Réseaux de Neurones Convolutifs

A. Carlier

26/11/2019

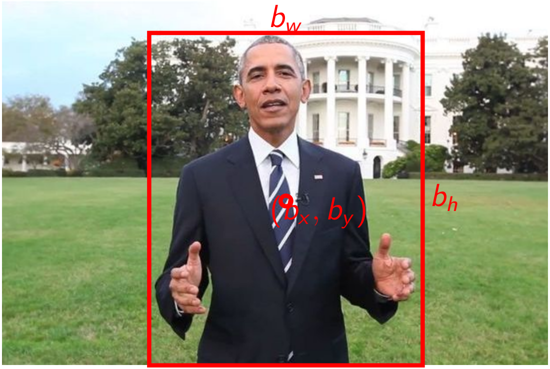
Problèmes classiques en traitement d'image

Classification

Ou encore : Annotation, *Labelling*

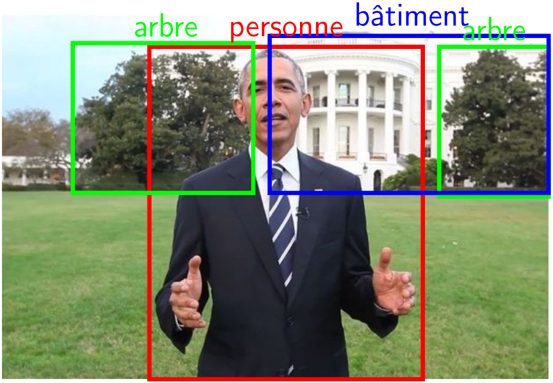
Classe principale : **Personne**
Classes secondaires : bâtiment, arbres, pelouse, ciel

Localisation



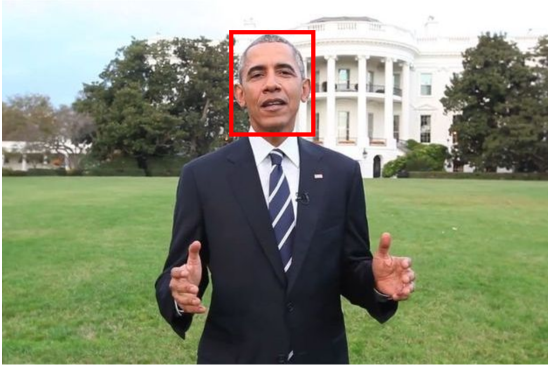
Objectif : délimiter la position de l'objet à l'aide d'une boîte englobante.

Détection

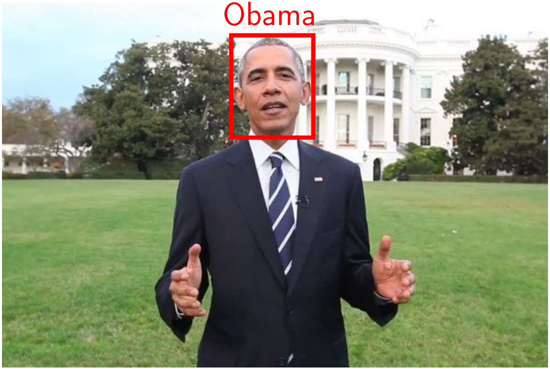


Objectif : délimiter la position d'objets multiples, avec éventuellement plusieurs instances, toujours à l'aide de boîtes englobantes.

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.

Aperçu du cours

- Problèmes classiques en traitement d'image
- Éléments constitutifs des réseaux convolutifs
- Classification d'image
  - Architectures simples : `LeNet`, `AlexNet`, `VGG-16`
  - Architectures avancées : `Inception`, `ResNet`
- Localisation et détection d'objet : `YOLO`, `R-CNN`
- Segmentation d'image : `UNet`, `FCN`, `DeepLab`
- Reconnaissance de visage : `FaceNet`

