

Réseaux de neurones

# IFT 780

Segmentation et localisation

Par

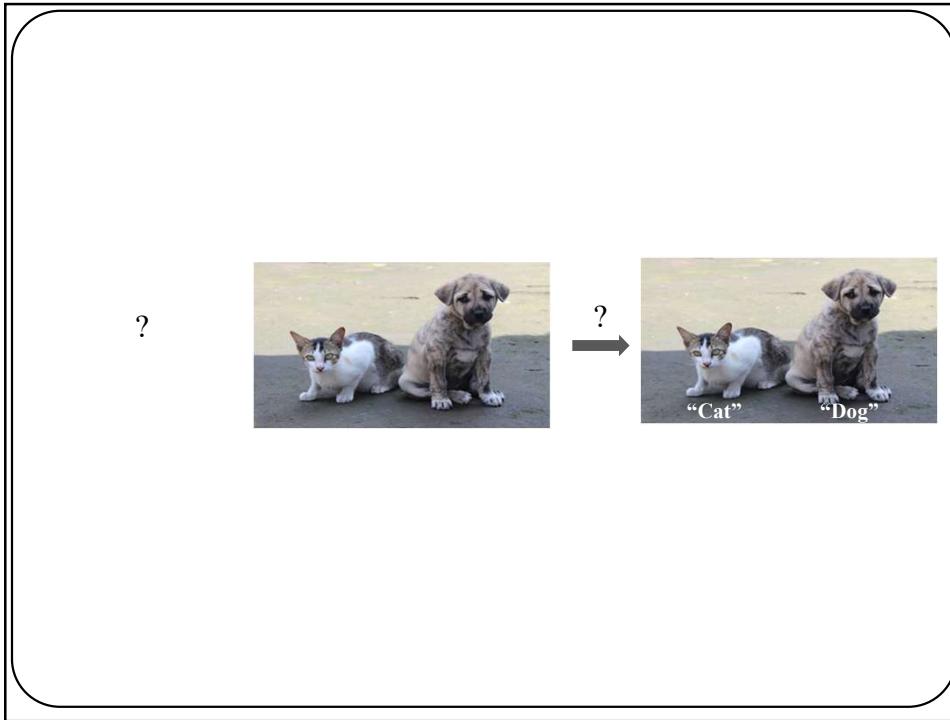
Pierre-Marc Jodoin, Antoine Théberge

1

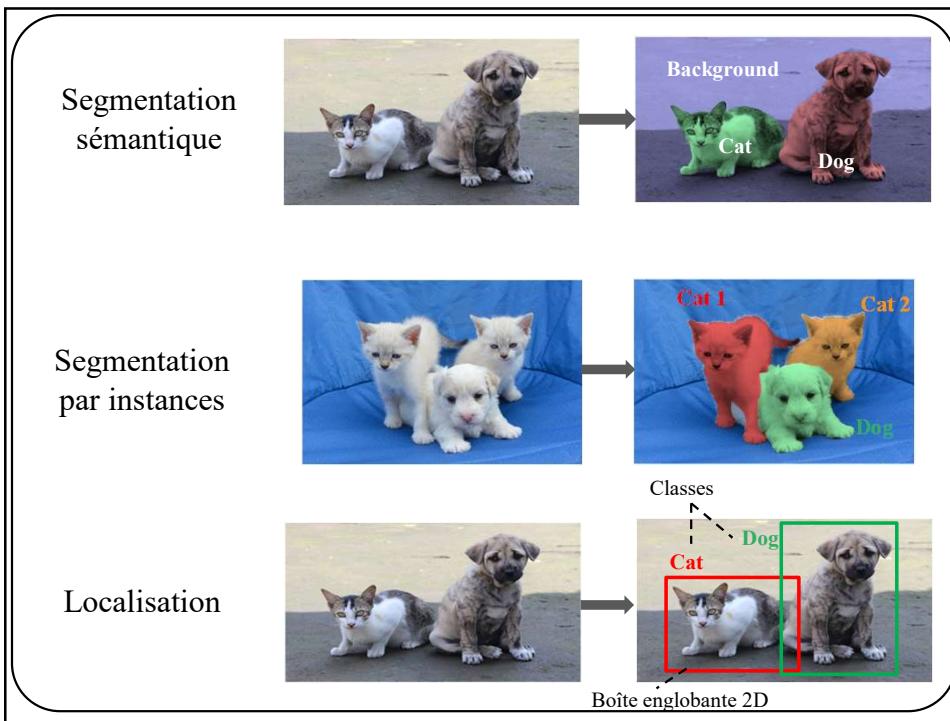
Classification



2

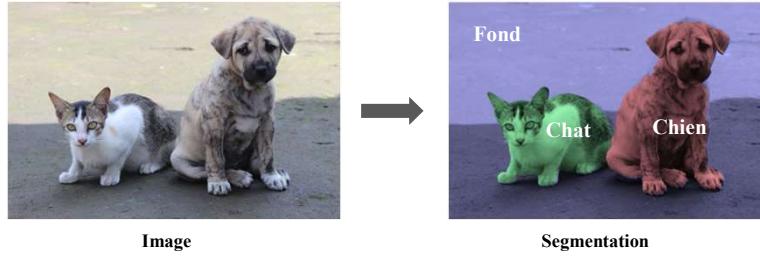


3



4

# Segmentation sémantique



**But:** Assigner la bonne étiquette de classe à **chaque pixel de l'image d'entrée**

Peut être vu comme un problème de classification **dense** et **structurée**

Possiblement des millions de pixels

Prédictions structure spatiale

5

# Segmentation sémantique

**Application :** identification+localisation des objets sur la route.



M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. [Bibtex]

6

# Segmentation sémantique

**Application :** identification+localisation des structures vues par satellite (« remote sensing »).



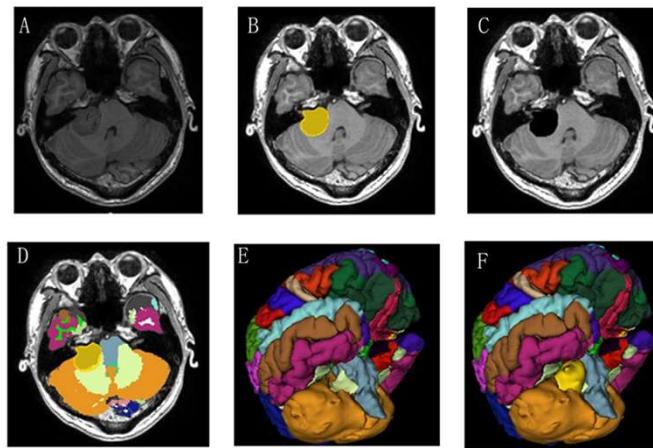
*Fig. 7: Left: Original satellite image. Right: Semantic segmentation of roads, buildings and vegetation.*

Ng, V., & Hofmann, D. (2018, July). Scalable feature extraction with aerial and satellite imagery. In Proceedings of the 17th Python in Science Conference (SCIPY 2018), Austin, TX, USA (pp. 9-15).

7

# Segmentation sémantique

**Application :** imagerie médicale, identification+localisation des tumeurs et régions du cerveau



Hou, X., Yang, D., Li, D., Liu, M., Zhou, Y., & Shi, M. (2020). A new simple brain segmentation method for extracerebral intracranial tumors. *PLoS one*, 15(4), e0230754.

8

# Segmentation sémantique

Comment mesurer la performance de la segmentation ?



Cible – Vérité terrain



Prédiction

9

# Segmentation sémantique

Comment mesurer la performance de la segmentation ?

Matrice de confusion:

		Vérité terrain	
		Positif	Négatif
Prédiction	Positif	<i>True positive (TP)</i>	<i>False Positive (FP)</i>
	Négatif	<i>False negative (FN)</i>	<i>True negative (TN)</i>



10

# Segmentation sémantique

Comment mesurer la performance de la segmentation ?

$$\text{Justesse} \quad \frac{TP + TN}{TP + TN + FP + FN}$$

(“Pixel accuracy”)

$$\text{Intersection over Union (IOU)/ Jaccard Index} \quad \frac{TP}{TP + FP + FN}$$

$$\text{Dice} \quad \frac{2TP}{2TP + FP + FN}$$

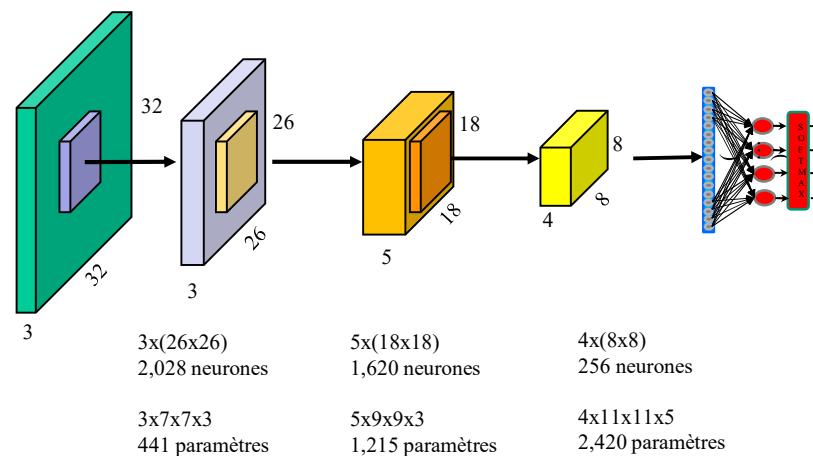


11

Image RGB : couche 1 : 3 filtres de taille 7x7  
couche 2 : 5 filtres de taille 9x9  
couche 3 : 4 filtres de taille 11x11  
convolution « valid »

Rappel  
classification  
d’images 32x32

Image: 32x32x3  
Stride : 1

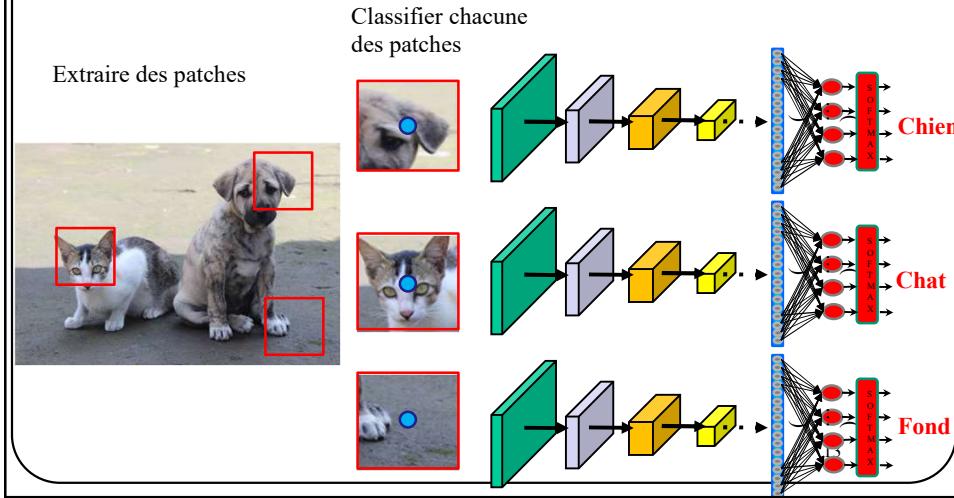


12

## Segmentation sémantique

Jusqu'à présent, on a vu comment classifier des images.

**Idée:** segmentation = classifier des sous-parties (*patches*) d'image

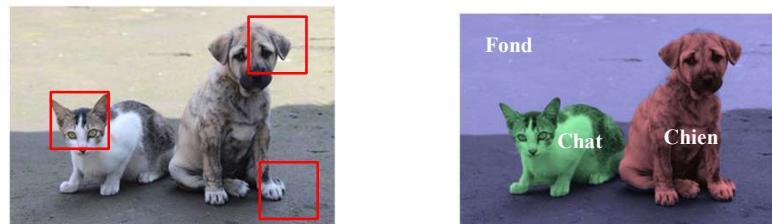


13

## Segmentation sémantique

Jusqu'à présent, on a vu comment classifier des images.

**Idée:** segmentation = classifier des sous-parties (*patches*) d'image



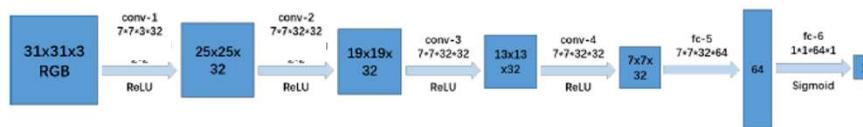
14

# Segmentation sémantique

Jusqu'à présent, on a vu comment classifier des images.

**Idée:** segmentation = classifier des sous-parties (*patches*) d'image

Exemple d'un réseau à convolution pour des patches RGB 31x31  
(Image tirée de l'article)



Wang Y, Luo Z., Jodoin P-M (2017) **Interactive Deep Learning Method for Segmenting Moving Objects Pattern Recognition Letters**, 96, p.66-75

15

## Plusieurs inconvénients

### 1. Très long tant en entraînement qu'en test

#### 1. Entraînement

Si 10,000 images 640x480 (300 000 pixels/image)  
= 3 milliards de patches!

1 epoch = 3 milliards de propagations avant  
et de rétro-propagations

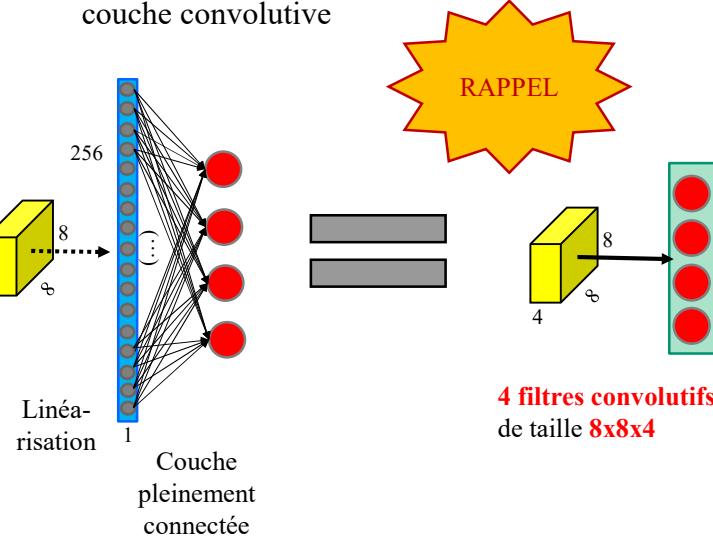
#### 2. Prédiction basée sur une **information locale (une patch)**

16

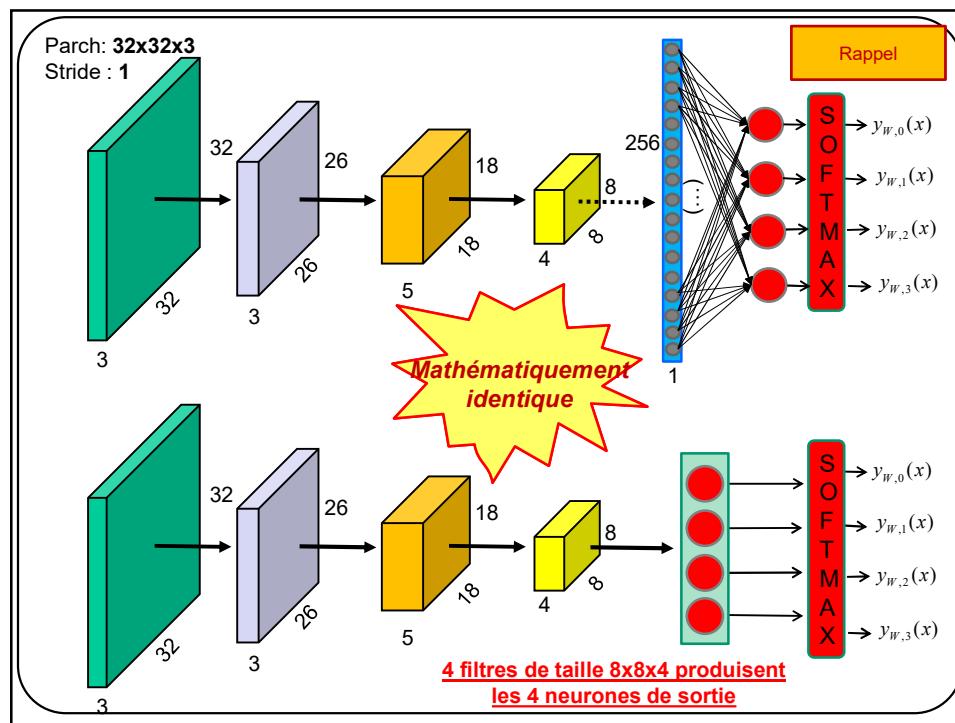
16

## Segmentation sémantique

**Amélioration 1:** remplacer la couche pleinement connectée par une couche convective



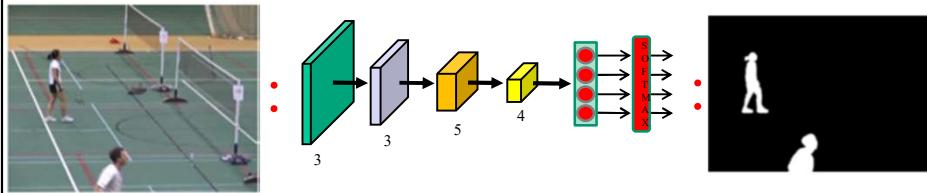
17



18

Image RGB : couche 1 : 3 filtres de taille 7x7  
couche 2 : 5 filtres de taille 9x9  
couche 3 : 4 filtres de taille 11x11  
convolution « valid »

Avec le réseau que voici, avec des conv « valid » et sans *pooling*, pour une image en entrée de 320x240, on aura en sortie 289x209 pixels, chacun ayant un vecteur de 4 prédictions.



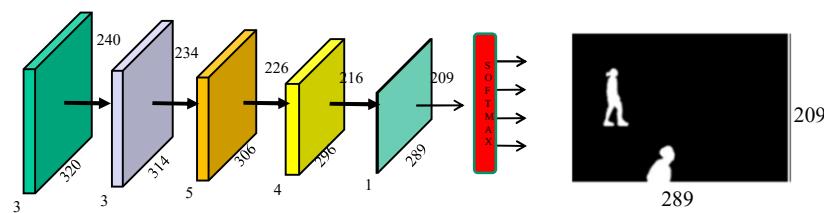
**Immense avantage** : fini les patches, on peut traiter une image avec 1 propagation avant et 1 rétropropagation

19

19

## Segmentation sémantique

Taille des cartes d'activation pour une image en entrée 320x240



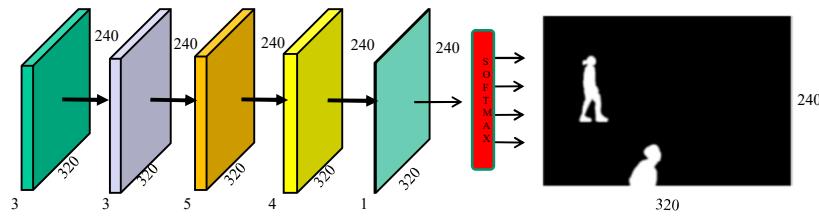
**Immense avantage** : fini les patches, on peut traiter une image avec 1 propagation avant et 1 rétropropagation

20

20

## Segmentation sémantique

Si on remplace les convolutions « valid » par des **convolutions « same »** (avec du *padding*) nous aurons en sortie une image de la même taille que l'image d'entrée



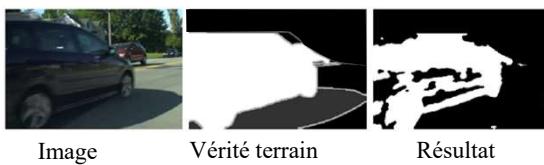
**Immense avantage**: fini les patches, on peut traiter une image avec 1 propagation avant et 1 rétropropagation

21

21

## Segmentation sémantique

Un réseau comme celui de la page précédente n'est jamais utilisé en pratique. Voici un exemple:



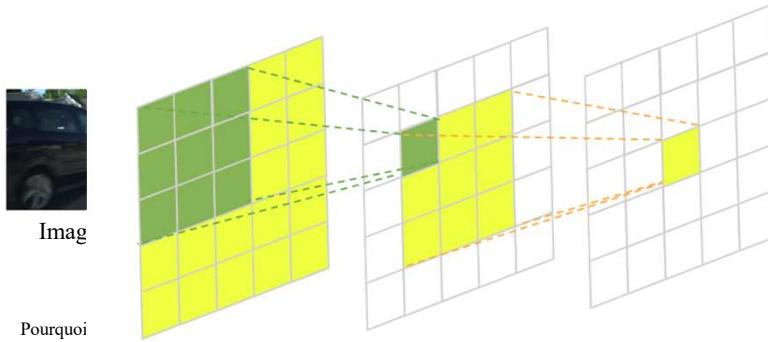
Pourquoi la prédiction est-elle bruitée ? Pourquoi autant de trous dans la prédiction ?

**Réponse : le “receptive field” est trop petit !**

Wang Y, Luo Z., Jodoin P-M (2017) Interactive Deep Learning Method for Segmenting Moving Objects Pattern Recognition Letters, 96, p.66-75

22

# Segmentation sémantique



**Réponse : le “receptive field” est trop petit !**

Wang Y, Luo Z., Jodoin P-M (2017) Interactive Deep Learning Method for Segmenting Moving Objects Pattern Recognition Letters, 96, p.66-75

23

## Note: taille du receptive field

$$r_0 = \sum_{l=1}^L \left( (k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$

*r* = receptive field

*k* = kernel

*s* = stride

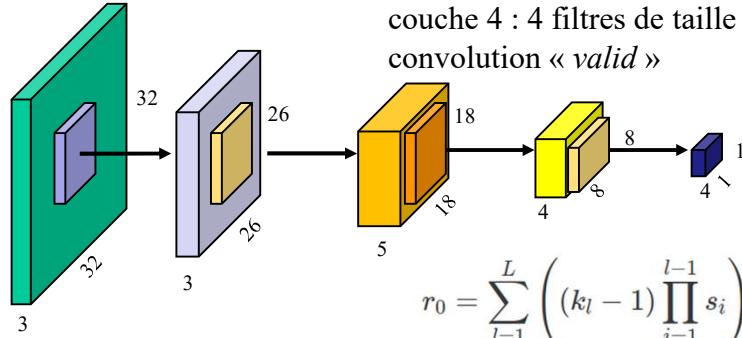
*l* = layers du réseau

<https://distill.pub/2019/computing-receptive-fields>

24

## Note: taille du receptive field

Image RGB : couche 1 : 3 filtres de taille 7x7  
couche 2 : 5 filtres de taille 9x9  
couche 3 : 4 filtres de taille 11x11  
couche 4 : 4 filtres de taille 8x8  
convolution « *valid* »



$$r_0 = \sum_{l=1}^L \left( (k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$

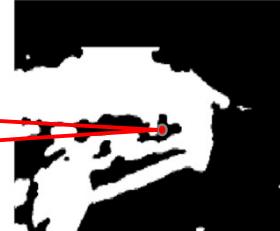
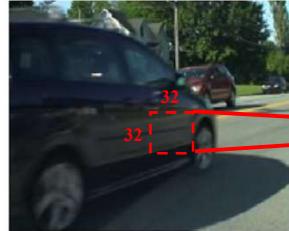
<https://distill.pub/2019/computing-receptive-fields>

$$\mathbf{6 + 8 + 10 + 7 + 1 = 32}$$

25

## Note: taille du *receptive field*

In  
S



Chaque prédiction en sortie se fait sur la base d'une portion de taille 32x32 de l'image d'entrée.  
En ce sens, le réseau n'a **qu'une vision locale et partielle de l'image.**

3

$$r_0 = \sum_{l=1}^L \left( (k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$

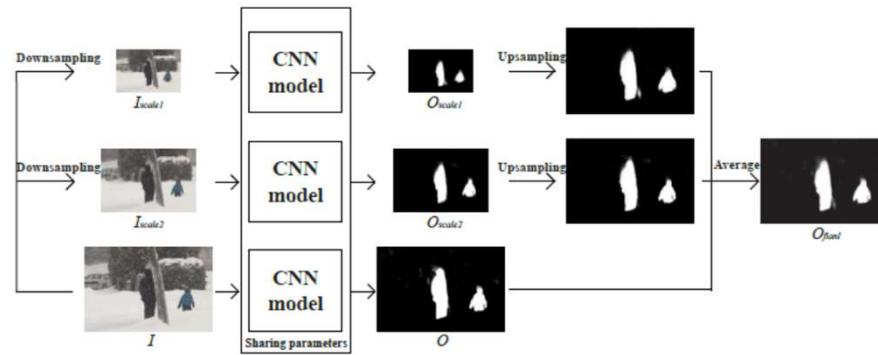
<https://distill.pub/2019/computing-receptive-fields>

$$\mathbf{6 + 8 + 10 + 7 + 1 = 32}$$

26

## Segmentation sémantique

**Amélioration 2:** pour avoir plus de contexte dans la prédiction, entraîner un CNN avec des **images multirésolution**. En test, **combiner les prédictions** (ensemble de modèles)

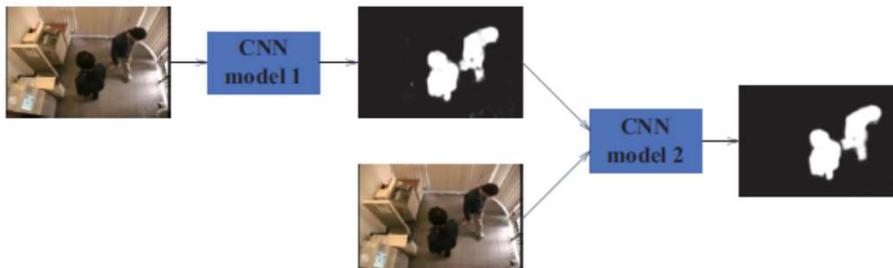


Wang Y, Luo Z., Jodoin P-M (2017) Interactive Deep Learning Method for Segmenting Moving Objects Pattern Recognition Letters, 96, p.66-75

27

## Segmentation sémantique

**Amélioration 3:** Pour raffiner les résultats, entraîner 2 modèles en cascade. Un premier qui segmente l'image d'entrée et le second qui segmente l'image d'entrée et la carte de segmentation du premier. Cela permet d'améliorer la cohésion spatiale.

