DETECTION D'OBJETS AVEC L'ALGORITHME YOLOX

Francesco Loiudice

---

1. Introduction aux algorithmes de détection d'objets

L'objectif du projet est l'amélioration du modèle de classification d'image du projet 6 à l'aide de l'algorithme de détection d'objet YOLOX. L'objectif des algorithmes de classification d'images est d'obtenir la classe d'appartenance d'un objet dans une image. Si on a plusieurs objets de classes différentes dans l'image on peut entrainer un classifieur multilabel mais on ne peut pas localiser les objets sur l'image.

L'objectif des algorithmes de détection d'objet est donc de localiser les objets dans une image et indiquer leur position à l'aide de 'Bounding Boxes' et ensuite de classifier chaque objet dans chaque Bounding Box.

Les architectures de réseaux de neurones pour la détection d'objets se divisent en deux catégories :

- Multi-stage detectors : détecteurs avec deux stages séparés pour l'identification des ROI (regions of interest) et la classification et la localisation des objets

	Description	mAP [%] [1]	Speed [s] [2]
RCNN (Region-based Convolutional Neural Network)	• Selective Search Algorithm, basé sur des techniques de segmentation, pour identifier les ROI (Regions Of Interest) • CNN pour extraction des features de chaque région • Classifieur SVM pour classification des objets • Modèle de régression pour prédiction des Bounding Boxes	58.5 - 66	9.8 - 47
Fast RCNN	• CNN pour extraction des features de l'image • Selective Search Algorithm pour identifier les ROI (Regions Of Interest) • Classification et prédiction des Bounding Boxes avec NN	57.1 - 66.9	0.1 – 0.32
Faster RCNN	• CNN pour extraction des features de l'image • RPN (Region Proposals Network) avec Anchor Boxes pour prédiction de la présence d'un objet dans chaque Box • Classification et optimisation des dimensions avec NN	76.1	0.1

- Single stage detectors : détecteurs avec identification des Bounding Boxes et classification en parallèle

	Description	mAP [%] [1]	Speed [s] [2]
SSD (Single-Stage-Detectors)	• Extraction des features de l'image avec VGG16 • Ajoute couches convolutionnelles et extraction de 4 ou 6 Anchor Boxes pour chaque pixel • Classification et non maximum suppression	74.3	0.016
YOLO (You Only Look Once)	• Extraction features maps de l'image avec Darknet (Googlenet architecture) • Transformation de l'image en Grid cells (SxS) • Prédiction probabilité présence d'un objet, Classification (C classes) et localisation (4 coordonnées) des Bounding Box (B) avec un tenseur de dimension SxSx(Bx5 +C) en sortie de Fully connected layers	63.4	0.022

[1] Evaluation avec VOC Pascal 2007 dataset de petits à larges objets
[2] Evaluation avec VOC Pascal 2007 dataset larges objets

La première version de l'algorithme YOLO a été ensuite amélioré en termes de précision et vitesse avec plusieurs versions suivantes.

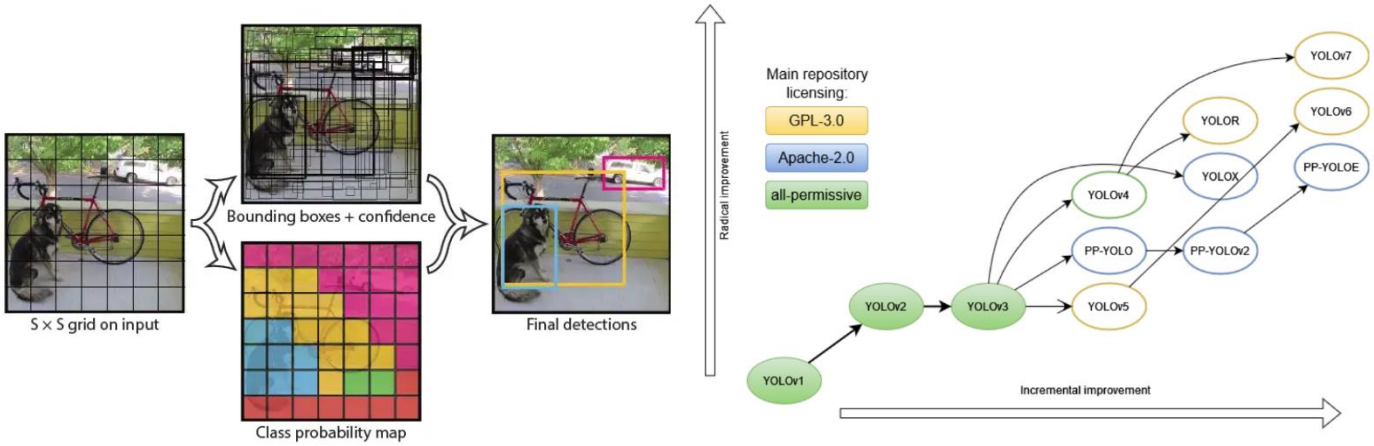


Fig 2. Picture from the original paper: https://paperswithcode.com/method/yolov1

2. YOLOX

La version YOLOX de l'algorithme est basée sur l'architecture YOLO vs3 et introduit les améliorations suivantes :

Methods	AP (%)	Parameters	GFLOPs	Latency	FPS
YOLOv3-ultralytics[2]	44.3	63.00 M	157.3	10.5 ms	95.2
YOLOv3 baseline	38.5	63.00 M	157.3	10.5 ms	95.2
+decoupled head	39.6 (+1.1)	63.86 M	186.0	11.6 ms	86.2
+strong augmentation	42.0 (+2.4)	63.86 M	186.0	11.6 ms	86.2
+anchor-free	42.9 (+0.9)	63.72 M	185.3	11.1 ms	90.1
+multi positives	45.0 (+2.1)	63.72 M	185.3	11.1 ms	90.1
+SimOTA	**47.3** (+2.3)	63.72 M	185.3	11.1 ms	90.1
+NMS free (optional)	46.5 (-0.8)	67.27 M	205.1	13.5 ms	74.1

Roadmap of YOLOX-Darknet53 in terms of AP (%) on COCO val.

