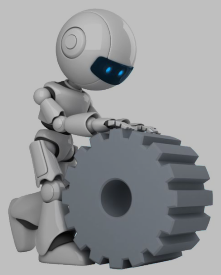
# Apprentissage Par Projet [APP]

# Comprendre et utiliser le *Machine learning*

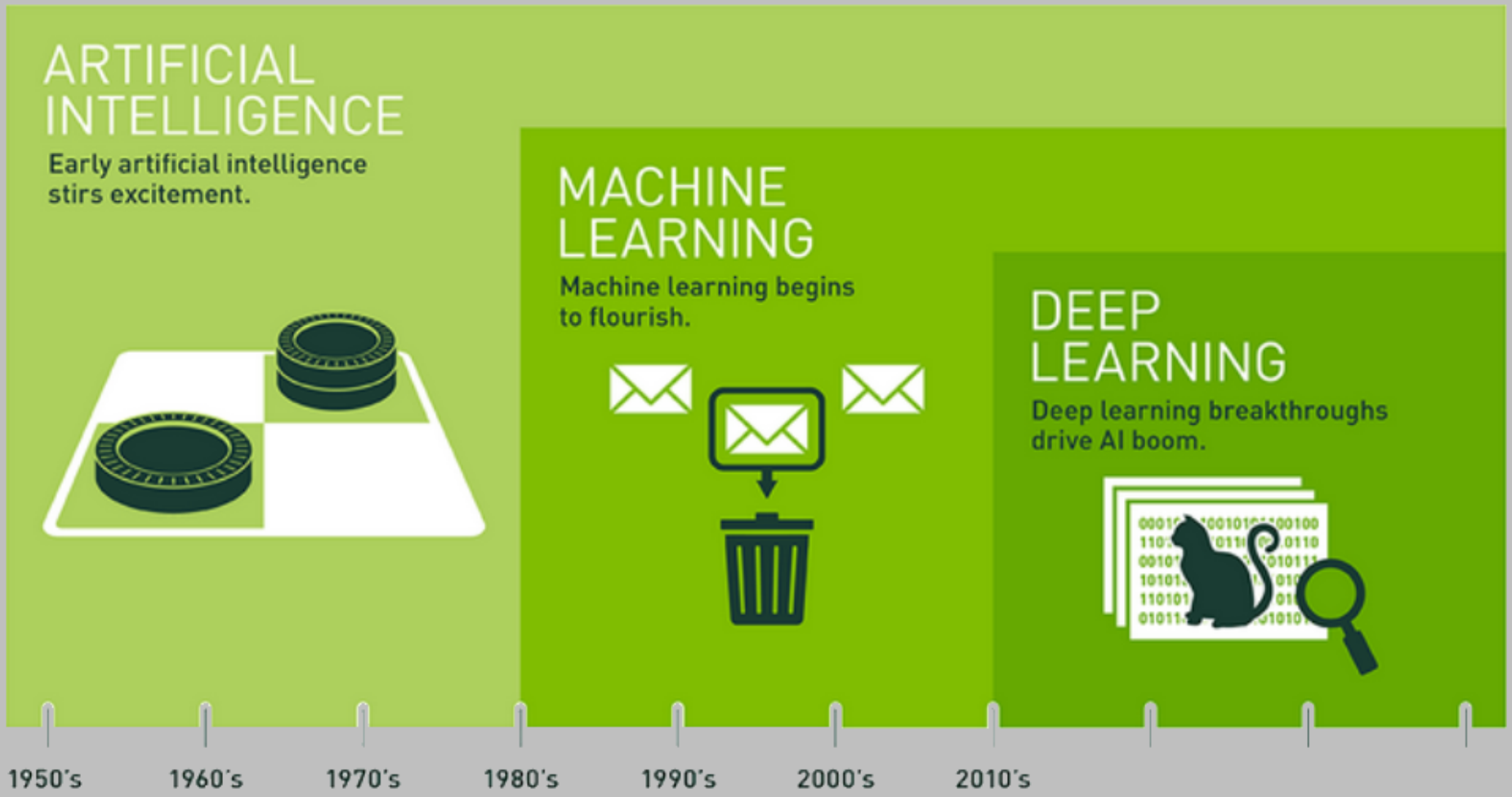
Jean-Luc.Charles@ENSAM.EU

Arts et Métiers Sciences et Technologies

avril 2022



# L'aspect historique...




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

# Intelligence Artificielle ?

**Intelligence Artificielle** [1] : reste un terme ambigu aux définitions multiples :

[1] utilisé la première fois en 1956 par John McCarthy, chercheur à Stanford lors de la conférence de Dartmouth

# Intelligence Artificielle ?

**Intelligence Artificielle** [1] : reste un terme ambigu aux définitions multiples :

- *"...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 d'*agent intelligent* ou d'*agent rationnel*
  *"...agent qui agit de manière à atteindre la meilleure solution ou, dans un environnement incertain, la meilleure solution prévisible."* Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015

[1] utilisé la première fois en 1956 par John McCarthy, chercheur à Stanford lors de la conférence de Dartmouth

# Intelligences Artificielles ?

***IA Forte*** (*Strong AI*)

***IA Faible*** (*Weak AI*)

# Intelligences Artificielles ?

***IA Forte*** (*Strong AI*)

- Vise à concevoir des systèmes qui pensent exactement comme les humains.
- Peut contribuer à expliquer comment les humains pensent...
- On en est encore loin... veut-on vraimment aller jusque là ?

***IA Faible*** (*Weak AI*)

# Intelligences Artificielles ?

***IA Forte*** (*Strong AI*)

- Vise à concevoir des systèmes qui pensent exactement comme les humains.
- Peut contribuer à expliquer comment les humains pensent...
- On en est encore loin... veut-on vraimment aller jusque là ?

