4. Réseaux de neurones convolutifs

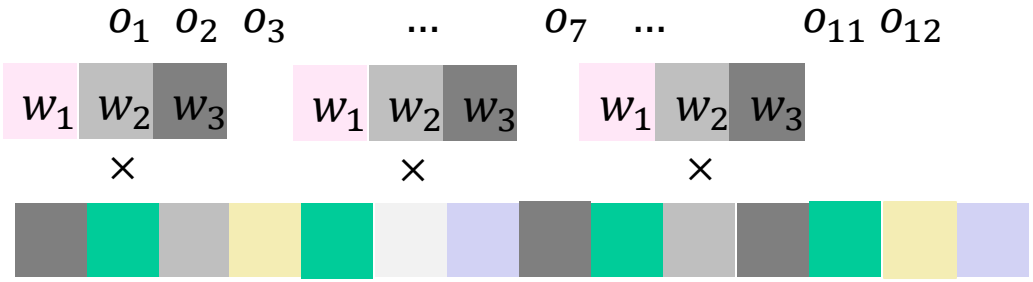
Réseaux de neurones convolutifs, ou à poids partagés

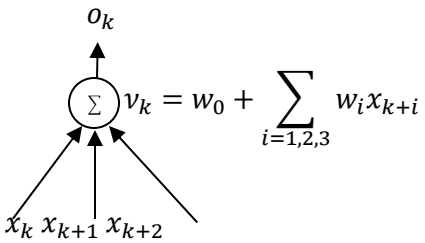
Y. LeCun, & al. , Backpropagation Applied to Handwritten Zip Code Recognition; AT&T Bell Lab., 1989.
L'un des premiers systèmes qui apprend les poids des filtres convolutifs

- ✓ Apprend à extraire des caractéristiques locales dans les images
- ✓ Chaque couche produit des cartes de caractéristiques
- ✓ Ils sont peu gourmands en paramètres car les poids sont les mêmes (partagés) pour différentes sous-parties des entrées
- ✓ mettent en œuvre un mécanisme de convolution (filtrage), induisant une fenêtre glissante
- ✓ Les poids des filtres sont appris par rétro-propagation du gradient

Cas 1D :

- ✓ le neurone se déplace le long du vecteur d'entrée (fenêtre glissante) selon un certain pas
- ✓ pour chaque position il produit une sortie, résultat d'une convolution





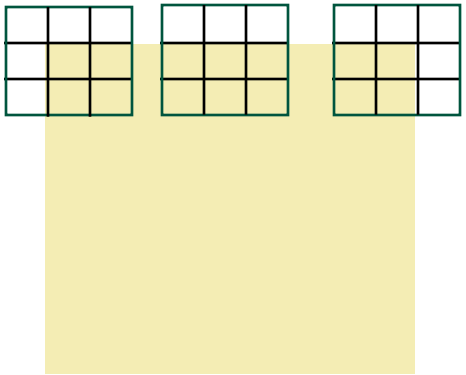
- ✓ les trois coefficients (éventuellement le biais) définissent le neurone, et sont « partagés » sur toutes les entrées

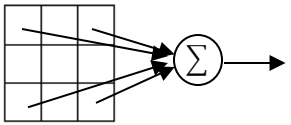
4. Réseaux de neurones convolutifs

Exemple en 2D, pour un seul filtre

1. une fenêtre glissante de taille 3×3 définie par 9 poids $w_{i,j}$ et un biais w_0 se déplace sur l'image d'entrée.
2. Elle produit une image filtrée (une carte) de même taille.
3. L'image produite peut être sous échantillonnée à travers une autre fenêtre glissante (2 X 2) dans laquelle on conserve la moyenne ou la valeur max (Max / Average Pooling)

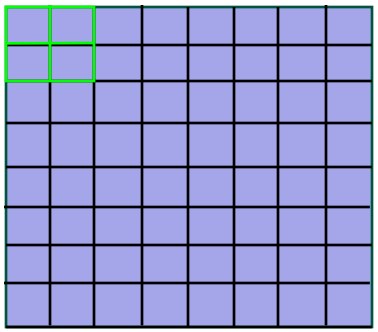
Convolution (3,3)





$$(v_{l,m}) = k_0 + \sum_{i,j=-1,0,1} k_{i,j} I_{l+i,j+m}$$

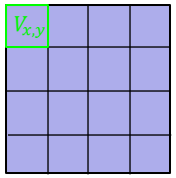
$$\forall\, l = 0, \ldots, W$$
$$\forall\, m = 0, \ldots, H$$

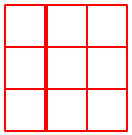


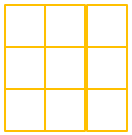
Max Pooling 2x2

$$V_{x,y} = \max_{l,m=0,1} y_{2x+l,2y+m}$$

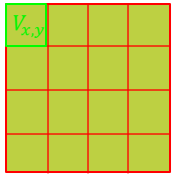
$$\forall\, x = 0, \ldots, W/2$$
$$\forall\, y = 0, \ldots, H/2$$

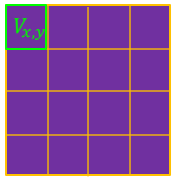






on répète l'opération avec N filtres différents
on produit N cartes de caractéristiques

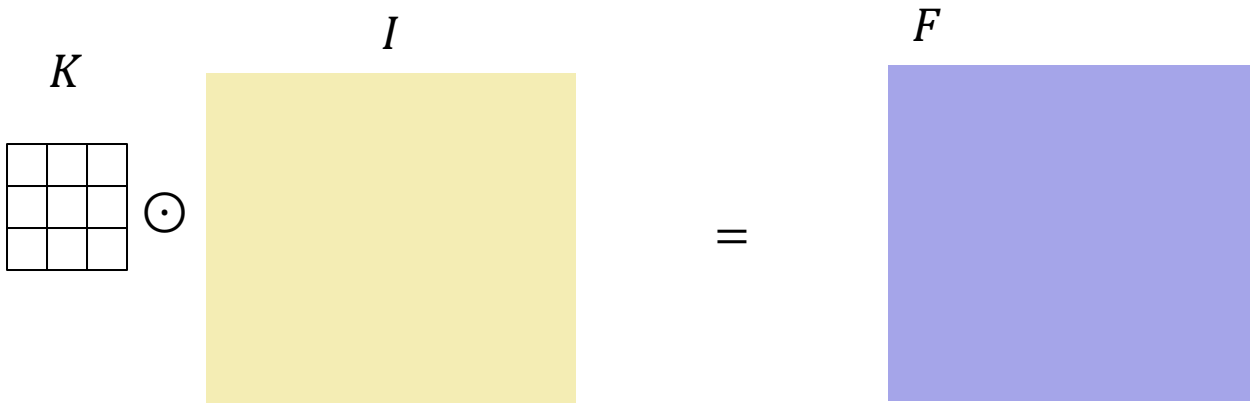




4. Réseaux de neurones convolutifs

Noyau de Convolution

$$K \odot I = F$$



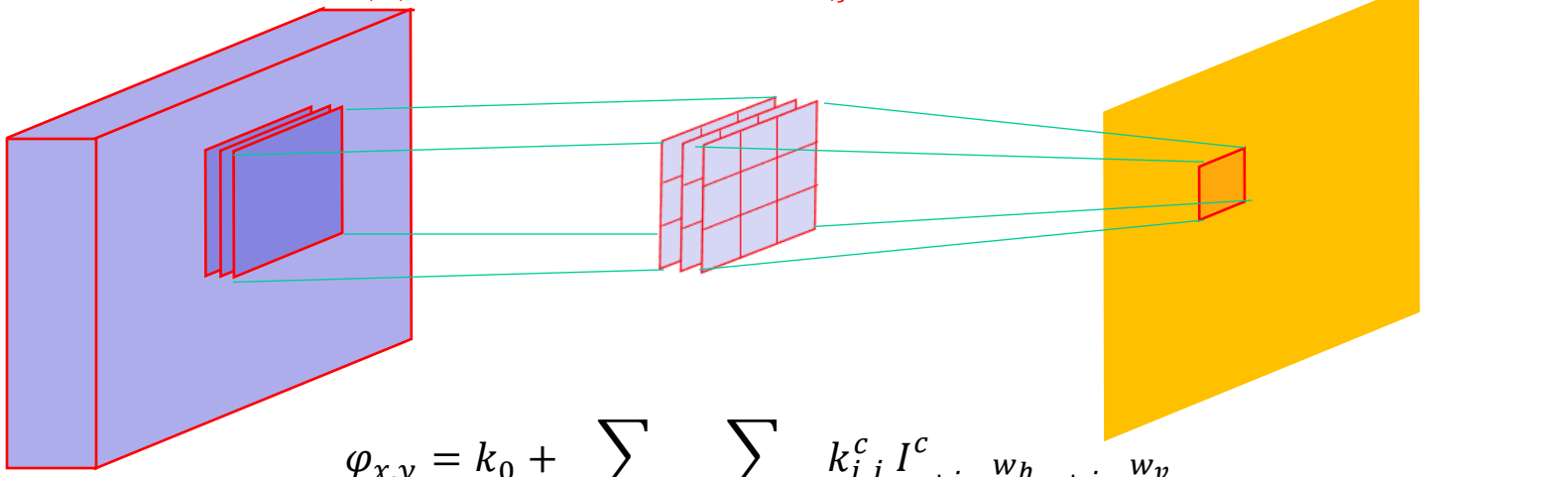
4. Réseaux de neurones convolutifs

Convolution multi-canal

- L'image d'entrée comporte plusieurs canaux (Rouge, Vert, Bleu…), ou bien elle est elle même le résultat d'une couche de filtrage par N filtres, dans ce cas elle comporte N canaux $C_i, i = 1, \dots, N$

