INTELLIGENCE ARTIFICIELLE EMBARQUÉE

Valentin Gies

Université de Toulon

Intelligence artificielle embarquée

- Contexte Général de l'Intelligence artificielle
  - Rappels sur les neurones
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
  - Generative models
- Contexte de l'AI embarquée
- Quel hardware pour l'AI embarquée ?
  - Architectures classiques
  - Approches neuromorphiques
  - Applications ultra-low power

CONTEXTE GÉNÉRAL DE L'AI

Valentin Gies

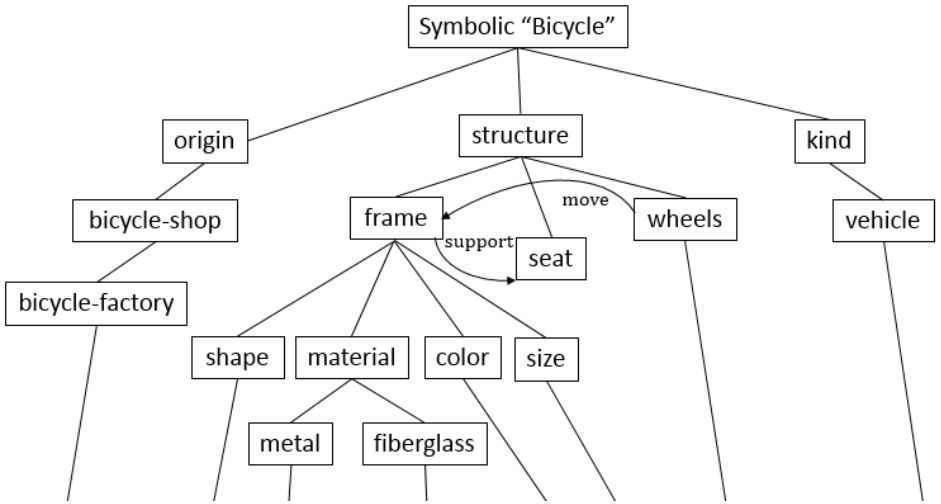
M2 ROC
Université de Toulon

Contexte général de l'AI
Définition

- Qu'est-ce que l'Intelligence artificielle
  - Définition : la capacité d'une machine à avoir certaines compétences d'un humain en termes de performance
    - Reconnaissance d'objets dans une image ou vidéo
    - Compréhension du langage
    - Prise de décisions
    - Résolution de problèmes
    - Elaboration de stratégies
  - Avoir des compétences au-delà de la performance
    - Energétique
    - Vitesse

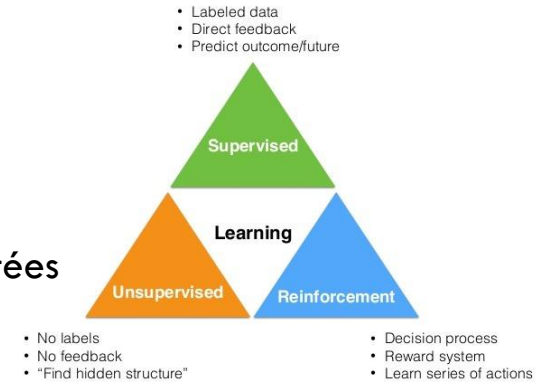
Contexte général de l'AI
Quelle AI pour quelle tâche ?

- Les différents types d'AI
  - GOFAI : Good Old Fashioned AI - Darmouth 1956
    - Approche ancienne – sans apprentissage
    - Basé sur la segmentation des problèmes en taches unitaires
      - Opérations dédiées à une taches et pouvant être complexes
    - Usage intensif des techniques de recherche sur des arbres
      - Limité si le « branching factor » est élevé : trop grande combinatoire
      - Inadapté pour l'image ou la parole.
    - Exemple :
      - Qu'est-ce qu'un vélo ?



Contexte général de l'AI
Quelle AI pour quelle tâche ?

- Les différents types d'AI
  - Qu'est-ce que le Machine Learning (ML) ?
    - Système d'apprentissage générique basé sur un réseau à structure simple mais massive
    - Le système apprend des données, sans programmation explicite
  - 3 catégories principales d'apprentissage
    - Supervised learning
      - Apprentissage sur des données étiquetées
      - Applications : identification de patterns…
    - Unsupervised learning
      - Apprentissage statistique sur des données non-étiquetées
      - Applications : détection d'évènement singuliers…
    - Reinforcement learning
      - Apprentissage de stratégies par essai-erreur-récompense
      - Applications : stratégies, jeu (AlphaGo, DeepBlue)…



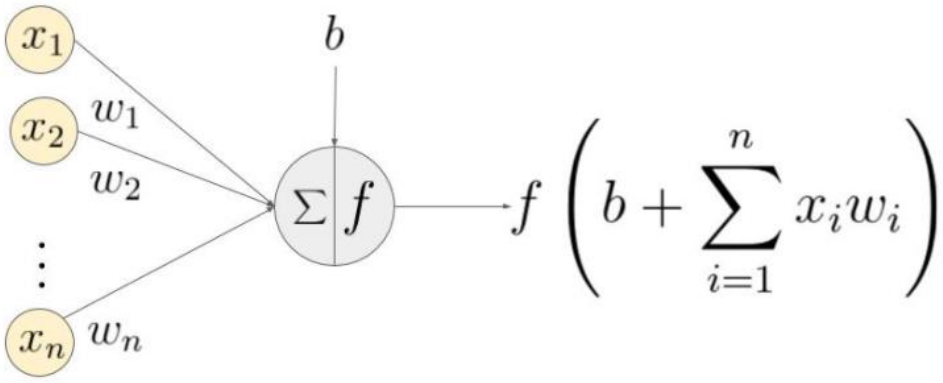
  - Exemples de : https://cs.stanford.edu/people/karpathy/convnetjs/
  - Une extension : réseaux génératifs
    - Génération automatique de données à l'aide de modèles appris.

Contexte Général de l'AI

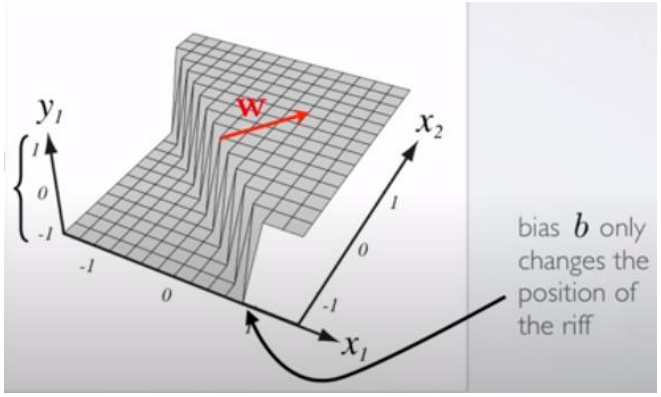
Rappels sur les neurones

Contexte général de l'AI
Rappels sur les neurones

- Neurone classique
  - Somme
    - Entrées pondérées
    - Biais
      - Décalage constant
  - Fonction d'activation
    - Apporte la non-linéarité
    - Partitionne l'espace des vecteurs d'entrée

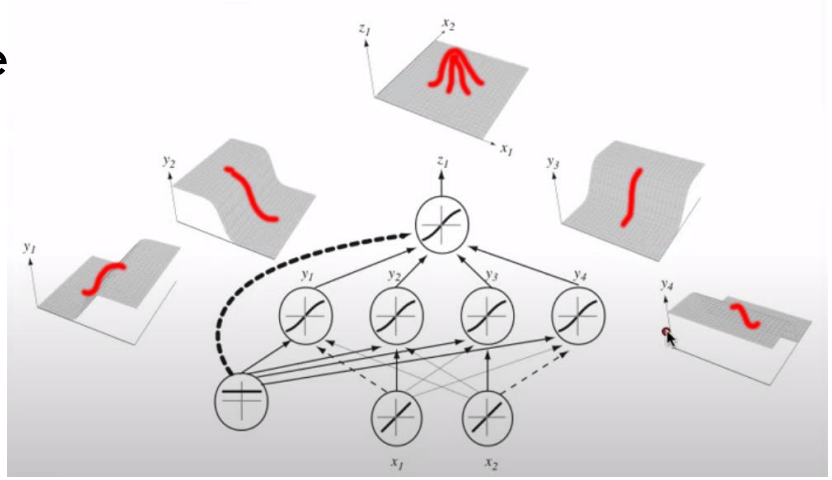


An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.



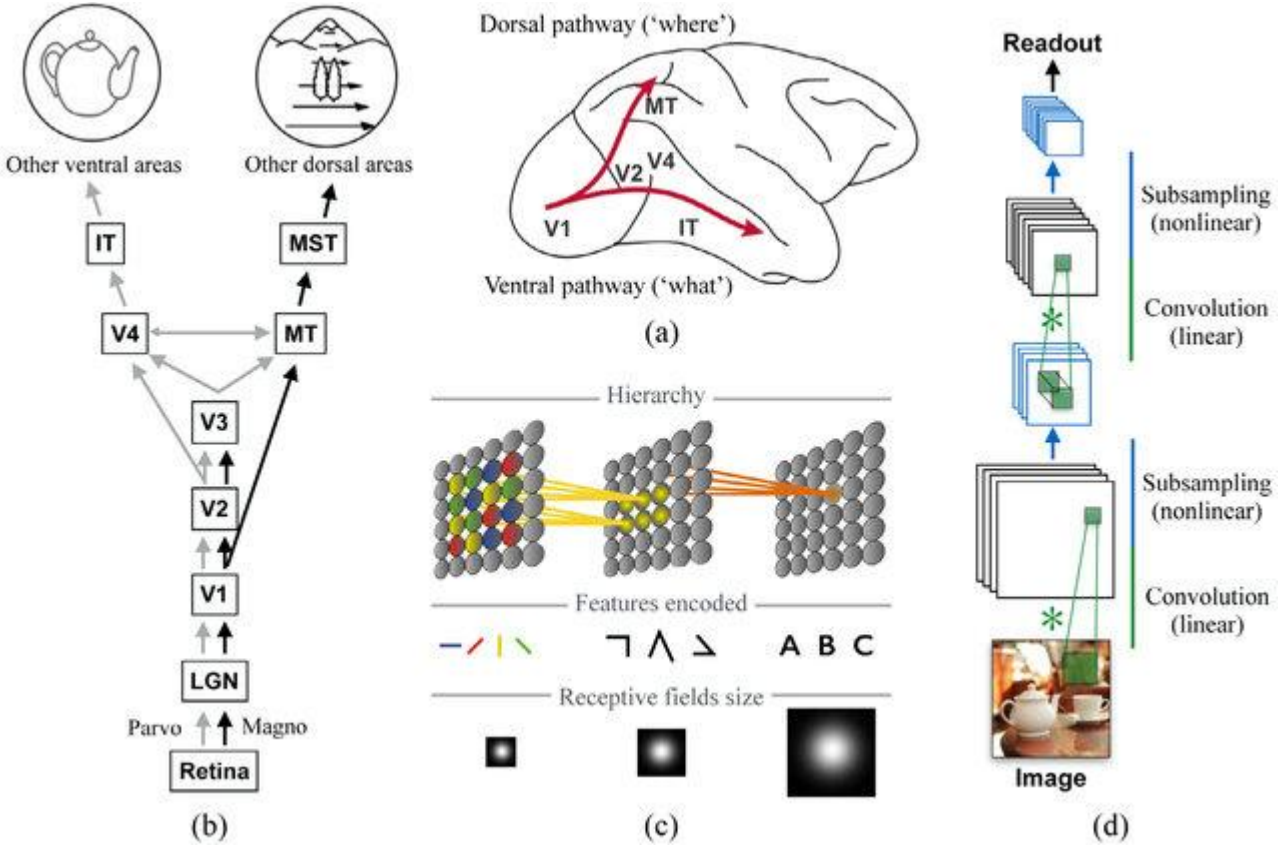
Contexte général de l'AI
Rappels sur les neurones

- Réseau de neurones
  - Couplage de partitions de l'espace permettant d'approximer n'importe quelle distribution de vecteurs
  - Théorème de l'approximation universelle (Hornik 1991)
    - « *Un réseau de neurones à une seule couche interne peut approximer n'importe quelle fonction continue, sous réserve d'avoir suffisamment de neurones* »
    - Ne donne pas de méthode de calcul des poids du réseau



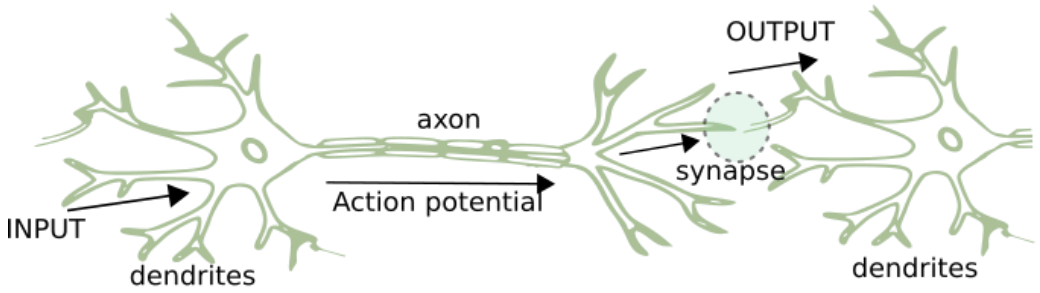
Contexte général de l'AI
Rappels sur les neurones

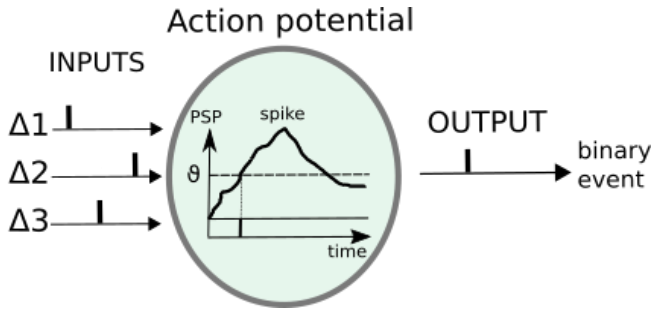
- Les réseaux multicouches : une approche bio-inspirée par la vision dans le cerveau



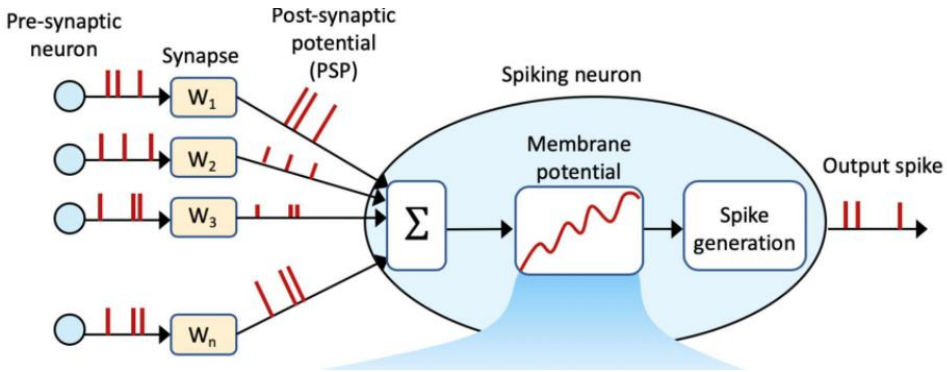
Contexte général de l'AI
Rappels sur les neurones

- Limitates de la bio-inspiration :
  - Dans les neurones réels : trains de spikes en entrée pour déclencher une sortie





  - Il existe des solutions électroniques de ce type : les *spiking neural networks*



Contexte Général de l'AI

Supervised Learning

Contexte général de l'AI
Supervised learning

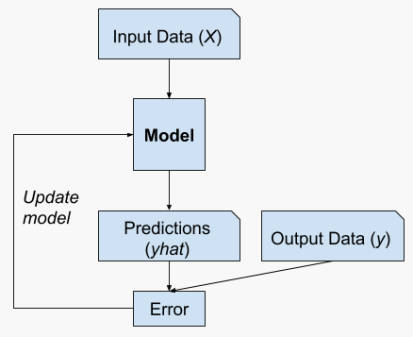
- Apprentissage dans un réseau de neurones
  - Objectif d'apprentissage : Optimiser le maximum de vraisemblance (maximum likelyhood estimation MLE) d'appartenance d'un training set (contenant m data) à un modèle :

$$\theta^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$$= \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right).$$

  - On passe en log :
    - Ok car log est monotone
    - Produit -> somme
    - Evite de multiplier des probabilités très faibles
      - Risque en calcul numérique d'underflow



Contexte général de l'AI
Supervised learning

- Apprentissage dans un réseau de neurones
  - Autre formulation d'objectif d'apprentissage : Minimiser la Kullback-Leibler divergence (D_{KL}) :
    - KL divergence : mesure de la non-similarité de deux distributions de probabilité :
    - L'objectif est de minimiser la non-similarité entre le training set et le modèle

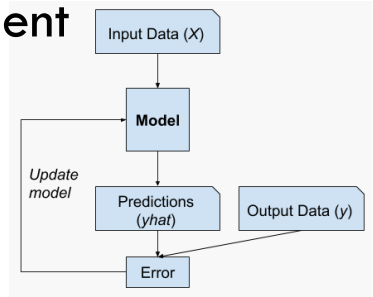
$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}} \left(p_{\mathrm{data}}(x) \| p_{\mathrm{model}}(x;\theta) \right).$$

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right).$$

    - Equivalent à maximiser la MLE (Maximum Likehood Estimation)

Contexte général de l'AI
Supervised learning

- Apprentissage dans un réseau de neurones
  - Objectif pratique en supervisé : minimiser conjointement
    - L'écart entre prédiction et étiquetage
    - La fonction de régularisation

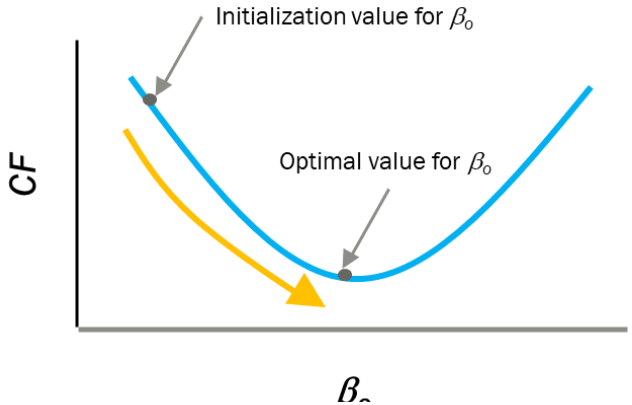


$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) + \lambda \Omega(\boldsymbol{\theta})$$

- θ : ensemble des paramètres du réseau (poids et biais).
- L : fonction de cout dépendant de la prédiction x(t); θ et de l'étiquetage y(t)
  - Plus la prédiction est éloignée de l'étiquetage, plus le cout est élevé.
- $\Omega(\theta)$ est un terme de régularisation
  - Permet d'exclure des cas non souhaités.
- T : taille du training set
- λ : hyper-paramètre permettant de doser l'optimisation entre fonction du cout et régularisation.

Contexte général de l'AI
Supervised learning

- Apprentissage dans un réseau de neurones
  - Méthode : optimisation des poids par descente de gradient

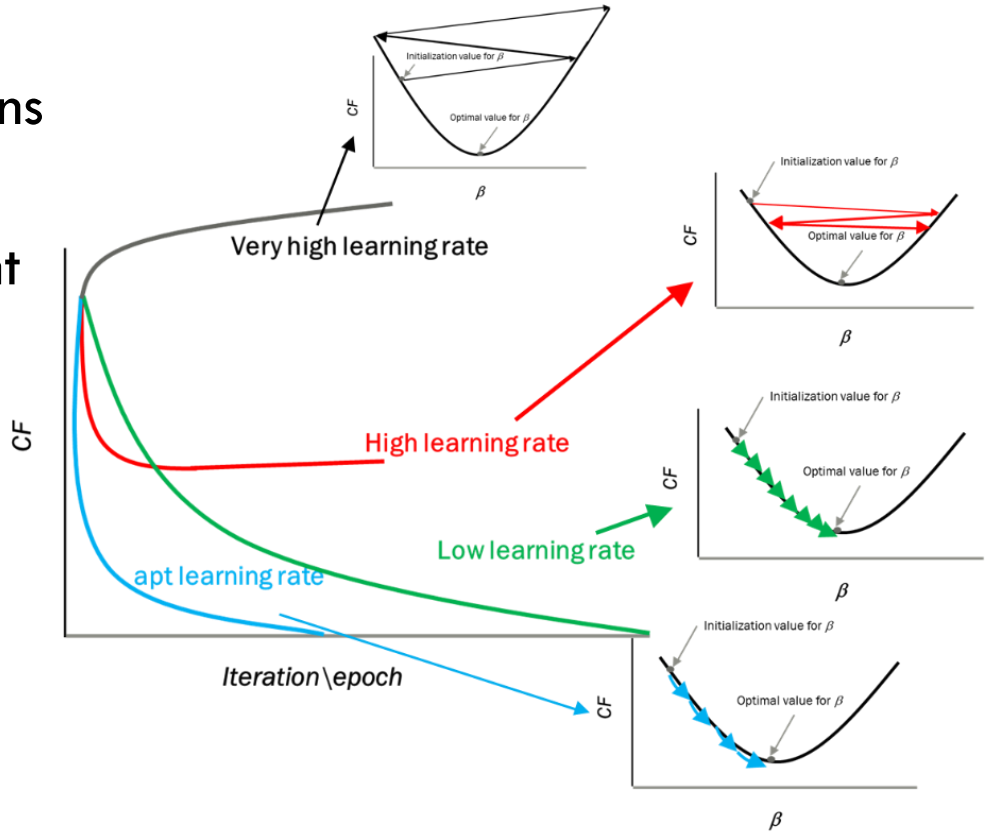


  - Algorithme :
    - initialize $\boldsymbol{\theta}$ ($\boldsymbol{\theta} \equiv \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{W}^{(L+1)}, \mathbf{b}^{(L+1)}\}$)
    - for N iterations
      - for each training example $(\mathbf{x}^{(t)}, y^{(t)})$
        - ✓ $\Delta = -\nabla_{\boldsymbol{\theta}} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$
        - ✓ $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \Delta$

} training epoch = iteration over **all** examples

Contexte général de l'AI
Supervised learning

- Les paramètres d'apprentissage
  - Learning rate
    - Trop grand : oscillations
    - Trop petit : temps de convergence important



Contexte général de l'AI
Rappels sur les neurones

- Apprentissage dans un réseau de neurones :
  - Fonction de cout (loss function)
    - 1ère possibilité : Mean Squared Error (MSE) - distance quadratique entre estimation et étiquetage

$$\underbrace{\text{MSE} = \frac{1}{n}}_{\text{test set}} \sum_{i=1}^{n} \underbrace{(y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i)^2}_{\text{actual value}}$$

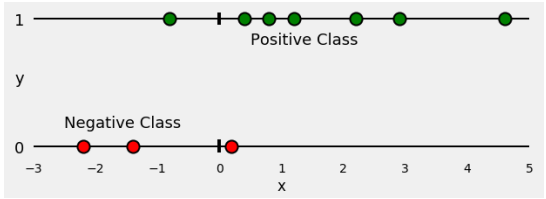
      - Pertinent pour des données continues, pas pour des données binaires.
      - Trop forte pondération des grande valeurs d'erreur.

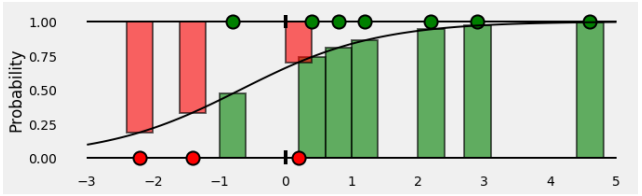
Contexte général de l'AI
Rappels sur les neurones

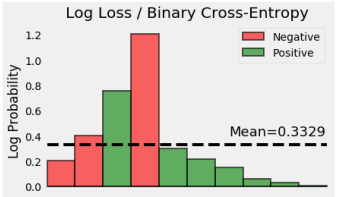
- Apprentissage dans un réseau de neurones :
  - Fonction de cout (loss function)
    - 2e possibilité : Binary Cross-entropy (prédiction binaire)
      - Supposons un tirage vert-rouge
        - y est le label : vert=1 et rouge=0
        - P(y) est la probabilité que le tirage soit vert

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

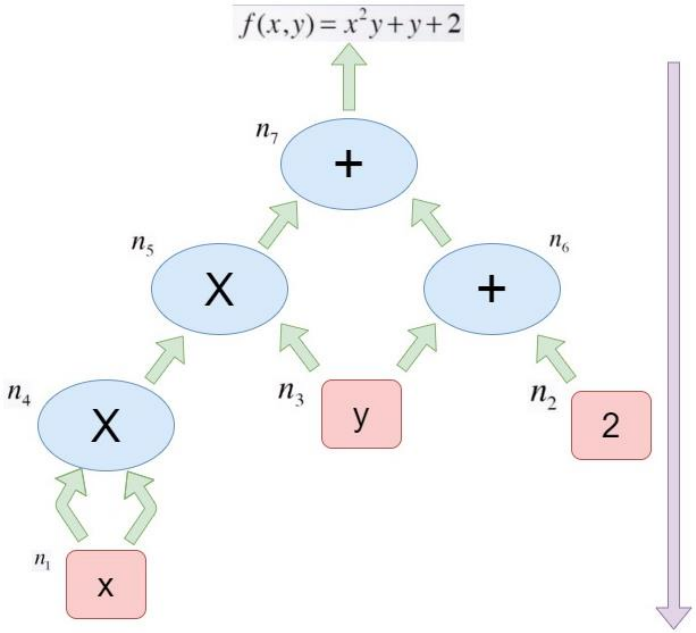






Contexte général de l'AI
Rappels sur les neurones

- Apprentissage par back-propagation :
  - Introduit par Rumelhart, Hinton et Williams en 1986
  - Chain rule : $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_i} \cdot \frac{\partial n_i}{\partial x}$



$$\frac{\partial f}{\partial n_7} = 1$$

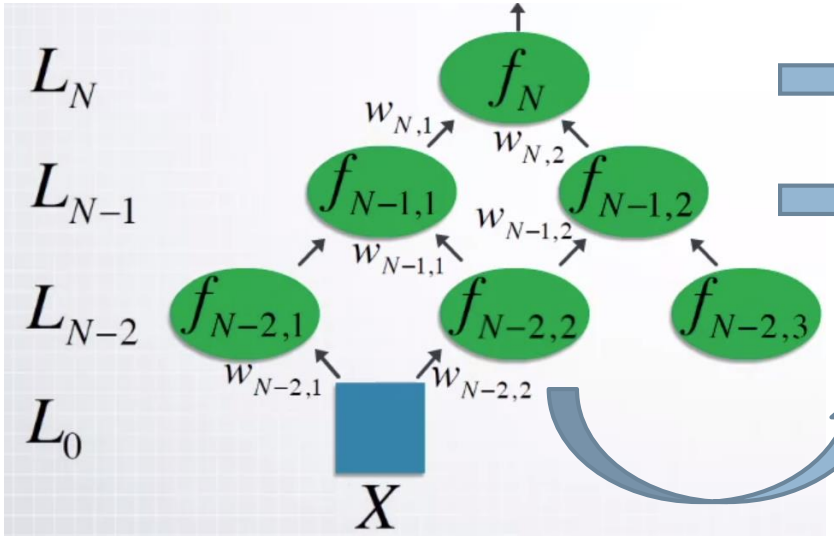
$$\frac{\partial f}{\partial n_5} = \frac{\partial f}{\partial n_7} \cdot \frac{\partial n_7}{\partial n_5} = \frac{\partial n_7}{\partial n_5} = 1 \qquad \frac{\partial f}{\partial n_6} = \frac{\partial f}{\partial n_7} \cdot \frac{\partial n_7}{\partial n_6} = \frac{\partial n_7}{\partial n_6} = 1$$

$$\frac{\partial f}{\partial n_4} = \frac{\partial f}{\partial n_5} \cdot \frac{\partial n_5}{\partial n_4} = \frac{\partial n_5}{\partial n_4} = n_3 \qquad \frac{\partial f}{\partial y} = \frac{\partial f}{\partial n_3} = \frac{\partial f}{\partial n_5} \cdot \frac{\partial n_5}{\partial n_3} + \frac{\partial f}{\partial n_6} \cdot \frac{\partial n_6}{\partial n_3} = n_4 + 1 = x^2 + 1$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_4} \cdot \frac{\partial n_4}{\partial x} = n_3 \cdot 2x = 2xy$$

Contexte général de l'AI
Rappels sur les neurones

- Apprentissage par back-propagation :
  - Application à un réseau de neurones
    - Objectif regarder comment la variation de chacun des poids va influencer la fonction de coût



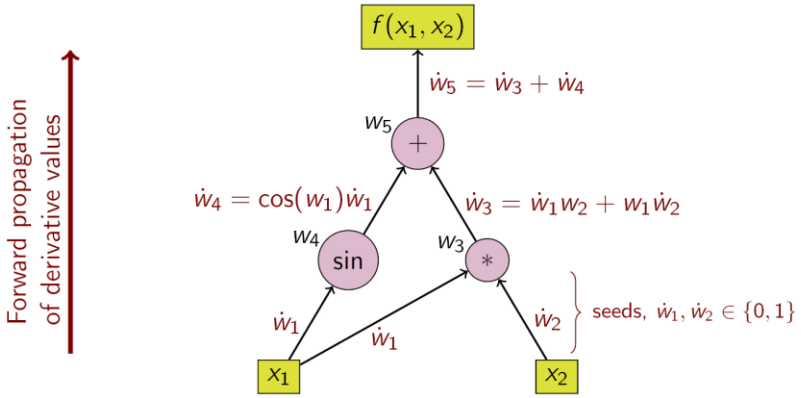
$$\frac{\partial f}{\partial w_{Ni}} = \frac{\partial f}{\partial f_N} \cdot \frac{\partial f_N}{\partial w_{Ni}} \equiv \delta_N \frac{\partial f_N}{\partial w_{Ni}}$$

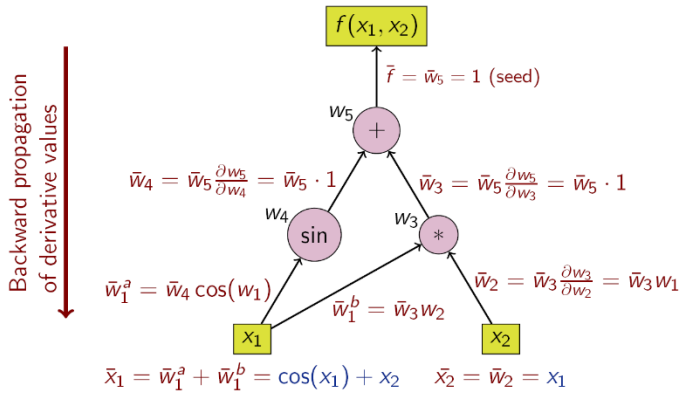
$$\frac{\partial f}{\partial w_{N-1,i}} = \frac{\partial f}{\partial f_N} \frac{\partial f_N}{\partial f_{N-1,i}} \cdot \frac{\partial f_{N-1,i}}{\partial w_{N-1,i}} = \delta_{N-1,i} \frac{\partial f_{N-1,i}}{\partial w_{N-1,i}}, \quad \delta_{N-1,i} = \delta_N \frac{\partial f_N}{\partial f_{N-1,i}}$$

$$\frac{\partial f}{\partial w_{N-2,2}} = \sum_{i=1,2} \frac{\partial f}{\partial f_N} \frac{\partial f_N}{\partial f_{N-1,i}} \frac{\partial f_{N-1,i}}{\partial f_{N-2,2}} \cdot \frac{\partial f_{N-2,2}}{\partial w_{N-2,2}} = \delta_{N-2,2} \frac{\partial f_{N-2,2}}{\partial w_{N-2,2}}, \quad \delta_{N-2,2} = \sum_{i=1,2} \delta_{N-1,i} \frac{\partial f_{N-1,i}}{\partial f_{N-2,i}}$$

Contexte général de l'AI
Rappels sur les neurones

- Comparaison des méthodes d'apprentissage
  - Hypothèse : réseaux de neurones à N inputs et M outputs
    - Si M>>N : forward propagation plus intéressante
    - Si N>>M : back propagation plus intéressante (cas des Neural Networks)





Contexte général de l'AI
Rappels sur les neurones

- Gradient stochastique VS gradient standard
  - Pb du gradient standard :
    - Très couteux à calculer sur l'ensemble du training set
  - Gradient stochastique
    - On calcul le gradient sur un petit ensemble de data du training set tirées au hasard
      - Beaucoup plus rapide !
      - Introduit un bruit stochastique
        - Permet d'éviter de rester piégé dans un minimum local de gradient quand celui-ci est assez plat.
      - A l'extrème, si on prend un sample à chaque itération
        - Online Stochastic Gradient Descent (SGD) : peut converger en fonction du learning rate

Contexte général de l'AI
Rappels sur les neurones

- Apprentissage dans un réseau de neurones
  - Régularisation
    - Objectif : pénaliser l'overfitting du training set
    - Régularisation L2 sur les poids :

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j \left(W_{i,j}^{(k)}\right)^2 = \sum_k ||\mathbf{W}^{(k)}||_F^2$$

    - Régularisation L1 sur les poids :
      - Permet de mettre certains poids à 0.

