Convolutional Networks for Image Classification

Mines Fontainebleau of doom

30 mars 2017

What the hell are convolutional neural networks ?

Une variante des réseaux de neurones adaptée au traitement d'image, en particulier pour la classification, mais aussi pour d'autres tâches (régression/segmentation ...)

Quelles différences ?

- Tient compte du fait que les entrées du système sont des images de taille fixe à n canaux.
- Moins de paramètres à apprendre qu'un réseau de neurones "naïf".

... et les GPU modernes sont très adaptés.

Bonne question

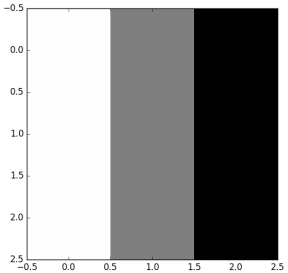
Les réseaux de neurones classiques sont mal adaptés aux entrées de grande dimension :

- Pour une image couleur de taille 256x256, *un seul* neurone doit apprendre 256x256x3 = 196 608 paramètres (+1 pour le biais mais c'est offert par la maison)
- Avec autant de paramètres, le risque d'overfitting est non négligeable ...

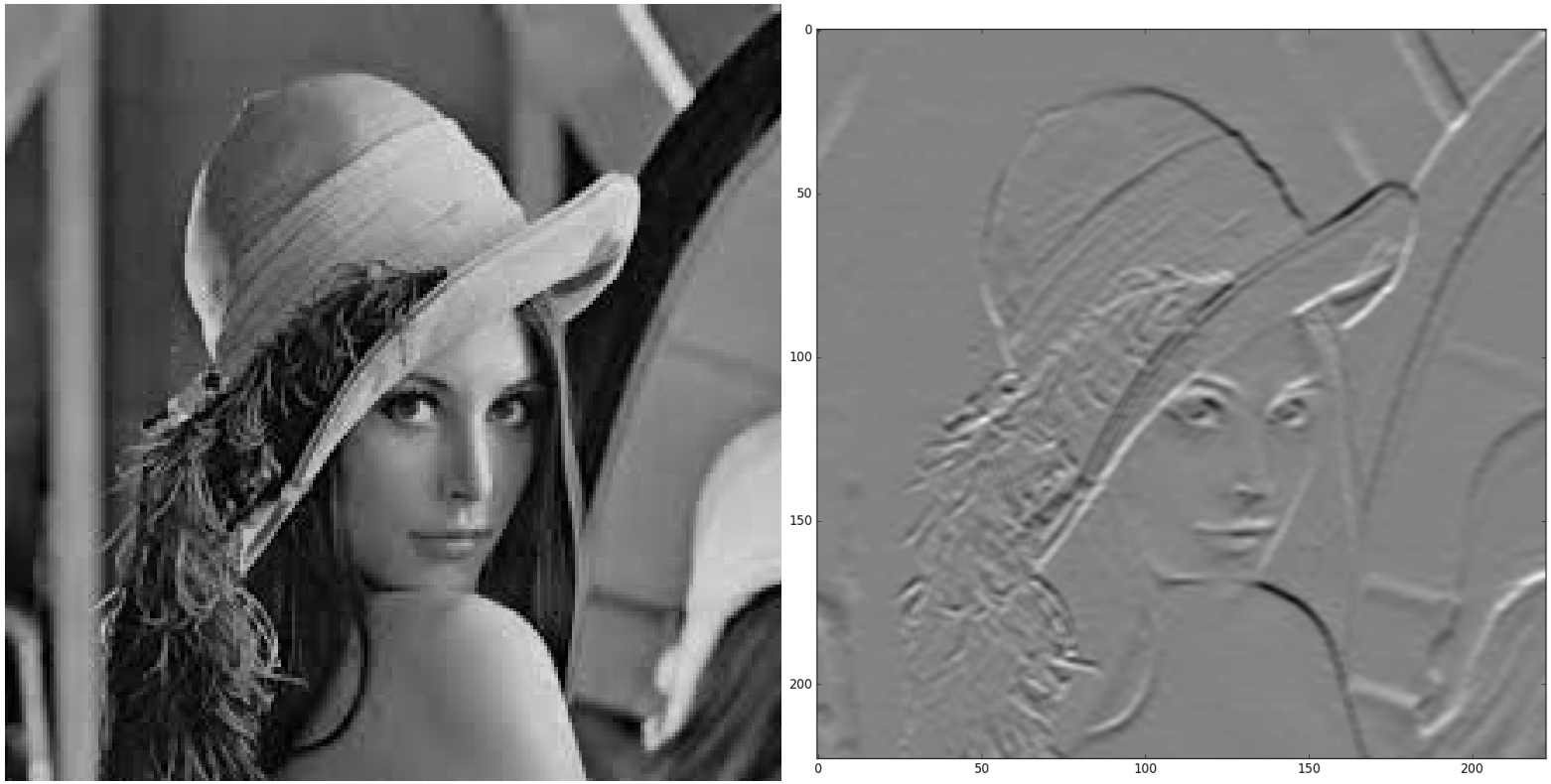
... on convertit tout en 16x16 grayscale ? 

