Réseaux de neurones convolutionnels (CNN)

Julie Delon

M2 MM, Apprentissage en Grande Dimension
Université Paris Descartes

Diapositives basées sur les cours de cs231n (Stanford) - Fei-Fei Li & Justin Johnson & Serena Yeung, et sur le cours de Mauricio Delbracio, José Lezama, Guillermo Carbajal, Instituto de Ingeniería Eléctrica Facultad de Ingeniería Universidad de la República

Invariance en vision

- Problèmes de vision par ordinateur nécessitent des solutions invariantes à certaines transformations (point de vue, illumination)

- Deux approches possibles :

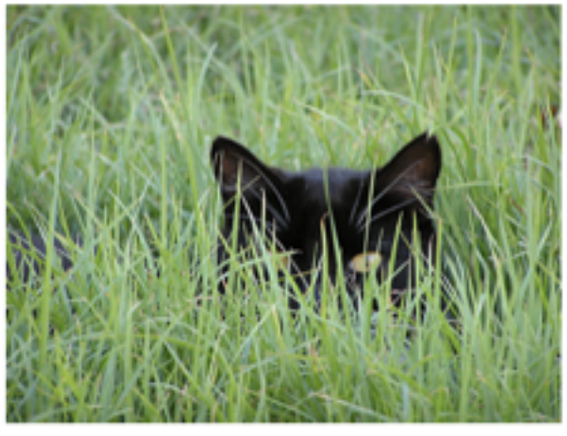
  - apprendre les invariances à partir d'une énorme base de données (let the data talk)

  - construire des représentations invariantes des données

Francesco Peri, Hidden Cat - Penny

Christina Gandolfo, Cat in the box

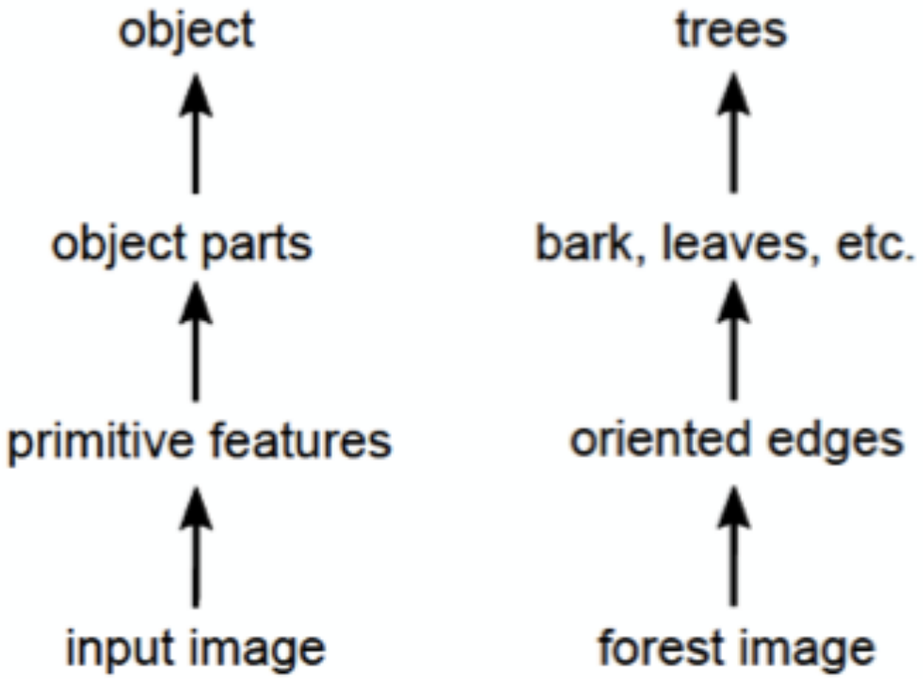
Matteo, hiding

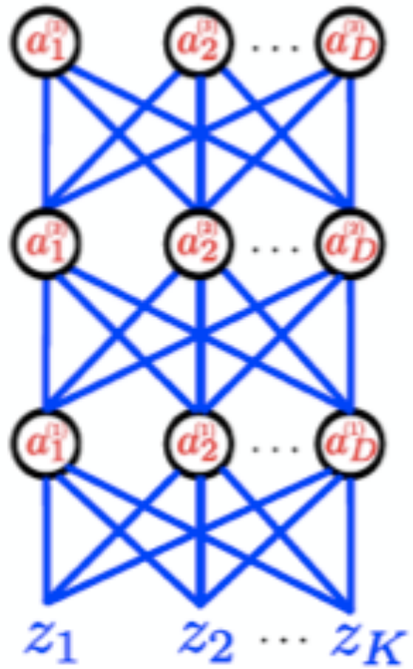


Grahford, Hidden Cat

Pourquoi des réseaux profonds ?

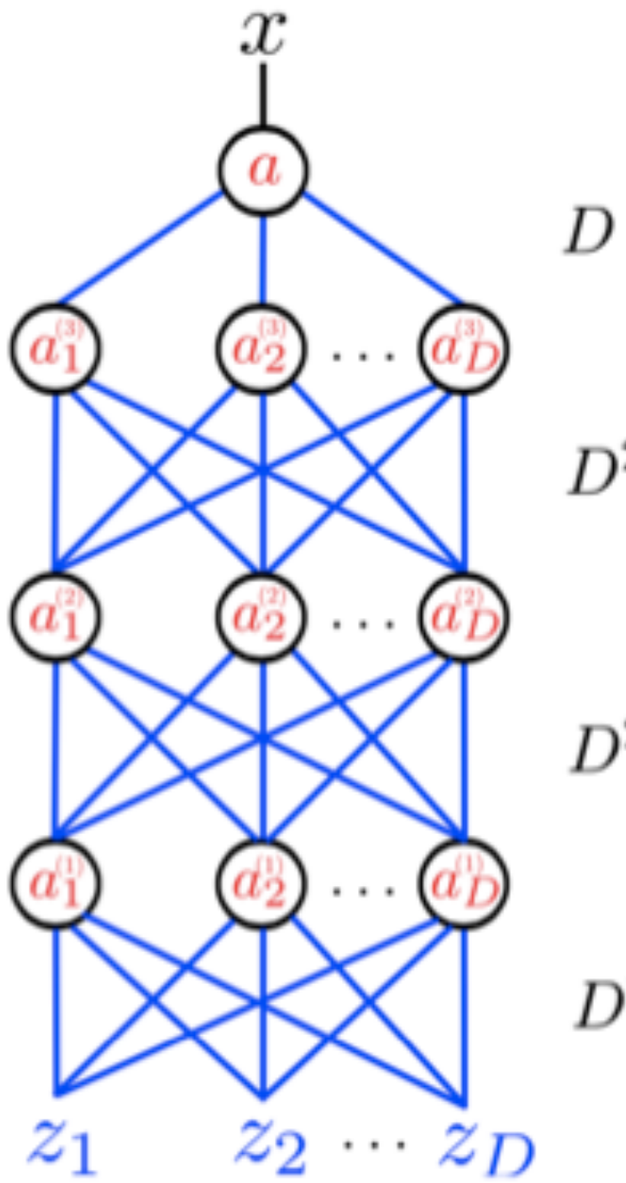
- Les données visuelles ont généralement une organisation hiérarchique



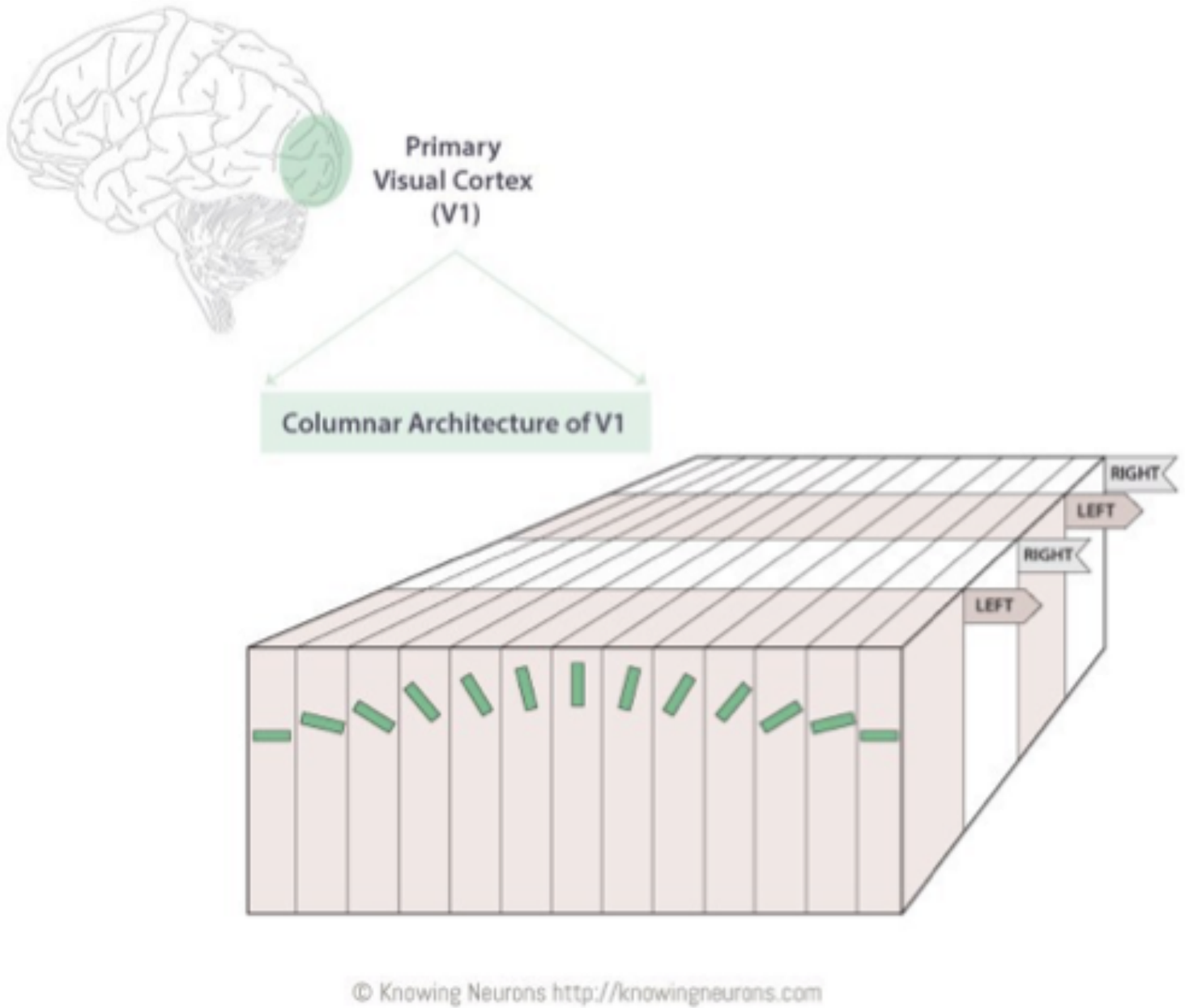


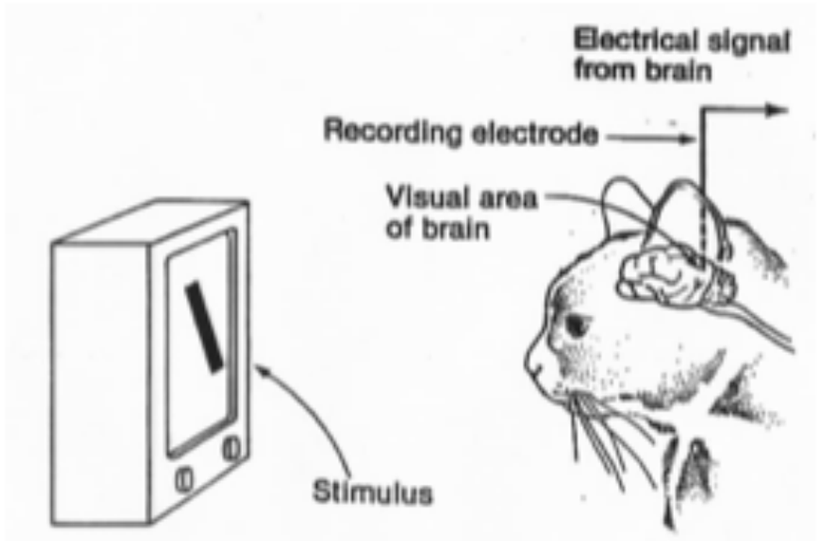
Pourquoi des réseaux profonds ?

- Combien de paramètres contient ce réseau ?
  - $3D^2+D$
- Pour une petite image 32x32
  - $3 \times (32^2)^2 + 32^2 \approx 3 \times 10^6$
- Difficile à entraîner : surajustement, initialisation
- Réseaux convolutionnels : permettent de diminuer le nombre de paramètres tout en forçant les invariances



Bases neurologiques de la perception visuelle





Hubel, David H., and Torsten N. Wiesel. The Journal of physiology, 1962
"Receptive fields, binocular interaction and functional architecture in the cat's visual cortex."

Convolutions

- Convolution d'image : opération linéaire entre une image et un filtre, produisant une nouvelle image

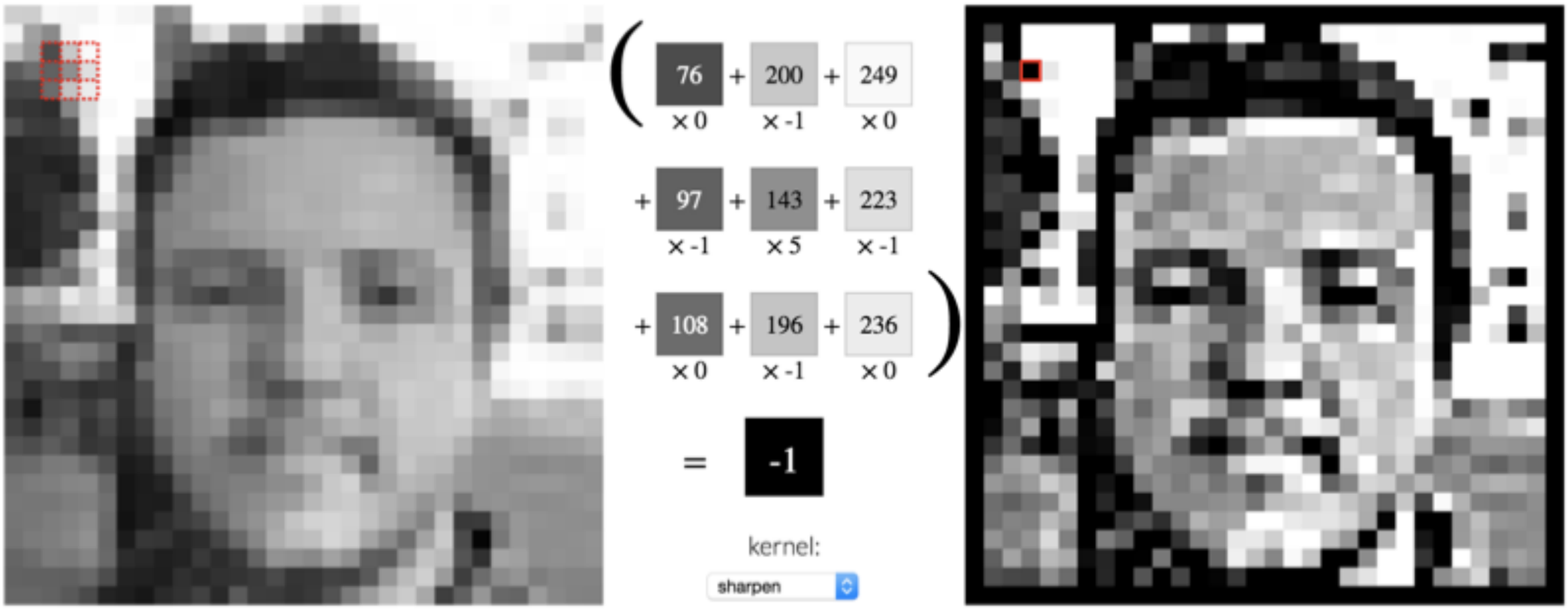
- Chaque pixel se calcule comme une somme pondérée des pixels de l'image d'entrée, translatée par rapport au noyau de convolution

$$(u * h)(i,j) = \sum_{k,l} u(i-k, j-l) h(k,l)$$

$$\underbrace{\begin{bmatrix} 130 & 136 & 53 & 44 & 231 & 67 & 108 \\ 130 & 89 & 77 & 58 & 250 & 154 & 130 \\ 208 & 239 & 120 & 111 & 112 & 181 & 22 \\ 203 & 223 & 59 & 79 & 28 & 57 & 67 \\ 164 & 140 & 215 & 235 & 66 & 30 & 204 \\ 97 & 159 & 50 & 110 & 104 & 76 & 7 \\ 207 & 150 & 58 & 47 & 152 & 81 & 237 \end{bmatrix}}_{u} * \underbrace{\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{h} = \underbrace{\begin{bmatrix} 79.2 & 81.6 & 62.0 & 77.2 & 118.4 & 112.0 & 61.0 \\ 111.4 & 134.2 & 79.4 & 108.0 & 161.0 & 156.4 & 82.8 \\ 156.0 & 175.8 & 121.2 & 96.0 & 136.4 & 105.2 & 80.0 \\ 159.6 & 172.8 & 139.2 & 102.4 & 68.4 & 72.6 & 70.0 \\ 120.8 & 180.2 & 139.8 & 141.0 & 92.6 & 86.6 & 61.6 \\ 125.4 & 119.2 & 118.4 & 109.2 & 101.6 & 59.6 & 104.8 \\ 90.8 & 114.8 & 61.0 & 73.4 & 76.8 & 109.2 & 65.0 \end{bmatrix}}_{u*h}$$

