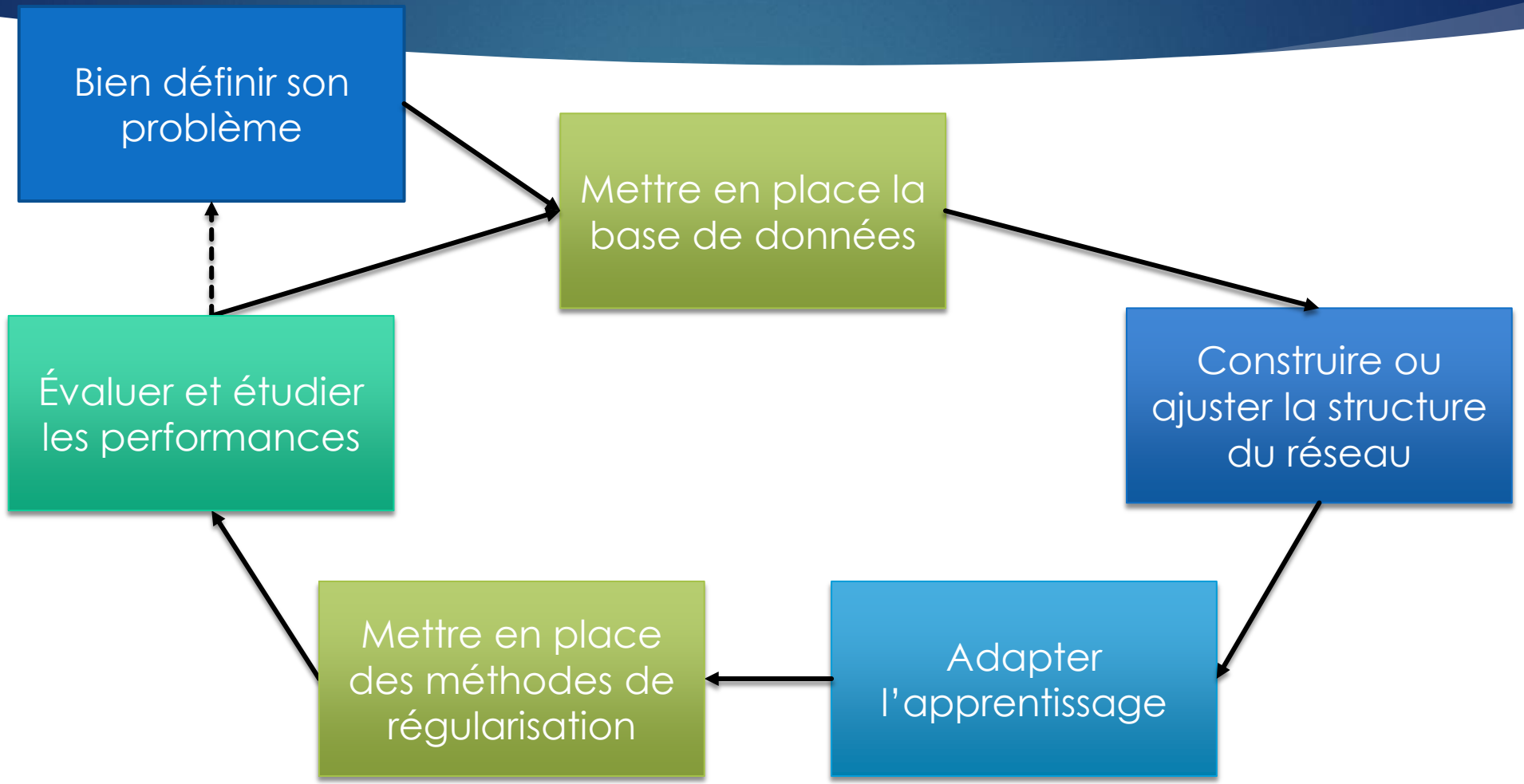
Réseaux de neurones et deep learning : Utilisation et méthodologie

GEOFFREY DANIEL – CEA/IRFU/DAP

Previously


Bien définir son problème
Mettre en place la base de données
Construire ou ajuster la structure du réseau
Adapter l'apprentissage
Mettre en place des méthodes de régularisation
Évaluer et étudier les performances

Réseaux de neurones et deep learning

À partir de mi-octobre

Séances de 30/40 mn suivies de 20 mn de discussion

- Séance 1 : Introduction aux réseaux de neurones
  - Utilisations courantes du deep learning
  - Bases générales
- Séance 2 : Architecture des réseaux, hyperparamètres et évaluation des performances
  - Comment construire mon réseau et adapter la phase d'apprentissage ?
  - Comment évaluer les performances de mon réseau de neurones ?
- Séance 3 : Construction de la base de données
  - Éléments méthodologiques sur la mise en place du problème à résoudre potentiellement par deep learning
  - Comment utiliser l'évaluation des performances pour améliorer la base de données et le réseau ?
- **Séance 4 : Réseaux de neurones convolutifs**
  - **Introduction à des structures plus avancées**

Les réseaux de neurones convolutifs (CNN)

- Motivations sur les images :
  - Corrélation locale des données
  - Détection de formes
  - Invariance par translation : les formes peuvent se trouver à différents endroits de l'image

Les réseaux de neurones convolutifs (CNN)

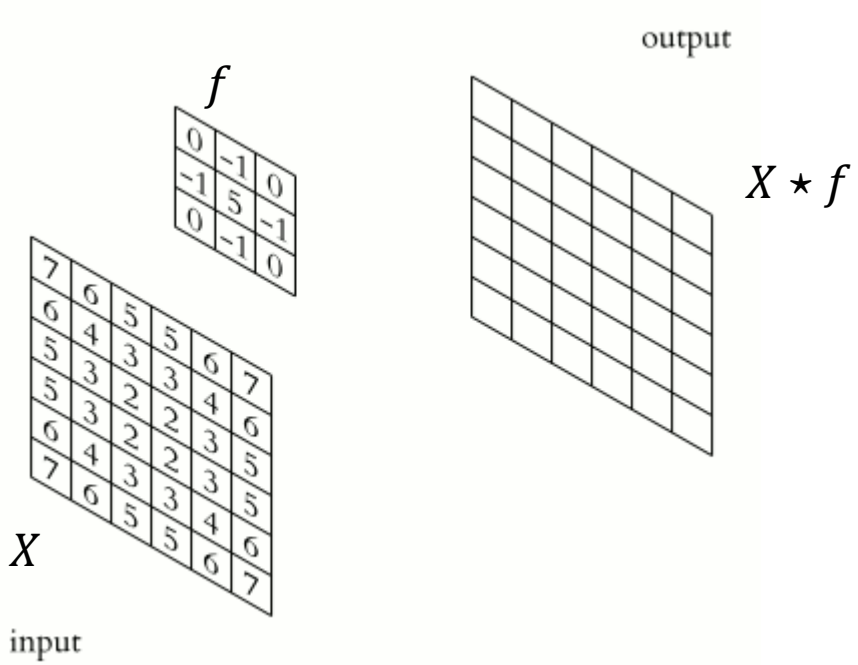
- L'opération de convolution :
  - On note X un tableau $n \times m$, f un filtre $n_f \times m_f$ avec $n_f \leq n$, $m_f \leq m$, la convolution de X par f pour un pixel de coordonnées (i, j), $i \in [\![0, n-1]\!], j \in [\![0, m-1]\!]$ est définie par :

$$(X \star f)[i,j] = \sum_{k=-\frac{n_f}{2}}^{\frac{n_f}{2}} \sum_{l=-\frac{m_f}{2}}^{\frac{m_f}{2}} X[i+k, j+l] \times f\left[k + \frac{n_f}{2}, l + \frac{m_f}{2}\right]$$

  - Dans le cas ci-dessus, si $i+k$ ou $j+l$ sort de l'image X (c'est-à-dire, $i+k \geq n$ par exemple) alors on prend $X[i+k, j+l] = 0$, on parle de « zéro-padding ». D'autres padding sont possibles (en reprenant la valeur du pixel le plus proche par exemple).
  - On peut aussi choisir de ne faire le calcul que sur les pixels (i,j) tels que $i+k$ et $j+l$ sont toujours dans l'image : il y aura alors réduction de la dimension de sortie

Les réseaux de neurones convolutifs (CNN)

- Plus visuellement :
- Le filtre ou noyau f est appelé filter ou kernel en anglais



Michael Plotke, https://commons.wikimedia.org

Les réseaux de neurones convolutifs (CNN)

- Effet des convolutions, exemple sur des images :

Deep Learning Methods for Vision, CVPR 2012 Tutorial

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Wikipedia article on Kernel (image processing)