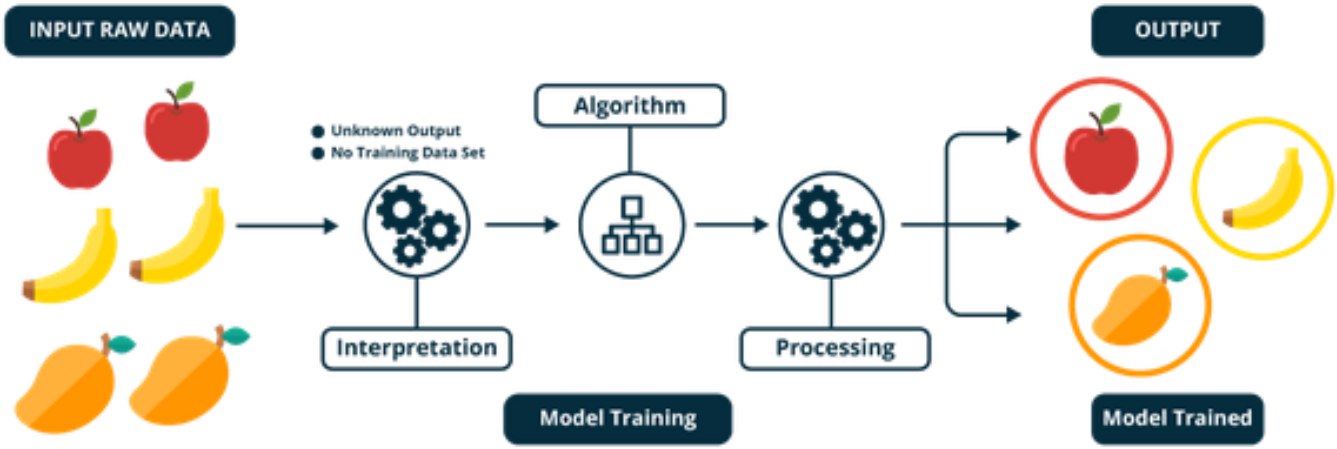
$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j |W_{i,j}^{(k)}|$$

Contexte Général de l'AI

Supervised learning

Contexte général de l'AI
Supervised learning

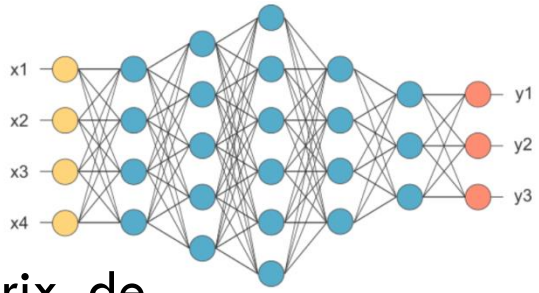
- Supervised learning
  - Apprentissage sur des données étiquetées
  - Reconnaissance basée sur des features (apprises ou prédéfinies)

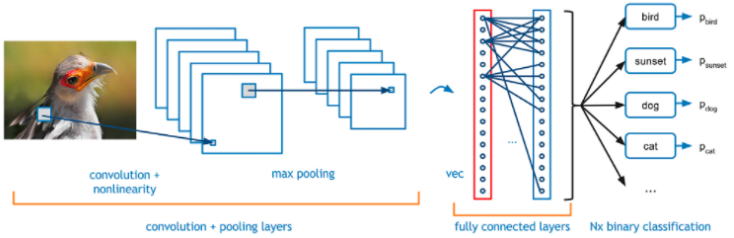


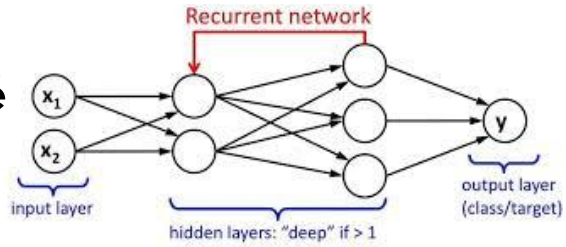
  - Applications : identification de patterns…

Contexte général de l'AI
Supervised learning

- Basé sur des réseaux de neurones
  - Standard neural networks
    - Utilise un extracteur de features en entrée
    - Application : évaluation de pertinence, de prix, de score…
  - Convolutional neural networks
    - Souvent multiscale
    - Appli : image principalement
  - Recurrent neural networks
    - Série de réseaux connectés entre eux avec liens entre les réseaux
    - Application : reconnaissance de texte ou de parole (permet de coder des probabilité d'apparition de succession de pattern)

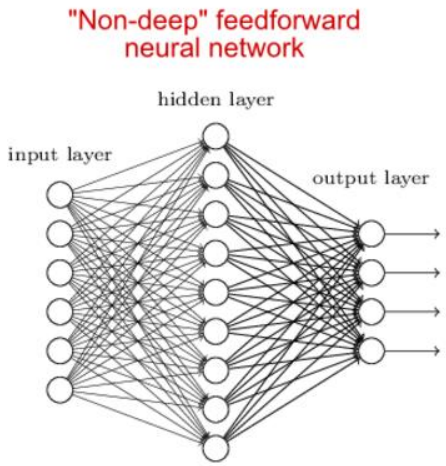


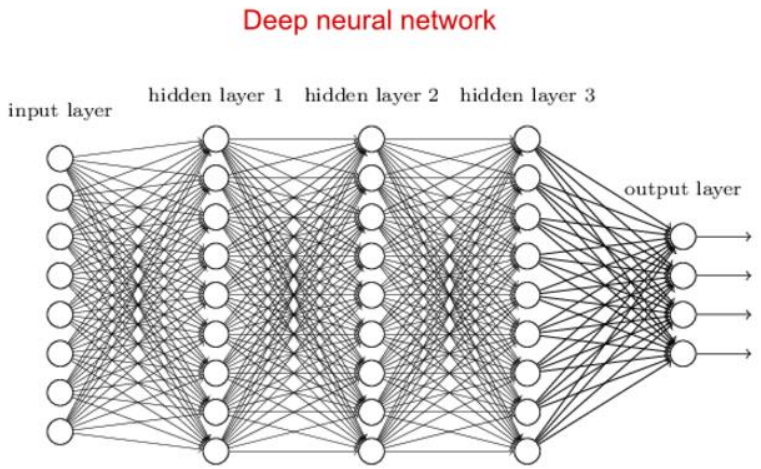




Contexte général de l'AI
Supervised learning

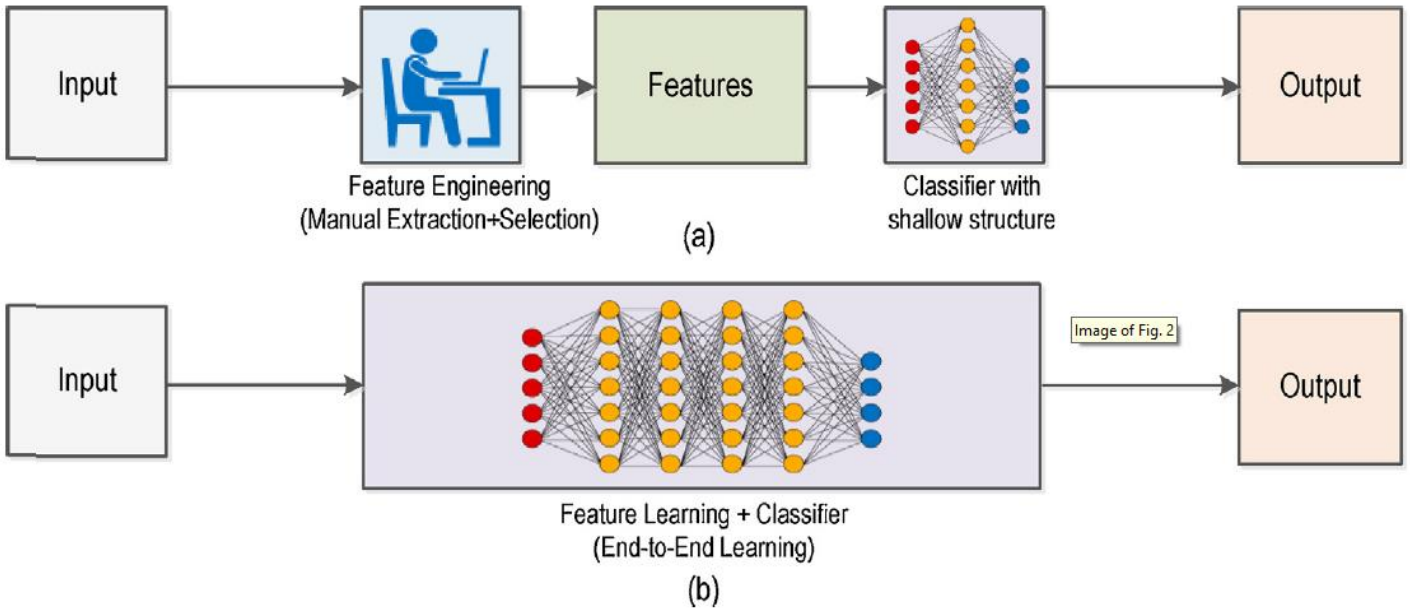
- Shallow et deep neural networks
  - Shallow networks
    - En général 1 couche cachée
  - Deep networks
    - Plus d'une couche cachée
- Différence de performance
  - A puissance de calcul égale :
    - Le deep donne en général de meilleurs résultats que le shallow





Contexte général de l'AI
Supervised learning

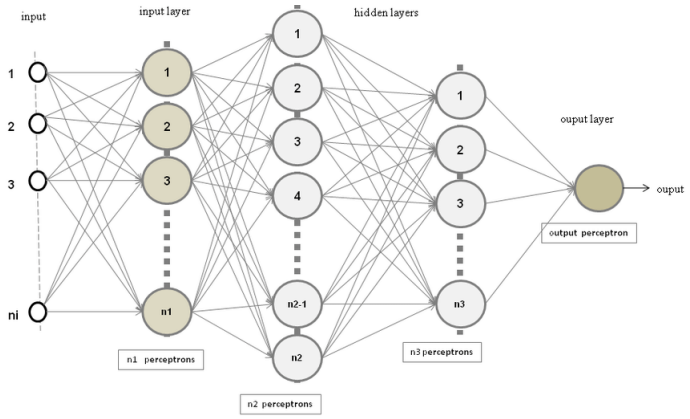
- Convolutional networks VS standard networks
  - Apprentissage simultané features+classifier



Source : Deep learning for smart manufacturing: Methods and applications – 2018 – Journal of Manufacturing Systems

Contexte général de l'AI
Supervised learning

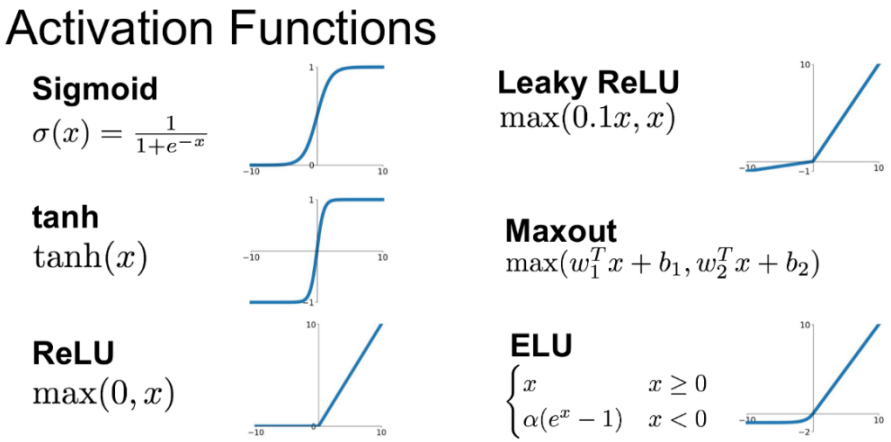
- Evaluation du cout algorithmique du shallow learning
  - Classifieur de taille limitée
    - Nb de MAC :
      - N1*N2 + N2*N3 + N3
      - N1 = 10, N2 = 100, N3 = 50 :
        - 6050 MAC



  - Extraction de features
    - Complexité :
      - N log N en FFT (Fast Fourier Transform)
      - N^2 en DFT (Discrete Fourier Transform)
    - Avec des échantillons de 1024 data (FFT à 48Hz en 48kHz audio)
      - 1MFLOP en FFT
      - 10kFLOP en DFT
  - L'extraction de features coute beaucoup plus cher que la classification (hors apprentissage)

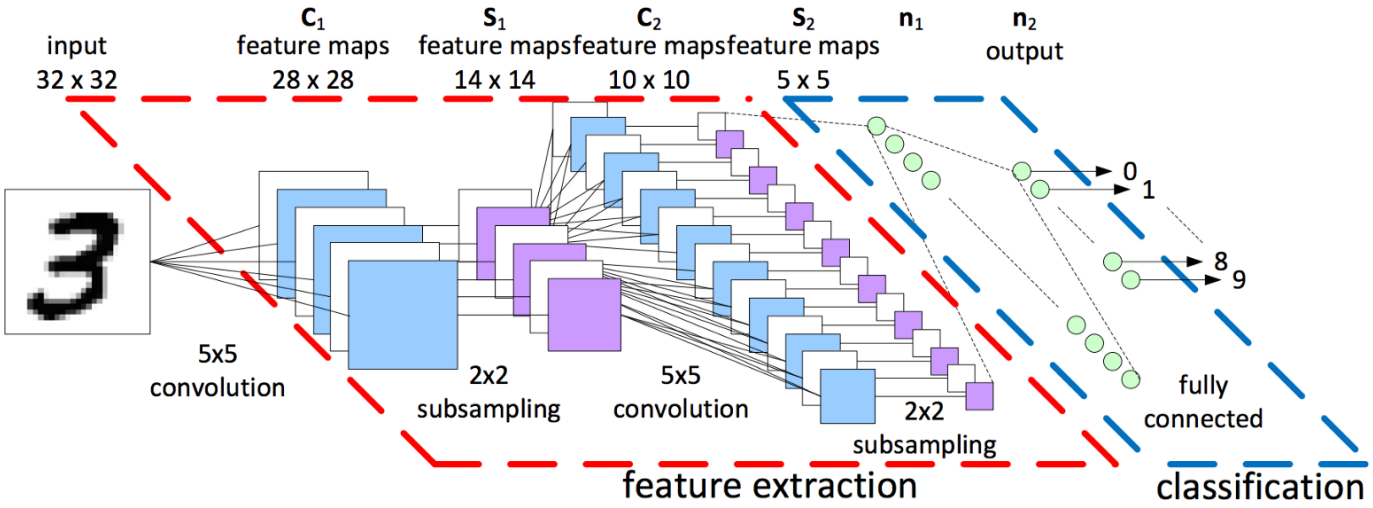
Contexte général de l'AI
Supervised learning

- Supervised learning à base de convolutional neural networks
  - Les noyaux de convolution sont appris :
    - adaptation des features à la tâche à effectuer
  - Les poids du réseau de classification sont également appris comme dans un réseau classique.
  - Le pooling est associé à une fonction d'activation non-linéaire (ReLu)
    - Permet de décrire des phénomènes non-linéaires.



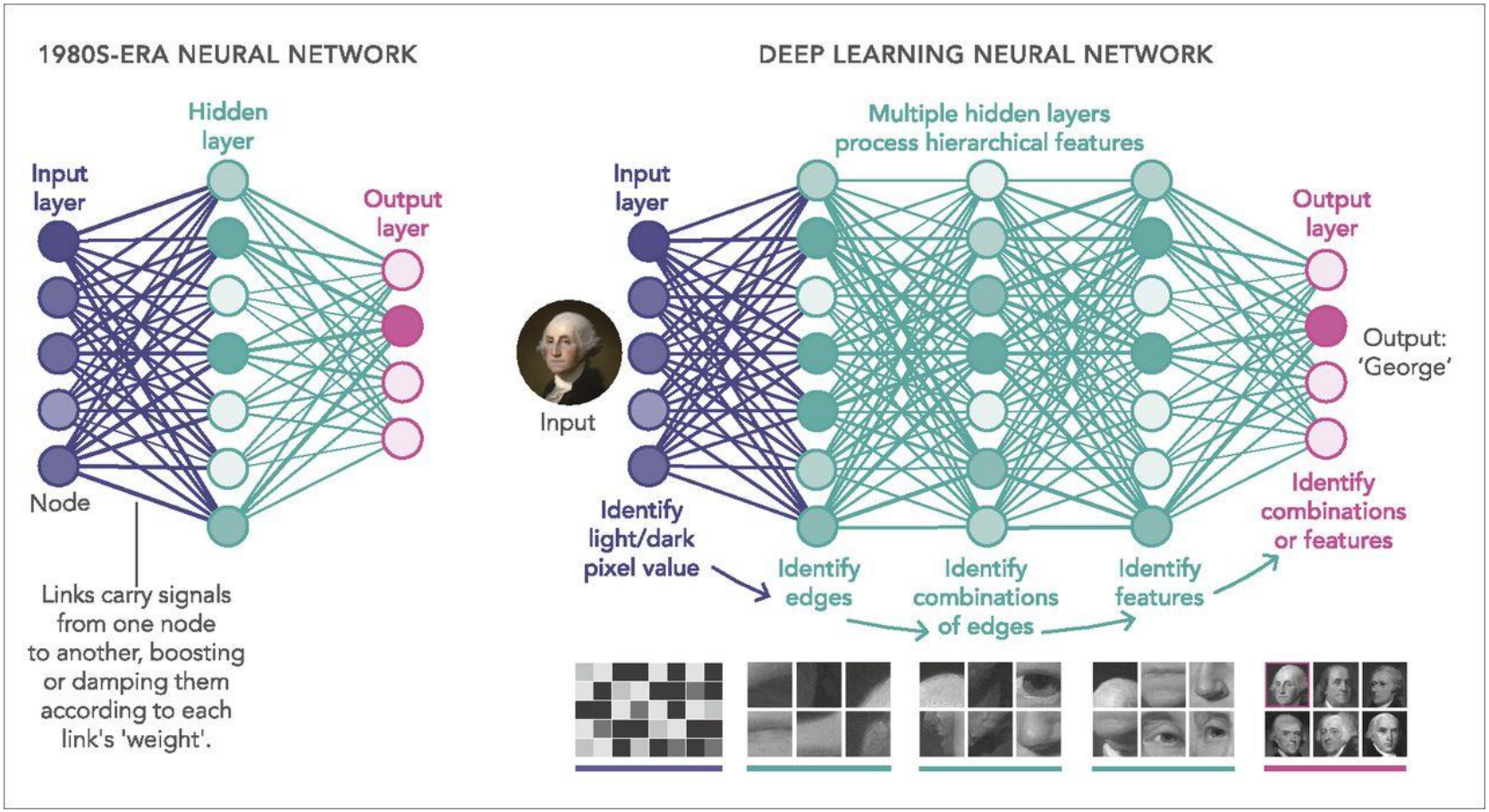
Contexte général de l'AI
Supervised learning

- Supervised learning à base de convolutional neural networks :
  - Multi-résolution (subsampling ou pooling)
  - Convolution successive à échelles différentes
  - Permet de détecter des pattern à échelle et position
  - Exemple d'application : https://www.cs.ryerson.ca/~aharley/vis/conv/flat.html



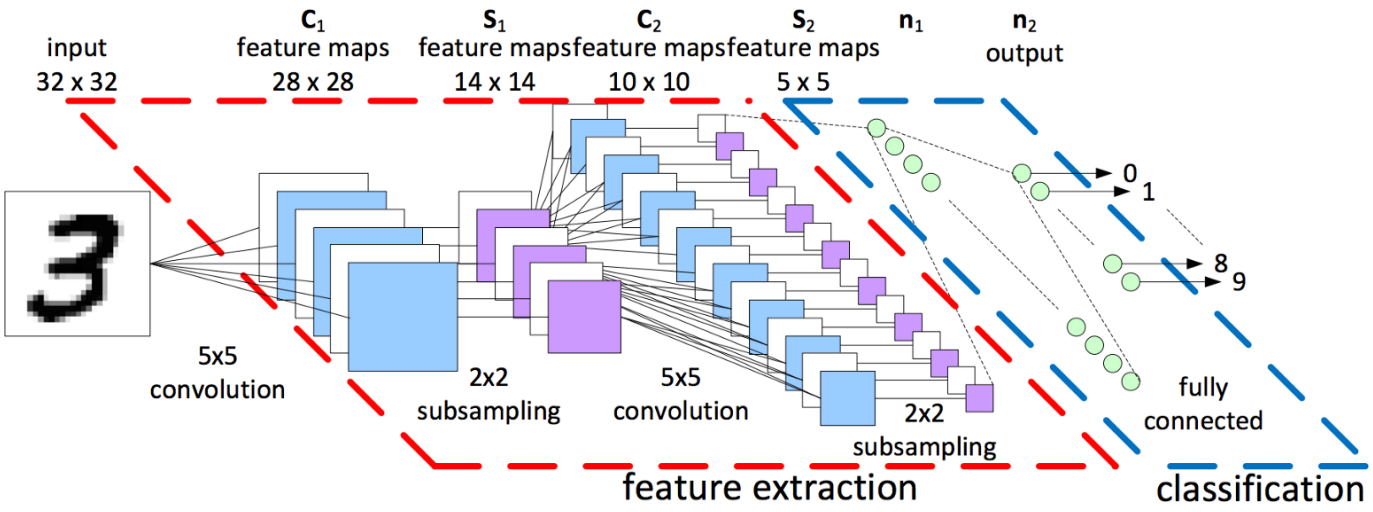
Contexte général de l'AI
Supervised learning

- Exemple de processing en deep learning



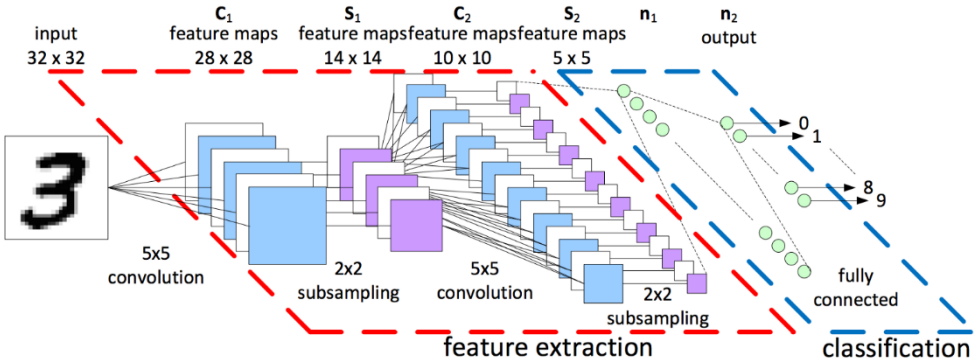
Contexte général de l'AI
Supervised learning

- Evaluation du cout algorithmique d'un deep learning :
  - A vous de le calculer !



Contexte général de l'AI
Supervised learning

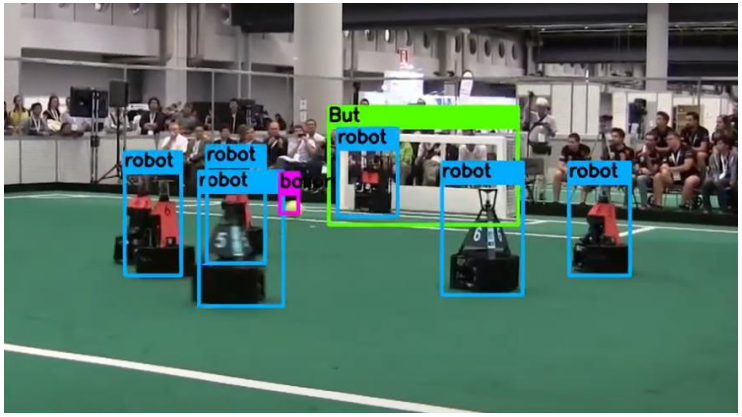
- Evaluation du cout algorithmique
  - A chaque image 32 * 32 :
    - Nb de MAC :
      - C1 = 6*28*28*5*5
      - S1 = 6*14*14*2*2
      - C2 = 16*10*10*5*5
      - S2 = 16*5*5*2*2
      - N2 = 10*16*5*5



      - Total = 117600+4704+40000+1600+4000 = 167904 MAC/im en 32*32 !
    - à 50 fps 32*32 : 8,2 Mflops (limite pour un MCU)
    - À 50 fps en 512*512 : >2 Gflops (nécessite des architectures parallèles)

Contexte général de l'AI
Supervised learning

- Implantation software des CNN
  - Apprentissage à l'aide d'outils dédiés
    - Format de sortie interopérable : Open Neural Network Exchange Format (ONNX)
  - Exécution sur plateforme GPU
    - Yolo V3 : Vidéo RoboCup

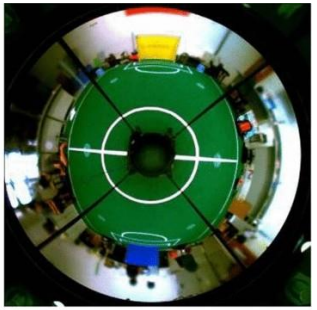


Caffe2 · Yandex CatBoost · Chainer · Cognitive Toolkit · K · LibSVM · MATLAB · mxnet · MyCaffe · NeoML · Neural Network Libraries · PaddlePaddle · PyTorch · sas · SIEMENS · SINGA · learn · Tengine · TensorFlow · dmlc XGBoost

Contexte général de l'AI
Supervised learning

- Limitations des CNN
  - Difficultés techniques
    - Nécessite de très grandes bases (très peu de disponibles)
    - Cout très important de l'apprentissage
    - Pas d'invariance par rotation de l'image
  - Problèmes éthique
    - Sensible à la composition de la base
      - Problème de représentativité.
      - Problème éthiques dans certains cas.

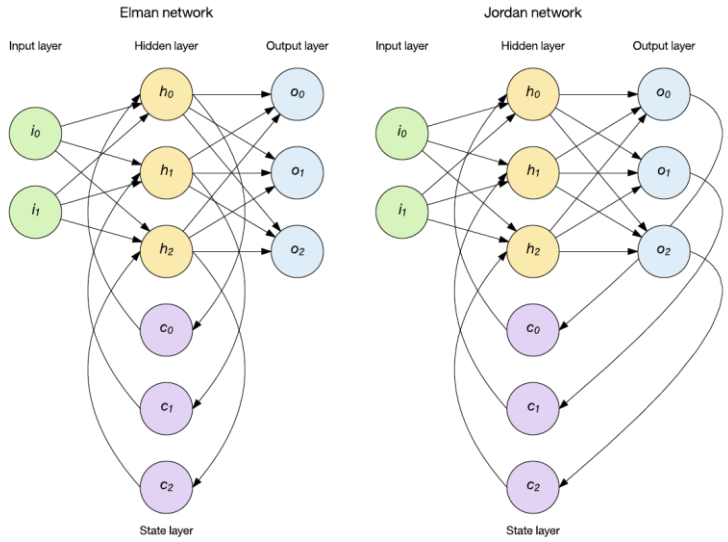
(a) omnidirectional camera



(b) omnidirectional image

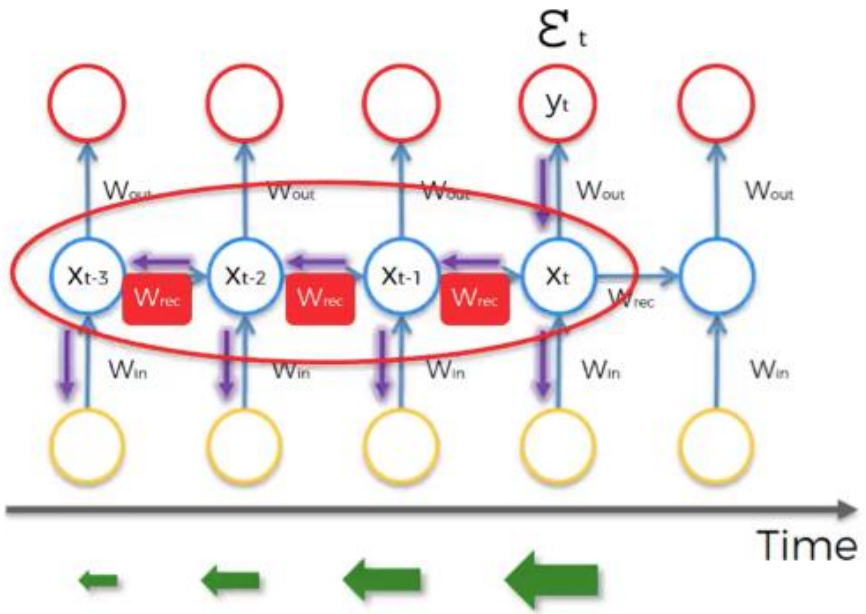
Contexte général de l'AI
Supervised learning

- Supervised learning à base de recurrent neural networks
  - Utilisé pour l'analyse des séries temporelles
  - Base : utilise des éléments de mémorisation des états précédents
    - De la couche interne
      - Réseau de Elman
    - Des sorties
      - Réseau de Jordan
  - Optimisé de manière classique
    - Back propagation through time
      - Somme les erreurs au cours du temps



Contexte général de l'AI
Supervised learning

- Supervised learning à base de recurrent neural networks
  - Pb de vanishing or explosion error
    - Multiplication de dérivées partielles très grandes ou petites



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

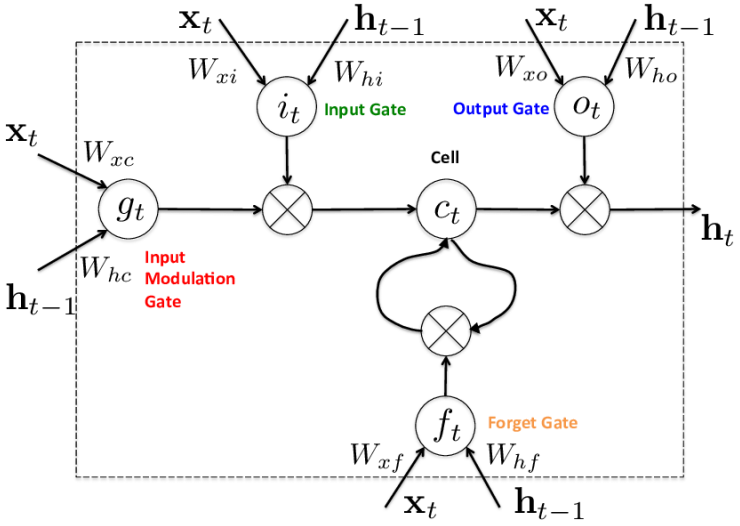
W_{rec} ~ small ⇨ Vanishing

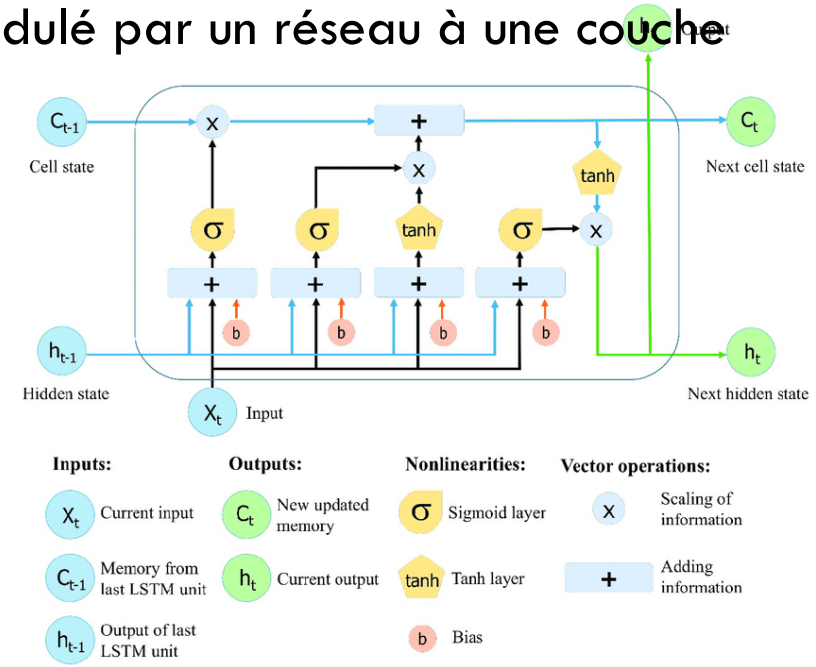
W_{rec} ~ large ⇨ Exploding

Formula Source: Razvan Pascanu et al. (2013)

Contexte général de l'AI
Supervised learning

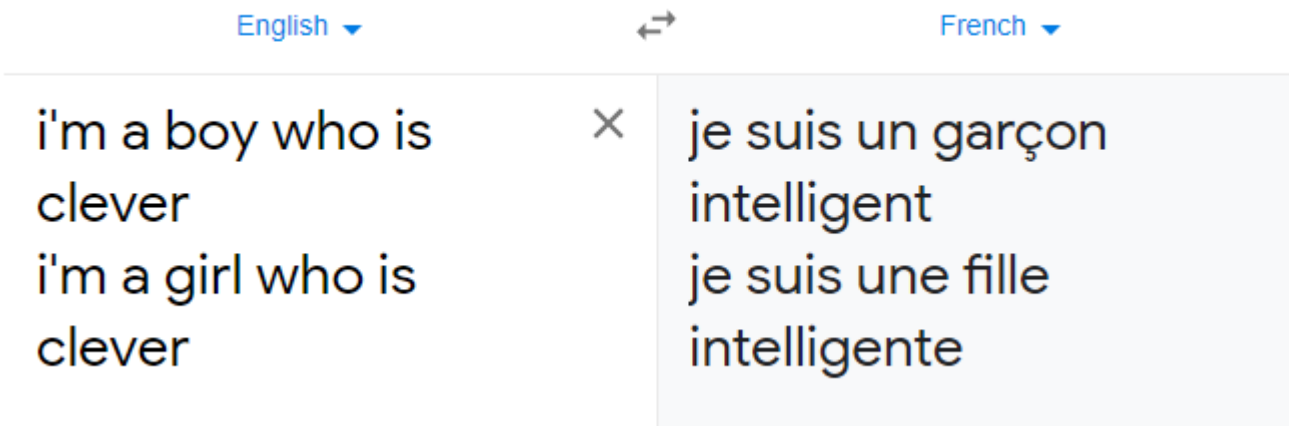
- Supervised learning à base de recurrent neural networks
  - Solution avancée : Long-Short Term Memory (LSTM)
    - Fonctionne comme des mémoires pondérées par des vannes.
    - L'état mémorisé Ct est la somme de Xt modulé et Ct-1 modulé par des réseaux à une couche.
    - La sortie est également Ct modulé par un réseau à une couche





Contexte général de l'AI
Supervised learning

- Supervised learning à base de recurrent neural networks
  - Applications des Long-Short Term Memory (LSTM)
    - Traduction automatique et contextuelle
      - Nécessite une mémorisation de proximité dans les mots pour les accords.

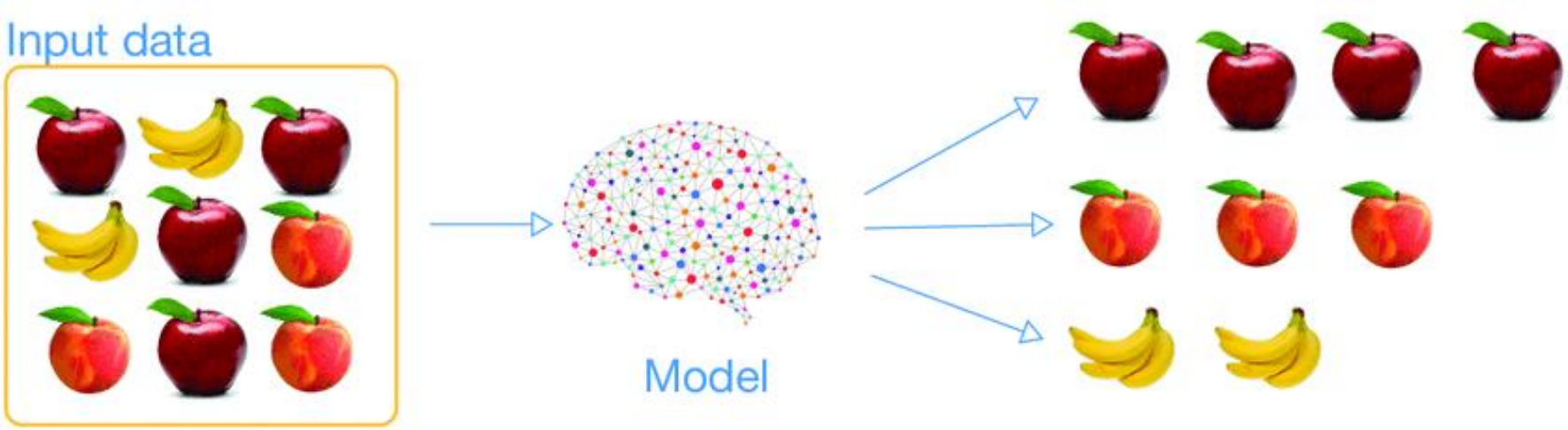


Contexte Général de l'AI

Unsupervised learning

Contexte général de l'AI
Unsupervised learning

- Apprentissage sur des données non étiquetées
- Clustering basé sur les similarités



- Applications :
  - Regroupement de data
  - Détection de phénomènes anormaux

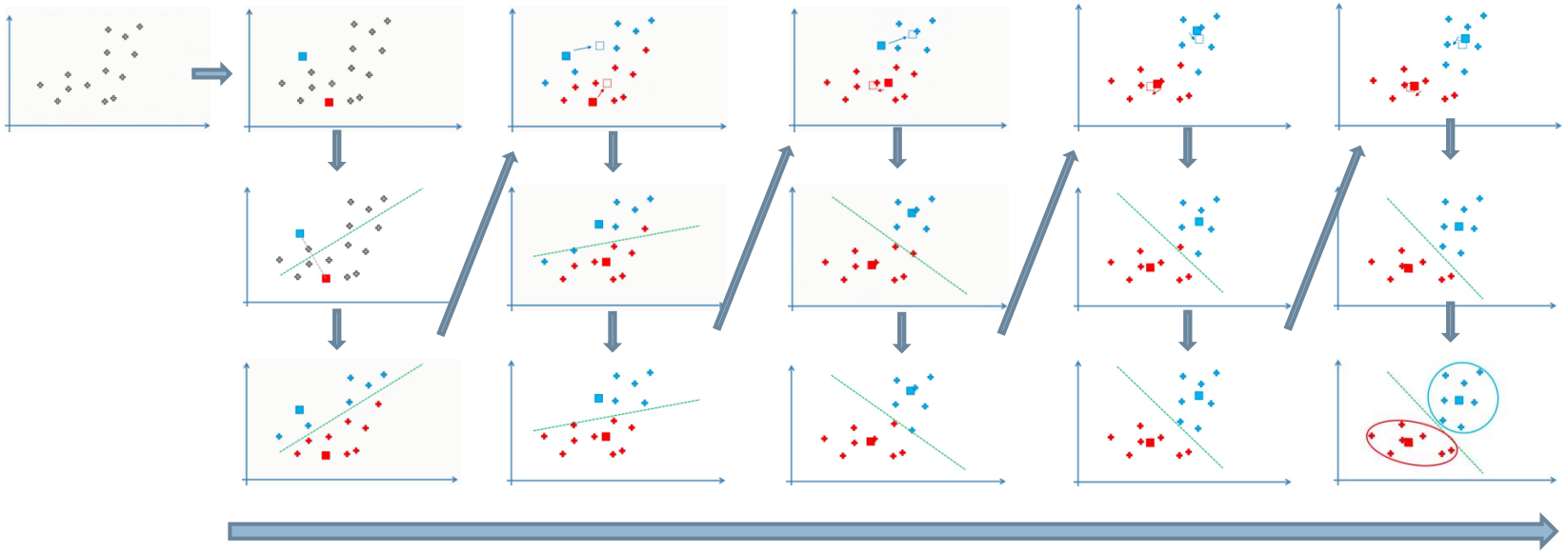
Contexte général de l'AI
Unsupervised learning

- Les données utilisées en unsupervised learning
  - Données brutes
    - Issues directement de capteurs
  - Données post-processées : features
    - Nécessitent une extraction de features préalable
      - FFT
      - Intégration
      - Filtrage
    - Les post-processing sont souvent une part importante du cout algorithmique lié à l'apprentissage.
      - Important pour la 2e partie !