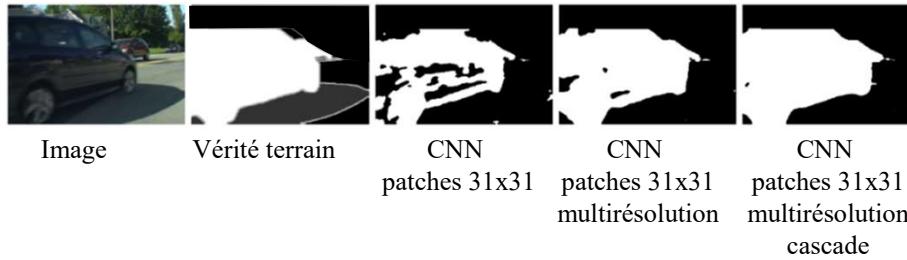


Wang Y, Luo Z., Jodoin P-M (2017) Interactive Deep Learning Method for Segmenting Moving Objects Pattern Recognition Letters, 96, p.66-75

28

# Segmentation sémantique

Receptive field

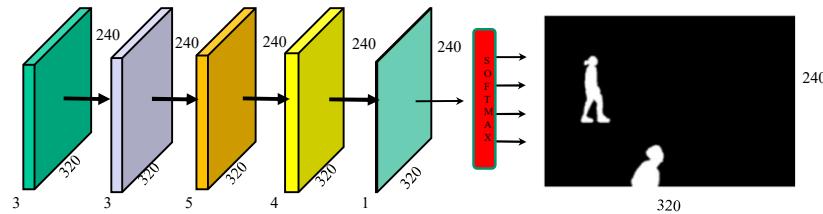


Wang Y, Luo Z., Jodoin P-M (2017) Interactive Deep Learning Method for Segmenting Moving Objects Pattern Recognition Letters, 96, p.66-75

29

# Segmentation sémantique

**Problème :** ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240)



**Solutions:**

- 1- ajouter beaucoup de couches
- 2- utiliser des convolutions dilatées (convolutions à trous)
- 3- mettre des couches de pooling après chaque bloc convolutionnel
- 4- faire un mélange de tout ça!

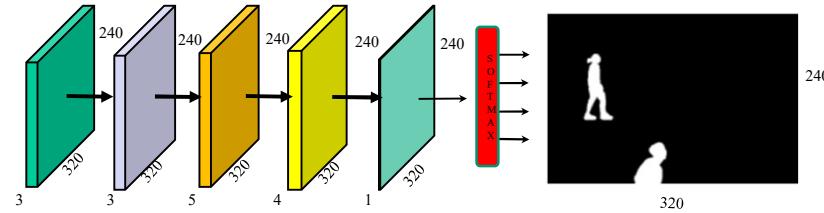
$$r_0 = \sum_{l=1}^L \left( (k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$

30

30

# Segmentation sémantique

**Problème** : ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240)



**Solutions:**

- 1- ajouter beaucoup de couches
- 2- utiliser des convolutions dilatées (convolutions *a trous*)
- 3- mettre des couches de pooling après chaque bloc convolutionnel
- 4- faire un mélange de tout ça!

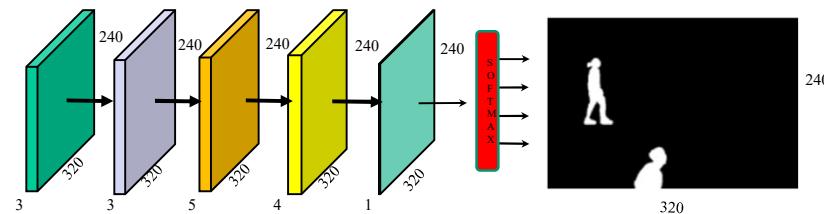
Avec des filtres 3x3  
Jusqu'à 120 couches!

31

31

# Segmentation sémantique

**Problème** : ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240)



**Solutions:**

- 1- ajouter beaucoup de couches
- 2- utiliser des convolutions dilatées (
- 3- mettre des couches de pooling après chaque bloc convolutionnel
- 4- faire un mélange de tout ça!

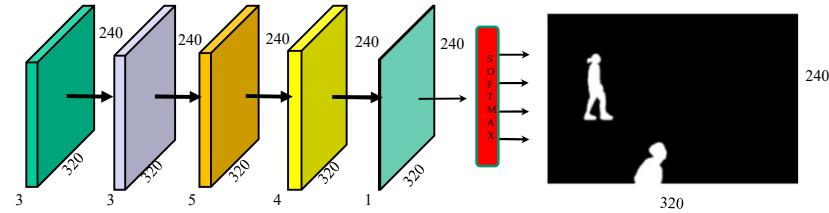
Explosion de la mémoire  
Et des temps de calculs  
Et problème de disparition  
du gradient

32

32

# Segmentation sémantique

**Problème** : ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240)



## Solutions:

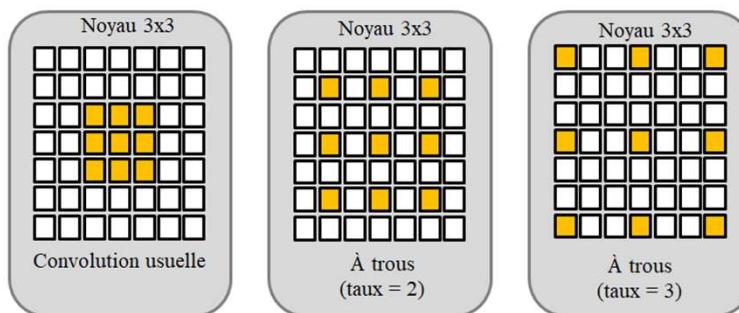
- 1- ajouter beaucoup de couches
- 2- utiliser des **convolutions dilatées** (convolutions à trous)
- 3- mettre des couches de pooling après chaque
- 4- faire un mélange de tout ça!

33

33

# Segmentation sémantique

Rappel



Champ récepteur=3x3

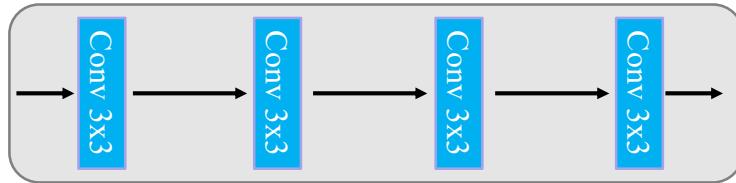
Champ récepteur=5x5

Champ récepteur=7x7

34

34

## Segmentation sémantique



Champ récepteur  
**(taux = 1)**

3x3

5x5

7x7

9x9

Champ récepteur  
**(taux = 2)**

5x5

9x9

13x13

17x17

Champ récepteur  
**(taux = 3)**

7x7

13x13

19x19

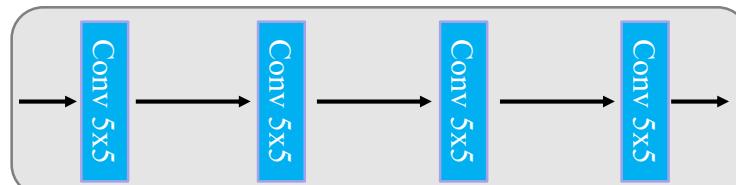
21x21

$$r_0 = \sum_{l=1}^L \left( (k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1 \quad k = \text{taux} * (k-1) + 1$$

35

35

## Segmentation sémantique



Champ récepteur  
**(taux = 1)**

5x5

9x9

13x13

17x17

Champ récepteur  
**(taux = 2)**

9x9

17x17

25x25

33x33

Champ récepteur  
**(taux = 3)**

15x15

29x29

43x43

57x57

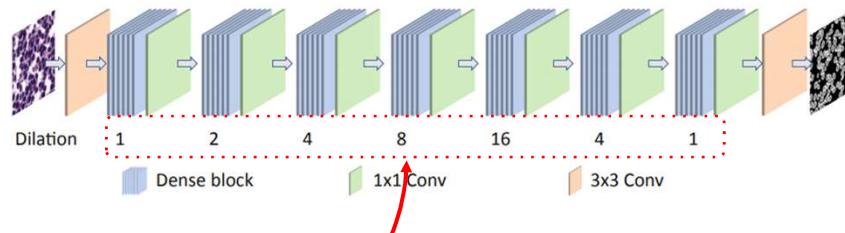
$$r_0 = \sum_{l=1}^L \left( (k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1 \quad k = \text{taux} * (k-1) + 1$$

36

36

# FullNet

Le « **FullNet** » [Qu et al. 2019] implémente ce type de réseau mais avec des blocs convolutifs **denses** comme ceux du **denseNet**.



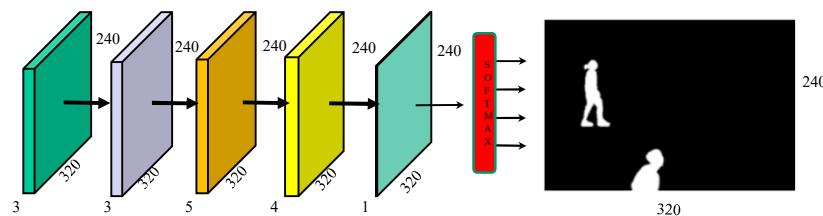
H. Qu, Z.Yan, G.M. Riedlinger, S.De and D.N. Metaxas  
“Improving Nuclei/Gland Instance Segmentation in Histopathology Images  
by Full Resolution Neural Network and Spatial Constrained Loss”, in proc of MICCAI 2019

37

37

# Segmentation sémantique

**Problème** : ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240).



## Solutions:

- 1- ajouter beaucoup de couches
- 2- utiliser des convolutions dilatées (convolutions à trous)
- 3- mettre des couches de pooling après chaque bloc convolutionnel
- 4- faire un mélange de tout ça!

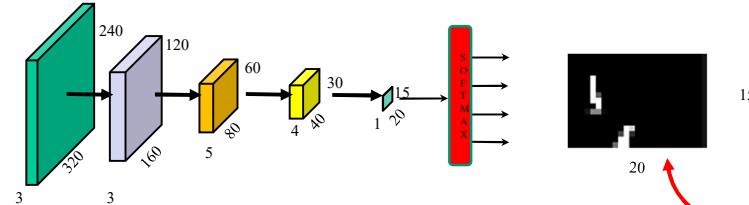
Image: Long et al. "Fully convolutional networks for semantic segmentation." ICCV, 2015.

38

38

# Segmentation sémantique

**Problème** : ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240).



**Solutions:**

- 1- ajouter beaucoup de couches
- 2- utiliser des convolutions dilatées (convolutions à trou)
- 3- mettre des couches de pooling après chaque bloc convolutionnel
- 4- faire un mélange de tout ça!



Résolution trop faible en sortie

Image: Long et al. "Fully convolutional networks for semantic segmentation." ICCV, 2015.

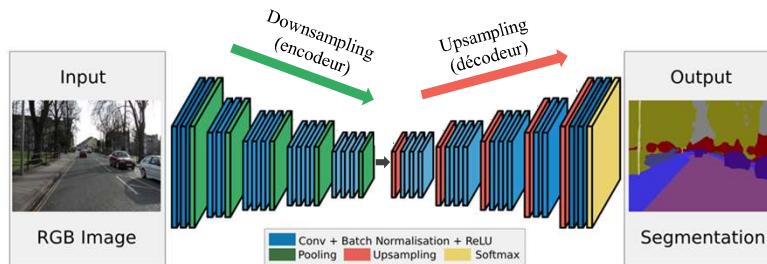
39

39

# Segmentation sémantique

**Solution** : augmenter la résolution en sortie à l'aide d'un **décodeur**.

Réseau encodeur-décodeur (ici "SegNet")



**Problème:** la résolution spatiale est perdue avec les couches de sous-échantillonnage  
(Conv + Pooling)

**Solution:** Augmenter la résolution à l'aide d'un décodeur et de couches de sur-échantillonage (??? + Conv)

Adapté de:  
Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." PAMI, 2017.

40

40

Pour **augmenter la taille** des cartes d'activation  
il faut une opération de “***upsampling***”

Deux types d'approches

- Méthodes sans paramètres => *unpooling*
- Méthode avec paramètres => *convolution transposée*

41

## Unpooling

7	8
2	1

2x2xC

« plus proche voisin »

7	7	8	8
7	7	8	8
2	2	1	1
2	2	1	1

4x4xC

7	9
2	1

« interpolation »

7	8	9	4.5
4.5	4.75	5	2.5
2	1.5	1	0.5
1	0.75	0.5	0.25

42

# Convolution transposée

L'idée ici est moins intuitive que pour du unpooling.

**Commençons par un exemple 1D...**

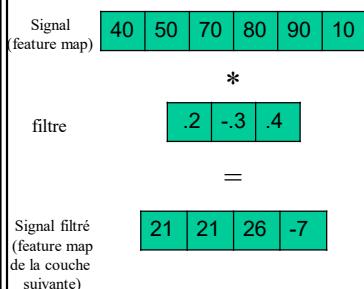
43

43

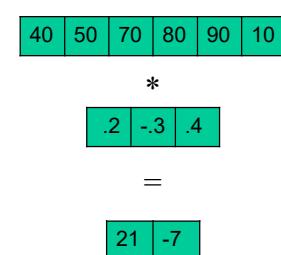
# Convolution de base

(exemple 1d)

Convolution “valid” stride =1



Convolution “valid” stride =3



44

44

## Opération matrice-vecteur

(exemple 1d)

Convolution “valid” stride =1

.2	-.3	.4	0	0	0
0	.2	-.3	.4	0	0
0	0	.2	-.3	.4	0
0	0	0	.2	-.3	.4

40
50
70
80
90
10

$$= \begin{matrix} 21 \\ 21 \\ 26 \\ -7 \end{matrix}$$

Convolution “valid” stride =3

.2	-.3	.4	0	0	0
0	0	0	.2	-.3	.4

40
50
70
80
90
10

$$= \begin{matrix} 21 \\ -7 \end{matrix}$$

45

45

## Opération matrice-vecteur

(exemple 1d)

Convolution “valid” stride =1

.2	-.3	.4	0	0	0
0	.2	-.3	.4	0	0
0	0	.2	-.3	.4	0
0	0	0	.2	-.3	.4

40
50
70
80
90
10

$$= \begin{matrix} 21 \\ 21 \\ 26 \\ -7 \end{matrix}$$

$$4 \times 6 \times 1 = 4 \times 1$$

Convolution “valid” stride =3

.2	-.3	.4	0	0	0
0	.2	-.3	.4	0	0
0	0	.2	-.3	.4	0

40
50
70
80
90
10

$$= \begin{matrix} 21 \\ -7 \end{matrix}$$

46

46

## Convolution transposée

Le but est d'avoir un filtre dont la taille des opérations **est inversée**

Convolution “valid”  
stride =1

$$\begin{array}{|c|c|c|c|} \hline .2 & 0 & 0 & 0 \\ \hline -.3 & .2 & 0 & 0 \\ \hline .4 & -.3 & .2 & 0 \\ \hline 0 & .4 & -.3 & .2 \\ \hline 0 & 0 & .4 & -.3 \\ \hline 0 & 0 & 0 & .4 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 40 \\ \hline 50 \\ \hline 70 \\ \hline 80 \\ \hline \end{array}$$

$$=$$

$$\begin{array}{|c|c|} \hline 8 & -2 \\ \hline -2 & 15 \\ \hline 15 & 15 \\ \hline 15 & 4 \\ \hline 4 & 32 \\ \hline \end{array}$$

Convolution “valid”  
stride =3

$$\begin{array}{|c|c|} \hline .2 & 0 \\ \hline -.3 & 0 \\ \hline .4 & 0 \\ \hline 0 & .2 \\ \hline 0 & -.3 \\ \hline 0 & .4 \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline 40 \\ \hline 50 \\ \hline \end{array}$$

$$=$$

$$\begin{array}{|c|c|} \hline 8 & -12 \\ \hline -12 & -16 \\ \hline -16 & 10 \\ \hline 10 & -15 \\ \hline -15 & 20 \\ \hline \end{array}$$

47

47

Convolution “valid”  
stride =1

$$\begin{array}{|c|c|c|c|} \hline .2 & 0 & 0 & 0 \\ \hline -.3 & .2 & 0 & 0 \\ \hline .4 & -.3 & .2 & 0 \\ \hline 0 & .4 & -.2 & 0 \\ \hline 0 & 0 & .4 & -.3 \\ \hline 0 & 0 & 0 & .4 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 40 \\ \hline 50 \\ \hline 70 \\ \hline 80 \\ \hline \end{array}$$

$$=$$

$$\begin{array}{|c|c|} \hline 8 & -2 \\ \hline -2 & 15 \\ \hline 15 & 15 \\ \hline 15 & 4 \\ \hline 4 & 32 \\ \hline \end{array}$$

Convolution “valid”  
stride =3

$$\begin{array}{|c|c|} \hline .2 & 0 \\ \hline -.3 & 0 \\ \hline .4 & 0 \\ \hline 0 & .2 \\ \hline 0 & -.3 \\ \hline 0 & .4 \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline 21 \\ \hline -7 \\ \hline \end{array}$$

$$=$$

$$\begin{array}{|c|c|} \hline 8 & -12 \\ \hline -12 & -16 \\ \hline -16 & 10 \\ \hline 10 & -15 \\ \hline -15 & 20 \\ \hline \end{array}$$

48

48

# Convolution transposée

Le but est d'avoir un filtre dont la taille des opérations **est inversée**

Convolution “valid”  
stride =1

.2	0	0	0
-.3	.2	0	0
.4	-.3	.2	0
0	.4	-.3	.2
0	0	.4	-.3
0	0	0	.4

Convolution “valid”  
stride =3

.2	0
-.3	0
.4	0
0	.2
0	-.3
0	.4

Matrices  
transposées

49

49

# Convolution transposée

Exemple chiffré avec 2x2 stride 1

Input	Kernel	Output
$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 2 \\ 4 & 6 \end{bmatrix} + \begin{bmatrix} 0 & 3 \\ 6 & 9 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{bmatrix}$

Autres noms:

- Déconvolution
- *Upconvolution*
- *Fractionally-strided convolution*

<https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba>

51

51

# Convolution transposée

Exemple chiffré avec 2x2 stride 1

Input	Kernel	=	Output
$\begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix}$	$\begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix}$	$=$	$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} + \begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix} + \begin{matrix} 0 & 2 \\ 4 & 6 \end{matrix} + \begin{matrix} 0 & 3 \\ 6 & 9 \end{matrix} = \begin{matrix} 0 & 0 & 1 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{matrix}$

Problème ?

<https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba>

52

52

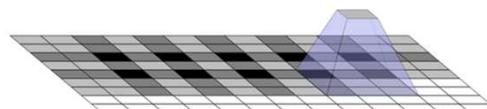
# Convolution transposée

Exemple chiffré avec 2x2 stride 1

Input	Kernel	=	Output
$\begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix}$	$\begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix}$	$=$	$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} + \begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix} + \begin{matrix} 0 & 2 \\ 4 & 6 \end{matrix} + \begin{matrix} 0 & 3 \\ 6 & 9 \end{matrix} = \begin{matrix} 0 & 0 & 1 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{matrix}$

Problème ?

Crée des artefacts  
de “damier” au chevauchement  
(*checkerboard*)



<https://distill.pub/2016/deconv-checkerboard/>

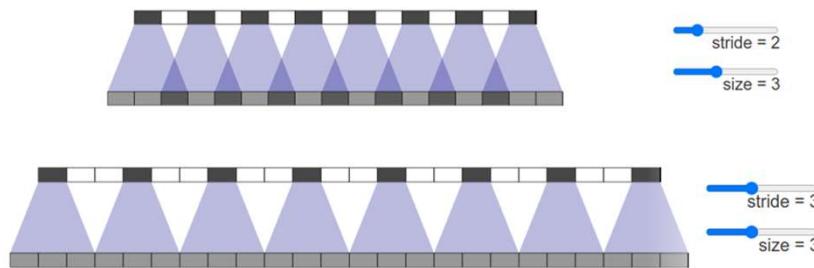
53

53

# Convolution transposée

## Solution ? (partielle)

Augmenter le *stride*  
pour éviter le chevauchement



<https://distill.pub/2016/deconv-checkerboard/>

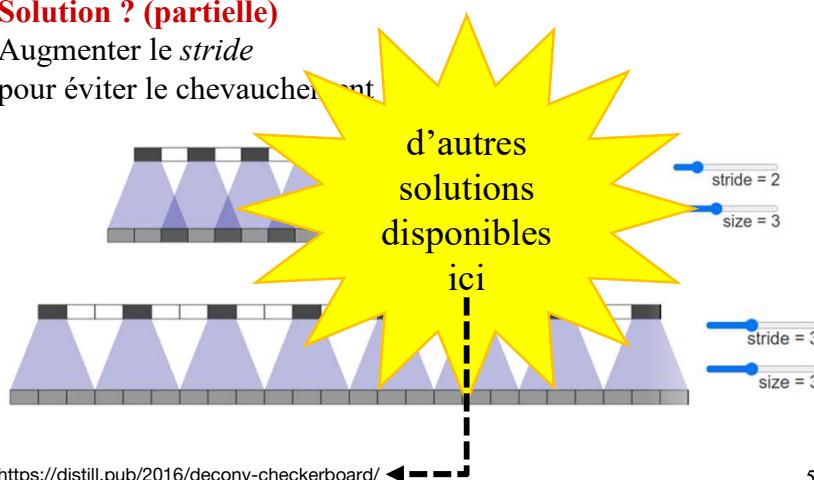
54

54

# Convolution transposée

## Solution ? (partielle)

Augmenter le *stride*  
pour éviter le chevauchement



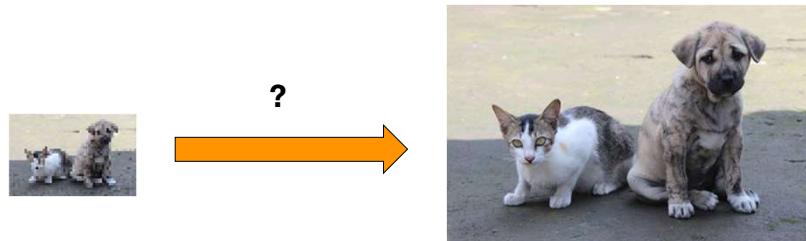
<https://distill.pub/2016/deconv-checkerboard/>

55

55

## Note: super-résolution

Comment entraîner un réseau à convolutions à faire de la super-résolution ?

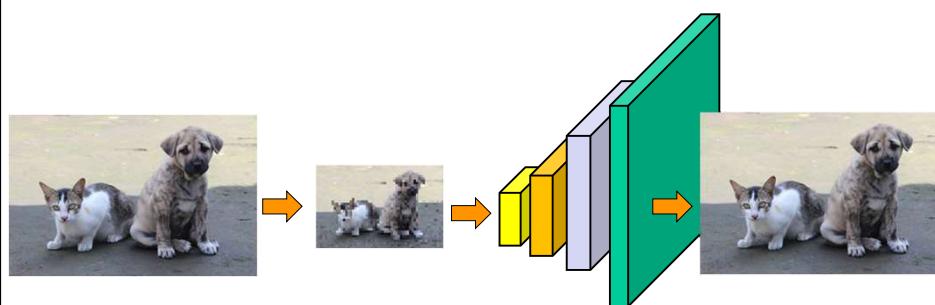


58

58

## Note: super-résolution

S'entraîner avec des images à haute-résolution qui ont été réduites !



