

Introduction aux Réseaux de Neurones Convolutifs

Statistique en grande dimension et Apprentissage profond

Juliette Chevallier

2023 – 2024

INSA Toulouse, juliette.chevallier@insa-toulouse.fr

Plan du cours

Introduction et motivation

Couches de convolution

Architectures convolutives

Interprétation des filtres de convolutions

Transfert d'apprentissage

Bonus

Quelques problèmes classiques en traitement d'image



Quelques problèmes classiques en traitement d'image



Classification

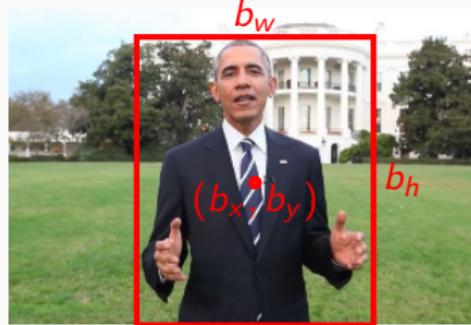
ou : Annotation, *Labelling*

Classe principale : **Personne**

Classes secondaires :

bâtiment, arbres, pelouse, ciel

Quelques problèmes classiques en traitement d'image



Classification

ou : 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.

Quelques problèmes classiques en traitement d'image



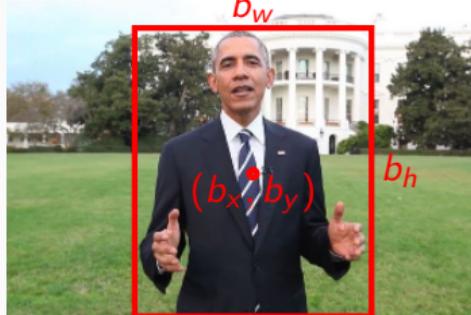
Classification

ou : Annotation, *Labelling*

Classe principale : **Personne**

Classes secondaires :

bâtiment, arbres, pelouse, ciel



Localisation

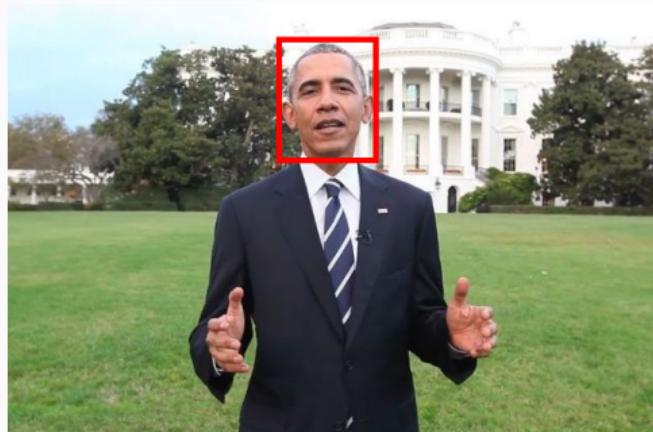
Objectif : Délimiter la **position** de l'objet, à l'aide d'une *boîte englobante*.



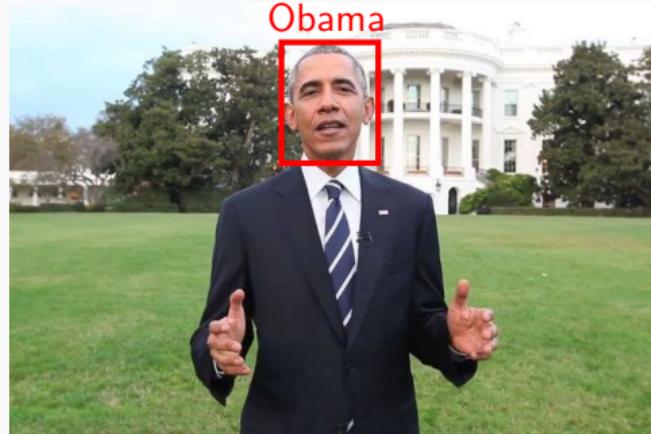
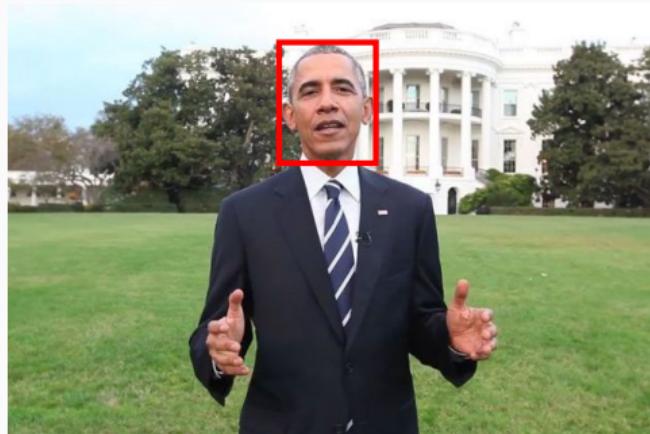
Détection d'objet

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



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



Segmentation d'objet

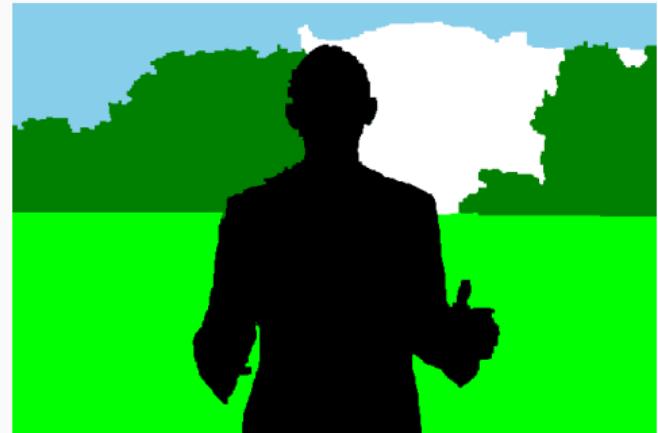
Objectif : Classification binaire des pixels de l'image comme faisant partie de l'objet ou du fond.

Segmentation



Segmentation d'objet

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



Segmentation d'image
ou *Image parsing*

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

Limite des réseaux dense (MLP)

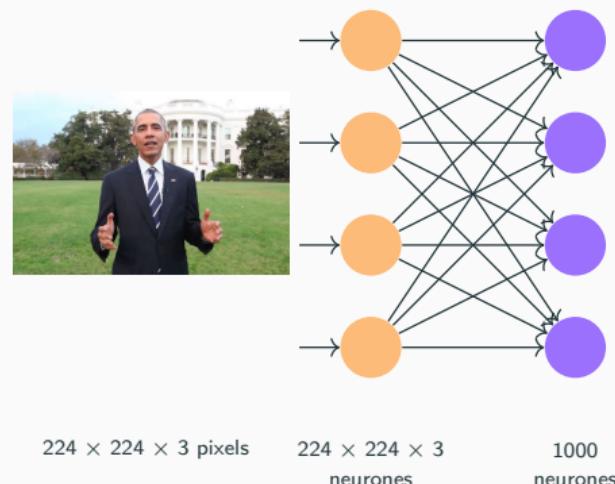
Exemple : classification d'images sur *ImageNet*.

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

```
model = keras.Sequential()
model.add(layers.Dense(1000, activation='relu', input_shape=(224*224*3,)))
model.summary()

Model: "sequential_1"

Layer (type)          Output Shape         Param #
=====
dense_1 (Dense)      (None, 1000)        150529000
=====
Total params: 150,529,000
Trainable params: 150,529,000
Non-trainable params: 0
```



- Un perceptron monocouche nécessite $224 \times 224 \times 3 \times 1000 + 1000$ paramètres, soit plus de 150 millions de paramètres !
- Les informations spatiales sont perdues avec l'aplatissement (*flatten*).

