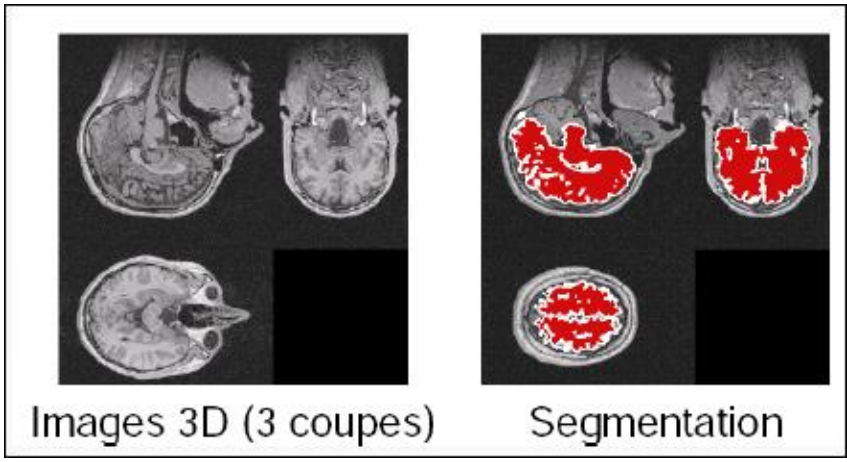
Segmentation, détection et suivi d’objets

Chapitre 3

Qu’est que la segmentation ?

- Décomposition d’une image en régions qui ont un sens
  - Etiquetage des pixels de l’image
  - Même couleur = même région
  - Critère d’homogénéité

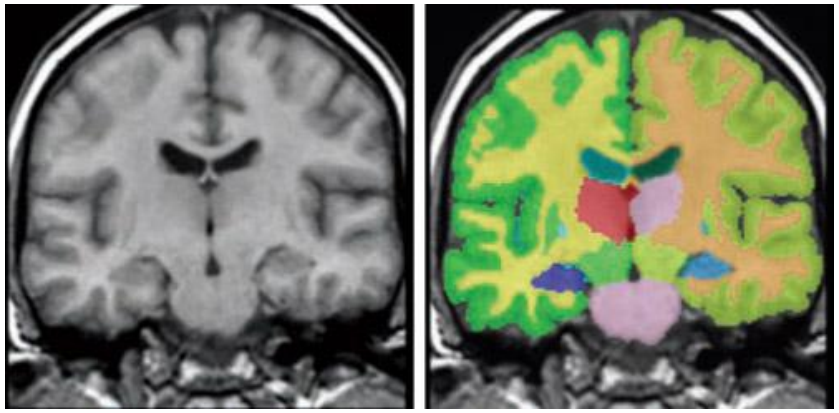


- Facile pour un être humain : connaissance préalables, image vue dans sa totalité,
  - Approches régions : grouper les pixels semblables et former des régions homogènes
  - Approches frontières : rechercher pixels dissemblables pour former des contours entre zones homogènes
  - Eventuellement des approches hybrides (frontières et régions)

Dans l'ancien temps…

- **Approche « Frontières » : détection des contours….**
  → Cf chapitre précédent

- **Approche « Région »**
  → Question : comment détecter des régions ?
  → Question : Comment découper une images en régions ?



Approche région : Croissance de région

- Idée: partir d'un point amorce (*seed*) et étendre en ajoutant les points de la frontières qui satisfont un critère d'homogénéité

amorce *croissance* *région finale*

- Le point amorce peut être choisi soit par un humain, soit de manière automatique en évitant les zones de fort contraste (gradient important) => *méthode par amorce*
- Le critère d'homogénéité est local (comparaison de la valeur du pixel candidat et du pixel de la frontière)

Segmentation : Croissance de régions

Algorithme ?

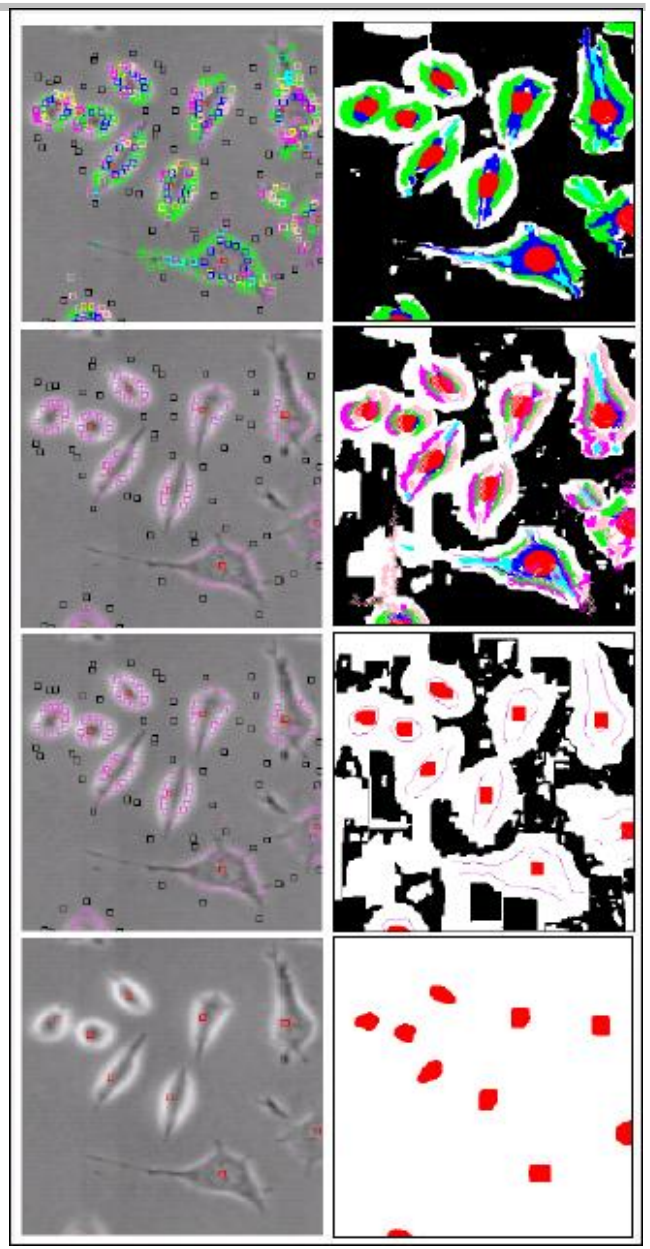
Pour chaque germe

  Pour chaque voisin

    Si (critère d'homogénéité vérifié
      ET pixel non labelisé)

      Propagation du label région

On applique la même procédure récursivement



Croissance de régions

Avantages

- Méthode rapide
- Conceptuellement très simple

inconvénients

- Problème du gradient 
  - Tenir compte de l'homogénéité globale donne un algorithme sensible à l'ordre de parcours des points (méthode par amorce)
  - Algorithme très sensible au bruit, peu stable

- Initialisation manuelle ou aléatoire
- Résultat dépendant des germes initiaux
- Méthode locale: aucune vision globale du problème. En pratique, il y a presque toujours un chemin continu de points connexes de couleur proche qui relie deux points d'une image…

Split and Merge

Principe :

- Découper l’image en arbre (SPLIT)
- Fusionner les parties homogènes (MERGE)

Algorithme ?

Split and Merge

Principe :

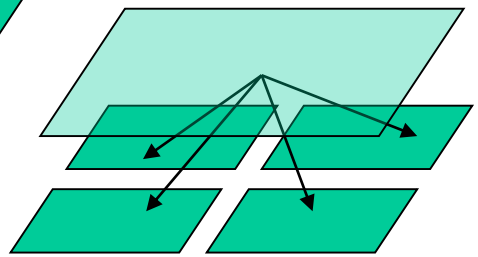
- Découper l'image en arbre (SPLIT)
- Fusionner les parties homogènes (MERGE)

SPLIT

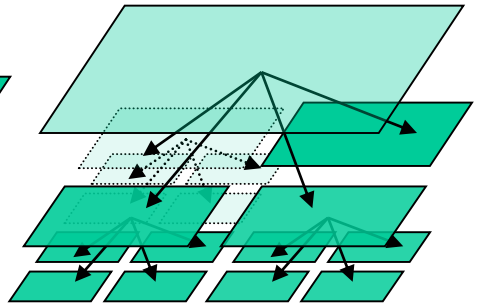
0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

Image initiale

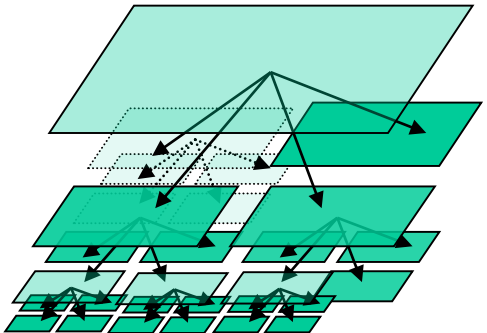
Split 1



Split 2



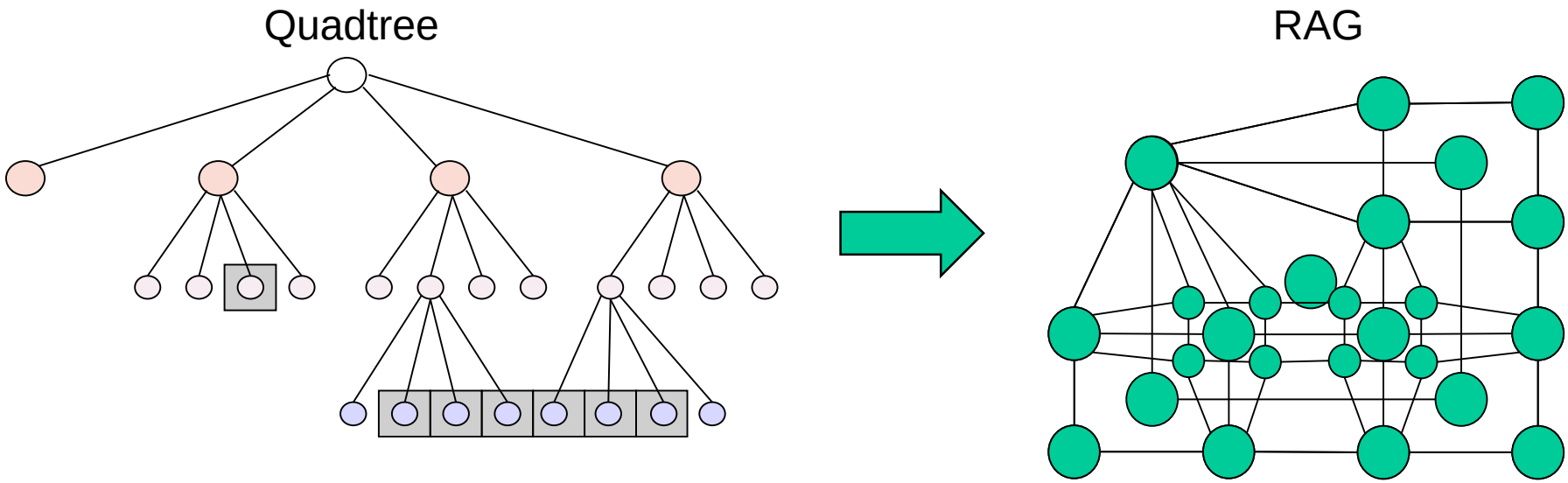
Split 3



Split and Merge

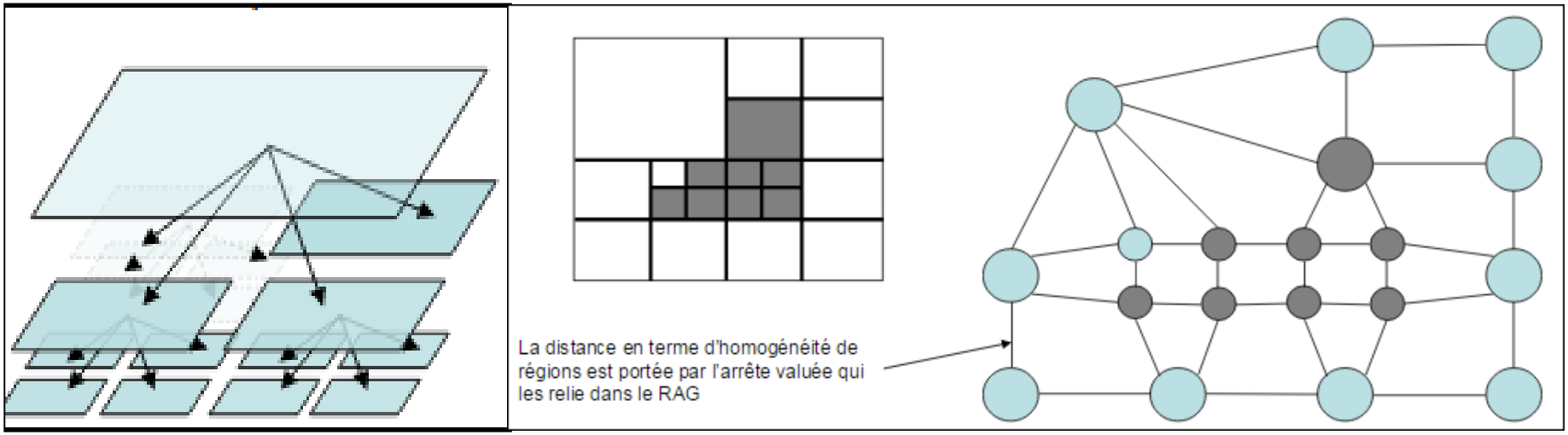
MERGE

- Construire le RAG : graphe qui prend en compte l'adjacence
- Valeur des arrêtes : mesure de la différence d'homogénéité
- Fusion itérative des nœuds



Split and Merge

- C’est l’approche duale de la croissance de région
- Temps de calcul assez longs
- Ces algorithmes s’adapte facilement aux images 3D

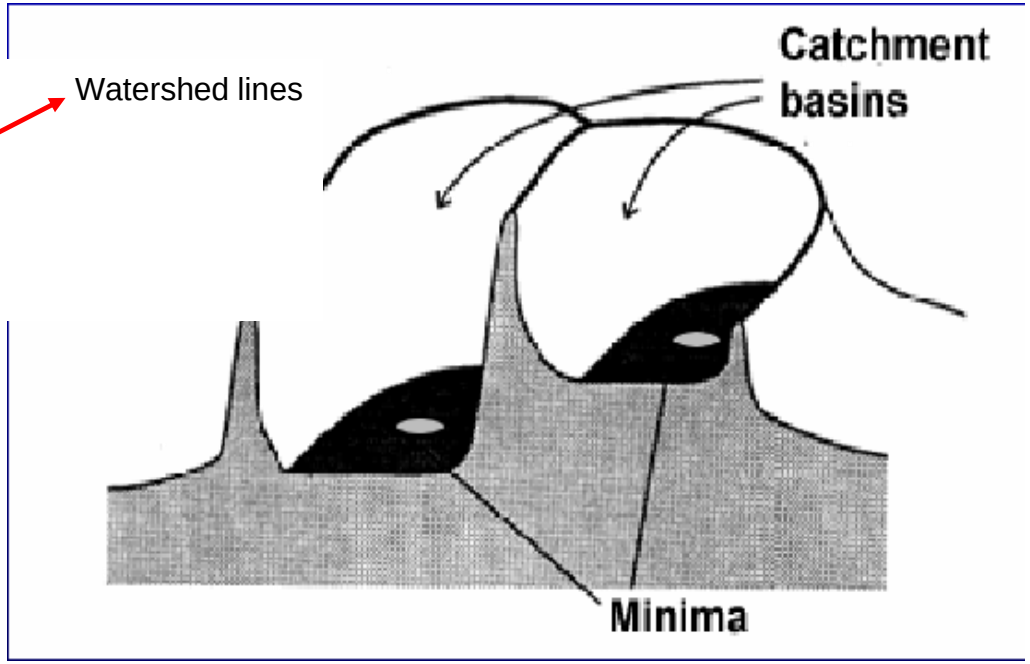


Ligne de partage des eaux

- Considérer une image comme une **carte topographique**.
- Principe : **inonder** cette carte topographique à partir de **sources** qui seront placées dans l'image.
- L'image finale sera composée de **bassins** (où l'eau s'est accumulée) et de **lignes** délimitant ces bassins.

1. Trouver les ***minima*** de l'image
   - n_bassin=n_minima
2. ***Tant que*** n_bassin>1 ***faire***
   - inonder l'image à partir des minima
   - ***Si*** deux bassins communiquent ***alors*** un maximum local est trouvé ***fin si***

   Fin tant que



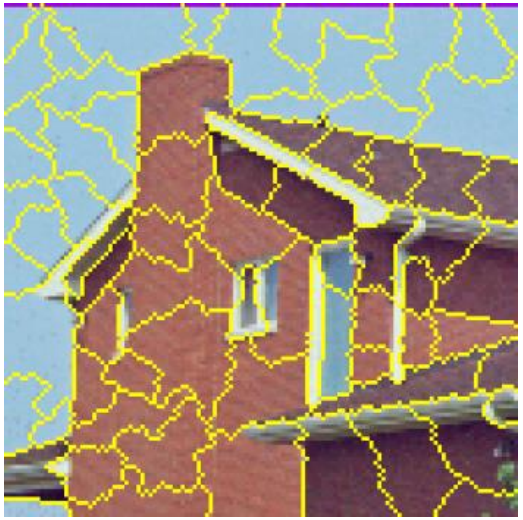
Ligne de partage des eaux

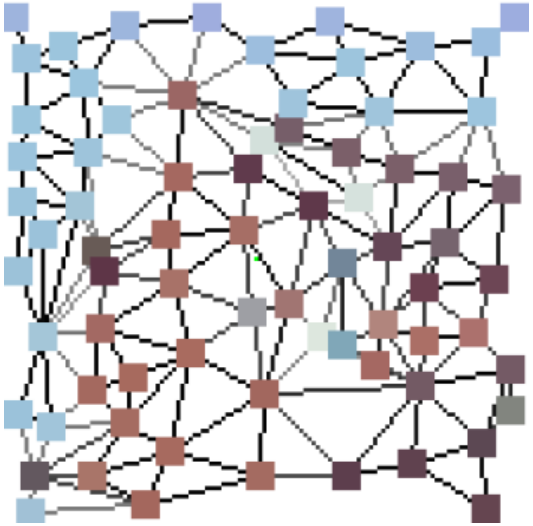
- Avantages:
  - Grande précision sur les frontières obtenues
  - Distinction parfaite de 2 régions collées

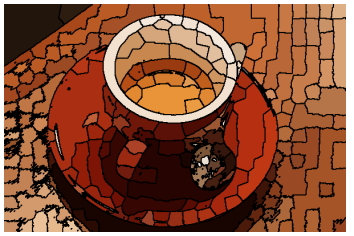
- Inconvénients:
  - Consommation de mémoire
  - Sensibilité au bruit
  - Sur-segmentation
  - Demande un post-traitement

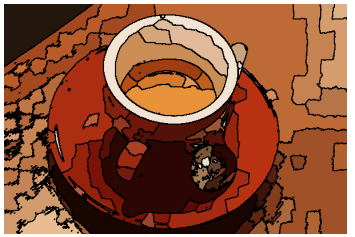
Region Adjacency Graph

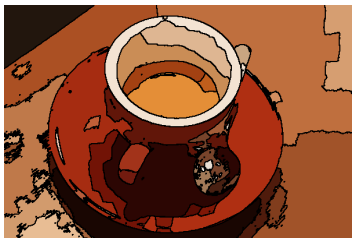
- Représentation d'images avec des graphes
- Manipulation de graphes
  - Nœuds ? Arcs ?
  - Fusion / Division
  - CF Option IA, GNN…







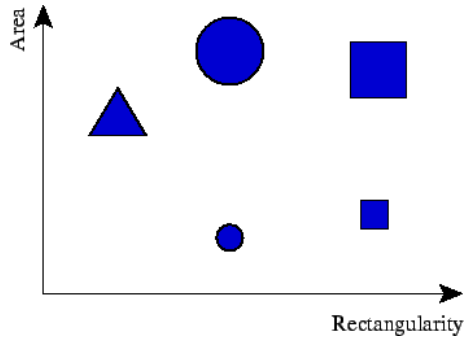


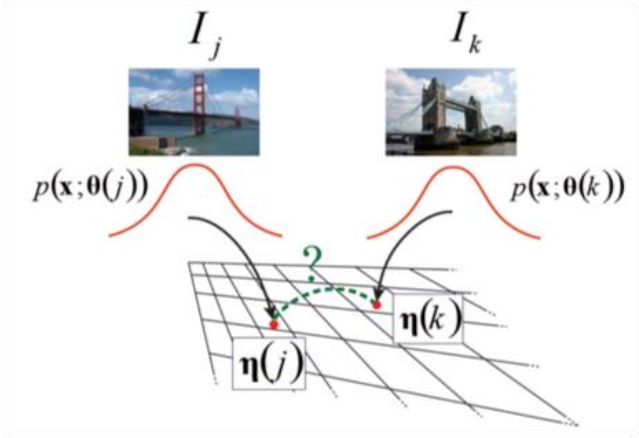


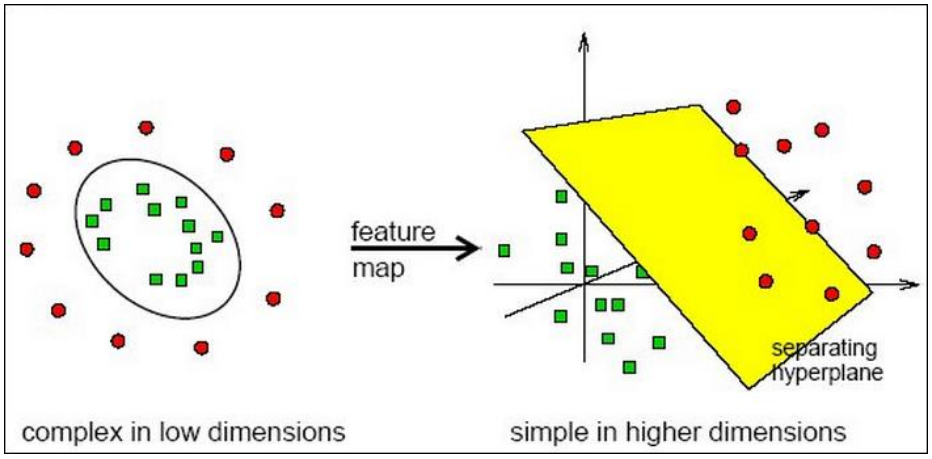
Approche basée classification

Principes

- p caractéristiques ⇨ Vecteur ⇨ Point ∈ R^p
- Analyse statistique des vecteurs (modalités/voxels)







Deux cas de figure :

