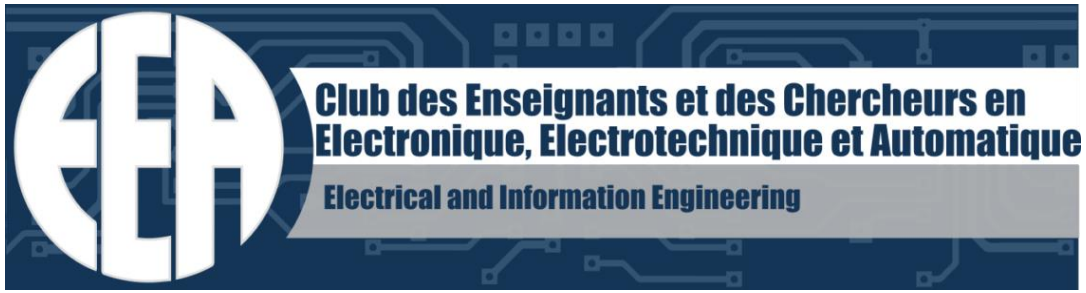
Architectures pour l'apprentissage profond

Bart.Lamiroy@univ-reims.fr

février 2024



CC BY NC

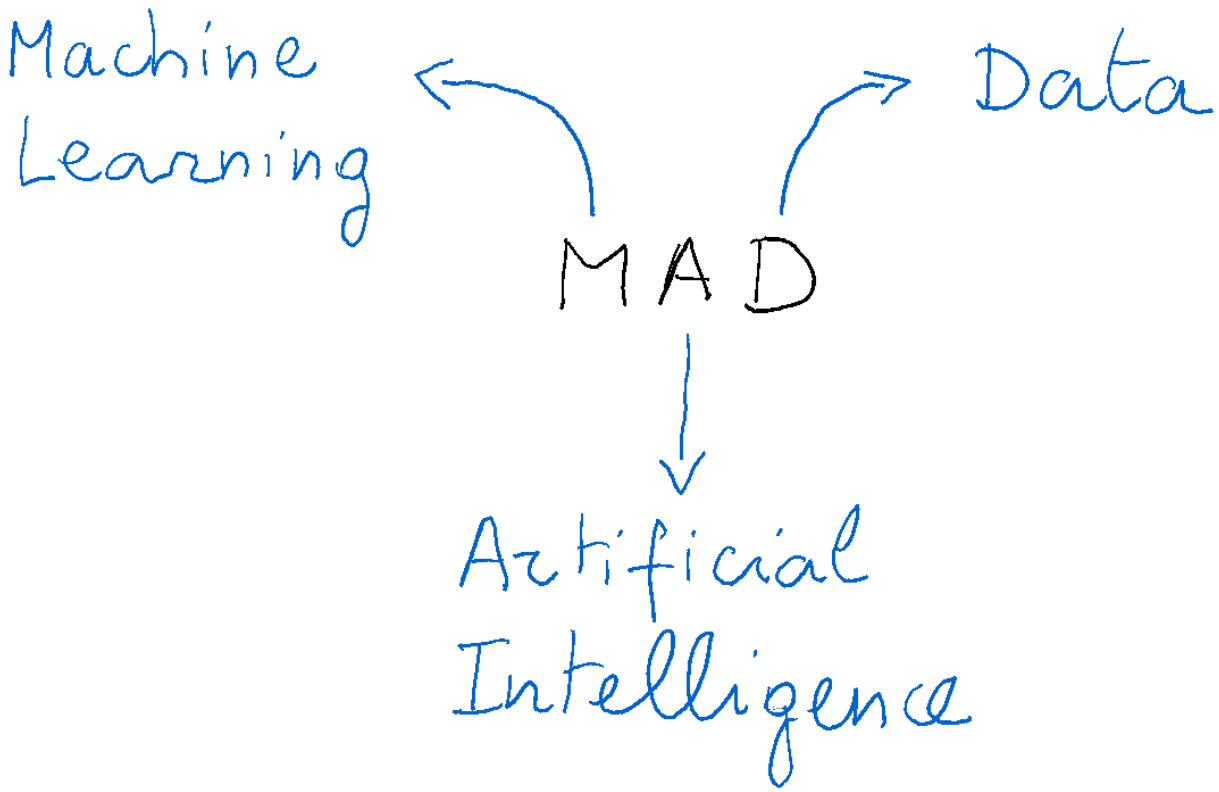
Intelligence Artificielle ?

- Evolution technologique
- Changement de culture
- Opportunité d'évolution

Source: giphy

CC BY NC

MAD …



CC BY NC

Quelques mises en garde (voire conseils)

- Attention au vocabulaire anthropocentrique

  « apprentissage », « intelligence », « décider », « voir », « choisir », « savoir », « comprendre » …

- Quelqu'un qui se prétend « Expert en IA » est ~~très probablement~~ un mytho.
- Les projets de mise en œuvre MAD nécessitent **obligatoirement** des compétences transversales

  (et donc des aller-retours entre experts en techniques de MAD et professionnels du domaine)

- Les moteurs actuels de l'adoption d'outils MAD et leur création viennent de « bricoleurs » éclairés … n'hésitez pas d'essayer.

CC BY NC

Objectifs généraux

Présentation des grandes familles de réseaux de neurones profonds

Comprendre

- Comment leur structure influe sur leurs capacités
- Comment ils sont entrainés
- Quels usages en découlent

Eclairage des LLM et agents conversationnels

CC BY NC

Plan de l’exposé

- Vocabulaire et fondements scientifiques
- Quelques rappels sur le fonctionnement des réseaux de neurones
- Catégories de réseaux de neurones profonds
- Applications

CC BY NC

Quelques rappels (ou pas)

- Apprentissage supervisé
- Apprentissage auto-supervisé
- Apprentissage non supervisé

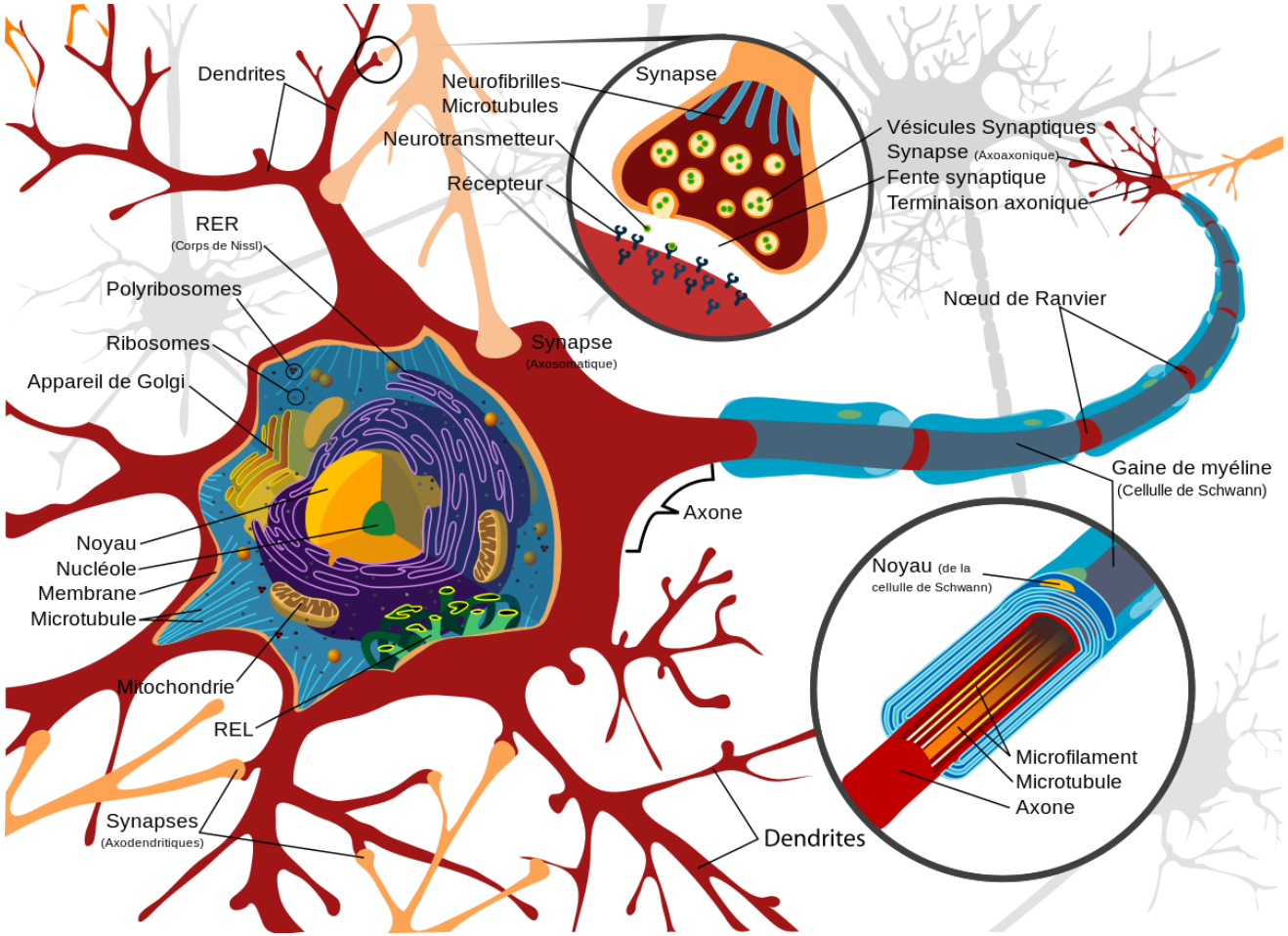
CC BY NC

Réseaux de neurones artificiels

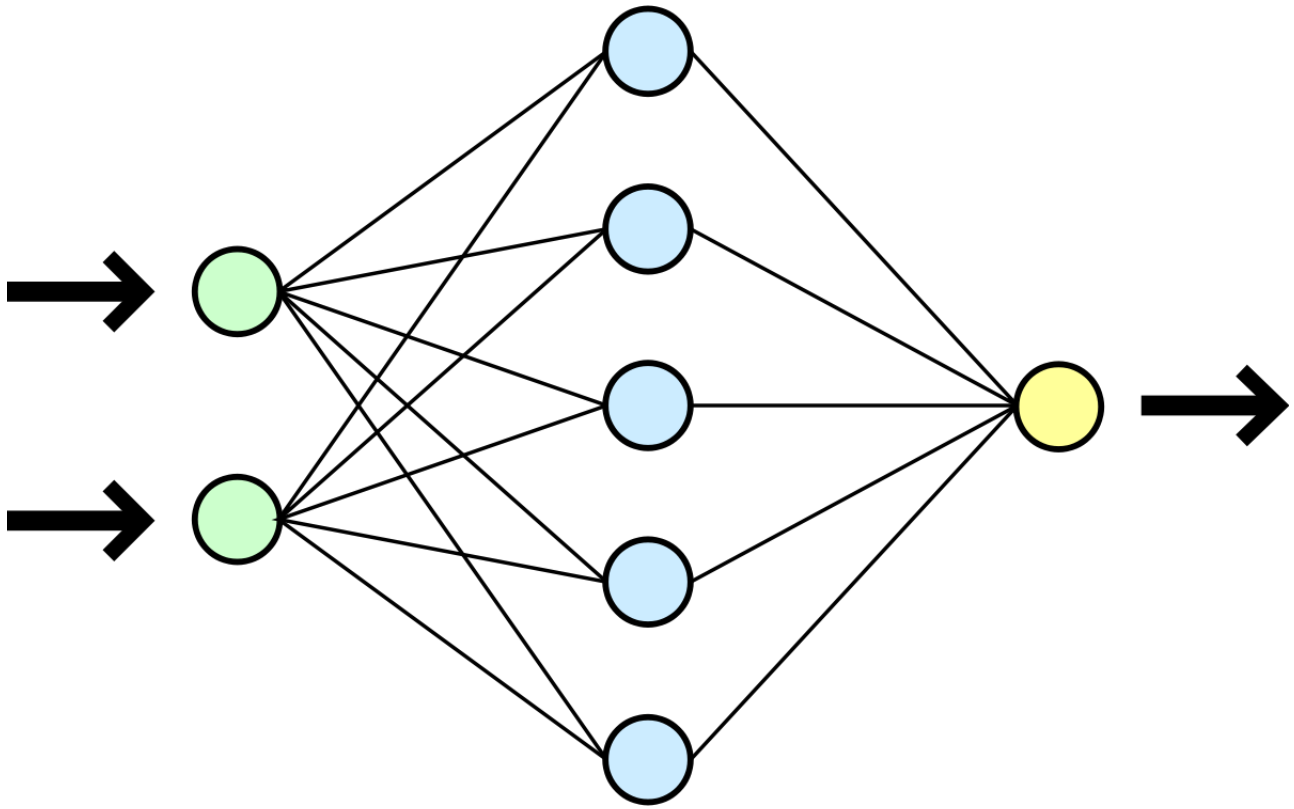
- Connexionnisme
- Fonction d’activation
- Fonction de coût
- Rétropropagation du gradient
- Théorème de l’approximateur universel

CC BY NC

Le neurone biologique et son fonctionnement

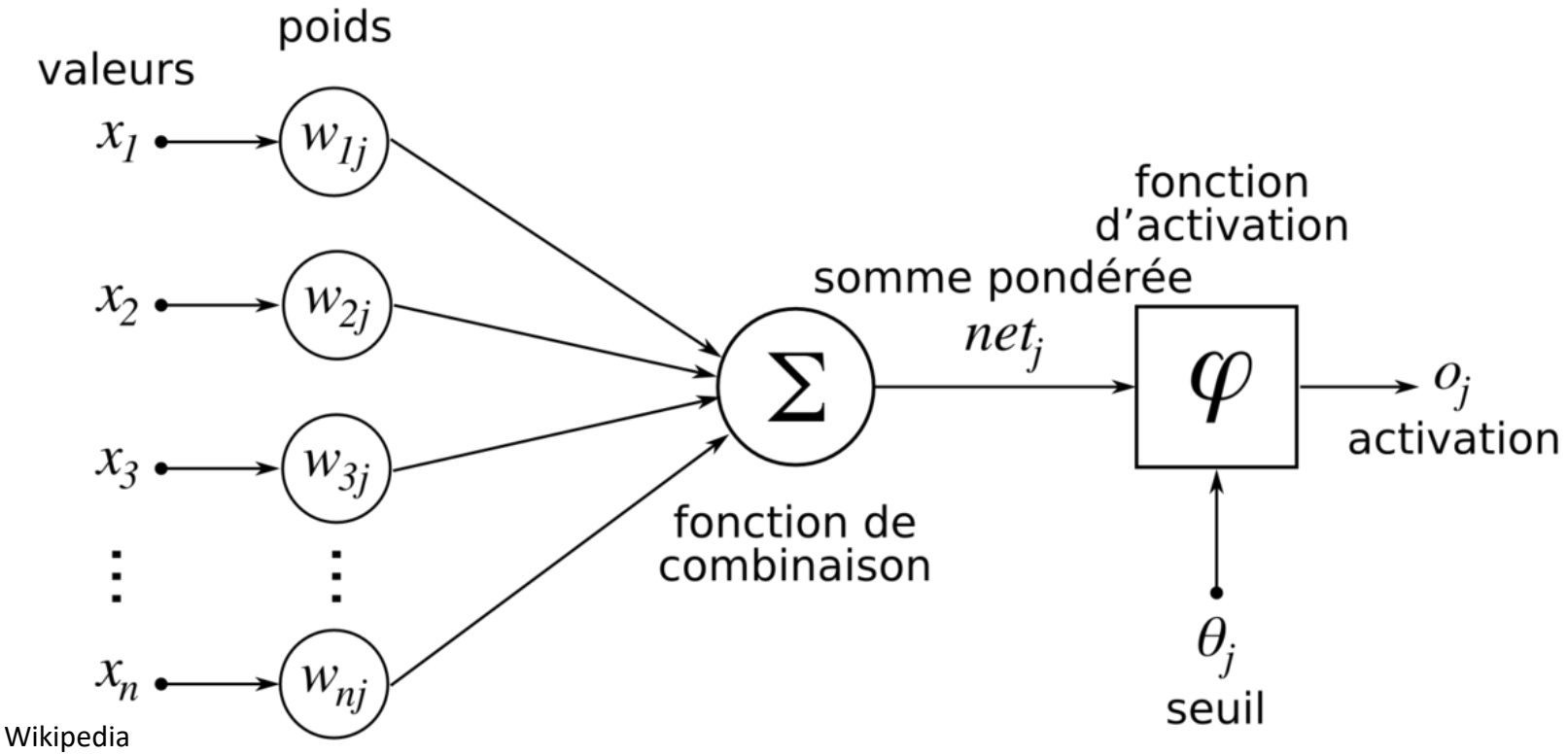


Modèle simplifié : graphe orienté acyclique



BY NC

Le neurone artificiel

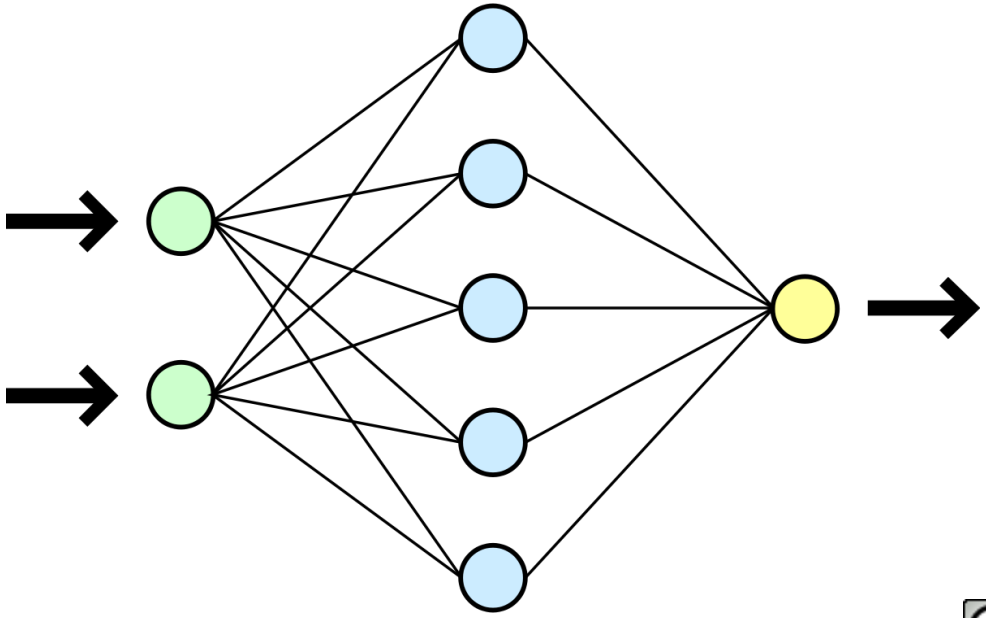


Source : Wikipedia

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

D’un point de vue calculatoire

- Des produits scalaires
- Très parallélisable
- Opérations SIMD (Single Instruction, Multiple Data) très adaptées à des GPU
- Approximateur universel



BY NC

Le théorème de l'approximateur universel

- Etant donnée ϕ une fonction strictement croissante bornée ;
- Soit $\boldsymbol{I_m}$ l'hypercube unite de dimension m et $\mathscr{C}$ l'ensemble des fonctions continues sur $\boldsymbol{I_m}$;
- Alors pour tout ε, il existe $N \in \mathbb{N}$ tel que pour toute fonction $f \in \mathscr{C}$ il existe $v_i, b_i \in \mathbb{R}$ et $w_i \in \mathbb{R}^m$ tels que pour tout $x \in \boldsymbol{I_m}$

$$|F(x) - f(x)| < \varepsilon$$

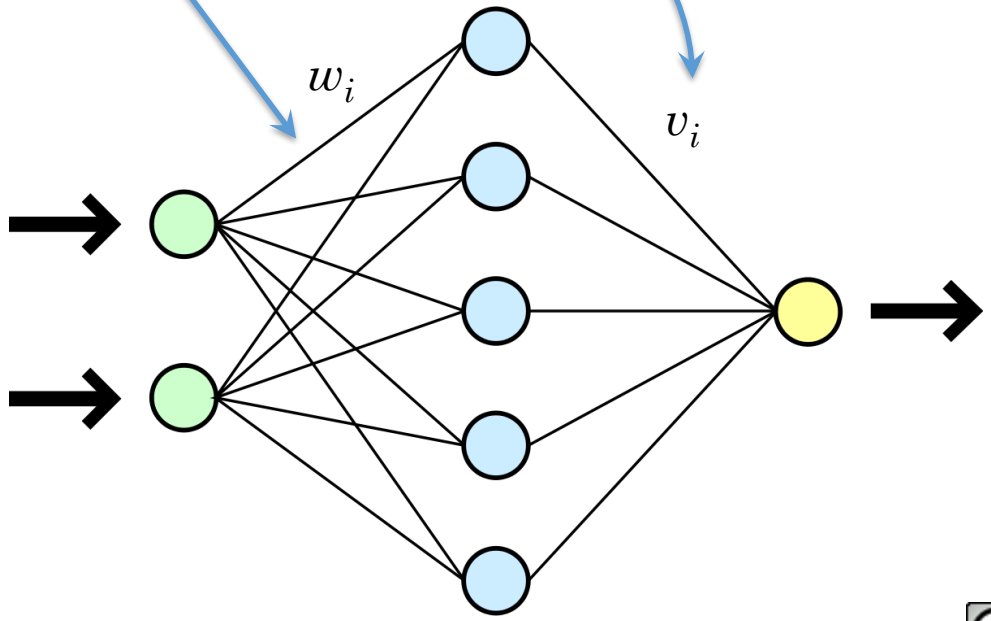
$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

CC BY NC

Le Perceptron

Formule de l'approximateur universel

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



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

CC BY NC

Le Perceptron

Défi : determiner les parameters $v_i, b_i \in \mathbb{R}$ et $w_i \in \mathbb{R}^m$

Solution : approche par *apprentissage supervisé* en formulant le problème comme une *minimisation par descente de gradient*.

