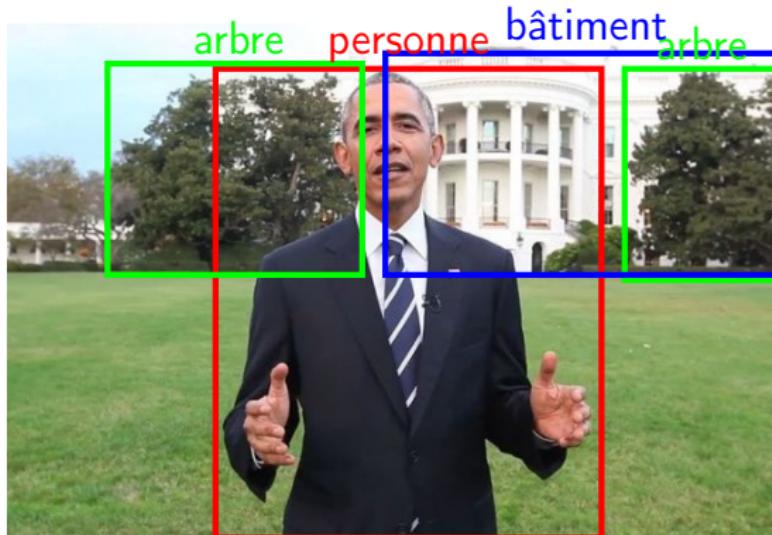


# Détection d'objet

A. Carlier

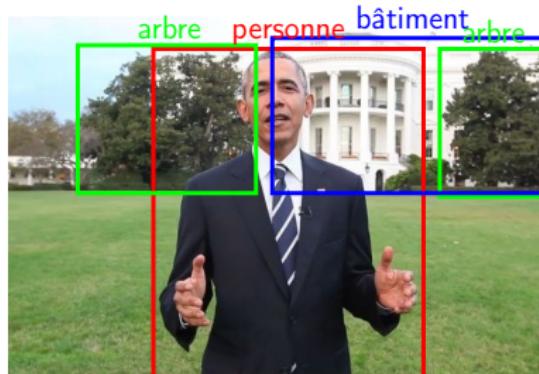
2025

# Détection d'objet



Objectif : délimiter la position d'objets multiples, avec éventuellement plusieurs instances, toujours à l'aide de boîtes englobantes.

# Détection d'objet



Difficultés de la détection d'objet :

- On ne connaît pas à l'avance le nombre d'objets présents sur l'image
- Comment encoder une boîte englobante ?
- Comment évaluer l'algorithme ?

# Plan du cours

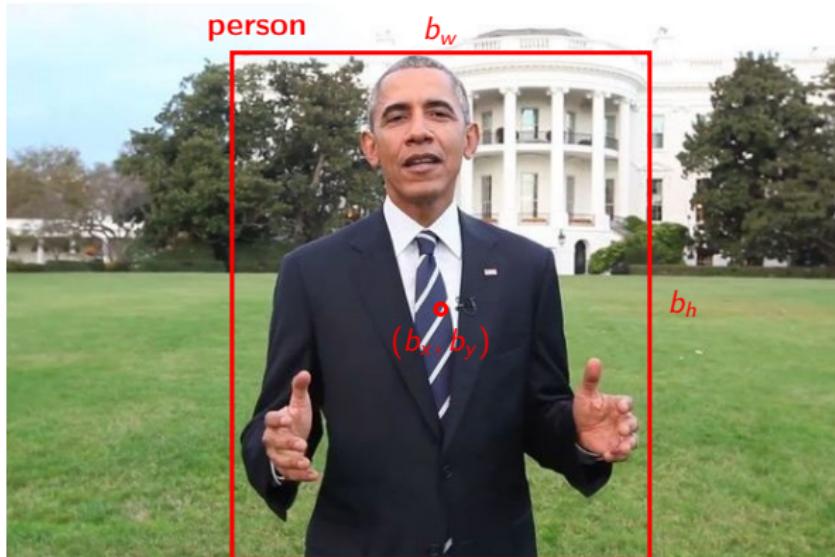
1 Localisation d'objet

2 Evaluation

3 Détection d'objet

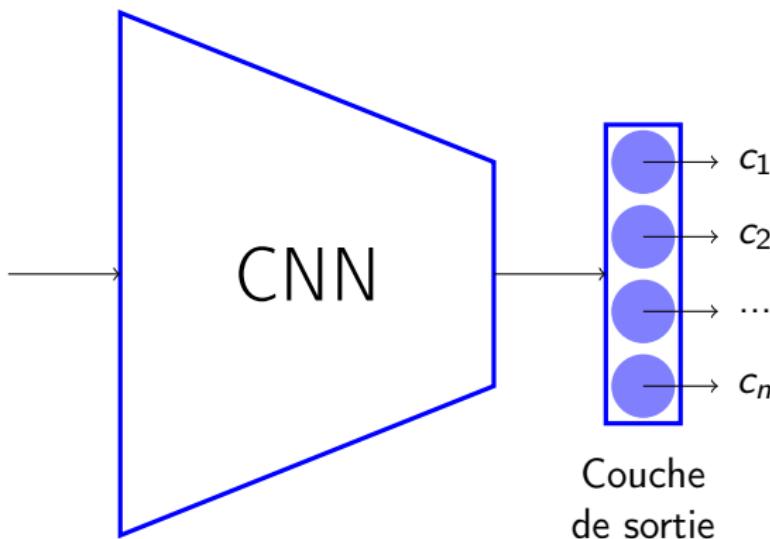
- Versions "naïves"
- La famille des R-CNN
- La famille des YOLO

