

Apprentissage Profond

Introduction aux Réseaux de Neurones Convolutifs

A. Carlier

2024

Plan du cours

1 Introduction et motivation

2 Couches de convolution

3 Architectures convolutives

4 Visualisation

5 Transfert d'apprentissage

Problèmes classiques en traitement d'image



Classification

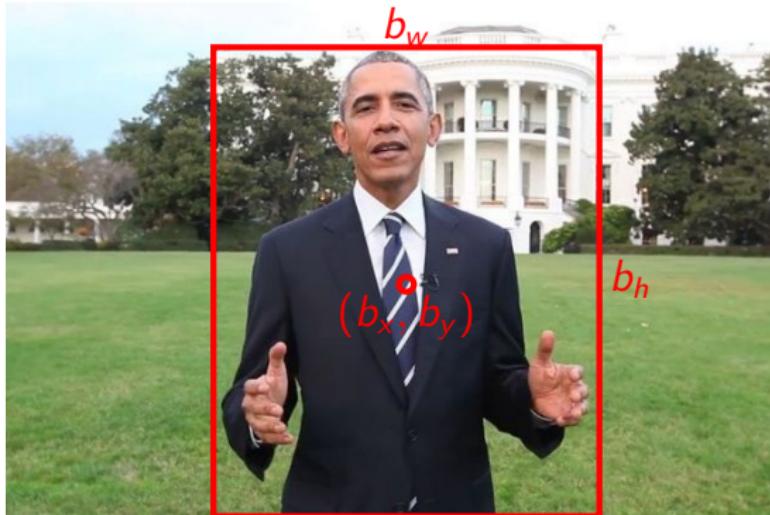
Ou encore : Annotation, *Labelling*



Classe principale : **Personne**

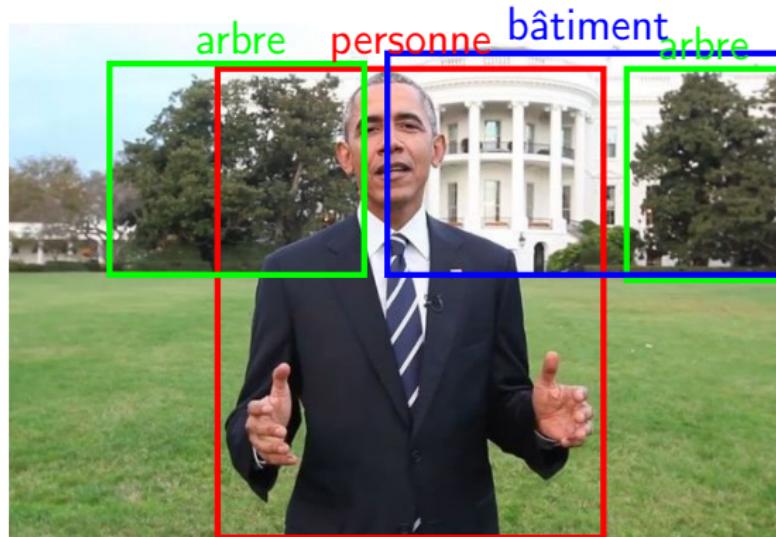
Classes secondaires : bâtiment, arbres, pelouse, ciel

Localisation



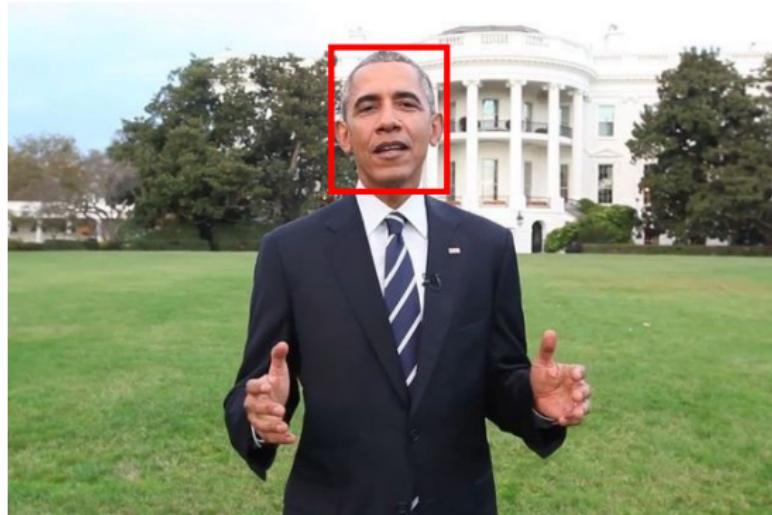
Objectif : délimiter la position de l'objet à l'aide d'une boîte englobante.

Détection

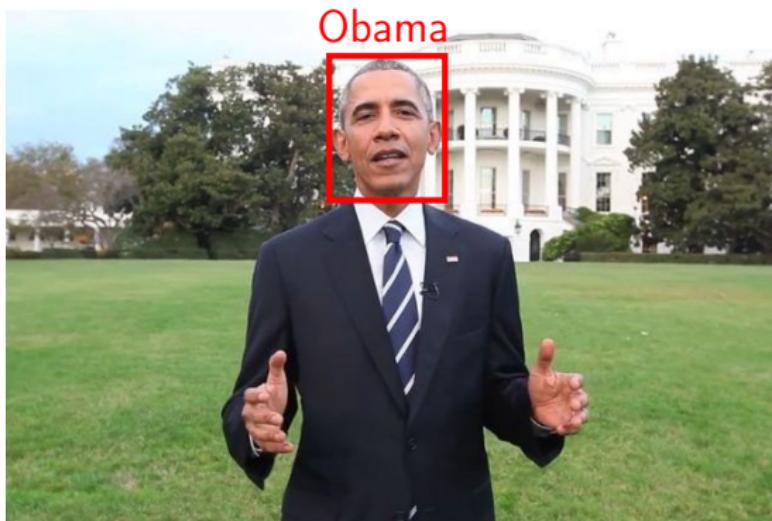


Objectif : délimiter la position d'objets multiples, avec éventuellement plusieurs instances, toujours à l'aide de boîtes englobantes.

Un sous-problème célèbre de la détection d'objet : la détection de visage...



... et la reconnaissance qui va de pair



Objectif : Étant donnés un visage détecté et une base de visages, reconnaître la personne.

Segmentation d'objet



Objectif : Classification binaire des **pixels** de l'image comme faisant partie de l'objet ou du fond.

Le Graal : la segmentation d'image

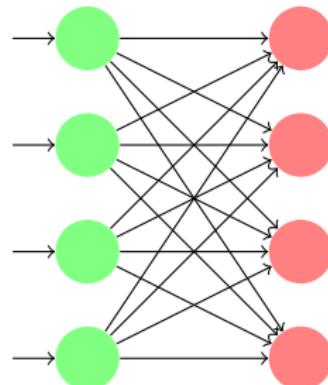
ou *Image Parsing*



Objectif : Classification n-aire des pixels de l'image.

