

UNIVERSITE ABDELMALEK ESSAADI  
FACULTE DES SCIENCES  
TETOUAN

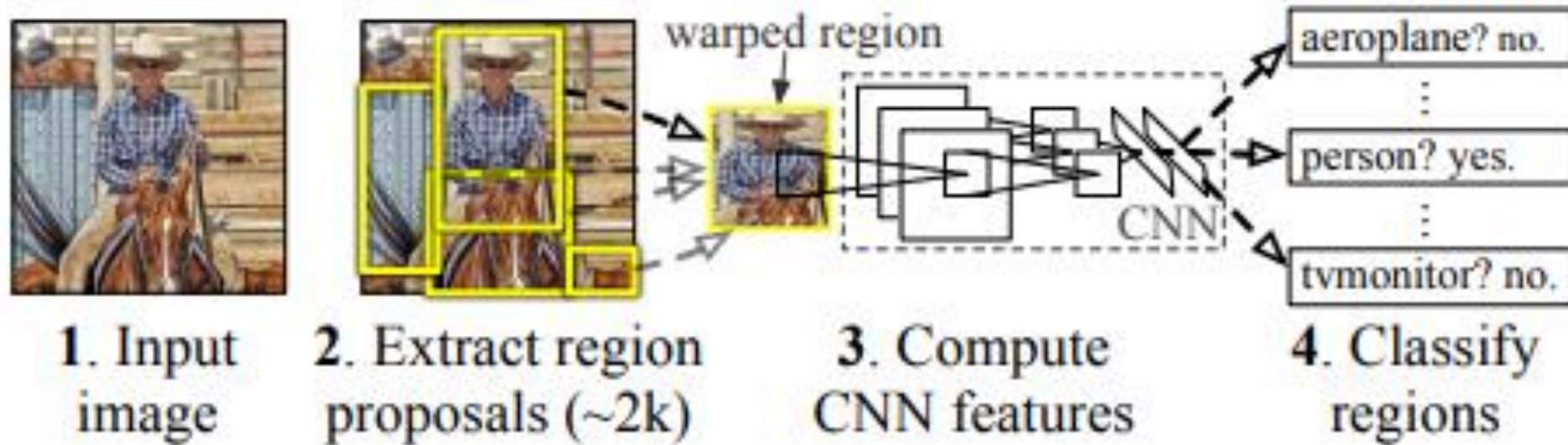


جامعة عبد المالك السعدي  
كلية العلوم  
تطوان

# Apprentissage profond pour la vision par ordinateur

Presented by Pr. Abdellatif El Ouissari

# Object Detection with Deep Learning

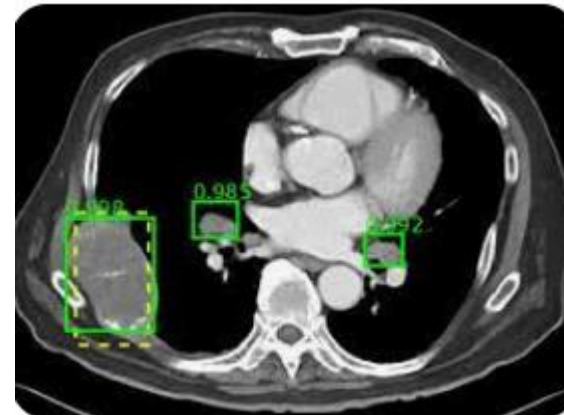


# Object Detection with Deep Learning

Dans le domaine de la vision par ordinateur, les réseaux neuronaux convolutionnels profonds (CNN) ont démontré d'excellentes performances.

Le traitement vidéo, la détection d'objets, la segmentation et la classification d'images, la reconnaissance vocale et le traitement du langage naturel sont quelques-uns des domaines d'application des CNN.

La détection d'objets est la tâche la plus cruciale et la plus difficile de la vision par ordinateur. Elle a de nombreuses applications dans les domaines de la sécurité, de l'armée, des transports et des sciences médicales.



# Object Detection with Deep Learning

Dans ce chapitre, la détection d'objets et ses différents aspects ont été couverts en détail. Avec l'évolution progressive des algorithmes d'apprentissage profond pour la détection d'objets, une amélioration significative des performances des modèles de détection d'objets a été observée.



# Object Detection with Deep Learning

Toutefois, cela ne signifie pas que les méthodes conventionnelles de détection d'objets, qui ont évolué pendant des dizaines d'années avant l'émergence de l'apprentissage profond, sont devenues obsolètes. Dans certains cas, les méthodes conventionnelles avec des caractéristiques globales constituent un choix supérieur.

Ce chapitre de synthèse commence par un aperçu rapide de la détection d'objets, suivi des cadres de détection d'objets, du réseau neuronal convolutionnel de base et d'un aperçu des ensembles de données courants ainsi que des mesures d'évaluation.

Les problèmes et les applications de la détection d'objets sont également étudiés en détail. Certains défis futurs de la recherche dans la conception de réseaux neuronaux profonds sont discutés

Enfin, les performances des modèles de détection d'objets sur les ensembles de données PASCAL VOC et MS COCO sont comparées et des conclusions sont tirées.

# PASCAL VOC Dataset

Les ensembles de données des défis VOC sont disponibles via les liens ci-dessous, et l'évaluation de nouvelles méthodes sur ces ensembles de données peut être réalisée via le serveur d'évaluation PASCAL VOC. Le serveur d'évaluation restera actif même si les défis sont maintenant terminés.



Details of each of the challenges can be found on the corresponding challenge page:

- [The VOC2012 Challenge](#)
- [The VOC2011 Challenge](#)
- [The VOC2010 Challenge](#)
- [The VOC2009 Challenge](#)
- [The VOC2008 Challenge](#)
- [The VOC2007 Challenge](#)
- [The VOC2006 Challenge](#)
- [The VOC2005 Challenge](#)



## COCO dataset.

La liste des étiquettes de classes pour l'ensemble de données COCO. Pour YOLOv3, cette base contient généralement 80 noms de classes, tels que :

skis  
snowboard  
sports ball  
kite  
baseball bat  
baseball glove  
skateboard  
surfboard  
tennis racket  
bottle  
wine glass  
cup  
fork  
knife  
Spoon  
bed  
diningtable

toilet  
tvmonitor  
laptop  
mouse  
remote  
keyboard  
cell phone  
microwave  
oven  
toaster  
sink  
refrigerator  
book  
clock  
vase  
scissors

teddy bear  
hair drier  
toothbrush  
bowl  
banana  
apple  
sandwich  
orange  
broccoli  
carrot  
hot dog  
pizza  
donut  
cake  
chair  
sofa  
pottedplant

person  
bicycle  
car  
motorbike  
aeroplane  
bus  
Train  
truck  
boat  
traffic light  
fire hydrant  
stop sign  
parking meter  
bench  
bird  
cat  
dog  
horse  
sheep  
cow  
elephant  
bear  
zebra  
giraffe  
backpack  
umbrella  
handbag  
tie  
suitcase  
frisbee



# Détection

La détection d'objets est une tâche fondamentale en vision par ordinateur et en compréhension d'images, visant à identifier et localiser les objets d'intérêt dans une image tout en leur attribuant des étiquettes de classe correspondantes.

Les méthodes traditionnelles, qui reposaient sur des caractéristiques conçues manuellement et des modèles peu profonds, peinaient à traiter des données visuelles complexes et offraient des performances limitées.

Ces méthodes combinaient des caractéristiques de bas niveau avec des informations contextuelles, mais ne parvenaient pas à capturer les sémantiques de haut niveau.

# Détection

L'apprentissage profond, en particulier les réseaux de neurones convolutifs (CNN), a permis de surmonter ces limites en apprenant automatiquement des caractéristiques riches et hiérarchiques directement à partir des données.

Ces caractéristiques incluent des représentations sémantiques et de haut niveau essentielles à une détection d'objets précise.

# Détection

Ce chapitre passe en revue les cadres de détection d'objets, en commençant par les méthodes classiques de vision par ordinateur. Nous classons les approches de détection d'objets en deux catégories :

- (1) les techniques classiques de vision par ordinateur et
- (2) les détecteurs basés sur les CNN.

Nous comparons les principaux modèles de CNN, en discutant de leurs forces et de leurs limites.

En conclusion, Ce chapitre met en lumière les avancées majeures en matière de détection d'objets grâce à l'apprentissage profond et identifie les axes clés de recherche à explorer pour améliorer les performances.

# Vision : tâches principales

Classification



CAT

Détection

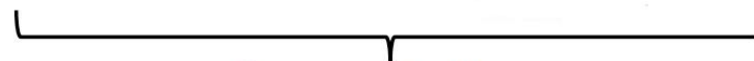


DOG, DOG, CAT

Segmentation



GRASS, CAT,  
TREE, SKY



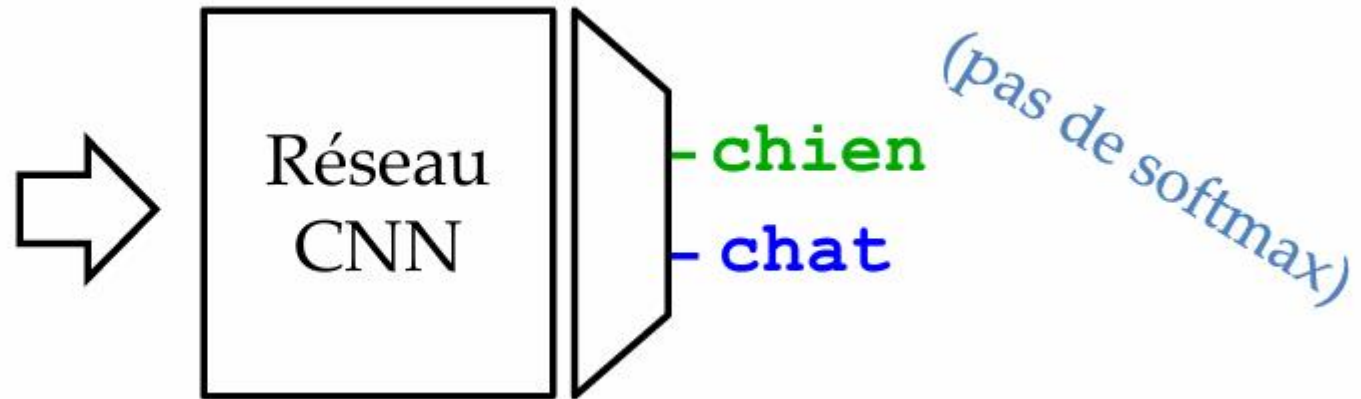
Segmentation  
d'instances



DOG, DOG, CAT

# Détection

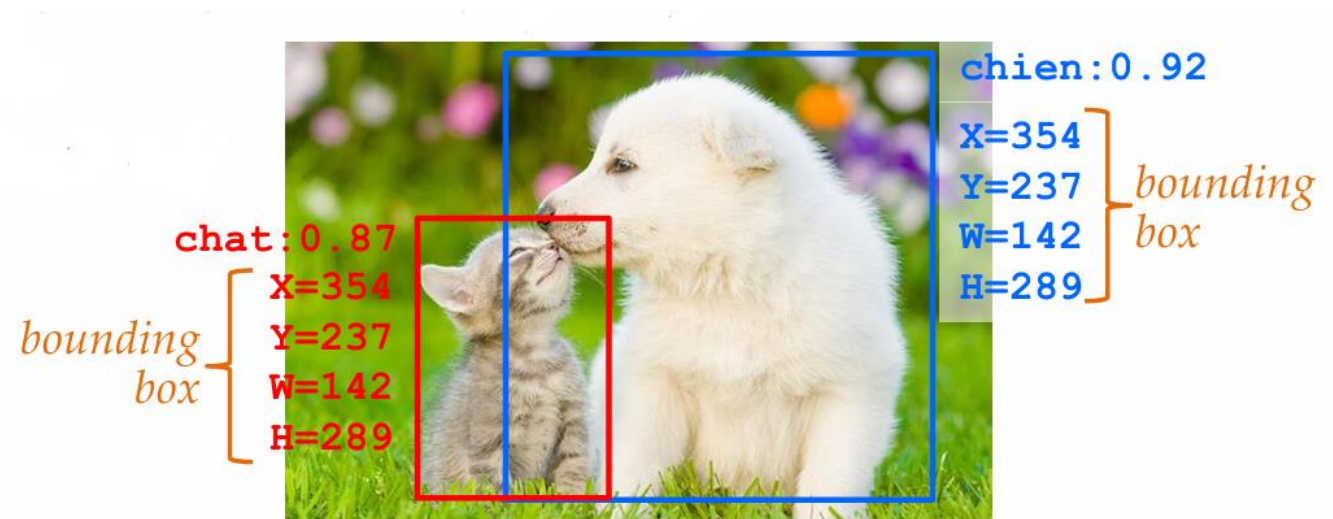
- Une des difficultés est que l'on ne connaît pas le nombre exact d'instances dans l'image
  - si on savait d'avance que les images ne contiennent au maximum qu'un chat et un chien :