2.1. Decoupled head

YOLOX utilise l'architecture Darknet-53 (backbone) de YOLO vs3 avec 53 couches convolutionnelles avec 1×1 convolutions, residual connections, and 3×3 convolutions afin d'obtenir un feature extractor très performant.
Le modèle consiste aussi d'un layer **FPN (Feature Pyramid Network)** (neck) qui extrait les informations de l'image avec feature maps de dimensions différentes obtenues à partir de trois sorties intermédiaires du modèle avec trois différentes prédictions à échelles différentes :

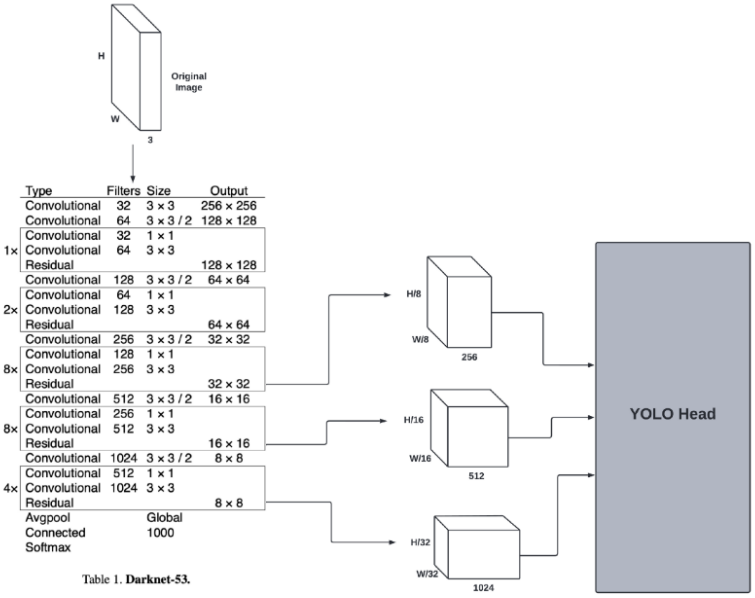
1. 256 feature maps (first transition output)
2. 512 feature maps (second transition output)
3. 1024 feature maps (third transition output)

Donc le nombre de filtres pour les trois sorties augmente avec la réduction de la dimension des features maps qui extraient détails des images à plus grande échelle.

L'input pour YOLOX Head sont les trois sorties du FPN avec les trois échelles de 1024, 512 et 256 channels.

L'output de YOLOX pour chaque input provenant du layer FPN sont trois tenseurs (différemment de YOLO vs3 qui a un seul tenseur en sortie) avec différentes informations et avec la structure suivante :

1. Cls: sortie avec la classe de l'objet dans chaque Bounding Box
2. Reg: sortie avec la localisation da la Bounding Box (x, y, w, h)
3. IoU (Obj): sortie avec la probabilité de présence d'un objet dans le Bounding Box



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

The Darknet backbone and its three outputs

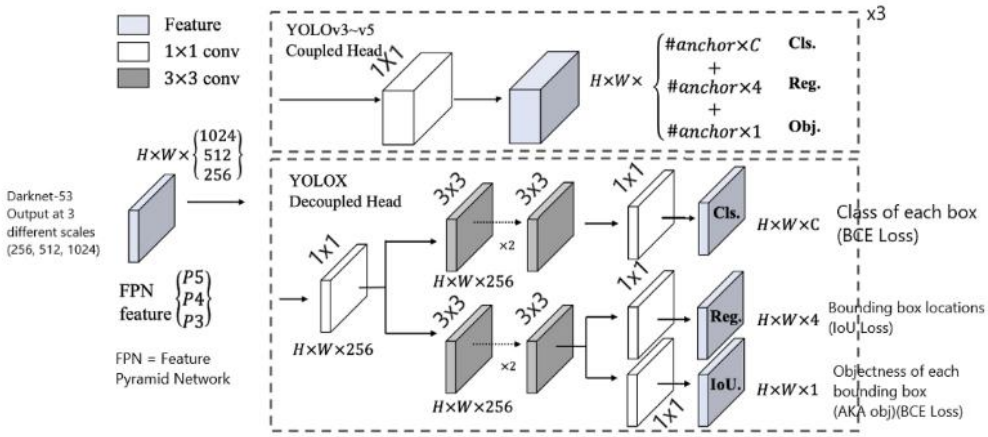
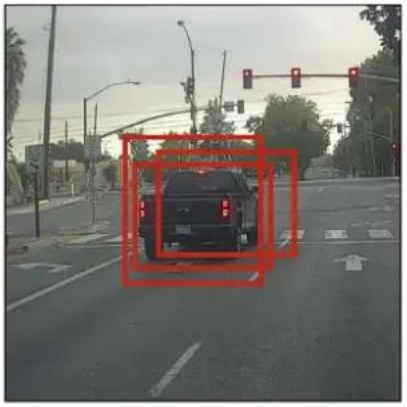


Figure 2: Illustration of the difference between YOLOv3 head and the proposed decoupled head. For each level of FPN feature, we first adopt a 1×1 conv layer to reduce the feature channel to 256 and then add two parallel branches with two 3×3 conv layers each for classification and regression tasks respectively. IoU branch is added on the regression branch.

