SCALIAN ACADEMY

Cultivons nos talents
Scalian Academy

DÉMYSTIFICATION DE L'IA

Prise en main du computer vision

COMPÉTENCES VISÉES

SCALIAN

- Avoir une vision technique de ce qu'est l'I.A
- Savoir explorer / comprendre / interpréter un jeu de données
- Savoir entraîner / test / évaluer un modèle de Deep Learning
- Connaître les principales architectures de Deep Learning
- Savoir utiliser la bonne architecture
- Identifier la faisabilité d'un projet de DL
- Connaître les technologies du marché
- Identifier les compétences recherchées

01

INTRODUCTION AU DEEP LEARNING

OBJECTIF PÉDAGOGIQUE

SCALIAN

CONTENU

- Qu’est-ce que le DL
- Pourquoi le DL
- Définition d’un perceptron
- Backpropagation

COMPÉTENCES ATTENDUES

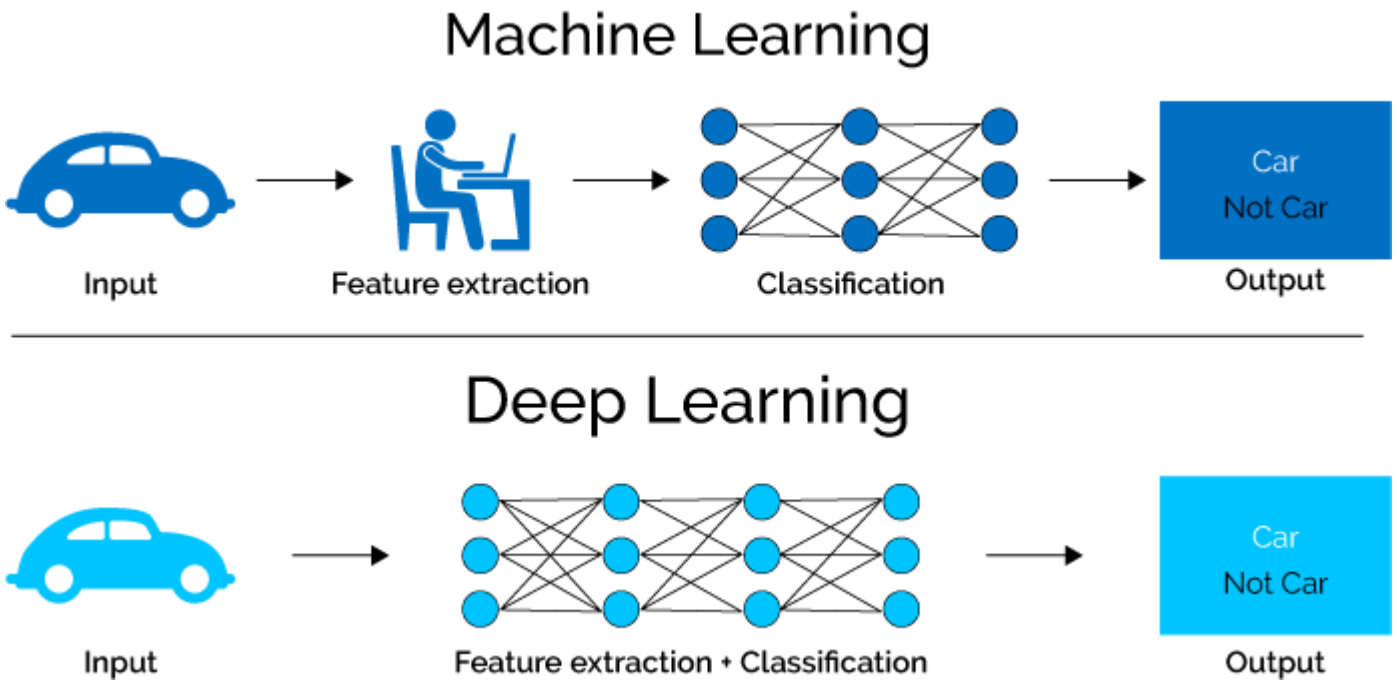
- Définir le Deep Learning (DL)
- Différencier l’approche DL de l’approche ML
- Comprendre la logique de l’apprentissage
- Evaluer un réseau de neurones

DÉFINITION

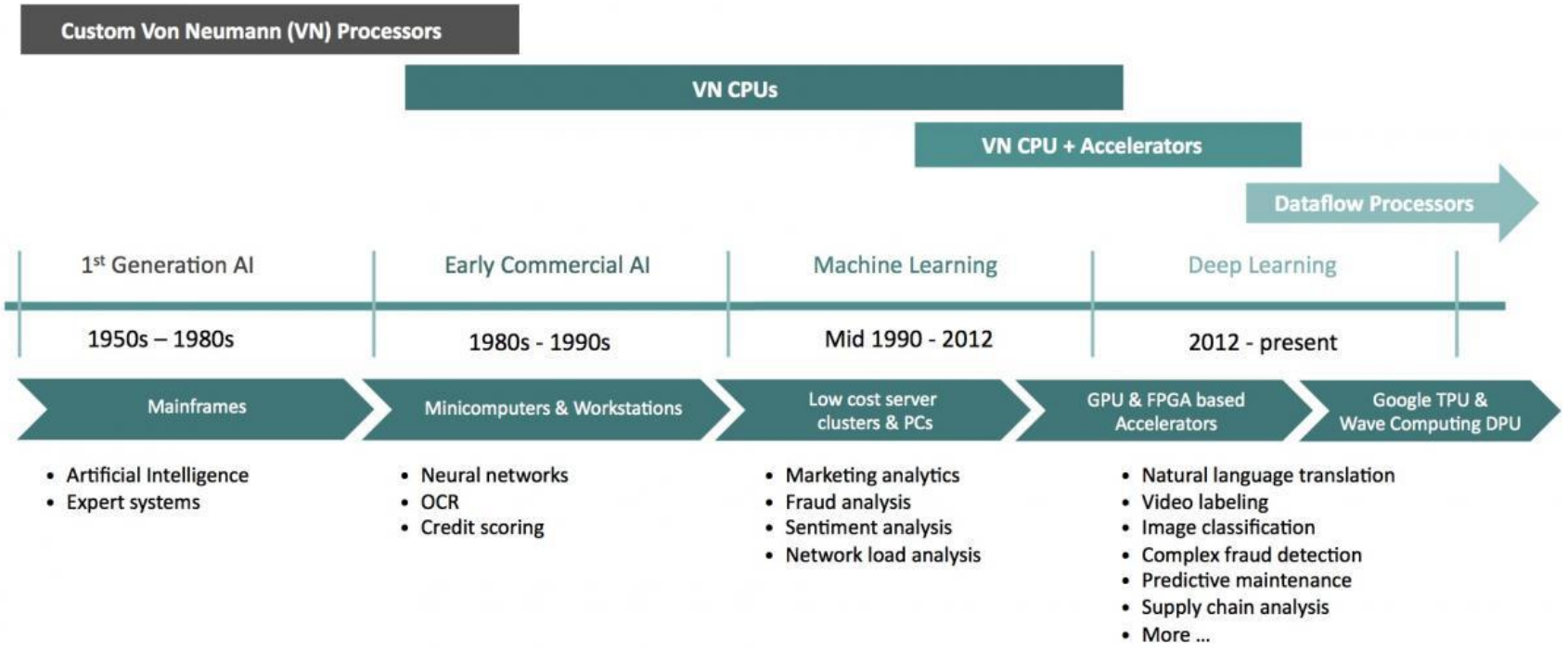
SCALIAN

“L'apprentissage profond est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d’abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.”
Wikipedia