Convolutions d’images

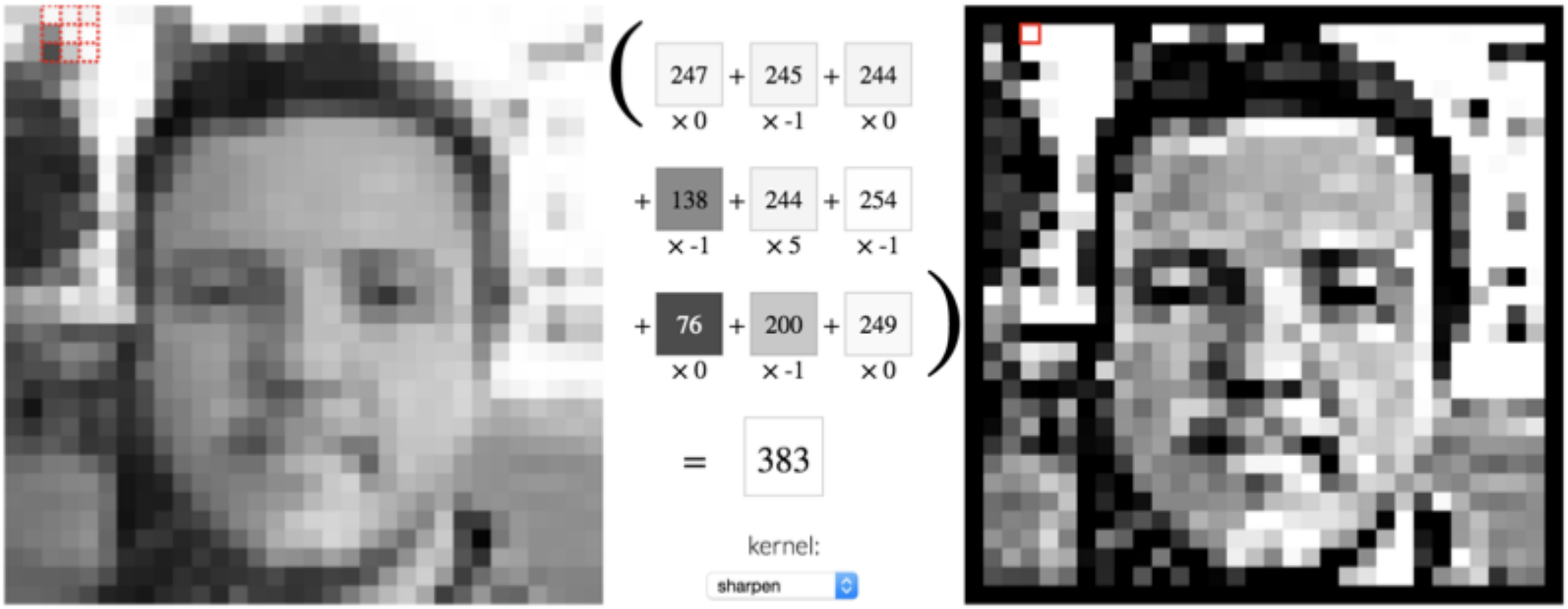
- Noyaux de convolution, expliqués visuellement par Victor Powell
  setosa.io/ev/image-kernels



$$\text{Kernel - } h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolutions d’images

- Noyaux de convolution, expliqués visuellement par Victor Powell
  setosa.io/ev/image-kernels



$$\text{Kernel - } h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Pourquoi des réseaux convolutionnels ?

1. Statistiques des images naturelles sont **invariantes à la translation**

   - On veut imposer l'invariance dans le modèle plutôt que l'apprendre

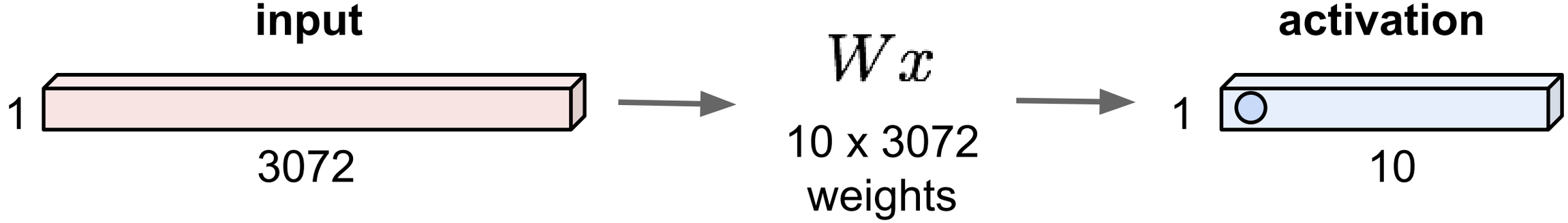
   - On veut minimiser le nombre de paramètres du réseau

2. Les caractéristiques de bas niveau sont locales (bords, blobs) -> utilisation de **filtres de petite taille**

3. Les caractéristiques de haut niveau sont globales -> besoin d'un **réseau profond**

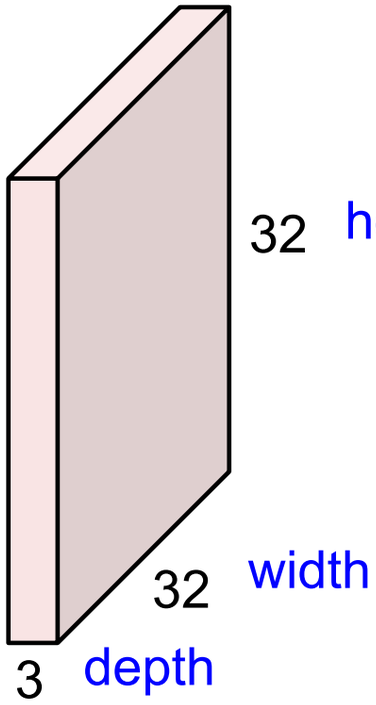
Réseau complètement connecté

32x32x3 image -> stretch to 3072 x 1



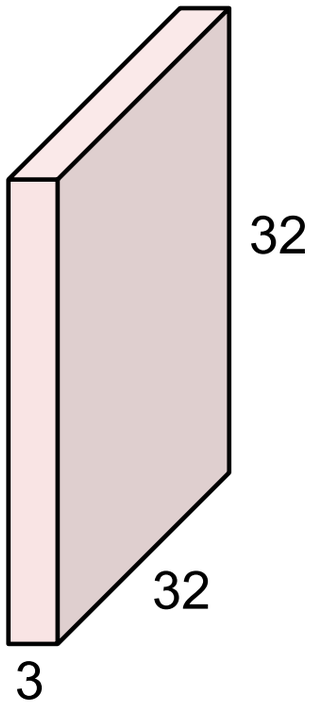
Réseau convolutionnel

32x32x3 image -> preserve spatial structure

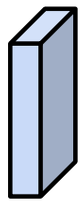


Réseau convolutionnel

32x32x3 image

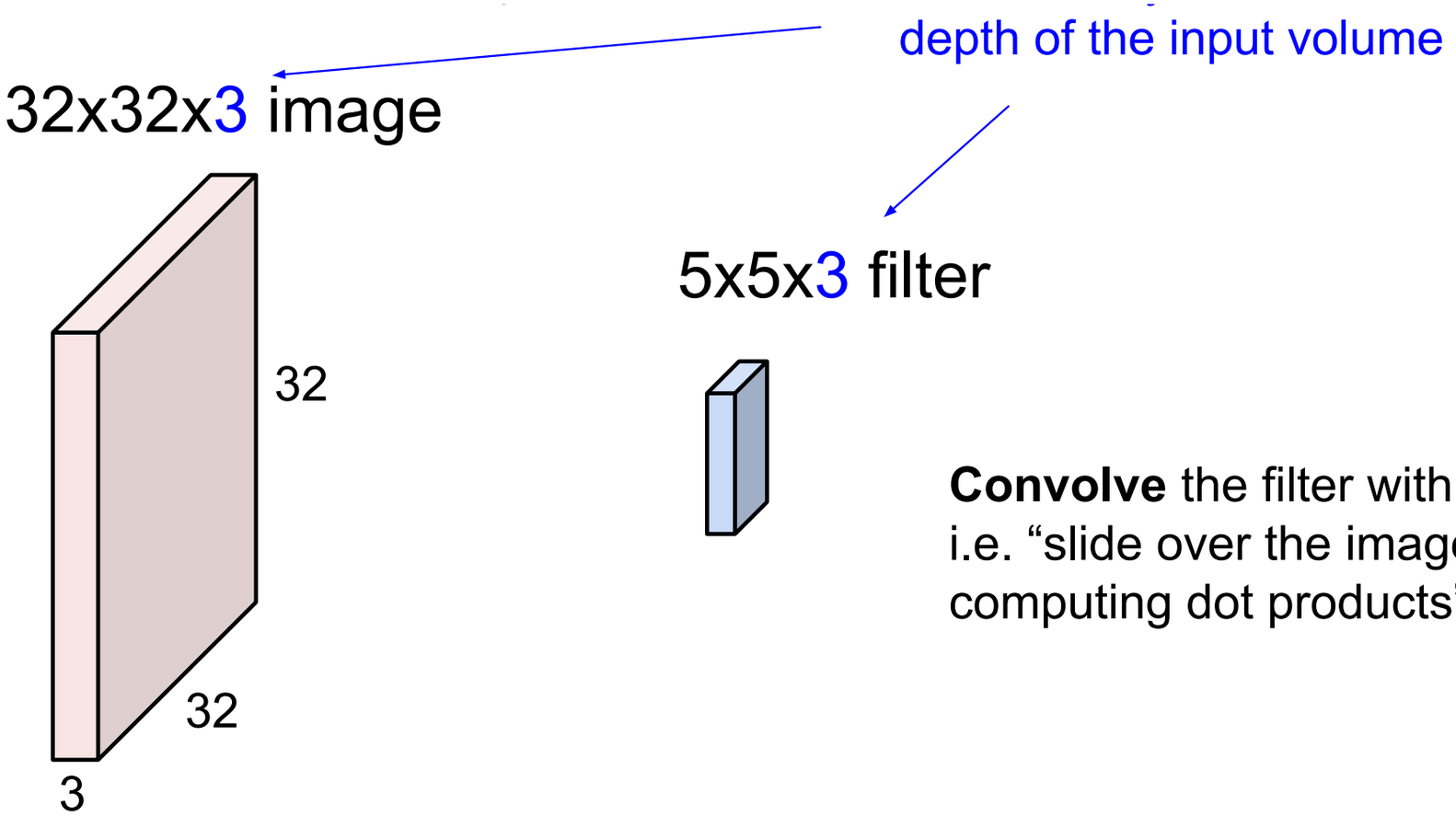


5x5x3 filter



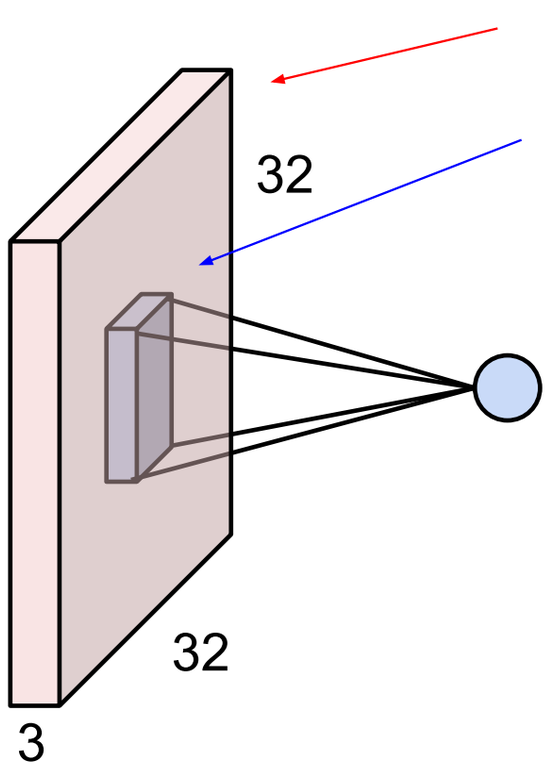
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Réseau convolutionnel