***IA Faible*** (*Weak AI*)

- Vise à concevoir des systèmes qui peuvent “se comporter” comme des humains.
- Ne nous dit rien sur la façon dont les humains pensent.
- On y est déjà... On l’utilise tous les jours ! reconnaissance faciale, vocale, anti-spam, traduction...

# *Machine Learning* et IA

Page extraite de 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*

# Les branches du *Machine Learning*

## *Supervised learning*
## Apprentissage supervisé

Requiert des **données labelisées**...

- **Classification**
  - Classification d'images
  - Détection d'objet sdans des images
  - Reconnaissance de la parole...
- **Régression**
  - Prédiction d'une valeur (continue)...
- **Détection d'Anomalies**
  - Détection de Spam
  - Manufacturing: reconnaissance de défauts (appris)
  - Prévision du temps
  - Classification de maladies...

# Les branches du *Machine Learning*

## *Unsupervised learning*
## Apprentissage non-supervisé

Ne requiert que des données (non labelisées)...

- **Clustering** – **Regroupement** de données non labelisées
  - Data mining, regroupement de données du web, de news...
  - Regroupement ADN
  - Traitement de données d'astronomie...
- **Detection d'Anomalie**
  - Détection de fraude
  - Manufacturing : détection de défauts (même nouveaux)
  - Monitoring : détéction d'activité anormale (panne, hacker, fraude...)
  - *Fake account* sur Internet...
- **Réduction de dimension**
  - Compression de données...

# Les branches du *Machine Learning*

## *Reinforcement learning*
## Apprentissage par renforcement

Un agent (le réseau de neurones) apprend à piloter un environnement (jeu, système mécatronique...)

- **Contrôle/commande**
  - Contrôle de robots, drones...
  - Financial (stock) trading...
- **Prise de décision**
  - jeux (video games)
  - analyse financière...

# Machine Learning et IA

Plusieurs approches permettent de concevoir des algorithmes de *Machine Learning* :

- Programmation Génétique (*Genetic programming*)
- Inférence bayésienne (*Bayesian inference*)
- Logique Floue (*Fuzzy logic*)
- Réseaux de neurones (*Neural Networks*)
- ...

# Machine Learning et IA

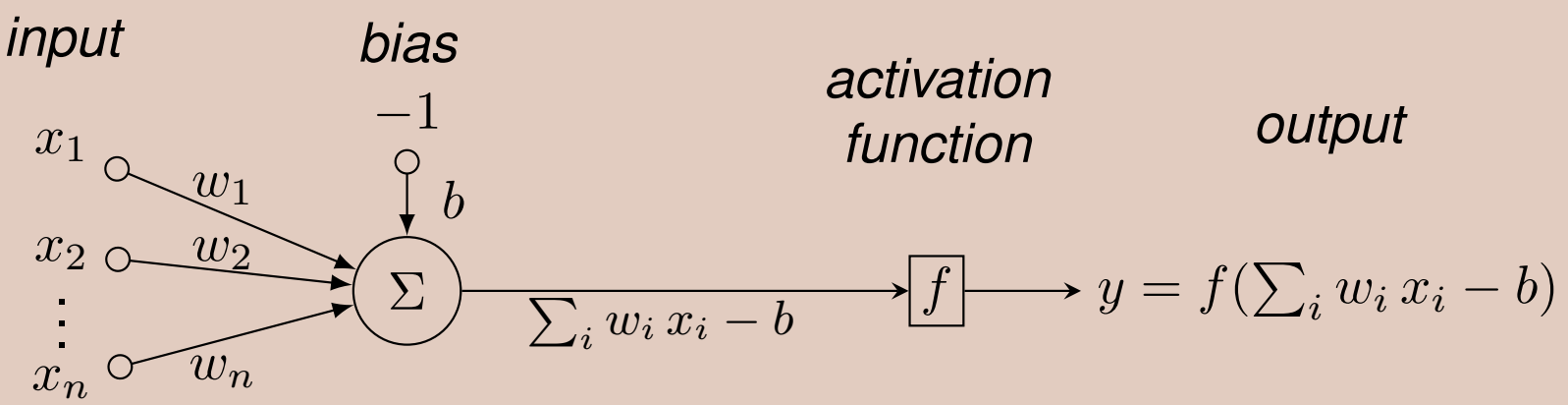
Plusieurs approches permettent de concevoir des algorithmes de *Machine Learning* :

- Programmation Génétique (*Genetic programming*)
- Inférence bayésienne (*Bayesian inference*)
- Logique Floue (*Fuzzy logic*)
- Réseaux de neurones (*Neural Networks*)
- ...

La suite traite uniquement des **Réseaux de neurones artificiels**.