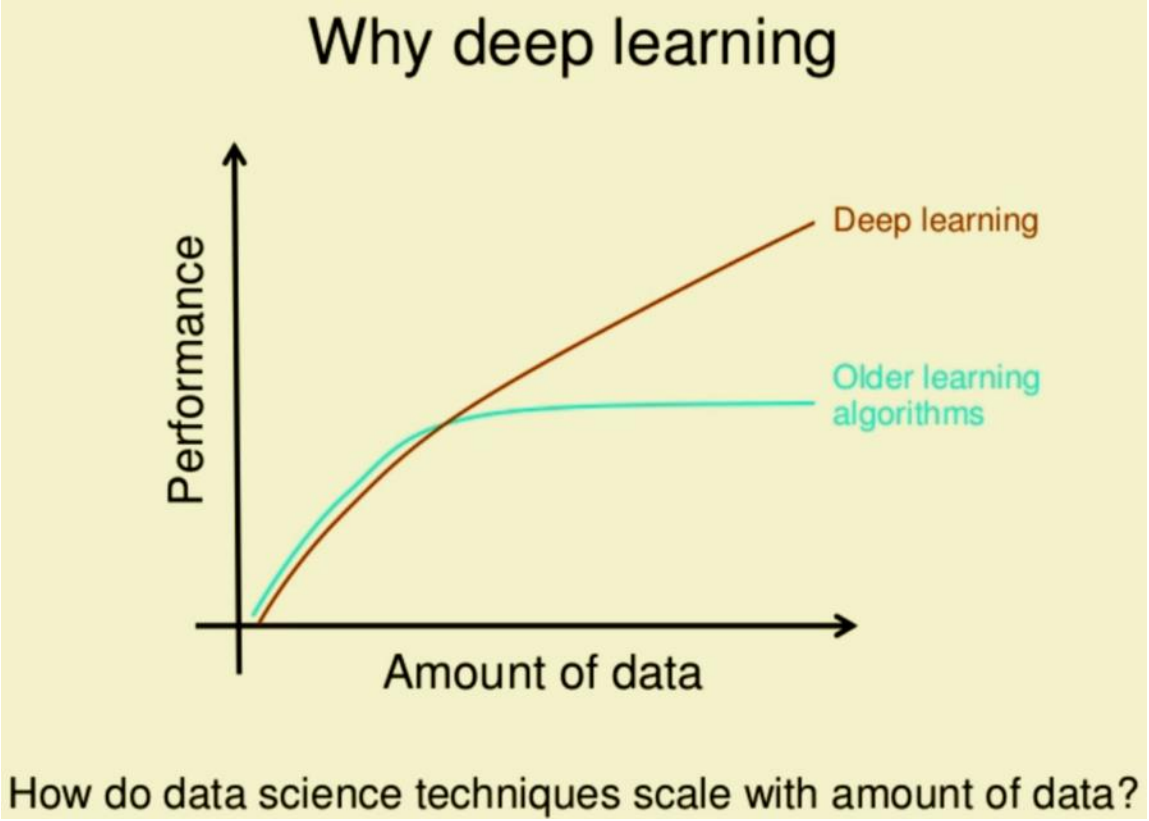
DÉFINITION


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

HISTORIQUE DES MÉTHODES

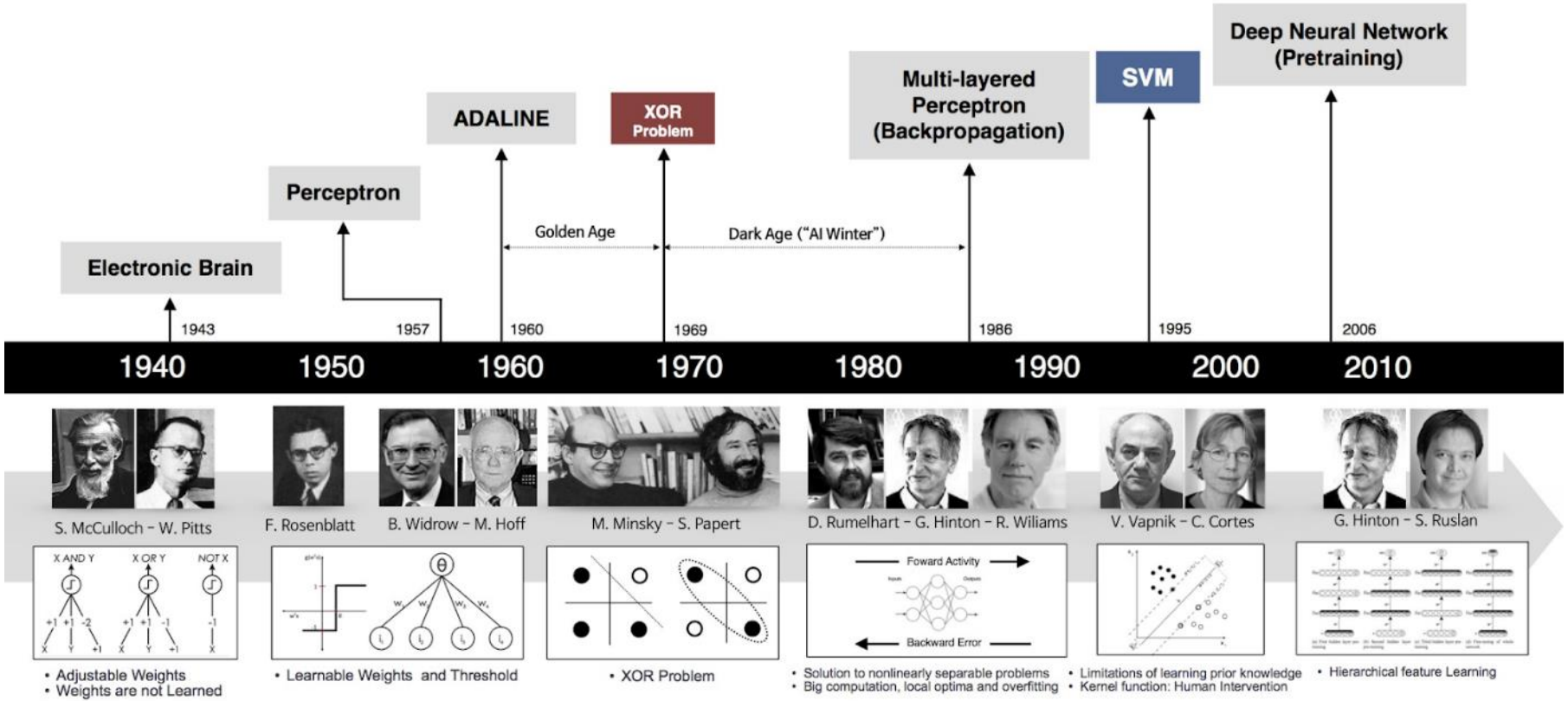


MACHINE LEARNING VS DEEP LEARNING



HISTORIQUE DES RÉSEAUX DE NEURONES

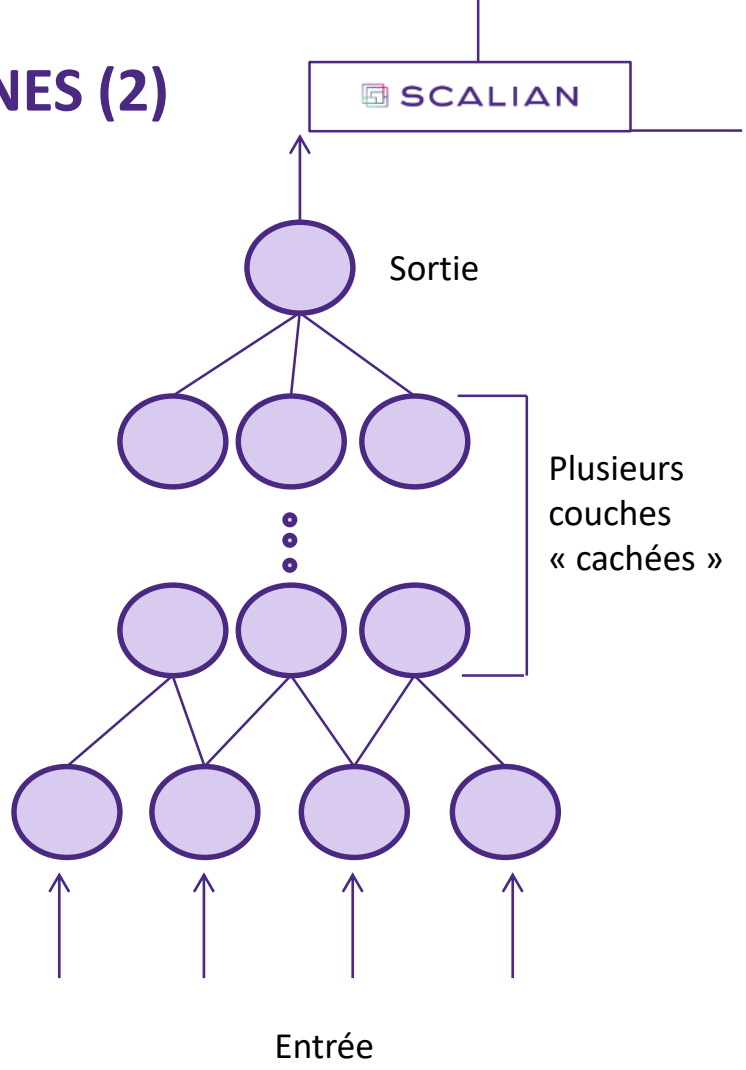
SCALIAN



HISTORIQUE DES RÉSEAUX DE NEURONES (2)