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

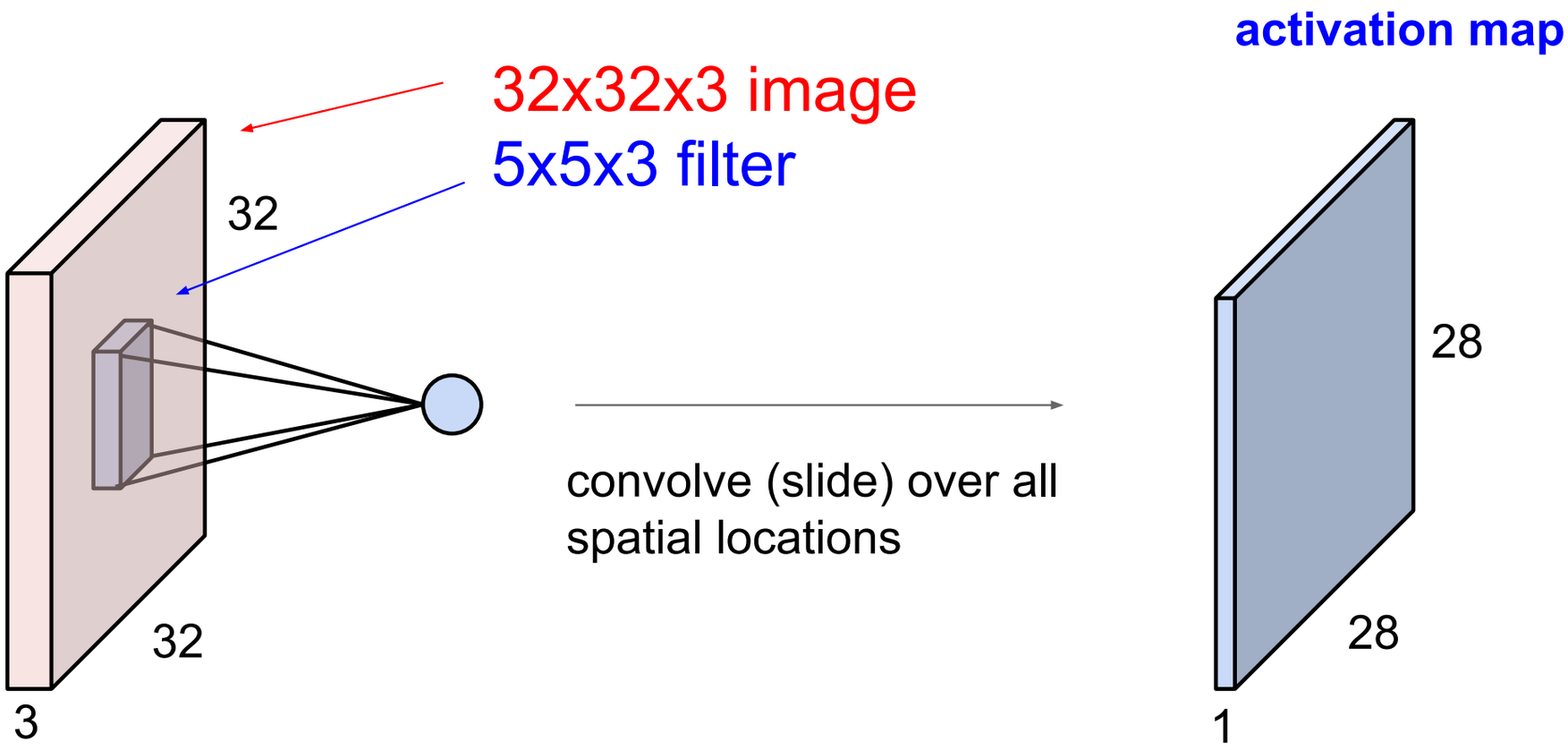
Réseau convolutionnel



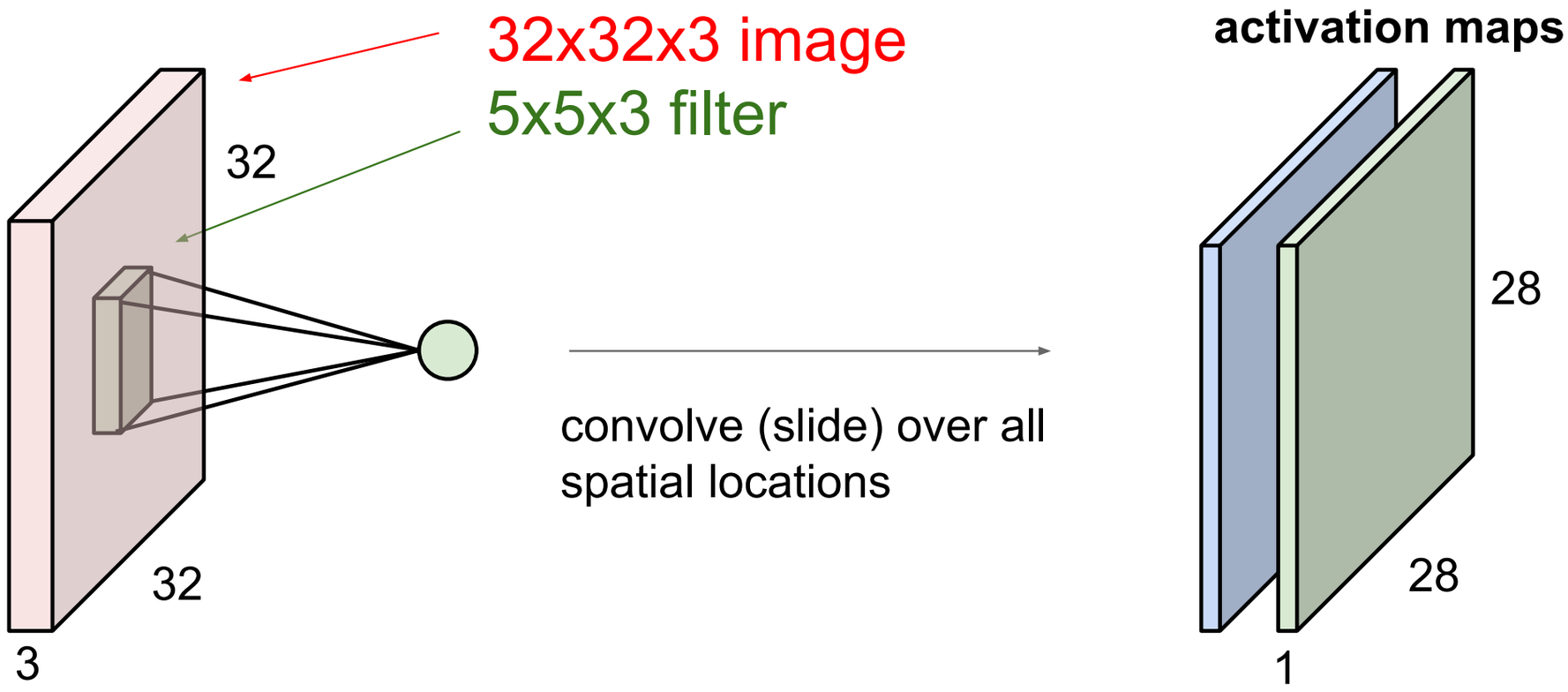
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

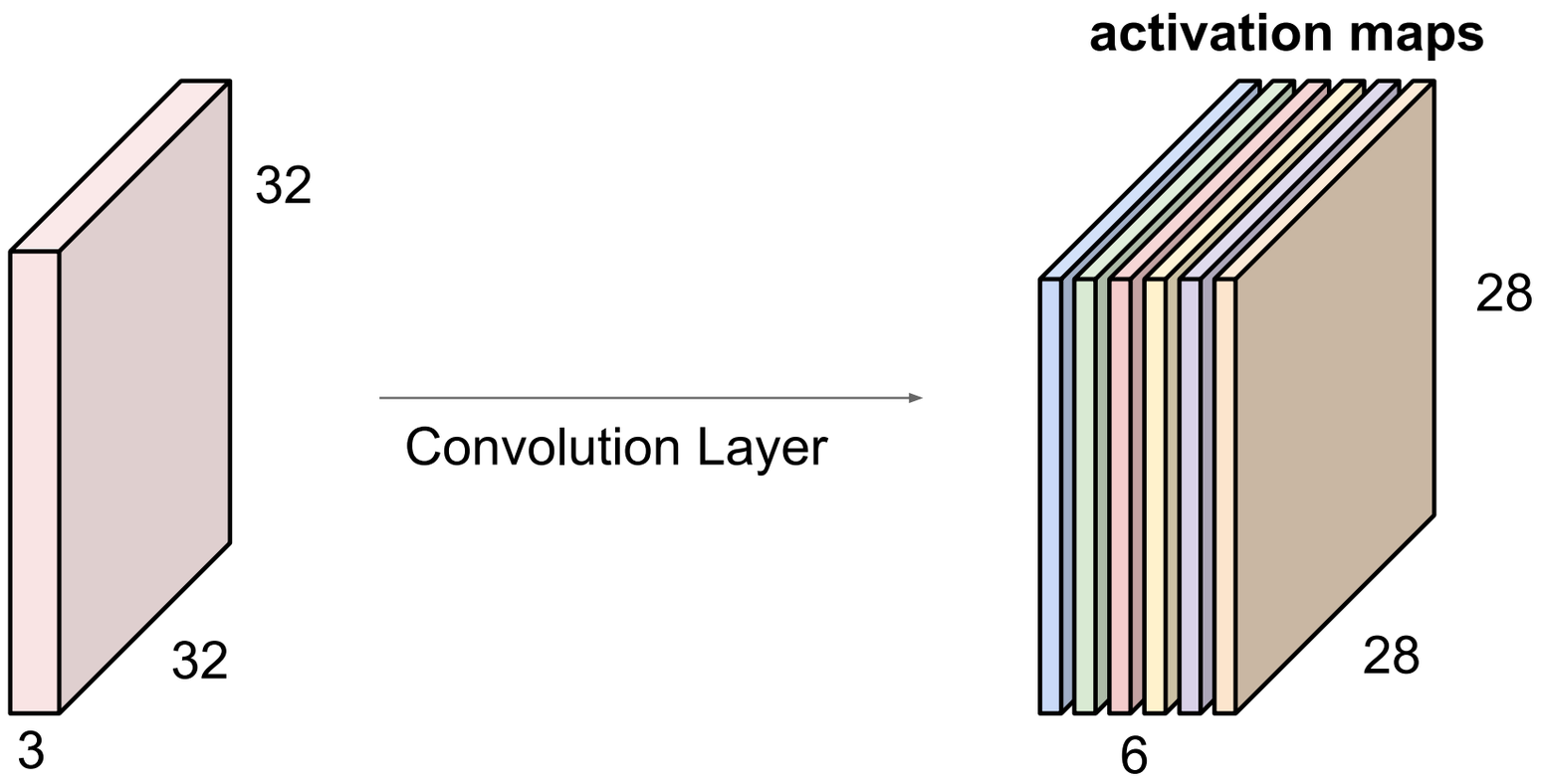
Réseau convolutionnel



Réseau convolutionnel



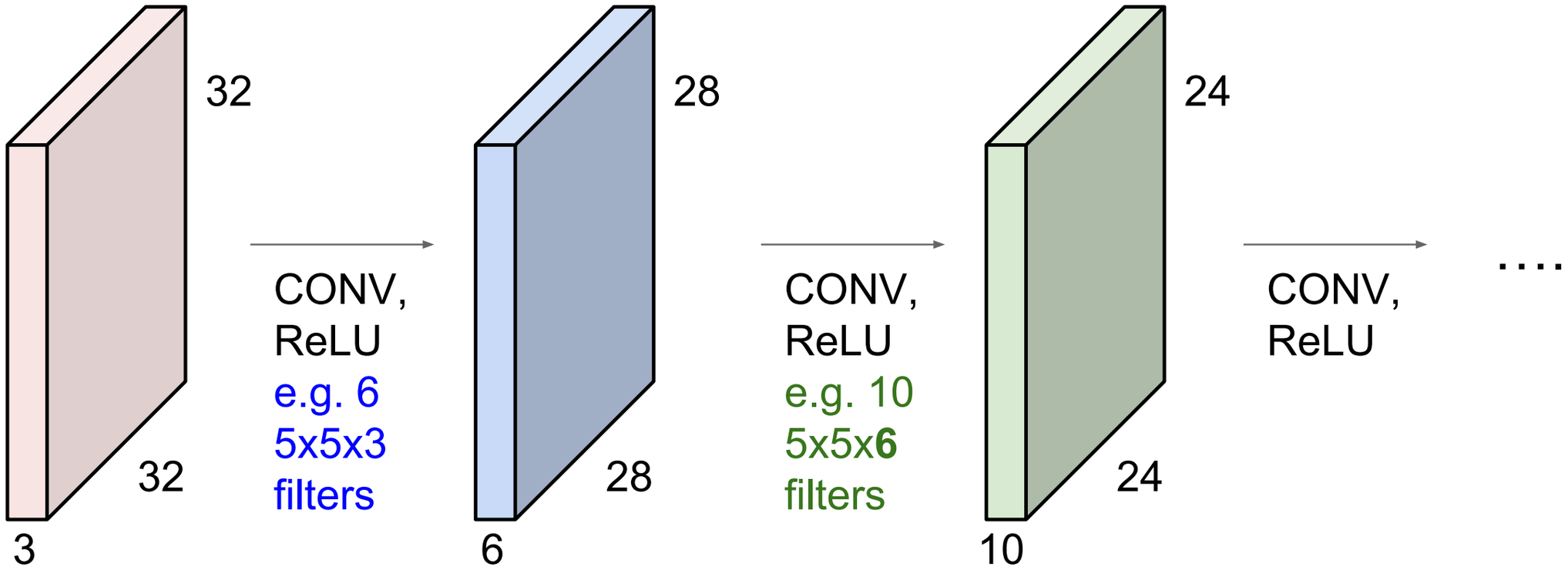
Réseau convolutionnel



We stack these up to get a “new image” of size 28x28x6!

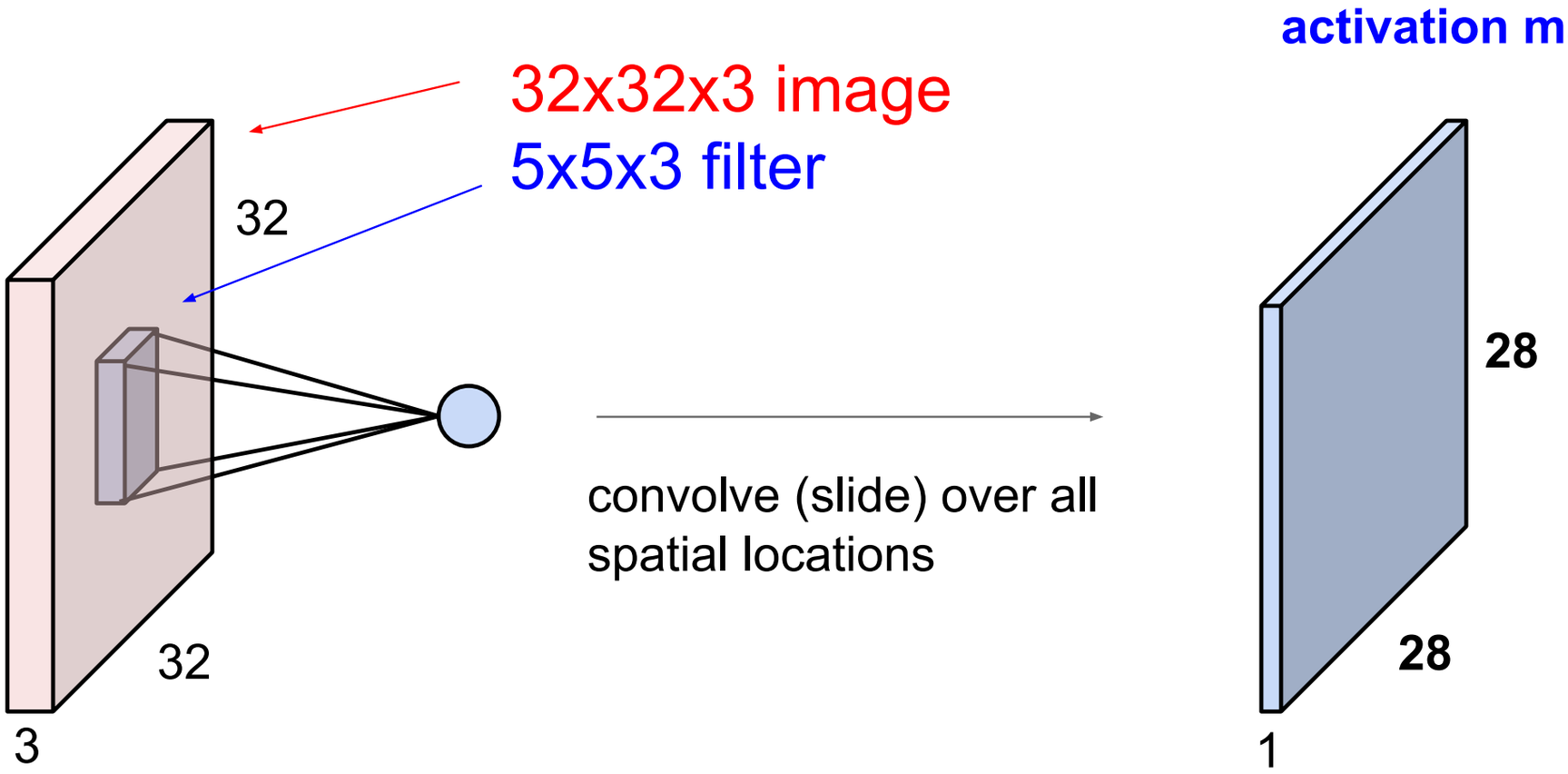
Réseau convolutionnel

- Les réseaux convolutionnels sont constitués de séquences de couches de convolution, intercalées avec des fonctions d'activation



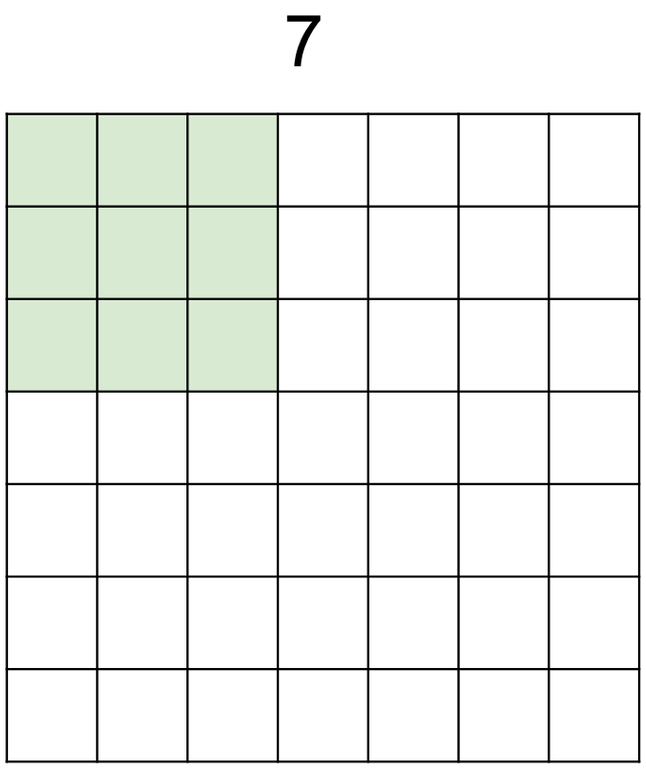
Dimensions spatiales

A closer look at spatial dimensions:



Dimensions spatiales

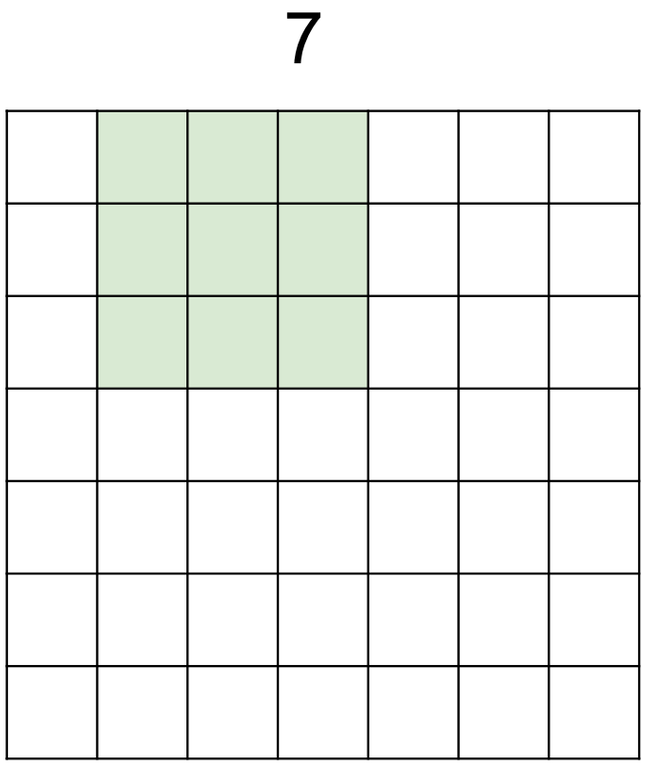
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Dimensions spatiales

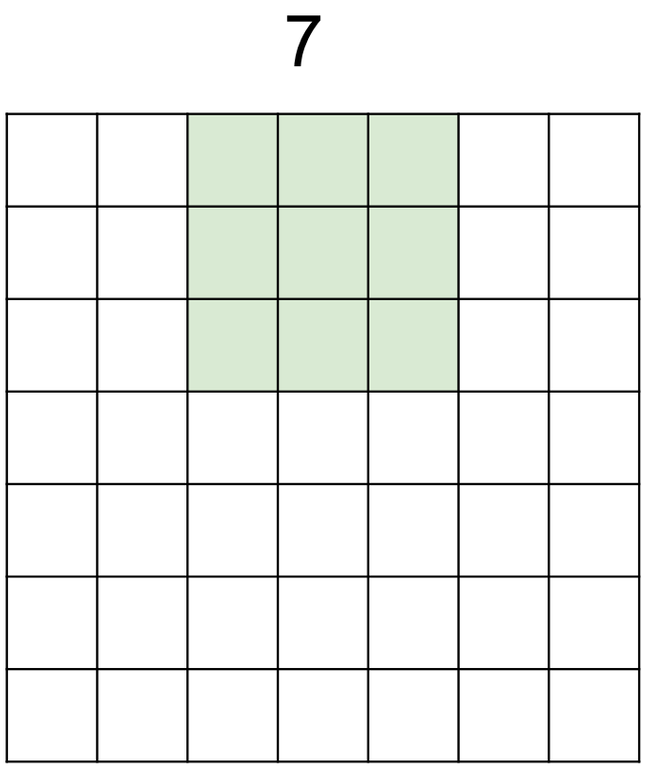
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Dimensions spatiales

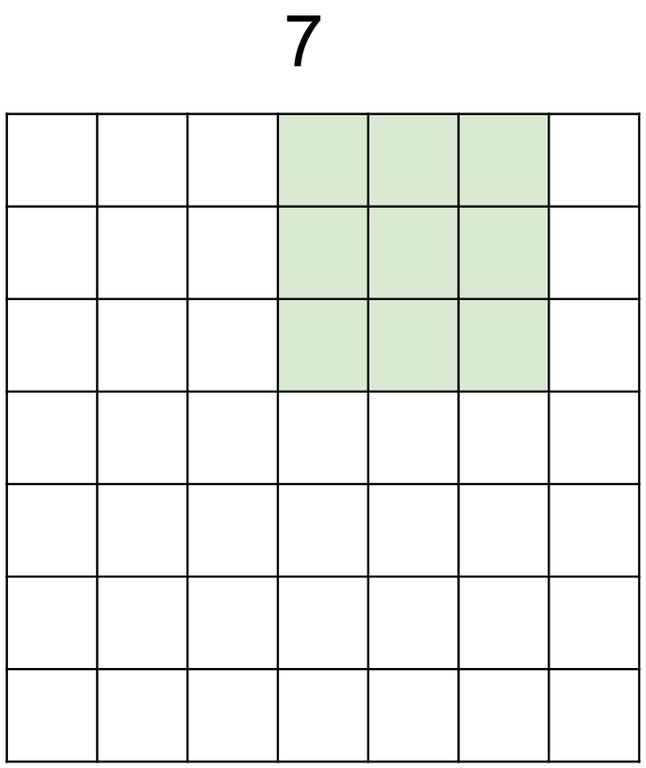
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Dimensions spatiales

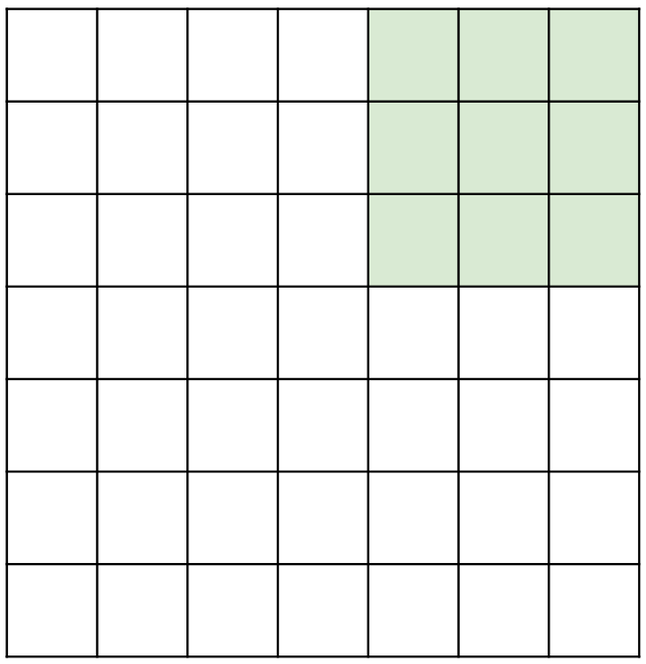
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Dimensions spatiales

A closer look at spatial dimensions:

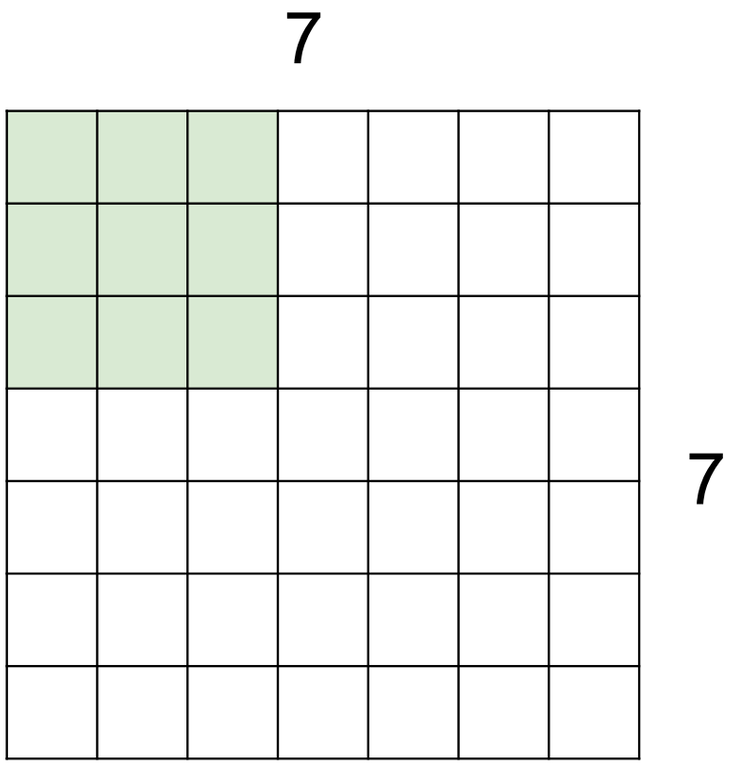


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Dimensions spatiales

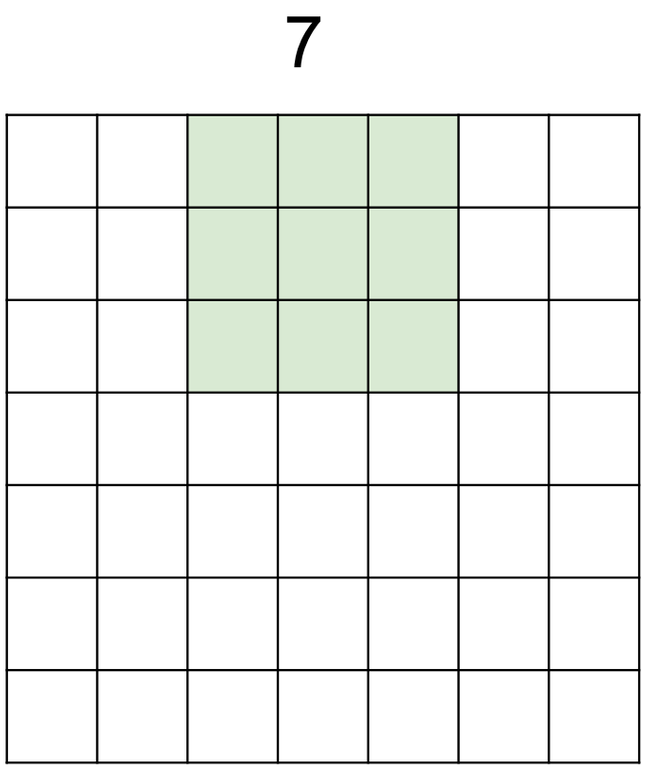
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Dimensions spatiales

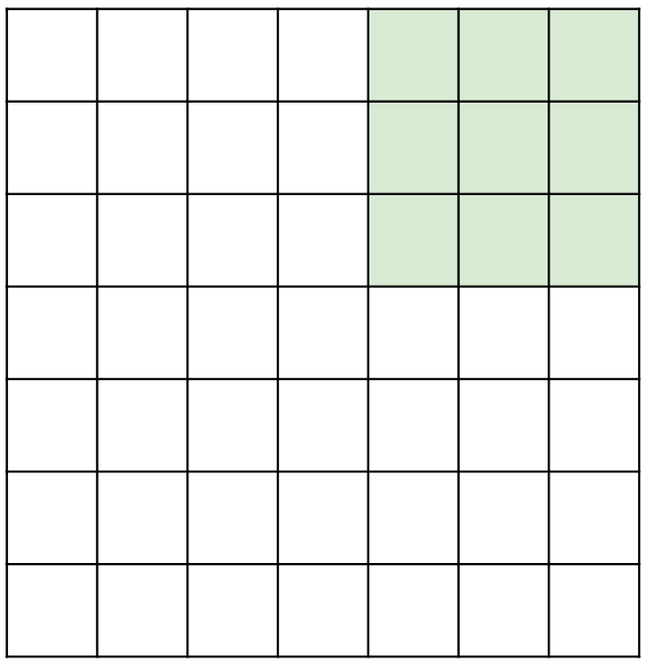
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Dimensions spatiales

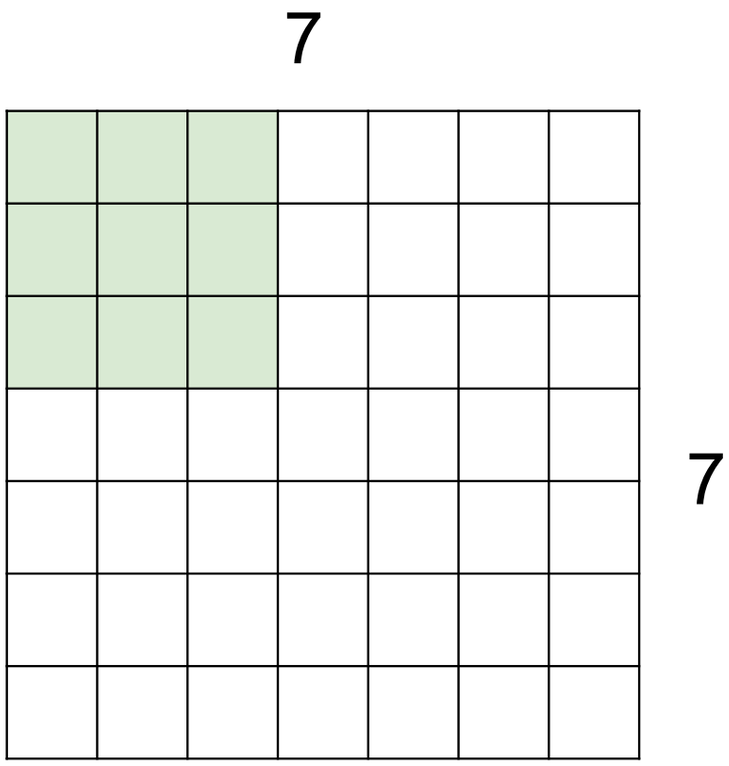
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Dimensions spatiales

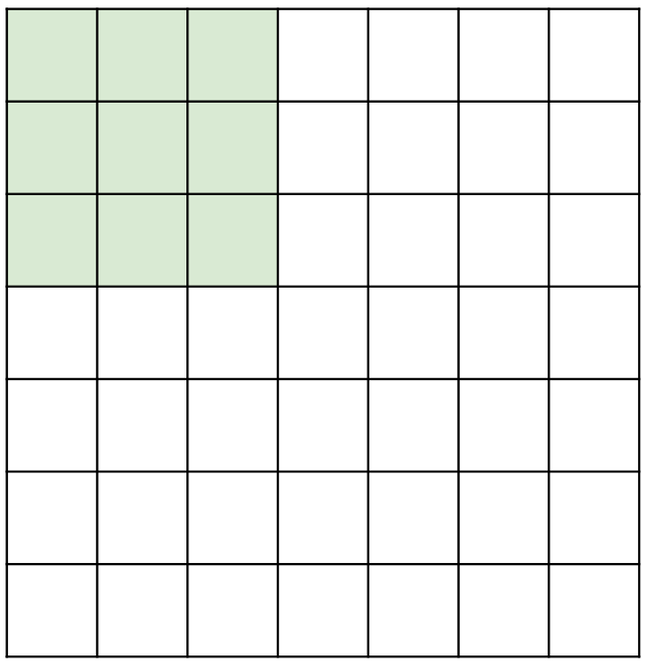
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Dimensions spatiales

