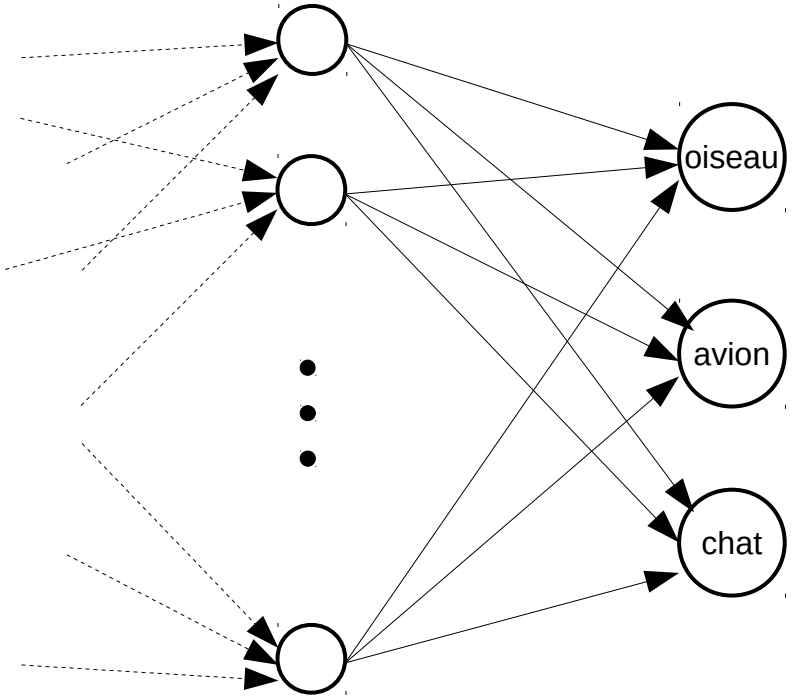
INTRODUCTION AUX RÉSEAUX DE NEURONES

Réseaux de neurones convolutifs («CNN» : *Convolutional neural networks*)

Pascal Germain*, 2018

* Merci spécial à Philippe Giguère pour m'avoir permis de réutiliser une partie de ces transparents.

Mais d'abord :
Réseau avec sortie multiclasse

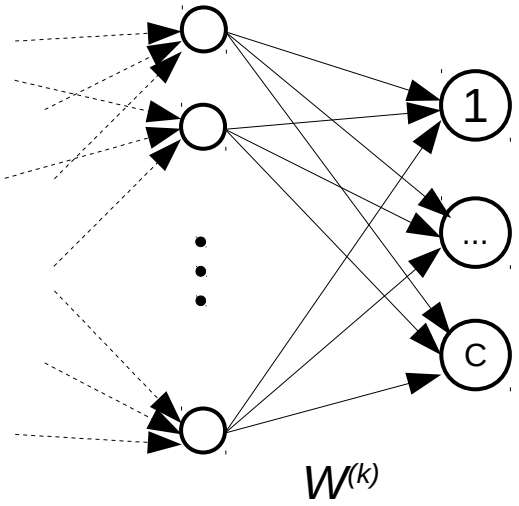


Réseau avec sortie multiclasse

Soit un problème à C classes.

1. Pour un exemple d'apprentissage (x, y), on représente chaque classe par un entier

$$y \in \{1, \ldots, C\}$$



Réseau avec sortie multiclasse

Soit un problème à C classes.

1. Pour un exemple d'apprentissage (x, y), on représente chaque classe par un entier
$$y \in \{1, \ldots, C\}$$

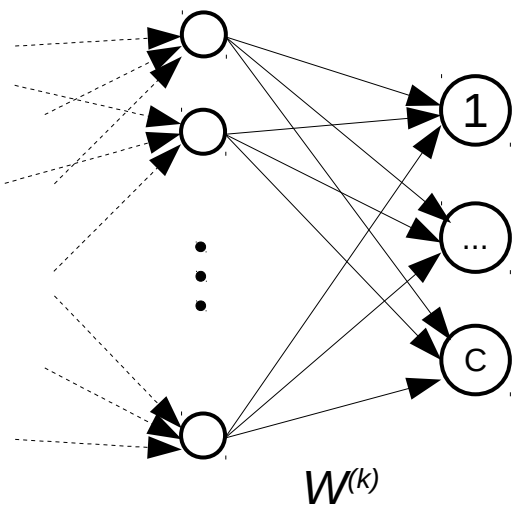
2. On converti y sous la forme d'un vecteur one-hot $\mathbf{y} \in \mathbb{R}^C$, possédant la valeur 1 à l'index correspondant à y, et les valeurs 0 autrement:
$$y = 0 \mapsto \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad y = 1 \mapsto \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

3. La couche de sortie sera donnée un vecteur $\hat{\mathbf{y}}$ obtenu en appliquant la fonction d'activation softmax aux valeurs $\mathbf{a} \in \mathbb{R}^C$ propagées par les couches précédentes:
$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_C \end{bmatrix} \qquad \text{avec } \hat{y}_i = \text{softmax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{C} e^{a_j}}$$

   On interprète le vecteur $\mathbf{y}$ comme une distribution de probabilité sur les classes.

4. Lors de l'optimisation du réseau, on minimise la perte du négatif log-vraisemblance:
$$L_{\text{nlv}}(\hat{\mathbf{y}}, y) = -\ln(\hat{y}_y),$$

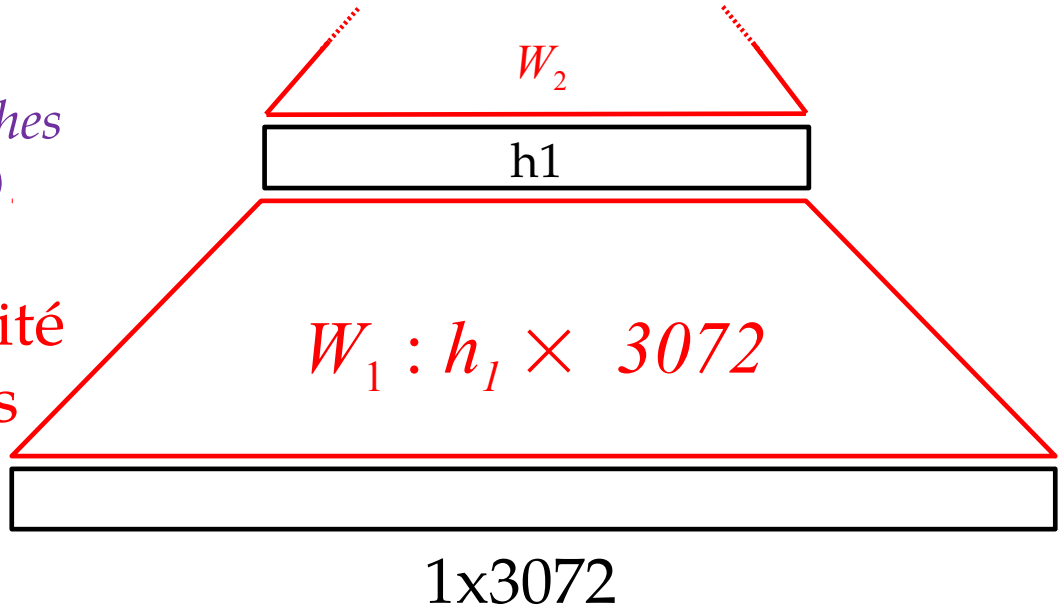
   où $\hat{y}_y$ corresponds à la sortie d'index y du réseau (la probabilité associée à la classe de l'exemple).



Réseau pleinement connecté
(*fully connected*)

Aussi appelé perception multi-couches
(multi-layers perceptron – MLP)

grande quantité
de paramètres



Vectorisation
(*flatten*) de l'image

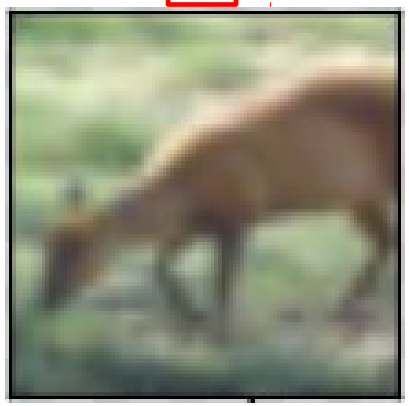


Image 32x32x3

- Vectorisation détruit :
  - relations spatiales
  - canaux de couleurs

La connaissance du problème influence la conception de l'architecture

- Forte corrélation locale dans les valeurs des pixels
  - structure 2D

Convolution $*$

- Opération mathématique très utilisée :
  - Traitement de signal
  - Probabilités (somme de 2 variables aléatoires)

$$(I * F)(i,j) = (F * I)(i,j) = \sum_m \sum_n F(m,n) I(i - m, j - n)$$

- Au sens strict, les réseaux utilisent plutôt la corrélation croisée $\star$

$$(F \star I)(i,j) = \sum_m \sum_n F(m,n) I(i + m, j + n)$$

Exemple « convolution »

(qui est plus une corrélation croisée, mais bon…)

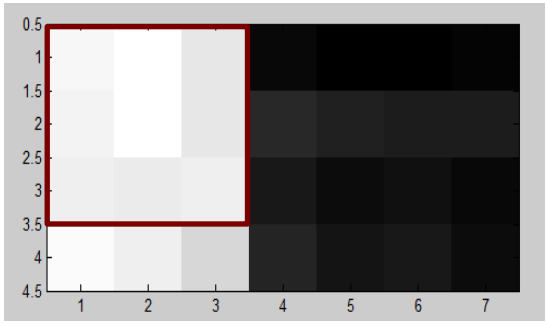
(Appellation de Filtre, *Filter*, ou *Kernel*)

$$F_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

207	210	195	63	57	56	59
204	212	197	82	76	74	75
202	198	202	72	65	67	63
209	201	187	78	69	71	64

$(I*F_1)(x,y)$

	-26	-533	-517	-28	
	-29	-505	-513	-25	

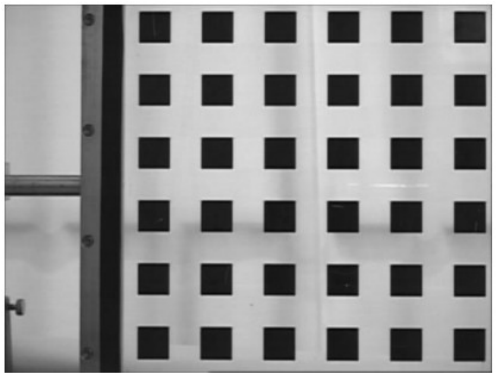


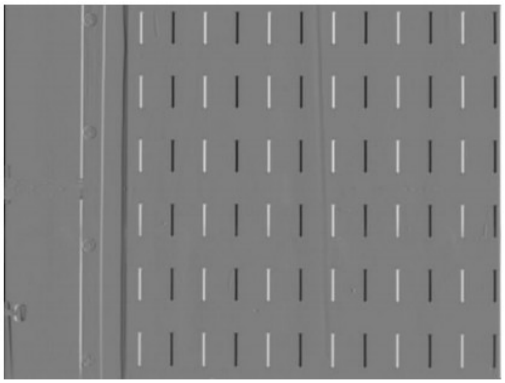
Exemples de filtres calibrés manuellement

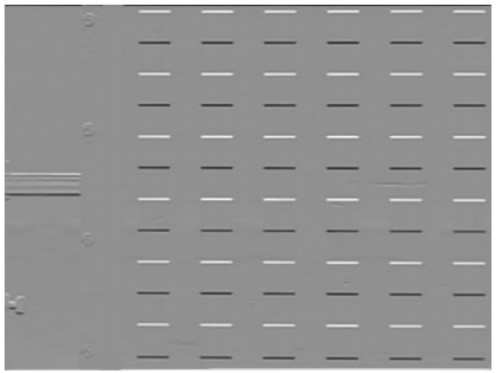
Détection bordure

Bordure verticale $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

Bordure horizontale $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$







Flou

$\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & 1 \\ 1 & 1 & 1 \end{bmatrix}$

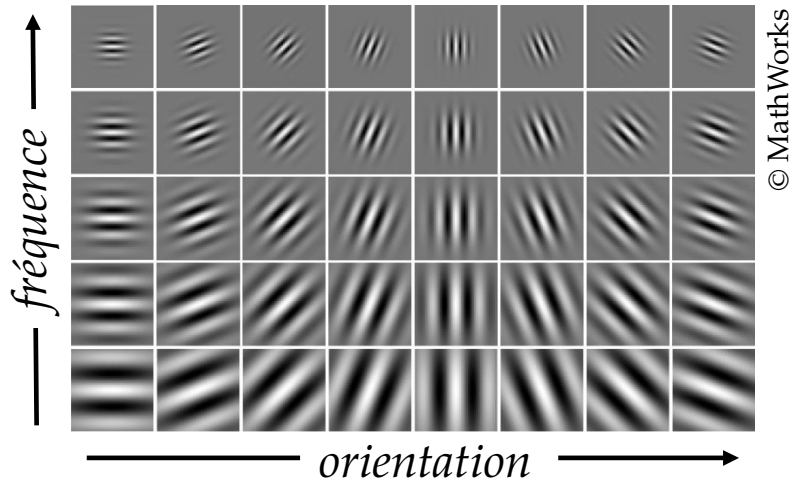
5x5

15x15

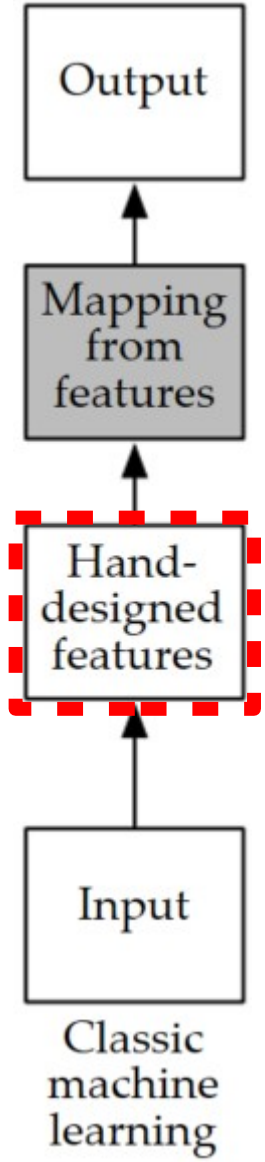
Adapté du cours de Design III de l'Université Laval (P. Hébert, D. Laurendeau)

Filtres

- Vont extraire des *features* de bas niveau
  - Filtres *Gabor* :



  - Filtres de bordure, ondelettes
- Longtemps été un domaine de recherche
  - que concevriez-vous comme filtre pour MNIST ?
- Comme les filtres CNN sont différentiables, le réseau pourra les modifier à sa convenance
  - les ajuster pour maximiser les performances sur les données d'entraînement

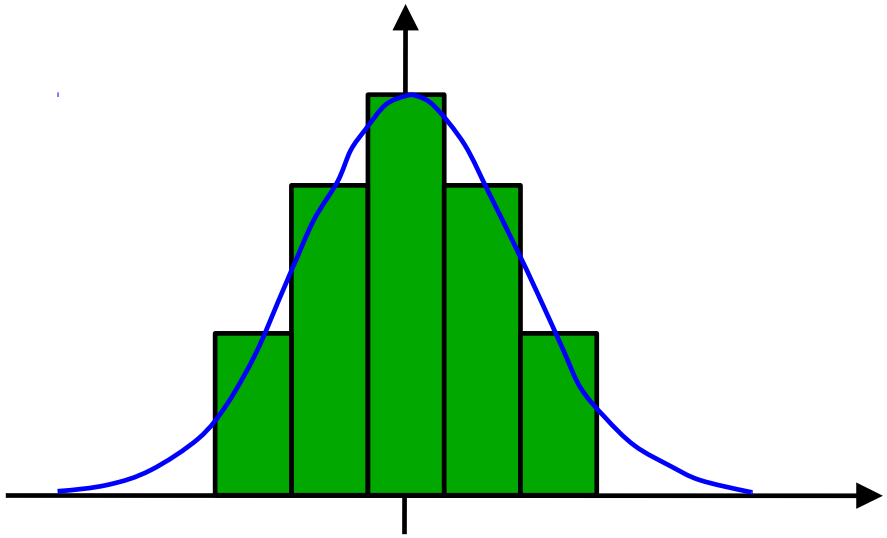


Exemple convolution

http://setosa.io/ev/image-kernels/

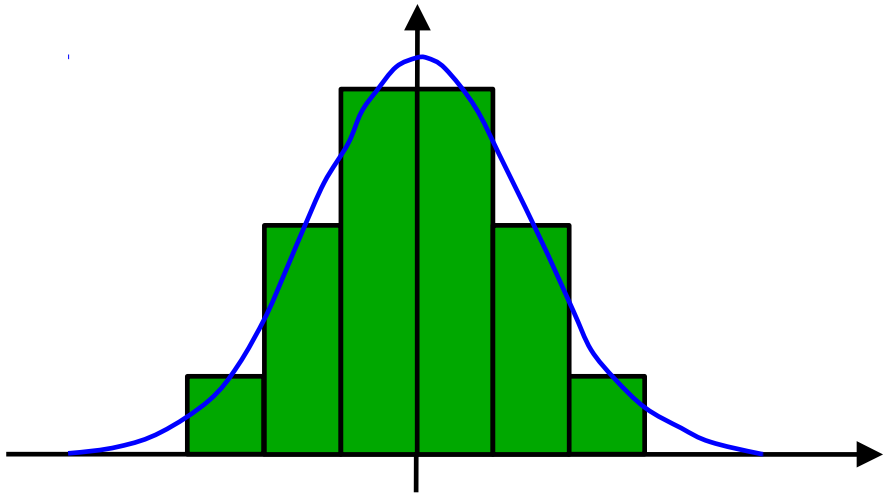
Pourquoi filtre taille impaire

- Pas de pixel « milieu » pour filtre taille paire
- Exemple : filtre radialement symétrique



filtre taille 5

La résultante est imputée à un seul pixel de l'image en sortie

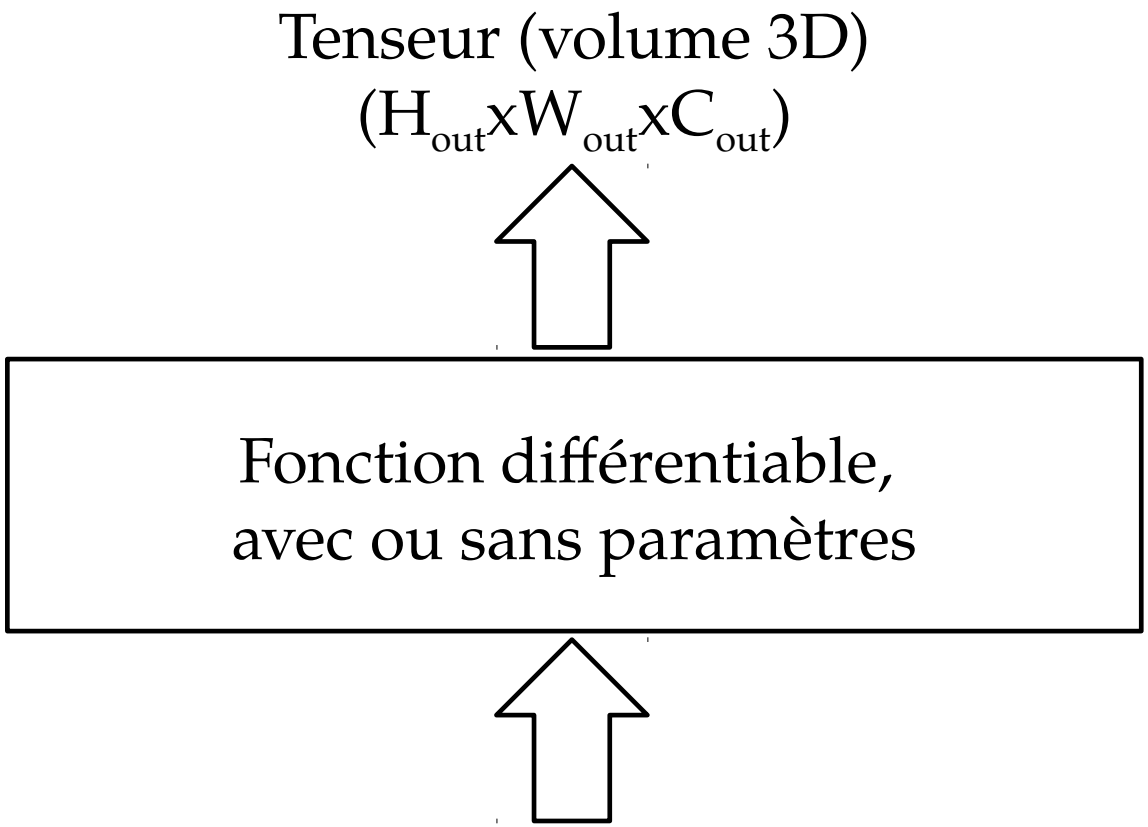


filtre taille 6

La résultante tombe à cheval entre des pixels de l'image en sortie (aliasing)