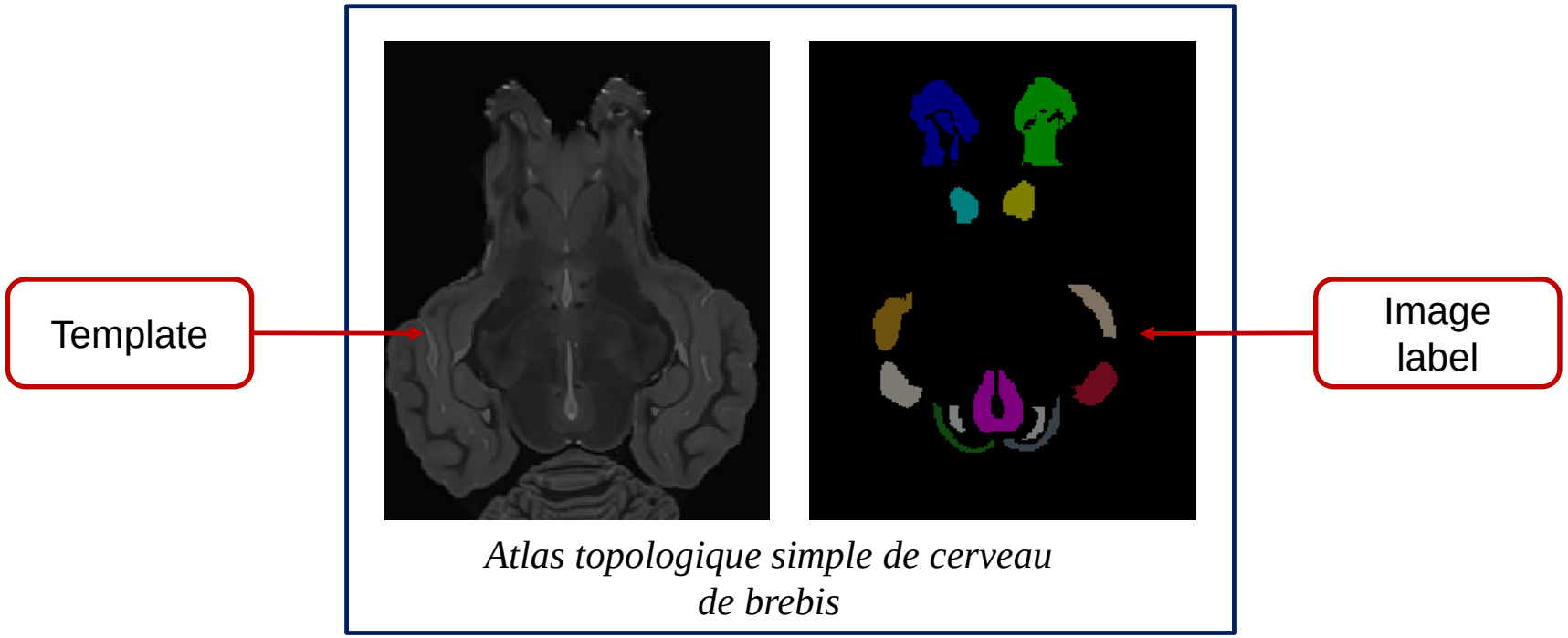
- Classification supervisée : on dispose d'un ensemble d'entrainement, à partir duquel on construit le classificateur
- Classification non supervisée : on dispose de vecteurs caractéristiques dont on ne connaît rien ⇨ Regroupement en vecteurs similaires pour former des agrégats (clusters)

Approche probabiliste (Atlas)

Atlas (volumique)

Information *a priori* spatiale sur la position et l'intensité des structures anatomiques

- **Différents types d'atlas :**
  - Simple



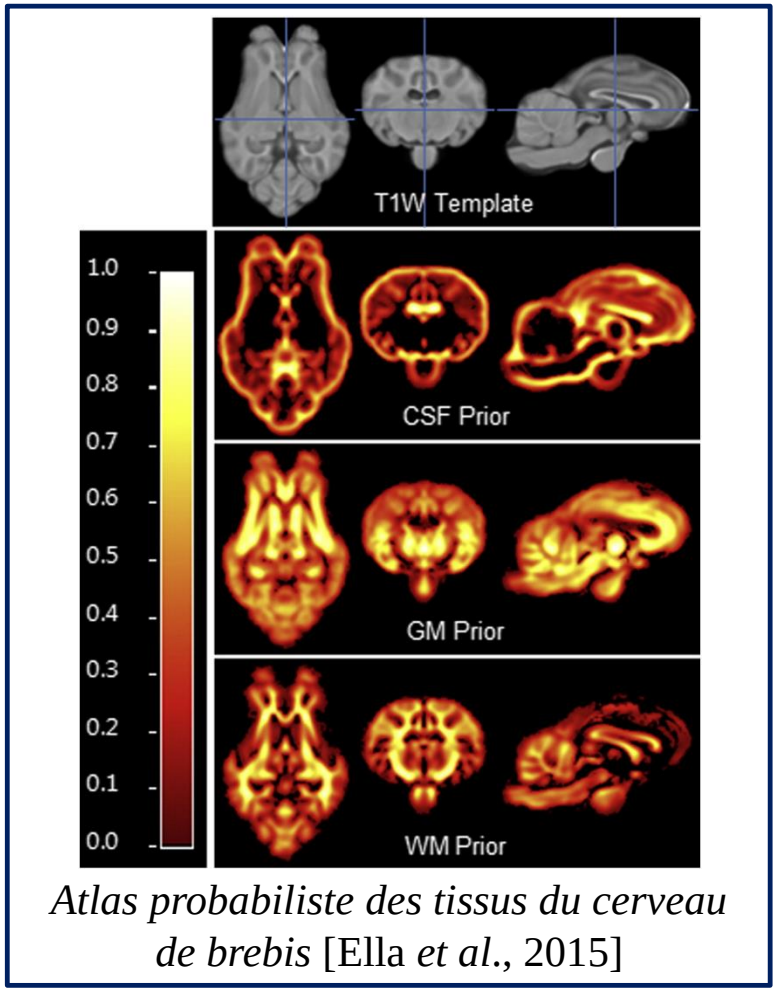
Atlas topologique simple de cerveau de brebis

Approche probabiliste (Atlas)

Atlas (volumique)

Information *a priori* spatiale sur la position et l'intensité des structures anatomiques

- **Différents types d'atlas :**
  - Simple
  - Probabiliste



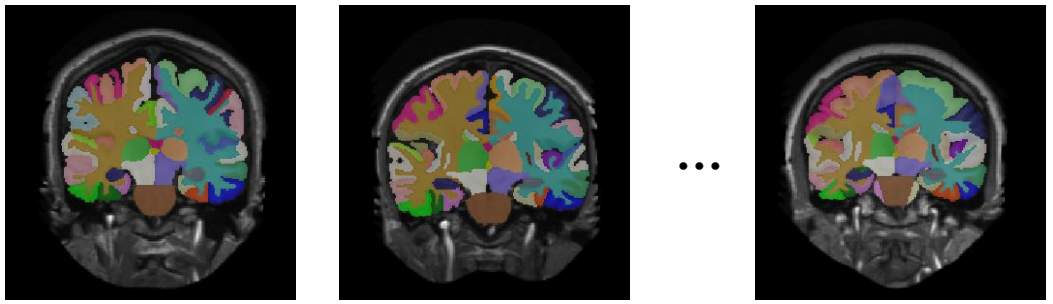
Atlas probabiliste des tissus du cerveau de brebis [Ella *et al.*, 2015]

Approche probabiliste (Atlas)

Atlas (volumique)

Information *a priori* spatiale sur la position et l'intensité des structures anatomiques

- **Différents types d'atlas :**
  - Simple
  - Probabiliste
  - Multi-Atlas



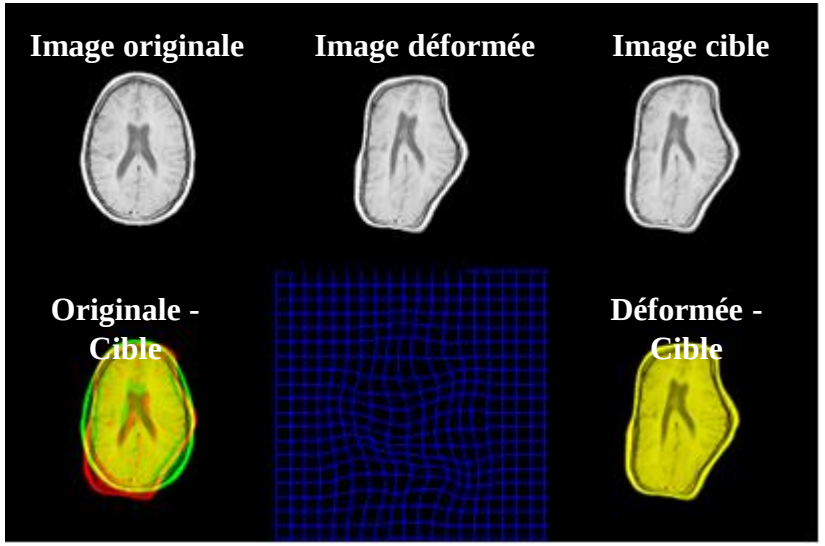
Multi-Atlas du cerveau humain de la base d'images utilisée lors du concours MICCAI'12 [2]

Approche probabiliste (Atlas)

Mode d'utilisation des atlas

Méthode de recalage nécessaire lors de l'utilisation des atlas

- **Transformation :**
  - Rigide
  - Affine
  - Non linéaire (TPS, B-spline, etc…)
- **Métrique :**
  - Somme des différences au carré
  - Information mutuelle
  - Cross-correlation
- **Propagation de label**
- **Fusion d'informations et décision :**
  - Méthode de vote, pondéré globalement [Artaechevarria *et al*., 2008] , pondéré localement [Isgum *et al*., 2009] Joint label fusion [Wang *et al*., 2013] , Generative model [Sabuncu *et al*., 2010] , etc…
  - Mélange de gaussiennes, Chaine de Markov [Bricq *et al*., 2006] , Champ de Markov [Scherrer *et al*., 2009] , etc…



Recalage non linéaire de deux images de cerveau [3]

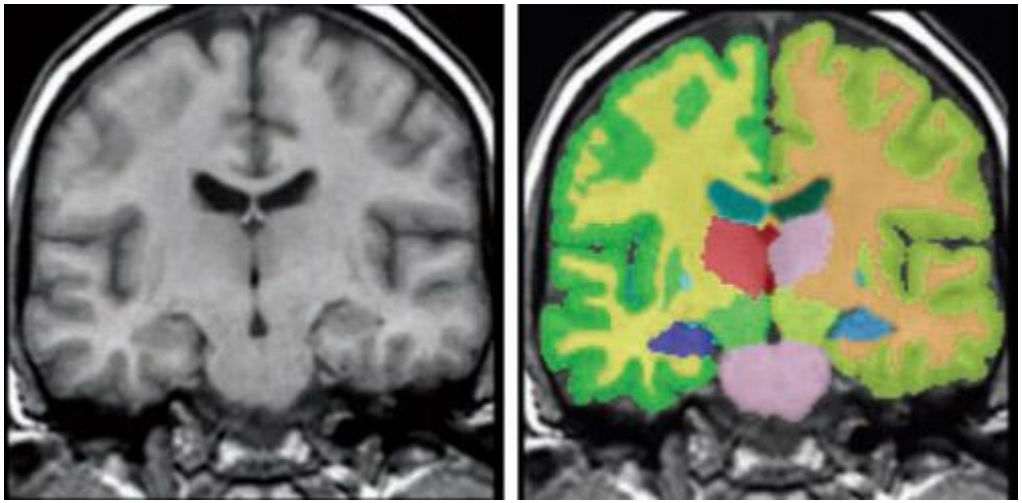
Approches probabilistes (Atlas)

Bilan

- **Idée :** les pixels sont des variables aléatoires représentant la probabilité d'appartenir à chaque classe
- Puis, estimation du maximum vraisemblance (MAP, EM,…)
- Chaine de Markov : 2D représenté en 1D (parcours d'hilbert, Peano)
- Champs de Markov : Seul les voisins comptent

Avantages/Inconvénients

- Exploitation de connaissance a priori (atlas prob.)
- Phase d’apprentissage
- Lent



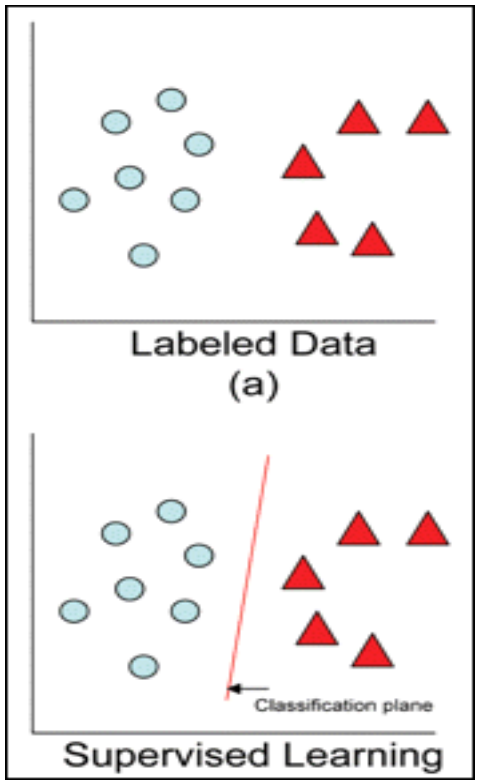
Retour en DI4….

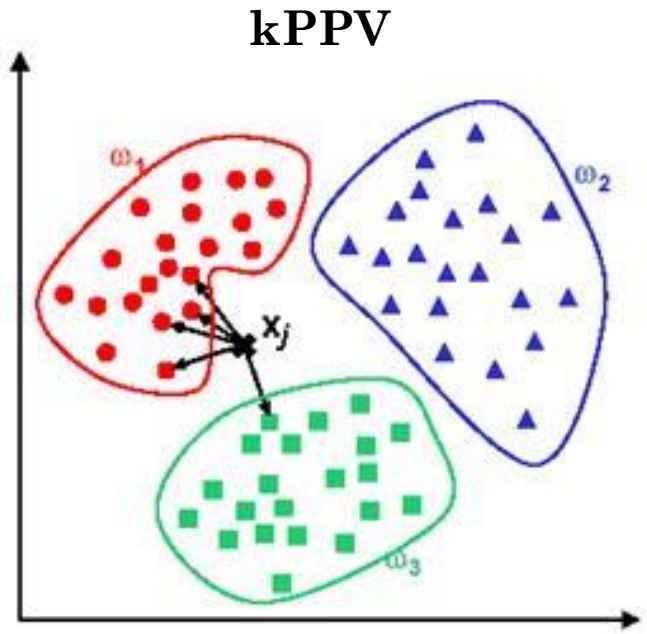
Comment segmenter une image avec une approche « analyse de données » / « classification de données » ?

Approche basée classification

Exemple de classificateur : k - plus proches voisins (kPPV/kNN)

- On dispose d'un ensemble d'entrainement composé de vecteurs caractéristiques et de leur classe
- On classe le vecteur inconnu x dans la classe la plus représentée parmi les k plus-proches voisins du vecteur x





Utilisation en CV ?

Approche basée classification

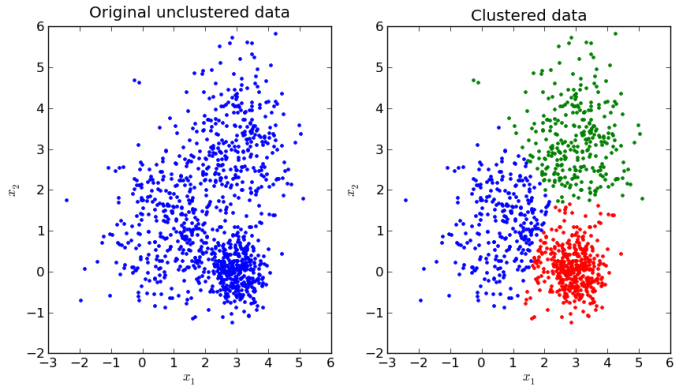
Exemple de méthodes de clustering : k-means

- Lorsqu'on ne dispose pas d'échantillons classifiés

⇨ apprentissage impossible

- On cherche à identifier K classes au moyen de leurs centres

Objectif : minimiser la variance intra-classe



Algorithme :

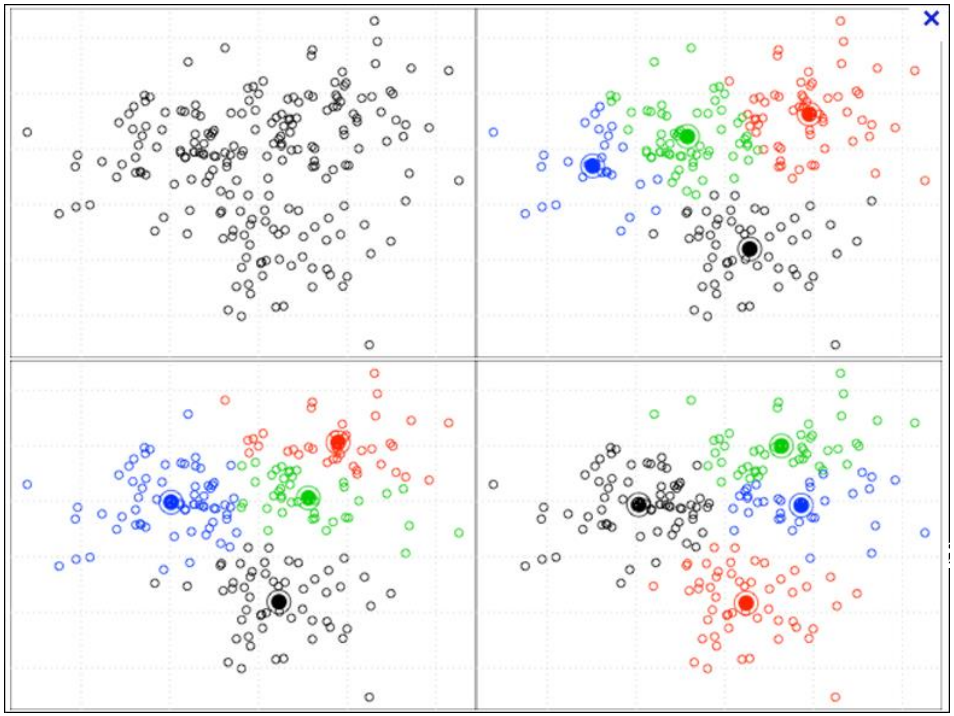
1 - choisir K centres arbitrairement

2 – répéter :

a/ affecter à chaque x son centre le plus proche

b/ recalculer le centre de chaque classe comme étant la moyenne des vecteurs de la classe

jusqu'à obtenir des moyennes stables.



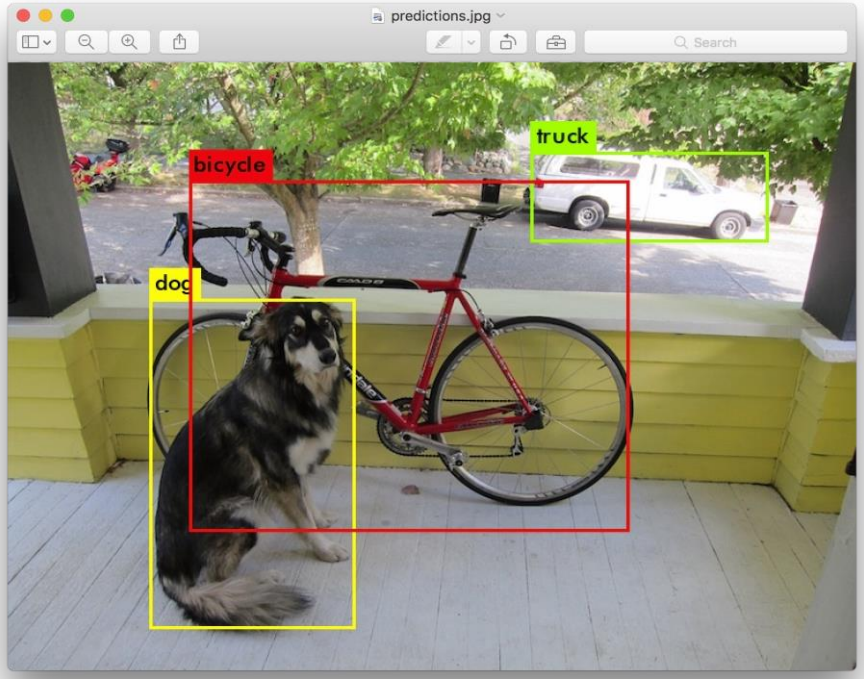
Approche basée classification

Deep learning et Images

Divers objectifs :

- Image classification
- Image tagging, object classification
- **Panoptic** / Semantic / Instance segmentation
- Visual Question Answering ...

The label encoding of pixels in panoptic segmentation involves assigning each pixel of an image two labels – one for semantic label, and other for instance id.

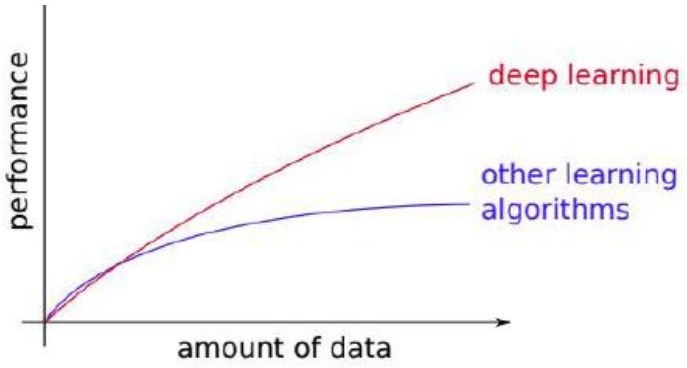


Définitions et justifications

- "Deep Learning is learning good representations by composition of learned functions." (From Yoshua Bengio)
  - Composition = Complicated functions are combinations of smaller, simpler functions.
- "Deep Learning is building a system by assembling parameterized modules into a computation graph, and training it to perform a task by optimizing the parameters using a gradient-based method." (From Yann Lecun)
- So deep learning is (From Yoshua Bengio)
  - (1) an approach to machine learning
  - (2) usually based on artificial neural networks but not only
  - (3) based on learning multiple levels of representations or based on composing learned functions to learn good representations
  - (4) in order to achieve the learning of such good representations and perform well in machine learning tasks.

- Why now ?