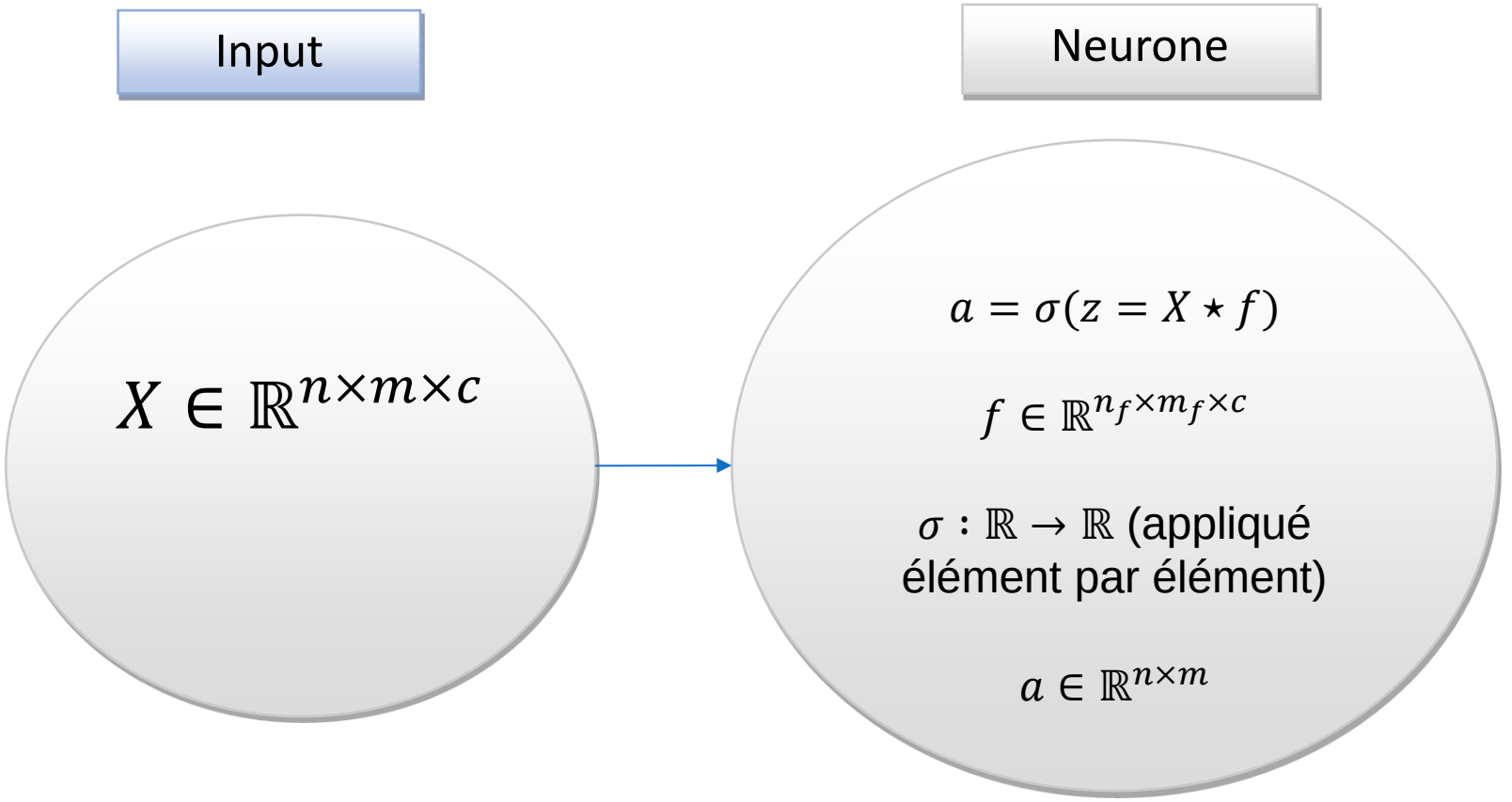
Les réseaux de neurones convolutifs (CNN)

- Un neurone de convolution :

On ne choisit pas les **filtres**, on les apprend !!! Ce sont des **paramètres entrainables** du réseau

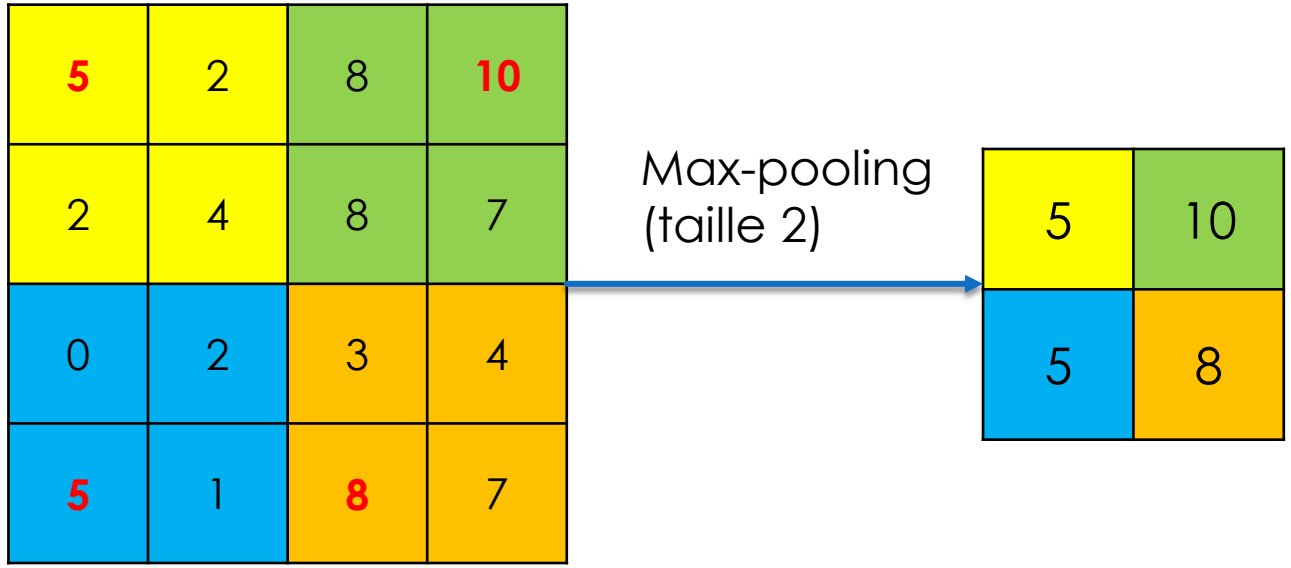
c s'appelle nombre de channels ou de canaux :

- $c = 1$ pour une image en niveaux de gris
- $c = 3$ pour une image RGB (une channel par couleur)
- Dans les couches intermédiaires, c correspond au nombre de neurones de la couche précédente



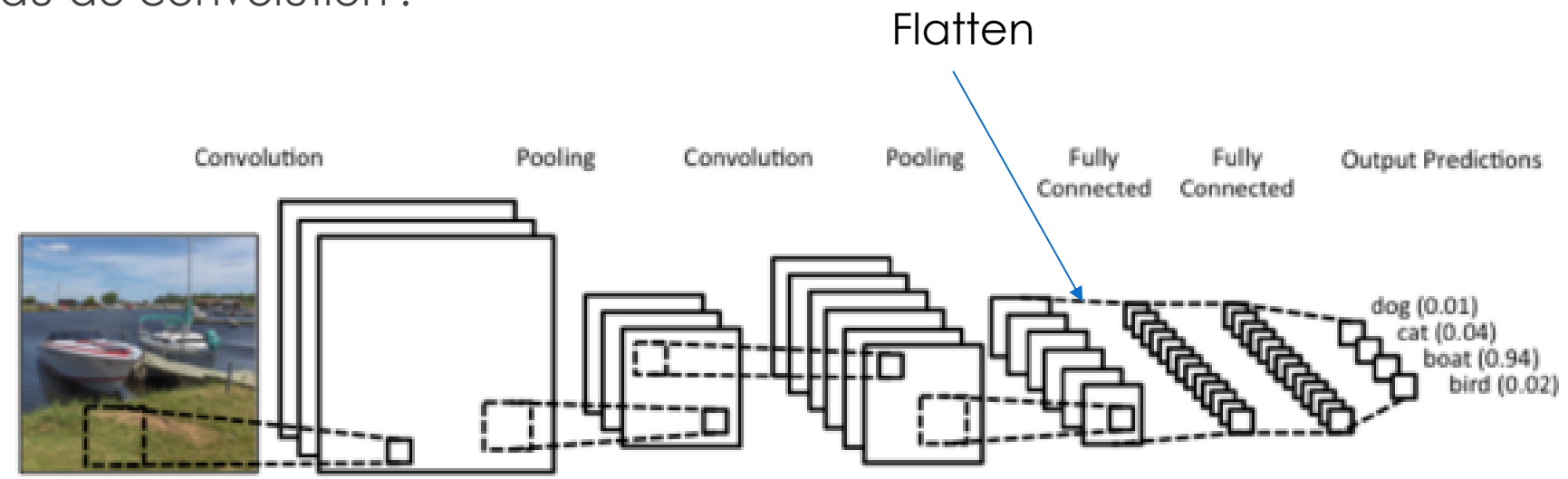
Les réseaux de neurones convolutifs (CNN)

- Le Pooling : opération utilisée pour réduire la dimension
  - Recherche de détails plus « grossiers », de plus grandes « structures » dans l'image
  - Max-pooling de taille l : on prend l'élément maximal de chaque sous-tableau de taille l
  - Sum-pooling de taille l : on fait la somme de tous les éléments de chaque sous tableau de taille l