Petit souci : on sait que N existe, mais on ne sait pas comment le calculer.

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

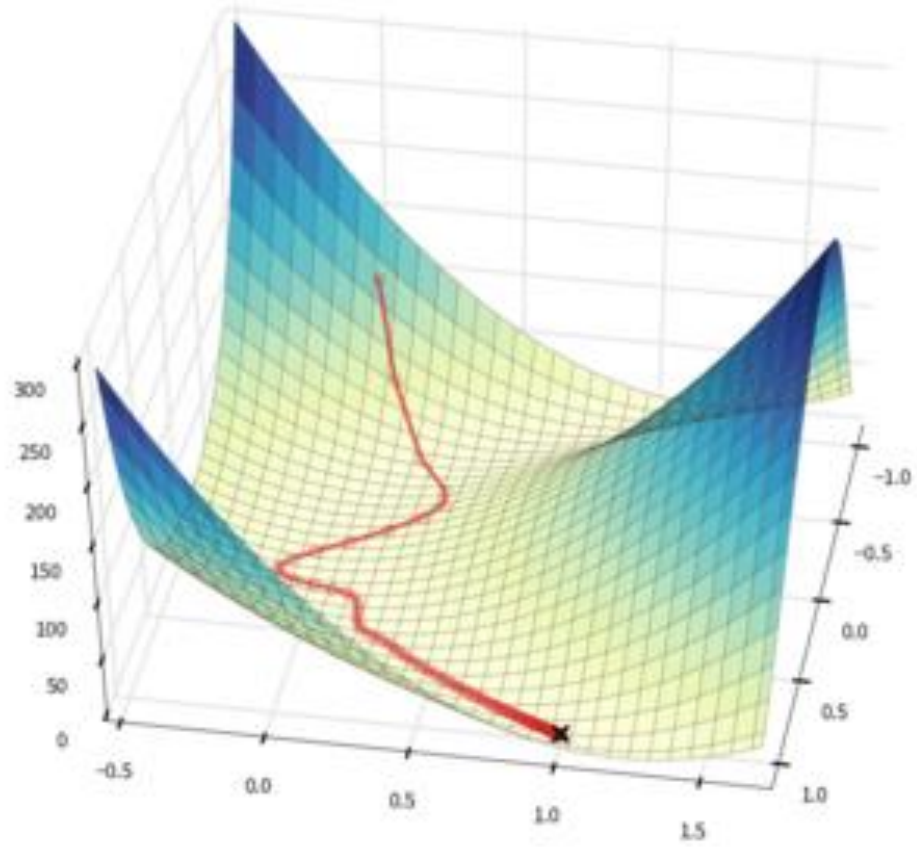
CC BY NC

Apprentissage supervisé et calcul de gradient

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

CC BY NC

Descente de gradient



Source : https://mrmint.fr

Calcul du gradient

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

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

Chaque poids $\boldsymbol{w_i}$ est ajusté avec l'incrément $\Delta \boldsymbol{w_i}$

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

CC BY NC

Calcul du gradient

Erreur observée (fonction de coût)

Sortie attentue

Sortie observée

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

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

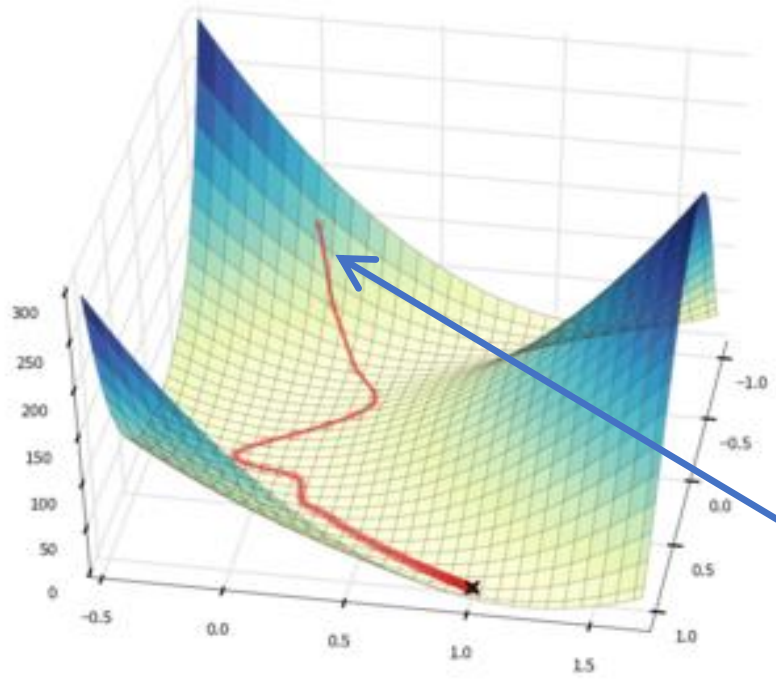
Poids du reseau de neurones

Chaque poids $\boldsymbol{w_i}$ est ajusté avec l'incrément $\Delta \boldsymbol{w_i}$

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

CC BY NC

Calcul du gradient

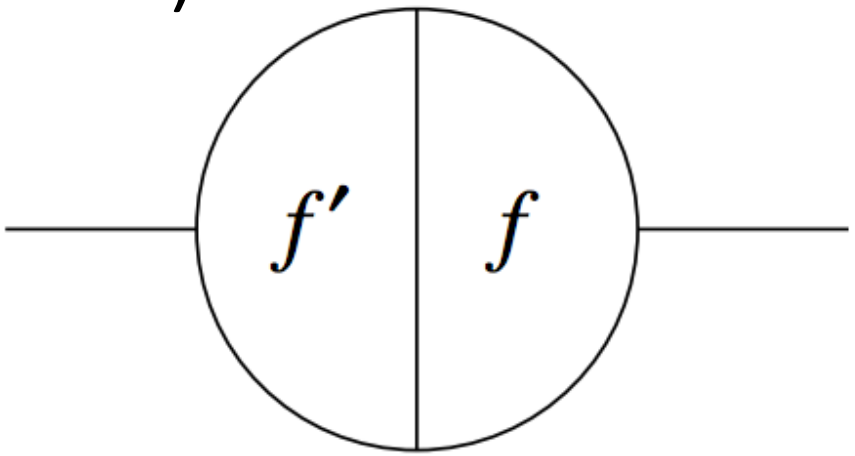


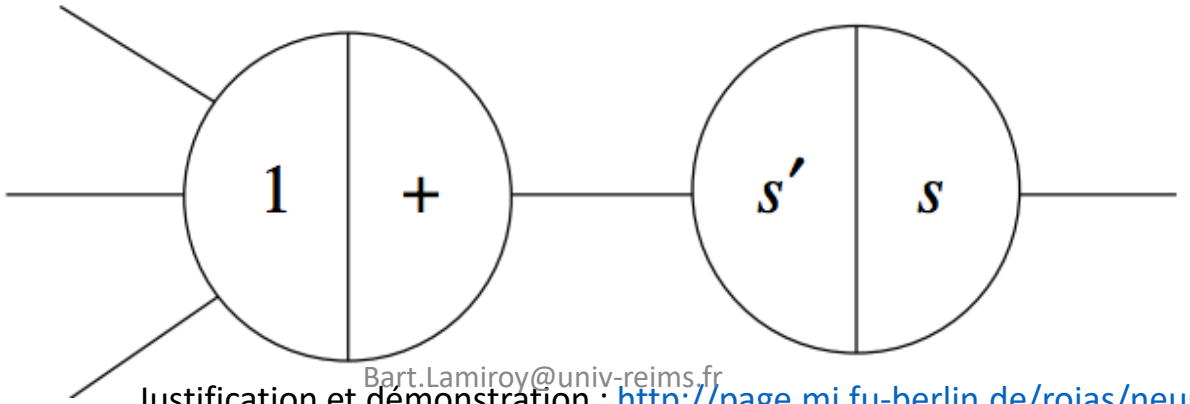
- Learning rate
- Decay
- Momentum

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

Practical Recommendations for Gradient-Based Training of Deep Architectures, Yoshua Bengio, https://arxiv.org/pdf/1206.5533v2.pdf

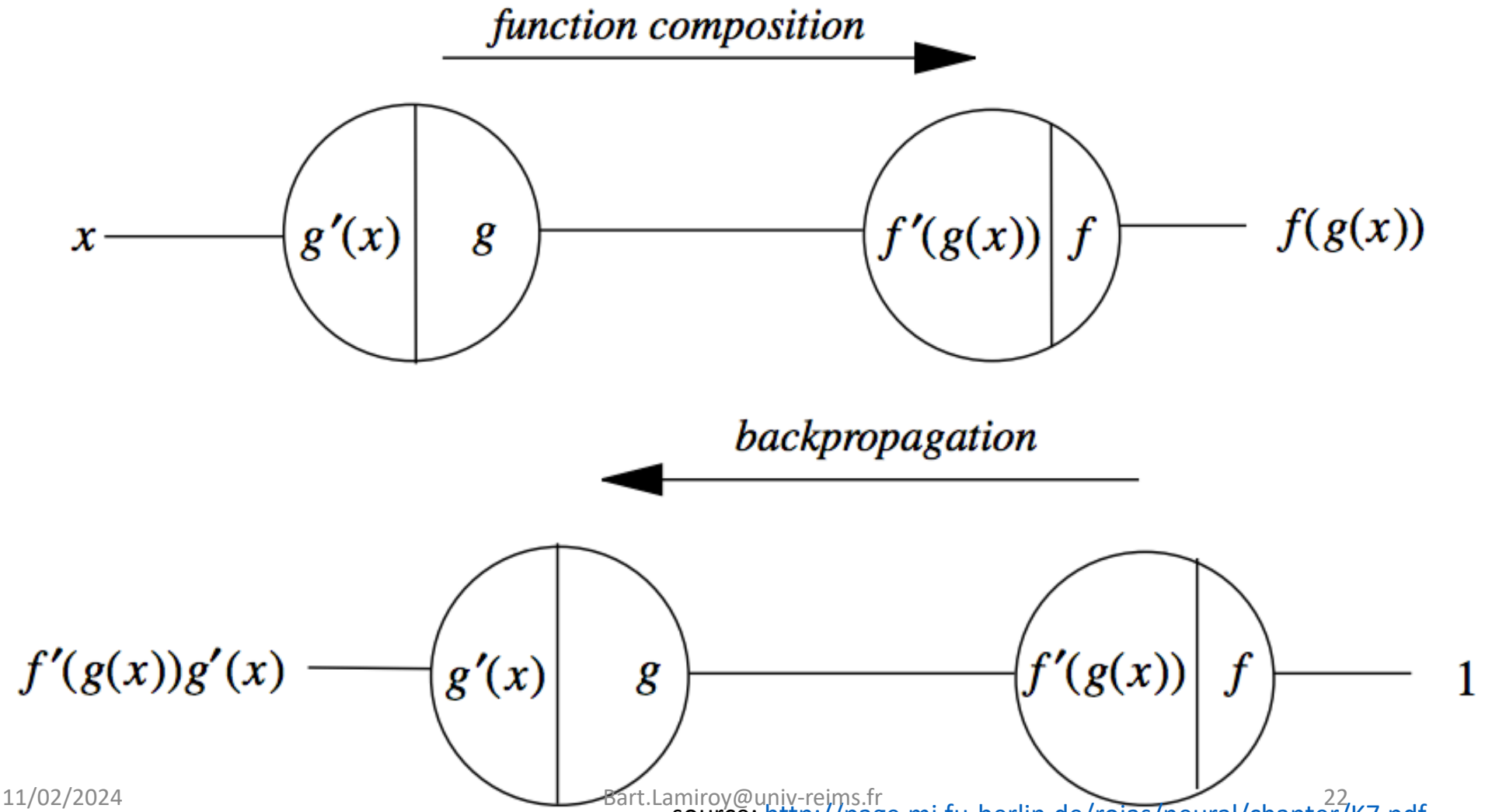
Algorithme de rétropropagation du gradient (cas à une variable)





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

Algorithme de rétropropagation



source: http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf

Résultat

Il est assez trivial de démontrer que

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

Où δ_j est la valeur de la fonction de coût rétropropagée au nœud j et où o_i est la valeur calculée au nœud i lors du calcul *feed-forward*.

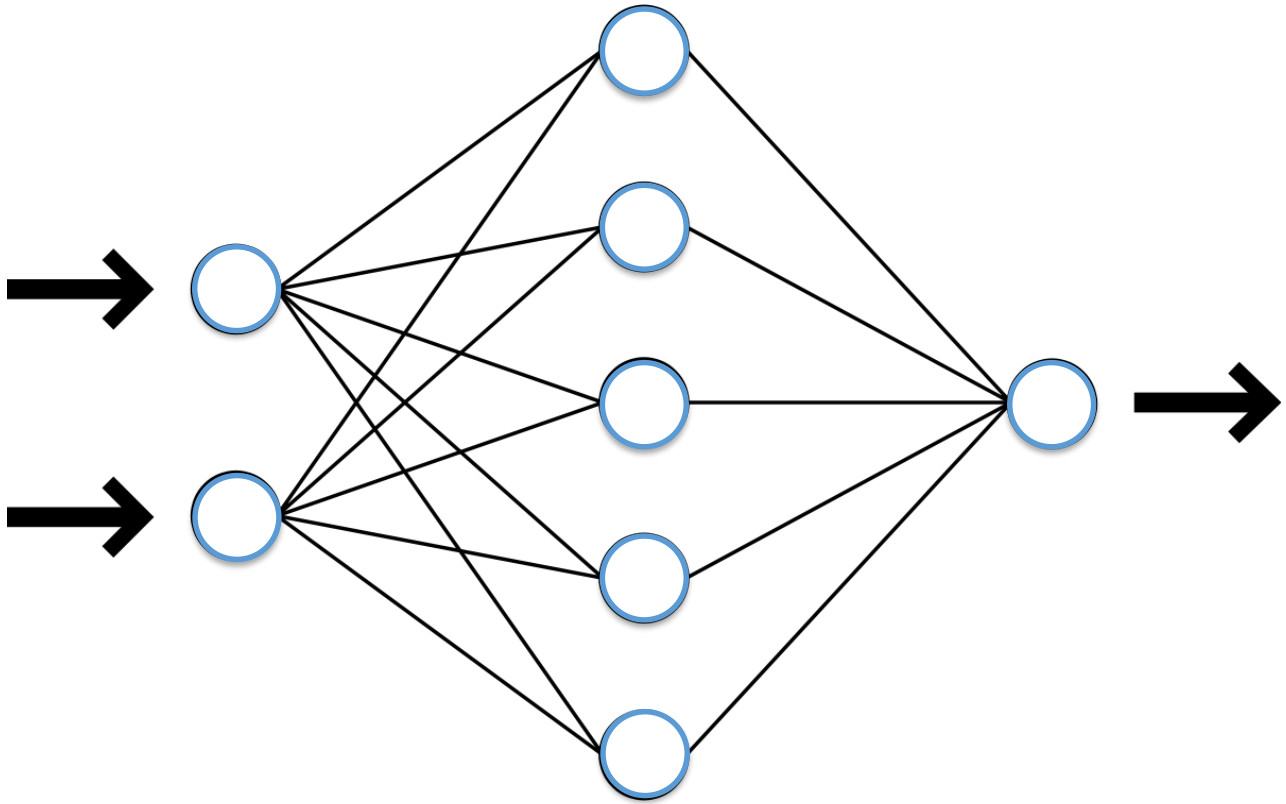
CC BY NC

Algorithme final d'apprentissage supervisé

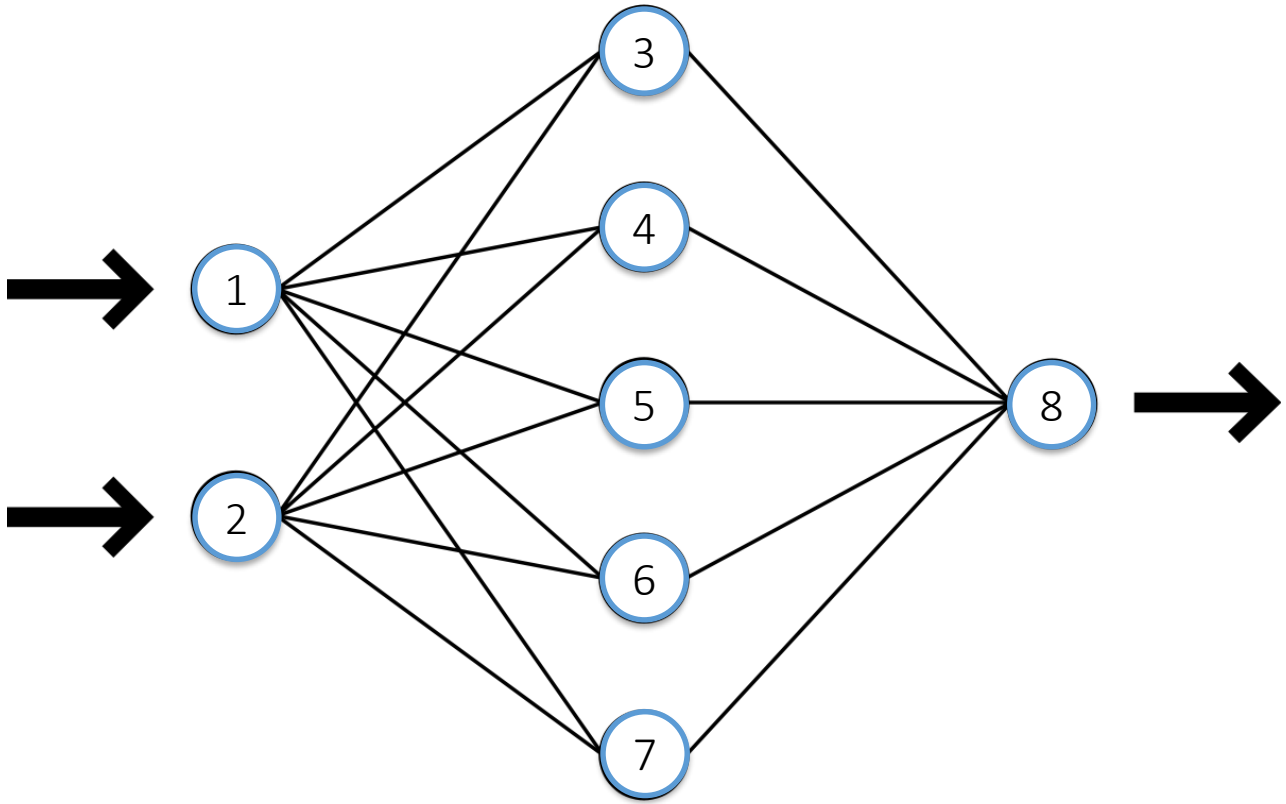
- Recueillir "suffisamment" de données de type $(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}))$
- Initialiser les poids $\boldsymbol{w_{ij}}$ du réseau de neurones à des valeurs aléatoires
- De façon itérative :
  - Exécuter le réseau en *feed-forward* avec une valeur $\boldsymbol{x_k}$ et mémoriser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