(à titre informatif)

CNN : couche typique

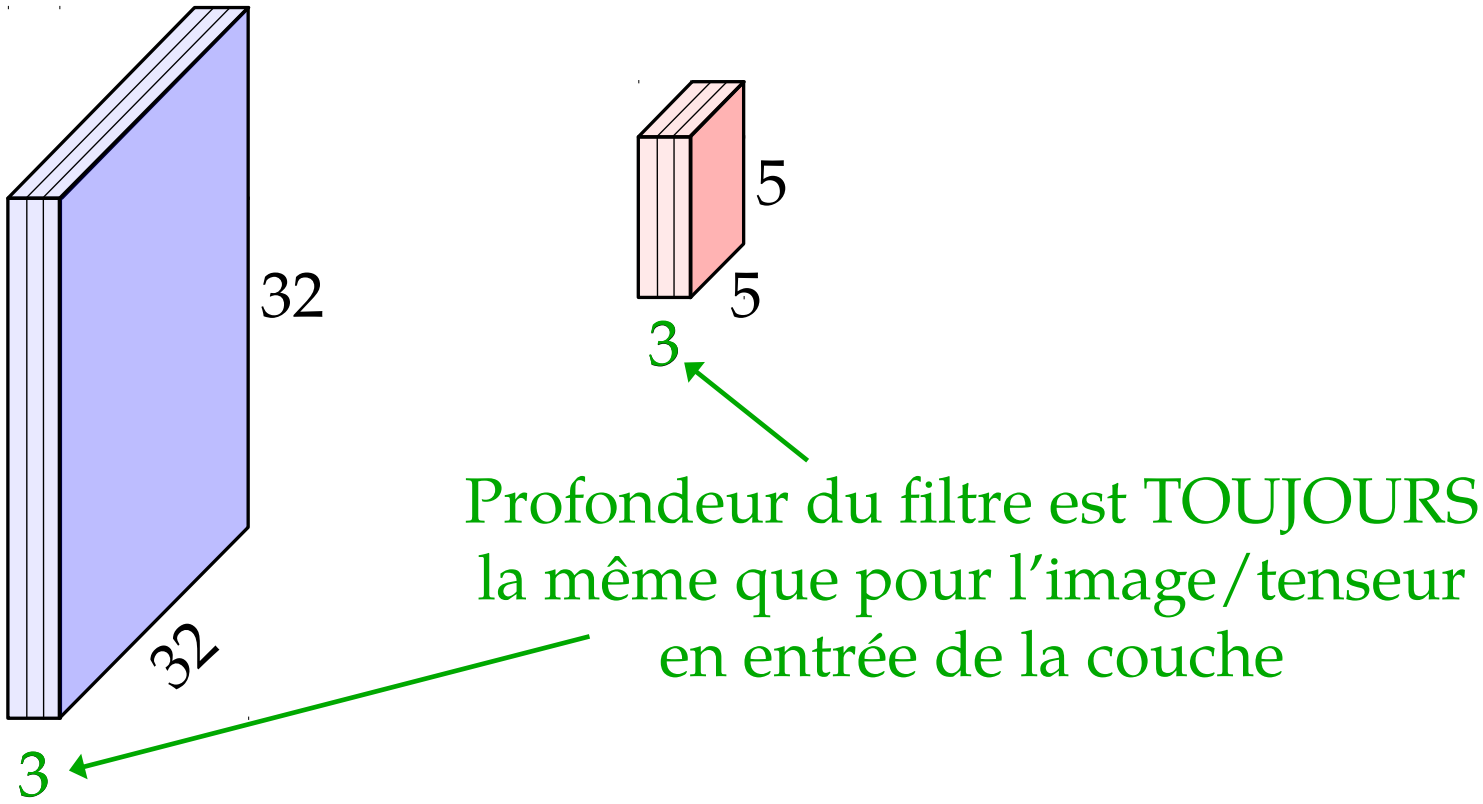


Principaux types de couche

- Pleinement connecté (*Fully-Connected)*
  ⇒ vous connaissez déjà
- Convolutive
- Pooling
  - MaxPooling
  - Average Pooling
  - Stochastic Pooling
  - …

Filtres convolutifs

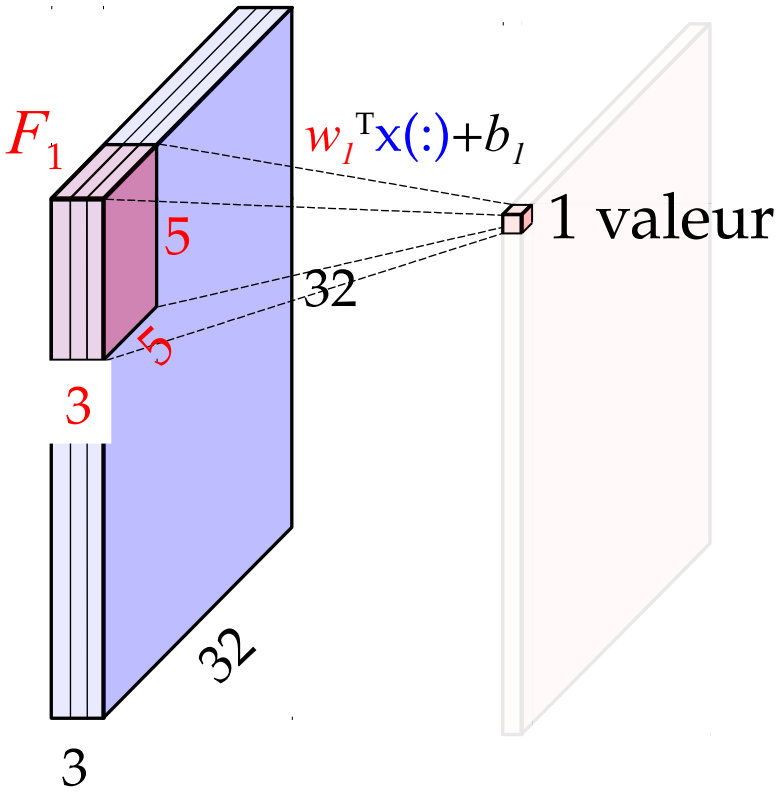
- Conserver la structure spatiale/couleur/*feature* de l'entrée



adapté de : cs231n, Université Stanford

Filtres convolutifs

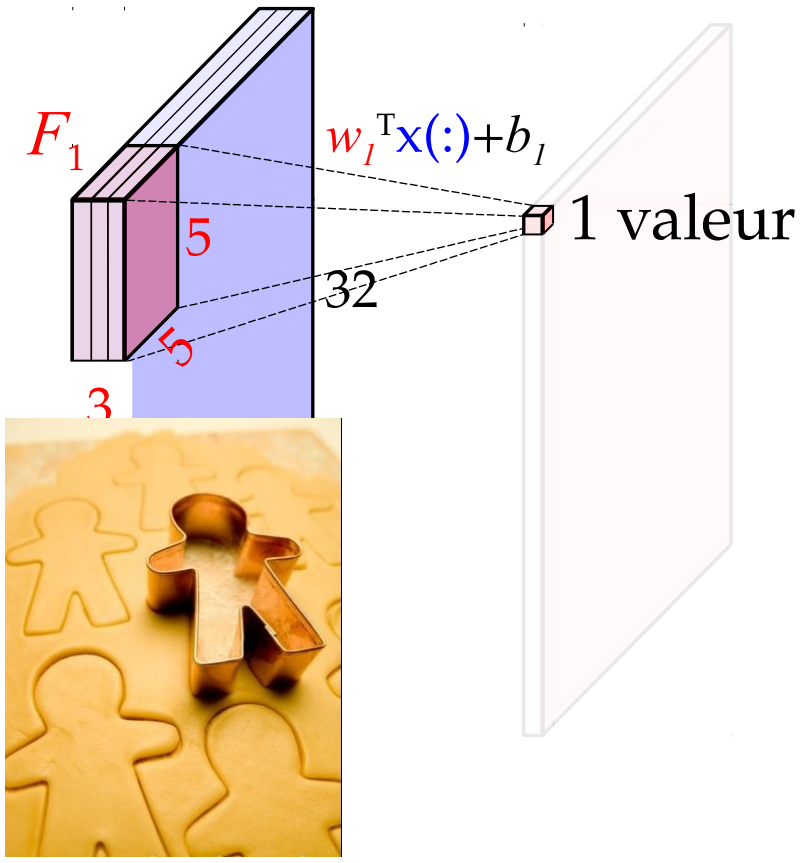
- Glisser spatialement le filtre F sur l'image, en calculant produit scalaire à chaque endroit de x



adapté de : cs231n, Université Stanford

Filtres convolutifs

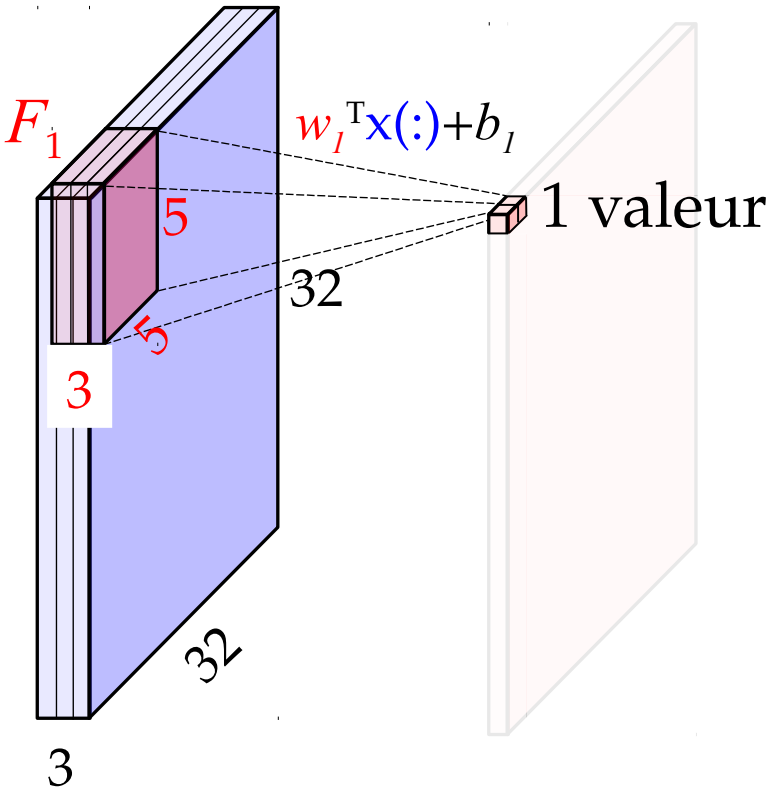
- Glisser spatialement le filtre F sur l'image, en calculant produit scalaire à chaque endroit de x



adapté de : cs231n, Université Stanford

Filtres convolutifs

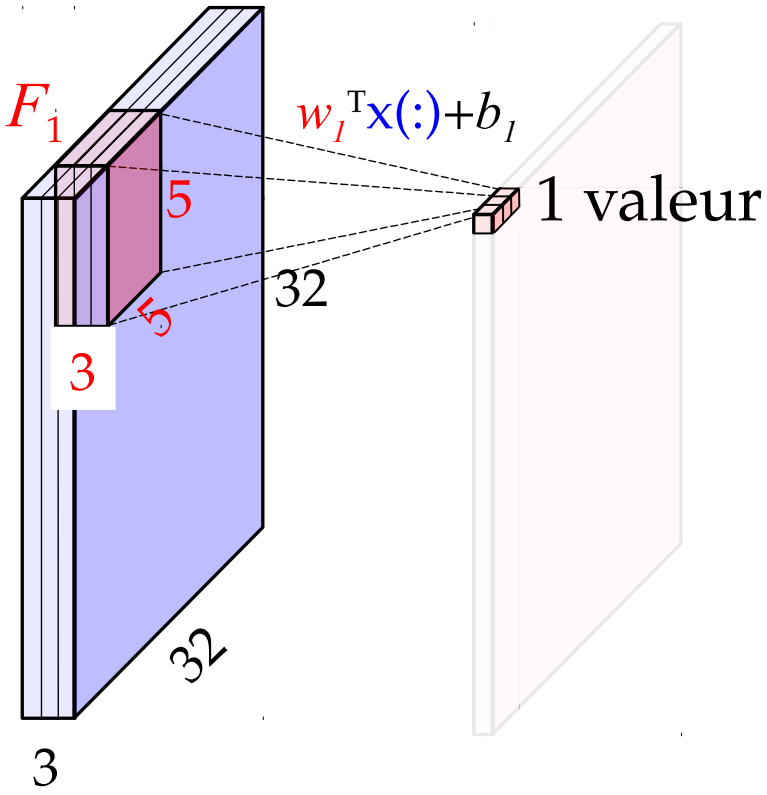
- Glisser spatialement le filtre F sur l'image, en calculant produit scalaire à chaque endroit de x



adapté de : cs231n, Université Stanford

Filtres convolutifs

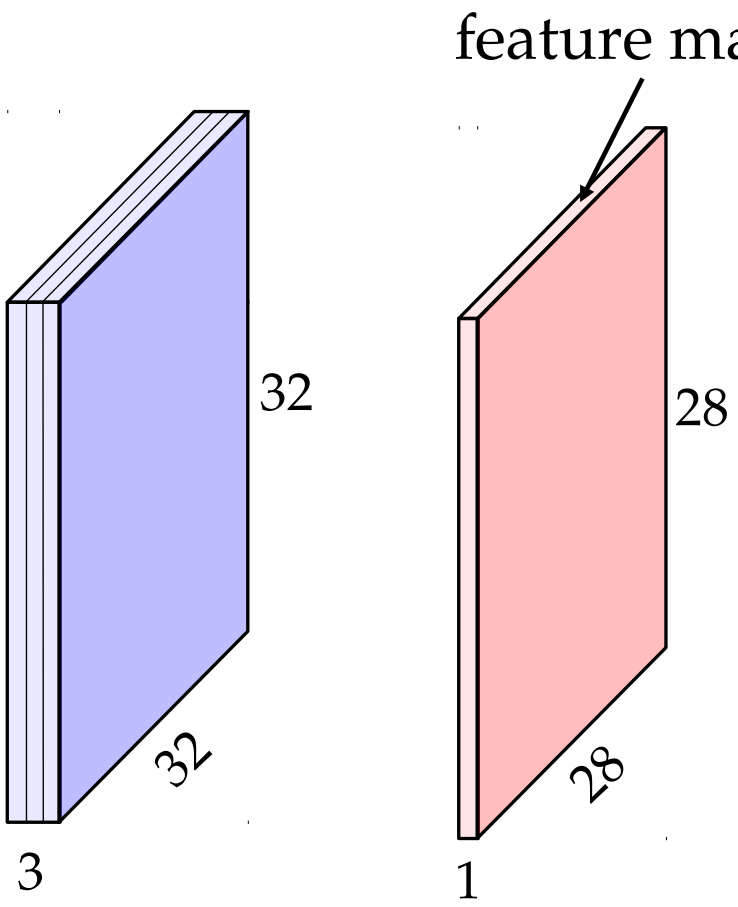
- Glisser spatialement le filtre F sur l'image, en calculant produit scalaire à chaque endroit de x



adapté de : cs231n, Université Stanford

Filtres convolutifs

- Sortie : *feature map*

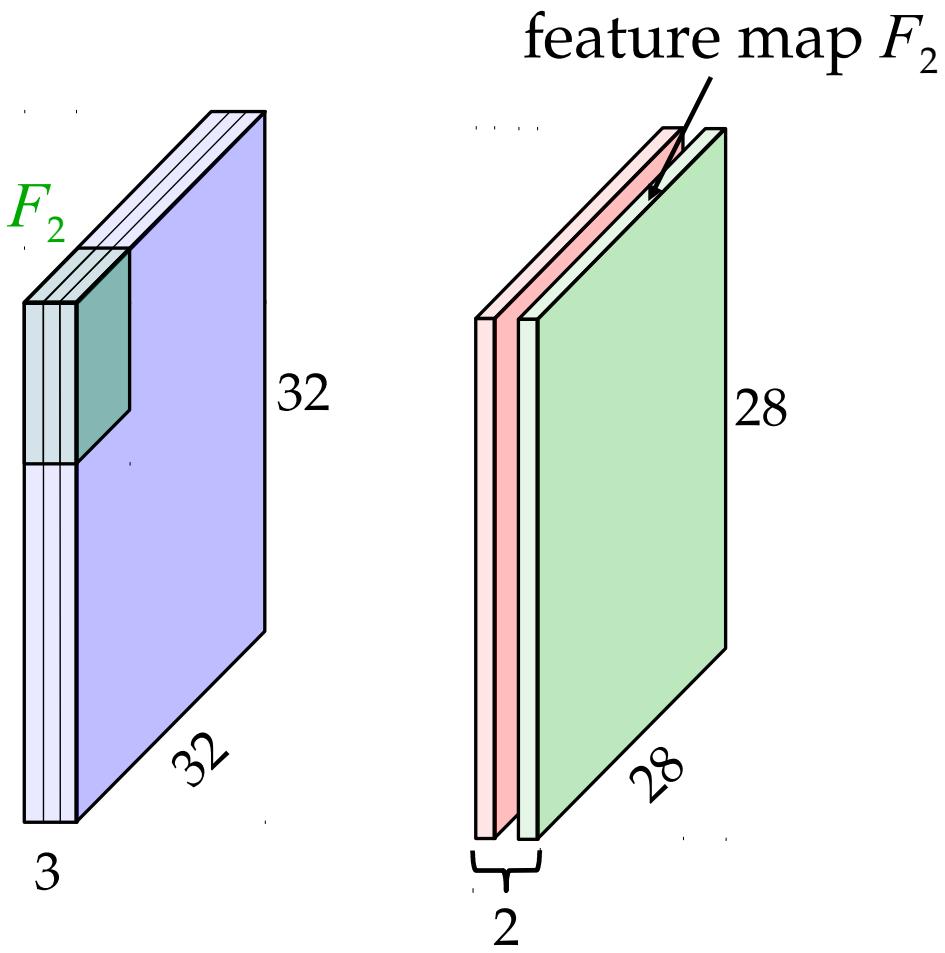


« combien présent est le feature F_1 à cet endroit? »

adapté de : cs231n, Université Stanford

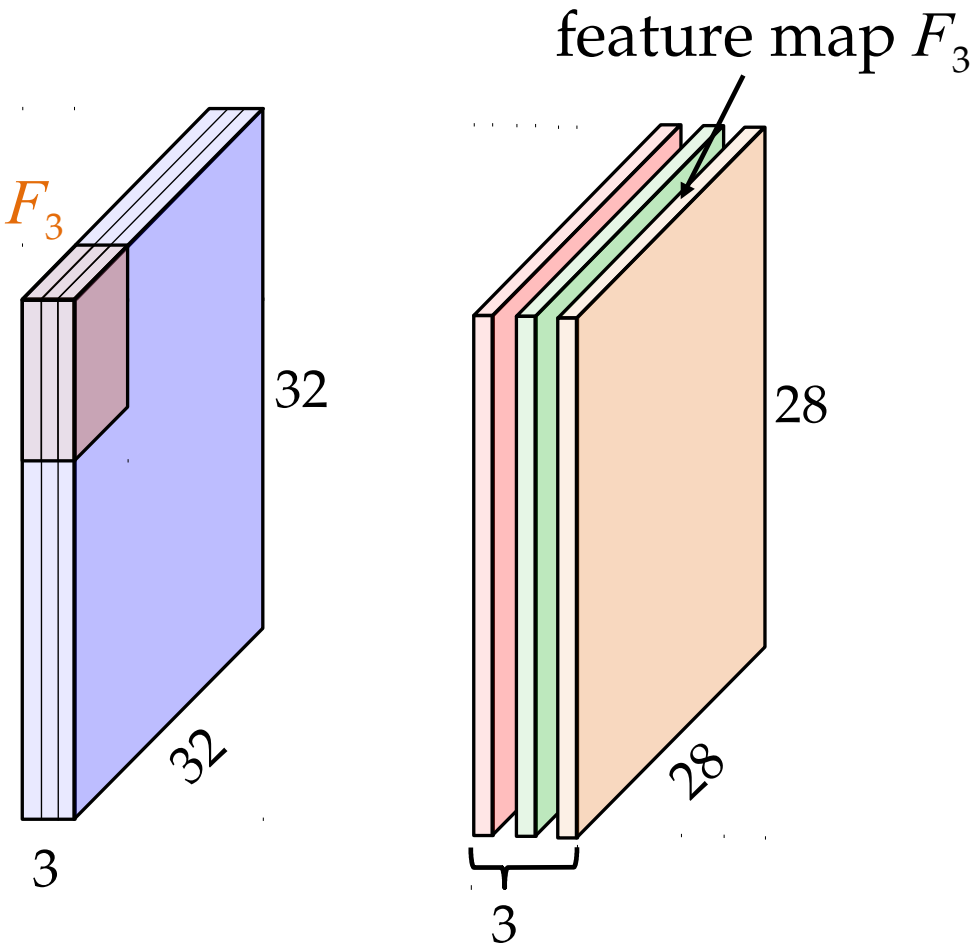
(Note : la taille de sortie est différente ici…)