Contexte général de l'AI
Unsupervised learning

- Algorithme K-Means
  - Répartition de vecteurs à n composantes en K clusters
    - Minimise le WCSS (Within Cluster Sum of Squares)

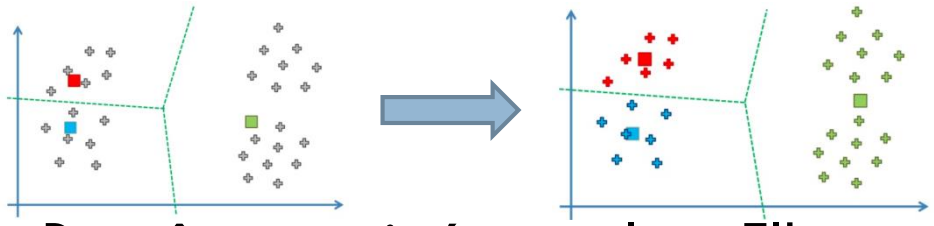
$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$



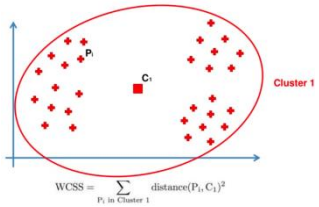
Itération des 3 étapes de base jusqu'à stabilisation

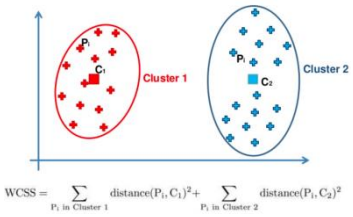
Contexte général de l'AI
Unsupervised learning

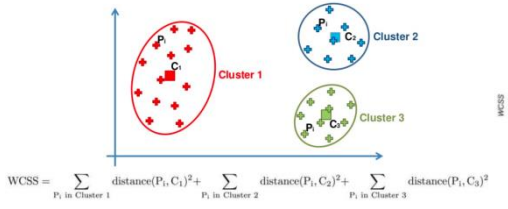
- Algorithme K-Means : limitations
  - Sensibilité à l'initialisation

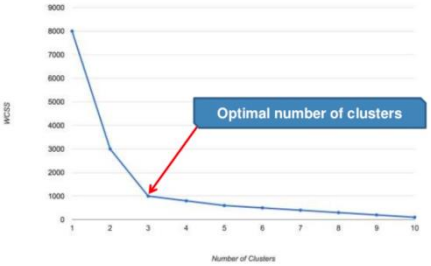


  - Peut être corrigée par la « Elbow method » (empirique)

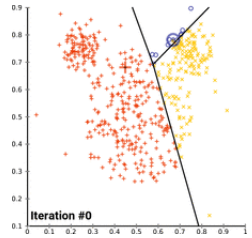








  - Ne marche pas toujours :
  - Nécessite d'autres algorithmes
    - EM clustering par exemple (basé sur la probabilité des pts d'appartenir à un des gaussiennes)



Contexte général de l'AI
Unsupervised learning

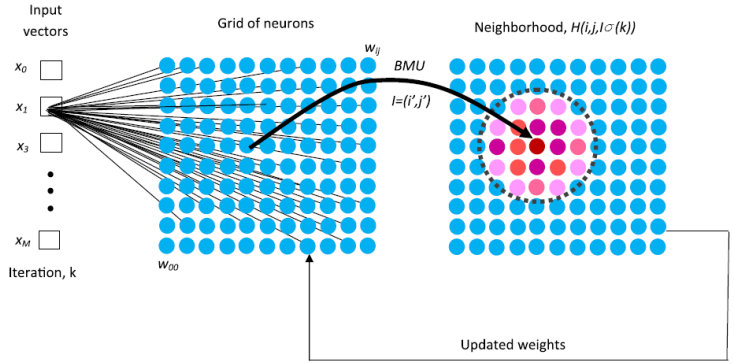
- Self Organizing Maps (Kohonen 1990)
  - Proche des K-Means avec une topologie (carrée ou hexagonale) liant les centroïdes des clusters
    - Chaque pt présenté est attribué à un cluster : Best matching Unit (BMU)
    - On update le réseau à l'aide du pt attribué
      - Le winner cluster est updaté
      - Ses voisins aussi

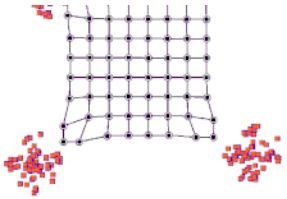
$$W_{t+1} = W_t + \eta_t L_t (X_t - W_t)$$

avec

$$L(t) = L_0 \mathrm{e}^{\left(\frac{-t}{\lambda}\right)}$$

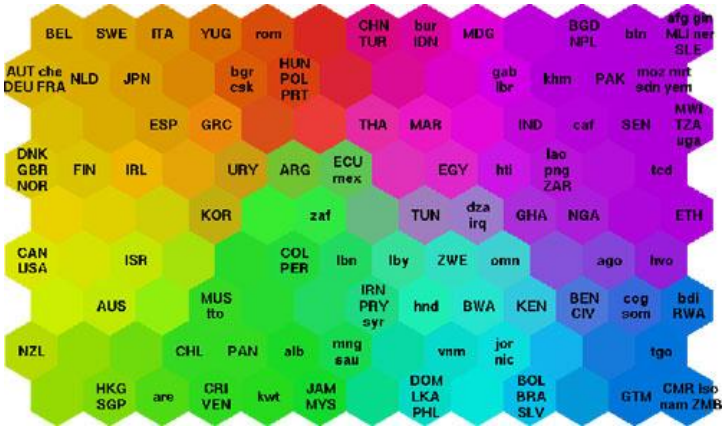
$$\eta(t) = \mathrm{e}^{\left(\frac{-distance^2}{2\,\sigma^2(t)}\right)}$$

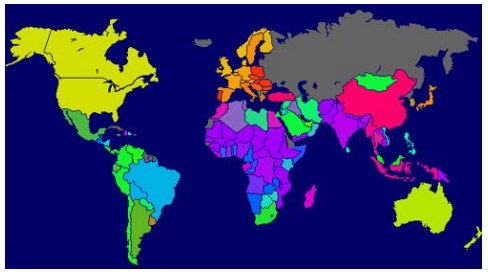




Contexte général de l'AI
Unsupervised learning

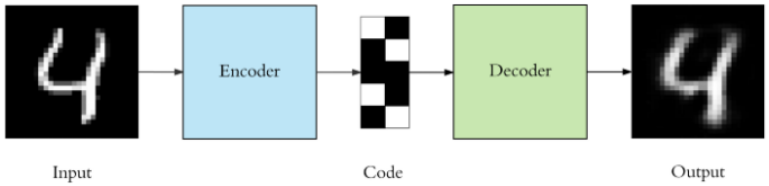
- Self Organizing Maps (Kohonen 1990)
  - Applications :
    - Clustering
      - Ex World Poverty Map
    - Fait apparaitre des corrélations
    - Sensibilité réduite au bruit
      - versus K-means





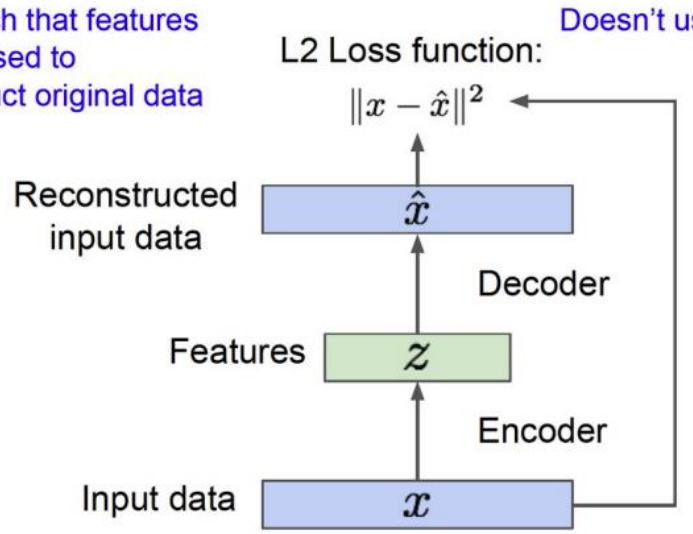
Contexte général de l'AI
Unsupervised learning

- Autoencoders
  - Input = Output



Train such that features can be used to reconstruct original data

Doesn't use labels!



  - Objectif :
    - Trouver les meilleurs features et poids pour reconstruire le signal d'entrée à partir d'un « code » de taille limitée.
      - Le code est une sorte de label généré (self-supervised networks)
    - Le code est la « signature » du signal d'entrée
      - Compression forte, mais dédiée à une tâche apprise
      - Compression avec perte

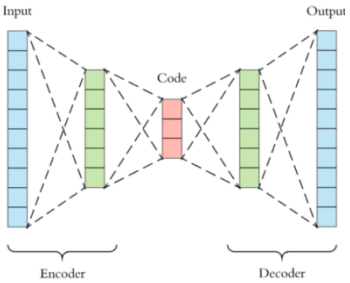
Contexte général de l'AI
Unsupervised learning

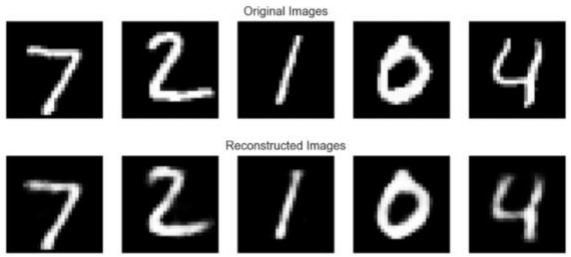
- Autoencoders
  - Paramètres de réglages
    - Taille du code : plus il est compact , plus la compression est importante.
    - Nombre de couches internes de l'encodeur et du décodeur.
    - Nombre de neurones par couche.
    - Choix de la loss function.

  - Exemple :
    - https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html

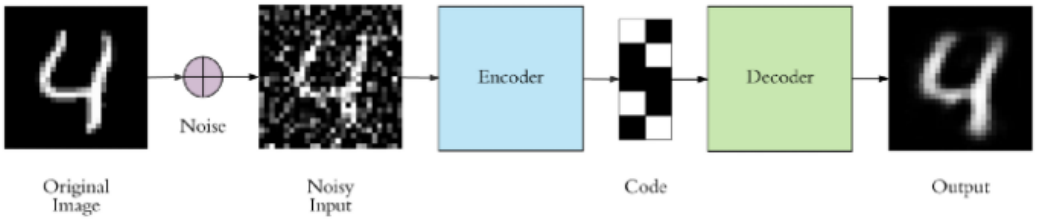
Contexte général de l'AI
Unsupervised learning

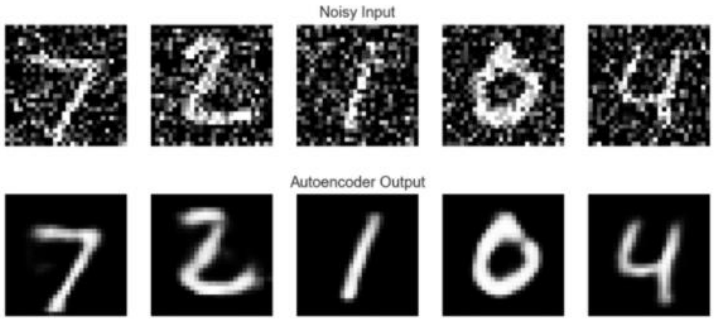
- Autoencoders : utilisation
  - Compression :
    - Example :
      - Code : 32 neurons
      - Hidden layer : 128 neurons
      - Input : 28*28 = 784 pixels
      - Activation :
        - Relu sur toutes les couches sauf la sortie
        - Sigmoid sur la sortie





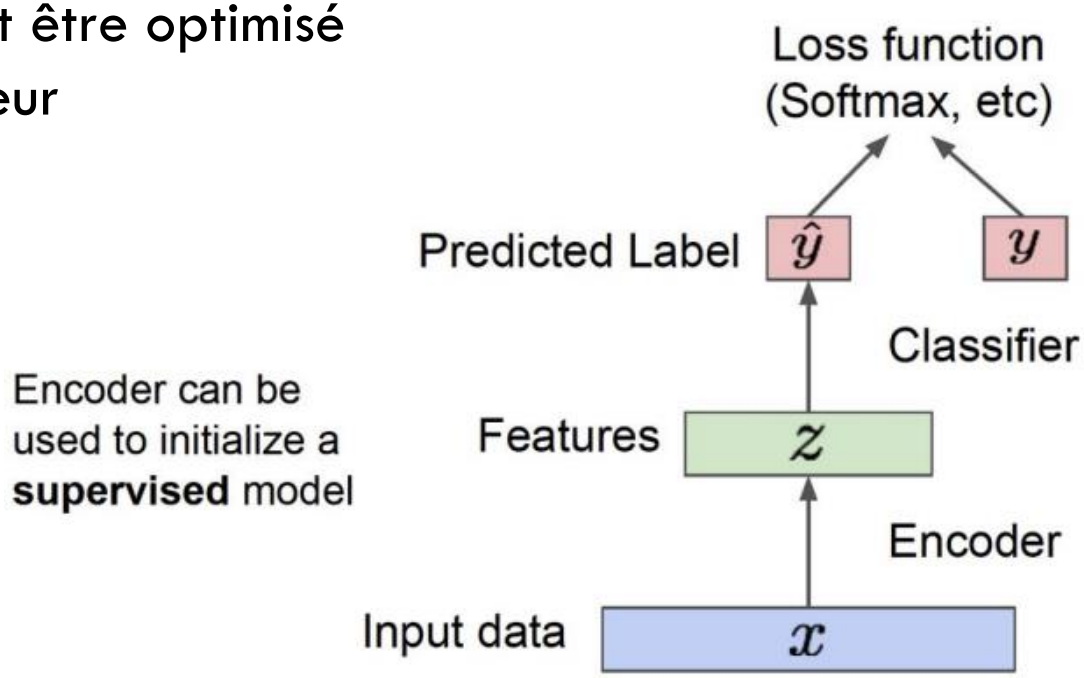
  - Suppression du bruit





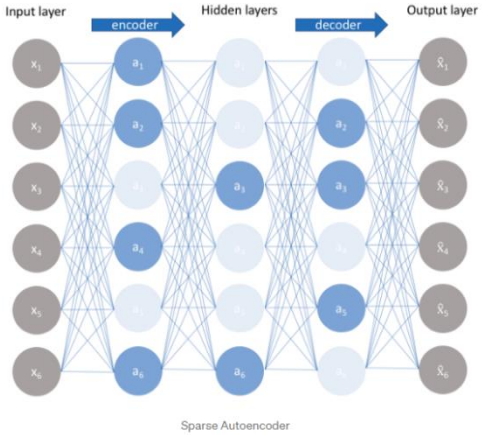
Contexte général de l'AI
Unsupervised learning

- Autoencoders : utilisation en supervisé possible
  - Utilisation des features pour faire de la classification supervisée
    - Après apprentissage
    - L'encodeur peut être optimisé avec le classifieur



Contexte général de l'AI
Unsupervised learning

- Sparse autoencoders
  - Auto-encoder à faible nombre de neurones actifs
    - Apprentissage avec une forte pénalisation de type L1 sur la fonction de régularisation.
    - Evite le sur-apprentissage
    - Peut donner de meilleurs résultats qu'un autoencoder classique
      - Se focalise sur l'essentiel



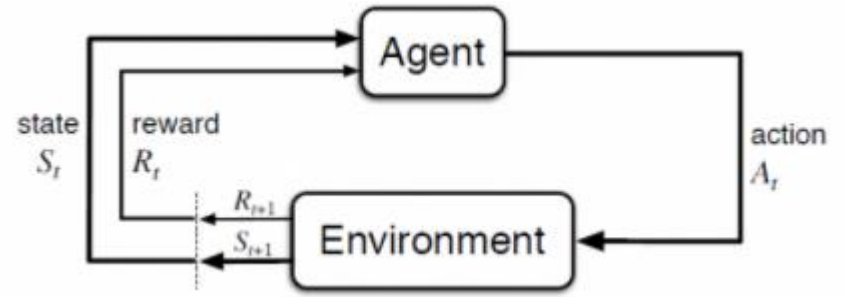
Sparse Autoencoder

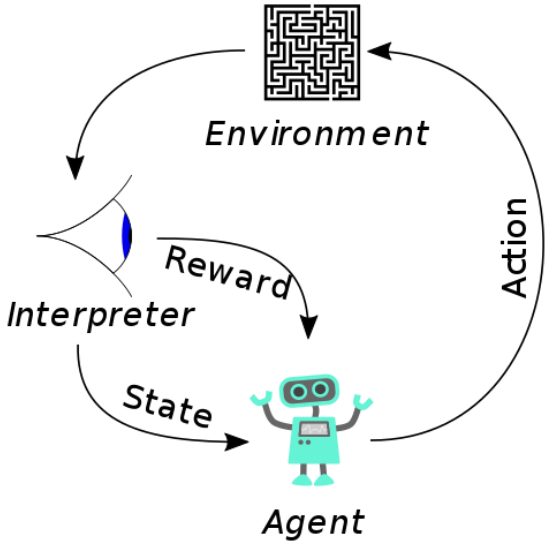
Contexte général de l'AI

Reinforcement learning

Contexte général de l'AI
Reinforcement learning

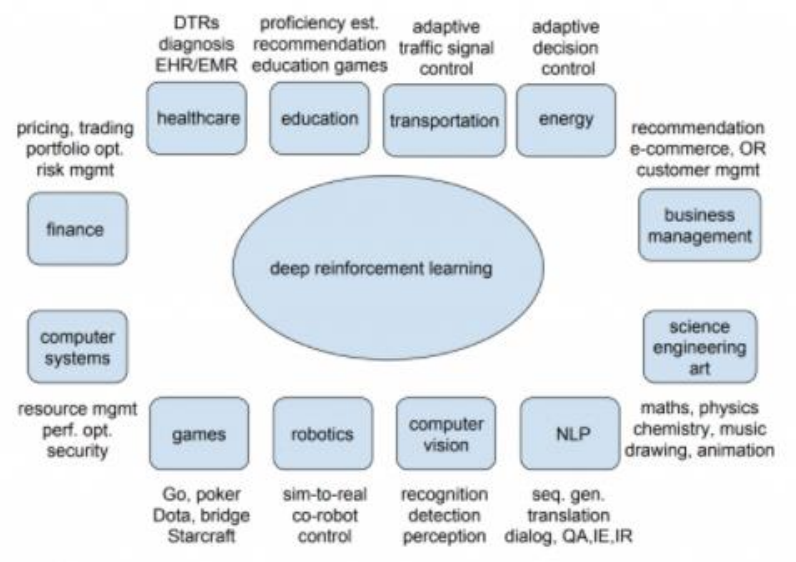
- Contexte :
  - Utilisé quand un agent doit prendre en compte les conséquences à long terme de ses actes, même si les conséquences à court terme ne sont pas favorables
    - Robotique
    - Stratégie de jeu
  - Basé sur des mécanismes d'incitation

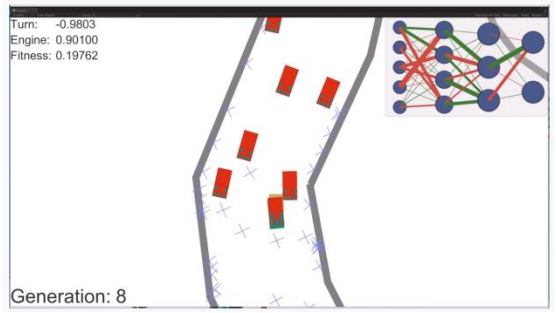




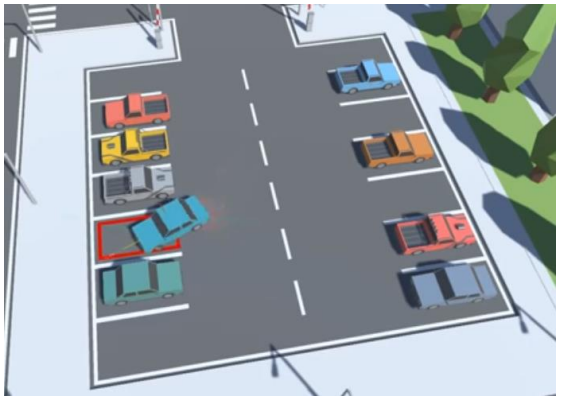
Contexte général de l'AI
Reinforcement learning

- Domaines d'application





- Deep Q Learning
  - Apprentissage de la conduite
  - Parking



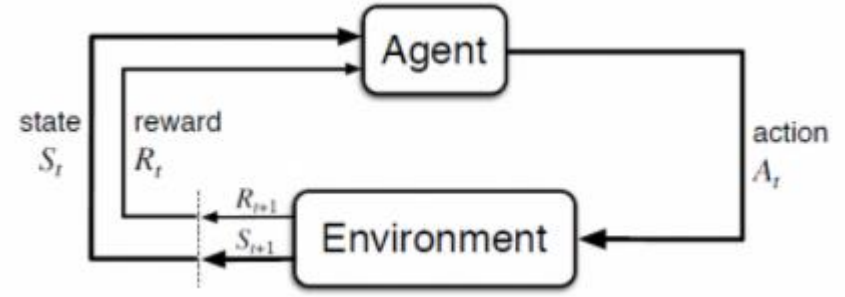
Contexte général de l'AI
Reinforcement learning

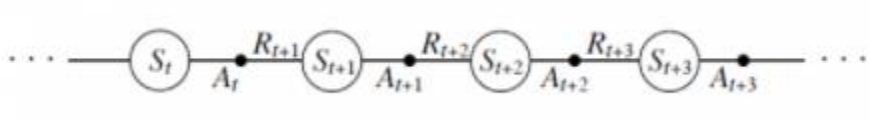
- Objectif :
  - Maximiser le gain cumulé au cours du temps

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

  - En réalité : on atténue les revenus du futur (incertains)

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$





Contexte général de l'AI
Reinforcement learning

- Hypothèse : chaque état ne dépend que de l'état précédent (Markov)
  - On *pourrait penser* a priori que la proba d'être dans un état S' et de gagner R dépend de tous les états précédents :

$$\Pr(s_{t+1}=s', r_{t+1}=r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0)$$

  - On simplifie
    - L'état S' et la récompense R du système à t dépend exclusivement de l'état s du système à t-1et des actions a à t

$$\Pr(s_{t+1}=s', r_{t+1}=r \mid s_t, a_t)$$

- On appelle cela les processus de Markov :
  - Pas de mémorisation des états passé, le futur ne dépend que de l'état courant

Contexte général de l'AI
Reinforcement learning

- Comment dire si un état est bon ou pas ?
  - Fonction de valeur v :

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

    - V(s) est l'espérance de récompense, définie pour chacun des états possibles si un agent suit la policy π
    - La policy π est l'ensemble des probabilités de prendre une action a connaissant l'état s du système.
  - Optimum policy :
    - La policy optimal est celle qui maximise l'espérance de récompense pour chacun des états s possibles.

$$\pi^* = \arg\max_\pi V^\pi(s) \quad \forall s \in \mathbb{S}$$

Contexte général de l'AI
Reinforcement learning

- Comment dire si un état est bon ou pas ?
  - Un outil : la Q-value
    - La Q-value est le gain espéré par l'agent **si il choisit l'action a** dans l'état s.
      - C'est aussi l'espérance de la sommes des reward (atténuées) à chaque step

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

      - Implantation : ce calcul peut être effectué par une méthode Monte-Carlo
        - Calcul de la valeur moyenne de la récompense obtenue sur l'état suivant, le tirage obéissant à une probabilité s dépendant de la policy π.

Contexte général de l'AI
Reinforcement learning

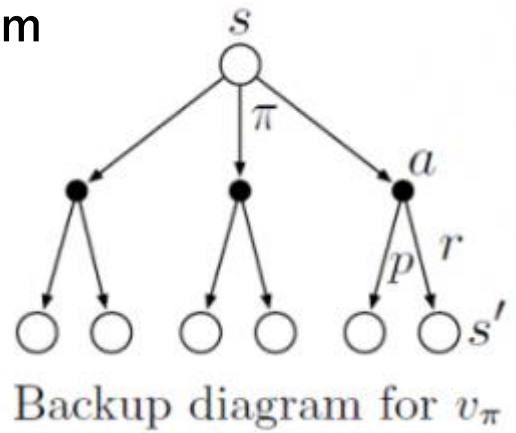
- Comment dire si un état est bon ou pas ?
  - Q-value : $$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t=s, A_t=a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \,\middle|\, S_t=s, A_t=a\right]$$
  - Equation de Bellman :
    - La fonction de valeur v peut se réécrire par récurrence e en développant sur tous les états possibles

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t=s] \\ &= \mathbb{E}_\pi[R_{t+1}+\gamma G_{t+1} \mid S_t=s] \\ &= \sum_a \pi(a|s)\sum_{s'}\sum_r p(s',r|s,a)\left[r+\gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1}=s']\right] \\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\left[r+\gamma v_\pi(s')\right], \quad \text{for all } s\in\mathcal{S}, \end{aligned}$$

    - Avec p(s', r | s , a), la proba d'arriver dans l'état s' avec la récompense r en étant dans l'état s et en ayant choisi l'action a

Contexte général de l'AI
Reinforcement learning

- Comment dire si un état est bon ou pas ?
  - L'équation de Bellman décrit l'espérance de v en balayant les états futurs possible en respectant leur proba de se produire.
    - La valeur de l'état initial est égale à la moyenne pondérée des valeurs (atténuées) de chacun des états espérables s' à laquelle vient s'ajouter la récompense r
    - Se décrit à l'aide d'un backup diagram



Backup diagram for v_π

Contexte général de l'AI
Reinforcement learning

- Comment trouver la policy optimale ?
  - La policy optimale maximise V pour chaque état

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) \quad \forall s \in \mathbb{S}$$

  - Les v* et q* correspondant sont :

$$v_*(s) \doteq \max_{\pi} v_{\pi}(s) \qquad q_*(s,a) \doteq \max_{\pi} q_{\pi}(s,a)$$

  - Ce qui peut s'écrire :

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \end{aligned}$$

Contexte général de l'AI
Reinforcement learning

- Comment trouver la policy optimale ?
  - En conséquences : la policy optimale qui maximise V pour chaque état est :

$$\begin{aligned}\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a]\end{aligned}$$

  - On peut résoudre cela de manière itérative
    - Programmation dynamique

Contexte général de l'AI
Reinforcement learning

- Exemple de résolution sur une grille de déplacement
  - On suppose que l'on veuille aller en 1 ou en 16 :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

actions 

Reward is -1 for all transition

- On pénalise de -1 chaque mouvement
- La policy π est random au départ (0.25 dans chaque direction)
- On initialise la fonction de valeur V0 à 0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Contexte général de l'AI
Reinforcement learning

- Exemple de résolution sur une grille de déplacement
  - Exemple de calcul de V1 à la case 6

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$$v_1(6) = \sum_{a\in\{u,d,l,r\}} \pi(a|6) \sum_{s',r} p(s',r|6,a)[r + \gamma v_0(s')]$$

$$= \sum_{a\in\{u,d,l,r\}} \underbrace{\pi(a|6)}_{=0.25\ \forall a} \sum_{s'} p(s'|6,a)[\underbrace{r}_{=-1} + \gamma \underbrace{v_0(s')}_{=0\ \forall s'}]$$

$$= 0.25 * \{-p(2|6,u) - p(10|6,d) - p(5|6,l) - p(7|6,r)\}$$

$$= 0.25 * \{-1 - 1 - 1 - 1\}$$

$$= -1$$

$$\Rightarrow v_1(6) = -1$$

  - On trouve la même chose sur chaque case non terminale
  - Sur les cases terminales, la probabilité de les quitter est nulle p(s'/s,a) = 0 et v0(1) ou v0(16)=0

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

Contexte général de l'AI
Reinforcement learning

- Exemple de résolution sur une grille de déplacement
  - On itère le processus avec V2

$$v_2(6) = \sum_{a\in\{u,d,l,r\}} \underbrace{\pi(a|6)}_{=0.25\ \forall a} \sum_{s'} p(s'|6,a)[\underbrace{r}_{=-1} + \gamma \underbrace{v_1(s')}_{=\begin{cases}-1, s'\in S\\ 0, s'\in S^+\setminus S\end{cases}}]$$

$$= 0.25 * \{p(2|6,u)[-1-\gamma] + p(10|6,d)[-1-\gamma] + p(5|6,l)[-1-\gamma] + p(7|6,r)[-1-\gamma]\}$$

$$\overset{\gamma=1}{=} 0.25 * \{-2-2-2-2\}$$

$$= -2$$

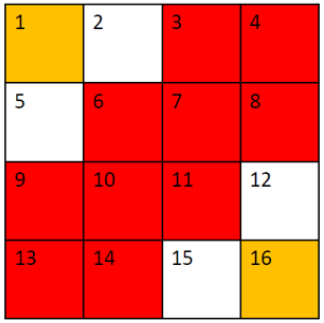
$$v_2(2) = \sum_{a\in\{u,d,l,r\}} \underbrace{\pi(a|2)}_{=0.25\ \forall a} \sum_{s'} p(s'|2,a)[\underbrace{r}_{=-1} + \gamma \underbrace{v_1(s')}_{=\begin{cases}-1, s'\in S\\ 0, s'\in S^+\setminus S\end{cases}}]$$

$$= 0.25 * \{p(2|2,u)[-1-\gamma] + p(6|2,d)[-1-\gamma] + p(1|2,l)[-1-\gamma*0] + p(3|2,r)[-1-\gamma]\}$$

$$\overset{\gamma=1}{=} 0.25 * \{-2-2-1-2\}$$

$$= -1.75$$

$$\Rightarrow v_2(2) = -1.75$$



$\Rightarrow v_2$ for the random policy:

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

Contexte général de l'AI
Reinforcement learning

- Exemple de résolution sur une grille de déplacement
  - On itère le processus n fois pour obtenir la fonction de valeur à l'infini

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$\cdots$

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$\cdots$

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

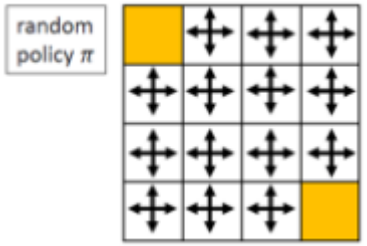
$\leftarrow v_\pi$

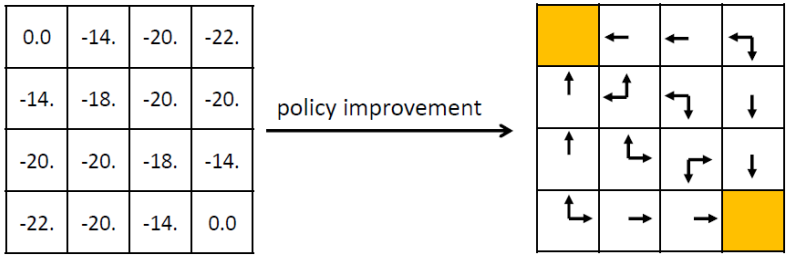
Contexte général de l'AI
Reinforcement learning

- Exemple de résolution sur une grille de déplacement
  - Optimisation de la policy à l'aide de la fonction de valeur

$$
\begin{aligned}
q_\pi(s,a) &\doteq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big].
\end{aligned}
$$

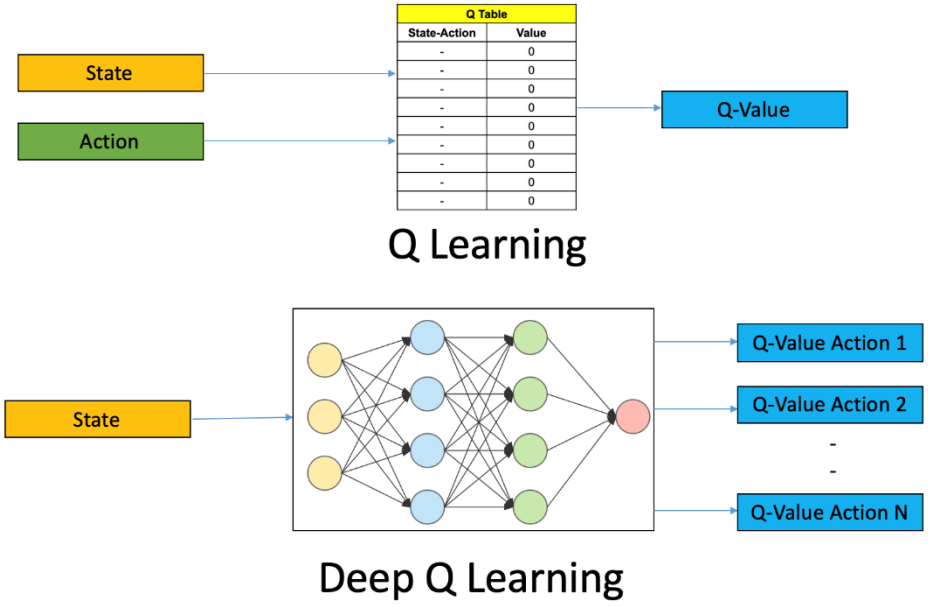
    - Optimiser la valeur de la policy revient à privilégier les chemins allant vers les petites v(s') et à éliminer les autres





Contexte général de l'AI
Reinforcement learning

- Algorithmes plus avancés de reinforcement learning
  - Deep Q-Learning : basé sur du Q-learning
    - On remplace la Q-Table (policy) par un réseau CNN



Contexte général de l'AI
Reinforcement learning

- Cout algorithmique
  - Après optimisation, le cout est limité
    - Très faible en Q-learning
    - Dépendant du CNN utilisé en deep Q-learning.

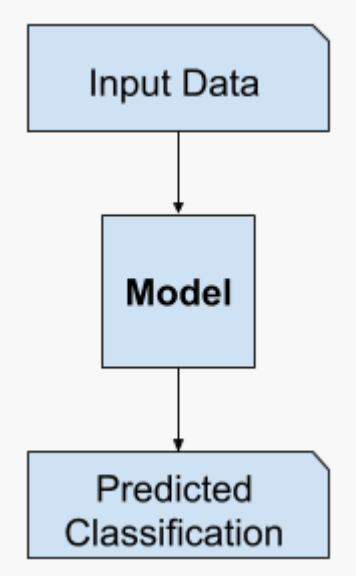
Contexte général de l'AI

Modèles Génératifs

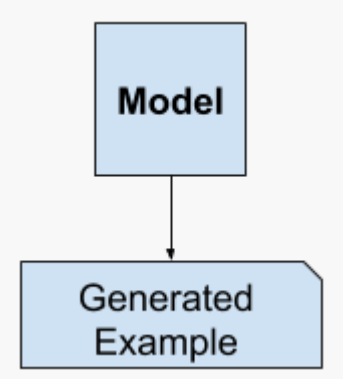
Contexte général de l'AI
Generative Models

- Qu'est-ce qu'un réseau génératif ?