Une histoire ancienne :

- **Premier modèle de Neurone formel en 1943 :**
  - **Pas d'apprentissage**
- **Regle de Hebb en 1949**
  - **Hypothèse sur l'apprentissage des connexions**
- **Le Perceptron (1958)**
  - **Introduction de l'apprentissage**
  - **Problème du « OU » Exclusif et abandon (1969)**
- **Rétro-propagation du gradient (1985)**
  - **Perceptron multi-couches**
  - **Ancêtre du deep learning**



HISTORIQUE DES RÉSEAUX DE NEURONES (3)

SCALIAN

Des succès dans de nombreux domaines à partir des années 2000 :

- **Premiers succès d'un Long short-term Memory (LSTM) pour la reconnaissance du dialogue (1998)**

- **Entrainement effectif de réseaux avec de très nombreuses couches (2006)**
  - **Début du deep learning avec des réseaux de plus en plus profond**

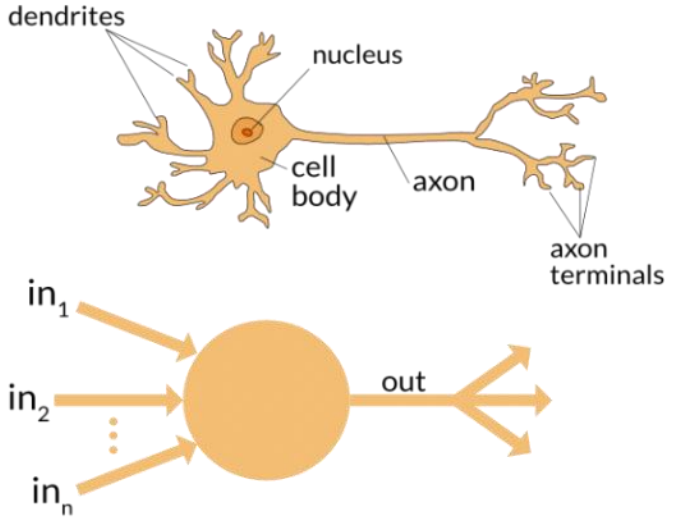
- **Les réseaux de neurones convolutifs deviennent state-of-the-art en classification d'images (2012).**

- **Alphago bat le champion du monde de Go en mai 2017**

QU'EST-CE QU'UN RÉSEAU DE NEURONES ?

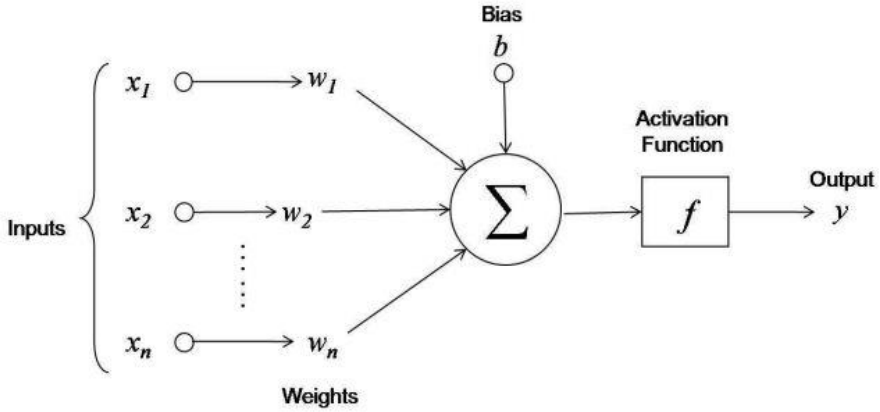
UN PEU DE VOCABULAIRE

- **Algorithme de Machine Learning inspiré du fonctionnement du cerveau**
- **Composé d'un ensemble d'unités appelées « neurones » fonctionnant en parallèle**
- **Ces neurones sont reliés entre eux par des connexions (« synapses »)**



QU'EST-CE QU'UN RÉSEAU DE NEURONES ?

UN PEU DE VOCABULAIRE



Pré-activation :

$$a(x) = b + \sum_i w_i x_i$$

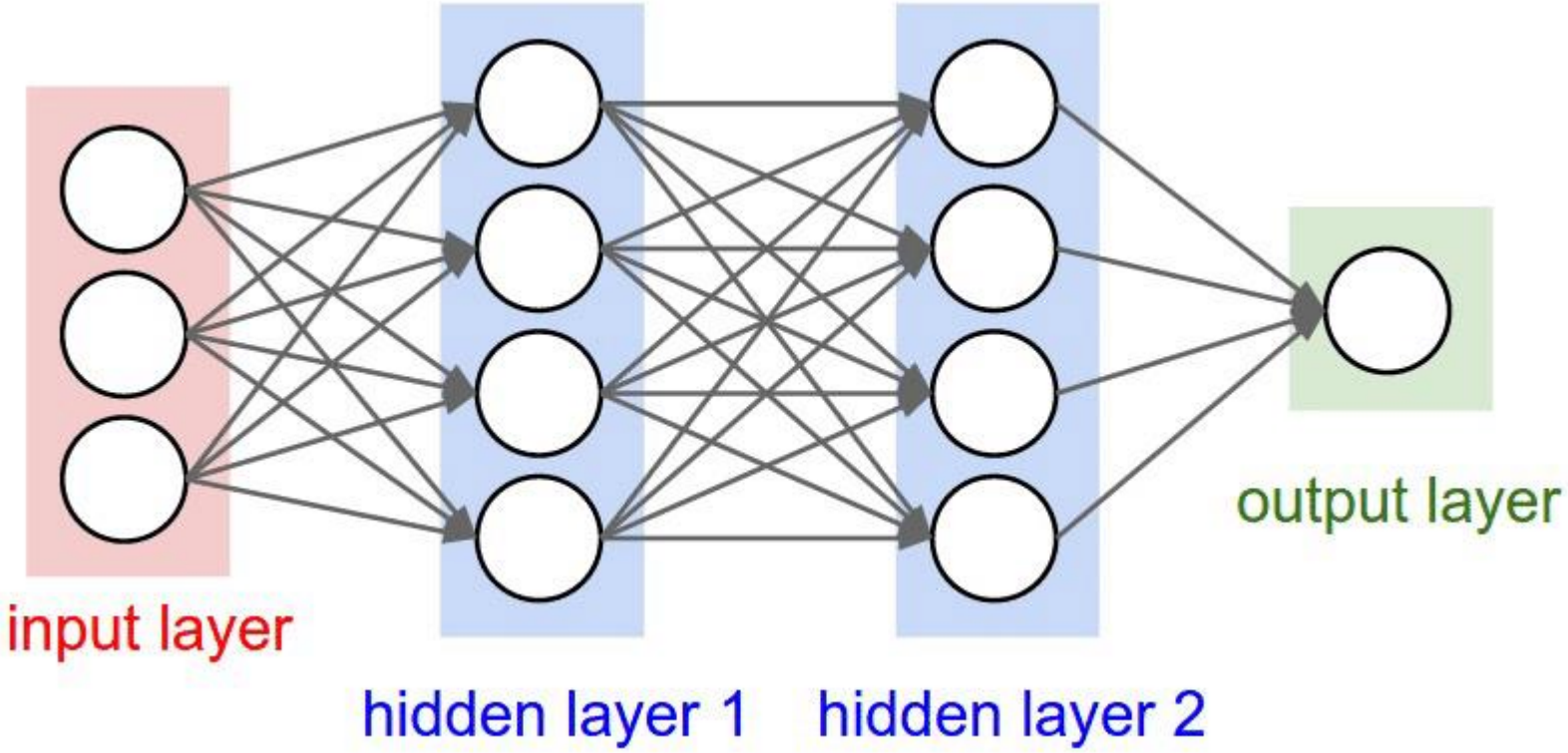
Activation (output) :