CC BY NC

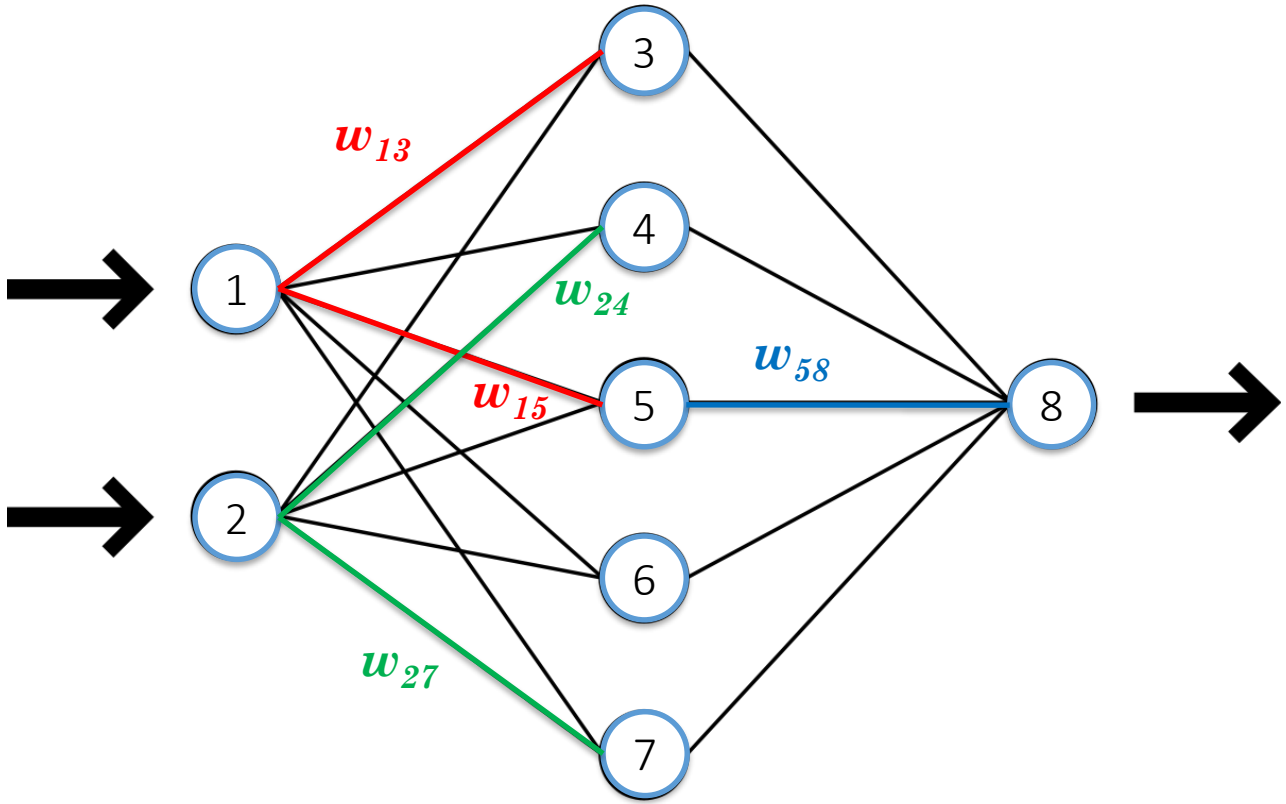
Feed-Forward Phase



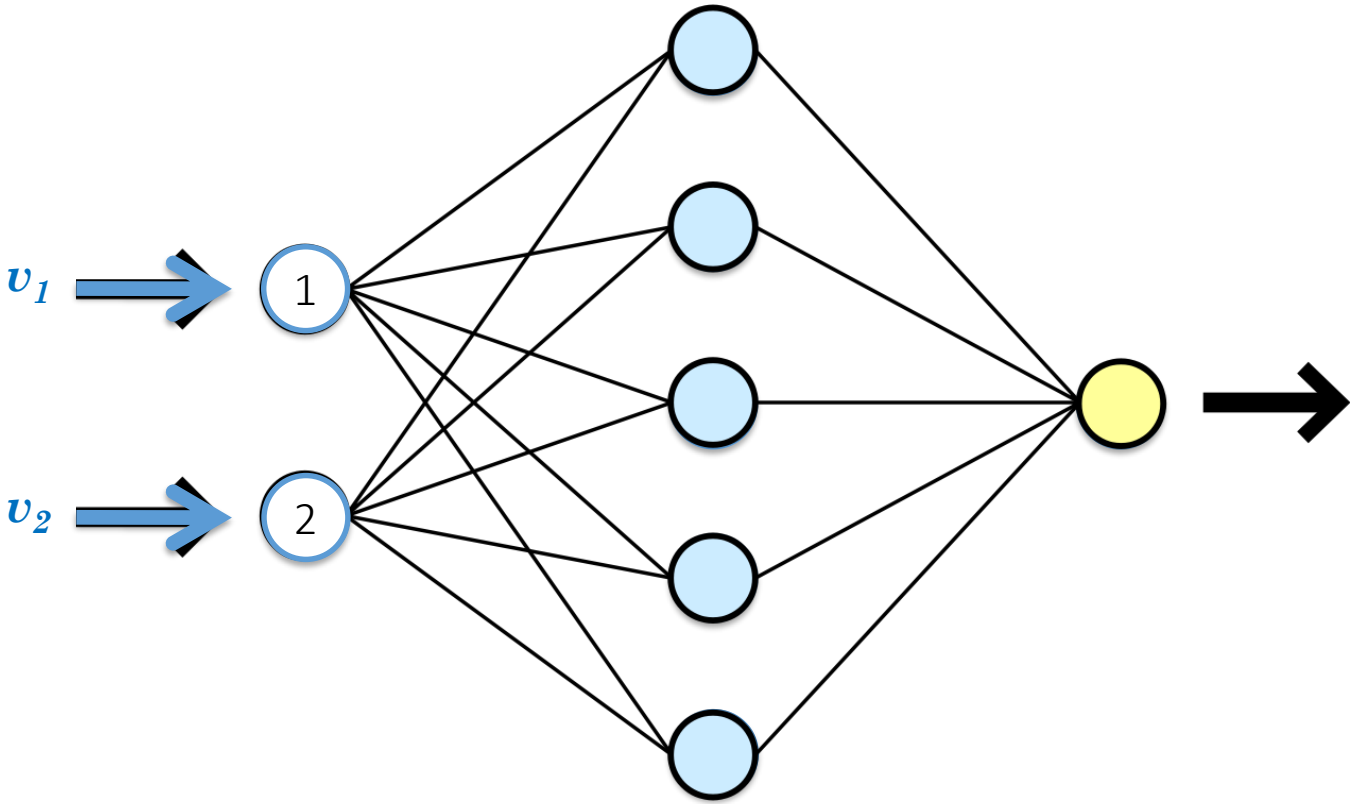
Feed-Forward Phase (notations)



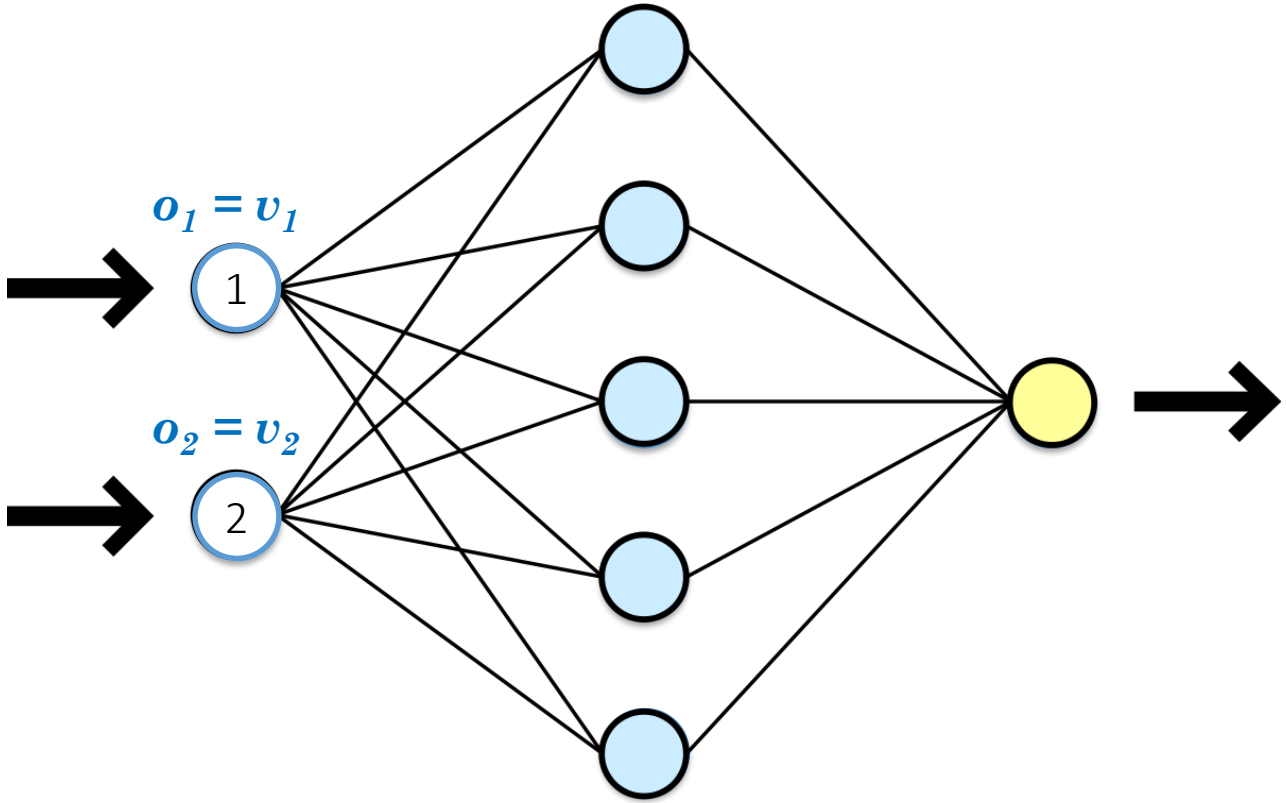
Feed-Forward Phase (notations)



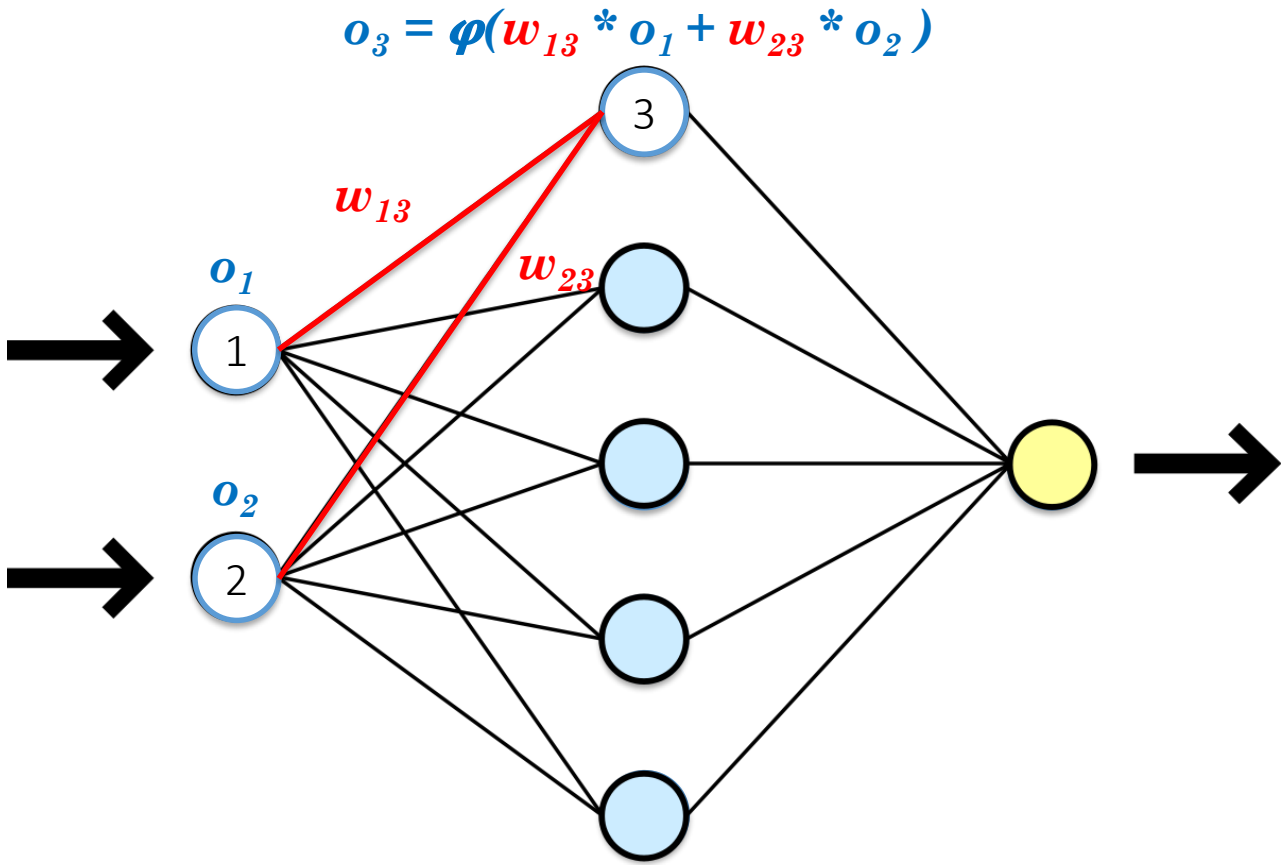
Feed-Forward Phase (Step 0)



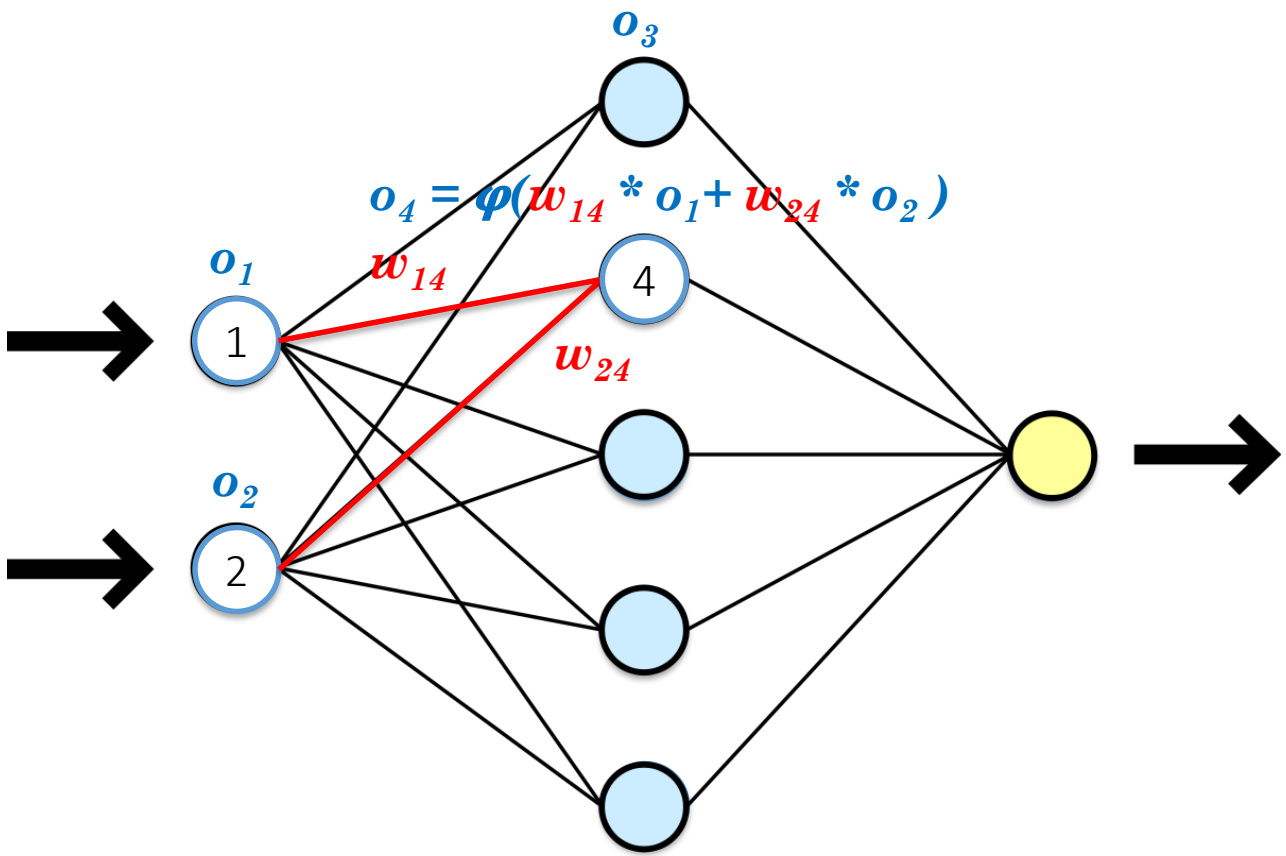
Feed-Forward Phase (Step 1)



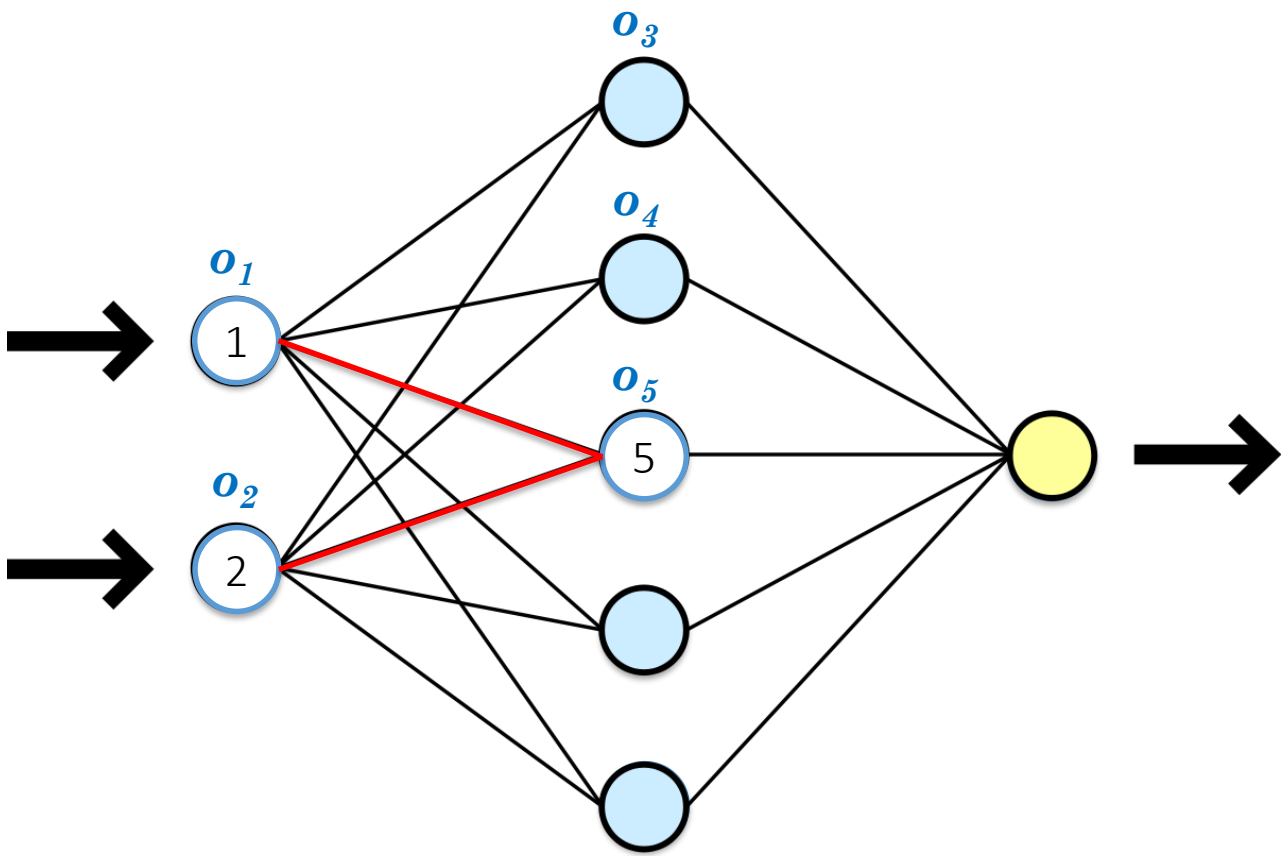
Feed-Forward Phase (Step 2)