Discriminative Model
Output = Classification



Generative Model
Output = Generated sample



Les données générées doivent être difficiles à distinguer des données réelles

Contexte général de l'AI
Generative Models

- Les modèles génératifs classiques (Ian GoodFellow)

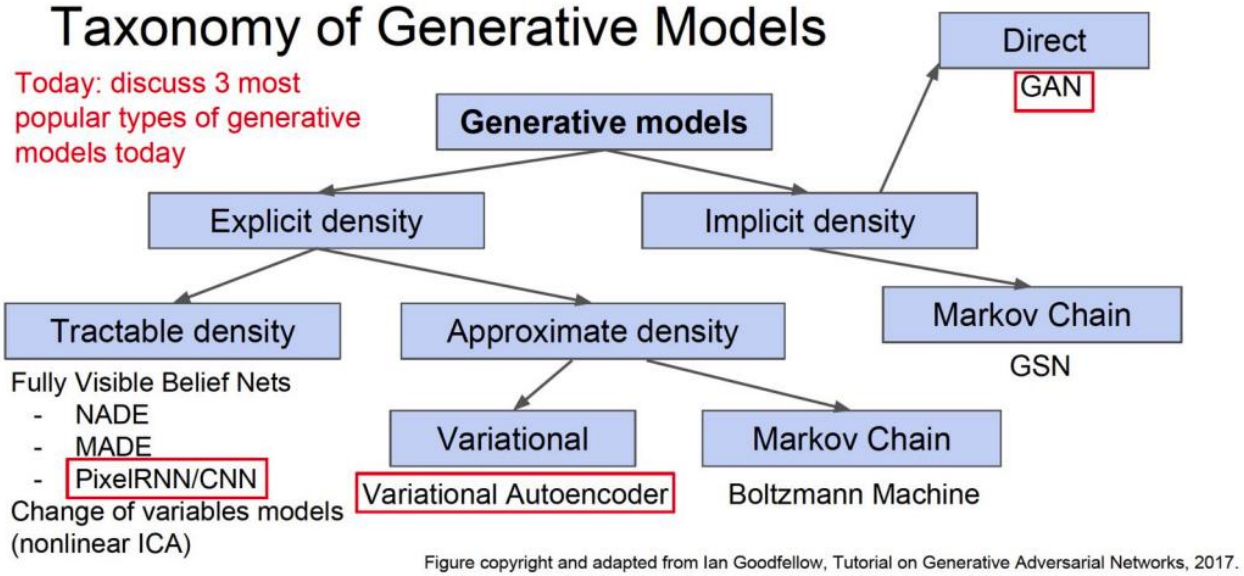


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

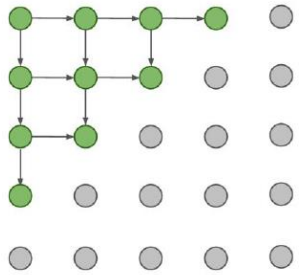
- Différent par
  - La possibilité d'exprimer formellement les probabilités conditionnelles entre données
  - La possibilité de l'optimiser formellement ou pas (tractability) les paramètres du réseau.

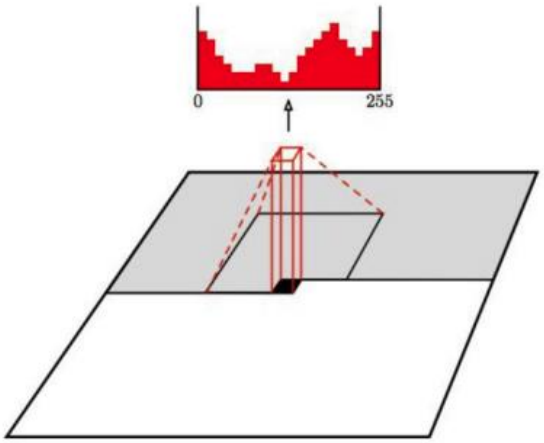
Contexte général de l'AI
Generative Models

- Quelques modèles génératifs classiques
  - Pixel RNN / Pixel CNN
    - Génération de données en utilisant le modèle MLE et les voisins déjà générés.
    - Entrainement effectué en cherchant le max de vraisemblance sur le training set.
    - Utilise une formulation explicite

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

    - Peut être remplacée par un CNN avec un noyau de taille limitée.



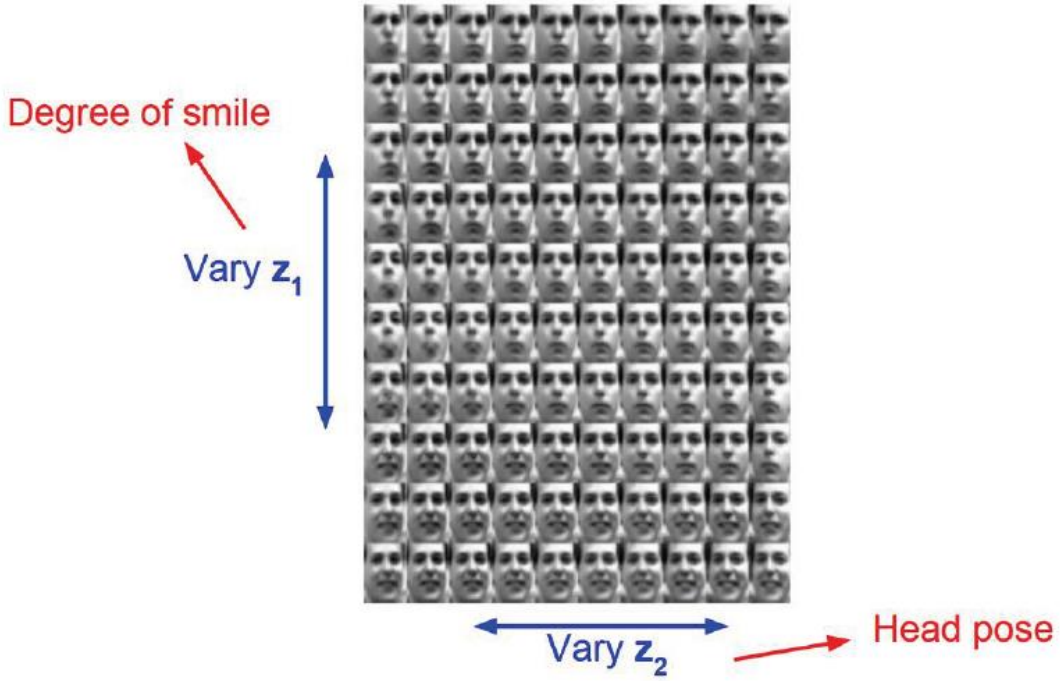


Contexte général de l'AI
Generative Models

- Quelques modèles génératifs classiques
  - Pixel RNN / Pixel CNN
    - Intérêt : modèle explicite et optimisable de manière rigoureuse
    - Inconvénient : Cout de calcul important
      - Calcul itératif pixel par pixel sur une image.
      - Difficile à paralléliser : x_i dépend de x_{i-1}
    - Optimisable avec :
      - Multiscale
      - Réduction du périmètre de connexion (CNN)
      - Heuristique d'entrainement

Contexte général de l'AI
Generative Models

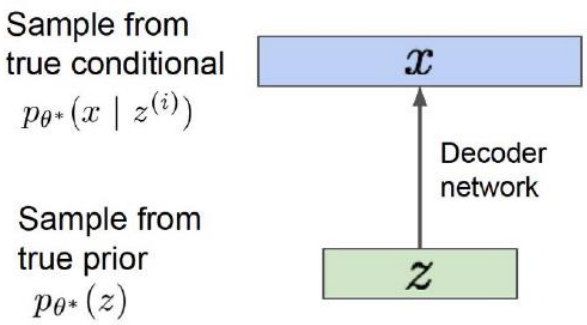
- Variational Autoencoder (VAE)
  - Idée : utiliser le code comme graine de génération de data
    - Les variations du code mènent à différentes datas



Contexte général de l'AI
Generative Models

- VAE et apprentissage
  - Maximisation de la vraisemblance avec :

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$



  - Problèmes : termes non calculables

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

Intractable data likelihood

Contexte général de l'AI
Generative Models

- VAE et apprentissage
  - Pour rendre l'apprentissage possible, on doit revenir au modèle complet avec un encodeur et un décodeur

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

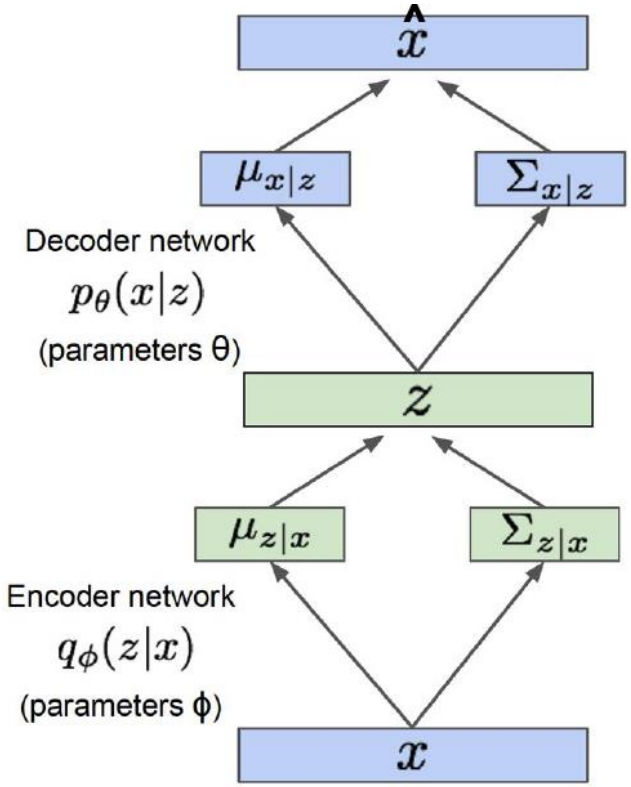
$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

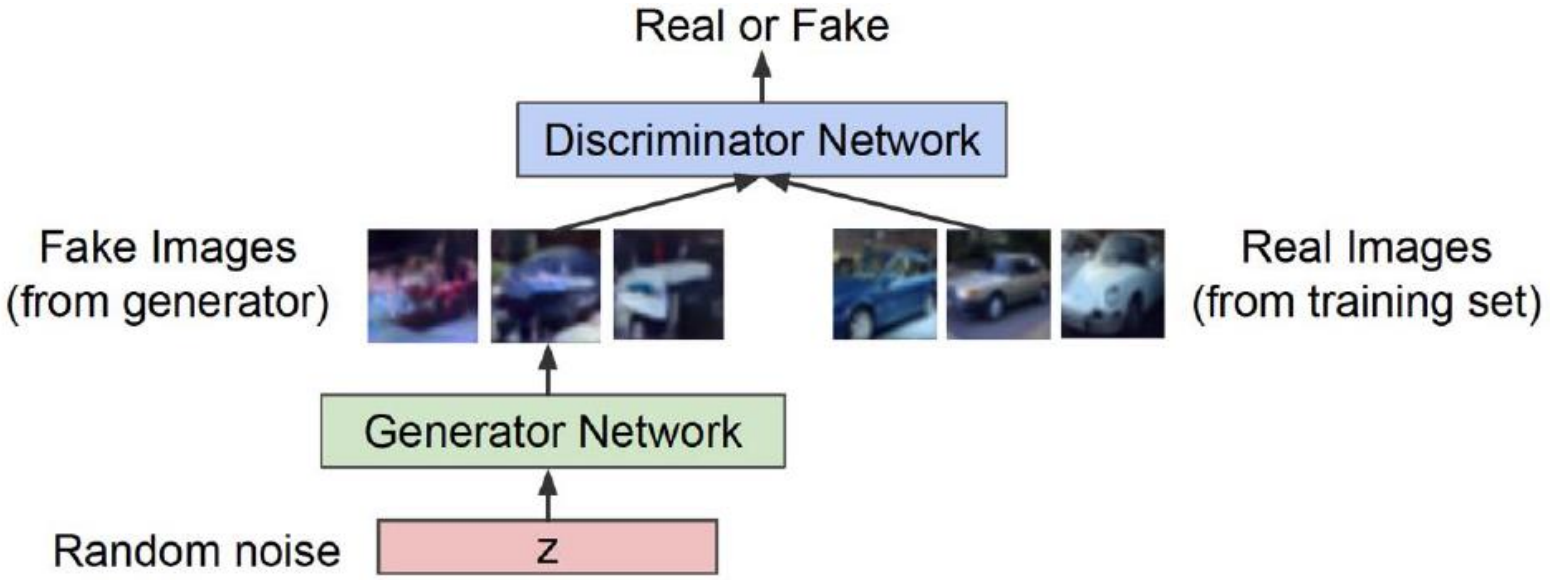


Contexte général de l'AI
Generative Models

- VAE : avantages et inconvénients
  - Avantages :
    - Méthode très utilisée
  - Inconvénient :
    - L'optimisation n'est pas exacte
      - Seule une borne basse est maximisée (le reste n'est pas calculable)
    - Résultat : les datas construites sont plus floues

Contexte général de l'AI
Generative Models

- Generative Adversarial Networks (GAN)
  - Basé sur deux réseaux
    - Un générateur essayant de générer le mieux possible
    - Un discriminateur essayant de distinguer le vrai du faux



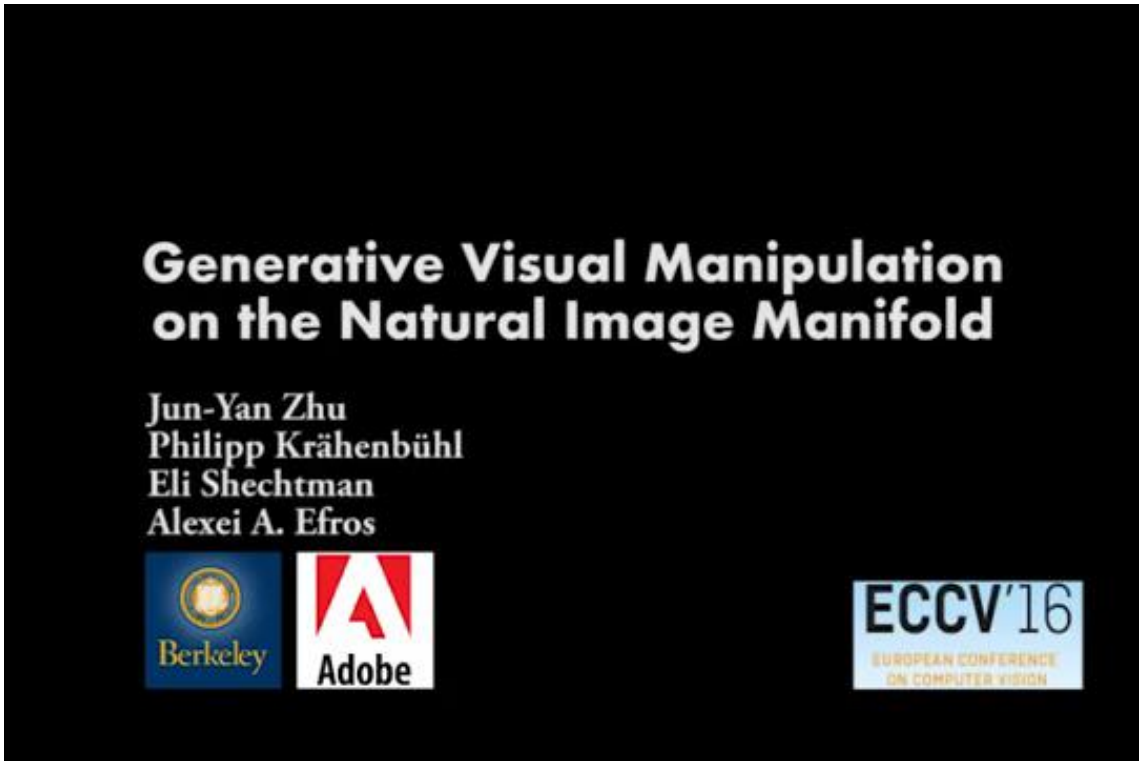
Contexte général de l'AI
Generative Models

- Generative Adversarial Networks (GAN)
  - Optimisation du réseau global

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

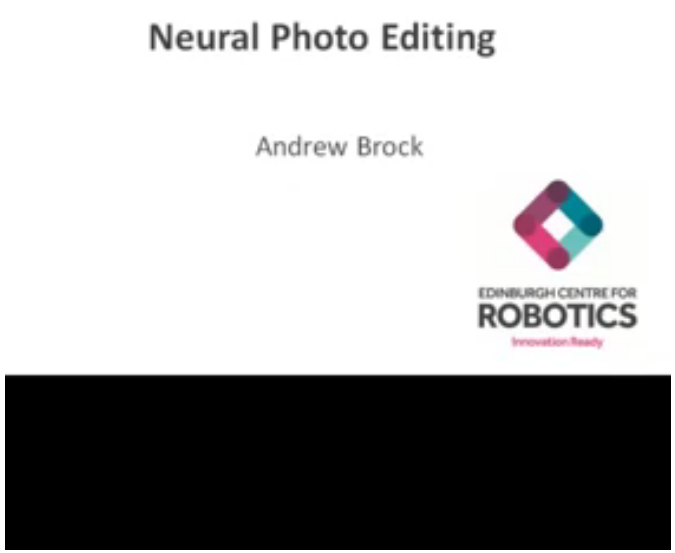
Contexte général de l'AI

- Exemples d'applications
  - Interactive GAN : https://youtu.be/9c4z6YsBGQ0



Contexte général de l'AI
Reinforcement learning

- Exemples d'applications
  - Introspective GAN : https://youtu.be/FDELBFSeqQs



INTELLIGENCE ARTIFICIELLE EMBARQUÉE

CONTEXTE DE L'AI EMBARQUÉE

Valentin Gies

M2 ROC

Université de Toulon

Contexte de l'AI embarquée

- Un concept déjà opérationnel :
  - Google « Smart Compose »
  - Apple « Siri »

- Des enjeux très forts pour les prochaines années
  - Véhicules autonomes
    - IA ultra-réactive : i.e. freinage

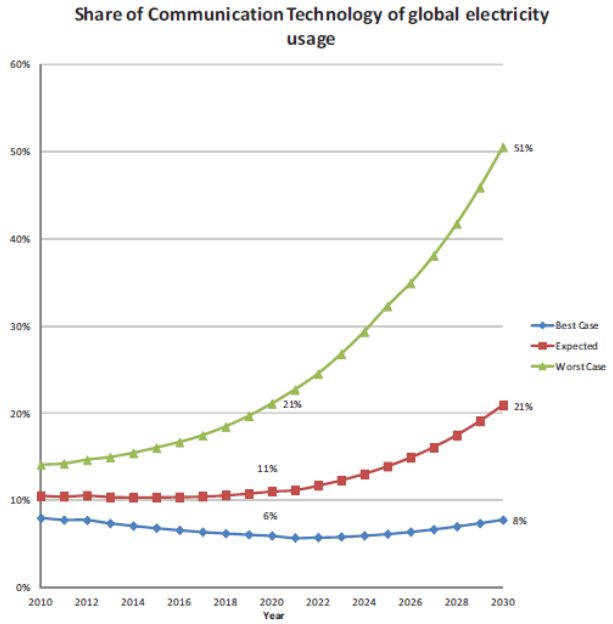
⇨ Impossible de compter uniquement sur le **Cloud**.
  - L'IA aura aussi besoin d'être traité en local, au cœur même des composants.

Contexte de l'AI embarquée : réglementaire

- Une règlementation de plus en plus restrictive
  - 2025 : 80 % du traitement des données devra être effectué dans les composants embarqués.
    - Règlementation européenne
  - RGPD : les data sensibles doivent être le moins possible transmises.
    - Risque de l'anonymisation a posteriori des données.
    - Pb du recueil de données médicales.

Contexte de l'AI embarquée : énergétique

- Consommation globale internet mondiale
  - 2015 :
    - Conso mondiale globale : 22,000 TWh
    - Conso ICT (information and communication technologies) : 1700 TWh – 7,7%
    - Référence :
      - 1réacteur nucléaire : 7 TWh
  - 2025 :
    - Conso ICT : 4600-8500 TWh
      - 14% - 25% conso mondiale
  - 2030 :
    - 21% - 50% conso mondiale



Source : Total Consumer Power Consumption Forecast – Anders S.G. – Nordic Digital Business Summit

Contexte de l'AI embarquée : énergétique

- La consommation énergétique des systèmes embarqués
  - 30Md de device connectés
    - 15Md en IoT (source Cisco 2019)
  - Le facteur d'échelle est important
    - Google Nest Thermostat
      - 0,16% of all connections – 55th mondial
    - Alexa – Siri :
      - 0,05% of all connections - 97th et 102nd mondiaux

Source : Estimating the end-to-end energy consumption of low-bandwidth IoT applications for WiFi devices - Loic Guegan, Anne-Cécile Orgerie - 2019

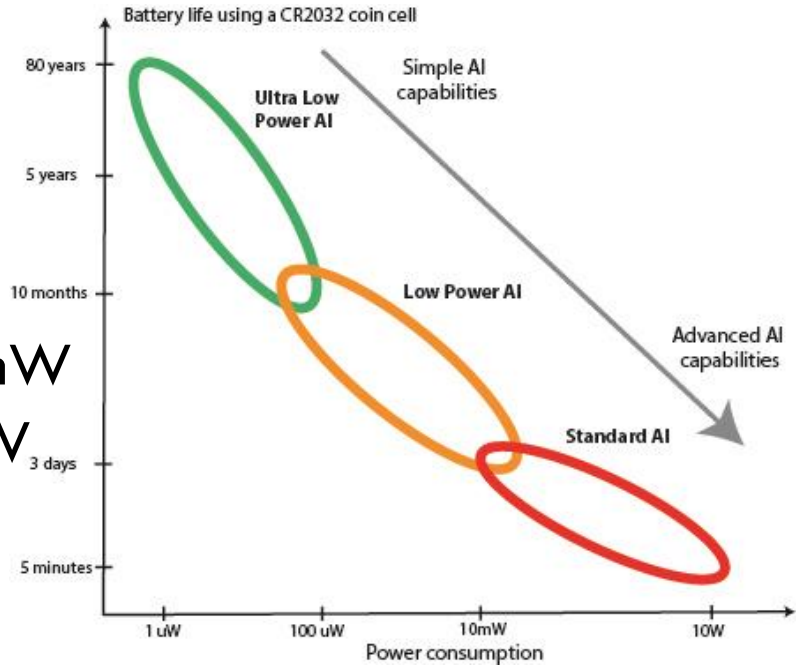
INTELLIGENCE ARTIFICIELLE EMBARQUÉE

QUEL HARDWARE POUR L'AI EMBARQUÉE ?

Objectifs de l'AI embarquée

- Cadre de travail
  - On cible l'implantation des opérations de traitement de signal après apprentissage.
    - L'apprentissage est fait offline, on ne se focalise pas dessus dans ce cours.

- Définition des types de low-power
  - Standard AI : >10mW
  - Low Power AI : 100uW – 10mW
  - Ultra Low Power AI : < 100uW
  - Reference pile CR2032
    - 240mAh sous 3V = 720mWh
    - 1uW -> autonomie : 80 ans



Objectifs de l'AI embarquée

- Puissance de calcul
  - Doit être adaptée au power mode ciblé
  - Fonctions couteuses en énergie :
    - Extraction de features en standard learning
      - Filtrage
      - FFT - Wavelets
    - Réseaux neuronaux incluant l'extraction de features de taille importante en deep learning
- Réduction énergétique
  - Réduire les échanges de données
    - Eviter les transferts de données en vue de leur traitement.
  - Utiliser des architectures low-power dédiées
    - Surtout pour l'ultra-low power

Objectifs de l'AI embarquée

- Diminution de la latence
  - Important dans les applications type robotique
- Amélioration de la sécurité
  - Résistance aux side channel attacks
  - Amélioration du respect de la vie privée
    - Respect de la RGPD

Quel hardware pour l'AI embarquée ? Primitives indispensables

- Les opérations les plus fréquentes en AI
  - Implantation efficace des neurones
    - Somme pondérée + fonction d'activation
    - Taille limitée à quelques couches
  - Extraction de features
    - Convolutions
    - Filtrage

- Le produits scalaire : l'opérateur clé en AI
  - Besoin de produits scalaires ayant un nombre arbitraire de termes
  - Besoin d'effectuer en parallèle de nombreux produits scalaires

Quel hardware pour l'AI embarquée ? Solutions technologiques

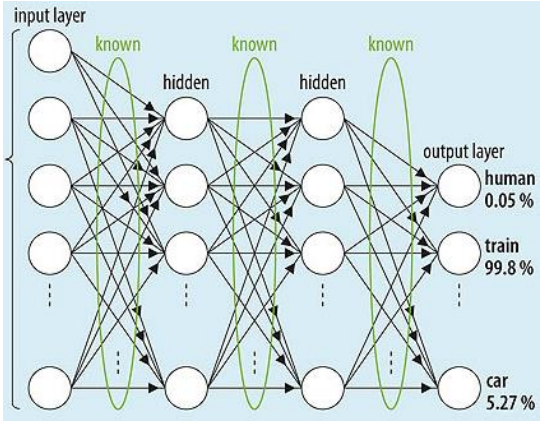
- Les solutions technologiques classiques
  - Microcontrôleurs
  - DSP : Digital Signal Processor
  - GPU : Graphic Processing Units
  - FPGA
- Les solutions dédiées AI :
  - Neural Processor / CNN processor
  - Computing In Memory (CIM) : Memristor – Crossbar
  - Reservoir computation
  - Spike Neural Networks
- What's next ?

Quel hardware pour l'AI embarquée

Architectures classiques

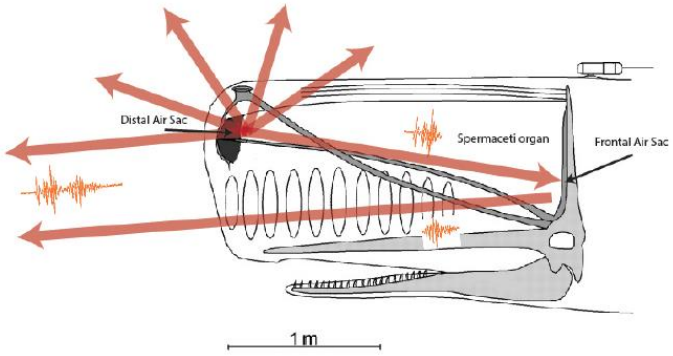
Quel hardware pour l'AI embarquée ?
ML sur microcontrôleurs

- Le microcontrôleur, un graal pour le ML low-power
  - Moins de 1$: très grand public
  - Mémoire très limitée, faible consommation
    - CC2652 : 80kB RAM
    - Conso : 71uA/MHz -> 10mW à 48MHz
  - Traitements limités
    - Ex : CC2652 - ARM M4
      - 48 MHz - 60MIPS (1,25 MIPS / MHz)
    - Cible les signaux à débit faible
      - Vibration
      - Signal acoustique
    - Ex : Always on Wake-up
    - Ex : Détection de pattern simples
  - Application souvent axée standard learning
    - Extraction de features
      - Majeur partie du processing : souvent >80 %
    - Classification
      - Effort limité car les poids sont fixés et le réseau de taille limitée



Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de système expert sur uC
  - Détection de cétacés (cachalots) sans neural network
    - Destiné à alerter les bateaux en cas de risque de collision
    - Cible de consommation : <100uW
    - Pattern : clicks
      - Bande de fréquence définie : 6kHz-20kHz
      - Répétition possible : intervalle [100ms ; 1s]



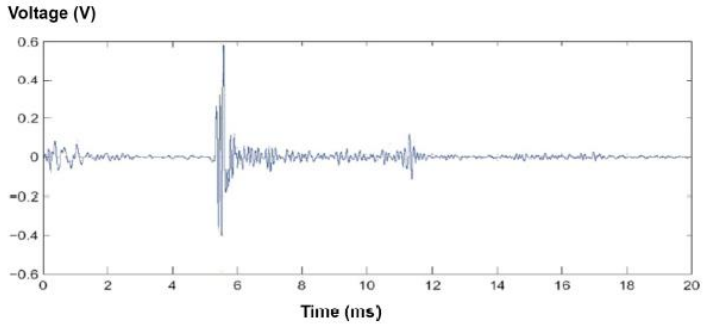
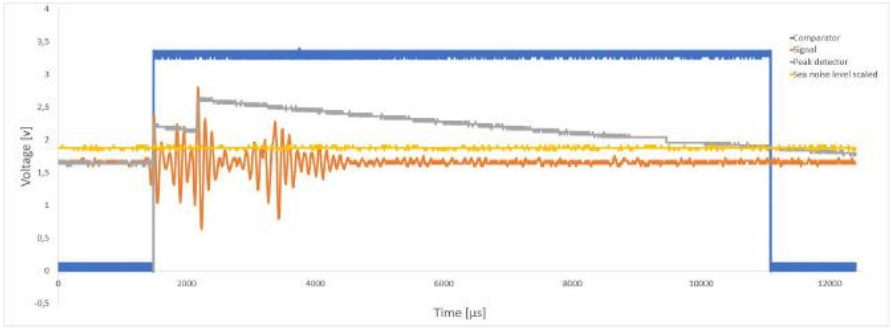
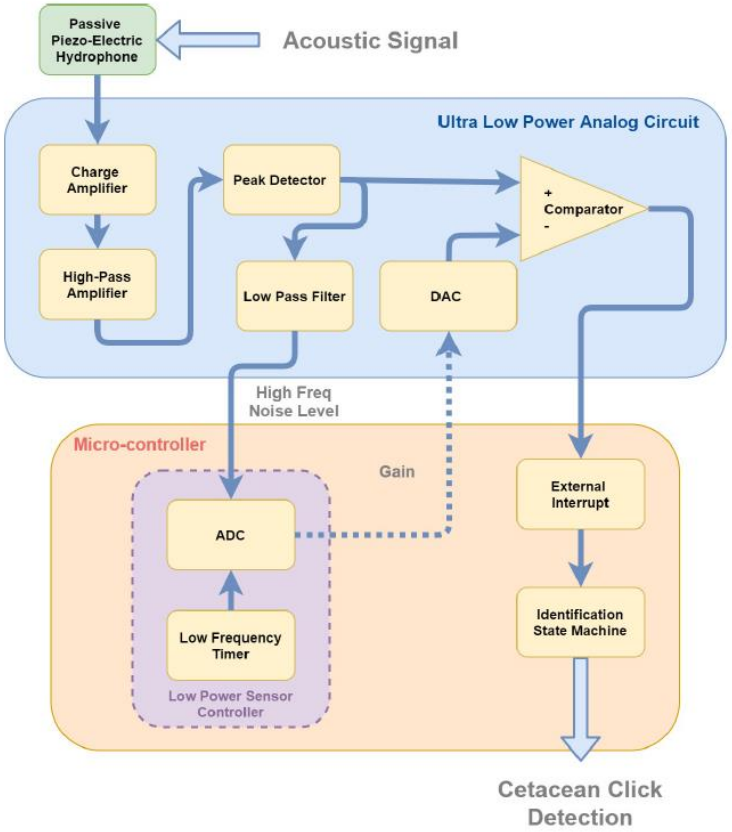


Fig. 1. Cetacean click signal.

Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

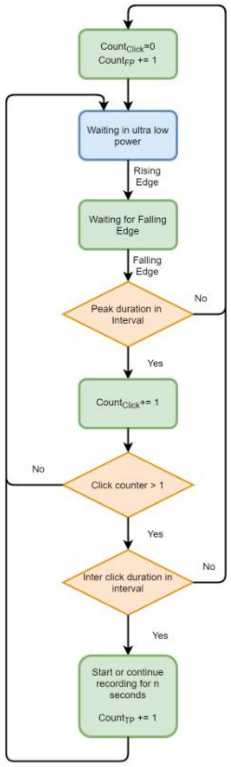
- Exemple de système expert sur uC
  - Détection de cétacés (cachalots) sans neural network
    - Architecture
      - Extraction de features
        - Analogique
        - Numérique ULP
      - Analyse des patterns par machine à état (State Machine) sur TI CC2652 (ARM M4F).

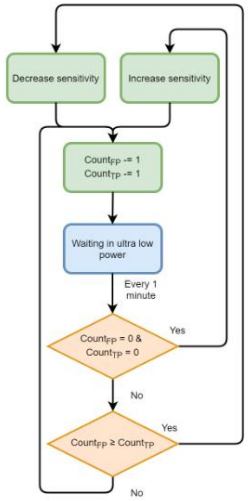




Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

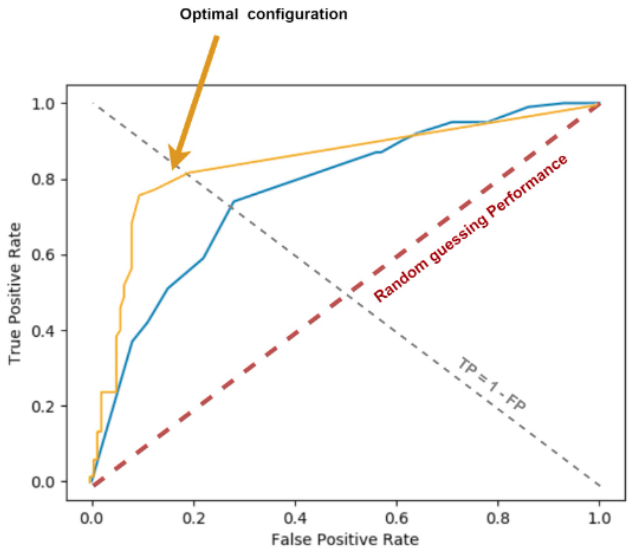
- Exemple de système expert sur uC
  - Détection de cétacés (cachalots) sans neural network
    - Analyse de patterns par State Machines
      - Machine de détection des codas
        - Implantation de règles expertes
          - Temps d'un click
          - Intervalle inter-click
      - Machine d'ajustement de la sensibilité
        - En fonction des TP et FP





Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

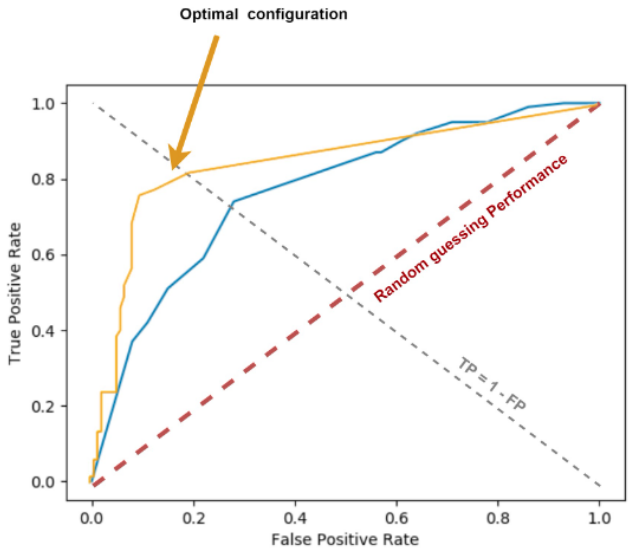
- Exemple de système expert sur uC
  - Détection de cétacés (cachalots) sans neural network
    - Résultats
      - Très bonne classification
        - AUC 85% avec state machine
        - AUC 75% sans state machine
      - Conso très réduite :
        - 12,5uA sans state machine
        - 17,5uA avec state machine.



  - Il est possible d'implanter de l'IA sur microcontrôleur classique en Ultra Low-Power

Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de système expert sur uC
  - Détection de cétacés (cachalots) sans neural network
    - Résultats
      - Très bonne classification
        - AUC 85% avec state machine
        - AUC 75% sans state machine
      - Conso très réduite :
        - 12,5uA sans state machine
        - 17,5uA avec state machine.