Convolution

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

Image

Filtre 3×3
$f = 3$

Réponse

Convolution

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \\ & \end{bmatrix}$$

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

$$\begin{bmatrix}1&1&0&1\\0&0&0&1\\1&1&1&0\\1&0&0&1\end{bmatrix} * \begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix} = \begin{bmatrix}1&1\\ & \end{bmatrix}$$

Image

Filtre 3×3
$f = 3$

Réponse

Convolution

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

Image

$*$

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

Filtre 3×3
$f = 3$

$=$

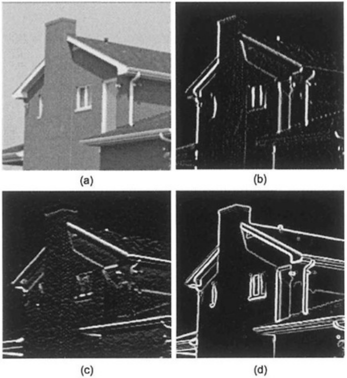
1	1
1	

Réponse

Convolution

$$\begin{array}{c}\begin{array}{|c|c|c|c|}\hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline\end{array} \\ \text{Image}\end{array} \quad * \quad \begin{array}{c}\begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline\end{array} \\ \text{Filtre } 3 \times 3 \\ f = 3\end{array} \quad = \quad \begin{array}{c}\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 0 \\ \hline\end{array} \\ \text{Réponse}\end{array}$$

La convolution en Vision par Ordinateur



$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

Gx

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

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.

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 $\mathrm{I} \circledast \mathrm{K}$ de l'image I au filtre K est $(w - f + 1) \times (h - f + 1)$

Convolution (2D) de volumes

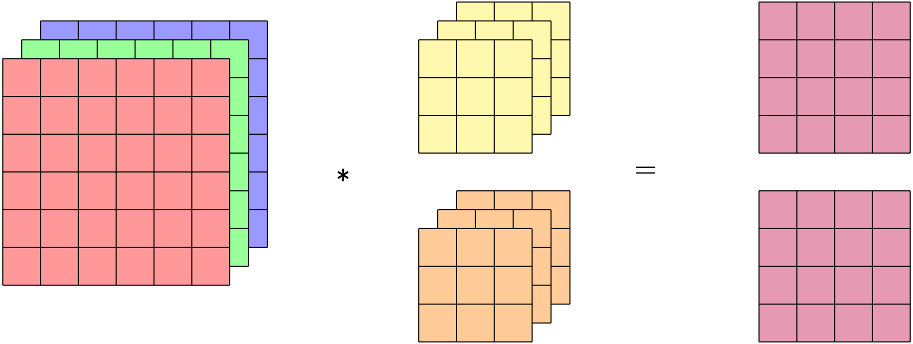


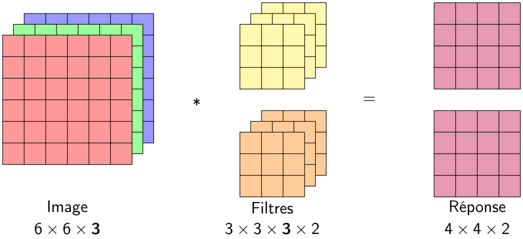
Image $6 \times 6 \times \mathbf{3}$

Filtres $3 \times 3 \times \mathbf{3} \times 2$

Réponse $4 \times 4 \times 2$

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 w *x* h *x* #canaux *x* #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 3x3x3x2 + 2 = 56 paramètres.

Padding

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

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.

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 $\mathrm{I} \circledast \mathrm{K}$ 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
$f = 3$

Réponse

$stride = 1$

Stride

$$\begin{bmatrix} 3 & 1 & 3 & 5 & 3 & 3 \\ 2 & 2 & 8 & 8 & 3 & 9 \\ 3 & 4 & 7 & 7 & 2 & 7 \\ 5 & 3 & 6 & 8 & 4 & 7 \\ 3 & 8 & 8 & 5 & 7 & 4 \\ 7 & 9 & 6 & 4 & 6 & 9 \end{bmatrix} * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} \boxed{10} & 7 \\ 9 & 2 \end{bmatrix}$$

Image
$p = 0$

Filtre
$f = 3$

stride $= 2$

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.