Exemple de la classification d'images sur ImageNet

Images de taille $224 \times 224 \times 3$, 1000 classes :



$224 \times 224 \times 3$ pixels

$224 \times 224 \times 3$
neurones

1000
neurones

Un perceptron monocouche nécessite $224 \times 224 \times 3 \times 1000 + 1000$ paramètres, soit 150 millions de paramètres !

Plan du cours

1 Introduction et motivation

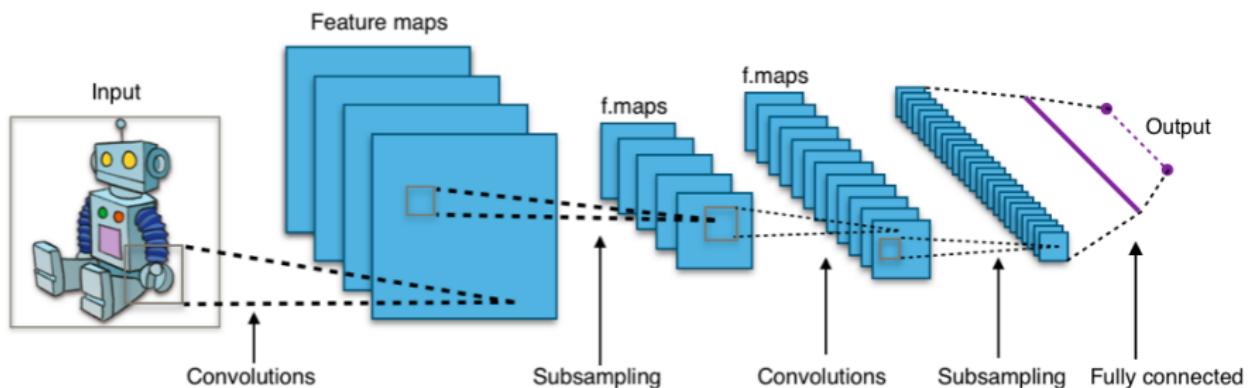
2 Couches de convolution

3 Architectures convolutives

4 Visualisation

5 Transfert d'apprentissage

Architecture classique d'un réseau de neurones convolutif



Convolution

$$\begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{green} & \text{green} \\ \hline \text{green} & \text{green} \\ \hline \end{array}$$

Convolution

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |

*

| | | |
|----|----|----|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

=

| | |
|---|--|
| 1 | |
| | |

Image

Filtre 3×3
 $f = 3$

Réponse

$$1 * -1 + 1 * -2 + 0 * -1 + 0 * 0 + 0 * 0 + 0 * 0 + 1 * 1 + 1 * 2 + 1 * 1 = 1$$

Convolution

$$\begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$$

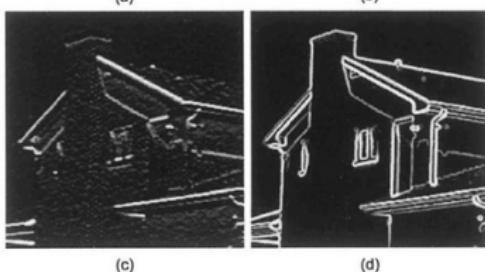
Convolution

$$\begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & \\ \hline \end{array}$$

Convolution

$$\begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & 0 \\ \hline \end{array}$$

La convolution en Vision par Ordinateur



| | | |
|----|---|----|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

G_x

| | | |
|----|----|----|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

G_y

- Les filtres (ou noyaux) de convolution sont utilisés depuis longtemps pour détecter des motifs dans les images, comme par exemple les contours (ici, filtres de Sobel)
- un pixel blanc indique une réponse élevée du filtre, c'est-à-dire un pixel situé sur le contour d'un objet, avec un fort gradient local.

Un point sur les dimensions des tenseurs

Étant donnés :

- une image I en niveaux de gris (un seul canal couleur) de dimension $w \times h$,
- un filtre K de dimension f ,

Alors la dimension de la réponse $I \circledast K$ de l'image I au filtre K est
 $(w - f + 1) \times (h - f + 1)$

Convolution (2D) de volumes

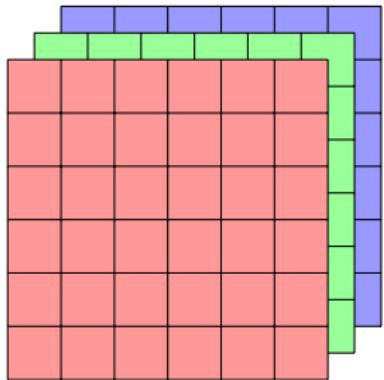
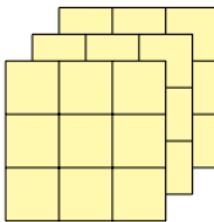
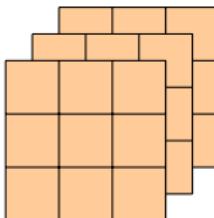


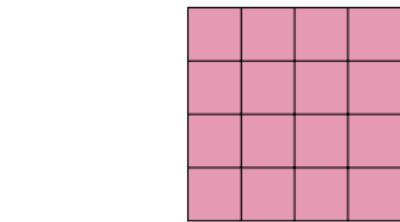
Image
 $6 \times 6 \times 3$



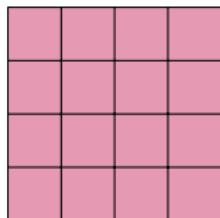
*



Filtres
 $3 \times 3 \times 3 \times 2$



=

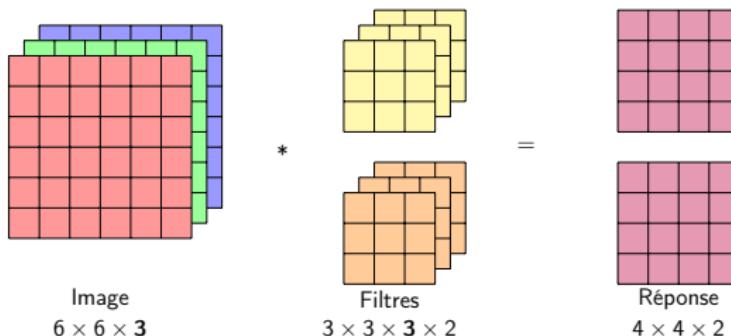


Réponse
 $4 \times 4 \times 2$

→ en Keras : `Conv2D(filters, kernel_size)`

Le nombre de canaux de l'image d'entrée et la profondeur des filtres de convolution sont nécessairement identiques.

Nombre de paramètres d'une couche de convolution

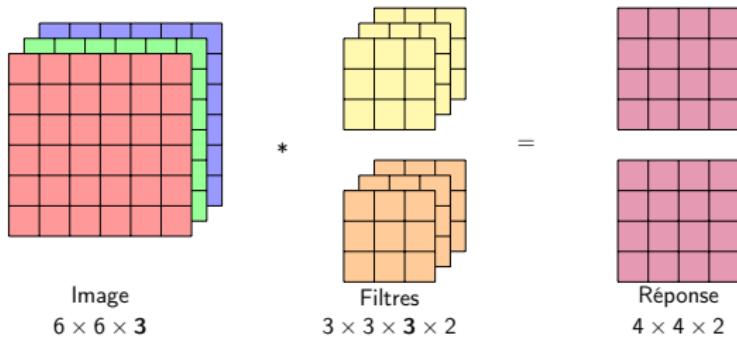


Il y a 2 types de paramètres dans une couche de convolution :

- Les **coefficients des filtres de convolution** : il y en a donc $f \times f \times \#\text{canaux} \times \#\text{filtres}$
- Les **biais** additionnés à la réponse des filtres de convolution, avant l'application de la fonction d'activation. Il y a exactement un biais par filtre de convolution.