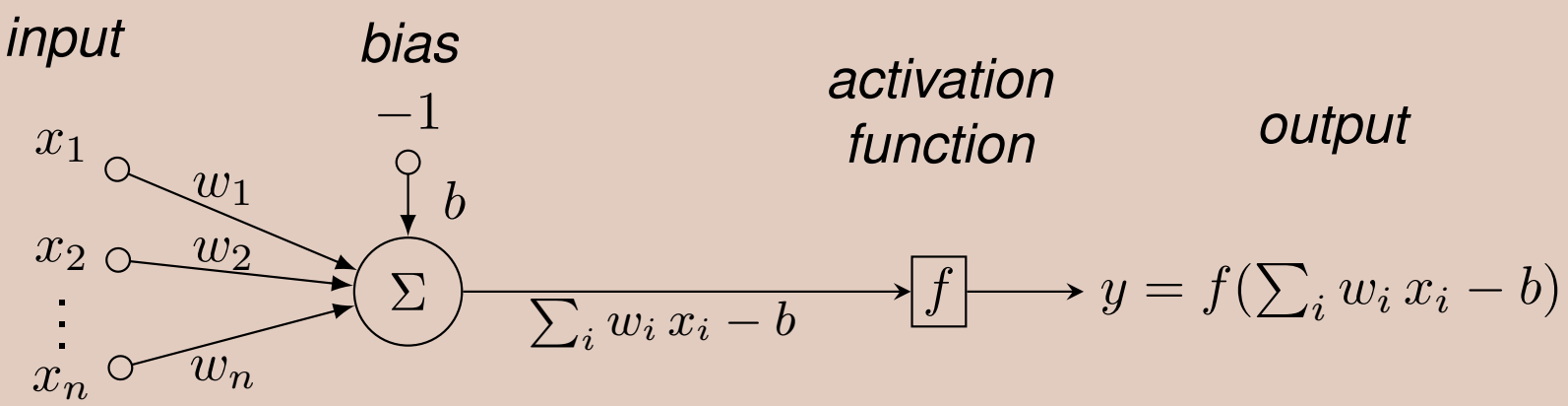
# Neurone artificiel

## Le modèle informatique du neurone artificiel



# Neurone artificiel

## Le modèle informatique du neurone artificiel

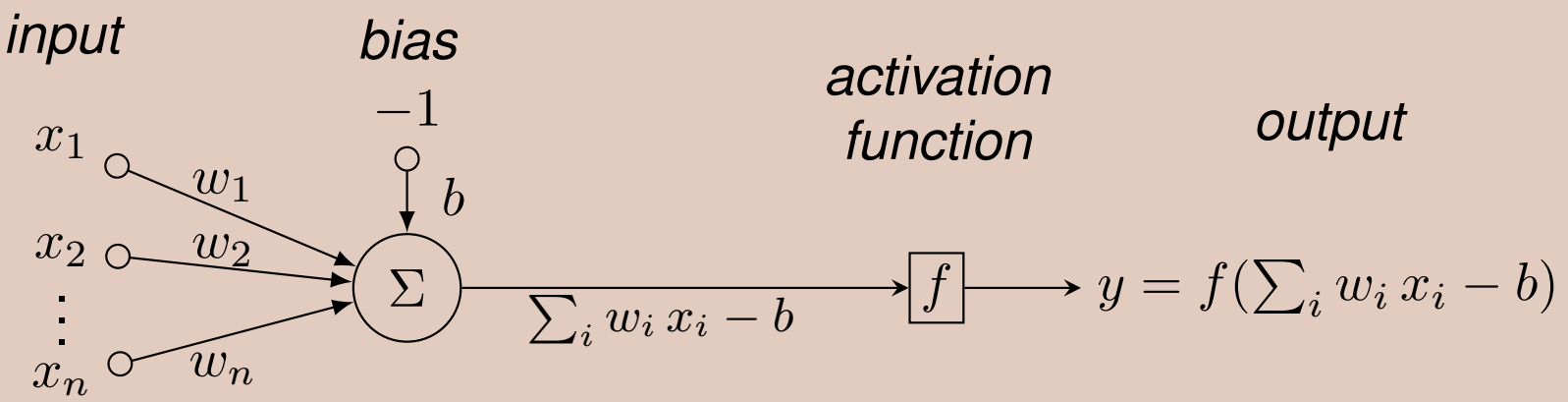



Un **neurone artificiel**:

- reçoit les données d'entrée $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$ (*weights*)

# Neurone artificiel

## Le modèle informatique du neurone artificiel

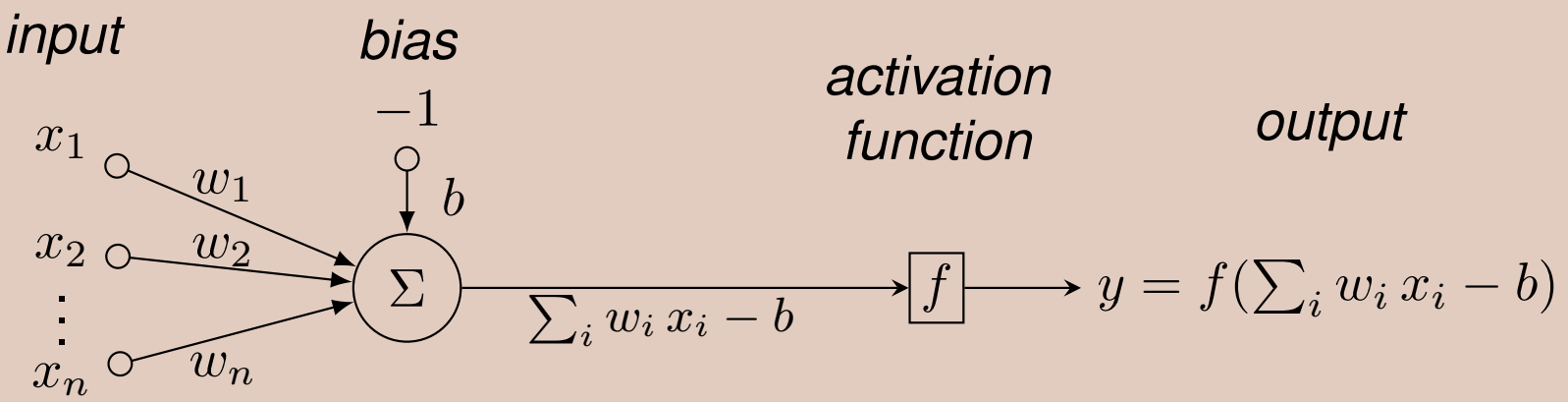



Un **neurone artificiel**:

- reçoit les données d'entrée $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$ (*weights*)
- calcule la **somme pondérée** de ses entrées moins le biais $\sum_i w_i\, x_i - b$

# Neurone artificiel

## Le modèle informatique du neurone artificiel




Un **neurone artificiel**:

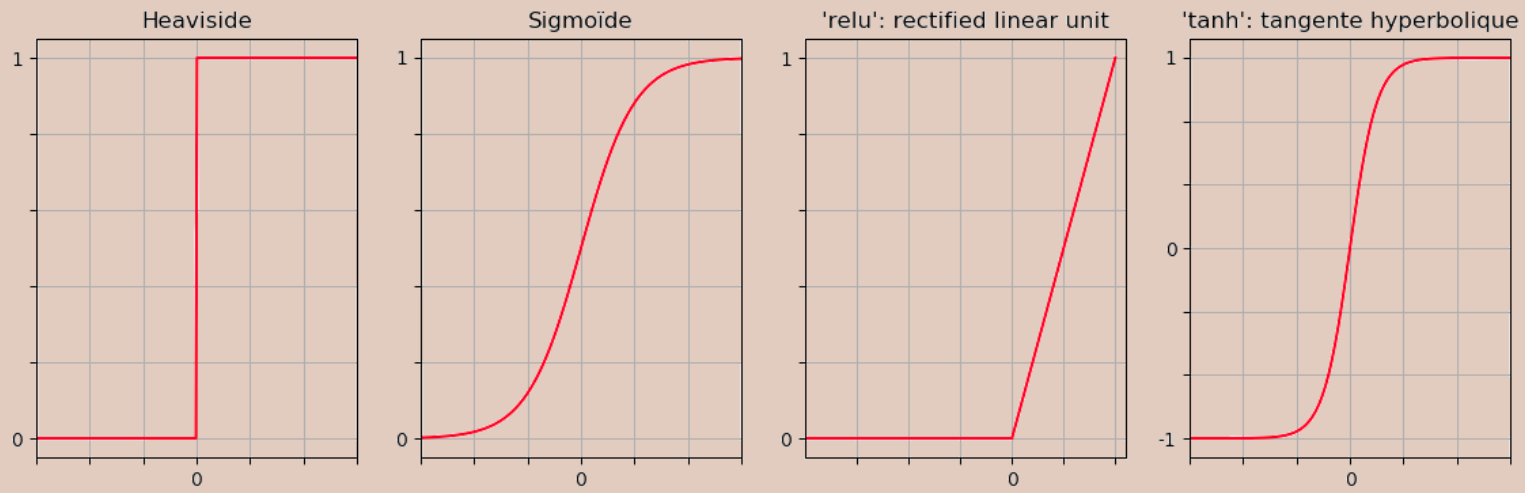
- reçoit les données d'entrée $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$ (*weights*)
- calcule la **somme pondérée** de ses entrées moins le biais $\sum_i w_i\, x_i - b$
- produit en sortie une **activation** $f(\sum_i w_i\, x_i - b)$, calculée avec une fonction d'activation $f$ (en général non-linéaire).

# Neurone artificiel

La fonction d'activation d'un neurone :

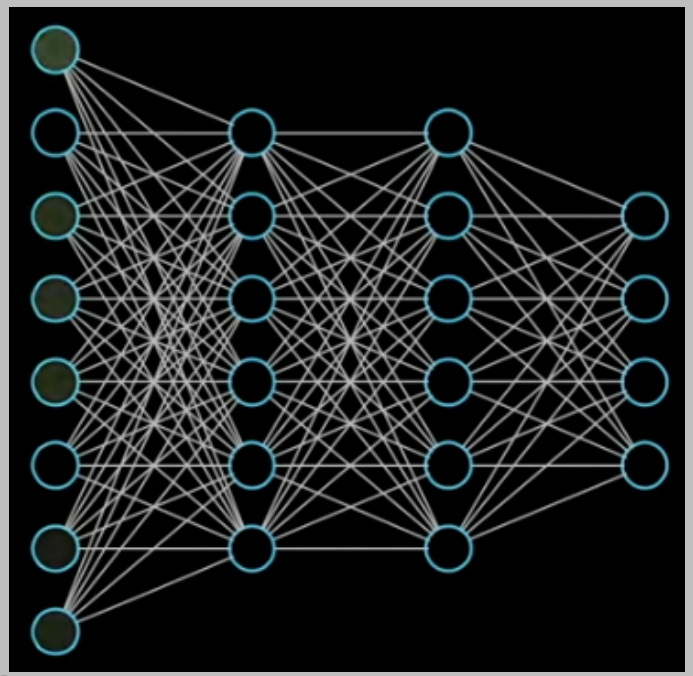
- indroduit un comportement non-linéaire,
- fixe la plage de la sortie du neurone,
  par exemple $[-1, 1]$, $[0, 1]$ ou encore $[0, \infty[$.

## Exemples de fonctions d'activation souvent utilisées




- Le biais $b$ fixe le seuil d'activation du neurone.

# Réseaux de neurones



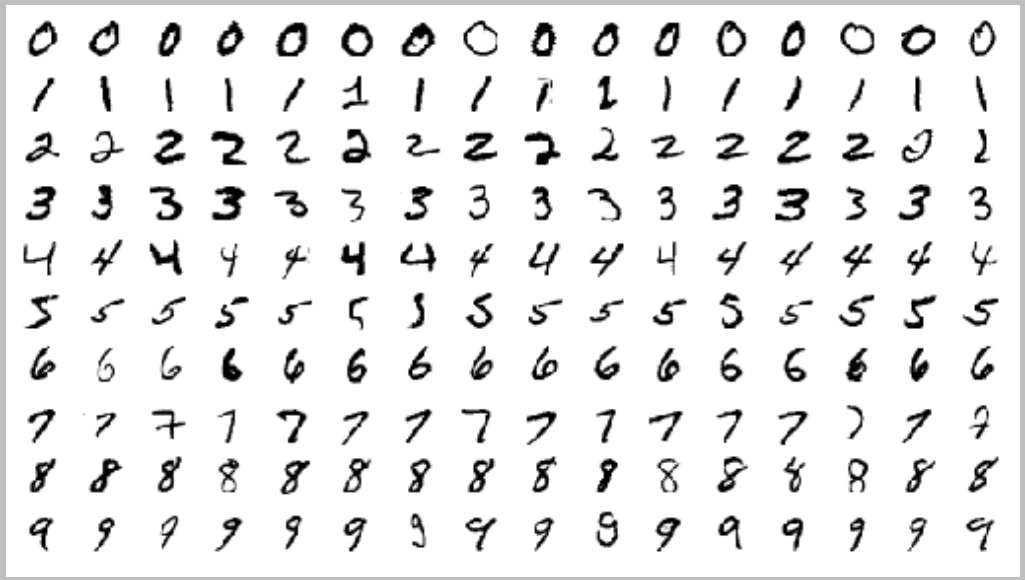
- Les réseaux de neurones sont des assemblages plus ou moins complexes de neurones artificiels organisés en couches.