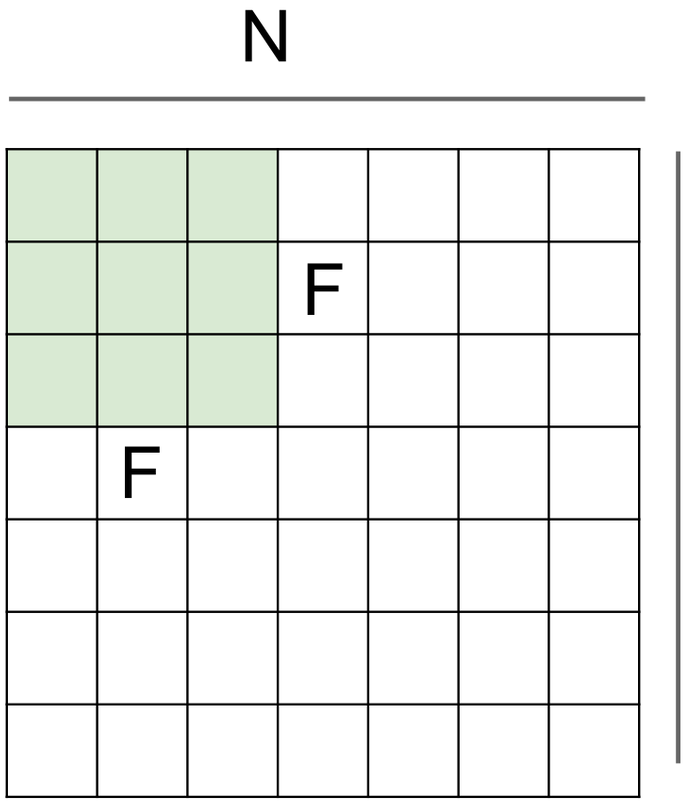
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn’t fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Dimensions spatiales

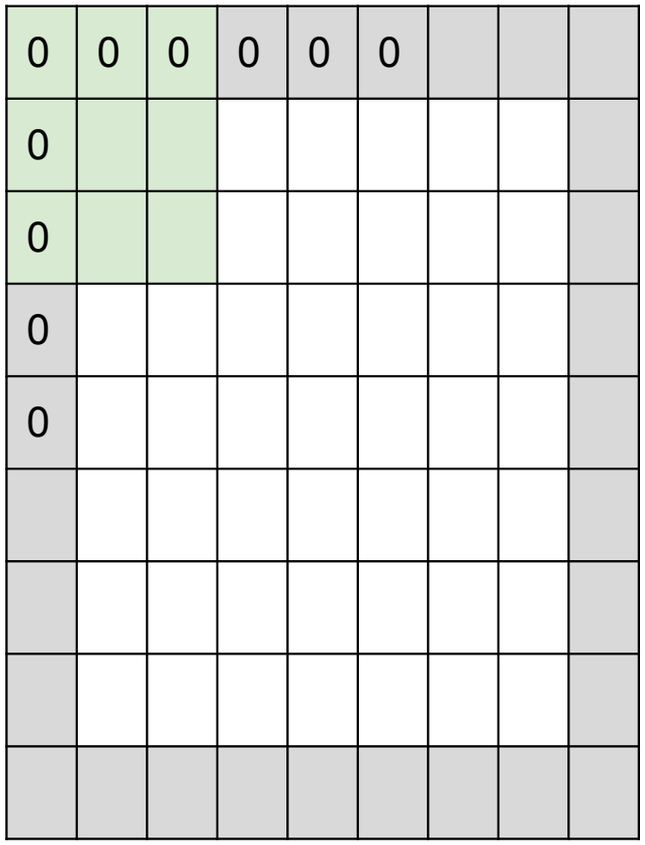


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Dimensions spatiales

In practice: Common to zero pad the border

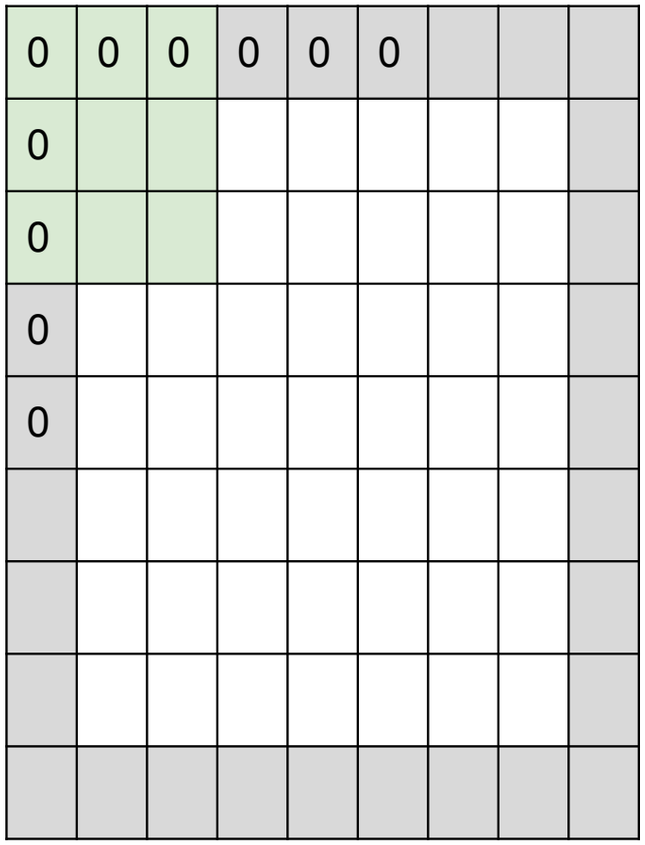


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Dimensions spatiales

In practice: Common to zero pad the border

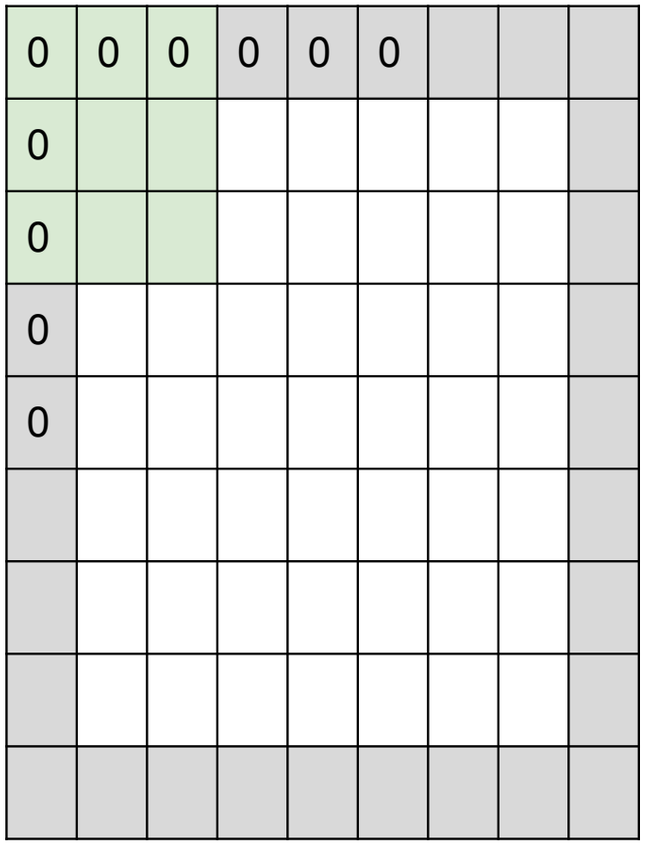


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Dimensions spatiales

In practice: Common to zero pad the border

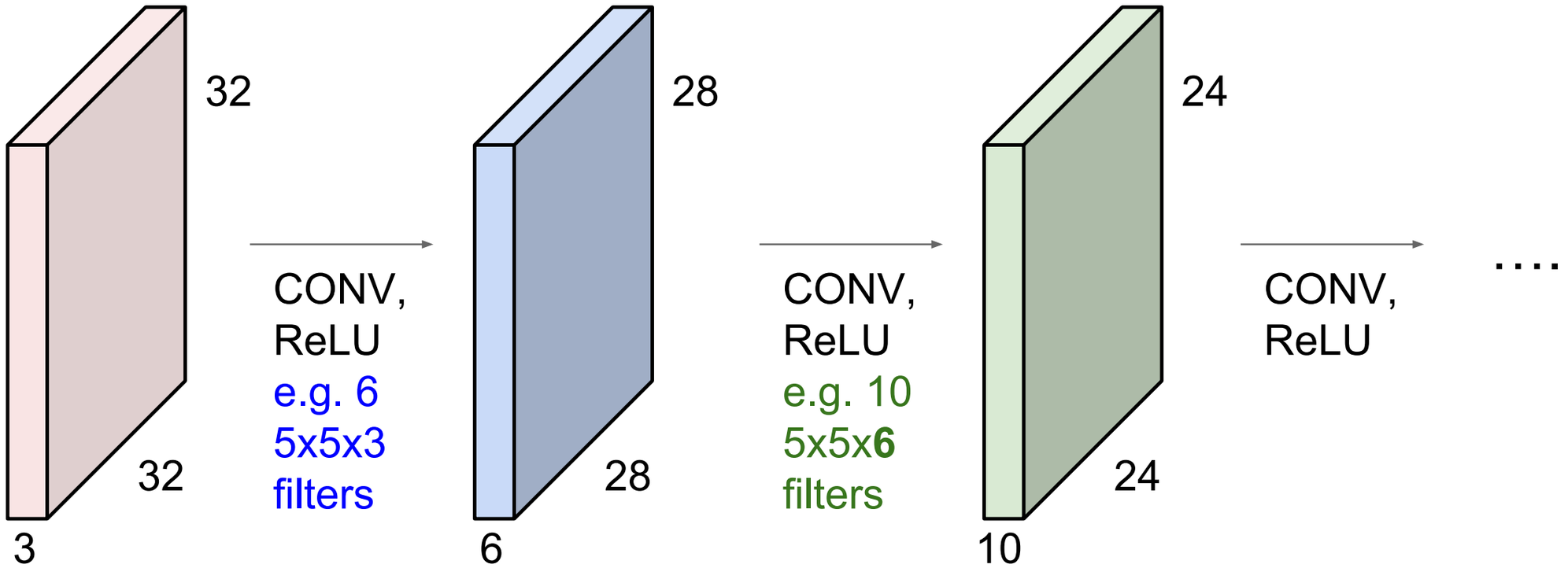


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

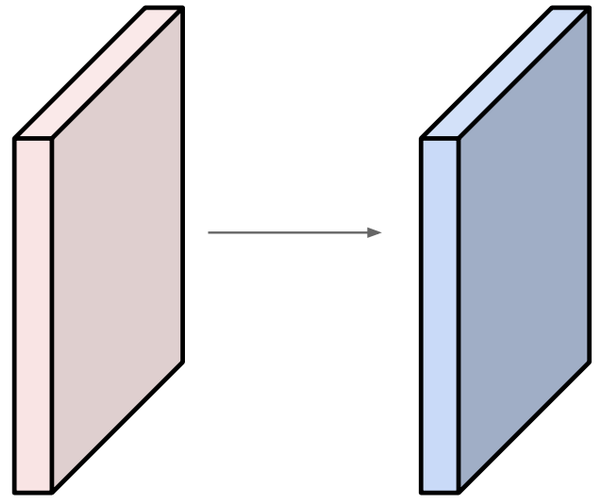
Dimensions spatiales

Remember back to…
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially!
(32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



Dimensions spatiales

Examples time:

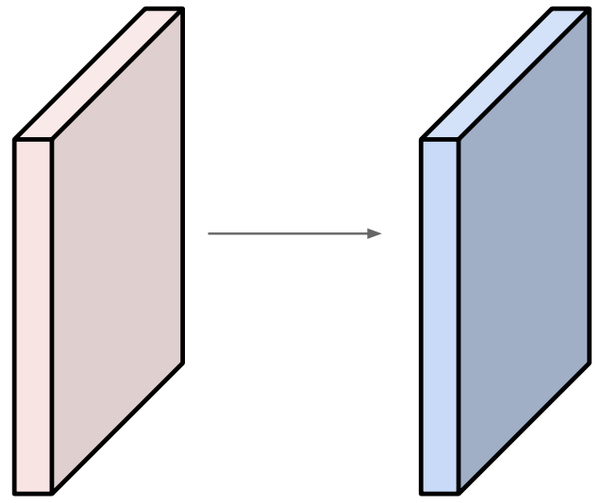


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

Dimensions spatiales

Examples time:

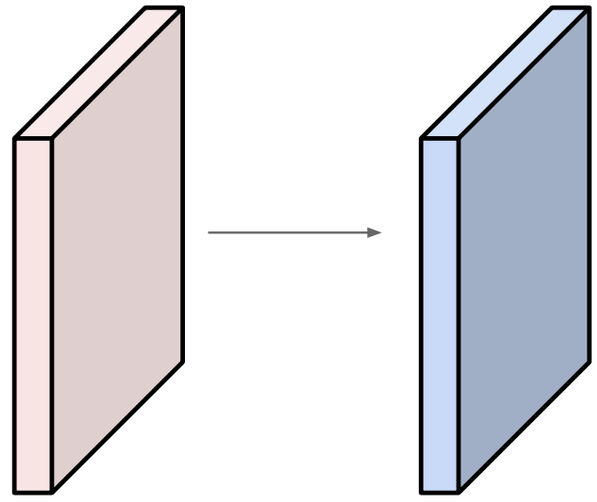


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Dimensions spatiales

Examples time:

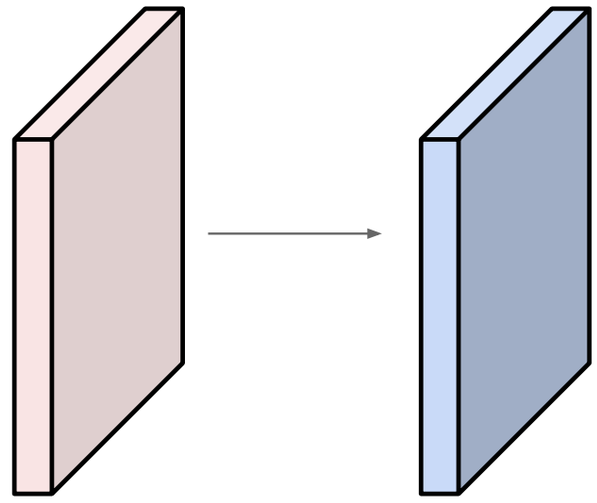


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Dimensions spatiales

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Quelques remarques

- Poids partagés : à la différence des réseaux complètement connectés, les poids de chaque couche de convolution sont réutilisés en tous les positions de l'entrée

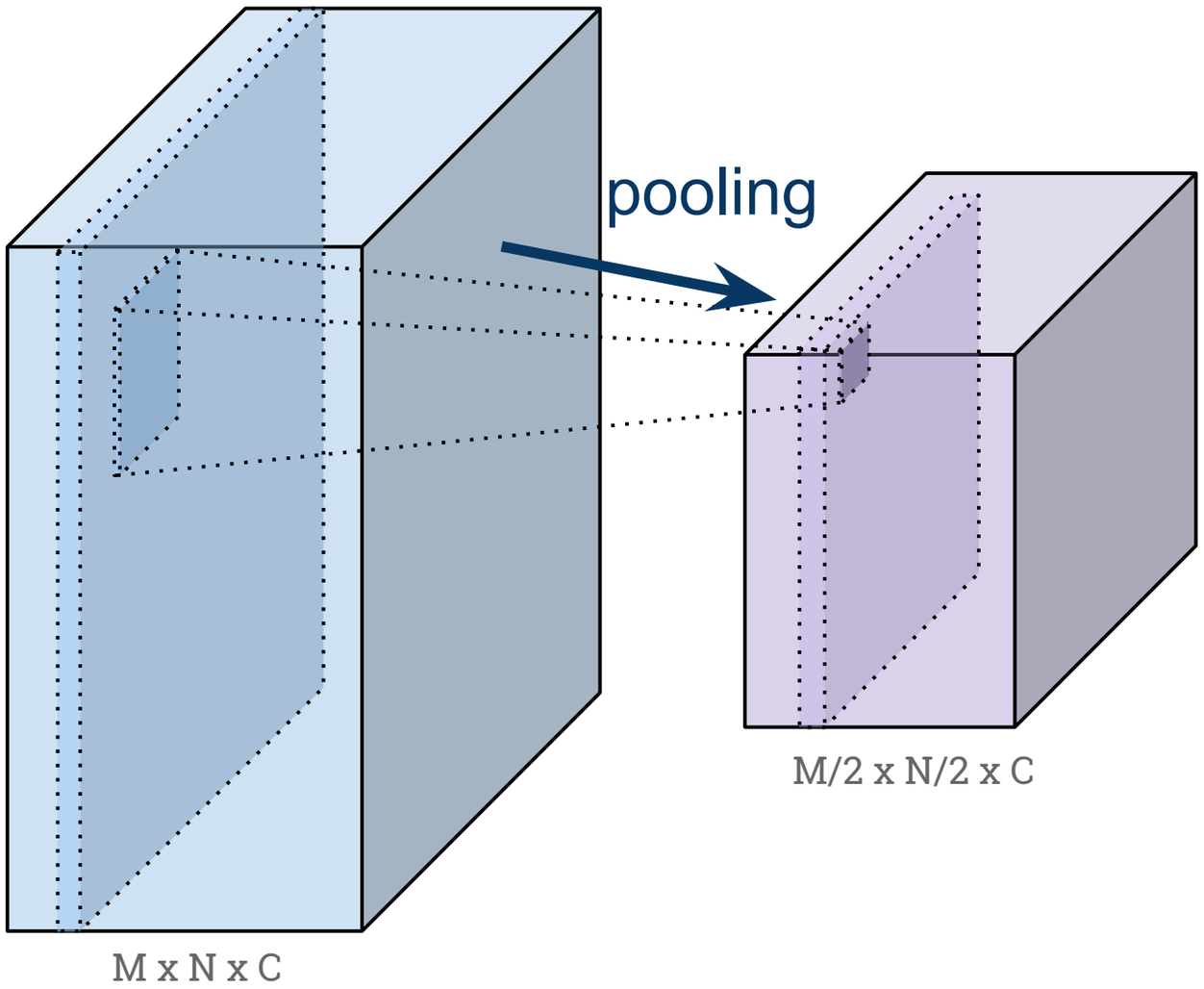
- Invariance : si on translate l'image d'entrée, la sortie du réseau est transitée

- Les couches de convolution permettent de traiter des données de différentes tailles (par exemple, des images), sans avoir besoin de changer l'architecture.