Ainsi dans l'exemple ci-dessus, il y a $3 \times 3 \times 3 \times 2 + 2 = 56$ paramètres.

Nombre d'opérations d'une couche de convolution



Il est intéressant de compter le nombre d'opérations d'un réseau de neurones pour caractériser sa complexité, et les ressources nécessaires à son exécution. On s'intéresse principalement aux additions et aux multiplications.

Ici, chaque élément de la réponse nécessite :

- $(f \times f \times \#\text{canaux})$ multiplications
- $(f \times f \times \#\text{canaux} - 1)$ additions entre les valeurs multipliées
- 1 addition supplémentaire pour l'ajout du biais

Ainsi dans l'exemple ci-dessus, il y a $(4 \times 4 \times 2) \times (3 \times 3 \times 3 \times 2) = 1728$ opérations.

Padding

$$\begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & 0 \\ \hline \end{array}$$

Image Filtre Réponse

Padding

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*

| | | |
|----|----|----|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

=

| | | | |
|----|----|----|----|
| 0 | 0 | 1 | 2 |
| 0 | 1 | 1 | -1 |
| 2 | 1 | 0 | 0 |
| -3 | -4 | -3 | -1 |

Image
 $p = 1$

Filtre

Réponse

- Ajout de zéros (*zero-padding*) aux bords.
- Permet par exemple d'obtenir une réponse de même dimension que l'image d'entrée (*convolution same*), ce qui facilite le chaînage des couches de convolution dans un réseau de neurones.

Impact sur la dimension des tenseurs

Étant donnés :

- une image I en niveaux de gris (un seul canal couleur) de dimension $w \times h$,
- un filtre K de dimension f , avec un *padding* p

Alors la dimension de la réponse $I \circledast K$ de l'image I au filtre K est
 $(w + 2p - f + 1) \times (h + 2p - f + 1)$

Stride

$$\begin{array}{|c|c|c|c|c|c|} \hline 3 & 1 & 3 & 5 & 3 & 3 \\ \hline 2 & 2 & 8 & 8 & 3 & 9 \\ \hline 3 & 4 & 7 & 7 & 2 & 7 \\ \hline 5 & 3 & 6 & 8 & 4 & 7 \\ \hline 3 & 8 & 8 & 5 & 7 & 4 \\ \hline 7 & 9 & 6 & 4 & 6 & 9 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 10 & 13 & 7 & 4 \\ \hline 3 & -3 & -1 & 0 \\ \hline 9 & 4 & 2 & 5 \\ \hline 14 & 2 & -6 & 2 \\ \hline \end{array}$$

Image
 $p = 0$

Filtre
 $f = 3$

$stride = 1$

Réponse

Stride

The diagram illustrates a convolution operation with a stride of 2. It shows an input image of size 6x6 and a filter of size 3x3. The stride of 2 means that the filter slides over the input at a distance of 2 units. The result is a response map of size 3x3.

Image
 $p = 0$

Filtre
 $f = 3$

stride = 2

Réponse

| | | | | | | | | |
|---|---|---|---|---|---|--|--|--|
| 3 | 1 | 3 | 5 | 3 | 3 | | | |
| 2 | 2 | 8 | 8 | 3 | 9 | | | |
| 3 | 4 | 7 | 7 | 2 | 7 | | | |
| 5 | 3 | 6 | 8 | 4 | 7 | | | |
| 3 | 8 | 8 | 5 | 7 | 4 | | | |
| 7 | 9 | 6 | 4 | 6 | 9 | | | |

* =

| | | | | | |
|----|----|----|--|--|--|
| -1 | -2 | -1 | | | |
| 0 | 0 | 0 | | | |
| 1 | 2 | 1 | | | |

=

| | | | | |
|----|---|--|--|--|
| 10 | 7 | | | |
| 9 | 2 | | | |

- Permet de **réduire la dimension** des tenseurs, en limitant la perte d'information du fait qu'un même coefficient influence plusieurs éléments de la réponse au filtre de convolution.

Impact sur la dimension des tenseurs

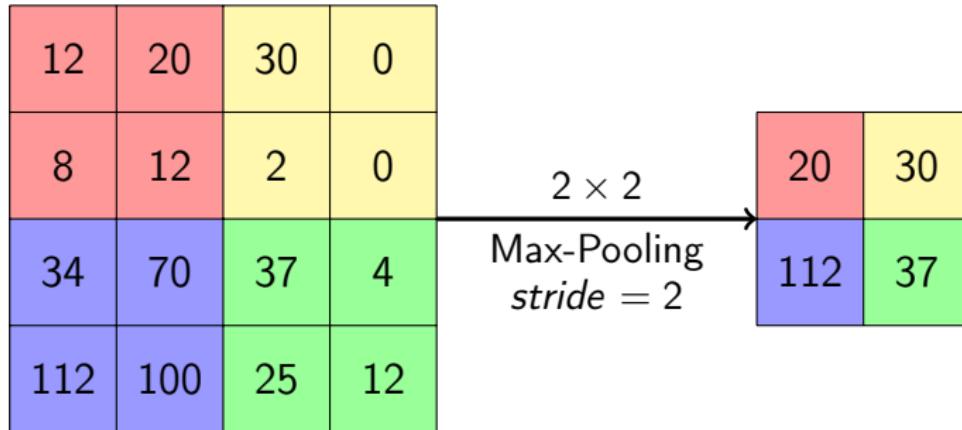
Étant donnés :

- une image I en niveaux de gris (un seul canal couleur) de dimension $w \times h$,
- un filtre K de dimension f , avec un *padding* p et un *stride* s

Alors la dimension de la réponse $I \circledast K$ de l'image I au filtre K est :

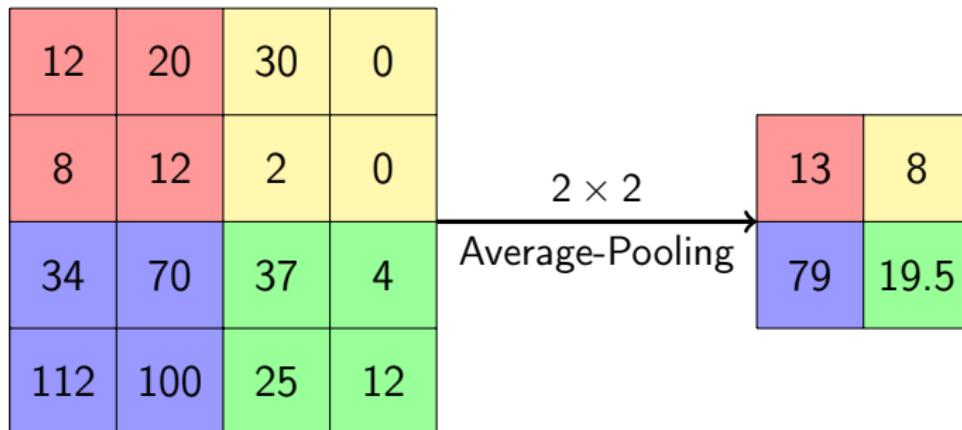
$$\left\lfloor \frac{w + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h + 2p - f}{s} + 1 \right\rfloor \quad (1)$$

