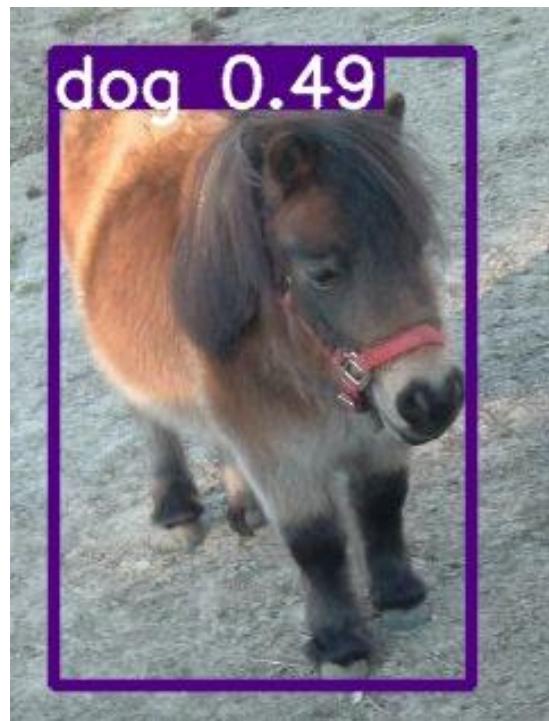


# Projet Vision Artificielle - Détection et Segmentation d'objets

17 Décembre 2021

---

BEAUREPAIRE Margot - LAUR Simon - LORGUE Paul - NAVARRO Emma



# Table des matières

<b>Introduction - Contexte du projet</b>	<b>1</b>
<b>Présentation des méthodes utilisées</b>	<b>2</b>
R-CNN	2
YOLO	3
<b>Résultats obtenus</b>	<b>3</b>
R-CNN	3
YOLO	9
<b>Conclusion</b>	<b>9</b>
Comparaison des deux méthodes	9
Discussions/Pistes d'amélioration	9
Apports du projet/Difficultés rencontrées	9

# I. Introduction - Contexte du projet

Dans le cadre du module de Vision Artificielle, le but de ce projet était de tester différents algorithmes et méthodes de détection et reconnaissance d'objets, afin de comparer leurs résultats et performances.

La détection d'objets consiste à localiser les régions de l'image qui contiennent un objet, et la reconnaissance correspond à labelliser l'objet qui s'y trouve.

La segmentation, elle, consiste à délimiter les contours des objets en classant chaque pixel de l'image. La segmentation peut être sémantique ou par instance. Dans une image contenant cinq personnes côte à côté, une segmentation sémantique détectera une seule forme englobant les cinq personnes, alors qu'une segmentation par instance séparera chacune des cinq personnes.

Nous avons donc choisi deux modèles de détection et reconnaissance d'objets à implémenter, tester et comparer : YOLO et RCNN. Nous nous sommes séparés en deux équipes de deux personnes, Paul et Emma travaillant sur RCNN, Simon et Margot sur YOLO.

Afin de pouvoir par la suite comparer les deux méthodes, nous avons choisi de tous utiliser le même jeu de données, et avons choisi le jeu de données PASCAL VOC 2012, sous-partie du jeu de données ImageNET.

Ce jeu de données contient 17125 images, dans lesquelles se trouvent des objets de 20 classes différentes. Pour chaque image du dataset, un fichier d'annotations est également fourni, donnant pour chaque objet présent sur l'image et qui appartient à l'une des 20 classes du dataset, la localisation et la taille de la région qui le contient, ainsi que le label de sa classe. Il y a au total 40138 objets annotés. La classe "personne" est beaucoup plus présente.

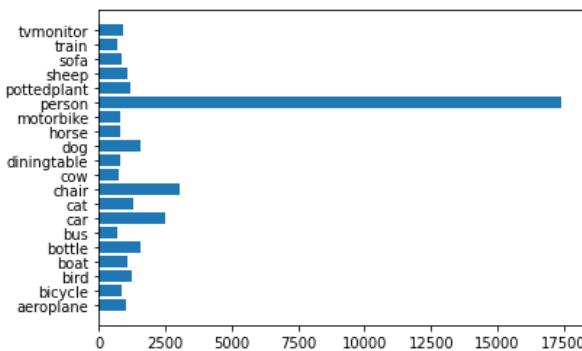


Figure n°1 - Répartition des classes dans le dataset

# II. Présentation des méthodes utilisées

Dans cette partie, nous allons décrire chacune des deux méthodes implémentées, en présentant leur architecture, les données utilisées, ainsi que les paramètres choisis pour entraîner les modèles.

## 1. R-CNN

### a. Architecture

Il existe plusieurs versions de modèle R-CNN (Region Based Convolutional Neural Networks), nous avons choisi la première version.