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

- **w** => poids
- **b** => biais
- **g(.)** => fonction d'activation

- **Le réseau apprend en faisant varier les valeurs des connexions, ou « poids »**
- **Le terme « Deep learning » désigne les réseaux de neurone avec un nombre important de couches**

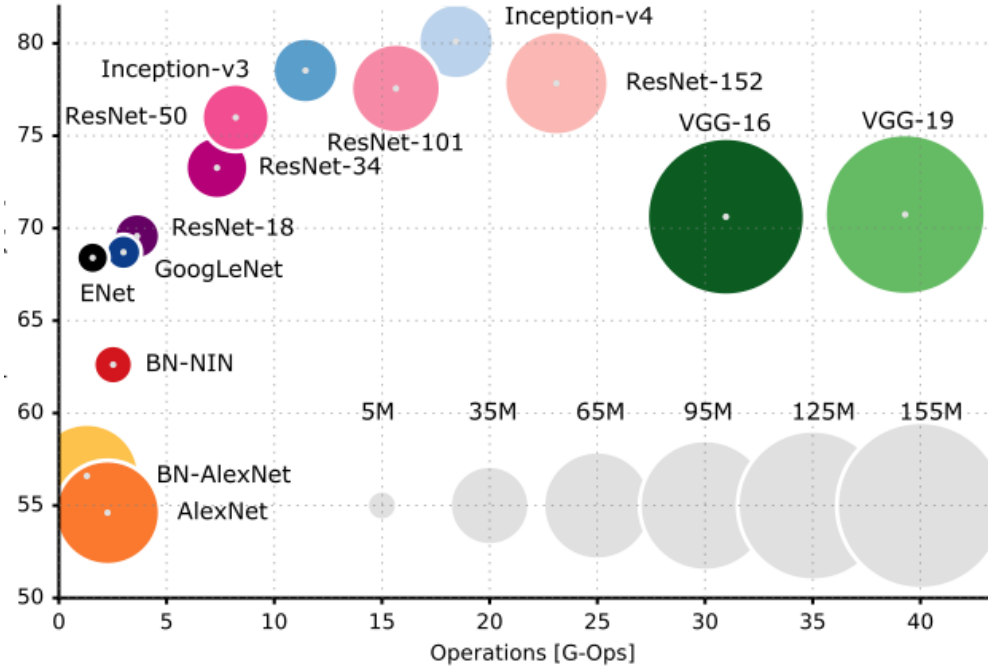
VOCABULAIRE



UN EXEMPLE D'ÉVOLUTION DES RÉSEAUX DE NEURONES

LE CHALLENGE IMAGENET

- **En 2011, 25% d'erreurs**
- **Envahi par les réseaux de neurones depuis 2012**
- **16% d'erreur par un réseau de convolution en 2012**
- **Meilleur que l'humain depuis 2015**
- **… mais sur moins de catégories**



FONCTIONNEMENT

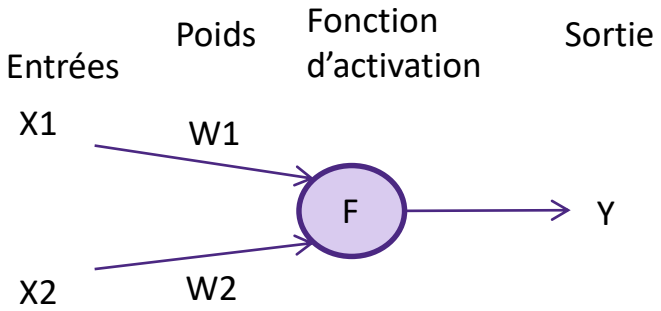
LE PERCEPTRON SIMPLE

- **Perceptron simple = Un neurone**
- **La sortie est une fonction des poids et des entrées**

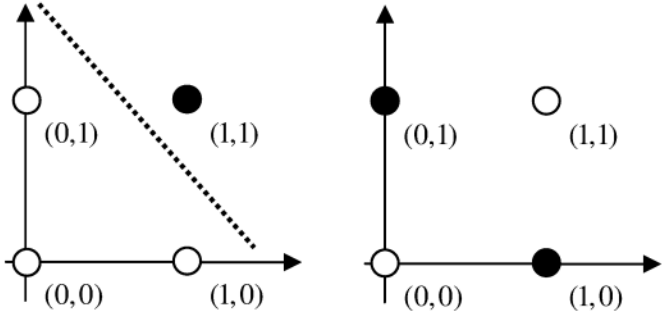
  $Y = f(W1\ X1 + W2\ X2)$

- **Ne peut pas résoudre le « Ou exclusif »**
- **En pratique, les calculs sont fait couche par couche avec des opérations matricielles**
  - **+ efficace, notamment par GPU**

Le perceptron simple



Le « Ou exclusif »



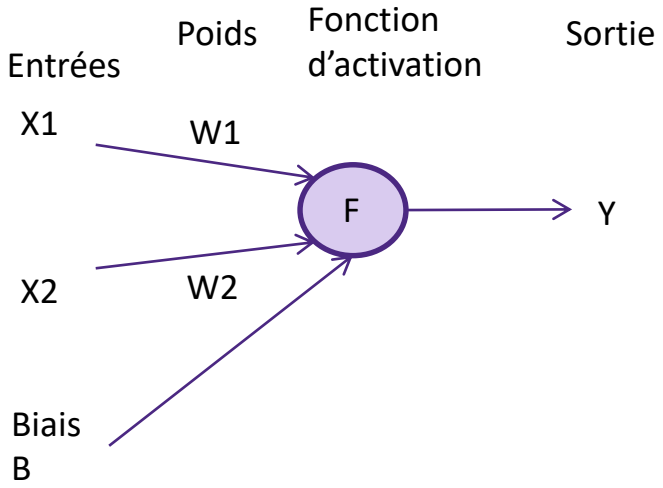
Réponse d'un neurone

LE BIAIS

- **Ajout d'un biais sur les neurones**

$$Y = f(W1\ X1 + W2\ X2 + B)$$

- **Permet de décaler la fonction d'activation**
- **Les poids changent la pente de la fonction**
- **Permet une réponse du neurone même si les entrées sont vides**



FONCTIONNEMENT

LES FONCTIONS D'ACTIVATION

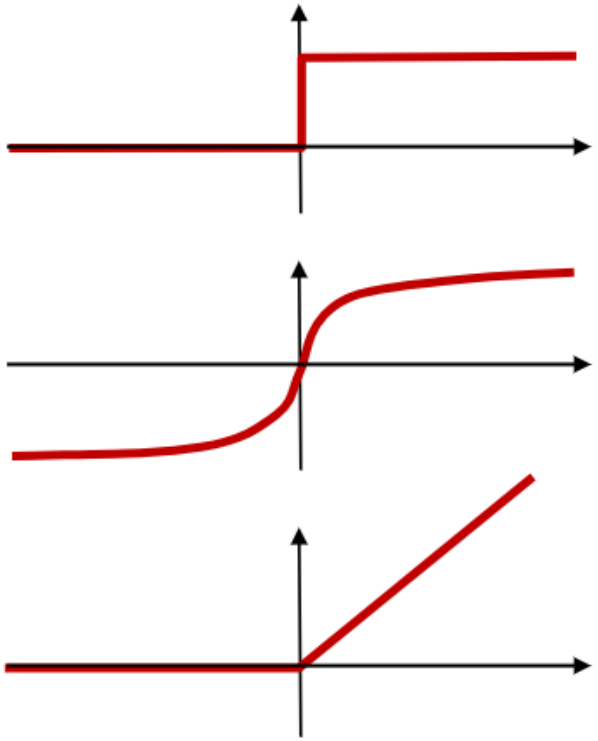
Différentes fonctions d'activation.