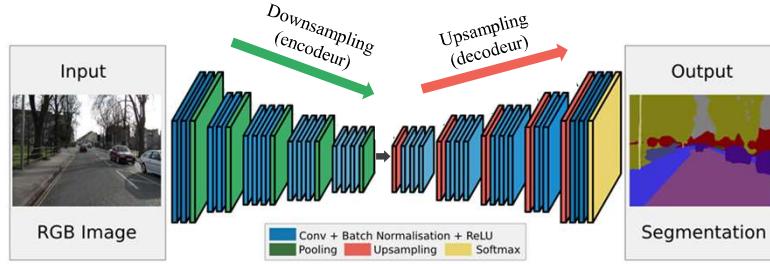
*Sub-pixel ou interpolation + convolution*

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., ... & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1874-1883).

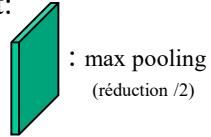
59

59

## Segmentation: Encodeur-décodeur



## Généralement:

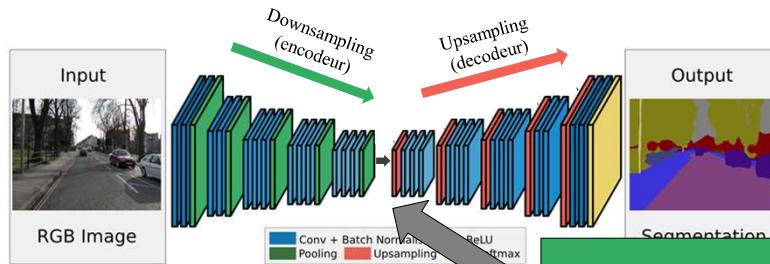


Interpolation  
: et/ou  
Convolution transposée  
(augmentation \*2)

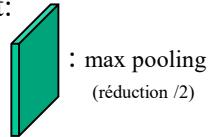
Adapté de:  
Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." PAMI, 2017.

60

## Segmentation: Encodeur-décodeur



Généralement:



Permet d'obtenir un *receptive field* couvrant toute l'image, puis d'obtenir la résolution souhaitée

Interpolation  
: et/ou  
Convolution transposée  
(augmentation \*)

Adapté de:  
Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." PAMI, 2017.

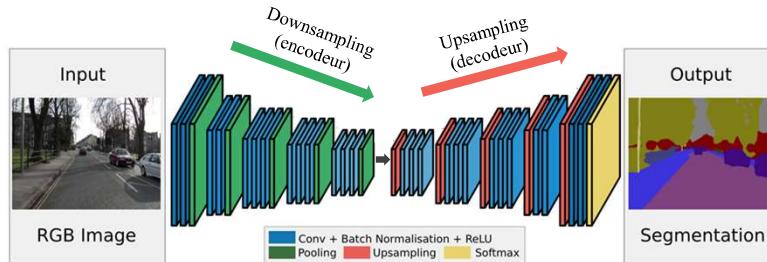
61

# Encodeur-décodeur

**Encodeur:** projette l'image d'entrée vers un espace de plus faible dimension

**Décodeur:** projette l'image de faible dimension vers l'espace souhaité

**Architecture généralement symétrique**



tiré de:

Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." PAMI, 2017.

62

62

# Encodeur-décodeur

**Encodeur:** projette l'image d'entrée vers un espace de plus faible dimension

**Décodeur:** projette l'image de faible dimension vers l'espace souhaité

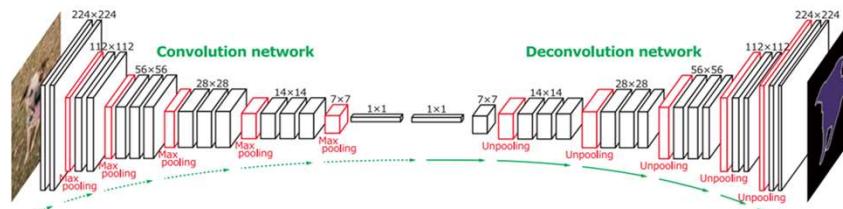


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

tiré de:

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1520-1528).

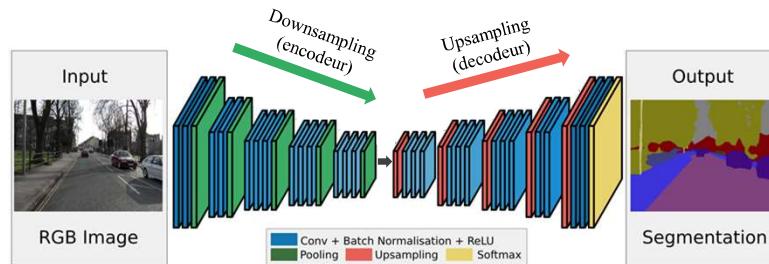
63

63

# Encodeur-décodeur

Un inconvénient des structures encodeur-décodeur

Perte d'information



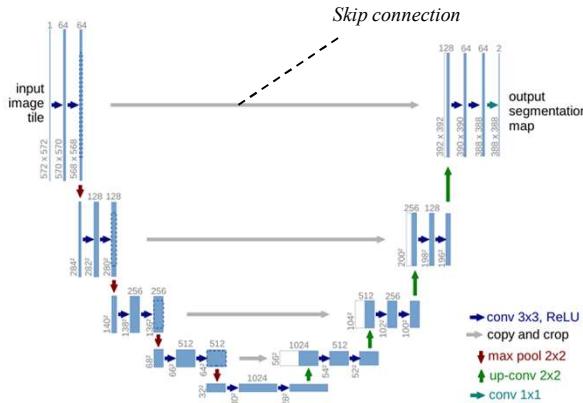
Adapté de:  
Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." PAMI, 2017.

65

65

# Solution : les *skip connections*

U-Net [Ronneberger et al., 2015]



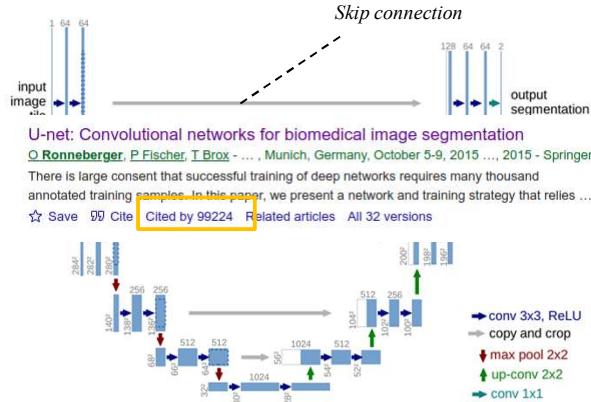
CNN le plus populaire en imagerie médicale

Image: Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." MICCAI, 2015.

66

## Solution : les *skip connections*

U-Net [Ronneberger et al., 2015]

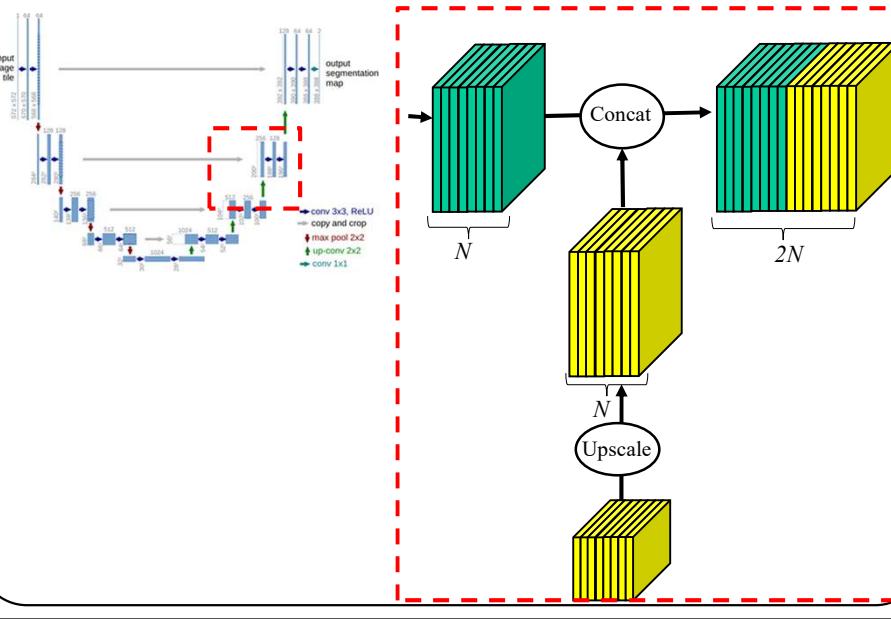


CNN le plus populaire en imagerie médicale

Image: Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." MICCAI, 2015.

67

## Solution : les *skip connections*



68

## 3D-UNet/V-Net

Identiques au Unet mais avec des **convolutions 3D**

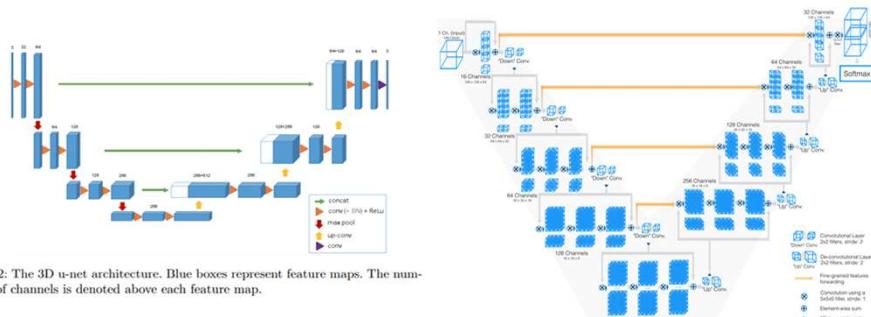


Fig. 2: The 3D u-net architecture. Blue boxes represent feature maps. The number of channels is denoted above each feature map.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016, October). 3D U-Net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention* (pp. 424-432). Springer, Cham.

Milletari, F., Navab, N., & Ahmadi, S. A. (2016, October). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)* (pp. 565-571). IEEE.

69

## 3D-UNet/V-Net

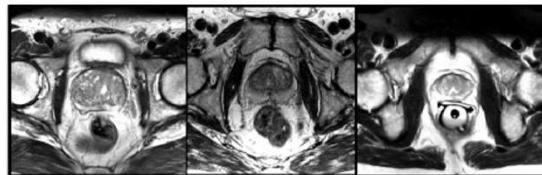


Fig. 1. Slices from MRI volumes depicting prostate. This data is part of the PROMISE2012 challenge dataset [7].



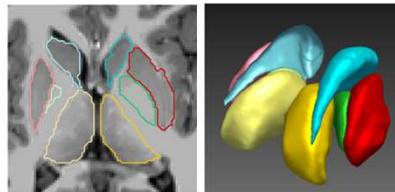
Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016, October). 3D U-Net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention* (pp. 424-432). Springer, Cham.

Milletari, F., Navab, N., & Ahmadi, S. A. (2016, October). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)* (pp. 565-571). IEEE.

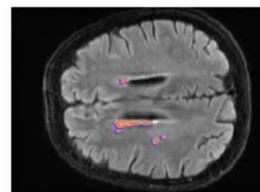
70

# Imagerie médicale

## Exemples d'images 3D en imagerie médicale



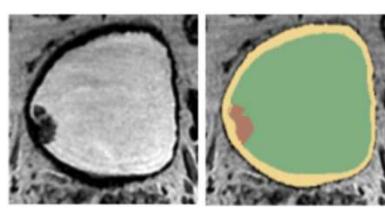
**Subcortical brain structures**  
[Dolz et al, 2018]



**White matter hyperintensities**  
[Dolz et al, 2019]



**Intervertebral disks**  
[Dolz et al, 2019]



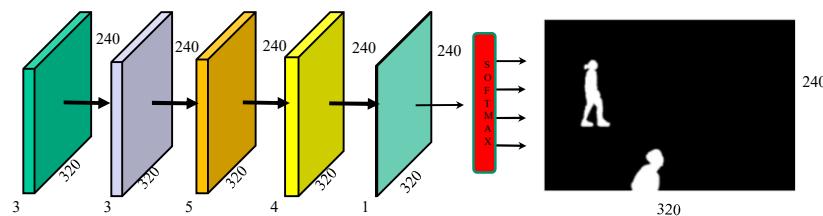
**Prostate wall and tumor**  
[Dolz et al, 2018]

71

71

# Segmentation sémantique

**Problème :** ce modèle a un champ récepteur (*receptive field*) relativement petit (ici 32x32). Au lieu, on aimerait que les pixels de sortie aient un champ récepteur de la taille de l'image d'entrée (ici 320x240).



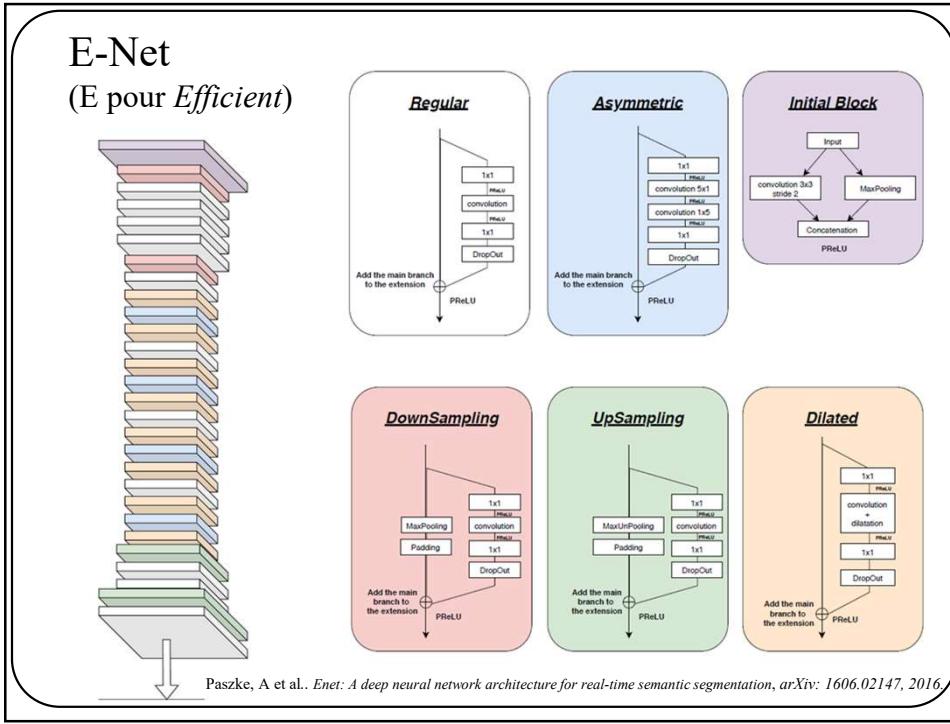
## Solutions:

- 1- ajouter beaucoup de couches
- 2- utiliser des convolutions dilatées (convolutions à trous)
- 3- mettre des couches de pooling après chaque bloc convolutionnel
- 4- faire un mélange de tout ça

Image: Long et al. "Fully convolutional networks for semantic segmentation." ICCV, 2015.

72

72



73

**E-Net : le “combo” ultime**  
(E pour Efficient)

Table 1: ENet architecture. Output sizes are given for an example input of  $512 \times 512$ .

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
$4 \times$ bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
<i>Repeat section 2, without bottleneck2.0</i>		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, arXiv: 1606.02147, 2016.

74

## E-Net

(E pour *Efficient*)

Table 2: Performance comparison.

Model	NVIDIA TX1						NVIDIA Titan X					
	480×320		640×360		1280×720		640×360		1280×720		1920×1080	
	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps
SegNet	757	1.3	1251	0.8	-	-	69	14.6	289	3.5	637	1.6
ENet	47	21.1	69	14.6	262	3.8	7	135.4	21	46.8	46	21.6

Table 3: Hardware requirements. FLOPs are estimated for an input of  $3 \times 640 \times 360$ .

	GFLOPs	Parameters	Model size (fp16)
SegNet	286.03	29.46M	56.2 MB
ENet	3.83	0.37M	0.7 MB

**Très efficace!!! 300 fois moins de calculs pour des résultats similaires à SegNet**

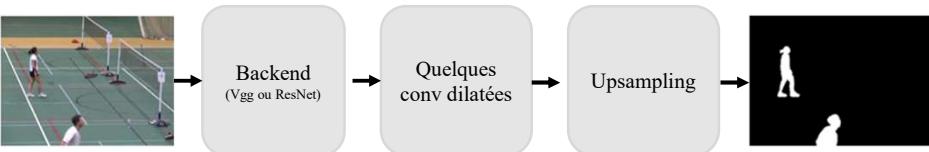
Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, *arXiv: 1606.02147*, 2016.

75

## DeepLab V1,V2,V3, PSPNet, MSCADC, etc.

Plusieurs méthodes utilisent à la fois des **convolutions dilatées** et du « **upsampling** ».

Configuration typique:



H.Zhao, J.Shi, X. Qi, X. Wang, J. Jia, Pyramid Scene Parsing Network, CVPR 2017

L. Chen, G. Papandreou, I.Kokkinos, K.Murphy, A.L. Yuille  
Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, ICLR 2015

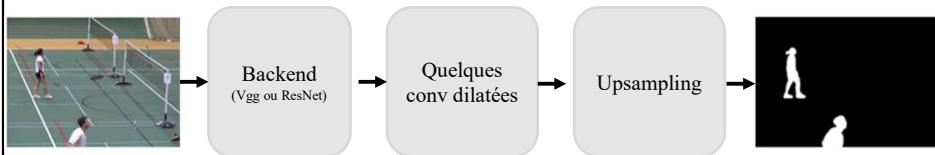
F. Yu, V. Koltun, Multi-Scale Context Aggregation by Dilated Convolution, ICLR 2016

L. Chen, G. Papandreou, I.Kokkinos, K.Murphy, A.L. Yuille  
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, PAMI 2016

76

## DeepLab V1,V2,V3, PSPNet, MSCADC, etc.

Une méthode très populaire : **DeepLab**



H.Zhao, J.Shi, X. Qi, X. Wang, J. Jia, Pyramid Scene Parsing Network, CVPR 2017

L. Chen, G. Papandreou, I.Kokkinos, K.Murphy, A.L. Yuille  
Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, ICLR 2015

F. Yu, V. Koltun, Multi-Scale Context Aggregation by Dilated Convolution, ICLR 2016

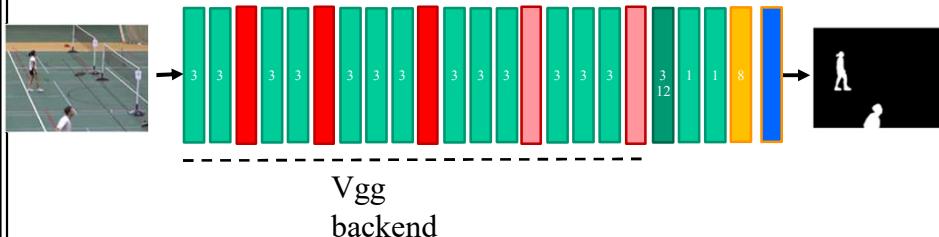
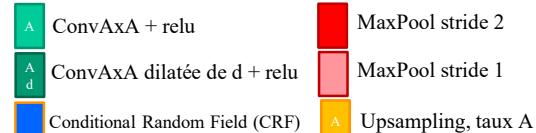
L. Chen, G. Papandreou, I.Kokkinos, K.Murphy, A.L. Yuille  
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, PAMI 2016

77

## DeepLab V1/V2

(certains détails peuvent varier)

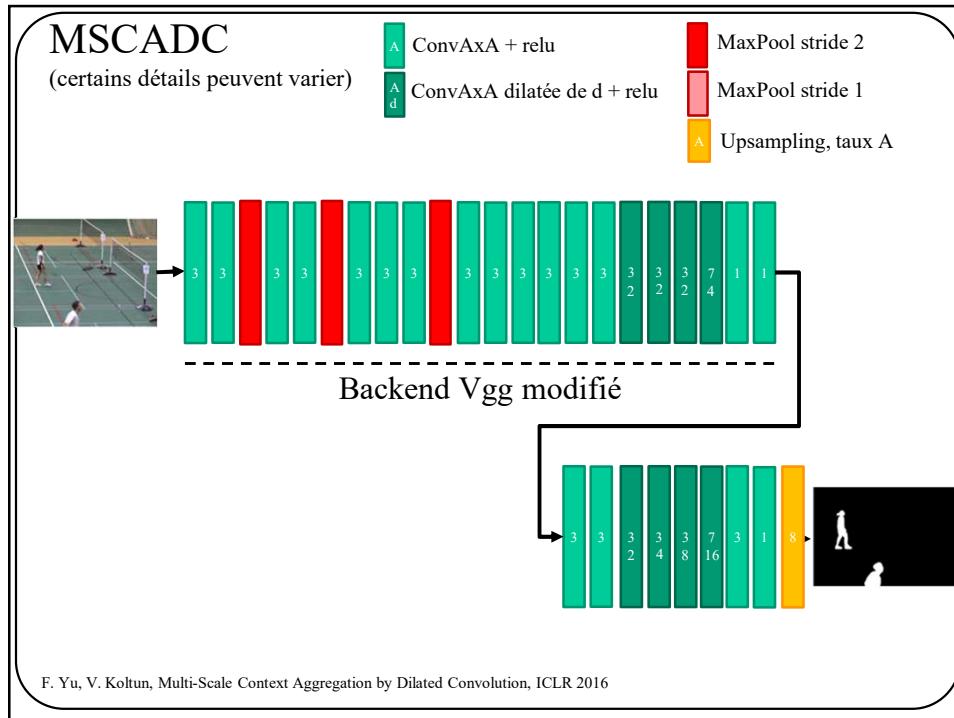
Ex.: Vgg backend



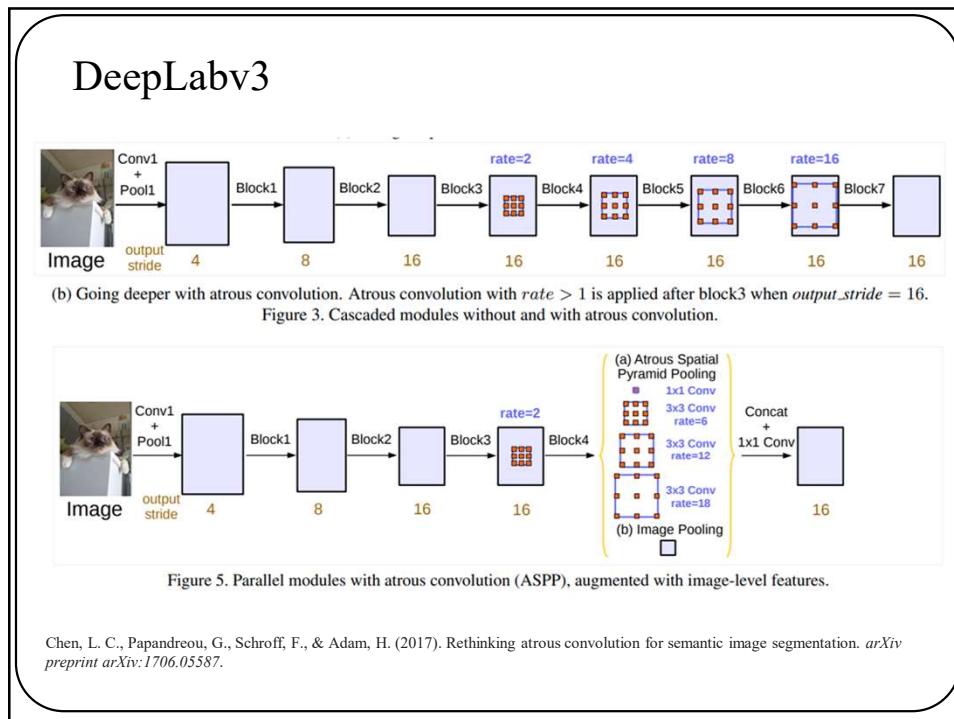
L. Chen, G. Papandreou, I.Kokkinos, K.Murphy, A.L. Yuille  
Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, ICLR 2015

On peut aussi mettre un backend ResNet

78



79

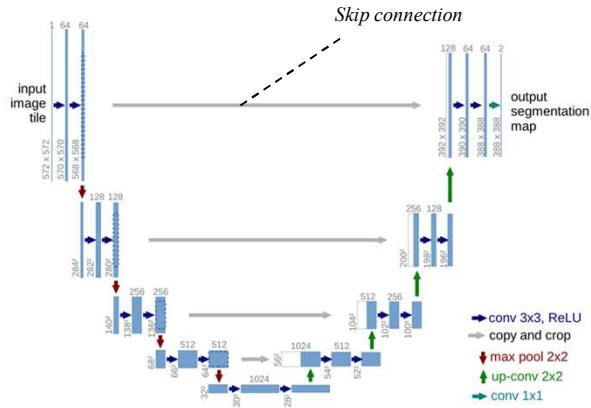


Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

80

## En pratique:

**U-Net [Ronneberger et al., 2015]**



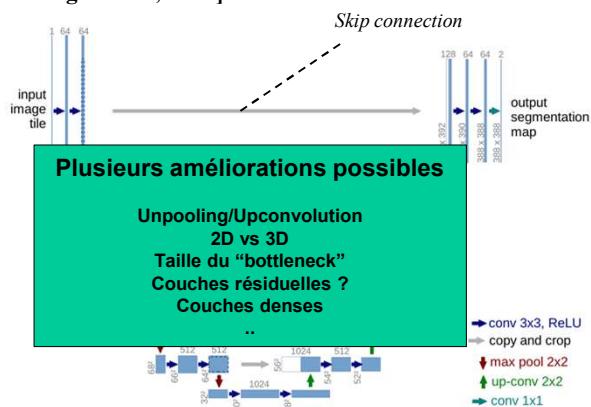
**CNN le plus populaire en imagerie médicale**

Image: Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." MICCAI, 2015.