- Sujet de recherche très fertile en soit

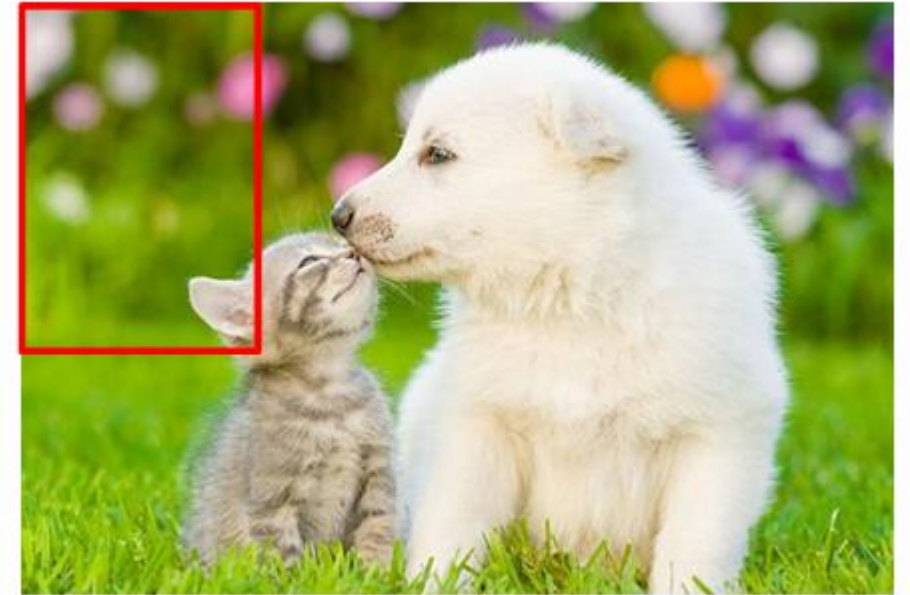
# Détection

- Pour :
  - une image d'entrée
  - une liste prédéterminée de classes
- Trouver, pour tous les objets présents :
  - la position (bounding box)
  - la classe
- Nombre de prédictions va varier d'une image à l'autre
  - Architecture doit en tenir compte



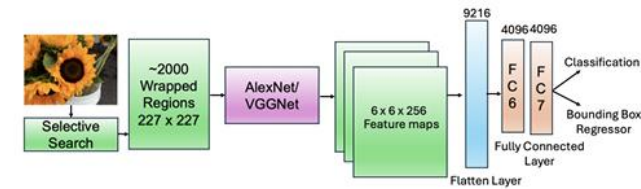
# Détection

- Possible de faire de la détection par un réseau de classification
- Approche par fenêtre coulissante (crops)
- Passe chaque crop dans un réseau classificateur
- Conserve n prédictions les plus confiantes
- Choix de la géométrie de la fenêtre :
  - taille, aspect ratio
- Fastidieux, car des milliers de passes dans le réseau classificateur :  
dizaines de secondes

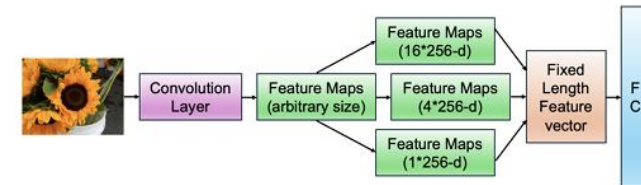


# Detection

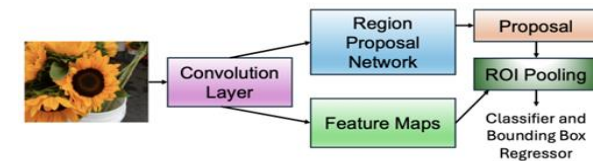
- Basé sur des régions proposées (region proposal)
  - R-CNN
  - Fast-RCNN
  - Faster RCNN
- Grille fixe (grid-based)
  - YOLO (v1, v2, v3,...)



R-CNN Architecture



SPP-Net Architecture

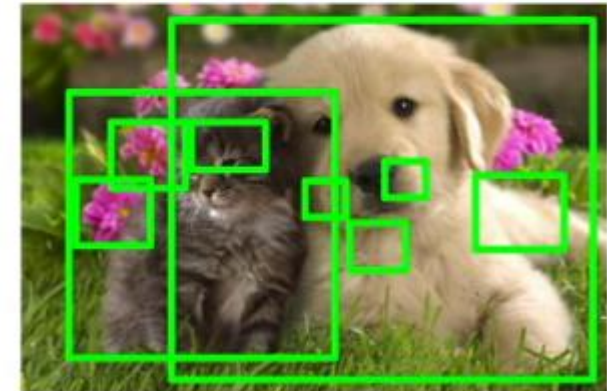
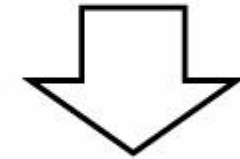


Faster R-CNN Architecture



# Region proposal (classique)

- Algorithmes qui proposent des régions prometteuses en terme de présence d'objets
- Basés parfois sur des heuristiques
  - Recherche de blobs– Distributions particulières de contours (edge)
- Selective Search propose 1000 régions en quelques seconds sur CPU (pas temps-réel)
- Voir aussi Edge Boxes



# Region Proposal Based Framework

Le cadre basé sur la proposition de région, un processus en deux étapes, correspond dans une certaine mesure au mécanisme d'attention du cerveau humain, qui effectue d'abord un balayage grossier de l'ensemble du scénario et se concentre ensuite sur les régions d'intérêt (RoI).

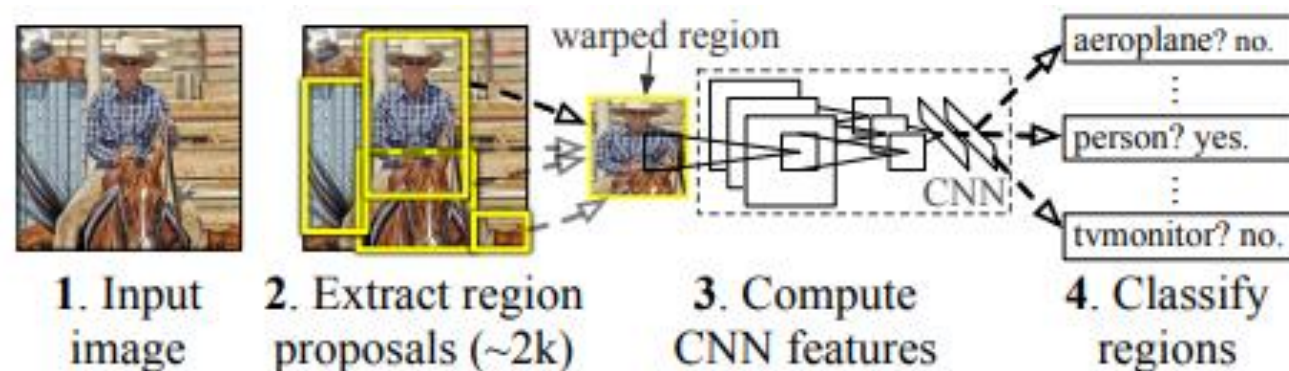
Parmi les travaux antérieurs, le plus représentatif est Overfeat. Ce modèle insère le CNN dans la méthode de la fenêtre coulissante, qui prédit les boîtes de délimitation directement à partir des emplacements de la carte des caractéristiques les plus élevées après avoir obtenu les confiances des catégories d'objets sous-jacentes.

# R-CNN

R-CNN : Il est important d'améliorer la qualité des boîtes englobantes candidates et d'adopter une architecture profonde pour extraire des caractéristiques de haut niveau.

Pour résoudre ces problèmes, R-CNN a été proposé par Ross Girshick en 2014 et a obtenu une précision moyenne (mAP) de 53,3 %, soit une amélioration de plus de 30 % par rapport au meilleur résultat précédent (DPM HSC) sur PASCAL VOC 2012.

La figure montre l'organigramme du R-CNN, qui peut être divisé en trois étapes comme suit.



# R-CNN

## **Génération de propositions de régions.**

Le R-CNN adopte la recherche sélective pour générer environ 2k propositions de régions pour chaque image.

La méthode de recherche sélective s'appuie sur un regroupement ascendant simple et sur des indices de saillance pour fournir rapidement des boîtes candidates plus précises de tailles arbitraires et pour réduire l'espace de recherche dans la détection d'objets.

# R-CNN

## **Redimensionnement des régions**

- Chaque région d'intérêt est redimensionnée à une taille fixe (ex :  $224 \times 224$  pixels) pour être compatible avec le CNN.
- Cela permet d'avoir un traitement uniforme, même si les RoIs ont des tailles variées.

# R-CNN

## **Extraction de caractéristiques profondes basée sur le CNN.**

À ce stade, chaque proposition de région est déformée ou recadrée dans une résolution fixe et le module CNN est utilisé pour extraire une caractéristique de 4096 dimensions comme représentation finale.

Grâce à la grande capacité d'apprentissage, au pouvoir expressif dominant et à la structure hiérarchique des CNN, il est possible d'obtenir une représentation de haut niveau, sémantique et robuste pour chaque proposition de région.

# R-CNN

## **Classification et localisation.**

Avec des SVM linéaires pré-entraînés pour plusieurs classes, différentes propositions de régions sont notées sur un ensemble de régions positives et de régions d'arrière-plan (négatives).

Les régions notées sont ensuite ajustées à l'aide d'une régression de boîtes de délimitation et filtrées avec une suppression non maximale gourmande (NMS) afin de produire des boîtes de délimitation finales pour les localisations d'objets préservées.