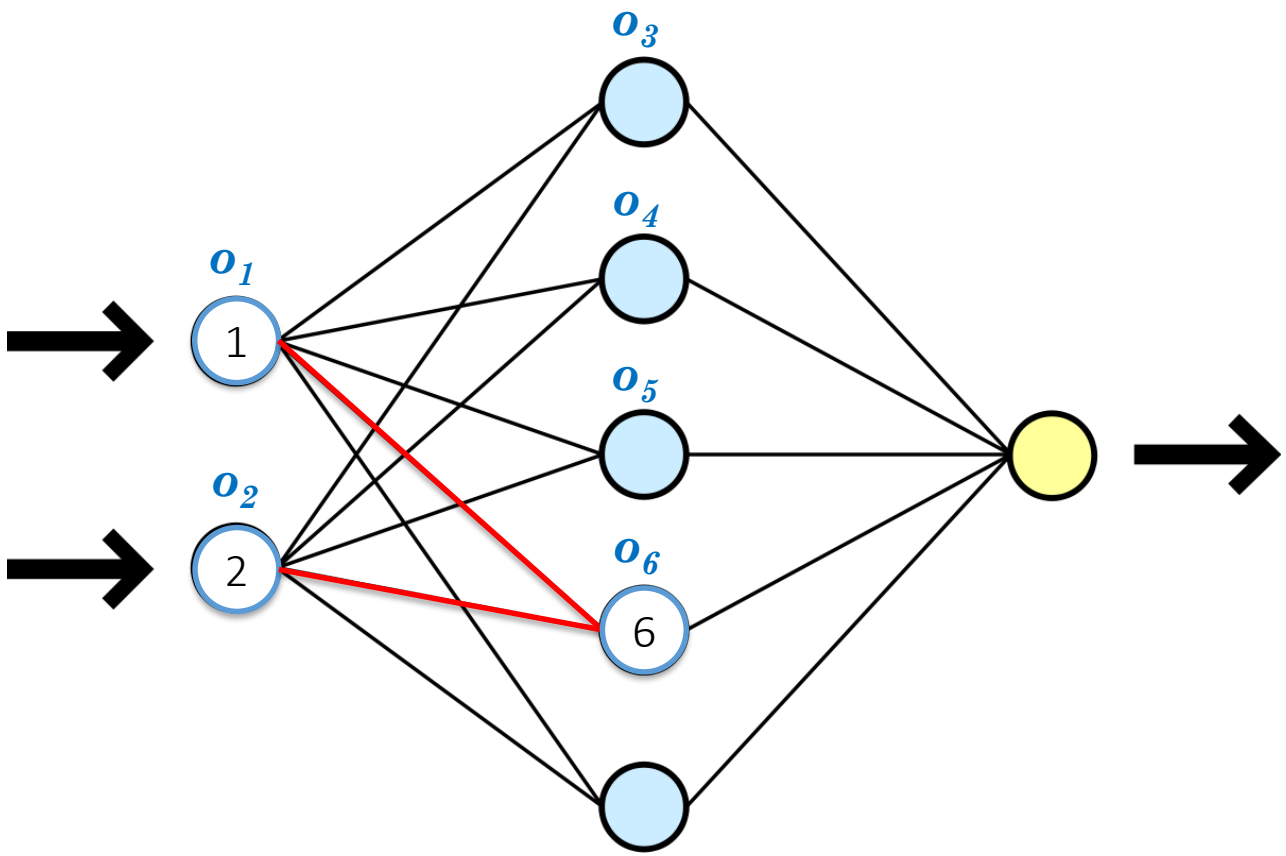
Feed-Forward Phase (Step 2)



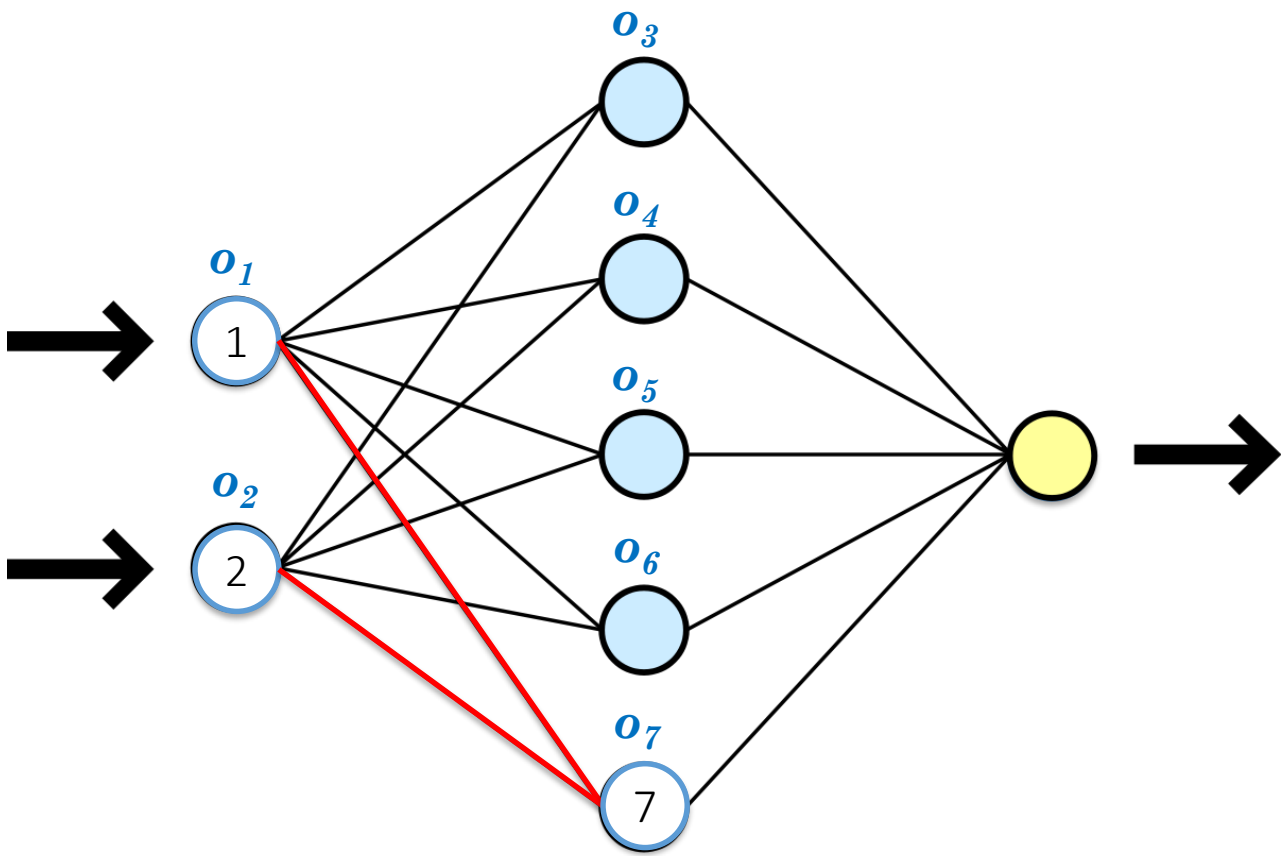
Feed-Forward Phase (Step 2)



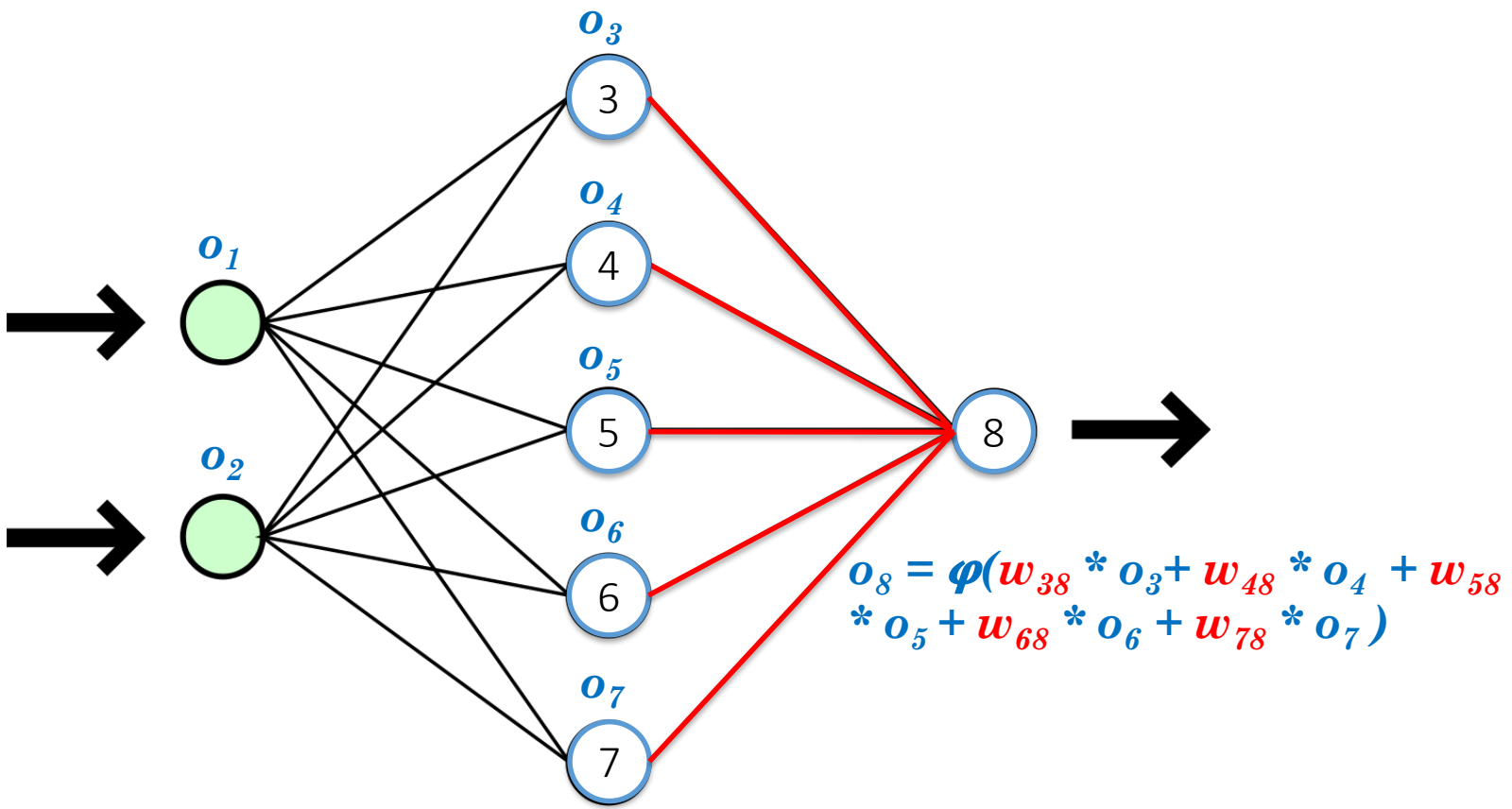
Feed-Forward Phase (Step 2)



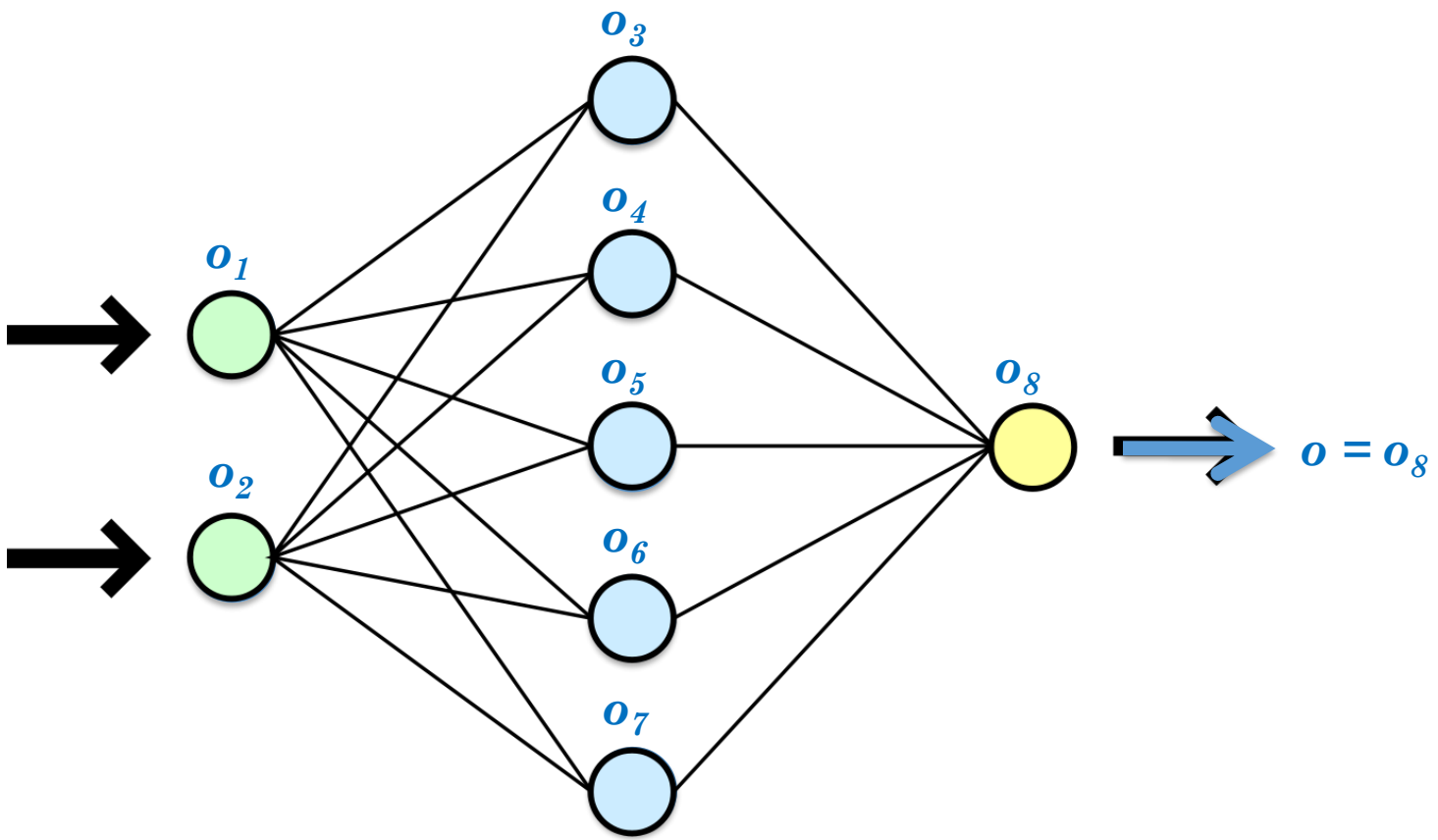
Feed-Forward Phase (Step 2)



Feed-Forward Phase (Step 3)