Banque de filtres convolutifs



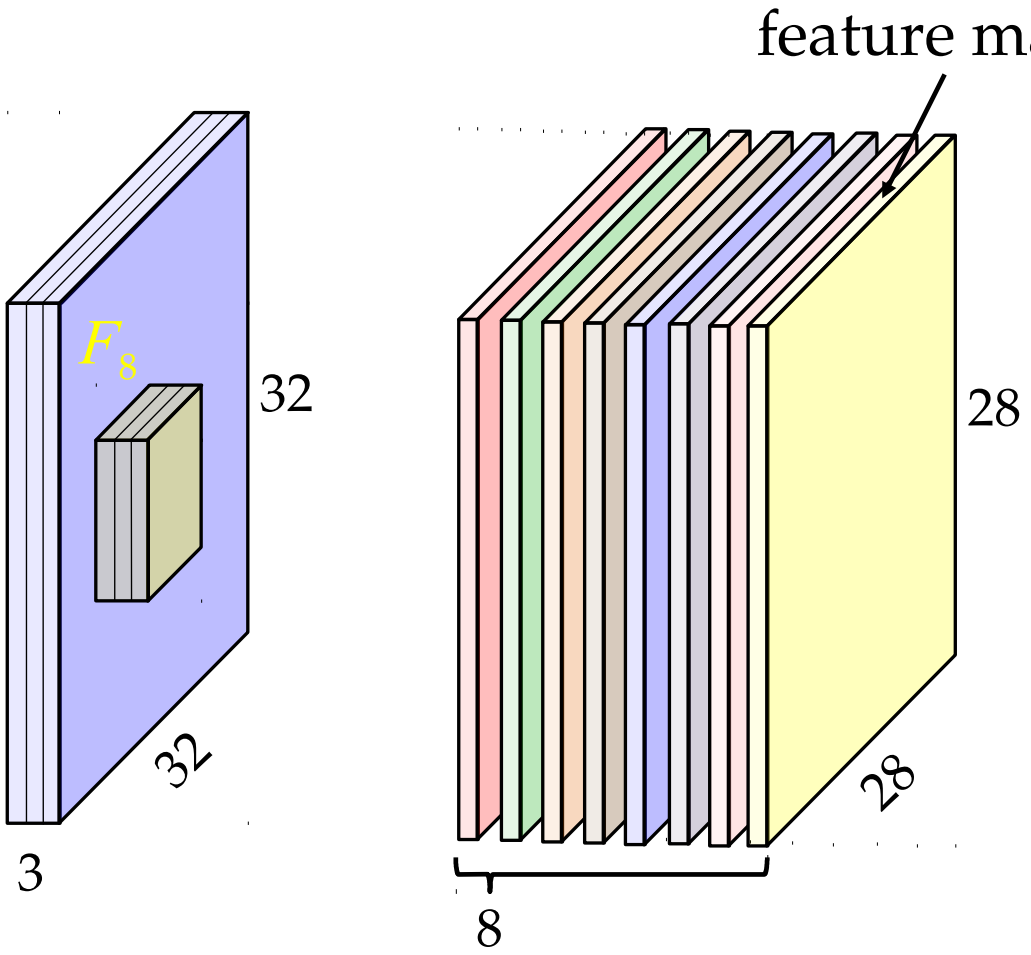
adapté de : cs231n, Université Stanford

Banque de filtres convolutifs



adapté de : cs231n, Université Stanford

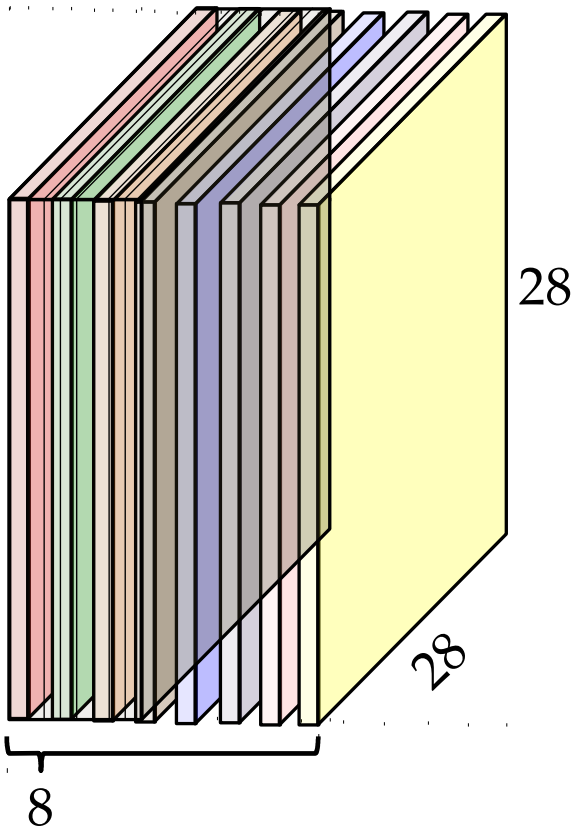
Banque de filtres convolutifs



adapté de : cs231n, Université Stanford

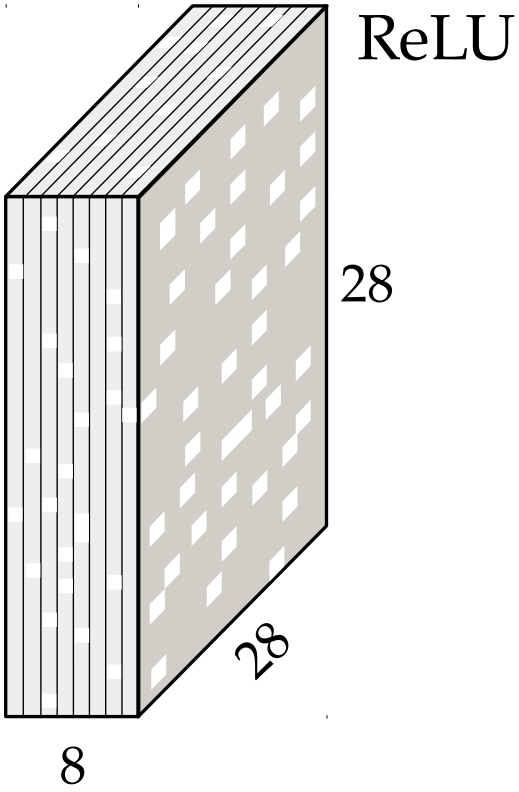
Résultante

Tenseur ordre 3

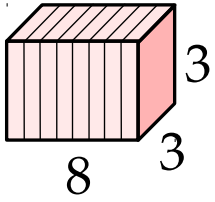


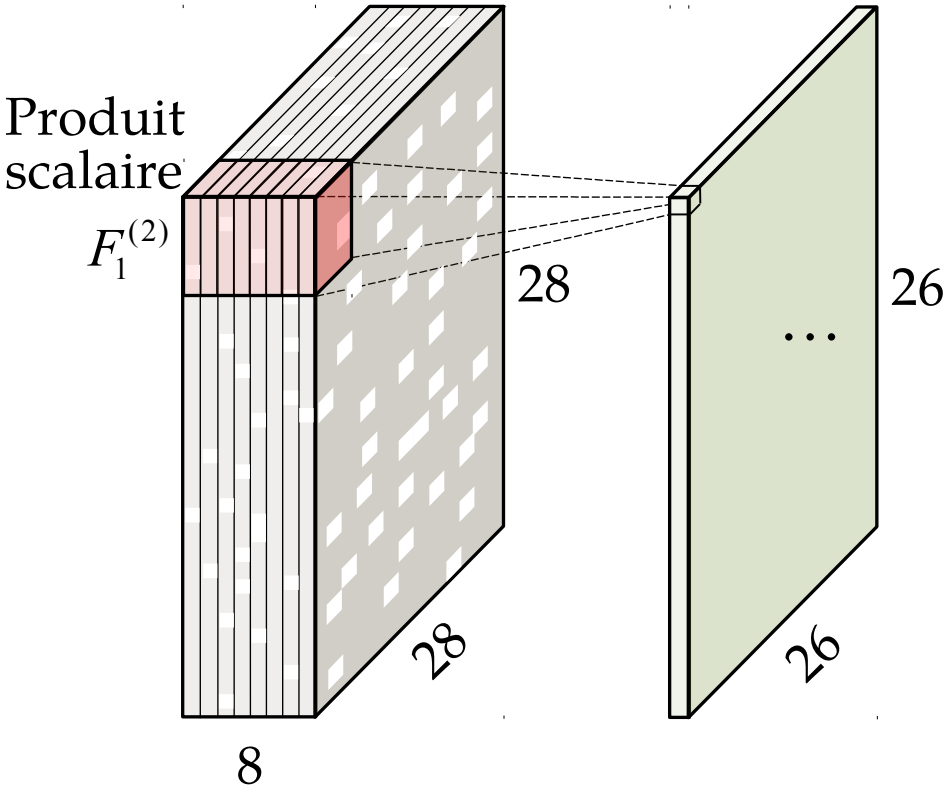
Résultante

- Applique la non-linéarité sur chaque entrée, individuellement
- Appelé *feature activation map*

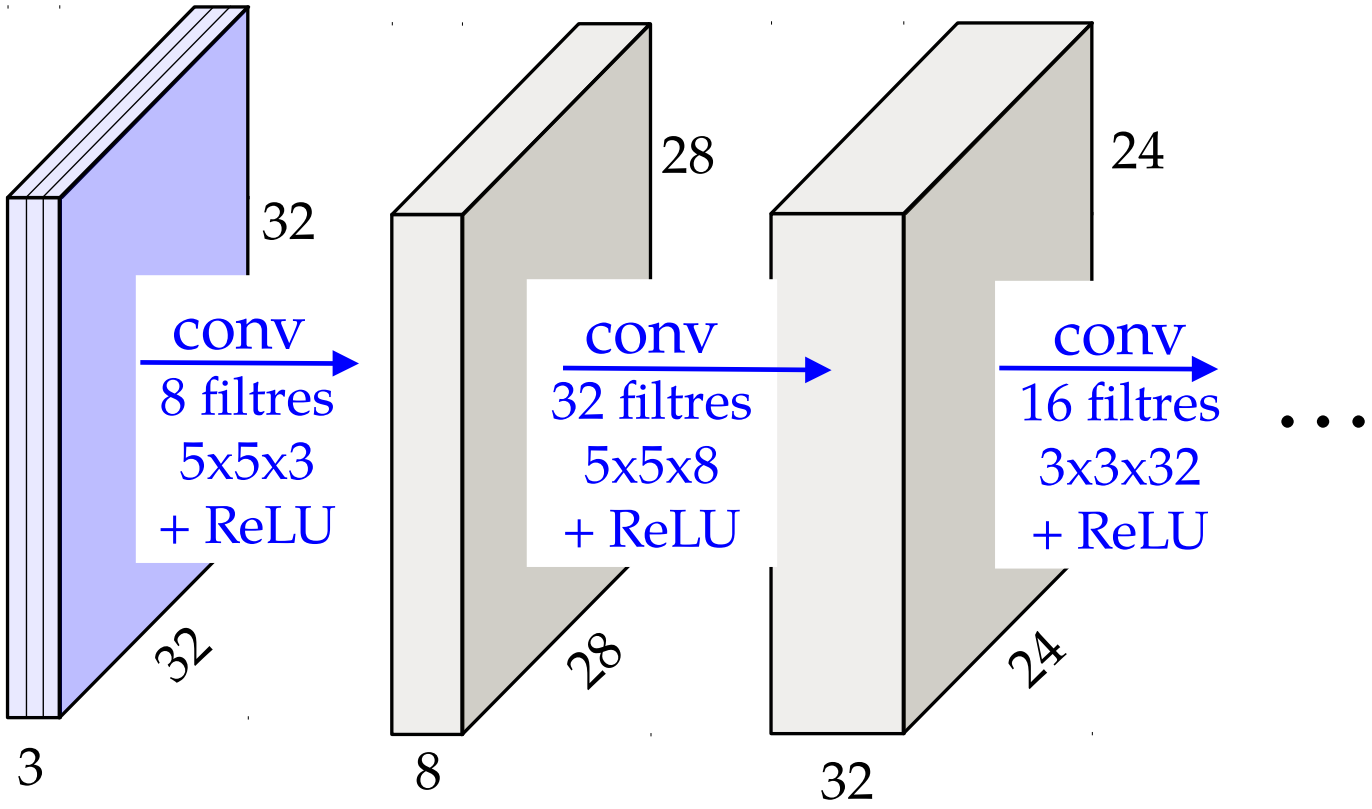


On recommence, avec une banque de filtres différents



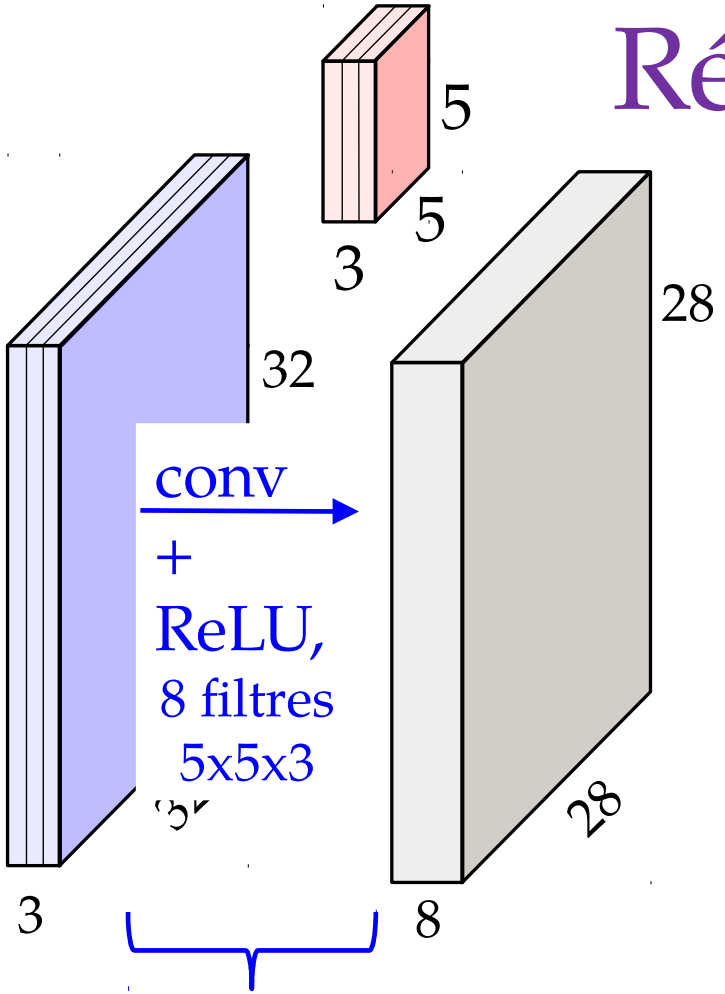


Idée de base



adapté de : cs231n, Université Stanford

Réduction # paramètres



- Perte de capacité d'expression vs. couche pleinement connectée
  - on ne peut pas établir de lien entre les pixels très éloignés
- Réduction importante du nombre de paramètre

Combien de paramètres pour cette couche?