# Classification et localisation conjointes



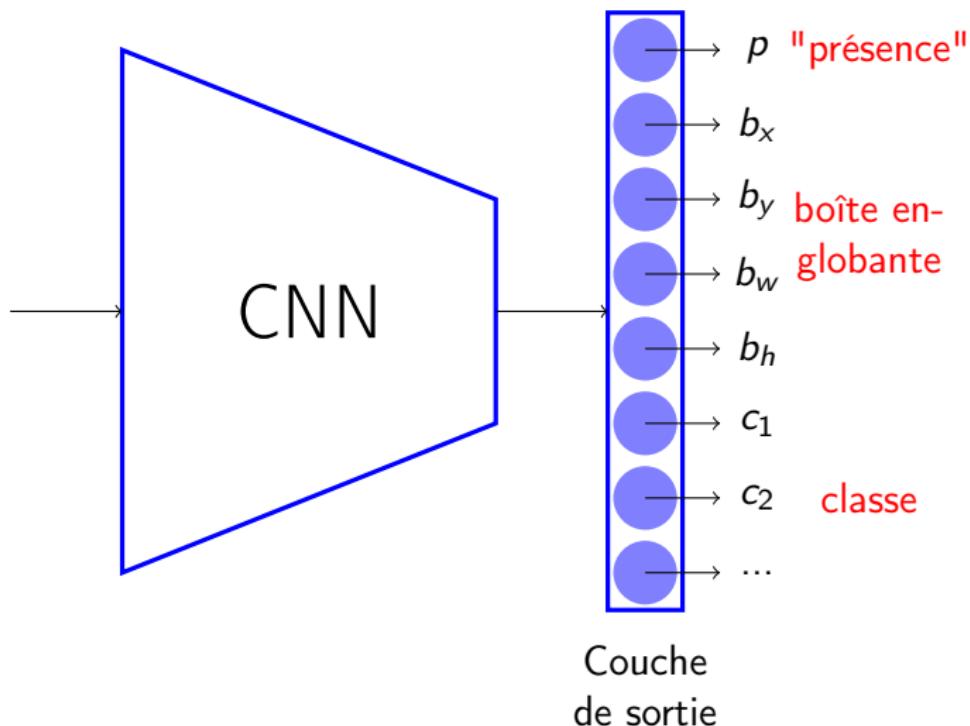
On commence par un problème simplifié : classifier et localiser **un seul objet** par image.

# Classification et localisation conjointes



Nous devons ajouter une information spatiale sur l'objet classifié.

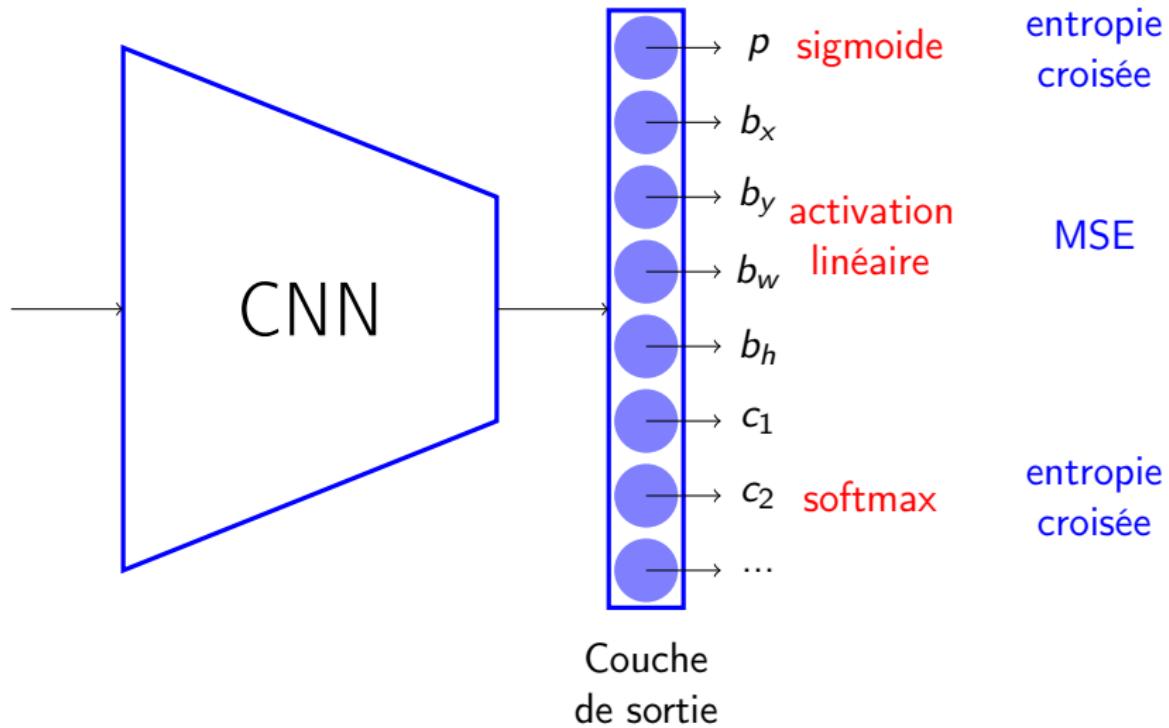
# Classification et localisation conjointes



Nous pouvons rajouter les coordonnées de la boîte englobante autour de l'objet.

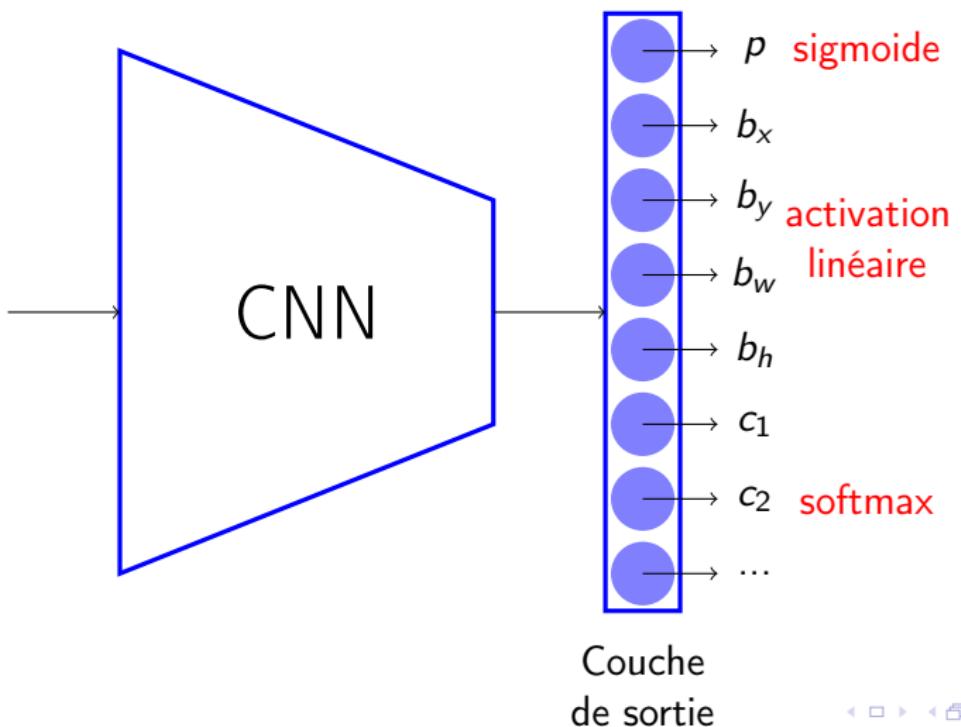
# Entraînement du modèle

Quelle fonction de coût ?