L'approche générale pour détecter et labelliser un objet avec un modèle RCNN consiste en trois étapes :

- générer des régions d'intérêt contenant potentiellement un objet
- associer à chacune de ces régions les probabilités qu'elles contiennent chacun des objets potentiels,
- sélectionner et regrouper les régions ayant la plus grande probabilité de contenir les objets.

### b. Générer des régions par Selective Search

La première étape est réalisée par Selective Search qui permet de proposer des régions de l'image en regroupant des ensembles de pixels avec des textures, couleurs, tailles et formes similaires.

Les regroupements sont effectués à différentes échelles de l'image pour obtenir différentes tailles de région. Cette méthode permet de générer entre 1500 et 2500 régions sur notre jeu de données.

Pour augmenter le jeu de données, nous avons appliqué le Selective Search sur les images annotées. Nous avons comparé les régions proposées avec les localisations réelles des objets, en se servant des coordonnées des boîtes englobantes fournies.

Ainsi, pour chaque région proposée, nous avons calculé son IoU (Intersection Over Union), avec les bounding boxes réelles.

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

Si l'IoU obtenu est supérieur à 0.8, nous associons à la région proposée le label de la classe correspondante et la région est caractérisée comme une région "foreground" car elle contient un objet. Cette région est alors ajoutée aux données d'entraînement du CNN.

Si l'IoU obtenu est inférieur à 0.05, nous associons à la région proposée une 21ème classe : "background", qui signifie que cette région ne contient pas d'objet. La région est aussi ajoutée au jeu de données d'entraînement du CNN.

### c. Classifier avec un CNN

La deuxième étape est effectuée par un classifieur CNN qui prend en entrée une région de l'image sur laquelle la détection doit être faite et donne en sortie une distribution de probabilités pour la classe contenue dans l'image. On peut alors déterminer que selon ce réseau l'image contient à  $p$  pourcents tel label et  $p'$  pourcents tel autre label.

Il est nécessaire d'entraîner ce réseau à reconnaître les objets que l'on souhaite. Nous avons choisi de finetuner le modèle VGG16 pré-entraîné avec les images d'IMAGNET pour reconnaître 1000 objets différents.

Nous avons figé les poids des quinze premières couches de convolution, puis nous avons modifié la tête du modèle avec 3 nouvelles couches: une couche dense de 512 neurones, une couche de Dropout de 50% et une autre couche dense de 256 neurones. La couche de sortie est activée par la fonction softmax et comporte 21 neurones, représentant les 20 classes du jeu de données et la classe background.

Toutes les images sont redimensionnées en (224,224, 3) avant d'être insérées dans le réseau CNN.

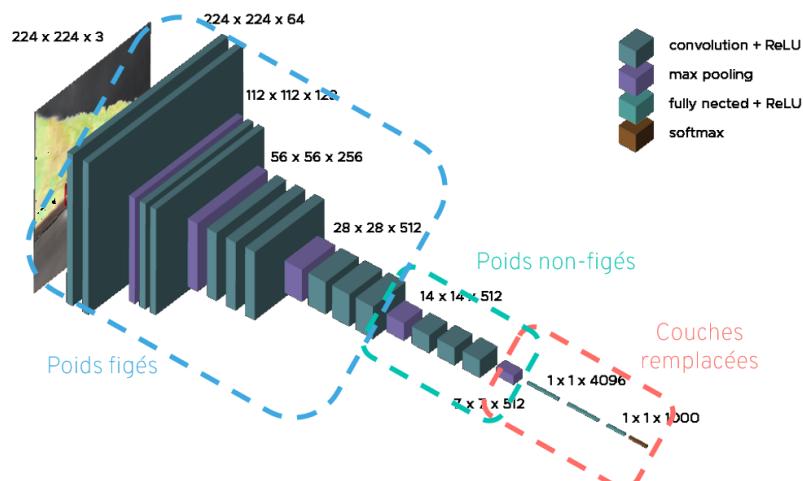


Figure n°2 - Image représentant l'architecture du réseau utilisé

Nous avons utilisé l'optimiseur Adam, une categorical crossentropy en tant que loss et pris comme métrique l'accuracy du modèle.

Les données d'entraînement comportent environ 40000 régions annotées réparties sur toutes les classes de manière équilibrée. Nous avions prévu d'entraîner le modèle sur 50 époques, cependant 19 ont suffit. Nous avions défini un callback avec un early stopping de patience 5 et la précision du modèle sur le jeu de données de validation ne s'était pas améliorée depuis l'époque 14. L'entraînement dure 4 à 5 heures sur une vingtaine d'époques grâce à l'utilisation d'un GPU (Nvidia GTX 1650 avec 4Go de mémoire dédié)