Chaque élément des matrices de sortie HxW pour feature map de l'FPN est une prédiction, donc on a 9 total outputs (3 matrices de sortie pour Cls, 3 matrices de sortie pour Reg et 3 matrices de sortie pour IoU). Lors de la prédiction on aura donc beaucoup de Bounding Boxes et la sélection est faite en deux étapes avec tout d'abord la suppression des sorties avec un confidence score (objectness) inferieur à une certaine seuil et ensuite avec la tecnique Soft-NMS (Soft-Non Max Suppression) qui supprime les prédictions avec une très grande superposition.

Avant Soft-NMS



Non-Max Suppression

Après Soft-NMS

2.1.1. Loss functions

Chaque sortie du modèle YOLOX a une fonction de perte différente afin de permettre une différente optimisation :

- Class optimization
- Regression optimization
- IoU/Objectness Loss

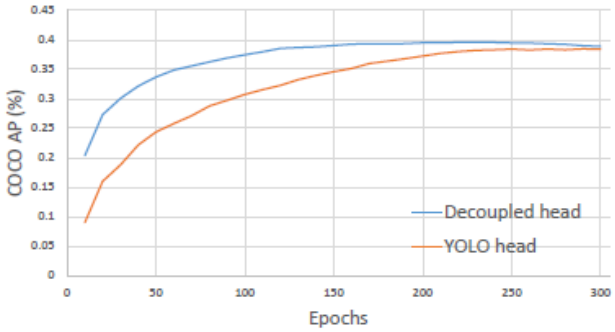
La fonction de perte finale est la combinaison des moyennes des trois fonctions de perte pour les prédictions avec étiquette positive :

$$L = \frac{1}{N_{pos}} L_{cls} + reg_{weight} * \frac{1}{N_{pos}} L_{reg} + \frac{1}{N_{pos}} L_{obj}$$

Final Loss function for YOLOX

où reg weight est un terme de balancing pour optimiser la Regression loss (valeur utilisé : 0.5).

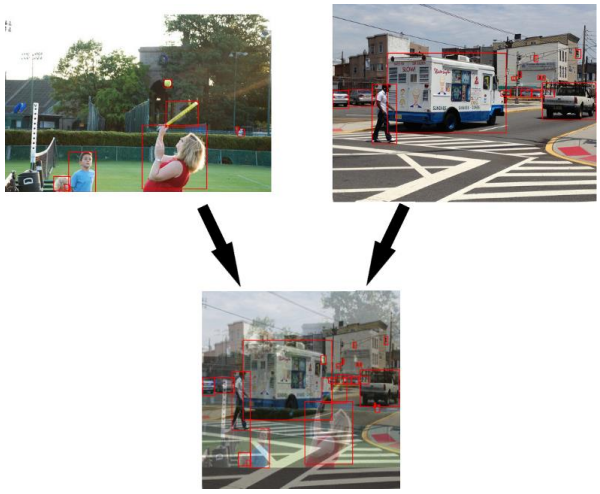
L'utilisation d'une Decoupled head améliore la vitesse de convergence de l'algorithme.



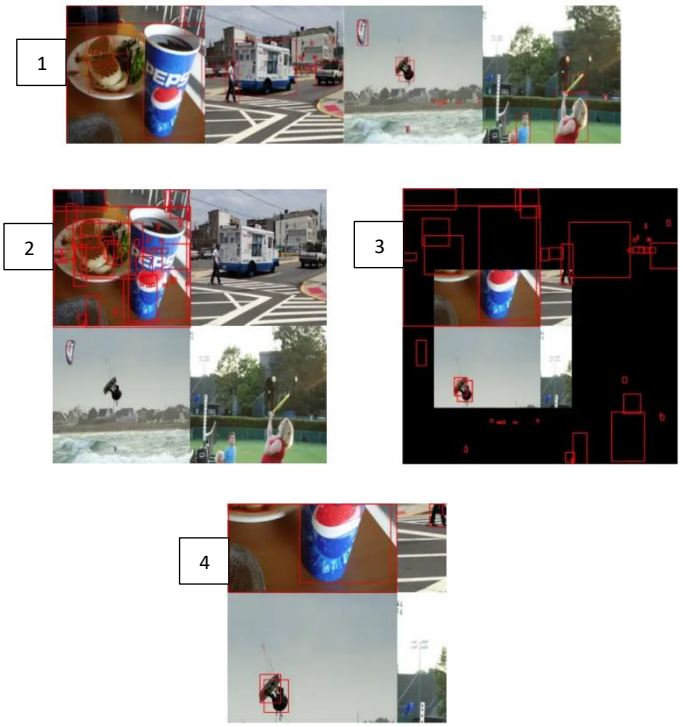
2.2. Strong data augmentation

YOLOX ajoute deux stratégies pour la data augmentation : Mosaic et MixUp.
La technique Mixup augmentation est une stratégie qui génère des images à partir de la somme pesée de deux images avec une combinaison de tous les Bounding Boxes.
La technique Mosaic fait un collage et ensuite un Random cropping de quatre images du jeu de training avec adaptation des Bounding Boxes. Cette technique aide le modèle pour la détection d'objets de taille réduite.