Impact sur la dimension des tenseurs

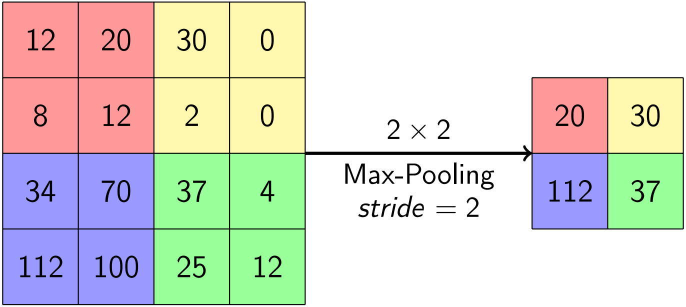
Étant donnés :

- 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 $\mathrm{I} \circledast \mathrm{K}$ de l'image I au filtre K est :

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

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

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

2×2

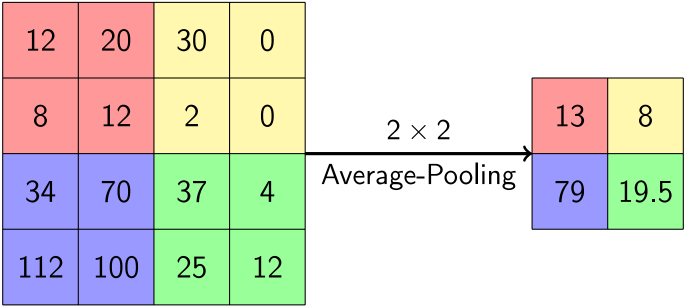
Max-Pooling

stride $= 1$

20	30	30
70	70	37
112	100	37

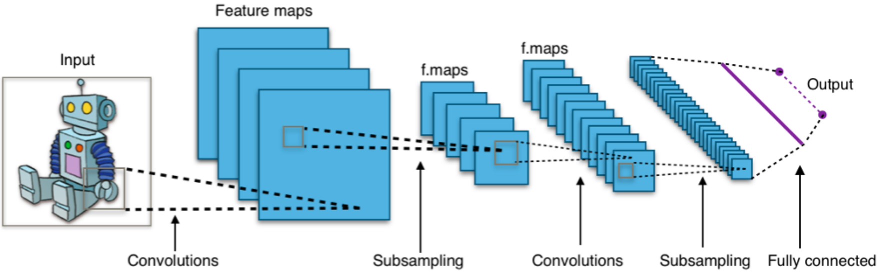
- En utilisant un *stride* de 1 et du *padding*, on peut obtenir une réponse de même dimension que l'entrée.
  $\rightarrow$ Ceci sera utile par la suite !

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.

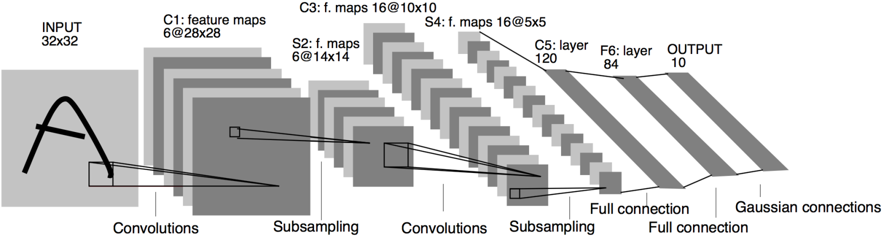
Intérêt des réseaux de neurones convolutifs

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.