- On étend la convolution à C canaux: $\mathrm{K} = \left(k_{i,j}^{c}\right)$ est le noyau de convolution 3D qui produit une carte de caractéristiques ϕ

$I = \left(I_{x,y}^{c}\right)_{c=1,\dots,C}$ $\quad K = \left(k_{i,j}^{c}\right)$ $\quad \phi = K \odot I = \left(\varphi_{x,y}\right)$



$$\varphi_{x,y} = k_0 + \sum_{c=1,\dots,C} \sum_{\substack{i=1,\dots,w_h \\ j=1,\dots,w_v}} k_{i,j}^{c} \, I_{x+i-\frac{w_h}{2},y+j-\frac{w_v}{2}}^{c}$$

- Exemple: un noyau 3×3 à 3 canaux comporte et $3 \times 3 \times 3 + 1 = 28$ paramètres $k_{i,j}^{c}$

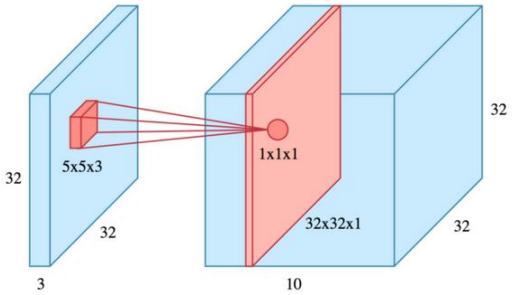
4. Réseaux de neurones convolutifs

Réseaux de neurones convolutifs

- Chaque noyau définit un filtre, et produit une carte de caractéristiques
- Chaque filtre travaille sur les N canaux de l'image d'entrée (conv. 3D)
- Une couche de N noyaux définit N filtres et produit N cartes de caractéristiques

Exemple

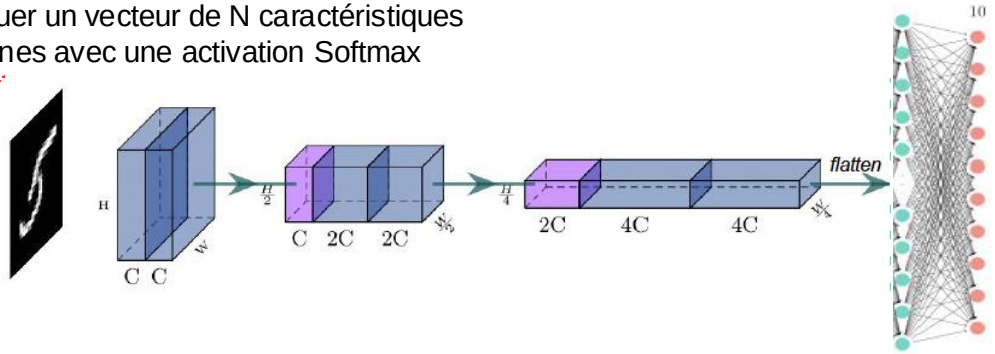
une image d'entrée de taille 32×32
comportant 3 *canaux d'entrée*
est filtrée par 10 *noyaux* 5×5
pour produire 10 cartes de taille 32×32
il y a au total $(5 \times 5 \times 3 + 1) \times 10 = 760$ *paramètres*



Exemple de réseau convolutif

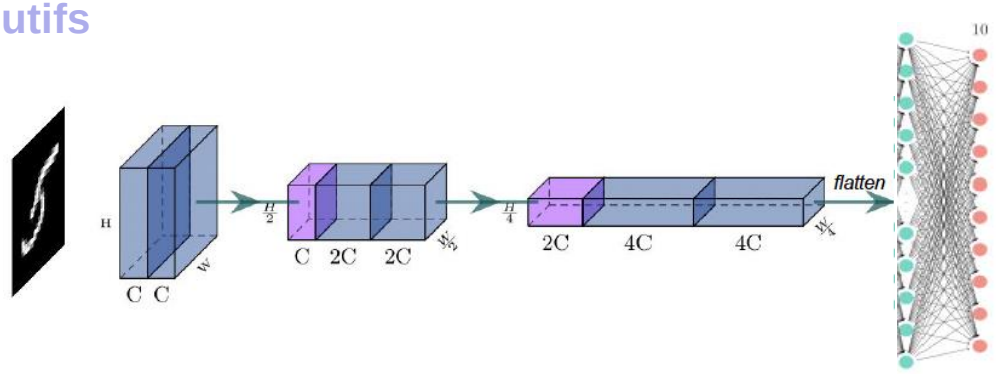
- ✓ 6 couches de convolution (bleu) et 2 couches de pooling (rose)
- ✓ La dernière carte est aplatie (flatten) pour constituer un vecteur de N caractéristiques
- ✓ Ce vecteur est réduit en une couche de 10 neurones avec une activation Softmax
- ✓ L'ensemble est entrainé par descente de gradien

$$L \times H = 28 \times 28$$
$$C = 8,\ 4C = 32$$
$$N = 32 \times \left(\frac{28 \times 28}{16}\right) = 32 \times 49 = 1568$$



4. Réseaux de neurones convolutifs

Réseaux de neurones convolutifs



$L \times H = 28 \times 28$
$C = 8,\ 4C = 32$
$N = 32 \times \left(\frac{28 \times 28}{16}\right) = 1568$

Nombre de paramètres pour des conv(3×3):

pour les deux premières couches
$C \times (9 + 1) + C \times (9 \times C + 1) = 8 \times 10 + 8 \times 73 = 664$
$+$
pour les deux couches suivantes
$2C \times (9 \times C + 1) + 2C \times (9 \times 2 \times C + 1) = 16 \times 73 + 16 \times 145 = 3488$
$+$
pour les 2 dernières couches
$4C \times (9 \times 2C + 1) + 4C \times (9 \times 4 \times C + 1) = 32 \times 145 + 32 \times 289 = 13888$
$+$
Pour la couche totalement connectée
$N \times 10 = 15680$

Soit un total de 33720 paramètres

dont 15680 sur la couche totalement connectée en sortie !!!!

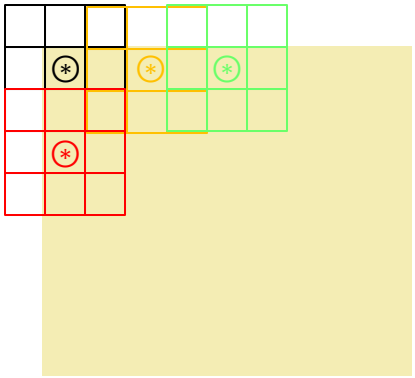
4. Réseaux de neurones convolutifs

Convolution: paramètres spatiaux

Déplacement (stride)

horizontal et vertical

exemple (2×2)