Les réseaux de neurones convolutifs (CNN)

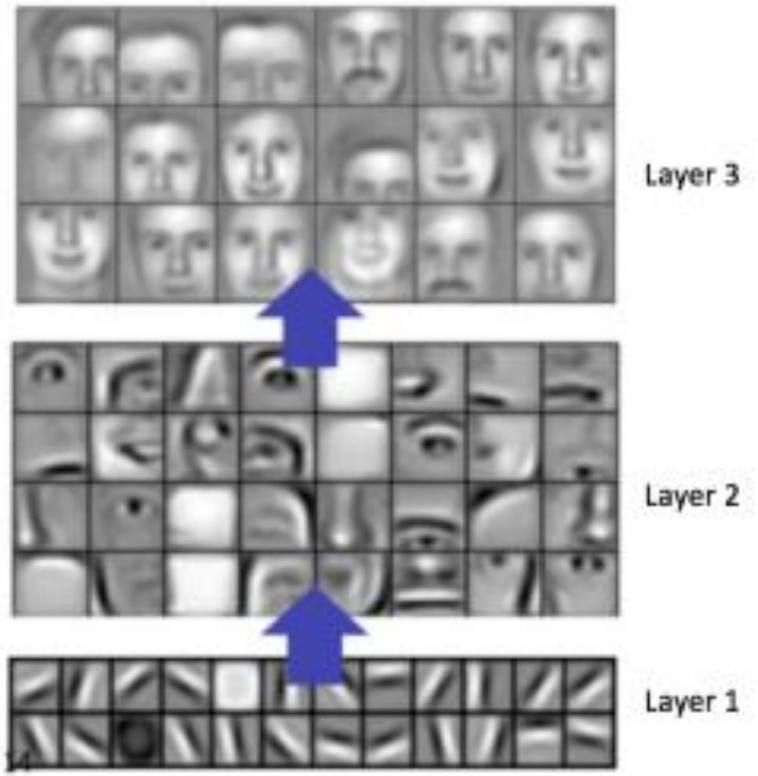
- Un réseau de convolution :



https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

Les réseaux de neurones convolutifs (CNN)

- Les filtres apprennent des formes de plus en plus complexes, globales



Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, Lee et al., Proceedings of the 26th International Conference on Machine Learning, 2009

Les réseaux de neurones convolutifs (CNN)

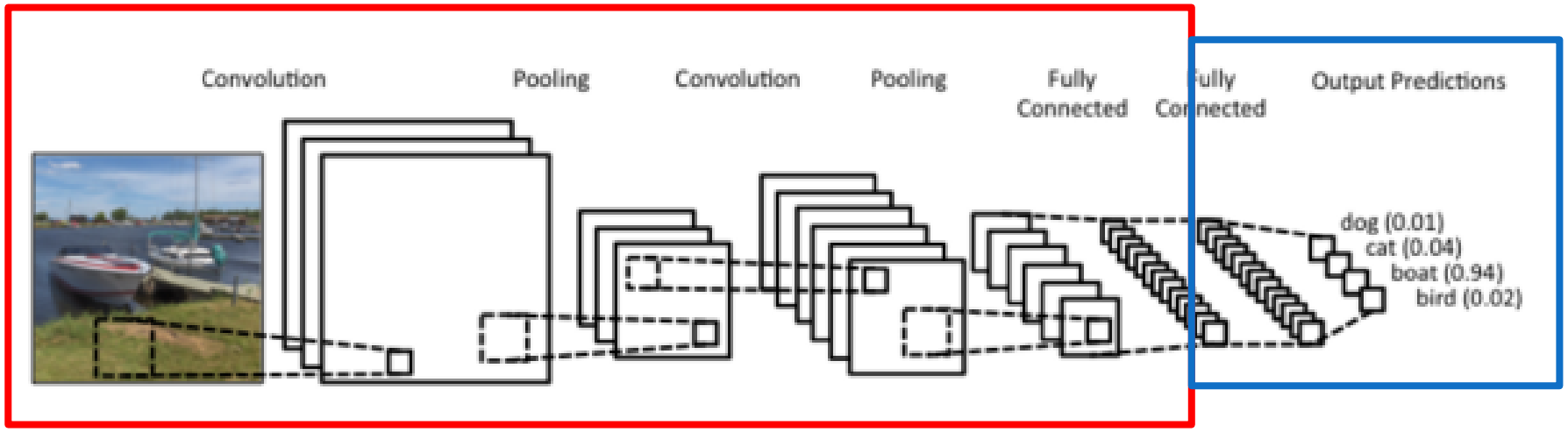
- Sous Keras (en **gras : ce qui est nouveau,** *en italique, ce qui est modifiable*) :
  - keras.layers.Conv2D(***filters***, ***kernel_size***, **strides=*(1, 1)***, **padding='*valid*'**, activation=*None*, use_bias=*True*, kernel_initializer='*glorot_uniform*', bias_initializer='*zeros*', kernel_regularizer=*None*, bias_regularizer=*None*)
  - keras.layers.MaxPooling2D(**pool_size=*(2, 2)***, **strides=*None***, **padding='*valid*'**)
  - keras.layers.UpSampling2D(**size=*(2, 2)***, **interpolation='*nearest*'**) : opération inverse du pooling
  - Ces trois couches existent aussi en version 1D et 3D
  - keras.layers.Flatten() (pour mettre les données sur une seule ligne avant des fully-connected layers)

- Quelques ressources :
  - *LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.*
  - *Zeiler, Matthew D., et al. "Deconvolutional networks." Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010.*
  - https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (explication générale sur les CNN)
  - http://scs.ryerson.ca/~aharley/vis/conv/flat.html (représentation d'un réseau de neurones et des couches intermédiaires sur les chiffres MNIST)

Tranfer learning

- Idée : conserver l'extraction des caractéristiques apprise sur d'autres problématiques
  - Revient à conserver des couches de convolution apprises sur un problème similaire
  - On ne change que la (ou éventuellement les) dernière(s) couche(s) d'identification

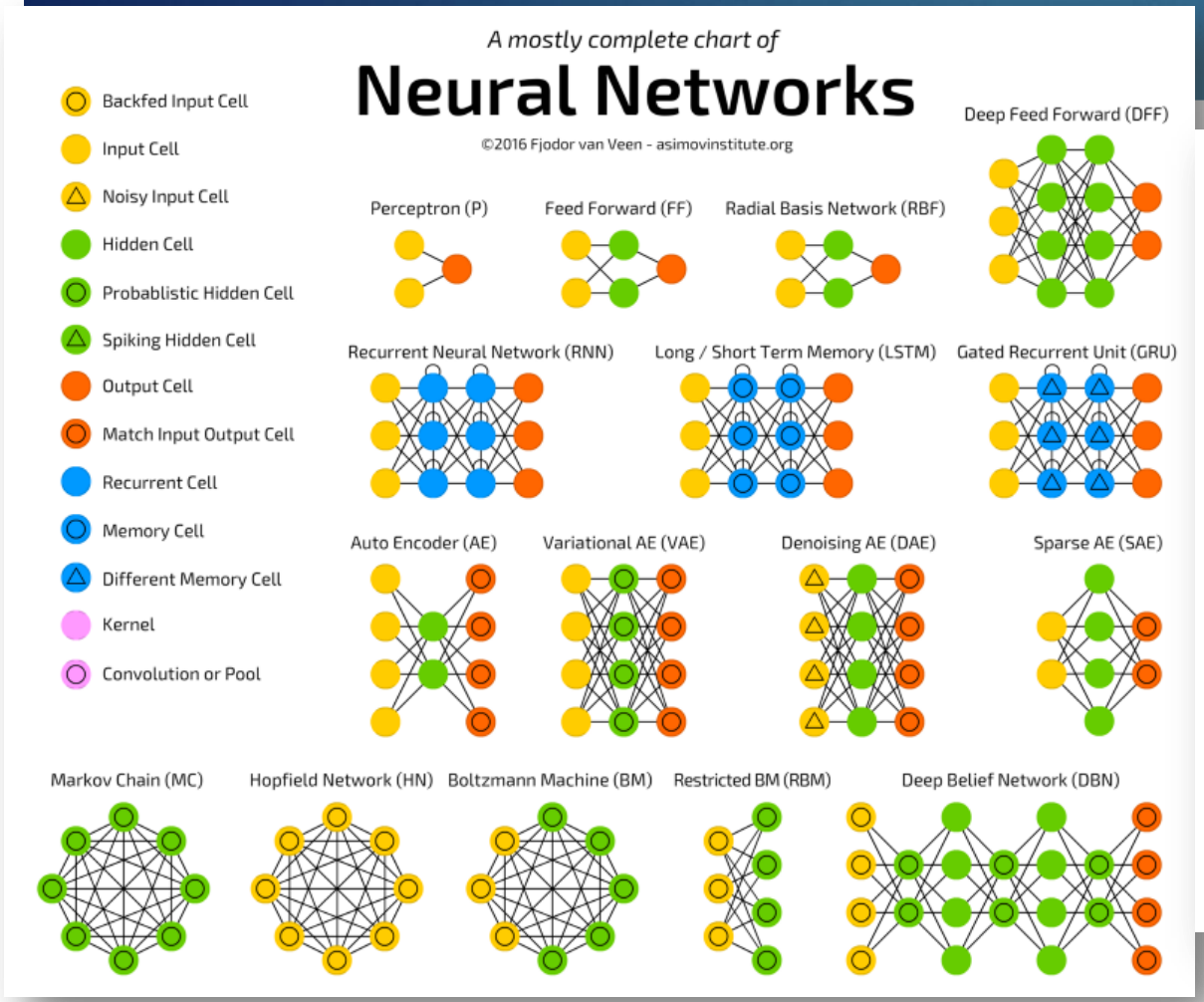
Fixée (déjà apprise)