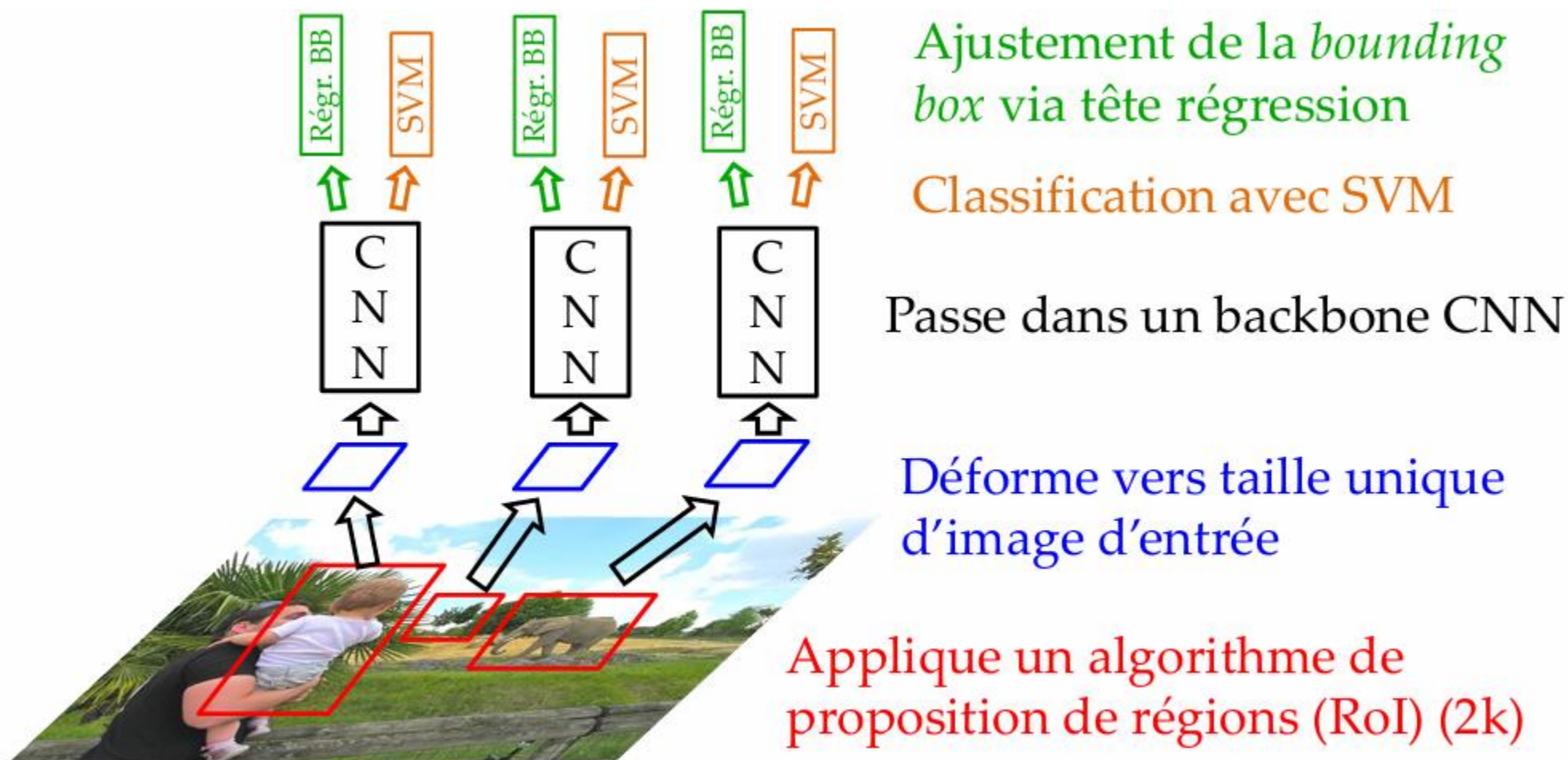
# R-CNN

Lorsque les données étiquetées sont rares ou insuffisantes, un préapprentissage est généralement effectué.

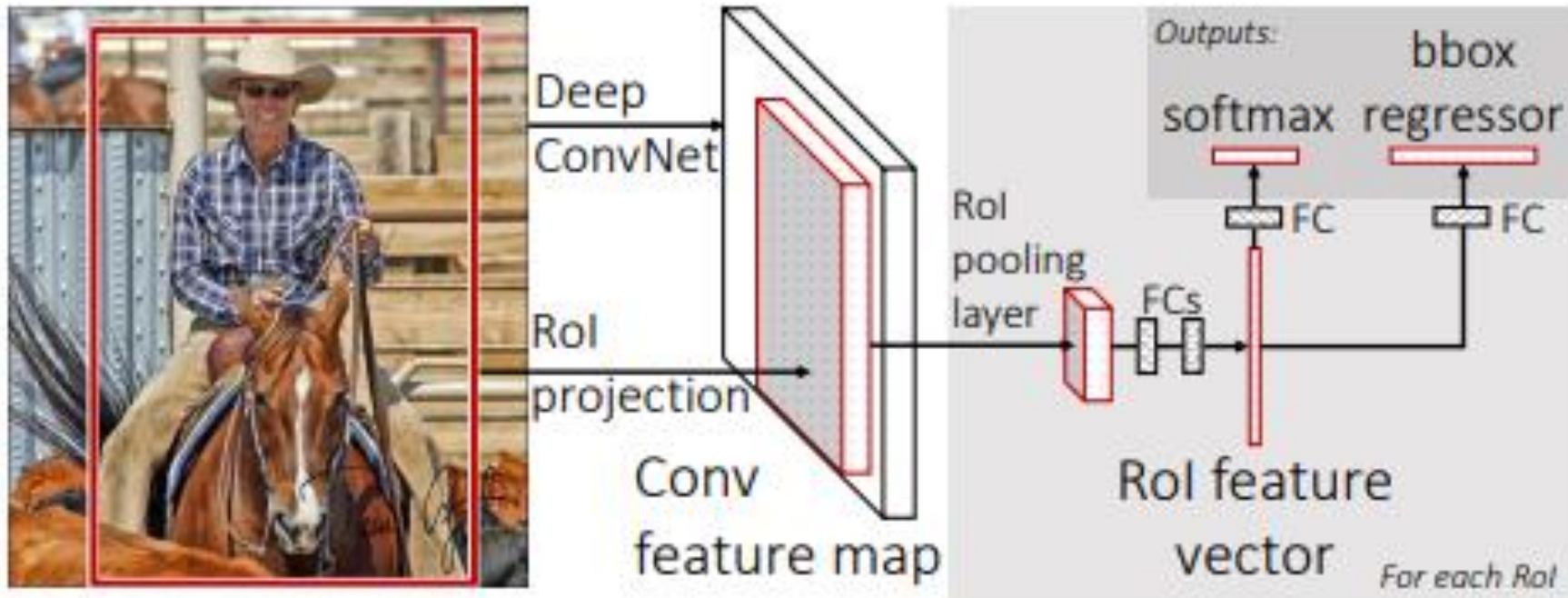
Au lieu d'un préapprentissage non supervisé, le R-CNN effectue d'abord un préapprentissage supervisé sur ILSVRC, un très grand ensemble de données auxiliaires, et procède ensuite à un réglage fin spécifique au domaine.

# R-CNN (2014)

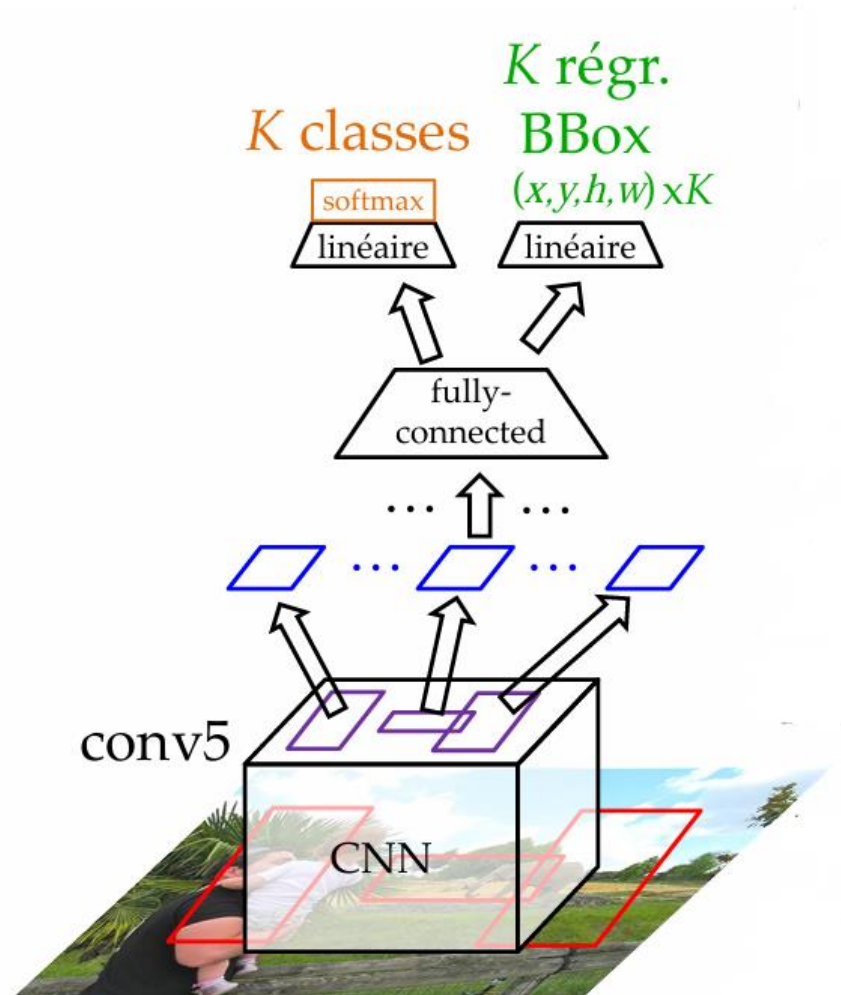
- RoI (region of interest) sont des rectangles
- Ajout d'une classe background
- Lent : inférence en 47 secondes par image, à cause de l'algorithme de proposition de régions



# Fast R-CNN



# Fast R-CNN (2015)



Perte régression

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Ajustement des la *bounding box* via tête de régression

Classification avec réseau linéaire + softmax

Déforme vers taille unique d'image d'entrée

Projette les RoI dans le feature map

Passe dans un backbone CNN

Applique un algorithme de proposition de régions (RoI) (Selective Search)

Temps de calcul

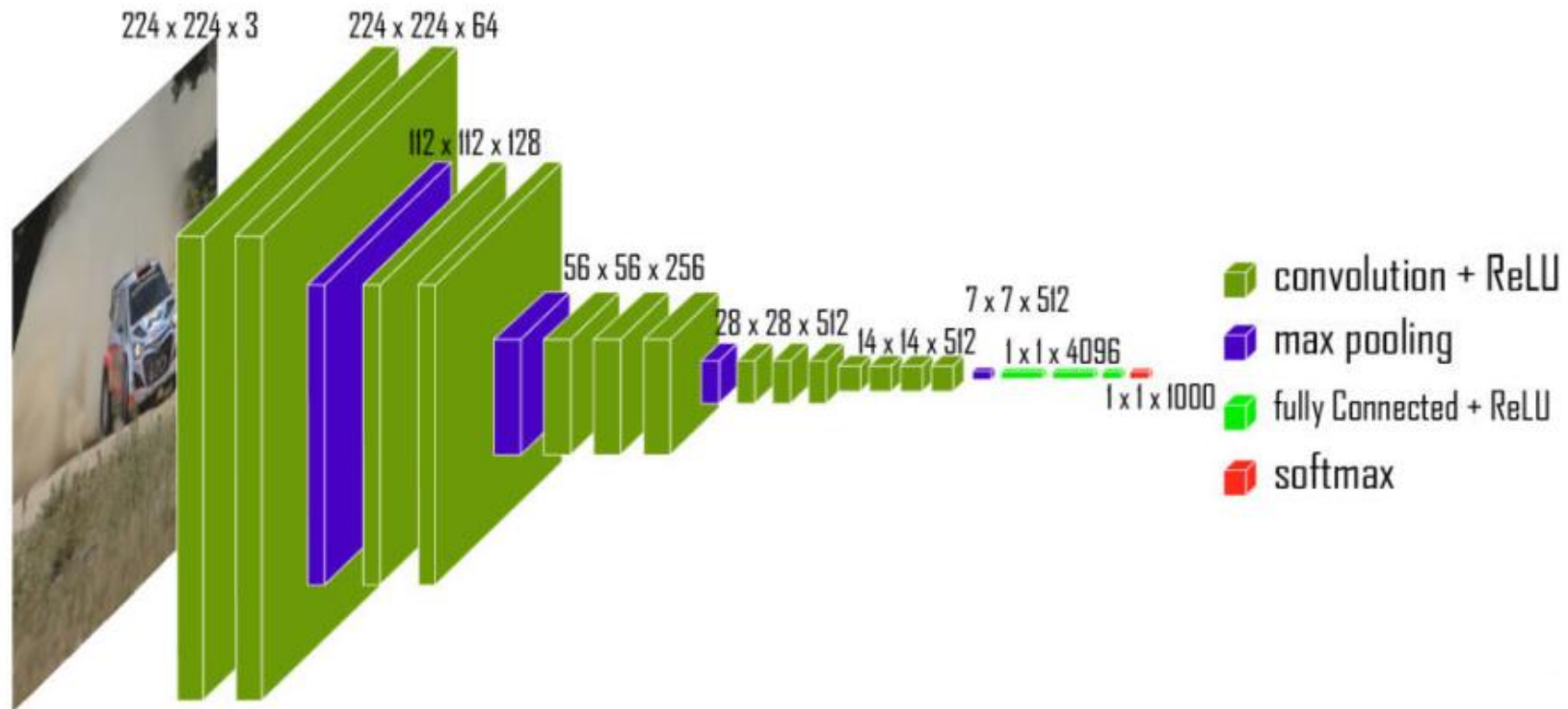
- 2 sec Prop RoI
- 0.3 sec partie réseau

# Fast R-CNN

## cropping feature

Le "cropping feature" dans des modèles comme Fast R-CNN correspond à un processus appelé ROI Pooling (Region of Interest Pooling). C'est une méthode qui permet d'extraire des régions spécifiques (propositions d'objets) directement depuis la feature map du CNN, plutôt que de découper l'image d'origine.