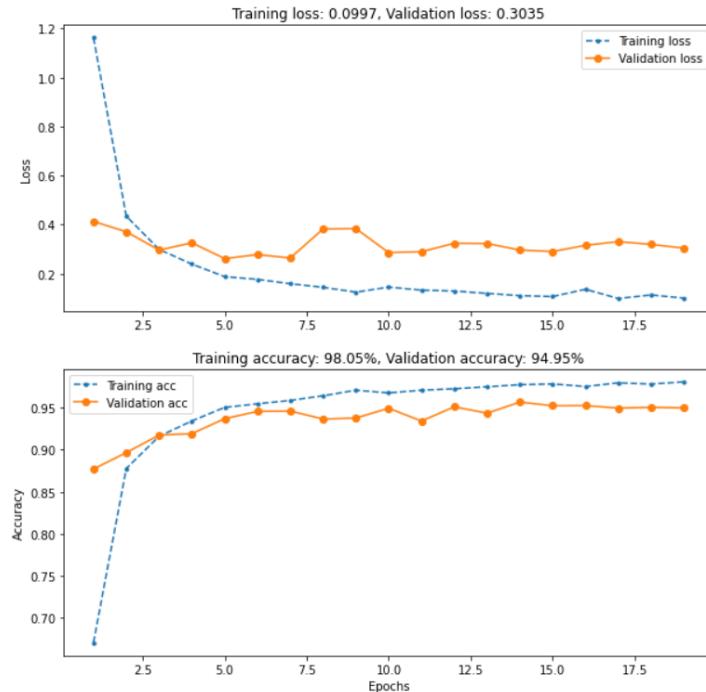


Figure n°3 - Evolution des valeurs de loss et d'accuracy sur les jeux de données d'entraînement et de validation en fonction des epochs

Le meilleur modèle est celui après entraînement de la 14ème epoch, pour laquelle nous avons obtenu une loss de validation d'environ 0.3, et une accuracy de validation d'environ 0.95.

#### d. Sélectionner et regrouper les meilleures régions

Une fois les probabilités d'apparition de chaque objet dans chaque image calculées, on doit déterminer quelles régions englobent le mieux les objets présents dans l'image initiale. Nous avons appliqué plusieurs méthodes pour essayer d'y parvenir:

- garder uniquement les régions dont le CNN prédit à plus de 70% l'existence d'un objet,
- supprimer les régions dont le label le plus probable n'apparaît que peu de fois,
- utiliser l'algorithme de Non Max Suppression avec les ensembles des régions avec leur distribution de probabilités associées,
- appliquer la médiane des régions pour chaque classe prédite par le CNN.

## 2. YOLO

YOLO ou You Only Look Once est un algorithme de détection d'objets dans les images mis en place par Redmond et. al. YOLO utilise un réseau de neurones pour à la fois classifier et prédire les bounding boxes des objets détectés.

### a. Principe de fonctionnement

Afin de détecter et classifier les objets, YOLO passe par trois étapes : la segmentation de l'image en images plus petites, la prédiction des boundings boxes et la sélection de la meilleure bounding box à l'aide d'un algorithme de non-max suppression.

Dans un premier temps, YOLO prend l'image en entrée et la divise en SxS cellules ayant toutes les mêmes dimensions. La détection d'objet va ensuite se dérouler dans chacune de ces cellules. La cellule représentant le centre de l'objet est la cellule responsable de la détection de cet objet.

Ensuite, chaque cellule va prédire B bounding boxes. Les bounding boxes sont un encadrement permettant de mettre l'objet détecté en valeur. Chacune de ces boundings boxes possède 4 propriétés :

1. Un score de confiance
2. Largeur de la bounding box
3. Hauteur de la bounding box
4. Les coordonnées du centre de la bounding box

Nous allons donc nous retrouver avec un grand nombre de bounding boxes. Alors, pour pallier ce problème, YOLO va ensuite utiliser l'IoU pour regrouper les boxes associées à un même objet pour ensuite effectuer un algorithme de non-max suppression afin de garder seulement la meilleure d'entre elles. Ainsi, en théorie, nous devrions nous retrouver avec une bounding box par objet présent sur l'image.

Pour utiliser YOLO, un notebook nommé YOLO\_Beaurepaire\_Laur est présent sur le git, dans la branche simonyolo.

### b. Architecture

Yolov3 utilise une variante de Darknet qui possédait au départ 53 couches convolutives entraînées sur ImageNET. Pour la détection, 53 couches supplémentaires sont ajoutées ce qui nous donne un réseau de 106 couches. La taille de ce réseau est notamment une des raisons principales de sa lenteur d'entraînement.