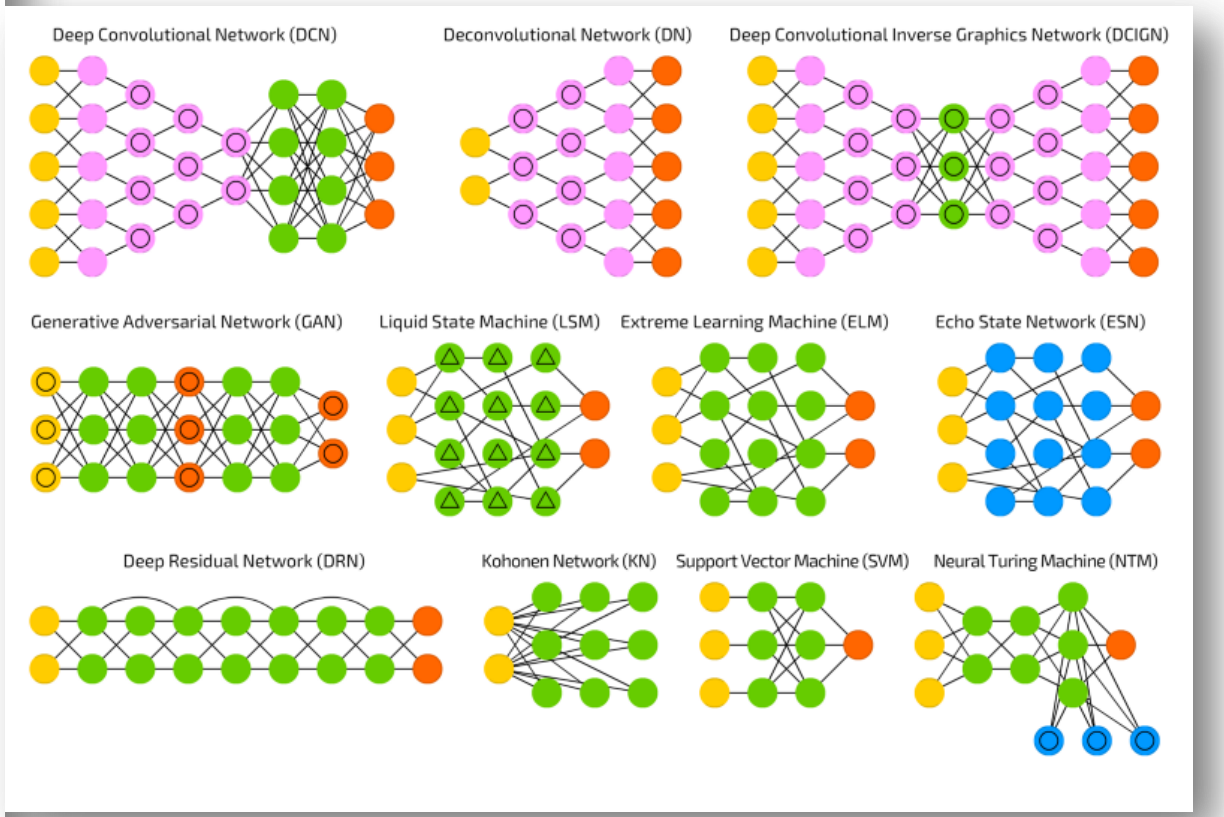


On apprend seulement ces couches

https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

D'autres structures : le zoo des réseaux

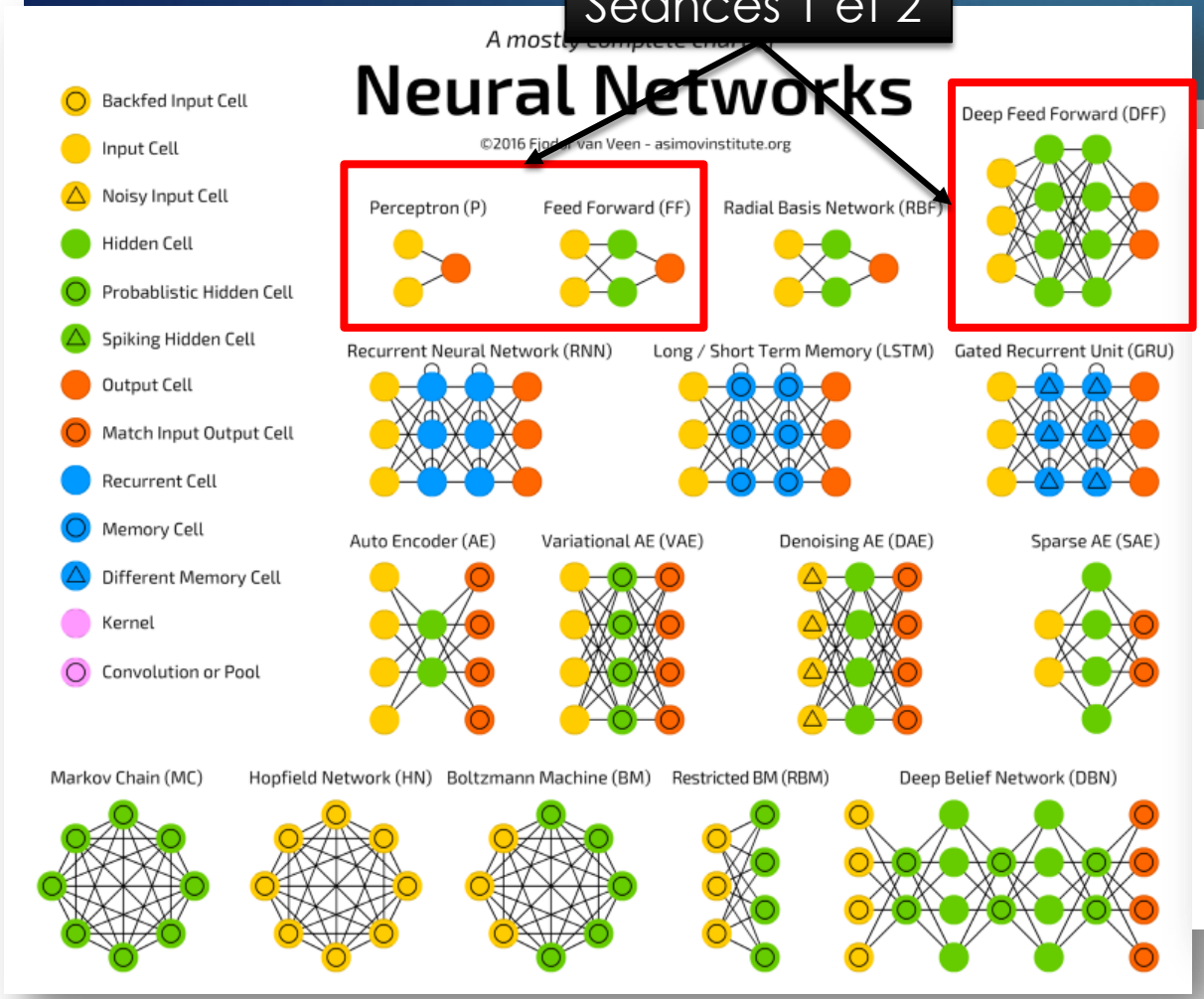


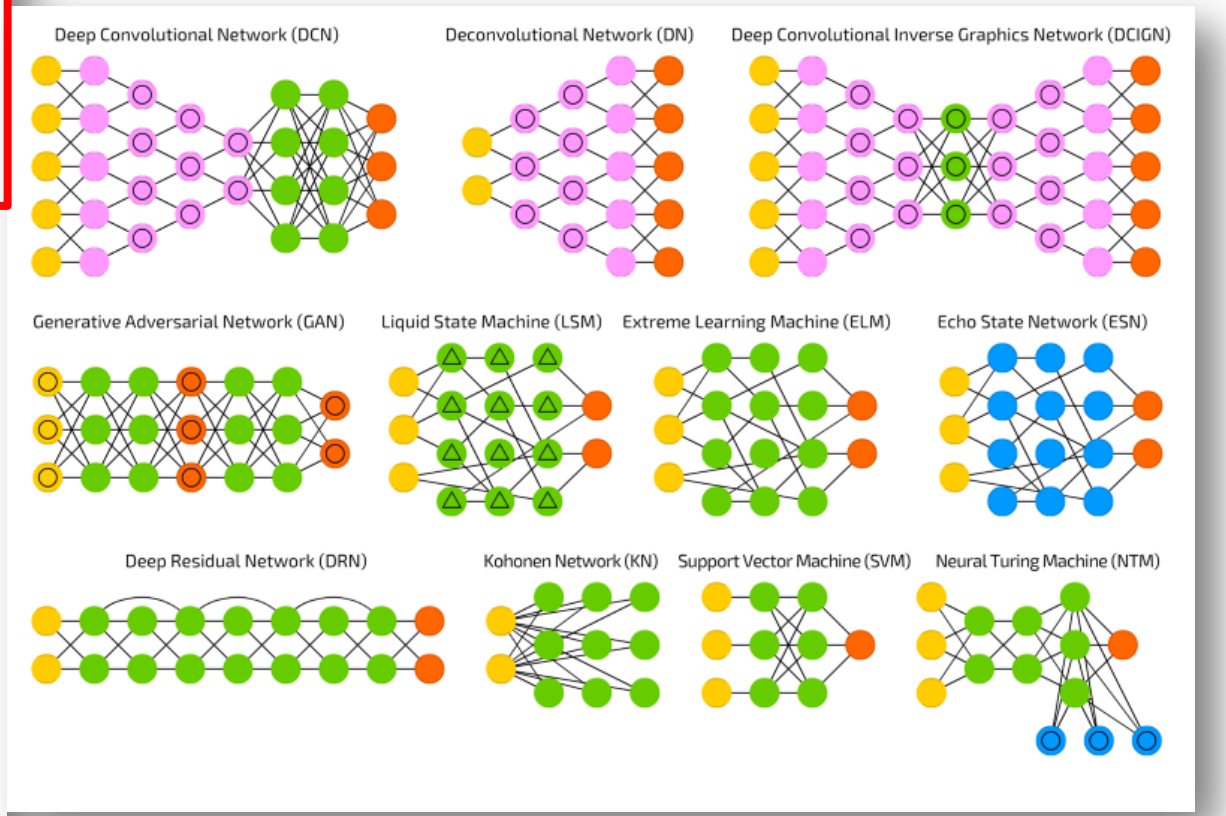


https://www.asimovinstitute.org/author/fjodorvanveen/

D'autres structures : le zoo des réseaux