Couche de Pooling



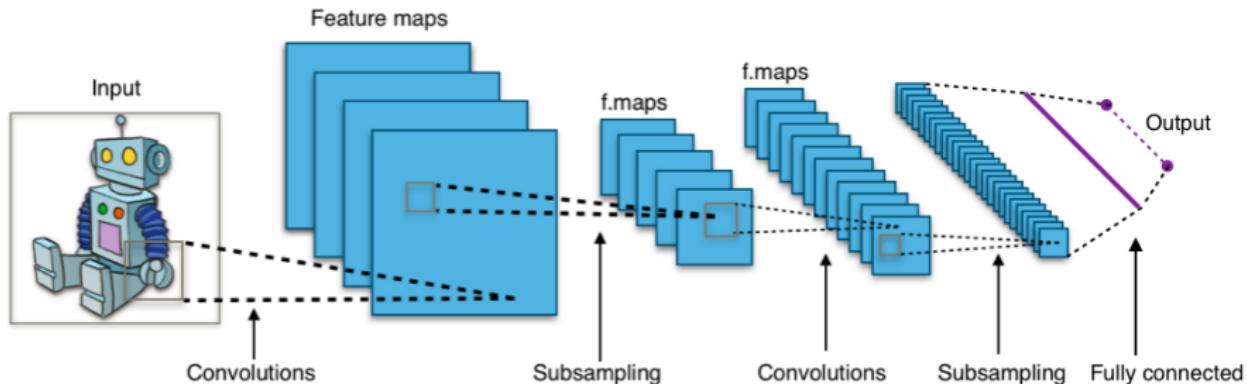
- Permet de **réduire la dimension** des tenseurs, pour contrebalancer la multiplication des réponses aux filtres de convolution.
- **Préserve les hautes réponses** des filtre de convolution.
- Introduit une **invariance à la translation**.
- **Pas de paramètres à apprendre** !

Couche de Pooling



- Alternativement, on peut aussi moyenner les valeurs plutôt qu'en conserver le maximum (on parle d'**Average-Pooling**).
- Les architectures classiques privilégient cependant le **Max-Pooling**.

Architecture classique d'un réseau de neurones convolutif



On trouve 3 types de couches dans un réseau de neurones convolutif typique :

- Des couches de **convolution**, combinées à des couches de **pooling** dans les premières couches du réseau.
- Des couches **complètement connectées** dans les dernières couches du réseau.

Partage de paramètres :

- Une couche de convolution est équivalente à une couche complètement connectée dans laquelle certains poids synaptiques sont partagés, et dont la majorité est à 0.
- Un même exemple de la base d'apprentissage permet de modifier ces poids (les coefficients de convolution) à de multiples reprises.

Ceci résulte en un nombre de paramètres bien plus faible pour un réseau convolutif que pour un réseau complètement connecté.

Plan du cours

1 Introduction et motivation

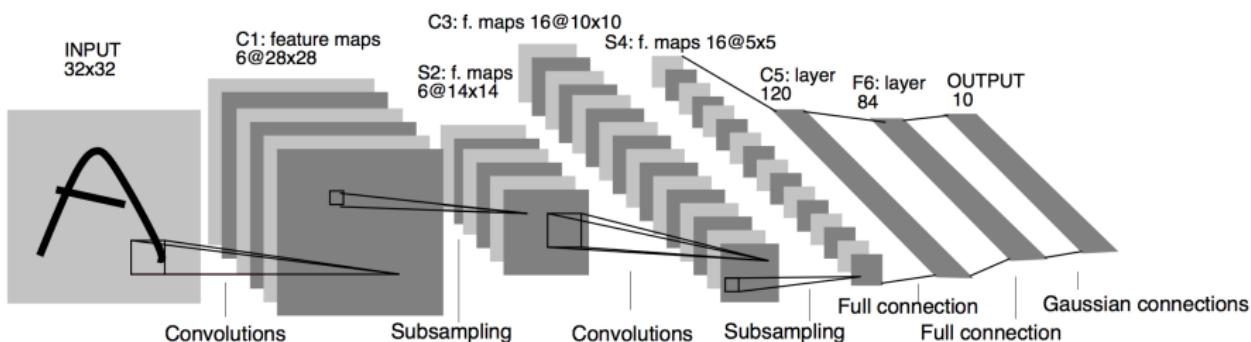
2 Couches de convolution

3 Architectures convolutives

4 Visualisation

5 Transfert d'apprentissage

Un pionnier : LeNet (1998)

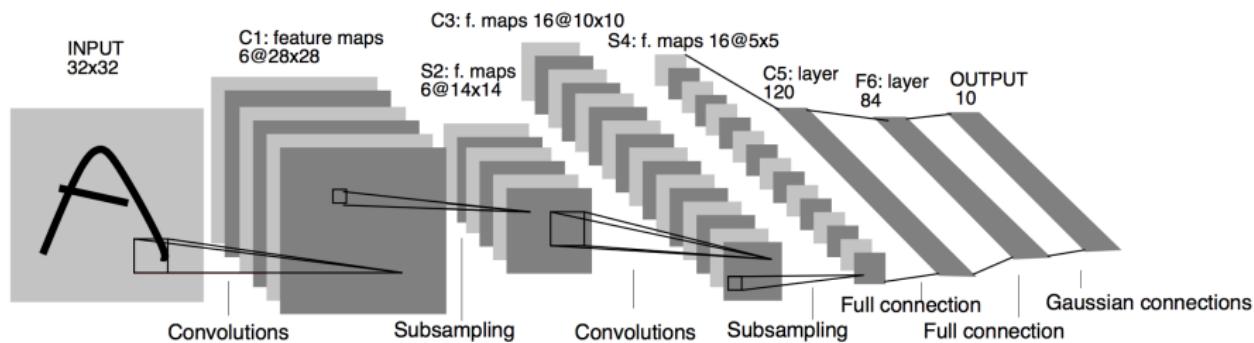


- $\simeq 60k$ paramètres
- 2 à 3 jours d'entraînement pour 20 epochs sur MNIST (en 1998!).
- Fonctions d'activation sigmoïdes en majorité.

Visualisation : https://adamharley.com/nn_vis/cnn/2d.html

[LeCun et al.] Gradient-based learning applied to document recognition.