  - Il est possible d'implanter de l'IA sur microcontrôleur classique en Ultra Low-Power

Quel hardware pour l'AI embarquée ?
ML sur microcontrôleurs

- Exemple de supervised learning sur uC
  - Classification de vibrations pour la détection de contraction tétaniques
    - Implanté sur uC (ARM M4F)

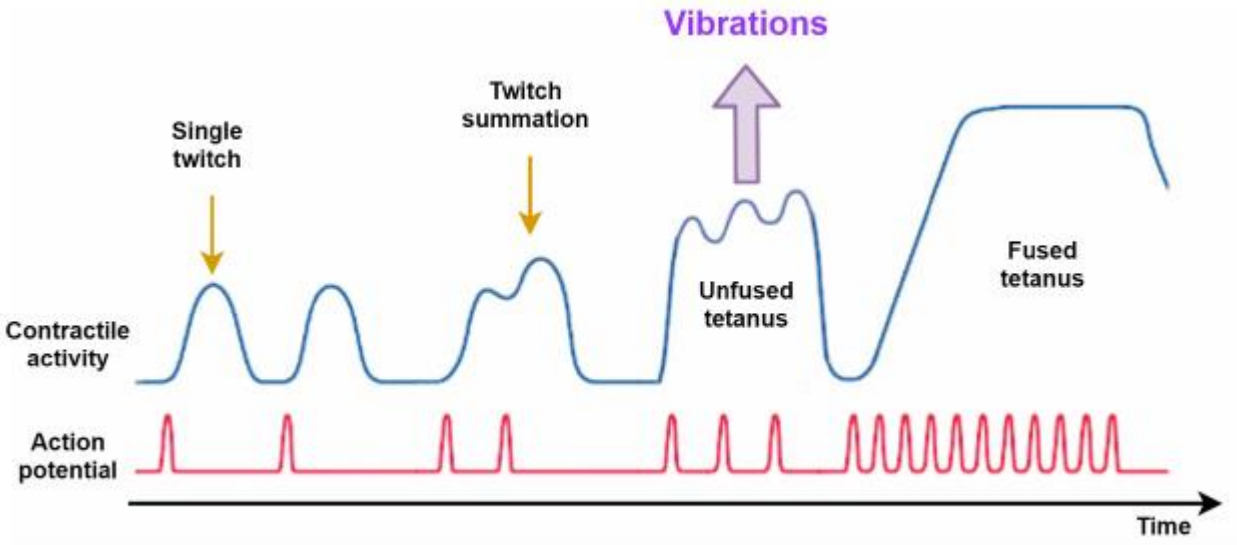


Figure 1. Tetanic contraction

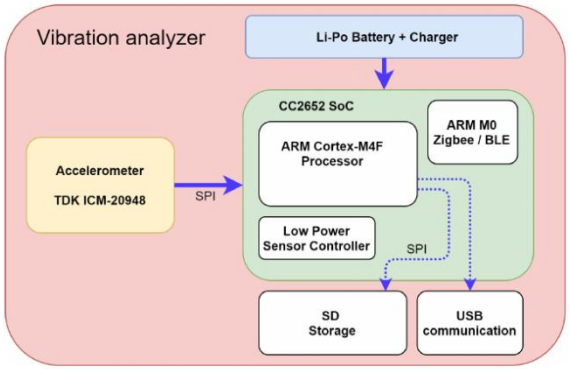


Figure 3. Vibration analyser architecture

Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de supervised learning sur uC
  - Classification de vibrations pour la détection de contraction tétaniques
    - Architecture

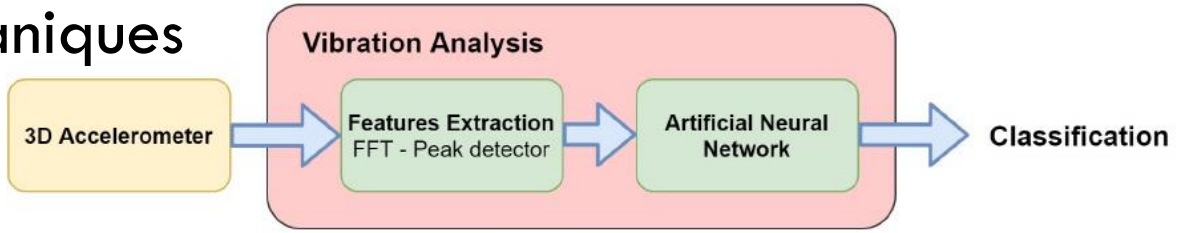


Figure 4. Feature extraction and pattern recognition chain

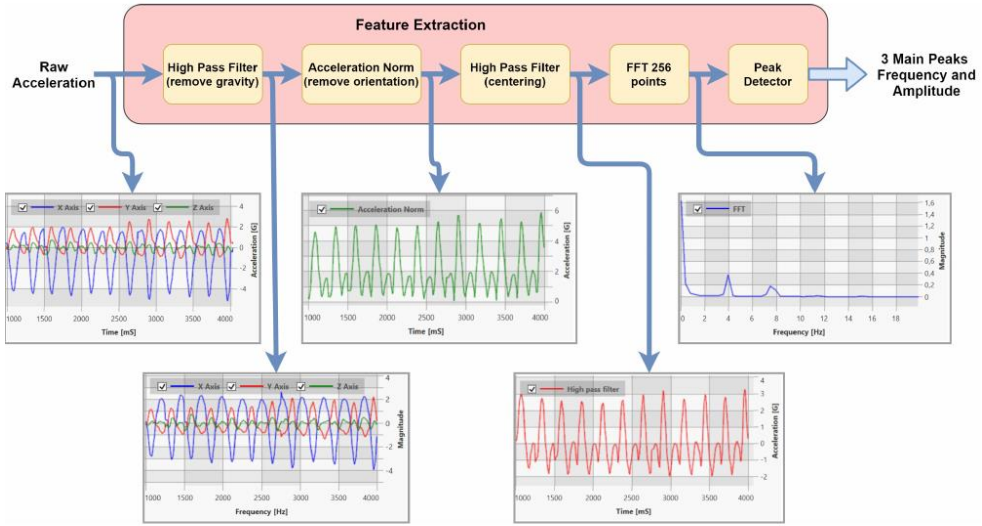


Figure 5. Features extraction module blocks diagram.

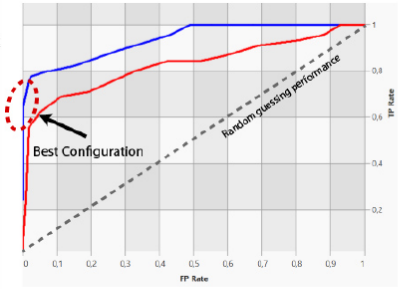


Figure 12. One hidden layer ANN ROC curves with 40 (in red) and 70 (in blue) neurons in hidden layer.

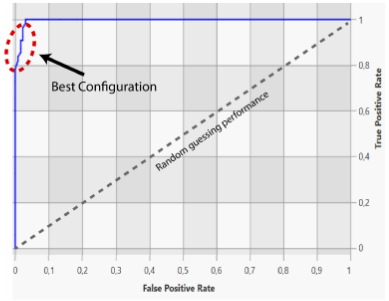


Figure 13. Two hidden layers ANN ROC curve.

Neurons in hidden layer 1	40	70	35
Neurons in hidden layer 2	—	—	35
Average Precision	0.739	0.886	0.991

Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de supervised learning sur uC
  - Classification de vibrations pour la détection de contraction tétaniques
    - Résultats avec deux hidden layers
      - 99% de précision
      - Perf avec des fenêtres de 256 samples à 100Hz
        - Tps de calcul 2,36%
          - Dont 85% pour les features

Neurons in hidden layer 1	40	70	35
Neurons in hidden layer 2	–	–	35
Average Precision	0.739	0.886	0.991

Feature extraction:

- Fast Fourier Transform: 1.774ms.
- Peak Detection: 175μs.

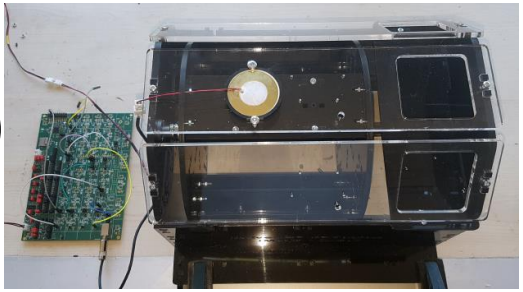
Neural Network Computation:

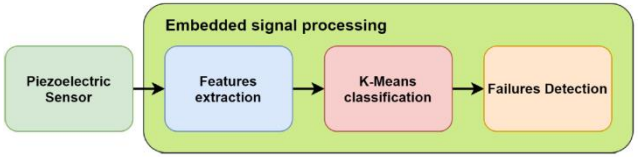
- One hidden layer of 40 neurons: 195μs.
- One hidden layer of 70 neurons: 345μs.
- Two Hidden layer, both of 35 neurons: 590μs.

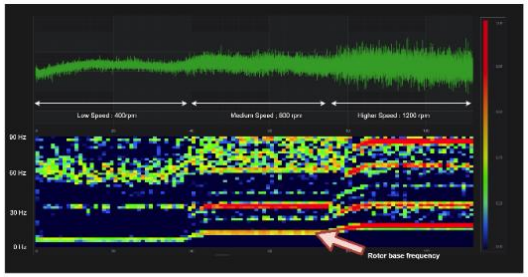
  - La classification a un temps de processing mineur devant l'extraction de features !

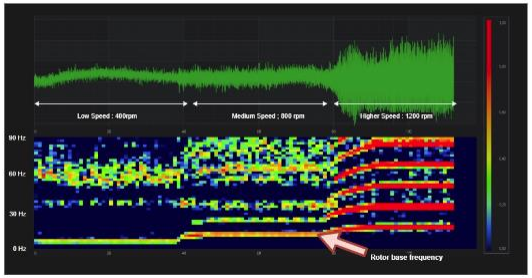
Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de unsupervised learning sur uC
  - Détection de fautes dans l'industrie 4.0
    - Implanté sur uC (ARM M4F)
      - Feature extraction : FFT (3 pics principaux)
      - Version avancée des K-Means
        - Basée sur les 3 pics de FFT
        - Temps réel
        - Implante l'oubli du passé
        - Permet l'ajout de clusters
          - On ne connait pas ce que l'on cherche





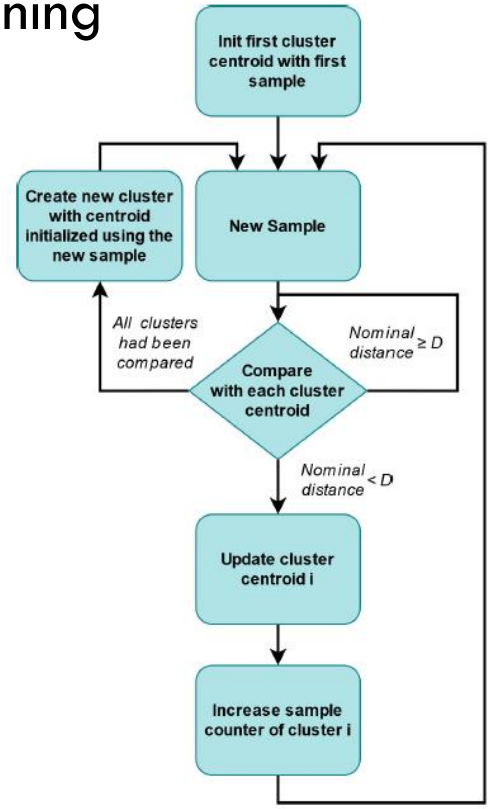




Quel hardware pour l'AI embarquée ?
ML sur microcontrôleurs

- Exemple de unsupervised learning sur uC
  - Détection de fautes dans l'industrie 4.0
    - Algorithme K-Means étendu durant le training

$$D = \sum_{k=1}^{3} \sqrt{\left(\frac{A_{i1} - \overline{A_{i1}}}{\sigma_{A_{ik}}}\right)^2 + \left(\frac{f_{i1} - \overline{f_{i1}}}{\sigma_{f_{ik}}}\right)^2}$$



    - Mise à jour

$$\overline{A_{ik}} = (1-\alpha)\overline{A_{ik}} + \alpha A_{ik}$$

$$\overline{f_{ik}} = (1-\alpha)\overline{f_{ik}} + \alpha f_{ik}$$

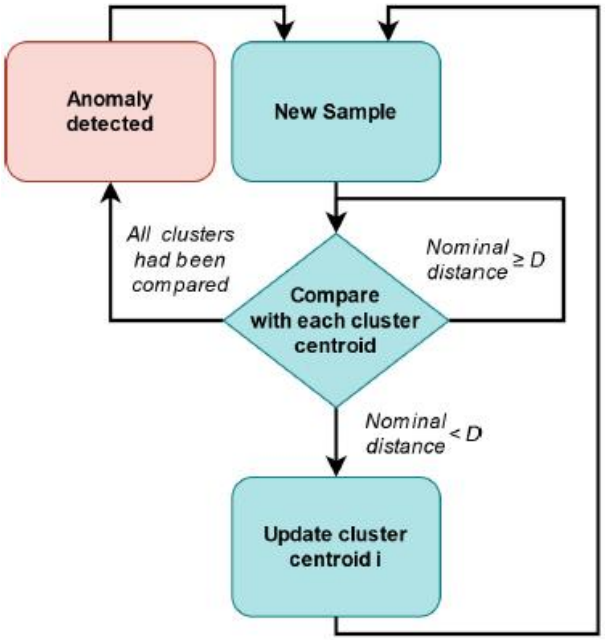
$$\sigma_{A_{ik}} = (1-\alpha)\sigma_{A_{ik}} + \alpha \left|A_{ik} - \overline{A_{ik}}\right|$$

$$\sigma_{f_{ik}} = (1-\alpha)\sigma_{f_{ik}} + \left|f_{ik} - \overline{f_{ik}}\right|$$

$$n_i = n_i + 1$$

Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de unsupervised learning sur uC
  - Détection de fautes dans l'industrie 4.0
    - Algorithme K-Means étendu durant la détection



Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs

- Exemple de unsupervised learning sur uC
  - Détection de fautes dans l'industrie 4.0
    - Résultats
      - Bonne qualité de détection

D_{Max}	0.25	0.5	0.75	1	1.25	1.5
Average Precision	0.73	0.88	0.88	0.88	0.88	0.88

TABLE I
CLASSIFICATION AVERAGE PRECISION DEPENDING ON D_{Max}.

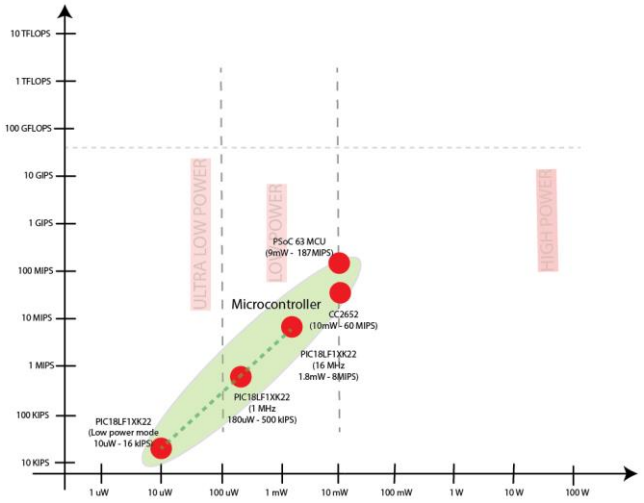
      - Usage réduit du processeur (à f_{ech} = 500Hz) :
        - Feature extraction :
          Fast Fourier Transform: $3.4ms$.
          Peak Detection: $250\mu s$.
        - K-Means Classification : $30\mu s$.
  - L'implantation de l'unsupervised learning est très légère
    - Ok sur microcontrôleur classique en Ultra Low-Power
    - Le cout en calcul vient principalement de l'extraction de features

Quel hardware pour l'AI embarquée ?
ML sur microcontrôleurs

- Exemple de unsupervised learning sur uC
  - Détection de fault dans un respirateur low-cost
    - CAS COVID 2020

Quel hardware pour l'AI embarquée ? ML sur microcontrôleurs : conclusion

- Le microcontrôleur, un graal pour le ML low-power
  - Pertinent dans le domaine des smart sensors :
    - Applications ayant des entrées à bande passante limitée
    - Permet l'ultra low-power en mode dégradé
  - Le cout algorithmique des extractions de features peut être réduit drastiquement en utilisant des circuits analogiques
    - Permet d'aborder les cas ultra low-power
  - Ne permet pas l'implantation des Convolutional Neural Networks (CNN)
    - Nécessité d'architectures dédiées et plus performantes
      - DSP
      - DPS multi-core
      - GPU



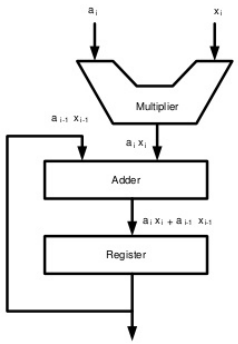
Quel hardware pour l'AI embarquée ?
ML sur Digital Signal Processor

- Les causes des limitations des uC
  - Le produit scalaire prend beaucoup de temps
    - Environ 30 cycles par terme
      - Principalement dû au multiplieur software

```
LOOP:   MOV.W   @r10,r13        /* Read of sample z[i] */          Source operand loading
        MOV.W   @r11,r12        /* Read of coefficient c[i] */
        CALL    #__mpyi         /* Multiplication */               MAC operation
        ADD.W   r12,r9          /* Accumulation  */
        MOV.W   r4,0(r15)       /* Shift data in the z buf */      Sample delay
        SUB.W   #2,r11          /* Pointer management */           Address management
        SUB.W   #2,r10
        SUB.W   #2,r15
        SUB.W   #1,r5           /* Loop management */              Loop management
        JNE     #LOOP
```

- Le DSP, une architecture axée produit scalaire :
  - Rend efficace:
    - Filtrage
    - Convolution
    - Opérateurs neuronaux

Single-Cycle MAC unit



$$\sum_{i=0}^{n}(a_i x_i)$$

Can compute a sum of *n*-products in *n* cycles

Quel hardware pour l'AI embarquée ? ML sur Digital Signal Processor

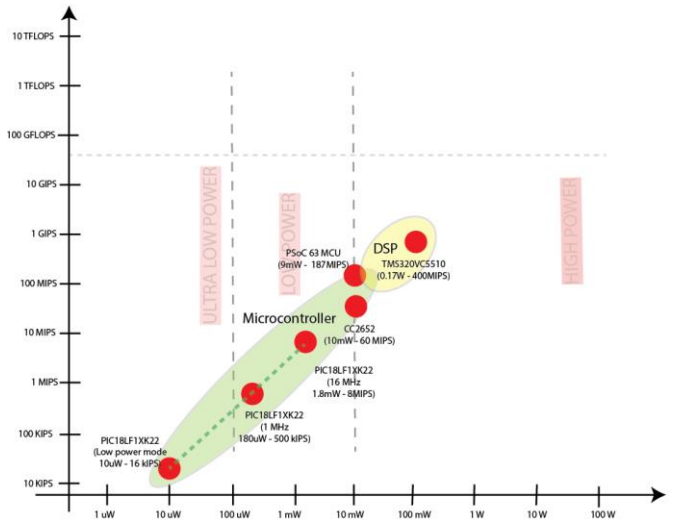
- Les particularités d'un DSP:
  - MAC unit
    - 1tap de produit scalaire en 1 cycle
      - Environ 20-30 fois plus rapide qu'un MCU.
    - Ne remplace celui de l'ALU
      - Code spécifique
  - Opérateurs arithmétiques et logiques câblés :
    - Rounding et Saturation
    - Barrel shifter
      - division ou multiplications par puissances de 2
  - Jeu d'instruction orienté traitement de signal
    - Filtrage : FIRS (symmetric FIR filter) / LMS
    - Math : ABS / Distance Quadratique (vecteurs)…
  - Multiples accès mémoire possibles en simultané

Quel hardware pour l'AI embarquée ? ML sur Digital Signal Processor

- Ce que l'on peut espérer avec un DSP:
  - Environ 20-30 fois plus rapide sur les produits scalaires.
    - TMS320VC5510 : 200MHz – 400 MIPS sur produit scalaire (MMACS)
  - Traitement de signal monodimensionnel à fréquence élevée devient possible
    - Ex : Parole
- Ce que l'on perd avec un DSP
  - Low Power : les DSP ne sont pas orientés low-power
    - Ex : TMS320VC5510 : 170mW à 200MHz
    - Compilation utilisant les MAC units non optimale
    - Une partie de ce qui pourrait être gagné en vitesse ne l'est pas.
      - Perte d'un facteur 4-5 en perf par rapport à l'optimal
  - MAC en virgule fixe
    - Nécessaire pour un maximum d'efficacité.
    - Complexe à mettre en œuvre

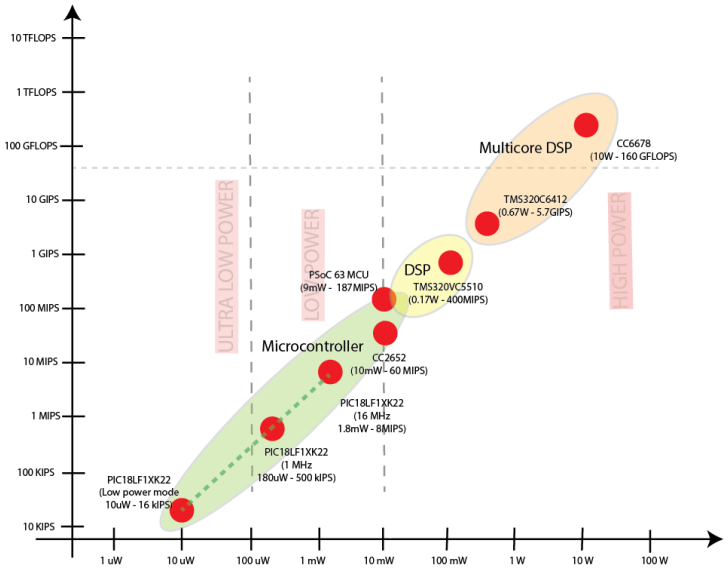
Quel hardware pour l'AI embarquée ?
ML sur DSP : conclusion

- Le DSP Classique : pertinent pour le traitement de signal à bande passante modérée
  - Traitement de la parole
    - Permet d'implanter du filtrage
    - Limité pour implanter des CNN sur des signaux unidimensionnels (trop de produits scalaires)
  - Fonctionnement de type low-power à normal
    - Perte du fonctionnement ultra low-power
    - Le low-power n'est accessible qu'en mode dégradé
  - Code bien plus complexe
    - Virgule fixe
    - Compilation non optimale



Quel hardware pour l'AI embarquée ? ML sur DSP Multi-Cœurs

- Les causes des limites du DSP
  - Les opérations parallélisables sont séquentialisées
    - Pas de parallélisme hardware
  - La fréquence de base un peu faible
- Les DSP multi-cœurs permettent de paralléliser
  - Ex : CC6678
    - 8 cœurs - 1,4GHz – 160 GFLOPs
    - 10W
  - Ex : TMS320C6412
    - 8 cœurs - 720 MHz – 5760 MIPS
    - 0,67W (0,93mW /MHz)

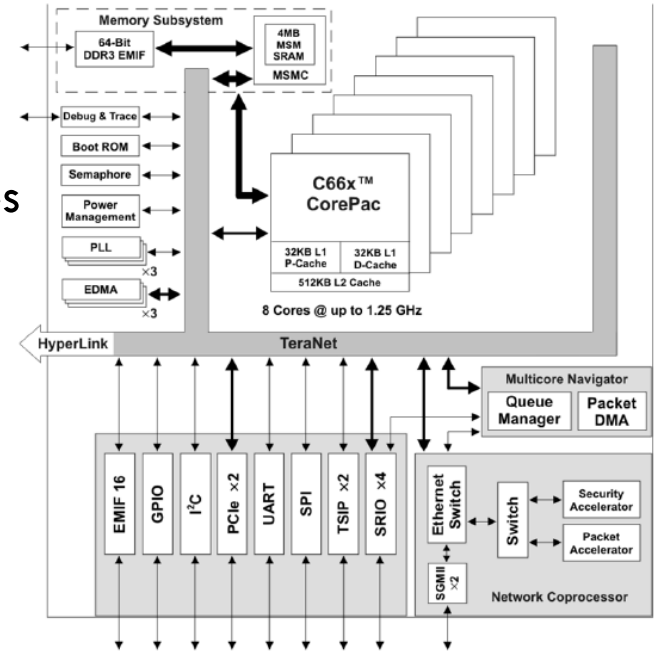


Quel hardware pour l'AI embarquée ? ML sur Processeurs Multi-Cœurs

- Intérêt d'un DSP multi-cœurs
  - Optimisation de la vitesse de calcul
    - Sur les parties parallélisables du code
      - Attention à l'écriture des algos !
    - Dépend du type de tâche
      - Gain Inférieur au nombre de cœurs
        - Ici de 3-6 sur un CC6678 8 cores

Table 2. Speedup of multi-core parallel.

FLOPs	Time of Single-Core	Time of Multi-Core
10M	0.01s	0.004s
100M	0.06s	0.02s
1G	0.61s	0.20s
10G	5.37s	1.07s
100G	55.52s	8.95s

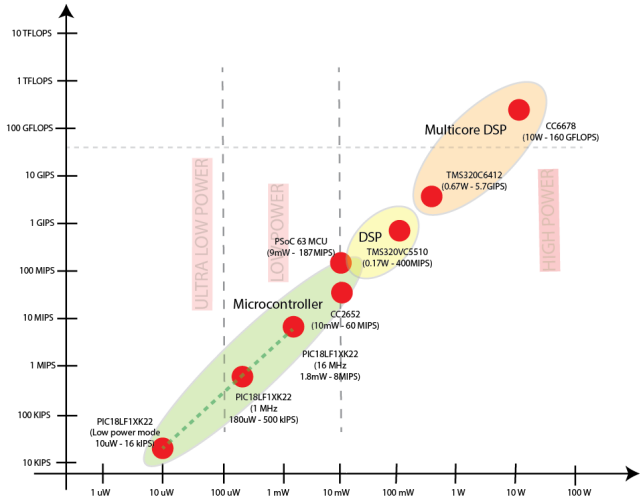


Quel hardware pour l'AI embarquée ?
ML sur DSP Multi-Cœurs

- Ce que l'on perd avec des processeurs multi-cœurs
  - Low Power impossible
    - Ex : TMS320C6412 : 720 MHz – 5760 MIPS
      - À 720 MHz : 0,67W (0,93mW /MHz)
    - Ex : CC6678 : 10W
  - Le gain n'est pas proportionnel au nombre de cœurs
    - Nécessité de synchroniser
  - Le nombre de cœurs est limité
    - Souvent 8, voir 16.

Quel hardware pour l'AI embarquée ? ML sur DSP Multi-Cœurs : conclusion

- Les processeurs multi-cœurs : pertinent pour le traitement de signal à bande passante élevée (sauf image)
  - Traitement de la parole
    - Permet d'implanter des CNN efficacement
  - Fonctionnement high power (10W)
    - Perte du fonctionnement low-power
  - Tout n'est pas parallélisable

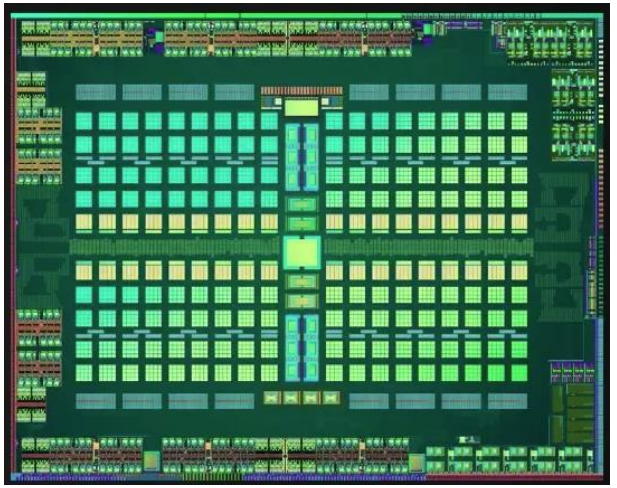


Quel hardware pour l'AI embarquée ?
ML sur GPU

- Les causes des limites des DSP multi-cœurs
  - Le nombre de cœurs est limité et pas du tout en rapport avec le nombre de neurones (au moins d'une couche).
- Pour aller plus loin en performance il faut paralléliser massivement :
  - GPU : milliers de cœurs
  - FPGA : possibilités limitées seulement par la taille du silicium (et le cout…)

Quel hardware pour l'AI embarquée ?
ML sur GPU

- Intérêt d'un GPU
  - Architecture
    - Parallélisation massive des cœurs
      - Ex : GTX 1080 :
        - 2560 cœurs CUDA
          - RAM :
        - 20 Streaming Processor
          - Allocation optimale de mémoire
        - 160 unités texture
          - Rotation, déformation, projection de bitmap
        - 7,2 Mds transistors
          - Max 50Mds sur FPGA
    - Puissance de calcul importante :
      - 8 TFLOPS
    - Consommation élevée : 165W

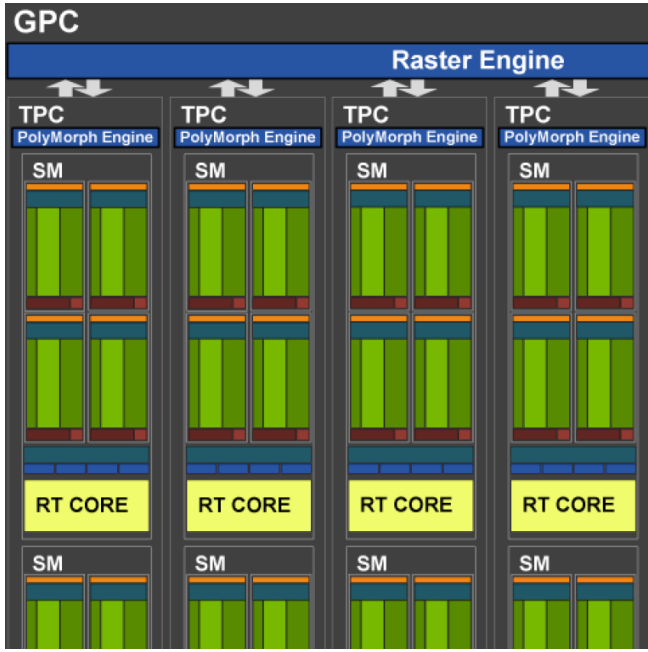


Quel hardware pour l'AI embarquée ?
ML sur GPU

- Une techno à développement rapide
  - Ex : RTX 2080 :
    - 4352 cœurs CUDA
    - 68 Streaming Processor CUDA Cores
    - 544 Tensor cores
      - Core mixte (fixed point et floating point)
      - Gain en vitesse de x10
      - Evite de sous-utiliser les cores
    - 8 Streaming Processor Tensor Cores
    - 68 RT Cores
      - Dédié au ray tracing
    - 34 Texture Processor Core (TPC)
      - Rotation, déformation, projection de bitmap
  - Caractéristiques de la RTX 1080
    - 18,6 Mds transistors
    - Puissance de calcul importante :
      - 28,5 TFLOPS en FP16
    - Consommation élevée : 260W

Quel hardware pour l'AI embarquée ?
ML sur GPU

- Architecture d'un GPU (Nvidia Pascal)
  - Ensemble de streaming multiprocessors (SM)
    - 64-128 GPU cores dans chaque SM
      - 1 FLOP/core par cycle d'horloge
    - Intègre un cache de niveau 2
      - Stockage variables intermédiaires
    - Host interface (NV Link) : communication avec le processeur -> 200 GB/s
    - DRAM interfaces : communications multiples avec la mémoire externe.
      - 4096 bit bus ! – 720 GB/s
  - GigaThread scheduler
    - Affectation des taches au SMs
  - Chez nVidia les architectures ont des noms : Turing > Pascal > Maxwell > Kepler > Fermi

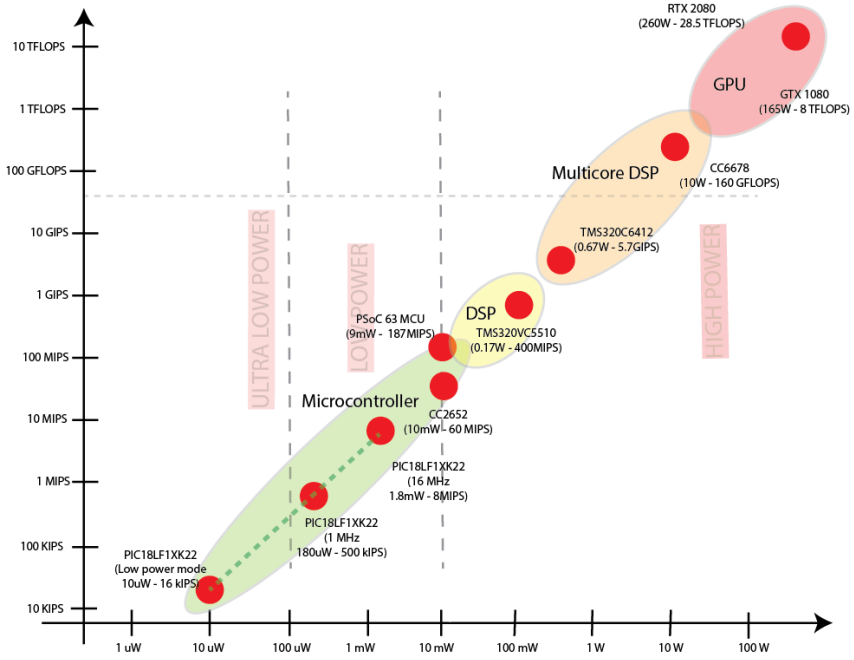


Quel hardware pour l'AI embarquée ?
ML sur GPU

- Programmation d'un GPU
  - Une programmation très orientée par le hardware
    - Les datas devant être traitées en parallèle doivent être placées dans des registres spécifiques.
    - Les opérations GPU utilisent des instructions particulières balisées dans des sections spécifiques de code et accessible via un framework (ex: CUDA sur GPU NVidia)
    - L'utilisateur ne gère pas la répartition des datas dans les cores, ni la mémoire du GPU. Les SM le font (efficacement, voir optimalement)
      - Les threads peuvent coopérer dans un même SM seulement
      - Les threads sont regroupés en warps (souvent 32 thread par warp au sein d'un même SM
        - 2 warps peuvent exécuter des codes différents dans un même SM
          - Le taux d'utilisation des cores peut être élevé.
  - Contraintes
    - Les branchements conditionnels sont peu adapté au parallélisme

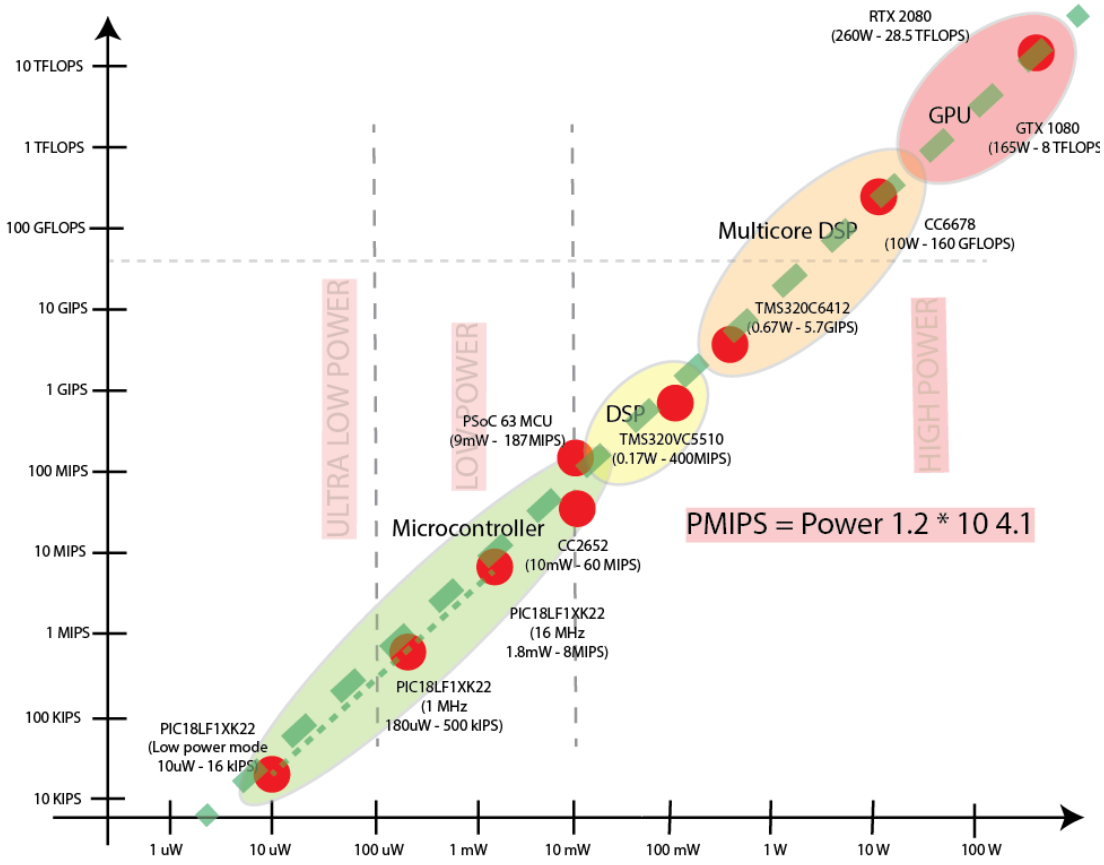
Quel hardware pour l'AI embarquée ?
ML sur GPU : conclusion

