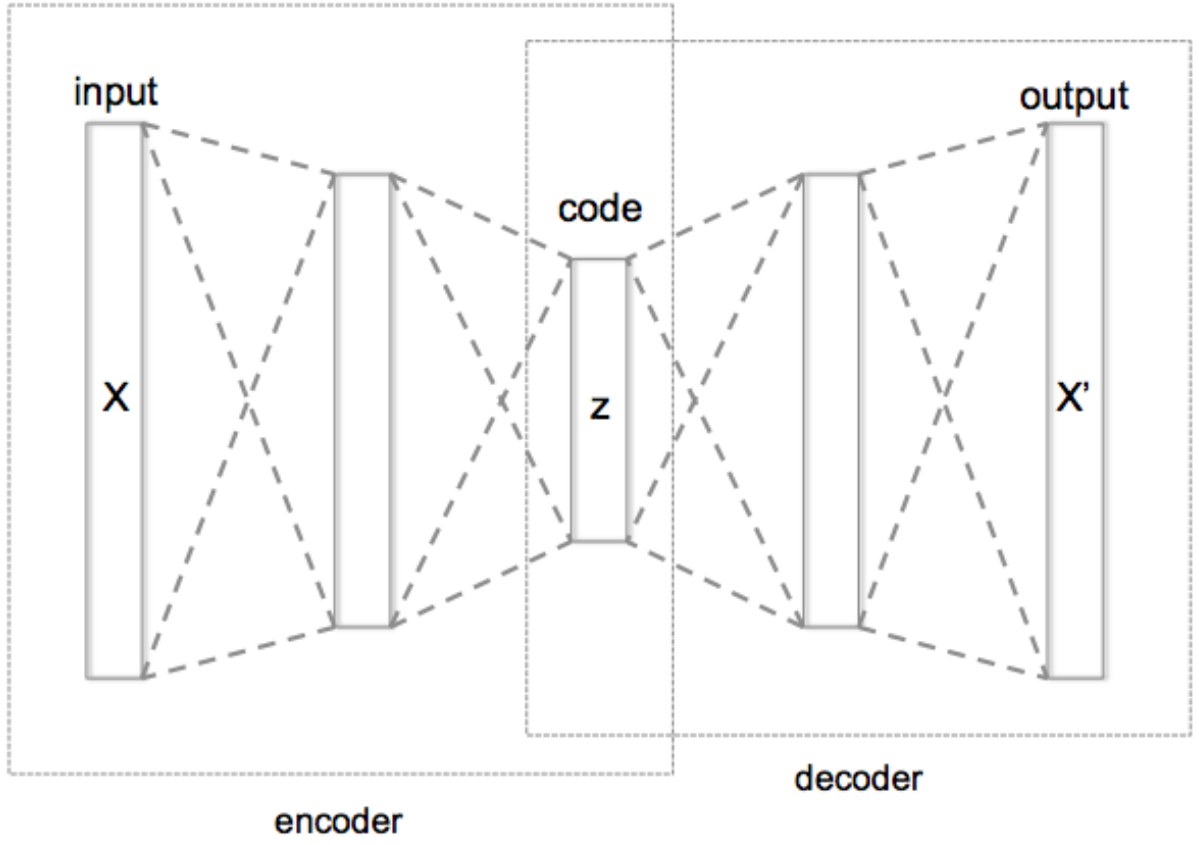
Source : https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization