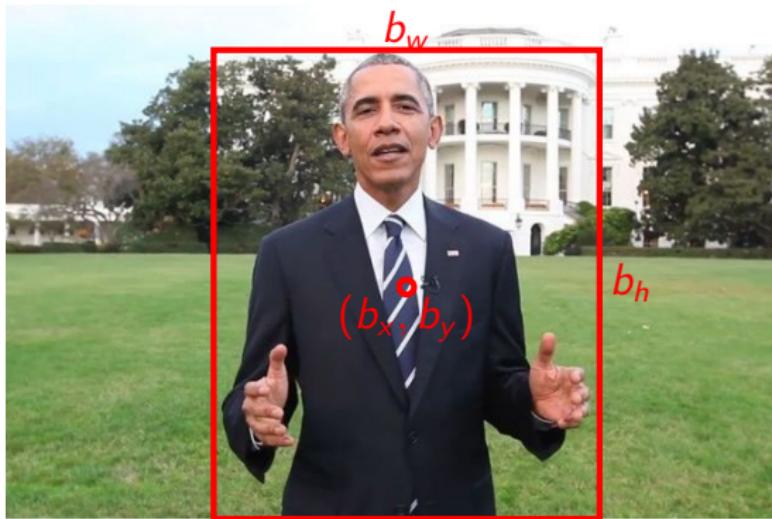
# Entraînement du modèle

Quelle fonction de coût ?



$$\begin{aligned}\mathcal{L} = & \text{entropie croisée} \quad \lambda_1 \mathcal{L}_1 \\ & + \\ & \text{MSE} \quad \lambda_2 \mathcal{L}_2 \\ & + \\ & \text{entropie croisée} \quad \lambda_3 \mathcal{L}_3\end{aligned}$$

## Format des labels



$$\hat{y} = \begin{pmatrix} 1 \\ 0.52 \\ 0.46 \\ 0.5 \\ 0.91 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{pmatrix}$$

# Plan du cours

1 Localisation d'objet

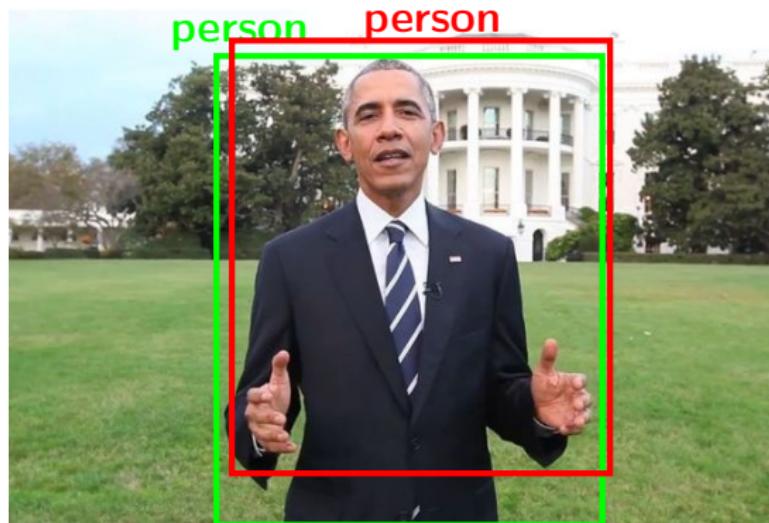
2 Evaluation

3 Détection d'objet

- Versions "naïves"
- La famille des R-CNN
- La famille des YOLO

# Evaluation

Qu'est-ce qu'une bonne détection ? On cherche à comparer une **prédition** à une **vérité-terrain**.



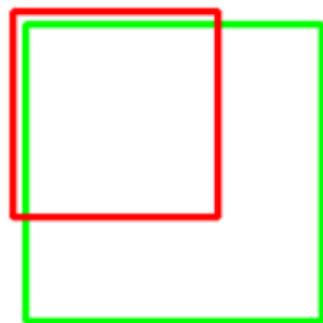
Une bonne prédition doit être bien localisée **et** bien reconnue.

## Intersection sur Union

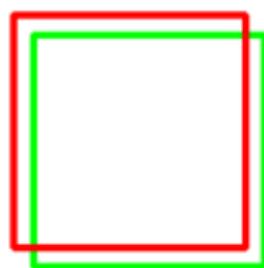
L'IoU (*Intersection over Union*) est une mesure classique de la qualité de la localisation :

$$\text{IoU}(R_1, R_2) = \frac{\mathcal{A}(R_1 \cap R_2)}{\mathcal{A}(R_1 \cup R_2)}$$

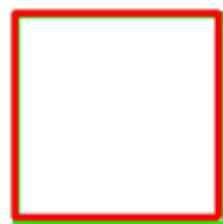
IoU: 0.4034



IoU: 0.7330



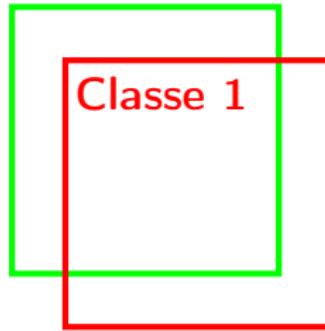
IoU: 0.9264



# Bonne détection : Vrai Positif (TP)

Prédiction et Vérité-Terrain.

Classe 1

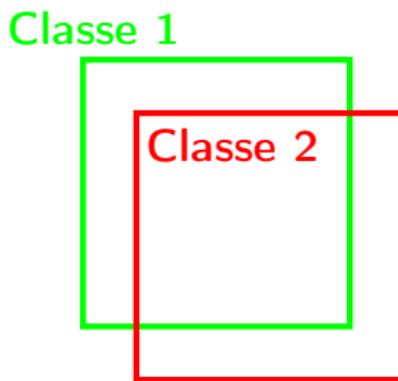


Classe 1 = Classe 1 et IoU > 0.5

On compte un Vrai Positif pour la classe 1.

# Erreurs de détection

Prédiction et Vérité-Terrain.