Computing power (GPUs)

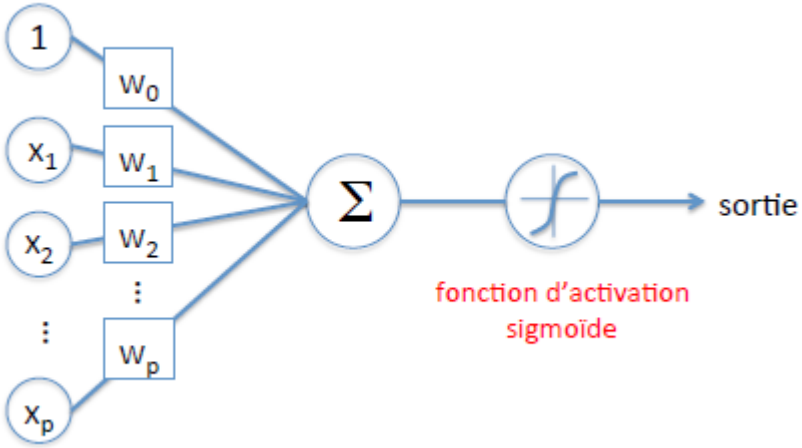


Labeled datasets

Principes de base…

Encore une histoire de « neurones »…

Entrée
(hand-crafted features)



Prédiction
(decision)

Sortie :

$$g(\mathbf{x}) = \frac{1}{1 + \exp\left[-(w_0 + w_1 x_1 + \ldots w_p x_p)\right]}$$

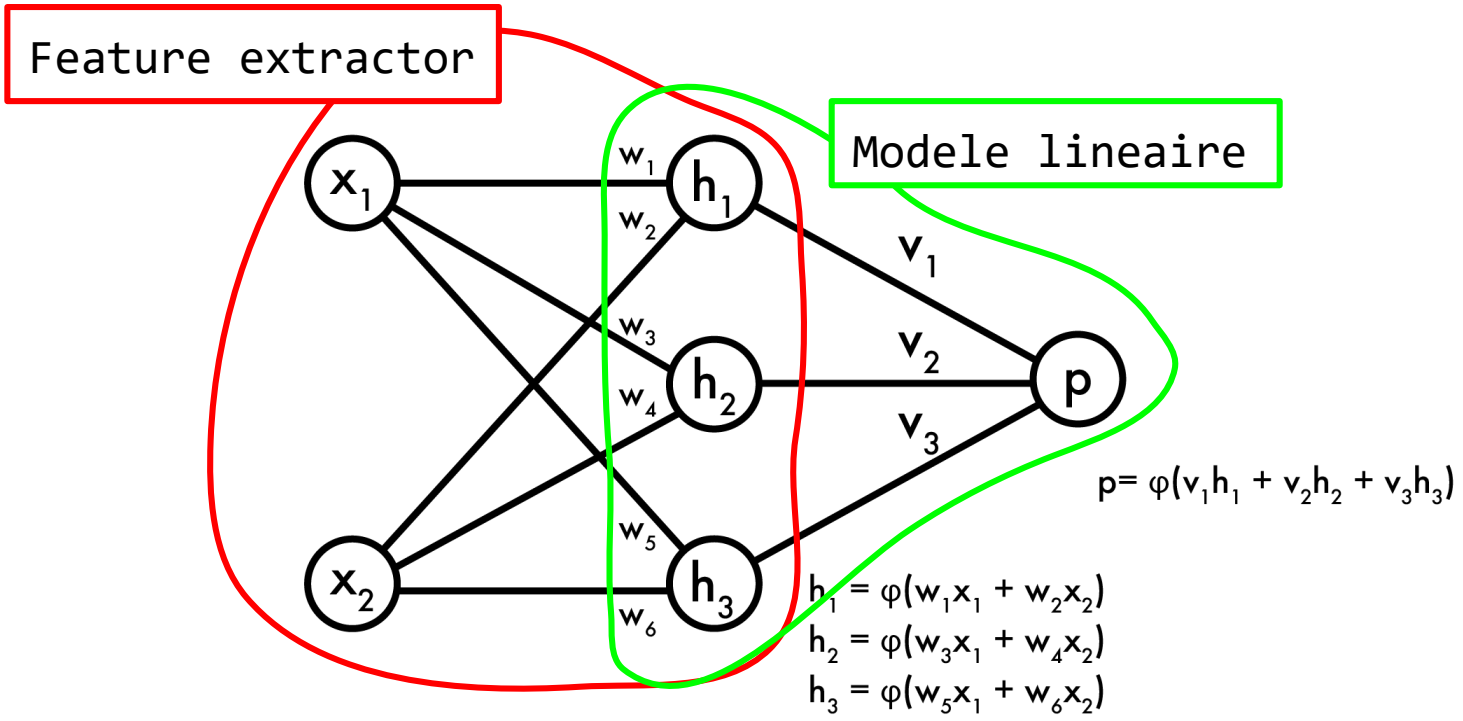
Au départ, le modèle de la régression logistique…

- Pas de couches cachées → On passe des entrées (features) à la décision directement
- Apprentissage des poids par minimisation de l'erreur de prédiction….

Principes de base…

Outrepasser le problème de "feature engineering"

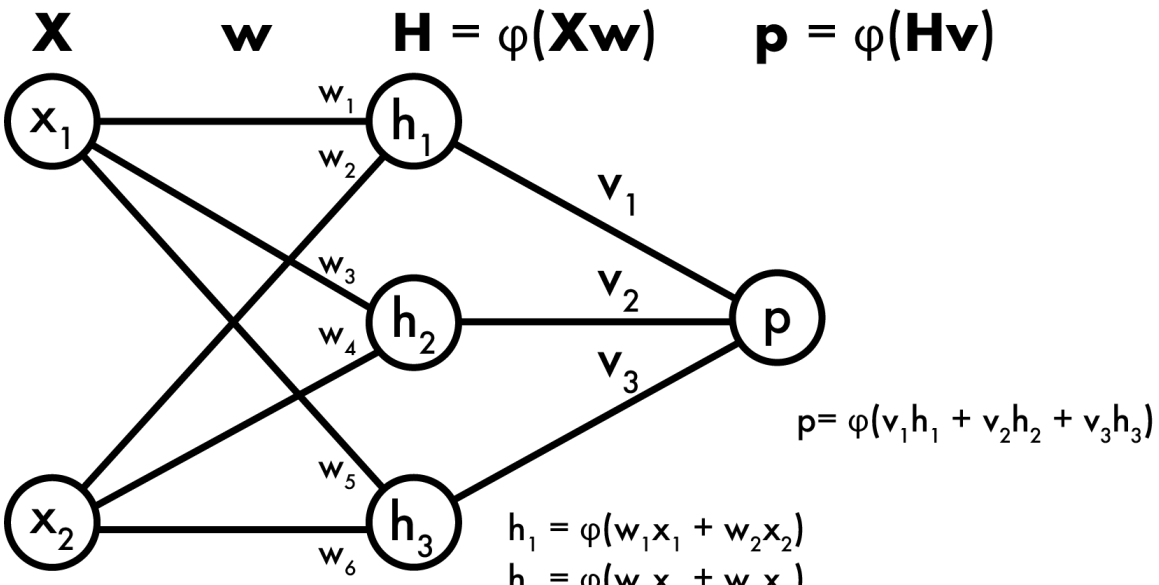
- Et si on ajoutait des transformations → couches cachées
- Maintenant, les prédictions p sont fonction de valeurs de la couche cachée



Principes de base…

Ici commence à apparaitre la notion / question du Deep Learning

- Il est possible de multiplier le nombre de couches cachées
- Chaque couche correspondant à des fonctions φ appliquées aux combinaisons linéaires de la couche précédente

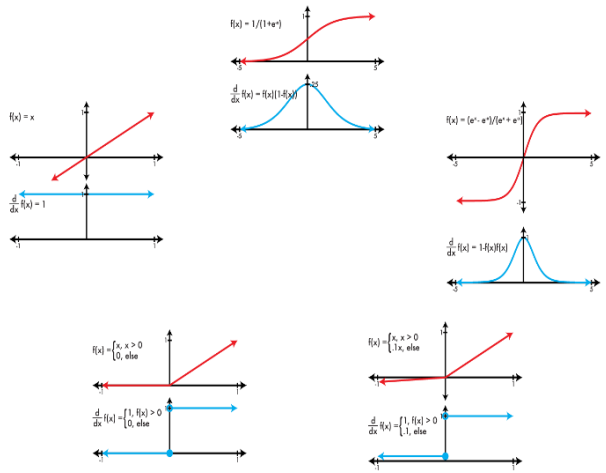


$p = \varphi(v_1 h_1 + v_2 h_2 + v_3 h_3)$

$h_1 = \varphi(w_1 x_1 + w_2 x_2)$
$h_2 = \varphi(w_3 x_1 + w_4 x_2)$
$h_3 = \varphi(w_5 x_1 + w_6 x_2)$

What about activation functions φ ?

- So many possible options
- want them to be easy to take derivative



Principes de base…

"Shallow" vs Deep Learning ?

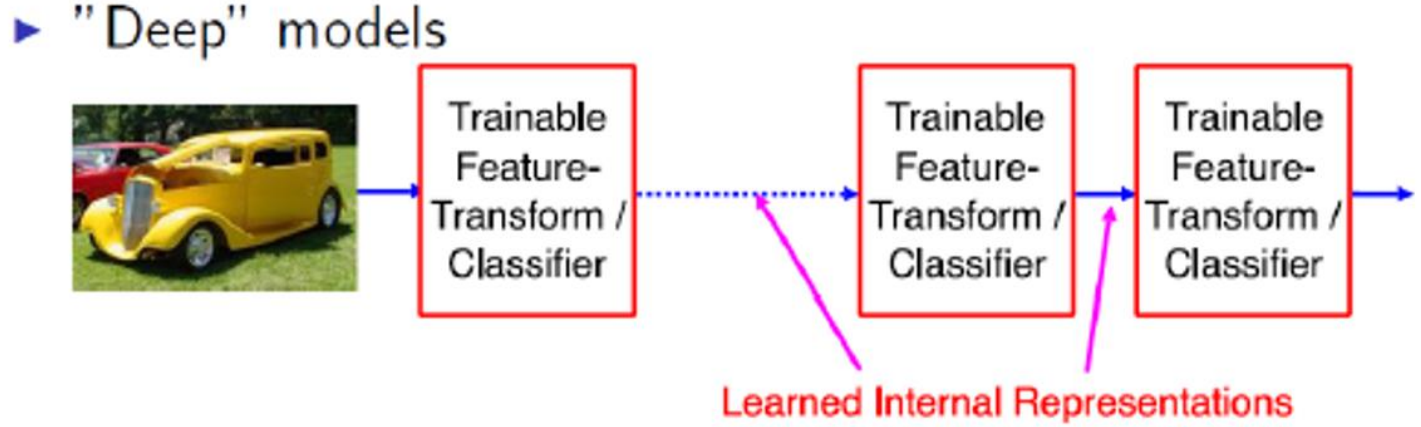
- "Shallow" models

- "Deep" models

Principes de base…

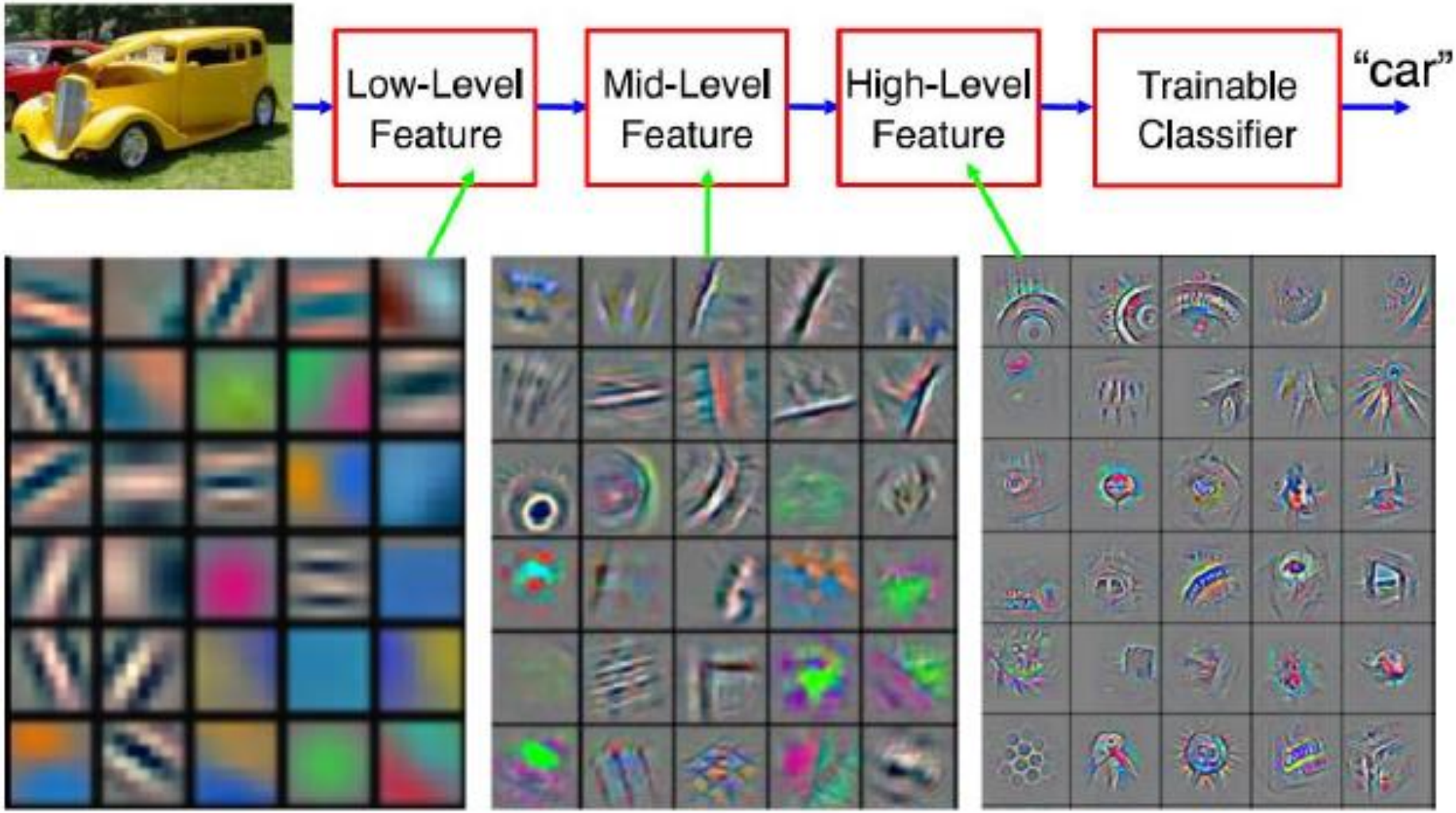
Deep Learning → End-to-End learning

- Un empilement de modules de transformation de représentations
- Chaque module transforme les données d'entrée en une représentation de plus haut niveau sémantique
- Les ***features*** de bas niveau sont partageables / génériques
- Les ***features*** de plus haut niveau sont généralement plus structurelles

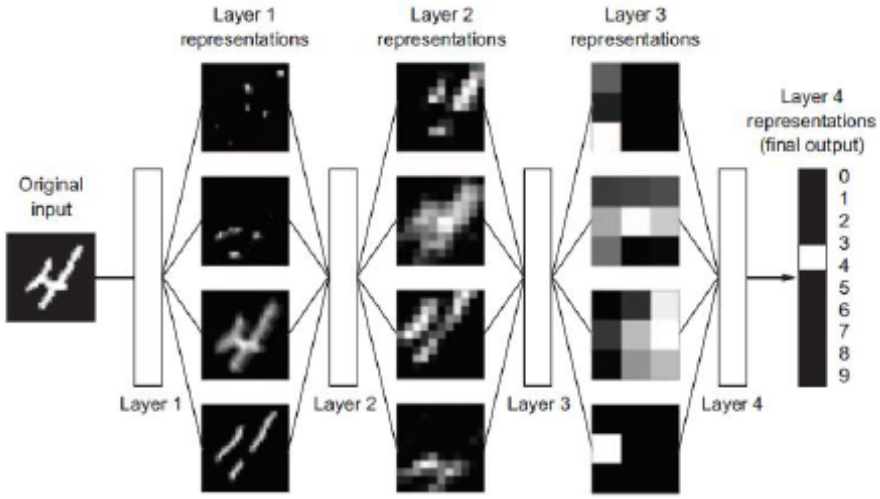


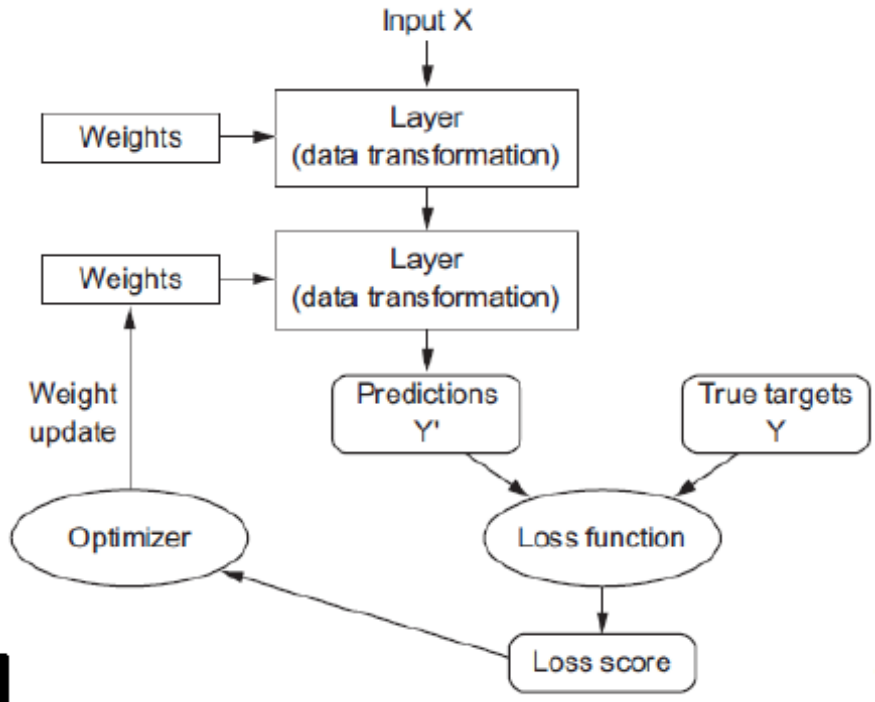
Principes de base…

Apprentissage d'un empilement de représentations "abstraites"



Apprentissage - Principes…



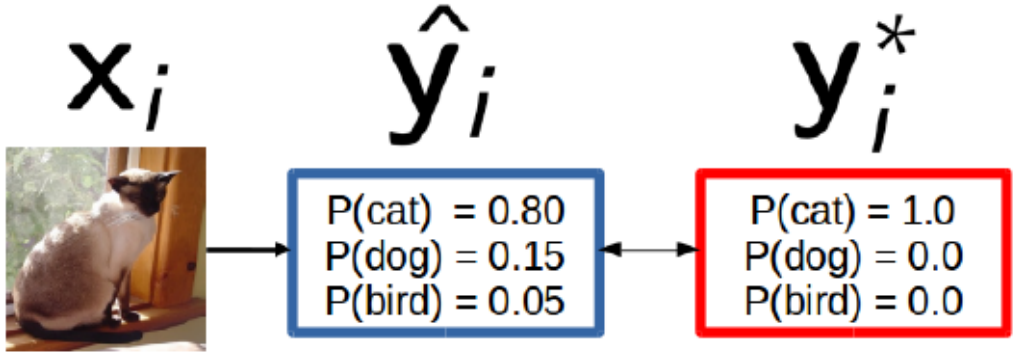


Deep Learning is about finding a good data represention with respect to an objective thanks to a composition of transformations (functions/layers)

Apprentissage

Apprentissage

- Apprentissage
  - Trouver les w_{ij} minimisant l'erreur d'apprentissage $\mathcal{L}(W)$ sur la base d'apprentissage



$$\ell_{CE}(\hat{y}_i, y_i^*) = KL(y_i^*, \hat{y}_i) = -\sum_{c=1}^{K} y_{c,i}^* log(\hat{y}_{c,i}) = -log(\hat{y}_{c^*,i})$$

  - $\mathcal{L}(W)$ servira de base à la définition de la fonction *Loss()* = ∩ → à bien définir !
  - Souvent Loss() MLP = ∩ = Moyenne des erreurs sur 1 lot d'apprentissage de taille N

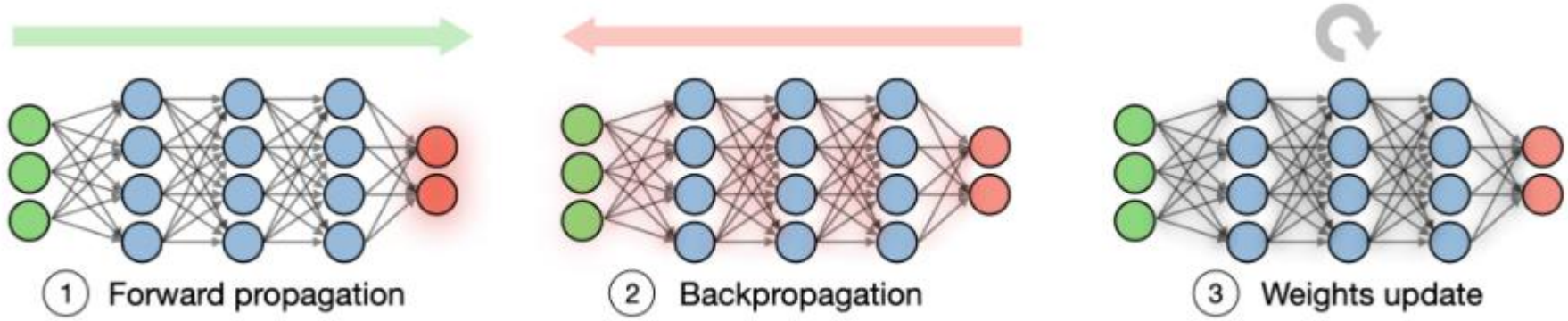
$$\mathcal{L}_{CE}(W, b) = \frac{1}{N} \sum_{i=1}^{N} \ell_{CE}(\hat{y}_i, y_i^*) = -\frac{1}{N} \sum_{i=1}^{N} log(\hat{y}_{c^*,i})$$

  - Utiliser la descente de gradient pour la minimisation de cette fonction

Apprentissage

Descente du gradient

- 3 étapes principales



- Processus itératif de modification des poids selon le signe du gradient :

$$W^{(t+1)} = W^{(t)} - \eta \frac{\partial \mathcal{L}_{CE}}{\partial W}$$

Sur la base d'appr.

$$\nabla_w^{(t)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \ell_{CE}(\hat{y}_i, y_i^*)}{\partial w} \left(w^{(t)}\right)$$