$(5x5x3 + 1) \times 8 = 608$

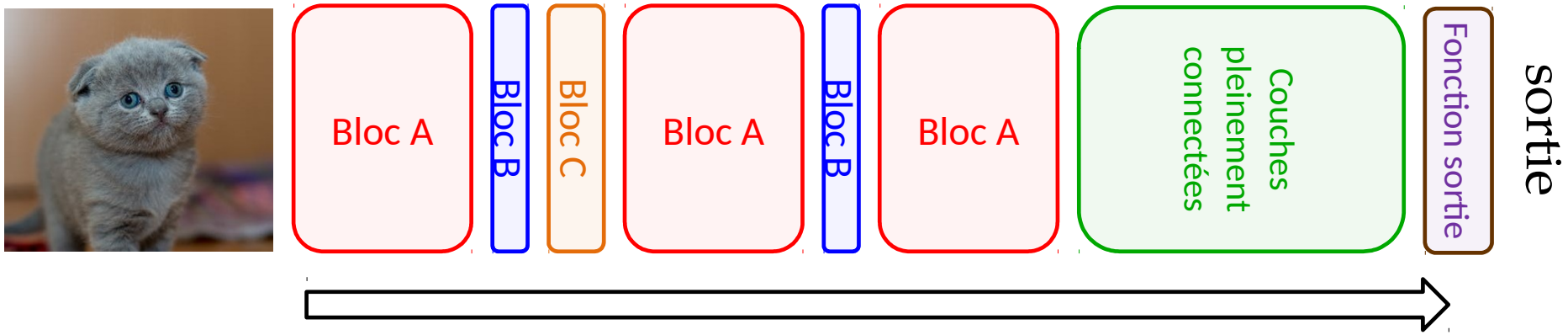
(biais : 1, filtres : 8)

Combien de paramètres pour une couche pleinement connectée ?
28x28x8 = 6272 neurones
32x32x3 + 1 : 3073 param./neurones
Total : 19,273,856 paramètres

adapté de : cs231n, Université Stanford

Approche par bloc

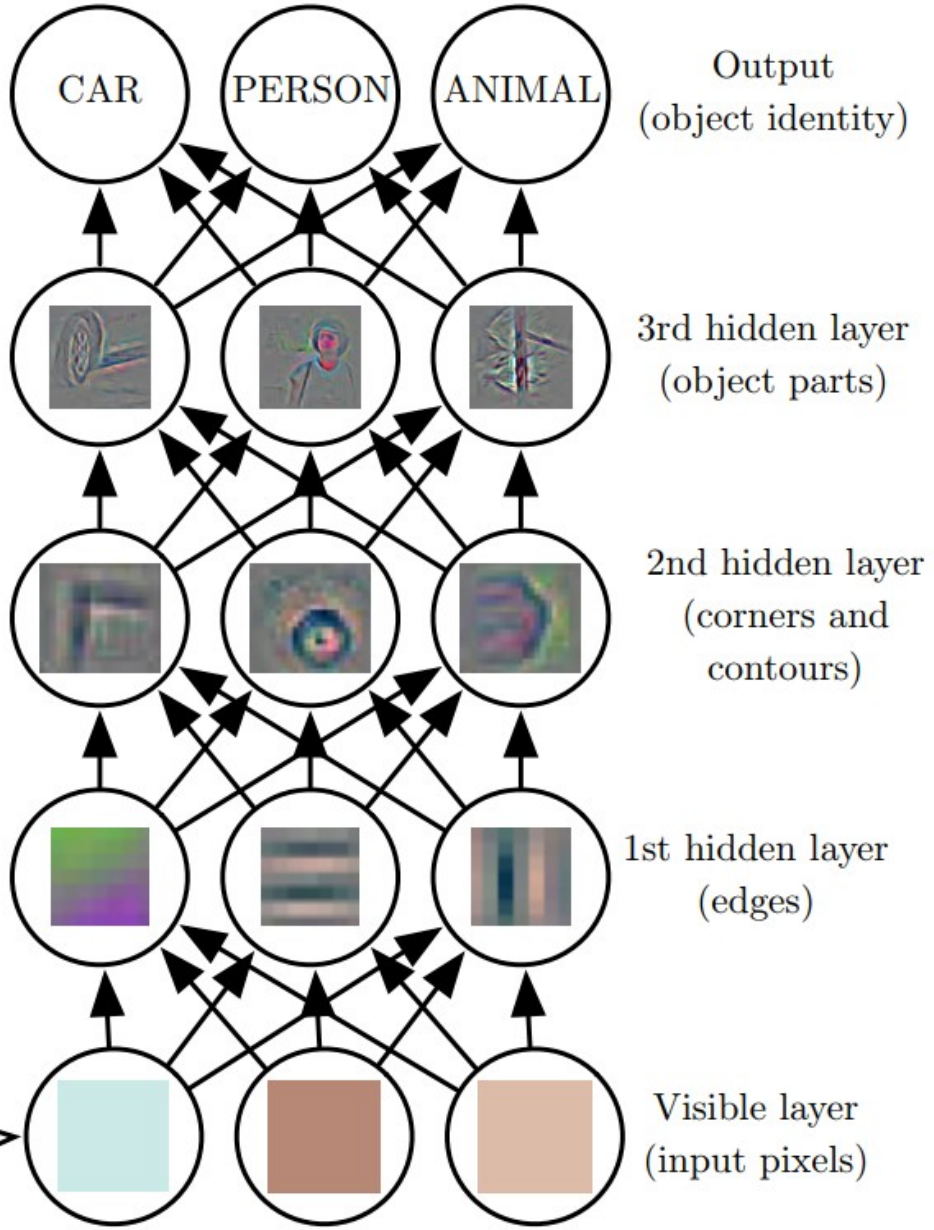
- La plupart des architectures sont organisées par alternance de blocs



- Facile de combiner des blocs de différents types, jouer sur la profondeur, réutiliser des idées
-

Hiérarchie de filtres

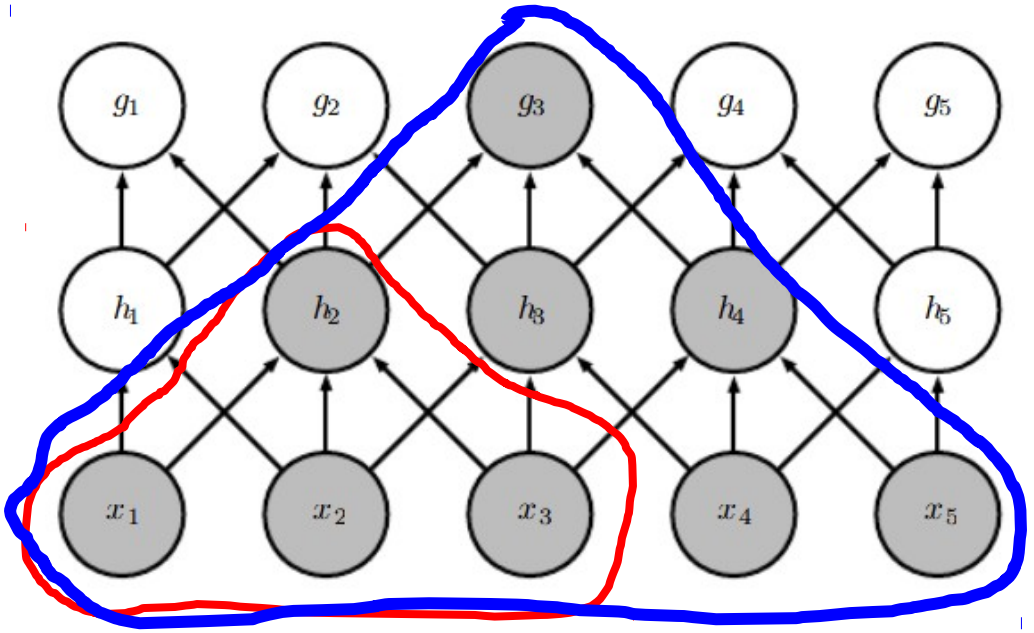
Va établir des liens entre des pixels de plus en plus éloignés



(I. Goodfellow) deeplearningbook.org

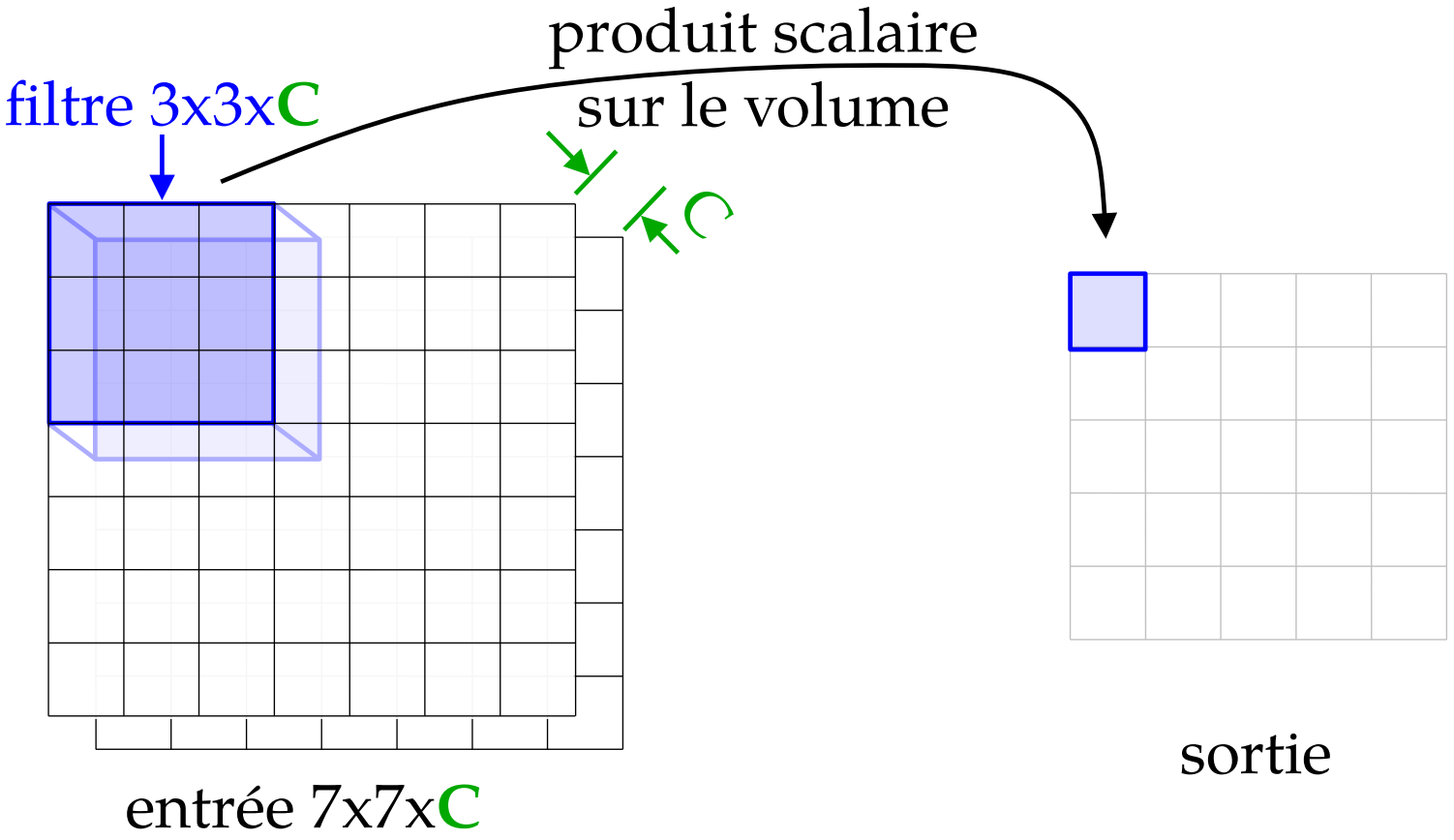
Champ récepteur

- Augmente avec la profondeur
- Les neurones plus haut peuvent prendre des décisions basées sur des champs plus grands