Autoencoder



Autoencoder

Latent representation space
embedding



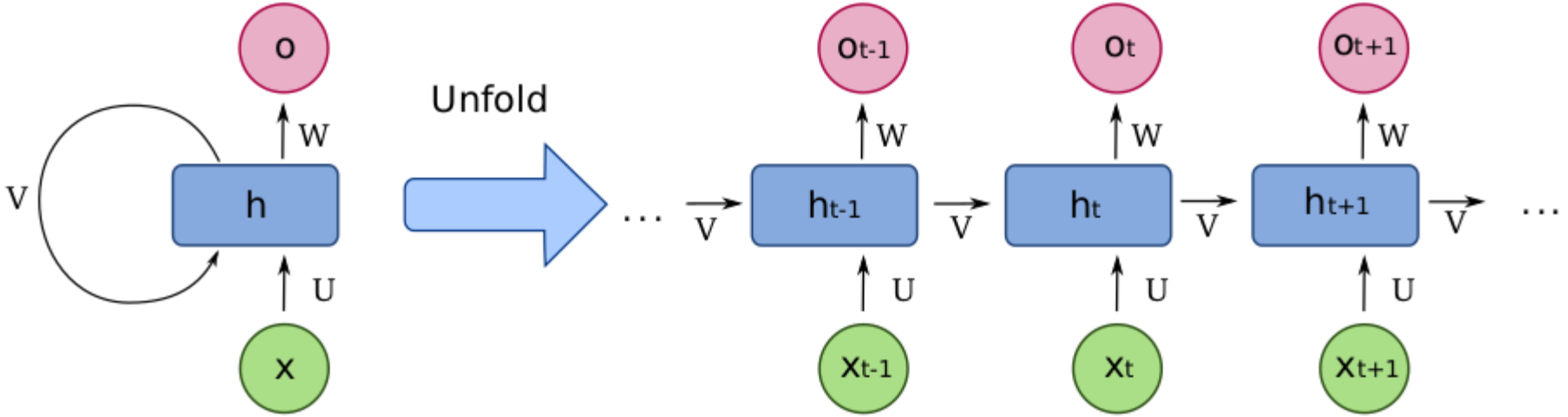
Autoencoder Types

Regularization

- Sparse (SAE)
- Denoising
- Variational (VAE)

Recurrent Neural Networks

- Introduction d'un temps de latence/mémoire dans la construction des réseaux (RNN : Recurrent Neural Networks)