  - η est le taux d'apprentissage

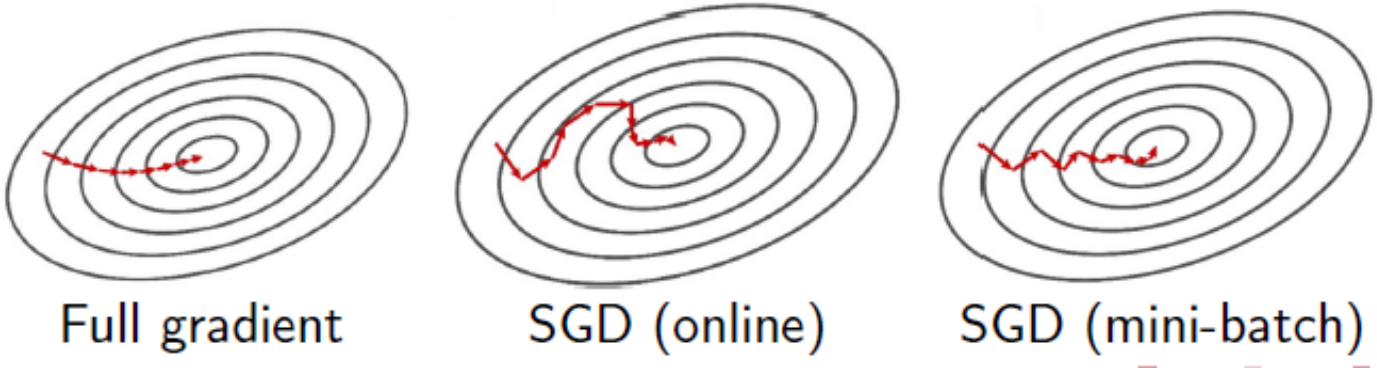
- La propriété principale exploitée : chaine de dérivation (chain rule)
- **Pour un MLP :**

$$\frac{\partial x}{\partial z} = \frac{\partial x}{\partial y}\frac{\partial y}{\partial z} \Rightarrow \frac{\partial \ell_{CE}}{\partial W} = \frac{\partial \ell_{CE}}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial s_i}\frac{\partial s_i}{\partial W}$$

Apprentissage

Descente du gradient

- Le gradient peut être calculé pour
  - Toute la base N exemples → Full-Gradient → Convergence très lente !
  - Chaque exemple → gradient stochastique (SGD)
  - Un mini-batch → 1 sous-partie B de la base d apprentissage N → Nb batches = N / B
  - SGD → Beaucoup mises à jour de paramètres : N,
  - Mini-batch → N / B → Convergence plus rapide
  - Approximation du vrai gradient



- 1 époque = 1 exploitation de tous les exemples de la base d'apprentissage

Apprentissage

Réglages des hyper-parametres

- Choix de la valeur du Learning Rate η ?
- Diminution pendant la progression de l'apprentissage
  - Décroissance inverse : $\eta = \eta_0 / (1 + \lambda.t)$, λ = taux de décroissance
  - Décroissance exponentielle : $\eta = \eta_0 . exp(\lambda.t)$,
  - Décroissance par paliers : $\eta = \eta_0 . r^{(t / t_u)}$

Method	Explanation	Update of w	Update of b
Momentum	• Dampens oscillations • Improvement to SGD • 2 parameters to tune	$w - \alpha v_{dw}$	$b - \alpha v_{db}$
RMSprop	• Root Mean Square propagation • Speeds up learning algorithm by controlling oscillations	$w - \alpha \frac{dw}{\sqrt{s_{dw}}}$	$b \longleftarrow b - \alpha \frac{db}{\sqrt{s_{db}}}$
Adam	• Adaptive Moment estimation • Most popular method • 4 parameters to tune	$w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \longleftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

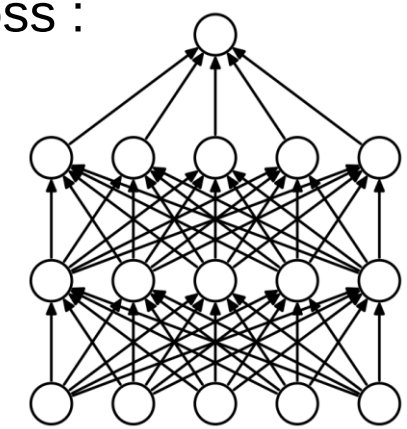
10)

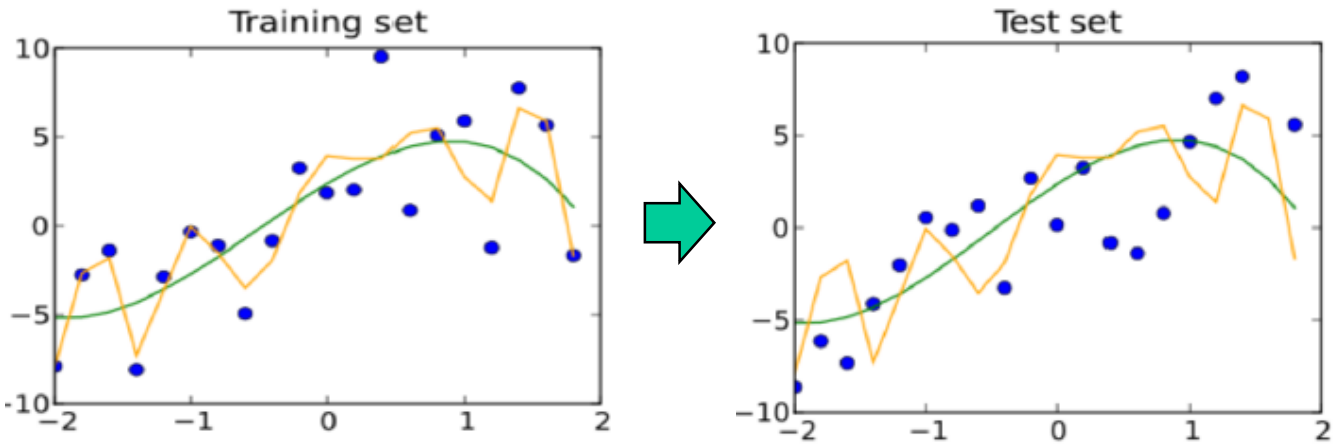
Remark: other methods include Adadelta, Adagrad and SGD.

Apprentissage

Réglages des hyper-parametres

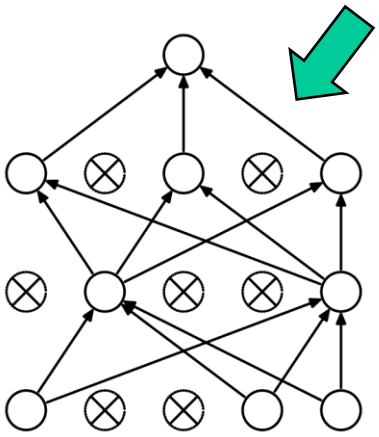
- Régularisation : amélioration de la généralisation, c'est-à-dire des performances sur la base de test
  - Régularisation structurelle: ajouter un terme *R(w)* dans la Loss :
  - $\mathcal{L}(w) = \mathcal{L}_{CE}(w) + \lambda.R(w)$
  - Régularisation dropout : désactivation aléatoire de neurones / epoch
  - Régilarisation L_2 : ↓↓ des poids, $R(w) = ||w||^2$
  - Régularisation L_1, …





$\mathcal{L}_{train} = 4$, $\mathcal{L}_{train} = 9$

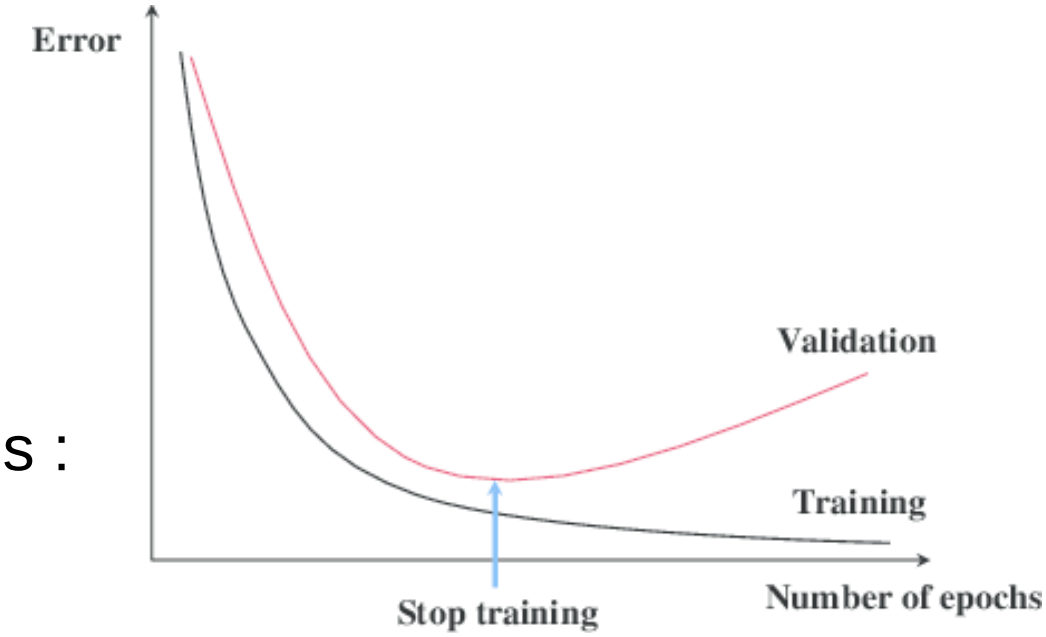
$\mathcal{L}_{test} = 15$, $\mathcal{L}_{test} = 13$



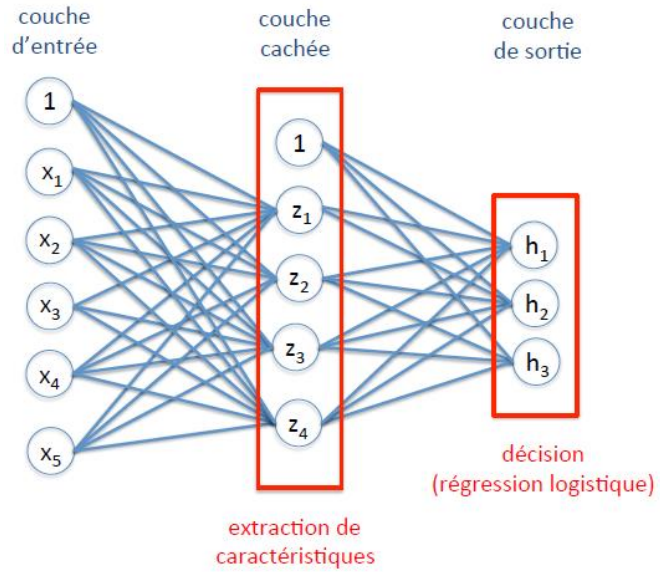
Apprentissage

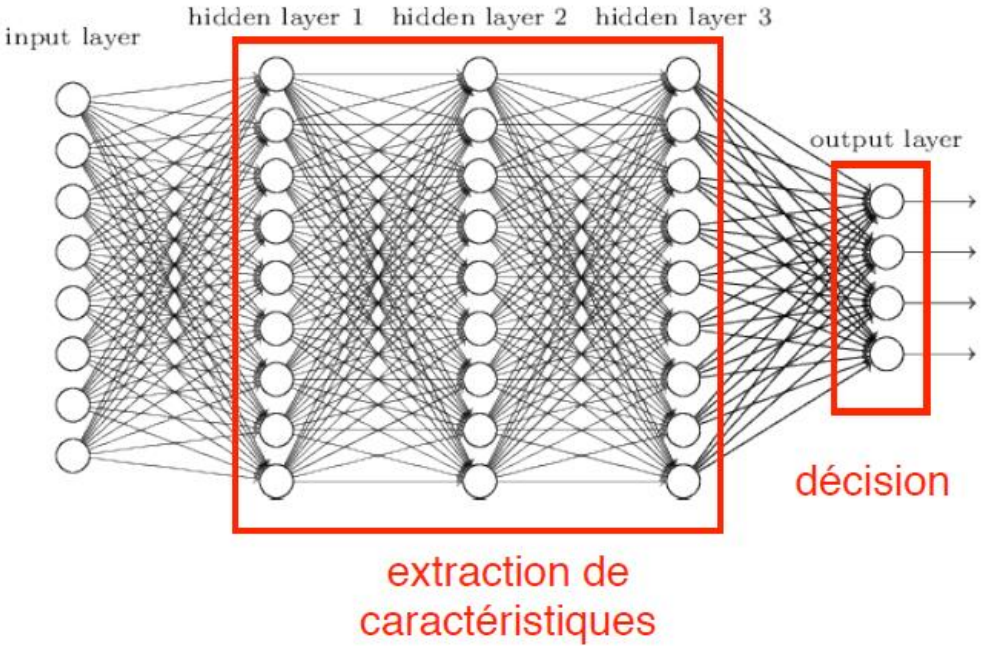
Réglages des hyper-parametres

- Paramètres d'apprentissage :
  - taux d'apprentissage, décroissance des poids, taille des mini batch, # époques, etc.
- Paramètres architecturaux :
  - nombre de couches, nombre de neurones,
  - type de non-linéarité,
  - etc.
- Réglage des hyper-paramètres :
  - estimer les performances sur un ensemble de validation en // de l'apprentissage



Fully Connected Networks (FCN)





Problèmes:

- Dimension d'entrée (3xd) - Nb pixels rgb !
- Très grand nombre de variables (W)
- L'apprentissage des poids se fait en minimisant une fonction d'erreur (Loss function) non-convexe → beaucoup de minima locaux.

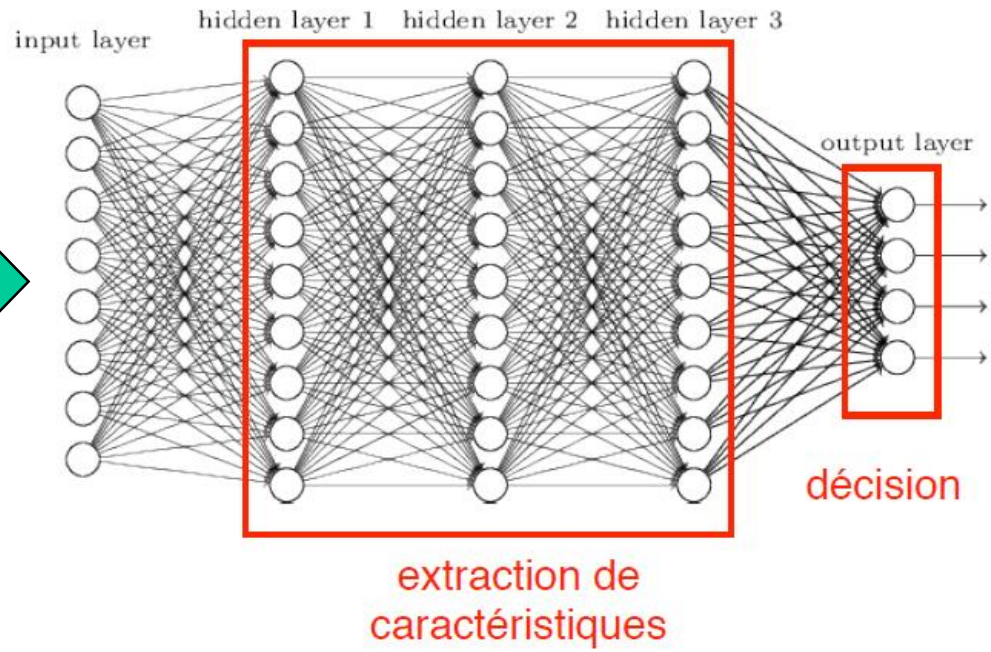
➔ Nécessité de grosses capacités de calcul

➔ Nécessité de simplifier le PB : FCN → CNN

m = nb images
n = nb outputs

- $X \in \mathbb{R}^{m \times d}$: input
- $X_i \in \mathbb{R}^{1 \times d}$
- d : 640x480 pixels = 307200
- W$\in \mathbb{R}^{d \times n}$
- B$\in \mathbb{R}^{n}$; $\sigma(a)$; $\hat{Y} \in \mathbb{R}^{m \times n}$: output
- $\hat{Y} = \sigma(XW + B)$
- Issues :
  - W : is huge (one parameter by pixel)
  - An image is reduced to a vector :
    - Spatiality is lost

Les CNN à la rescousse…



Problèmes:

- Mêmes performances d'un FCN sur des images permutés car pas de prise en compte de la topologie (images → Vecteur 1D → FCN)

- Prise en compte de la topologie + Réduction du nombre de paramètres

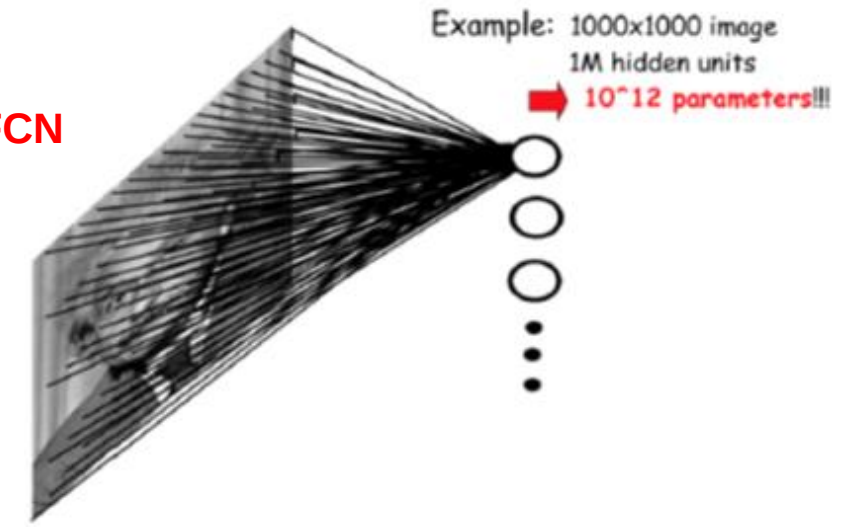
➔ ***Sparse connectivity dans les CNN***

Les CNN à la rescousse…

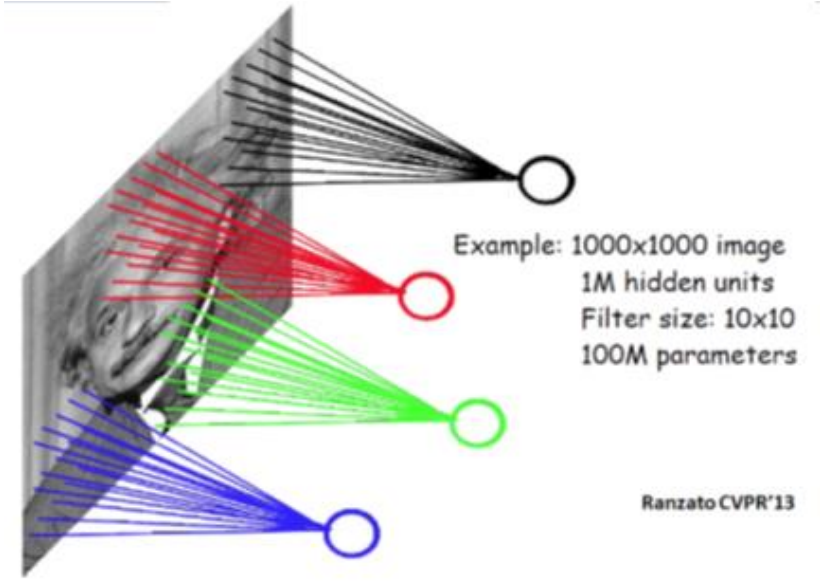
Réduction de complexité

Prior knowledge	CNN operation
Self similarity (Stationarity)	Shared parameters
Locality	Local connection
Translation invariance	Pooling
Compositionnality	Stackable

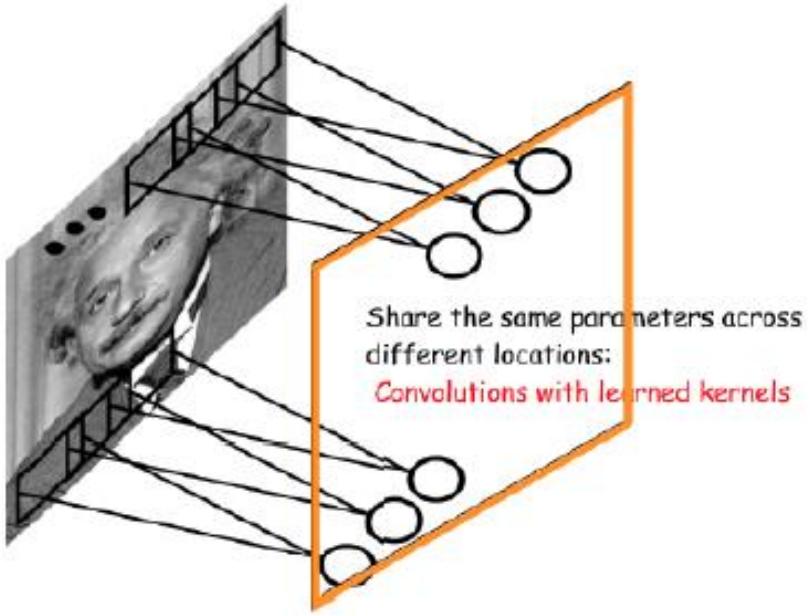
FCN



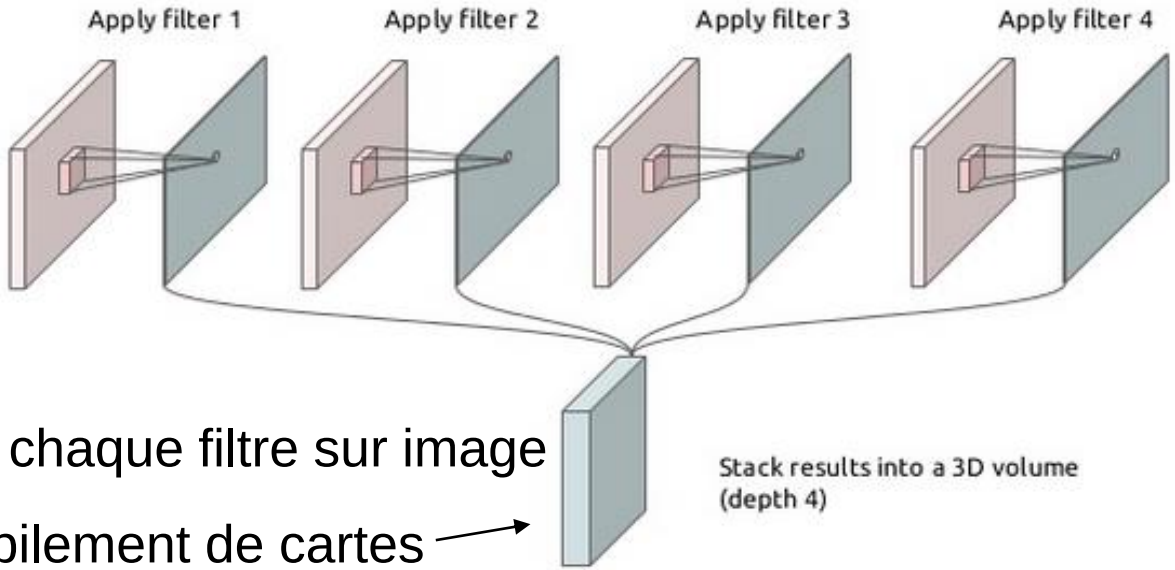
CNN → Localité → Topologie



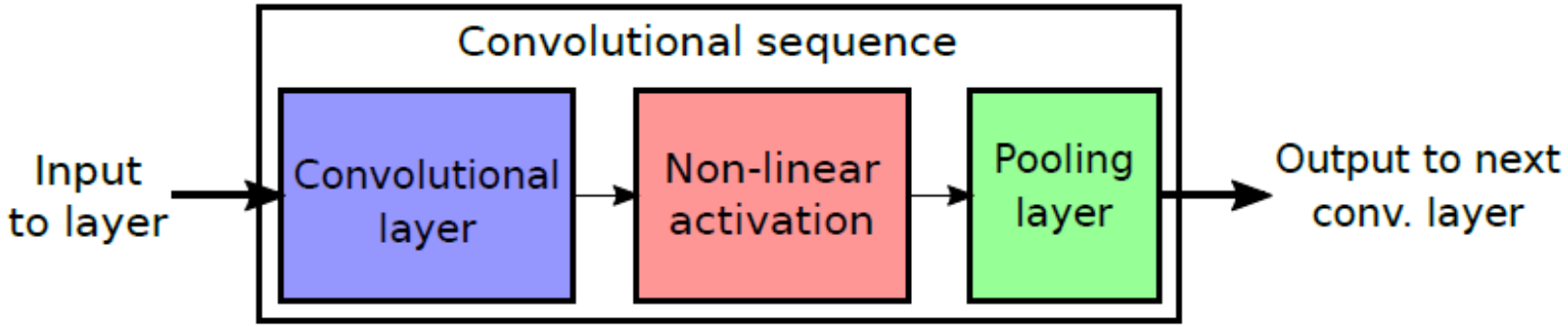
CNN → partage de W → auto-similarité



Les CNN : Tensors + Séquence de convolution



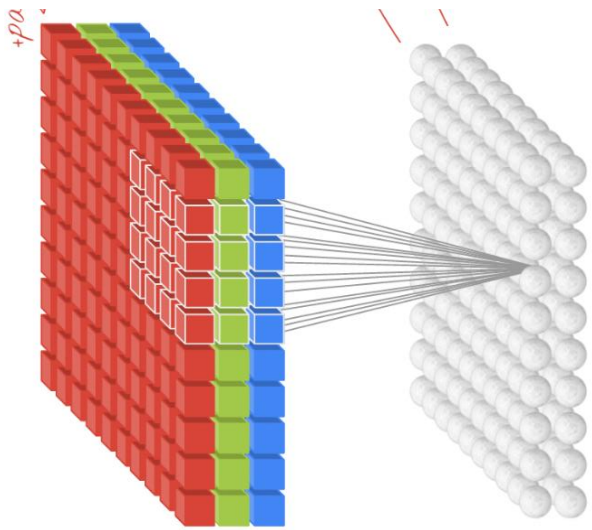
- Convolution 2D → 1 carte pour chaque filtre sur image
- Couches de convolution → empilement de cartes provenant de plusieurs filtres
- Tensor = tableau multidimensionnel
- **Séquence de convolution classique :**