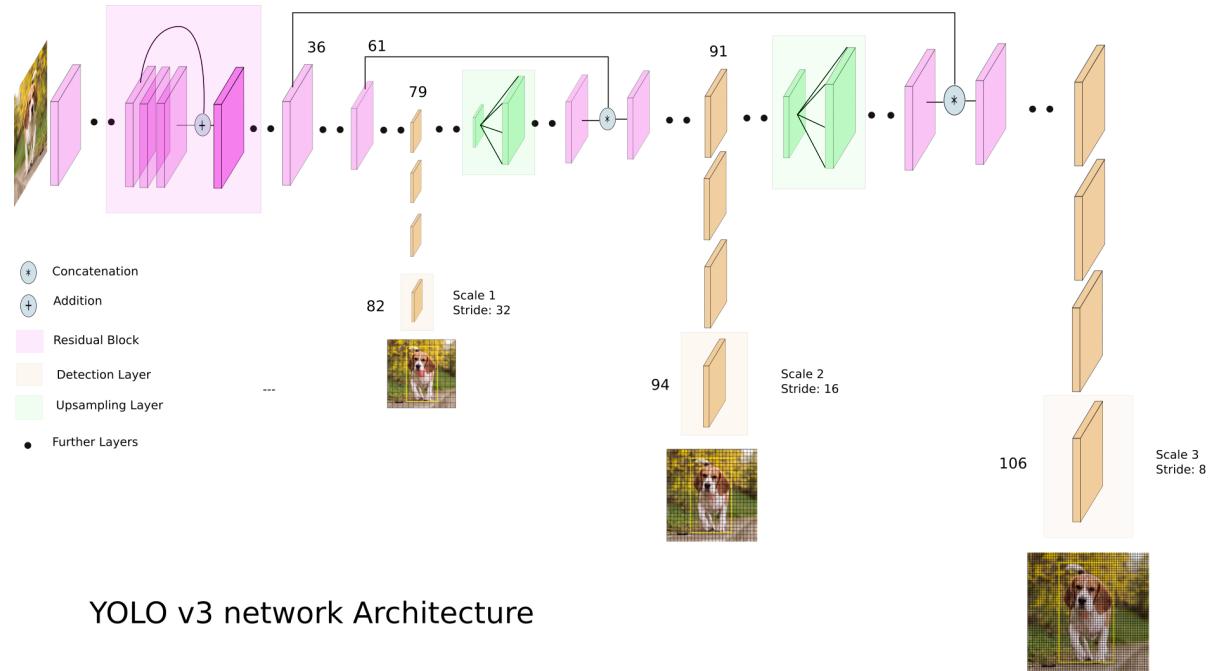


Figure n°4 - Schéma de l'architecture de YOLOv3

### III. Résultats obtenus

Maintenant que les différentes méthodes ont été présentées, nous allons vous présenter les résultats obtenus avec celles-ci.

#### 1. RCNN

Le jeu de données de test pour le R-CNN est composé d'un peu plus de 5000 régions annotées. Pour évaluer le classifieur de notre modèle RCNN, nous avons calculé le f1-score qui se calcule à partir de la précision et du rappel:

$$f1_{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

$$\text{avec } Precision = \frac{\text{True Positives}}{\text{Total Predicted Positives}} \text{ et } Recall = \frac{\text{True Positives}}{\text{Total Actual Positives}}$$

Nous avons obtenu un f1-score de 96% sur le jeu de données de test, ce qui est très satisfaisant.

Pour observer plus en détail les différentes prédictions, nous avons ensuite affiché la matrice de confusion des prédictions des images de test.

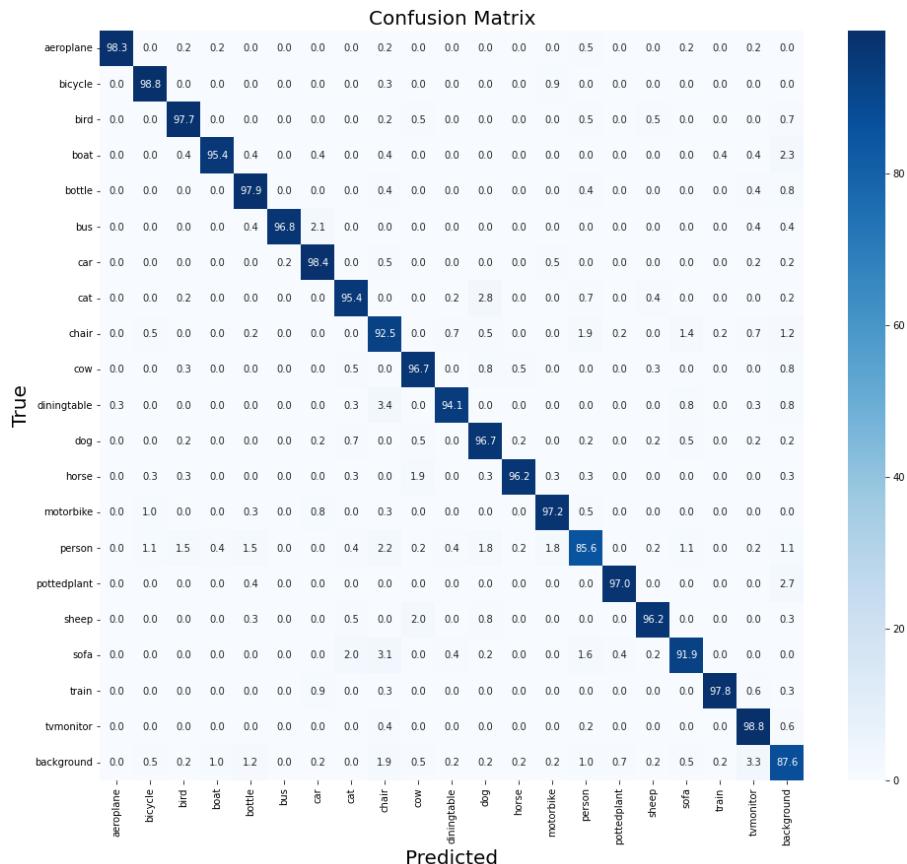
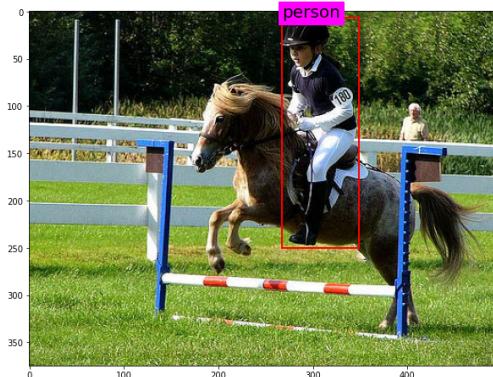