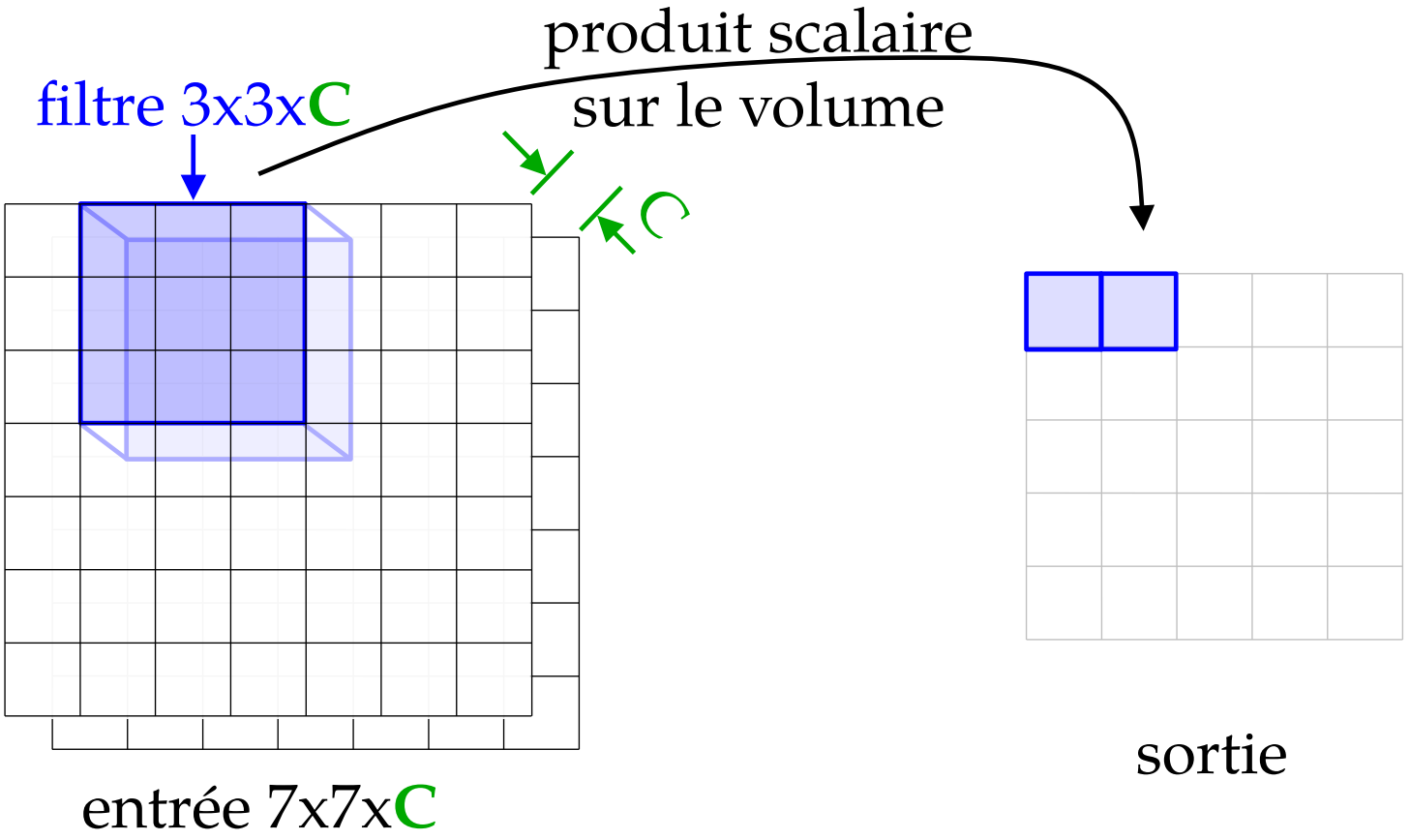


(I. Goodfellow) deeplearningbook.org

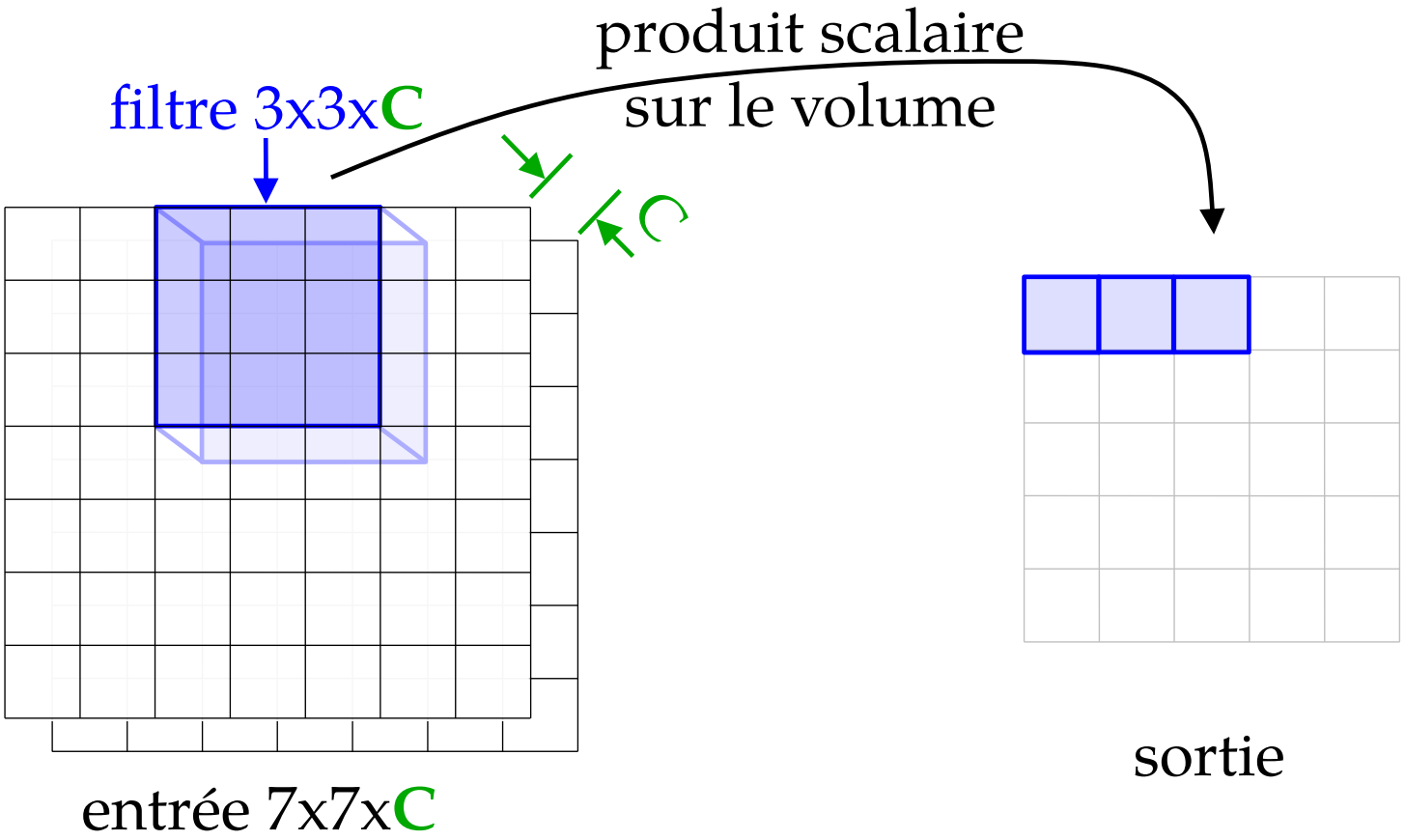
Effet de bord : redimensionalisation



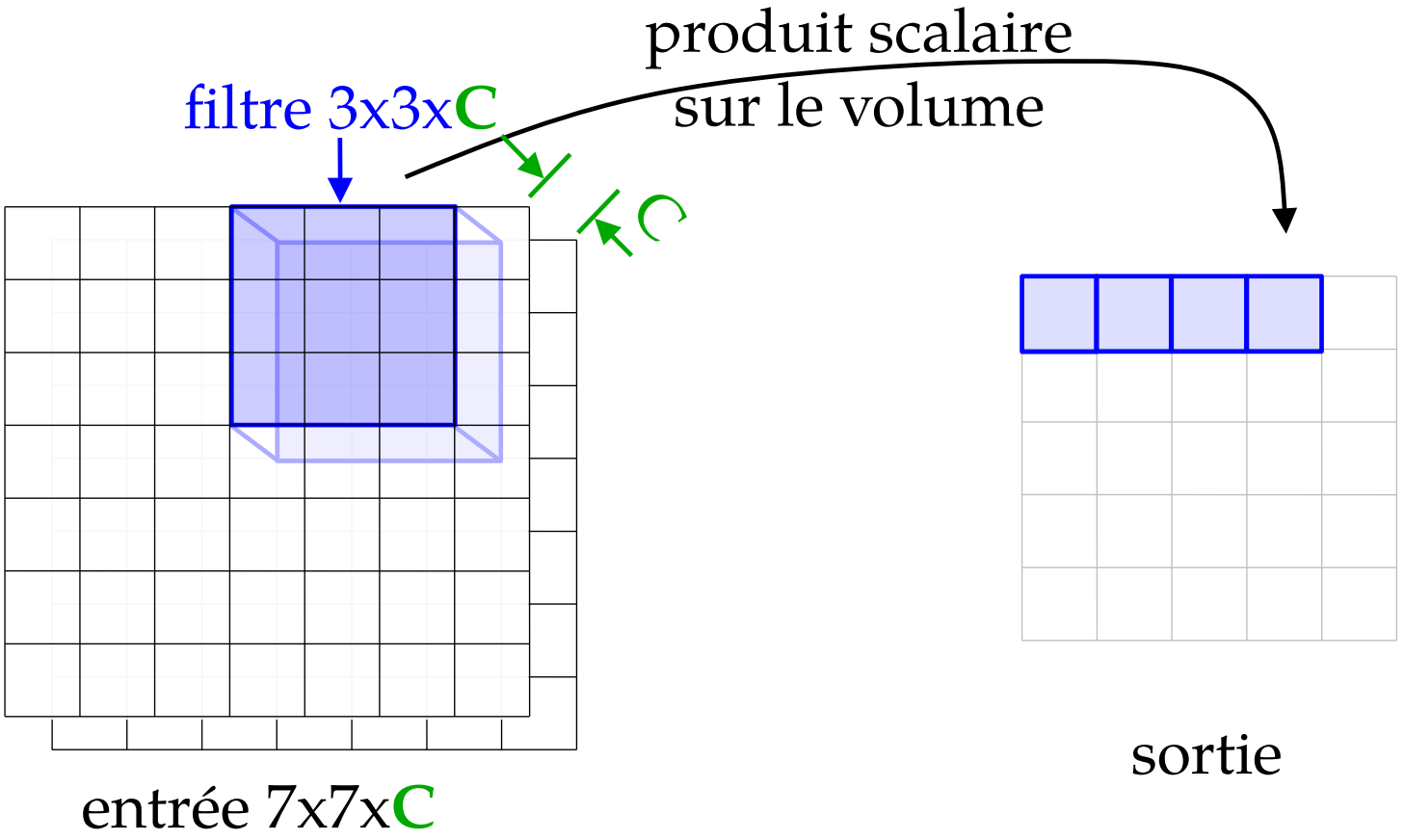
Effet de bord : redimensionalisation



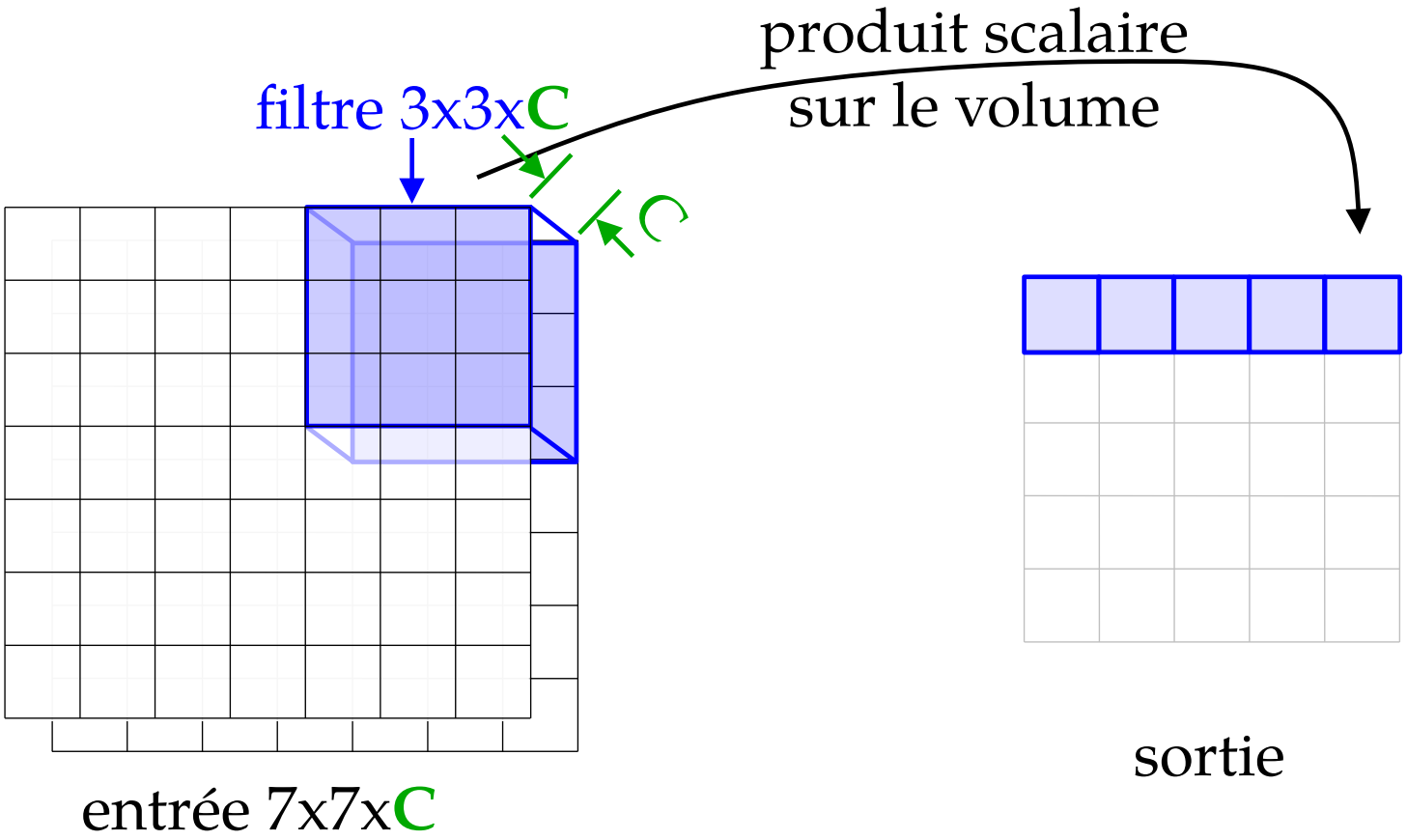
Effet de bord : redimensionalisation



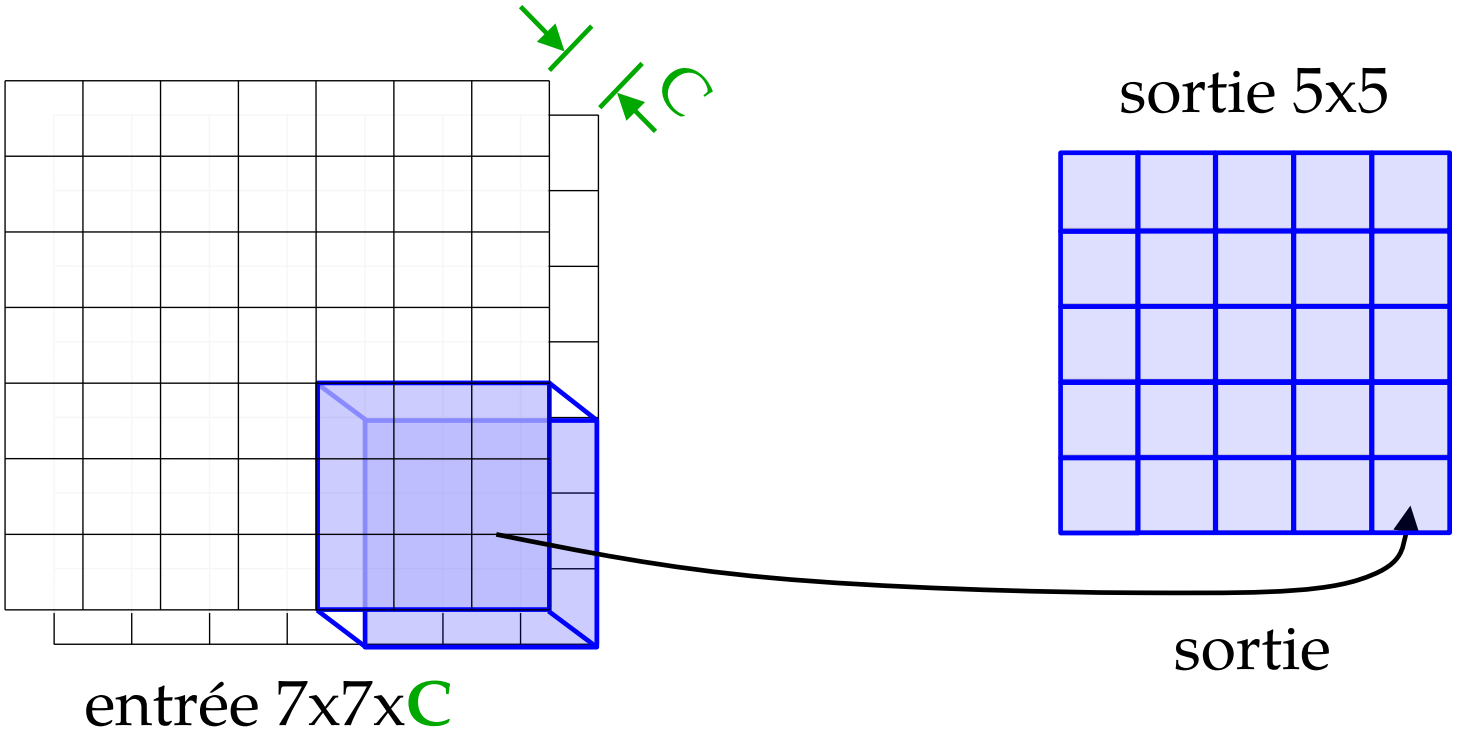
Effet de bord : redimensionalisation



Effet de bord : redimensionalisation

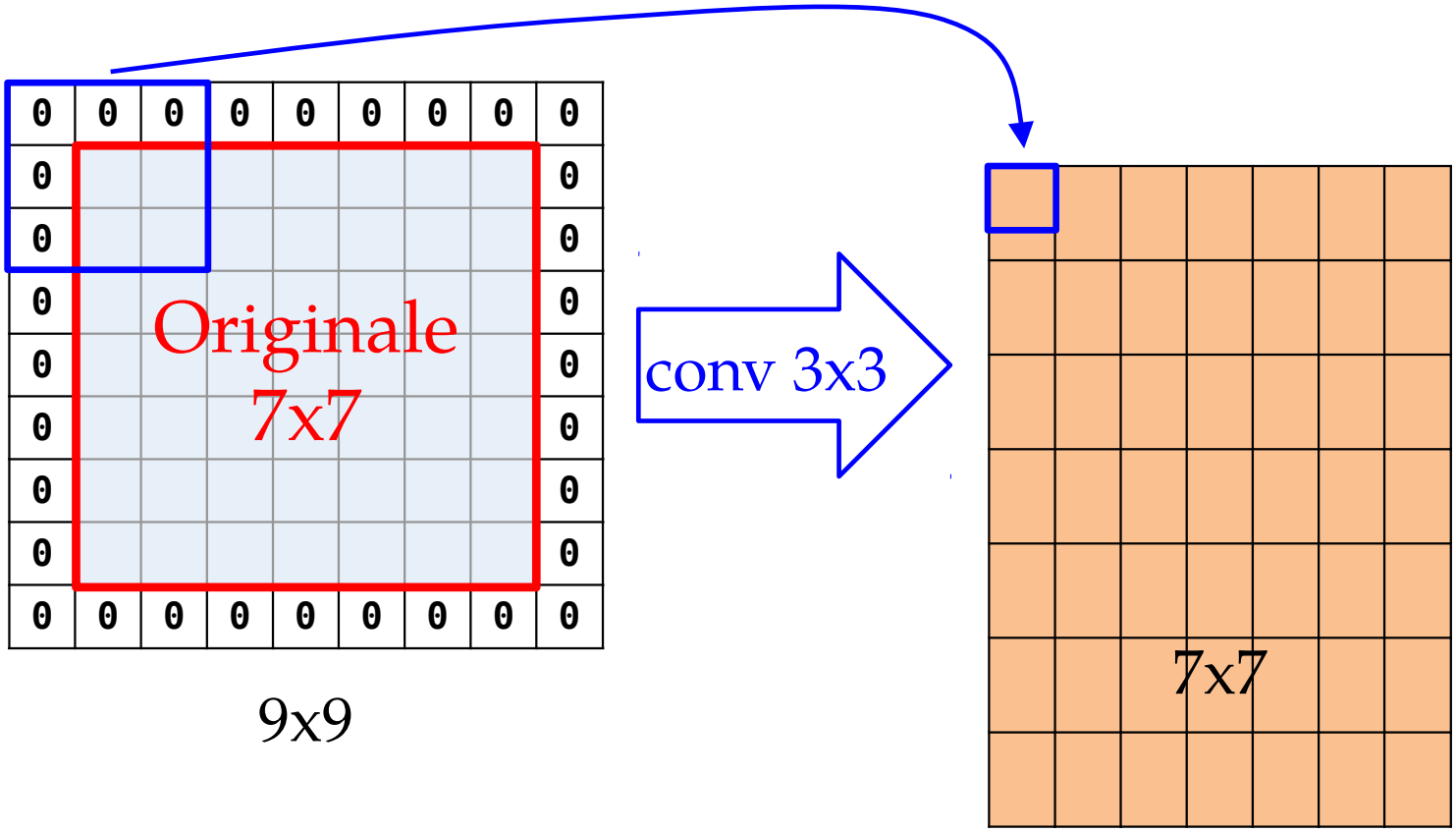


Effet de bord : redimensionalisation



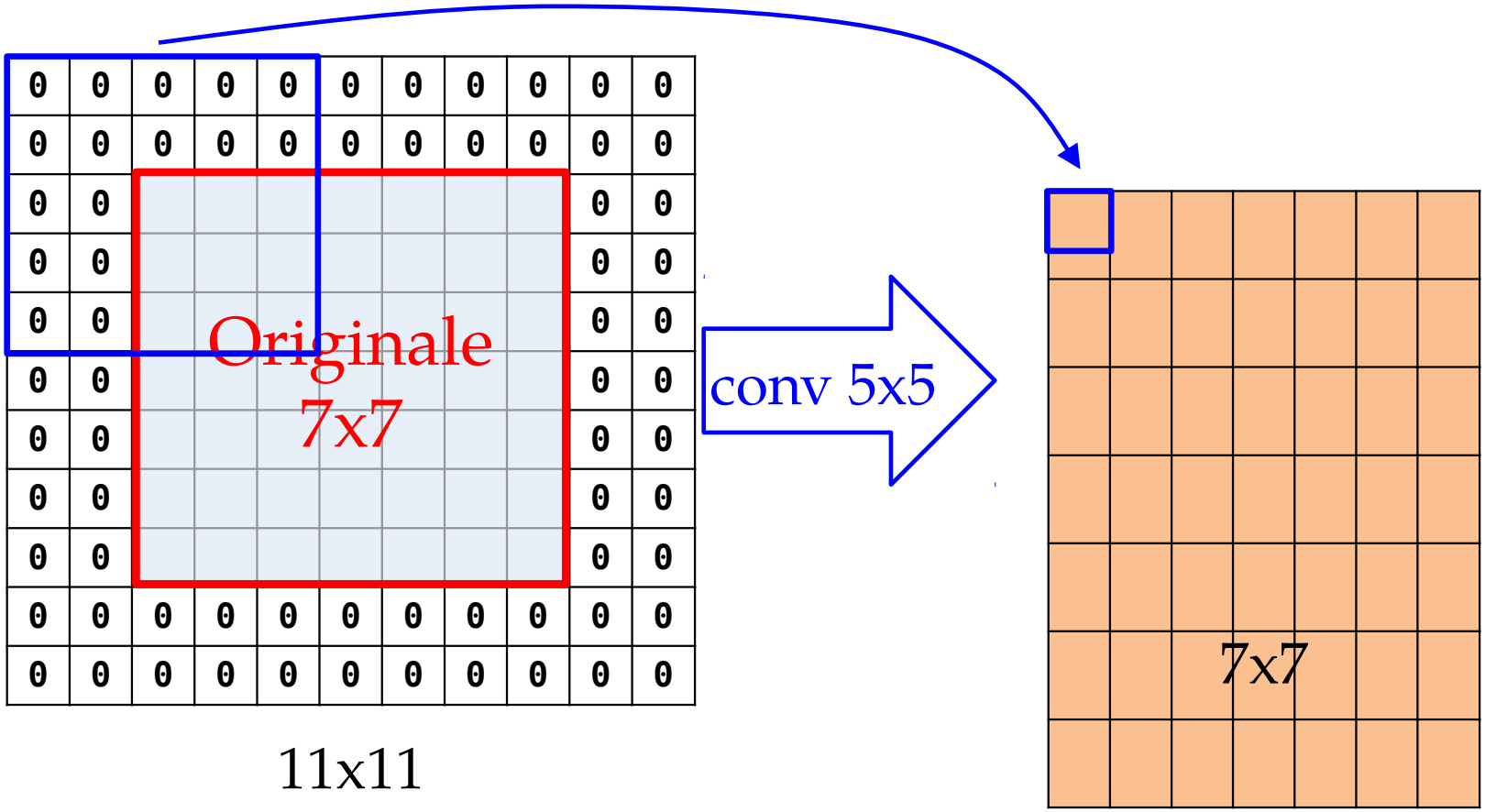
Ajout de zéros : *zero padding*

- Ajoute des entrées à 0 en bordure



(par simplicité, j'ai retiré la 3ème dimension)

Ajout de zéros : *zero padding*

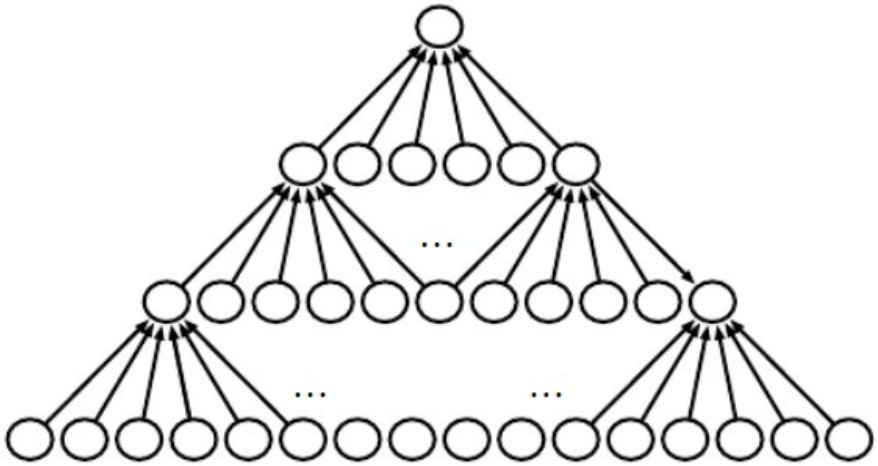


Pour filtre 3x3, padding de largeur 1
Pour filtre 5x5, padding de largeur 2
Pour filtre 7x7, padding de largeur 3

Zero-padding

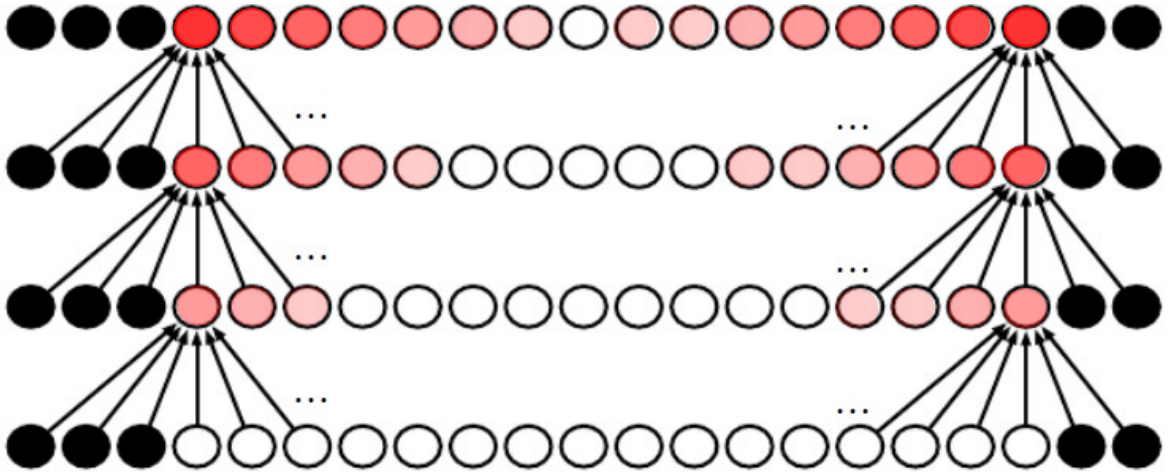
- Conserve la largeur du pipeline
- Attention aux effets de bord, où les entrées sont moins informatives à cause du 0-padding

sans 0-padding



rougeur : impact du 0-padding

avec 0-padding



(ne semble pas trop problématique)

Quelques questions!