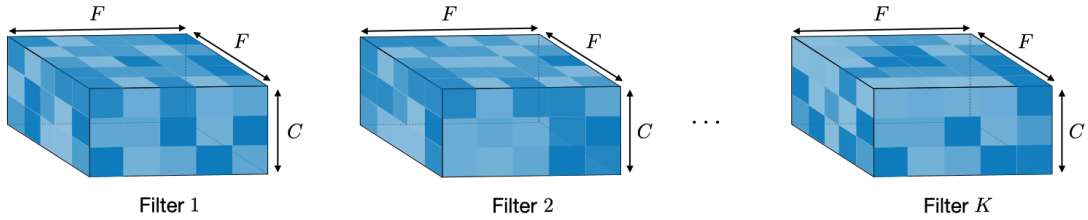


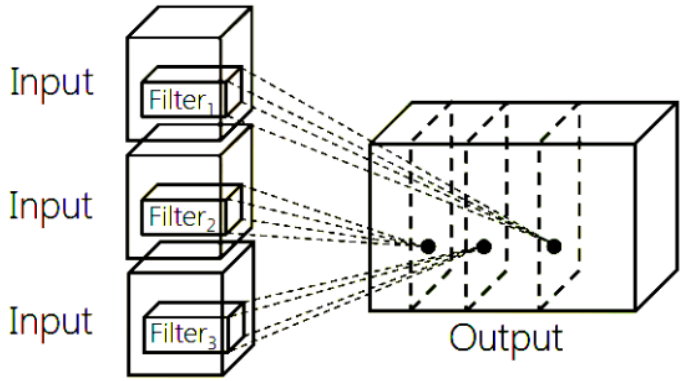
Les CNN : Convolution

- Convolution 2D *(FxF)* vs Convolution sur Tenseurs *(FxFxC)*
- Notion de profondeur *(C=depth)* - Images RVB *(C=3)*
- **RN → Ajout d'un biais dans les convolutions sur tenseur** (+1 paramètre)

$$f'(i,j) = (f \star h)(i,j) = \sum_{k=1}^{K} \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, m-j, k)h(n,m,k) \; + \; \mathrm{b}$$



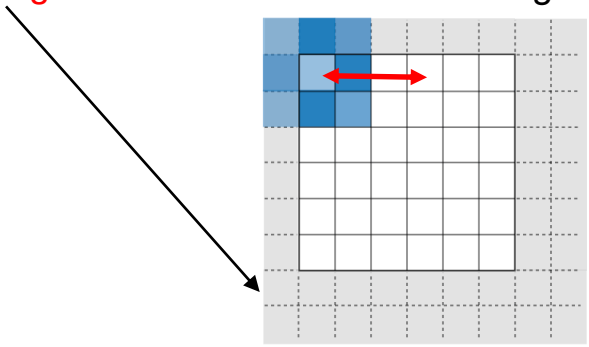


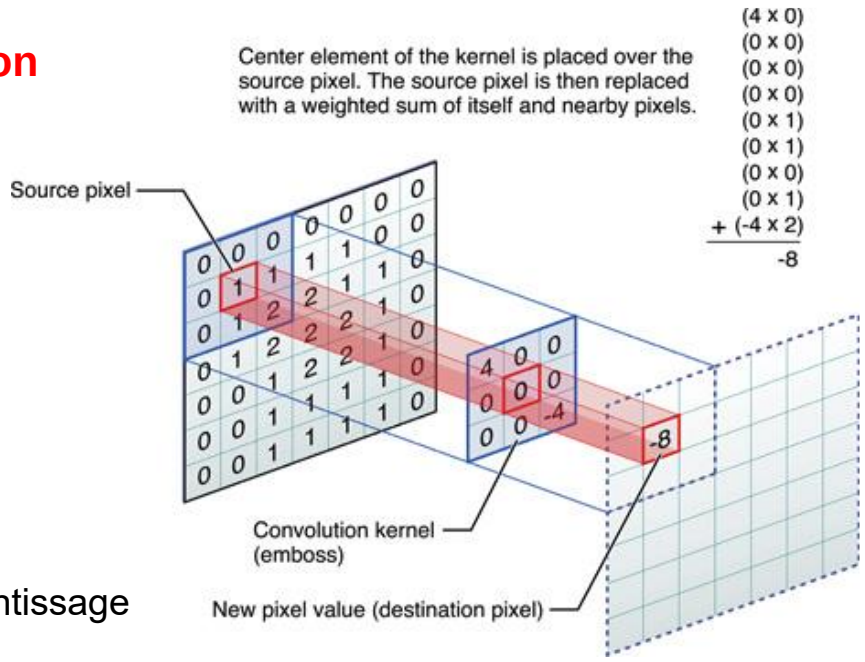


Les CNN : Convolution

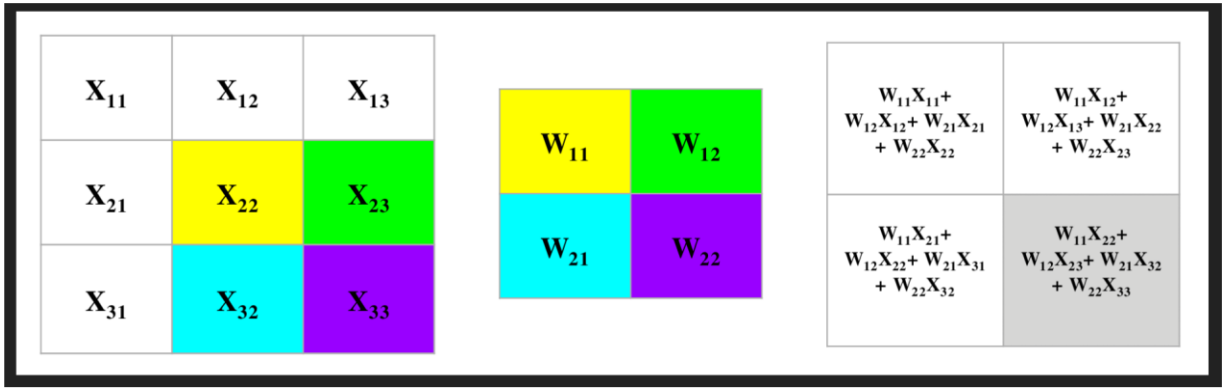
Les paramètres associés aux couches de convolution

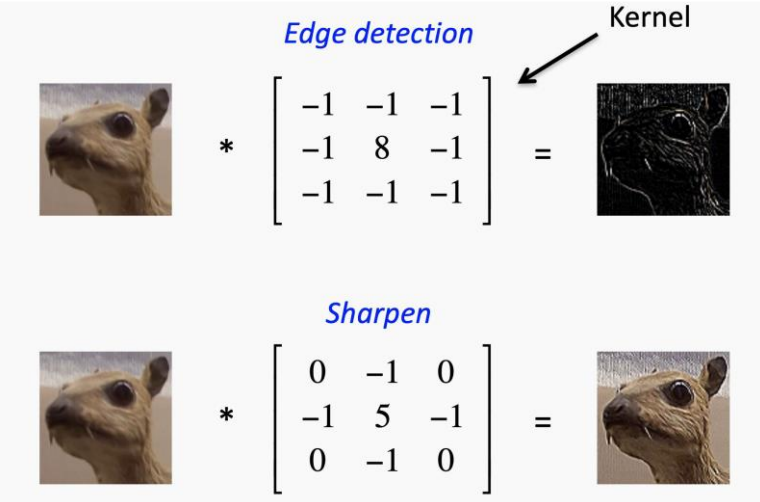
- Nombre et taille des filtres de convolution
- Stride = Décalage en nb de pixels
- Padding = Gestion des bords de l'image



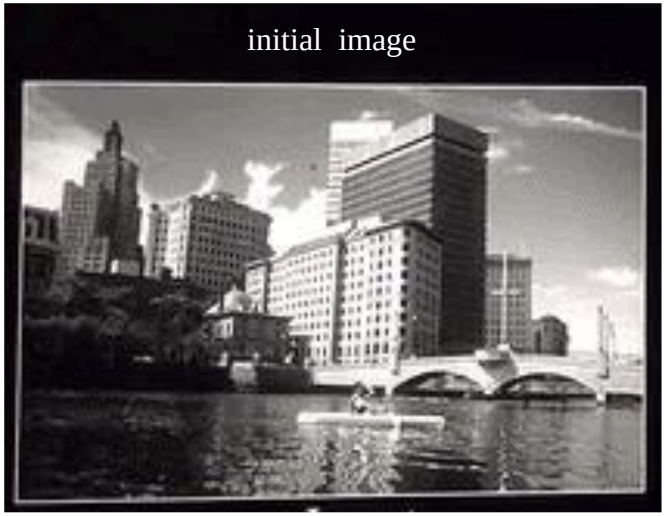


- Les poids w des filtres et les biais sont appris durant l'apprentissage
- Pour produire des ***Feature maps***
- Dépendentes de la taille des *receptive fields* (taille des filtres)





Les CNN : Activation non lineaire



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

tanh
$\tanh(x)$

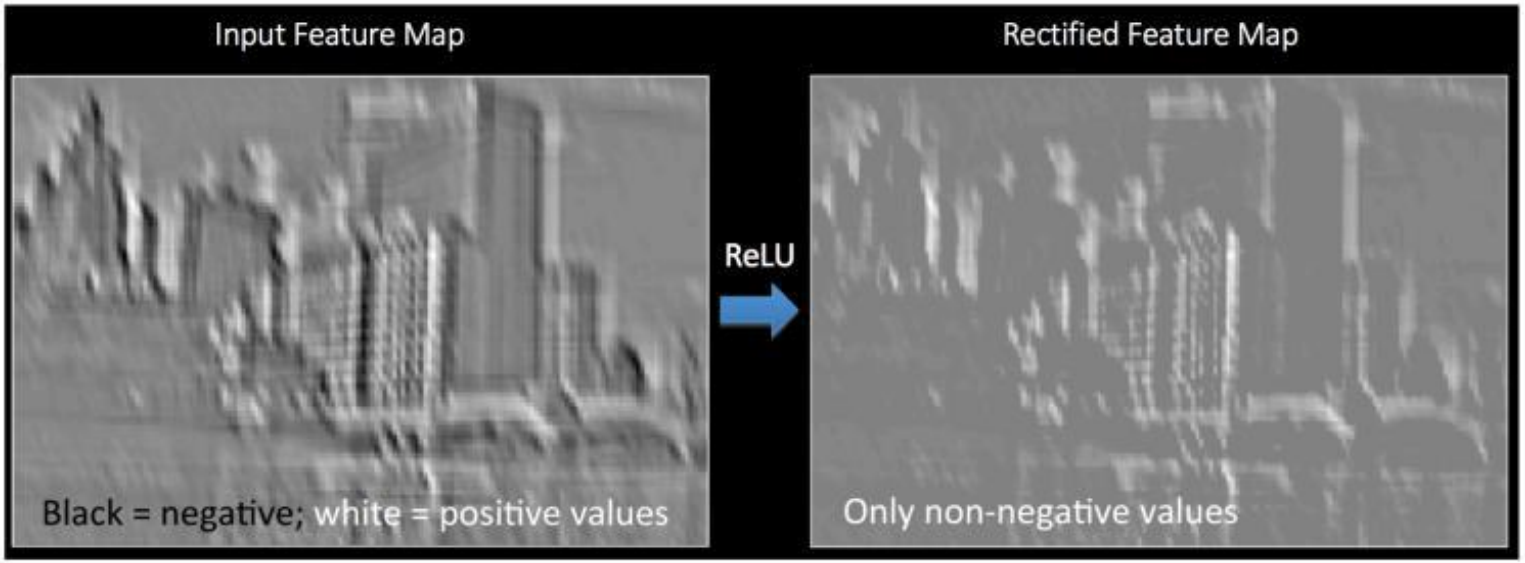
ReLU
$\max(0, x)$

Leaky ReLU
$\max(0.1x, x)$

Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

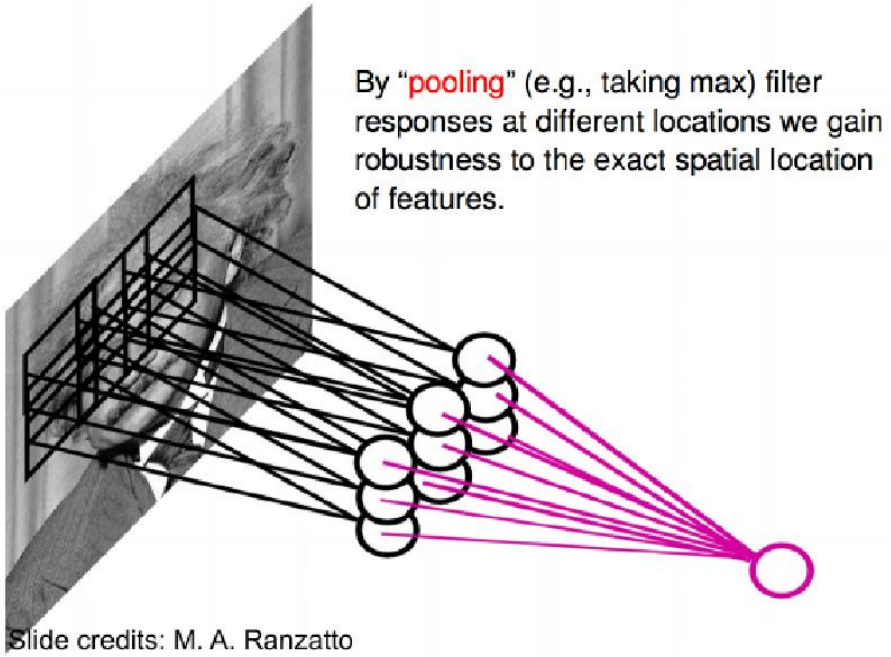
ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

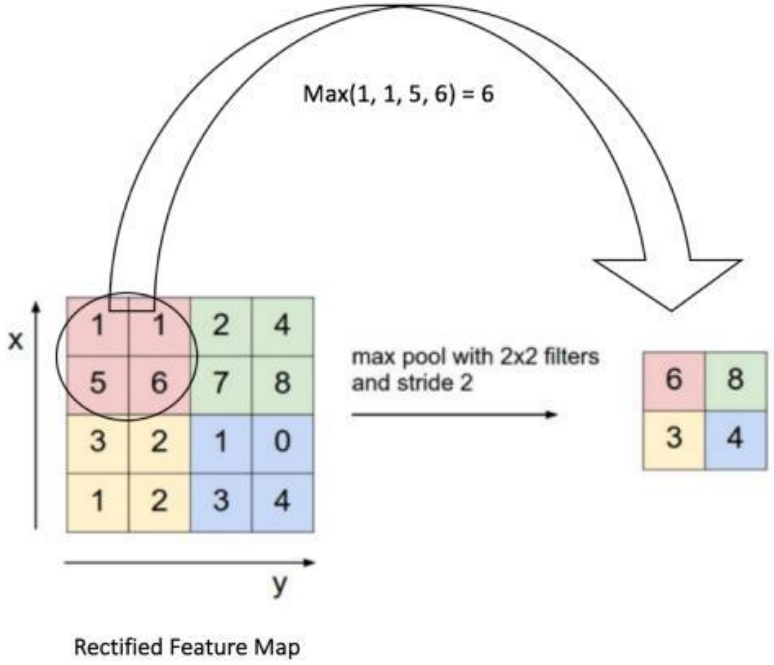
FIGURE 2.5: Activation functions. Figure from http://cs231n.stanford.edu/.



Activation non-lineaire (Relu, …)