- RNN initialement conçus pour traiter des séries temporelles



Source: Wikipedia

Extensions des RNN

- ***LSTM***

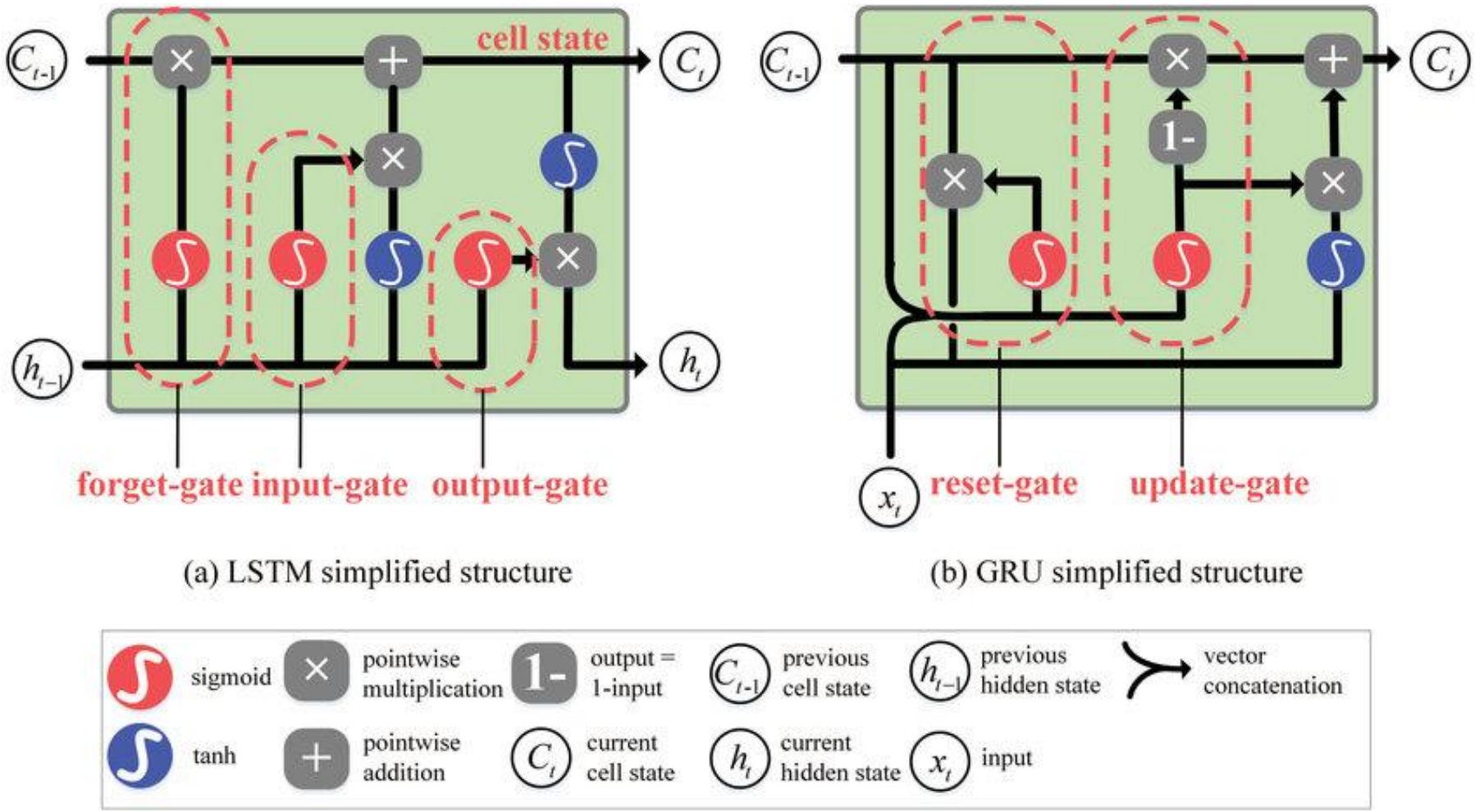
  Long Short-Term Memory, S. Hochreiter, J. Schmidhuber, Neural Computation Volume 9, Issue 8, November 15, 1997, p.1735-1780, https://doi.org/10.1162/neco.1997.9.8.1735