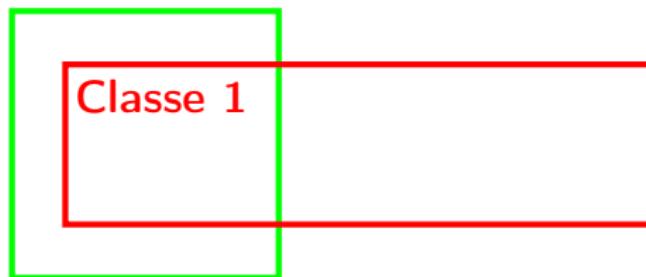
$\text{Classe 1} \neq \text{Classe 2}$  et  $\text{IoU} > 0.5$

On compte un Faux Positif pour la classe 2 et un Faux Négatif pour la classe 1.

# Erreurs de détection

Prédiction et Vérité-Terrain.

Classe 1



Classe 1 = Classe 1 et IoU < 0.5

On compte un Faux Positif pour la classe 1 et un Faux Négatif pour la classe 1.

# Précision, Rappel

On peut ainsi définir les deux métriques complémentaires de la Précision ( $P$ ) et du Rappel ( $R$ ) :

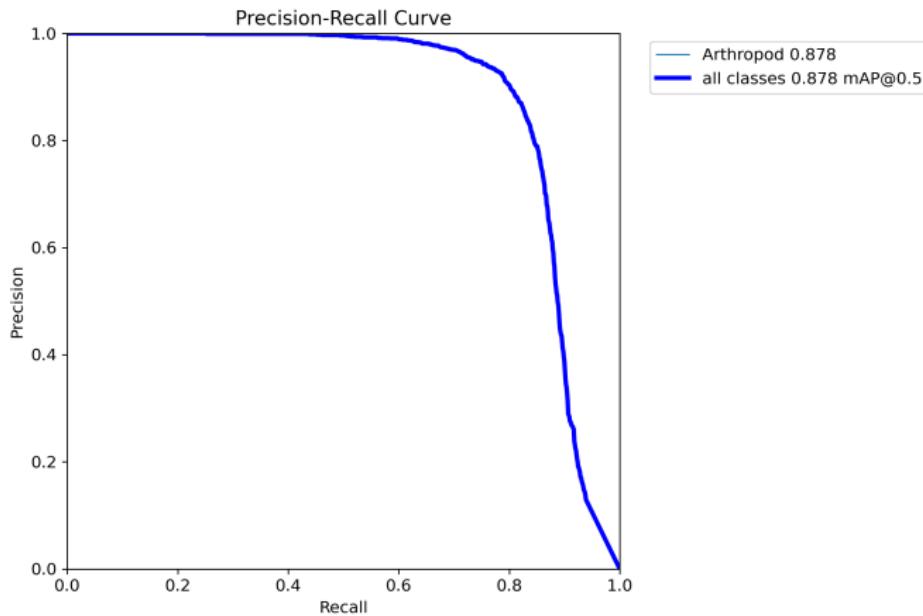
$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

La Précision désigne, parmi les objets identifiés par le détecteur, le ratio d'objets corrects.

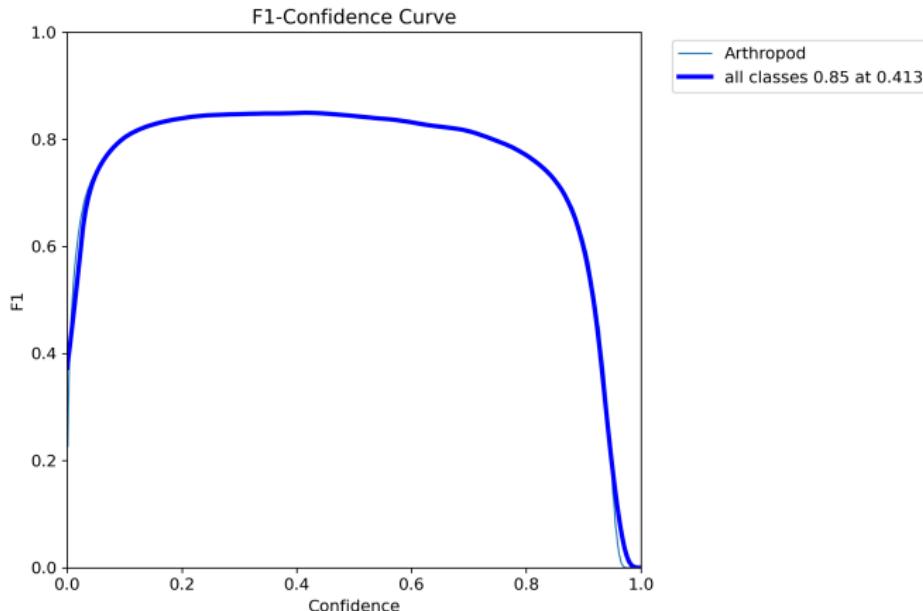
Le Rappel désigne, parmi les objets qu'il aurait fallu identifier, le ratio réellement identifié par le détecteur.

# Courbe Precision-Rappel pour un détecteur



On fait varier le seuil de confiance en deça duquel on élimine les détections du modèle, et on calcule à chaque fois la précision et le rappel obtenus.