- Idée clé : *parameter sharing*
- lié à l'invariance par translation

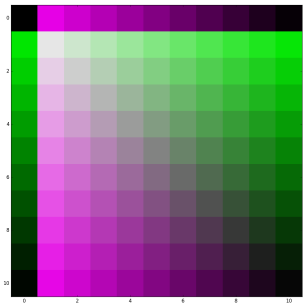
Opération de base : on fait glisser une petite fenêtre partout sur l'image et on calcule la corrélation.



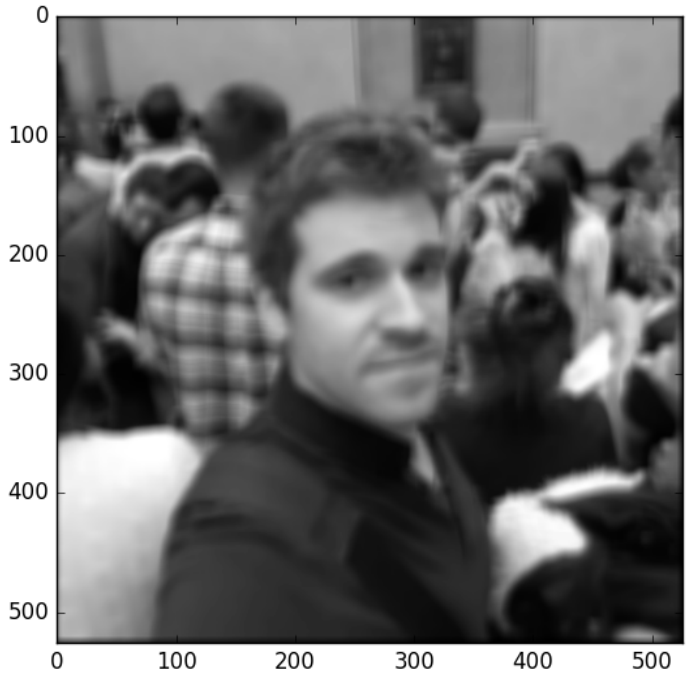
Résultat



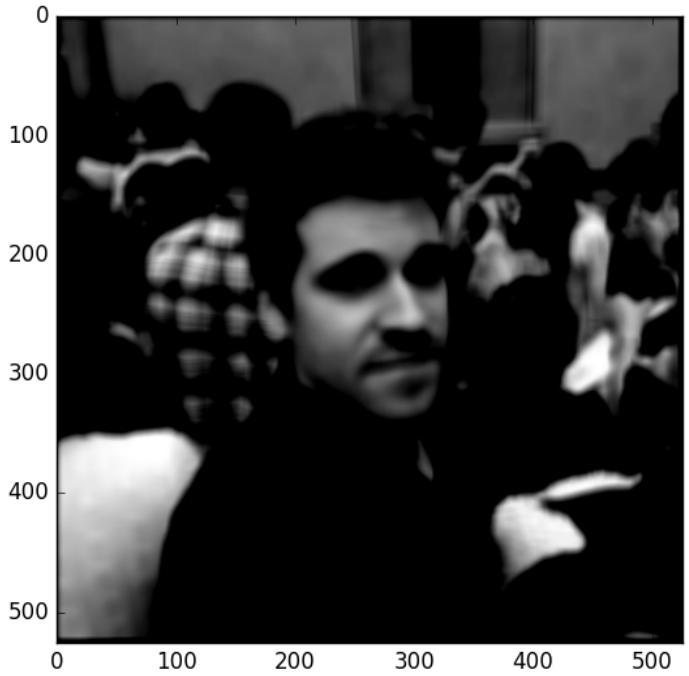
En couleurs



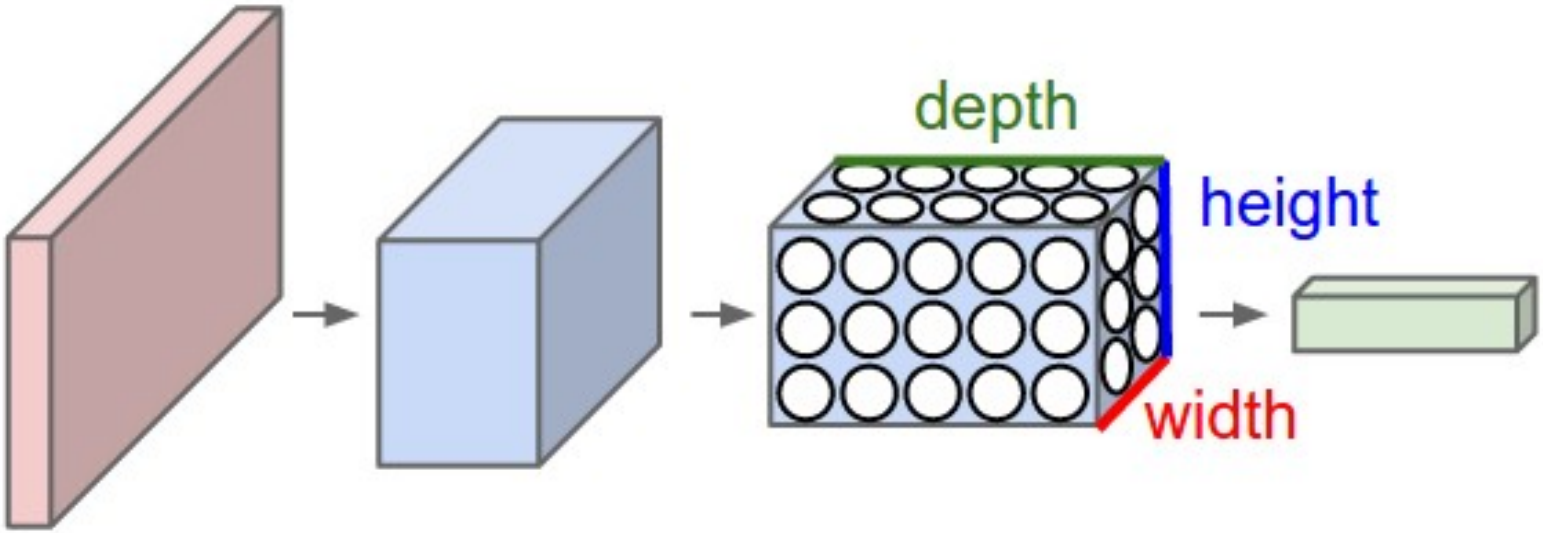
Résultat



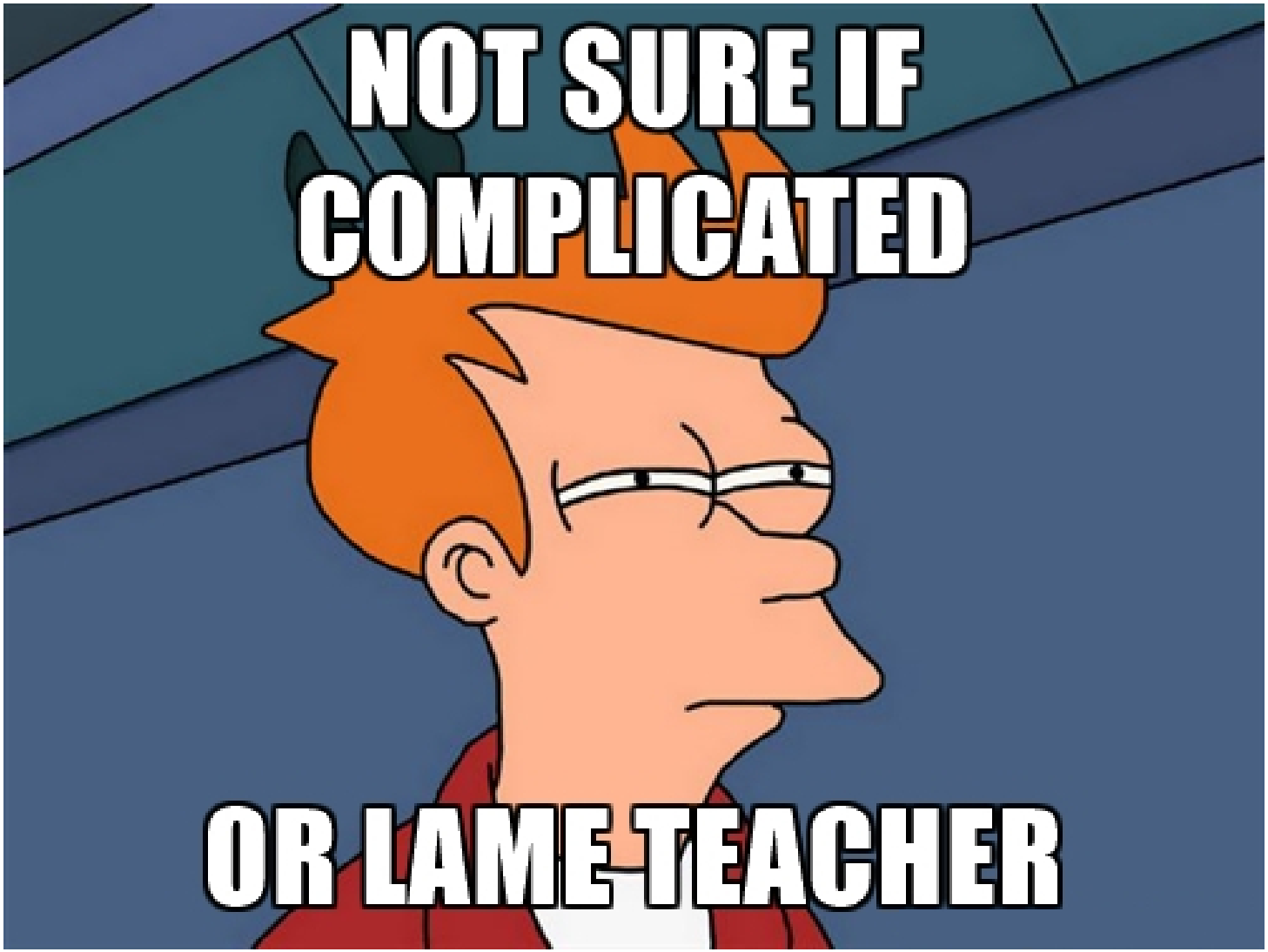
Seuillage



Merci Google Images



A ConvNet is made up of layers. Every layer transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.


NOT SURE IF
COMPLICATED
OR LAME TEACHER

"Pour faire simple"