Plan du cours

Introduction et motivation

Couches de convolution

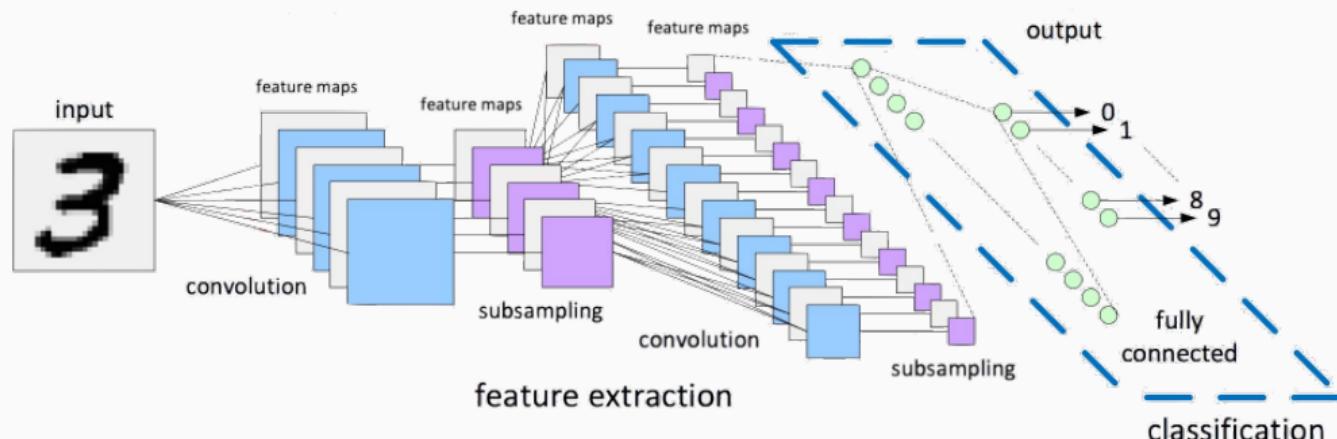
Architectures convolutives

Interprétation des filtres de convolutions

Transfert d'apprentissage

Bonus

Architecture classique d'un réseau de neurones convolutif



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

=

Image 4×4

Filtre 3×3
 $f = 3$

Réponse 2×2

Convolution

1 ₋₁	1 ₋₂	0 ₋₁	1
0 ₀	0 ₀	0 ₀	1
1 ₁	1 ₂	1 ₁	0
1	0	0	1

⊗

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

=

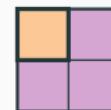


Image 4×4

Filtre 3×3
 $f = 3$

Réponse 2×2

Convolution

1 ₋₁	1 ₋₂	0 ₋₁	1
0 ₀	0 ₀	0 ₀	1
1 ₁	1 ₂	1 ₁	0
1	0	0	1

⊗

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

=

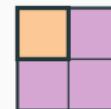


Image 4×4

Filtre 3×3
 $f = 3$

Réponse 2×2

$$-1 \times 1 - 2 \times 1 - 1 \times 0 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times 1 + 2 \times 1 + 1 \times 1 = 1$$

Convolution

1 ₋₁	1 ₋₂	0 ₋₁	1
0 ₀	0 ₀	0 ₀	1
1 ₁	1 ₂	1 ₁	0
1	0	0	1

⊗

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

=

1	

Image 4×4

Filtre 3×3
 $f = 3$

Réponse 2×2

$$-1 \times 1 - 2 \times 1 - 1 \times 0 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times 1 + 2 \times 1 + 1 \times 1 = 1$$

Convolution

1	1 ₋₁	0 ₋₂	1 ₋₁
0	0 ₀	0 ₀	1 ₀
1	1 ₁	1 ₂	0 ₁
1	0	0	1

⊗

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

=

1	1
1	1

Image 4×4 Filtre 3×3
 $f = 3$ Réponse 2×2

$$-1 \times 1 - 2 \times 0 - 1 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times 1 + 2 \times 1 + 1 \times 0 = 1$$

Convolution

$$\begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline 0_{-1} & 0_{-2} & 0_{-1} & 1 \\ \hline 1_0 & 1_0 & 0_0 & 0 \\ \hline 1_1 & 0_2 & 0_1 & 1 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & \\ \hline \end{array}$$

Image 4×4 Filtre 3×3 Réponse 2×2
 $f = 3$

$$-1 \times 0 - 2 \times 0 - 1 \times 1 + 0 \times 1 + 0 \times 1 + 0 \times 0 + 1 \times 1 + 2 \times 0 + 1 \times 0 = 1$$

Convolution

1	1	0	1
0	0_{-1}	0_{-2}	1_{-1}
1	0_0	0_0	1_0
1	0_1	1_2	1_1

Image 4×4

\circledast

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

Filtre 3×3
 $f = 3$

=

1	1
1	0

Réponse 2×2

$$-1 \times 0 - 2 \times 0 - 1 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 1 + 1 \times 0 + 2 \times 1 + 1 \times 1 = 0$$

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	1
1	0

Image 4×4

Filtre 3×3
 $f = 3$

Réponse 2×2

Convolution 2D discrète

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	1
1	0

Image 4×4

Filtre 3×3

$$f = 3$$

Réponse 2×2

Convolution 2D discrète :

$$(F \otimes I)(x, y) = \sum_m \sum_n F(n, m) \cdot I(x-m, y-n)$$

Convolution 2D discrète

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

Image 4×4 Filtre 3×3 Réponse 2×2
 $f = 3$

Étant donnés :

- Une image I en niveaux de gris (un seul *canal couleur*) de dimension $w \times h$,
- Un filtre F de dimension f ,

Alors la dimension de la réponse $F * I$ est

$$(w - f + 1) \times (h - f + 1)$$

Convolution 2D discrète :

$$(F * I)(x, y) = \sum_m \sum_n F(n, m) \cdot I(x-m, y-n)$$

Convolution 2D discrète

1	1	0	1
0	0	0	1
1	1	1	0
1	0	0	1

Image 4×4

\circledast

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

Filtre 3×3
 $f = 3$

=

1	1
1	0

Réponse 2×2

Étant donnés :

- Une image I en niveaux de gris (un seul *canal couleur*) de dimension $w \times h$,
- Un filtre F de dimension f ,

Alors la dimension de la réponse $F \circledast I$ est

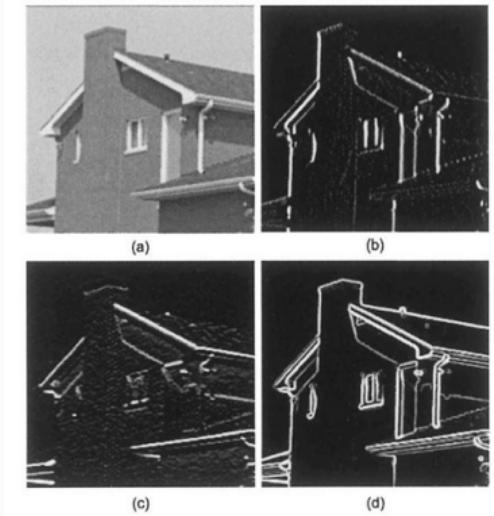
$$(w - f + 1) \times (h - f + 1)$$

Convolution 2D discrète :

$$(F \circledast I)(x, y) = \sum_m \sum_n F(n, m) \cdot I(x-m, y-n)$$

- tient compte l'**organisation spatiale**,
- invariant en translation,
- partage des paramètres.

La convolution en vision par ordinateur



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

Gx

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

Gy

- 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.

Convolution 2D de volumes : Plusieurs canaux de couleur

Soit une image colorée, en 3 dimensions : hauteur, largeur, canaux (RGB).