Séances 1 et 2

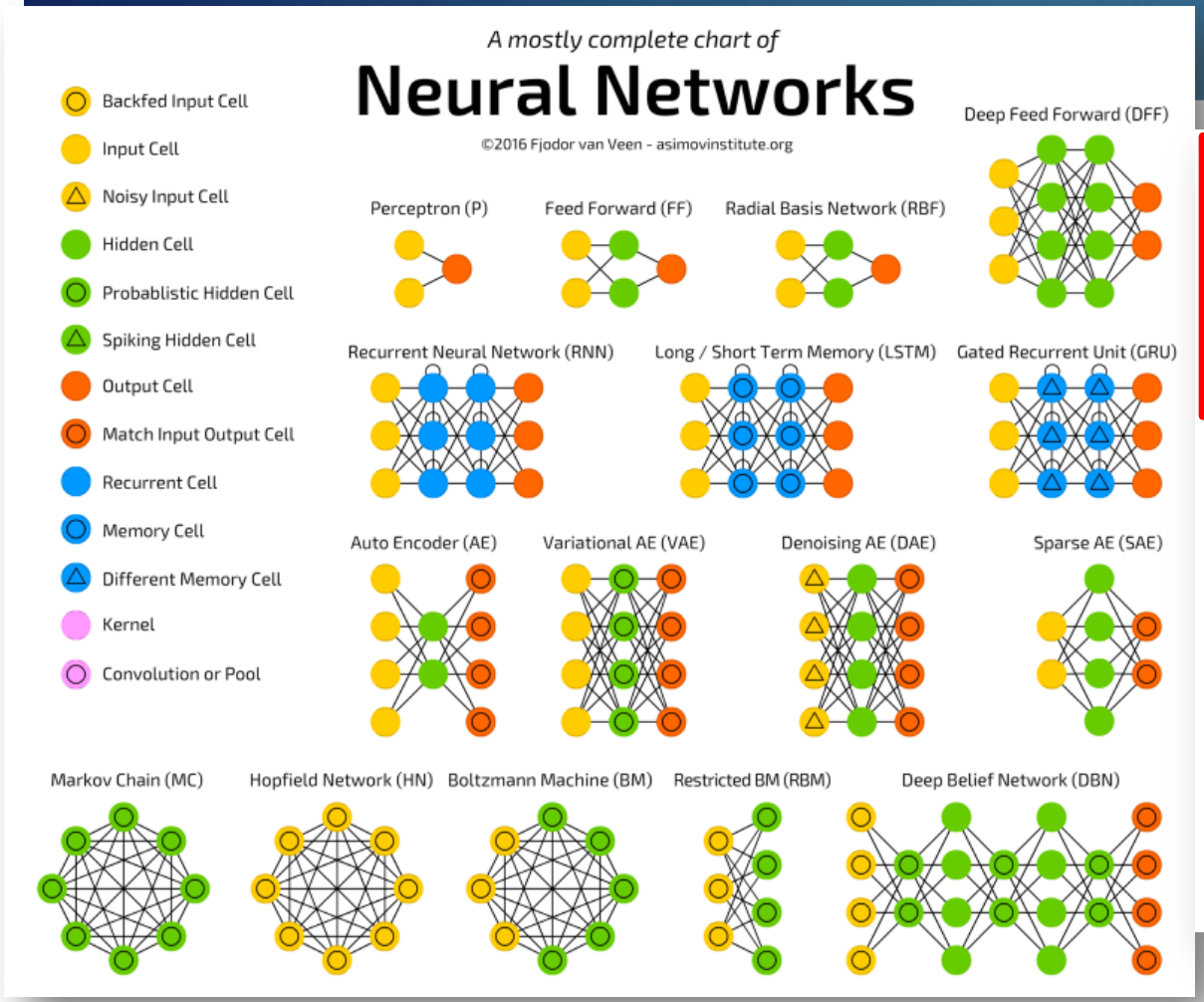


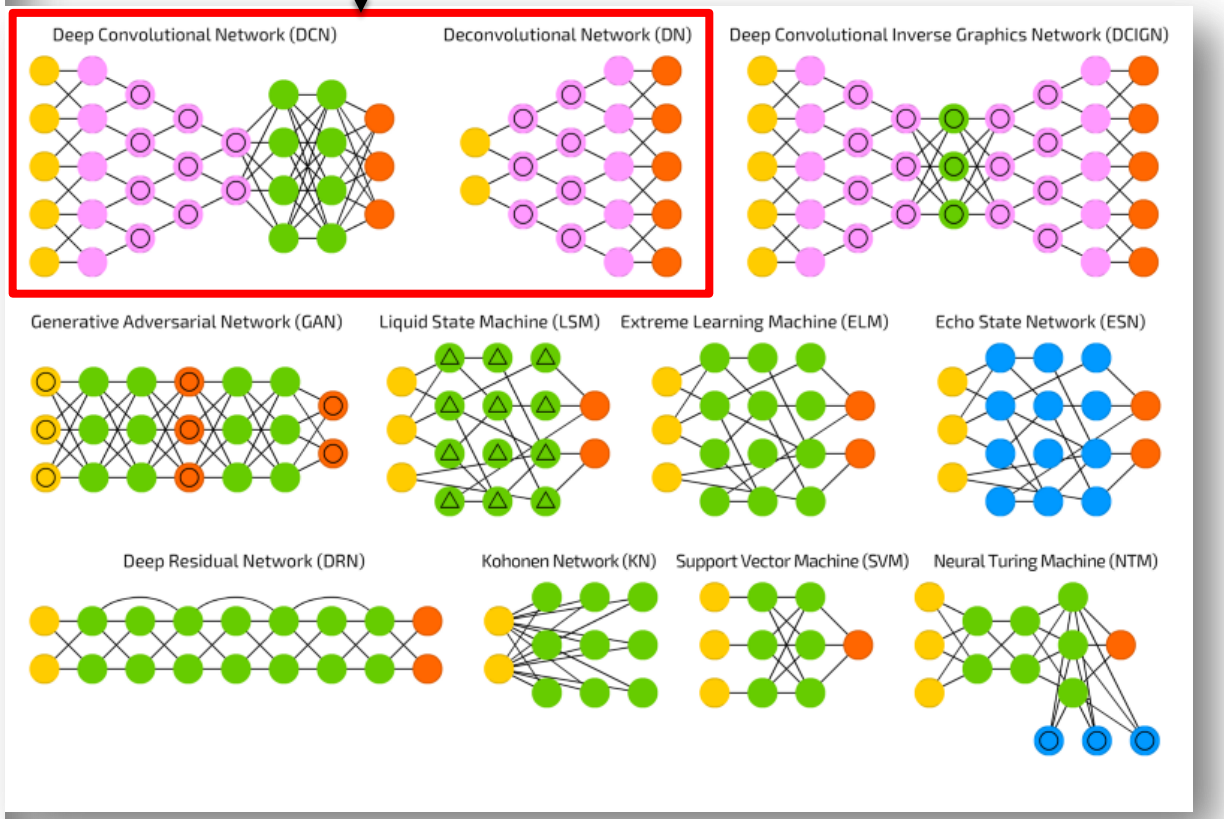


https://www.asimovinstitute.org/author/fjodorvanveen/

D'autres structures : le zoo des réseaux

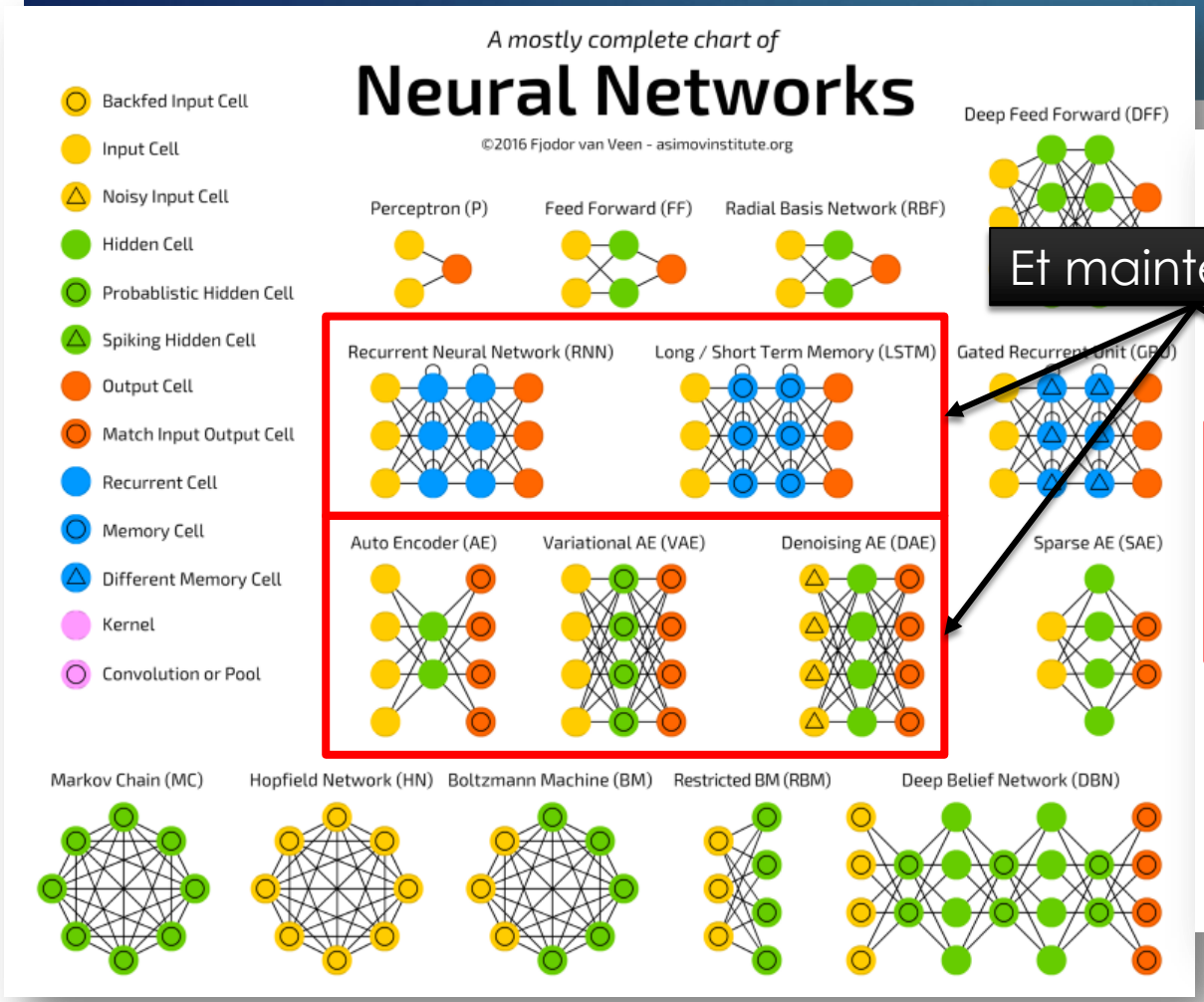
Séance 4



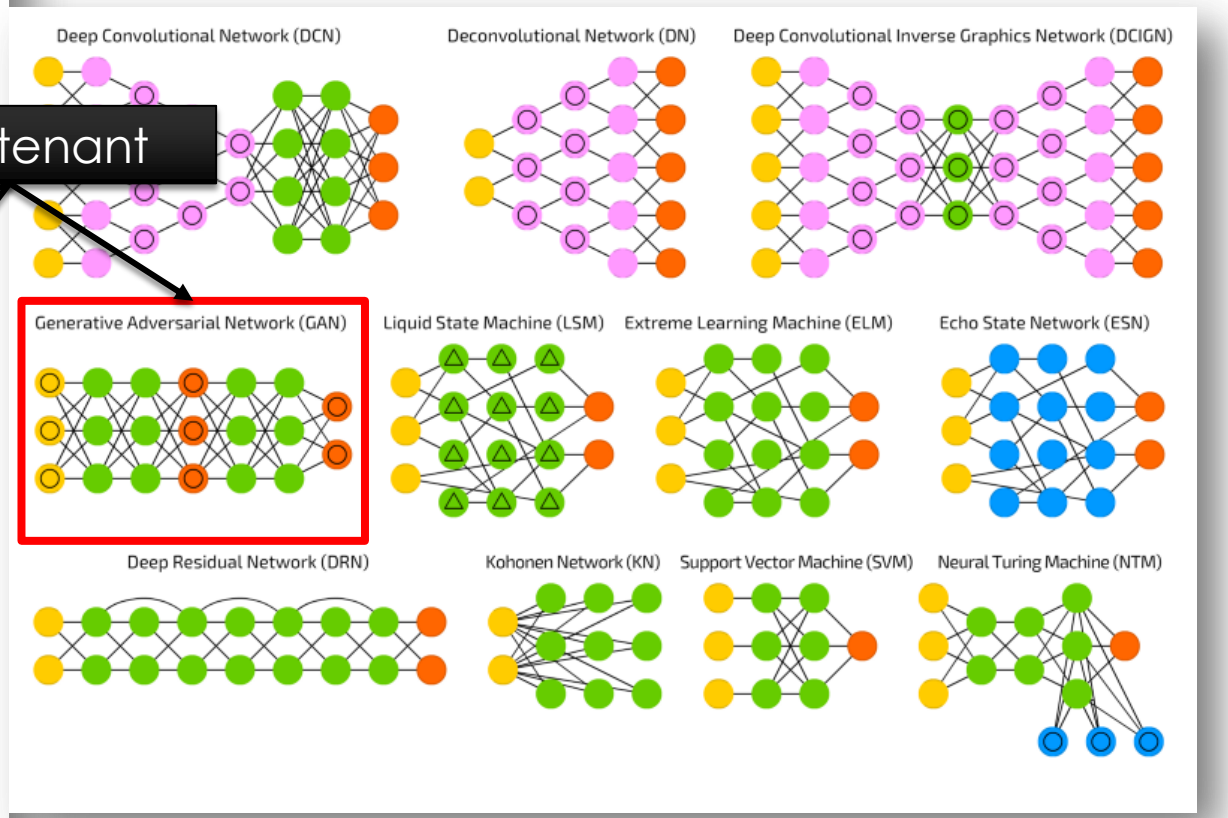


https://www.asimovinstitute.org/author/fjodorvanveen/