Comment les choisir ?

- **Fonction Heaviside :**
  - **Pour prendre une décision**
  - **Ex: pour la couche de sortie du réseau**

- **Sigmoïde (et Linéaire) :**
  - **Evite que l'activité dans les réseaux n'explose**
  - **Problème de disparition du gradient de l'erreur**
  - **Adaptée si peu de couches**

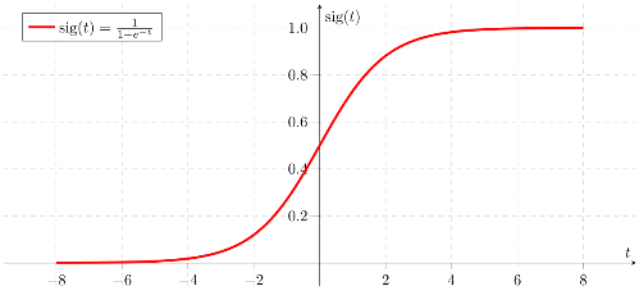
- **Rectified Linear Unit (ReLU) :**
  - **Pas de problème de disparition du gradient**
  - **« Sparse »**
  - **State-of-the-art dans des réseaux à nombreuses couches.**



FONCTION D'ACTIVATION - SORTIE

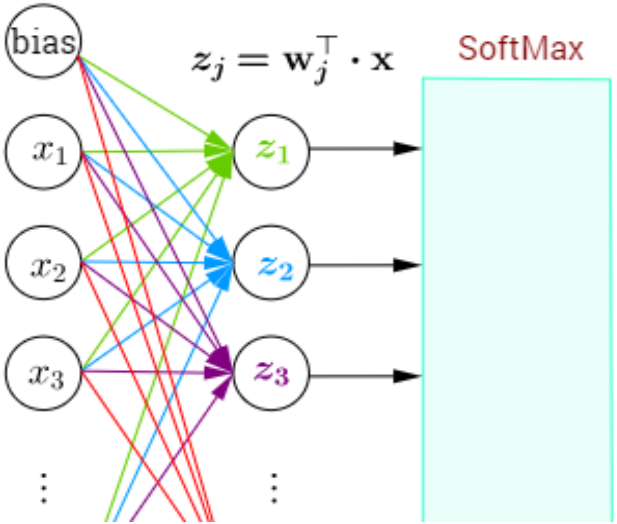
SCALIAN

Binaire



Sigmoid : $\sigma(x) = \frac{1}{1+e^{-x}}$

Multi-classes



Softmax : $p(y = j|\mathbf{x}) = \frac{e^{(\mathbf{w}_j^T \mathbf{x}+b_j)}}{\sum_{k\in K} e^{(\mathbf{w}_k^T \mathbf{x}+b_k)}}$

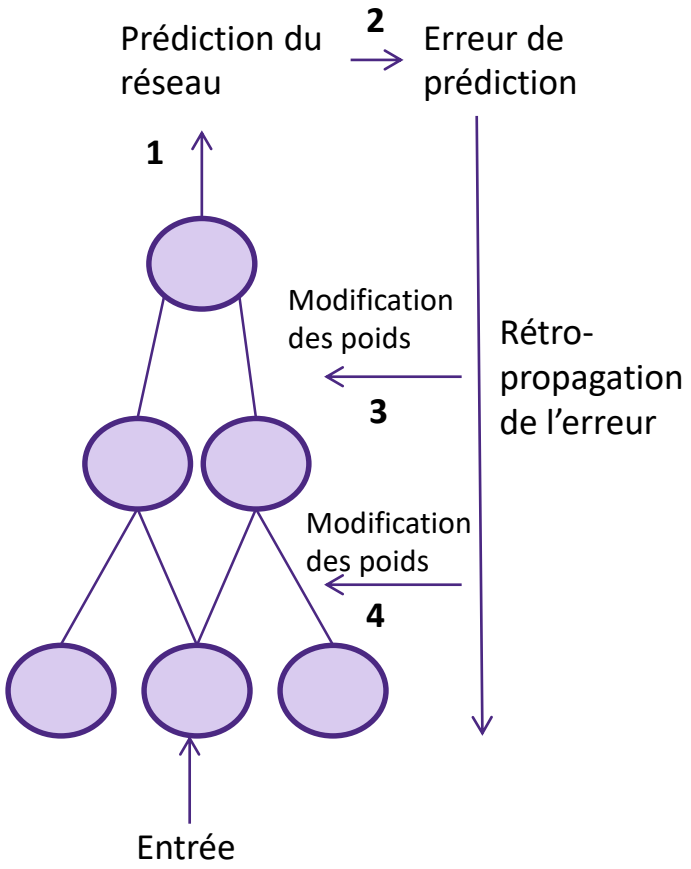
L'APPRENTISSAGE

Objectif : minimiser l'écart entre la réponse du réseau et la réponse voulue en faisant varier les poids du réseau.

Comment savoir quel poids bouger ?

- **Algorithme de rétro-propagation du gradient de l'erreur**
  - **Permet de déterminer les nouveaux poids par calcul de dérivée**
  - **Intuitivement : la dérivée permet de voir quel effet aurait la variation de chaque poids sur la réponse finale.**

- **Différents algorithmes de descente du gradient**
  - **Appelés « Optimizer »**
  - **Différences de performance pour éviter les minima locaux.**