Mixup Augmentation



Mosaic Augmentation



2.3. Anchor-free model

2.3.1. Versions YOLO précédentes

Les versions de YOLO précédentes à YOLOX prédisent le box pour la détection d'objet sur la base d'un offset à partir d'un box prédéfinis nommé « Anchor Box » choisi entre plusieurs boxes de tailles différentes disponibles pour chaque élément de la grille et superposés à l'image.

Une limite des Anchor Boxes est la nécessité de définir des hyperparamètres additionnels comme le nombre de Anchors boxes à disposer sur l'images et leur taille.

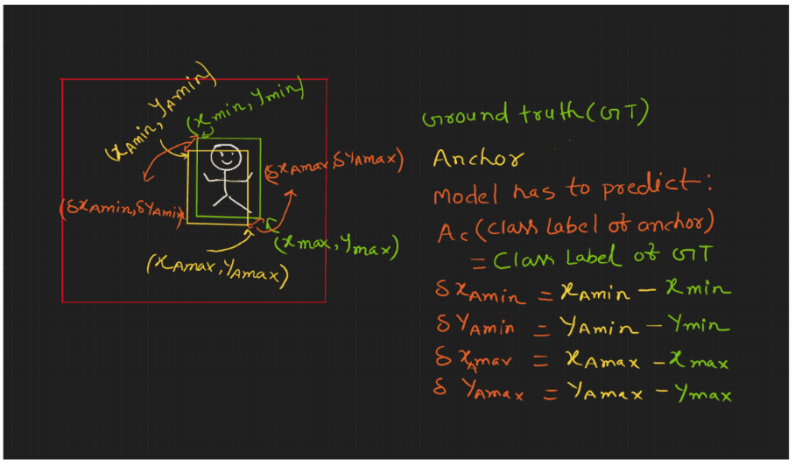


fig. 3: Anchor-based Object Detection Model Prediction

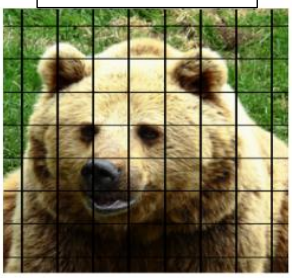
2.3.2. YOLOX

YOLOX a l'avantage de prédire directement la taille de chaque Bounding Box plutôt que comme offset par rapport à une Anchor Box et sa position par rapport à un points de référence fixe sur l'image.

L'image est divisée avec trois grilles sur la base des trois échelles différentes utilisées pour les prédictions et chaque élément des trois matrices de sortie peut être directement lié à un point unique sur la grille qui ensuite sera utilisé comme offset pour déplacer le Bounding Box. On définit « stride » la distance en termes de nombre de pixels entre deux points d'intersection consécutives sur la grille. Les points d'intersections sur la grille s'appellent « Anchors points ». YOLOX utilise donc trois « strides » différentes : 32, 16, 8 :

Image 256 x 256

64 prédictions



Level 1
(Stride of 32)

256 prédictions

Level 2
(Stride of 16)

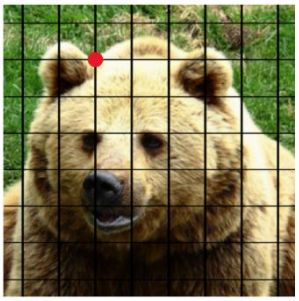
1024 prédictions

Level 3
(Stride of 8)

Bear Grids

Avec l'utilisation des grilles le nombre de prédictions du modèle se réduit énormément car elles sont faites seulement pour les Anchors points et pas pour tous les pixels de l'image.

x = s/2 + s*i x,y = Anchor-point, s = stride
y = s/2 + s*j i = ith intersection point on the x-axis j = jth intersection point on the y-axis



Bear Grid with a point at (2, 1) assuming index starts at 0 and (0, 0) is the top left corner

Pour la prédiction les Anchor points sont utilisée comme offsets des Bounding Boxes.

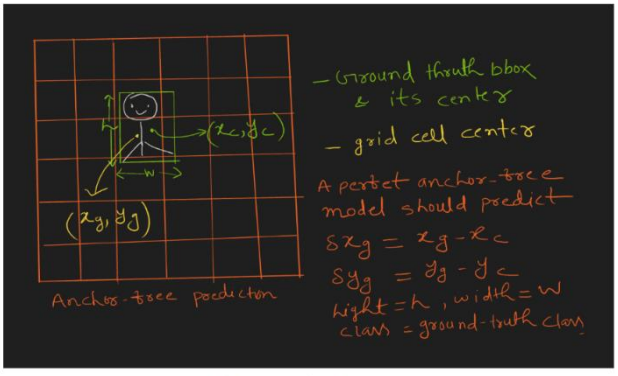
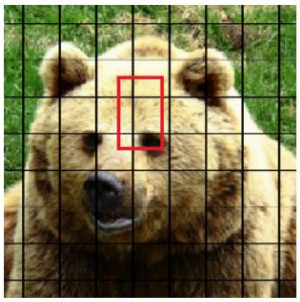


fig. 4: Anchor free Prediction

Les Bounding Boxes prédites sont ensuite transférées sur l'images initiale à l'aide des équations suivantes :

$$
\begin{aligned}
l\_x &= p\_x + x \\
l\_y &= p\_y + y \\
l\_w &= s*e^{(p\_w)} \\
l\_h &= s*e^{(p\_h)}
\end{aligned}
$$

p_x, p_y = prédiction du centre du box
p_w, p_h = prédiction des dimensions du box