- Si j'ai une image de 224x224x3 en entrée, complétez la taille du filtre : 5x5x**?**

Réponse : 5x5x**3**

- Si j'ai 32 filtres 5x5 appliqués sur cette image 224x224x3, sans 0-padding, quelle sont les dimensions du tenseur de sortie?

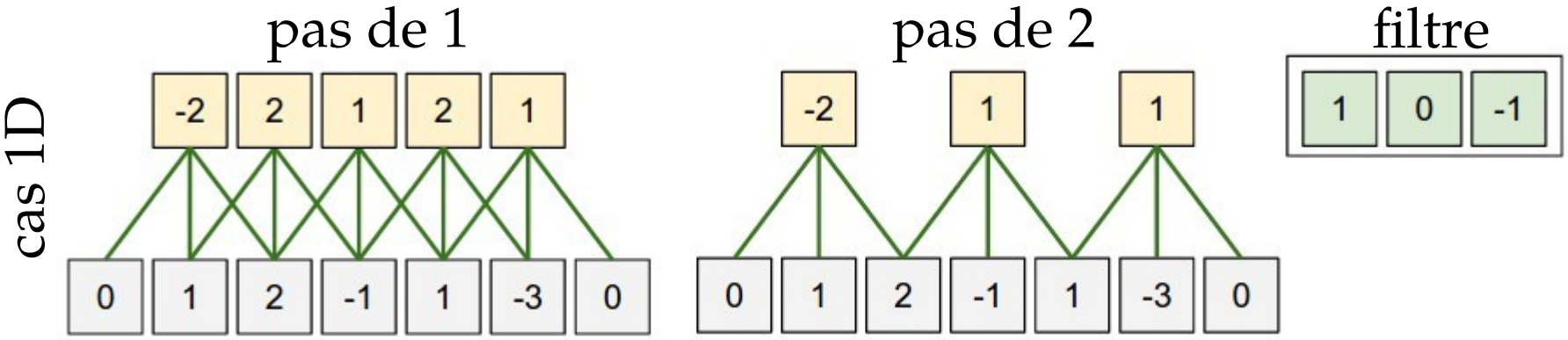
Réponse : 220x220x32

- Quelle est la largeur du 0-padding pour un tenseur de 64x64x3, si on applique un filtre 9x9?

Réponse : largeur de 4

Pas (*stride*)

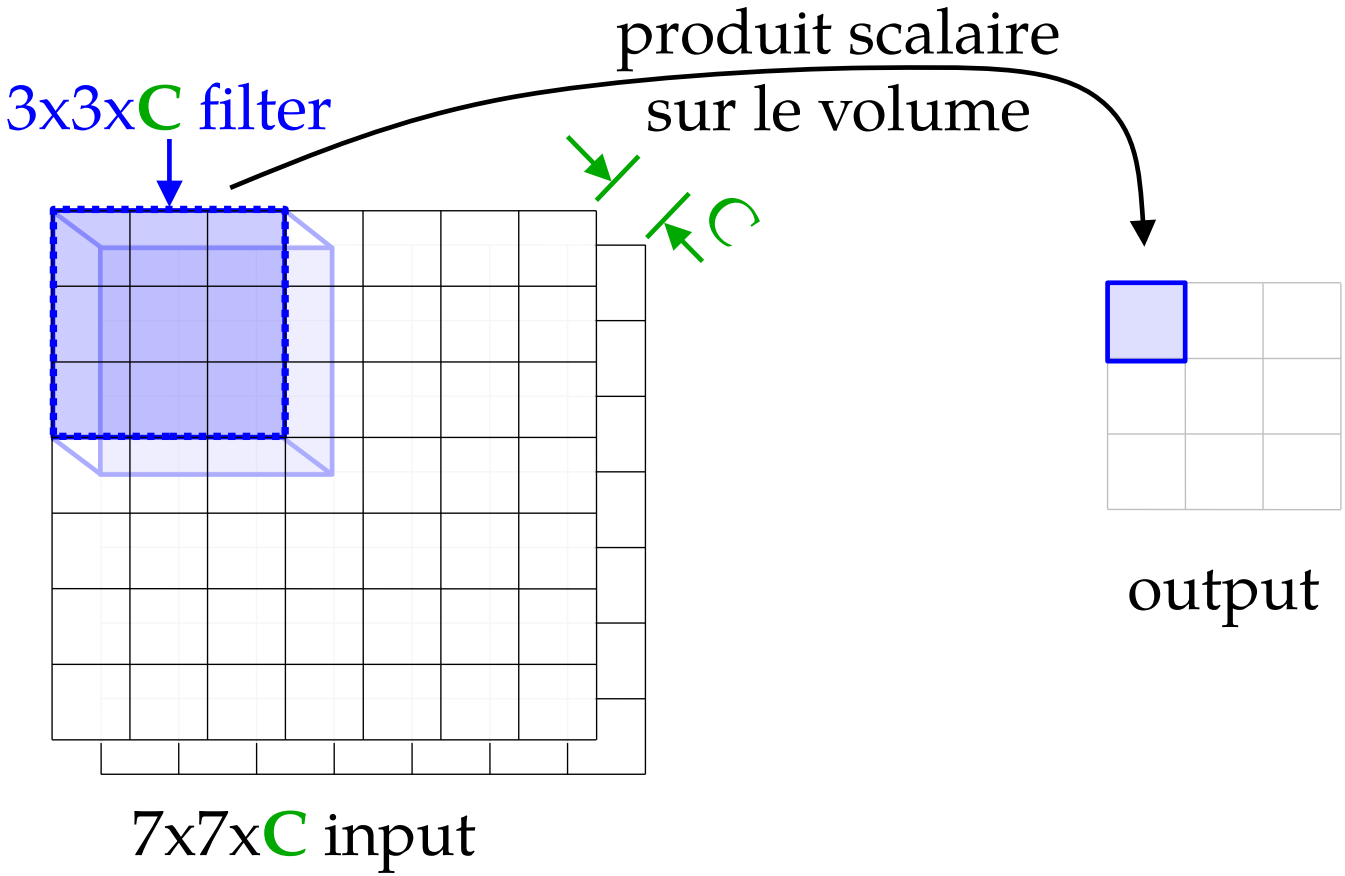
- ***Pas*** (stride) : saut dans l'application de la convolution



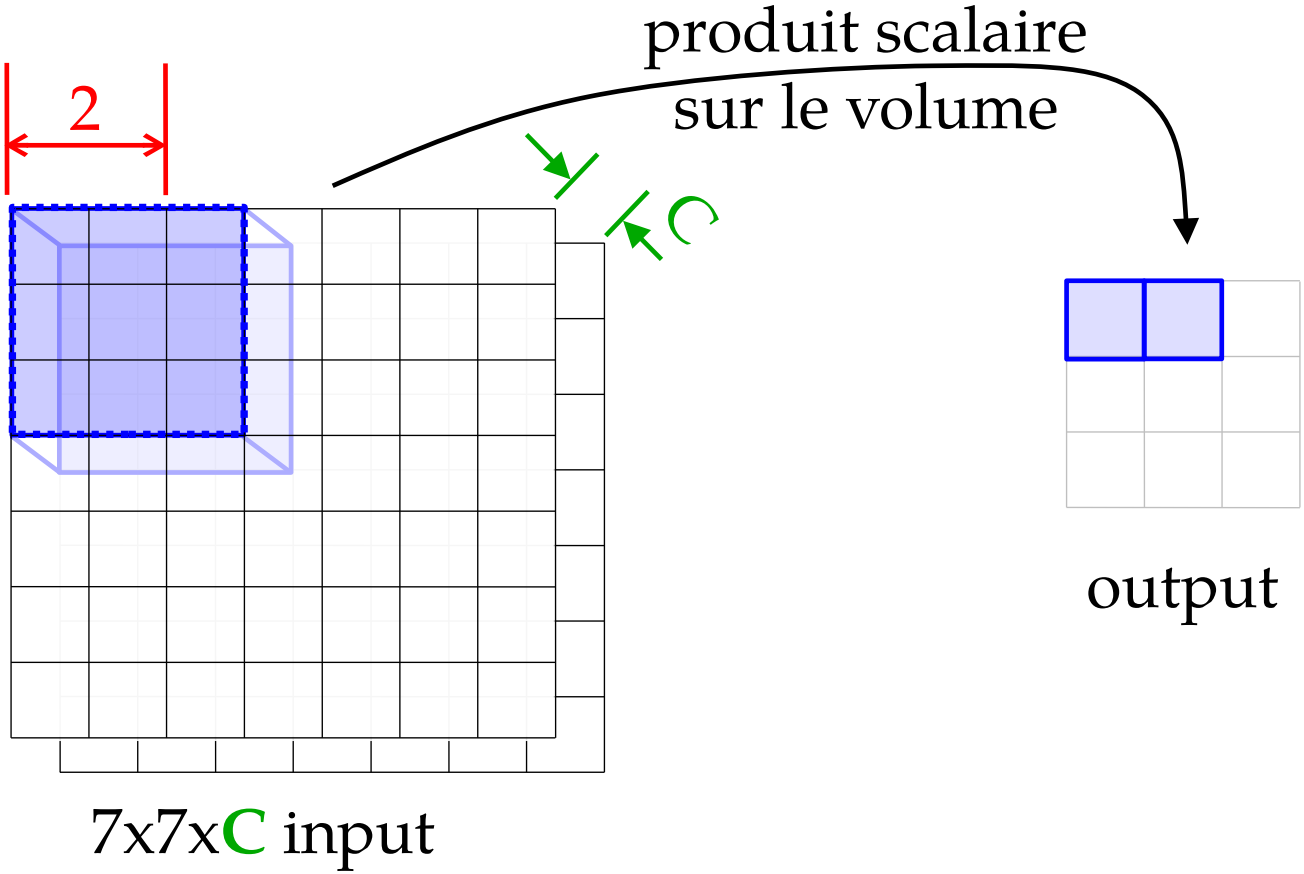
- Le ***pas*** est rarement plus de 3
  - Si plus d'1, réduit grandement la taille de sortie HxW
- Pas toujours possible d'avoir un nombre entier d'application de convolution, si le ***pas*** n'est pas 1
  - Par exemple, entrée 7x7, filtre 3x3, ***pas*** de 3
  - Libraire peut automatiquement faire du 0-padding, couper l'image (*crop*) ou lancer une exception

http://cs231n.github.io/convolutional-networks/

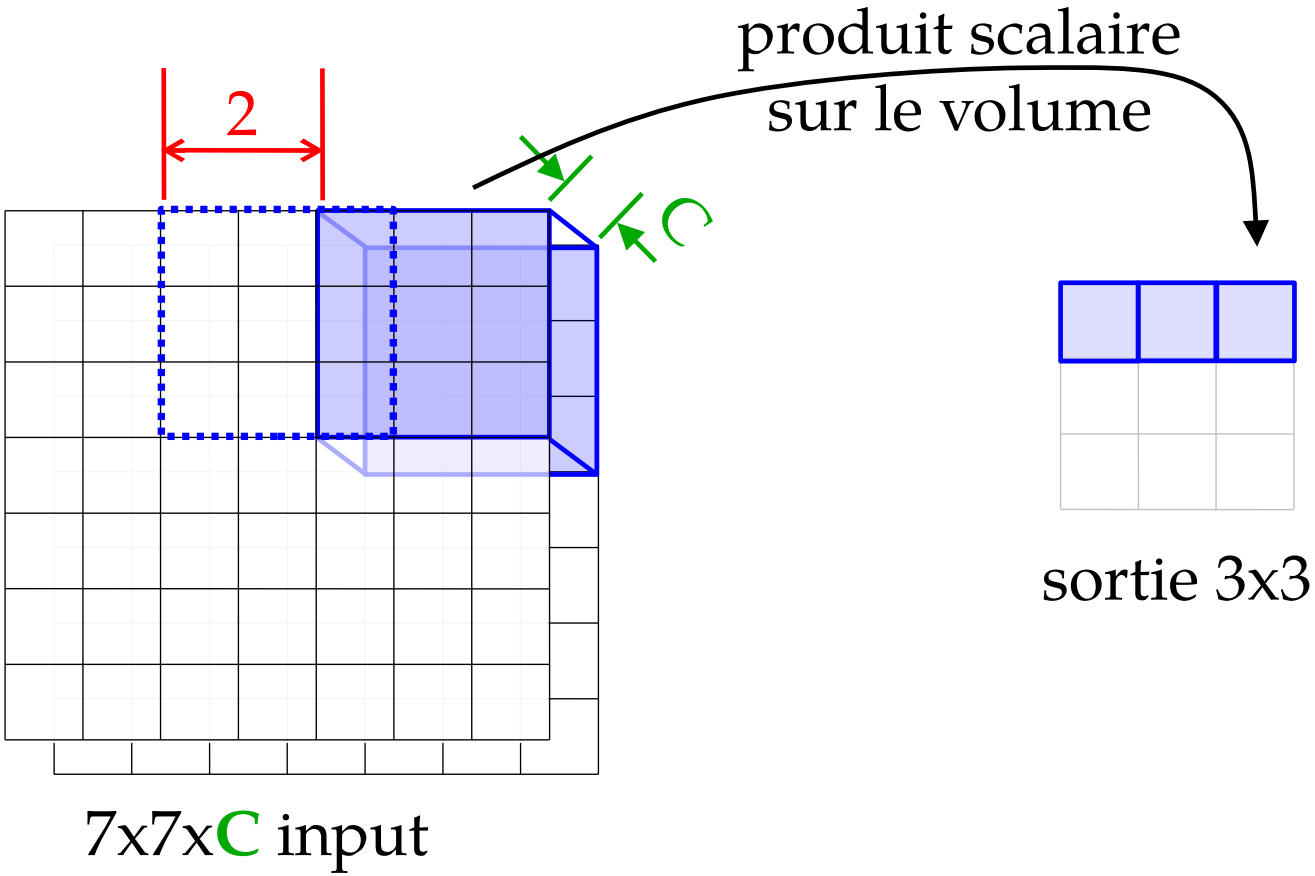
Pas de 2


produit scalaire
sur le volume
3x3xC filter
C
output
7x7xC input

Pas de 2



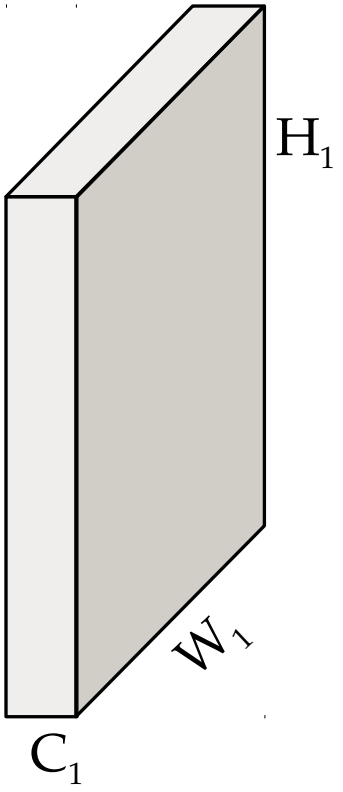
Pas de 2

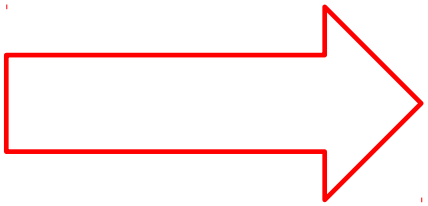


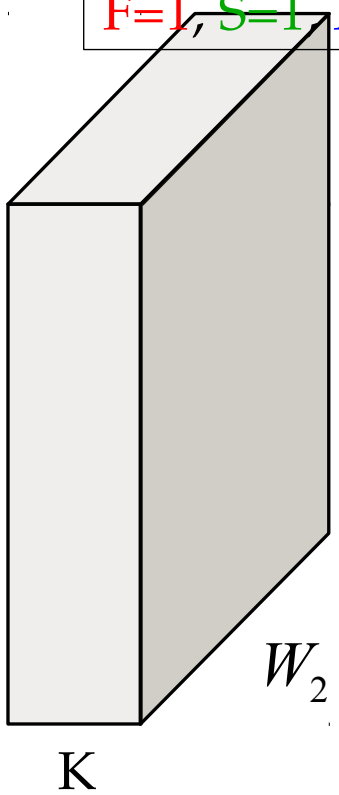
Tailles entrée et sortie

K : nombre de filtres
F : taille du filtre (FxF)
S : pas (stride)
P : quantité de 0-padding

Valeurs typiques (cs231n)
K : puissances de 2
F=3, S=1, P=1
F=5, S=1, P=2
F=5, S=2, P=autant que nécessaire
F=1, S=1, P=0