D'autres structures : le zoo des réseaux



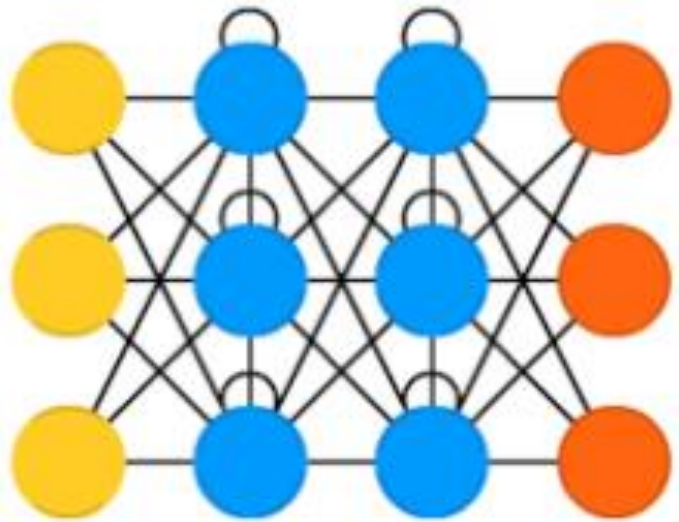
Et maintenant



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

D'autres structures : les réseaux récurrents

- Problème à traiter : analyse de séquences (textes, enregistrement audio/vidéo)
  - Input de tailles différentes
  - Ordre dans les données