Deux architecture sont très souvent utilisées :

- **Réseau de Neurones Dense** (*Dense Neural Network, RND*)
  simple, généraliste, peut atteindre des scores de réussite importants.
- **Réseau de Neurones Convolutif** (*Convolutional Neural Network, CNN*)
  plus complexe, spécialisé dans le traitement des images, peut atteindre des scores supérieurs à 99% dans la reconnaissance d'images.

# Données utilisées pour l'APP

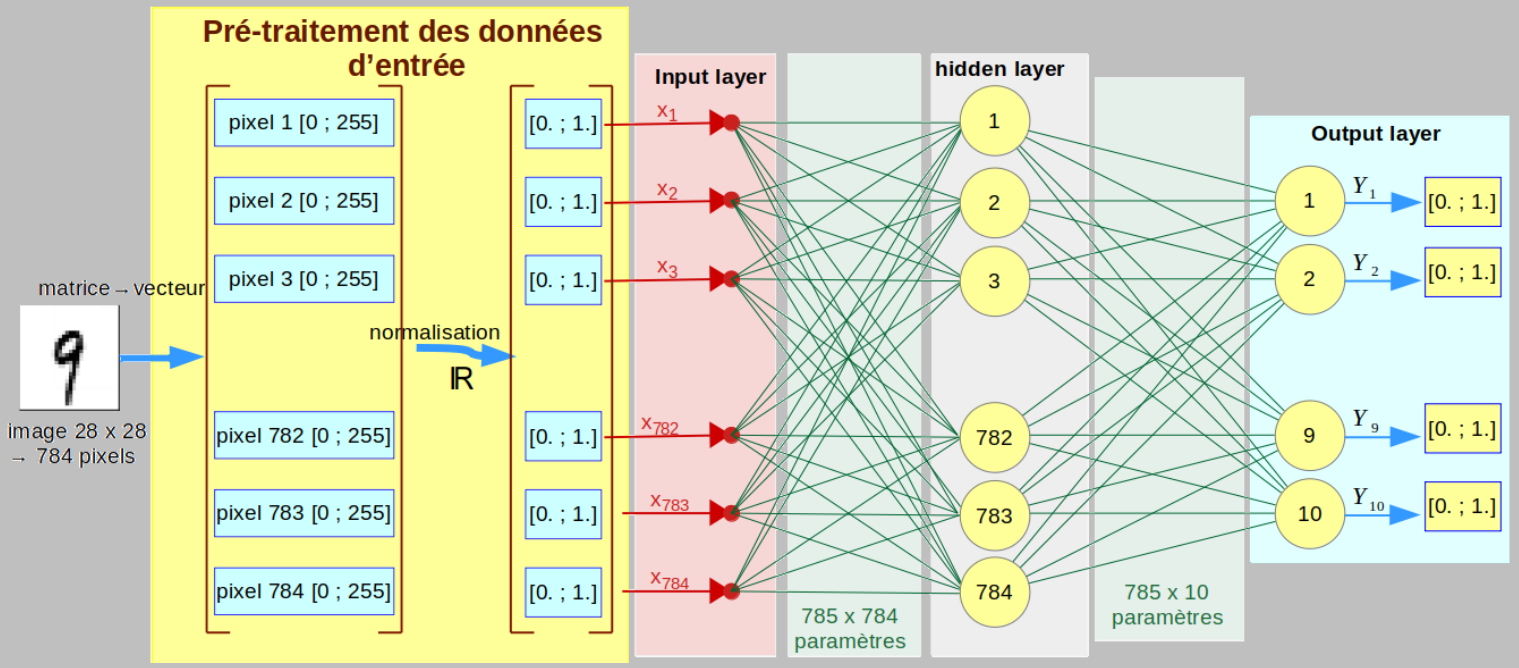
- MNIST banque de 70000 **images labellisées**
  (60000 images d'entraînement – 10000 images de test.



- Images en ton de gris de $28 \times 28$ pixels
- Réseau dense $\rightsquigarrow$ scores pouvant atteindre 98 % de succès...
- État de l'art pour la reconnaissance d'image : réseau convolutifs

# 1 - Réseau de neurones dense (séquentiel)

Chaque matrice $28 \times 28 \rightsquigarrow$ vecteur normalisé de 784 composantes `float` $\in [0; 1]$.