DÉMO

SCALIAN

tf playground

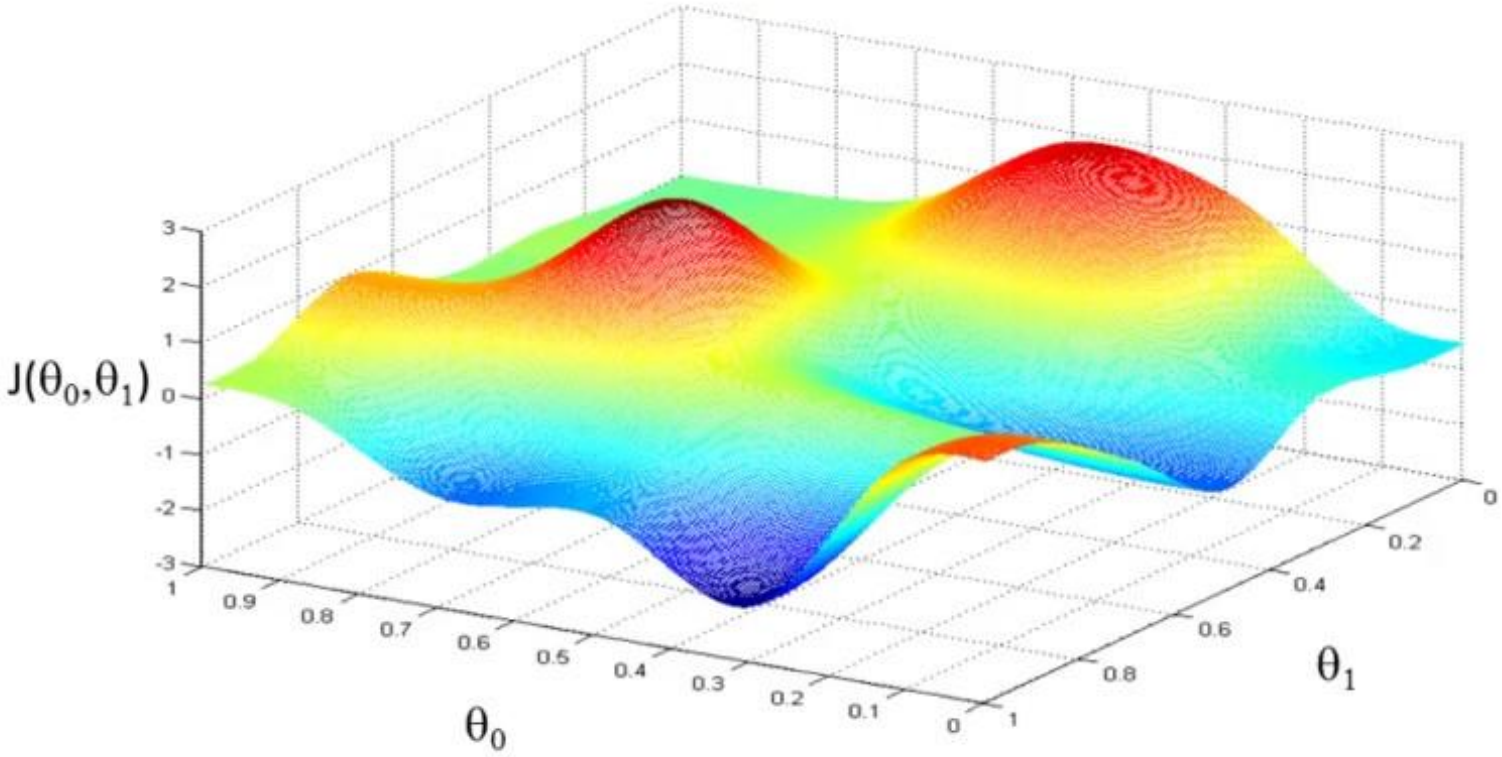
COST FUNCTION

$$J(\theta) = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n))^2$$

=> Objectif : Minimiser la Cost function

COST FUNCTION

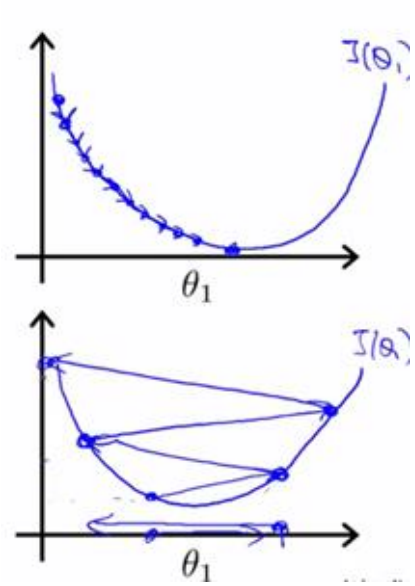
SCALIAN



DESCENTE DE GRADIENTS

$$J(\theta) = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n))^2$$

Dérivée partielle : $$\frac{\partial J(\theta)}{\partial \theta_1} = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n)) x_n$$

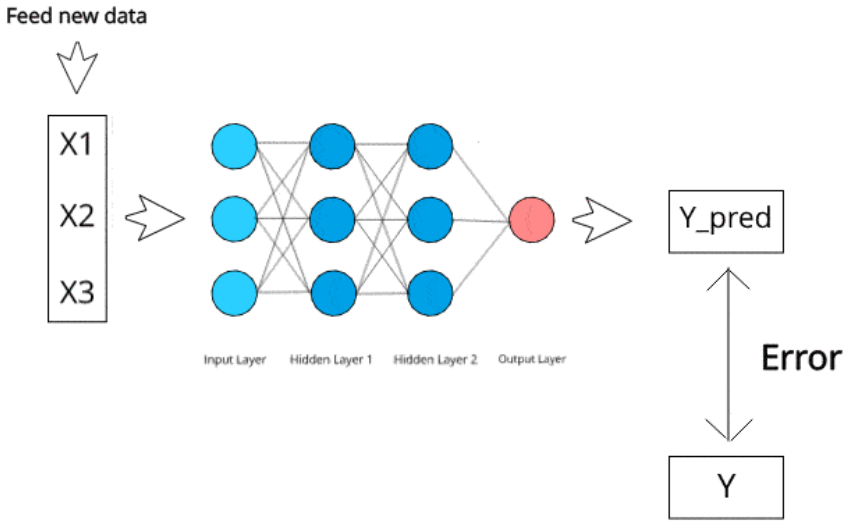


Pour i allant de 1 à nombre_choisi :

$$\theta_1 = \theta_1 - \alpha \frac{\partial J(\theta)}{\partial \theta_1}$$

Avec $\alpha > 0$, le pas d'avancement

DÉRIVATION DES FONCTIONS COMPOSÉES



Idée :

- Grâce au théorème de la dérivée en chaîne, on peut calculer plus facilement les dérivées de fonctions composées
- Un réseau de neurones est constituées de fonctions composées
- On calcule l'erreur de chaque couche de la dernière à la première
- On utilise l'erreur pour calculer les dérivées partielles
- On met à jour les poids du réseau (Learning rate)