81

## En pratique:

**U-Net [Ronneberger et al., 2015]**



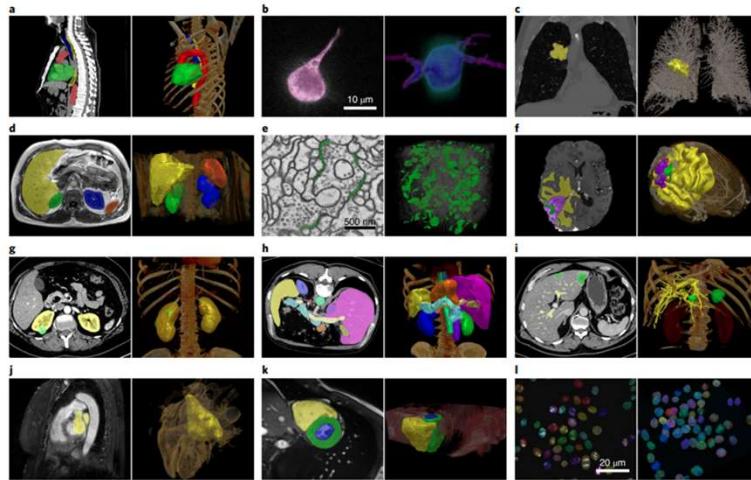
**CNN le plus populaire en imagerie médicale**

Image: Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." MICCAI, 2015.

82

## En pratique : nnU-Net

L'imagerie médicale est un domaine vaste et varié

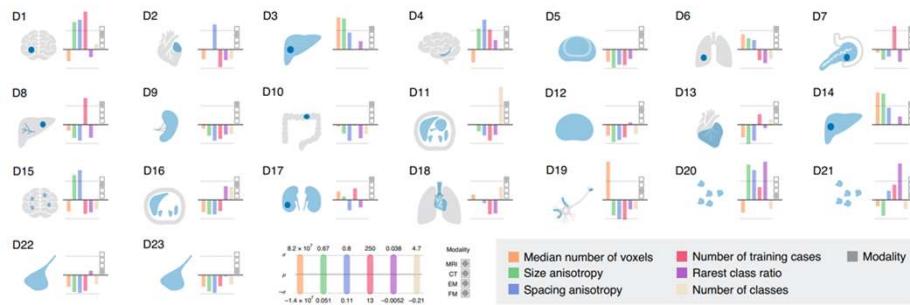


Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2), 203-211.

83

## En pratique : nnU-Net

L'imagerie médicale est un domaine vaste et varié

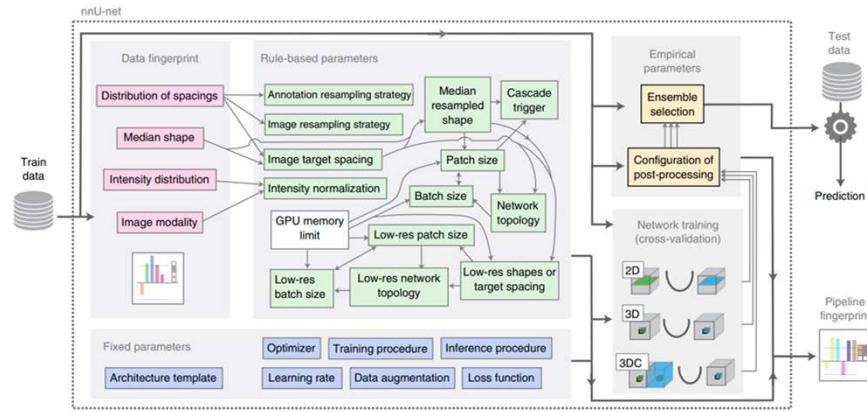


Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2), 203-211.

84

## En pratique : nnU-Net (*no-new-net*)

L'imagerie médicale est un domaine vaste et varié



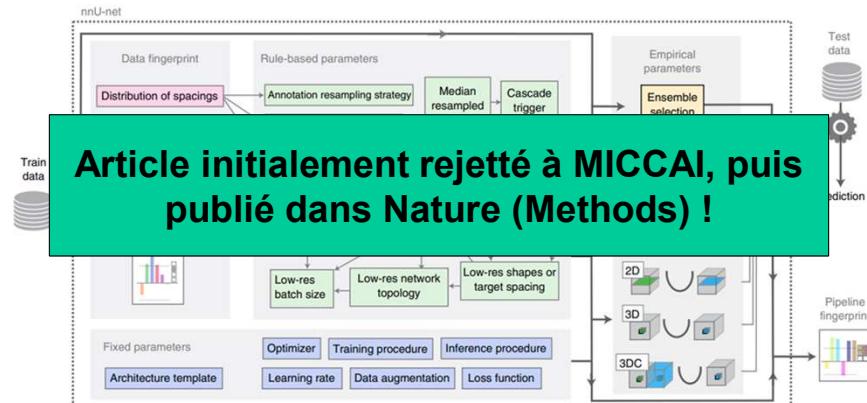
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2), 203-211.

85

## En pratique : nnU-Net (*no-new-net*)

L'imagerie médicale est un domaine vaste et varié

**Article initialement rejeté à MICCAI, puis publié dans Nature (Methods) !**



Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2), 203-211.

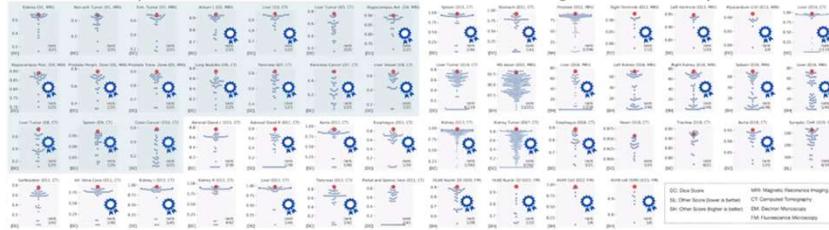
86

# En pratique : nnU-Net (*no-new-net*)

nnU-Net est de loin le meilleur!

## nnU-Net Application: Out-of-the-box Quantitative Results

Evaluation performed on 23 datasets from biomedical segmentation competitions



First place in 33 out of 53 segmentation tasks...

...despite competing against specialized handcrafted solutions\*

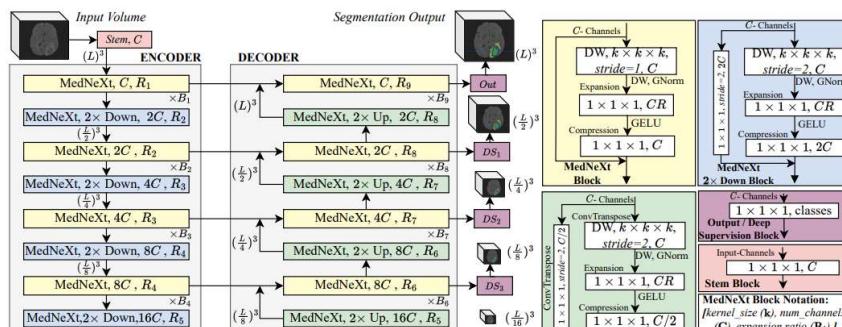
Beyond the Patterns 29 - Fabian Isensee - nnU-Net: self-configuring deep learning image segmentation  
<https://www.youtube.com/watch?v=C6tpnJRpt90>

Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2), 203-211.

87

# La recherche continue...

MedNeXt, un réseau parmi les plus performants



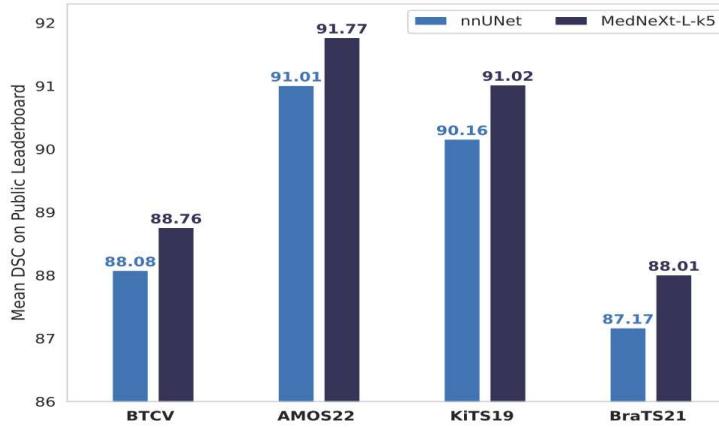
(a) MedNeXt macro and block architecture

Roy et al. *MedNeXt: Transformer-driven Scaling of ConvNets for Medical Image Segmentation*, in proc of MICCAI 2023

88

## La recherche continue...

MedNeXt, meilleur que nnUNet  
(4 bases de données de segmentation médicale)



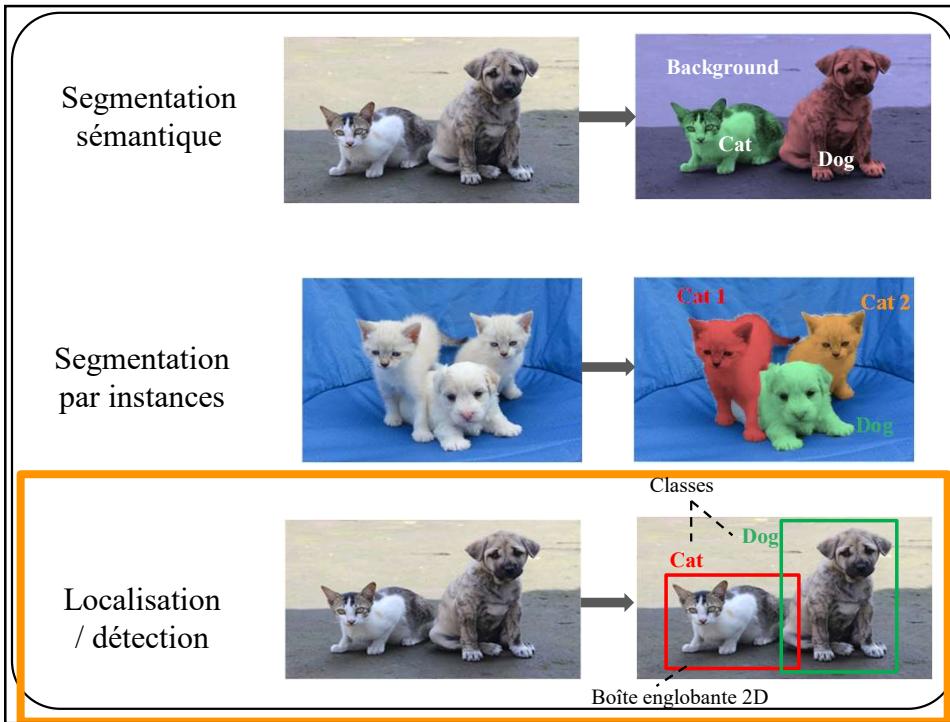
Roy et al. *MedNeXt: Transformer-driven Scaling of ConvNets for Medical Image Segmentation*, in proc of MICCAI 2023

89

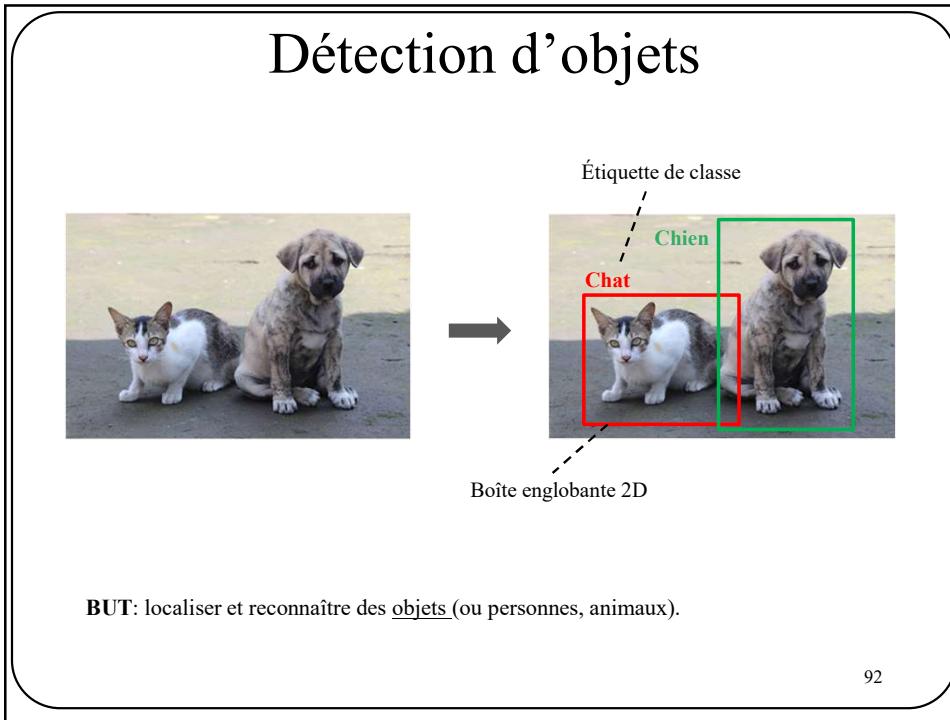
## DÉTECTION D'OBJETS

90

90



91



92

92

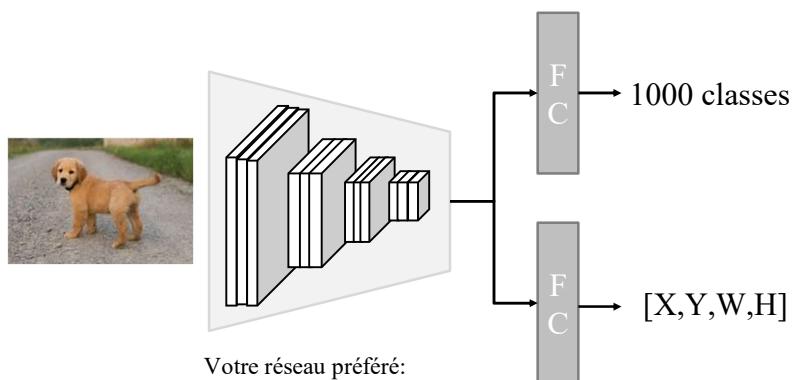
## Détection d'un seul objet



93

93

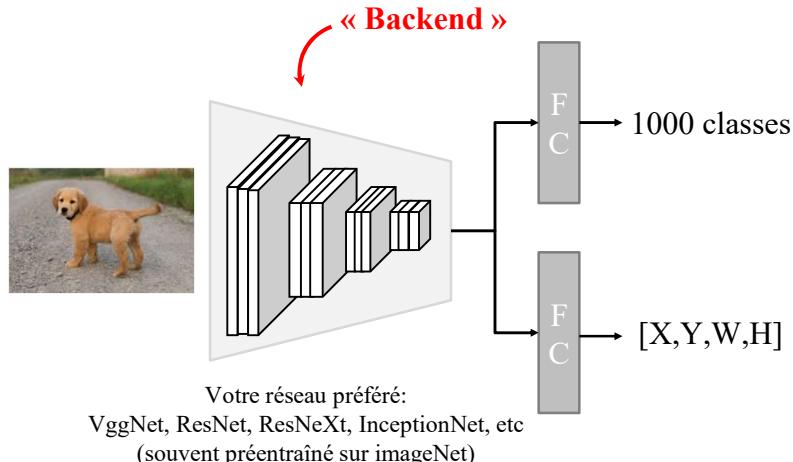
## Détection d'un seul objet



94

94

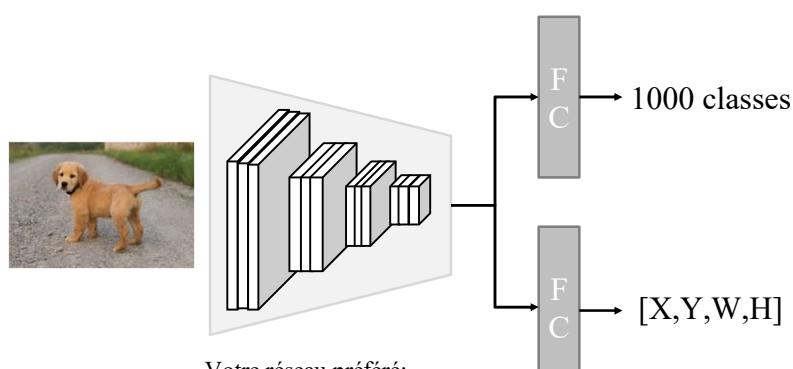
## Détection d'un seul objet



95

95

## Détection d'un seul objet



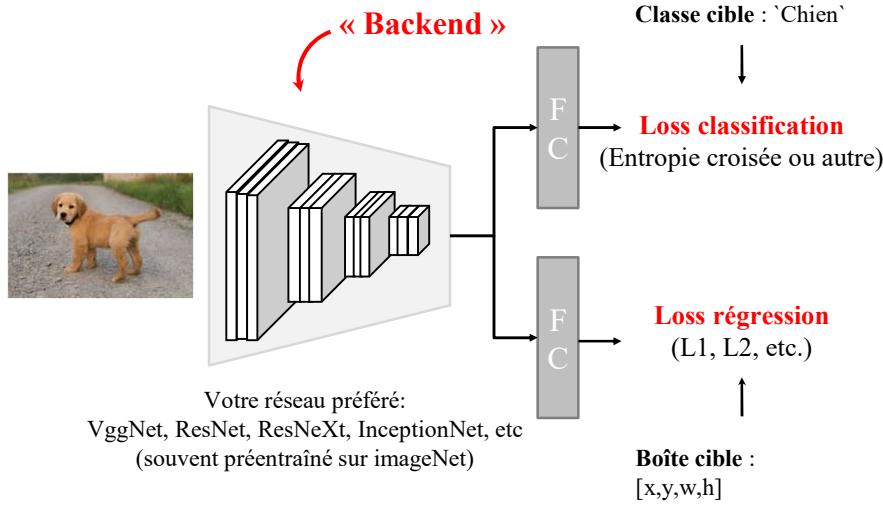
### Deux têtes de sortie :

- 1- Classification
- 2- Régression

96

96

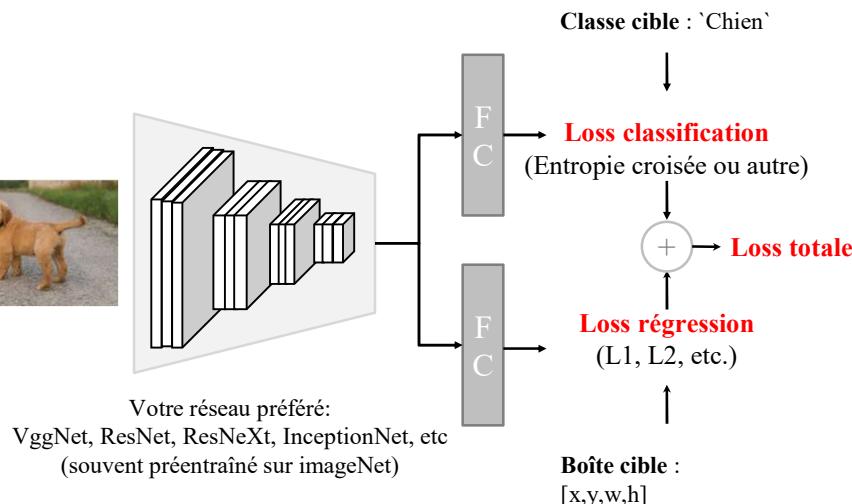
## Détection d'un seul objet



97

97

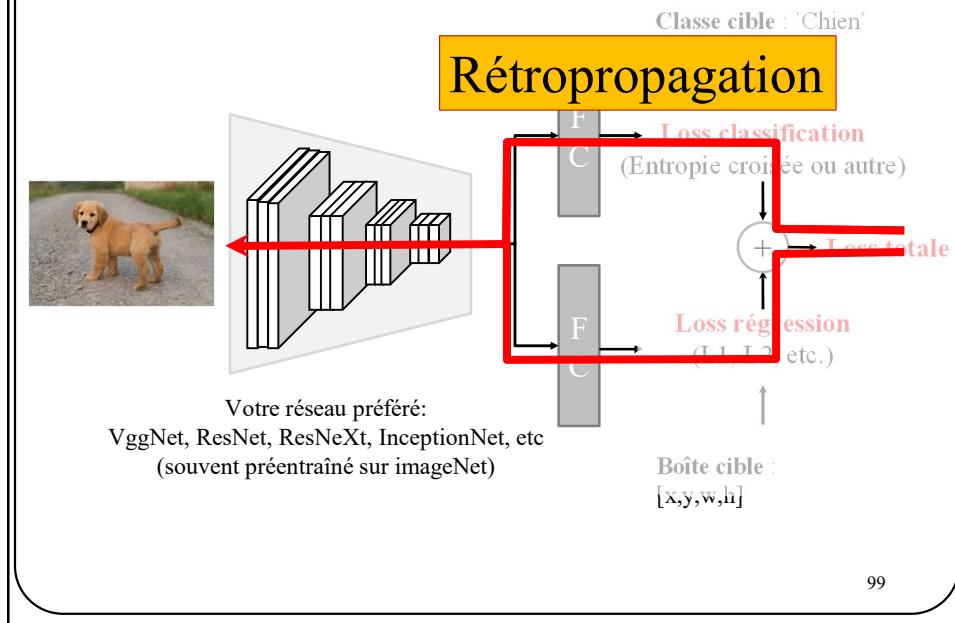
## Détection d'un seul objet



98

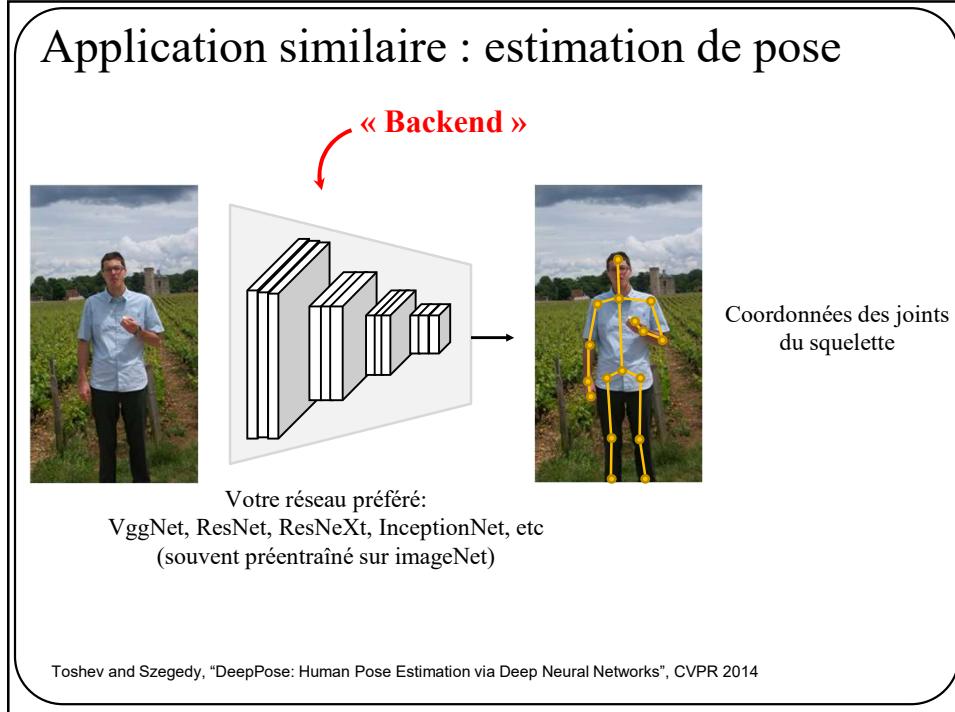
98

## Détection d'un seul objet



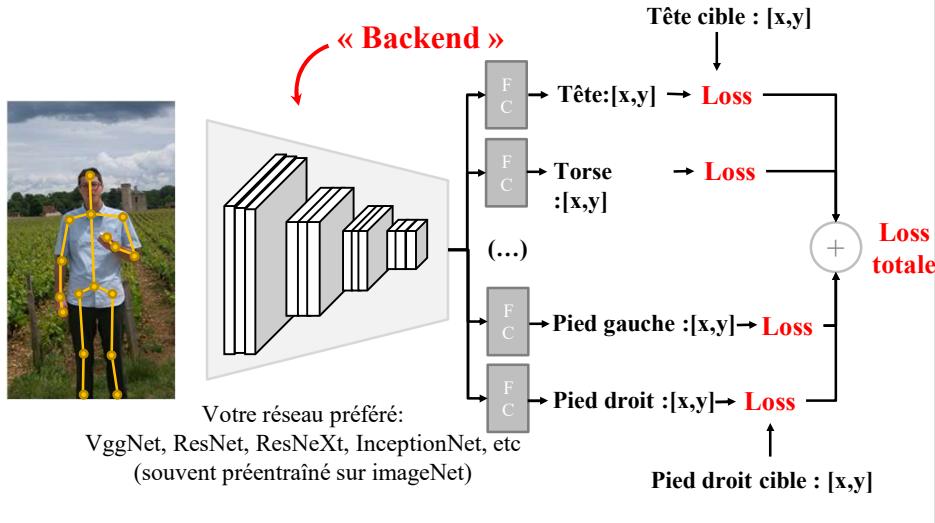
99

## Application similaire : estimation de pose



100

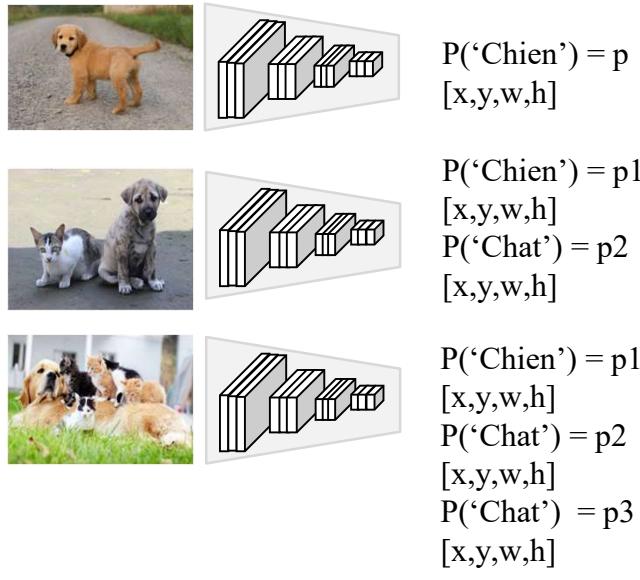
## Application similaire : estimation de pose



Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

101

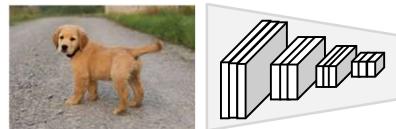
## Localisation de plusieurs objets



102

## Localisation de plusieurs objets

**Problème:** chaque image commande une sortie différente



‘Chien’  
[x,y,w,h]



‘Chien’  
[x,y,w,h]  
‘Chat’ [x,y,w,h]



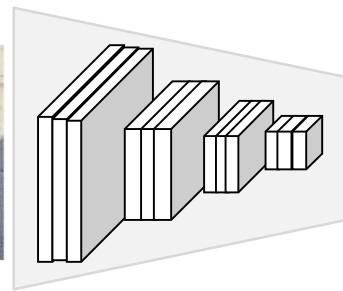
‘Chien’  
[x,y,w,h]  
‘Chat’ [x,y,w,h]  
‘Chat’ [x,y,w,h]  
...