Couche de Pooling (MaxPool)

- But : compresser (sous-échantillonner) la représentation, permet de réduire la taille des données tout en gardant une partie significative

- Appliquée sur chaque couche d'activation séparément



Couche de Pooling

- But : compresser (sous-échantillonner) la représentation, permet de réduire la taille des données tout en gardant une partie significative

- Appliquée sur chaque couche d'activation séparément

Couche de Pooling

- But : compresser (sous-échantillonner) la représentation, permet de réduire la taille des données tout en gardant une partie significative

- Appliquée sur chaque couche d'activation séparément

CNN et Pooling

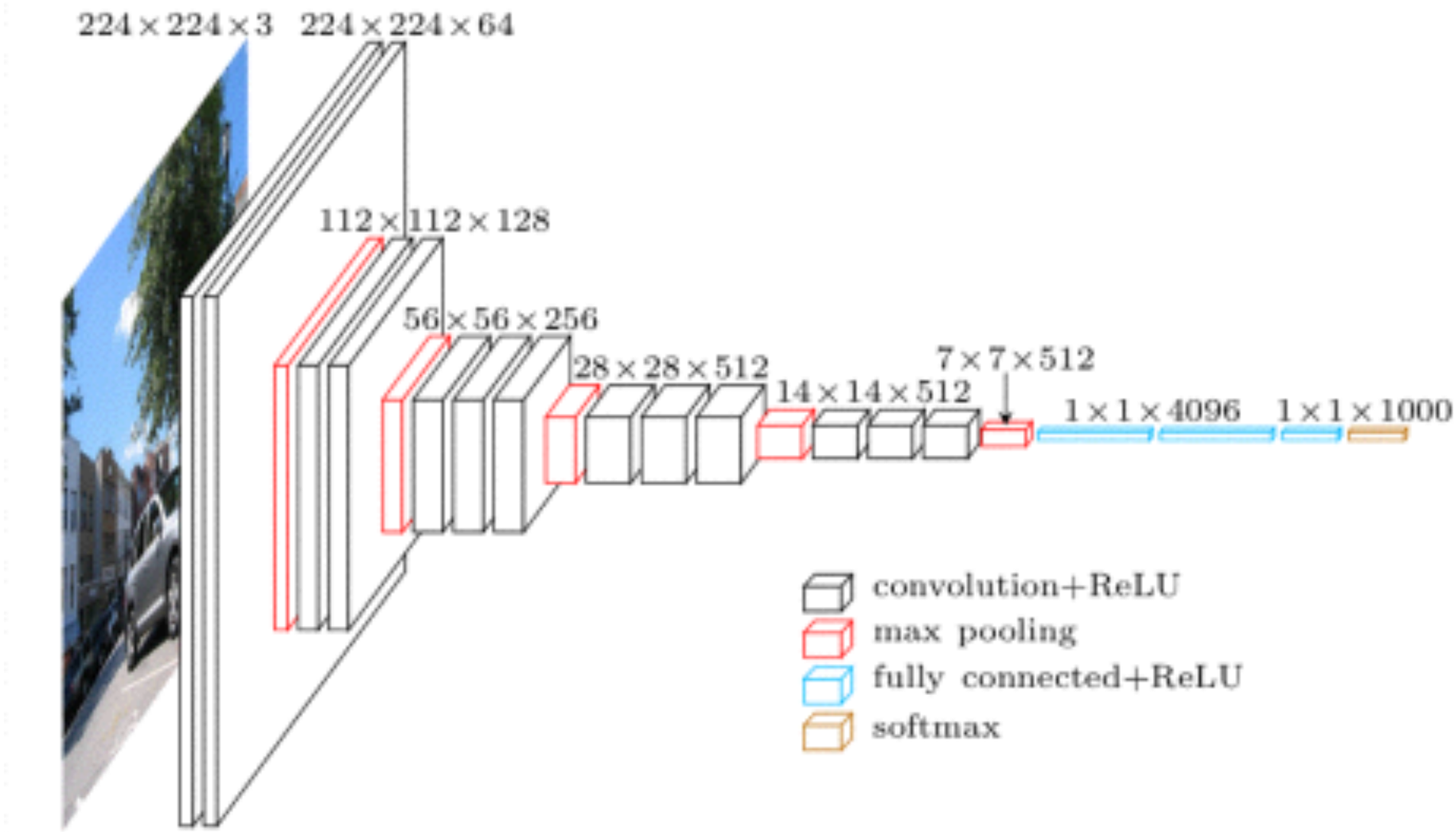
- Un réseau convolutionnel peut être vu comme un cas particulier de réseau de neurones complètement connecté : certains poids sont partagés (identiques) et d'autres mis à zéro

- Cela peut être vu comme un prior sur la distribution des poids d'un réseau complètement connecté, afin que la sortie soit invariante aux translations

- Attention, le fait que le réseau soit convolutionnel, avec des étapes de Pooling, peut provoquer des problèmes de sous-ajustement

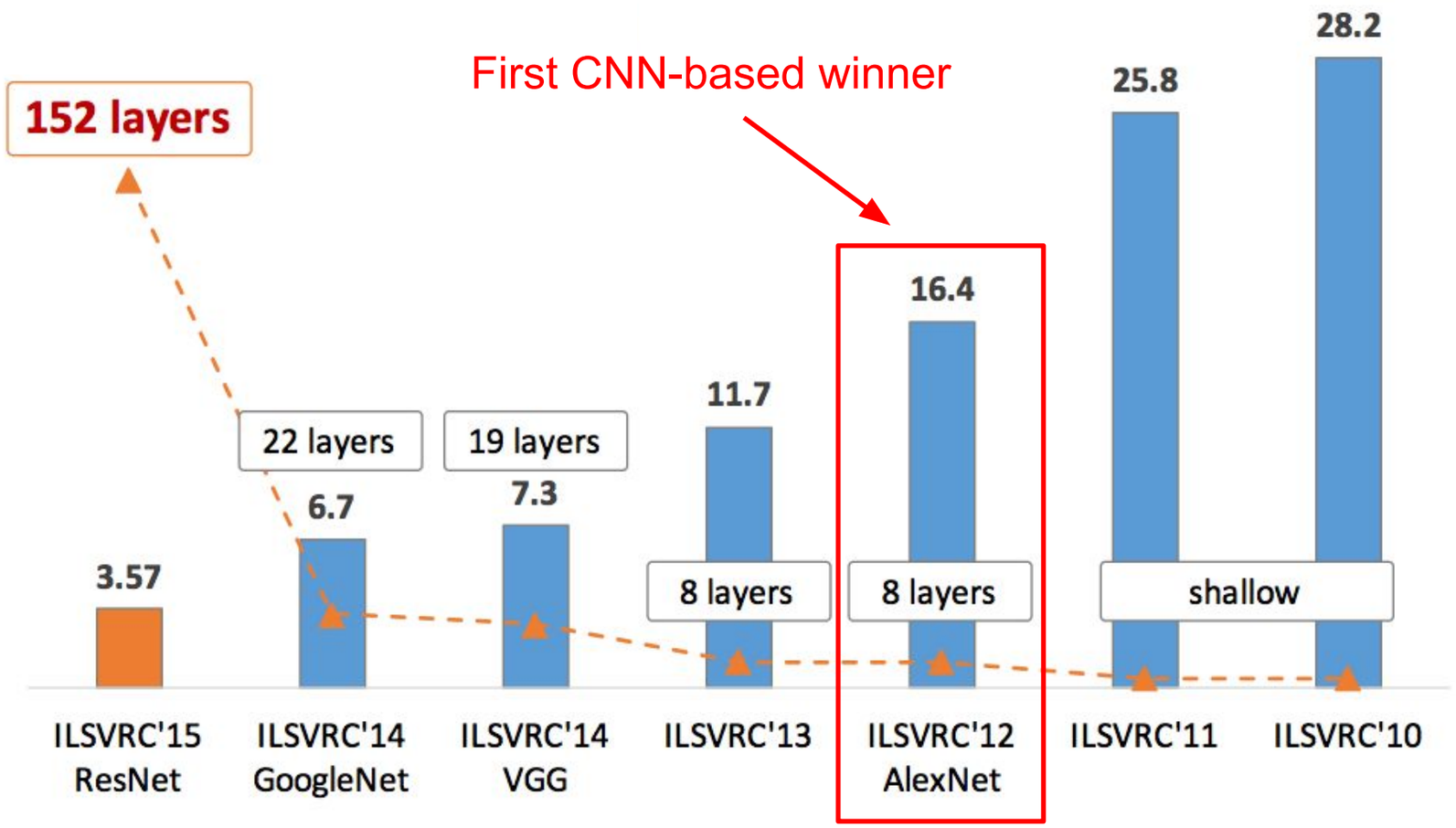
Exemple de CNN : VGG-16

- CNN proposé par K. Simonyan et A. Zisserman (Oxford) en 2014 dans l'article "*Very Deep Convolutional Networks for Large-Scale Image Recognition*". Le réseau atteint 92.7% de précision dans le challenge ImageNet (base de 14 million images avec 1000 classes). 500Mo sur le disque



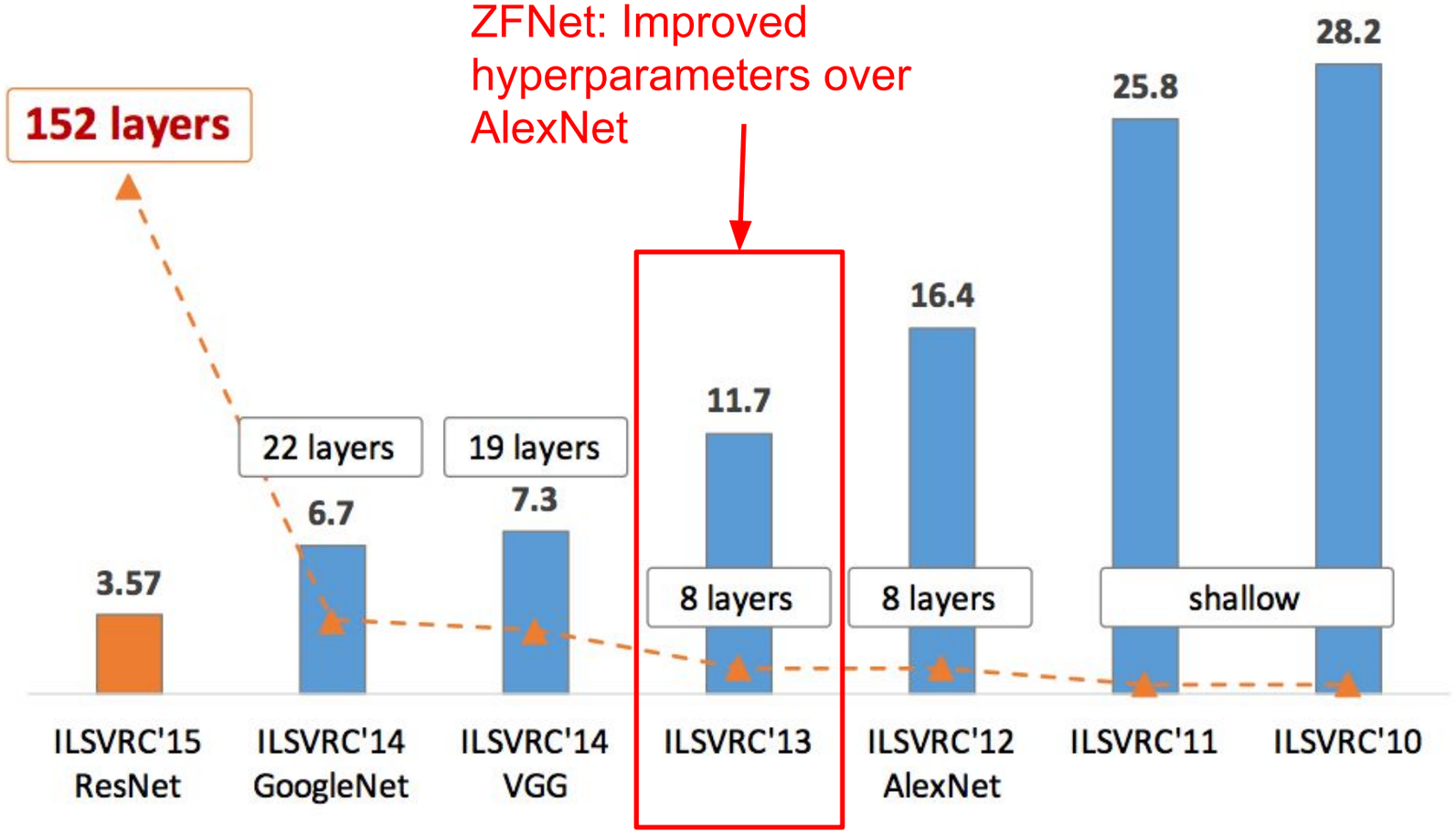
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