On utilise principalement 3 types de couches (layers) :

- **Convolutional layer** Chaque neurone (i, j, d) est connecté à un voisinage du point (i,j) de l'image d'entrée. Les paramètres (poids) à apprendre dépendent de d mais pas de (i, j). (détails à suivre)
- **Pooling layer** Sous-échantillonne l'image d'entrée (par moyenne ou max). Typiquement, transforme une image (N,N,num_channels) en image (N/2,N/2,num_channels) en prenant le max sur des imagettes 2x2.
- **Fully connected layer** Comme dans les réseaux de neurones habituels, chaque neurone est connecté à toutes les entrées de la couche précédente. Utilisé dans les dernières couches du CNN.

Convolutional layer

Un filtre de convolution transforme une image X (I, J, n_ch) en une nouvelle représentation R (I', J') (un seul canal) via une opération du type :

$$R(i,j) = \sum_{c=1}^{n\_ch} \sum_{x=-1}^{1} \sum_{y=-1}^{1} X(i+x, j+y, c) W(x, y, c)$$

(ici pour un filtre W de taille (3, 3, n_ch))

Autrement dit : pour certaines coordonnées (i,j) de l'image originale, on calcule une combinaison linéaire de toutes les valeurs au voisinage de (i,j) (sur tous les canaux).

Attention : la dimension (I', J') de l'image sortie R n'est pas nécessairement la même que celle de l'image d'entrée (là, j'aurais dû parler de stride, zero-padding et receptive field).