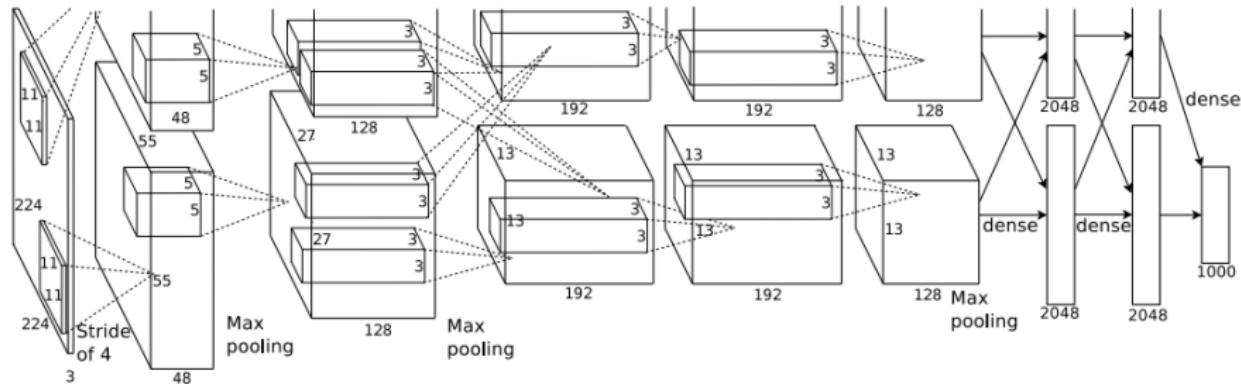
Question



Nombre de paramètres ? Nombre d'opérations à l'inférence ?

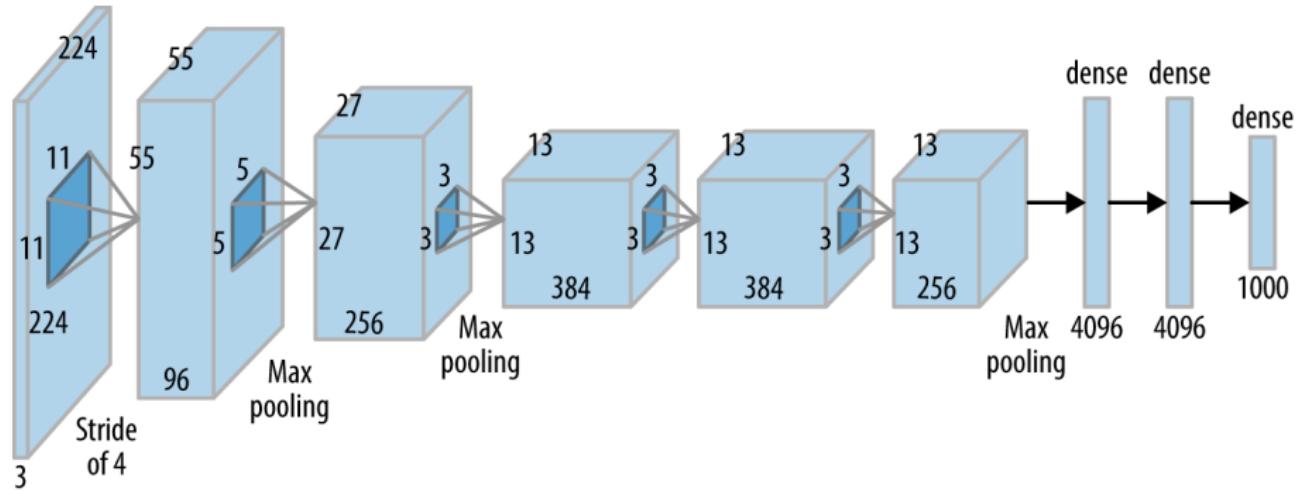
Question subsidiaire : poids en mémoire des calculs intermédiaires ?

La bascule : AlexNet (2012)



- $\simeq 60M$ paramètres, 8 couches
- l'architecture est conçue pour être implémentée sur 2 GPUs.
- introduit l'usage de la fonction `ReLU` comme un standard pour l'entraînement de réseaux profonds.

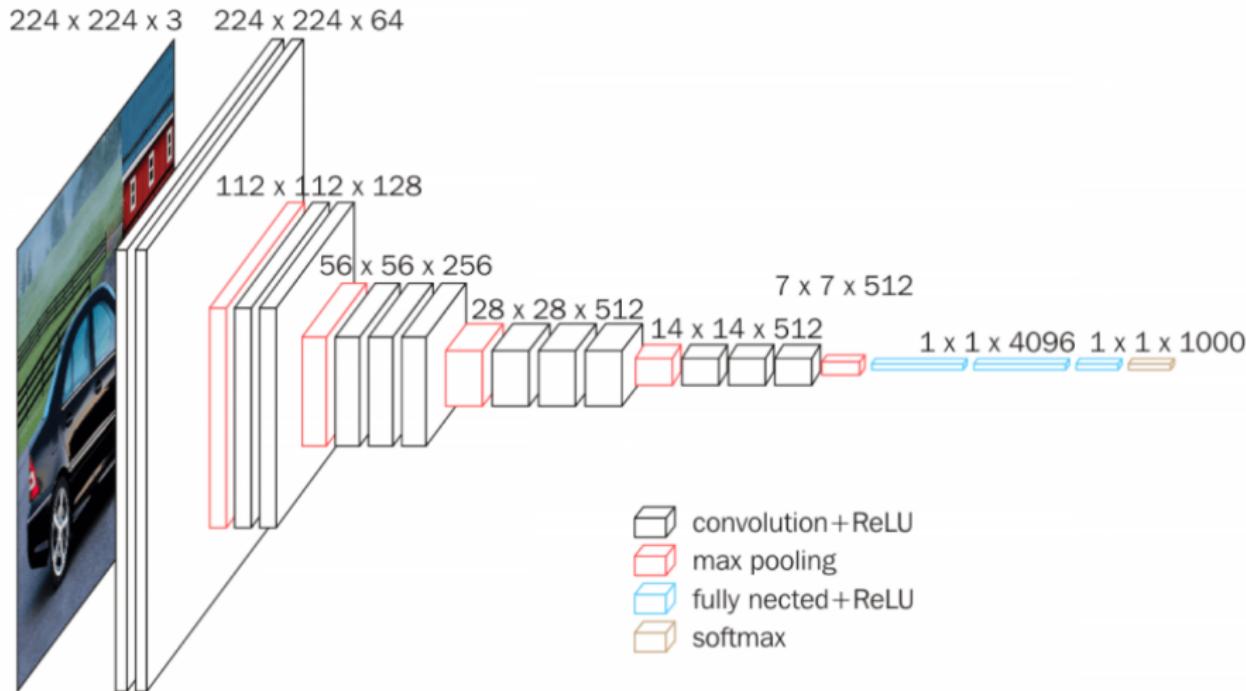
La bascule : AlexNet (2012)



Observations :

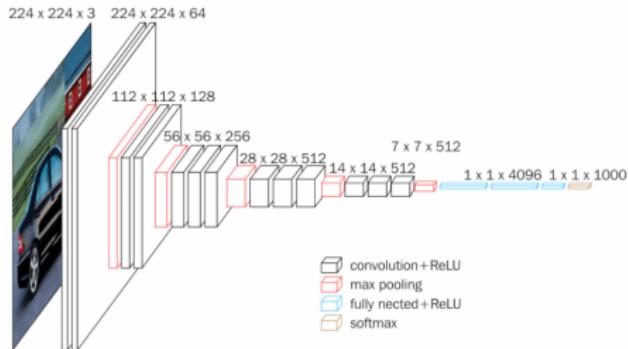
- Diminution progressive de la taille des filtres ($11 \rightarrow 5 \rightarrow 3$)
- Diminution progressive de la taille de l'image ($224 \rightarrow 55 \rightarrow 27 \rightarrow 13$)
- Augmentation progressive du nombre de filtres ($96 \rightarrow 256 \rightarrow 384$)
- *Stride puis Max Pooling*

Une version "simplifiée" : VGG-16 (2014)



$\simeq 138M$ paramètres, 16 couches dans sa version standard.