End of *Feed-Forward* Phase



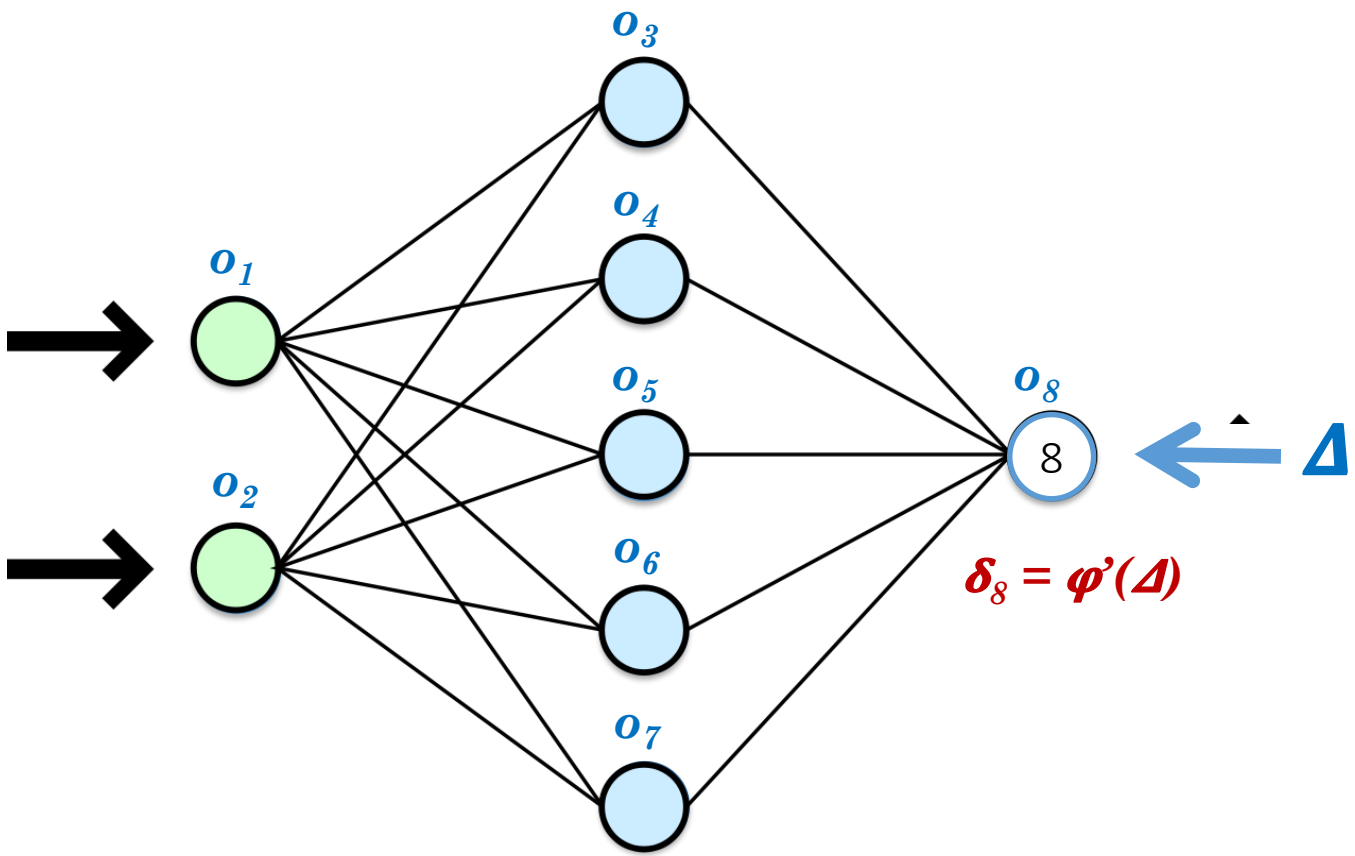
Bart.Lamiroy@univ-reims.fr

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 $\Delta = |o - t|$

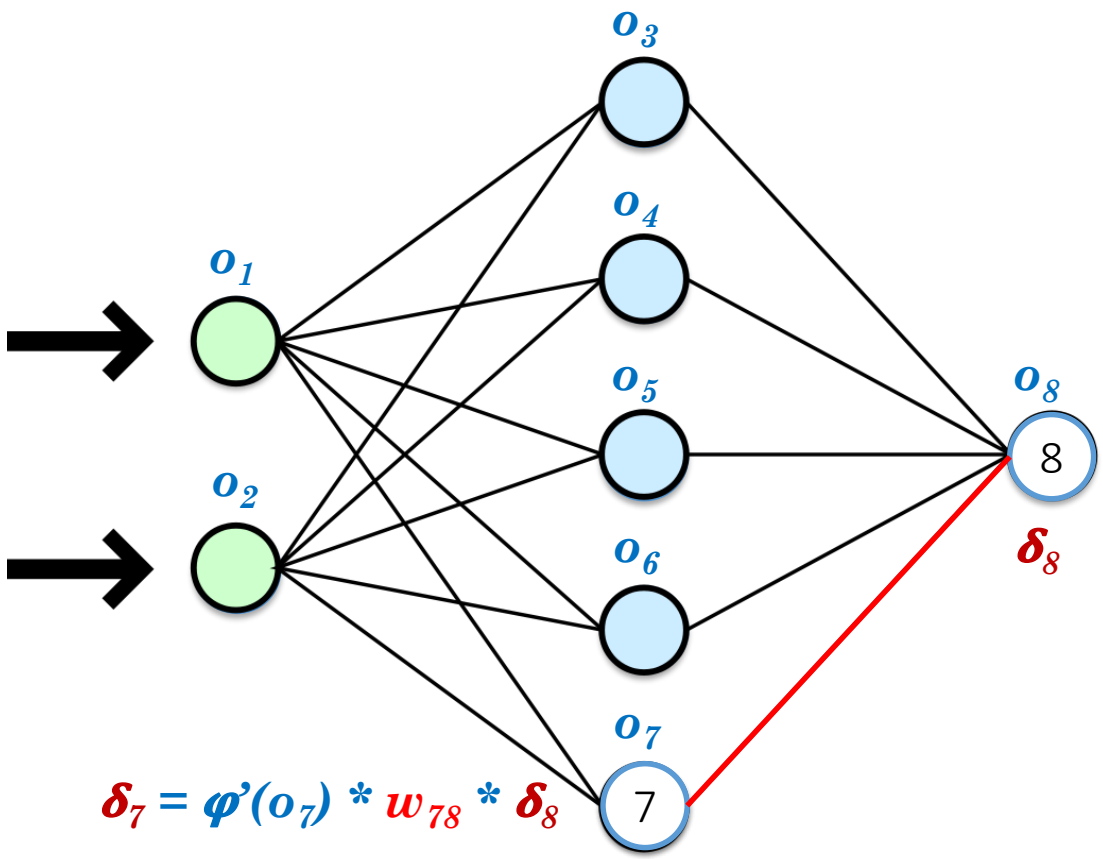
CC BY NC

Phase de rétropropagation



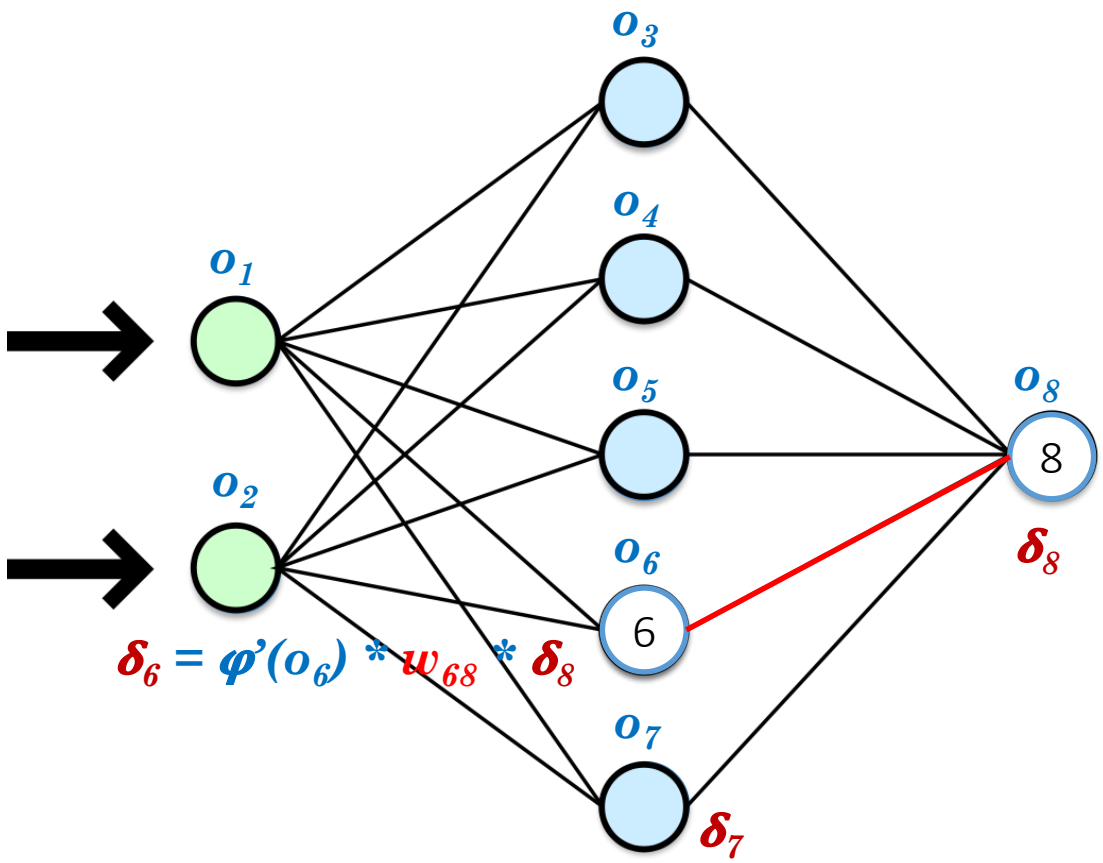
Bart.Lamiroy@univ-reims.fr

Phase de rétropropagation

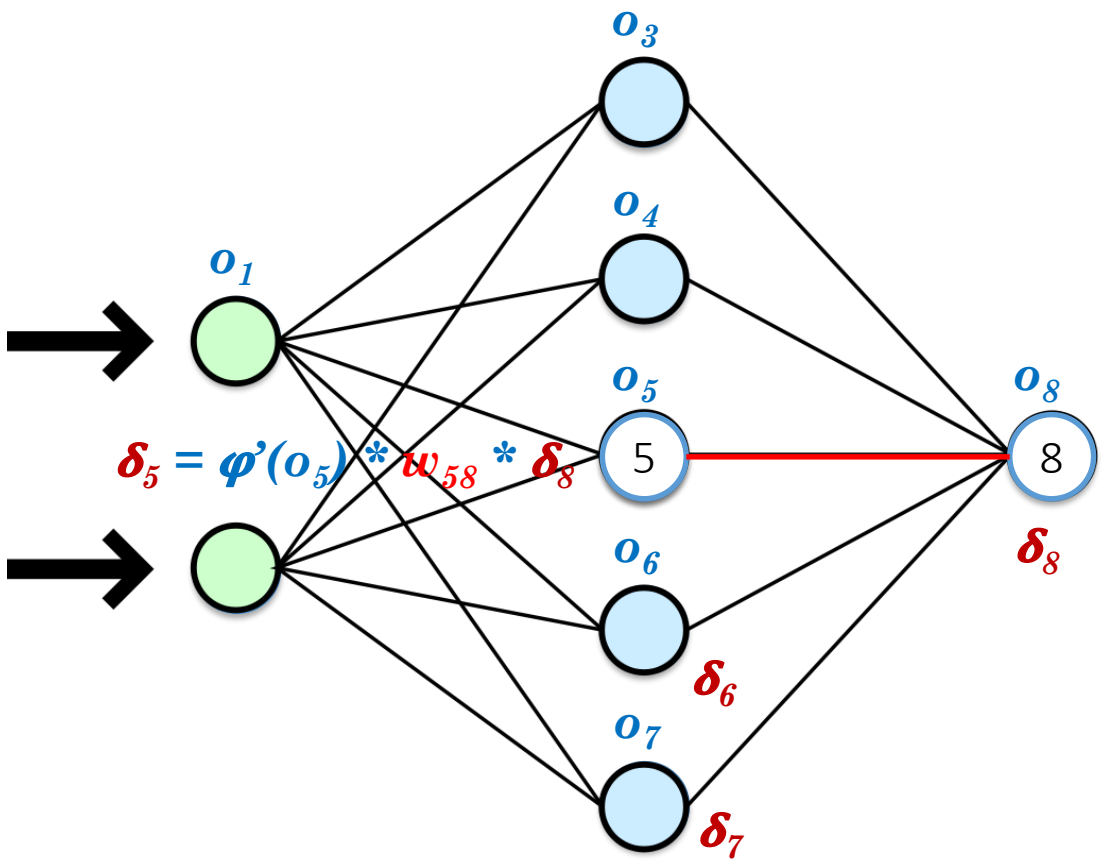


Bart.Lamiroy@univ-reims.fr

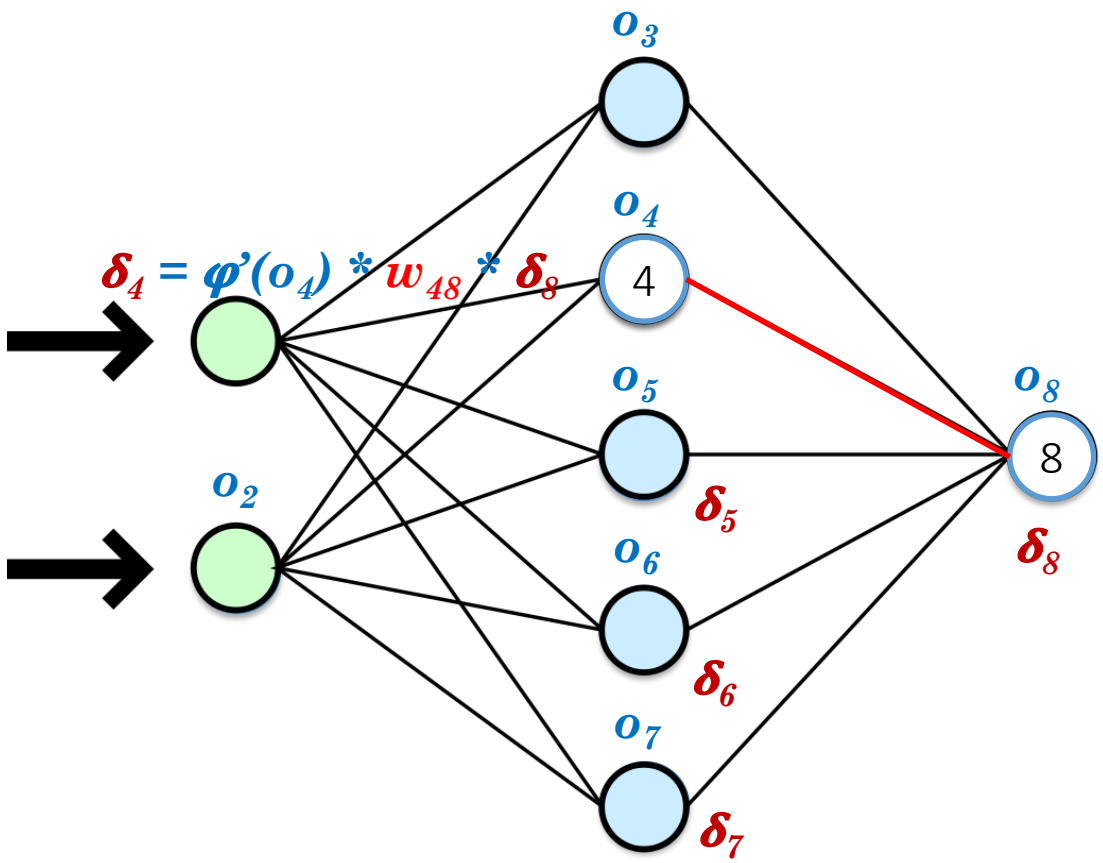
Phase de rétropropagation



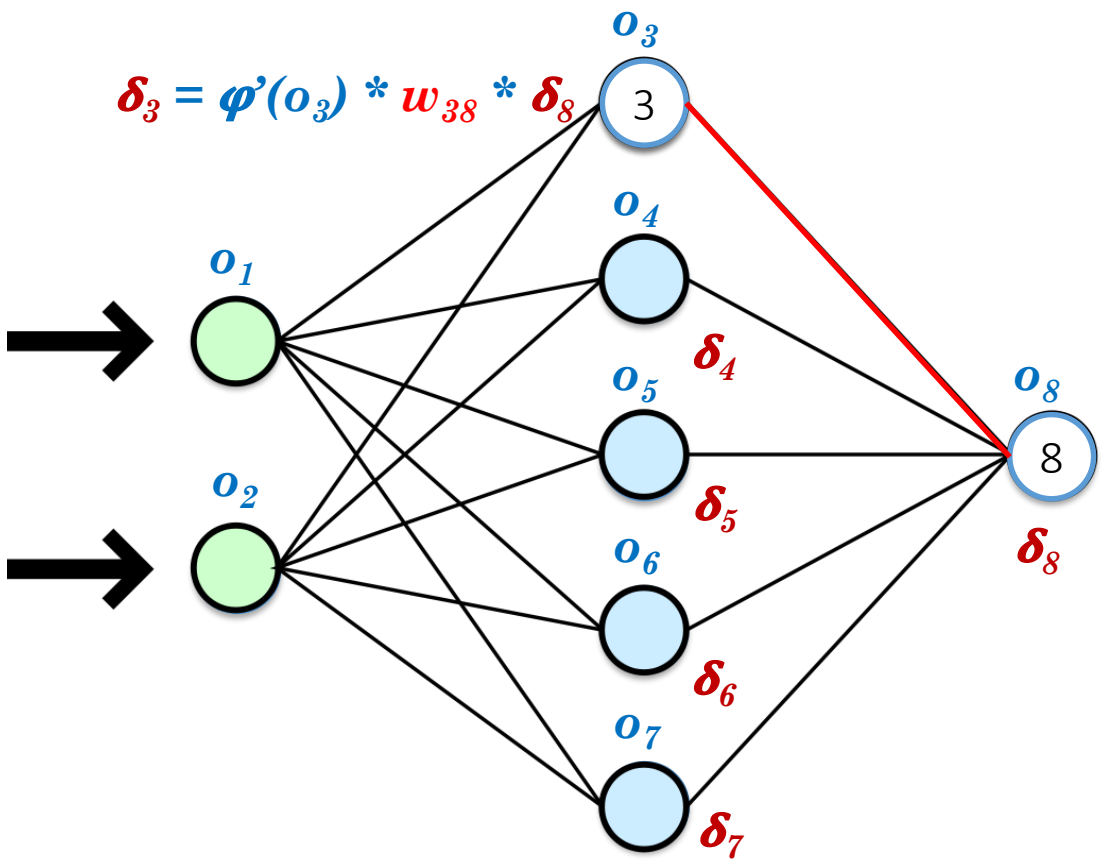
Phase de rétropropagation



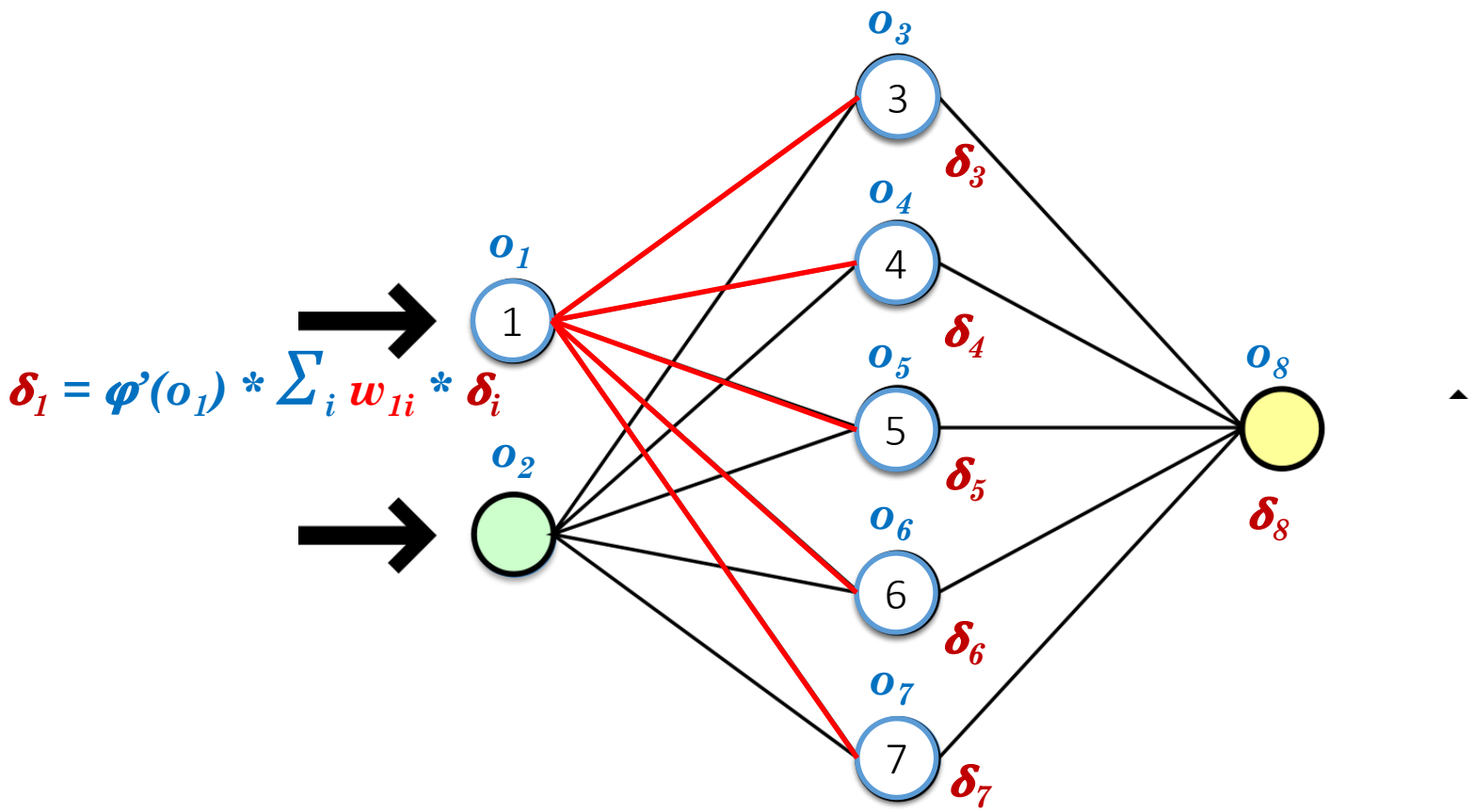
Phase de rétropropagation



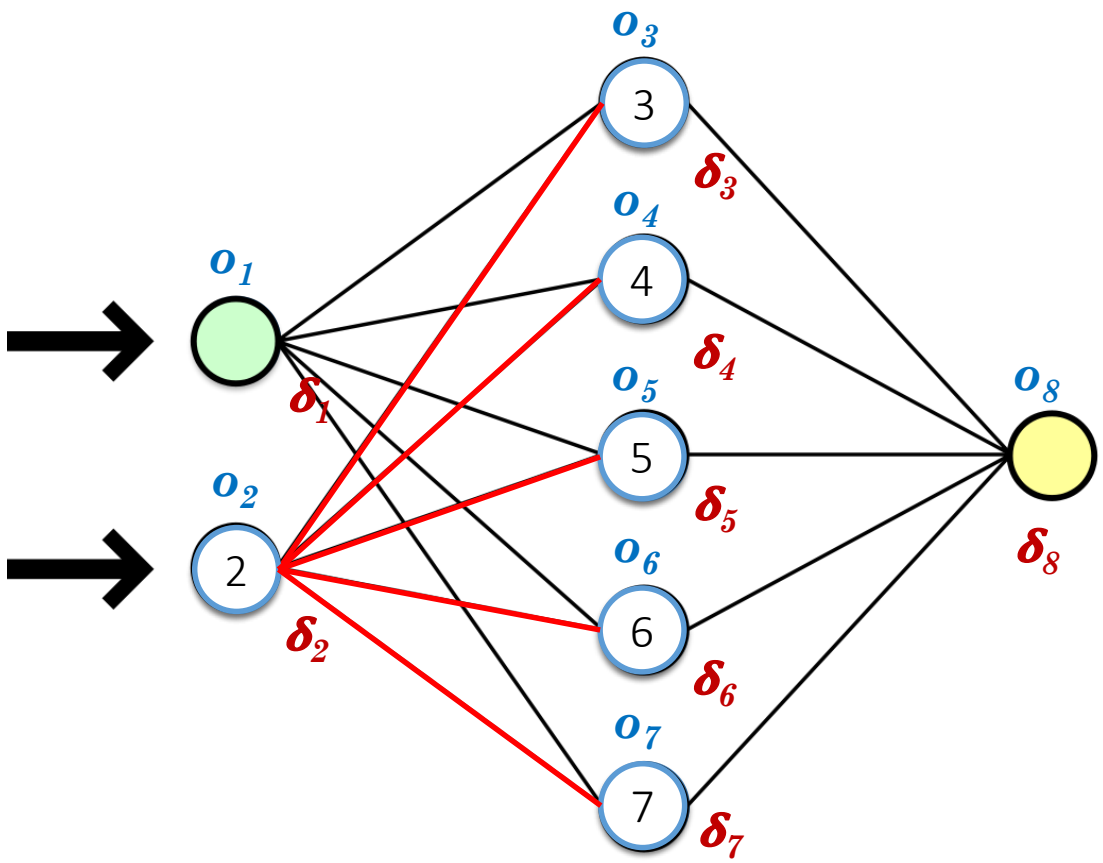
Phase de rétropropagation



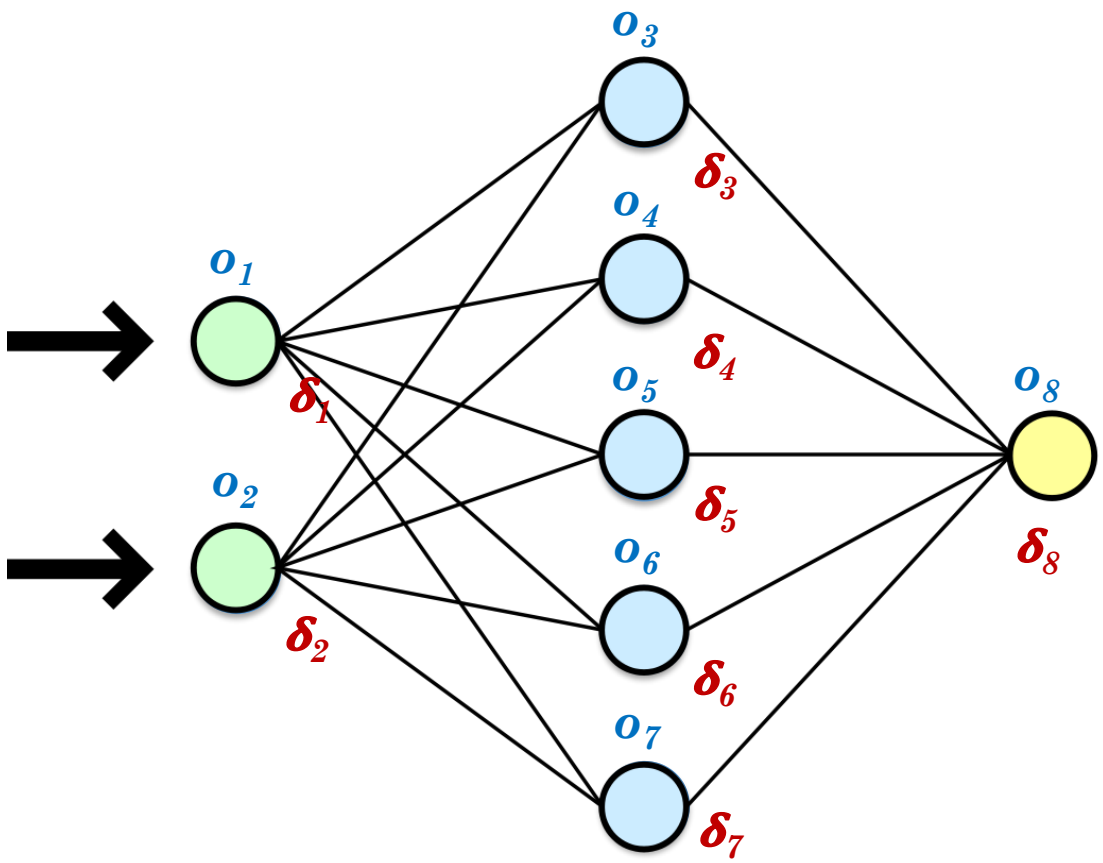
Phase de rétropropagation



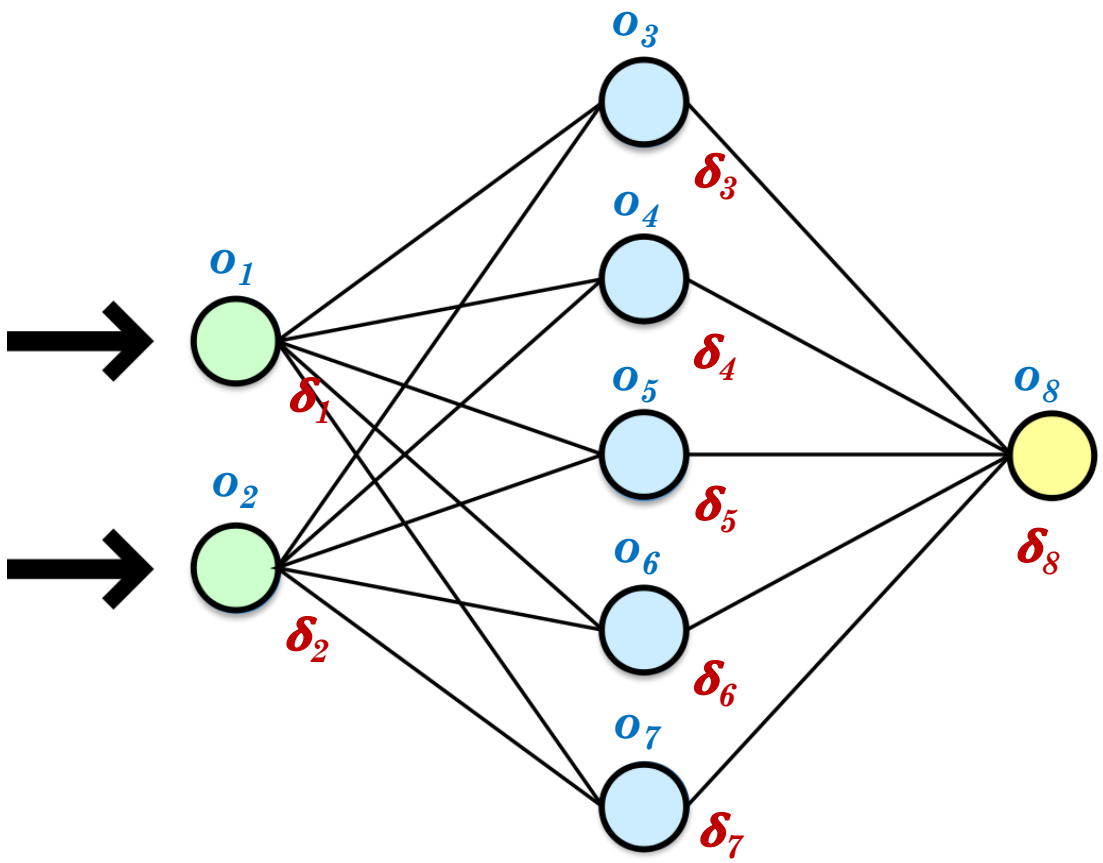
Phase de rétropropagation