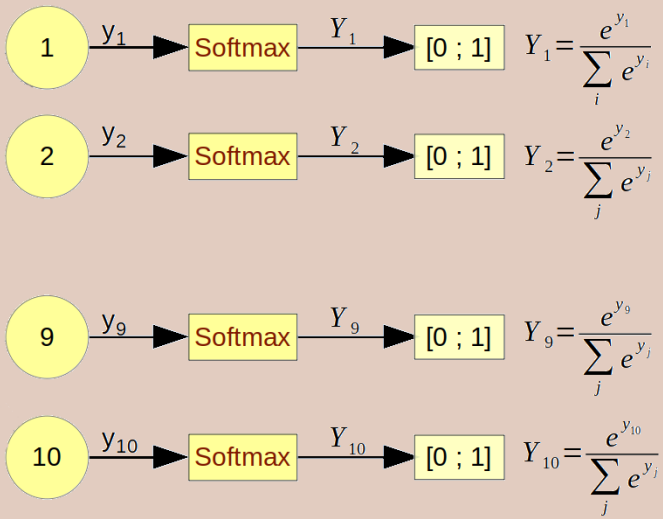

Constitution du réseau :

- La couche d'entrée (*Input layer*) fixe la dimension des entrées du réseau à 784 valeurs. **Elle ne comporte aucun neurone !**
- Une couche "cachée" (*Hidden layer*) de 784 neurone (on pourrait en avoir plus, ou moins...), reçoit les données d'entrées. Elle est connectée à la couche suivante.
- La couche de sortie (*Output layer*) contient 10 neurones (1 neurone pour chaque chiffre à reconnaître).

# 1 - Réseau de neurones dense

- Dans les couches intermédiaires ("cachées") la fonction d'activation *relu* favorise souvent l'apprentissage du réseau [2].
- La classification (dernière couche) utilise la fonction *softmax* :

## Fonction d'activation *softmax*




- L'activation du neurone $k$ est $Y_k = e^{y_k} / \sum_i e^{y_i}$ avec $y_k = \sum_i \omega_i x_i - b$ calculé par le neurone $k$.
- Les sorties des neurones s'interprêtent comme des probabilités dans l'intervalle [0,1].

La réponse du réseau est le label associé au neurone de plus grande probabilité (activation).

[2] évite le *vanishing gradient* qui apparaît dans l'algorithme de *back propagation*

# 1 - Réseau de neurones dense

## Codage *One-hot* des labels

But : mettre les label des images au format de la sortie du réseau

- Labels des images : **nombres entiers** de 0 à 9.
- Sortie du réseau : **vecteur de 10** `float` dans l'intervalle [0,1] calculés par les fonctions *softmax* des 10 neurones de sortie.
- Codage *one-hot* d'une collection ordonnée de $N$ éléments uniques :

  - chaque élément est codé par un vecteur de $N$ composantes toutes nulles sauf une,
  - le *i ème* élément $\rightsquigarrow$ vecteur avec un 1 pour *i ème* composante.

# 1 - Réseau de neurones dense

## Codage *One-hot* des labels

But : mettre les label des images au format de la sortie du réseau

- Labels des images : **nombres entiers** de 0 à 9.
- Sortie du réseau : **vecteur de 10** `float` dans l'intervalle [0,1] calculés par les fonctions *softmax* des 10 neurones de sortie.
- Codage *one-hot* d'une collection ordonnée de $N$ éléments

élément est codé par un vecteur de $N$ composantes
lles sauf une,
élément $\rightsquigarrow$ vecteur avec un 1 pour *i ème* composante.

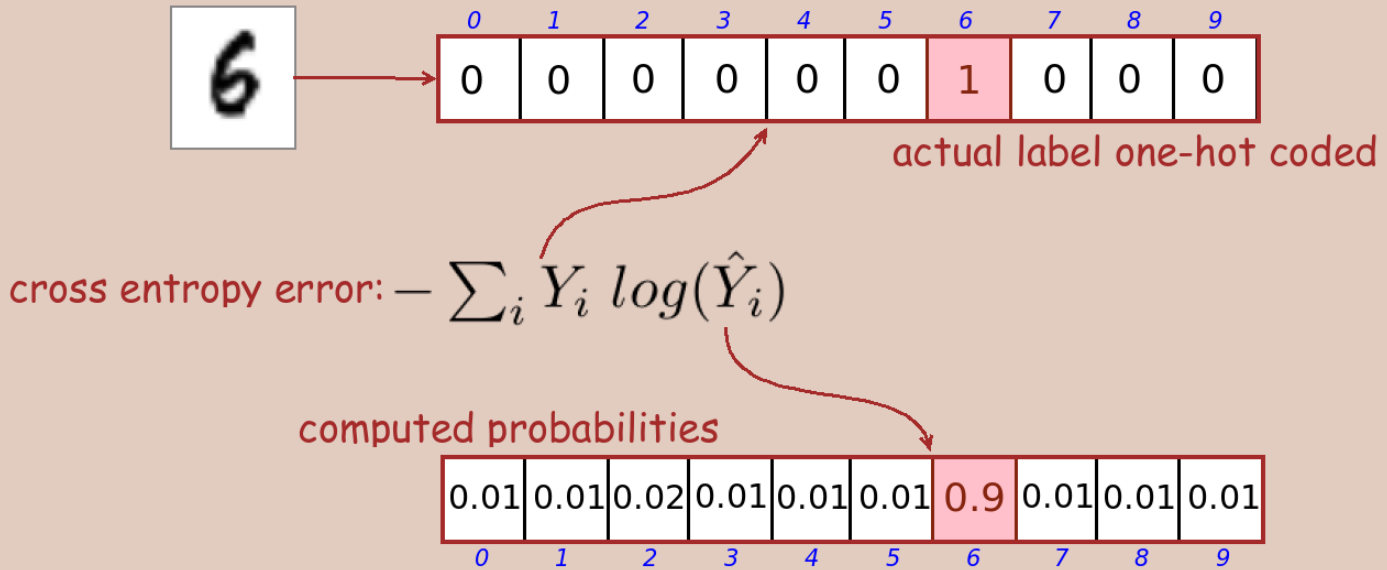
| chiffre | vecteur *one-hot* |
|---|---|
| 0 | [1 0 0 0 0 0 0 0 0 0] |
| 1 | [0 1 0 0 0 0 0 0 0 0] |
| 2 | [0 0 1 0 0 0 0 0 0 0] |
| 3 | [0 0 0 1 0 0 0 0 0 0] |
| 4 | [0 0 0 0 1 0 0 0 0 0] |
| 5 | [0 0 0 0 0 1 0 0 0 0] |
| 6 | [0 0 0 0 0 0 1 0 0 0] |
| 7 | [0 0 0 0 0 0 0 1 0 0] |
| 8 | [0 0 0 0 0 0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

Le codage *one-hot* des labels '0' à '9' donne un vecteur à 10 composantes, comme celui calculé par le réseau de neurones.