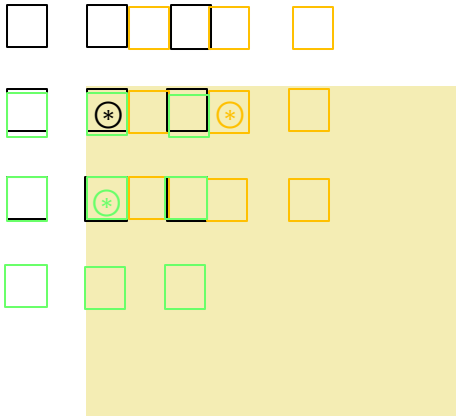


Dilatation (dilation)

sous-échantillonne l'entrée

exemple (2×2) avec stride de 2

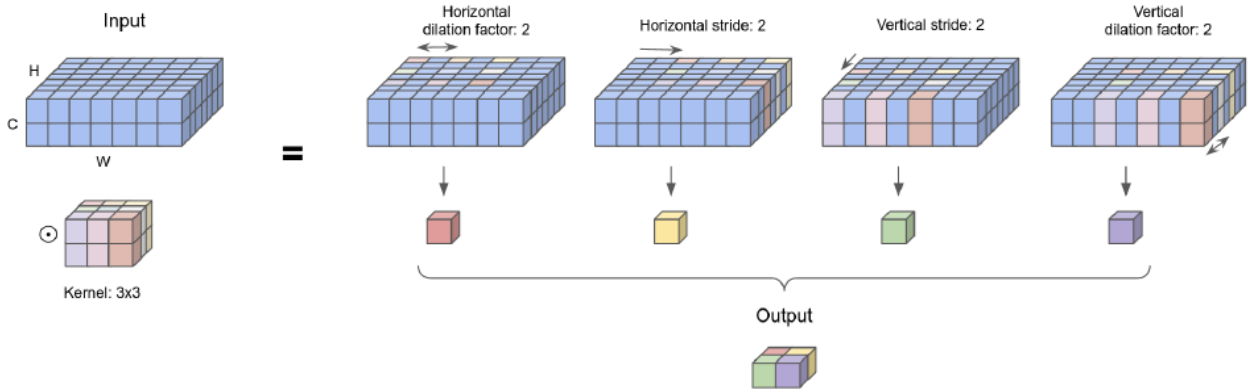
moins local avec le même nombre de paramètres



4. Réseaux de neurones convolutifs

Convolution paramètres spatiaux

Exemple



$$\boldsymbol{Y} = \boldsymbol{X} \odot \boldsymbol{k},$$

$$\boldsymbol{Y}_{i,j} = \sum_{h=1}^{k_\mathrm{h}} \sum_{w=1}^{k_\mathrm{w}} \sum_{c=1}^{C} \boldsymbol{X}_{s_\mathrm{h}*(j-1)+d_\mathrm{h}*(h-1)+1, s_\mathrm{w}*(i-1)+d_\mathrm{w}*(w-1)+1, c} \boldsymbol{k}_{h,w,c}$$

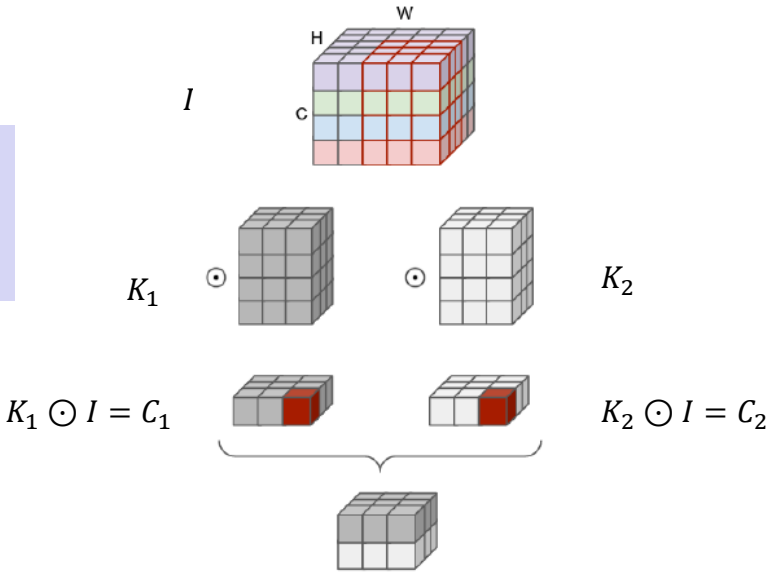
4. Réseaux de neurones convolutifs

Convolution séparable

Rappel: Convolution standard

nombre de paramètres
$N = C \times w_h \times w_v \times nk$

nk le nombre de noyaux (filtres)



pour C=4 canaux et 2 noyaux
$(3 \times 3 \times 4 + 1) \times 2 =$ 74 paramètres

pour $C = 512$ et 512 noyaux
2.359.808 paramètres

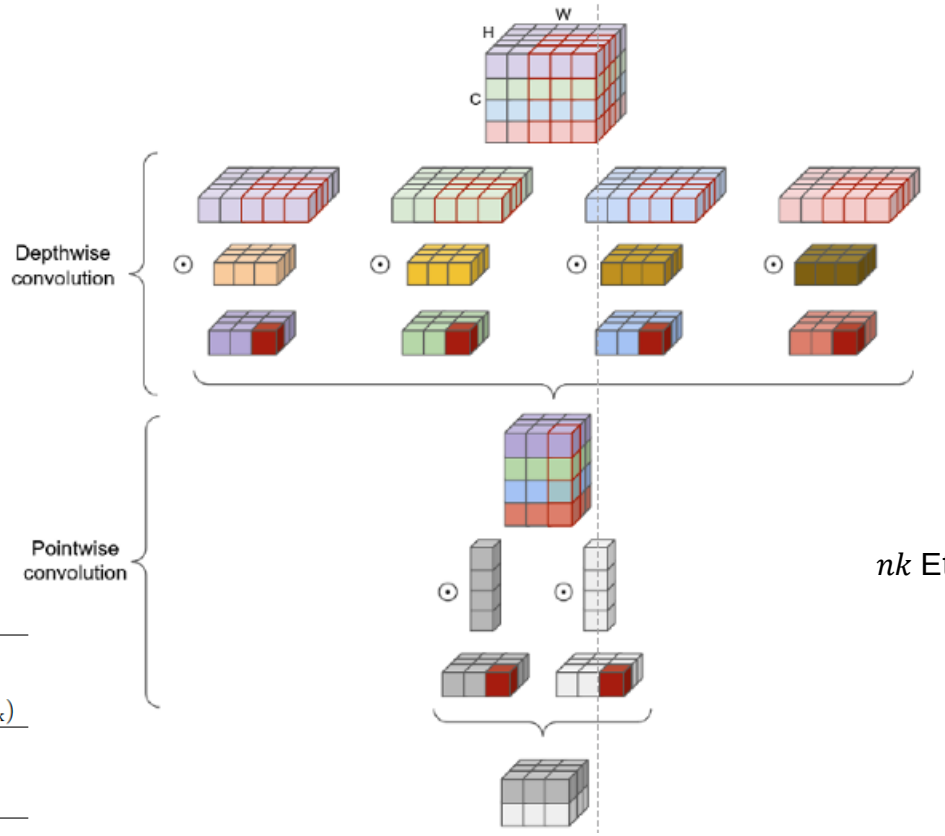
4. Réseaux de neurones convolutifs

Convolution séparable

Réduire le nombre de paramètres en décomposant un noyau (3D) en deux étapes

nombre de paramètres

$$N = C \times (w_h \times w_v) + C \times nk$$



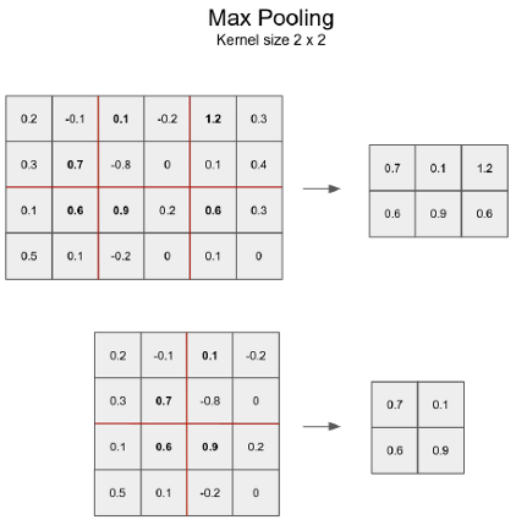
Comparaison Conv / Conv séparable

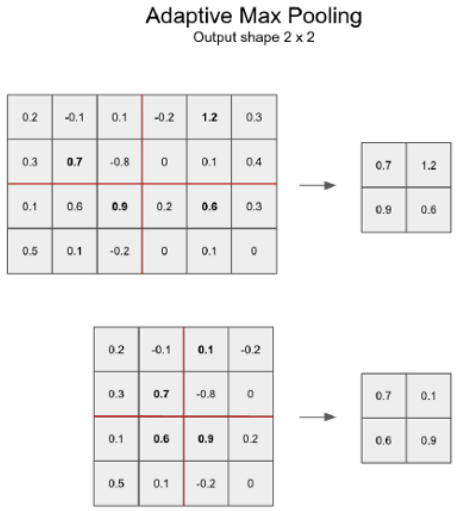
C	n_k	k_H	k_W	# weights convolution $C \times k_H \times k_W \times n_k$	# weights DSC $C \times (k_H \times k_W + n_k)$
512	512	3	3	2.4M	0.3M
1028	1028	3	3	9.5M	1.1M
1028	1028	5	5	26.4M	1.1M

4. Réseaux de neurones convolutifs

Les couches de Pooling (regroupement)

- Réduire la taille des cartes, donc le temps de calcul
- Aucun paramètre
- Quelle stratégie de sous-échantillonnage : à pas fixe, à taille fixe ?
- Quelle information : le maximum, la moyenne ?