ALGORITHME

Initialiser les poids du réseau

Initialiser $\Delta w_{kj} \leftarrow 0, \Delta w_{ji} \leftarrow 0$

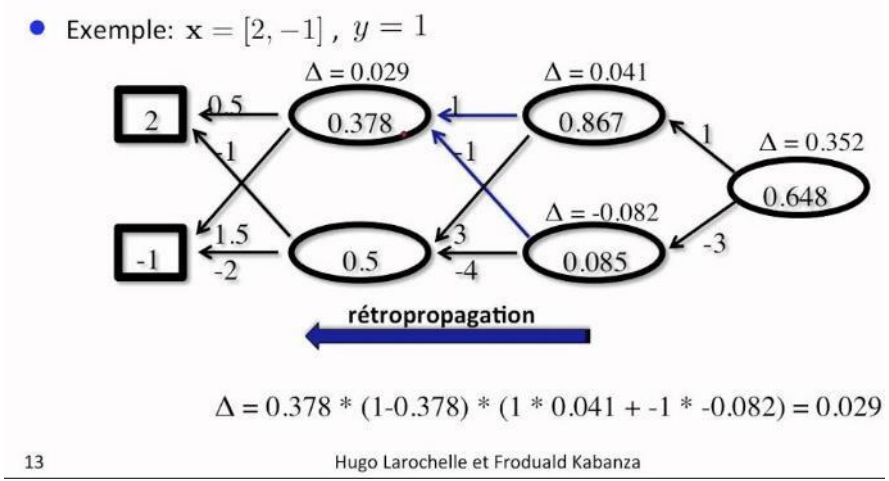
Répéter jusqu'à terminaison

Pour chaque exemple d'apprentissage faire

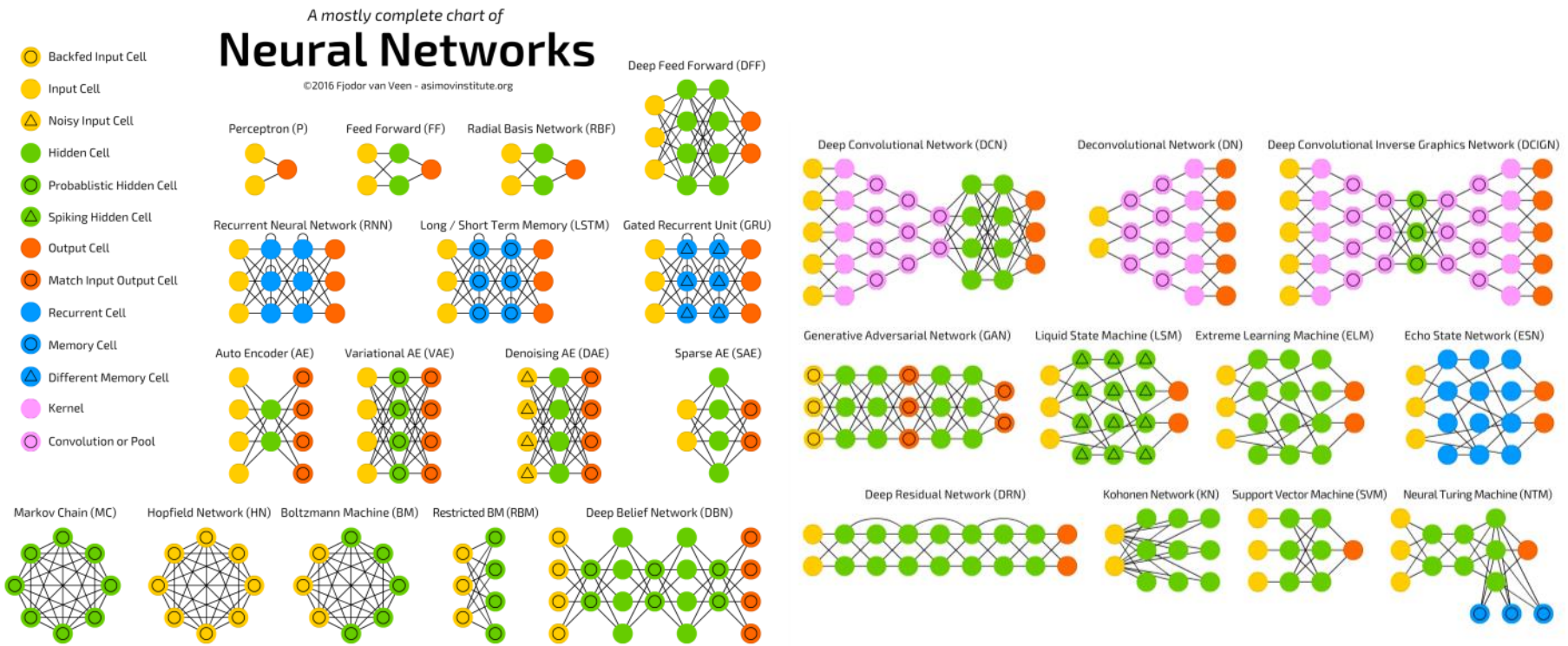
1. *Appliquer le réseau et calculer les sorties*
2. *Calculer et cumuler les deltas*
   - *Pour chaque unité de sortie* k
     Calculer δ_k; $\Delta w_{kj} \leftarrow \Delta w_{kj} - \eta \delta_k z_j$
   - *Pour chaque unité cachée* j
     Calculer δ_j; $\Delta w_{ji} \leftarrow \Delta w_{ji} - \eta \delta_j x_i$

Ajuster les poids

- $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$
- $w_{kj} \leftarrow w_{kj} + \Delta w_{kj}$



LES DIFFÉRENTES ARCHITECTURES DE RÉSEAUX DE NEURONES



Source : http://www.asimovinstitute.org/neural-network-zoo/

LES DIFFÉRENTES ARCHITECTURES DE RÉSEAUX DE NEURONES

SCALIAN

La liste précédente est très loin d'être exhaustive !

Quelques grandes familles que nous explorerons plus en détail demain :

- **Les réseaux de neurones convolutifs (CNN)**
- **Les réseaux de neurones récurrents (RNN)**
- **Les auto-encodeurs**
- **Les GANs**

ENTRAINER ET ÉVALUER SON RÉSEAU

ENTRAÎNER SON RÉSEAU DE NEURONES

SCALIAN

Les bonnes pratiques d'un projet de Machine Learning :

- **Isoler le jeu de test des création du jeu de donnée, si possible sur une machine séparée.**
- **Ne pas prendre de décision sur le jeu de test, uniquement pour confirmation !**
- **Les jeux de validation et de tests doivent coller le plus près aux exemples réels**
  - **Proportions respectées avec la réalité**
  - **Exemples les plus proches des cas réels**
- **Equilibrer les exemples positifs/négatifs dans le jeu d'entraînement mène généralement à de meilleurs performances.**
  - **Undersampling/oversampling/smote**
  - **La meilleure méthode dépend souvent du cas d'usage**
- **Sauvegarder les données après pré-processing si le pré-processing est couteux en calcul**

ENTRAÎNER SON RÉSEAU DE NEURONES

SCALIAN

Quelques particularités liés aux réseaux de neurones :

- **Si peu de données, du feature engineering donnera de meilleurs résultats qu'un algorithme de type deep learning.**

- **Beaucoup de paramètres (architecture/apprentissage/…)**
  - **On ne peux généralement pas faire de méthode automatique sur tous ces paramètres**
  - **On se limite souvent au taux d'apprentissage (voir pas du tout).**
  - **Ajouter un « decay » (décroissance) au taux d'apprentissage améliore presque toujours les performances**

- **L'apprentissage peux être très long**
  - **Certains réseaux mettent des semaines à apprendre sur plusieurs GPU**
  - **Souvent trop long pour faire de la validation croisée.**

- **Il est difficile de « comprendre » ce qui se passe à l'intérieur du modèle**
  - **Intérêt de la visualisation**
  - **Intérêt d'afficher de nombreuses informations sur l'état du réseau en temps réel au cours de l'entraînement**

- **Mettre en concurrence différentes architectures.**

EVALUER UN RÉSEAU DE NEURONE

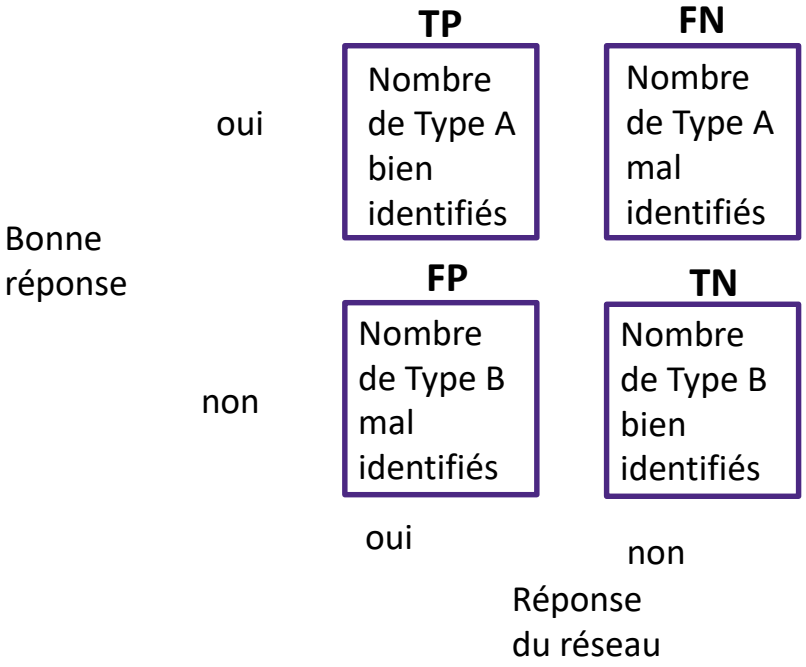
SCALIAN

Utiliser les métriques classiques de Machine learning :

- **Afficher la performance sur le jeu de validation.**
- **L'accuracy (pourcentage de bonnes réponse) ne suffit pas !**
- **La matrice de confusion, la précision, le rappel et l'aire sous la courbe roc sont de bons indicateurs.**
- **La courbe ROC permet de définir le seuil de décision comportant le meilleur compromis par rapport au besoin.**

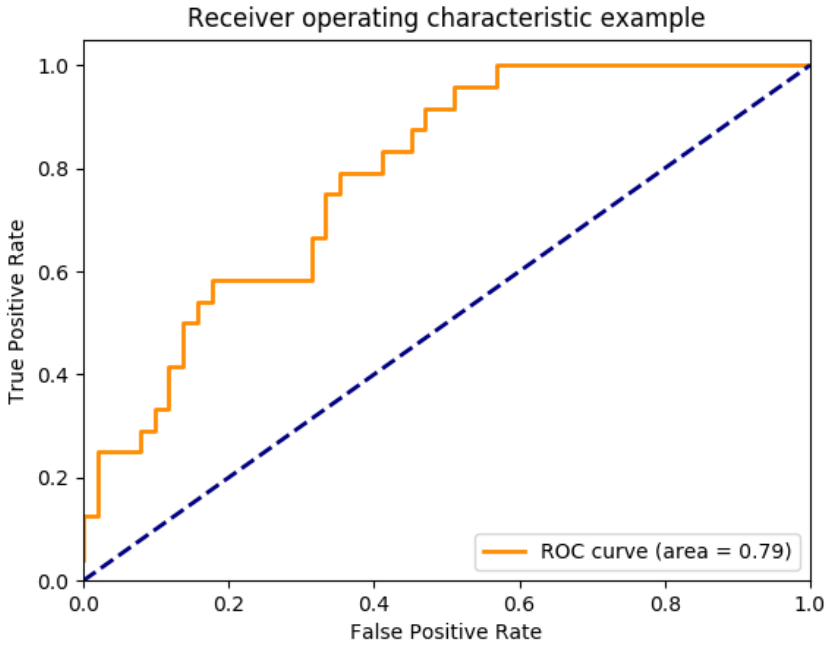
LES MÉTRIQUES

LA MATRICE DE CONFUSION



- **Permet de trouver où se situent les erreurs**
- **Adaptable aux problèmes avec de multiples classes**
- **Permet de calculer la précision TP/(TP+FP) et le rappel TP/(TP+FN)**