# CNN Network of Fast R-CNN



VGG-16 architecture

Le R-CNN rapide est expérimenté avec trois réseaux ImageNet pré-entraînés, chacun avec 5 couches de mise en commun maximale et 5-13 couches de convolution (telles que VGG-16). Certains changements ont été apportés à ce réseau pré-entraîné :

Le réseau est modifié de telle sorte qu'il utilise en entrée l'image et la liste des propositions de régions générées sur cette image.

Deuxièmement, la dernière couche de mise en commun (ici  $7*7*512$ ) avant les couches entièrement connectées doit être remplacée par la couche de mise en commun de la région d'intérêt (RoI).

Troisièmement, la dernière couche entièrement connectée et la couche softmax sont remplacées par deux couches de classificateur softmax et un régresseur de boîtes englobantes spécifiques à  $K+1$  catégories avec une couche entièrement connectée.

# Region of Interest (RoI) pooling:

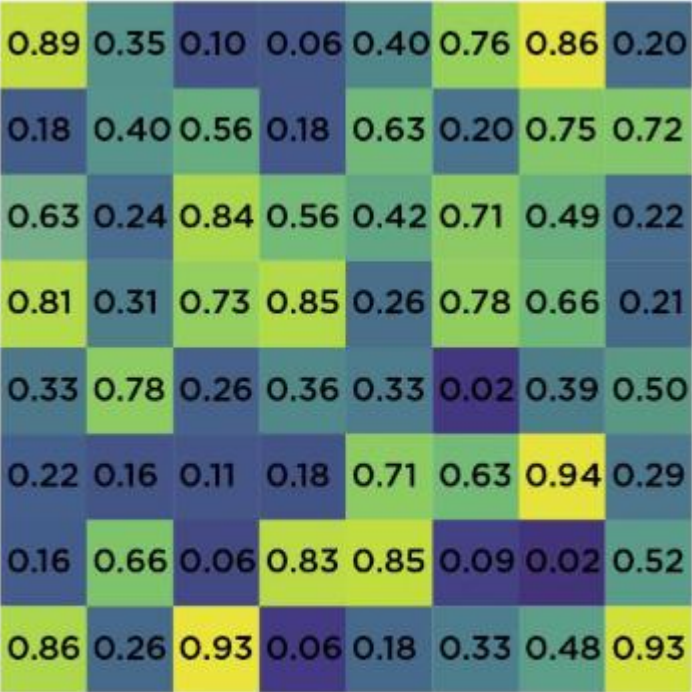
La mise en commun des RoI est une nouveauté qui a été introduite dans l'article sur le R-CNN rapide. Son objectif est de produire des cartes de caractéristiques uniformes et de taille fixe à partir d'entrées non uniformes (RoI). Il prend deux valeurs en entrée :

Une carte de caractéristiques a été obtenue à partir de la couche CNN précédente (14 x 14 x 512 dans VGG-16).

Une matrice  $N \times 4$  représente les régions d'intérêt, où  $N$  est le nombre de RoI, les deux premières représentent les coordonnées du coin supérieur gauche de la RoI et les deux autres représentent la hauteur et la largeur de la RoI, désignées par  $(r, c, h, w)$ .

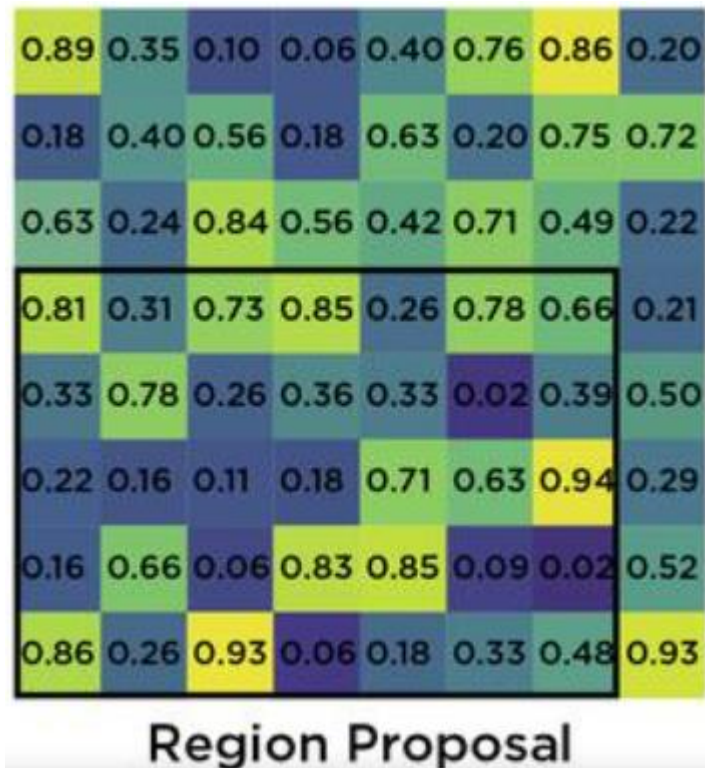


Considérons que nous avons 8\*8 cartes de caractéristiques, nous devons extraire une sortie de taille 2\*2. Nous suivrons les étapes ci-dessous.

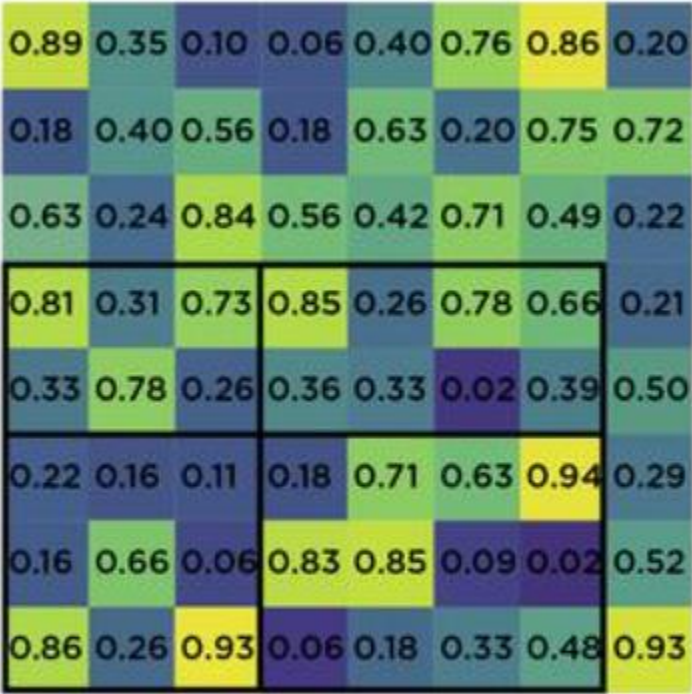


Input Activation

Supposons que l'on nous donne les coordonnées du coin gauche du RoI (0, 3), la hauteur et la largeur (5, 7).

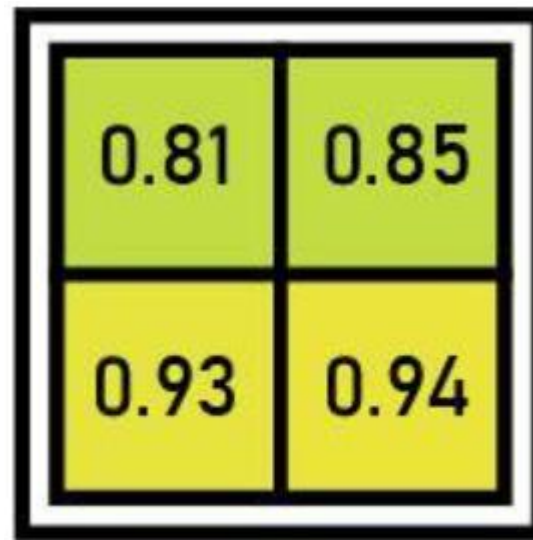


Maintenant, si nous devons convertir cette proposition de région en un bloc de sortie 2 x 2 et que nous savons que les dimensions de la section de mise en commun ne sont pas parfaitement divisibles par la dimension de la sortie. Nous prenons la mise en commun de manière à ce qu'elle soit fixée à des dimensions de 2 x 2.



Pooling Sections

Nous appliquons maintenant l'opérateur de regroupement maximal pour sélectionner la valeur maximale de chacune des régions que nous avons divisées.



0.81	0.85
0.93	0.94

*Max pooling output*

# Quelques résultats de Fast R-CNN sur différentes bases de données VOC

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11] <sup>†</sup>	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	<b>44.6</b>	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	<b>35.6</b>	66.8	67.2	70.4	<b>71.1</b>	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	<b>79.0</b>	68.6	57.0	39.3	79.5	<b>78.6</b>	81.9	<b>48.0</b>	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	<b>77.0</b>	78.1	<b>69.3</b>	<b>59.4</b>	38.3	<b>81.6</b>	<b>78.6</b>	<b>86.7</b>	42.8	<b>78.8</b>	<b>68.9</b>	<b>84.7</b>	<b>82.0</b>	<b>76.6</b>	<b>69.9</b>	31.8	<b>70.1</b>	<b>74.8</b>	<b>80.4</b>	70.4	<b>70.0</b>

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without “difficult” examples, **07+12**: union of **07** and VOC12 trainval. <sup>†</sup>SPPnet results were prepared by the authors of [11].

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.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	<b>82.3</b>	75.2	67.1	50.7	<b>49.8</b>	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	<b>41.5</b>	<b>71.9</b>	62.2	73.2	<b>64.6</b>	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	<b>77.8</b>	<b>71.6</b>	<b>55.3</b>	42.4	<b>77.3</b>	<b>71.7</b>	<b>89.3</b>	<b>44.5</b>	<b>72.1</b>	<b>53.7</b>	<b>87.7</b>	<b>80.0</b>	<b>82.5</b>	<b>72.7</b>	36.6	68.7	<b>65.4</b>	<b>81.1</b>	62.7	<b>68.8</b>

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

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	<b>43.0</b>	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	<b>38.6</b>	<b>68.3</b>	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.

# Fast R-CNN : Résumé

Tout d'abord, nous générons la proposition de région à partir d'un algorithme de recherche sélective. Cet algorithme de recherche sélective génère jusqu'à environ 2000 propositions de régions.

Ces propositions de régions (projections de RoI) se combinent aux images d'entrée transmises à un réseau CNN. Ce réseau CNN génère la carte de caractéristiques de convolution en sortie.

Ensuite, pour chaque proposition d'objet, une couche de regroupement des régions d'intérêt (RoI) extrait le vecteur de caractéristiques de longueur fixe pour chaque carte de caractéristiques.

Chaque vecteur de caractéristiques est ensuite transmis à deux couches de classificateur softmax et de régression Bbox pour la classification de la proposition de région et l'amélioration de la position de la boîte englobante de cet objet.