- **GRU** – Gated Recurrent Unit

  Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, 2014. arXiv:1406.1078

- **RCNN** – Recurrent Convolutional Neural Networks

LSTM *vs*. GRU

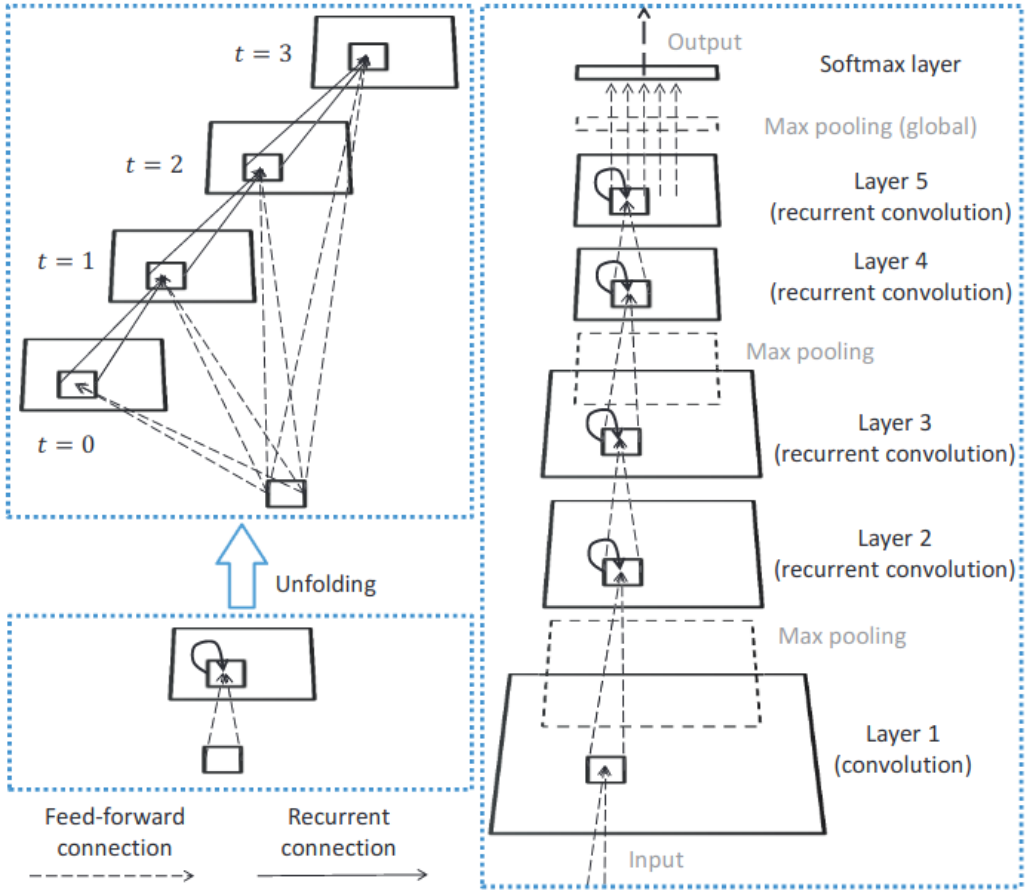


(a) LSTM simplified structure

(b) GRU simplified structure

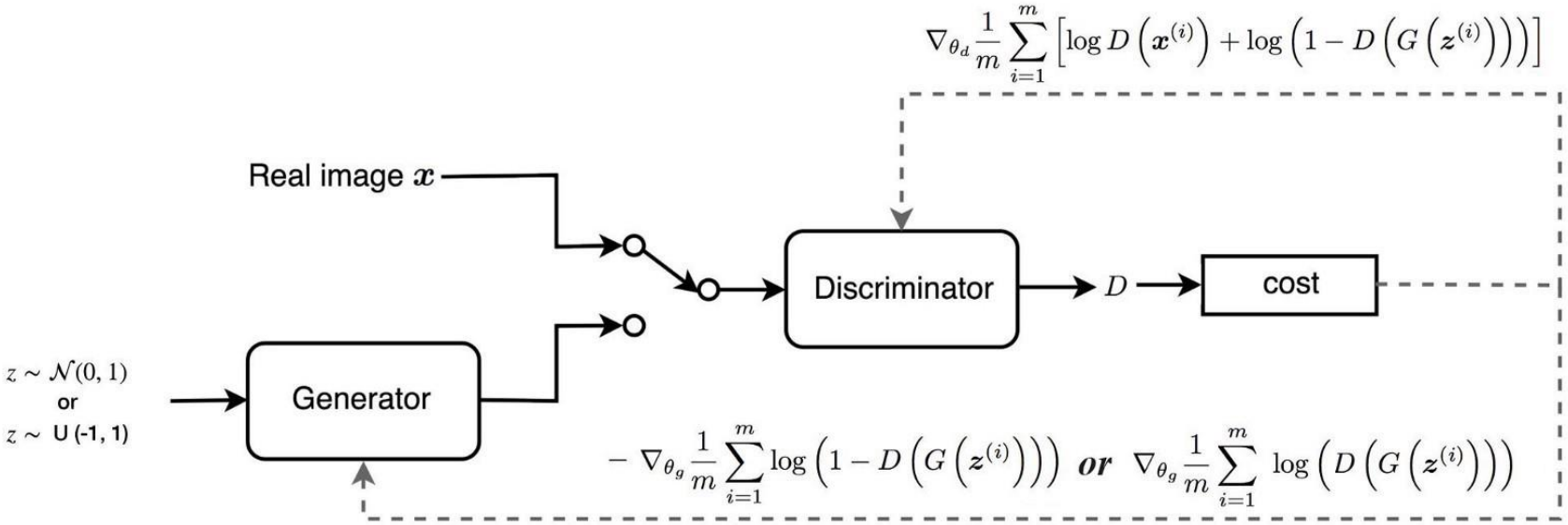
Zhao, Junhui & Nie, Yiwen & Shanjin, Ni & Sun, Xiaoke. (2020). Traffic Data Imputation and Prediction: An Efficient Realization of Deep Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2978530.

Recurrent Convolutional Networks



Source : [Recurrent Convolutional Neural Network for Object Recognition](), M. Liang, X. Hu, CVPR 2015

Generative Adversarial Networks



Source : https://medium.com/@jonathan_hui