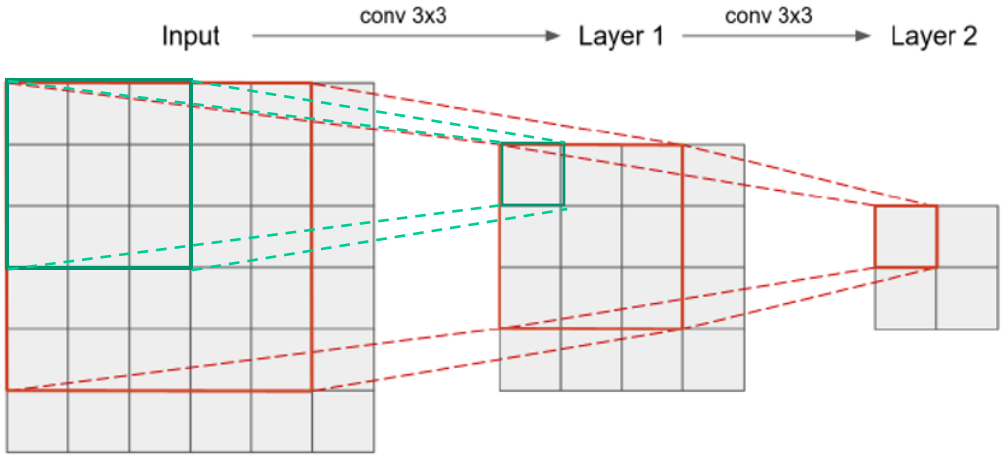




… Idem pour la moyenne

4. Réseaux de neurones convolutifs

Champ réceptif d'un réseau convolutif

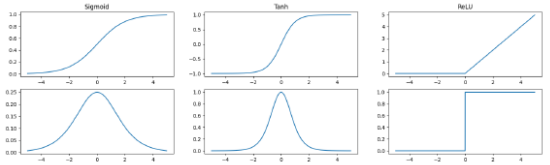


- C'est la taille de la fenêtre de pixels qui contribuent à un point d'une carte de la dernière couche
- Tenir compte du pooling, du stride, et de la taille du noyau sur chaque couche

Exemple: 2 couches conv 3x3 avec stride de 1 => champ réceptif de taille 5x5 pixels

4. Réseaux de neurones convolutifs

Les couches de normalisation



➢ En statistique :

centrer ($m = 0$) et réduire ($\sigma^2 = 1$) les données = standardiser

permet de comparer les distributions des données indépendamment de leur plage de variation

$$\hat{x}_i = \frac{x_i - \mu_X}{\sqrt{\sigma_X^2 + \varepsilon}} \quad \text{avec} \quad \mu_X = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2$$

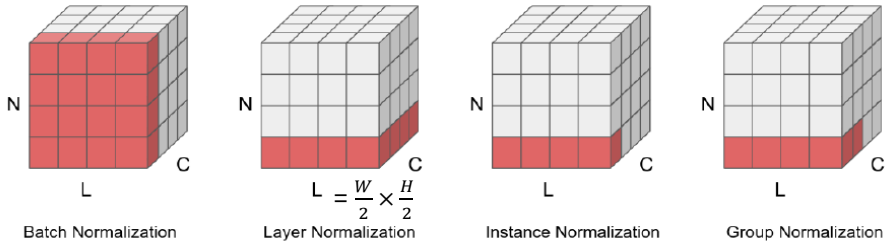
➢ En deep learning :

- ➢ la mise à jour synchrone des poids sur toutes les couches en même temps (pour une même itération) peut avoir des effets d'underflow ou d'overflow à l'itération suivante
- ➢ On s'assure que les variables des couches de sortie (les canaux de sortie) évoluent dans la bonne plage des fonctions d'activation.
- ✓ On évite les zones de gradient nul, on apprend plus vite.
- ✓ Les différents canaux ont des plages de variation qui deviennent comparables, on peut les « mélanger », et apprendre avec un même learning rate.

on calcule $y_i = \gamma \hat{x}_i + \beta$: on normalise chaque caractéristique indépendamment des autres, puis on fait une correction globale d'échelle et de biais, quels que soit les canaux, γ et β sont appris.

4. Réseaux de neurones convolutifs

Les couches de normalisation



Batch normalization

- pour chaque carte C_i, pour l'ensemble des N exemples du lot et des pixels $(\frac{W}{2} \times \frac{H}{2})$ de la cartes C_i: on normalise ensemble les caractéristiques d'une carte pour tous les exemples du batch

Layer normalization

- pour chaque exemple N_j du lot, pour tous les éléments de chaque carte $(\frac{W}{2} \times \frac{H}{2})$ et pour toutes les carte C_i : on normalise ensemble toutes les caractéristiques

Instance normalization

- pour chaque exemple N_j du lot et pour chaque carte C_i , pour tous les éléments de la carte $(\frac{W}{2} \times \frac{H}{2})$: on normalise les caractéristiques d'une carte

Group normalization

- pour chaque exemple N_j du lot , pour tous les éléments de la carte $(\frac{W}{2} \times \frac{H}{2})$ et pour un groupe de cartes C_i: on normalise ensemble les caractéristiques de certaines cartes

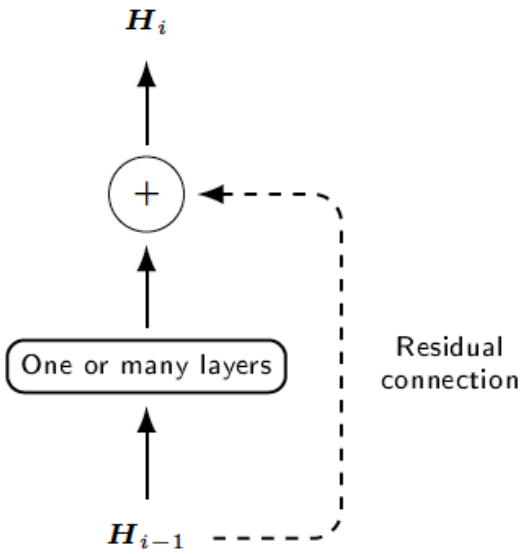
4. Réseaux de neurones convolutifs

Les connections résiduelles (skip connections)

Introduit pour la première fois dans le réseau ResNet

- Une couche peut voir plus d'une couche précédente (multi-résolution)
- La rétropropagation du gradient est plus efficace car elle se propage rapidement dans les couches les plus profondes

Somme
Multiplication
Concaténation

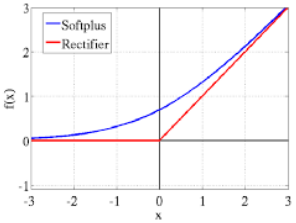


4. Réseaux de neurones convolutifs

Fonctions d'activation

Relu: éviter la saturation de l'activation softmax

$$ReLU(x) = \max(0, x)$$



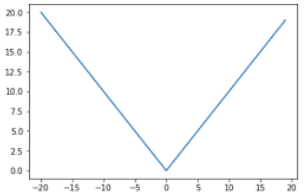
Softplus:

$$Softplus(x) = \frac{1}{\beta} \times log\big(1 + exp(\beta \times x)\big)$$

Extention Relu: apprendre dans la partie négative

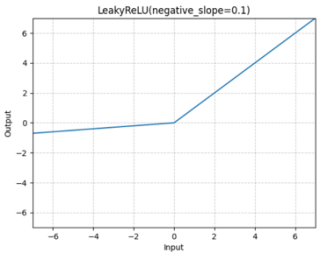
$$AbsRectification(x) = \max(0, x) - \min(0, x)$$

pour la vision



$$LeakyReLU(x) = \max(0, x) - 0.01 \min(0, x)$$

dans les réseaux adverses



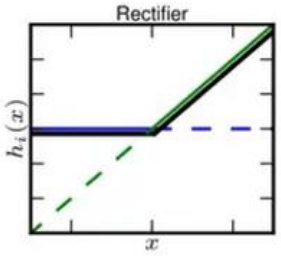
4. Réseaux de neurones convolutifs

Fonctions d'activation

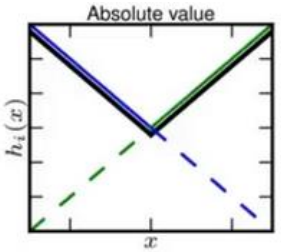
maxout: apprendre la fonction d'activation comme une fonction linéaire par morceaux