# La différence entre R-CNN et Fast R-CNN

## R-CNN (2014) :

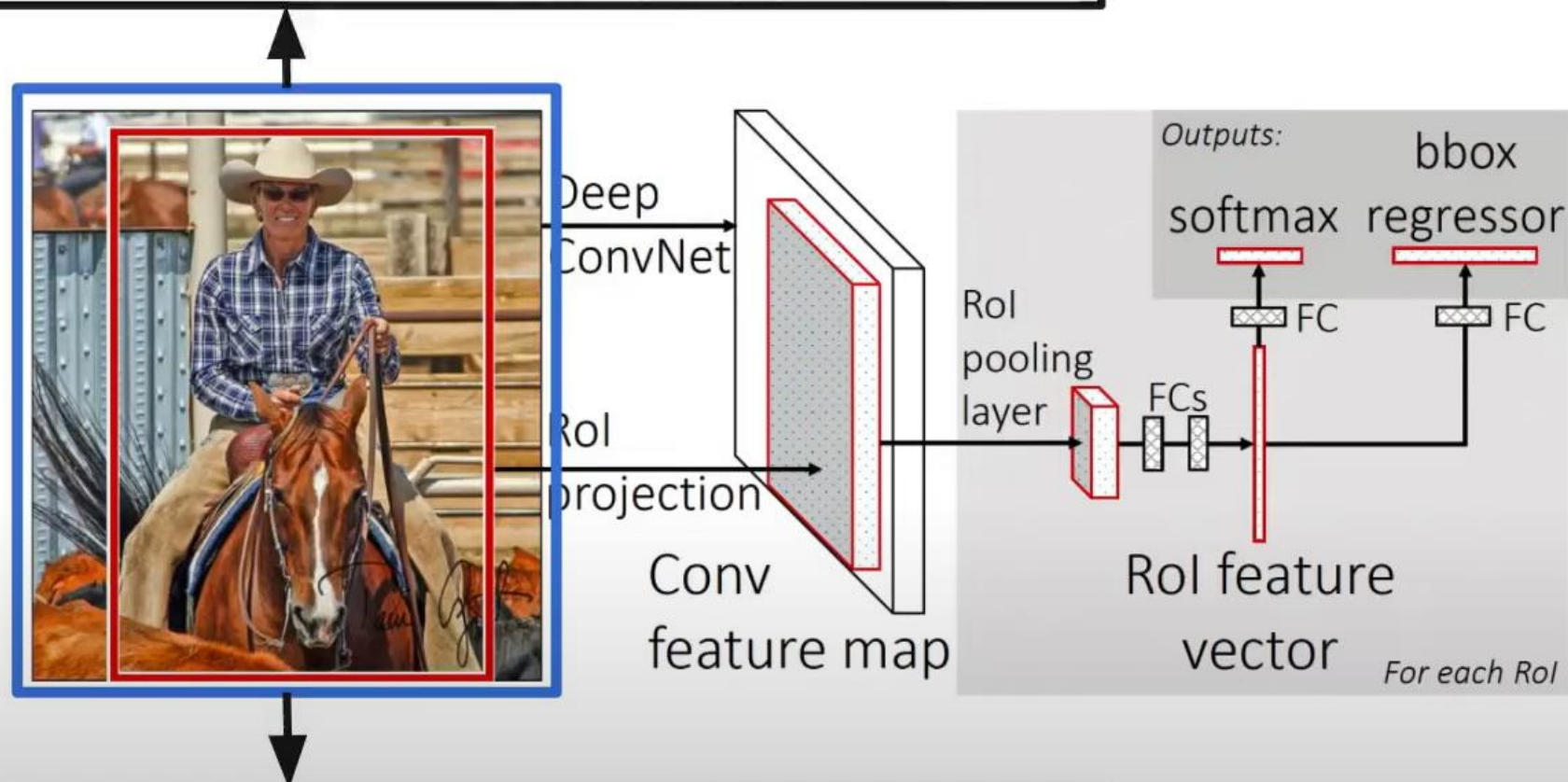
- Génère d'abord des propositions de régions (~2000) avec Selective Search.
- Pour chaque proposition, elle extrait les caractéristiques avec un CNN (ex: AlexNet).
- Chaque région est redimensionnée et passée indépendamment dans le CNN.
- Ensuite, un classifieur SVM est utilisé pour la classification, et une régression pour ajuster les boîtes englobantes.

## Fast R-CNN (2015) :

- L'image complète passe une seule fois dans le CNN pour générer une feature map partagée.
- Ensuite, les propositions de régions (toujours générées avec Selective Search) sont projetées sur cette feature map.
- Chaque région est passée par un RoI pooling pour obtenir une taille fixe, puis classifiée via un seul réseau fully connecté.



Still quite slow for most of practical applications

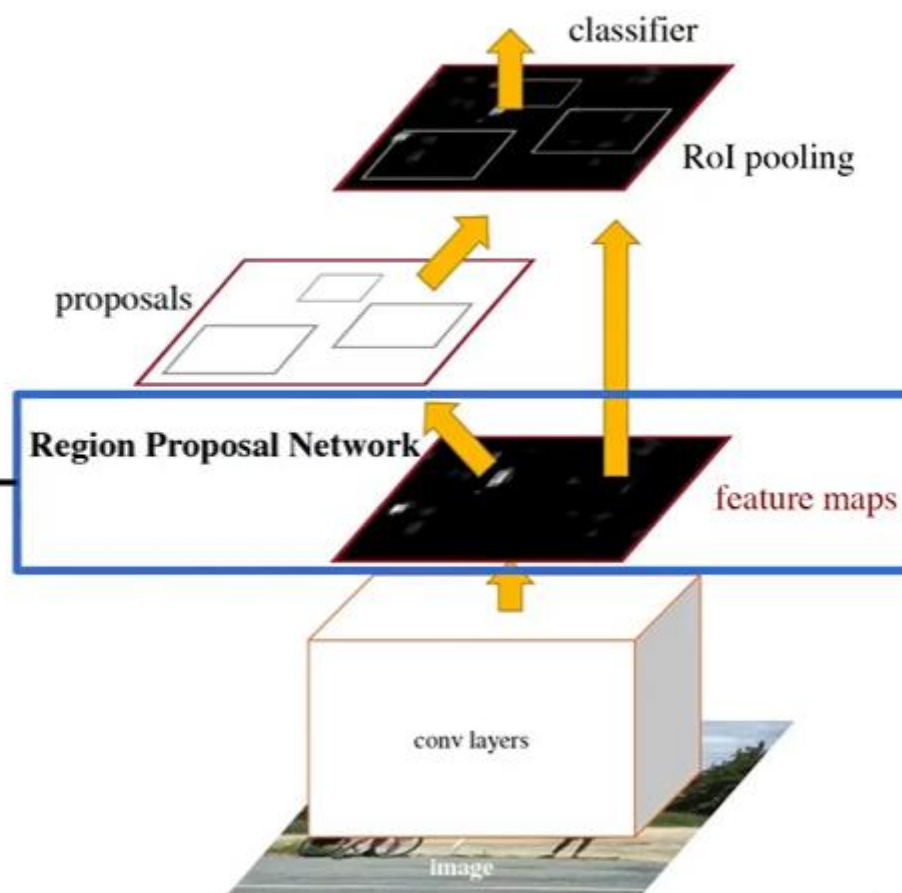


The region proposal algorithm is an external algorithm that was not specifically tuned for object detection



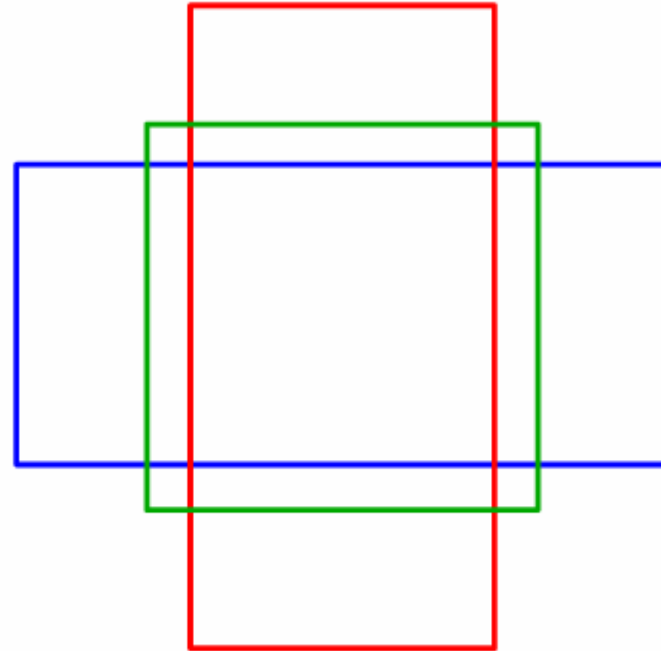
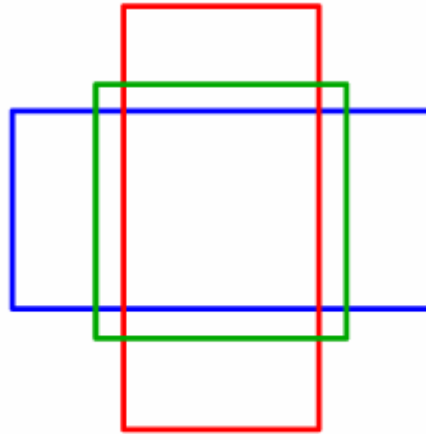
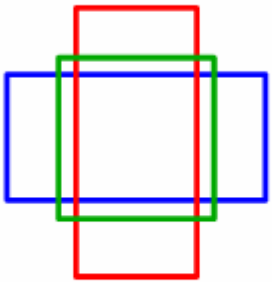
# Faster R-CNN

Replaces the region proposal algorithm with a significantly faster neural network that can learn to propose better regions for the task at hand.



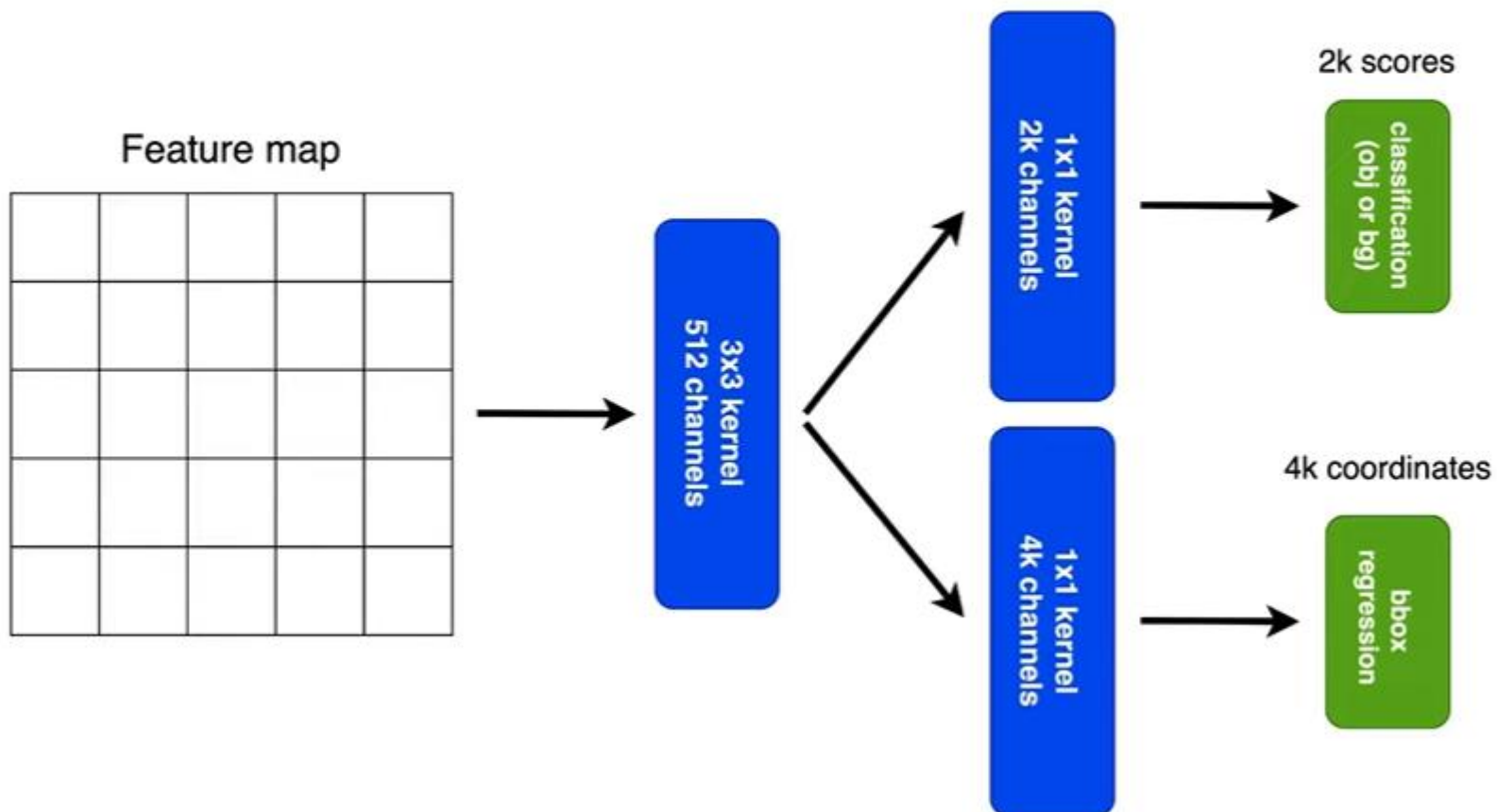
# Faster R-CNN

- Laisser le réseau faire les propositions : RPN (Region Proposal Network)
- Accélère grandement le processus
- RPN s'améliore via backprop des pertes
- Introduction de 9 prototypes de bounding box (anchor box) :




