103

## Localisation de plusieurs objets

**Solution 1 :** appliquer un CNN à une fenêtre coulissante

3 classes : ‘Chien’, ‘Chat’, ‘Fond’



$$P(\text{‘Fond’}) = 1$$

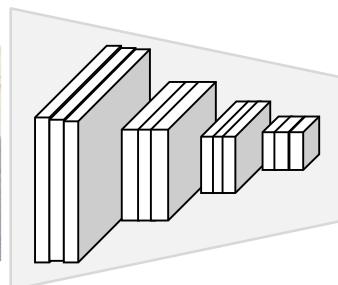
Votre réseau préféré:  
VggNet, ResNet, ResNeXt, etc  
(souvent préentraîné sur imageNet)

104

## Localisation de plusieurs objets

**Solution 1 :** appliquer un CNN à une fenêtre coulissante

3 classes : 'Chien', 'Chat', 'Fond'



$$P(\text{'Fond'}) = 0.7$$

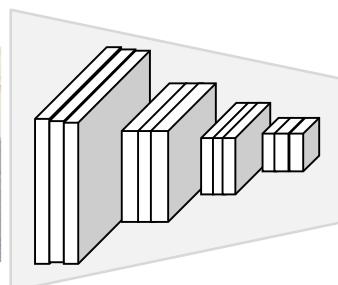
Votre réseau préféré:  
VggNet, ResNet, ResNeXt, etc  
(souvent préentraîné sur imageNet)

105

## Localisation de plusieurs objets

**Solution 1 :** appliquer un CNN à une fenêtre coulissante

3 classes : 'Chien', 'Chat', 'Fond'



$$P(\text{'Chien'}) = 0.9$$

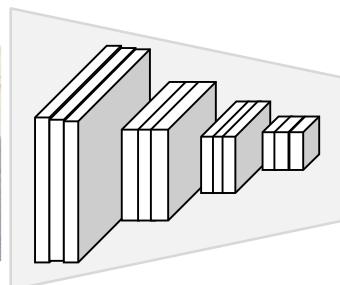
Votre réseau préféré:  
VggNet, ResNet, ResNeXt, etc  
(souvent préentraîné sur imageNet)

106

## Localisation de plusieurs objets

**Solution 1 :** appliquer un CNN à une fenêtre coulissante

3 classes : 'Chien', 'Chat', 'Fond'



$$P(\text{'Fond'}) = 0.5$$

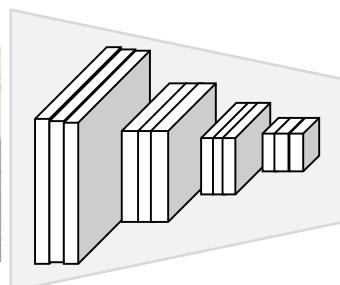
Votre réseau préféré:  
VggNet, ResNet, ResNeXt, etc  
(souvent préentraîné sur imageNet)

107

## Localisation de plusieurs objets

**Solution 1 :** appliquer un CNN à une fenêtre coulissante

3 classes : 'Chien', 'Chat', 'Fond'



$$P(\text{'Chat'}) = 0.9$$

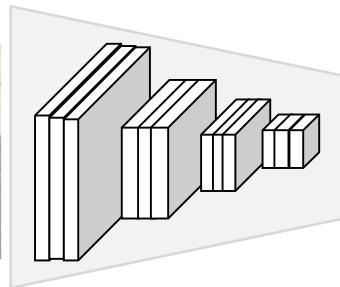
Votre réseau préféré:  
VggNet, ResNet, ResNeXt, etc  
(souvent préentraîné sur imageNet)

108

## Localisation de plusieurs objets

**Solution 1 :** appliquer un CNN à une fenêtre coulissante

3 classes : 'Chien', 'Chat', 'Fond'



$$P(\text{'Chat'}) = 0.9$$

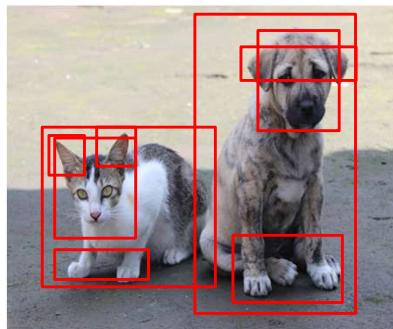
L'inconvénient de cette méthode est qu'elle requière de traiter un très grand nombre de fenêtres coulissantes de plusieurs dimensions.

109

## Localisation de plusieurs objets

**Solution 2 :** Présélectionner un nombre restreint de fenêtres.

Il est relativement facile et rapide de trouver ~1000 fenêtres susceptibles de contenir un objet d'intérêt



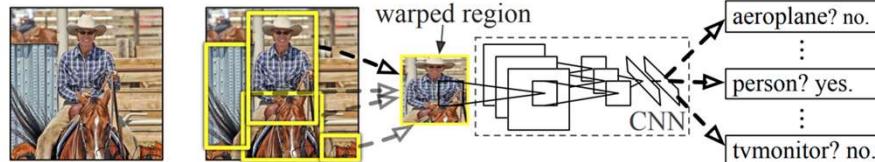
On appelle ce type de méthodes  
« *Region proposal method* »

Alexe et al., "Measuring the objectness of image windows", TPAMI 2012  
Uijlings et al., "Selective Search for Object Recognition", IJCV 2013  
Cheng et al., "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014  
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

110

## R-CNN [Girshick et al, 2014]

### R-CNN: Regions with CNN features



1. Extraire des régions (de 1000 à 2000)  
à l'aide d'une « *region proposal method* »

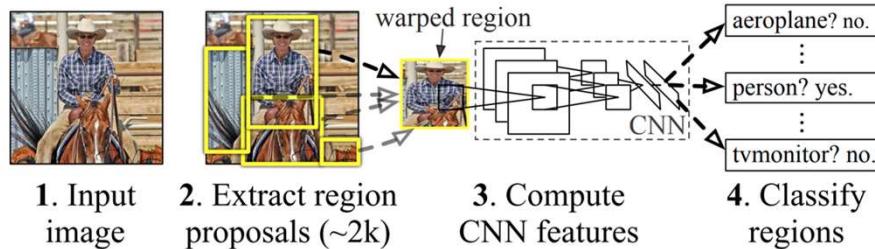
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

111

111

## R-CNN [Girshick et al, 2014]

### R-CNN: Regions with CNN features



1. Extraire des régions (1000 à 2000)  
à l'aide d'une « *region proposal method* »
2. Crop + réajuster la taille de chaque région afin  
qu'elles soient toutes identiques

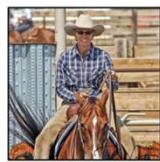
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

112

112

## R-CNN [Girshick et al, 2014]

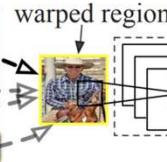
### R-CNN: Regions with CNN features



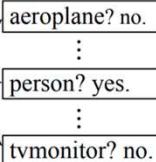
1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features



4. Classify regions

1. Extraire des régions (1000 à 2000)  
à l'aide d'une « *region proposal method* »

2. Crop + réajuster la taille de chaque région afin  
qu'elles soient toutes identiques

3. Extraire des caractéristiques profondes  
(sortie de AlexNet ou VggNet avant le Softmax)

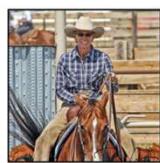
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

113

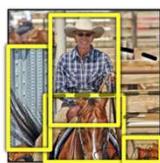
113

## R-CNN [Girshick et al, 2014]

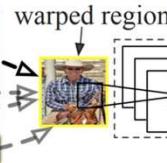
### R-CNN: Regions with CNN features



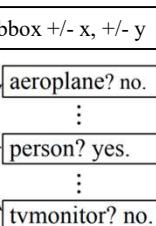
1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features



4. Classify regions

1. Extraire des régions (1000 à 2000)  
à l'aide d'une « *region proposal method* »

4. Classification & localization  
(localisation pour ajuster la position des régions)

2. Crop + réajuster la taille de chaque région afin  
qu'elles soient toutes identiques

3. Extraire des caractéristiques profondes  
(sortie de AlexNet ou VggNet avant le Softmax)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

114

114

## R-CNN [Girshick et al, 2014]

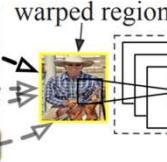
### R-CNN: Regions with CNN features



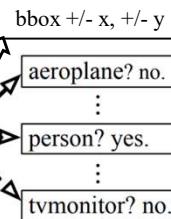
1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features



4. Classify regions

1. Extraire des régions (1000 à 2000) à l'aide d'une « *region proposal method* »

2. Crop + réajuster la taille de chaque région afin qu'elles soient toutes identiques

3. Extraire des caractéristiques profondes (sortie de AlexNet ou VggNet avant le Softmax)

4. Classification & localisation (localisation pour ajuster la position des régions)

5. Éliminer les fenêtres qui se chevauchent en ne gardant que les plus “probables”

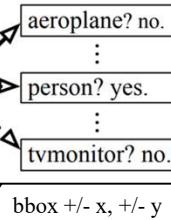
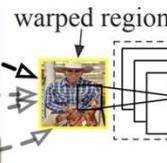
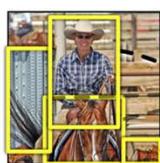
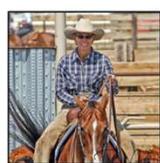
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

115

115

## R-CNN [Girshick et al, 2014]

### R-CNN: Regions with CNN features



Composantes:

- *Region-proposal method*
- “Backend” (AlexNet ou VGG16) pré-entraîné sur ImageNet puis *finetuned* sur Pascal VOC
- SVM *par classe* pour la classification
- Régresseur pour bouger les régions

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

116

116

## (Parenthèse)

Pour mieux comprendre

117

117

### *Intersection over Union*

Aussi appelé “Jaccard Index”

Comment comparer les régions ?

$$\frac{|\text{Intersection}|}{|\text{Union}|}$$

0.5 est “correct”

0.7 est bon

0.9 est excellent



118

## *Intersection over Union*

Aussi appelé “Jaccard Index”

Comment comparer les régions ?

$$\frac{\text{Intersection}}{\text{Union}}$$

|Union|

0.5 est “correct”

0.7 est bon

0.9 est excellent



119

## *Intersection over Union*

Aussi appelé “Jaccard Index”

Comment comparer les régions ?

$$\frac{\text{Intersection}}{\text{Union}}$$

|Union|

0.5 est “correct”

0.7 est bon

0.9 est excellent

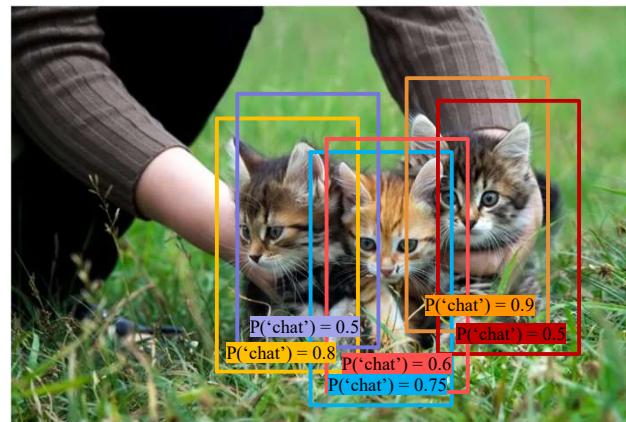


120

## “Non-Max Suppression”

Comment éviter les duplicitas (2 boîtes sur un même objet) ?

1. Sélectionner la boîte avec le plus haut score
2. Éliminer les boîtes avec un  $\text{IoU} > \epsilon$  (p.e. 0.7) ayant un score moins élevé
3. Répéter jusqu'à ce qu'aucune boîte ne puisse être éliminée

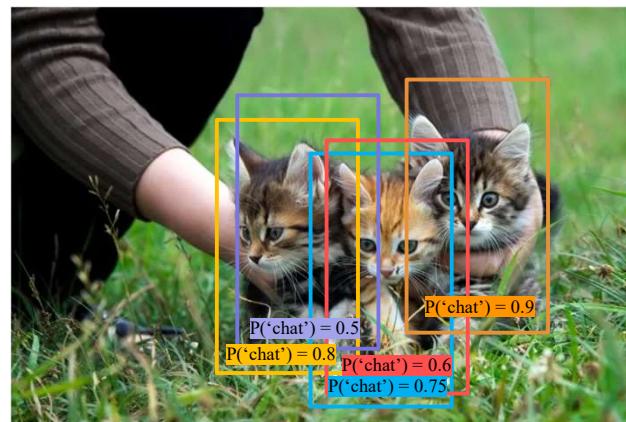


121

## “Non-Max Suppression”

Comment éviter les duplicitas (2 boîtes sur un même objet) ?

1. Sélectionner la boîte avec le plus haut score
2. Éliminer les boîtes avec un  $\text{IoU} > \epsilon$  (p.e. 0.7) ayant un score moins élevé
3. Répéter jusqu'à ce qu'aucune boîte ne puisse être éliminée

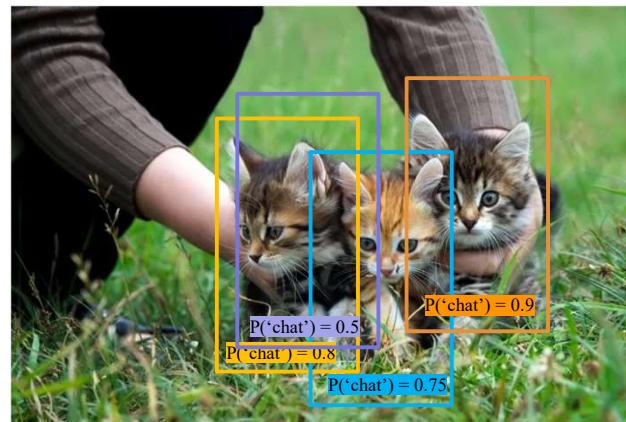


122

## “Non-Max Suppression”

Comment éviter les duplicitas (2 boîtes sur un même objet) ?

1. Sélectionner la boîte avec le plus haut score
2. Éliminer les boîtes avec un  $\text{IoU} > \epsilon$  (p.e. 0.7) ayant un score moins élevé
3. Répéter jusqu'à ce qu'aucune boîte ne puisse être éliminée

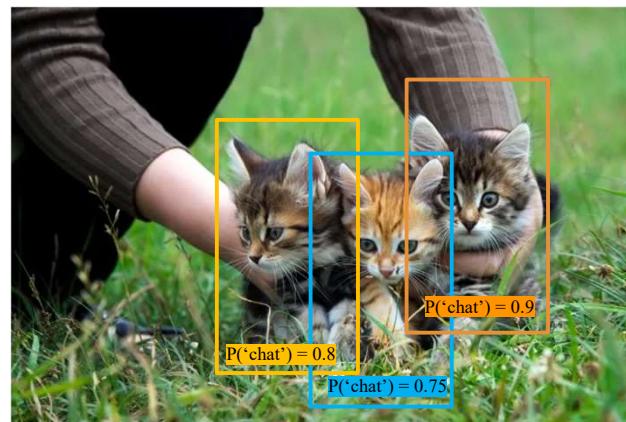


123

## “Non-Max Suppression”

Comment éviter les duplicitas (2 boîtes sur un même objet) ?

1. Sélectionner la boîte avec le plus haut score
2. Éliminer les boîtes avec un  $\text{IoU} > \epsilon$  (p.e. 0.7) ayant un score moins élevé
3. Répéter jusqu'à ce qu'aucune boîte ne puisse être éliminée



124

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction ET de la localisation

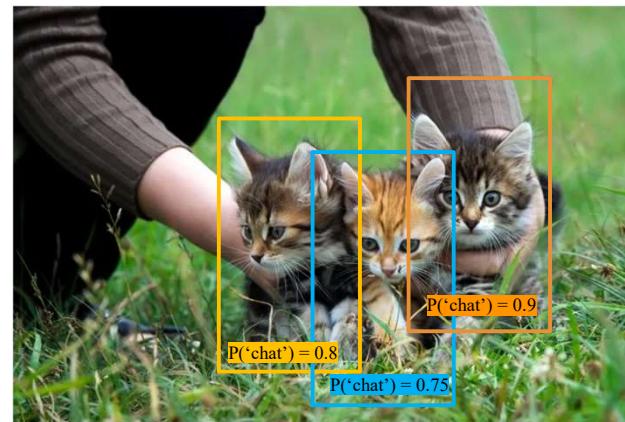
Classification:

- “Top 1%”
- “Top 5%”
- Top ..%

Segmentation:

- (Sørensen-)Dice/F1
- IoU/Jaccard Index
- Précision

Localisation ?



125

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction ET de la localisation



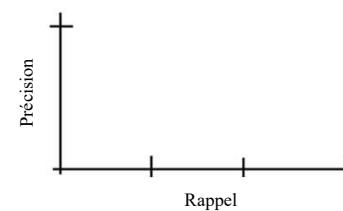
1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe

a. Pour chaque détection

- i. Ordonner les détections par probabilité
- ii. Prendre la détection la plus probable
  1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante
  2. Sinon, l'indiquer comme faux positif
- iii. Tracer le point

b. Calculer l'aire sous la courbe

3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

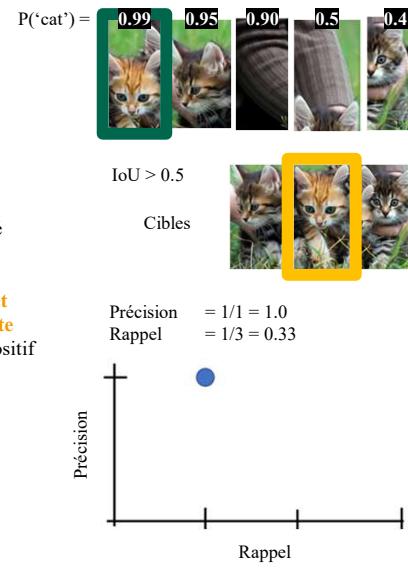
126

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction *ET* de la localisation

1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe
  - a. Pour chaque détection
    - i. Ordonner les détections par probabilité
    - ii. Prendre la détection la plus probable
      - 1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante**
      2. Sinon, l'indiquer comme faux positif
    - iii. Tracer le point
  - b. Calculer l'aire sous la courbe
3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

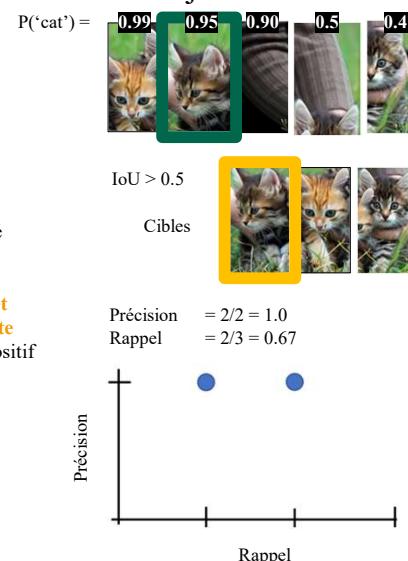
127

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction *ET* de la localisation

1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe
  - a. Pour chaque détection
    - i. Ordonner les détections par probabilité
    - ii. Prendre la détection la plus probable
      - 1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante**
      2. Sinon, l'indiquer comme faux positif
    - iii. Tracer le point
  - b. Calculer l'aire sous la courbe
3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

128

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction *ET* de la localisation

1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe

a. Pour chaque détection

- i. Ordonner les détections par probabilité
- ii. Prendre la détection la plus probable
  1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante
- 2. Sinon, l'indiquer comme faux positif**

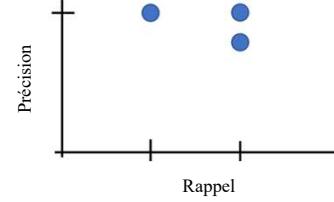
iii. Tracer le point

b. Calculer l'aire sous la courbe

3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes



Précision = 2/3 = 0.67  
Rappel = 2/3 = 0.67



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

129

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction *ET* de la localisation

1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe

a. Pour chaque détection

- i. Ordonner les détections par probabilité
- ii. Prendre la détection la plus probable
  1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante
- 2. Sinon, l'indiquer comme faux positif**

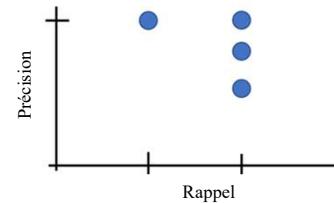
iii. Tracer le point

b. Calculer l'aire sous la courbe

3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes



Précision = 2/4 = 0.5  
Rappel = 2/3 = 0.67



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

130

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction *ET* de la localisation

1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe

a. Pour chaque détection

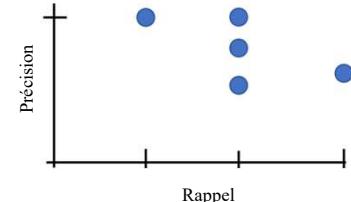
- i. Ordonner les détections par probabilité
- ii. Prendre la détection la plus probable
  1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante
  2. Sinon, l'indiquer comme faux positif
- iii. Tracer le point

b. Calculer l'aire sous la courbe

3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes



Précision =  $3/5 = 0.6$   
Rappel =  $3/3 = 1.0$



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

131

## “mean Average Precision”

Comment mesurer la performance d'un détecteur d'objet ?