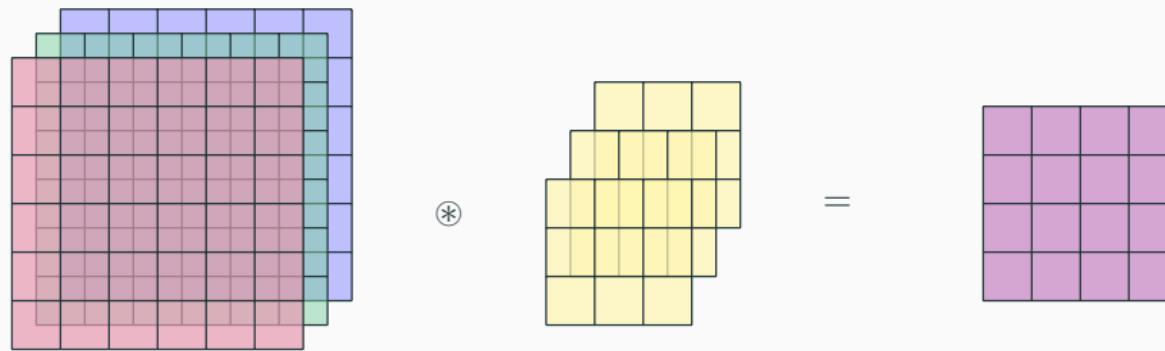


Image
 $6 \times 6 \times 3$

Filtres
 $3 \times 3 \times 3$

Réponse
 $4 \times 4 \times 1$

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

Convolution 2D de volumes : Plusieurs canaux de couleur

Soit une image colorée, en 3 dimensions : hauteur, largeur, canaux (RGB).

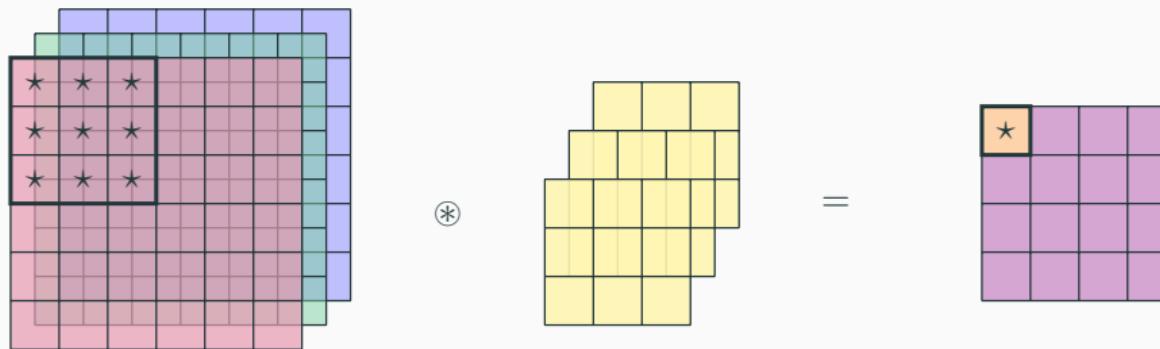


Image
 $6 \times 6 \times 3$

Filtres
 $3 \times 3 \times 3$

Réponse
 $4 \times 4 \times 1$

$$(F * I)(x, y) = \sum_{\textcolor{brown}{c}} \sum_m \sum_n F^{\textcolor{brown}{c}}(n, m) \cdot I^{\textcolor{brown}{c}}(x - m, y - n)$$

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

Couche de convolution 2D de volumes

Soit une image colorée, en 3 dimensions : hauteur, largeur, canaux (RGB).

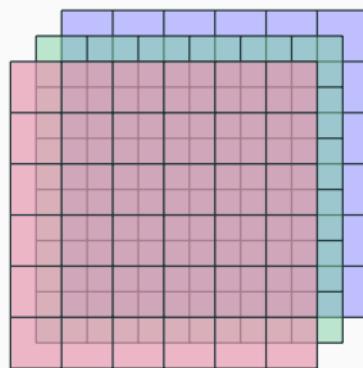
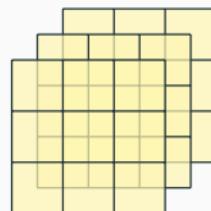
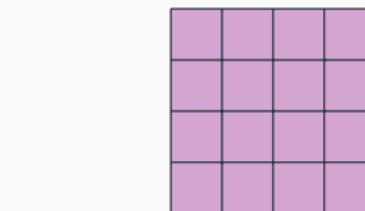


Image
 $6 \times 6 \times 3$



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

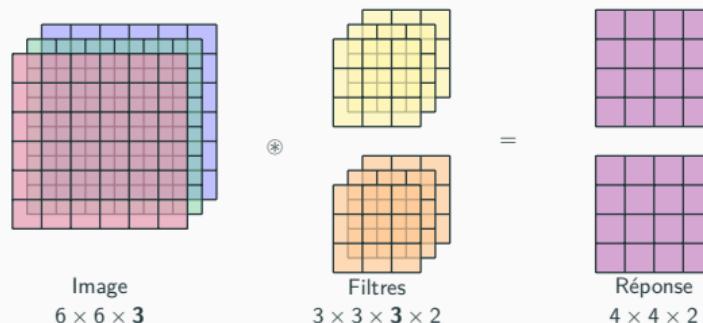


Réponse
 $4 \times 4 \times 2$

*

=

Nombre de paramètres d'une couche de convolution

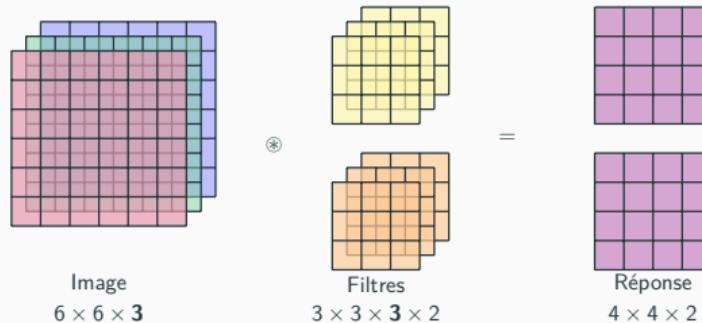


Il y a **deux** types de paramètres dans une couche de convolution :

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

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

Nombre d'opérations dans 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.

Remarque : CPU vs. GPU

CPU :

- Peu de cœurs....
 - ...mais extrêmement complexe et rapide.
 - Système de mémoire partagée.
- ~~> *Tâches séquentielles.*

GPU :

- Beaucoup de cœurs...
 - ...mais simple et lent.
 - Système à mémoire non partagée.
- ~~> *Tâches parallèles.*

Remarque : CPU vs. GPU

CPU :

- Peu de cœurs....
 - ...mais extrêmement complexe et rapide.
 - Système de mémoire partagée.
- ~~> *Tâches séquentielles.*

GPU :

- Beaucoup de cœurs...
 - ...mais simple et lent.
 - Système à mémoire non partagée.
- ~~> *Tâches parallèles.*

- En particulier, le GPU n'est pas adapté à tous les algorithmes ML
- Le temps de chargement des données sur le GPU peut être très élevé : Le GPU n'est utile que si le temps de calcul est supérieur au temps de chargement, c'est-à-dire qu'il est utile pour les **modèles complexes**.

Retour sur la classification d'images sur *ImageNet*

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

```
model = keras.Sequential()
model.add(layers.Conv2D(1000, kernel_size=(3,3), activation='relu', input_shape=(224,224,3)))
model.summary()

Model: "sequential_2"
-----  

Layer (type)          Output Shape         Param #
-----  

conv2d (Conv2D)        (None, 222, 222, 1000)    28000  

-----  

Total params: 28,000
Trainable params: 28,000
Non-trainable params: 0
```

- Avec un perceptron *mono-couche*, plus de 150 **millions** de paramètres,
- Avec une simple couche de convolution, filtre de taille 3, un seul filtre : “seulement” 28 000 ($= (3^3 + 1) \times 1000$) paramètres.

Stride

The diagram illustrates a convolution operation with a stride of 1. The input image (Image) is a 6x6 matrix with values ranging from 1 to 9. The filter (Filtre) is a 3x3 matrix with values [-1, -2, -1], [0, 0, 0], and [1, 2, 1]. The resulting output (Réponse) is a 4x4 matrix where each element is the result of the convolution step between the corresponding image patch and the filter. The output values are 10, 13, 7, 4 in the first row; 3, -3, -1, 0 in the second; 9, 4, 2, 5 in the third; and 14, 2, -6, 2 in the fourth.

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	13	7	4
3	-3	-1	0
9	4	2	5
14	2	-6	2

Image

Filtre
 $f = 3$

$stride = 1$

Réponse

Stride

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

Image Filtre $f = 3$ Réponse

$stride = 2$

- 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.