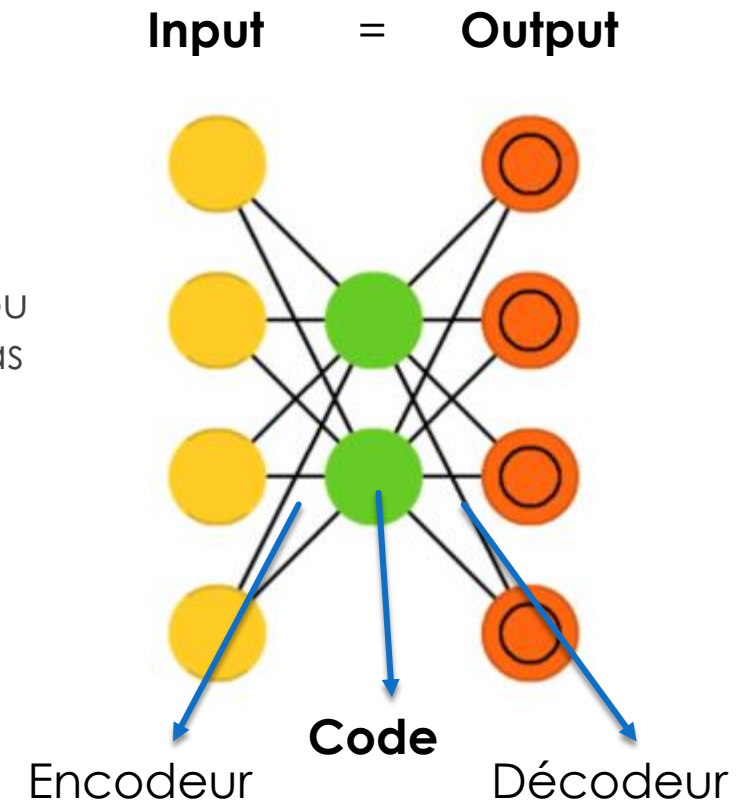
- Recurrent Neural Network (+ LSTM) :
  - Les neurones récurrents prennent en input :
    - La nouvelle information (le nouveau mot, la nouvelle image…)
    - Sa propre sortie précédente (ce que le neurone avait calculé avant)

- Quelques références :
  - *Elman, Jeffrey L. "Finding structure in time." Cognitive science 14.2 (1990)*
  - *Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997)*
  - *Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014)*

D'autres structures : les auto-encodeurs

- Problème : on veut réduire la dimension de nos données
  - Compresser les données

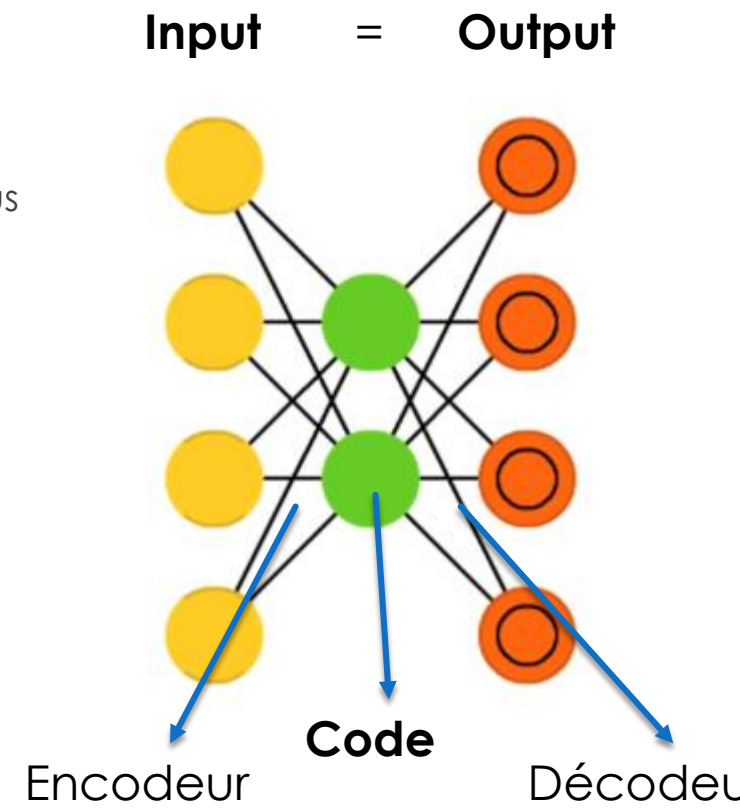
- Auto-encodeur :
  - On prend les mêmes données comme input et output ! Pas besoin de labels ou de « pré-travail ». Attention : on ne fait que compresser les données, il n'y a pas de classification à proprement parler.
  - Réseaux symétriques en général (en termes de nombre de neurones et de couches dans l'encodeur et le décodeur)
  - Il n'y a pas forcément une unique couche, on peut aussi combiner avec des couches de convolution
  - La couche du milieu correspond au « code »



D'autres structures : les auto-encodeurs

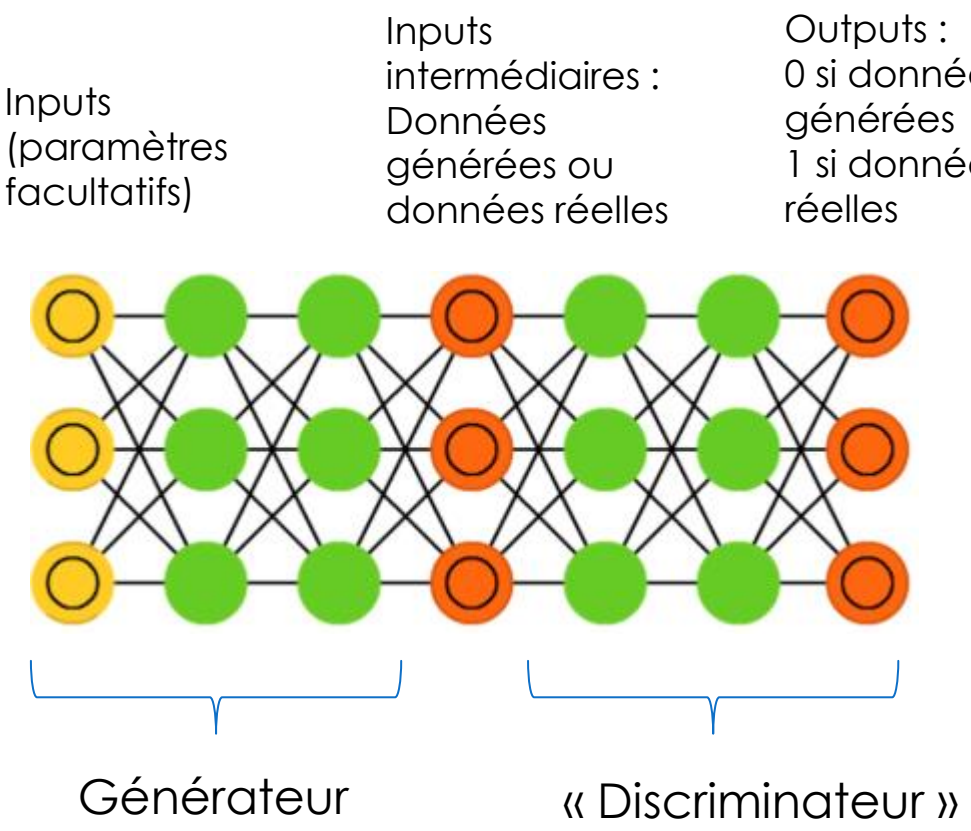
- Utilisation :
  - Débruiter les données
  - Réduire la dimension avant d'appliquer un autre algorithme :
    - Mettre des couches d'identification à la place du décodeur : le « code » peut être plus simple à traiter
    - Garder le code et faire du clustering, de la classification non supervisée sur le « code »

- Quelques ressources :
  - *Bourlard, Hervé, and Yves Kamp. "Auto-association by multilayer perceptrons and singular value decomposition." Biological cybernetics 59.4-5 (1988)*
  - *Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. "Efficient learning of sparse representations with an energy-based model." Proceedings of NIPS. 2007*
  - *Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. ACM, 2008.*