LES MÉTRIQUES



Exemple

- **L'aire sous la courbe est une bonne mesure de la performance du réseau**
- **Adaptable aux problèmes avec de multiples classes**
- **Chaque point de la courbe peut être un compromis atteint par le réseau**
- **Fixer le seuil de décision du réseau permet de choisir quel point de la courbe prendre**
- **Seuil souvent ajouté par l'ajout d'une couche/fonction d'activation spécifique au réseau**

LES PROBLÈMES « CLASSIQUES » D'UN RÉSEAU DE NEURONE

SCALIAN

1. **Mon réseau ne converge pas !**

Problème : l'accuracy reste aux alentours du niveau de chance sur les jeux d'entraînement et de validation

Causes potentielles :

- **Les données et les labels sont mal alignés et ne correspondent pas.**
- **Les données sont mal fournies au réseau**
- **L'apprentissage ne fonctionne pas**
  - **Les couches de sorties ne sont pas touchées par l'optimizer**
  - **Les poids des couches ont été initialisés à 0**
  - **Le taux d'apprentissage est trop faible/le temps d'apprentissage trop court**
  - **Changer d'optimizer peut aider**

LES PROBLÈMES « CLASSIQUES » D'UN RÉSEAU DE NEURONE

SCALIAN

1.(bis) Mon réseau ne converge pas !

Problème : La sortie du réseau explose (Nan)

Causes potentielles :

- **Les poids ou les biais initiaux sont trop élevés**
- **Le taux d'apprentissage est trop fort**
  - **Le baisser ou augmenter son decay**
- **Changer de fonction d'activation si le réseau le permet**

LES PROBLÈMES « CLASSIQUES » D'UN RÉSEAU DE NEURONE

SCALIAN

2. Mon réseau a de bonnes performances sur le jeu d'entraînement mais pas sur le jeu de validation

Causes potentielles :

- **Le ratio d'exemples de chaque type est différent dans les jeux d'entraînement et validation**
  - **Modifier les proportions du jeu de d'entraînement (ajout de donnée ou sampling)**

- **Le réseau fait du sur-apprentissage (overfitting) du jeu d'entraînement**
  - **Ajouter de la variation aléatoire dans les données à chaque présentation au réseau (« data augmentation »).**
  - **Normaliser les batchs**
  - **Arrêter l'entraînement plus tôt quand les performances divergent (« early stopping »)**
  - **Régularisation L1 et L2**
  - **Ajouter du « Dropout »**

LES DIFFÉRENTES TECHNOLOGIES DE DEEP LEARNING

SCALIAN

Caffe

- ✓ Plus de communauté
- ✓ Migration Caffee 2

Caffe 2

- ✓ Créer pour la prod
- ✓ Utilisé Facebook, MS, Amazon.
- ✓ Peu de documentation, d' example, de communauté

MXNet

- ✓ Framework de Deep Learning d'Amazon !
- ✓ Structure de données propriétaire, pas compatible avec tout numpy

Torch

- ✓ Ecrit en LUA/C
- ✓ Import des modèles « caffe »
- ✓ Google migre de Torch vers Tensorflow.
- ✓ Twitter est le plus gros utilisateur.

DeepLearning4j

- ✓ Java
- ✓ Gère les formats des différentes techno « caffe, torch, theano, … »

Tensorflow

- ✓ La solution technique la plus plebiscitée.
- ✓ Soutenue par Google.
- ✓ Utilisation en Prod
- ✓ API Keras compatible avec Theano

Watson

- ✓ Impossible de choisir le modèle à utiliser.
- ✓ Uniquement solution Cloud
- ✓ Coûte très cher

Azure Cognitive Services

- ✓ Impossible de choisir le modèle à utiliser.
- ✓ Pas possible d'entrainer un modèle

Google Cloud ML Engine

- ✓ 100 % compatible avec Tensorflow
- ✓ Coût de 1 server GPU 1,62$/heure

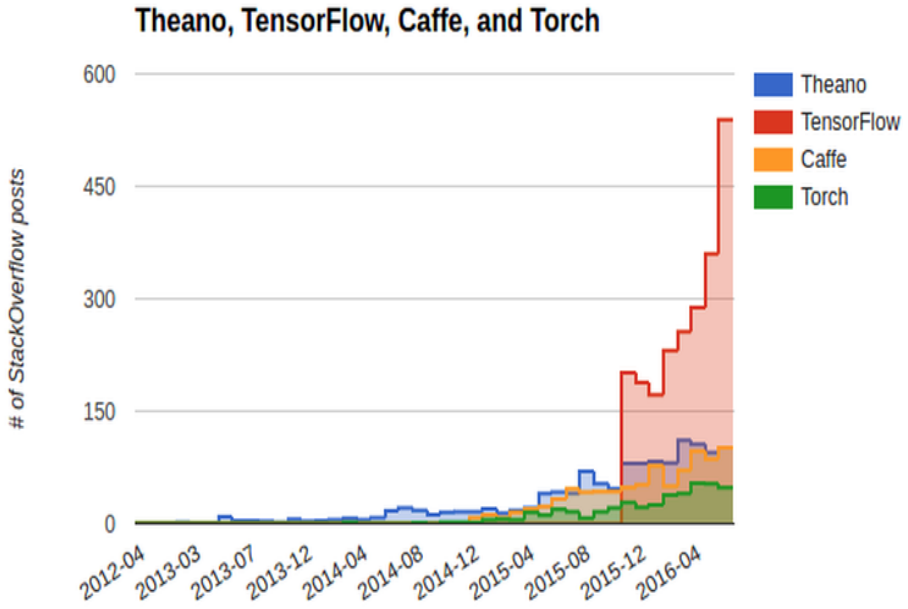
Microsoft Cognitive Toolkit

- ✓ Prometteur
- ✓ Fonctionne uniquement avec des GPU
- ✓ Solution encore jeune

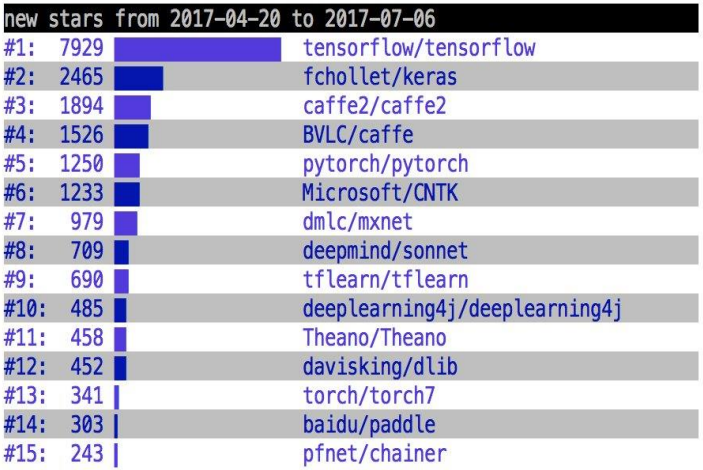
POPULARITÉS

SCALIAN

Nombre de posts sur StackOverflow



Nombre de stars github



CONCLUSION

Les réseaux de neurones sont constitués d'un petit nombre de « briques » et d'architectures.

La majorité des réseaux complexes sont des combinaisons de ces éléments de base.

Ce sont souvent les mêmes problèmes que l'on rencontre lors de la création et l'entraînement de réseaux.

Tensorflow est devenu la référence en bibliothèque de réseaux de neurones.