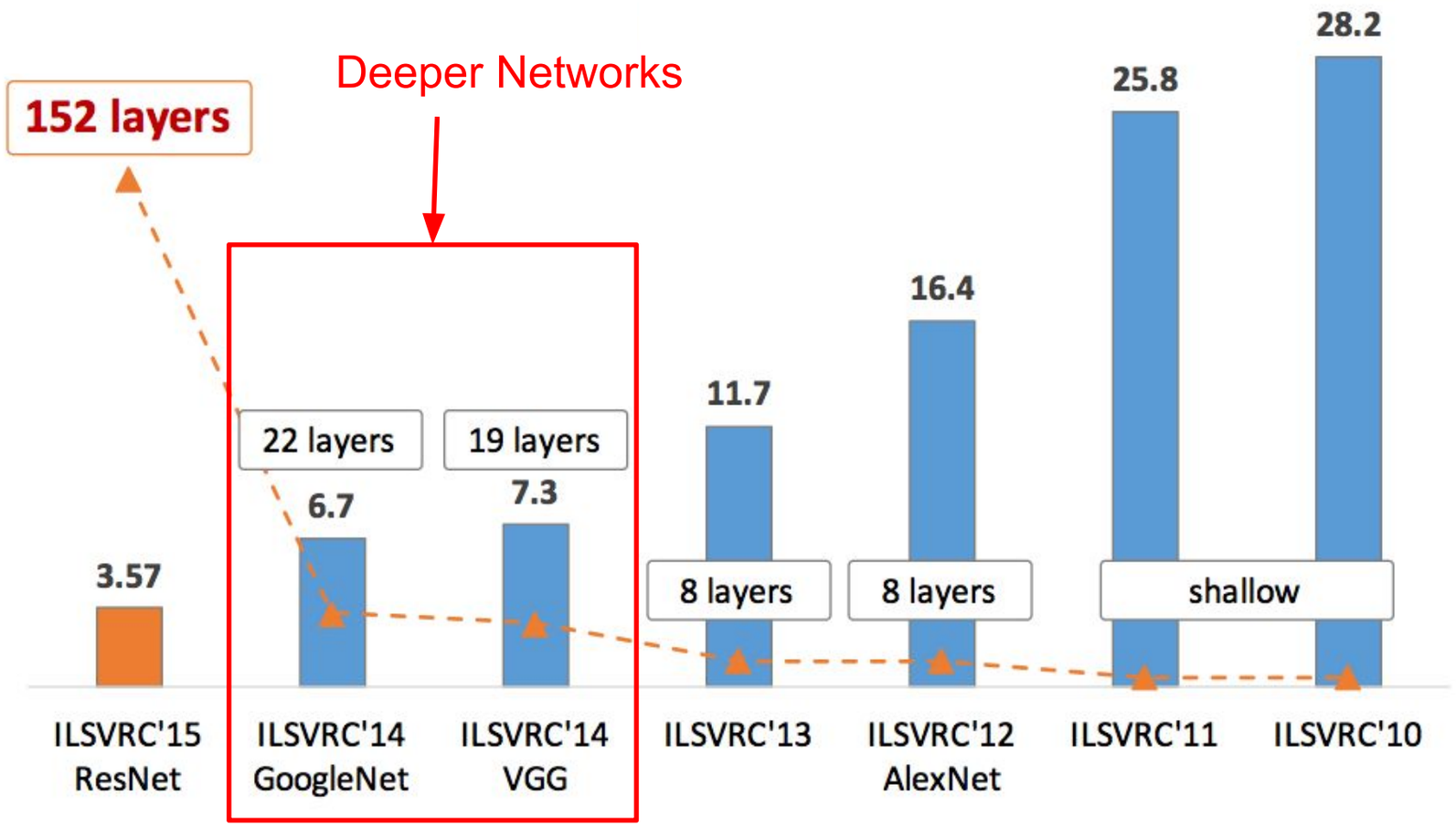
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

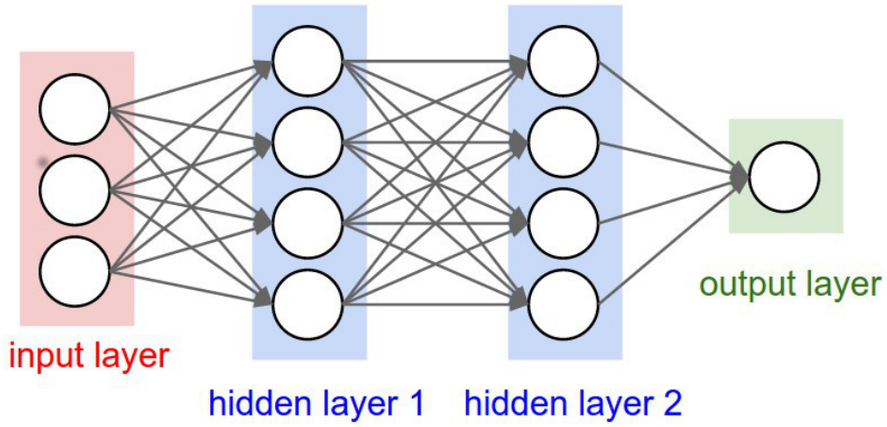


ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

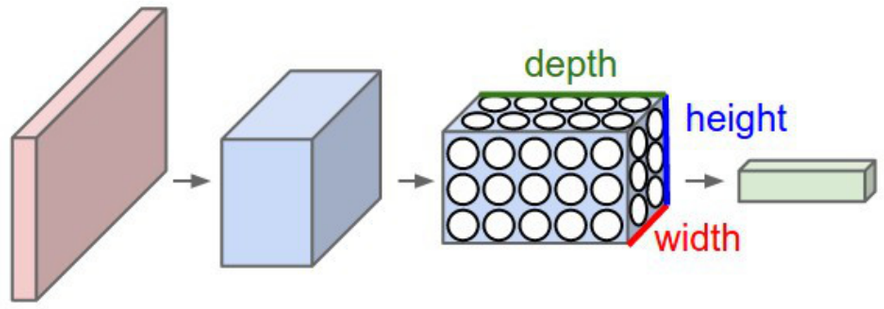
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Résumé



réseau complètement connecté

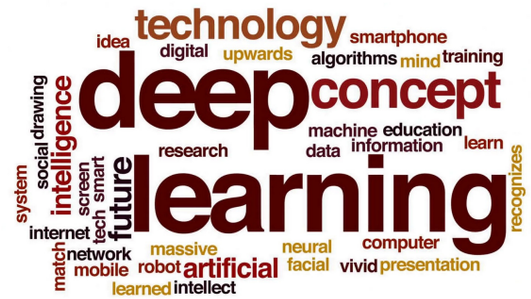


réseau convolutionnel

- Réseau convolutionnel : Trois hyperparamètres contrôlent la taille du volume généré dans une couche de convolution:
  - Profondeur: nombre de filtres (hyperparamètre)
  - *Stride* : pas de convolution S (généralement 1 ou 2, rarement plus grand)
  - *Zero-padding* : le volume est agrandi pour contrôler le volume de sortie. La largeur de l'image de sortie est $W_o = (W - F + 2P) / S + 1$ (similaire pour la hauteur)
- Il faut aussi préciser la taille du filtre de convolution $F \times F$.

Environnements pour le Deep Learning

theano



Caffe2

PYTORCH

TensorFlow

torch

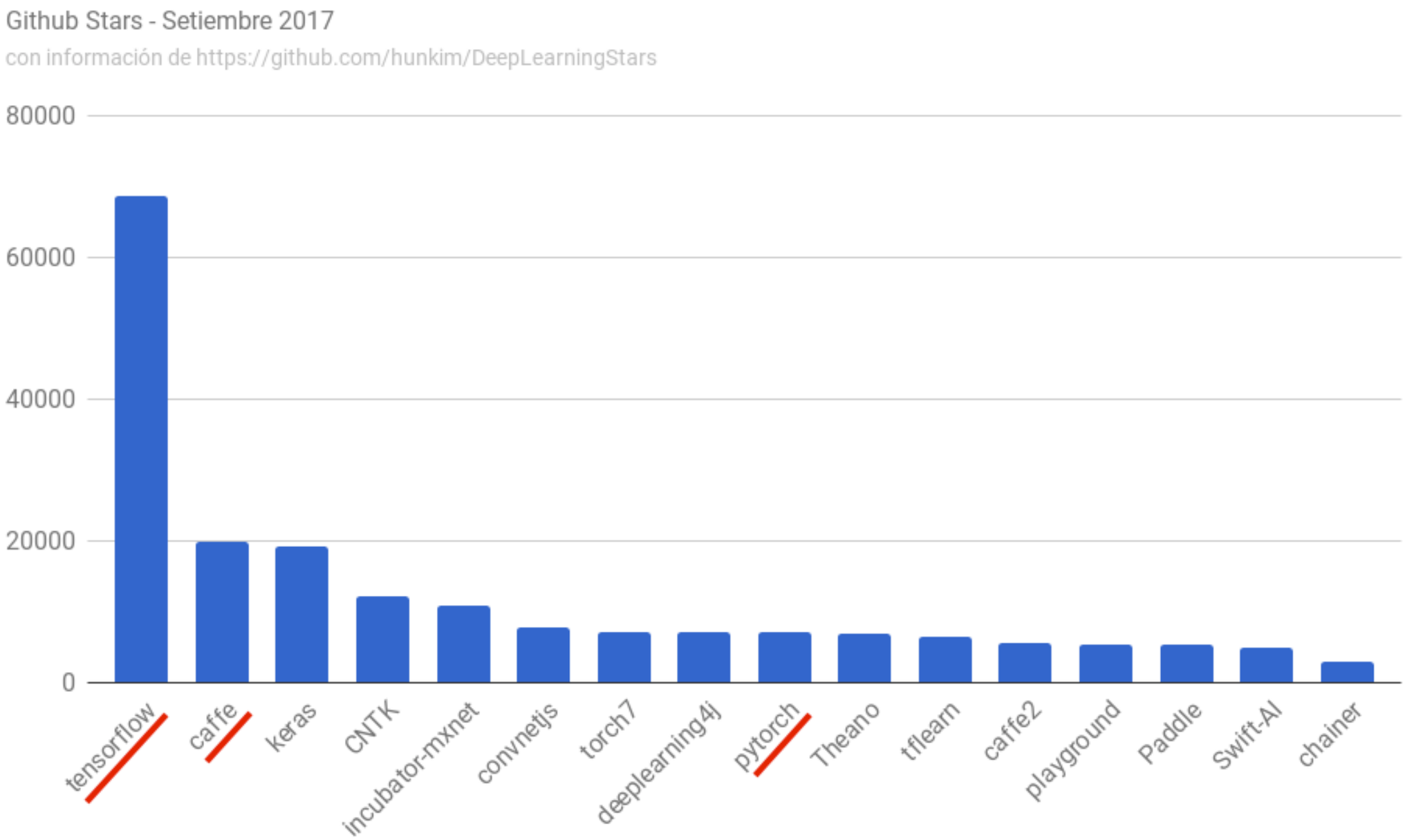
GitHub

Caffe

Environnements pour le Deep Learning

- Caffe → Caffe2 (UC Berkeley, Facebook)
- Theano → **TensorFlow** (U. Montreal → Google)
- Torch → **PyTorch** (NYU → Facebook)
- Autres : MatConvNet (Oxford U.), MXNet (Amazon), CNTK (Microsoft), Paddle (Baidu), etc...

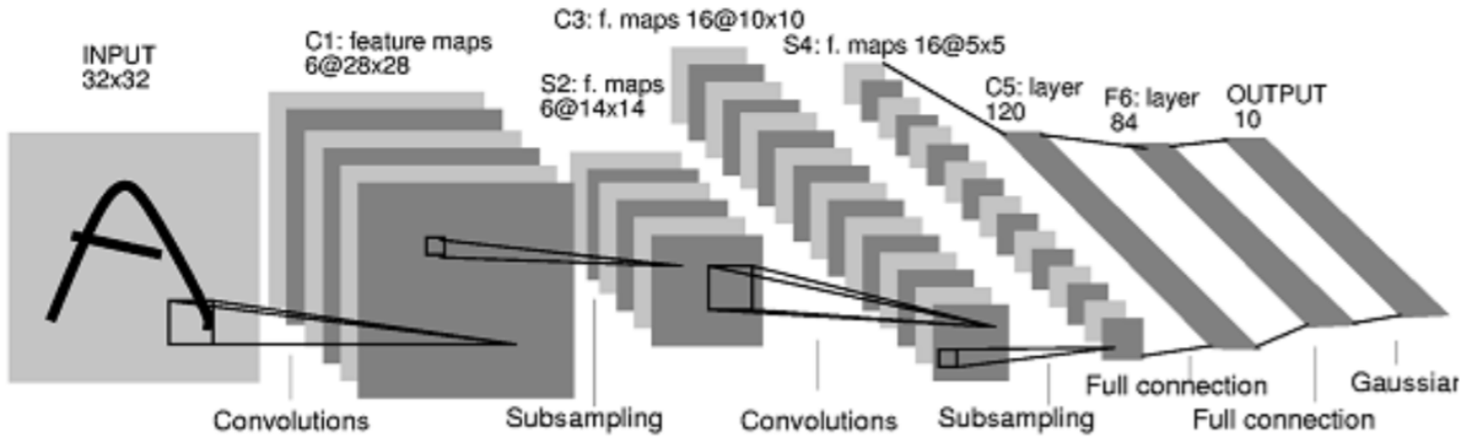
Environnements pour le Deep Learning



Evolution des architectures

1998

LeCun et al.



# of transistors

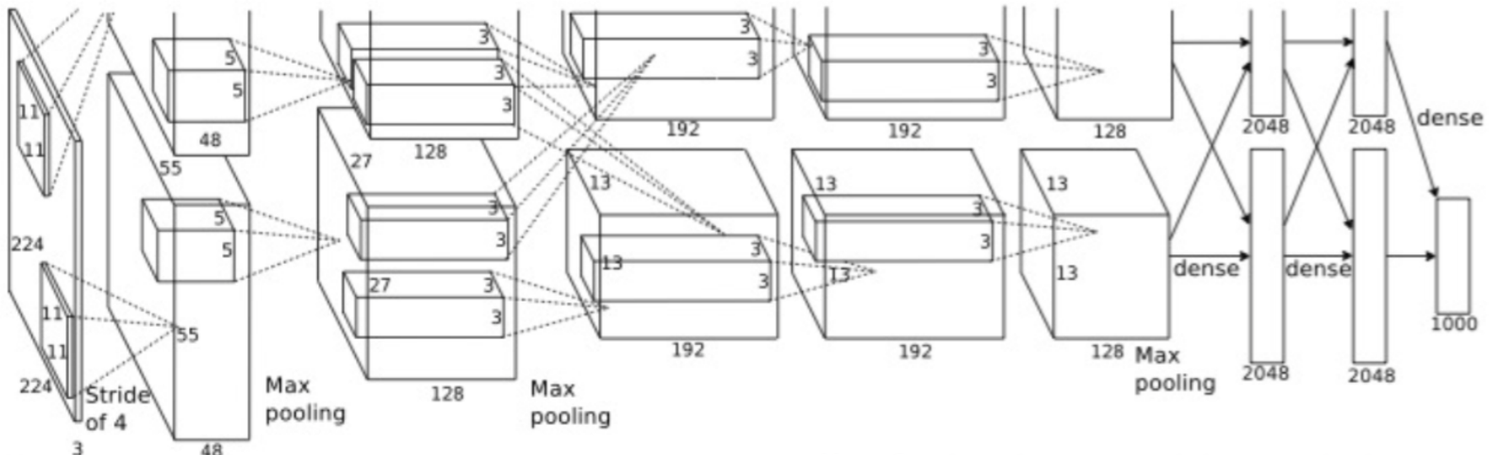
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors GPUs