## Une représentation équivalente avec le F1-Score



$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

# Plan du cours

1 Localisation d'objet

2 Evaluation

3 Détection d'objet

- Versions "naïves"
- La famille des R-CNN
- La famille des YOLO

# Plan du cours

1 Localisation d'objet

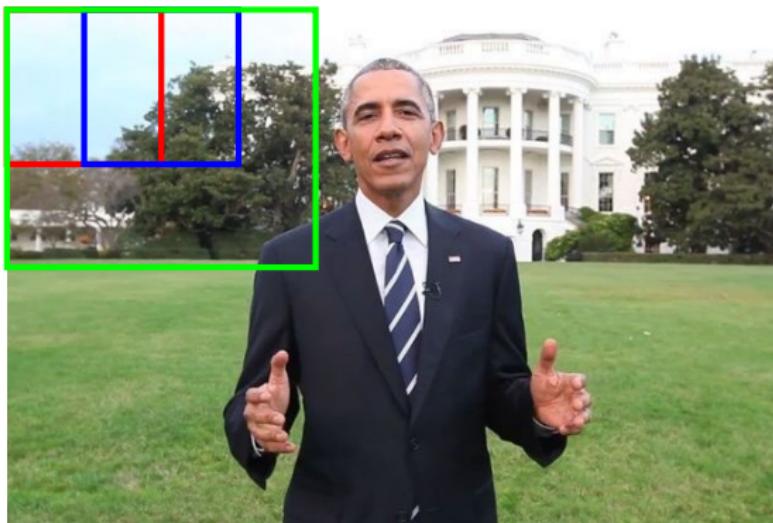
2 Evaluation

3 Détection d'objet

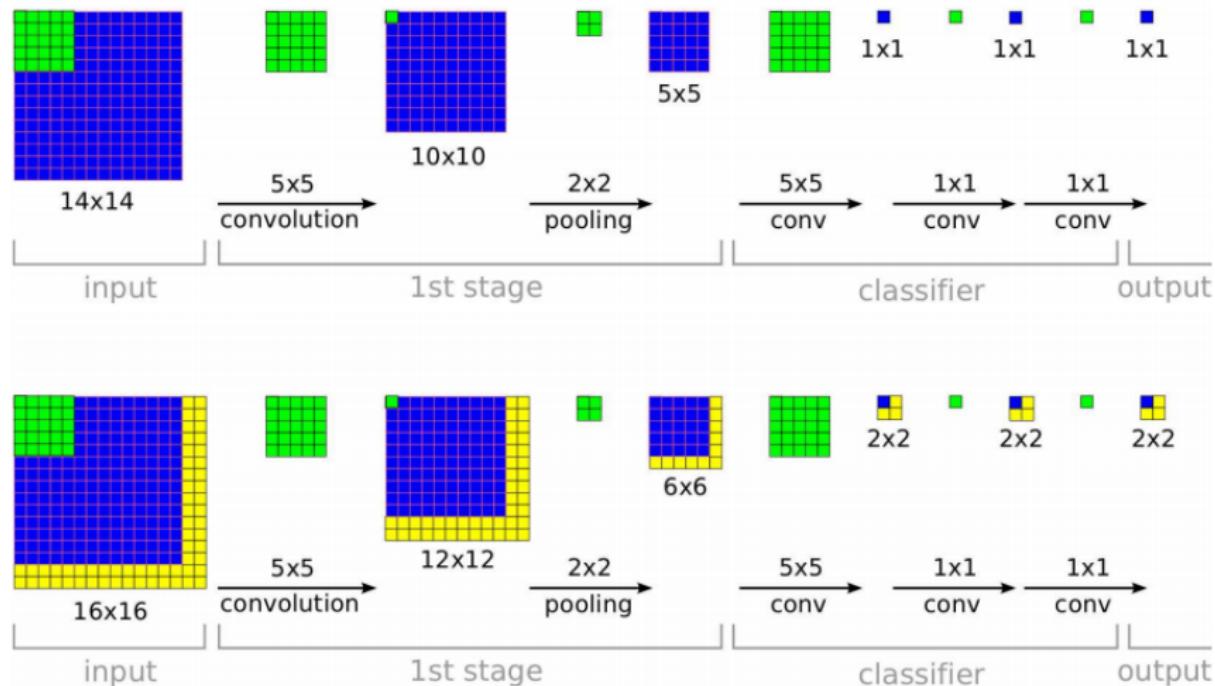
- Versions "naïves"
- La famille des R-CNN
- La famille des YOLO

# De la localisation à la détection d'objet

Une première idée : utiliser une fenêtre glissante multi-échelle pour localiser sur chaque sous-partie de l'image les objets présents.



# Fenêtre glissante convolutionnelle



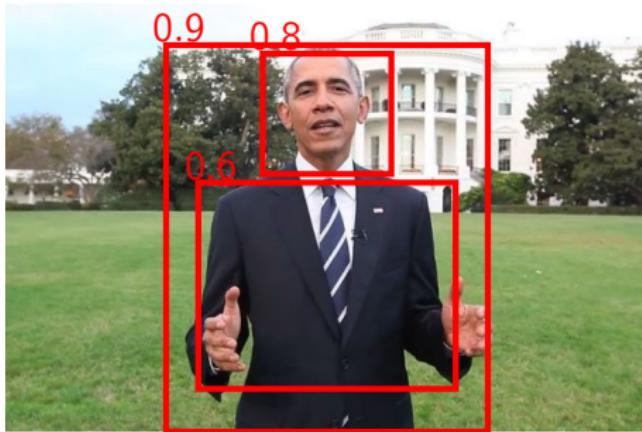
[Sermanet et al.] OverFeat : Integrated Recognition, Localization and Detection using Convolutional Networks

# Fenêtre glissante convolutionnelle



[Sermanet et al.] OverFeat : Integrated Recognition, Localization and Detection using Convolutional Networks

## Suppression des doublons (*Non-Max Suppression*)



On définit deux seuils : `IoU_threshold` et `confidence_threshold`.

- ① Suppression des boîtes avec un indice de confiance inférieur à `confidence_threshold`.
- ② Sélection parmi les boîtes restantes de la boîte  $b_{max}$  avec le plus haut indice de confiance.
- ③ Suppression de toutes les boîtes ayant une intersection sur union supérieure à `IoU_threshold` avec  $b_{max}$ .

# Plan du cours

1 Localisation d'objet

2 Evaluation

3 Détection d'objet

- Versions "naïves"
- La famille des R-CNN
- La famille des YOLO

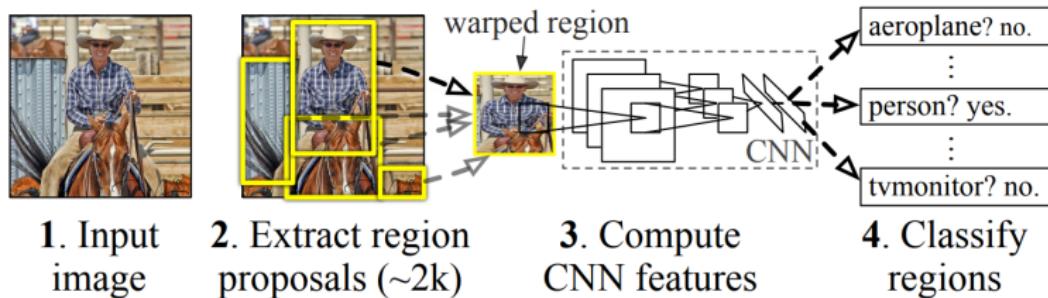
# Famille des R-CNN

Pour éviter les fenêtres glissantes : génération de régions candidates.