Bear Grid with bounding box

2.4. SimOTA

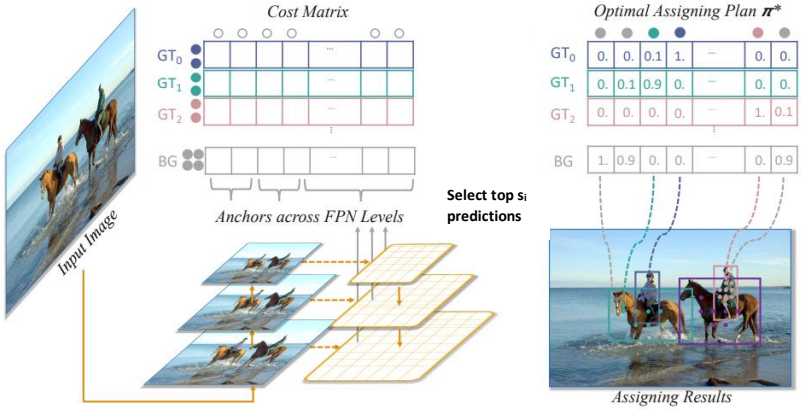
L'algorithme SimOTA est utilisé pendant l'entrainement pour faire la différence entre bonnes et mauvaises prédictions à l'aide aussi des techniques « Dynamic label assignment » et « Center prior ».

Les tâches de l'algorithme SimOTA sont les suivantes :

1) Attribution d'étiquettes positives aux prédictions qui appartiennent à un GT (Ground Truth Object)
2) Attribution d'étiquettes négatives aux prédictions qui appartiennent au background
3) Attribution des Ground Truth Bounding Boxes (GT) à chaque Anchor point avec étiquette positive dans l'image

Algorithme :

1. Assigner m et n as comme nombre de GT (Ground Truths) and nombre de Anchor points
2. Obtenir la prédiction du modèle pour la classe P^{cls} et pour le box P^{box}
3. Créer le supplying vector s avec m +1 valeurs. Utiliser la technique de « dynamic k estimation » pour obtenir les valeurs pour chaque GT et remplir le vecteur : on sélectionne les q prédictions avec les valeurs IoU plus élevées, on les additionne et on utilise la valeur obtenue comme k value pour le nombre de Anchor points à assigner aux GT.
4. La valeur pour le background est s[m+1] = n — sum(s).
5. Evaluer la perte Cls loss pour la couple de chaque prédiction jth et la i-eme GT : $c^{cls} = FocalLoss(P^{cls}, G^{cls})$
6. Evaluer la perte Reg loss pour la couple de chaque prédiction jth et la i-eme GT : $c^{reg} = IoULoss(P^{box}, G^{box})$
7. Evaluer la perte additionnelle liée à l'application de la technique « center prior » qui consiste à définir une perte additionnelle pour la couple de chaque prédiction jth et la i-eme GT : $c^{cp} = CenterPrior(A_j, G^{box})$ pour les Anchor points avec une distance supérieur à r^2 par rapport au centre du i-ème GT Box
8. Evaluer la perte pour le background : $c^{bg} = FocalLoss(P^{cls}, \emptyset)$
9. Evaluer la perte « foreground » : $c^{fg} = c^{cls} + \alpha c^{reg} + c^{cp}$
10. Evaluer la cost matrix c finale de dimension (m+1, n) comme union de c^{bg} et c^{fg}
11. Itérer pour tous les éléments de s et évaluer les meilleures prédictions pour s_i avec les pertes c_i plus basses.
12. La matrice obtenue a m valeurs et, pour chaque m_i, maximum s_i prédictions.



Après avoir exécuté SimOTA, la sortie sera un tableau de taille m où chaque élément dans le tableau résultant est un Anchor point prédit avec une étiquette positive correspondante au GT object G_i. Les autres prédictions qui ne figurent pas dans le tableau obtenu sont considérées comme des prédictions négatives sans affectation de GT.

2.5. Approche Multi-Positives

Avec les méthodes basées sur des Anchor Boxes, il y avait une quantité significative de vrais positifs par GT object. Cependant, après le passage à la méthode centrée, le nombre total de Anchor points « true » est fortement réduit par rapport aux méthodes basées sur Anchor boxes car on devrait considérer exclusivement le centre de chaque GT object sur l'image. Donc on ignore des points avec une très bonne qualité sur l'image et on obtient un jeu de training très déséquilibré avec une valeur de rappel très réduite.

YOLOX introduit l'approche de multi-positive center qui attribue des étiquettes positives aussi au voisinage de 3x3 pixels de chaque Anchor point réel avec une amélioration de 2.1 % sur la performance du modèle.

3. Implémentation du modèle et performances

3.1. Jeu de données

On a utilisé le jeu de données 'Stanford dogs dataset' pour l'entrainement du modèle YOLOX. Les formats de données acceptés en entrée du modèle YOLOX sont COCO et VOC Pascal. Le jeu de donnée 'Stanford dogs dataset' utilise VOC Pascal.