- Intérêt des GPU
  - Permet d'implanter l'exécution de CNN sur des images en temps réel
    - Yolo / Tensor Flow / PyTorch
    - Vidéo RoboCup
- Inconvénient
  - Consommation élevée !
  - Difficile à intégrer dans un système autonome



Quel hardware pour l'AI embarquée ? ML sur architectures classiques

- Relation puissance de calcul – consommation
  - $P_{Calcul}\ (FLOPS) = P_{elec}(W)^{1,2} * 10^{4,1}$
    - $R^2 = 99\%$
  - Course à la puissance

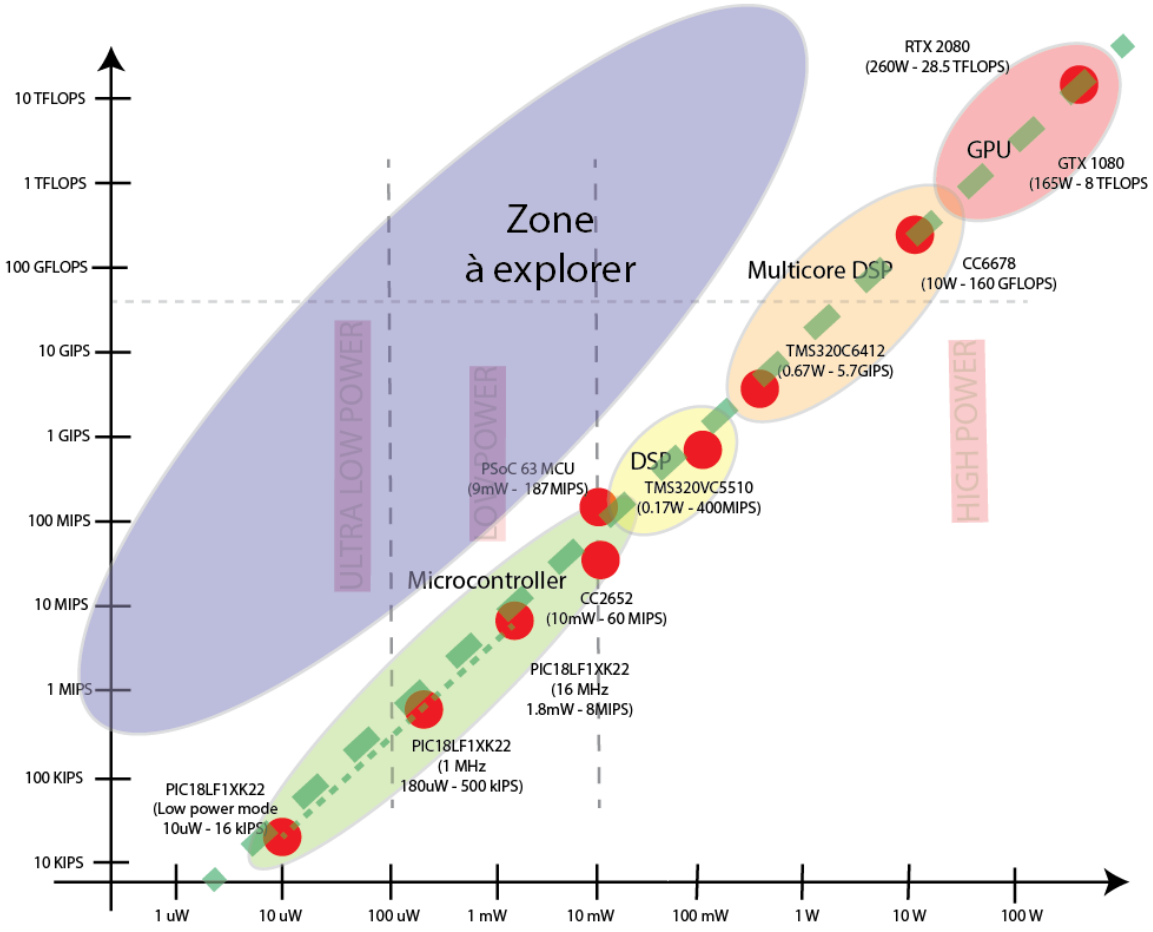


Quel hardware pour l'AI embarquée ? ML sur architectures classiques

- Chacune des architectures classiques a son domaine de pertinence
  - Low power Smart sensors : MCU
  - Advanced sensors : DSP
  - Speech recognition : DSP multi-core
  - Video analysis : GPU
- Les architectures classiques sont… classiques
  - Impossible de quitter la relation puissance de calcul – consommation

Quel hardware pour l'AI embarquée ?
ML sur architectures classiques

- Enjeu : aller au-delà de ce paradigme

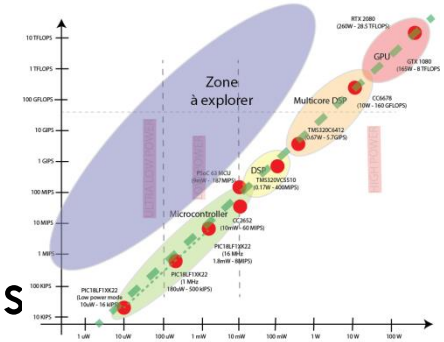


Quel hardware pour l'AI embarquée

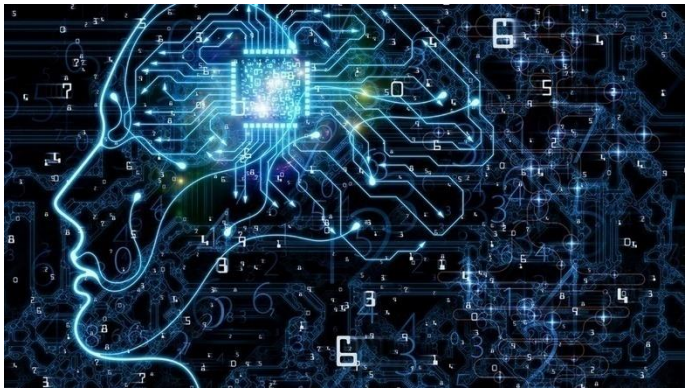
Approches neuromorphiques

Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Enjeu :
  - Réduire d'un facteur 100-1000 la conso à puissance de calcul constante
  - Ne peut pas se faire avec les architectures classiques

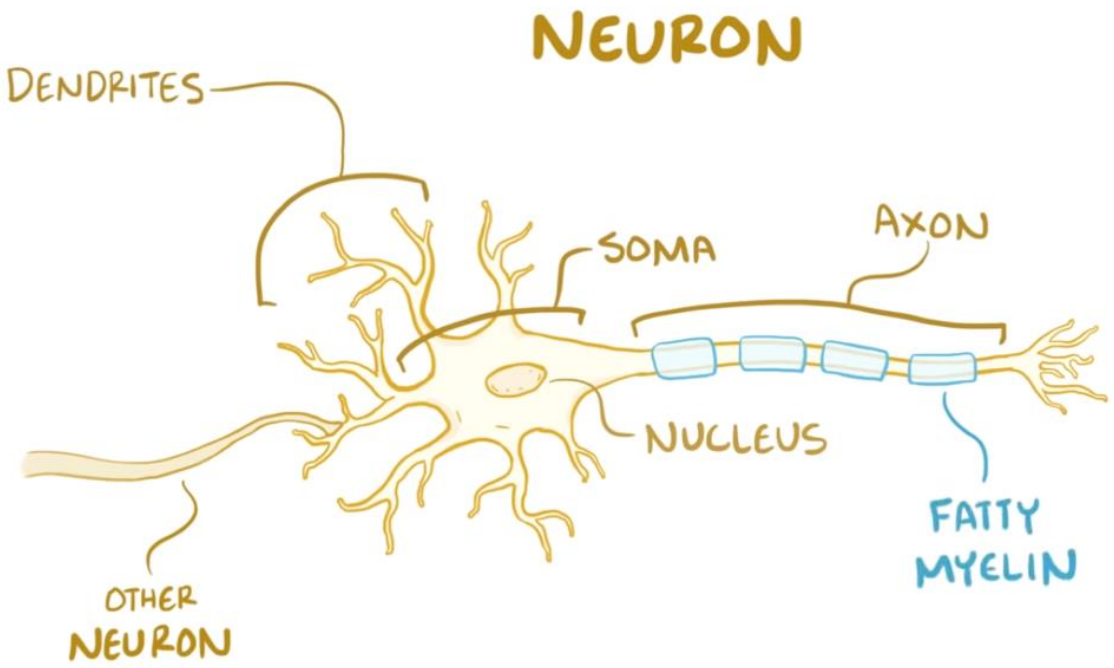


- Idée : aller vers le fonctionnement du cerveau
  - AlphaGo (2016) : 20kW !!!! – cerveau : 20W
  - Cerveau
    - 100 Mds de neurones
    - 10,000 fois plus de synapses
    - Pas de séparation calcul-mémoire
  - Transistor = interrupteur



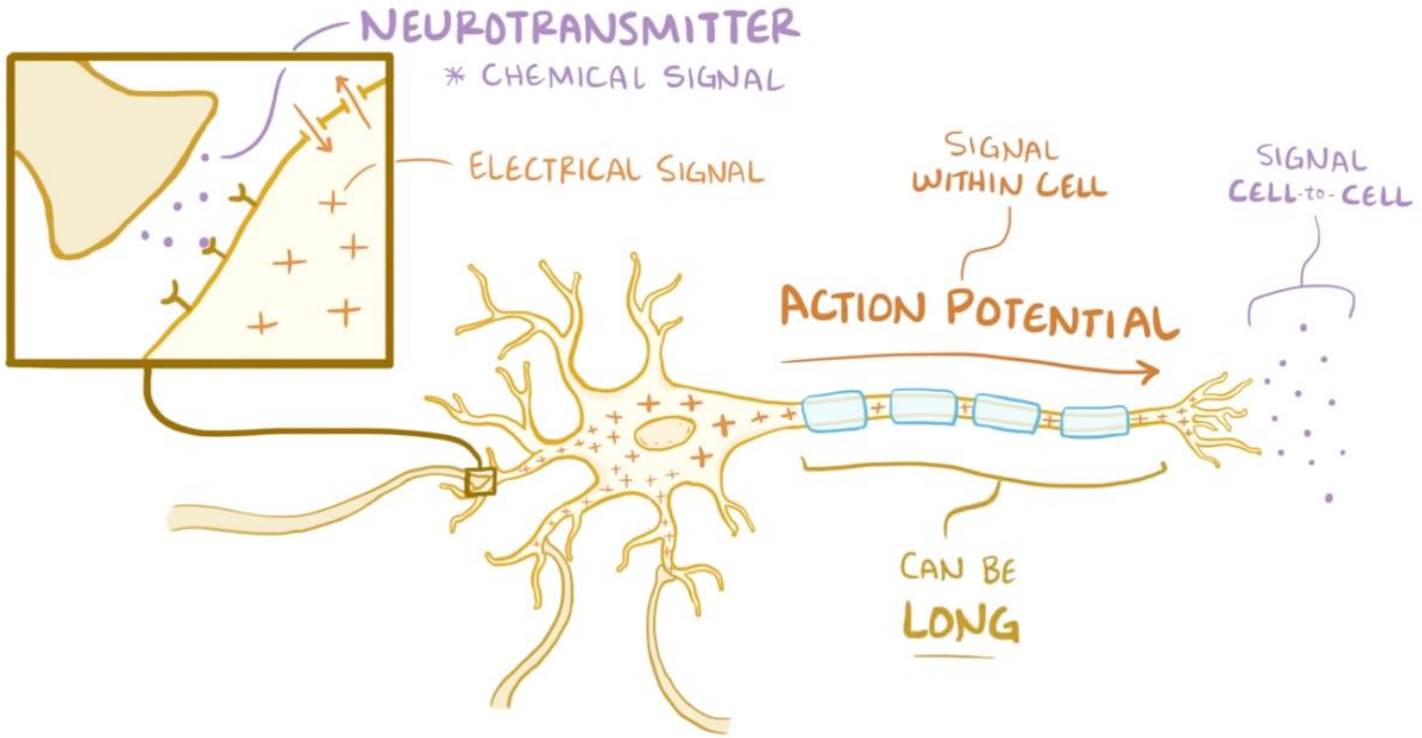
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
- Présentation (source Osmose) : https://youtu.be/BbUcWbtVjT4



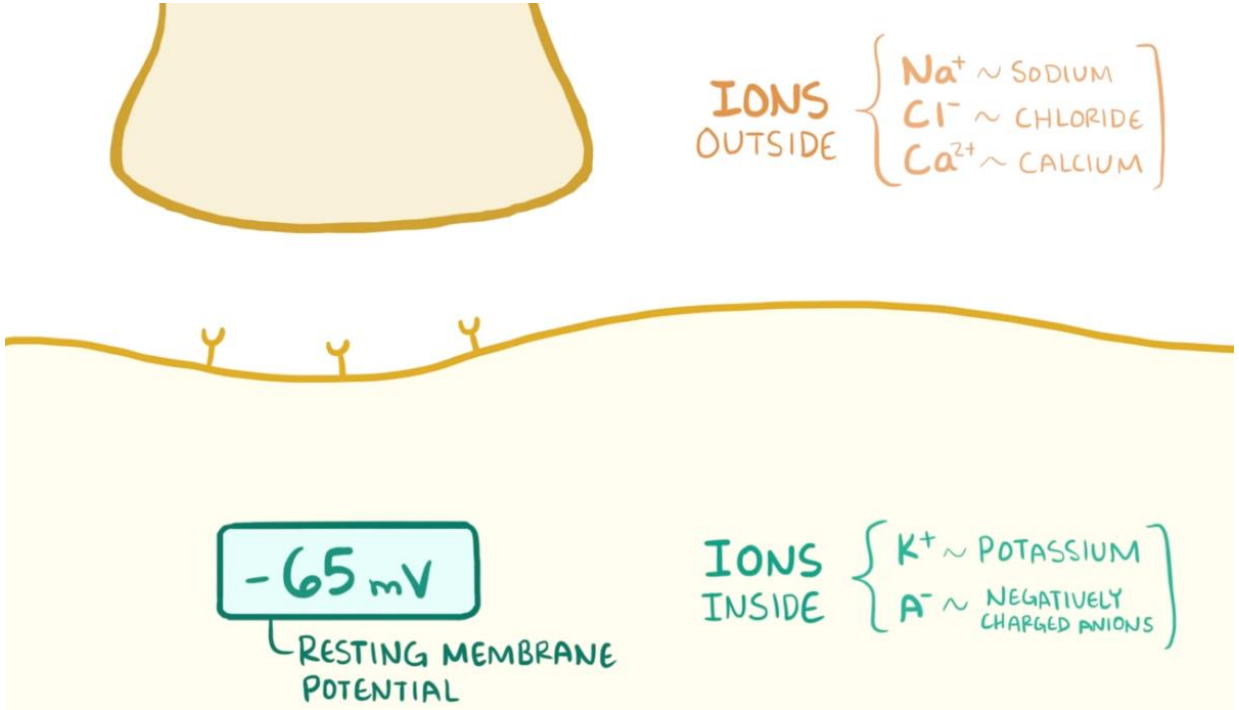
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Présentation : https://youtu.be/BbUcWbtVjT4



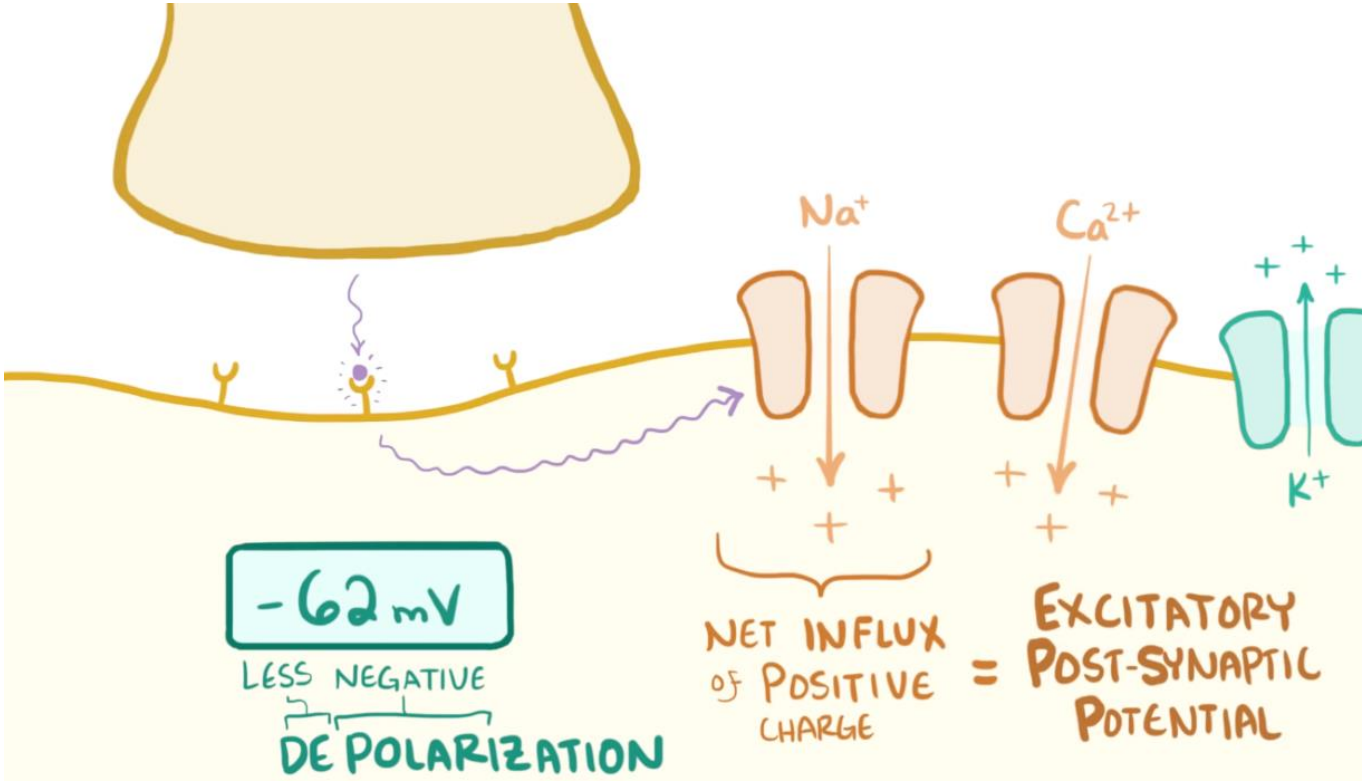
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Etat de repos d'un neurone



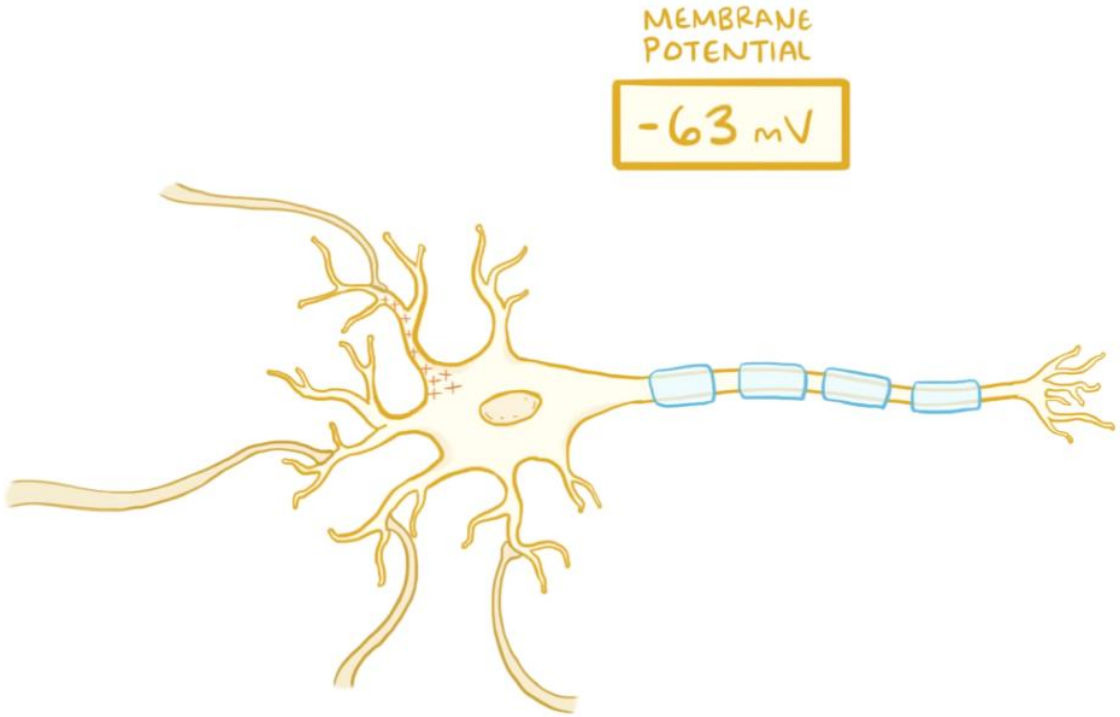
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Effet local de l'arrivée de neurotransmetteurs sur les synapses



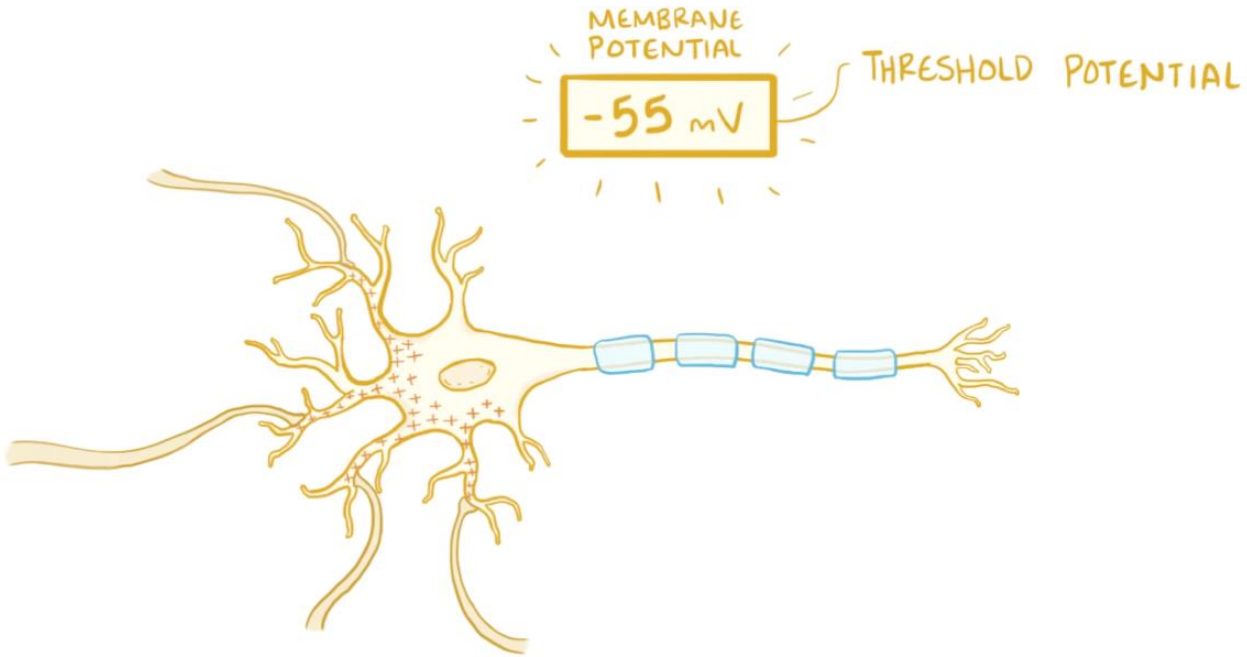
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Effet global de l'arrivée de neurotransmetteurs sur quelques synapses : le potentiel global monte un peu



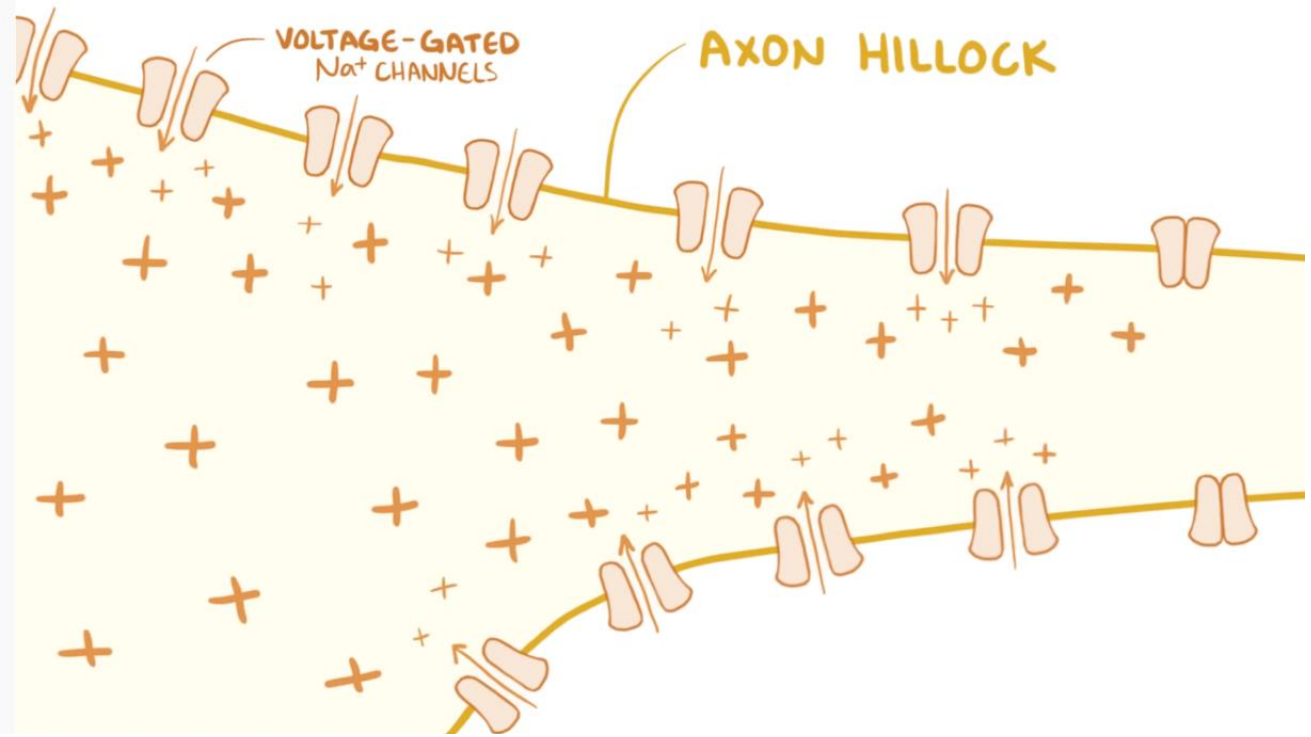
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Effet global de l'arrivée de neurotransmetteurs sur beaucoup de synapses : on dépasse le potentiel de déclenchement



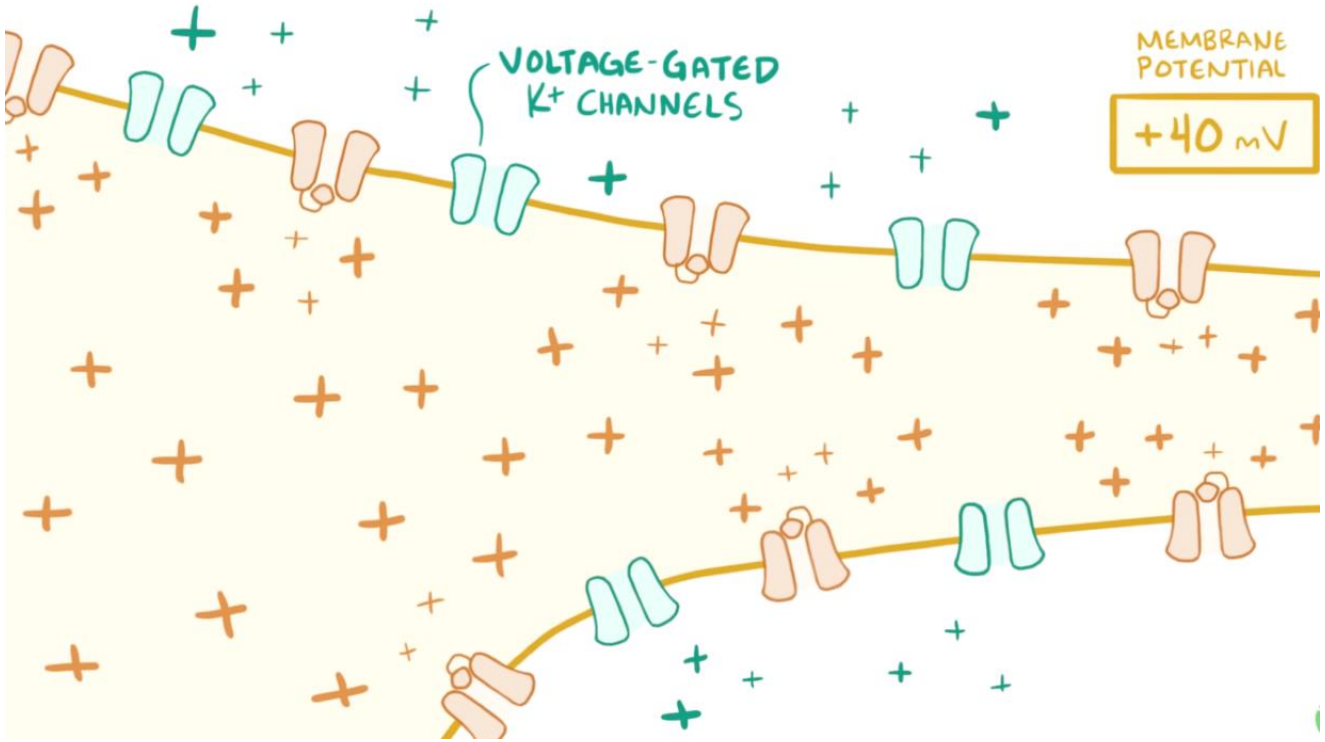
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Le franchissement du seuil déclenche une propagation de potentiel (Na+ rush)



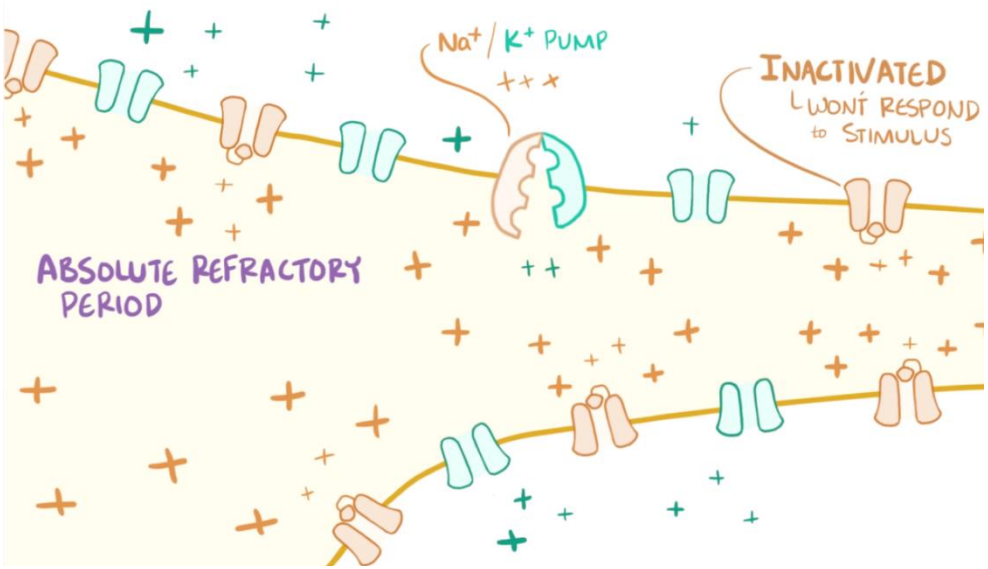
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Après le rush : équilibrage par sortie d'ions K+



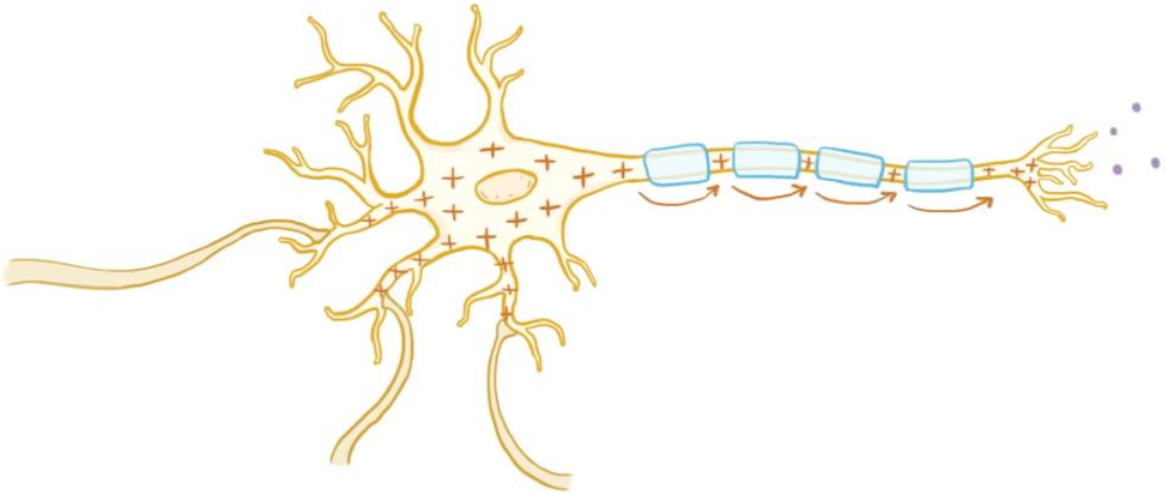
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Puis pompage des ions Na+ vers l'extérieur et K+ vers l'intérieur pour revenir en état d'origine
  - Important : le neurone ne peut être stimulé durant cette phase



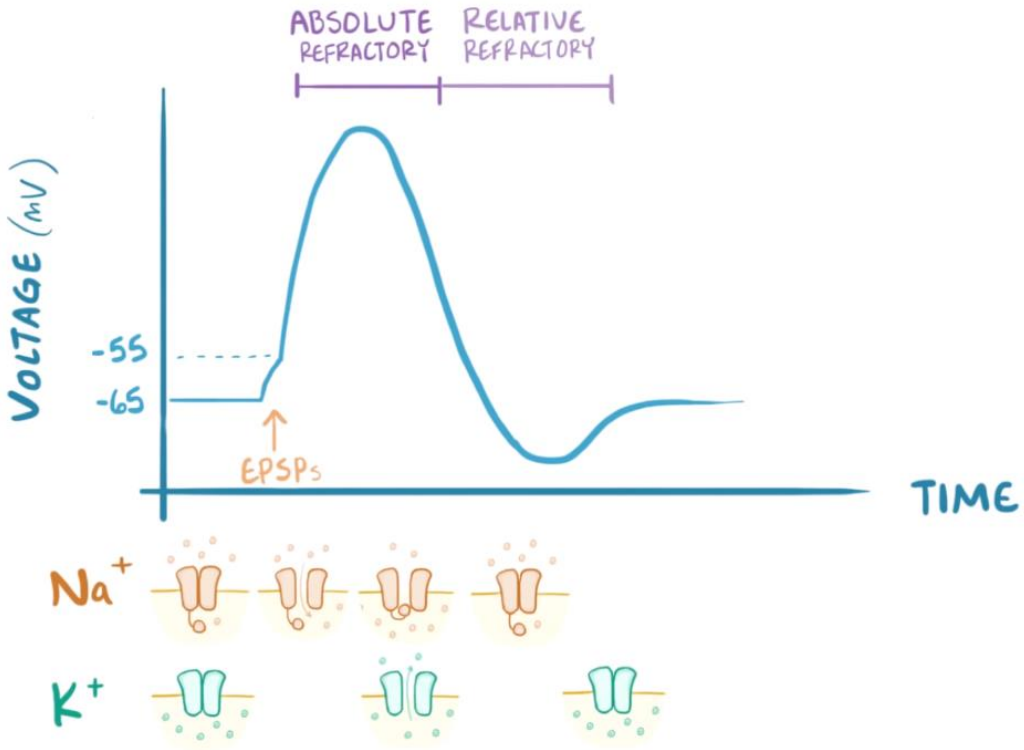
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Propagation du potentiel dans l'axone :
    - Renforcement régulier via des canaux Na+ : le signal ne peut remonter
    - Propagation rapide dans les zones de myéline