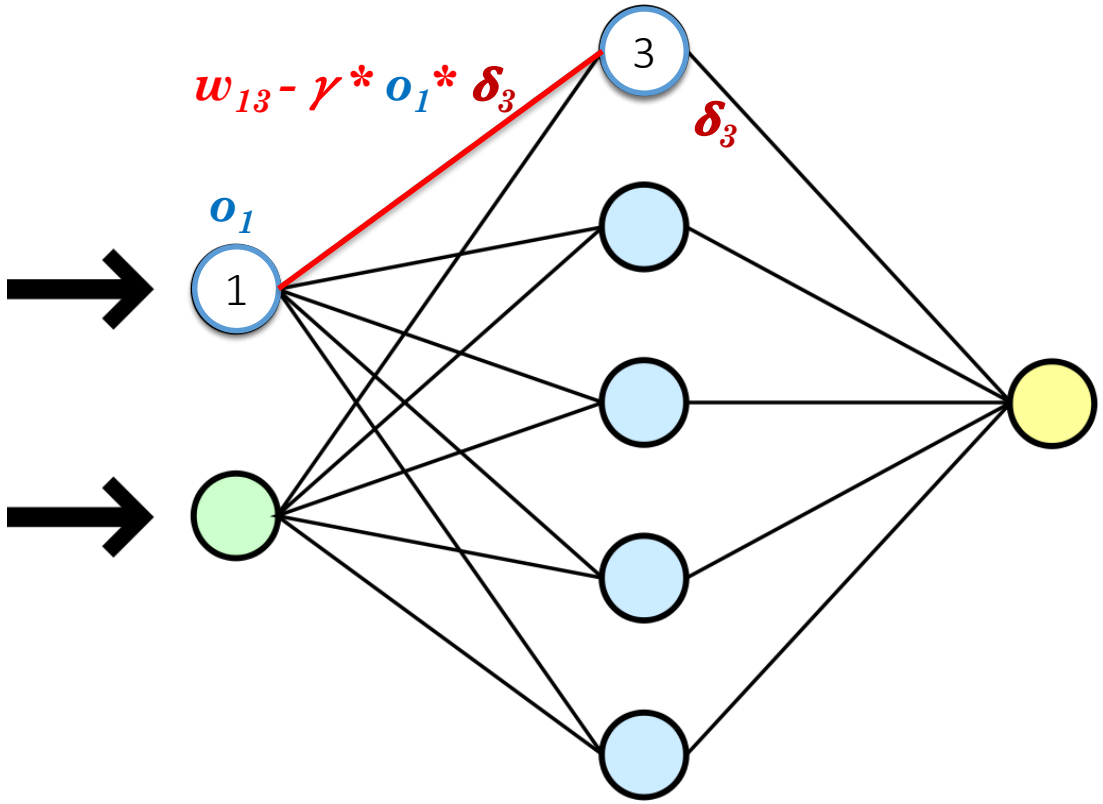
Résultat de la rétropropagation



Mise à jour des poids

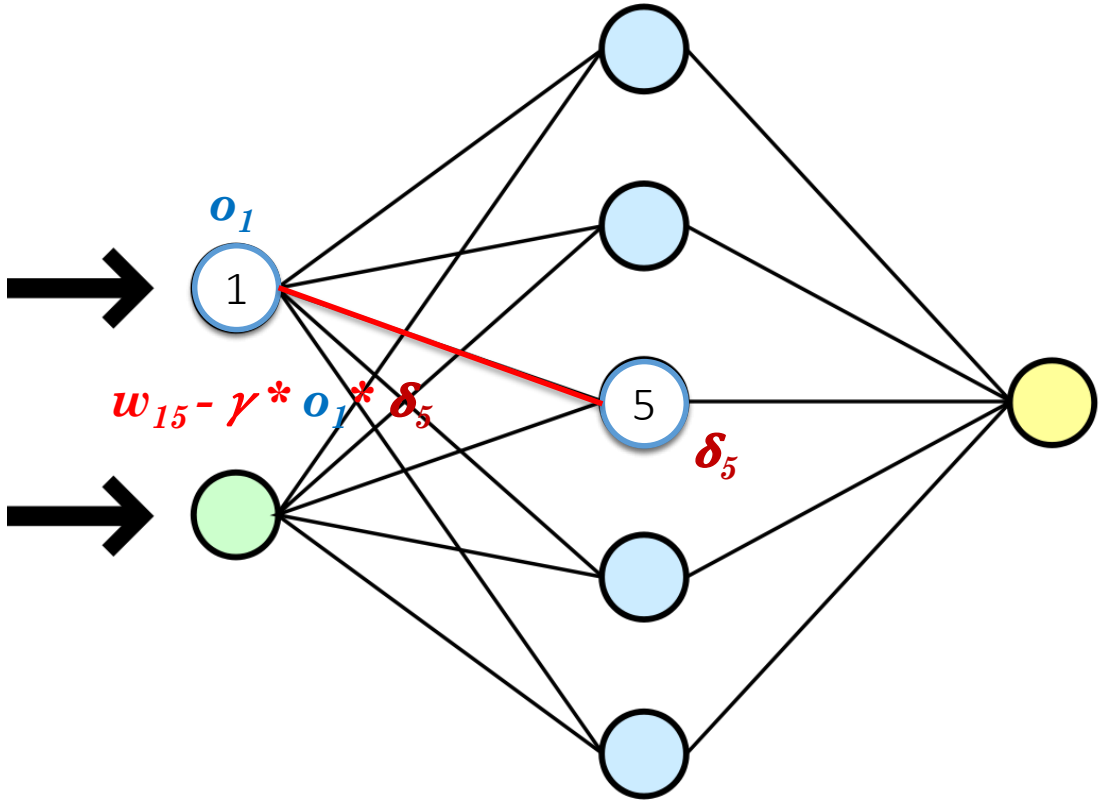


Mise à jour des poids



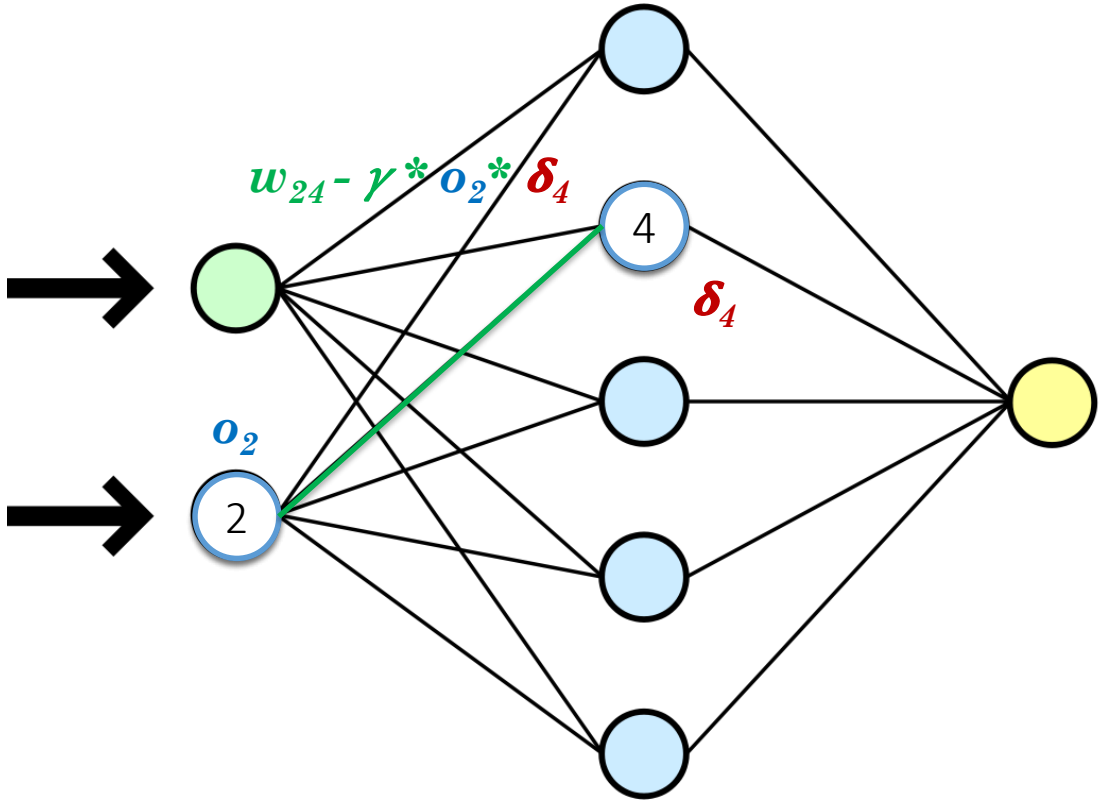
Bart.Lamiroy@univ-reims.fr

Mise à jour des poids



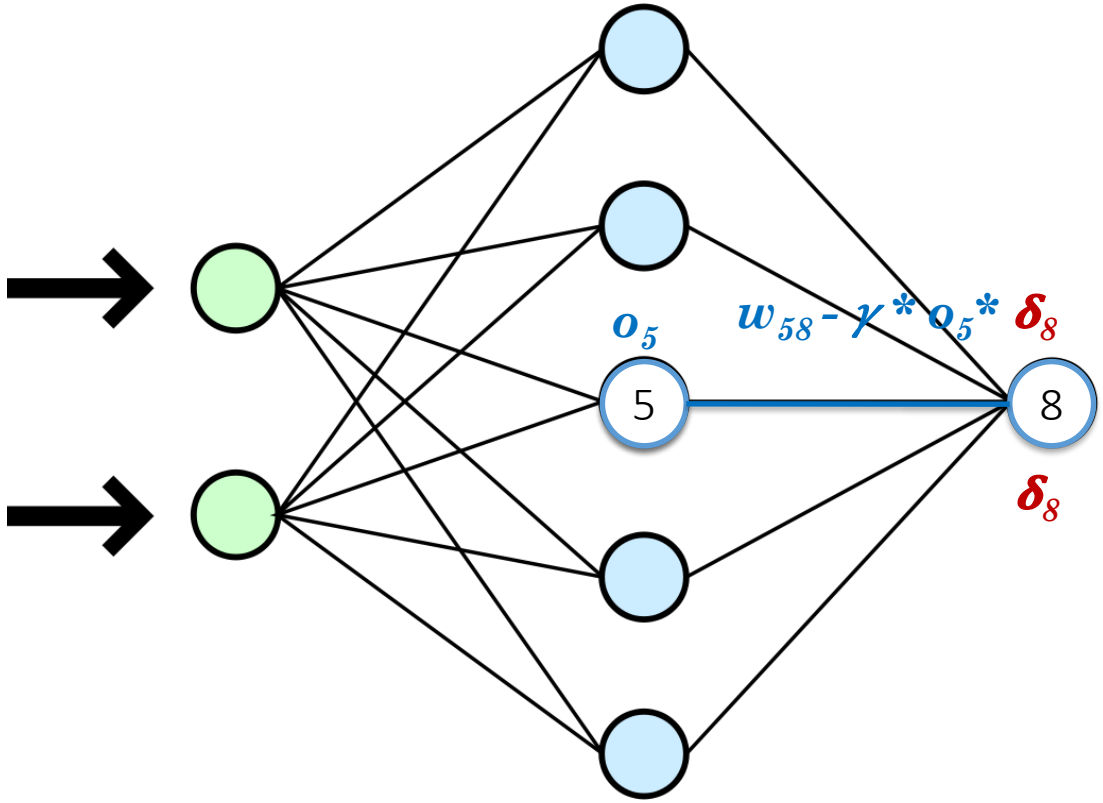
Bart.Lamiroy@univ-reims.fr

Mise à jour des poids



Bart.Lamiroy@univ-reims.fr

Mise à jour des poids



Bart.Lamiroy@univ-reims.fr

Conclusions intermédiaires

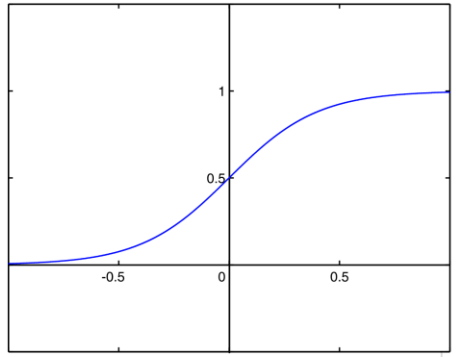
- Paradigme générique de “programmation” de fonctions
- Pas besoin de modèle, uniquement de données

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

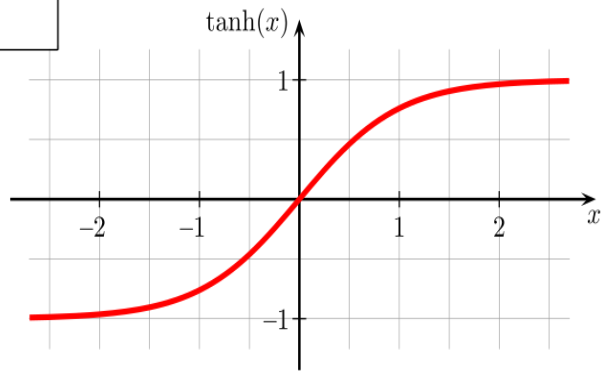
CC BY NC

Rapide focus sur les fonctions d’activation

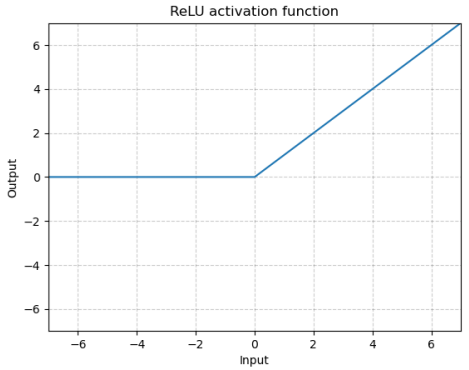
- Sigmoïde



- Tangente hyperbolique



- ReLU (Rectified Linear Unit)