Les CNN : Pooling

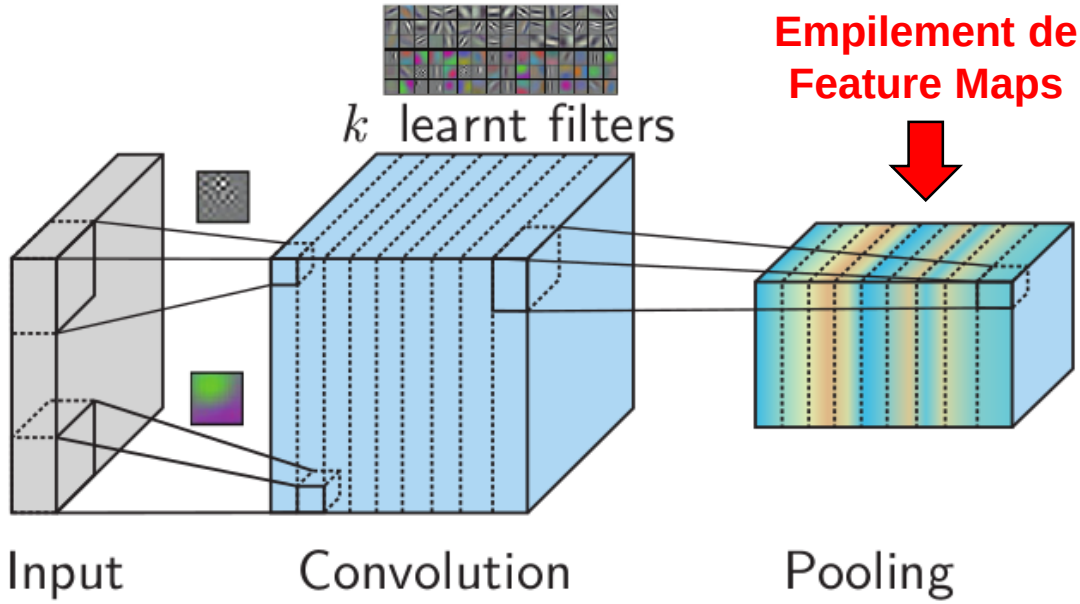




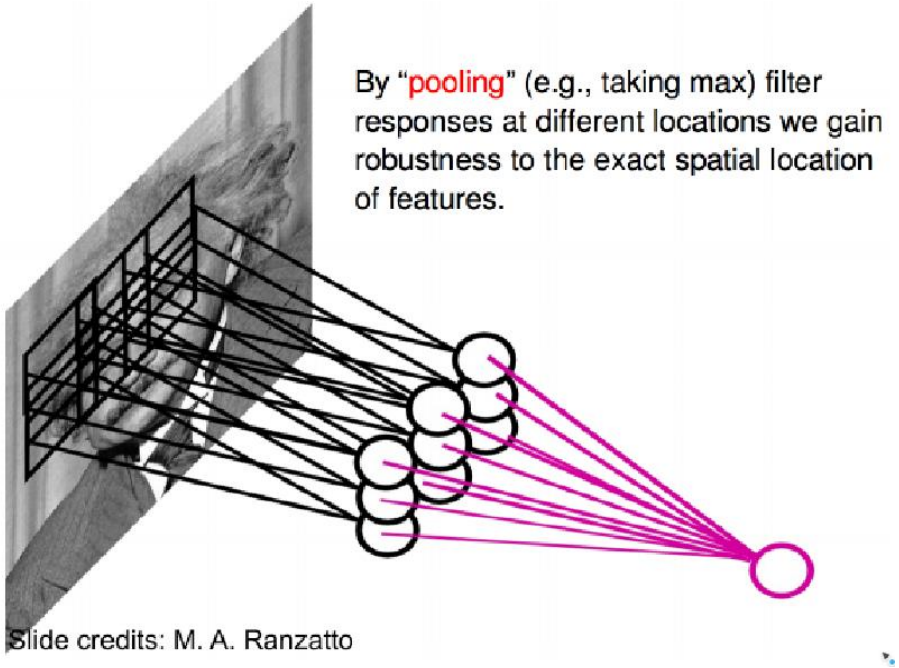
Pooling

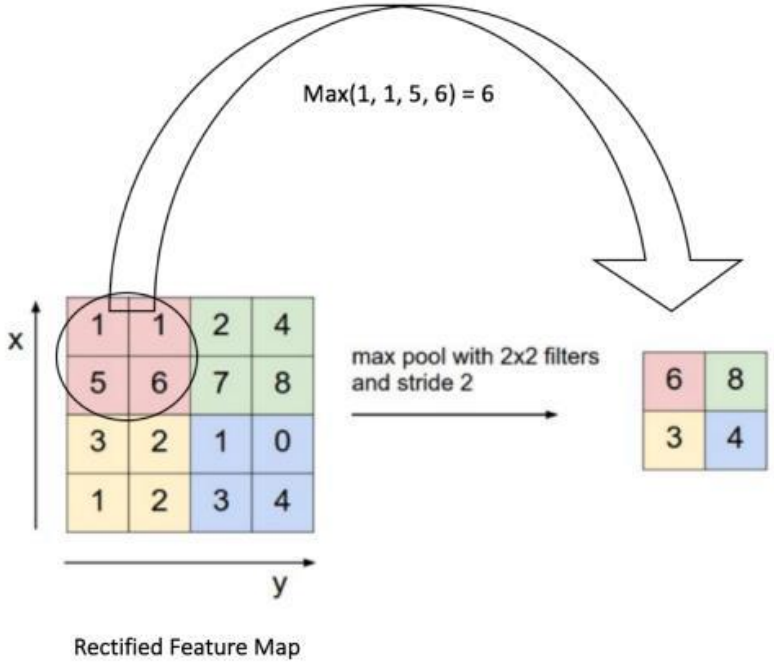
- Max
- Sum
- Average

…



Les CNN : Pooling

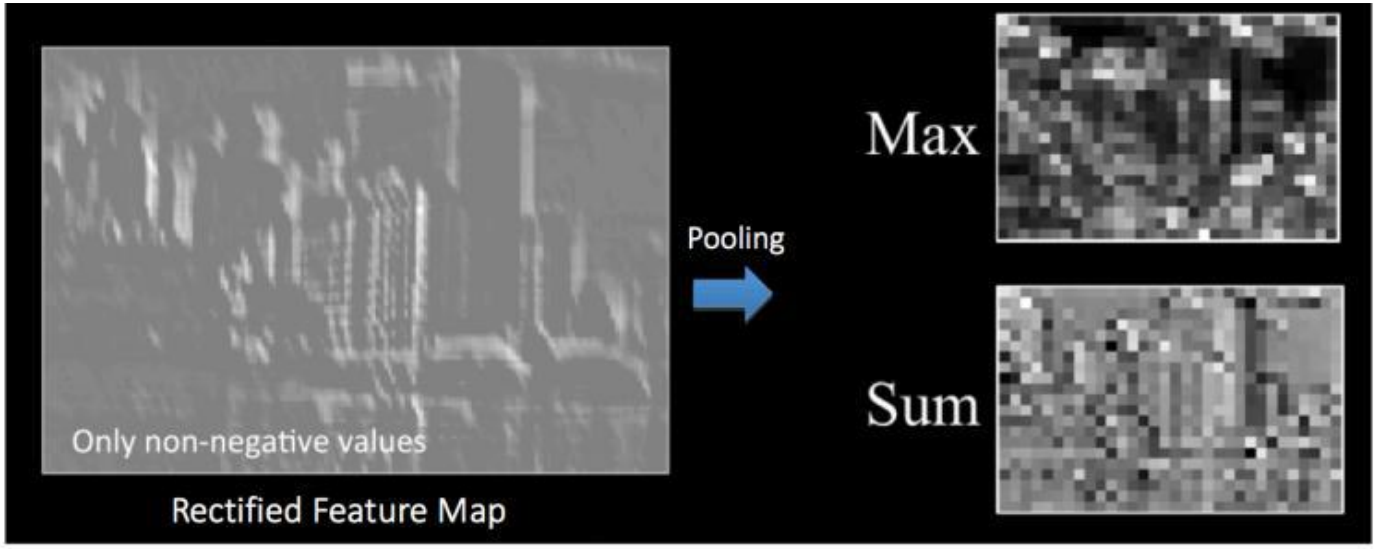




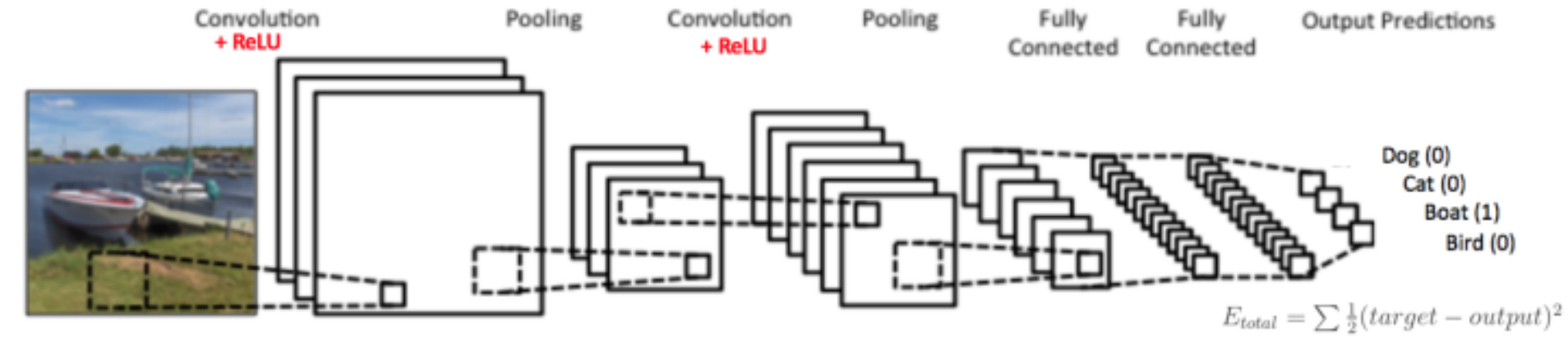
Pooling → Regroupement

- Max
- Sum
- Average

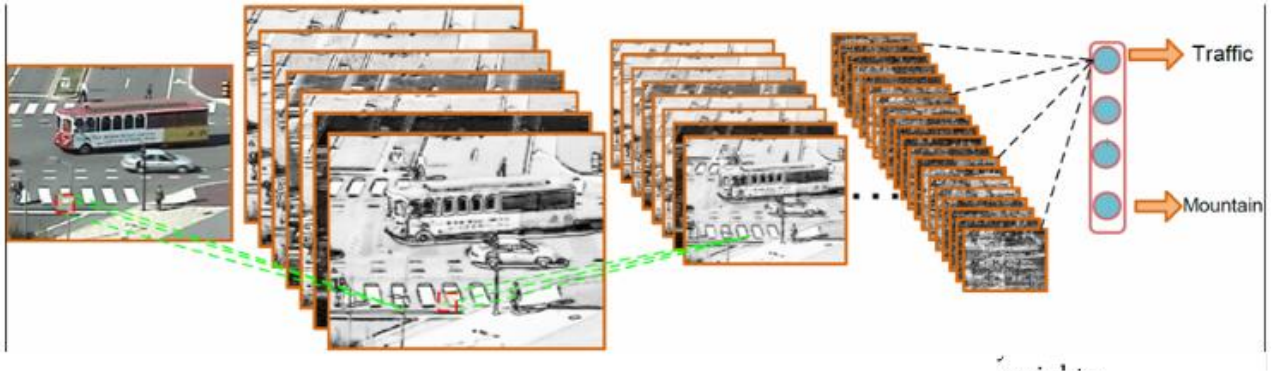
…



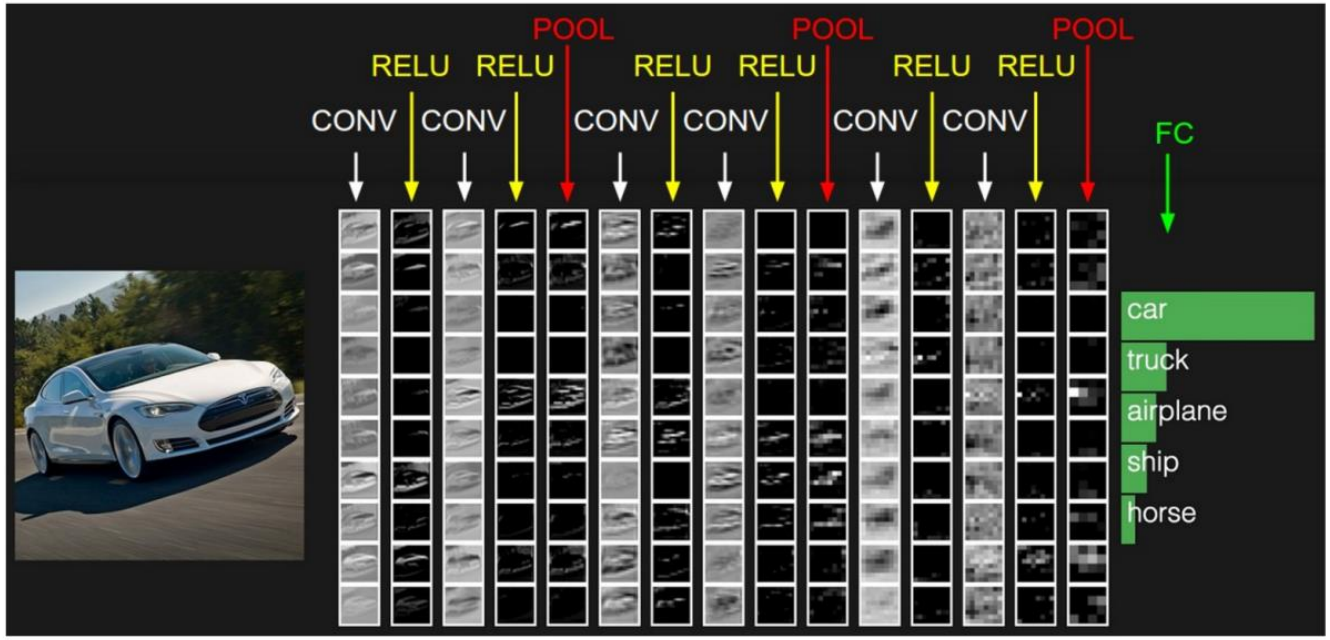
« Fully Connected Layer » pour finir…



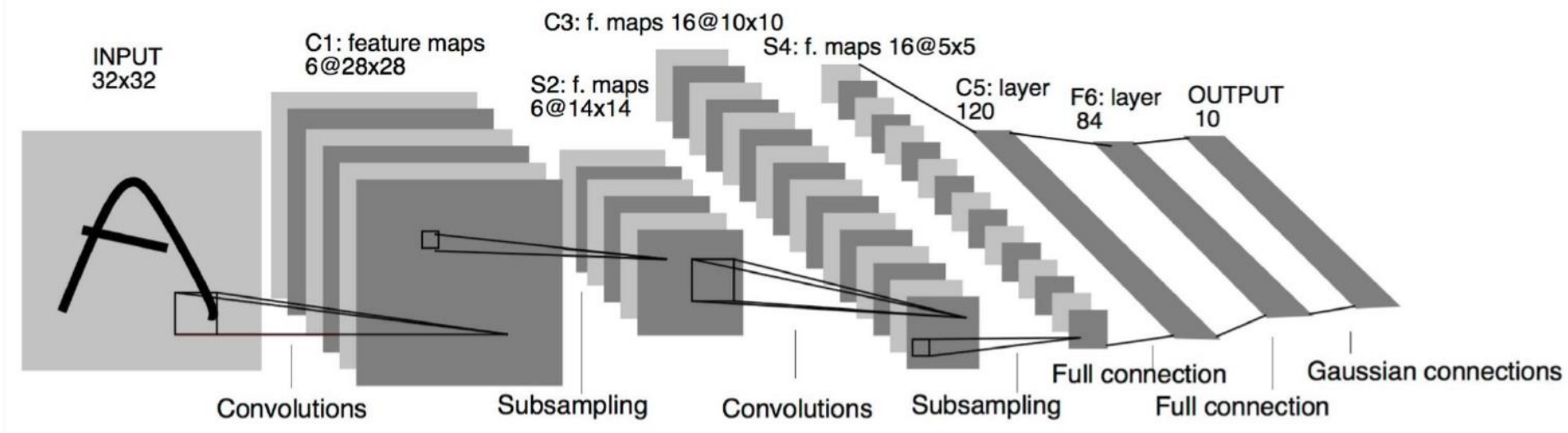
FCN → MLP classique
(connexions non représentées)



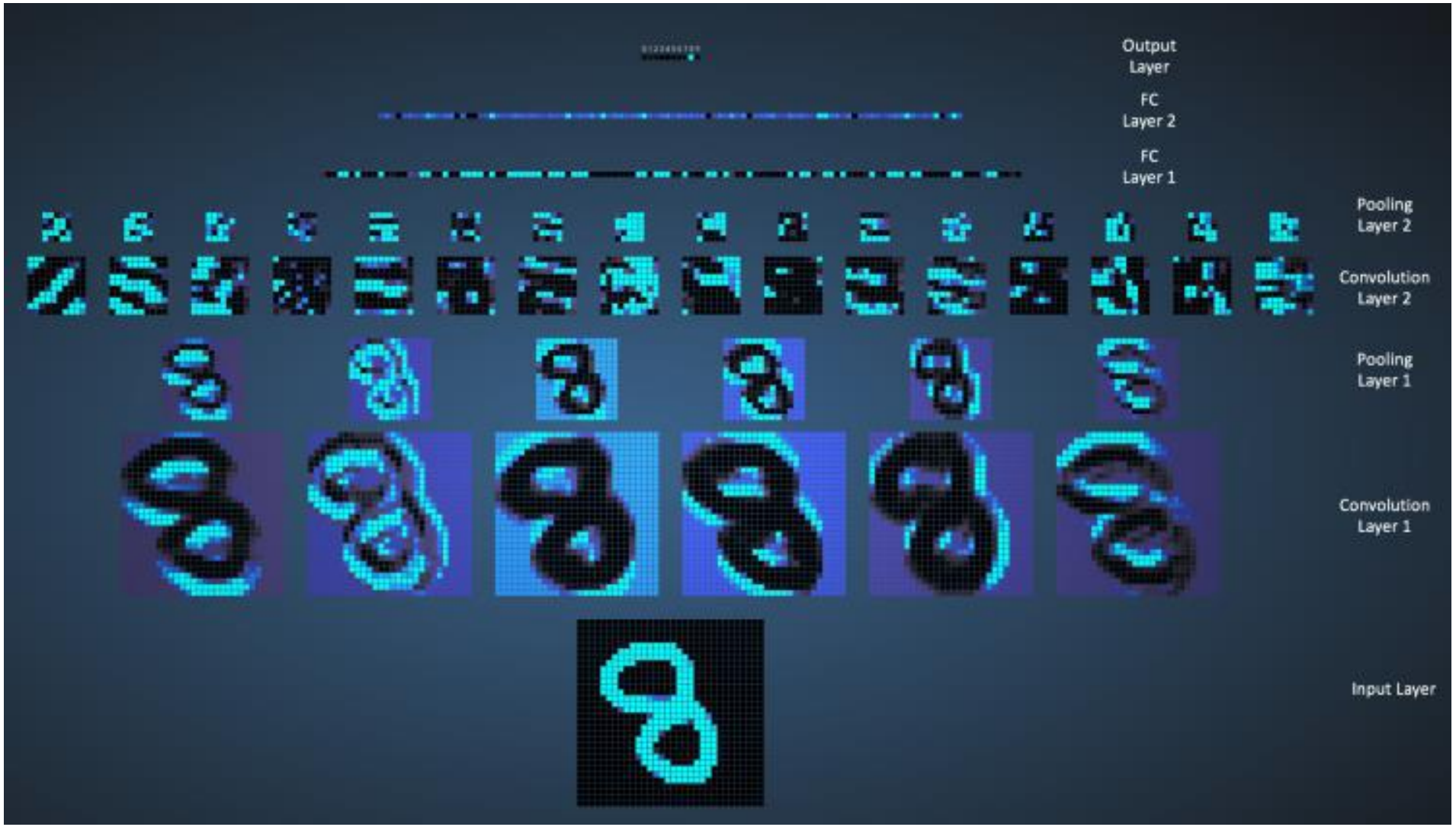
CNN : Architecture finale



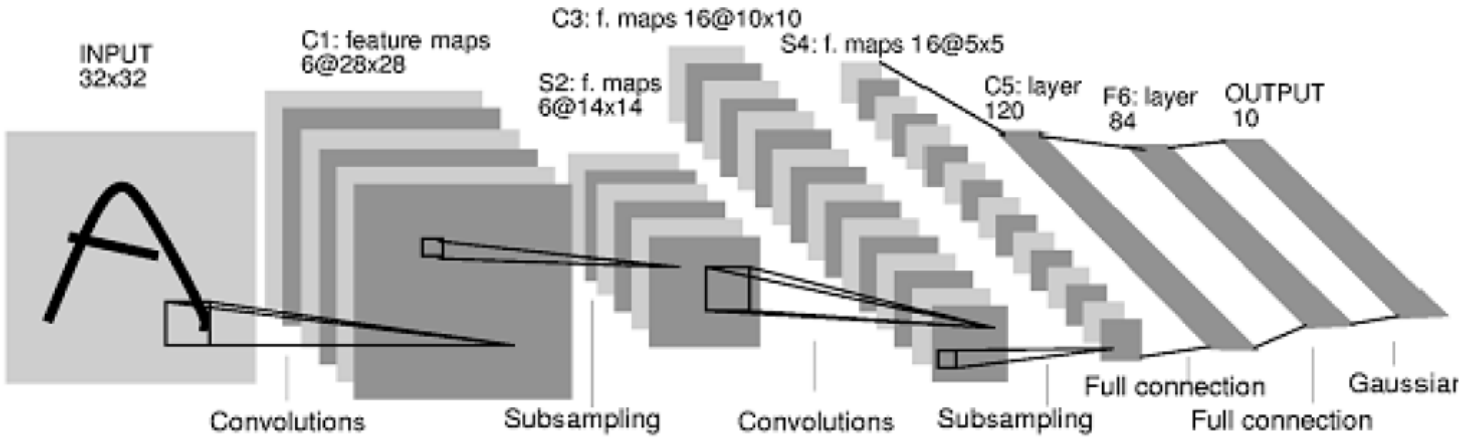
Remarque:
Même après ‘simplification”, reste tous de même des Milliers/Millions de paramètres à apprendre…



Visualisation des « Feature Maps »



Etude de cas LeNET 5



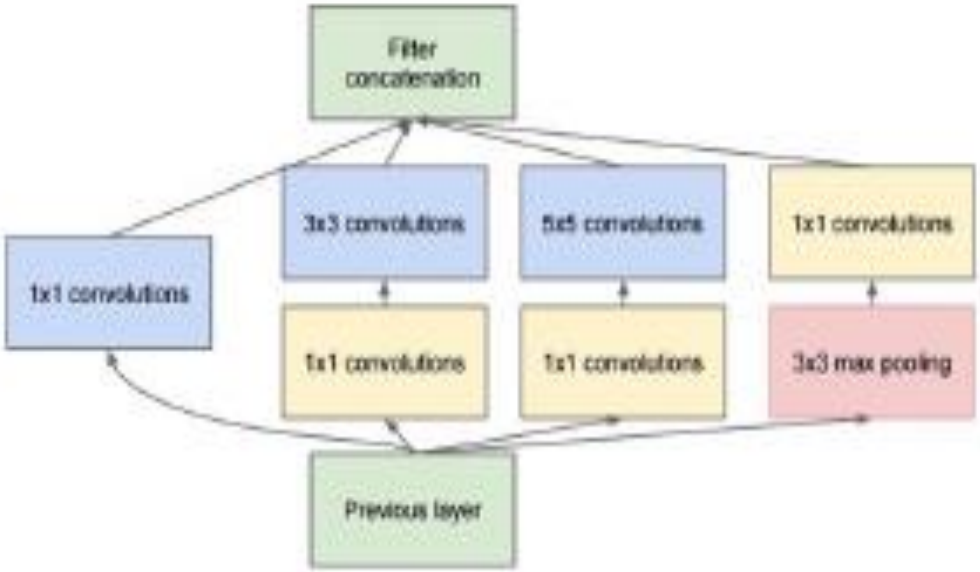
F6 Layer: Fully Connected layer

- 84 fully connected units.
- # Parameters: 84*(120+1)=10164

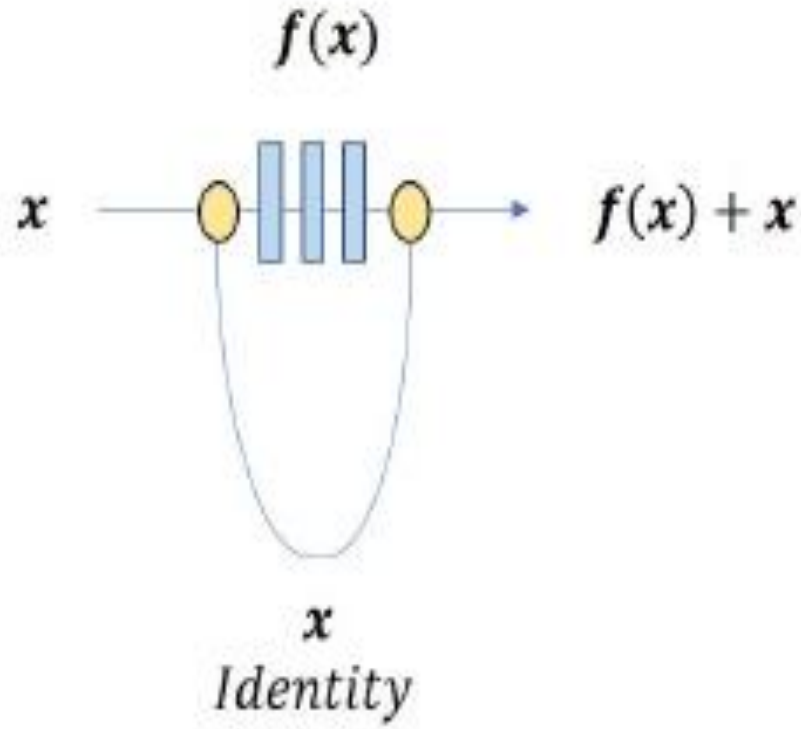
F7 Layer (output): Fully Connected layer

- 10 (# classes) fully connected units.
- # Parameters: 10*(84+1)=850

Apparition de différents types de module (blocks)