Le format VOC Pascal pour le jeu de données prévoit une structure de trois dossiers respectivement avec les images, les annotations en format xml qui décrivent les infos par rapport au format de l'image, la classe de l'objet et les dimensions de la Bounding Box et deux fichiers de texte avec les listes des images du jeu de train et de validation. La percentage des images entre jeu de train et validation est 80/20.

```
VOCdevkit
        |____ VOC2007
        |____ VOC2012
                |____ Annotations
                |____ JPEGImages
                |____ ImageSets
                        |____ Main
                                |____ test.txt
                                |____ train.txt
```

Les classes du jeu de données « Stanford dogs dataset » sélectionnées pour le modèle sont 15 :

"Chihuahua","Japanese_spaniel", "Maltese_dog", "Pekinese","ShihTzu", "Blenheim_spaniel", "papillon","toy_terrier", "Rhodesian_ridgeback", "Afghan_hound", "basset", "beagle", "bloodhound", "bluetick", "black-and-tan_coonhound"

3.2. Paramétrage du modèle

Les modèles de détections d'objet YOLOX utilisés sont yolox_m et yolox_s qui ont été installé directement de la source https://github.com/Megvii-BaseDetection/YOLOX

	Nano	Tiny	Small	Medium	large	Extra-Large
Depth	0.33	0.33	0.33	0.67	1.0	1.33
Width	0.25	0.375	0.50	0.75	1.0	1.25

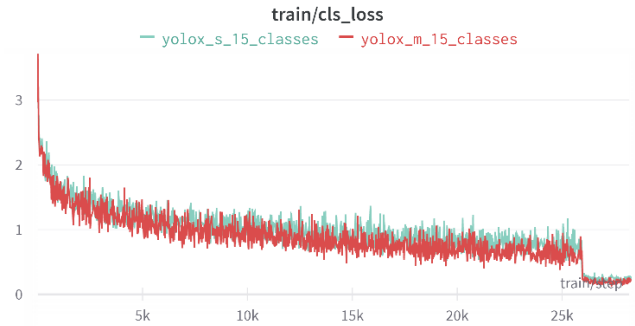
Depth = nombre de layers du réseau, Width = nombre maximale de neurones dans un layer

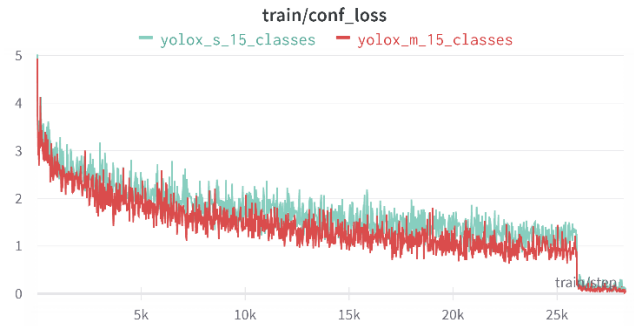
Les étapes de paramétrage dans COLAB des modèles sont :

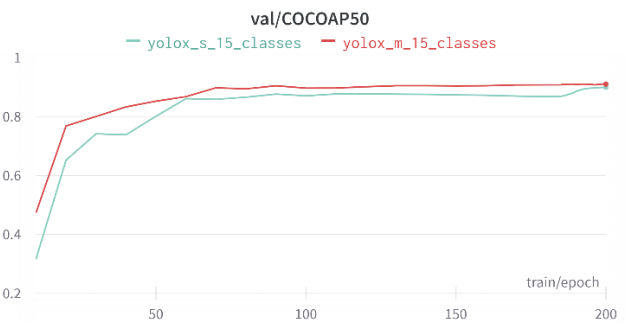
1) Import des weights des modèles pré-entrainés avec le jeu de données COCO
2) Téléchargement du jeu de données dans le format VOC Pascal
3) Définition des classes dans le fichier voc_classes.py du folder yolox.
4) Paramétrage du nombre de classes num_classes = 15, du nombre d'epochs max_epoch = 200 et du chemin du jeu de données d'entrainement et validation dans les fichiers yolox_voc_s.py et yolox_voc_m.py du dossier exps.

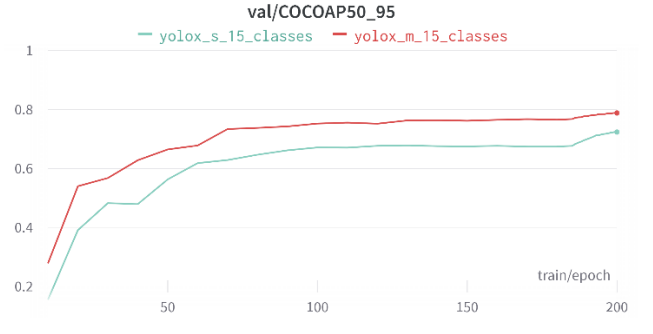
3.3. Entrainement du modèle et performances