Exemples:

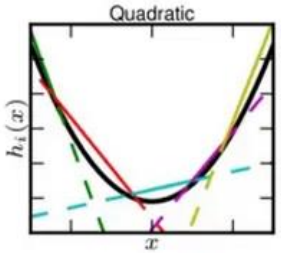
$ReLU(x) = \max(0, x)$



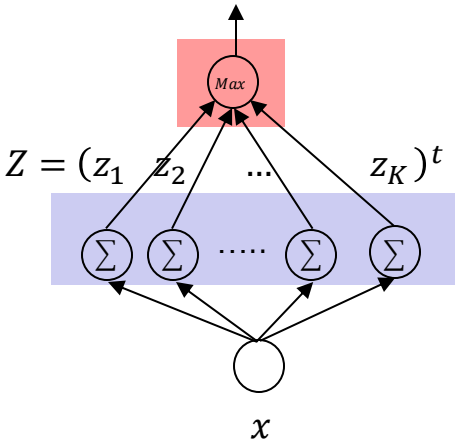
$abs(x) = \max(x, -x)$



$abs(x) = \max(z_1, z_2, z_3, z_4, z_5)$



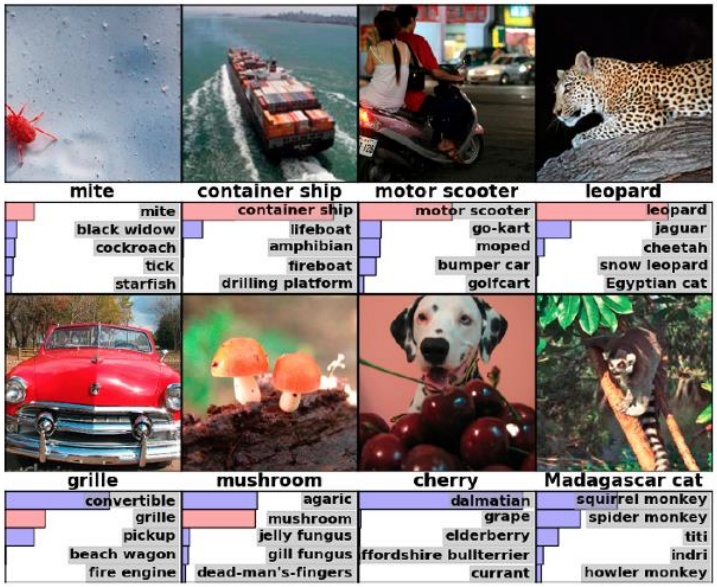
$h(X) = \max(z_1 \quad z_2 \ldots z_K)$



4. Réseaux de neurones convolutifs

Des CNN pour quelle tâche ?

- **Classification d'images**



- Détection d'objets

- Segmentation sémantique

4. Réseaux de neurones convolutifs

Des CNN pour quelle tâche ?

- Classification d'images
- **Détection d'objets**
- Segmentation sémantique

4. Réseaux de neurones convolutifs

Des CNN pour quelle tâche ?

- Classification d'images

- Détection d'objets

- **Segmentation sémantique**

4. Réseaux de neurones convolutifs

Dataset pour la classification d'images

IMAGENET

EntleBucher

Appenzeller

1.5 M d'images couleur (512×512)

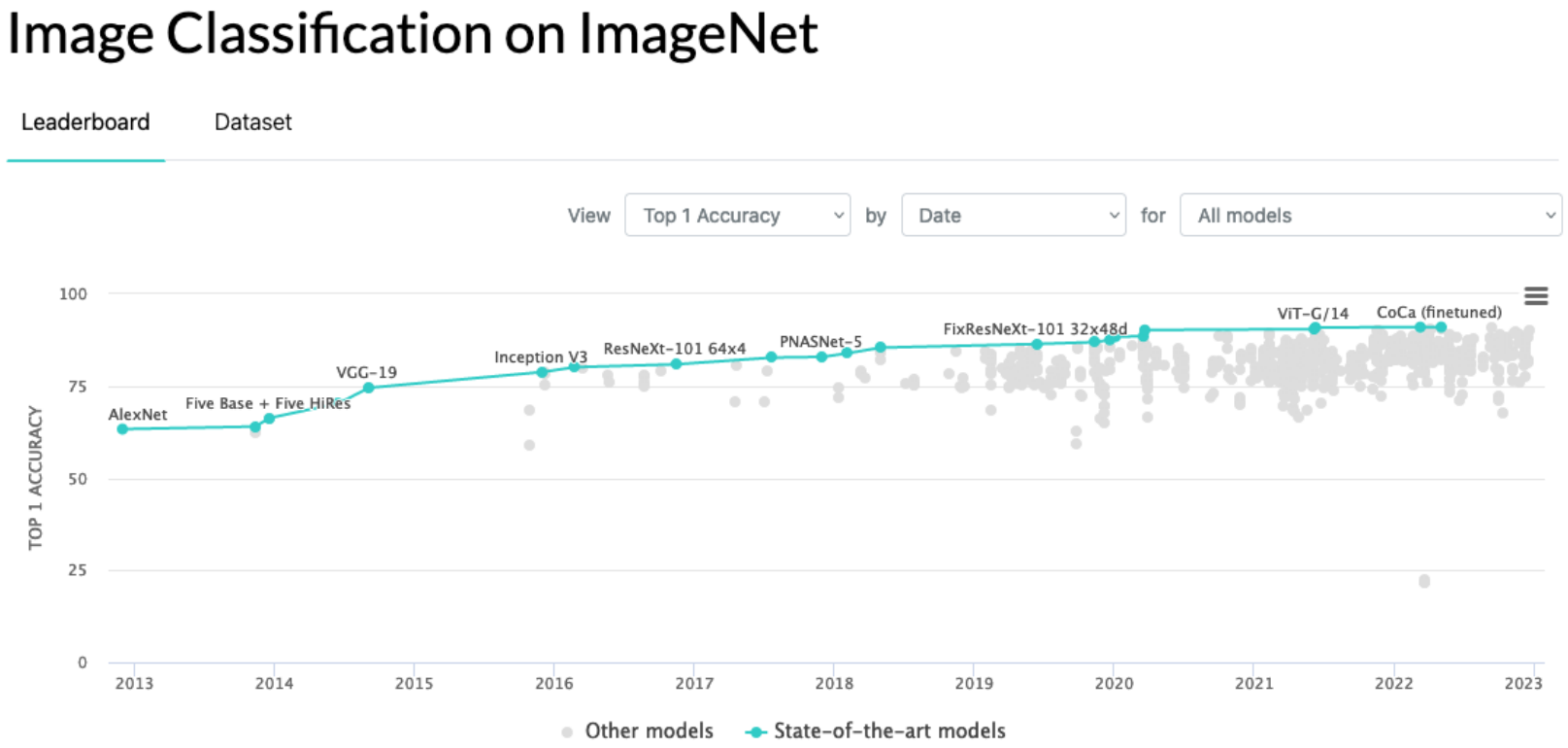
- ✓ **annotées: (catégorie,** (x, y, l, h)**)**
- ✓ **Utilise la structure WordNet de catégorisation d'objets + 120 catégories de races de chiens**
- ✓ **1000 catégories d'objets, plus de 1000 exemples par catégorie**
- ✓ **plusieurs objets par image** $(\text{catégorie}, (x, y, l, h))$.