Une version "simplifiée" : VGG-16 (2014)

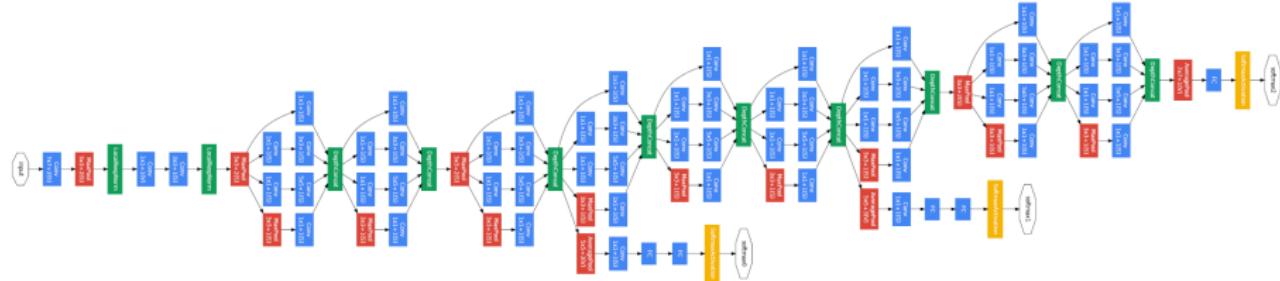


Objectif : étudier l'impact de la profondeur sur les performances du réseau.
→ Pour cela, les auteurs ont rendu l'architecture du réseau très régulière :

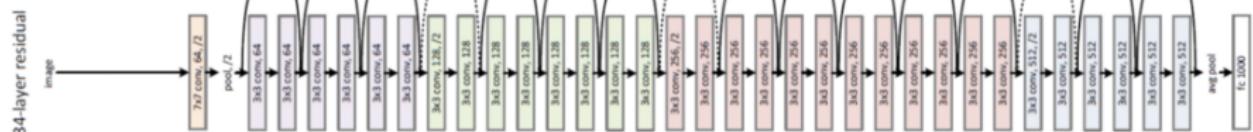
- Utilisation systématique de convolutions 3×3
- Reprise des grandes caractéristiques d'AlexNet, en les régularisant :
 - ▶ Diminution progressive de la taille de l'image ($224 \rightarrow 112 \rightarrow 56 \dots$)
 - ▶ Augmentation progressive du nombre de filtres ($64 \rightarrow 128 \rightarrow 256\dots$)

[Simonyan et Zisserman] Very Deep Convolutional Networks for Large-Scale Image Recognition

Architectures plus avancées

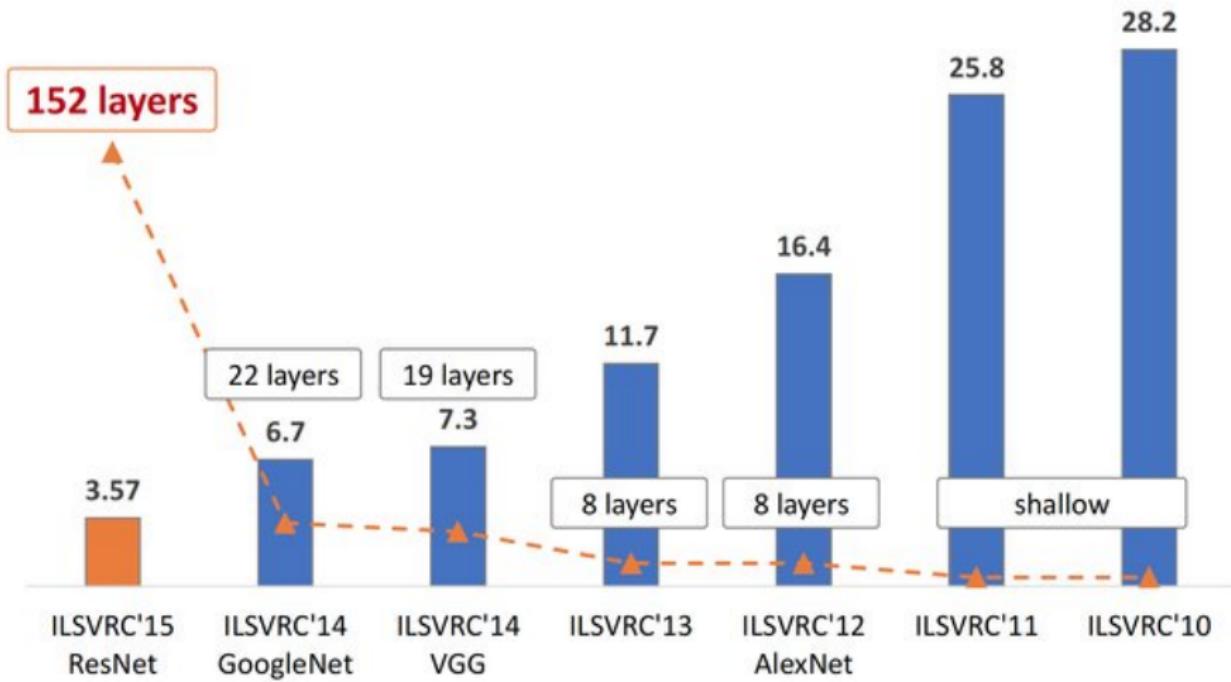


[Szegedy et al.] Going deeper with convolutions.