Doit évaluer la qualité de la prédiction *ET* de la localisation

1. Faire la détection sur toutes les images de test
2. Pour chaque image et chaque classe

a. Pour chaque détection

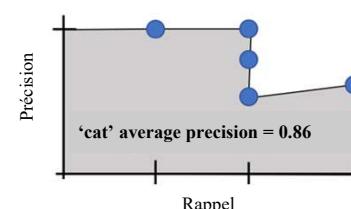
- i. Ordonner les détections par probabilité
- ii. Prendre la détection la plus probable
  1. Si elle correspond à une cible, l'indiquer comme vrai positif et éliminer la cible correspondante
  2. Sinon, l'indiquer comme faux positif
- iii. Tracer le point

b. Calculer l'aire sous la courbe  
=> *average precision*

3. Faire la moyenne pour chaque image
4. Faire la moyenne pour toutes les classes => **mAP**

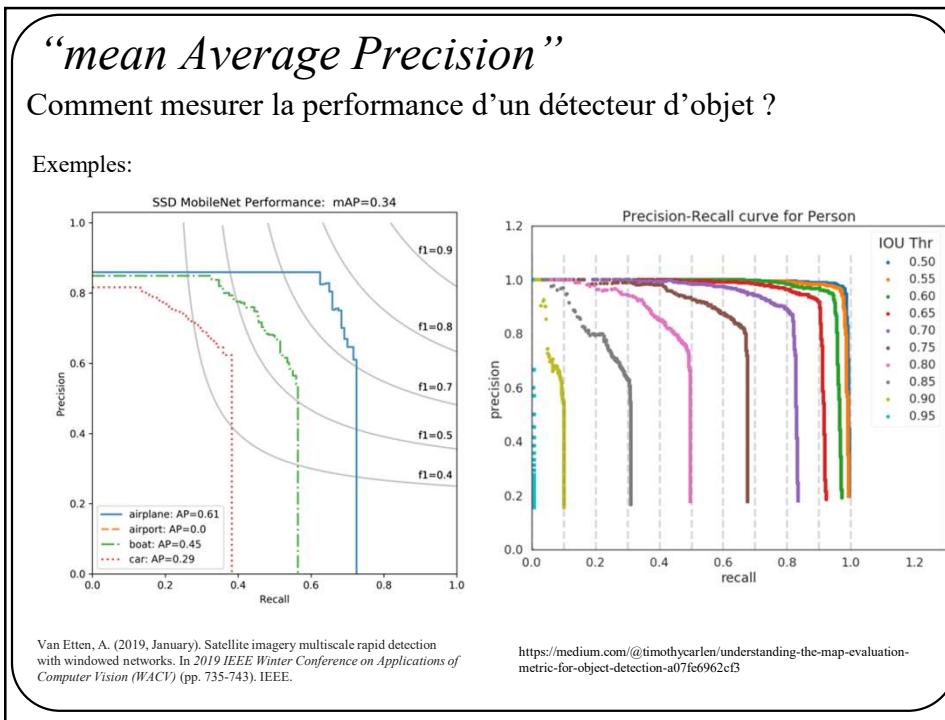


Précision =  $3/5 = 0.6$   
Rappel =  $3/3 = 1.0$



Adapté de [https://web.eecs.umich.edu/~justincj/slides/eecs498/498\\_FA2019\\_lecture15.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf)

132



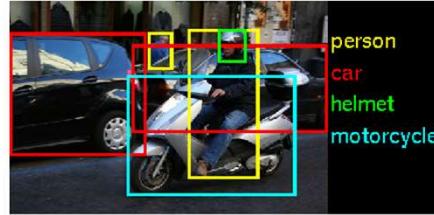
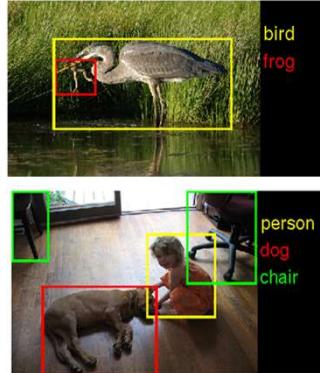
133



134

# Jeux de données populaires

ImageNet (2013+)



## Comparative scale

		PASCAL VOC 2012	ILSVRC 2013
Number of object classes		20	<b>200</b>
Training	Num images	5717	395909
	Num objects	13609	345854
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---

<https://image-net.org/challenges/LSVRC/2013/>

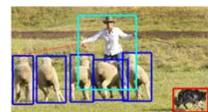
135

# Jeux de données populaires

MS-COCO (2014+) 100 classes



### (a) Image classification



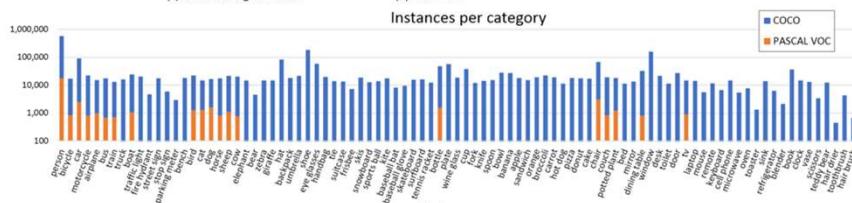
### (b) Object localization



### (c) Semantic segmentation



(d) This wo



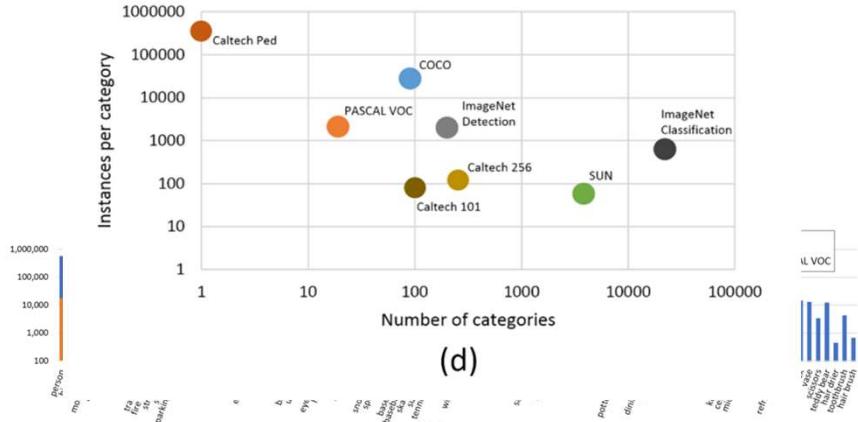
Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

136

# Jeux de données populaires

MS-COCO (2014+) 100 classes

Number of categories vs. number of instances



Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

137

De retour au programme principal

138

138

**R-CNN** [Girshick et al, 2014]

**R-CNN: Regions with CNN features**

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

**VOC 2010 test**

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	soft	train	tv	mAP
DPM v5 [30] <sup>†</sup>	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18] <sup>†</sup>	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	61.2	52.4	53.7

**Table 1: Detection average precision (%) on VOC 2010 test.** R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section C. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. <sup>†</sup>DPM and SegDPM use context rescoring not used by the other methods.

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014      139

139

**R-CNN** [Girshick et al, 2014]

### Problème du R-CNN

3 entraînements séparés (pas d'entraînement bout-en-bout)

- Finetuning du CNN (entropie croisée)
  - pré-entraîné sur ImageNet
  - ré-entraîné sur Pascal VOC
- Entraînement du SVM (Hinge loss)
- Entraînement de la régression (loss L2)

Entraînement lent et complexe

- 84h

Détection lente

- 47secondes / image avec VGG16

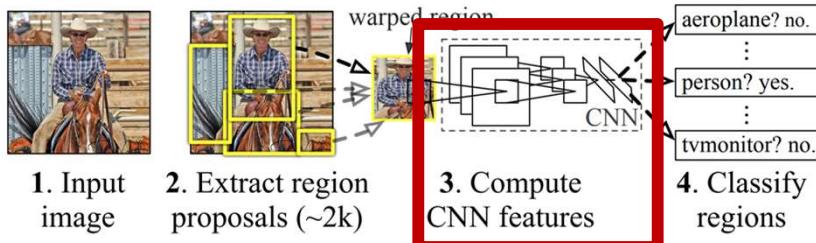
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014      140

140

## R-CNN [Girshick et al, 2014]

### Problème du R-CNN

#### R-CNN: Regions with CNN features



2000 propagations avant par image !

Training time (Hours)

R-CNN 84

Test time (seconds)

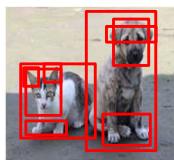


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

141

141

## Fast R-CNN [Girshick, 2015]



640x480x3

1. Localiser des régions

Girshick, "Fast R-CNN", ICCV 2015.

142

**Fast R-CNN** [Girshick, 2015]

« Backend »

The diagram illustrates the Fast R-CNN architecture. It starts with an input image of size 640x480x3, which is processed by a CNN backbone consisting of five convolutional layers. Red arrows point from the input image to the backbone and from the backbone to the classification heads. The classification heads are represented by three 3D blocks, each with dimensions 512x20x15, corresponding to the 15 categories of the PASCAL3D+ dataset.

1. Localiser des régions  
2. Propagation avant de l'image dans 5 blocs convolutionnels

Girshick, "Fast R-CNN", ICCV 2015.

143

**Fast R-CNN** [Girshick, 2015]

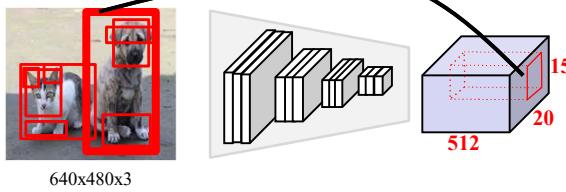
The diagram illustrates the Fast R-CNN architecture. It starts with an input image of size 640x480x3, which is processed by a CNN backbone consisting of five convolutional layers. Red arrows point from the input image to the backbone and from the backbone to the classification heads. The classification heads are represented by three 3D blocks, each with dimensions 512x20x15, corresponding to the 15 categories of the PASCAL3D+ dataset.

1. Localiser des régions  
2. Propagation avant de l'image dans 5 blocs convolutionnels.  
Cartes d'activation **20x15x512**

Girshick, "Fast R-CNN", ICCV 2015.

144

## Fast R-CNN [Girshick, 2015]

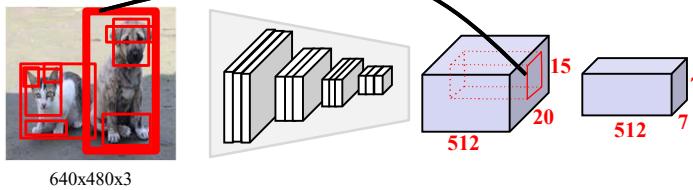


1. Localiser des régions
2. Propagation avant de l'image dans 5 blocs convolutionnels.  
Cartes d'activation **20x15x512**
3. À tour de rôle, projeter chaque région vers les cartes d'activation.

Girshick, "Fast R-CNN", ICCV 2015.

145

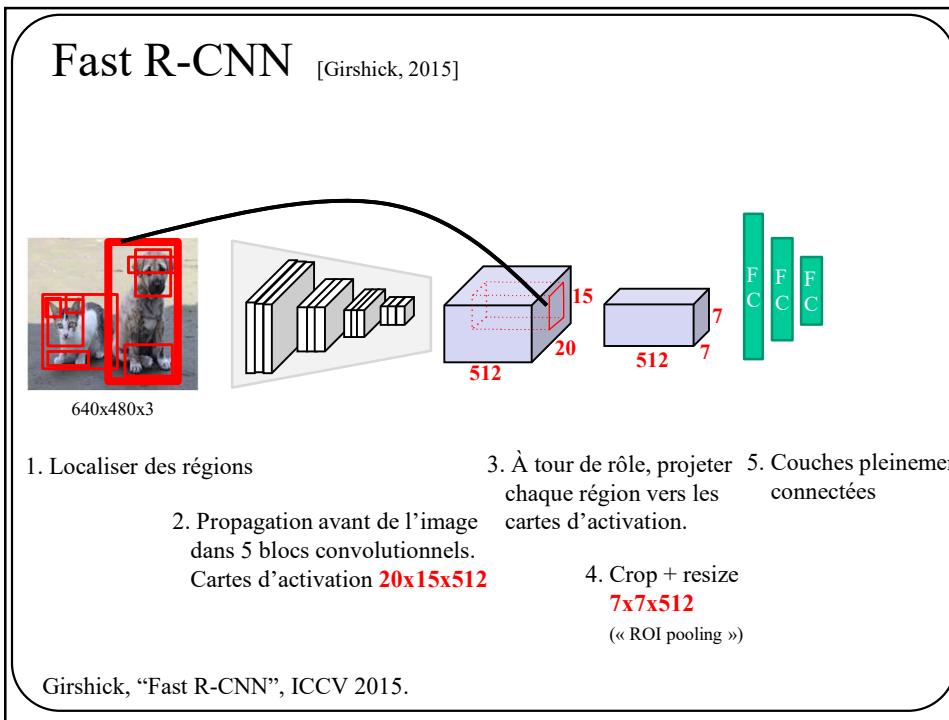
## Fast R-CNN [Girshick, 2015]



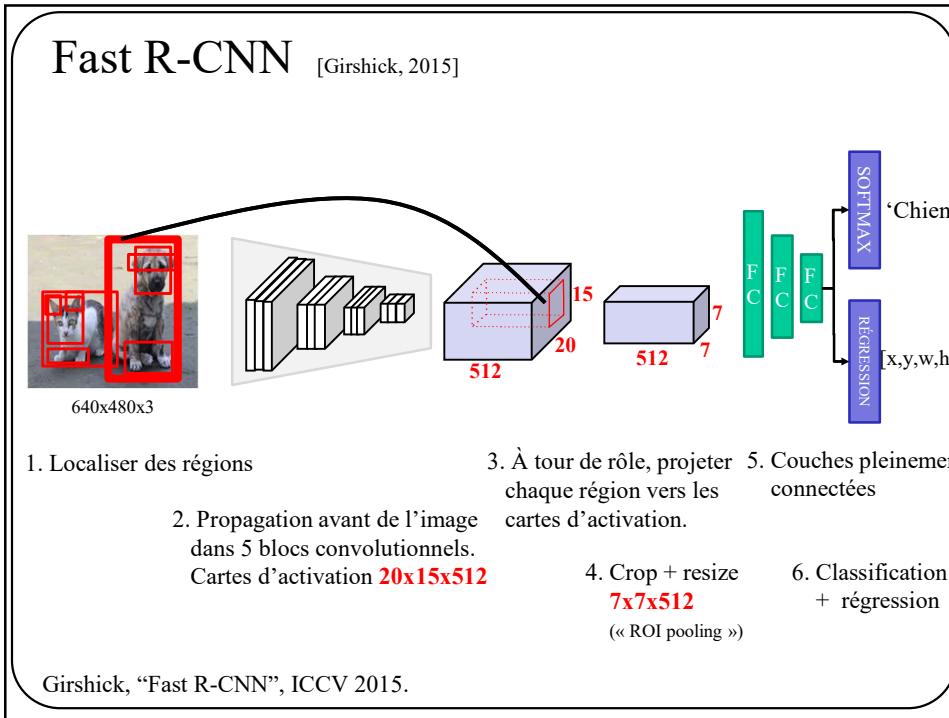
1. Localiser des régions
2. Propagation avant de l'image dans 5 blocs convolutionnels.  
Cartes d'activation **20x15x512**
3. À tour de rôle, projeter chaque région vers les cartes d'activation.
4. Crop + resize  
**7x7x512**  
("ROI pooling")

Girshick, "Fast R-CNN", ICCV 2015.

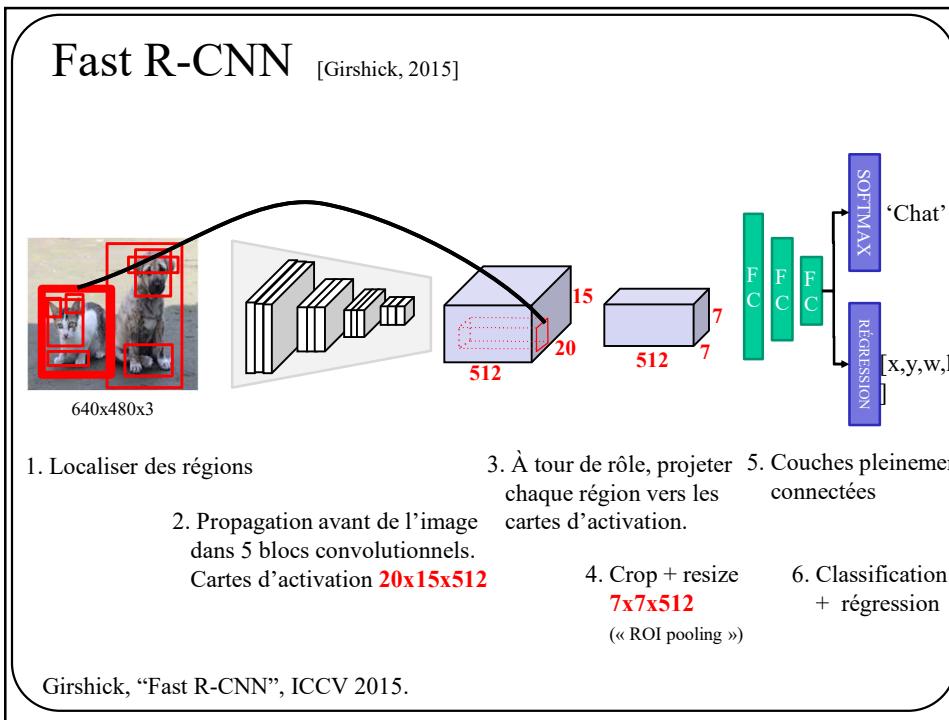
146



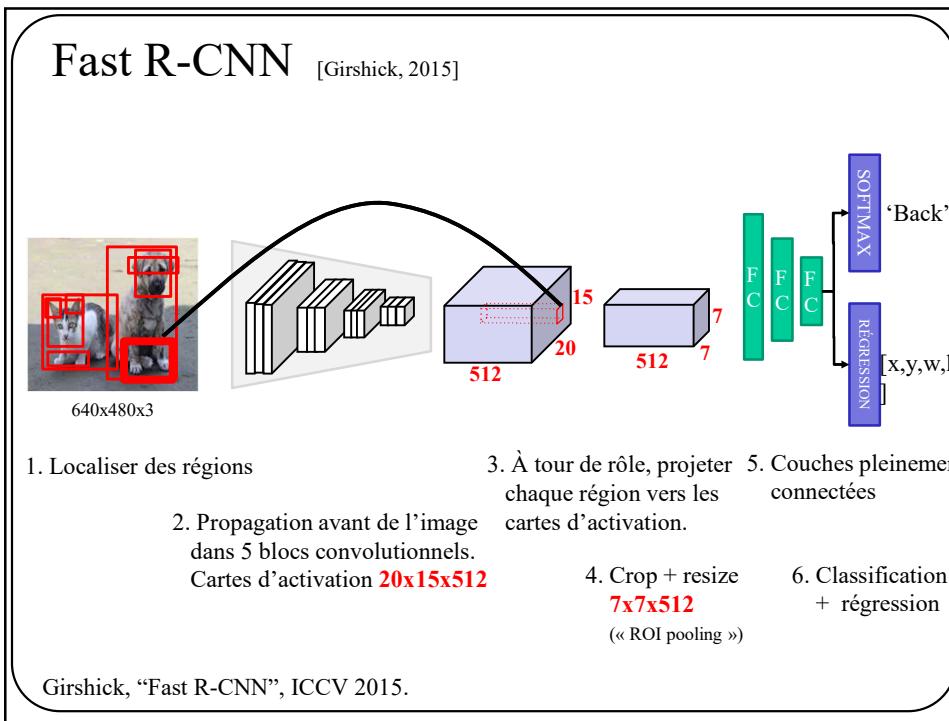
147



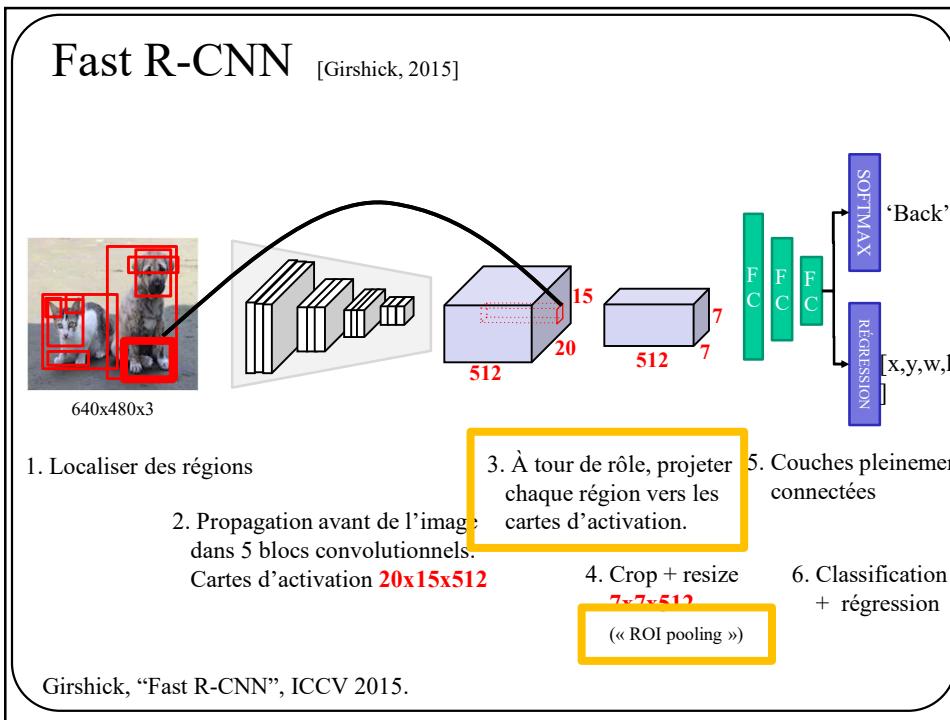
148



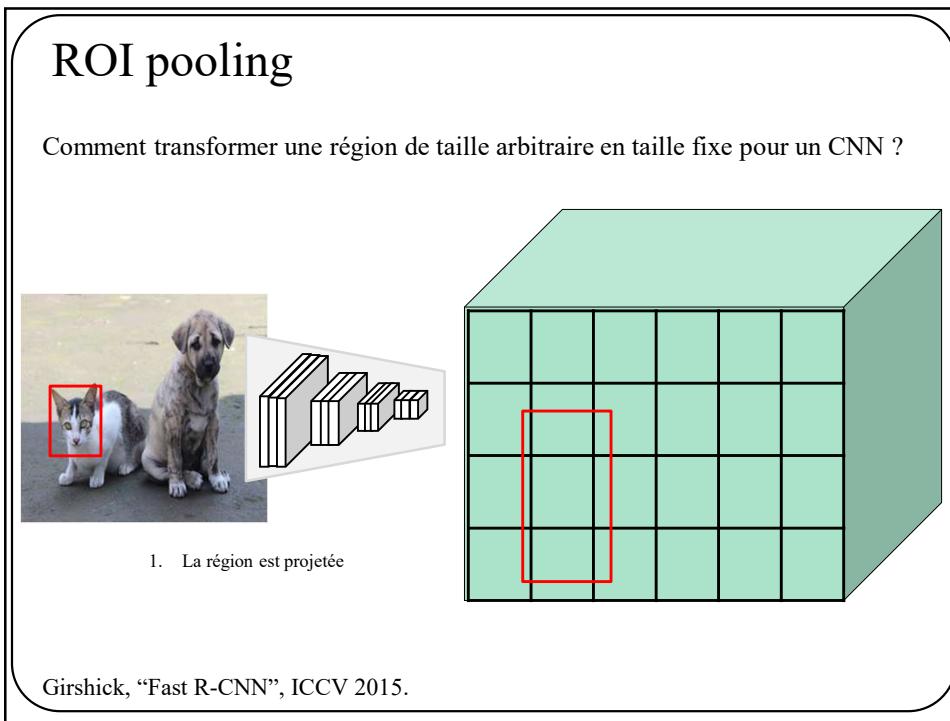
149



150



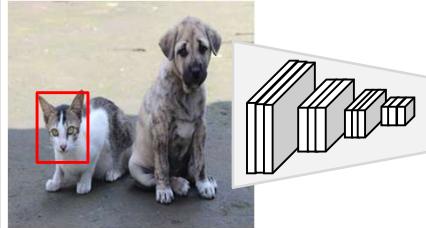
151



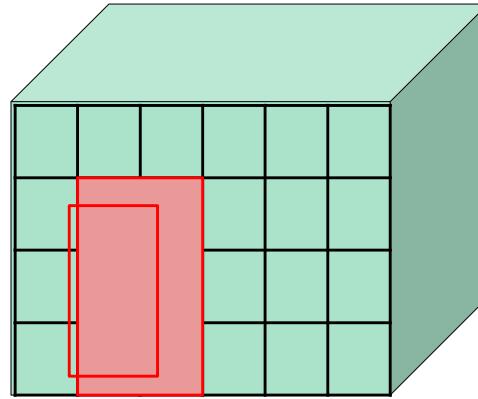
152

## ROI pooling

Comment transformer une région de taille arbitraire en taille fixe pour un CNN ?