Figure n°5 - Matrice de confusion des prédictions des classes

La diagonale est plutôt bien définie cependant il est important de noter que les personnes et l'arrière-plan ne sont reconnus qu'à 85,6 et 87,6% du temps. L'arrière-plan occupe la majorité des images donc beaucoup de régions devront être labellisées "background", ce fort taux de faux négatifs est alors d'autant plus problématique.

Au delà de ces résultats, nous avons voulu regarder les prédictions du modèle sur une image du dataset que nous avons choisi aléatoirement.

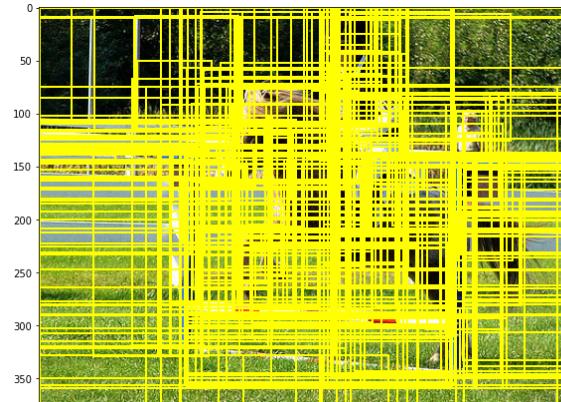


*Figure n°6 - Image aléatoire du dataset Pascal VOC, avec annotation correspondante*

Comme l'on peut le voir, dans le fichier d'annotations de cette image, il n'y avait qu'une bounding box correspondant à une personne, comme représenté sur l'image par le cadre rouge.

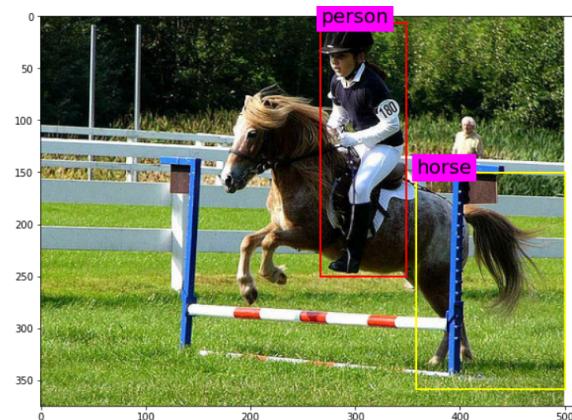
Nous avons ensuite effectué une prédiction grâce à notre modèle sur cette image. Nous obtenons les bounding boxes suivantes, correspondant aux zones dans lesquelles le modèle a prédit un objet qui n'est pas un background, avec une probabilité de plus de 70%. On remarque qu'il reste encore beaucoup de régions après ce seuillage.

Cette image n'étant pas vraiment lisible à cause du nombre de prédictions effectuées par le modèle, nous avons choisi d'en afficher une aléatoirement, comme présenté sur la figure ci-dessous.



*Figure n° 7 - Prédiction du modèle sur l'image d'entrée*

Nous pouvons voir sur l'exemple ci-contre que la prédiction n'ets pas mauvaise car dans le rectangle jaune se trouve effectivement une partie d'un cheval, mais elle n'englobe quand même pas le cheval en entier. De plus, ce n'est qu'une prédiction parmi environ 300 trouvées par le modèle, et nous aimeraisons donc trouver un moyen de réduire ce nombre de prédictions afin de garder uniquement les meilleures.



*Figure n°8 - Exemple d'une prédiction (jaune) par rapport à l'annotation (rouge)*

Pour commencer, nous avons observé la répartition des classes présentes dans les prédictions, et obtenu la figure ci-dessous.