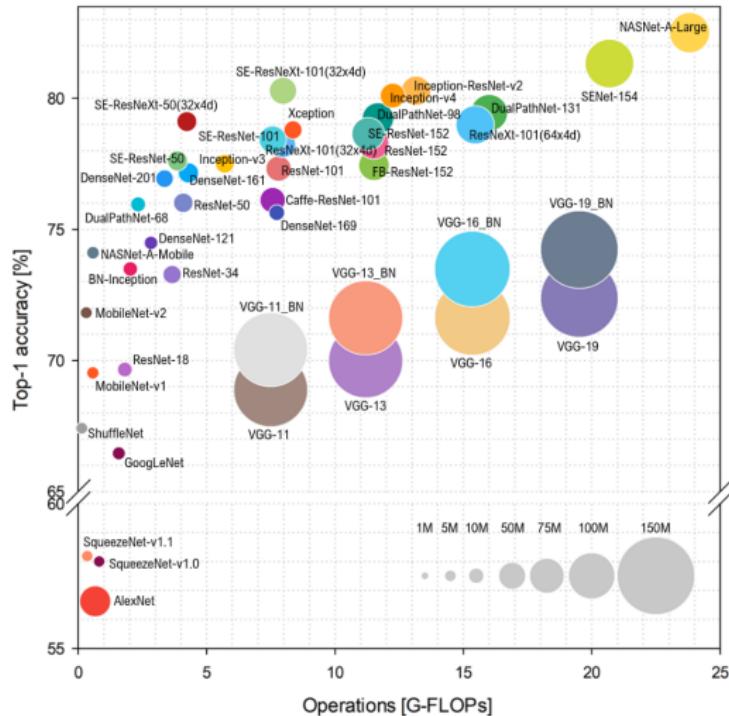


[He et al.] Deep Residual Learning for Image Recognition

2015 : fin du challenge ImageNet pour la classification d'image



Depuis 2015



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

Plan du cours

1 Introduction et motivation

2 Couches de convolution

3 Architectures convolutives

4 Visualisation

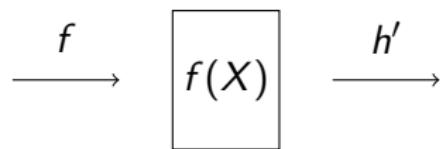
5 Transfert d'apprentissage

Apprentissage par représentation

X

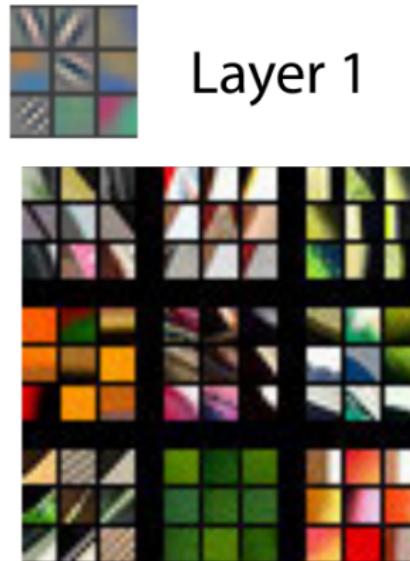


Y



Caractéristiques

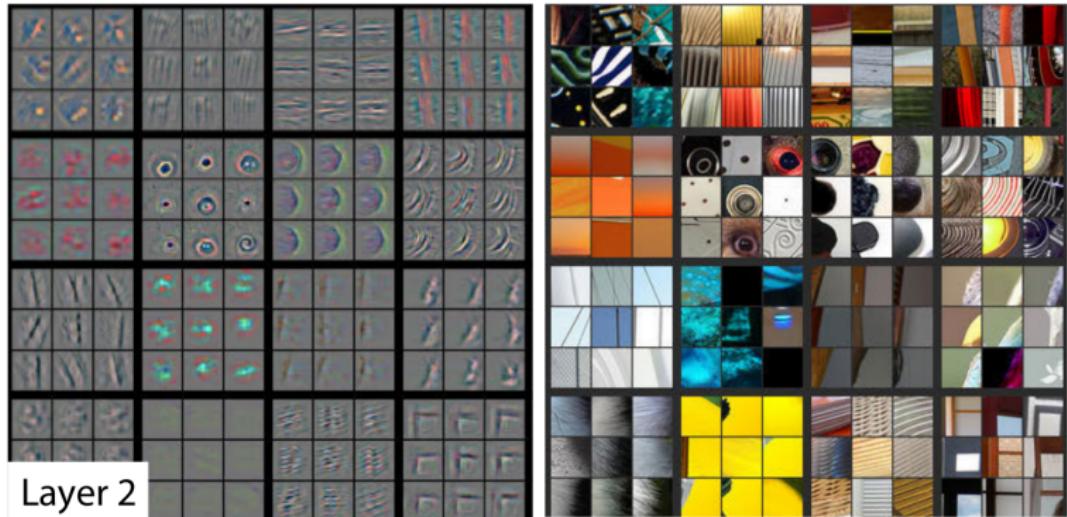
Qu'apprennent les réseaux convolutifs ?



Exemple de filtres appris sur la première couche d'un réseau (proche d'AlexNet), et pour chaque filtre, énumération des 9 patches d'images occasionnant la plus forte activation de ces filtres.

[Zeiler et al.] Visualizing and understanding convolutional networks.

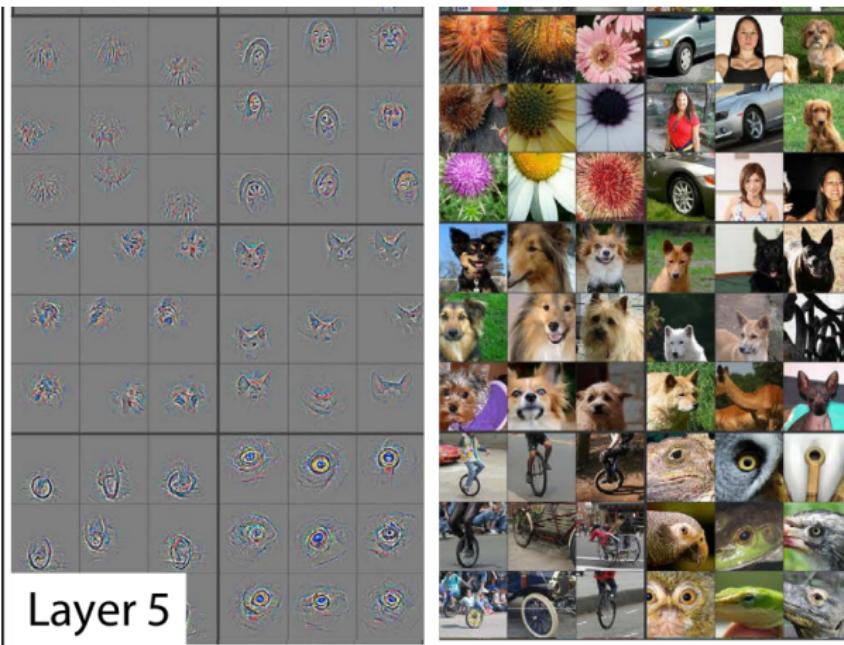
Qu'apprennent les réseaux convolutifs ?



Même visualisation que précédemment, mais pour les filtres de la deuxième couche. A noter que les filtres de la partie gauche ne sont pas présents tels quels dans le réseau, mais que cette représentation visuelle est reconstruite.

[Zeiler et al.] Visualizing and understanding convolutional networks.