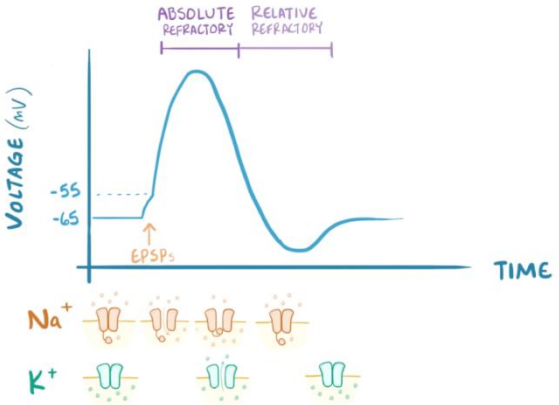
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

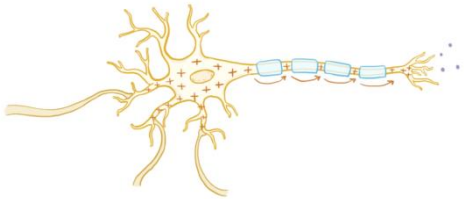
- Les neurones biologiques
  - Cycle de stimulation du neurone



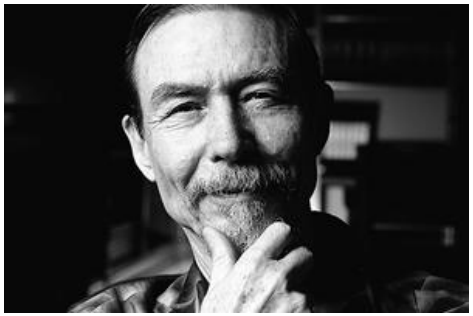
Quel hardware pour l'AI embarquée ? Approches neuromorphiques

- Les neurones biologiques
  - Caractéristiques de fonctionnement
    - Accumulation de stimuli jusqu'à un seuil
    - Envoi d'un stimulation forte
      - Potentiel de stimulation beaucoup plus élevé que le delta de déclenchement
    - Inhibition de la restimulation pendant un certain temps
    - Propagation dans le neurone monodirectionelle rapide





Quel hardware pour l'AI embarquée ?
Chips neuromorphiques



- Calcul neuromorphique (Carver Mead – CalTech - 1980)
  - Initialement : VLSI avec analog circuits
  - Aujourd'hui :
    - Analog, digital, mixed VLSI + code pour émuler des circuits neuronaux
  - Deux axes principaux :
    - Spiking neural networks
      - TruNorth (IBM)
      - Loihi (INTEL)
    - Memristors
      - RRAMs

Quel hardware pour l'AI embarquée ?
Chips neuromorphiques

- Spiking Neural Networks
  - Example of Loihi (Intel)
    - Intel 14nm FinFET : 2,07Mds de transistors
    - Die : 60mm²
      - 2184 neuron / mm²
    - 128 neuromorphic cores
      - 1024 neurones max dans chaque (130.000 au total)
      - 16.000 synapses max par core (8 octets mémoire par synapse) -> 128kB par core (
      - 4096 axones sortant au max d'un core (vers les autres)
        - Attention, un axon inter-core peut irriguer plusieurs neurones
      - 4096 axones entrant au max dans un core (depuis les autres)

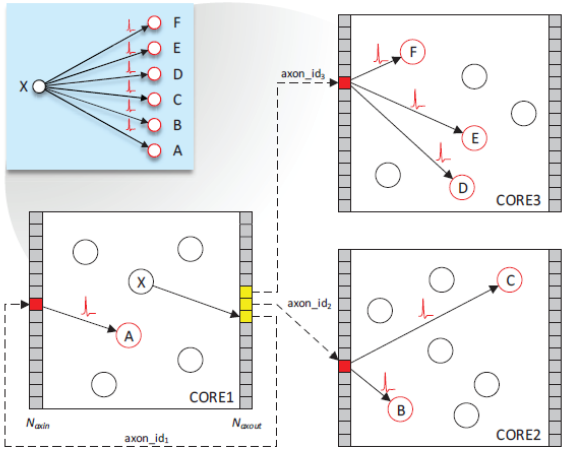
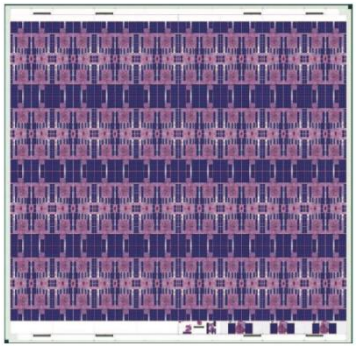
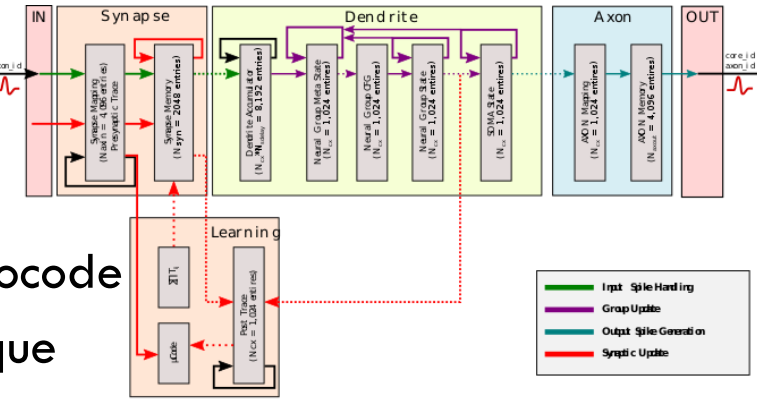


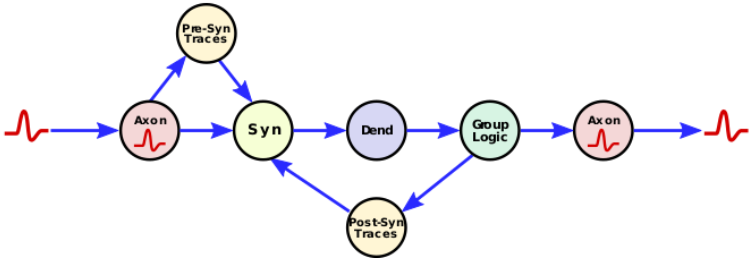
Fig. 3: Neuron-to-neuron mesh routing model



Quel hardware pour l'AI embarquée ? Chips neuromorphiques

- Spiking Neural Networks
  - Example of Loihi (Intel)
    - Fait pour implanter un réseau de neurone de type Spiking Neural Network (SNN)
    - A la capacité d'apprentissage
      - Learning engine
      - Règles implantables dans le microcode
      - Update des synapses fait à chaque learning epoch
    - Mécanisme de synchro globale
    - Implante





Quel hardware pour l'AI embarquée ?
Chips neuromorphiques

- Spiking Neural Networks
  - Example of Loihi (Intel)
    - Un routage avancé des messages
      - Communication entre cores possible
      - Routage asynchrone des messages
        - Basé sur la techno des routeurs
      - Possibilité de coupler les chips
        - Pohoiki Springs : 768 chips !
          - 10 Mds de synapses
          - 10 M neurones
    - Programmation à base de Loihi Python API
    - Permet d'implanter l'AI classique
      - CNN
      - Autoencoder
      - Reinforcement learning

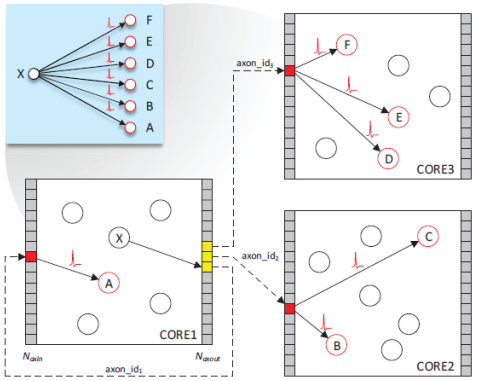
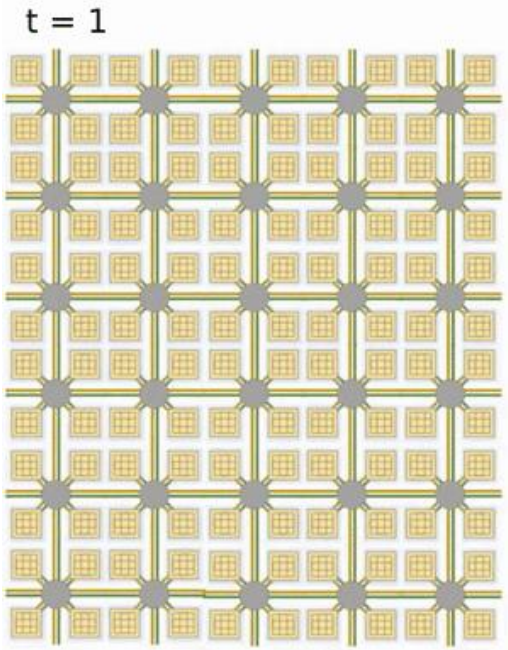
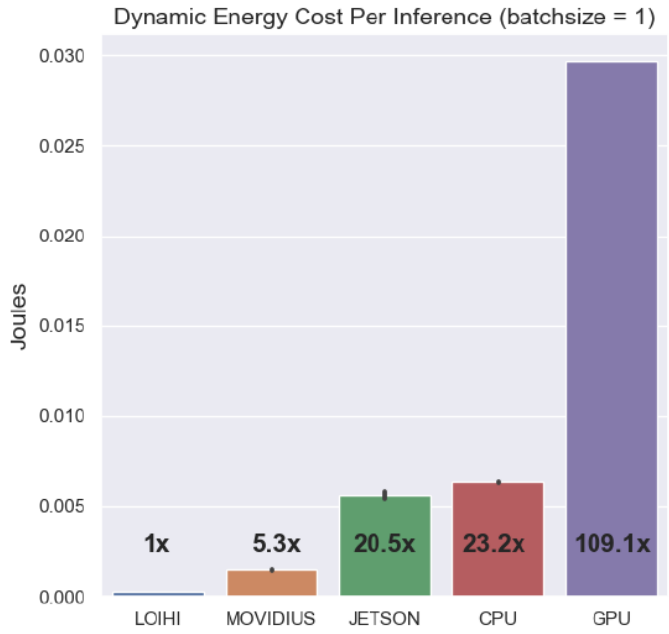


Fig. 3: Neuron-to-neuron mesh routing model



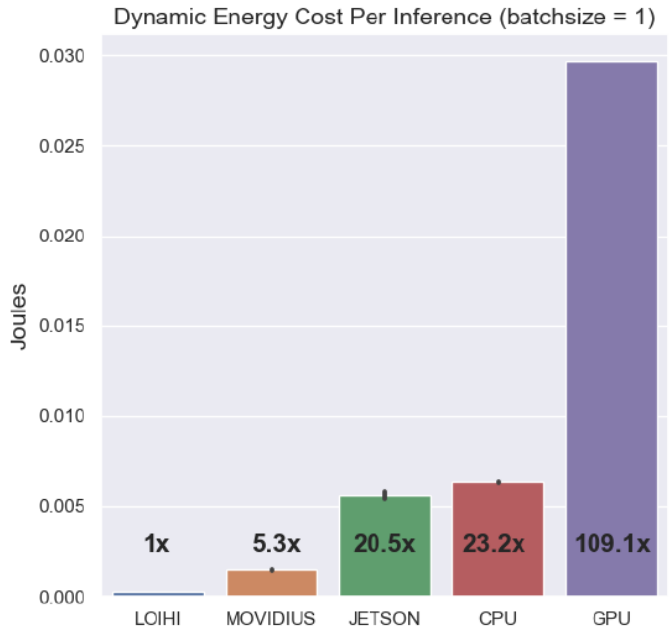
Quel hardware pour l'AI embarquée ? Chips neuromorphiques

- Spiking Neural Networks
  - Performance
    - Tache : reconnaissance vocale
    - 100 fois moins consommateur qu'un GPU !
      - Exemple pas forcément favorable au GPU (sous-utilisé ici)
    - 23 fois moins qu'un CPU
    - Pourquoi ?
      - Les neurones consomment uniquement quand ils sont actifs
      - Le parallelisme réduit le nombre d'étape pour passer de l'entrée à la sortie



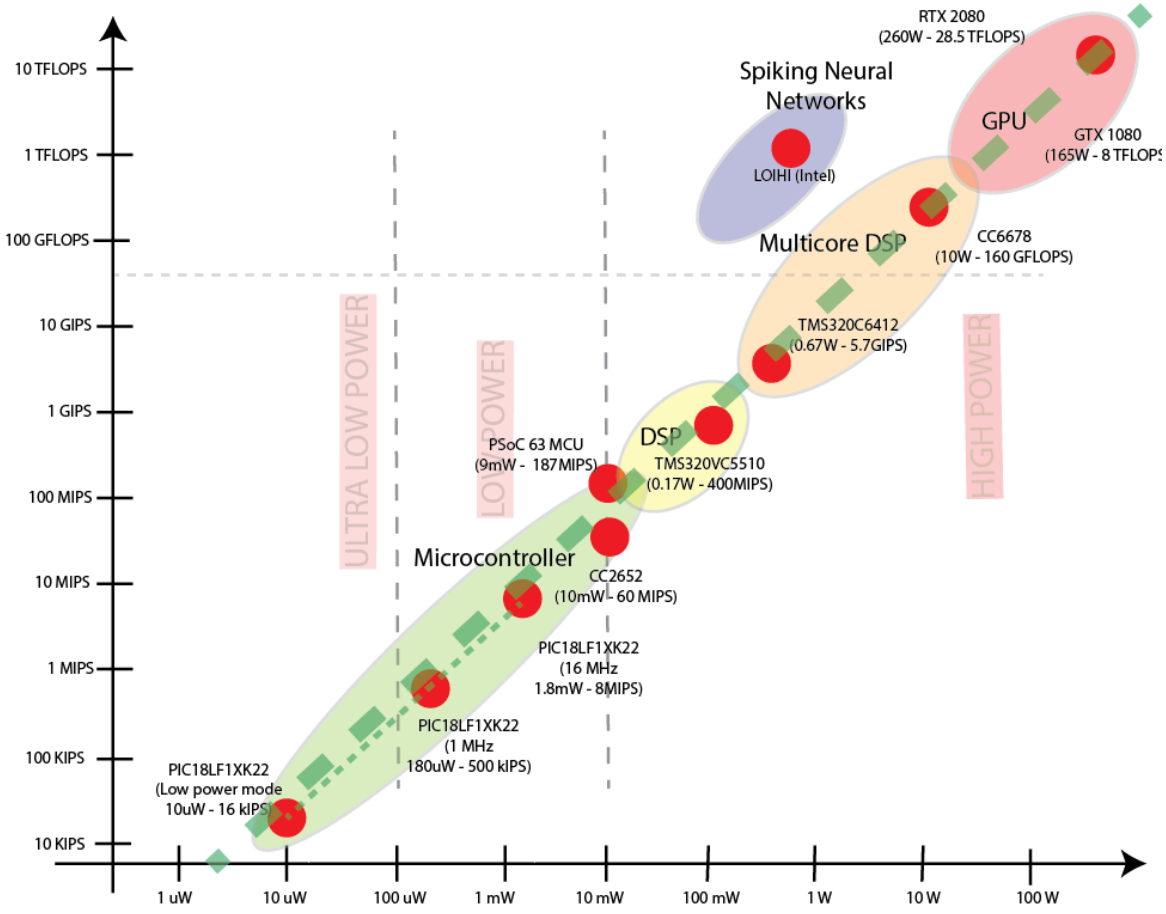
Quel hardware pour l'AI embarquée ? Chips neuromorphiques

- Spiking Neural Networks
  - Performance
    - Tache : reconnaissance vocale
    - 100 fois moins consommateur qu'un GPU !
      - Exemple pas forcément favorable au GPU (sous-utilisé ici)
    - 23 fois moins qu'un CPU
    - Pourquoi ?
      - Les neurones consomment uniquement quand ils sont actifs
      - Le parallelisme réduit le nombre d'étape pour passer de l'entrée à la sortie



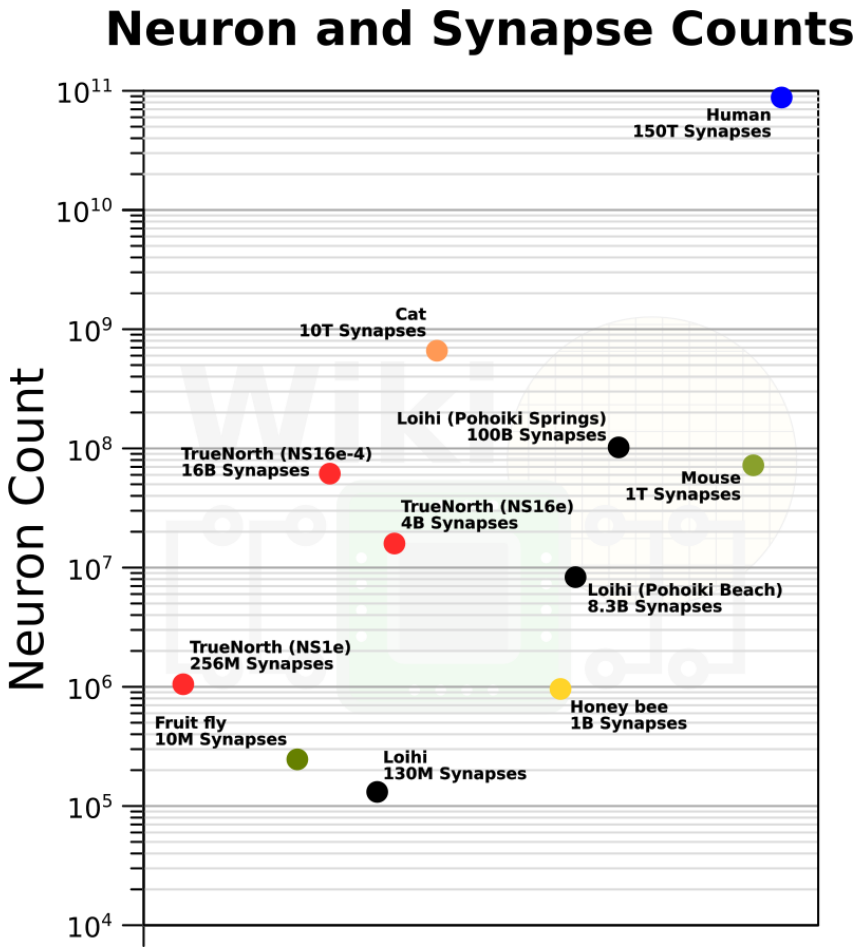
Quel hardware pour l'AI embarquée ?
Chips neuromorphiques

- Spiking Neural Networks
  - Performance



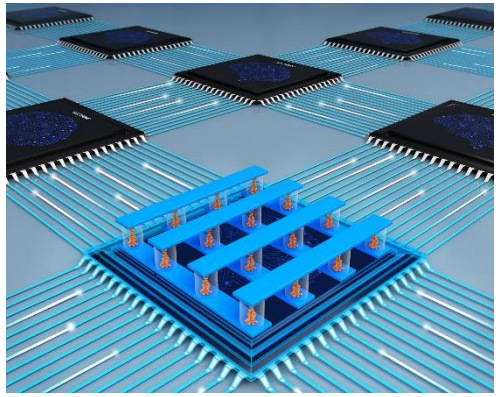
Quel hardware pour l'AI embarquée ?
Chips neuromorphiques

- Spiking Neural Networks
  - Autres solutions techno
    - TrueNorth
      - 16 Mds synapses
      -



Architectures neuromorphiques
Memristors

- Une autre solution de rupture : les memristors
  - Elements passifs capables de mémoriser les impulsions de courant ayant eu lieu dans le passé.
  - Souvent implantés sous forme matricielle (crossbars) comme mémoire RRAM (Resistive RAM)
    - Multi bits : 100 états discernables par cellule
    - Extremely low power consumption : deplacement de quelques ions pour changer d'état : 10fJ / switch
    - Temps de switch : 85 ps (nitride)
  - Peuvent servir de synapse



Source : [2] G. C. Adam, A. Khiat, and T. Prodromakis, "Challenges Hindering Memristive Neuromorphic Hardware from Going Mainstream," Nature Communications, vol. 9, no. 1, p. 5267, 2018.

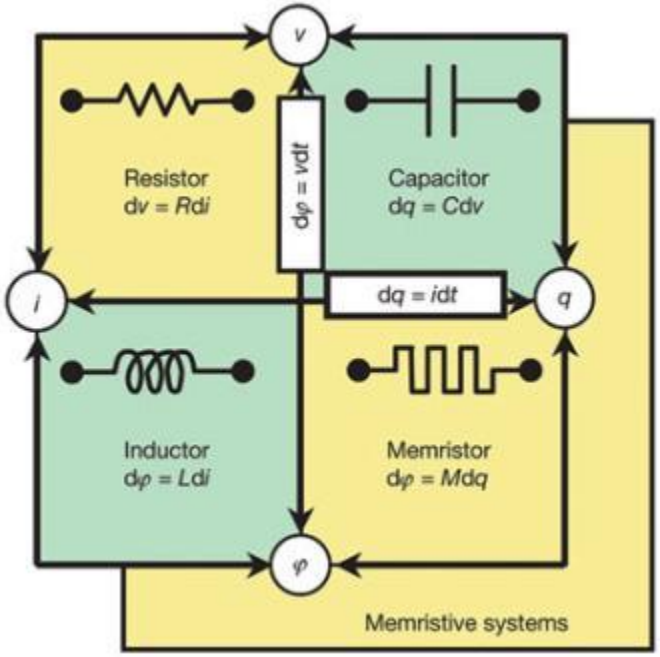
Architectures neuromorphiques
Memristors

- Qu'est-ce qu'un memristor : L.O. Chua 1971
  - Présentation par Leon Chua : https://www.youtube.com/watch?v=ieMdXnjQTXc
  - Présentation HP https://www.youtube.com/watch?v=Z5UTRTOfgo4

Source : [2] G. C. Adam, A. Khiat, and T. Prodromakis, "Challenges Hindering Memristive Neuromorphic Hardware from Going Mainstream," Nature Communications, vol. 9, no. 1, p. 5267, 2018.

Architectures neuromorphiques
Memristors

- Qu'est-ce qu'un memristor ?
  - Théorisé en 1971 – réalisé en 2008

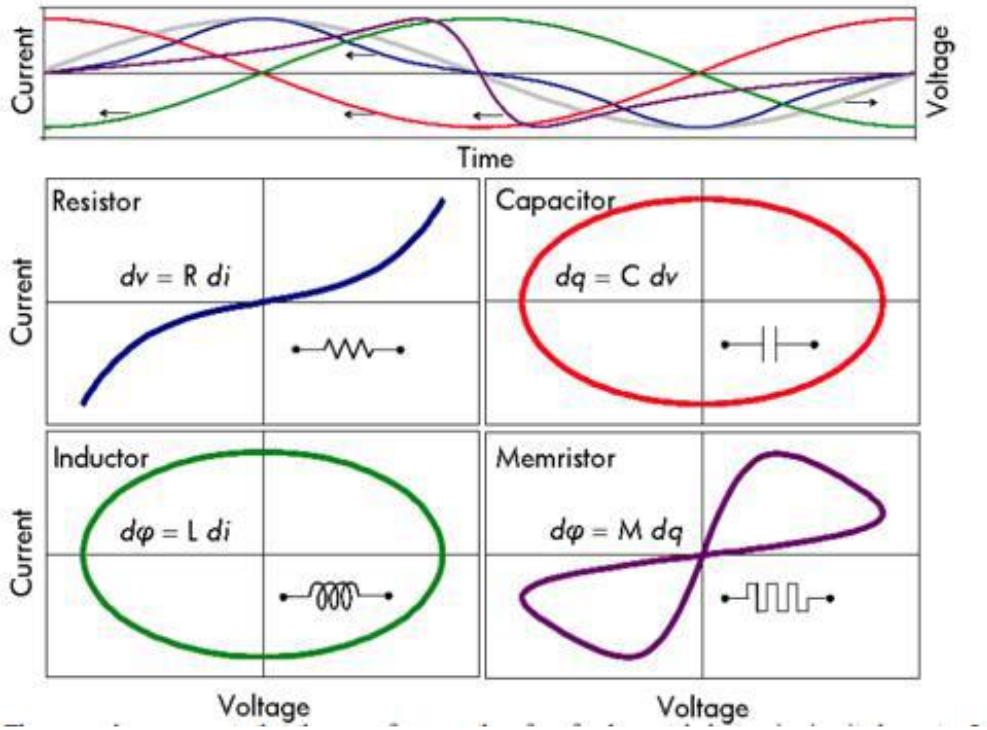


Architectures neuromorphiques
Memristors

- Qu'est-ce qu'un memristor ?
  - $M(q) = \frac{d\varphi}{dq}$
    - $\frac{d\varphi}{dt} = M(q)\frac{dq}{dt}$
    - Donc : $V = M(q)I$
  - Dimension de M : Ohm
  - M dépend de l'intégrale du courant passé
    - En théorie du moins…
    - D'où le nom de **Memristor**

Architectures neuromorphiques
Memristors

- Caractéristique d'un memristor ?
  - Cycle d'hysteresis (V,I) pincé en 0 car $V = M(q)I$

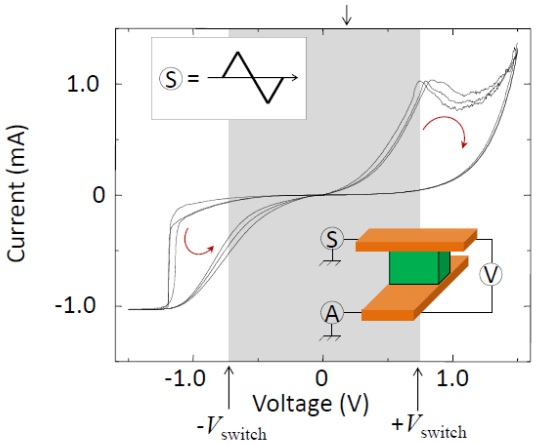


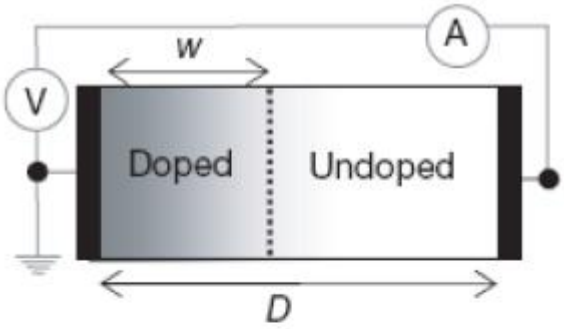
- *"If it's pinched, it's a memristor" (L.Chua)*

Architectures neuromorphiques
Memristors

- Modèles d'un memristor
  - TiO2 based RRAM
    - Jonctions TiO_2/TiO_{2-x}
      - TiO_2 : isolant
      - TiO_{2-x} : conducteur
    - Résistance équivalente :

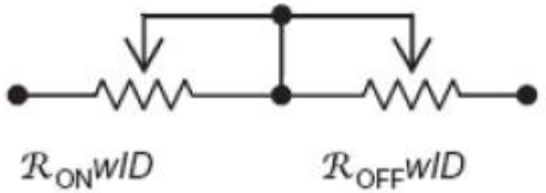
$$R_{mem} = R_{ON}\frac{w}{D} + R_{OFF}\frac{D-w}{D}$$





Undoped:

Doped:



Architectures neuromorphiques
Memristors

- Modèle d'un memristor
  - Comment varie R_{mem} ?
    - Modélisé par la migration des ions présents dans le conducteur sous l'effet du courant :

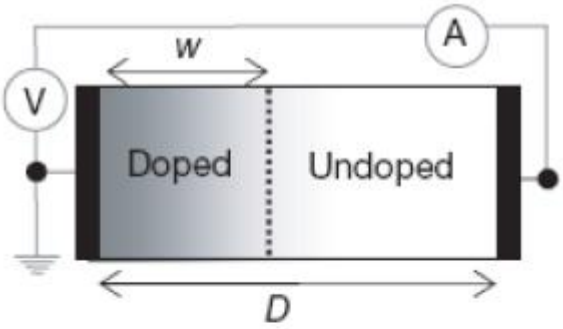
$$\frac{dw}{dt} = \frac{\mu_v R_{ON}}{D} i(t)$$

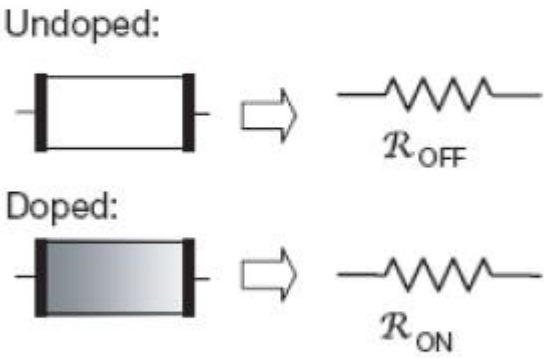
    - w varie avec q, donc R_{mem} aussi :
  - $R_{mem} = RON\frac{w}{D} + ROFF\frac{D-w}{D}$

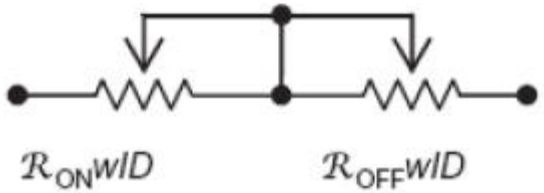
$$= R_{OFF} + (R_{ON} - ROFF)\frac{w}{D}$$

$$= ROFF + (R_{ON} - ROFF)\frac{\mu_v R_{ON}}{D^2} q$$

$$= R_{OFF}(1 - \frac{\mu_v R_{ON}}{D^2} q) \text{ si } R_{ON} \ll R_{OFF}$$







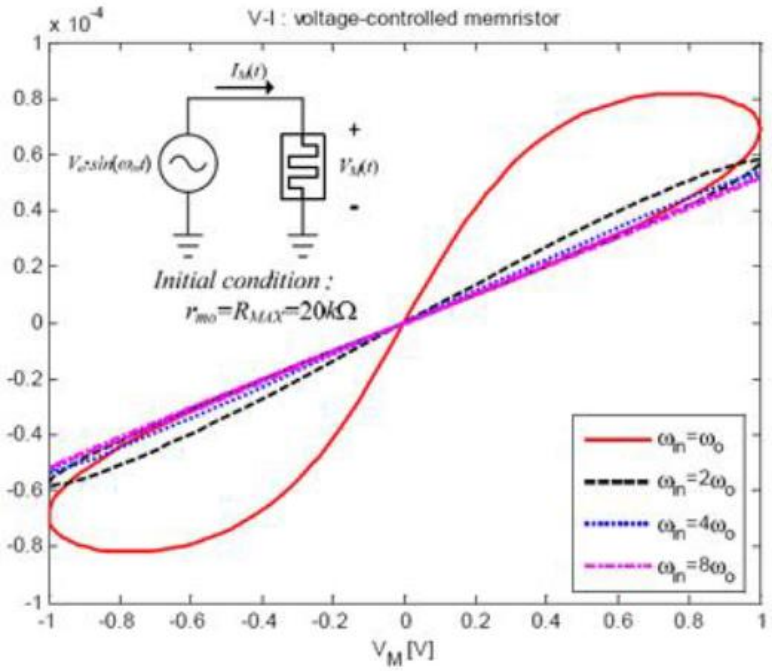
Architectures neuromorphiques
Memristors

- Comment lire la résistance d'un memristor ?
  - Impossible de lire en continu : intégration du courant
  - Il faut appliquer un signal sinusoïdal avec un nombre entier de périodes !
    - Sinon, on change la charge

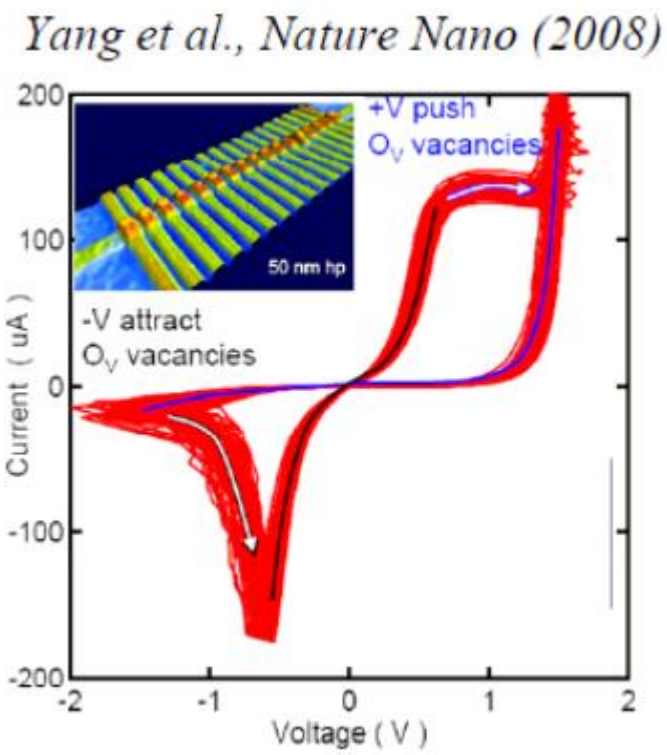
Architectures neuromorphiques
Memristors

- Différences entre un memristor réel et le modèle

Memristor théorique



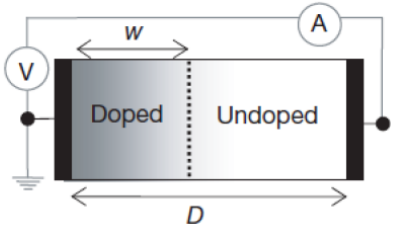
Memristor réel

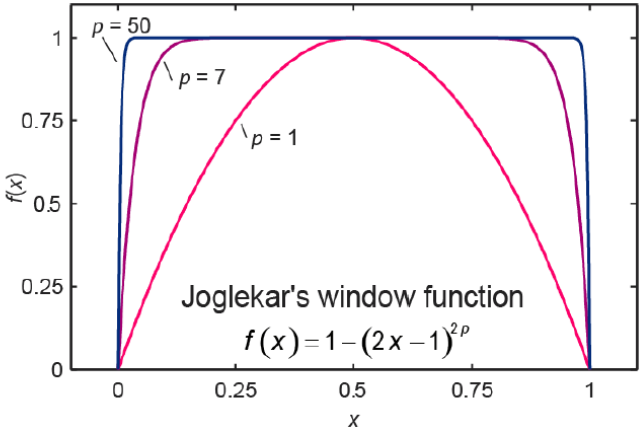


Architectures neuromorphiques
Memristors

- Limites du modèle
  - $R_{OFF}\left(1-\frac{\mu_v R_{ON}}{D^2}q\right)$
  - La vitesse de diffusion des porteurs n'est pas la même dans le deux sens.
  - Nécessité de borner l'évolution de w dans [0;1] : fonction *fenêtre f*