Stride

The diagram shows a convolution operation with a stride of 2. The input image (left) is a 6x6 matrix with values ranging from 1 to 9. The filter (middle) is a 3x3 matrix with values -1, -2, -1; 0, 0, 0; 1, 2, 1. The resulting response (right) is a 2x2 matrix with values 10, 7; 9, 2. The filter is applied to the input image with a stride of 2, resulting in a smaller output response.

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

Image

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

Filtre
 $f = 3$

10	7
9	2

Réponse

- 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.

stride = 2

Étant donnés :

- Une image I en niveaux de gris (un seul *canal couleur*) de dimension $w \times h$,
- Un filtre F de dimension f , de **stride** s ,

Alors la dimension de la réponse $F * I$ est

$$\left\lfloor \frac{w-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h-f}{s} + 1 \right\rfloor$$

Padding

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

Image

Filtre

Réponse

Padding

$$\begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|c|c|} \hline 0 & 0 & 1 & 2 \\ \hline 0 & 1 & 1 & -1 \\ \hline 2 & 1 & 0 & 0 \\ \hline -3 & -4 & -3 & -1 \\ \hline \end{array}$$

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.

Padding

0	0	0	0	0	0	0
0	1	1	0	1	0	
0	0	0	0	1	0	
0	1	1	1	0	0	
0	1	0	0	1	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
 $p = 1$

- 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.

Filtre

Étant donnés :

- Une image I en niveaux de gris (un seul *canal couleur*) de dimension $w \times h$,
- Un filtre F de dimension f , et de **padding** p ,

Alors la dimension de la réponse $F \circledast I$ est

$$(w + 2p - f + 1) \times (h + 2p - f + 1)$$

Impact sur la dimension des tenseurs

The diagram illustrates a convolution operation. On the left, an 6×6 **Image** matrix with padding $p = 1$ is shown. It has a green block of size 3×3 at position (1,1), a purple block of size 3×3 at (2,2), and a blue block of size 3×3 at (3,3). A dashed box indicates the receptive field of the central element (3,3) which is 0. In the middle, a 3×3 **Filtre** matrix with stride $s = 2$ is shown. It has yellow elements at positions (1,1), (2,2), and (3,3). The result of the convolution is a 3×2 **Réponse** matrix with values 1, 0, 0, 2. The multiplication symbol (*) is placed between the image and the filter.

0	0	0	0	0	0
0	1	1	0	1	0
0	0	0	0	1	0
0	1	1	1	0	0
0	1	0	0	1	0
0	0	0	0	0	0

*

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

=

1	0
0	2

Image
 $p = 1$

Filtre
 $stride = 2$

Réponse

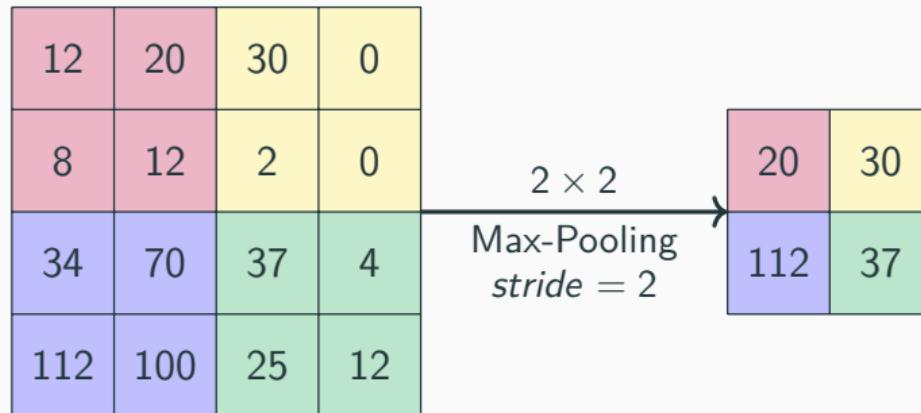
Étant donnés :

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

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

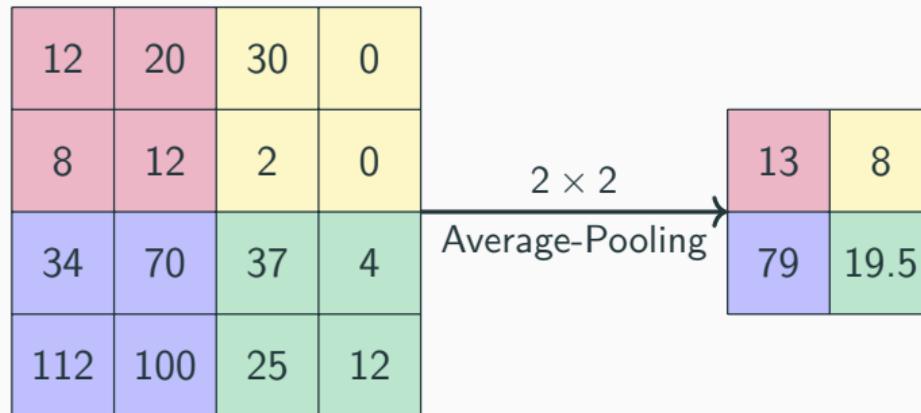
$$\left\lfloor \frac{w + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h + 2p - f}{s} + 1 \right\rfloor$$

Couche de Pooling



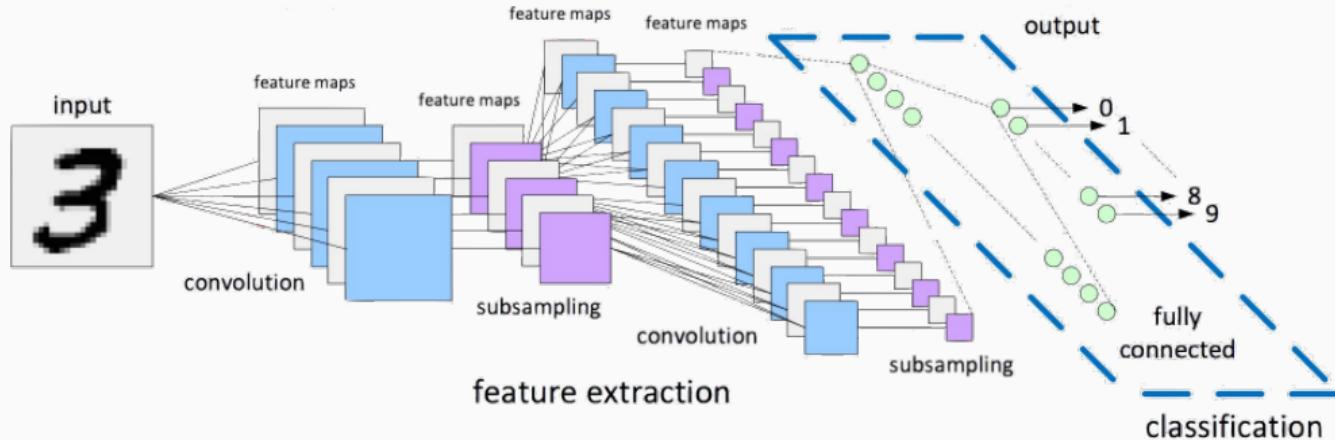
- Permet de **réduire la dimension** des tenseurs, pour contrebalancer la multiplication des réponses aux filtres de convolution,
- *Préserve les hautes réponses* des filtre de convolution,
- Introduit une *invariance à la translation*,
- **Pas de paramètres** à apprendre !