www.image-net.org/challenges/LSVRC

Compétition : Large Scale Visual Recognition Challenge (ILSVRC)

de 2012 à 2017

www.image-net.org/challenges/LSVRC

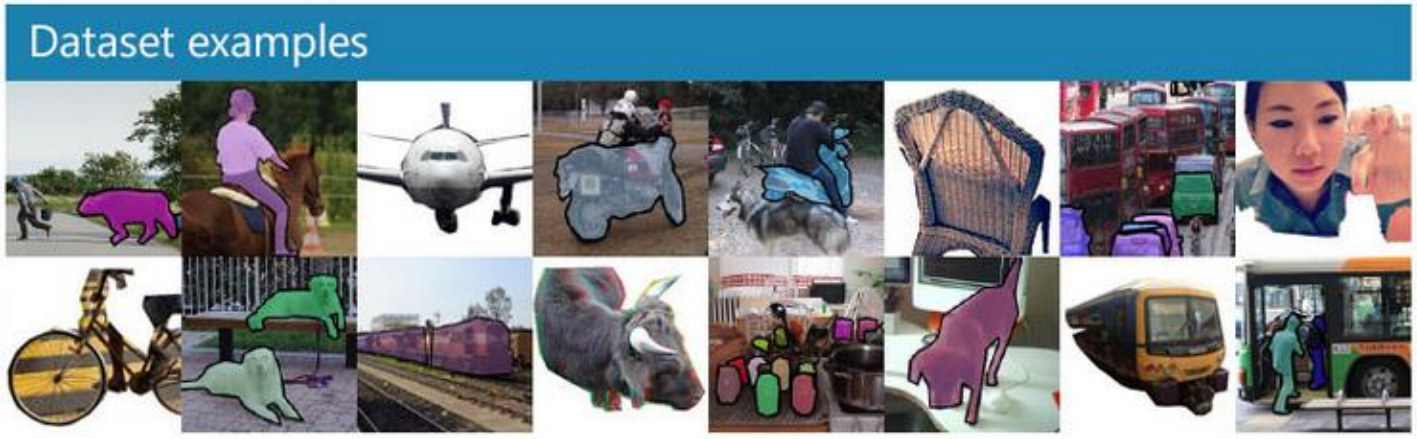


4. Réseaux de neurones convolutifs

Microsoft COCO: Common Objects in Context : https://cocodataset.org

- ✓ Object segmentation
- ✓ Recognition in context
- ✓ Superpixel stuff segmentation
- ✓ 330K images (>200K labeled)
- ✓ 1.5 million object instances

- ✓ 80 object categories
- ✓ 91 stuff categories
- ✓ 5 captions per image
- ✓ 250,000 people with keypoints



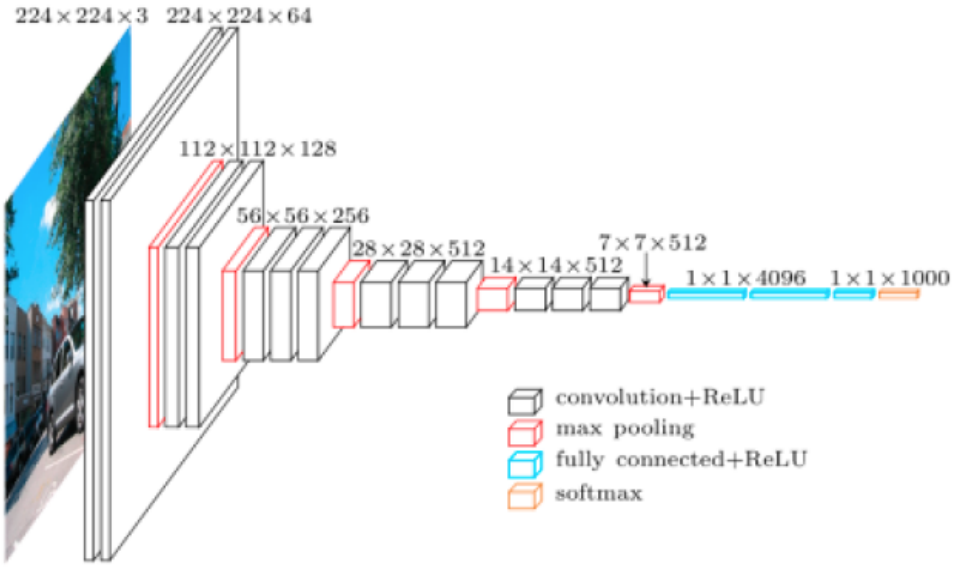
4. Réseaux de neurones convolutifs

Architecture VGG16 pour la classification d'images

16 couches:

13 conv + pooling + 3 denses

138 M de paramètres dont 124 M pour les 3 couches denses

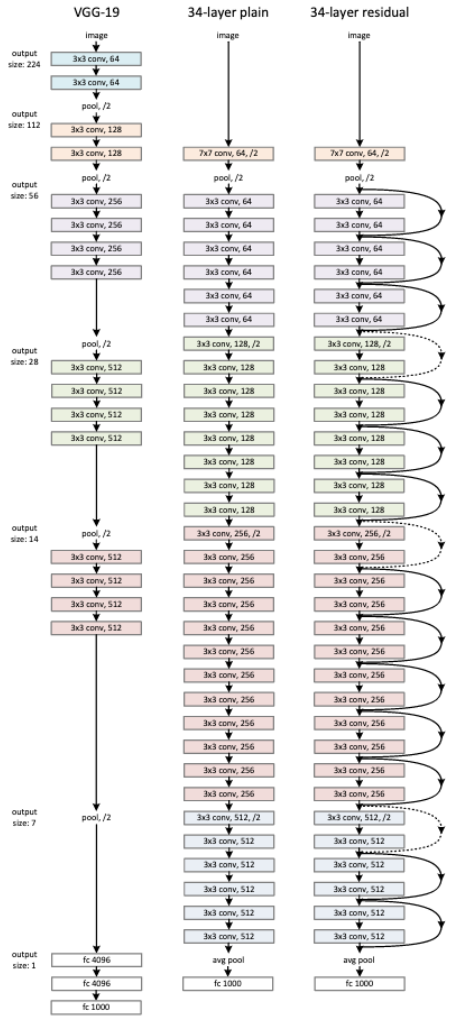


Performance : 92.7% sur ImageNet

4. Réseaux de neurones convolutifs

ResNet pour la classification d'images

L'introduction des connexions résiduelles pour des réseaux extrêmement profonds



Performance :

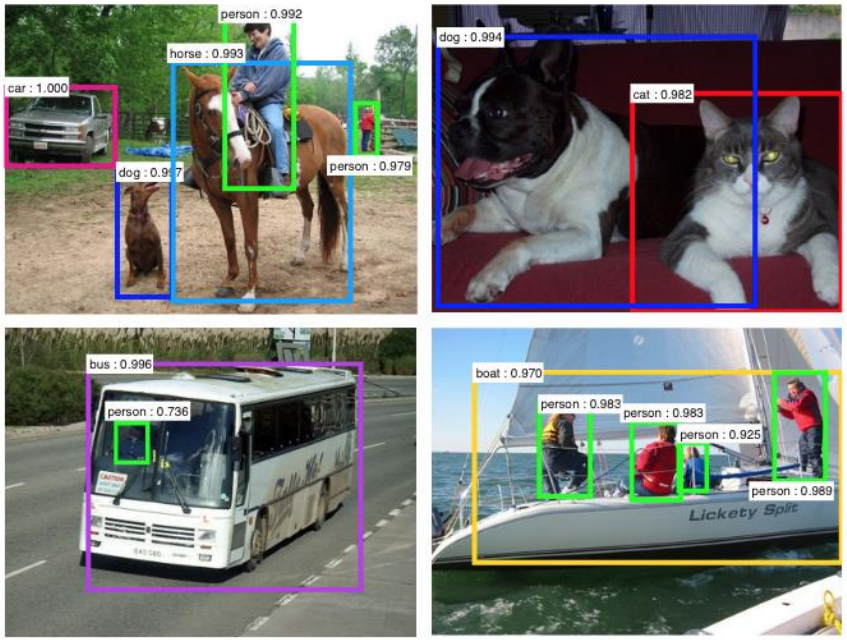
method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

4. Réseaux de neurones convolutifs

Faster-RCNN pour la détection d'objets

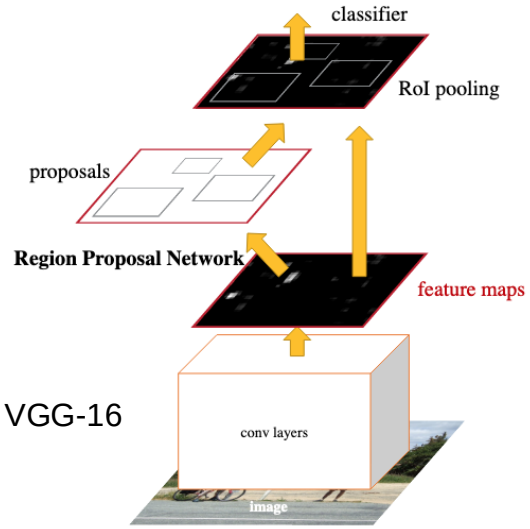
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