[Uijlings et al.] Selective Search for Object Recognition

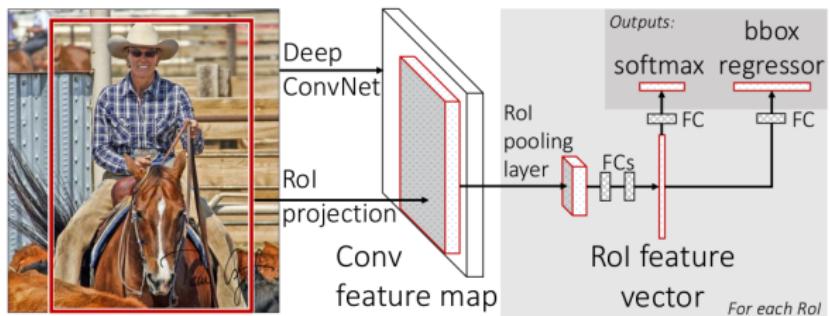
Génération des régions candidates et classification de ces régions.



Inconvénient de cette méthode : nécessité de faire de nombreuses prédictions par un CNN pour chaque image. **Très peu optimisé !**

[Girshick et al.] Rich feature hierarchies for accurate object detection and semantic segmentation

# Fast R-CNN



La partie droite est entraînée séparément, par-dessus la base convulsive, à l'aide d'une fonction de coût hybride :

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \quad (1)$$

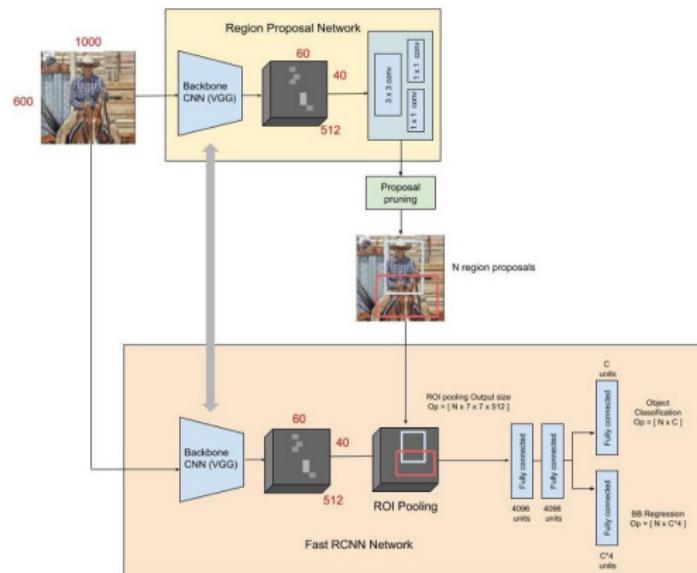
où

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x}, \text{y}, \text{w}, \text{h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

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

# Faster R-CNN

Un même réseau (poids partagés) pour générer des régions candidates et évaluer si elles contiennent un objet (plus rapide mais pas temps réel !)



[Ren et al.] Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks

## En pratique

### Detectron2 :

<https://github.com/facebookresearch/detectron2>

Faster R-CNN:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
<a href="#">R50-C4</a>	1x	0.551	0.102	4.8	35.7	137257644	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R50-DC5</a>	1x	0.380	0.068	5.0	37.3	137847829	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R50-FPN</a>	1x	0.210	0.038	3.0	37.9	137257794	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R50-C4</a>	3x	0.543	0.104	4.8	38.4	137849393	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R50-DC5</a>	3x	0.378	0.070	5.0	39.0	137849425	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R50-FPN</a>	3x	0.209	0.038	3.0	40.2	137849458	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R101-C4</a>	3x	0.619	0.139	5.9	41.1	138204752	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R101-DC5</a>	3x	0.452	0.086	6.1	40.6	138204841	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">R101-FPN</a>	3x	0.286	0.051	4.1	42.0	137851257	<a href="#">model</a>   <a href="#">metrics</a>
<a href="#">X101-FPN</a>	3x	0.638	0.098	6.7	43.0	139173657	<a href="#">model</a>   <a href="#">metrics</a>

# Plan du cours

1 Localisation d'objet

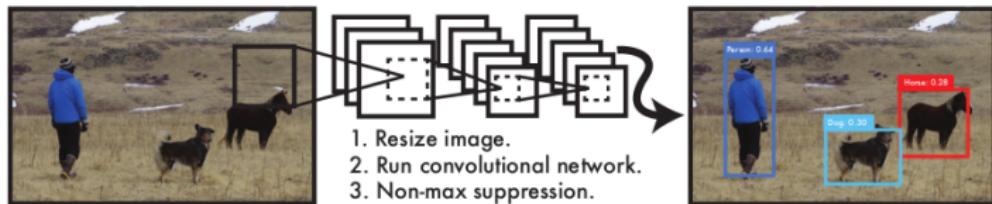
2 Evaluation

3 Détection d'objet

- Versions "naïves"
- La famille des R-CNN
- La famille des YOLO

# YOLO

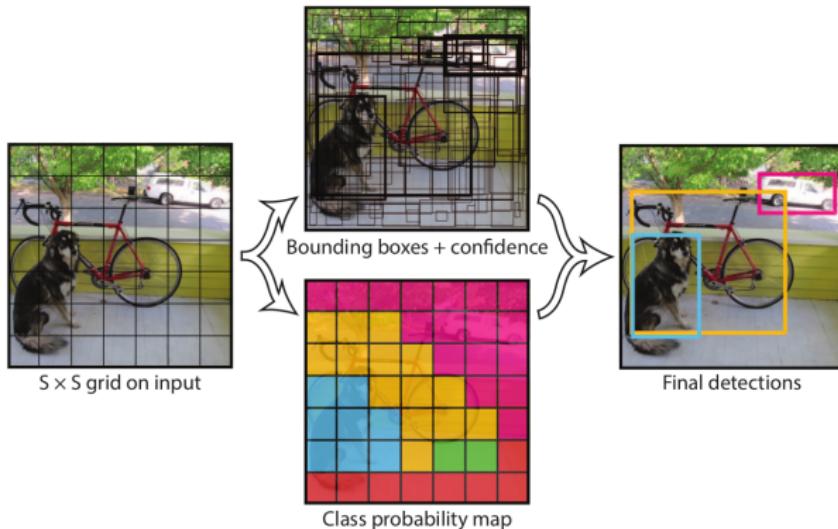
You Only Look Once = une seule inférence du réseau permet de générer toutes les boîtes en même temps.