(A) Inception module



(B) Residual block

Apparition de différents types d'architectures

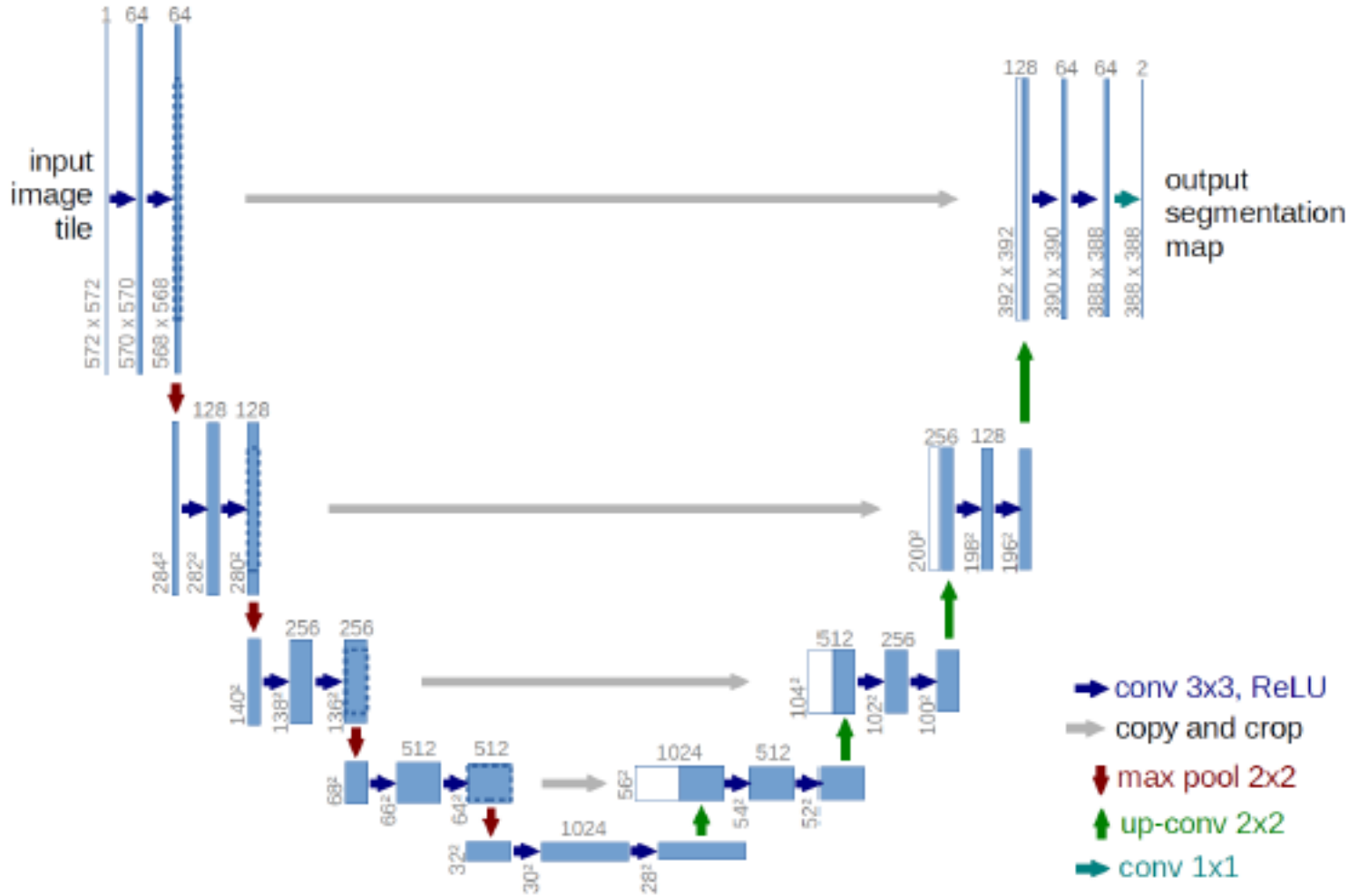
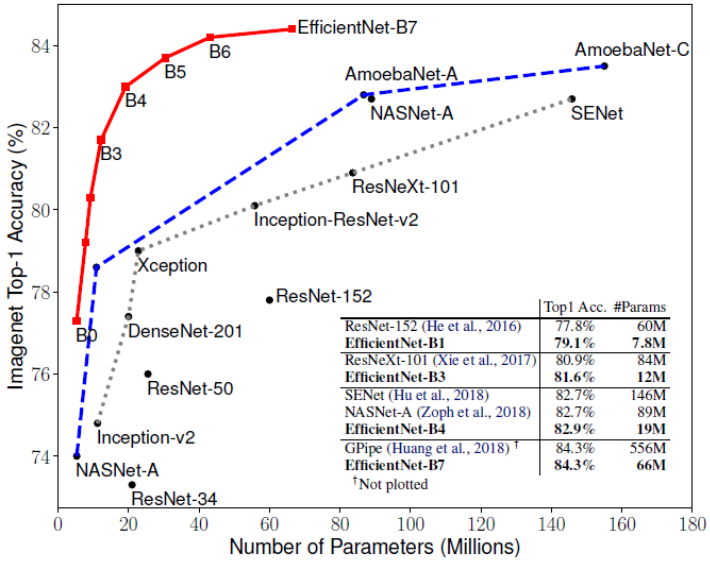


FIGURE 2.11: U-Net architecture. Figures from (Ronneberger et al., 2015).

Optimiser les architectures



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

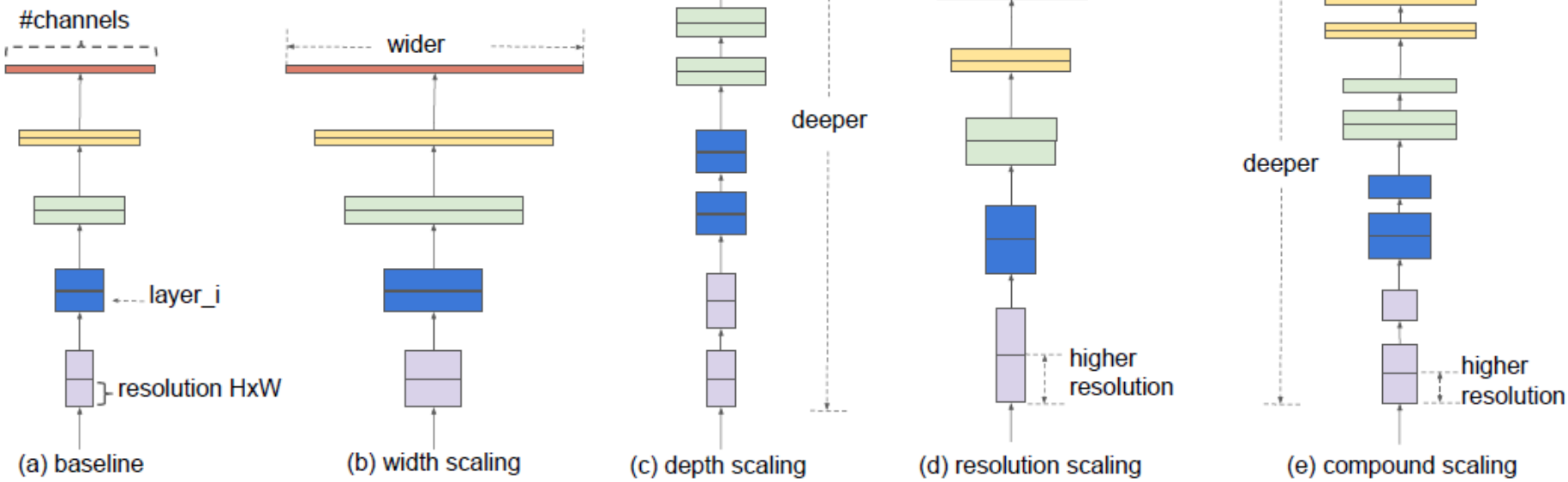
Figure 1. **Model Size vs. ImageNet Accuracy.**

$$\max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big)$$

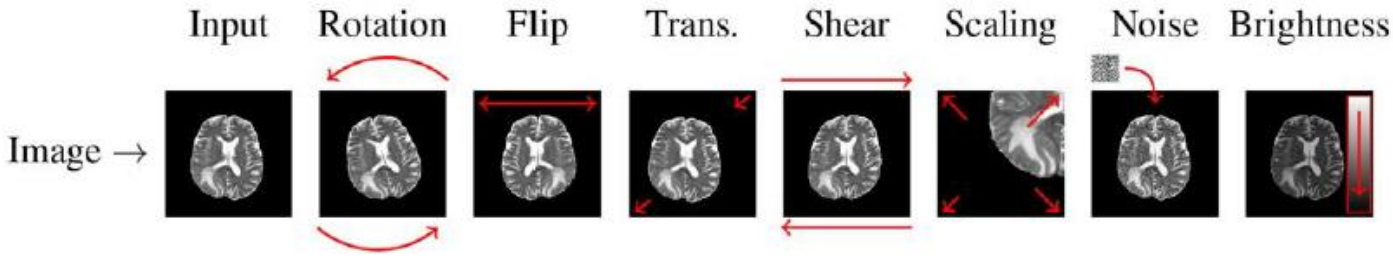
$$\text{Memory}(\mathcal{N}) \le \text{target\_memory}$$

$$\text{FLOPS}(\mathcal{N}) \le \text{target\_flops}$$



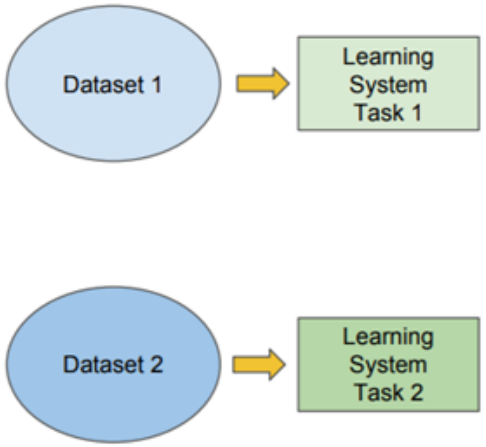
Data augmentation, Transfer Learning, fine-tuning

Data augmentation

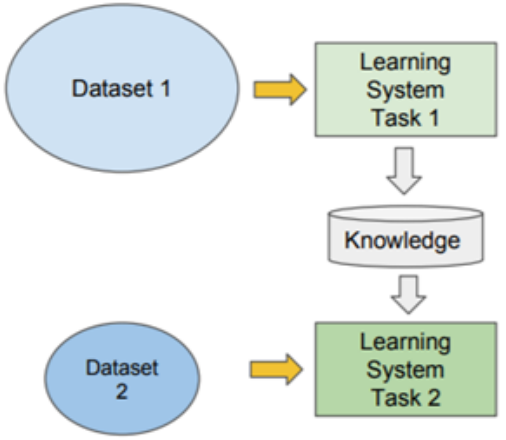


Apprentissage traditionnel VS transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks

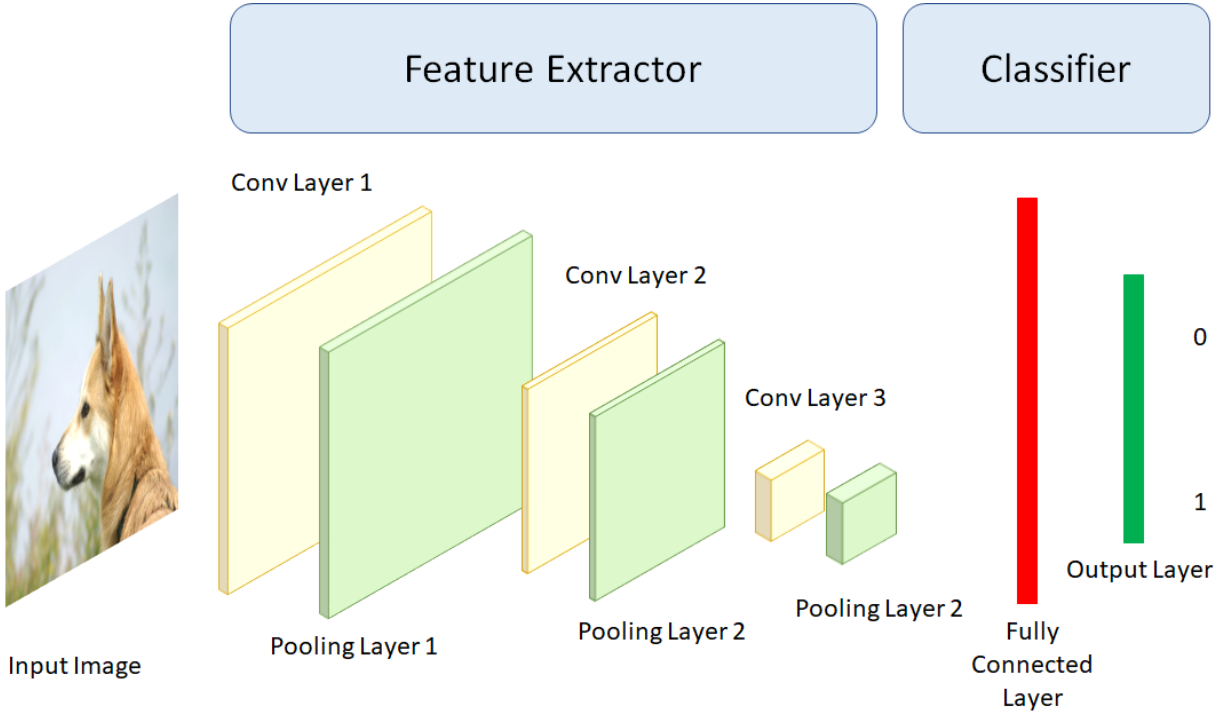


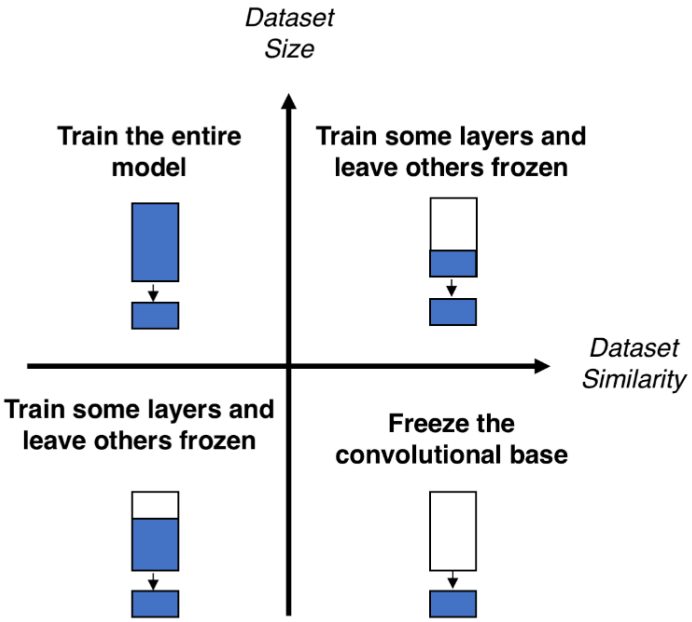
- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



Transfer Learning and fine-tuning

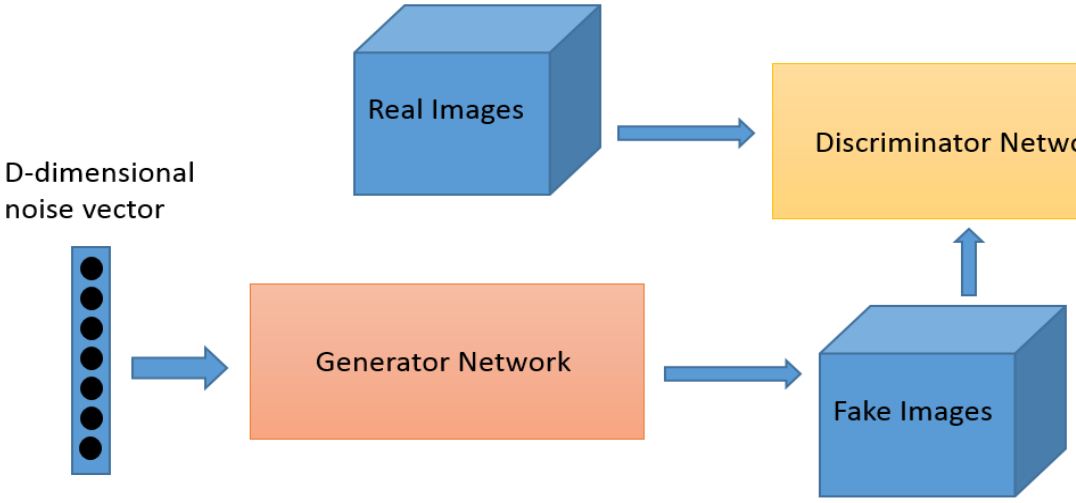
- **Que garder (freeze) ? Que ré-apprendre ?**
  - **Transfer Learning** → Récupération des poids d'un modèle appris et adaptation des poids des couches finales, souvent de la partie "classifier" (en rouge) uniquement
  - **Fine tuning** → Après le Transfer Learning, ré-apprentissage d'une partie des poids des couches de convolution (mais pas tous !) avec un *Learning rate* très faible
  - **Choix selon la « ressemblance »** entre dataset 1 (image.net) et dataset 2 (target)

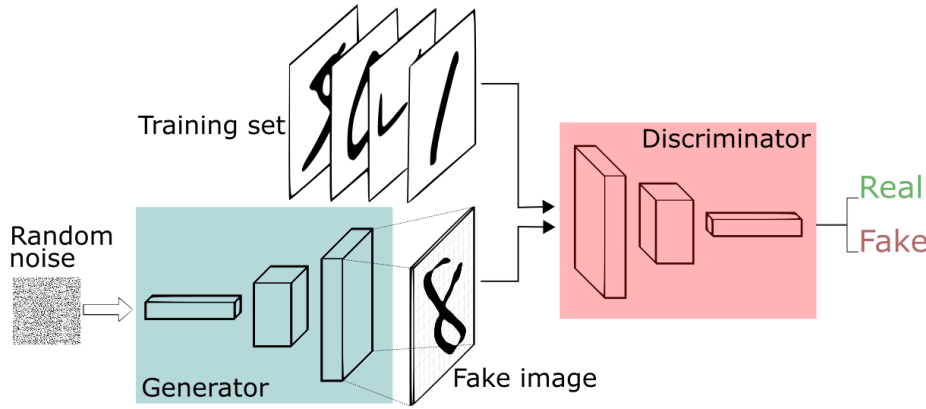


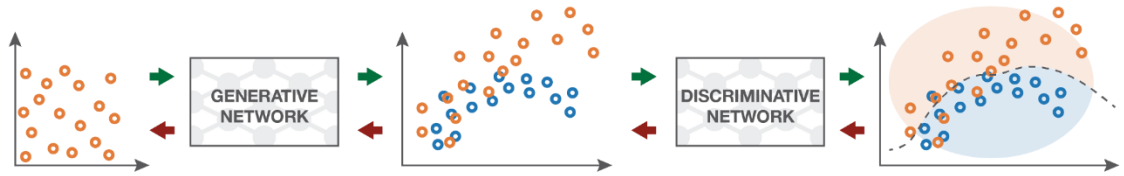


Self-supervised learning / GAN

- **Generative Adversarial Networks (GANs)**
  - Mise en compétition de 2 parties / architectures :
    - Generator → cherche aà maximiser l'erreur
    - Discriminator → cherche à minimiser l'erreur
    - Les 2 modèles doivent être appris (l'un après l'autre alternativement)
  - Génération de données d'apprentissage (ou de copies / données « stylisées »)

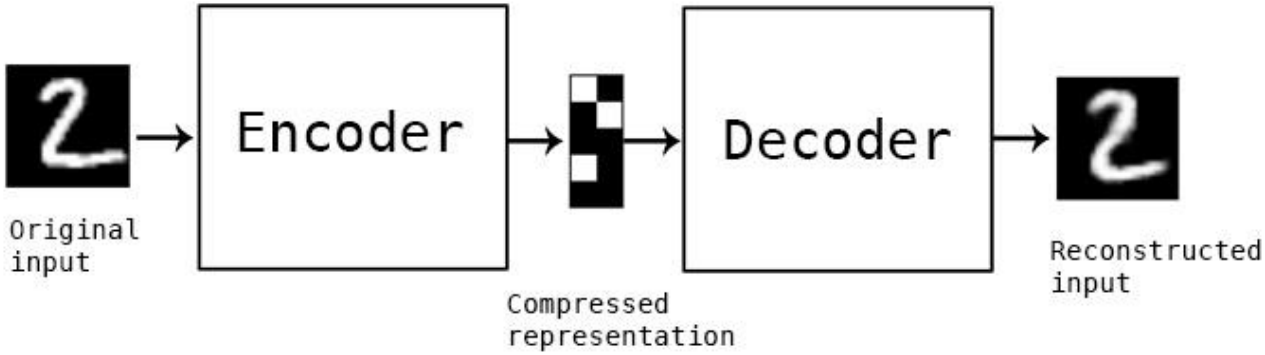




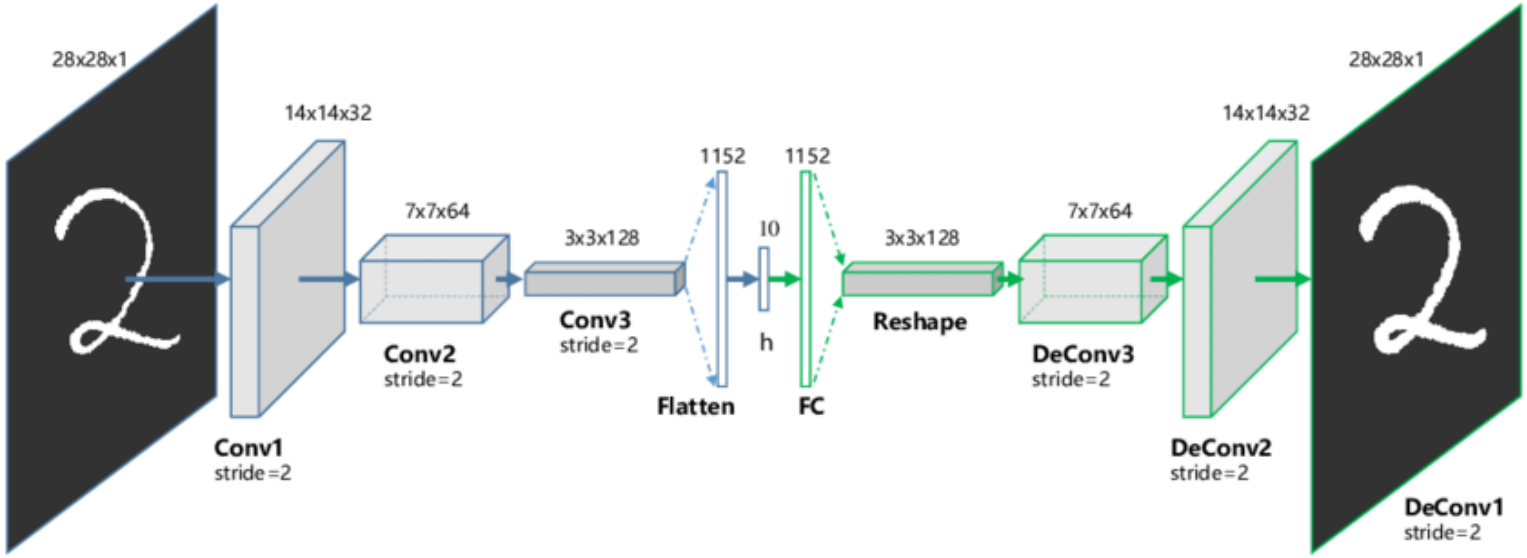


Self-supervised learning / encoder-decoder

- Les sorties (y) générées peuvent être comparés aux entrées (x)
  - Plus besoin de données annotées manuellement pour apprendre
  - Mise en compétition de 2 parties / architectures : encoder et decoder