1. La région est projetée vers la dimension du *feature map*



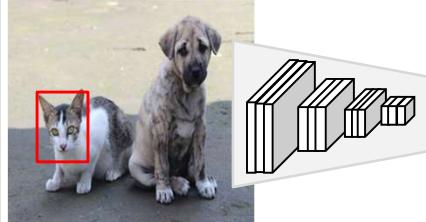
2. La région est ajustée à la grille (*snap*)

Girshick, "Fast R-CNN", ICCV 2015.

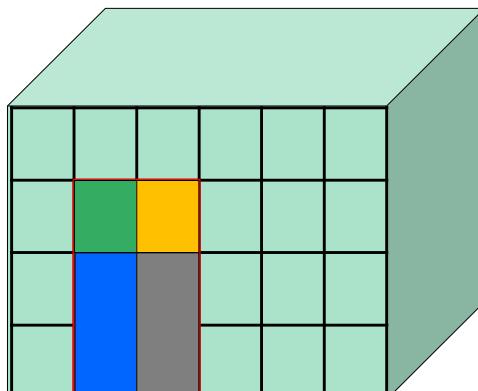
153

## ROI pooling

Comment transformer une région de taille arbitraire en taille fixe pour un CNN ?



1. La région est projetée vers la dimension du *feature map*



2. La région est ajustée à la grille (*snap*)

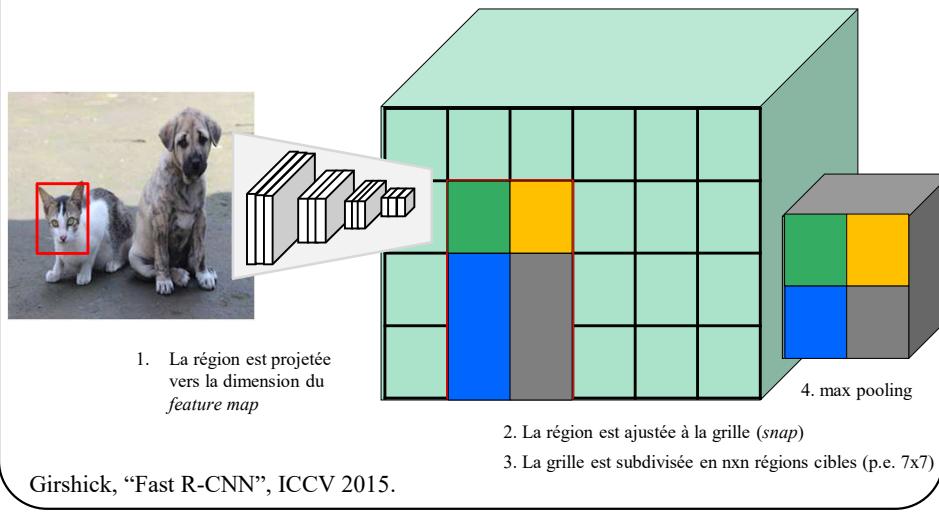
3. La grille est subdivisée en nxn régions cibles (p.e. 7x7)

Girshick, "Fast R-CNN", ICCV 2015.

154

## ROI pooling

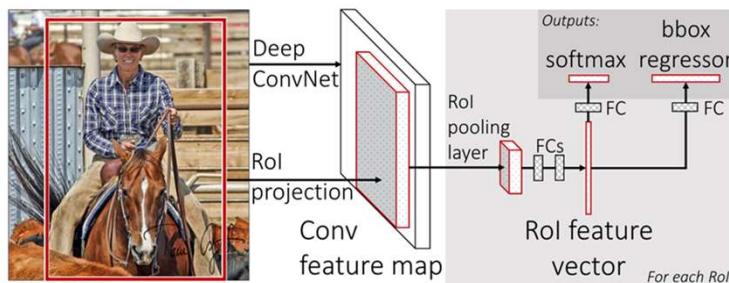
Comment transformer une région de taille arbitraire en taille fixe pour un CNN ?



155

## Fast R-CNN [Girshick, 2015]

Autre illustration (de Girshick):



### Avantages:

- 1 propagation avant par image au lieu de 1 par région
- Entraînement bout-en-bout

### Inconvénients:

- La "*region proposal method*" indépendante du réseau

Girshick, "Fast R-CNN", ICCV 2015.

156

## Fast R-CNN [Girshick et al, 2015]

Tiré de l'article

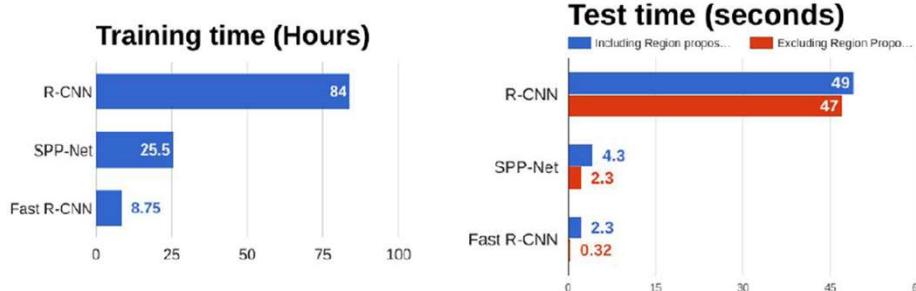
method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	<b>82.3</b>	<b>78.4</b>	<b>70.8</b>	<b>52.3</b>	38.7	<b>77.8</b>	<b>71.6</b>	<b>89.3</b>	<b>44.2</b>	<b>73.0</b>	<b>55.0</b>	<b>87.5</b>	<b>80.5</b>	<b>80.8</b>	<b>72.0</b>	35.1	<b>68.3</b>	<b>65.7</b>	<b>80.4</b>	<b>64.2</b>	<b>68.4</b>

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS\_NIN\_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2. **Unk.**: unknown.

Girshick, “Fast R-CNN”, ICCV 2015.

157

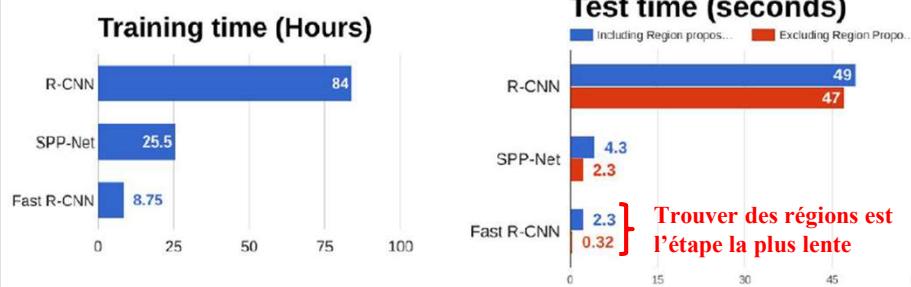
## Fast R-CNN [Girshick et al, 2015]



Girshick, “Fast R-CNN”, ICCV 2015.

158

## Fast R-CNN [Girshick et al, 2015]



CNN: exécutés sur GPU => rapide !

*Region proposal*: exécuté sur CPU => lent :(

Girshick, "Fast R-CNN", ICCV 2015.

159

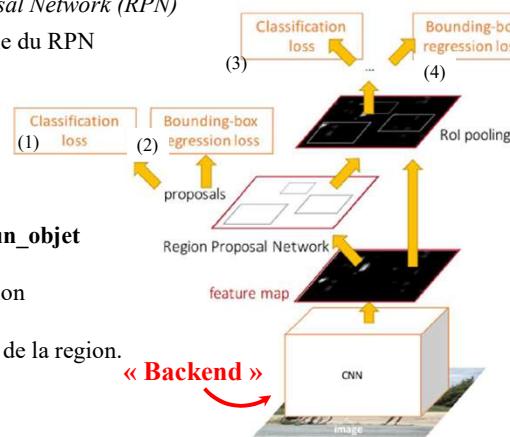
## Faster R-CNN [Ren et al, 2015]

### Idée:

- Trouver les régions à même les cartes d'activation avec un *Region Proposal Network (RPN)*
- Chaque boîte possible est une sortie du RPN

4 loss utilisées simultanément:

1. RPN : Classification objet vs pas\_un\_objet
2. RPN : régression [x,y,w,h]
3. Classification de l'objet dans la région (chien,chat...)
4. Régression des coordonnées finales de la région.

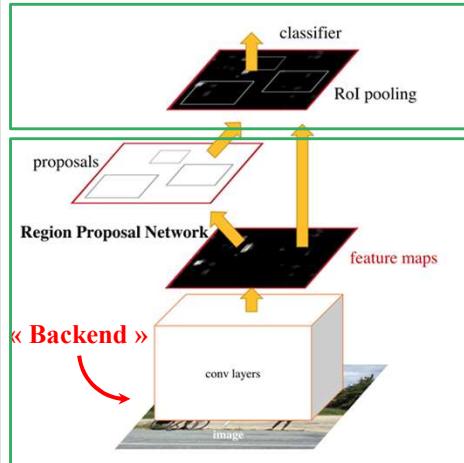


Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

160

160

## Faster R-CNN [Ren et al, 2015]



- Deux étapes:
- Classifier les régions
- Proposer les régions

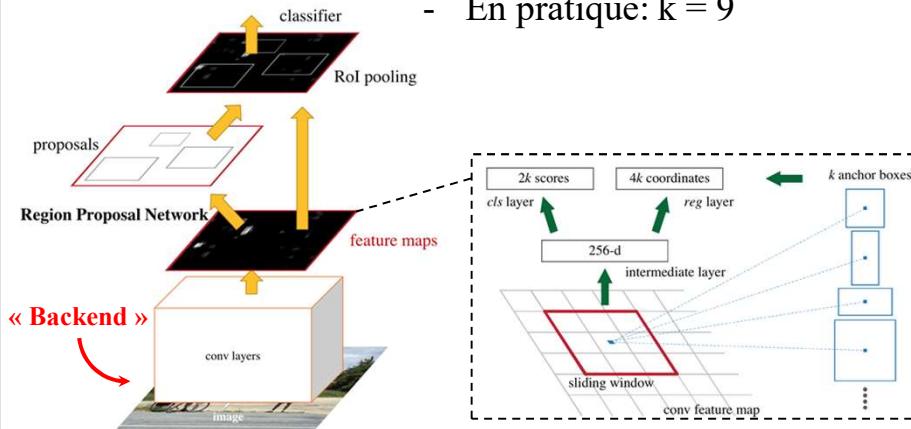
Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

161

161

## Faster R-CNN [Ren et al, 2015]

- Utilisation de “anchors”
- Boîtes de forme prédéterminée
- En pratique:  $k = 9$



Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

162

162

## Faster R-CNN [Ren et al, 2015]

R-CNN Test-Time Speed

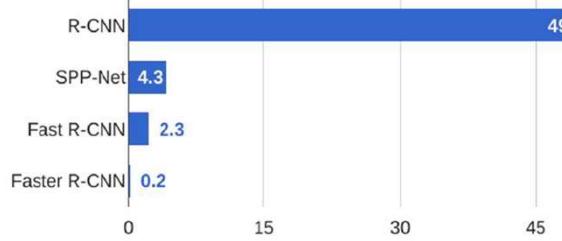


Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. “Region-wise” includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	10	47	198	5 fps
ZF	RPN + Fast R-CNN	31	3	25	59	17 fps

Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

163

163

## Faster R-CNN [Ren et al, 2015]

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

method	# box	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
SS	2000	07+12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
RPN	300	12	67.0	82.3	76.4	71.0	48.4	45.2	72.1	72.3	87.3	42.2	73.7	50.0	86.8	78.7	78.4	77.4	34.5	70.1	57.1	77.1	58.9
RPN	300	07+12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
RPN	300	COCO+07+12	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2

### Avantages:

- 1 propagation avant par image au lieu de 1 par région
- Entraînement bout-en-bout
- *Region proposal method* apprise !
- Inférence très rapide

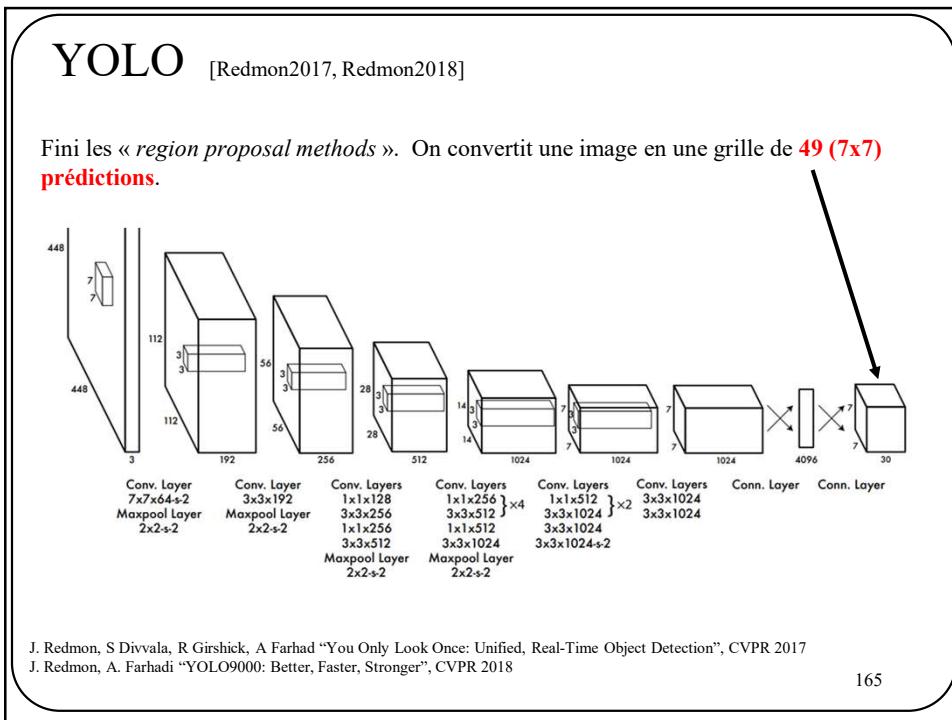
### Inconvénients:

- Architecture complexe avec plusieurs *moving parts*
- Entraînement pas vraiment bout en bout (anchors)
- Détection en plusieurs étapes

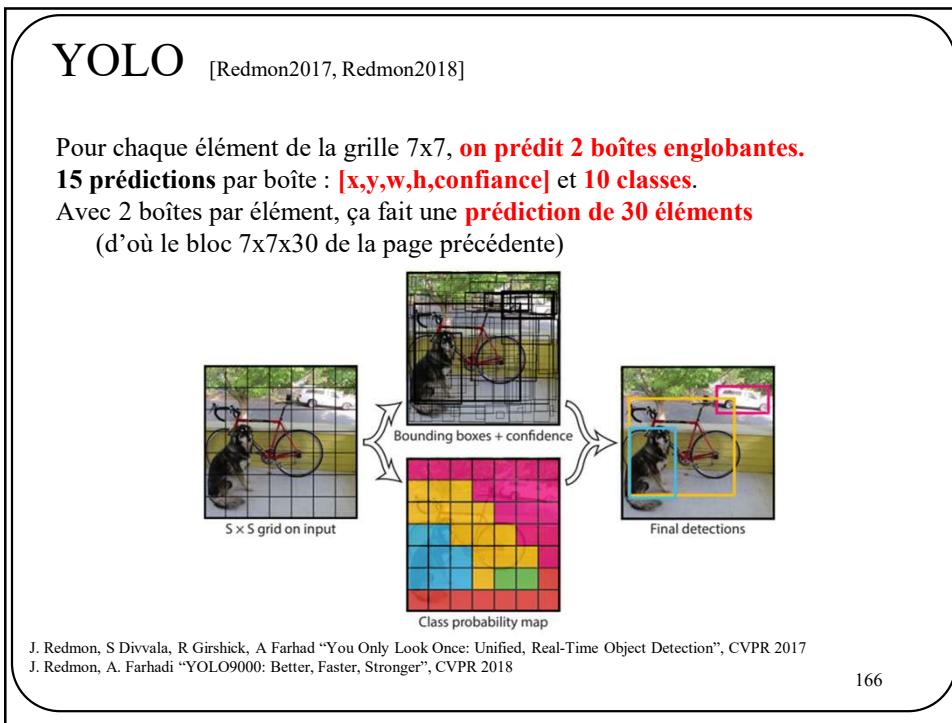
Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

164

164



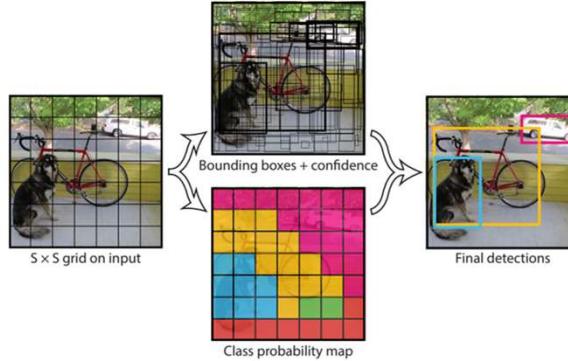
165



166

## YOLO [Redmon2017, Redmon2018]

Ainsi, YOLO prédit TOUJOURS **49x2=98 objets possibles** chacune avec un **indice de confiance**.



J. Redmon, S Divvala, R Girshick, A Farhad “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2017  
J. Redmon, A. Farhad “YOLO9000: Better, Faster, Stronger”, CVPR 2018

167

167

## YOLO [Redmon2017, Redmon2018]

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR.CNN, MORE DATA [11]	73.9	<b>85.5</b>	<b>82.9</b>	<b>76.6</b>	<b>57.8</b>	<b>62.7</b>	<b>79.4</b>	<b>77.2</b>	<b>86.6</b>	<b>55.0</b>	<b>79.1</b>	<b>62.2</b>	<b>87.0</b>	<b>83.4</b>	<b>84.7</b>	<b>78.9</b>	<b>45.3</b>	<b>73.4</b>	<b>65.8</b>	<b>80.3</b>	<b>74.0</b>
HyperNet.VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	<b>79.8</b>	87.7	49.6	74.9	52.1	86.0	81.7	83.3	<b>81.8</b>	<b>48.6</b>	<b>73.5</b>	59.4	79.9	65.7
HyperNet.SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
<b>Fast R-CNN + YOLO</b>	<b>70.7</b>	83.4	78.5	73.5	55.8	43.4	79.1	<b>73.1</b>	<b>89.4</b>	49.4	75.5	57.0	<b>87.5</b>	80.9	81.0	74.7	41.8	71.5	68.5	<b>82.1</b>	<b>67.2</b>
MR.CNN.S,CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	86.4	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.6	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP.ENS,COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	<b>68.8</b>	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	<b>87.5</b>	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH,FGS,STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS,NIN,C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS,NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	72.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
<b>YOLO</b>	<b>57.9</b>	<b>77.0</b>	<b>67.2</b>	<b>57.7</b>	<b>38.3</b>	<b>22.7</b>	<b>68.3</b>	<b>55.9</b>	<b>81.4</b>	<b>36.2</b>	<b>60.8</b>	<b>48.5</b>	<b>77.2</b>	<b>72.3</b>	<b>71.3</b>	<b>63.5</b>	<b>28.9</b>	<b>52.2</b>	<b>54.8</b>	<b>73.9</b>	<b>50.8</b>
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.0	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

**Table 3: PASCAL VOC 2012 Leaderboard.** YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

J. Redmon, S Divvala, R Girshick, A Farhad “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2017  
J. Redmon, A. Farhad “YOLO9000: Better, Faster, Stronger”, CVPR 2018

168

168

# YOLO

[Redmon2017, Redmon2018]

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	
MR.CNN.MORE.DATA [11]	73.9	<b>85.5</b>	<b>82.9</b>	<b>76.6</b>	<b>57.8</b>	<b>62.7</b>	<b>79.4</b>	77.2	86.6	<b>55.0</b>	<b>79.1</b>	<b>62.2</b>	87.0	<b>83.4</b>	<b>84.7</b>	78.9	45.3	73.4	65.8	80.3	74.0	
HyperNet.VGG	71.4	84.2	79.5	72.6	55.6	62.7	78.7	<b>70.8</b>	87.7	10.6	74.0	52.1	86.0	81.7	82.3	<b>81.8</b>	<b>48.6</b>	<b>73.5</b>	59.4	79.9	65.7	
HyperNet.SP	71.3	84.2	79.5	72.6	55.6	62.7	78.7	70.8	87.7	10.6	74.0	52.1	86.0	81.7	82.3	81.6	48.4	73.2	59.3	79.7	65.6	
<b>Fast R-CNN + YOLO</b>	70.7	83.4	77.9	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1	
MR.CNN.S.CNN [11]	70.7	83.4	77.9	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1	
Faster R-CNN [28]	70.4	83.4	77.9	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1	
DEEP.ENS.COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	<b>68.8</b>	75.9	71.4	
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1	
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	<b>87.5</b>	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2	
UMICH.FGS.STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2	
NUS.NIN.C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3	
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	
NUS.NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7	
R-CNN VGG BB [13]	62.4	79.4	73.7	64.0	41.2	44.0	65.0	67.4	84.6	38.2	67.3	46.7	82.0	74.0	76.4	65.2	35.6	65.4	54.2	67.4	60.3	
R-CNN VGG [13]	59.2	76.																				
<b>YOLO</b>	57.9	77.																				
Feature Edit [33]	56.3	74.																				
R-CNN BB [13]	53.3	71.																				
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6	

**Table 3: PASCAL VOC 2012 Leaderboard.** YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

J. Redmon, S Divvala, R Girshick, A Farhad “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2017

169

## YOLOv2-3

v2:

- Batch norm
- Images de plus haute résolution
- Anchors
- et autres

v3:

- Réseau plus profond
- Détection à plusieurs résolutions
- Plus de boîtes par élément de grille

J. Redmon, S Divvala, R Girshick, A Farhad “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2017  
J. Redmon, A. Farhad “YOLO9000: Better, Faster, Stronger”, CVPR 2018  
Redmon, J., & Farhad, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

170

## YOLOv2-v3

Yolo v2 et James Bond

<https://www.youtube.com/watch?v=VOC3huqHrss>

<https://pjreddie.com>

(Site web du premier auteur)

J. Redmon, S Divvala, R Girshick, A Farhad "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2017  
 J. Redmon, A. Farhadi "YOLO9000: Better, Faster, Stronger", CVPR 2018  
 J. Redmon, & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

171

171

## YOLOv2-v3

À ce jour,  
nous sommes  
à YOLOv9!