RPN

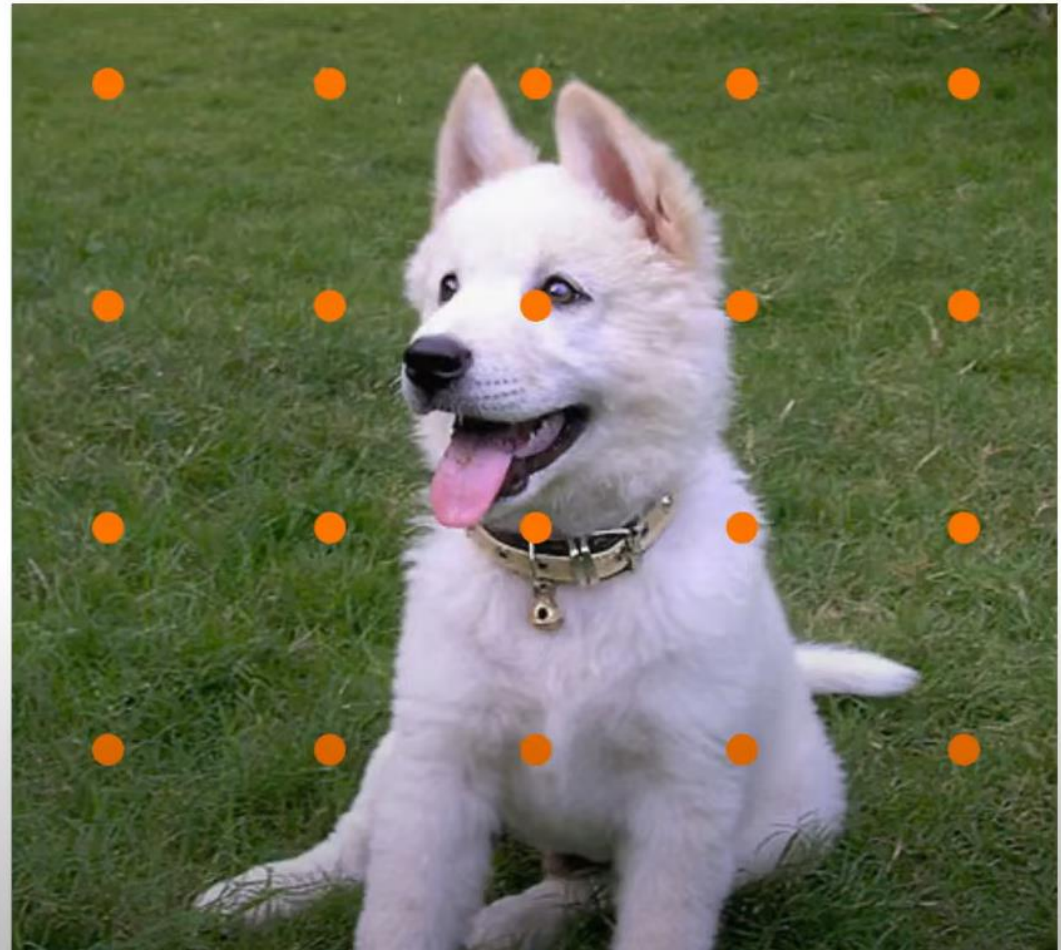
# Region Proposal Network



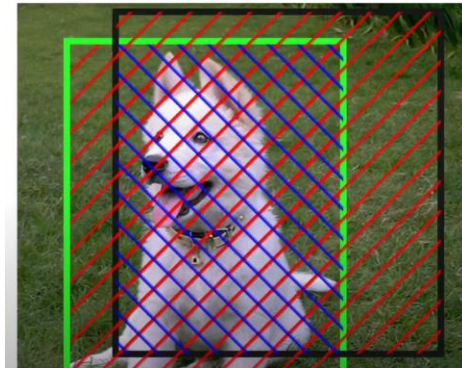
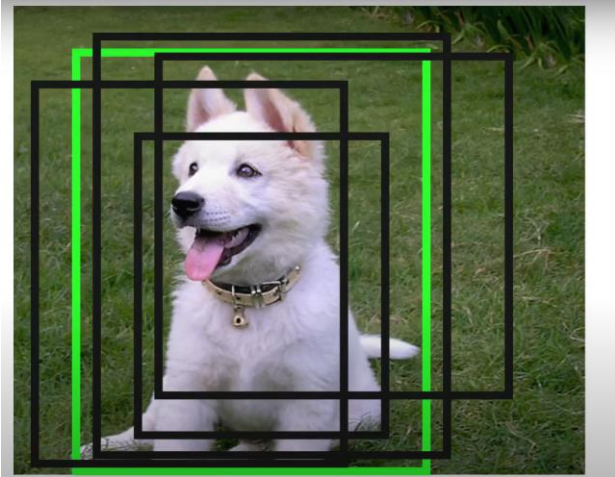
# Intersection over union

Feature map anchor points

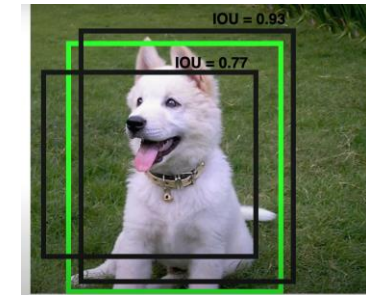
				
				
				
				
				



# Intersection over union



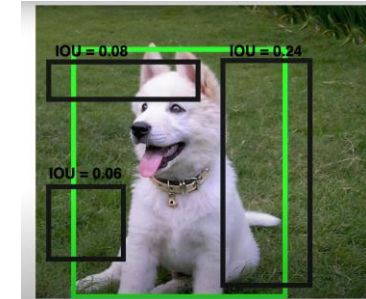
$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$



$\text{IoU} > 0.7$



positive

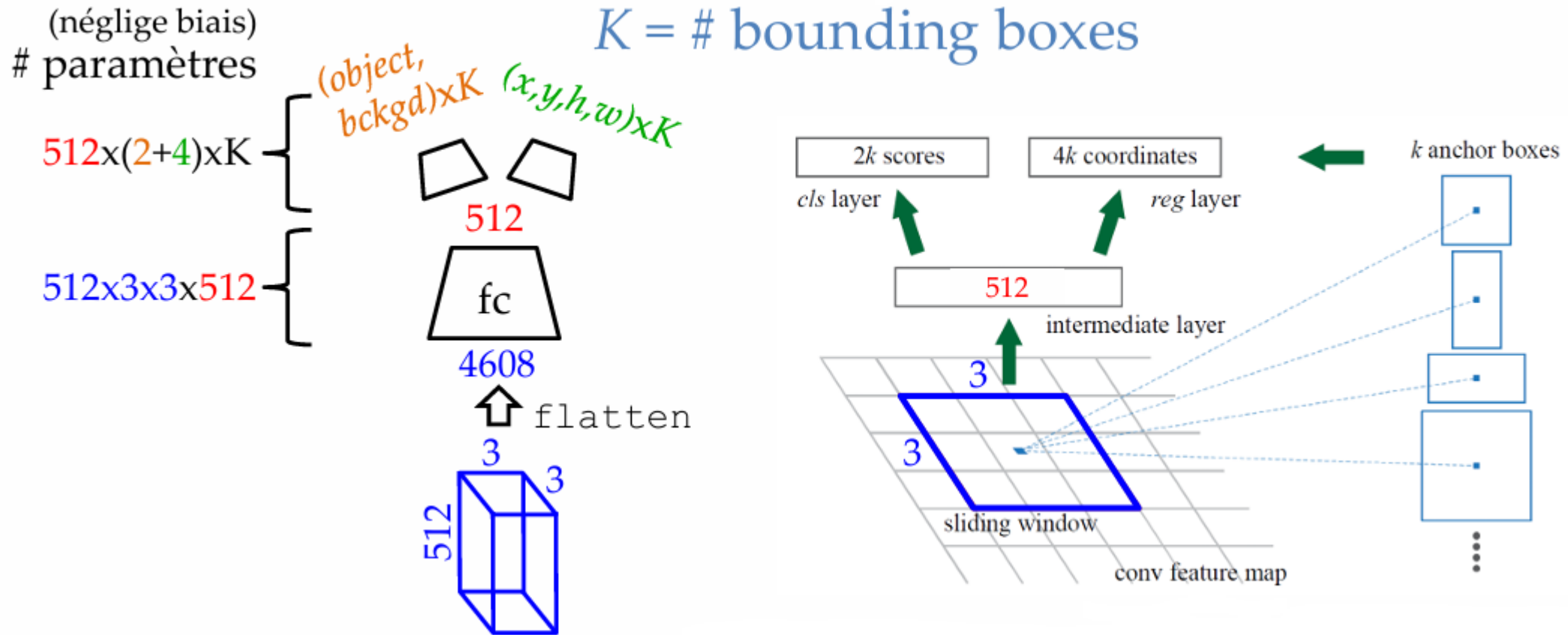


$\text{IoU} < 0.3$



negative

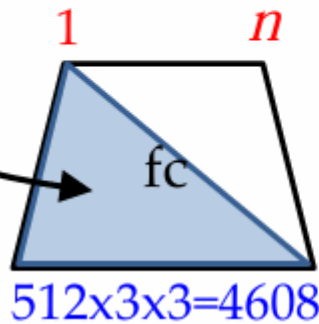
# Faster R-CNN : Fully-convolutional





# Faster R-CNN

filtre  
conv  
3x3 !