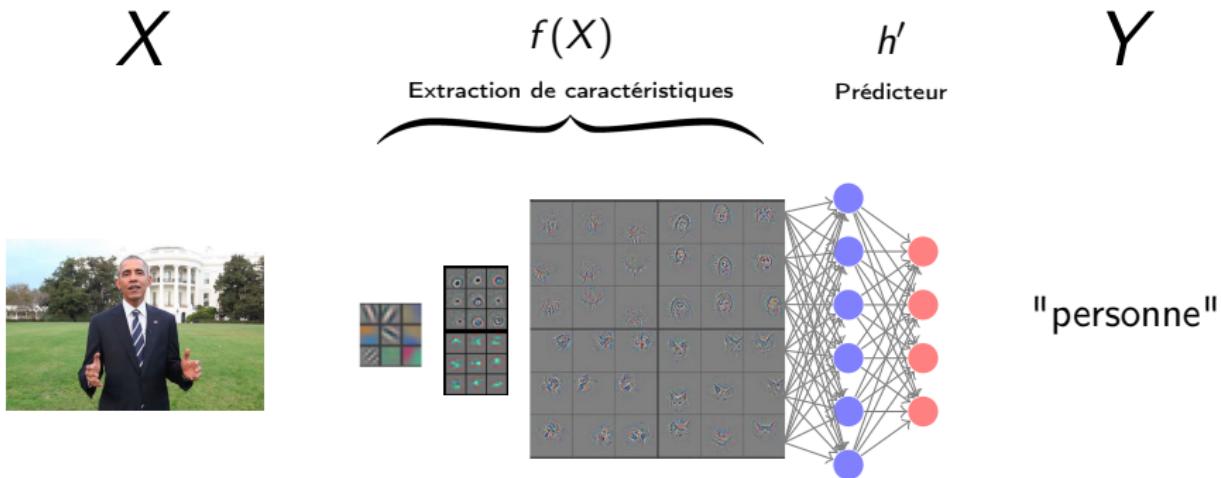
Qu'apprennent les réseaux convolutifs ?



Les motifs détectés sont d'un plus haut niveau sémantique à mesure que l'on progresse dans les couches du réseau.

[Zeiler et al.] Visualizing and understanding convolutional networks.

Une manière d'interpréter les CNN



On peut interpréter un CNN au regard de l'apprentissage par représentation : la partie convulsive est un extracteur de caractéristiques $f(X)$, et les couches denses de fin constituent notre prédicteur h' . L'apprentissage profond permet ainsi d'apprendre l'extracteur de caractéristiques en plus du prédicteur !

Plan du cours

1 Introduction et motivation

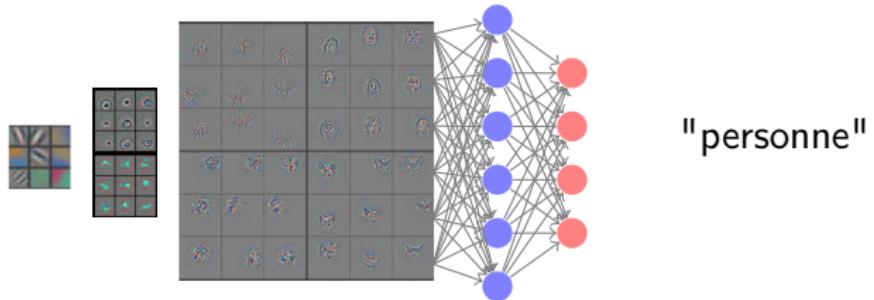
2 Couches de convolution

3 Architectures convolutives

4 Visualisation

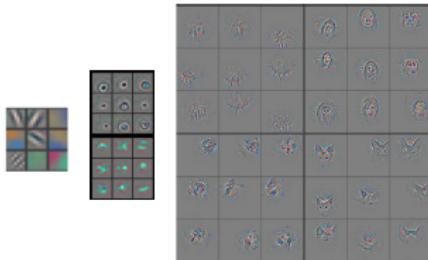
5 Transfert d'apprentissage

Transfert d'apprentissage



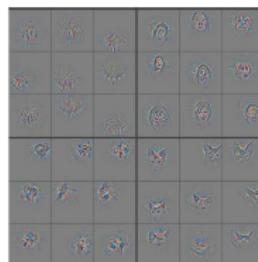
Supposons que l'on dispose d'un CNN entraîné sur une large base de données, comme ImageNet (≈ 14 millions d'images).

Transfert d'apprentissage

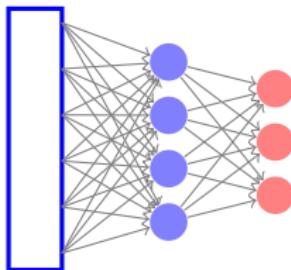


On peut extraire la base convulsive qui agit comme extracteur de caractéristiques, et la réutiliser pour une autre tâche. C'est ce que l'on appelle le **transfert d'apprentissage** (*transfer learning*).

Transfert d'apprentissage "statique"



1- Extraction de caractéristiques

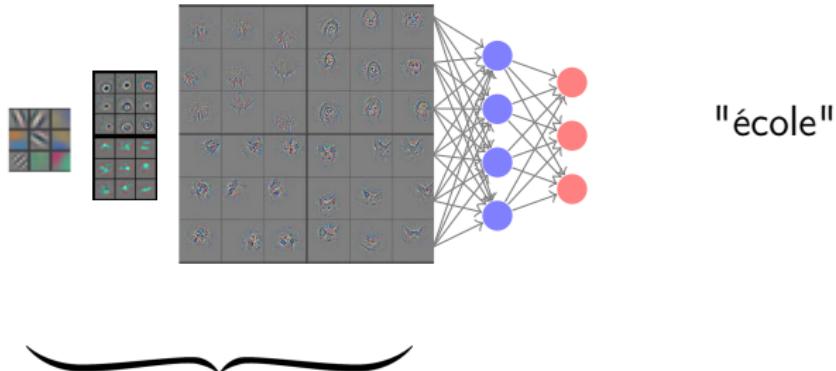


"école"

2- Entraînement d'un classifieur de ces caractéristiques

On peut utiliser un réseau pré-entraîné pour extraire les caractéristiques d'une nouvelle base de données, puis entraîner un simple classifieur de ces caractéristiques.

Transfert d'apprentissage

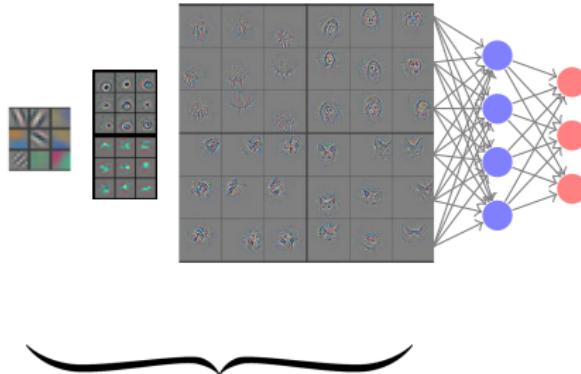


1- Gel des paramètres
de l'extracteur de
caractéristiques

2- Entraînement des
dernières couches
du classifieur

Si l'extraction de caractéristiques est incluse au classifieur, mais que ses paramètres sont bloqués, le transfert d'apprentissage supporte l'augmentation de données.

Fine-Tuning



Dégel des paramètres de l'extracteur
et ré-entraînement complet du classifieur

Une fois les dernières couches du classifieur entraînées, on peut ensuite débloquer les paramètres de la base convolutive et ré-entraîner l'ensemble du réseau, pour le "spécifier" à la nouvelle tâche : c'est le *fine-tuning*.
Attention : le taux d'apprentissage utilisé doit être très petit pour ne pas risquer de détruire les filtres généraux qui avaient été obtenus lors du pré-entraînement.