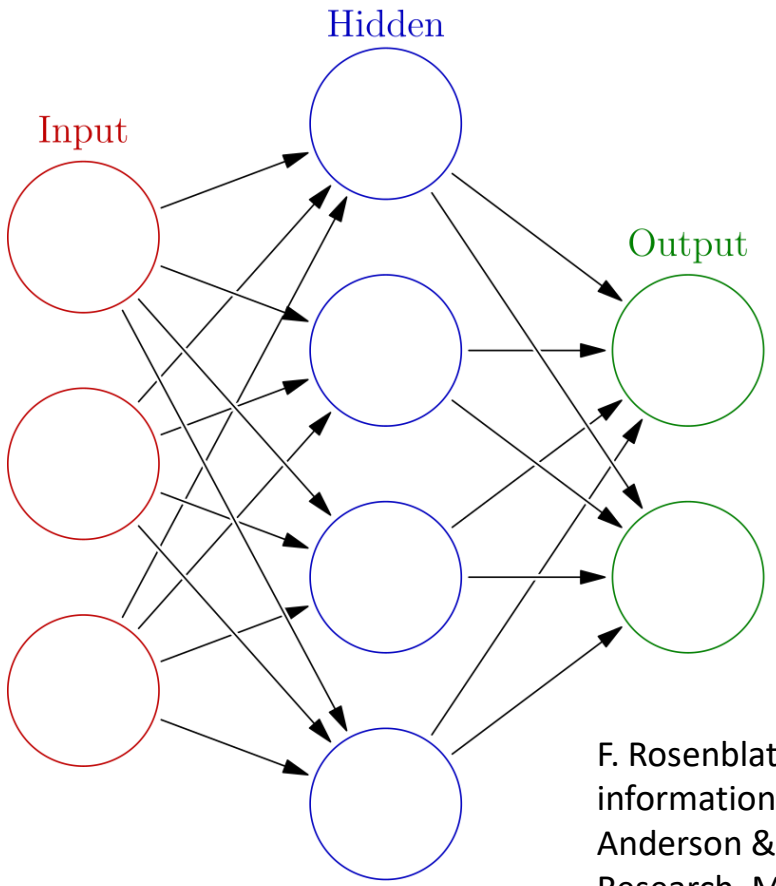


Types de réseaux

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

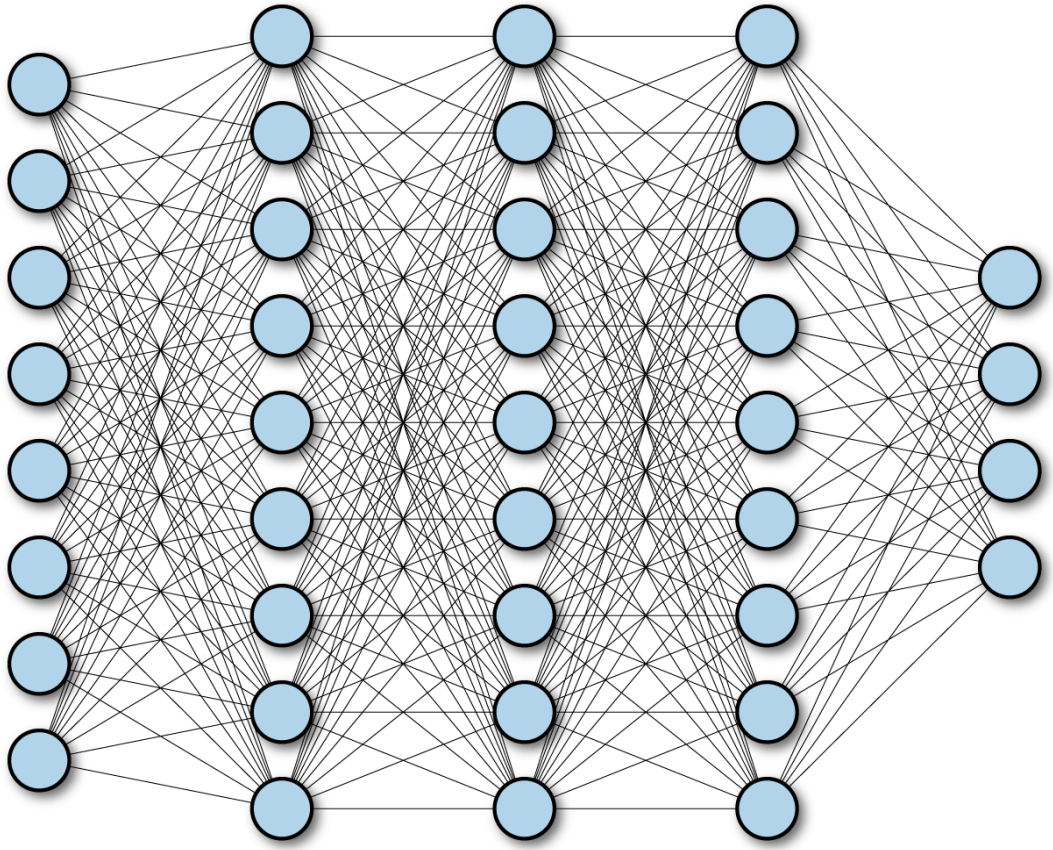
CC BY NC

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



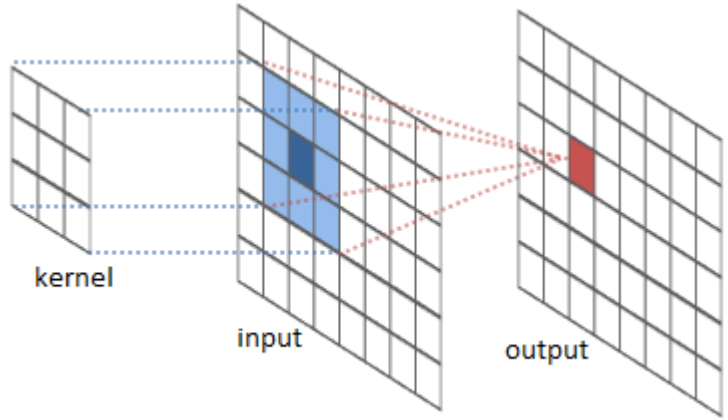
https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/assets/tfdl_0402.png

CC BY NC

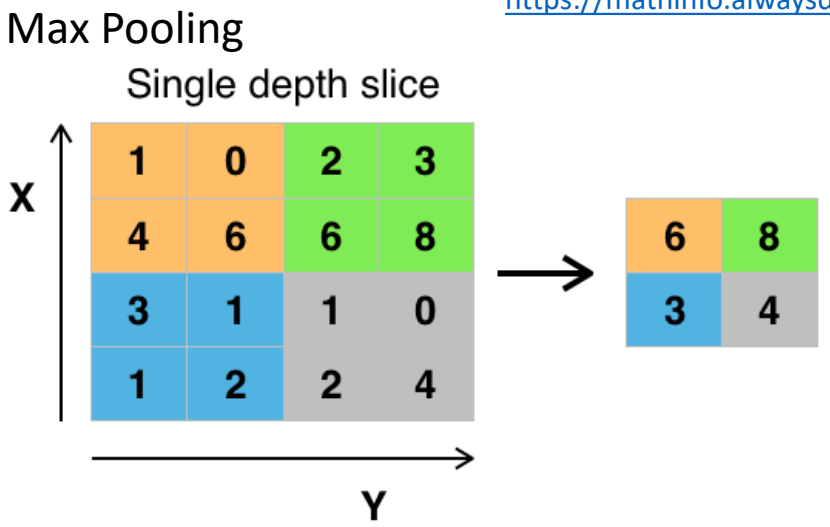
CNN – Convolutional Neural Network

- Convolution
- *Pooling* (sous-échantillonage)
- Aggrégation

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



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



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

CNN – Convolutional Neural Network

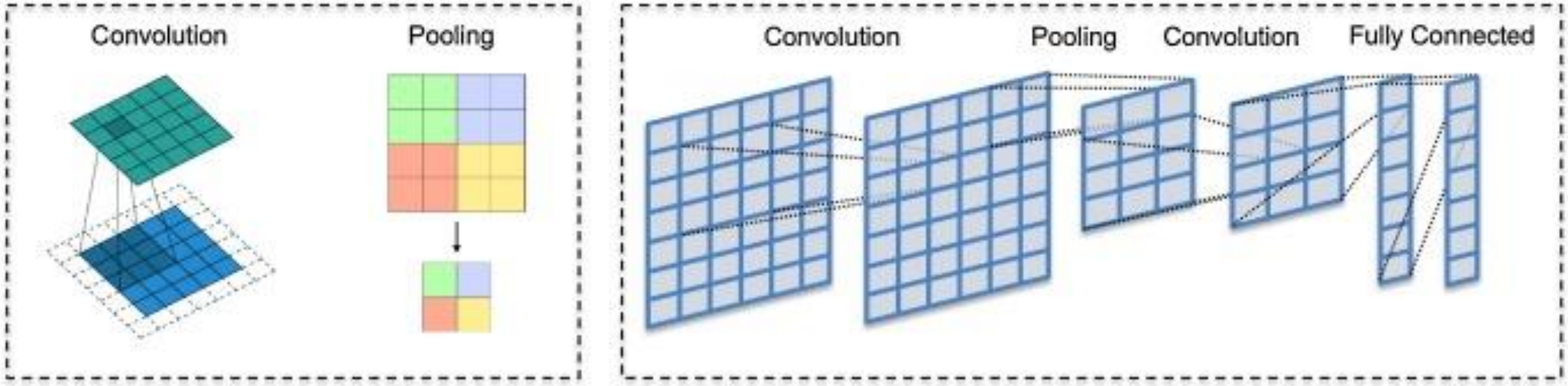
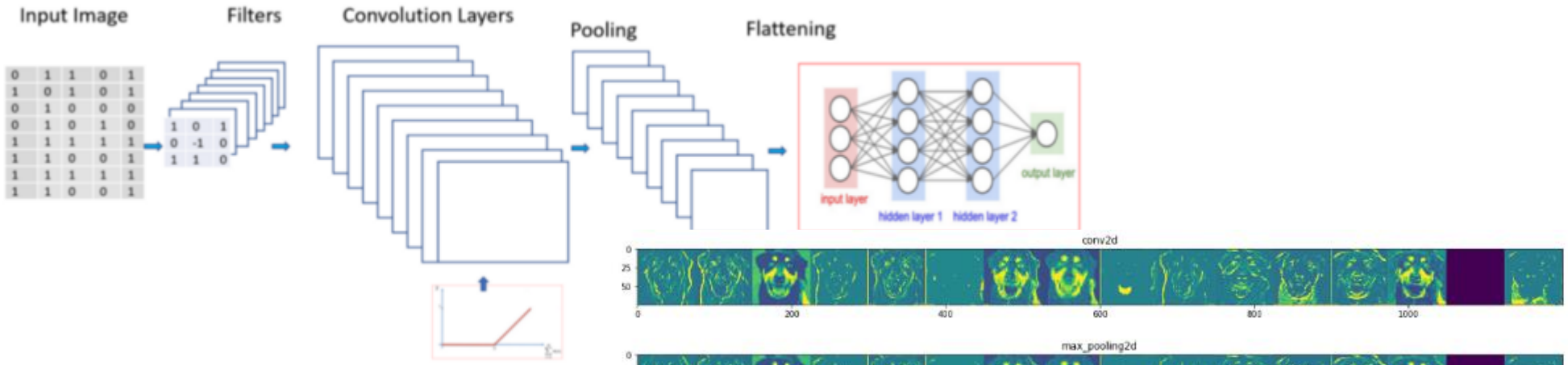
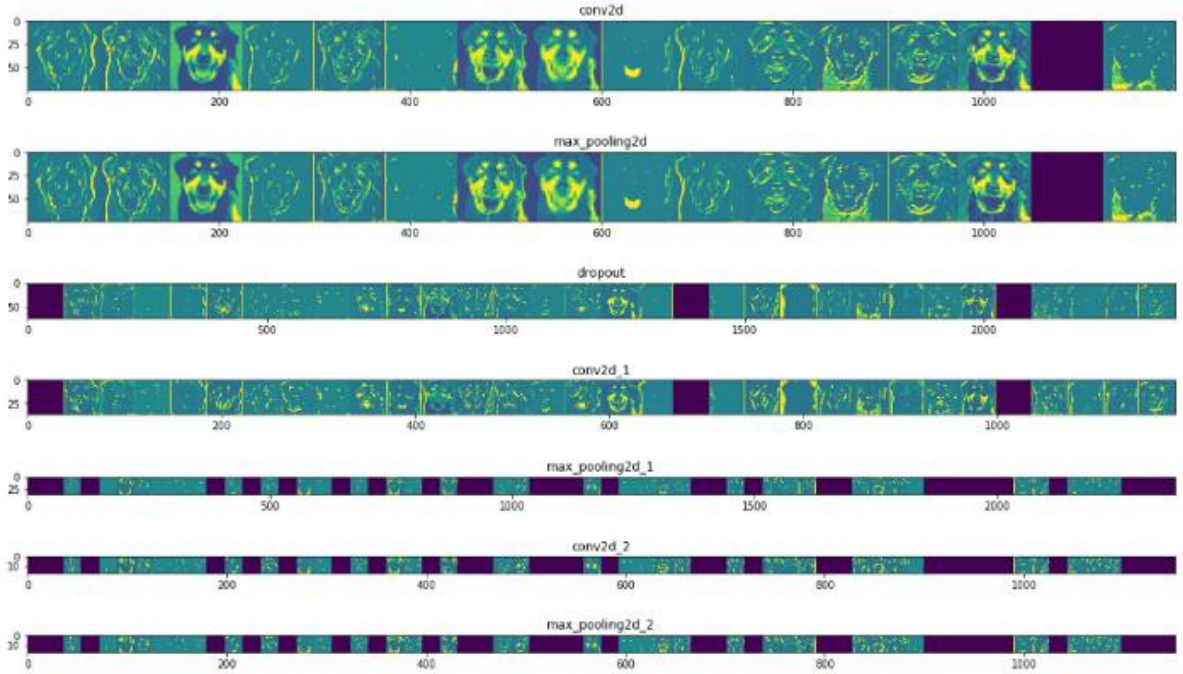


Figure 6 from Andreas Maier, Christopher Syben. Tobias Lasser. Christian Riess. "A gentle introduction to deep learning in medical image processing". Zeitschrift für Medizinische Physik Volume 29, Issue 2, May 2019, Pages 86-101.
https://www.sciencedirect.com/science/article/pii/S093938891830120X#fig0010

CNN end-to-end



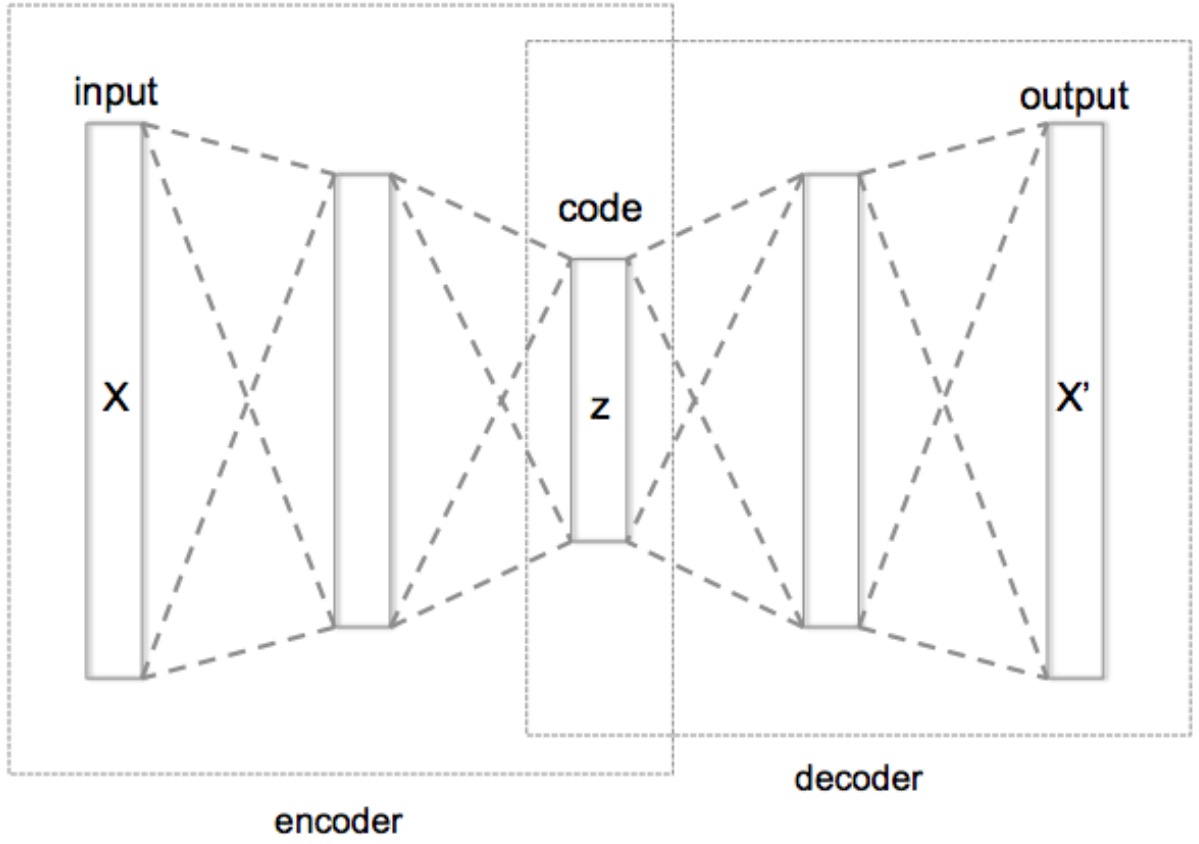


Source : https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization

CC BY NC

Autoencoder

Espace de représentation latent
« plongement » (*embedding*)



Par Chervinskii — Travail personnel, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45555552

Autoencoder Types

Regularization

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

Exemple d'utilisation des plongements :

`reine` = $(\alpha_0, \alpha_1 \ \ldots \alpha_n)$

`femme` = $(\beta_0, \beta_1 \ \ldots \beta_n)$

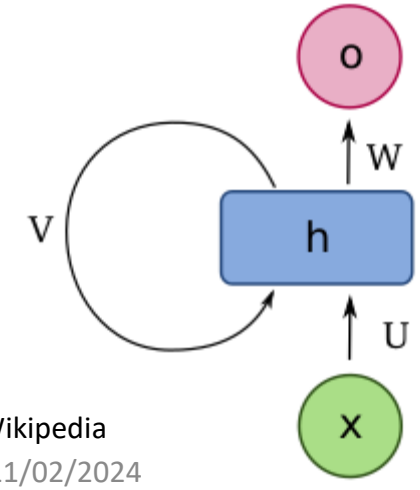
`bâtiment` = $(\gamma_0, \gamma_1 \ \ldots \gamma_n)$

`reine − femme + bâtiment = château`

CC BY NC

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**
- Principe d'une mémoire (*hidden state*) h_t qui dépend de l'entrée x_t au temps t et de la mémoire h_{t_1} au temps $t-1$

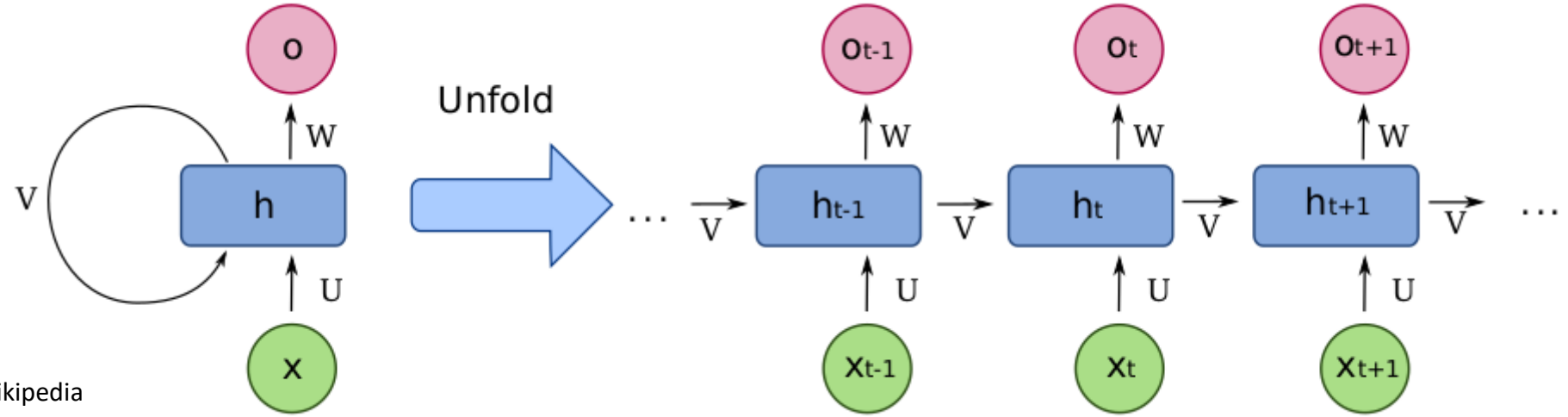


Source: Wikipedia

$$h_0 = [\,0, 0 \ldots 0\,]$$
$$h_t = V\, h_{t-1} + U\, x_t$$
$$o_t = W\, h_t$$

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
- Principe d'une mémoire (*hidden state*) h_t qui dépend de l'entrée x_t au temps t et de la mémoire h_{t-1} au temps t-1