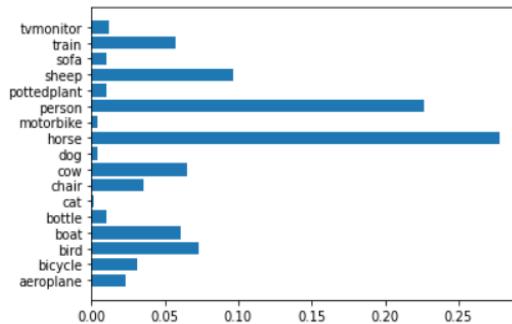


Figure n°9 - Répartition des classes prédictes à plus de 70% de certitude

Pour cette exemple et pour la plupart des images que nous avons fournies au modèle, il y a beaucoup plus de classe représentées qu'il n'en faudrait d'autant plus qu'on ne garde ici que les régions dont le classifieur a prédit à plus de 70% de certitude qu'elle contenait tel ou tel objet.

En observant la répartition des classes sur plusieurs images, nous pouvons tout de même noter que les classes le plus souvent prédictes sont les classes des objets réellement présents. Nous avons donc fixé un seuil de manière arbitraire à 10%, afin de sélectionner uniquement les classes prédominantes (qui sont censées être les seules de valides).

Après application de ce seuil, le nombre de bounding boxes a diminué, comme on peut le voir sur la figure suivante.

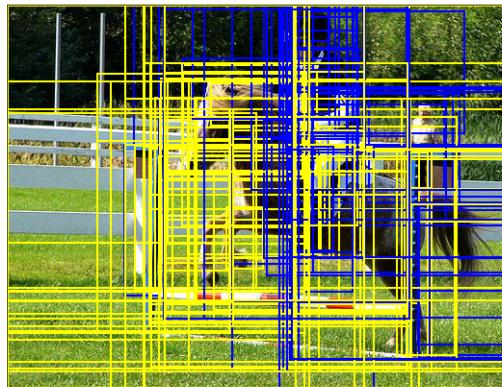


Figure n°10 - Bounding boxes des deux classes sélectionnées avec application du seuil  
(jaune: horse, bleu: person)

Pour regrouper ces régions et ne garder que les plus pertinentes nous avons appliqué l'algorithme de non max suppression (avec seuil iou à 0.7) sur les régions correspondant aux deux classes sélectionnées après application du seuil.

Le Non max suppression prenant en-compte le niveau de certitude que chaque région contienne telle ou telle classe nous nous sommes intéressés à la distribution des probabilités associés à chaque région.

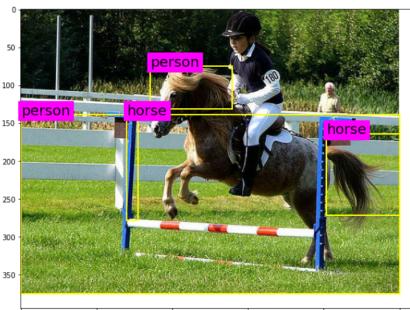


Figure n°11 - Résultat de l'algorithme de non max suppression

On peut voir que la majeure partie des probabilités sont autour de 1 ce qui ne permet pas de pondérer la pertinence d'une région par rapport à une autre, l'algorithme de non max suppression est alors peu efficace.

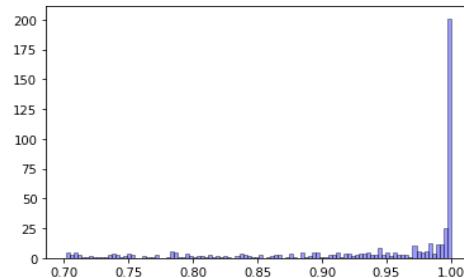


Figure n°12 - Répartition des probabilités associées à chaque région

Partant du constat que toutes les régions ont la même importance nous avons choisi de faire la médiane des régions (médiane de chaque coordonnée) pour chacune des classes.

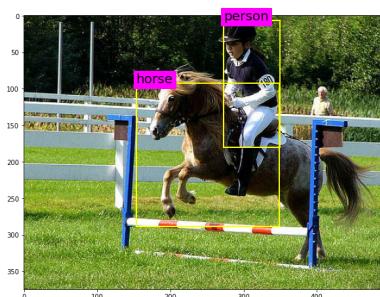


Figure n°13 - Résultat de l'application de la médiane sur les bounding boxes des classes "person" et "horse"

Le résultat obtenu sur cette image est plus cohérent.

La méthode de la médiane est plutôt basique et donne des résultats erronés dès lors qu'il y a plus d'une instance d'une classe dans l'image, comme on peut le constater avec l'image ci-dessous avec plusieurs moutons présents.



Figure n°14 - Résultats du calcul de la médiane sur une image avec plusieurs instances d'une même classe

Il nous faut donc trouver une solution pour les images avec plusieurs instances d'une même classe, et peut-être essayer de trouver une méthode plus efficace que la médiane pour les images avec une seule instance d'objets par classe.