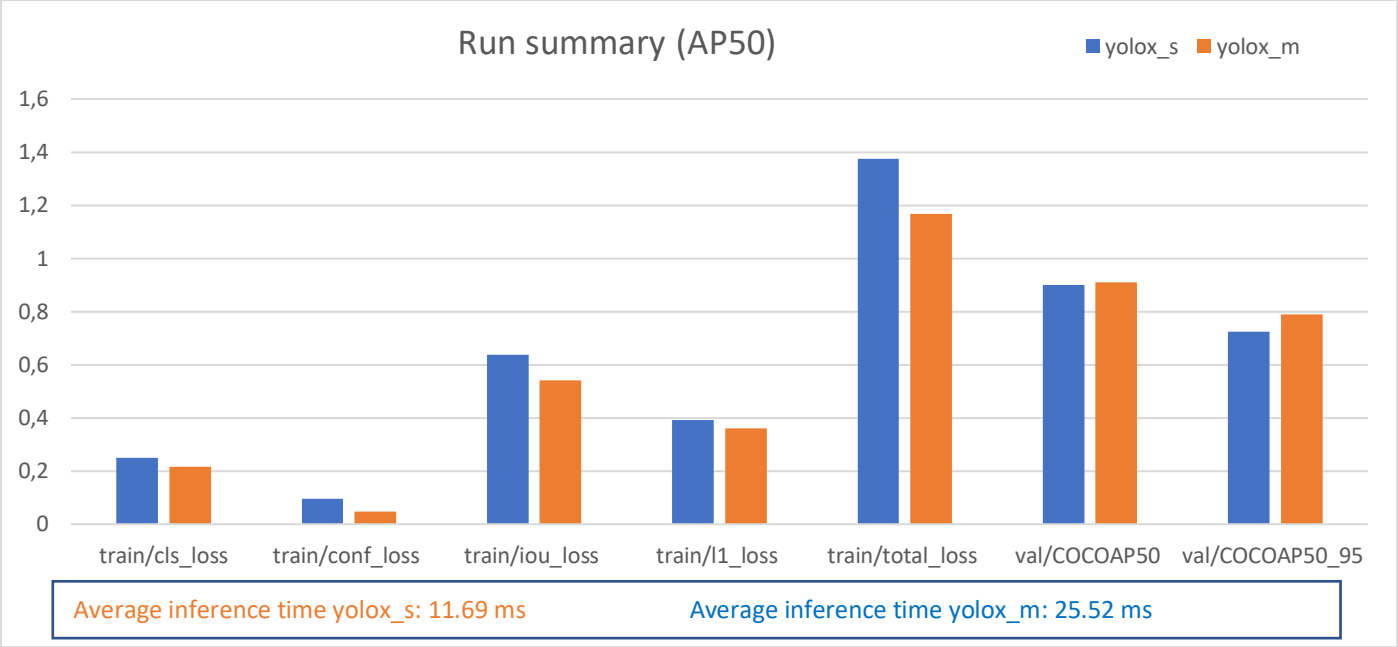
Les modèles yolox_m et yolox_s ont été entrainé dans COLAB pendant 200 epochs avec les performances suivantes, évaluées à l'aide de la plateforme Weight & Bias (https://wandb.ai/floiudice1968/YOLOX?workspace=):





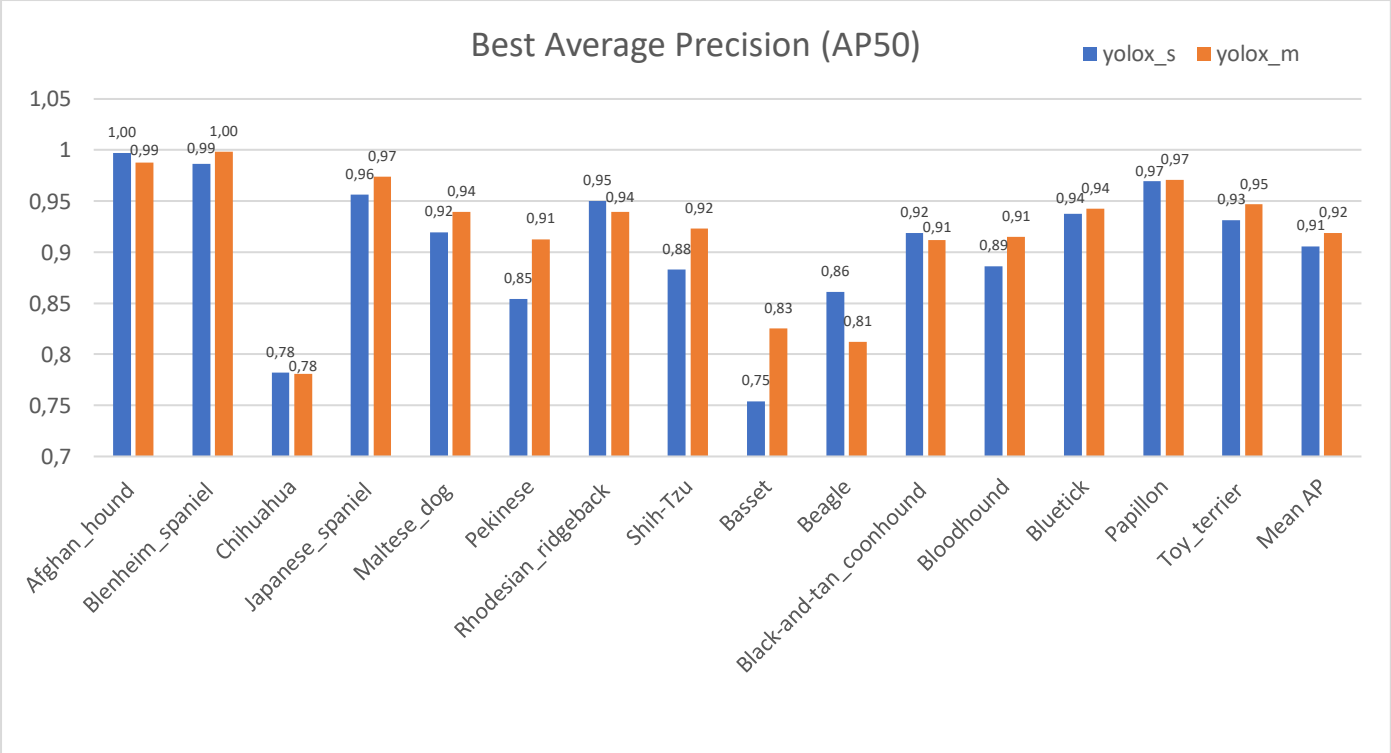






Les deux modèles de « object detection » yolox_s et yolox_m ont été évalué avec les respectives meilleurs jeux de paramètres obtenus lors de l'entrainement.