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

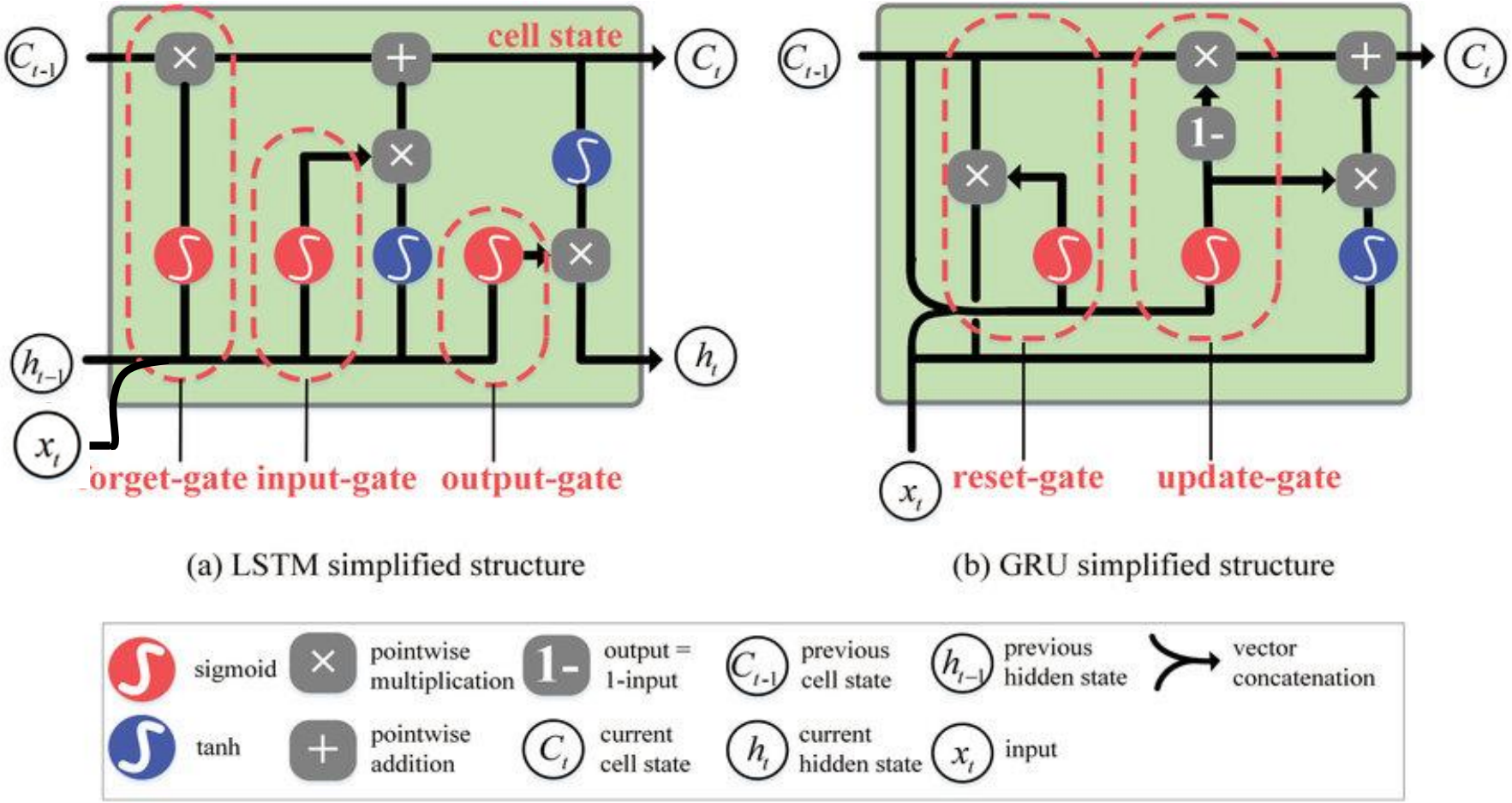
  Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, 2014. arXiv:1406.1078

- **CRNN** –Convolutional Recurrent Neural Networks

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

CC BY NC

LSTM *vs*. GRU

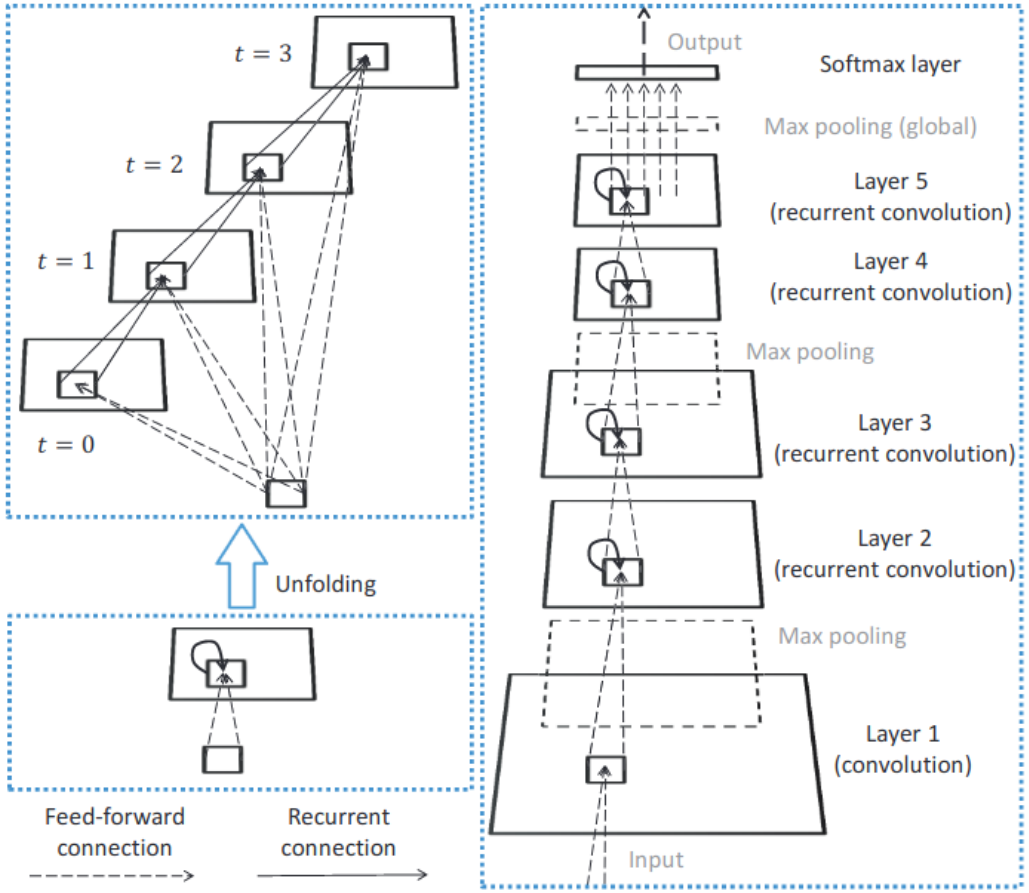


(a) LSTM simplified structure

(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

CC BY NC

ChatGPT
Transformers et mécanisme d'attention

© Philippe Geluck

CC BY 2.0 Deed
Erwin Soeputra machie from Singapore, Singapore

CC BY-SA 2.0
chuck624 from Upstate NY, USA

CC BY-SA 3.0 Deed license, Malene Thyssen, http://commons.wikimedia.org/wiki/User:Malene

Droits appartenant aux propriétaires respectifs

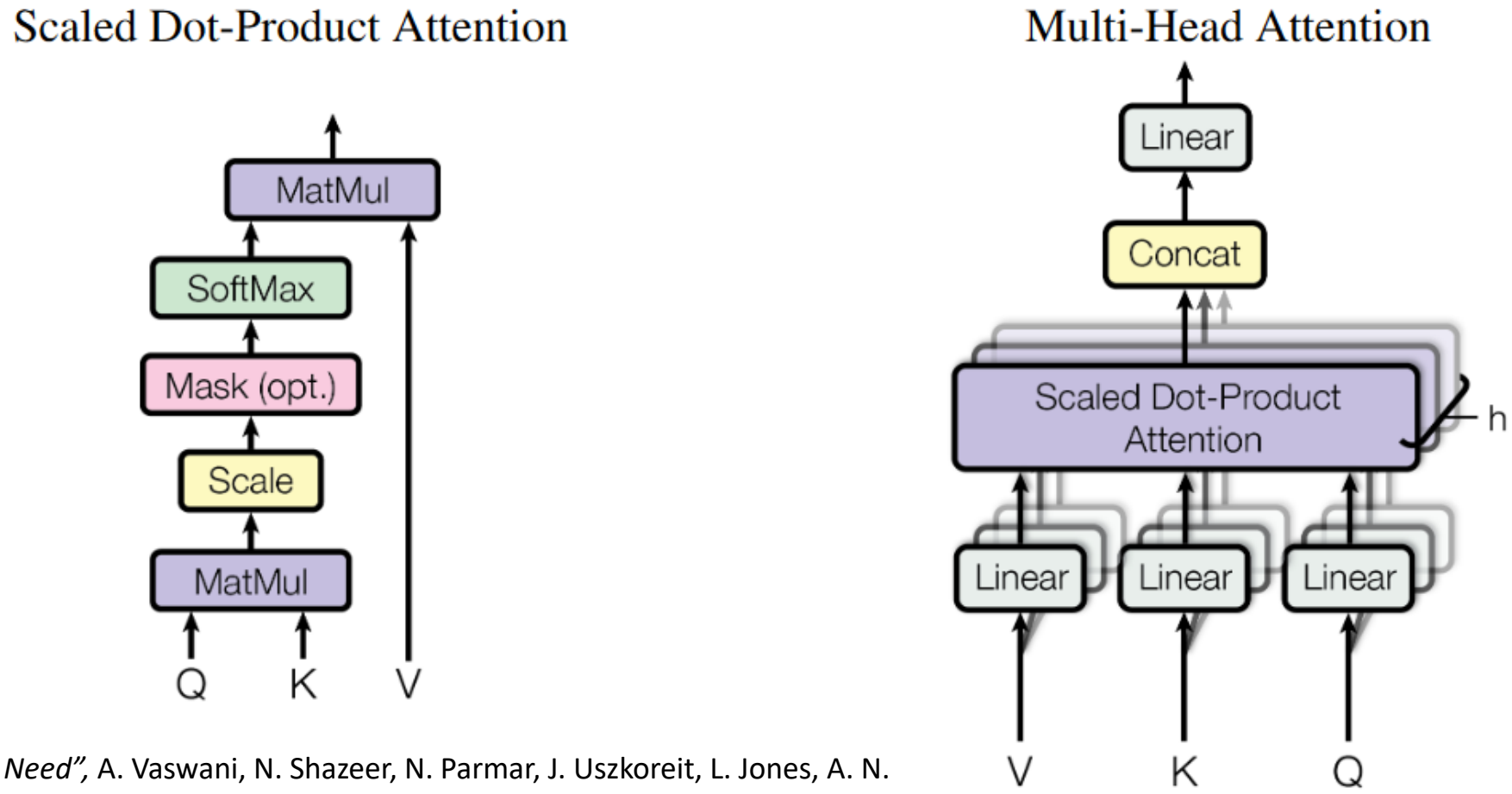
Pourquoi « transformers »

- Transformation de séquences d'entrée en séquences de sortie (*seq2seq*)
- Mécanisme d'auto-régression
  - Principe de régression type série temporelle (prédire le « futur » en observant le « passé »)
  - Enrichissement la séquence initiale (le « passé ») par les prédictions successives à chaque étape.

"Attention is all you Need", A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I Polosukhin
Advances in Neural Information Processing Systems 30 (NIPS 2017)
https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

CC BY NC

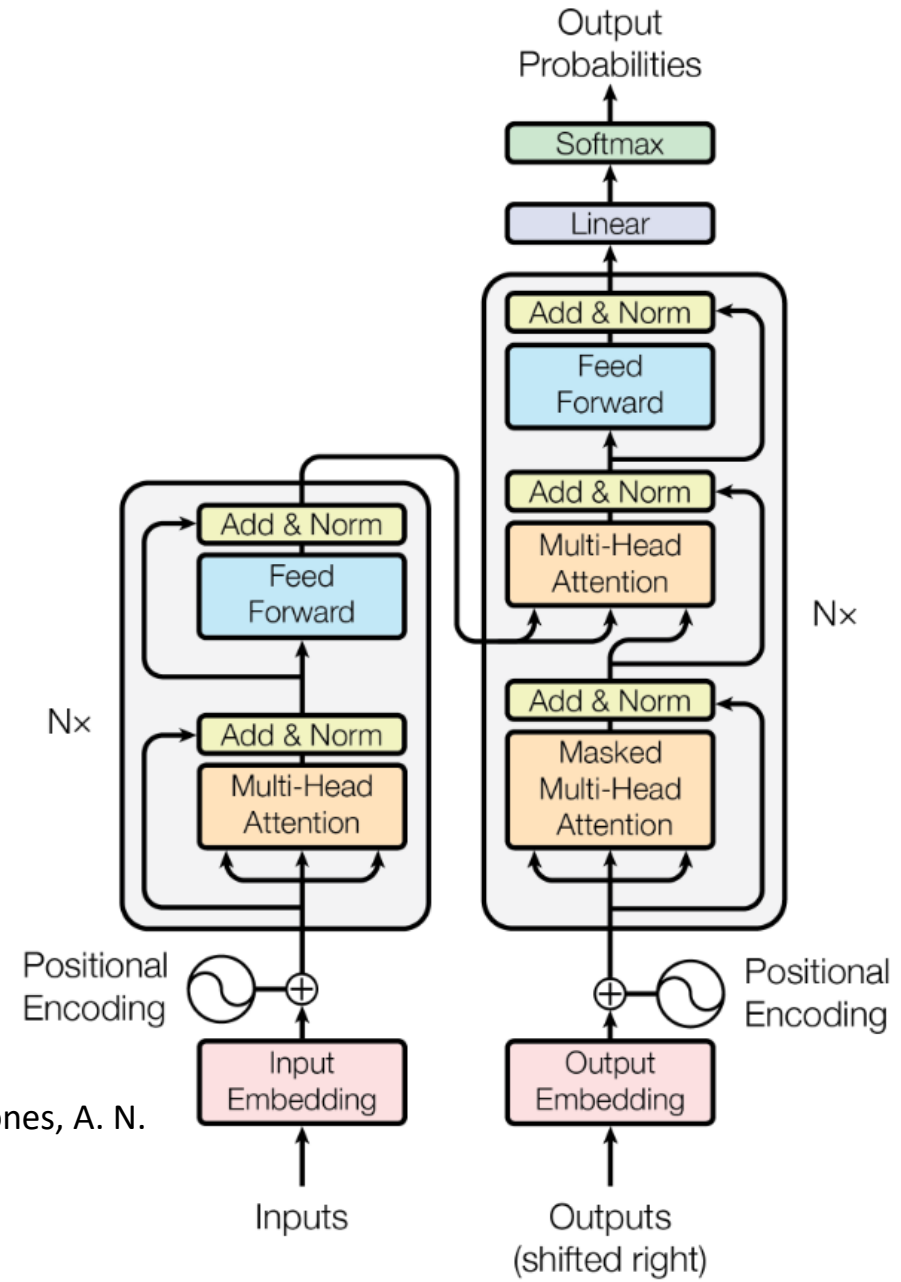
Mécanisme d'attention



"Attention is all you Need", A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I Polosukhin
Advances in Neural Information Processing Systems 30 (NIPS 2017)

CC BY NC

Transformer



"Attention is all you Need", A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I Polosukhin
Advances in Neural Information Processing Systems 30 (NIPS 2017)

Résultat : LLM

- LLM = Large Language Model
- Apprentissage auto-supervisé
- Principe de base : seq2seq autorégressif
  - Entrée = séquence
  - Sortie = le mot le plus probable qui complète la séquence
- Perroquet stochastique

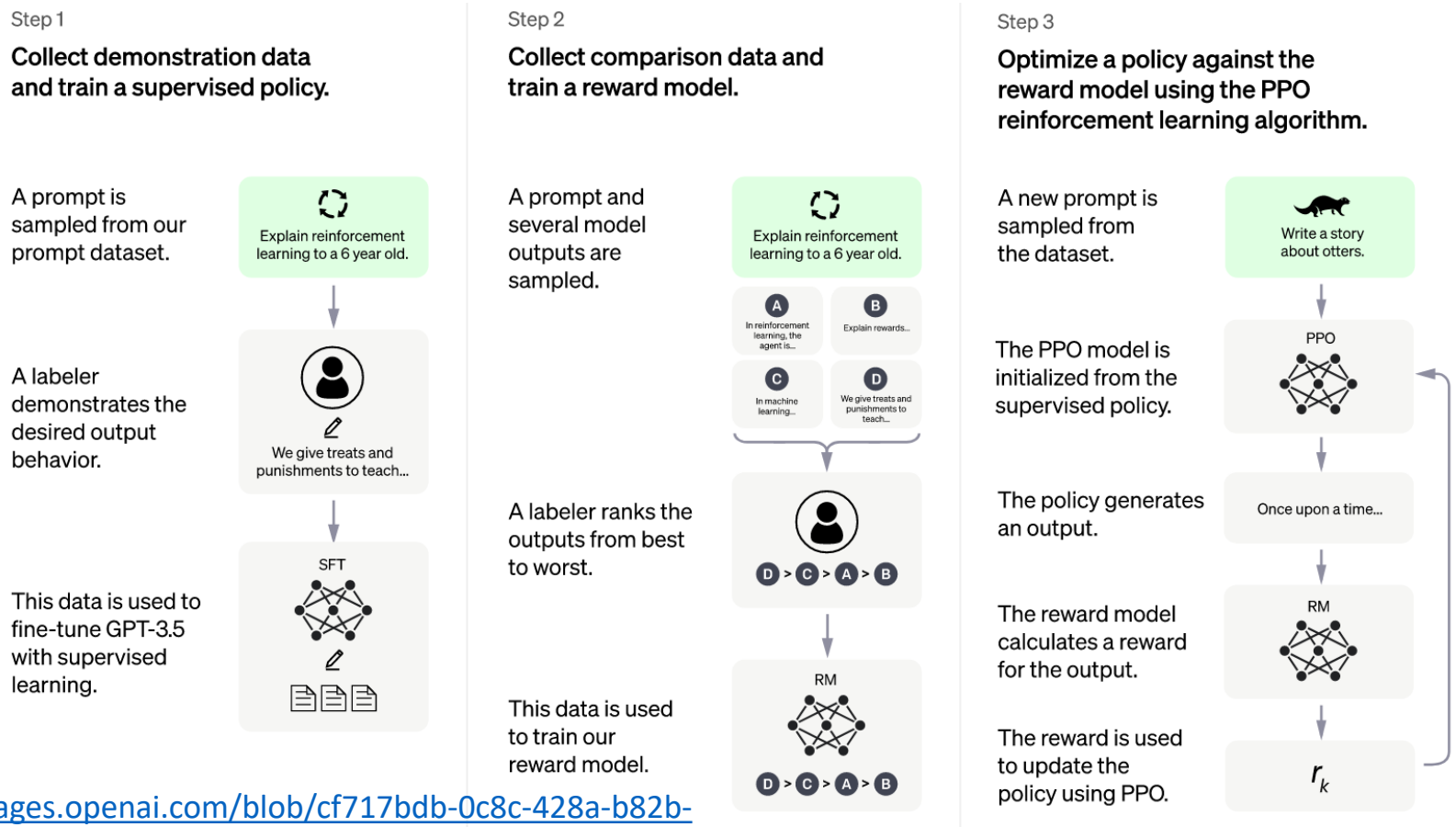
  « *On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?* 🦜 »

  E. M. Bender, T. Gebru, A. McMillan-Major, M. Mitchell, FAccT '21: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, March 2021, Pages 610–623

  https://dl.acm.org/doi/10.1145/3442188.3445922

CC BY NC

ChatGPT : Entrainement humain



Source : https://images.openai.com/blob/cf717bdb-0c8c-428a-b82b-3c3add87a600/ChatGPT_Diagram.svg?width=10&height=10&quality=50

CC BY NC

Métrique chinchilla

$$\begin{cases} C = C_0 N D \\ L = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + L_0 \end{cases}$$

where the variables are

- C is the cost of training the model, in FLOPs.
- N is the number of parameters in the model.
- D is the number of tokens in the training set.
- L is the average negative log-likelihood loss per token (nats/token), achieved by the trained LLM on the test dataset.
  - L_0 represents the loss of an ideal generative process on the test data
  - $\frac{A}{N^\alpha}$ captures the fact that a Transformer language model with N parameters underperforms the ideal generative process
  - $\frac{B}{D^\beta}$ captures the fact that the model trained on D tokens underperforms the ideal generative process

A retenir (pour des LLM)

On peut atteindre la même performance (**L constant**) sur un modèle plus complexe (***augmenter N d'un facteur α***) à condition d'augmenter la quantité de données d'entraînement de la même proportion (**augmenter D d'un facteur α**) et d'augmenter le coût de calcul façon quadratique (**augmenter C d'un facteur α^2**).

Training Compute-Optimal Large Language Models
Jordan Hoffmann et al. DeepMind
https://arxiv.org/pdf/2203.15556.pdf

CC BY NC

On pourrait continuer longtemps …

- *Hallucinations as a feature*

  “One of the sort of non-obvious things is that a lot of value from these systems is heavily related to the fact that they do hallucinate. If you want to look something up in a database, we already have good stuff for that.”

  — Sam Altman, CEO OpenAI, Dreamforce 2023, San Francisco
- Modèles multi-langues
- Modèles multi-modaux
- Modèles par diffusion

…

CC BY NC

… très longtemps

- L'important est-ce l'algorithme ou la data ?
- Comment évaluer les algorithmes
- Comment corriger les algorithmes

…

CC BY NC

Source : http://www.signwithrobert.com/

Questions ?

CC BY NC