## 2. YOLO

Après avoir pris en main YOLO, nous avons ré-entraîné le modèle sur le jeu de données Pascal VOC 2012 dans le but de pouvoir ensuite comparer nos résultats avec ceux obtenus pour le R-CNN.

Dans le temps imparti et les capacités des machines utilisées, nous avons dû faire des choix pour optimiser au maximum le temps d'apprentissage du modèle. Ainsi, sur les 17125 images du jeu de données nous en avons

sélectionné seulement 300 pour l'apprentissage du modèle et 200 pour les données de validation. De plus, le modèle n'a été entraîné que sur 22 epochs.

Voici, ci-dessous, les résultats que nous obtenons sur les données de validation.

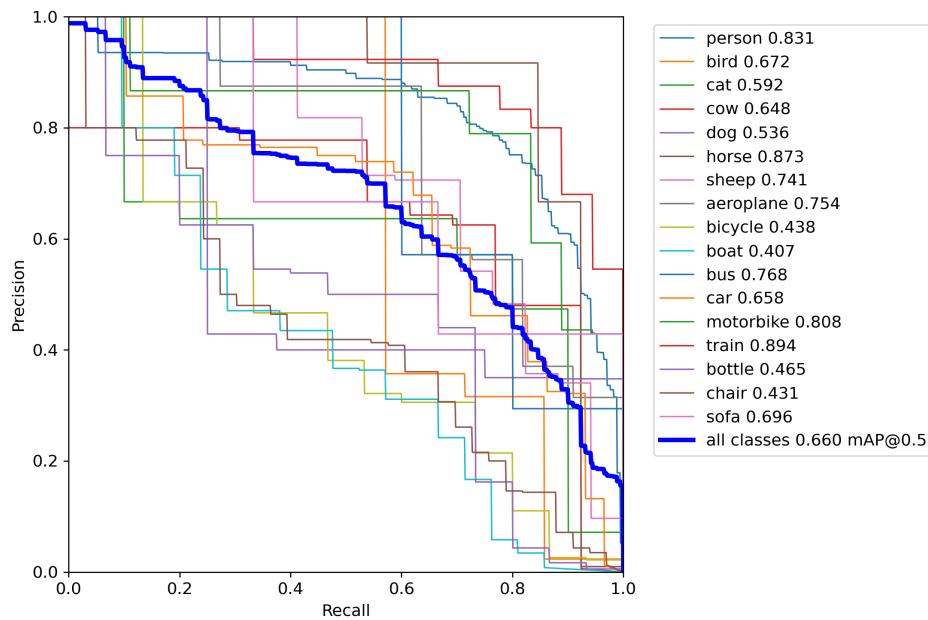


Figure n° 15 - Precision en fonction du Recall sur les données de validation

$$\text{Precision} = \frac{TP}{TP + FP} \quad \boxed{\text{Recall} = \frac{TP}{TP + FN}}$$

En regardant la PR-curve ci-dessus, nous pouvons voir que nos résultats semblent plutôt bons. Cependant, nous voyons que certaines classes ont une meilleure courbe que d'autres.