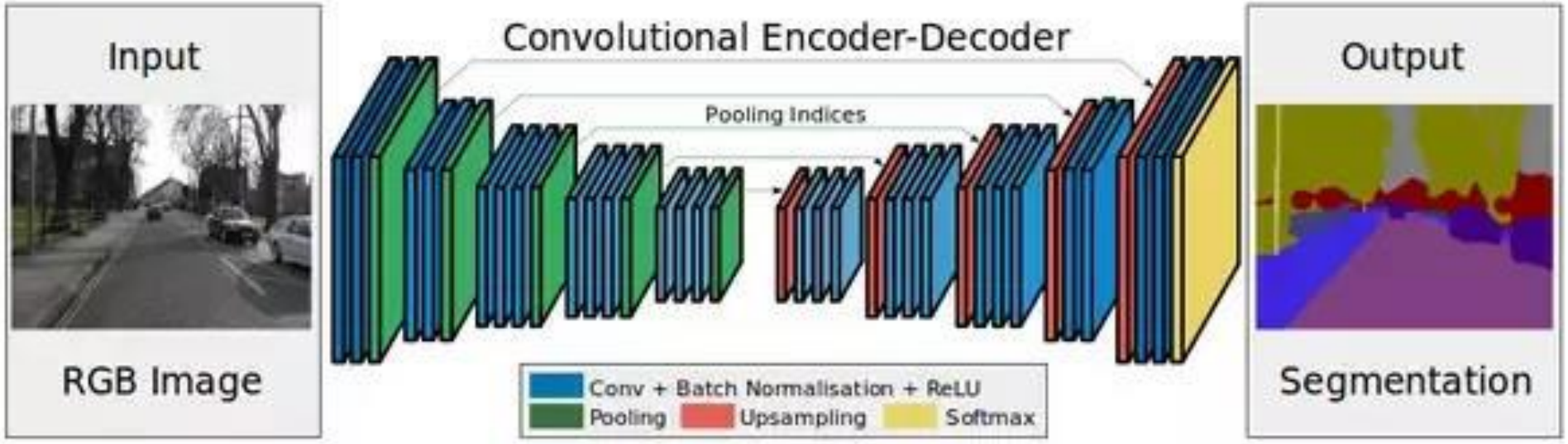


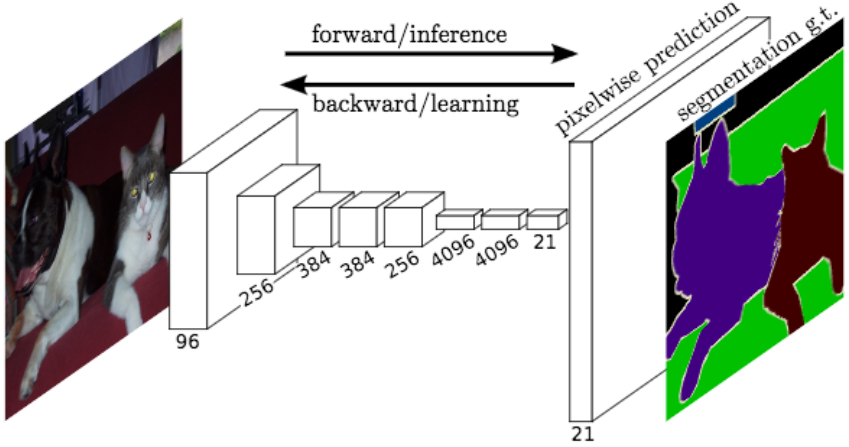
Loss = ||x - y||



Self-supervised learning / encoder-decoder

- Segmentation sémantique
  - Les sorties générées (^y) peuvent être comparés aux sorties attendues (y)



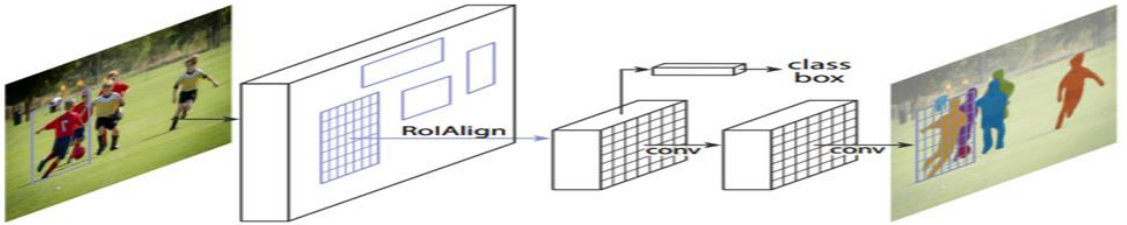


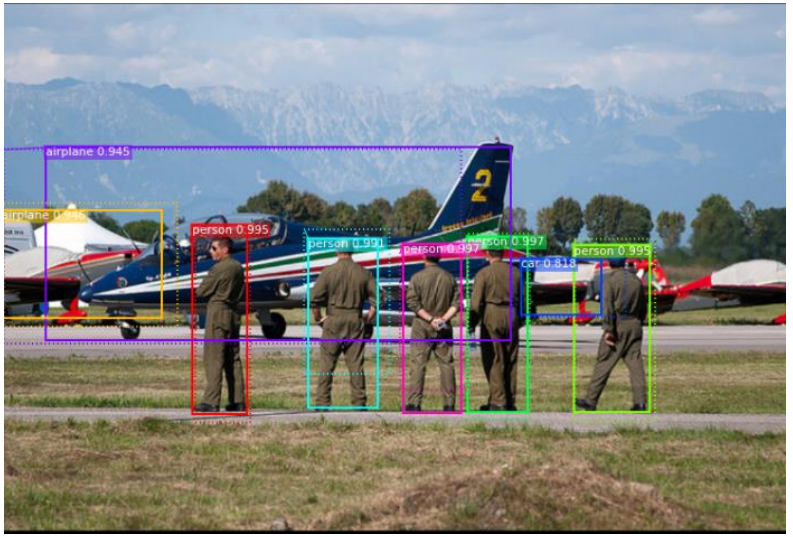
Blue : Pavement
Green : Tree
Purple : Cars
….

Instance segmentation / Tracking

- Différent de la segmentation sémantique → Détection de « bounding boxes » contenant des instances d'objets

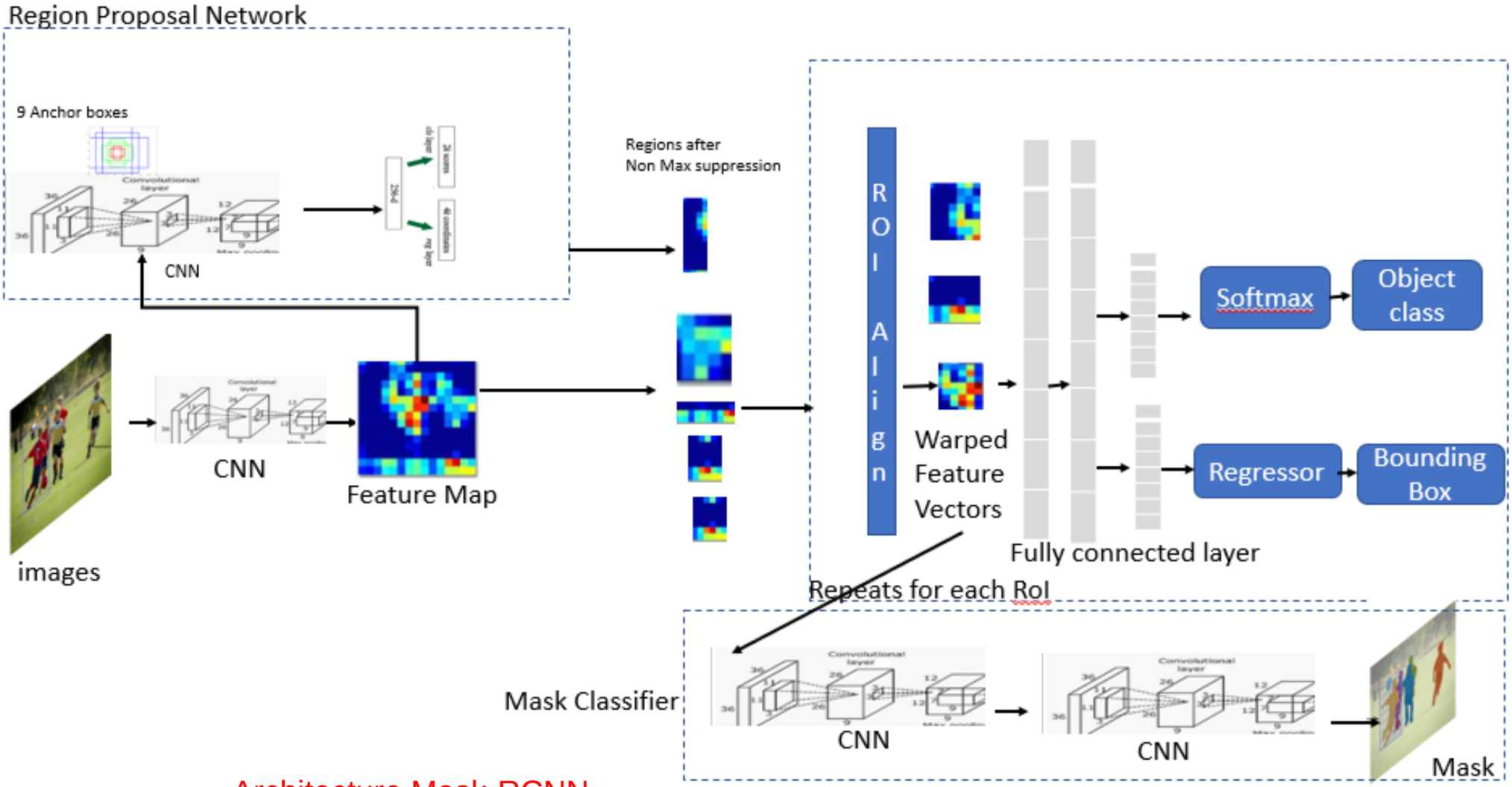
- Mask R-CNN, Faster R-CNN, …
  - CNN ResNet101 pour extraire des caractéristiques de l'image
  - Réseau de proposition de régions (RPN) → Génération de RoI / Mask
  - Binary Mask Classification (pour chaque classe d'objets)
  - Suppression des non-maxima





Instance segmentation / Tracking

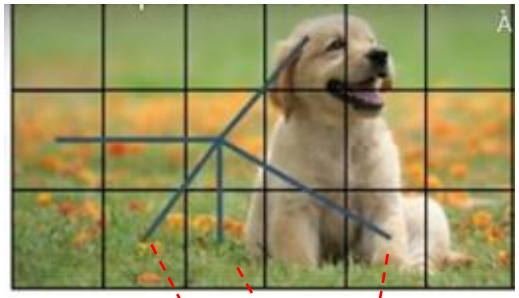
R-CNN, Fast R-CNN, Faster R-CNN, YOLO, ...

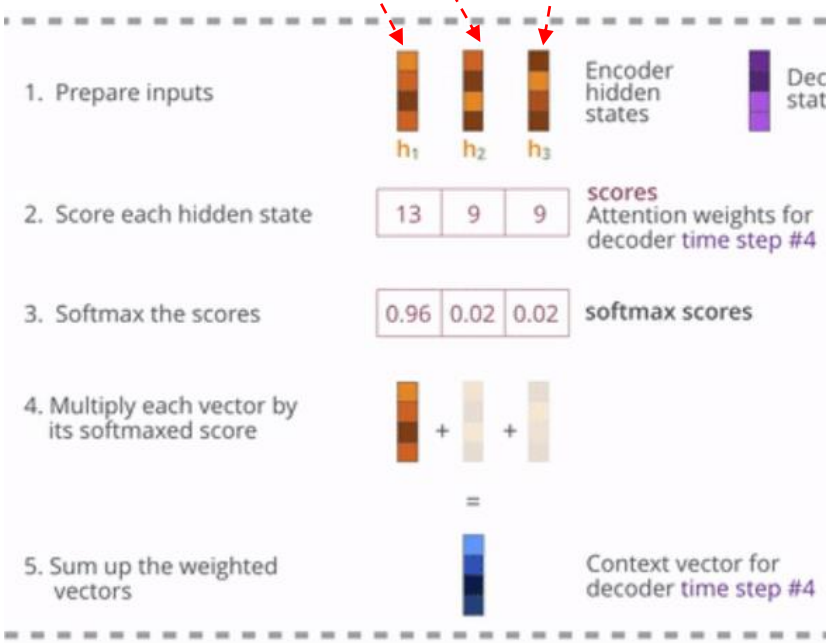


Architecture Mask-RCNN

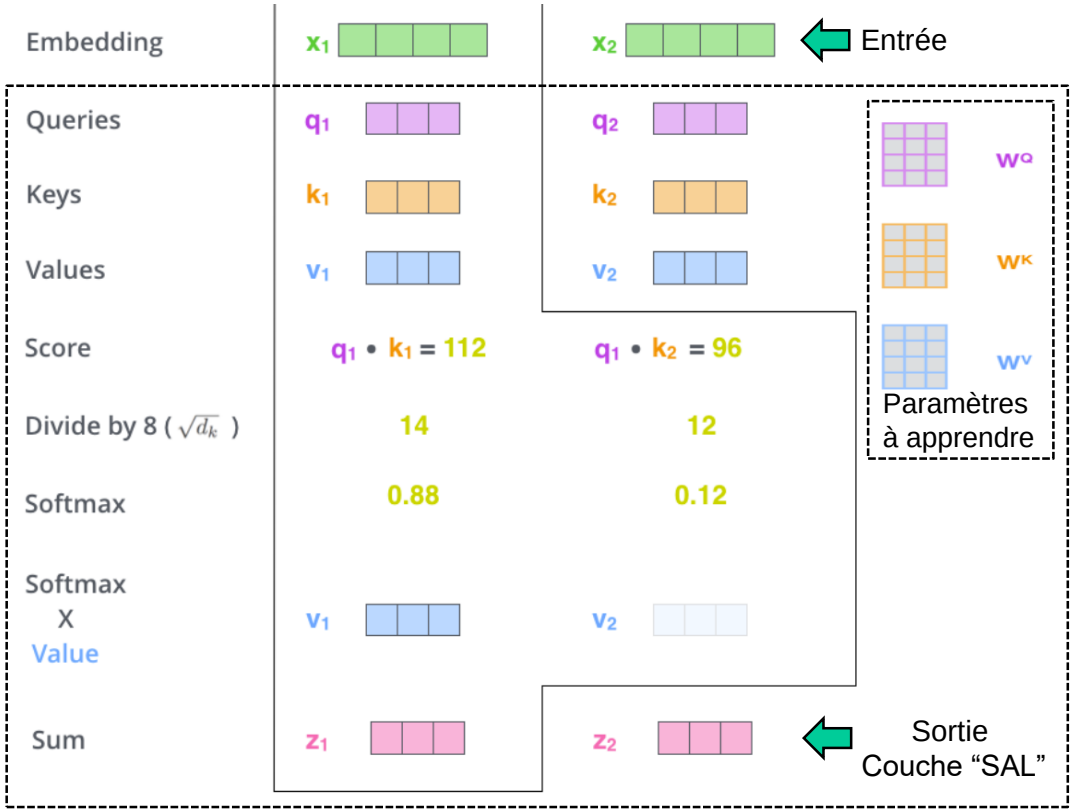
Des CNN aux Transformers

- Utilisation d'un mécanisme d'encodage → décodage
- Mécanismes **d'attention** → Pondération de l'importance relative des zones (patches) les unes par rapport aux autres localement





- **Mécanisme 'self-attention Layer'**
  - Avec triplet : Query, Key, Value



Transformers

Les details

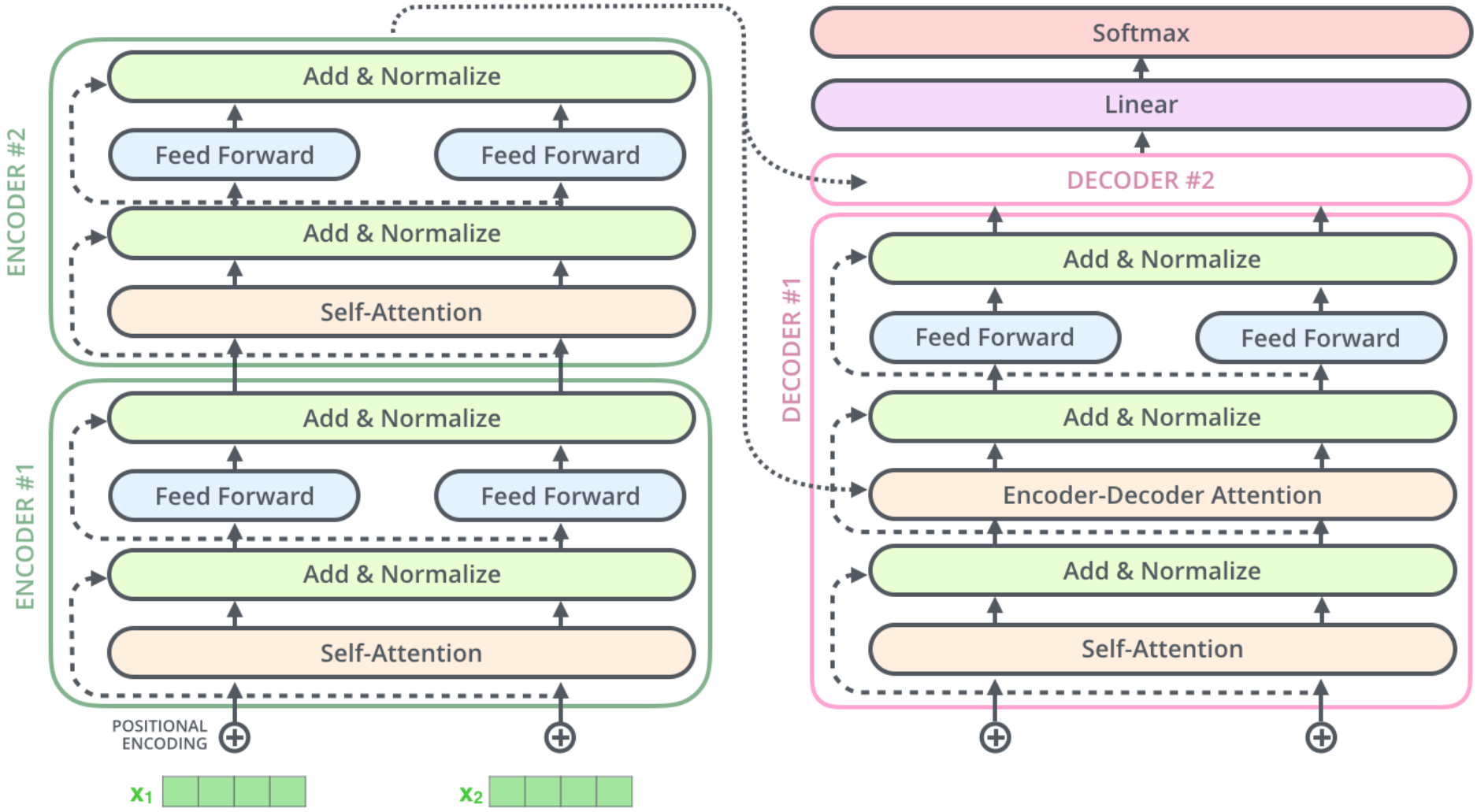


Image Patches

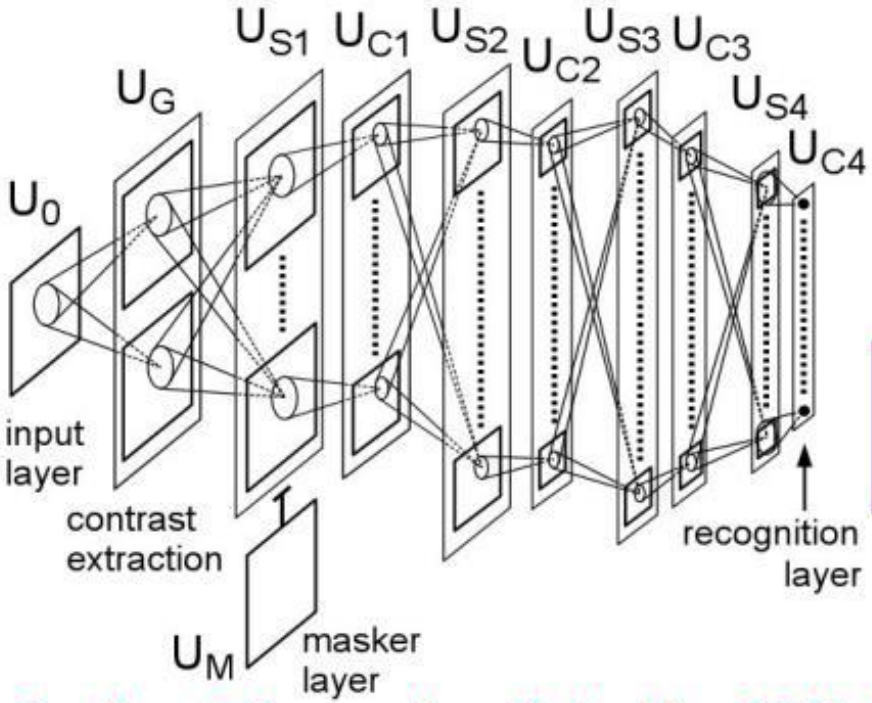
Deep learning Architecte…

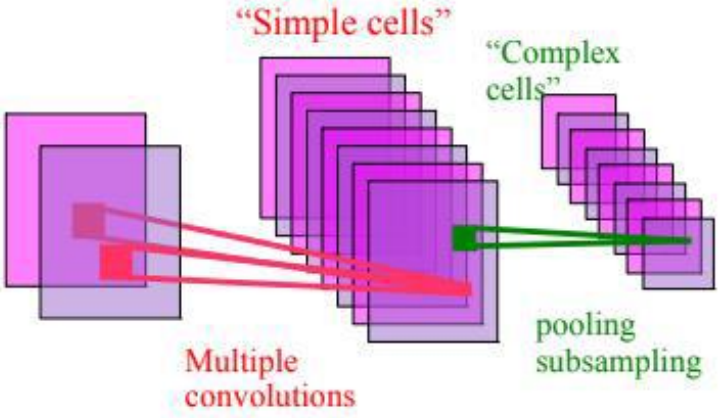
- De nombreuses librairies
  - Google → https://www.tensorflow.org/
  - FaceBook → https://pytorch.org/
  - Differentiable computer vision → https://kornia.github.io/

- Partage de modèles
  - **Model Zoo :** open source deep learning code and pretrained models → https://modelzoo.co/
  - Etc…

Epistémologie…

- [Hubel & Wiesel 1962]:
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.





Cognitron & Neocognitron [Fukushima 1974-1982]

Petit contrôle…

Approche Régions ? CNN ? Autres ?
Utilisation pour le TP image Tagging ?

- Expliquez les problématiques de la détection de régions ? Et les principales méthodes ?

- Pourquoi peut on dire que le résultat de la segmentation est un ensemble d'images binaires ?