<https://viso.ai/computer-vision/yolov9/>

J. Redmon, S Divvala, R Girshick, A Farhad "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2017  
 J. Redmon, A. Farhadi "YOLO9000: Better, Faster, Stronger", CVPR 2018  
 J. Redmon, & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

172

172

## YOLO v9

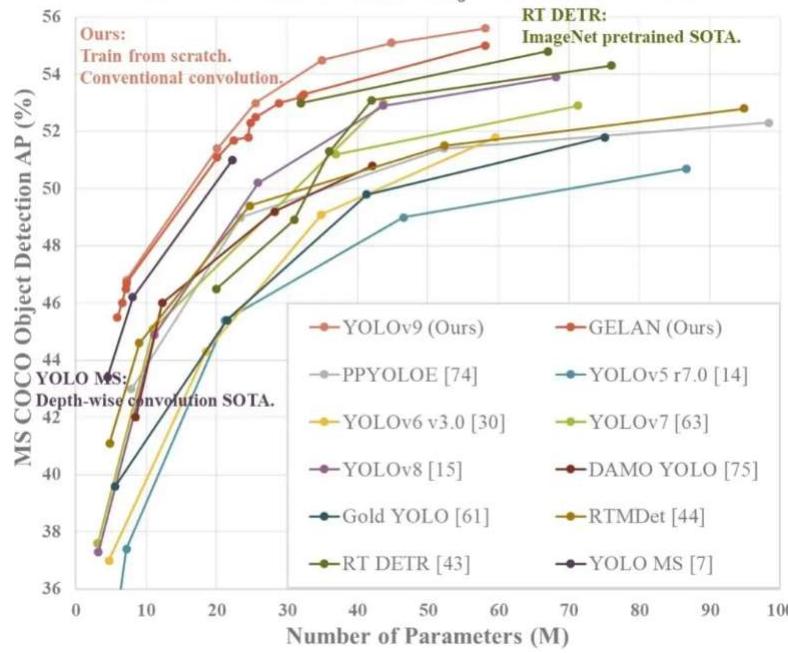


<https://viso.ai/computer-vision/yolov9/>

173

173

Performance on MS COCO Object Detection Dataset



174

## SSD (single shot detector) [Liu et al. 2016]

Tout comme YOLO:

- Pas de « *region proposal method* » pour SSD
- Prédiction d'un **nombre fixe** de boîtes englobantes.
  - 98 pour YOLO
  - **8732 pour SSD300**
  - **24564 pour SSD512 (!)**
- Prédit 25 éléments : **[x,y,w,h,confiance,c<sub>1</sub>,..., c<sub>20</sub>]**
- Élimine les boîtes avec une confiance faible

W Liu, D Anguelov, D Erhan, C Szegedy, S Reed, C-Y Fu, A.C. Berg "SSD: Single Shot MultiBox Detector", ECCV 2016

175

## SSD300 (single shot detector) [Liu et al. 2016]

Utilise les **10 premières couches de VGG16** comme *backend*

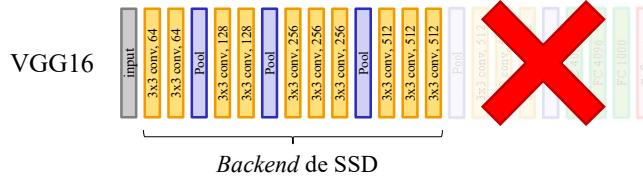


176

176

## SSD300 (single shot detector) [Liu et al. 2016]

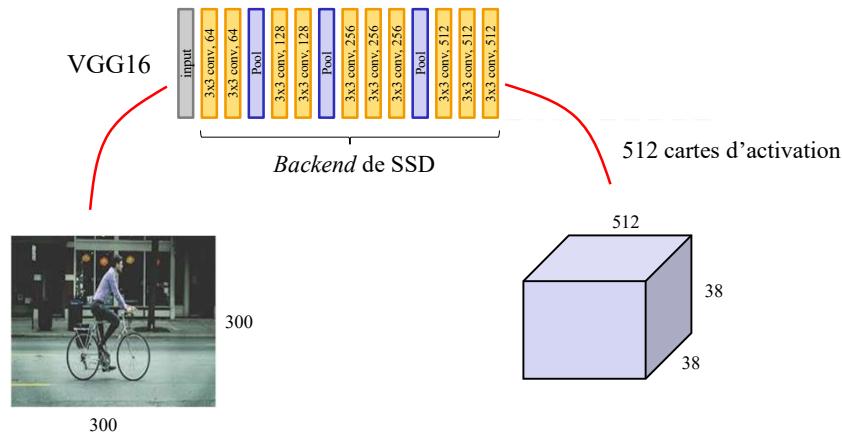
Utilise les **10 premières couches de VGG16** comme *backend*



177

## SSD300 (single shot detector) [Liu et al. 2016]

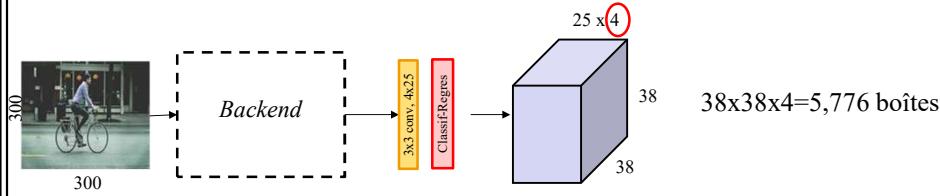
Utilise les **10 premières couches de VGG16** comme *backend*



178

## SSD300 (single shot detector) [Liu et al. 2016]

Utilise les **10 premières couches de VGG16** comme *backend*



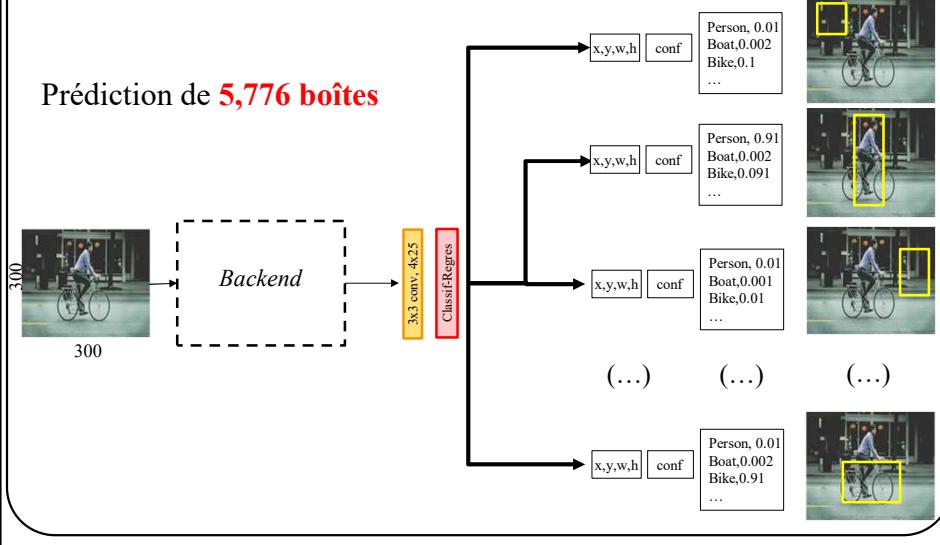
4 prédictions par pixel

1 prédition= 25 éléments **[x,y,w,h,confiance,c<sub>1</sub>,..., c<sub>20</sub>]**

179

## SSD300 (single shot detector) [Liu et al. 2016]

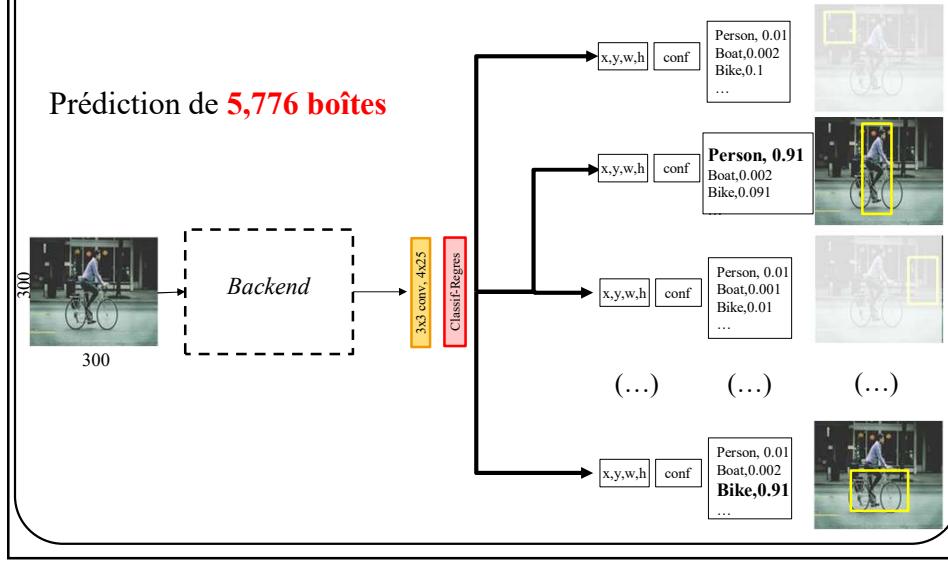
Prédiction de **5,776 boîtes**



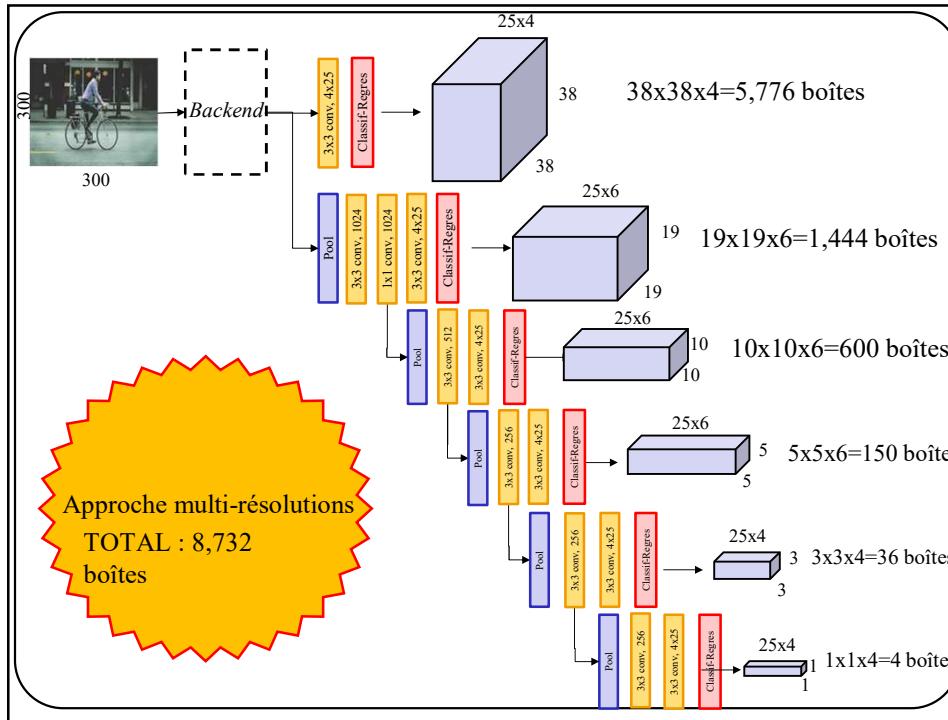
180

## SSD300 (single shot detector) [Liu et al. 2016]

On ne garde que les boîtes **sans chevauchement** et avec une **confiance élevée**



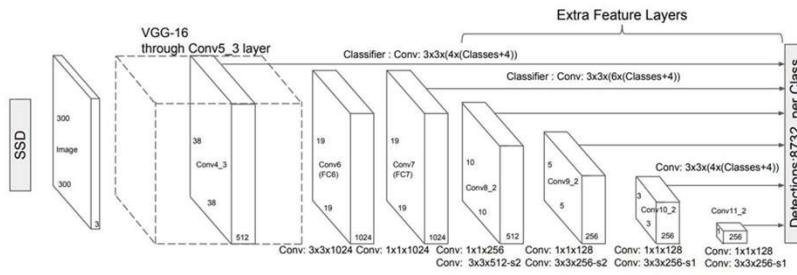
181



182

## SSD (single shot detector) [Liu et al. 2016]

Autre illustration tirée de l'article (plus compacte)



W Liu, D Anguelov, D Erhan, C Szegedy, S Reed, C-Y Fu, AC. Berg "SSD: Single Shot MultiBox Detector", ECCV 2016

183

## SSD (single shot detector) [Liu et al. 2016]

Method	mAP	FPS
Faster R-CNN (VGG16)	73.2	7
Fast YOLO	52.7	155
YOLO (VGG16)	66.4	21
SSD300	74.3	46
SSD512	76.8	19
SSD300	74.3	59
SSD512	76.8	22

**mAP** : mean average precision

W Liu, D Anguelov, D Erhan, C Szegedy, S Reed, C-Y Fu, AC. Berg "SSD: Single Shot MultiBox Detector", ECCV 2016

184

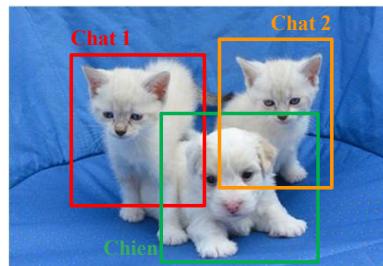
## SSD (single shot detector) [Liu et al. 2016]

[https://medium.com/@jonathan\\_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06](https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06)



185

## Segmentation par instance



Localisation



Segmentation  
par instance

186

186

## Mask R-CNN [He et al., 2017]

Idée de base:



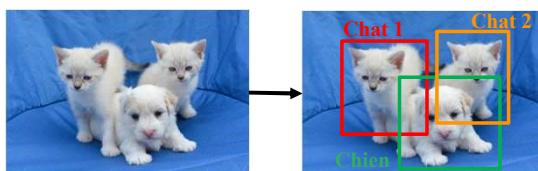
Images: He et al. "Mask R-CNN." ICCV, 2017

187

## Mask R-CNN [He et al., 2017]

Idée de base:

1. Localisation

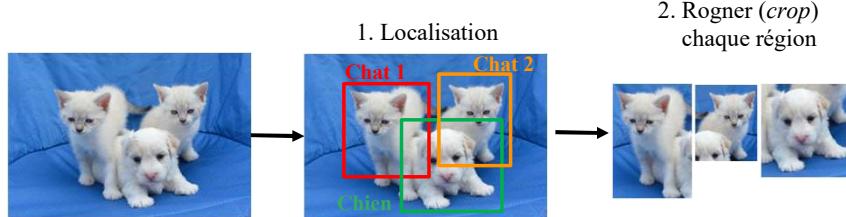


Images: He et al. "Mask R-CNN." ICCV, 2017

188

## Mask R-CNN [He et al., 2017]

Idée de base:

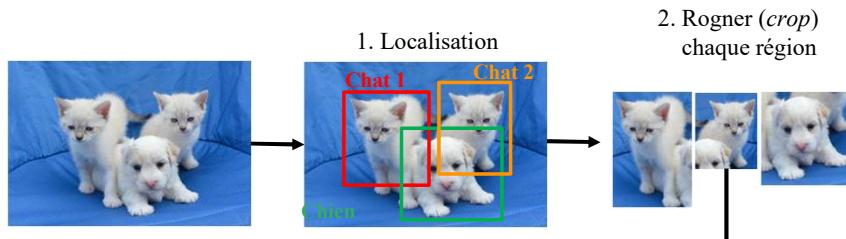


Images: He et al. "Mask R-CNN." ICCV, 2017

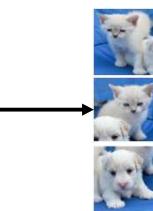
189

## Mask R-CNN [He et al., 2017]

Idée de base:



3. Redimension

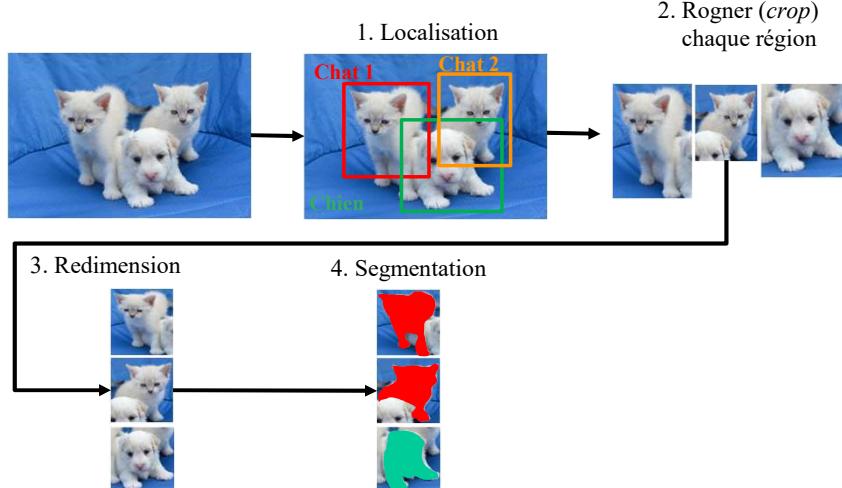


Images: He et al. "Mask R-CNN." ICCV, 2017

190

## Mask R-CNN [He et al., 2017]

Idée de base:

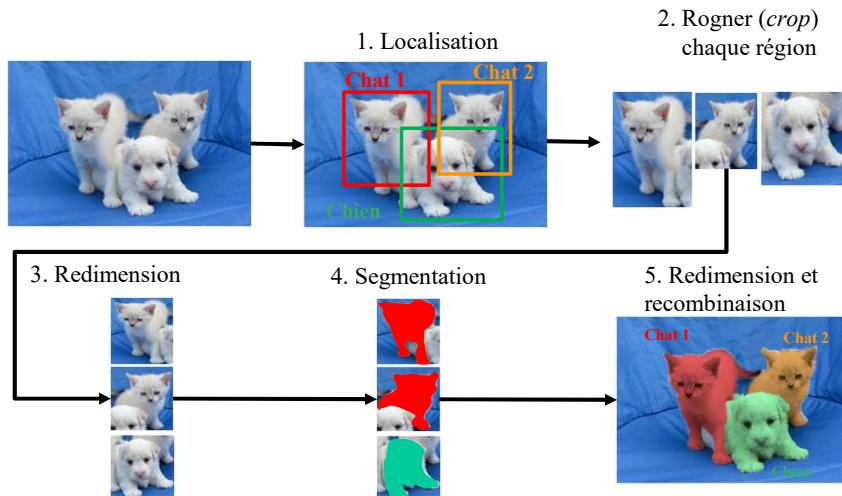


Images: He et al. "Mask R-CNN." ICCV, 2017

191

## Mask R-CNN [He et al., 2017]

Idée de base:



Images: He et al. "Mask R-CNN." ICCV, 2017

192

Mask R-CNN  
=

CNN localization + CNN segmentation

193

193

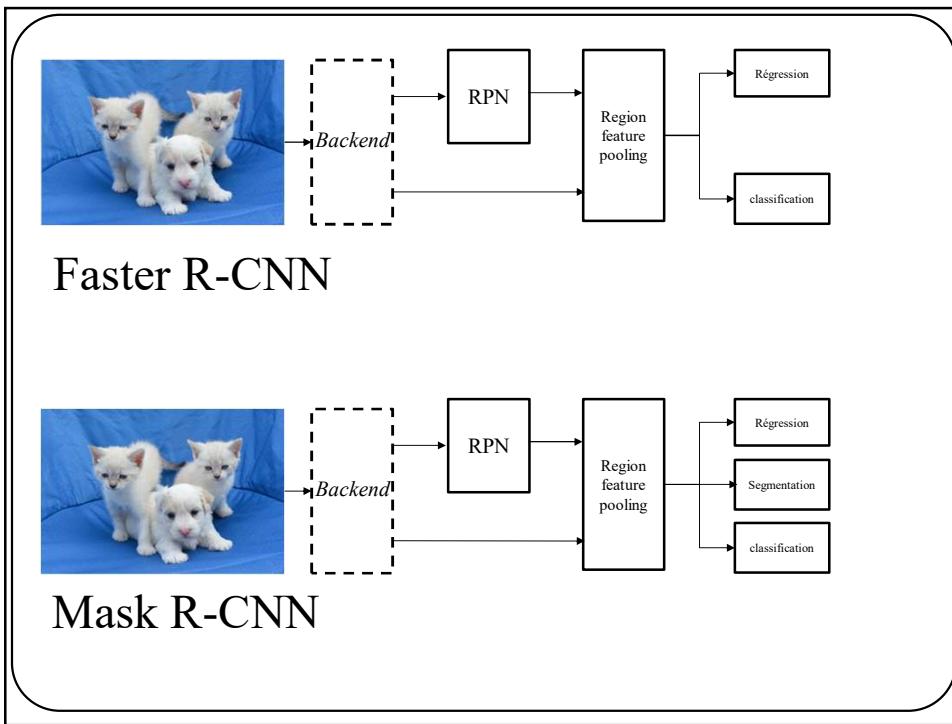
Mask R-CNN  
=

Faster R-CNN (avec backend ResNet) + CNN segmentation

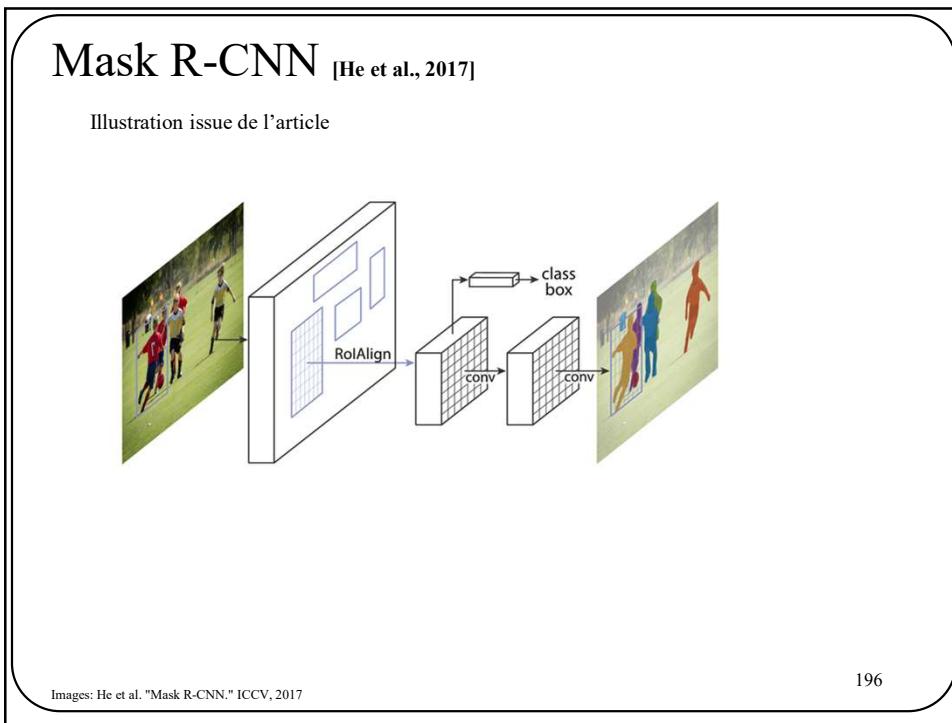
(Config de l'article d'origine. D'autres versions de Mask R-CNN utilisent d'autres backends.)

194

194



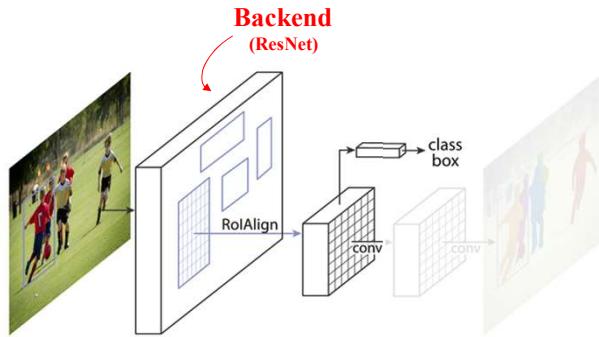
195



196

## Mask R-CNN [He et al., 2017]

Illustration issue de l'article



## Faster R-CNN

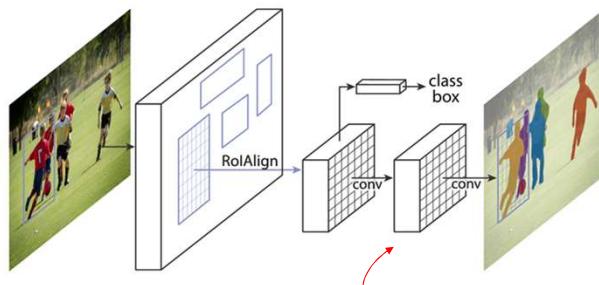
Images: He et al. "Mask R-CNN." ICCV, 2017

197

197

## Mask R-CNN [He et al., 2017]

Illustration issue de l'article



**Séquence de convolutions  
Pour la segmentation**

Images: He et al. "Mask R-CNN." ICCV, 2017

198

198

## Mask R-CNN [He et al., 2017]



Figure 5. 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).

Images: He et al. "Mask R-CNN." ICCV, 2017

199

199

## Mask R-CNN [He et al., 2017]

<https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/>



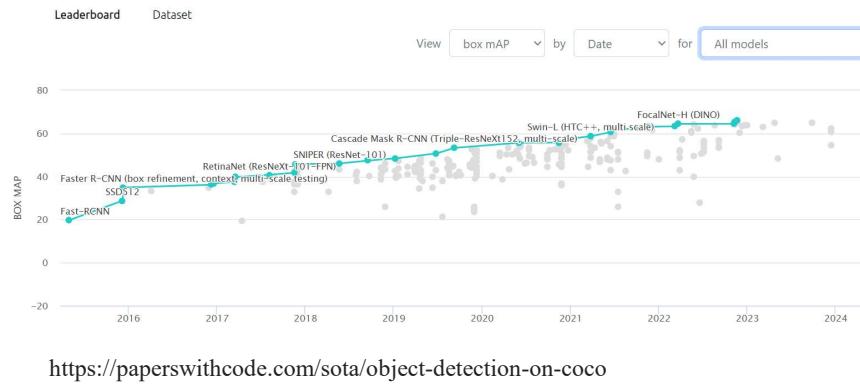
200

### En résumé

### Bons survols:

- Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R. (2019). A survey of deep learning-based object detection. *IEEE access*, 7, 128837-128868.
  - Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310-7311).

## Object Detection on COCO test-dev



201

### En résumé

La détection d'objets et segmentation d'instances est complexe avec beaucoup d'astuces pour entraîner et prédire.

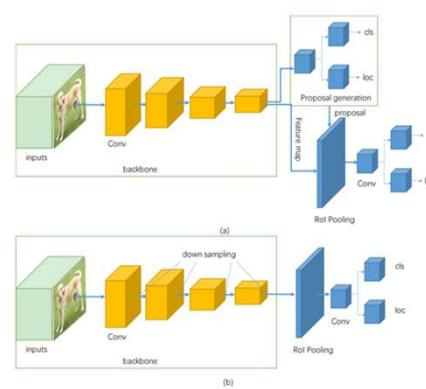
Deux grandes familles de méthodes:

*Two-stage detectors (e.g. fasterRCNN)*

- Plus complexes
  - Plus lents
  - Plus performants

### *Single Stage detectors (e.g. yolo, SSD)*

- Plus simples
  - Plus rapides
  - Moins performants



Bibliothèque de détection d'objets/segmentation en Pytorch

Bibliothèque de détection d'objets/segmentation en temps réel : <https://github.com/facebookresearch/detectron2>

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R. (2019). A survey of deep learning-based object detection. *IEEE access*, 7, 128837–128868.

202