**Problème :** Comment faire en sorte de générer un nombre de boîtes adapté (et potentiellement différent) pour chaque image ?

**Solution :** Générer trop de boîtes, puis filtrer !

# YOLO



YOLO divise l'image en  $S \times S$  cellules, et prédit  $B$  boîtes englobantes pour chaque cellule. Une boîte englobante est définie par 4 coordonnées et un indice de confiance.

[Redmon et al.] You Only Look Once : Unified, Real-Time Object Detection

## Indice de confiance

L'indice de confiance  $c_{\mathcal{B}_p}$  d'une boîte englobante prédite  $\mathcal{B}_p$  est défini comme suit :

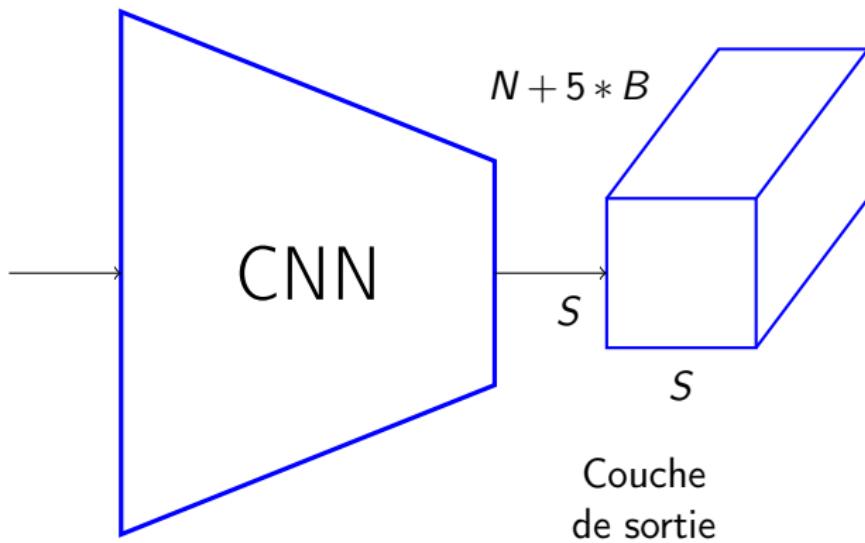
$$c_{\mathcal{B}_p} = \mathbb{1}^{\text{obj}} \text{ IOU}(\mathcal{B}_p, \mathcal{B}_{GT})$$

Ainsi, s'il n'y a pas d'objet dans la cellule contenant  $\mathcal{B}_p$ ,  $c_{\mathcal{B}_p} = 0$ .

S'il y a un objet dans la cellule, alors  $c_{\mathcal{B}_p} = \text{IOU}(\mathcal{B}_p, \mathcal{B}_{GT})$ .

On entraîne ainsi le modèle à détecter des objets mais aussi à prédire avec quelle précision il a détecté un objet.

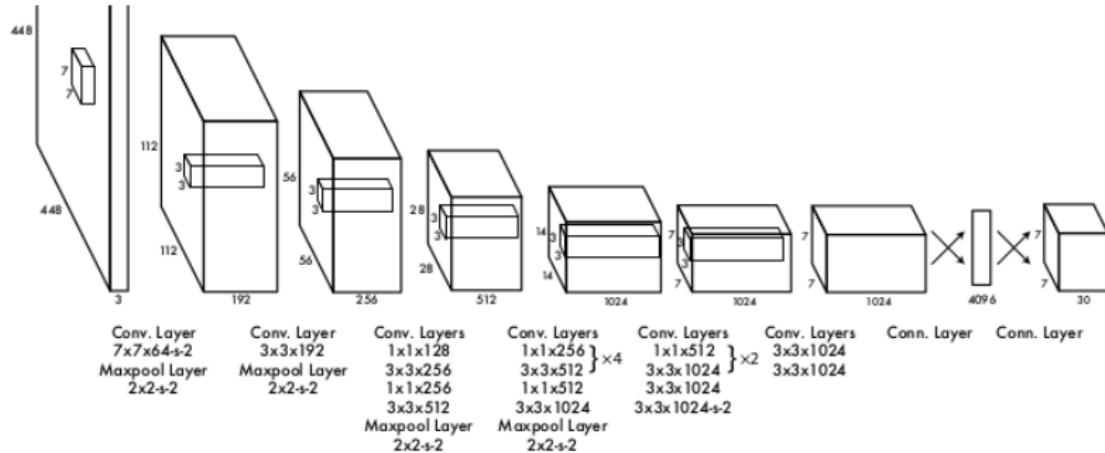
# YOLO



où :

- $S$  est la taille de la grille (sur PASCAL VOC,  $S = 7$ )
- $B$  est le nombre d'objets détectés par cellule (PASCAL :  $B = 2$ )
- $N$  est le nombre de classes (PASCAL :  $N = 20$ )

# Darknet



Architecture librement inspirée d'Inception !

# YOLO : fonction de perte

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad \text{coord loss} \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2 \\ & \text{no box loss} \quad + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2 \\ & \text{box loss} \quad + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \quad (3)$$

Une erreur quadratique moyenne sur tous les termes !

# Le problème des petits objets



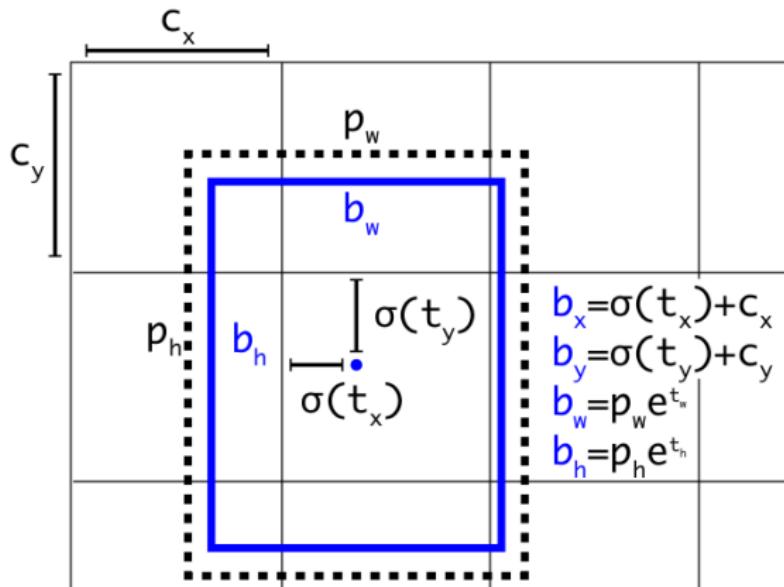
# YOLO v2

De nombreuses améliorations sont réalisées, par exemple :

- Utilisation de "modèles" de boîtes englobantes (*anchor boxes*).
- Modification de la manière d'encoder les boîtes englobantes.
- Prédictions de probabilités de classe différentes pour chaque boîte englobante
- Entraînement multi-échelle.
- Optimisations sur l'architecture Darknet.