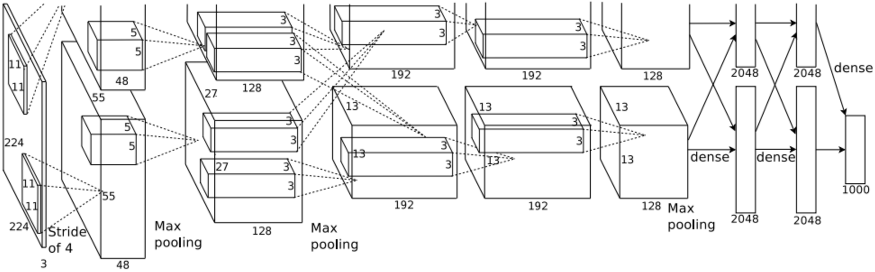
Un pionnier : LeNet (1998)



$\simeq$ 60k paramètres

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

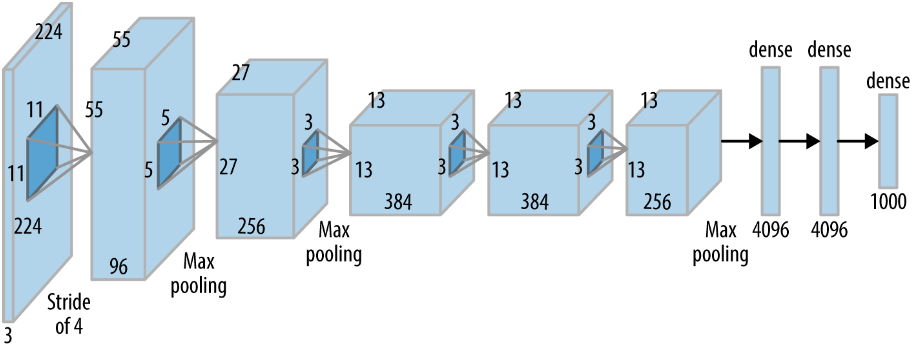
La bascule : AlexNet (2012)



$\simeq$ 60M paramètres, 8 couches

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

La bascule : AlexNet (2012)

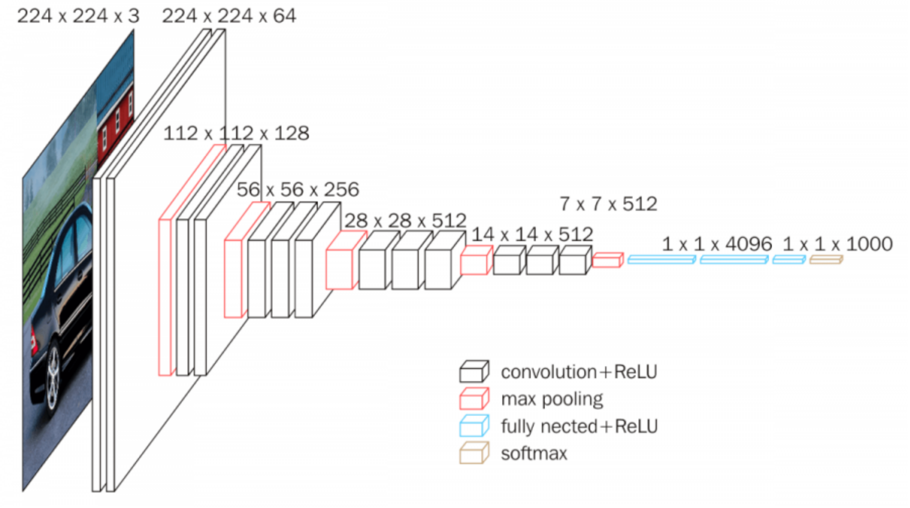


Observations :

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

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

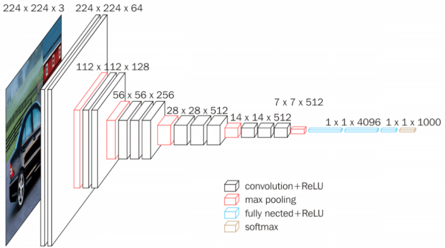
Une version "simplifiée" : VGG-16 (2014)