$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

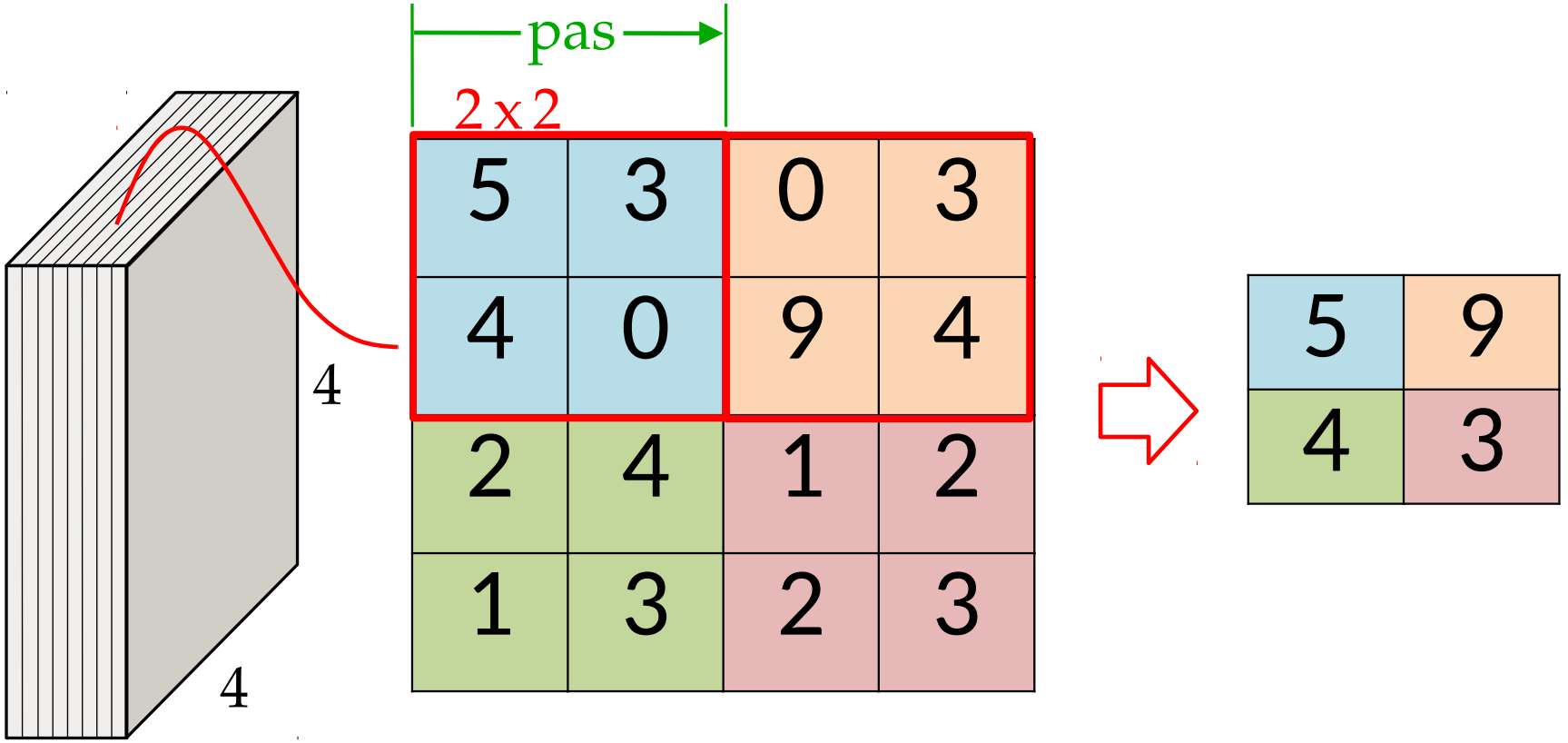
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

paramètres : F•F•C_1•K poids + K biais

Pooling

Max Pooling

- Appliqué pour chaque tranche, indépendamment



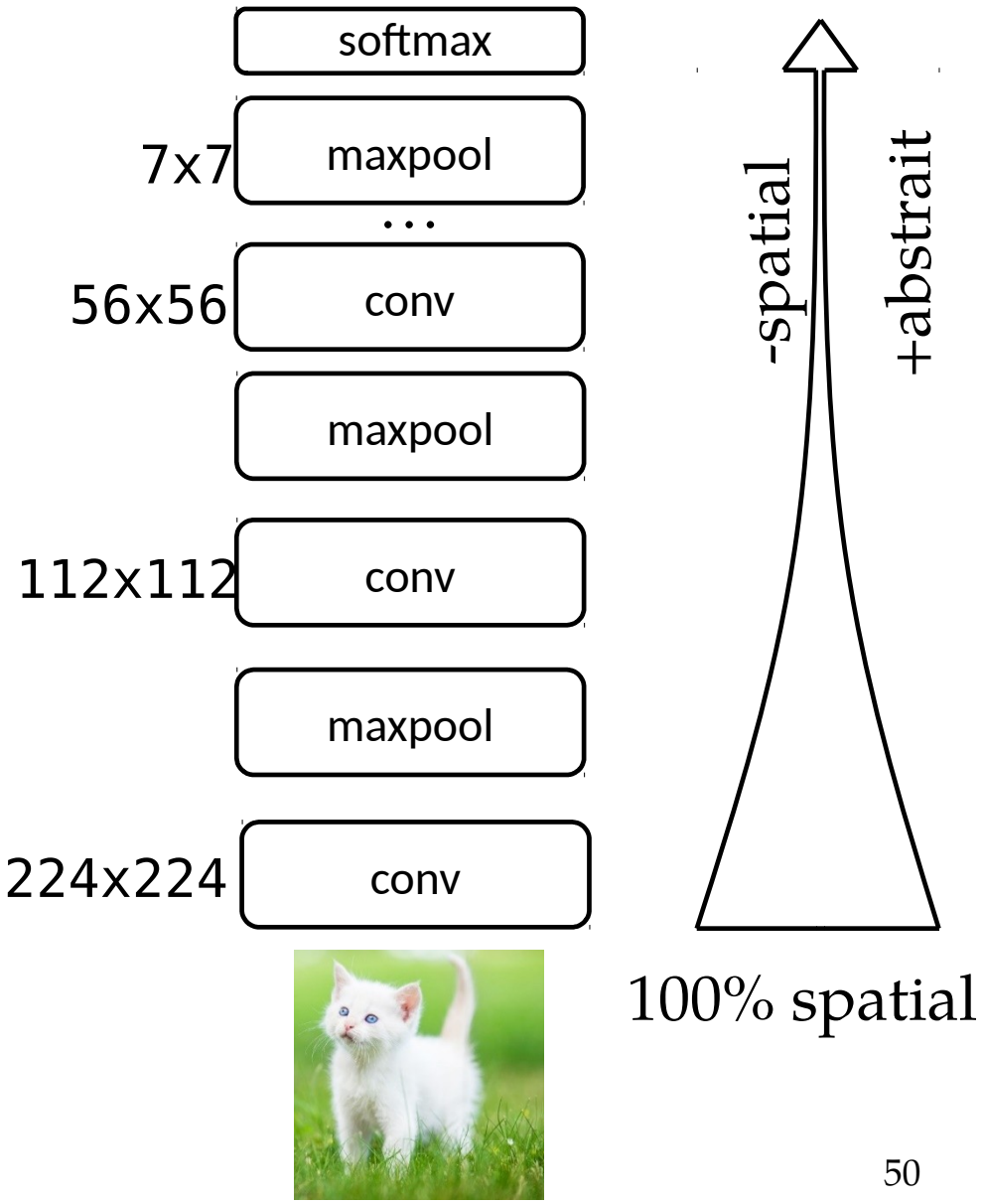
On doit spécifier le «pas»

Max Pooling

- Réduit la dimension du feature map
- Souvent, on en profite pour augmenter le nombre de filtre
  - la « quantité » d'information reste similaire
  - augmente la richesse de la représentation / abstraction
- Pourquoi maxpool au lieu de faire une moyenne?
  - maxpool: détecte la présence d'un feature dans une région
  - avgpool: va en partie noyer cette valeur (ou compter le nombre)

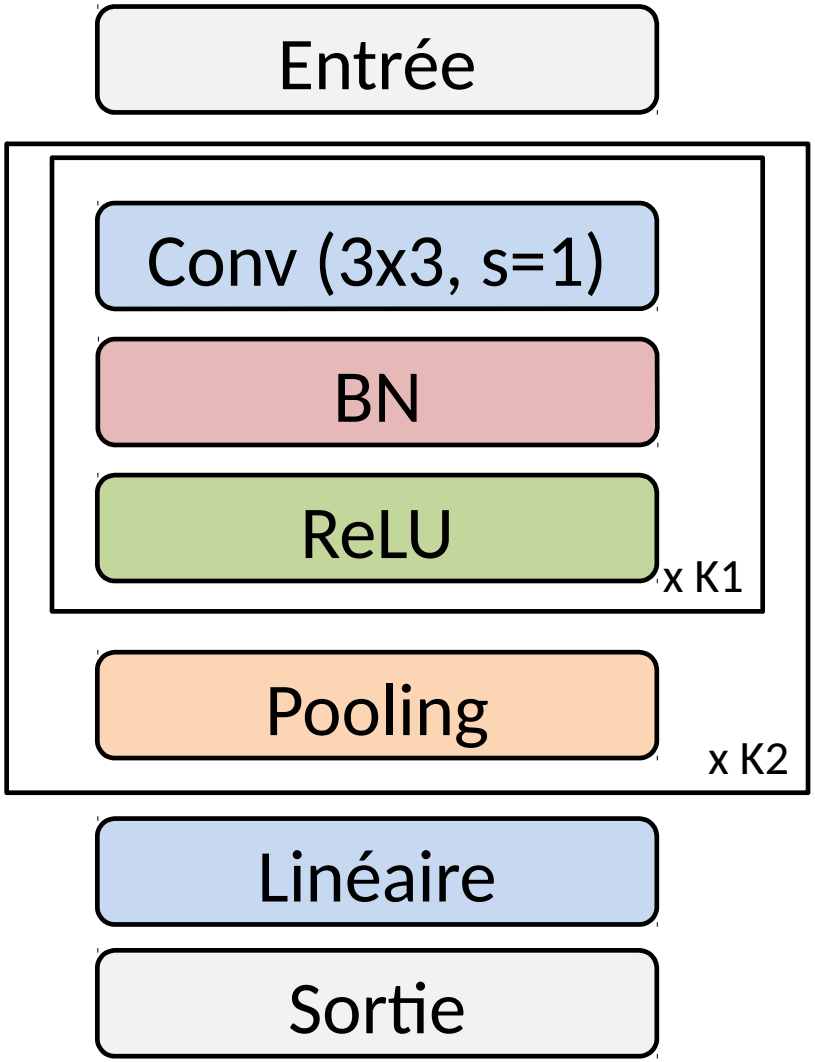
Pooling

- Augmente champ réceptif rapidement
- Réduit le nombre de paramètre
- Confère une certaine invariance aux transformations géométriques (rotation, translation, échelle)

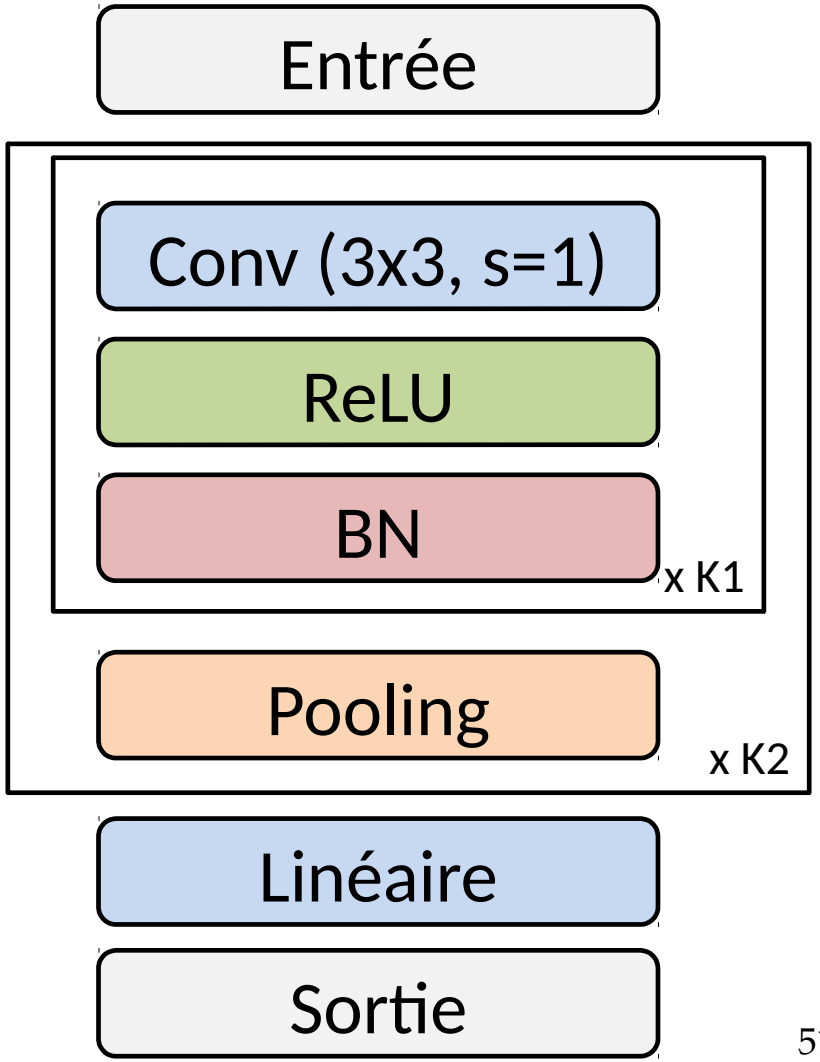


Position de la batch norm

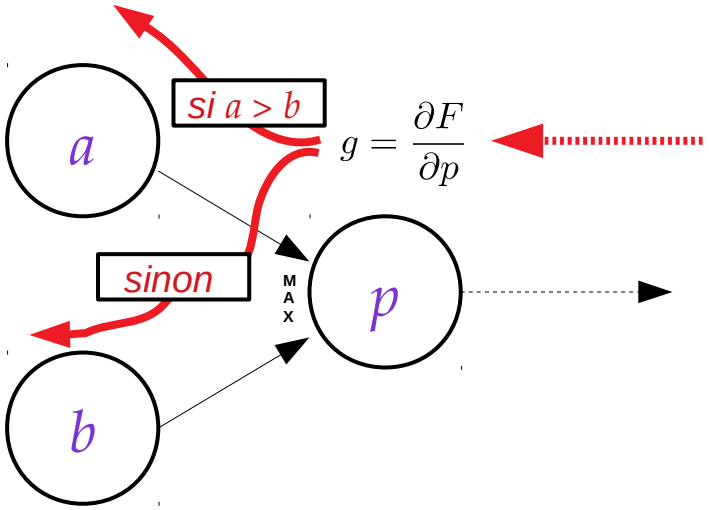
Classique



Nouvelle tendance



Rétropropagation du gradient



$$p = \max(a, b) = \begin{cases} a & \text{si } a > b \\ b & \text{sinon.} \end{cases}$$

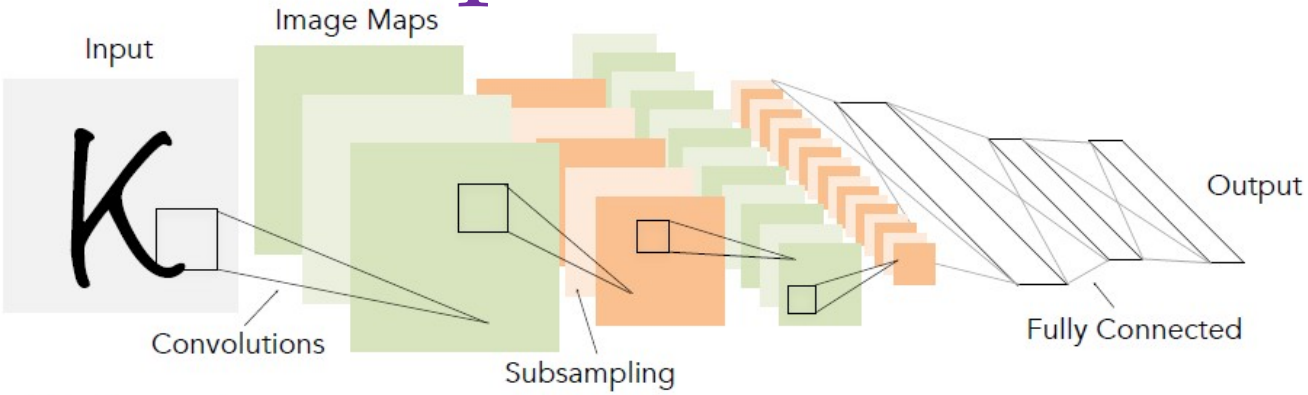
$$\frac{\partial p}{\partial a} = \max(a, b) = \begin{cases} 1 & \text{si } a > b \\ 0 & \text{sinon.} \end{cases}$$

$$\frac{\partial p}{\partial b} = \max(a, b) = \begin{cases} 0 & \text{si } a > b \\ 1 & \text{sinon} \end{cases}$$