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

Introduction et motivation

Couches de convolution

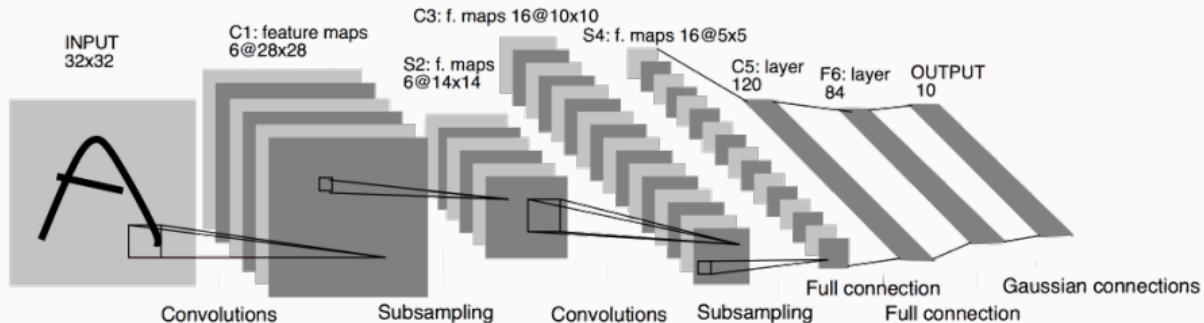
Architectures convolutives

Interprétation des filtres de convolutions

Transfert d'apprentissage

Bonus

Un pionnier : LeNet (1998)

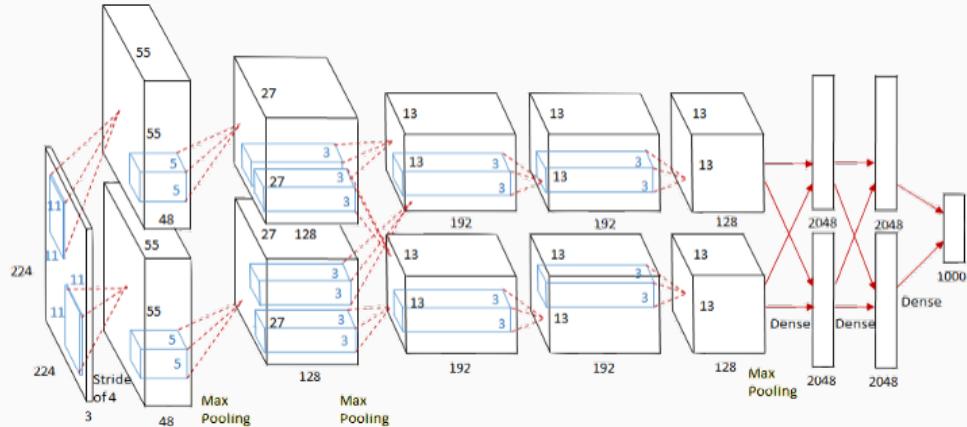


- $\simeq 60k$ paramètres
- 2 à 3 jours d'entraînement pour 20 epochs sur MNIST (en 1998 !).
- Fonctions d'activation sigmoïdes en majorité.

Visualisation : scs.ryerson.ca/~aharley/vis/

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d (AveragePooling2D)	(None, 14, 14, 6)	0
activation (Activation)	(None, 14, 14, 6)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	2416
average_pooling2d_1 (AveragePooling2D)	(None, 5, 5, 16)	0
activation_1 (Activation)	(None, 5, 5, 16)	0
conv2d_2 (Conv2D)	(None, 1, 1, 120)	48120
flatten (Flatten)	(None, 120)	0
dense (Dense)	(None, 84)	10164
dense_1 (Dense)	(None, 10)	850
<hr/>		
Total params: 61,706		
Trainable params: 61,706		
Non-trainable params: 0		

La bascule : AlexNet (2012)



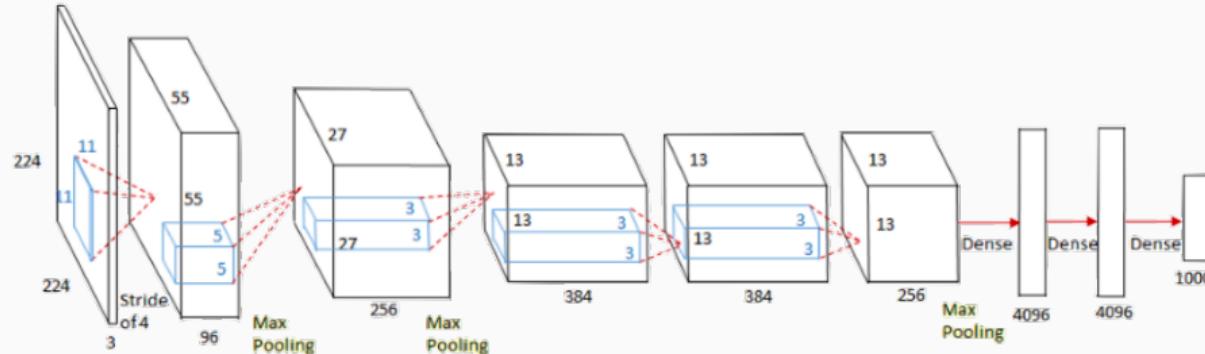
- $\approx 60M$ paramètres, 8 couches,
- L'architecture est conçue pour être implémentée sur 2 GPUs,
- Introduit l'usage de la fonction **ReLU** comme un standard pour l'entraînement de réseaux profonds.

[Krizhevsky et al.] ImageNet Classification with Deep Convolutional Neural Networks.

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 56, 56, 96)	34944
lambda (Lambda)	(None, 56, 56, 96)	0
activation (Activation)	(None, 56, 56, 96)	0
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 7, 7, 256)	614656
lambda_1 (Lambda)	(None, 7, 7, 256)	0
activation_1 (Activation)	(None, 7, 7, 256)	0
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 256)	0
conv2d_2 (Conv2D)	(None, 1, 1, 384)	885120
activation_2 (Activation)	(None, 1, 1, 384)	0
conv2d_3 (Conv2D)	(None, 1, 1, 384)	1327488
activation_3 (Activation)	(None, 1, 1, 384)	0
conv2d_4 (Conv2D)	(None, 1, 1, 256)	884992
activation_4 (Activation)	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 4096)	1052672
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 10)	40970

Total params: 21,622,154
Trainable params: 21,622,154
Non-trainable params: 0

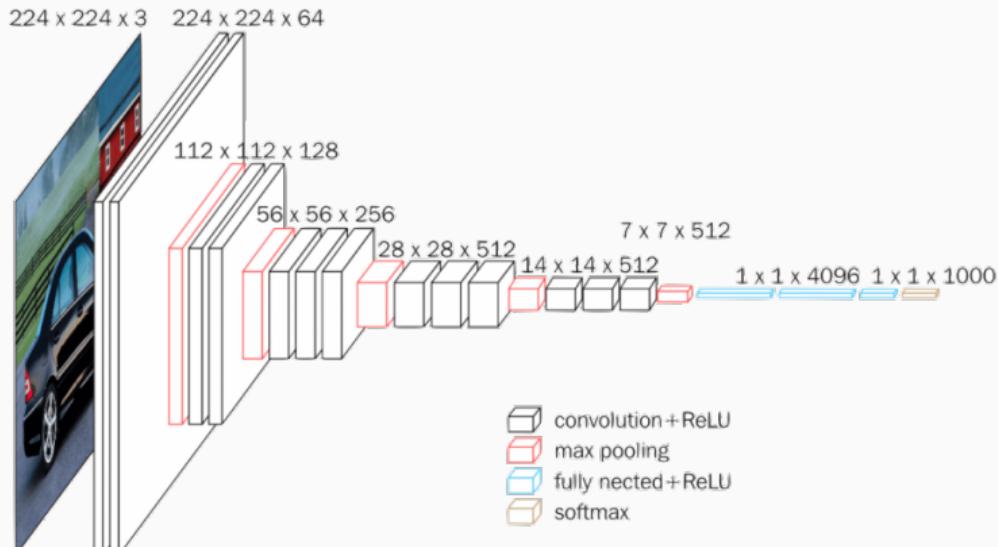
La bascule : AlexNet (2012)



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)

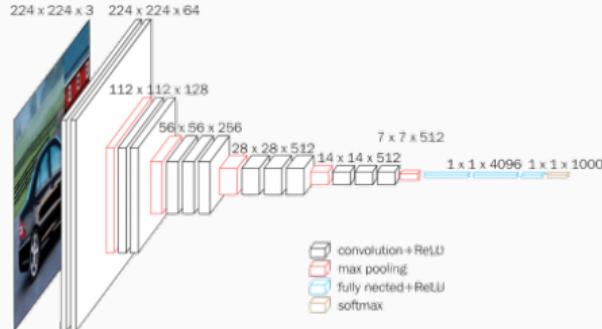


$\simeq 138M$ paramètres, 16 couches dans sa version standard.