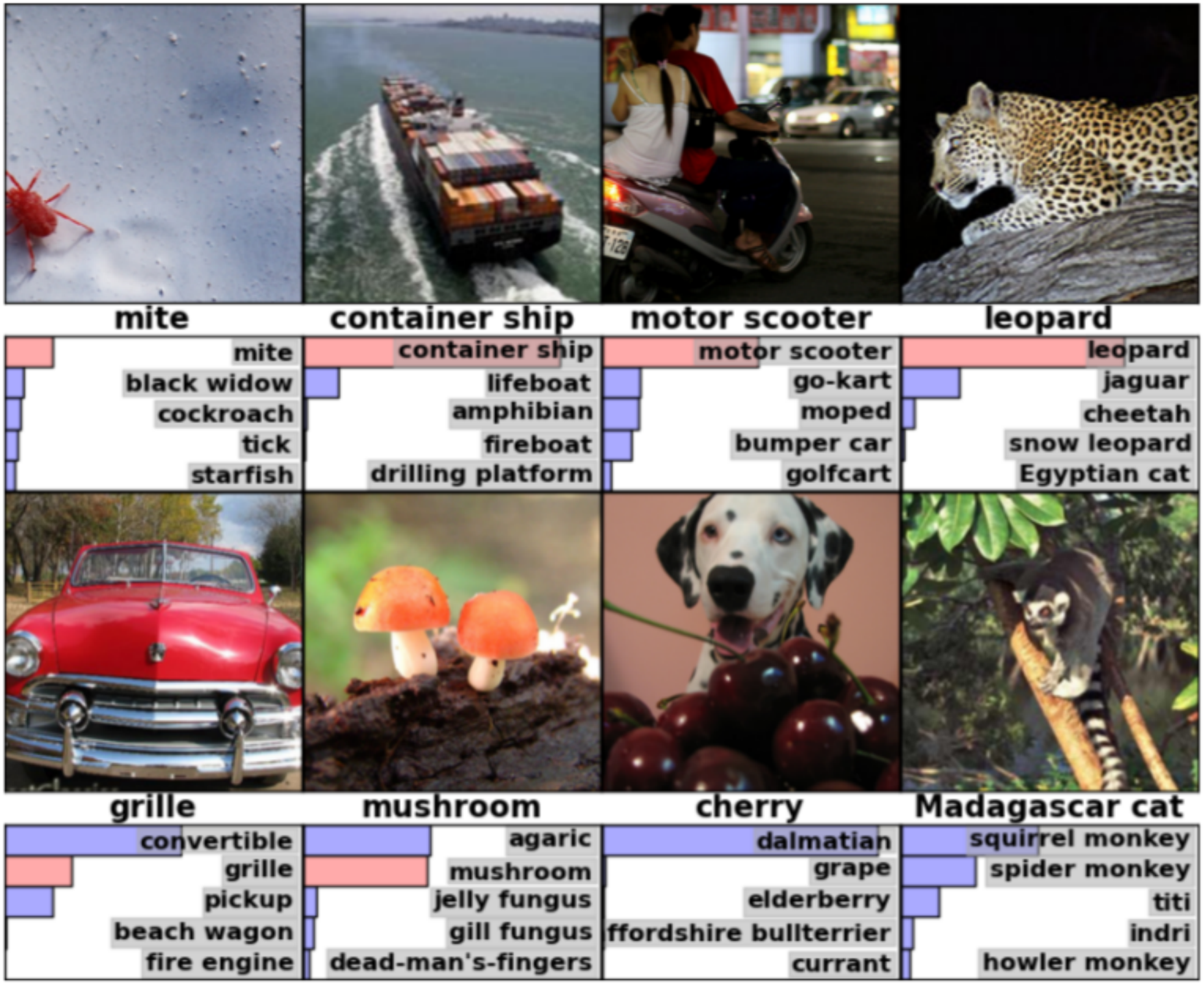
10^9

# of pixels used in training

10^{14} IMAGENET

Quelques applications

Classification d’images



Krizhevsky, Alex, Ilya Sutskever, Geoffrey E. Hinton.
“Imagenet classification with deep convolutional neural networks.”, NIPS 2012. (13k citas)

Vision par ordinateur

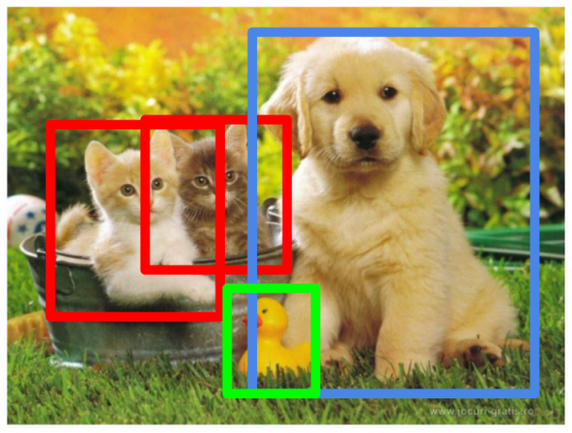
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

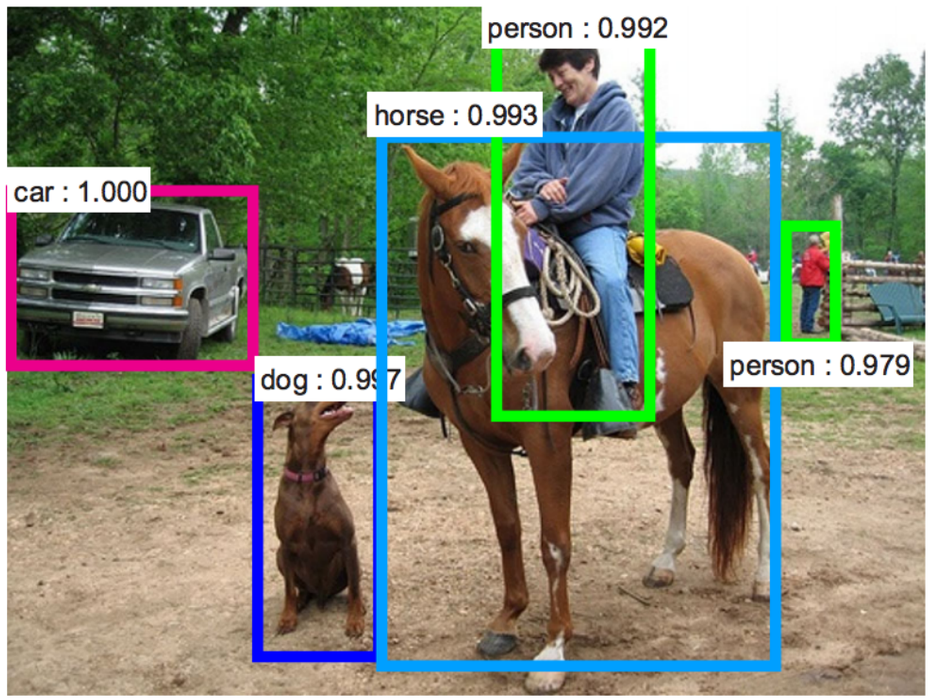
Instance Segmentation

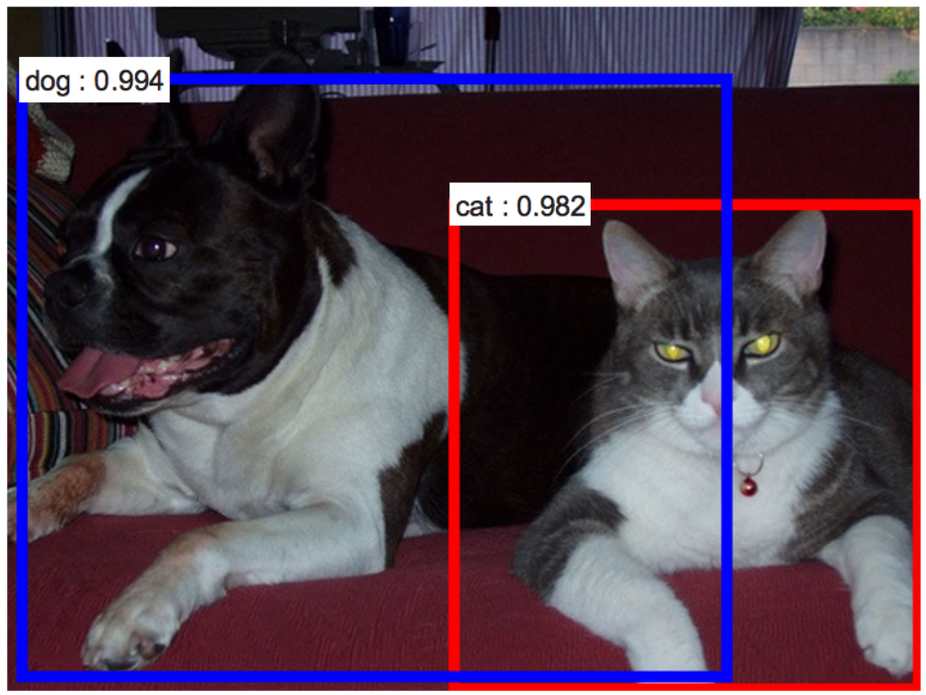
CAT, DOG, DUCK

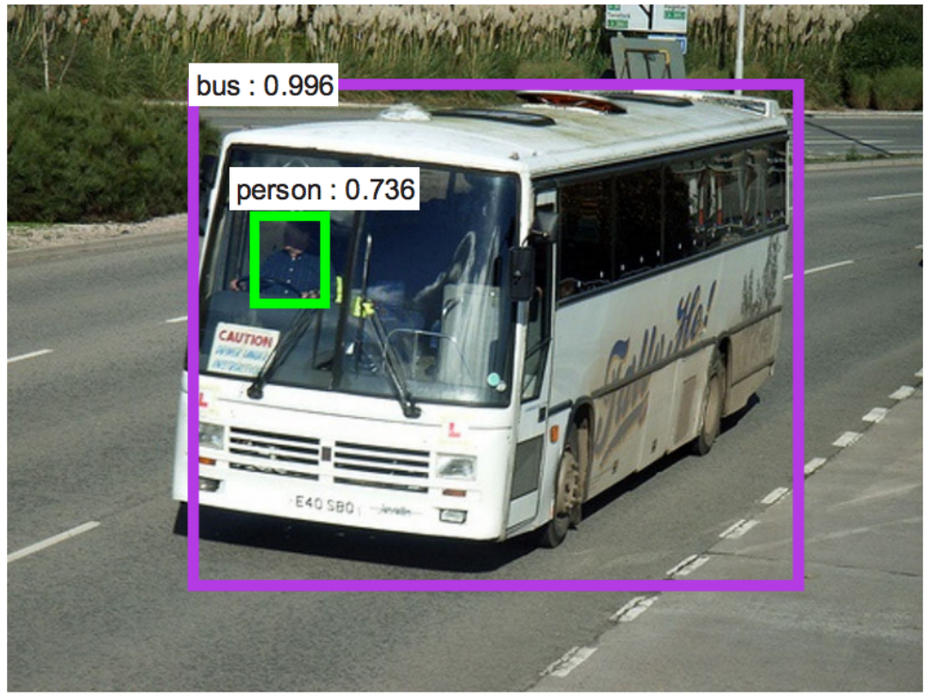
Single object

Multiple objects

Détection d'objets







Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun.
"Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS 2015 (1.3k cit

Segmentation

Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick , “Mask R-CNN.”, ICCV 2017

Détection de pose

Figure 10. Results containing viewpoint and appearance variation, occlusion, crowding, contact, and other common imaging artifacts.

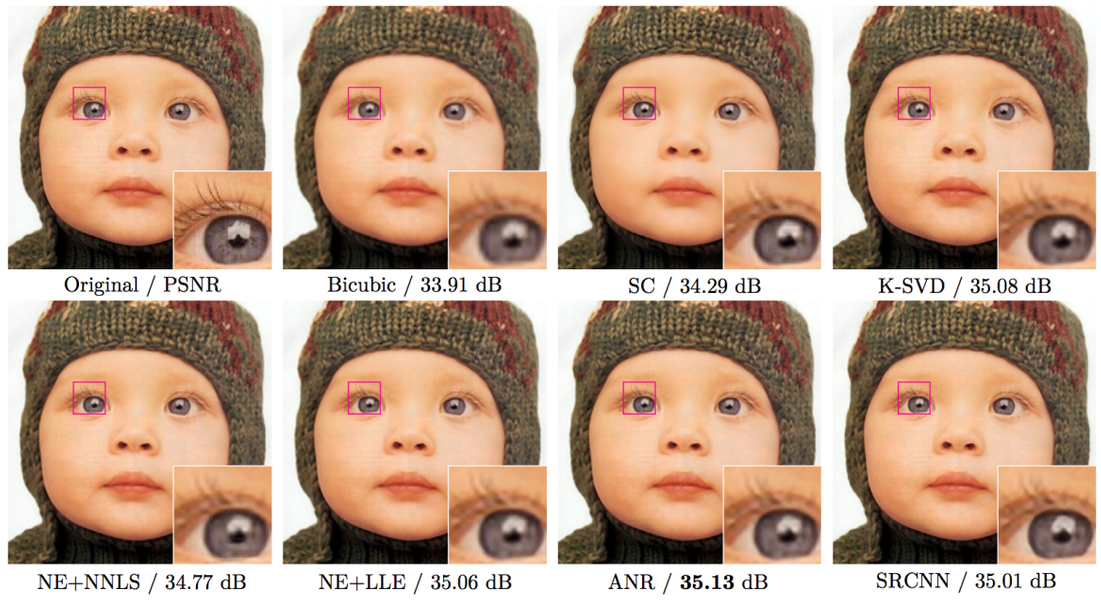
▸ OpenPose

Zhe Cao and Tomas Simon and Shih-En Wei and Yaser Sheikh
"Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.", CVPR 2017

https://www.youtube.com/watch?v=pW6nZXeWlGM

Restauration d'image : défloutage, superrésolution

Su, Shuochen, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang "Deep video deblurring.", CVPR 2017



Dong, Chao, Chen Change Loy, Kaiming He, Xiaoou Tang. "Learning a deep convolutional network for image super-resolution.", ECCV 2014

Sous-titrage automatique

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

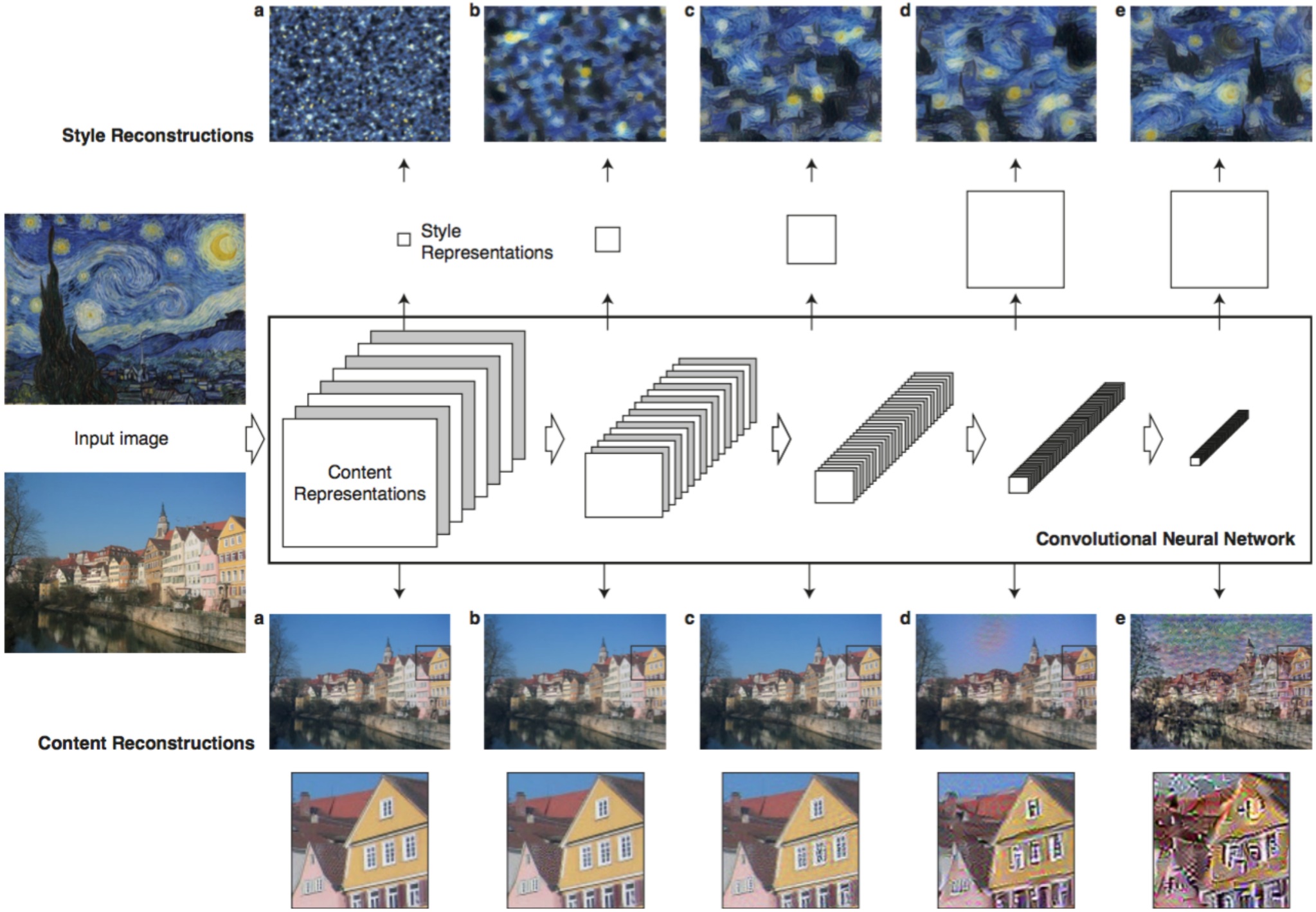
Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

Captions generated by Justin Johnson using Neuraltalk2

Transfert de style

Transfer Learning : réutilisation de réseaux déjà appris pour d'autres tâches (ici VGG-16)



Gatys et al. 2015

Transfert de style

Gatys et al. 2015