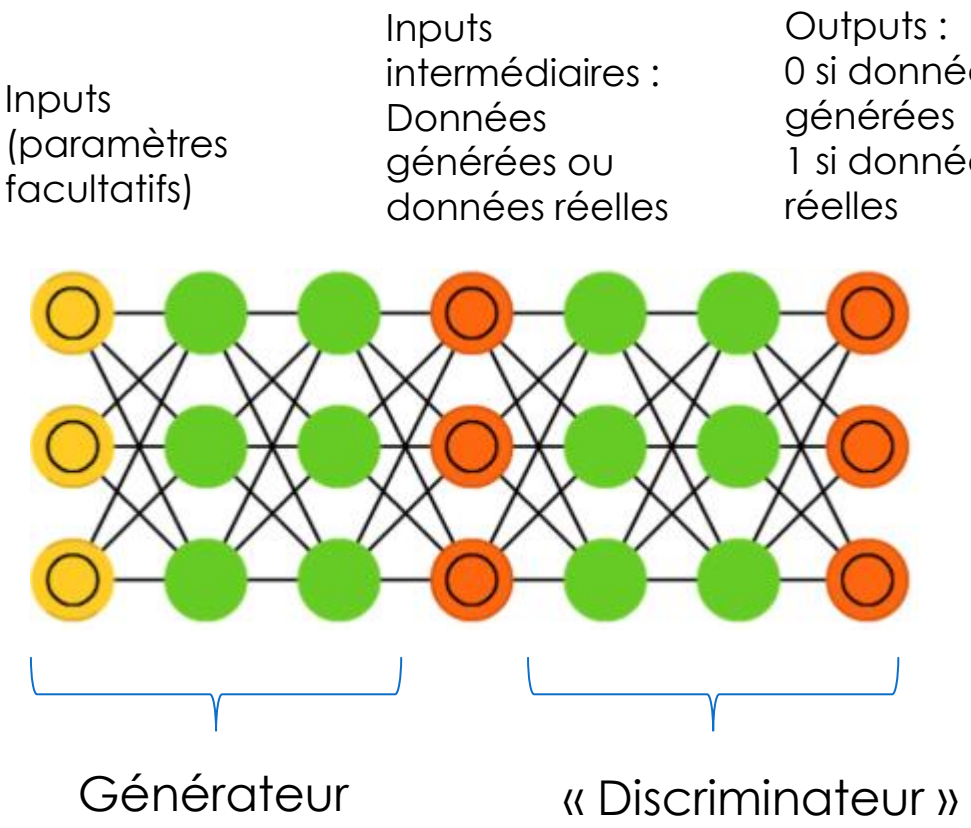
D'autres structures : les GANs

- **Generative Adversarial Networks**
- Problème : on veut générer des données qui semblent « vraies »
  - Exemple : générer des images de personnes qui n'existent pas mais qui ont l'air réelles
  - Utilisation possible en simulation
- Deux réseaux travaillent en compétition :
  - Un réseau générateur essaie de générer des inputs qui ont l'air « vrais »
  - Un réseau discriminateur essaie de distinguer les données générées des données réelles



D'autres structures : les GANs

- Méthode : on dispose de données réelles
  - Première étape : générateur initialisé aléatoirement, le discriminateur est entraîné (Inputs : données réelles ou générées, Outputs : 0 si généré, 1 si réel)
  - Deuxième étape : on fixe le discriminateur, et on entraîne le générateur à tromper le discriminateur (il faut que le discriminateur donne 1 au maximum de données générées)
  - Ensuite : on reprend l'étape 1 avec le nouveau générateur entraîné, puis l'étape 2 et ainsi de suite
- Quelques ressources :
  - "*Generative adversarial nets.*" Goodfellow, Ian, et al. Advances in Neural Information Processing Systems. 2014.
  - *Generative models for fast simulation*, S.Vallecorsa, CERN
  - *Generative adversarial networks simulate gene expression and predict perturbations in single cells,* Ghahramani, Watt, Luscombe (2018)



Évaluer l'incertitude du réseau

- Objectifs :
  - Classification : savoir si le réseau est certain ou non de sa prédiction
  - Régression : avoir une barre d'erreur sur le résultat

- Idée : utiliser les concepts bayésiens
  - On voudrait que le réseau apprenne la distribution des paramètres θ qui expliquent les inputs X_{train} et les ouptuts Y_{train} :

$$p(\theta|X_{train}, Y_{train}) = \frac{p(Y_{train}|X_{train}, \theta)p(\theta)}{p(Y_{train}|X_{train})}$$

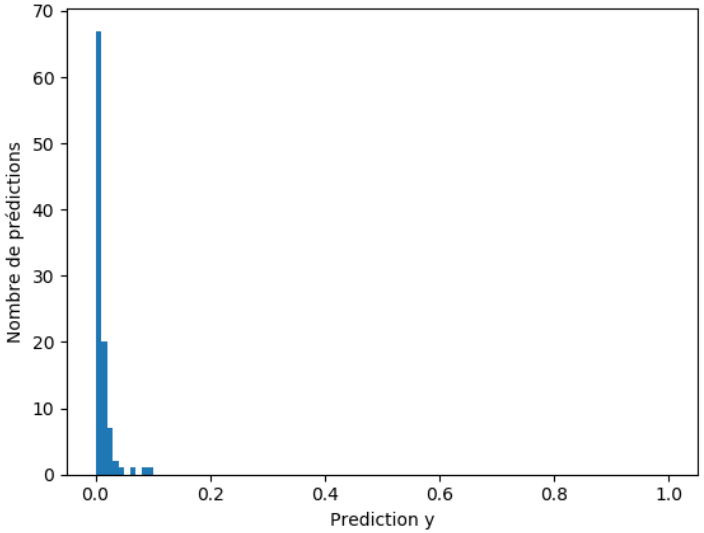
  - Bayesian Neural Network : les poids W sont tirés suivant une loi normale $\mathcal{N}(\mu, \sigma)$ où μ et σ sont les paramètres à apprendre.
  - Chaque neurone tire aléatoirement ses paramètres suivant cette loi, puis applique les calculs habituels du réseau. Plusieurs tests par le même réseau va donner un résultat différent : on obtient une distribution de résultats qui permet de caractériser $p(Y_{test}|X_{test})$ et donc d'obtenir une erreur (en calculant par exemple $Var(p(Y_{test}|X_{test}))$

Évaluer l'incertitude du réseau

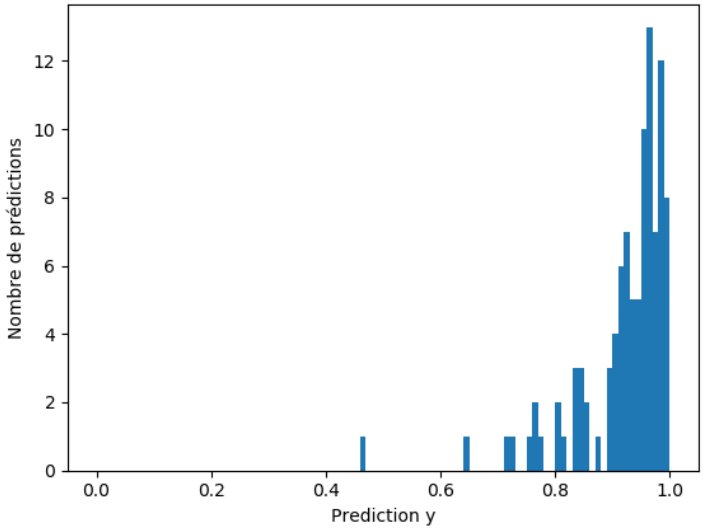
- En pratique : difficile d'utiliser et d'apprendre des poids tirés aléatoirement
- Approximation de réseau bayésien : utiliser le dropout et l'activer dans la phase de tests
  - Rappel : dropout = extinction aléatoire des neurones → mise en place comme régularisation dans l'apprentissage, on la garde dans la phase de test
  - Méthode :
    - On effectue l'apprentissage normalement
    - Pour tester un exemple x_{test}, on garde le dropout actif. En appliquant n fois le même réseau, mais avec un dropout aléatoire, on obtient n prédictions $(y_1, \dots, y_n)$ prédictions pour ce même test.
    - Attention : sur Keras, le Dropout est activé par défaut seulement pour l'apprentissage et non pour le test. Il faut le « forcer » à rester actif pour le test.
- Pour aller plus loin :
  - *Uncertainty in Deep Learning,* Yarin Gal, thèse Université de Cambridge

Évaluer l'incertitude du réseau

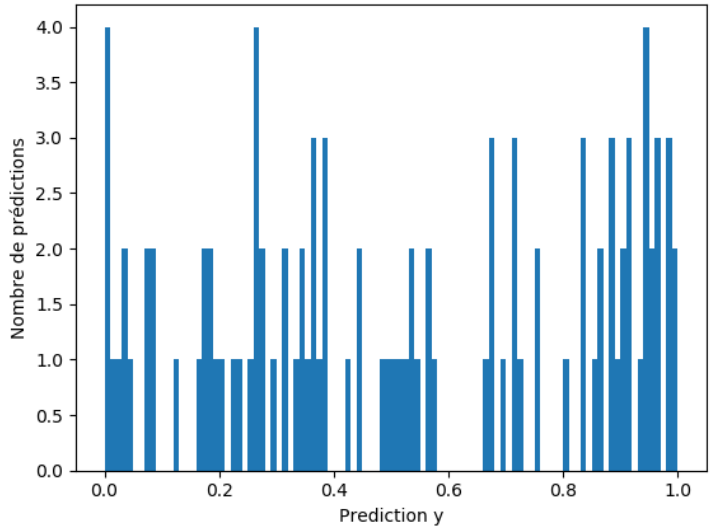
Exemple pour la classification (0 ou 1) : histogramme en sortie du sigmoïde



Prédiction négative (0) certaine



Prédiction positive (1) (relativement) certaine



Prédiction totalement incertaine

Pour résumer

- Réseaux de neurones convolutifs :
  - Permet d'extraire les caractéristiques pertinentes dans les inputs : caractéristiques qui sont invariantes par translation (forme dans les images), corrélation locale
- De nombreuses autres structures existent :
  - Récurrentes : pour les séquences
  - Auto-encoder, GANs…
- Évaluation de l'incertitude du réseau :
  - En utilisant le dropout sur le jeu de test