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

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 224, 224, 64)	1792
activation (Activation)	(None, 224, 224, 64)	0
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36928
activation_1 (Activation)	(None, 224, 224, 64)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73856
activation_2 (Activation)	(None, 112, 112, 128)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147584
activation_3 (Activation)	(None, 112, 112, 128)	0
max_pooling2d_1 (MaxPooling)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295168
activation_4 (Activation)	(None, 56, 56, 256)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590080
activation_5 (Activation)	(None, 56, 56, 256)	0
conv2d_6 (Conv2D)	(None, 56, 56, 256)	65792
activation_6 (Activation)	(None, 56, 56, 256)	0
max_pooling2d_2 (MaxPooling)	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1180160
activation_7 (Activation)	(None, 28, 28, 512)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2359808
activation_8 (Activation)	(None, 28, 28, 512)	0
conv2d_9 (Conv2D)	(None, 28, 28, 512)	262656
activation_9 (Activation)	(None, 28, 28, 512)	0
max_pooling2d_3 (MaxPooling)	(None, 14, 14, 512)	0
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2359808
activation_10 (Activation)	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
activation_11 (Activation)	(None, 14, 14, 512)	0
conv2d_12 (Conv2D)	(None, 14, 14, 512)	262656
activation_12 (Activation)	(None, 14, 14, 512)	0
max_pooling2d_4 (MaxPooling)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102764544
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 10)	40970
Total params:	129,582,922	
Trainable params:	129,582,922	
Non-trainable params:	0	

Une version “simplifiée” : VGG-16 (2014)

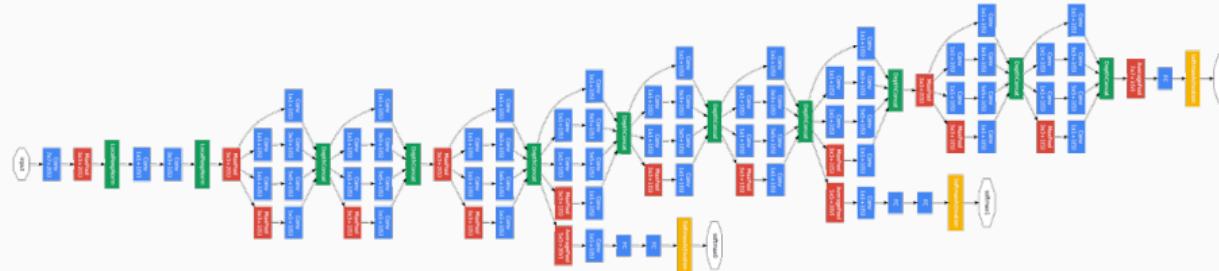


Objectif : Étudier l'impact de la **profondeur** sur les performances du réseau.

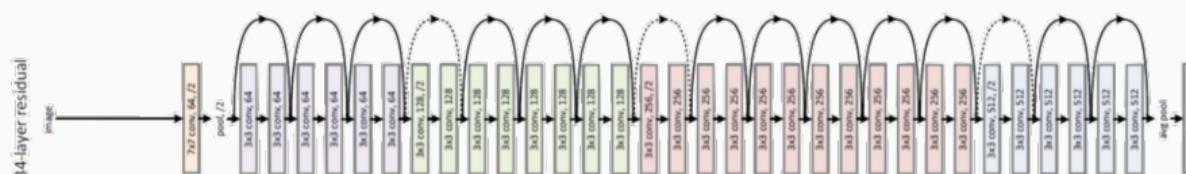
~~> Pour cela, les auteurs ont rendu l'architecture du réseau très régulière :

- 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 : $64 \rightarrow 128 \rightarrow 256$

Architectures plus avancées

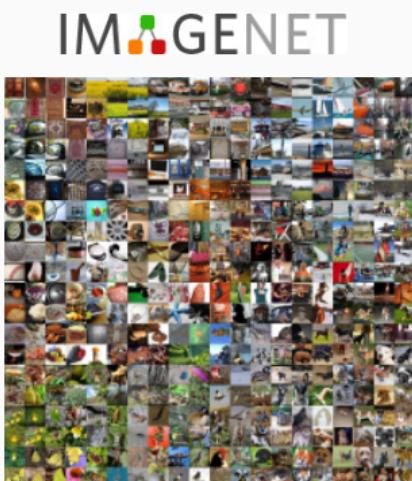


[Szegedy et al.] Going deeper with convolutions.

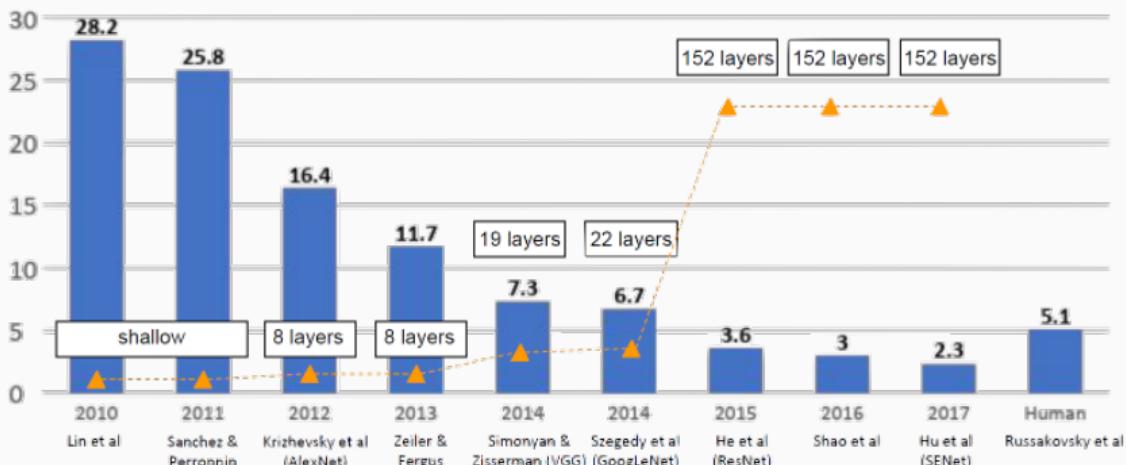


[He et al.] Deep Residual Learning for Image Recognition

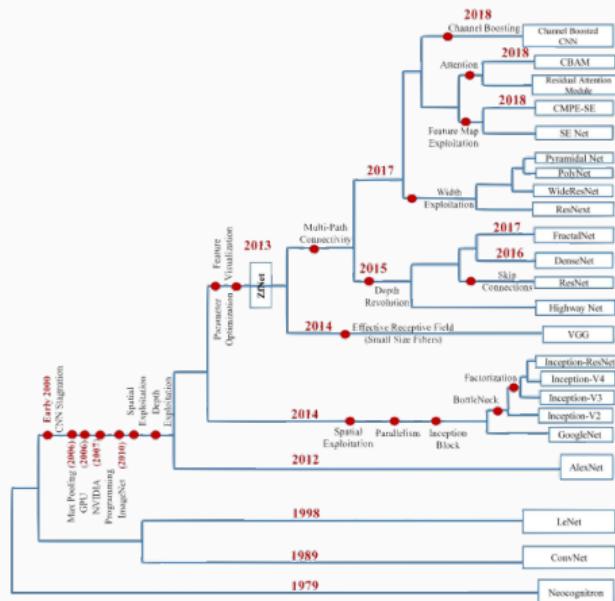
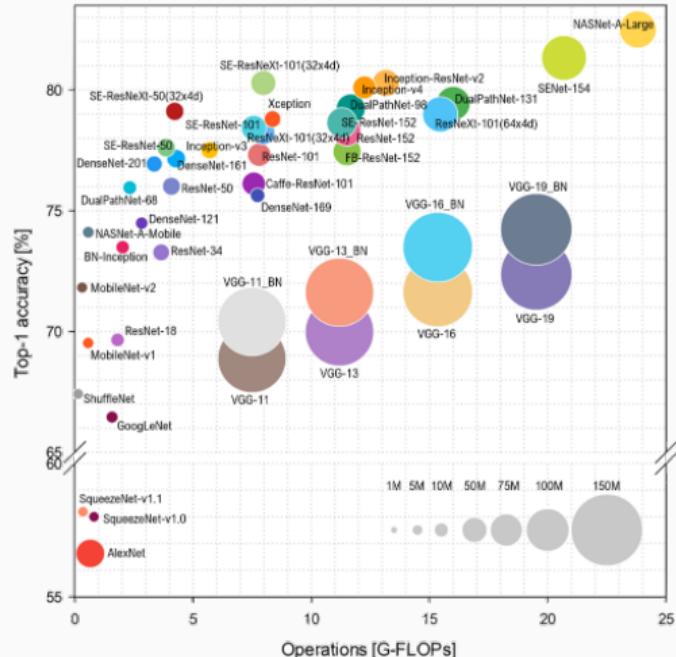
2012-1017 : Challenge ImageNet pour la classification d'image (ILSVRC)