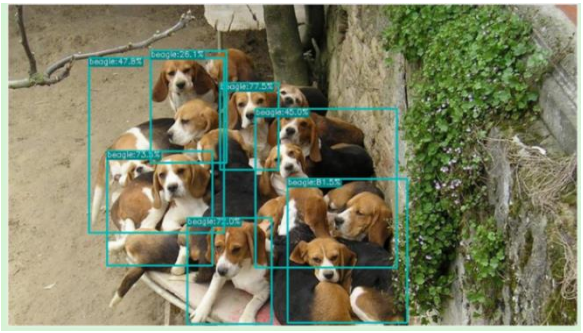
La performance est évaluée en termes de AP (Average Precision) avec un seuil IoU 0.5 pour chaque classe et en moyenne.

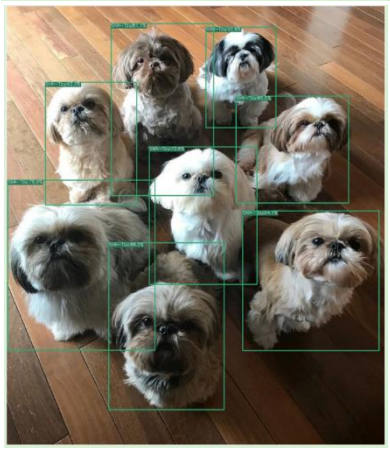


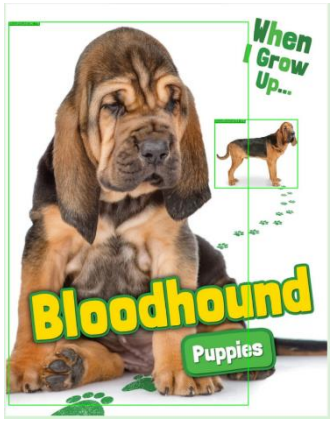
Le modèle yolox_m montre une meilleure performance en termes de qualité de la prédiction mAP50 avec une vitesse inferieure per rapport à yolox_s.

Le meilleur jeu de paramètres obtenus pour le modèle a été utilisé dans le code Python 'Classifieur_object_detection.py » qui permet d'effectuer une classification ou détection d'objets dans une image ou une détection d'objets dans une vidéo en format mp4 à l'aide d'une interface utilisateur qui permet de sélectionner et importer une image ou une vidéo.

Ci-dessous les résultats obtenus avec images qui mettent en évidence la capacité du modèle d'identifier plusieurs objets de classes différentes aussi en présence d'occlusions et avec différentes échelles.





Petit lexique YOLOX

- **Anchor-box** : Bounding Box de taille prédéfinie utilisés dans différentes version YOLO anchor-based. YOLOvs3 utilise 5 anchor-boxes pour chaque grid
- **Anchor-point** : points d'intersection sur chacune des trois grilles superposées sur l'image par YOLOX ensuite utilisés pour la définition des Bounding Boxes.
- **AP50 (Average precision)** : précision moyenne évaluée sur la courbe Precision-Recall avec interpolation sur 11 points et IoU_threshold 0.5
- **Backbone** : CNN pour l'extraction des features d'une image
- **Bounding Box** : Box pour localisation d'un objet dans une image
- **COCO (Microsoft Common Objects in Context)** : dataset avec 328K images utilisé pour la détection d'objets avec annotations en format JSON
- **cp (center prior)** : perte additionnelle utilisée lors de l'entrainement par l'algorithme simOTA pour pénaliser les Anchor points avec le carré de la distance du centre du GT Box
- **Darknet-53** : réseau de neurones (backbone) pour l'extraction des features avec 53 couches convolutionnelles utilisé in YOLO vs3 et YOLOX
- **FPN (Feature Pyramid Network)** : réseau de neurones qui extrait les features d'une image avec différentes échelles à partir de différents états de transition du backbone
- **GT (Ground Truth) Box** : True Box pour un objet sur l'image défini dans les annotations du jeu de données
- **IoU** : métrique, définie comme : Area of Overlap/Area of Union, qui quantifie le niveau de superposition entre deux régions
- **mAP50 (mean Average Precision)** : moyenne des AP de toutes les classes d'objets
- **RCNN (Region-based Convolution Neural Network)** : détecteur d'objets multistage avec stages de détection et classification séparés
- **ROI (Region of Interest)** : région identifiée sur une image envoyée ensuite à un classifieur
- **simOTA (Simplified Optimal Transport Assignment)** : algorithme utilisé pendant l'entrainement du modèle pour assigner étiquettes positives ou négatives aux Anchor points
- **SSD (Single Stage Detector)** : détecteur d'objet single stage avec stages de détection et classification parallèles
- **VOC Pascal (PASCAL Visual Object Classes)** : dataset avec 11530 images utilisé pour la détection d'objets avec annotations en format xml
- **YOLO (You Only Look Once)**: détecteur d'objet single stage basé sur l'architecture Darknet

Sources :

https://arxiv.org/abs/2107.08430
https://arxiv.org/pdf/2103.14259.pdf
https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/
https://github.com/Megvii-BaseDetection/YOLOX
https://towardsdatascience.com/object-detection-neural-network-building-a-yolox-model-on-a-custom-dataset-77d29d85ae7f
https://medium.com/p/3e5c89f2bf78
https://medium.com/mlearning-ai/yolox-explanation-simota-for-dynamic-label-assignment-8fa5ae397f76