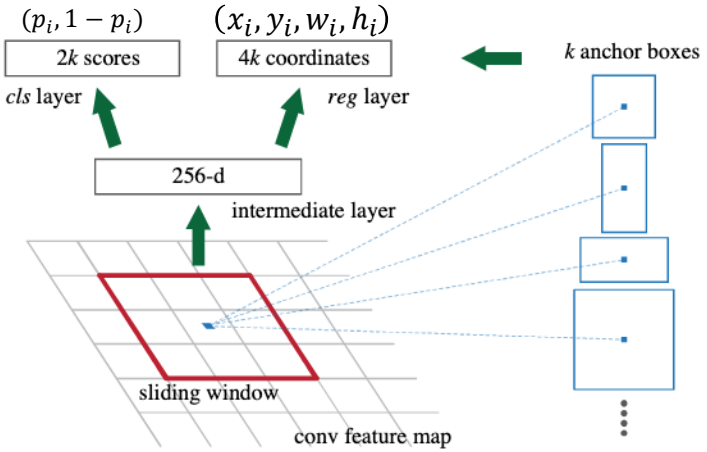


4. Réseaux de neurones convolutifs

Faster-RCNN pour la détection d'objets

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks





fenêtre glissante 3x3 en sortie de VGG
soit un champs réceptif de 228x228

p_i : probabilité d'avoir un objet en ce point

$t_i = (x_i, y_i, w_i, h_i)$: coordonnées de chaque fenêtre

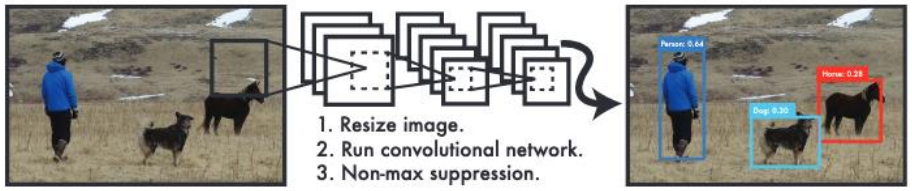
k fenêtres d'aspect prédéfini
3 échelles et 3 aspect
k=9

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

4. Réseaux de neurones convolutifs

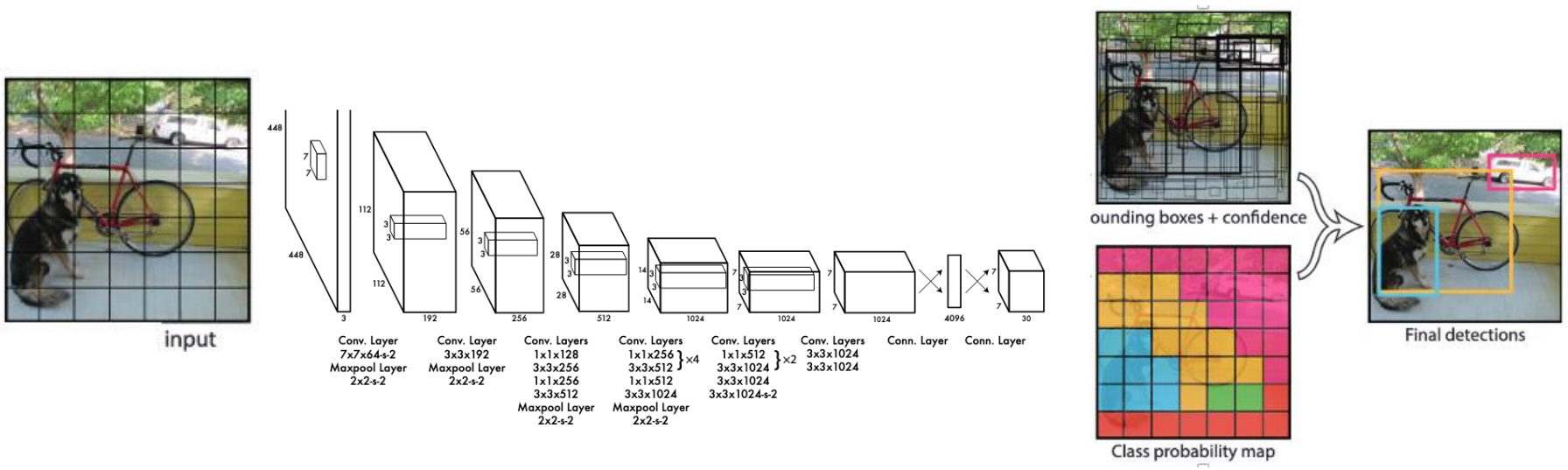
YOLO You Only Look Once pour la détection d'objets

Unified, Real-Time Object Detection



Une seule architecture convolutive très rapide prédit toute l'information

Une version à 24 couches, une autre à 9 couches



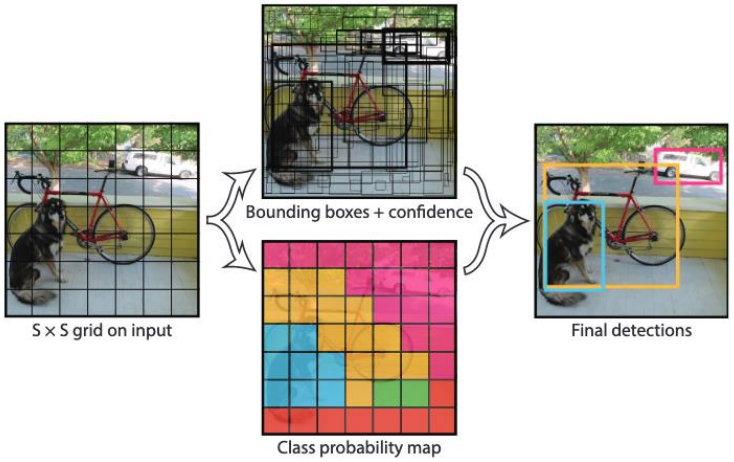
4. Réseaux de neurones convolutifs

YOLO You Only Look Once pour la détection d'objets

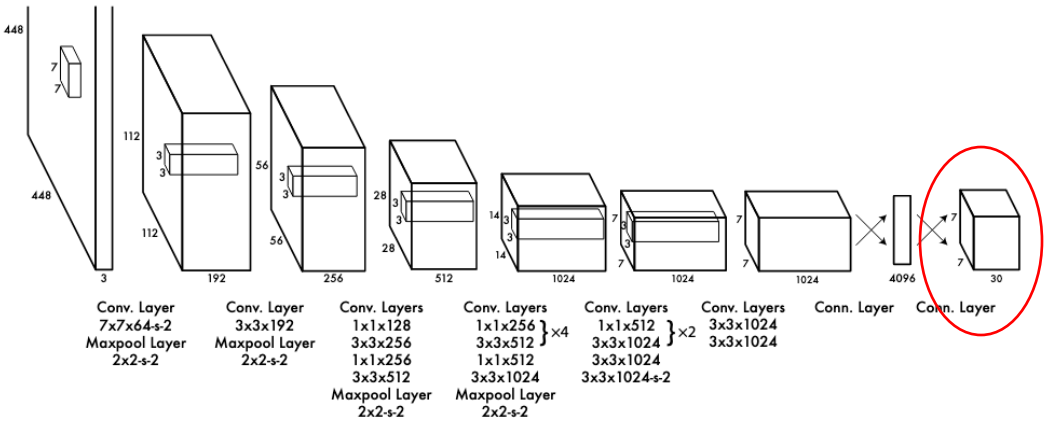
Unified, Real-Time Object Detection

Pour chaque cellule de la grille le système prédit

- B boites et leurs confiances $\left((x_i, y_i, w_i, h_i), p_i\right)_{i=1,\dots,B}$
  avec $p_i = P(objet) \times IoU(boite_{prédite}, boite_{verité})$
- Des probabilités de K classe d'objets $P(c_k|objet)_{k=1,\dots,20}$



En pratique il y a 7 x 7 cellules et B=2 boites et K=20 classes soit 7x7 x (10 + 20) prédictions



Il est possible qu'un même objet grand recouvrant plusieurs cellules soit prédit plusieurs fois: on garde la meilleure prédiction en appliquant une opération nommée non maximal suppression

YOLO You Only Look Once pour la détection d'objets

Unified, Real-Time Object Detection

la loss YOLO :

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

- S x S cellules
- B boites et leurs confiances $\left((x_i, y_i, w_i, h_i), p_i\right)_{i=1,\dots,B}$
  avec $p_i = P(objet) \times IoU(boite_{prédite}, boite_{verité})$
- probabilités de K classes d'objets $P(c_k|objet)_{k=1,\dots,20}$

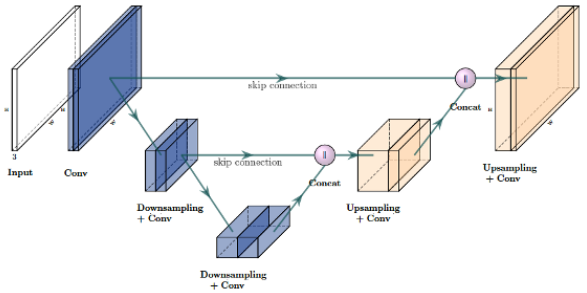
4. Réseaux de neurones convolutifs

Segmentation sémantique: l'architecture FCN et U-net

Ronneberger, O., P. Fischer et T. Brox (oct. 2015). « U-Net : Convolutional Networks for Biomedical Image Segmentation ». In : 18th Medical Image Computing and Computer-Assisted Intervention (MICCAI), p. 234-241.