ImageNet Large Scale Visual Recognition Challenge – ILSVRC



Depuis 2015



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures (2018)

Plan du cours

Introduction et motivation

Couches de convolution

Architectures convolutives

Interprétation des filtres de convolutions

Transfert d'apprentissage

Bonus

Apprentissage par représentation

X



Y



But :

Apprentissage par représentation

X



Y

h

"personne"

But :

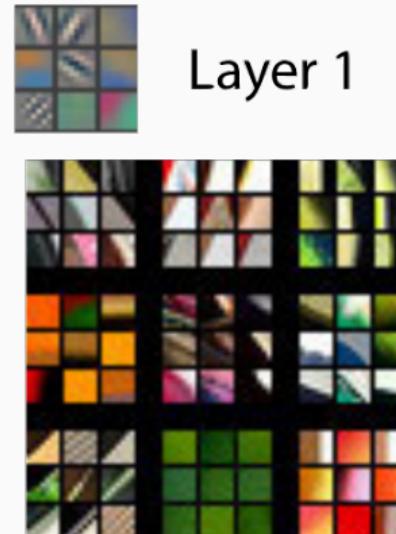
Apprentissage de
caractéristiques
(features) :

f

$f(X)$

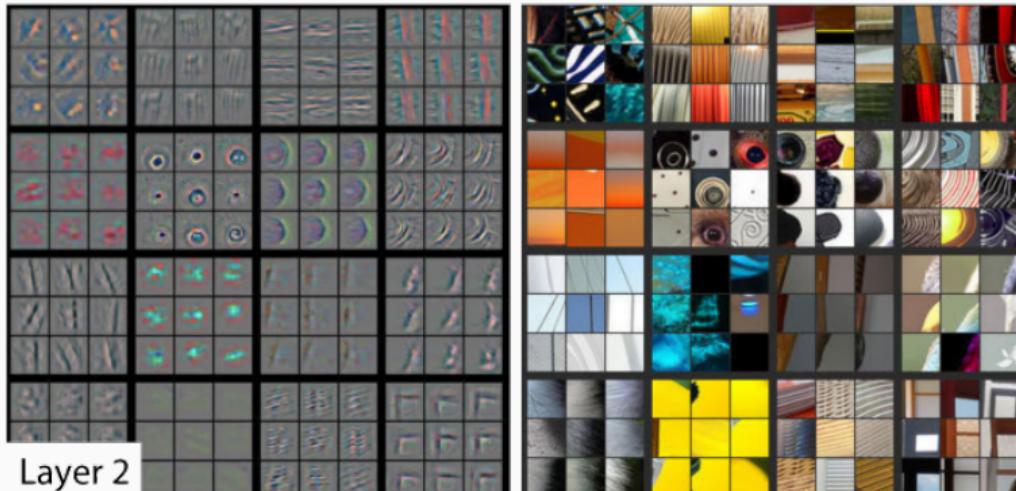
h'

Qu'apprennent les réseaux convolutifs ?



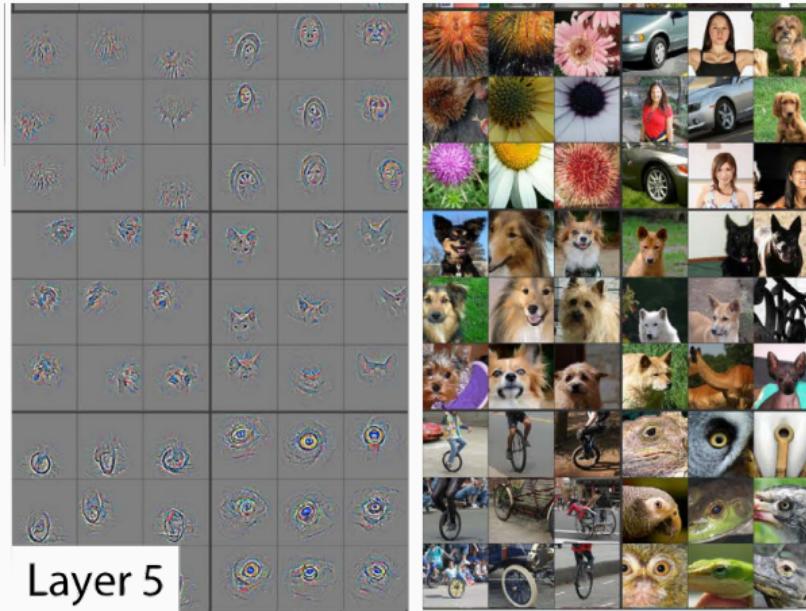
Exemple de filtres appris sur la première couche d'un réseau (proche d'AlexNet), et pour chaque filtre, énumération des 9 patches d'images occasionnant la plus forte activation de ces filtres.

Qu'apprennent les réseaux convolutifs ?



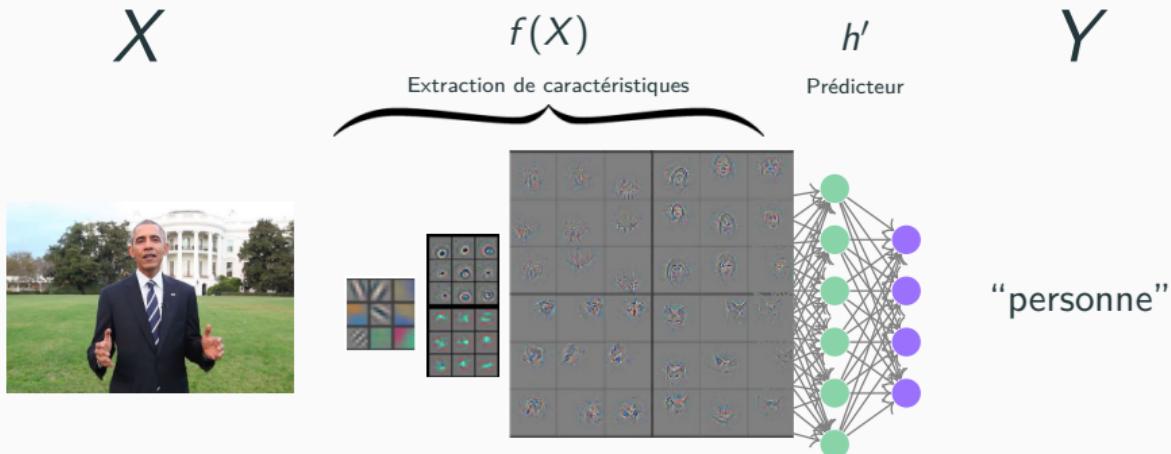
Même visualisation que précédemment, mais pour les filtres de la deuxième couche. A noter que les filtres de la partie gauche ne sont pas présents tels quels dans le réseau, mais que cette représentation visuelle est reconstruite.

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.

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 les caractéristiques en plus du prédicteur !