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

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

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

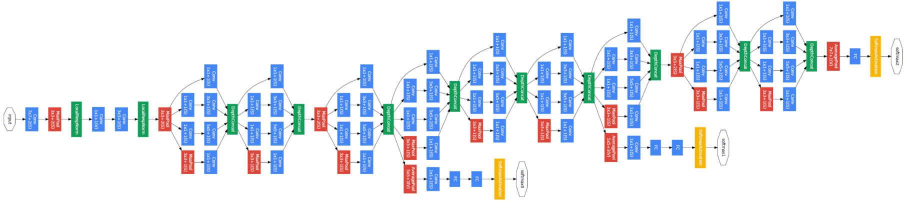


Objectif : étudier l'impact de la profondeur sur les performances du réseau.
$\rightarrow$ Pour cela, les auteurs ont rendu l'architecture du réseau très régulière :

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

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

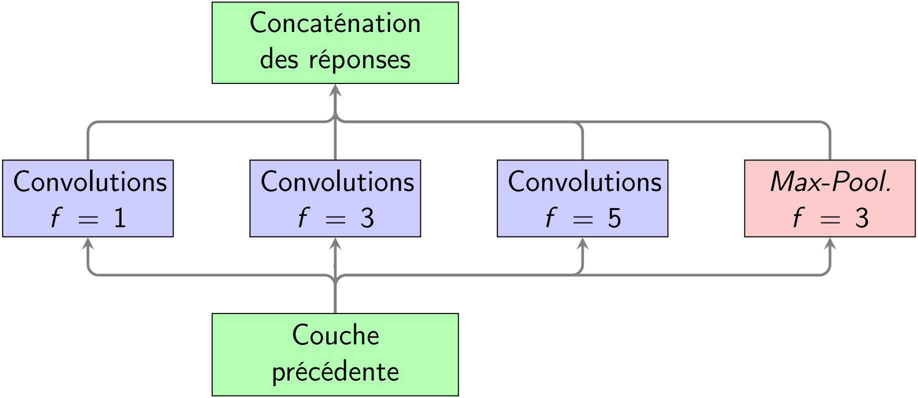
Choisir, c'est renoncer : GoogLeNet (ou Inception, 2014)



$\simeq$ 7M paramètres, 22 couches

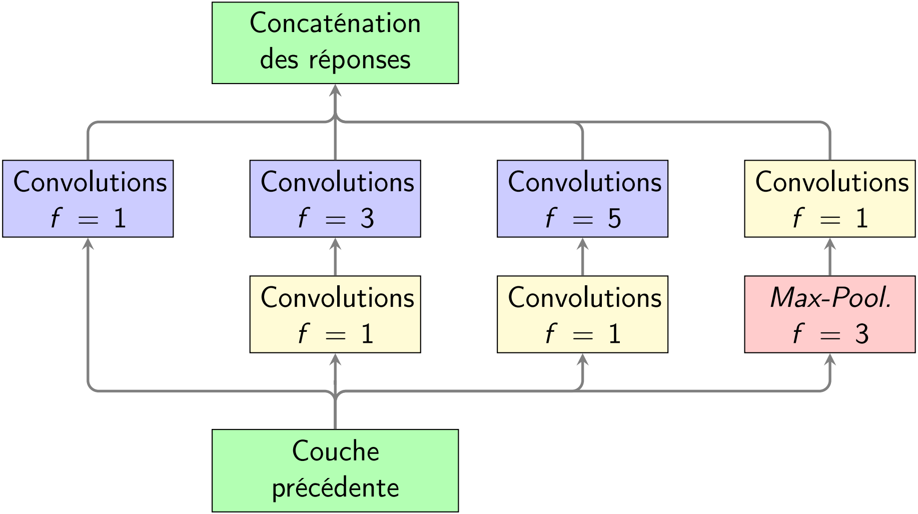
[Szegedy et al.] Going deeper with convolutions.