Interprétation

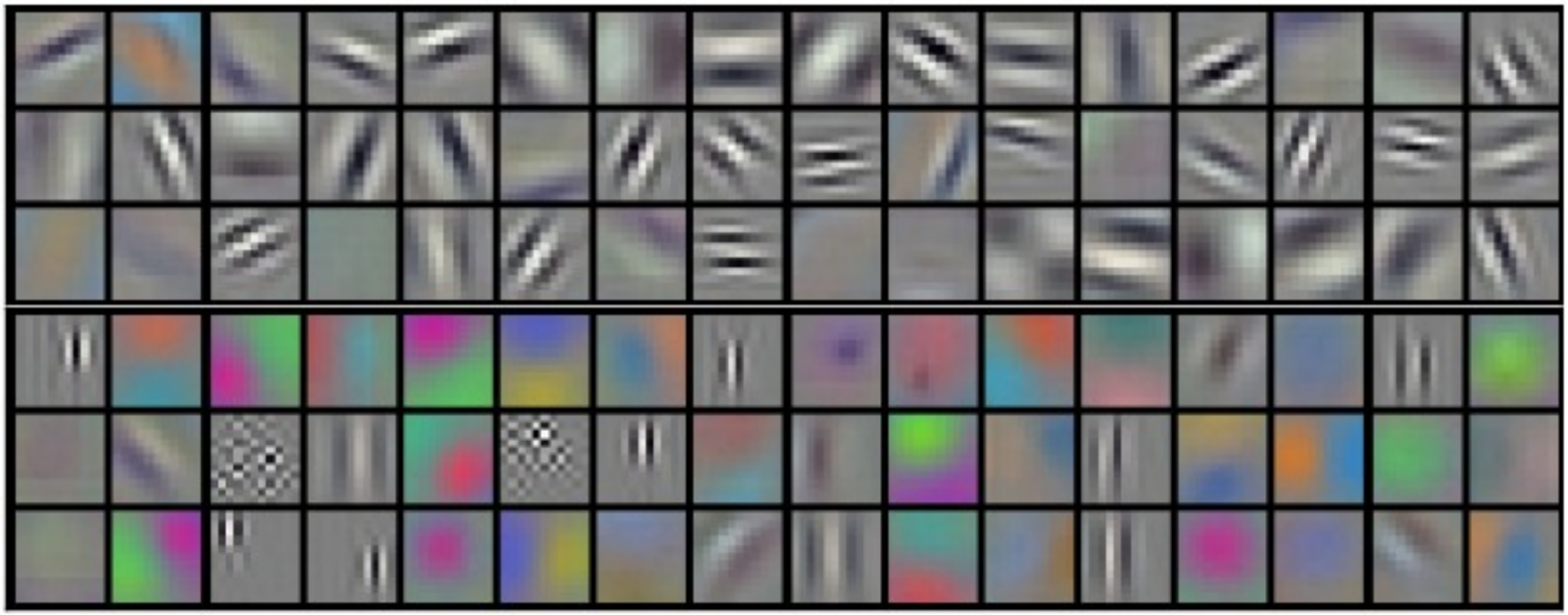


Figure : Examples of 11x11 filters learned by AlexNet

3 layer types

Exemple avec 10 classes et des images RGB 32x32 : un réseau simple serait [INPUT - CONV - RELU - POOL - FC] :

- INPUT [32x32x3] contient l'image originale
- CONV [32x32x12] apprend 12 filtres
- RELU [32x32x12] applique la fonction max(x,0) (d'autres non-linéarités sont possibles)
- POOL [16x16x12] sous-échantillonne la couche précedente selon les dimensions spatiales
- FC (fully-connected) : considère la couche précédente comme un vecteur 16x16x12, multiplie par une matrice (10,16x16x12) et calcule le softmax pour prédire les probabilités de chaque classe.

Ajoutez 2 œufs et 100g de sucre

La recette actuelle :

INPUT -> [[CONV -> RELU]*N -> POOL ?]*M -> [FC -> RELU]*K -> FC

avec :

- $N \leq 3$
- $M \geq 0$
- $K \geq 0$ (usually $K \leq 3$)

Trust me, I was a teacher

Ce dont je n'ai pas eu le temps de parler (mais dont je peux vous parler si vous avez le temps) :

- **Optimizer** Utiliser RMSProp (mais what the hell is RMSProp ?)
- **Dropout** Pour éviter le surapprentissage (valable aussi pour les neural networks classiques)
- **Sliding mask** Une manière élégante de transformer classification en régression, ou de vérifier que le réseau apprend bien ce que vous voulez.

Your turn

Pour en savoir plus :

- Cette présentation est fortement inspirée du cours de Stanford `http://cs231n.github.io`
- Voir aussi le cours de Geoff Hinton sur coursera.org
- Me demander directement (c'est mieux si vous avez du café)