Plan du cours

Introduction et motivation

Couches de convolution

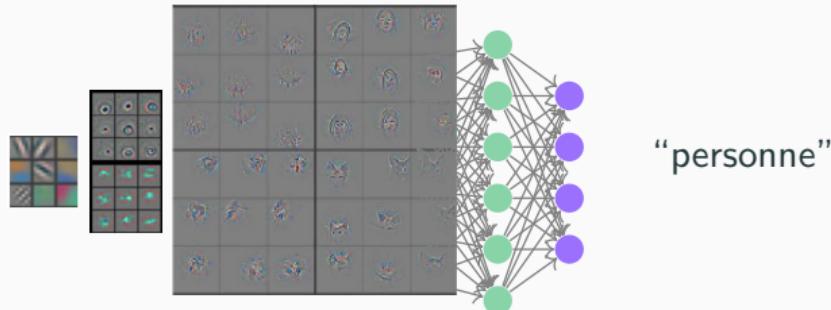
Architectures convolutives

Interprétation des filtres de convolutions

Transfert d'apprentissage

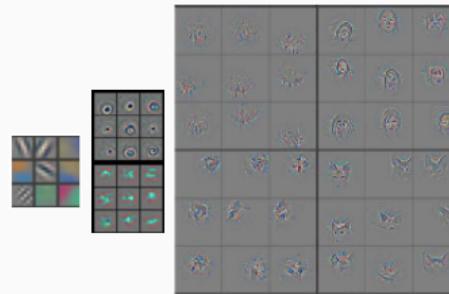
Bonus

Transfert d'apprentissage



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

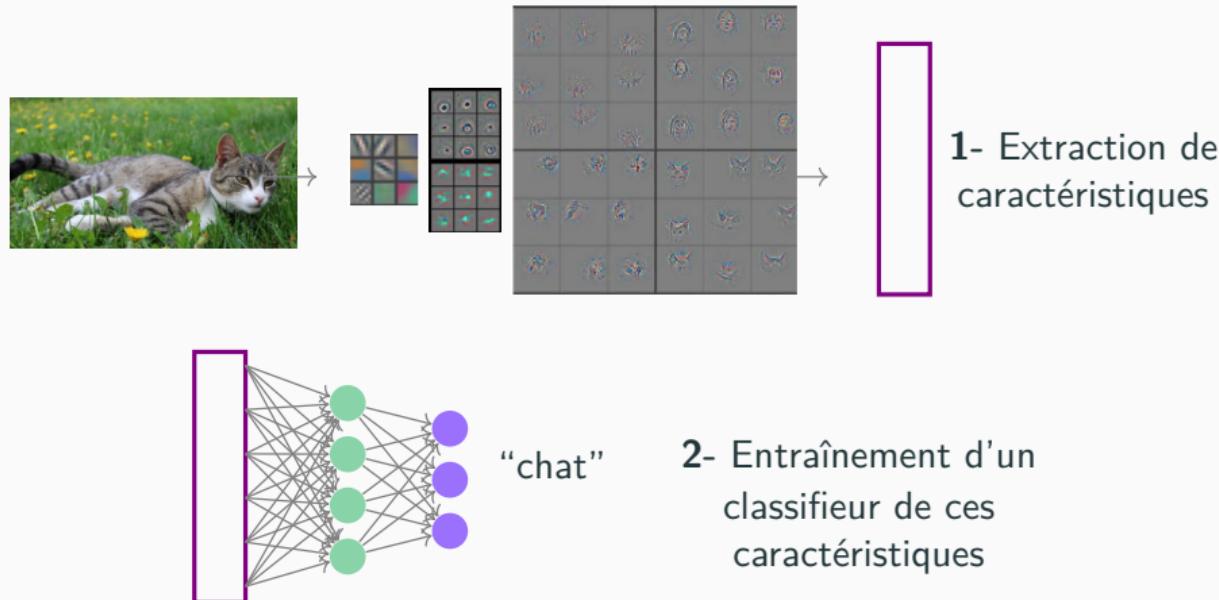
Transfert d'apprentissage



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

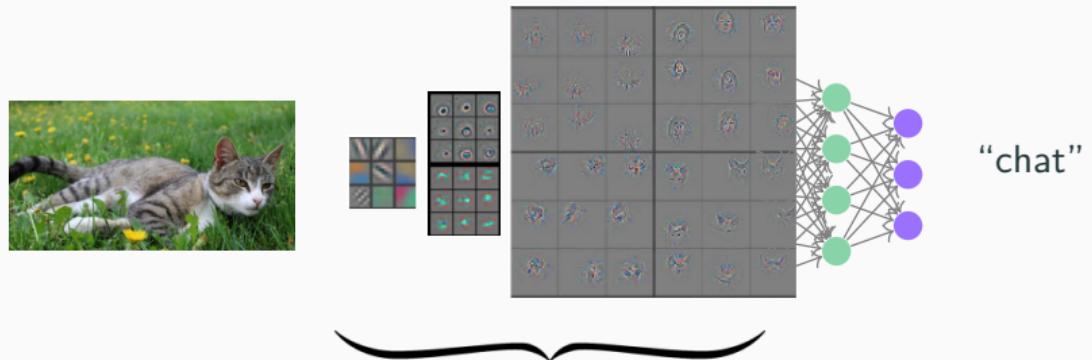
On peut extraire la base convolutive qui agit comme *extracteur de caractéristiques*, et la réutiliser pour une autre tâche. C'est ce que l'on appelle le **transfert d'apprentissage** (**transfer learning**).

Transfert d'apprentissage “statique”



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

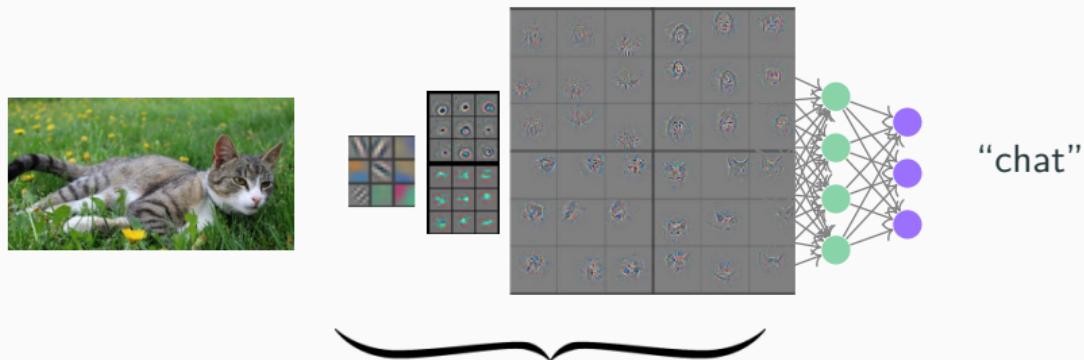


1- Gel des paramètres
de l'extracteur de
caractéristiques

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*.

Spécificité du réseau ou 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.

Plan du cours

Introduction et motivation

Couches de convolution

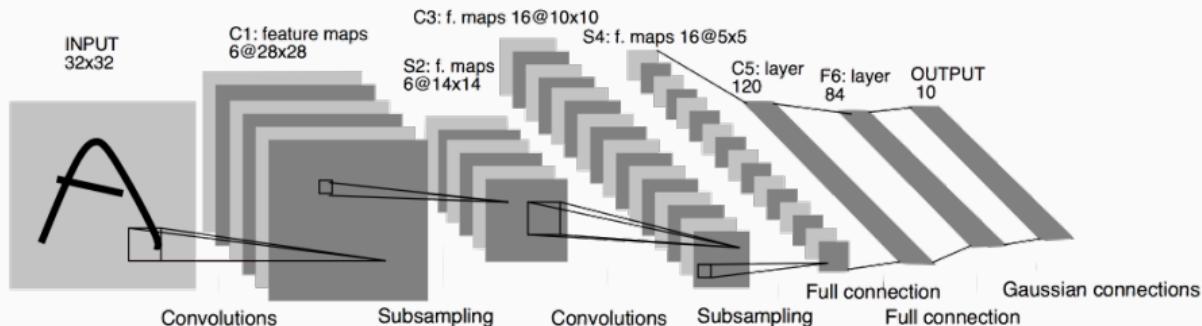
Architectures convolutives

Interprétation des filtres de convolutions

Transfert d'apprentissage

Bonus

Exercice pour mercredi 15 (TP)



Questions :

1. Nombre de paramètres ?
2. Nombre d'opérations à l'inférence ?