La brique de base d'Inception



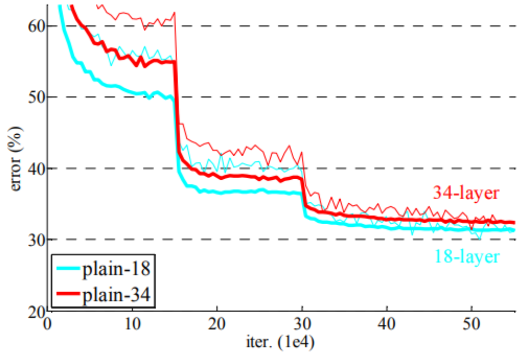
[Szegedy et al.] Going deeper with convolutions.

La brique de base d'Inception



[Szegedy et al.] Going deeper with convolutions.

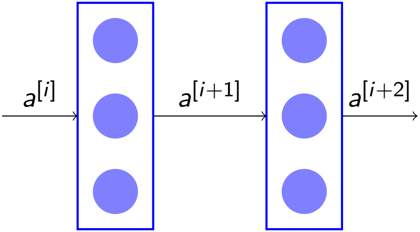
Le problème de l'évanescence des gradients



Les réseaux plus profonds devraient en théorie permettre d'approximer de plus en plus efficacement l'ensemble d'apprentissage. En pratique, ils posent de nombreux problèmes d'optimisation, comme celui de l'evanescence des gradients (mais aussi celui de l'explosion des gradients).

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

Blocs résiduels



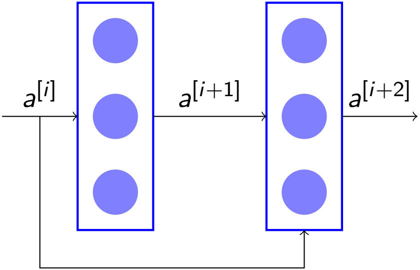
Cette vue abstrait les liaisons synaptiques entre les 2 couches $i+1$ et $i+2$.
On a ici par exemple :

$$a^{[i+1]} = \texttt{ReLU}(W^{[i+1]}a^{[i]} + b^{[i+1]})$$

et

$$a^{[i+2]} = \texttt{ReLU}(W^{[i+2]}a^{[i+1]} + b^{[i+2]})$$

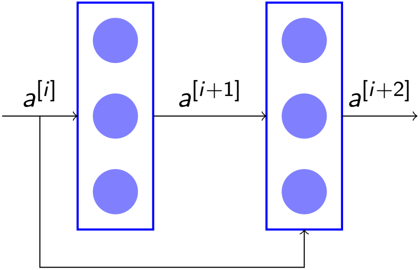
Blocs résiduels



On peut ajouter une connexion comme présenté ci-dessus (*skip connection*) pour former un **bloc résiduel**, d'équation :

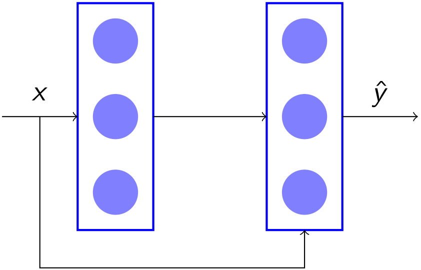
$$a^{[i+2]} = \mathtt{ReLU}(W^{[i+2]}a^{[i+1]} + b^{[i+2]} + a^{[i]})$$

Blocs résiduels



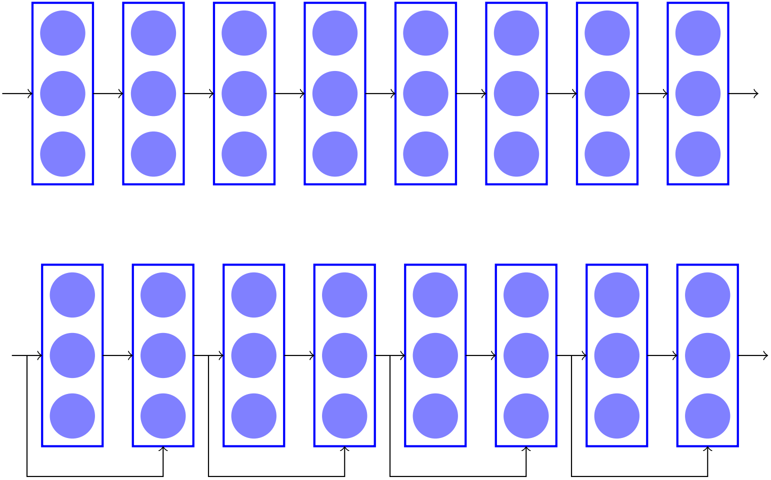
Une condition importante est que les dimensions de $a^{[i]}$ et $a^{[i+2]}$ soient identiques ! On utilise donc souvent des convolutions qui conservent la dimension (*same*) dans les réseaux résiduels.