Architectures

Ne date pas d'hier

1998
LeCun et al.



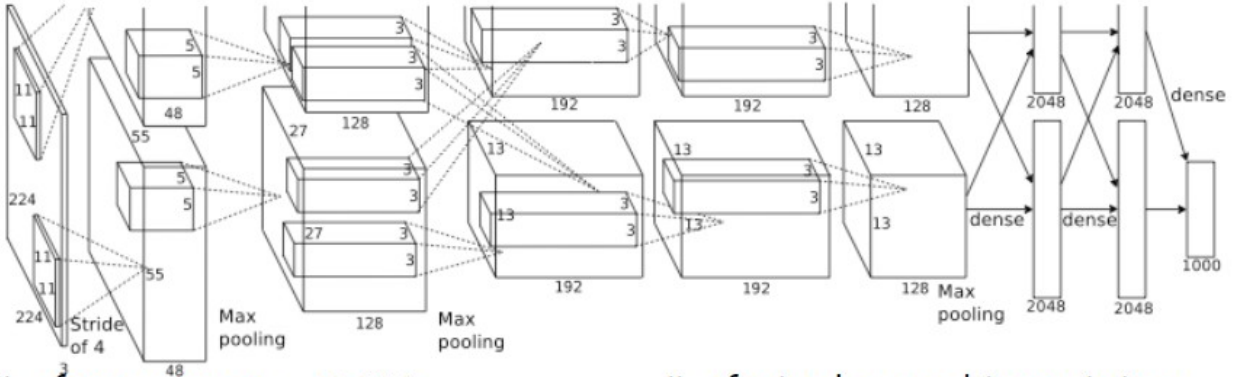
# of transistors

10^6

# of pixels used in training

10^7 NIST

2012
Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

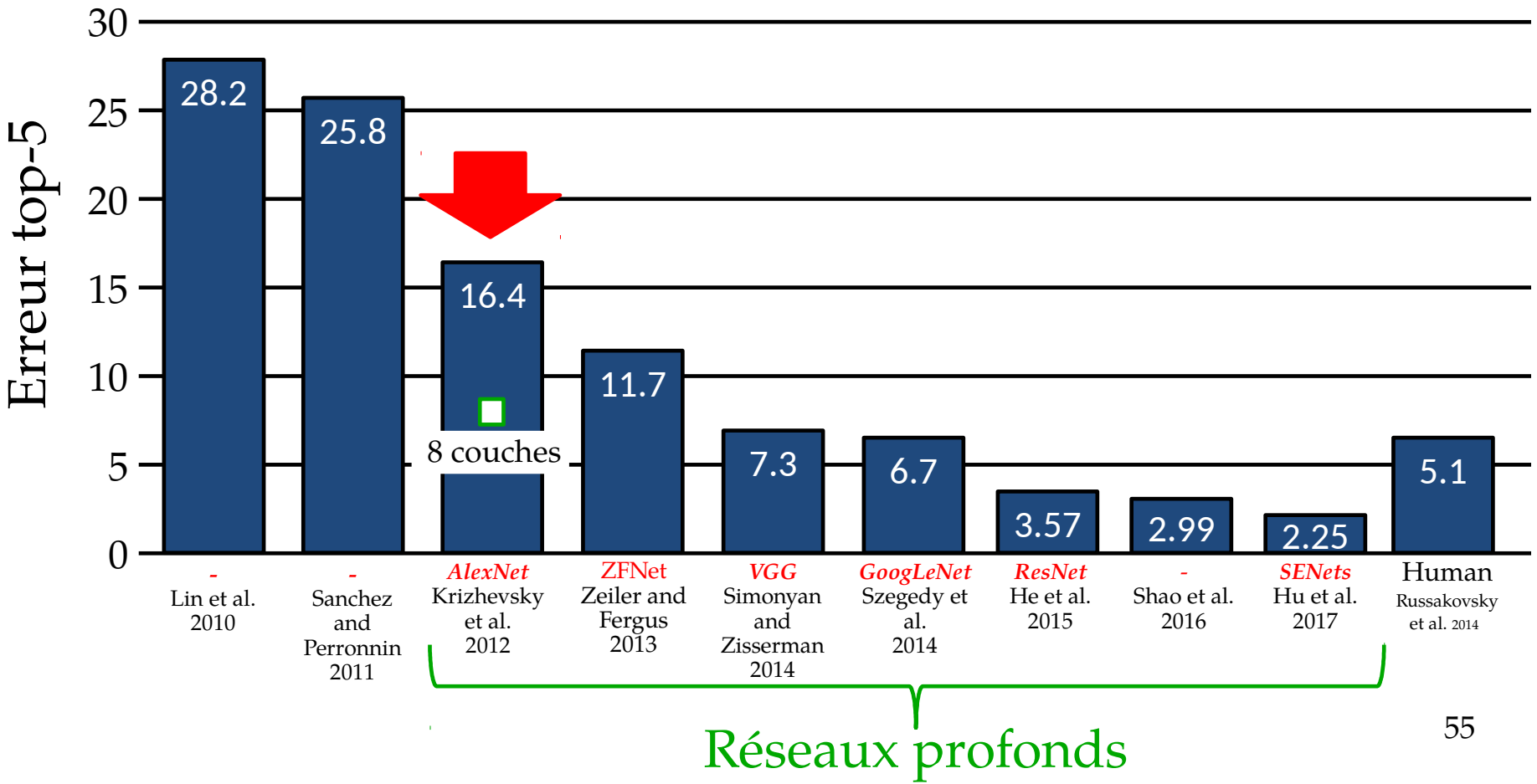
10^{14} IMAGENET

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

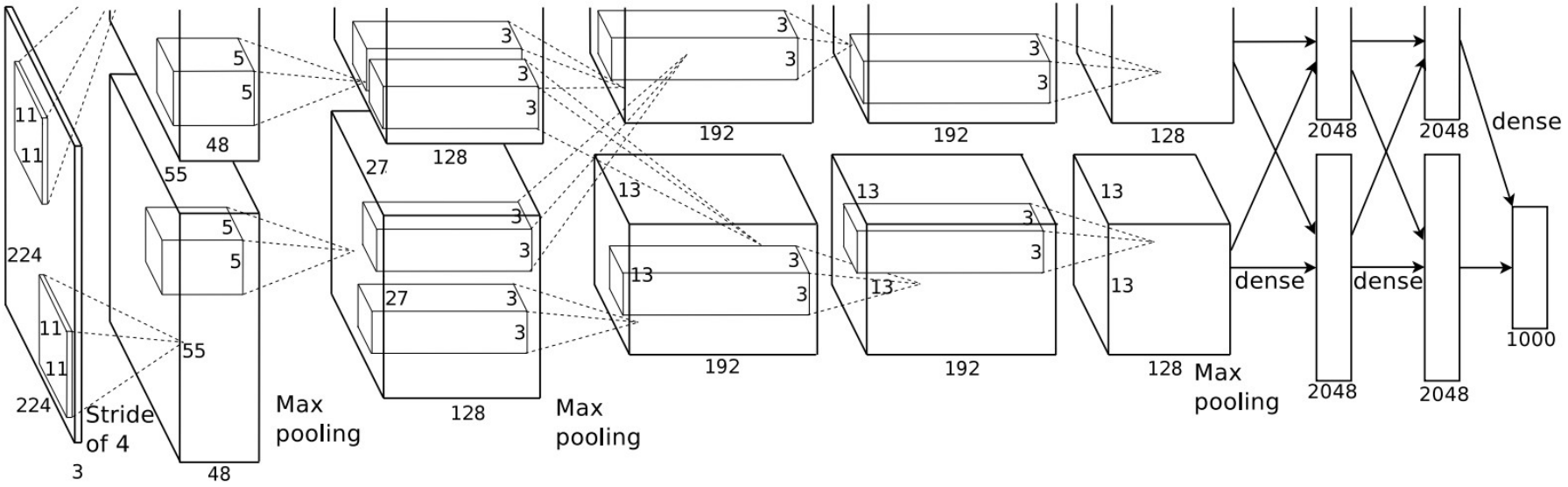
tiré de : cs231n, Université Stanford

Large Scale Visual Recognition Challenge

- *Image Classification Challenge* :
  - 1,000 classes d'objets
  - 1,431,167 images



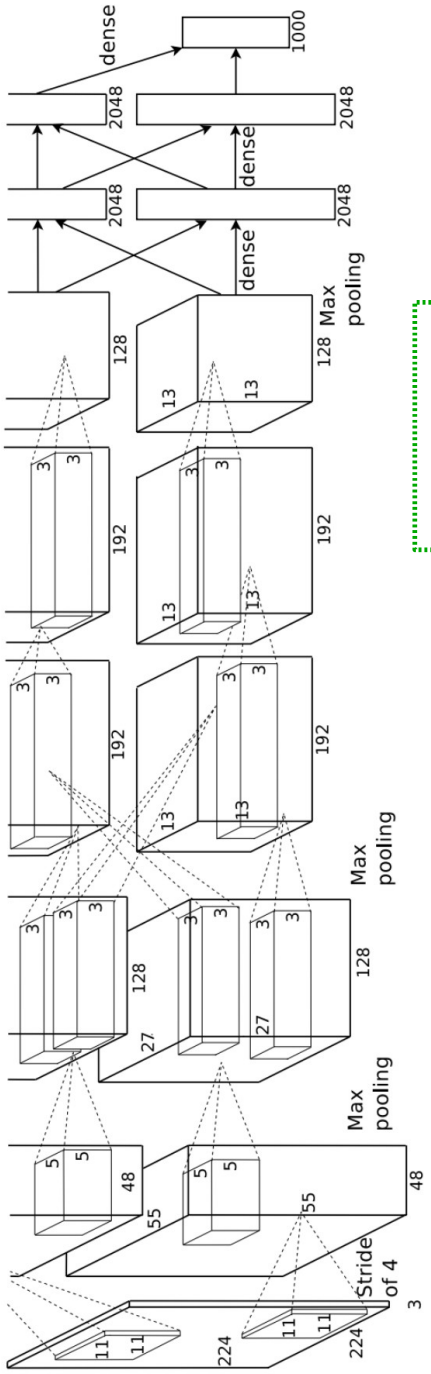
AlexNet



- 8 couches
- 60M paramètres
- Apparition des ReLU
- Dropout de 0.5
- Entraîné sur deux cartes GTX 580 (3 Go) en parallèle

adapté de : cs231n

AlexNet



adapté de : cs231n

majorité des paramètres

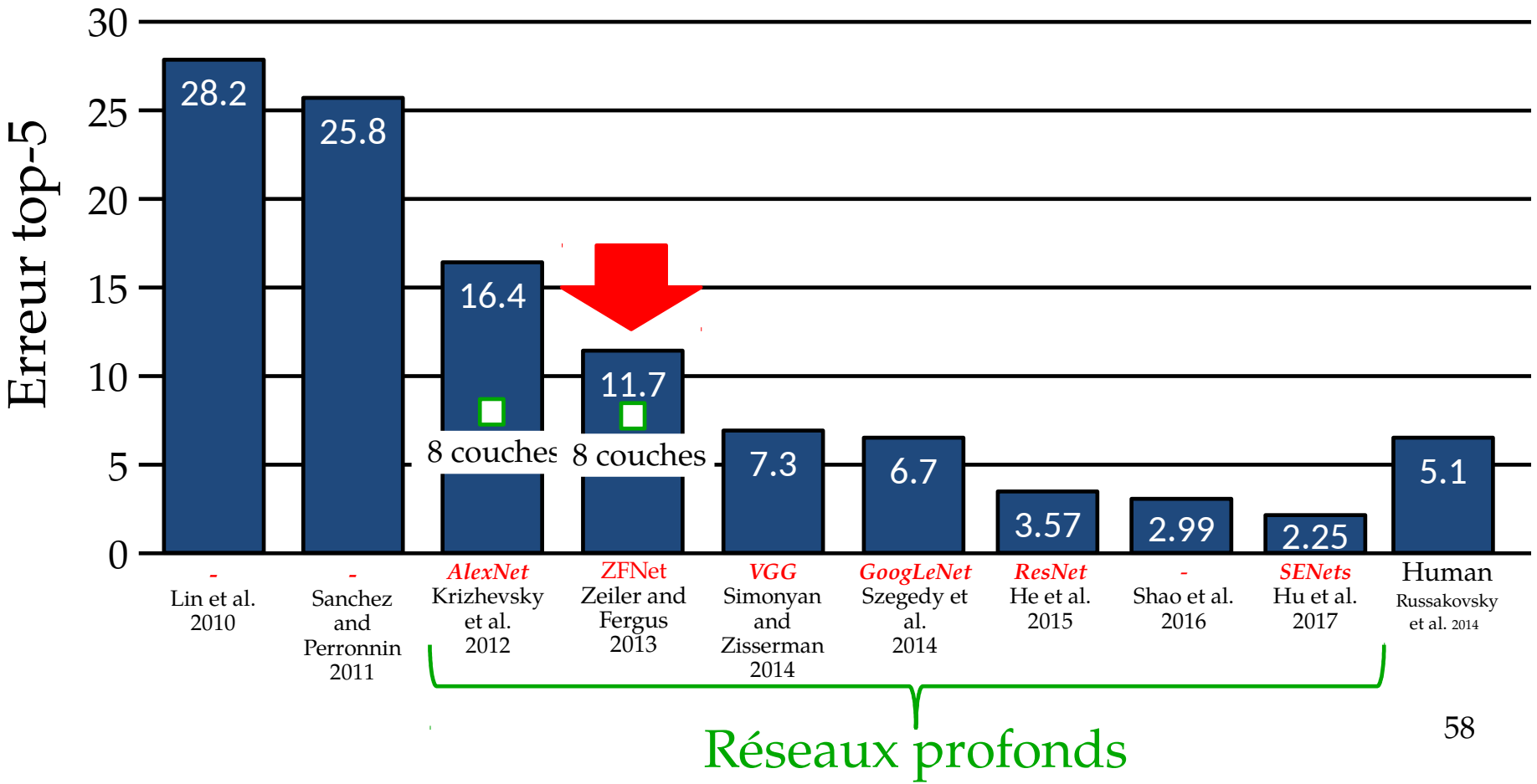
[1000] FC8: 1000 neurons (class scores)
[4096] FC7: 4096 neurons
[4096] FC6: 4096 neurons

Classificateur puissant (besoin dropout)

[6x6x256] MAX POOL3: 3x3 filters at stride 2
[13x13x256] CONV5: 256 **3x3** filters at stride 1, pad 1
[13x13x384] CONV4: 384 **3x3** filters at stride 1, pad 1
[13x13x384] CONV3: 384 **3x3** filters at stride 1, pad 1
[13x13x256] NORM2: Normalization layer
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[27x27x256] CONV2: 256 **5x5** filters at stride 1, pad 2
[27x27x96] NORM1: Normalization layer
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[55x55x96] CONV1: 96 **11x11** filters at stride 4, pad 0
[227x227x3] INPUT

réduction rapide

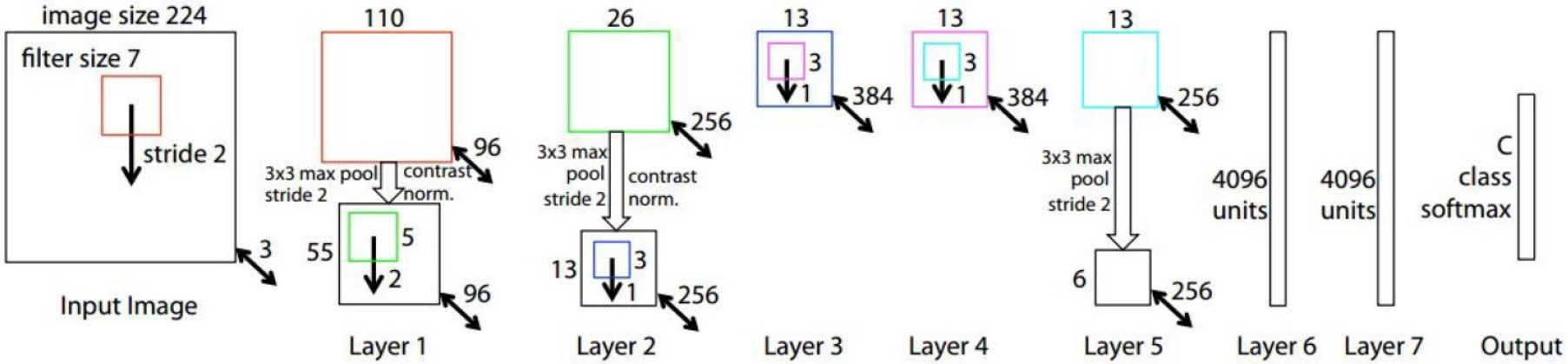
Large Scale Visual Recognition Challenge



ZFNet

ZFNet

[Zeiler and Fergus, 2013]



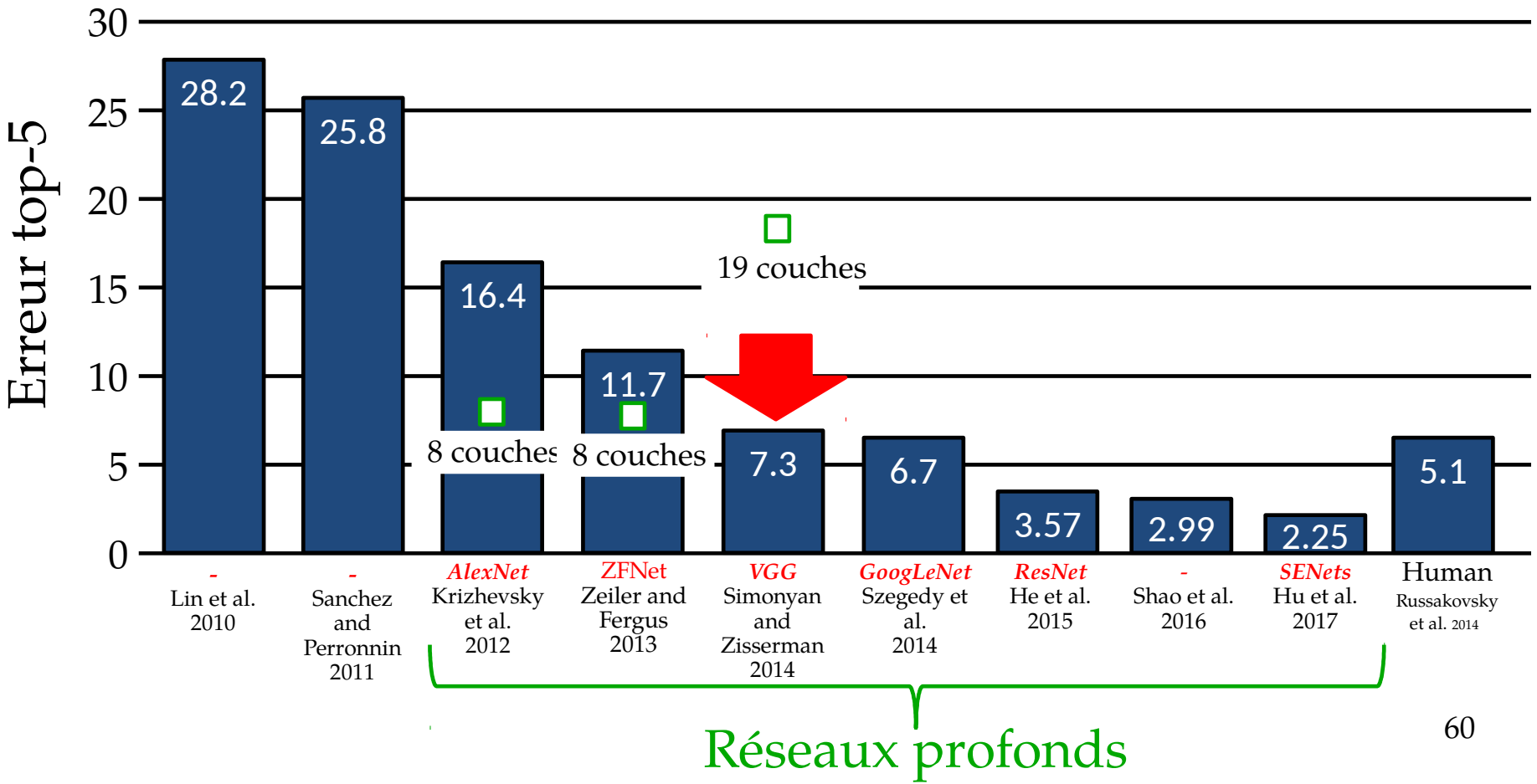
TODO: remake figure

AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

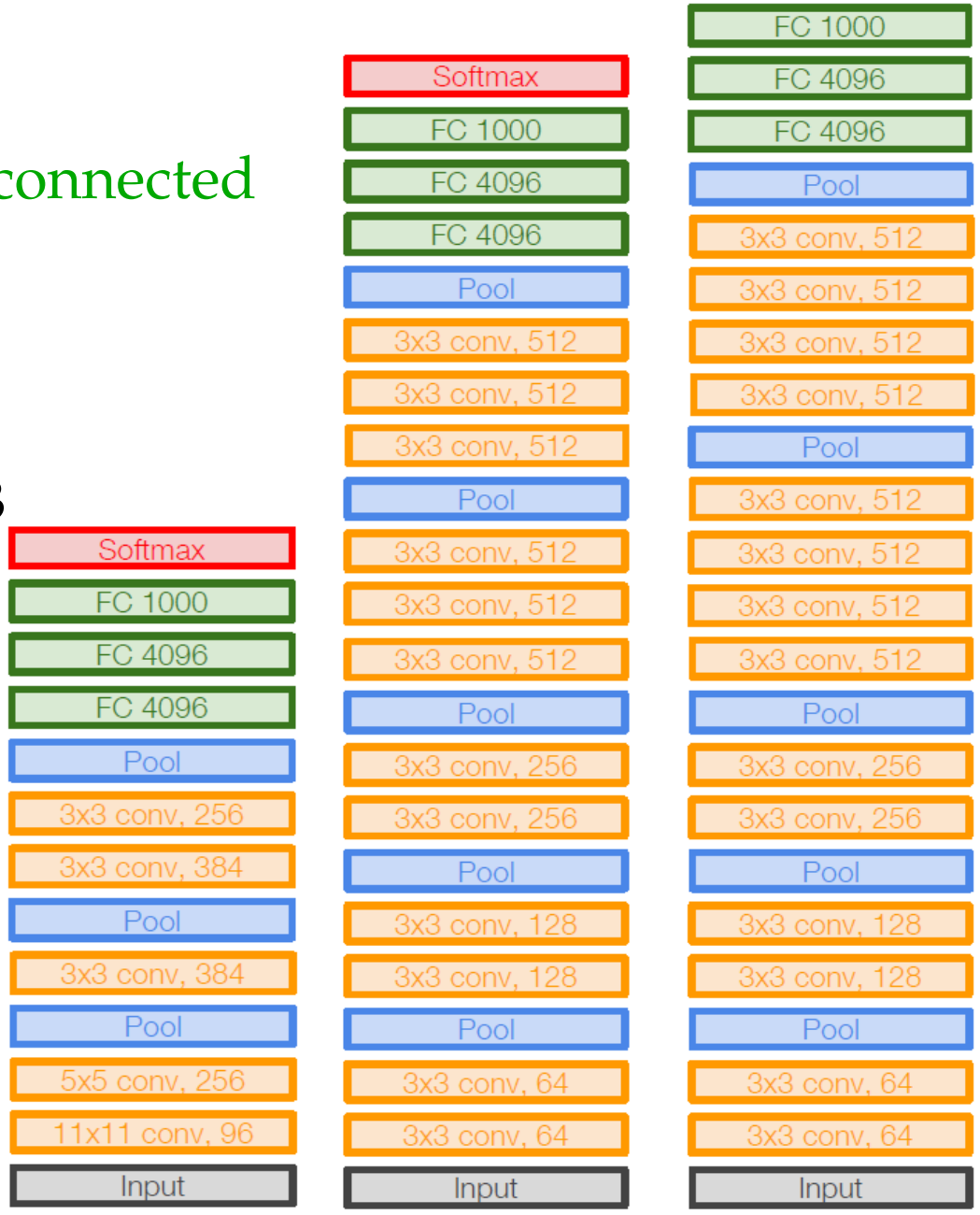
tiré de : cs231n

Large Scale Visual Recognition Challenge



VGGNet

- Toujours 3 couches fully-connected comme classificateur
- 16-19 couches
- 138M paramètres
- Que des convolutions 3x3
- Empilement de 3 convolution 3x3 a le même champ récepteur qu'un filtre 7x7
  - Mais plus de non-linéarité (si ReLU)
  - Moins de paramètres : $3(3^2C^2)$ vs. 7^2C^2 , avec C channels en entrée-sortie
    (économie de 45%)



adapté de : cs231n