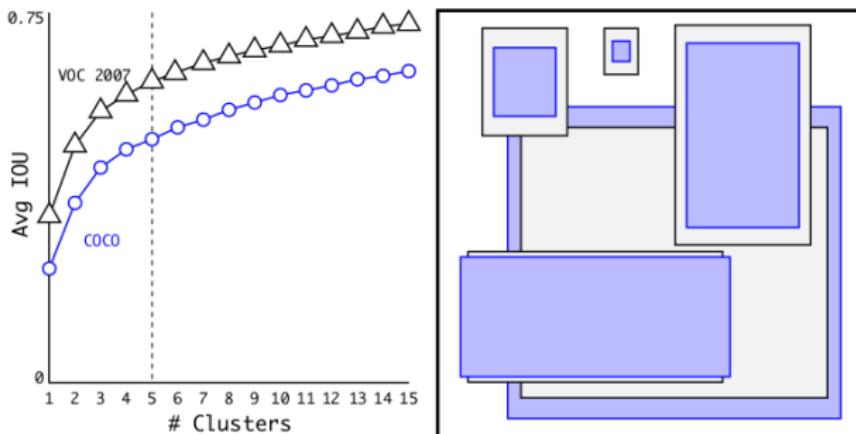
[Redmon et al.] YOLO9000 : Better, Faster, Stronger

## Encodage basé ancrages



$t_x, t_y, t_w, t_h$  peuvent prendre n'importe quelle valeur réelle, le problème est plus facile à apprendre pour le modèle.

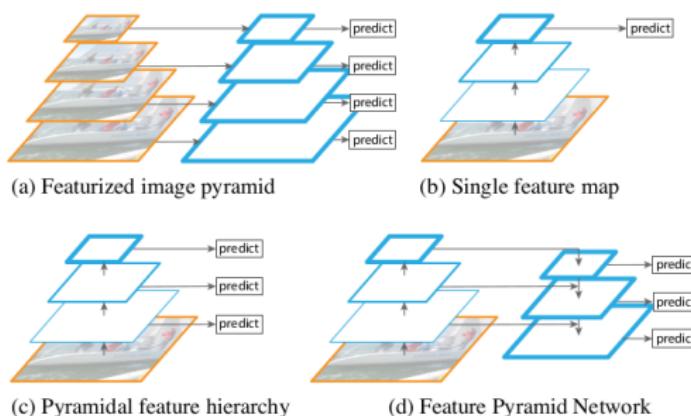
# Choix des ancre



Algorithme des  $k$ -moyennes en utilisant l'IOU pour mesurer la similarité entre les boîtes.

# YOLOv3

- Darknet-53, un ResNet-like !
- Remplacement de softmax par sigmoid pour la prédiction des classes (utile pour certains datasets où les classes ne sont pas mutuellement exclusives)
- L'utilisation de Feature Pyramid Networks résoud en grande partie le problème des petits objets.



# YOLOv4 en détail

## YOLOv4 consists of:

- Backbone: CSPDarknet53 [81]
- Neck: SPP [25], PAN [49]
- Head: YOLOv3 [63]

## YOLO v4 uses:

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing
- Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multi-input weighted residual connections (MiWRC)
- Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler [52], Optimal hyperparameters, Random training shapes
- Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

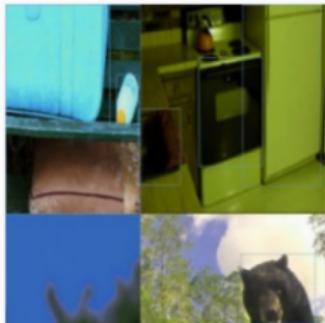
# Un nouveau type d'augmentation : mosaïques



aug\_-319215602\_0\_-238783579.jpg



aug\_-1271888501\_0\_-749611674.jpg



aug\_1462167959\_0\_-1659206634.jpg



aug\_1474493600\_0\_-45389312.jpg



aug\_1715045541\_0\_603913529.jpg



aug\_1779424844\_0\_-589696888.jpg

## Et après ?

- ① **YOLOv5** : <https://github.com/ultralytics/yolov5>
  - ▶ **YOLOR** Wang et al. 2021 : You Only Learn One Representation : Unified Network for Multiple Tasks)
  - ▶ **YOLOv7** Wang et Bochkovskiy 2022 : YOLOv7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors
- ② **YOLOX** Ge et al. 2021 YOLOX : Exceeding YOLO Series in 2021
- ③ **YOLOv6** Li et al. 2022 YOLOv6 : A Single-Stage Object Detection Framework for Industrial Applications

**YOLOv8** : <https://github.com/ultralytics/yolov8>

**YOLOv11** :

<https://github.com/ultralytics/ultralytics>

# YOLOv11



## Performance

[Détection \(COCO\)](#)[Segmentation \(COCO\)](#)[Classification \(ImageNet\)](#)[Pose \(COCO\)](#)[OBB \(DOTAv1\)](#)

Voir [Detection Docs](#) pour des exemples d'utilisation de ces modèles formés sur [COCO](#), qui comprennent 80 classes préformées.

Modèle	taille (pixels)	mAPval 50-95	Vitesse CPU ONNX (ms)	Vitesse T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	$56.1 \pm 0.8$	$1.5 \pm 0.0$	2.6	6.5
YOLO11s	640	47.0	$90.0 \pm 1.2$	$2.5 \pm 0.0$	9.4	21.5
YOLO11m	640	51.5	$183.2 \pm 2.0$	$4.7 \pm 0.1$	20.1	68.0
YOLO11l	640	53.4	$238.6 \pm 1.4$	$6.2 \pm 0.1$	25.3	86.9
YOLO11x	640	54.7	$462.8 \pm 6.7$	$11.3 \pm 0.2$	56.9	194.9