Appliquer un réseau *fully-connected* avec  $n$  sorties, de manière coulissante = Appliquer  $n$  filtres de convolution

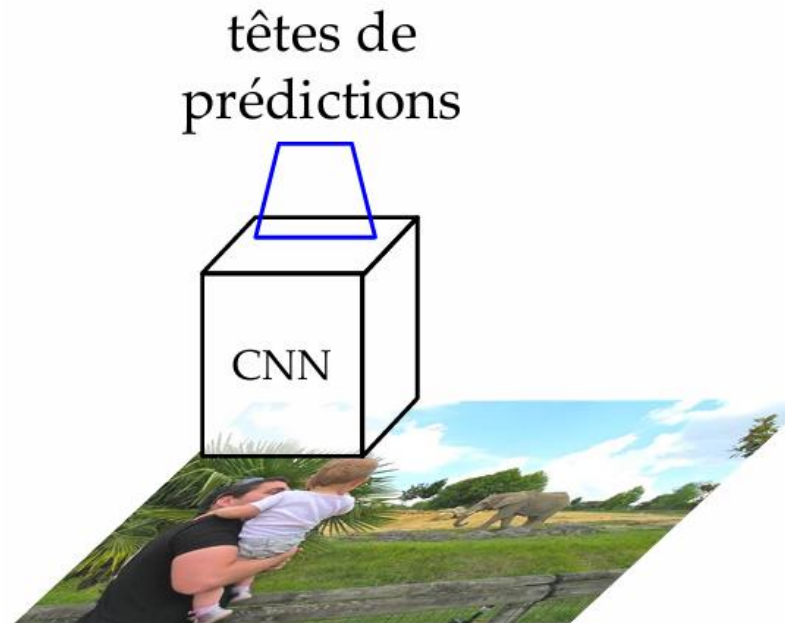
**Fully-convolutional network (FCN)**

**Approche détection par fenêtre coulissante mais très efficace!**

# Faster R-CNN

- **Coulissage via convolution : très rapide**

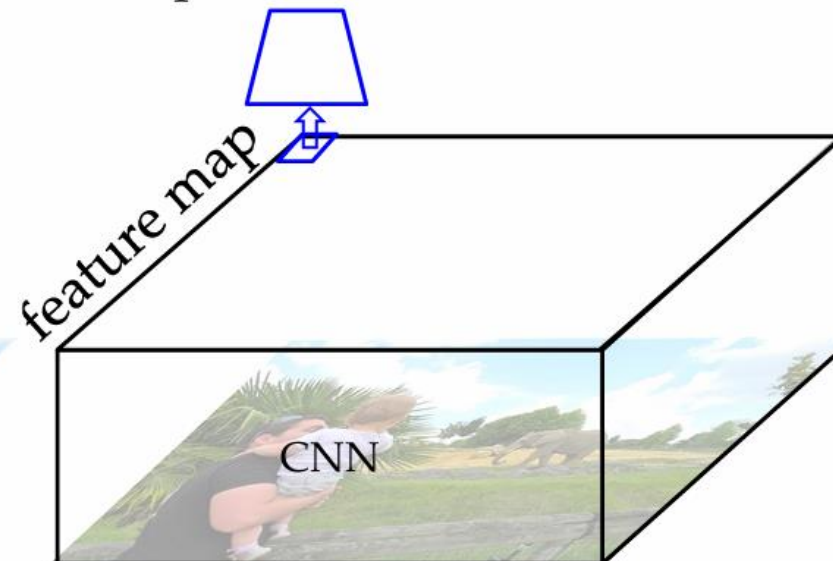
Détection par coulissage  
« traditionnel »



**Faster R-CNN**

...simplement en ajoutant quelques  
couches de convolution en guise de têtes

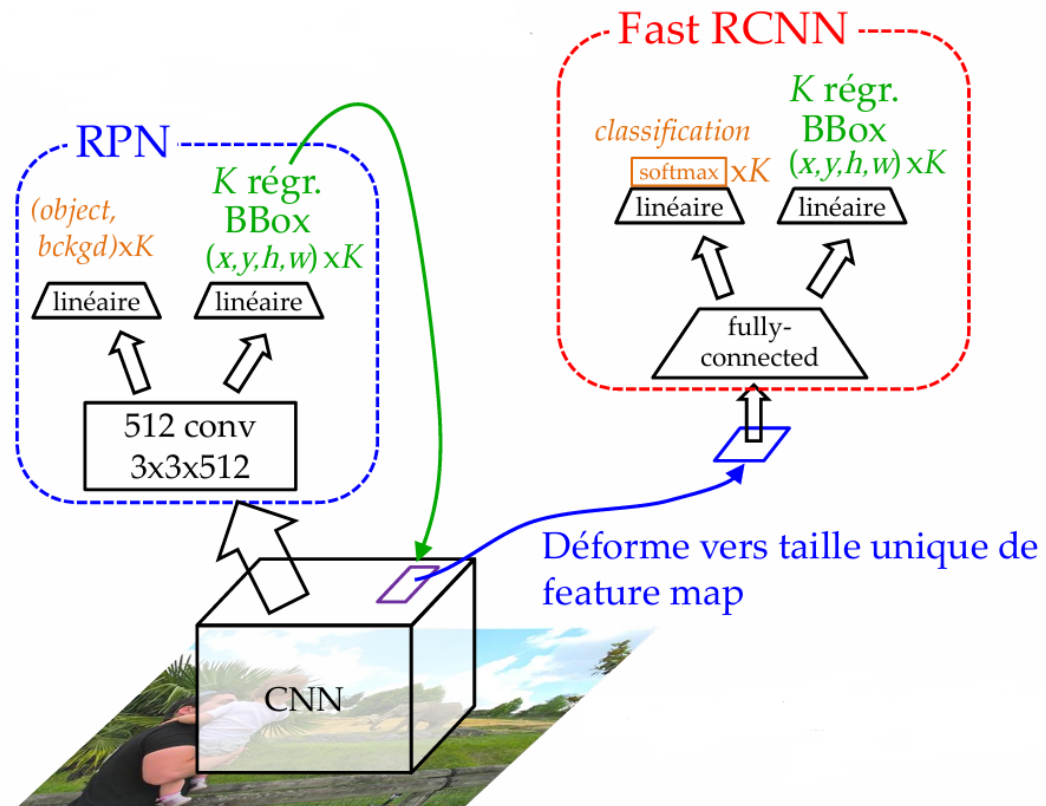
têtes de  
prédictions





# Faster R-CNN

Prendre les (300-2000) RoI avec les scores d'objectedness les plus élevés



Gradients sur les pertes vont aider les deux réseaux

# YOLO

La détection d'objets est une tâche fondamentale en Computer Vision: elle permet aux intelligences artificielles de localiser et classifier les objets présents dans des images ou des vidéos.

La capacité à détecter précisément les objets a de nombreuses applications, allant des voitures autonomes aux systèmes de surveillance.

Ces dernières années, un algorithme a gagné en popularité pour ses performances exceptionnelles dans la détection d'objets: You Only Look Once (YOLO).

# You Only Look Once (YOLO)

You Only Look Once (YOLO) est un détecteur d'objets en une seule étape conçu par Redmon, J. où la détection d'objets est effectuée comme un problème de régression.

Il prédit les coordonnées des boîtes de délimitation des objets et détermine la probabilité de la catégorie à laquelle il est associé. L'utilisation d'un seul réseau permet une optimisation de bout en bout .

Il prédit les détections directement à l'aide d'une sélection limitée de régions candidates. Contrairement aux approches basées sur les régions, qui utilisent les caractéristiques d'une région spécifique, YOLO utilise les caractéristiques de l'ensemble de l'image.

# YOLO

YOLO ou "You Only Look Once", est un outil spécial qui aide les ordinateurs à voir rapidement et avec précision les choses dans les images, les fichiers textes ou les vidéos.

Créé par les experts Joseph Redmon et Ali Farhadi en 2015, YOLO est plus rapide que les outils plus anciens car il analyse l'image entière en une seule fois. Cette vérification rapide permet à YOLO d'identifier rapidement s'il y a d'autres objets, comme des voitures, des arbres ou des animaux, et où ils se trouvent dans l'image.



Ali Farhadi

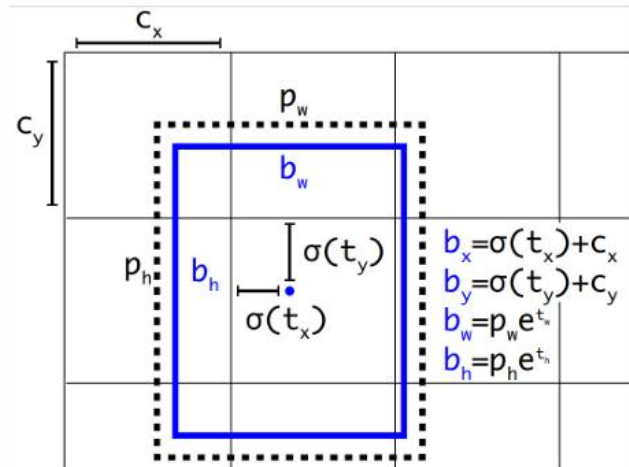
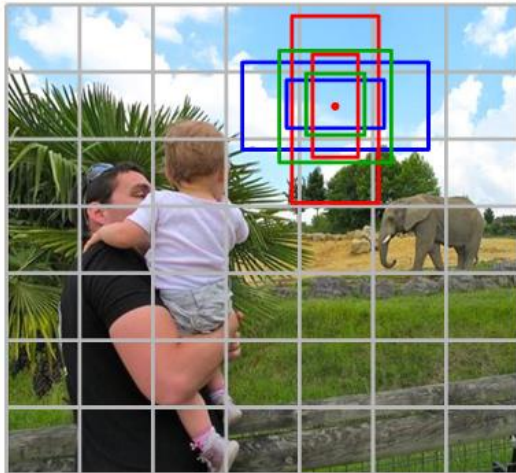
Professor, Computer Science and Engineering, [University of Washington](#)  
Adresse e-mail validée de cs.uw.edu - [Page d'accueil](#)

[Computer Vision](#) [Machine learning](#) [Artificial Intelligence](#)

Citée par	TOUT AFFICHER	
	Toutes	Depuis 2020
Citations	174901	147125
<a href="#">indice h</a>	91	82
indice i10	178	159

# Approche sans proposal (Yolo v3)

- Grille régulière (7x7 → feature map)
- Pour chaque position centrale de la grille
  - pour chaque anchor box, prédire :
    - classe (incluant la classe background)
    - régression ( $b_x, b_y, b_w, b_h$ ) sur les paramètres de l'anchor box + confiance (objectness)
- Prédiction obtenues via Fully-convolutional (FCN) d'une 1x1 : très rapide! (20 ms)

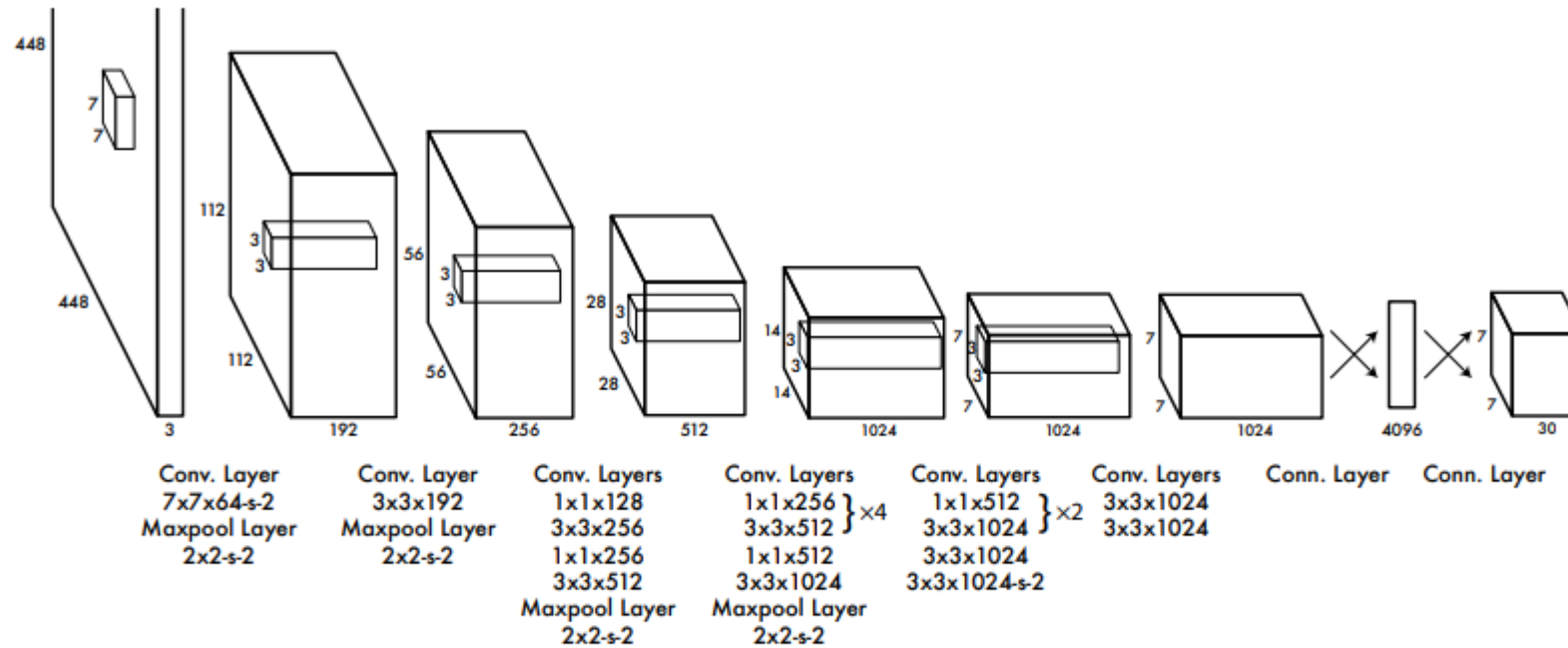


Le modèle est implémenté sous forme de réseau de neurones convolutifs et évalué sur le jeu de données de détection PASCAL VOC.