Blocs résiduels

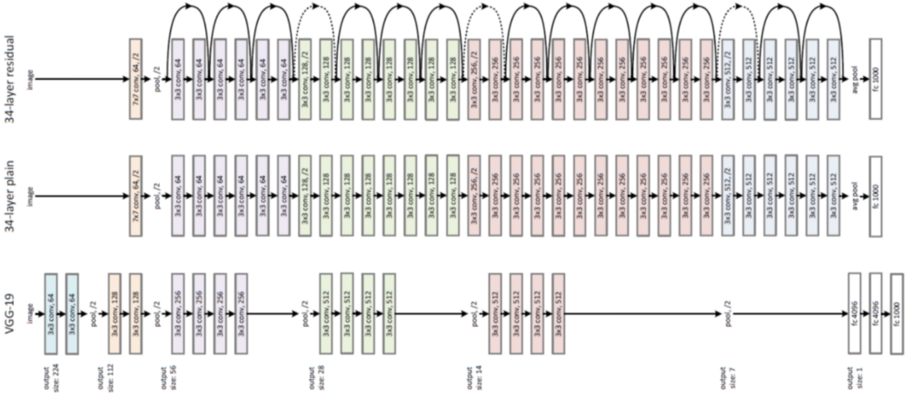


Alors que l'entraînement vise d'ordinaire à estimer une fonction F telle que $\hat{y} = F(x)$, on cherche ici la fonction telle que : $\hat{y} = F(x) + x$.
En d'autres termes, la fonction F estime le **résidu** $\hat{y} - x$ (d'où le nom de bloc résiduel).

Rendre "résiduel" un réseau de neurones

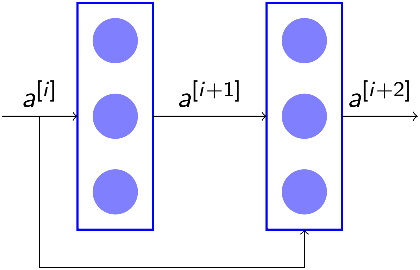


ResNet (2015)



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

Une bonne propriété des blocs résiduels



$$a^{[i+2]} = \mathtt{ReLU}(W^{[i+2]}a^{[i+1]} + b^{[i+2]} + a^{[i]})$$

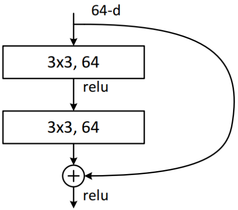
Si les poids synaptiques $W^{[i+2]}$ et les biais $b^{[i+2]}$ sont proches de 0, alors :

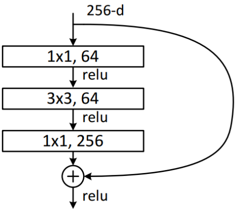
$$a^{[i+2]} \simeq \mathtt{ReLU}(a^{[i]}) \simeq a^{[i]}$$

La fonction identité est facile à modéliser par un bloc résiduel.

ResNet (2015)

Les blocs résiduels ont permis aux auteurs d'entraîner des réseaux de plusieurs centaines de couches. Le réseau qui a remporté le *challenge* `ImageNet` en 2015 comptait d'ailleurs 152 couches.
Les problèmes de ces architectures très profondes ne sont plus liés à l'optimisation mais aux performances, ce qui a motivé les auteurs à introduire des blocs résiduels de format différent :





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