# 1 - Réseau de neurones dense

## Fonction d'erreur : *Cross entropy error*

- Une image traitée par le réseau $\rightsquigarrow$ vecteur $\hat{Y}$ de 10 `float` à comparer au codage *hot-one* $Y$ du label de l'image.
- On utilise la fonction d'erreur (ou de perte) *cross entropy* adaptée au codage *one-hot* : $e(Y, \hat{Y}) = -\sum_i Y_i \; log(\hat{Y}_i)$



# 1 - Réseau de neurones dense

## Optimisation et *Back Propagation*

- Pendant la phase d'apprentissage un algorithme d'optimisation calcule le gradient de la fonction d'erreur par rapport aux poids du réseau.

# 1 - Réseau de neurones dense

## Optimisation et *Back Propagation*

- Pendant la phase d'apprentissage un algorithme d'optimisation calcule le gradient de la fonction d'erreur par rapport aux poids du réseau.
- L'algorithme de *Back Propagation* **modifie** les poids du réseau couche par couche grâce au gradient de la fonction d'erreur, en itérant de la dernière couche à la première couche.

# 1 - Réseau de neurones dense

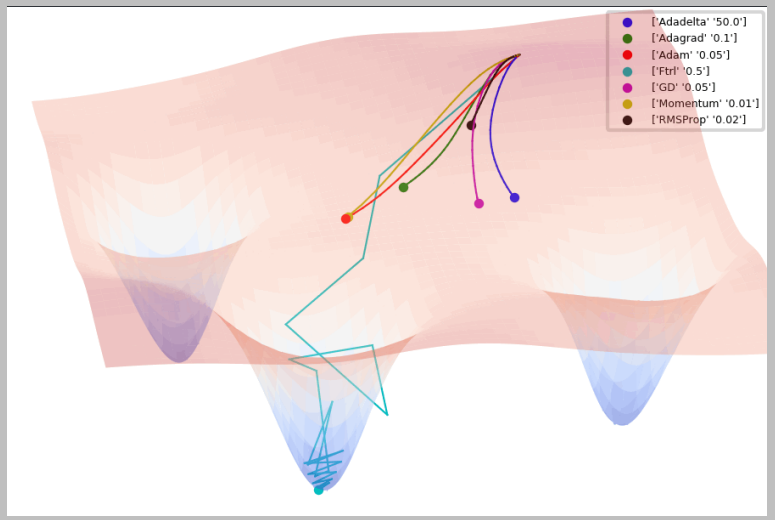
## Optimisation et *Back Propagation*

- Pendant la phase d'apprentissage un algorithme d'optimisation calcule le gradient de la fonction d'erreur par rapport aux poids du réseau.
- L'algorithme de *Back Propagation* **modifie** les poids du réseau couche par couche grâce au gradient de la fonction d'erreur, en itérant de la dernière couche à la première couche.
- Exemples d'algorithme d'optimisation :
  - Descente de Gradient (*Gradient Descent (GD)*)
  - Descente de Gradient Stochastique (*Stochastic Gradient Descent (SGD)*)
  - *Adam* (version améliorée de descente de gradient)...

  Le module tf.keras.optimizers propose l'implémentation Python de plusieurs algorithmes d'optimisation.

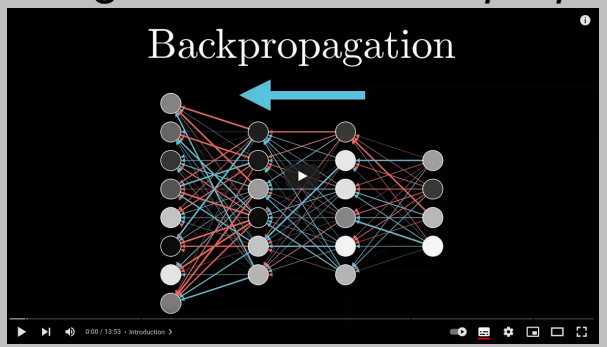
# 1 - Réseau de neurones dense

Visualisation des itérations d'algorithmes de descente de gradient pour une fonction de perte ultra-simple à seulement 2 variables :




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

Vidéo d'explication de l'algorithme de *back propagation* :



# Mise en oeuvre dans l'APP

## 1 – Auto-formation/ Réseau dense

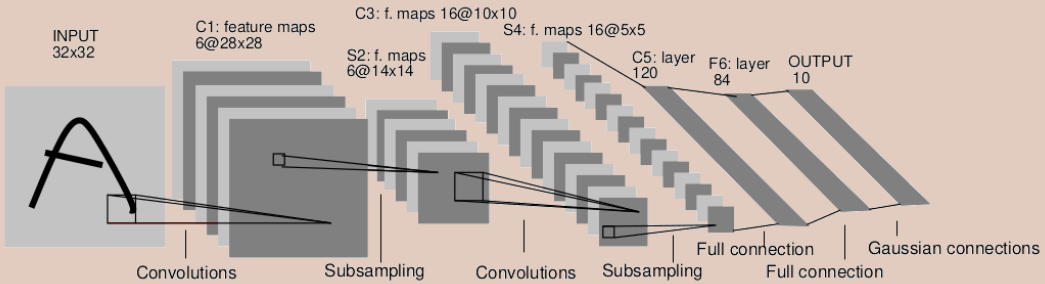
- Les trois *notebooks* `ML1_MNIST.ipynb`, `ML2_DNN.ipynb` et `ML3_DNN_suite.ipynb` visent les savoir-faire :
  - charger et pré-traiter les images du MNIST,
  - construire un réseau de neurones **dense**,
  - entraîner le réseau à reconnaître les images du MNIST,
  - évaluer et exploiter le réseau entraîné.
- Les modules Python utilisés pour créer les réseaux de neurones et les entraîner : tensorflow et keras.
- Les scores obtenus avec des réseaux denses peuvent atteindre 98% de réussite.