- réaliser une prédiction au niveau pixel sur des images de taille variable, très utilisé en imagerie médicale

FCN : Fully convolutionnal network sans couche dense, entrée image , sortie image de même taille
U-net : un FCN avec une architecture encodeur décodeur en U



✓ l'encoder est un réseau convolutif avec pooling qui produit une représentation compressée de l'image d'entrée

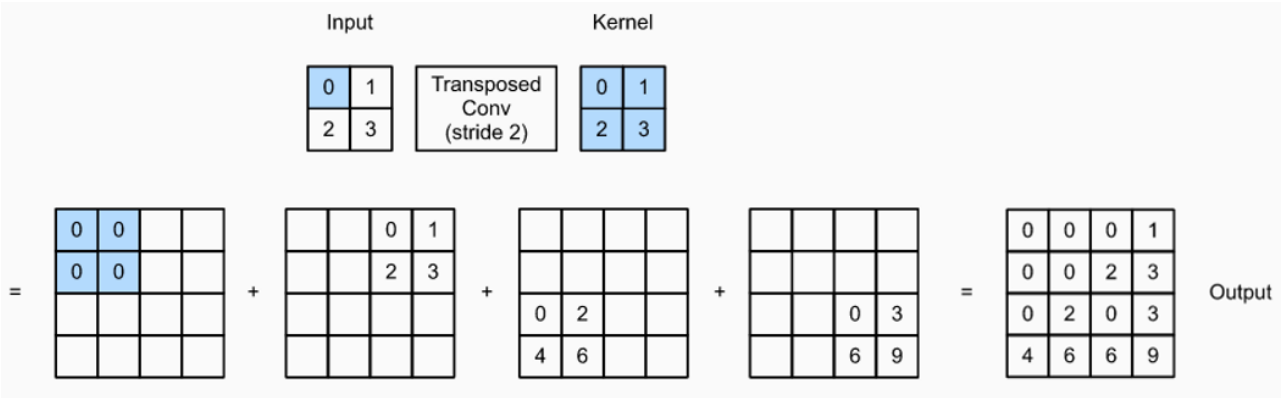
✓ le décodeur est chargé de construire l'image de segmentation à partir de la représentation compressée. Il met en œuvre une fonction de sur-échantillonnage (Transpose convolution), et des connexions résiduelles pour exploiter les détails (haute résolution)

4. Réseaux de neurones convolutifs

Segmentation sémantique: l'architecture FCN et U-net

Ronneberger, O., P. Fischer et T. Brox (oct. 2015). « U-Net : Convolutional Networks for Biomedical Image Segmentation ». In : 18th Medical Image Computing and Computer-Assisted Intervention (MICCAI), p. 234-241.

Convolution transposée: réaliser l'opération inverse de conv + pooling



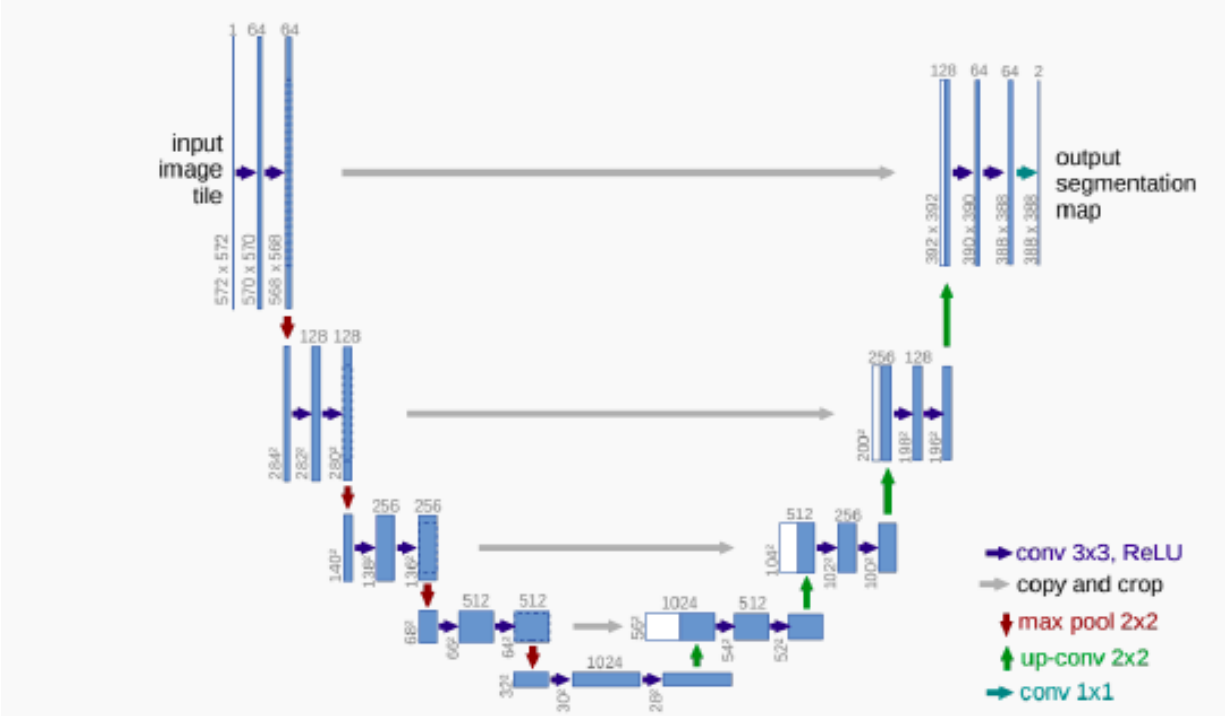
Transposée: l'image et le kernel jouent des rôles inversés

mais ce n'est pas une transposition au sens strict

4. Réseaux de neurones convolutifs

Segmentation sémantique: l'architecture FCN et U-net

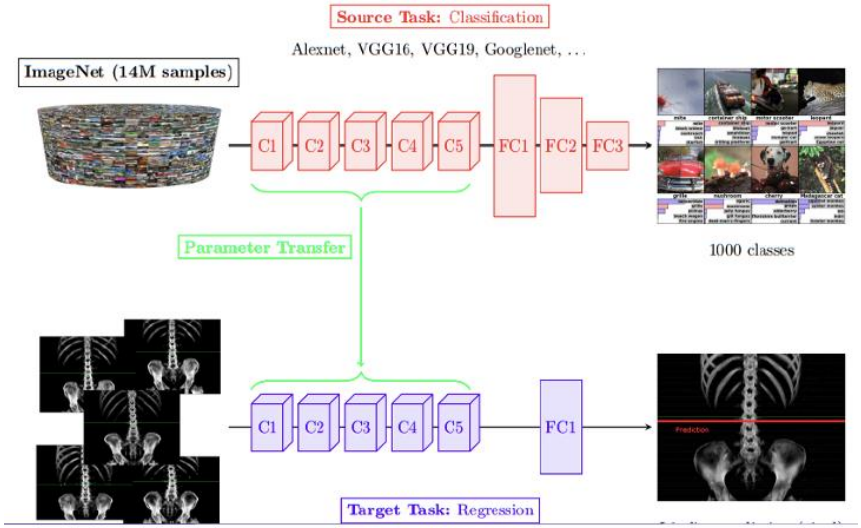
Ronneberger, O., P. Fischer et T. Brox (oct. 2015). « U-Net : Convolutional Networks for Biomedical Image Segmentation ». In : 18th Medical Image Computing and Computer-Assisted Intervention (MICCAI), p. 234-241.



Transfer Learning

Comment faire quand on a peu de données?

- utiliser un réseaux pré-entrainé (AlexNet, VGG16, ResNet, Yolo etc…)
- n'apprendre que les couches de sortie sur le problème considéré
- Fine tuning de couches de sortie



4. Réseaux de neurones convolutifs

Modèle de Classification disponibles avec Pytorch

AlexNet	GoogLeNet	MobileNet V3	SqueezeNet
ConvNeXt	Inception V3	RegNet	SwinTransformer
DenseNet	MaxVit	ResNet	VGG
EfficientNet	MNASNet	ResNeXt	VisionTransformer
EfficientNetV2	MobileNet V2	ShuffleNet V2	Wide ResNet

https://pytorch.org/vision/stable/models.html