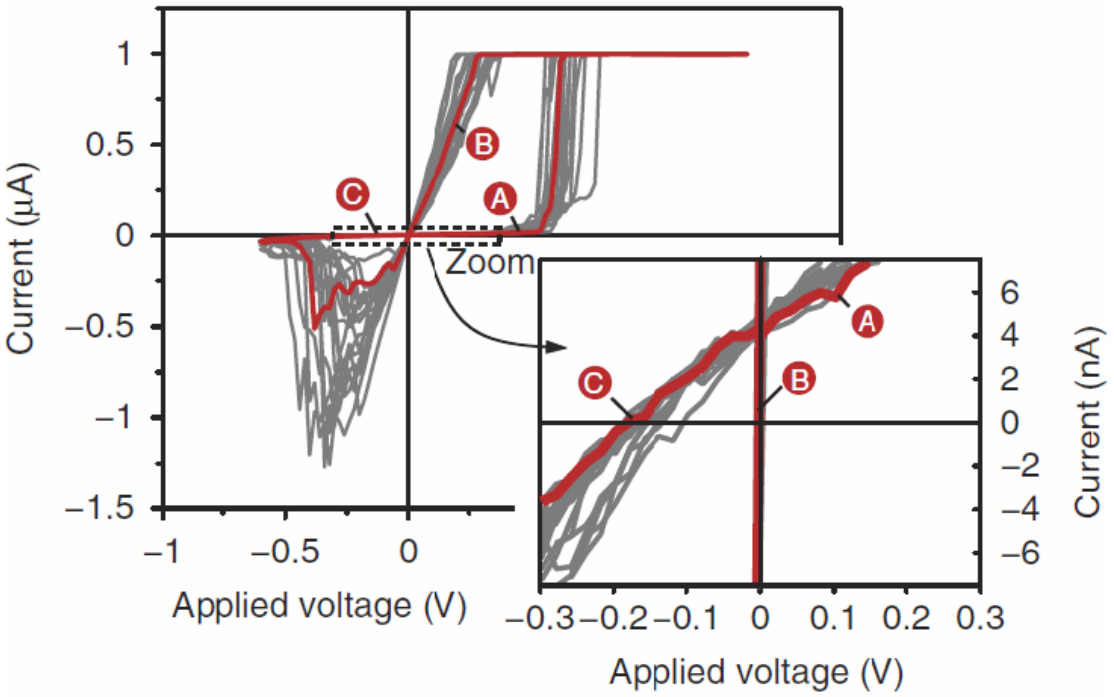
$$\dot{x} = h(x,I) = K_1 \cdot I \cdot f(x,I)$$
$$V = R(x) \cdot I = \left(\left(R_{\mathrm{LRS}} - R_{\mathrm{HRS}}\right) \cdot x + R_{\mathrm{HRS}}\right) \cdot I$$





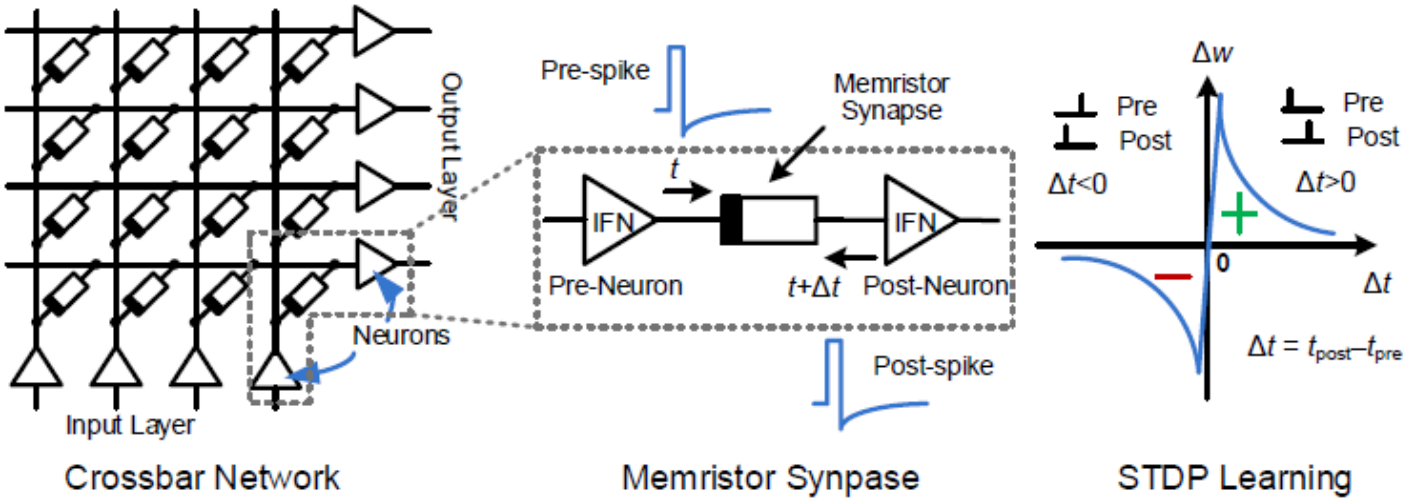
Architectures neuromorphiques
Memristors

- Limites du modèle (suite)
  - Effet de nanobatterie (Nature 2013)
    - Remet en cause le passage par (0;0)



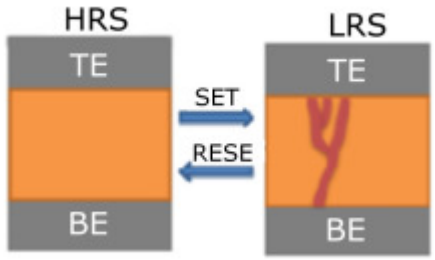
Architectures neuromorphiques
Memristors

- Le memristor : une implantation neuronale ?
  - Exemple : *A CMOS Spiking Neuron for Dense Memristor- Synapse Connectivity for Brain-Inspired Computing (Wu, Saxena, Zhu) 2016*

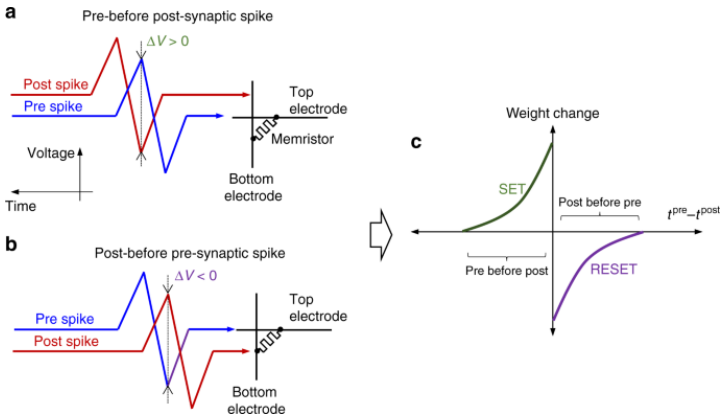


Architectures neuromorphiques
Memristors

- Le memristor : une implantation neuronale ?
  - Idée : moduler les poids des synapses grâce aux memristors
    - HRS (High Resistance State)
    - LRS (Low Resistance State)
    - Possible de représenter jusqu'à 100 états intermédiaires

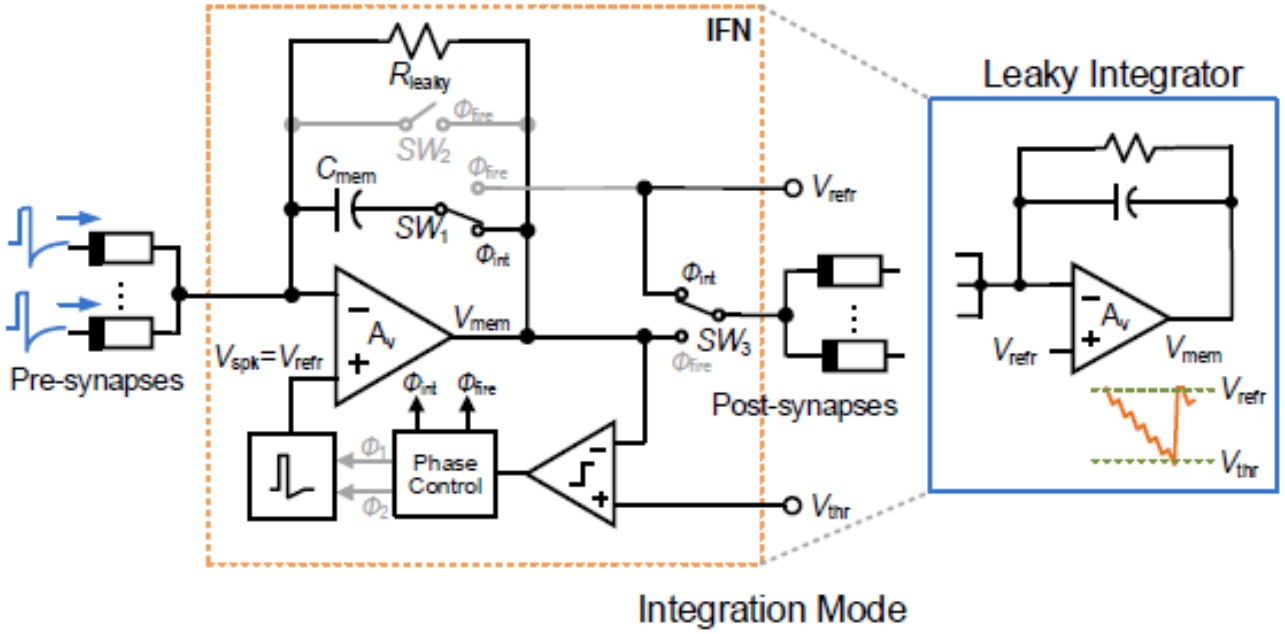


  - La résistance est ajustable par impulsions électriques
    - STDP : Spike-Timing Dependent Plasticity



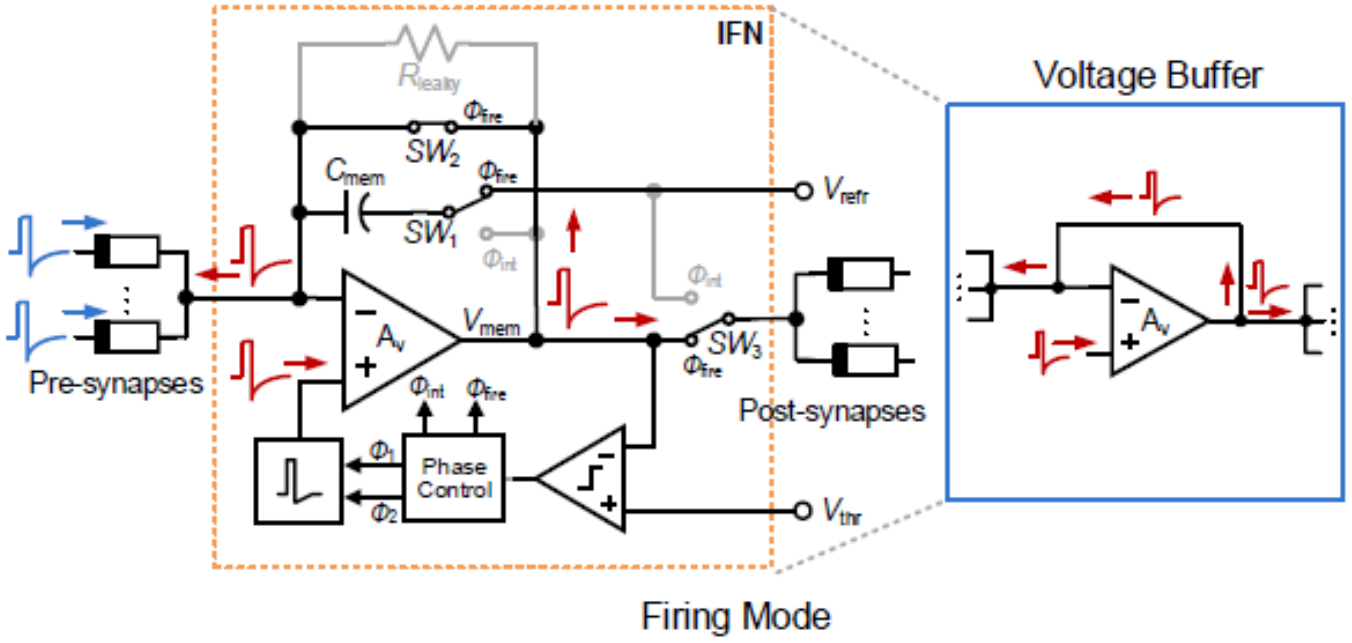
Architectures neuromorphiques
Memristors

- Le memristor : une implantation neuronale ?
  - Mode de fonctionnement : 2 phases
    - Intégration de spikes synaptiques



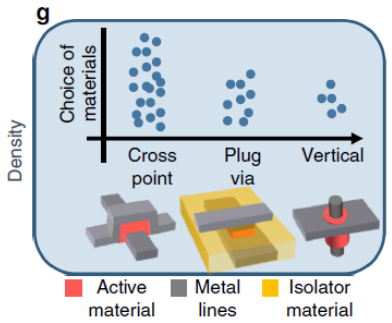
Architectures neuromorphiques
Memristors

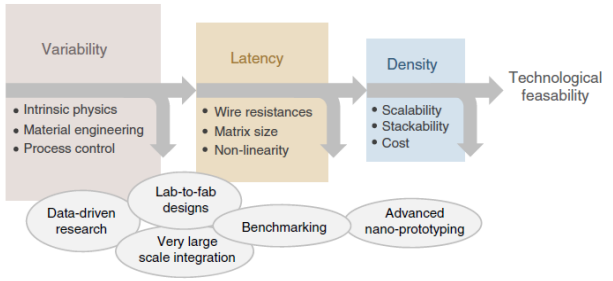
- Le memristor : une implantation neuronale ?
  - Mode de fonctionnement : 2 phases
    - Emission d'un spike (Firing)



Architectures neuromorphiques
Memristors

- Avantages et inconvénients hardware des memristors
  - Variabilité :
    - Complexe à gérer sur de la production industrielle
  - Latence :
    - Les lignes résistives engendrent des temps d'accès importants aux devices.
  - Densité :
    - Des avantages et inconvénients selon la topologie choisie
      - Crosspoint :
        - + : fabrication simple, densité (2nm, 0,7Tb.cm²)
        - - : stepping -> uncontrolled film thinning -> variabilité
      - Plug-via :
        - - : Etching de la via -> variabilité
      - Vertical design :
        - + : moins intégré
        - - : high cost – choix réduits de matériaux

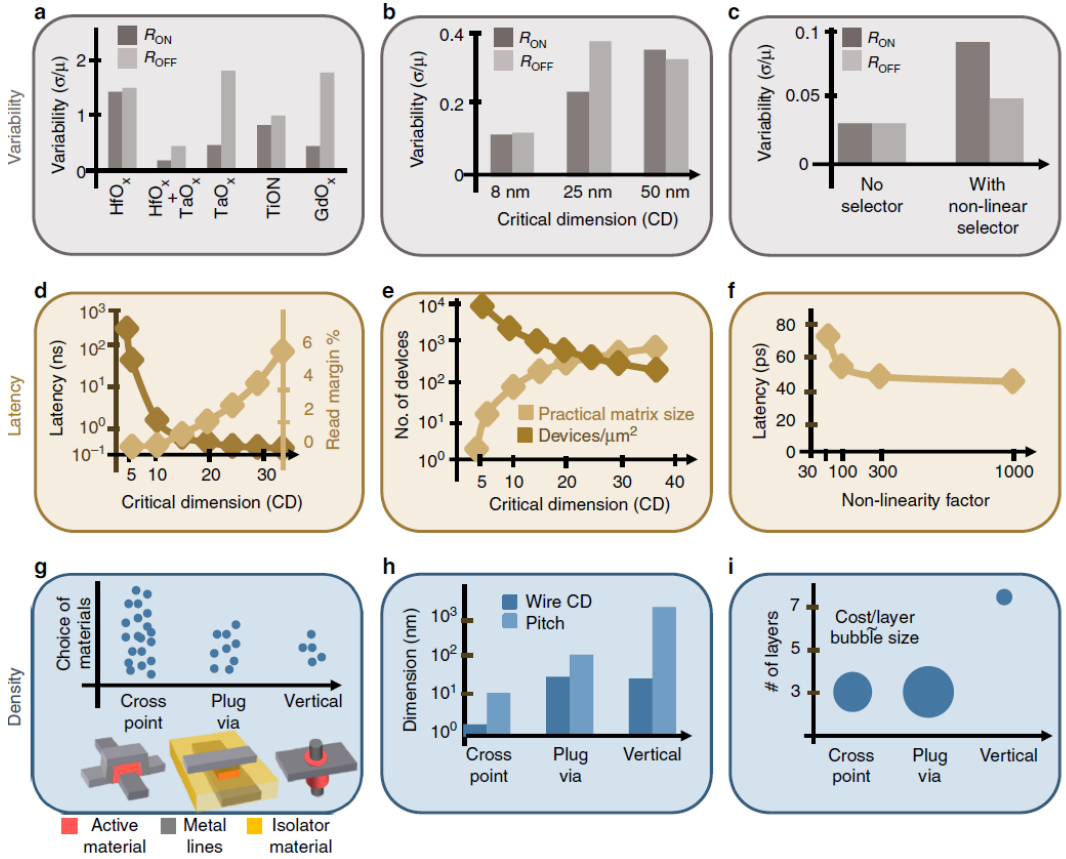




Source : [2] G. C. Adam, A. Khiat, and T. Prodromakis, "Challenges Hindering Memristive Neuromorphic Hardware from Going Mainstream," Nature Communications, vol. 9, no. 1, p. 5267, 2018.

Architectures neuromorphiques
Memristors

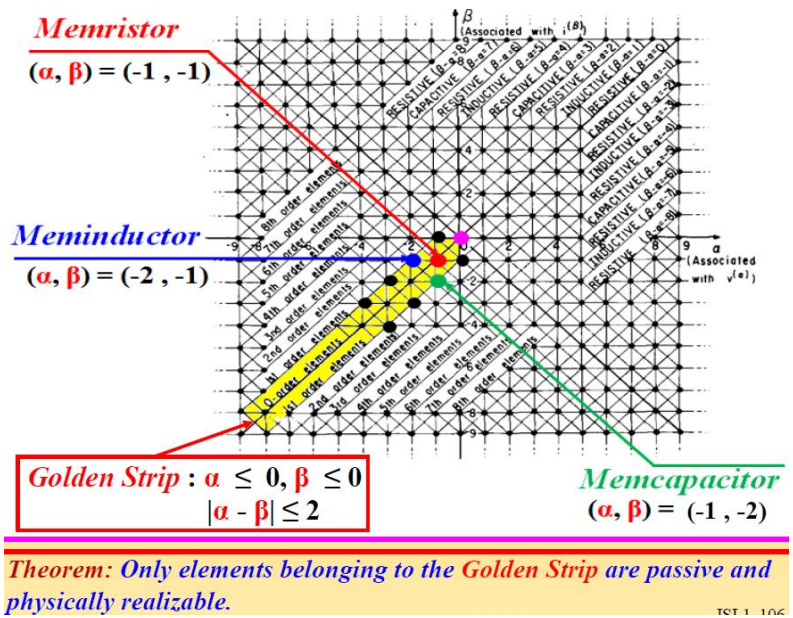
- Avantages et inconvénients hardware



Source : [2] G. C. Adam, A. Khiat, and T. Prodromakis, "Challenges Hindering Memristive Neuromorphic Hardware from Going Mainstream," Nature Communications, vol. 9, no. 1, p. 5267, 2018.

Architectures neuromorphiques
Memristors

- Extension :
  - Les memcapacitors : une capacité variable à mémoire.
  - Les meminductors : une bobine variable à mémoire.
  - Les deux modèles ne sont qu'au début :
    - Golden strip



Quel hardware pour l'AI embarquée

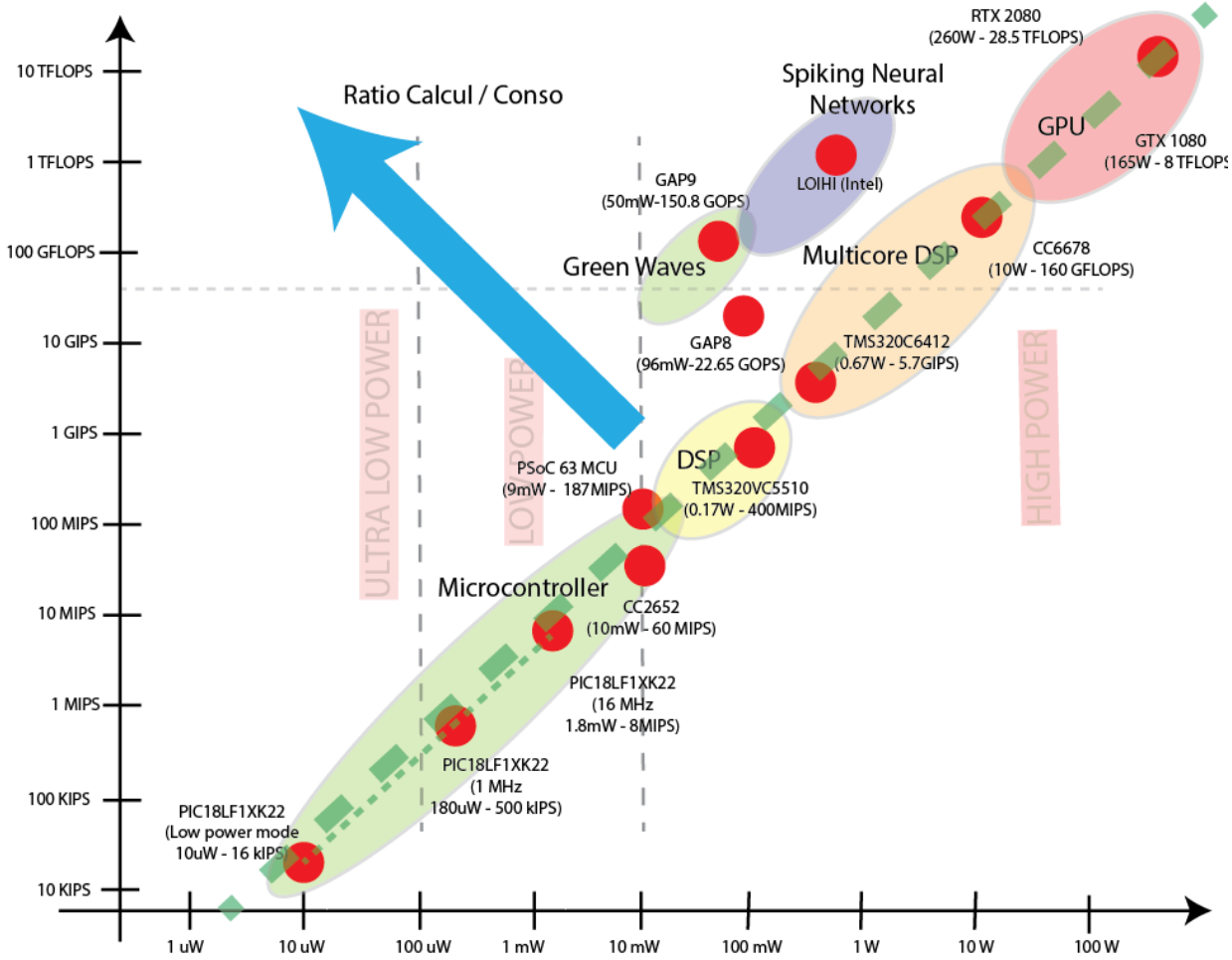
AI ultra-low power

Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Objectifs :
  - Longue autonomie :
    - Consommation inférieure à 100uW
    - Autonomie : 1 an sur une pile bouton CR2032 (720mWh)
  - Permet de réduire les opérations de maintenance
    - Applications industrielles autonomes
    - Application grand public sans recharge
  - Un but à poursuivre : réduire l'impact énergétique de l'IoT sur notre planète

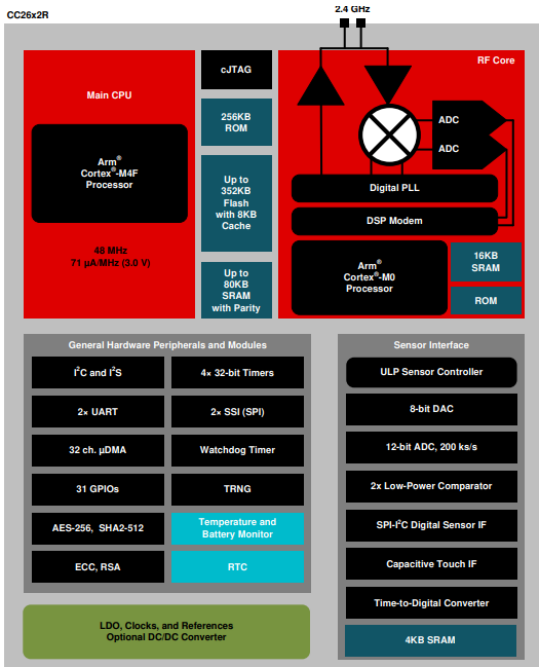
Architectures neuromorphiques
AI ultra-low power

- La plupart des architectures sont > 10mW



Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Première piste : utiliser des uC à basse fréquence
  - Exemples de consommation
    - TI CC2652 :
      - ARM M4 principal : 71uA/MHz (1,25MIPS/MHz)
        - 128uW / MHz sous 1.8V
      - Intègre un sensor controller ultra low-power :
        - Boucle infinie à 2MHz : 30uA
        - ADC 12 bits à 1Hz : 1uA
          - 8 inputs / 200 ksps max
        - 3 timers
    - Cypress PSoC63 :
      - ARM M4 principal : 22uA /MHz (1.25MIPS/MHz) annoncé
        - D'après datasheet sur le M4 : 1,8mW à 8MHz soit 170uW/MHz
  - Pertinent si les signaux sont à basse fréquence



Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Première piste : utiliser des uC à basse fréquence
  - Si les signaux ont une grande dynamique : compliqué
    - Exemple de signaux audio à 20kHz
      - Le cout de conversion ADC est très élevé à cette fréquence
        - 180uW à 20kHz
    - Le cout des traitement est élevé aussi à cette fréquence :
      - Filtre passe-bas ordre 1 (Euler) :
        - 40k MAC = 1 MIPS = 128uW (1 MAC = 25 IPS)
      - FFT (Window size : 1024) 20 fois par seconde :
        - Environ 2n ln(n) = 500kIPS = 970uW

Quel hardware pour l'AI embarquée ?
AI ultra-low power

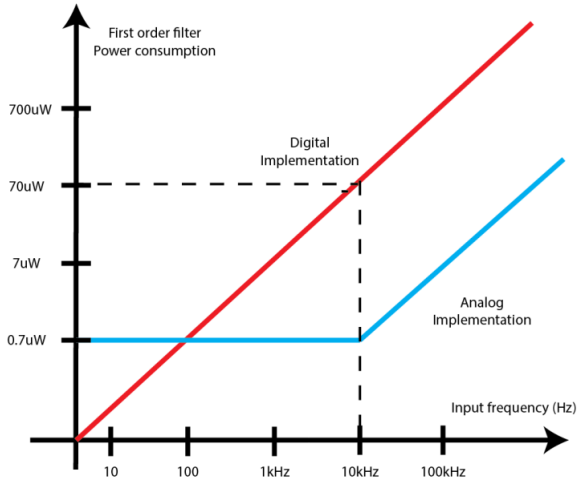
- La piste des uC à basse fréquence ne permet pas de traiter des signaux à fréquence élevée
  - filtrage simple à 20kHz > budget max en ULP
- On doit envisager autre chose :
  - Il faudrait récupérer des features prétraitée à très basse fréquence
  - Comment faire à votre avis ?

Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Calcul analogique :
  - Prétraitements analogiques à haute fréquence pour extraire des features du signal
    - Filtre LP : 130 uW
    - Conversion ADC : 180uW
    - Op Amp (MAX4464 – GBP 40kHz) : 0.75uA – 1.8V
      - Conso : 1.35uW
      - 100 opAmp = 1 filtre ordre 1 digital
      - 140 opAmp = Conversion ADC à 20kHz
  - La consommation directement liée à la fréquence
    - Consommation linéaire avec le GBP
    - Consommation linéaire avec la fréq de calcul uC
    - Le rapport 100 opAmp = 1 filtre ordre 1 uC
      - Reste vrai si on augmente la fréquence
      - A condition d'ajuster le GBP à la freq du signal à traiter.

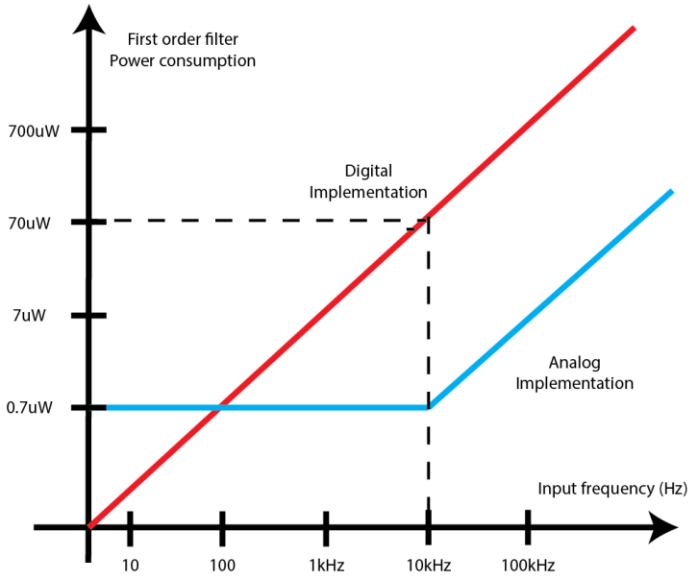
Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Calcul l'analogique :
  - A très basse fréquence, le numérique redevient intéressant.
    - GBP mini d'un AOP : 10kHz
      - 100 opAmp GBP 10kHz = 1 filtre ordre 1 Euler à 10kHz
      - 1 opAmp GBP 10kHz = 1 filtre ordre 1 Euler à 100Hz
      - En dessous de 100Hz, le numérique coute moins cher.



Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Domaine de pertinence des preprocessing analogiques :
  - Au-delà de 10kHz :
    - L'analogique permet de gagner un facteur 100 en efficacité énergétique.
  - En dessous de 100Hz :
    - Le numérique est plus performant
      - cout fixe trop important pour l'analogique
  - Entre les deux, l'analogique est meilleur mais moins nettement



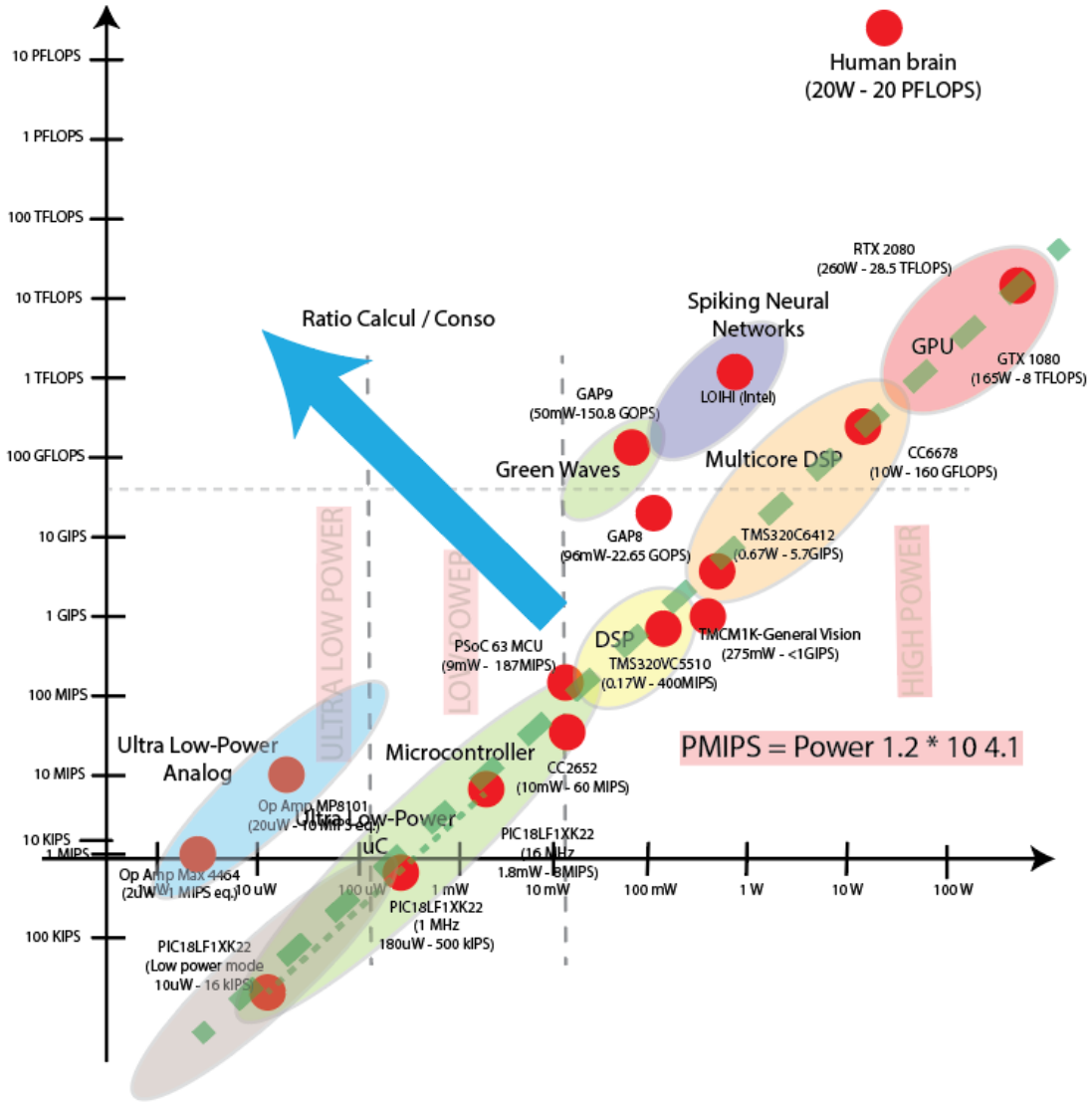
Quel hardware pour l'AI embarquée ?
AI ultra-low power

- Une stratégie pertinente :
  - Analog preprocessings
    - Fréquence élevée : gain important en analog
  - Moments extraction
    - Réduction de la fréquence utile du signal par extraction de ses caractéristiques variant lentement.
  - Digital sampling et post-processing
    - Fréquence faible : efficace en digital
    - Traitements plus complexes : FSM, NN, …

Quel hardware pour l'AI embarquée ?
AI ultra-low power

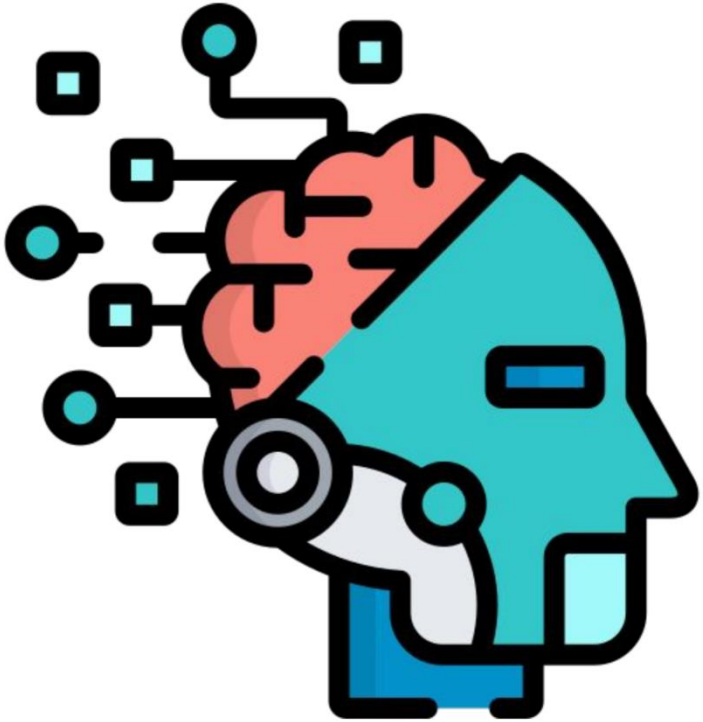
- Le retour de l'analogique
  - Possible de faire des traitements en analogiques pour moins cher qu'en digital
    - Si les signaux ont une bande passante entre 1kHz et 20kHz, on est gagnant en analogique.
      - Conso constante en analogique = conso en numérique à 20kHz
      - Au-delà, pas de gain, il faut des AOP à plus grand GBP
      - En-dessous, la conso constante de l'analogique est trop pénalisante.
      - Cout de la conversion analogique-numérique
        - Il faut essayer de réduire la fréq de conversion en transformant le signal (extraction de feature en analogique)
    - Exemple d'évaluation :
      - Op Amp GBP=20kHz : conso XXX uW

Quel hardware pour l'AI embarquée ?
Les différentes architectures pour l'AI



AI embarquée

The end !



That is life

C'est la vie

Это жизнь

Bubomi obo

それが人生です