# Mise en oeuvre dans l'APP

Améliorer significativement les scores de réussite $\rightsquigarrow$ réseau spécialisé dans le traitement des images : **réseau de neurones convolutifs** *Convolutional Neural Network (CNN)*.

## 2 – Auto-formation / Réseau convolutif

- Le *notebook* `ML4_CNN.ipynb` vise les savoir-faire :
  - construire un réseau de neurones **convolutif** inspiré du réseau **LeNet5** (un des premiers RNC proposé par Yann LeCun *et al.* dans les années 90),




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

  - entraîner le réseau à reconnaître les images du MNIST,
  - évaluer et exploiter le réseau entraîné.

# Mise en oeuvre dans l'APP

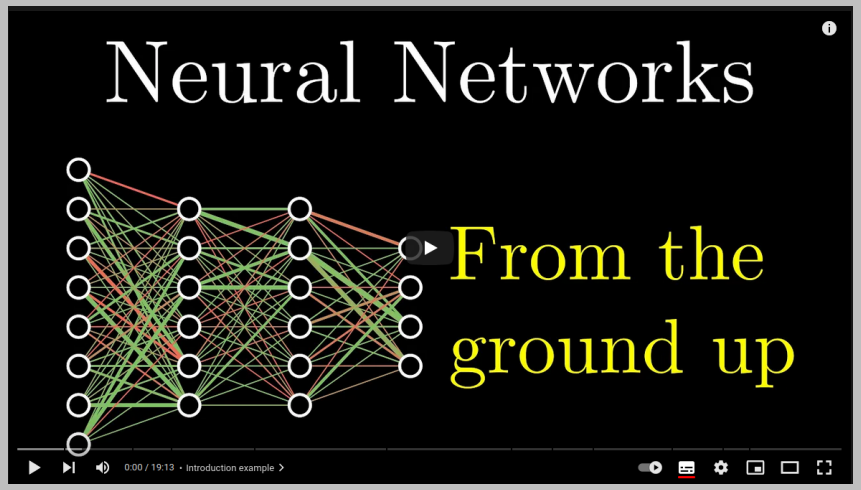
## 3 – Projet : Entraîner un réseau de neurone avec une banque de données spécifique

Pour ce projet d'équipe, les étapes sont :

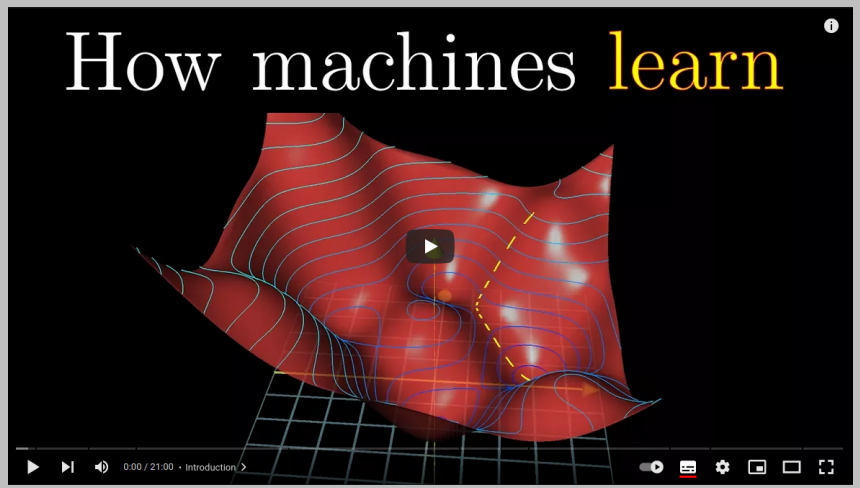
- Choix d'une banque de données spécifique à votre projet à trouver sur Internet (images ou autre...).
- Choix du réseau (dense ou convolutif) à entraîner, en utilisant les acquis d'apprentissage de l'auto-formation.
- Entraînement supervisé du réseau de neurones avec la banque de données choisie, évaluation du réseau entraîné.

# Vidéographie

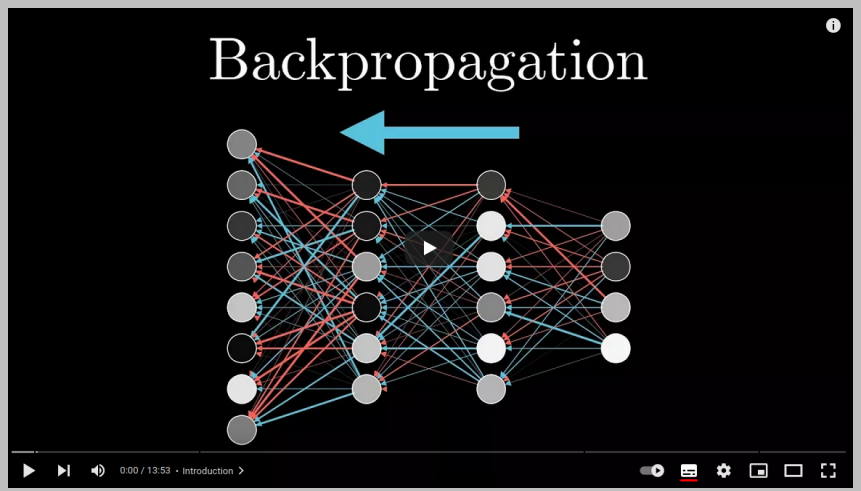

1/ Local: "Le deep learning - YouTube.webm"




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