Les premières couches convolutionnelles extraient les caractéristiques de l'image d'entrée, tandis que les couches entièrement connectées prédisent les probabilités de sortie et les coordonnées des boîtes englobantes. L'architecture du réseau s'inspire du modèle GoogLeNet utilisé pour la classification d'images.

Elle est composée de 24 couches convolutionnelles suivies de 2 couches entièrement connectées. Au lieu des modules inception utilisés dans GoogLeNet, la conception utilise des couches de réduction  $1 \times 1$  suivies de couches convolutionnelles  $3 \times 3$ , conformément à l'approche de Lin.

# L'architecture complète



**Figure 3: The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.



# YOLO

Une version rapide de YOLO est également entraînée afin d'explorer les limites de la détection d'objets en temps réel.

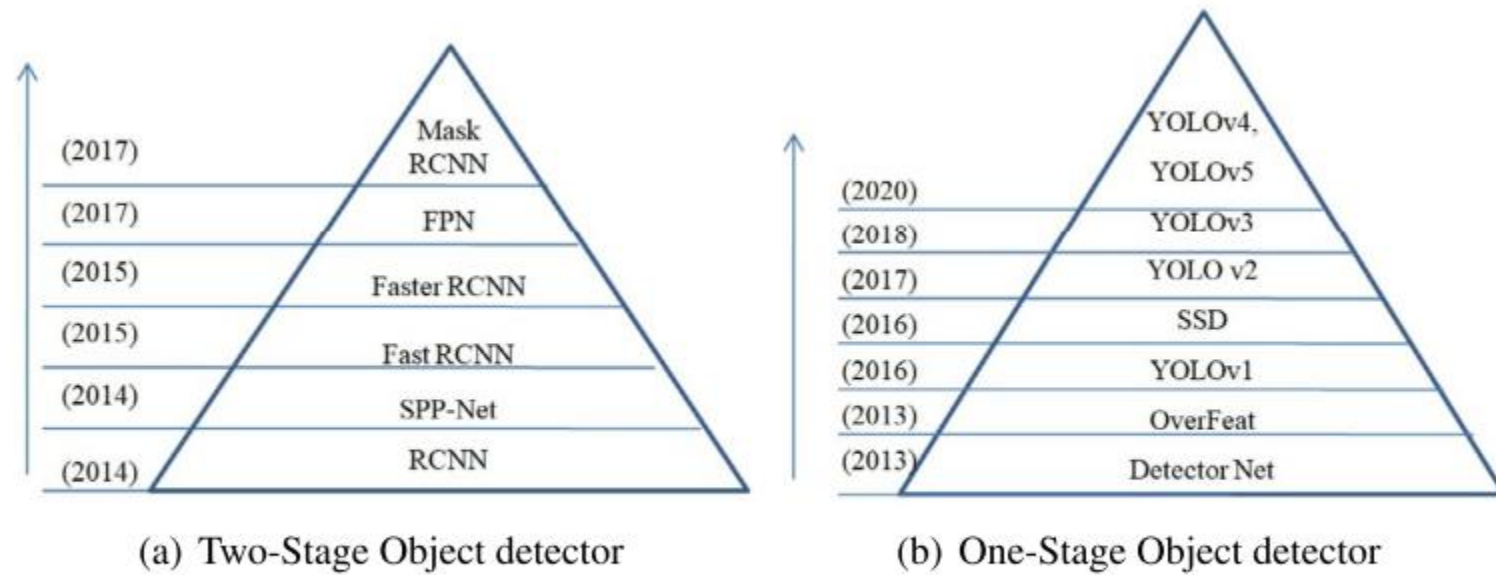
Fast YOLO utilise un réseau de neurones avec un nombre réduit de couches convolutionnelles (9 au lieu de 24) et un plus petit nombre de filtres par couche.

À part la taille du réseau, tous les autres paramètres d'entraînement et de test sont identiques entre YOLO et Fast YOLO.

**TABLE I: Comparison of CNN-Based Object Detection Architectures**

<b>Model</b>	<b>Strengths</b>	<b>Limitations</b>
R-CNN (2013)	Simple, foundational; applies CNNs for classification.	High computation for 2000 region classifications; slow (47 sec/image); no end-to-end training.
SPPNet (2015)	Faster than R-CNN; supports multi-scale input via spatial pyramid pooling.	Does not update conv. layers before SPP layer during fine-tuning.
Fast R-CNN (2015)	Faster than SPPNet; introduces ROI pooling to handle varied input sizes.	Relies on selective search for region proposals, not learned during training.
Faster R-CNN (2015)	Uses RPN for fast region proposals; improves efficiency.	Limited in detecting small objects due to single feature map.
Mask R-CNN (2017)	Adds instance segmentation, detecting objects and masks simultaneously.	High computational demand; struggles with motion blur at low resolution.
YOLO (2015)	Real-time detection at 45 fps; single forward pass.	Poor detection of small objects; produces coarse features.
SSD (2016)	Handles various resolutions; uses multi-scale feature maps for detection.	Default boxes may not match all shapes; possible overlapping detections.

Object Detector	Year	Image Input size	Backbone DCNN	Region Proposal Method	Learning Method / (Optimization method)	Loss / Cost function	Strengths	Shortcomings
RCNN [25]	2014	Fixed	AlexNet	Selective Search	SGD,BP	Hinge loss, Bounding box regressor loss	<ol style="list-style-type: none"> <li>1. First Neural Network based on region proposal for higher detection quality.</li> <li>2. Increase in the performance is seen over traditional state-of-the-art methods</li> </ol>	<ol style="list-style-type: none"> <li>1. Training is costly as huge amount of space and time is required.</li> <li>2. Multi-stage pipeline training is used.</li> <li>3. CNN is frequently applied to 2000 image regions, so the feature extraction is the main time constraint in testing.</li> <li>4. Extracting 2000 image regions is a difficult task as features are extracted for every image region.</li> </ol>
SPP-Net [26]	2014	Arbitrary	ZFNet	Selective Search	SGD	Hinge loss, Bounding box regressor loss	<ol style="list-style-type: none"> <li>1. Extracts the features of entire image at once.</li> <li>2. Outputs the fixed length features regardless of image size.</li> <li>3. Faster than RCNN.</li> </ol>	<ol style="list-style-type: none"> <li>1. Architecture is identical to RCNN, so it has same drawbacks as RCNN.</li> <li>2. No end-to-end training.</li> </ol>
Fast RCNN [27]	2015	Arbitrary	Alexnet VGGM VGG16	Selective Search	SGD	Classification loss, Bounding box regression loss	<ol style="list-style-type: none"> <li>1. First end-to-end detector training.</li> <li>2. Faster and accurate than RCNN and SPP-Net.</li> <li>3. Single-stage training network.</li> <li>4. RoI pooling layer used.</li> </ol>	<ol style="list-style-type: none"> <li>1. Sluggish for real time applications because of selective search.</li> <li>2. Region proposal computation is a bottleneck.</li> <li>3. No end-to-end training.</li> </ol>
Faster RCNN [20,21]	2015	Arbitrary	ZFNet VGG16	Region Proposal Network	SGD	Classification loss (class log loss), Bounding box regression loss	<ol style="list-style-type: none"> <li>1. Introduces RPN that generates cost free region proposals.</li> <li>2. Established translation-invariant and multi-scale anchors.</li> <li>3. An integrated network comprising RPN and Fast RCNN is designed with common Convolutional layers.</li> <li>4. Provides end-to-end training.</li> </ol>	<ol style="list-style-type: none"> <li>1. Training is complex; inefficient for real time applications.</li> <li>2. Lack of performance for small and multi-scale objects.</li> <li>3. Speed is slow.</li> </ol>
Feature Pyramid Network [28]	2017	Arbitrary	ResNet50 ResNet101	Region Proposal Network	Synchronized SGD	Classification loss (Class log loss), Bounding box regression loss	<ol style="list-style-type: none"> <li>1. Multi-level feature fusion FPN is designed.</li> <li>2. Accurate solution to multi-scale object detection.</li> <li>3. Follows top-down structure with lateral connections.</li> </ol>	<ol style="list-style-type: none"> <li>1. It is still required to use pyramid representation to tackle multi-scale challenges.</li> <li>2. Speed is yet the bottleneck for detection purpose; cannot fulfill real time needs.</li> </ol>
Mask RCNN [29]	2017	Arbitrary	ResNet101 ResNext101	Region Proposal Network	SGD	Classification loss, Bounding box regression loss, Mask loss (average binary cross entropy loss)	<ol style="list-style-type: none"> <li>1. RoIAlign pooling layer is used rather than RoI pooling layer; thus increase in the detection accuracy.</li> <li>2. Simple and flexible architecture for object instance segmentation.</li> <li>3. Pixel to pixel alignment is carried out.</li> </ol>	<ol style="list-style-type: none"> <li>1. Detection speed is low to satisfy real time requirements.</li> </ol>



**Fig. 1.** Classification of generic object detection models (a) Two-stage detectors from period 2014 to 2017 [20,21,25–29]. (b) One-stage detectors from period 2013 to 2020 [22,23,30–37].

# YOLO implementation

# YOLOV8 implementation

YOLOv8 est écrit en PyTorch et maintenu par **Ultralytics**. Il est beaucoup plus facile à utiliser que les versions précédentes.

```
from ultralytics import YOLO
```



Chargement d'un modèle pré-entraîné

```
model = YOLO('yolov8n.pt')
```

Vous pouvez utiliser d'autres versions comme `yolov8s.pt`, `yolov8s`, `yolov8m`, `yolov8l`, `yolov8x`.

# YOLOV8 implementation

**Lancer une inférence sur une image**

```
results = model("C:/Users/hp/Pictures/Capture.png")
```

**Afficher les résultats**

Affiche l'image avec les boîtes de délimitation

```
results[0].show()
```

Sauvegarde de l'image avec les prédictions

```
results[0].save('output.jpg')
```

# Segmentation



FIN Bon courage

# Annexe

# Dataset of Cancer

*Serving the Clinical and Computer Vision Communities (<https://www.isic-archive.com/>)*

- TorchVision est une bibliothèque open source qui fournit aux développeurs et aux chercheurs un large éventail d'outils et de fonctionnalités pour s'attaquer à diverses tâches de vision par ordinateur, allant de la classification d'images à la détection et à la segmentation d'objets

# SPPNet

Bien que SPPNet soit plus performant que RCNN en termes d'efficacité et de précision, il présente encore quelques problèmes car il suit grosso modo la même procédure que RCNN, qui comprend le réglage fin du réseau, l'extraction des caractéristiques et la régression de la boîte englobante (bounding box). Girshick, R. a montré qu'il était possible d'améliorer encore le RCNN et le SPPNet, et a proposé un nouveau détecteur appelé Fast RCNN. Il permet une formation de bout en bout du détecteur qui apprend simultanément le classificateur softmax et la régression de la boîte englobante spécifique à la classe, avec une perte multitâche, plutôt que de les former séparément comme dans RCNN et SPPNet. Dans le RCNN rapide, au lieu d'exécuter le CNN 2000 fois par image, il n'est exécuté qu'une seule fois et obtient toutes les régions d'intérêt. Une couche de mise en commun des Rdl a ensuite été ajoutée entre la couche convolutive finale et la couche initiale entièrement connectée, de sorte qu'une caractéristique de longueur fixe est extraite pour toutes les propositions de régions.