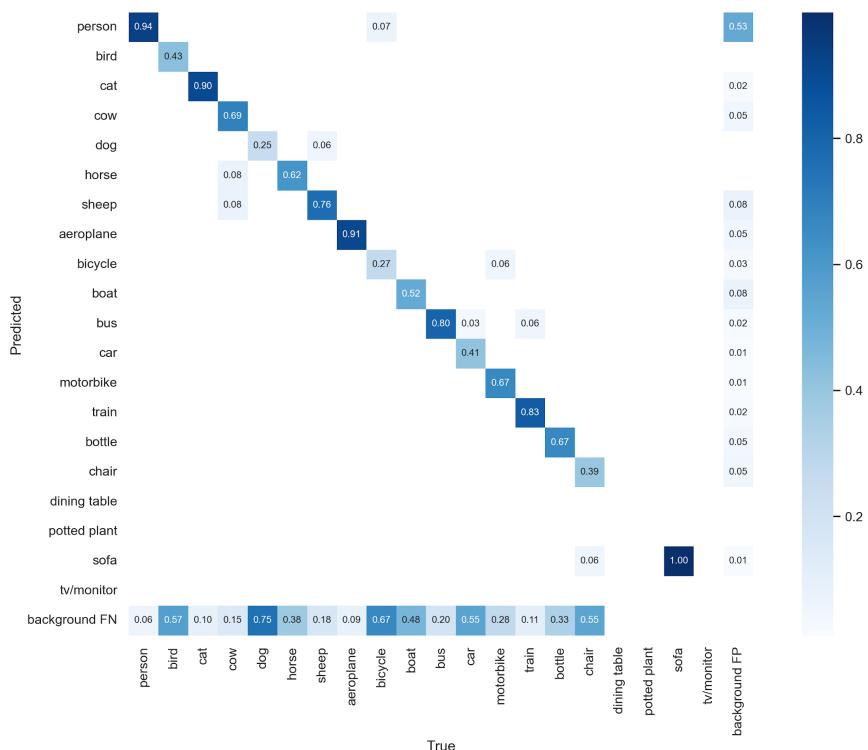


Figure n° 16 - Matrice de confusion sur les données de validation

Ces résultats ont aussi été confirmés par la matrice de confusion sur laquelle nous pouvons observer une diagonale qui n'est pas parfaite. En effet, pour que le modèle soit parfait, il faudrait des 1 sur toute la diagonale et des 0 partout ailleurs. Ainsi, nous voyons que les classes les mieux prédictes sont *person*, *cat*, *aeroplane* et *sofa*. Pour le reste, nous pouvons voir que beaucoup d'objets sont parfois associés à du *background FP*. C'est un problème car le modèle ne va donc pas détecter tous les objets présents sur une image. Cependant, ce problème est moins grave que si le modèle confondait deux objets. Ainsi, nous sommes sûrs que si le modèle prédit un *sofa*, ce ne sera pas une *person*. Cependant, le modèle peut ne pas détecter le chien présent sur l'image et l'associer à l'arrière-plan.

## IV. Conclusion

### 1. Discussions/Pistes d'amélioration

Concernant les R-CNNs, le problème majeur est que l'algorithme prévoit des objets dans des bounding boxes avec de très hautes probabilités à chaque fois, il est donc difficile de choisir par la suite les meilleures prédictions.

Ceci est peut être dû au fait que pour les données d'entraînement, nous ne sélectionnons uniquement les régions avec des  $\text{IOU} > 0.8$  pour les objets et des  $\text{IOU} < 0.05$  pour les backgrounds. Il faudrait peut-être sélectionner des bounding boxes avec des IOU plus variés pour remédier à ce problème.

Par ailleurs notre classifieur fait trop d'erreurs dans la prédiction de l'arrière-plan ce qui engendre énormément de faux positifs pour les autres classes ainsi les véritables régions contenant un objet bien labélisé sont noyées dans ce bruit. Il serait peut être judicieux de finetuner CNN sur plus d'images d'arrière plan.

Le RCNN reste plutôt lent à entraîner, car pour chaque image des données d'entraînement, on propose au réseau environ 2000 bounding boxes issues de la Selective Search. Un autre problème est qu'il n'y a pas de possibilités d'entraînement de la Selective Search, qui peut donc être amenée à proposer des régions candidates non pertinentes.

D'autres algorithmes utilisant la même approche que les R-CNNs permettent de répondre en partie à ces problèmes, notamment Fast-RCNN et Faster-RCNN. En effet, au lieu de donner des régions de l'image en entrée au CNN, ils lui donnent l'image entière, ce qui fait beaucoup moins d'entrées pour le réseau et réduit donc considérablement les temps d'entraînement.

Concernant la sélection de région, Fast-RCNN utilise Selective Search, ce qui pose le même problème mis en lumière lors de l'utilisation du R-CNN. Cependant, le Faster-RCNN utilise un réseau séparé pour proposer des régions, ce qui répond au problème de l'aspect "fixe" de la Selective Search.

Concernant YOLO, il faudrait augmenter la taille des données d'entraînement et le nombre d'epochs afin d'observer des améliorations. Cependant, un problème connu de YOLO est le regroupement d'objets d'une même classe : par exemple, s'il y a une superposition de plusieurs objets d'une même classe, comme un chat devant un autre, il pourrait en compter qu'un seul. Pour minimiser au maximum ce problème, nous pourrions par exemple tenter de jouer sur la valeur de l'IoU qui regroupe les boxes d'une même classe avant de réaliser l'algorithme de non-max suppression.

### 2. Apports du projet/Difficultés rencontrées

Au lancement du projet, nous avons eu du mal à définir nos objectifs et nous avons donc sûrement perdu du temps sur le démarrage que nous n'avons pas pu rattraper par la suite. En effet, découvrant les techniques de détection d'objets pour la première fois nous ne savions pas quelles seraient les sorties et résultats de chaque algorithme. Étant séparés en deux groupes, il nous était compliqué de comprendre les travaux réalisés par le

second groupe. Aussi, pour les deux techniques, les temps d'entraînement étaient assez longs avec les machines que nous avions à disposition, un seul GPU ne suffisant pas pour nos deux groupes. Alors, nous n'avons pas réellement eu le temps de pouvoir adapter les résultats des deux groupes afin d'avoir des données numériques comparables. De même, nous aurions aimé pouvoir commencer la segmentation d'objets.

Bien que difficile, ce projet nous a tout de même permis de découvrir, de comprendre et d'approfondir des méthodes de détection d'objets performantes et populaires. Ces notions nous sont déjà utiles dans les autres projets scolaires et il ne fait aucun doute qu'elles seront aussi très appréciées dans nos futurs stages et emplois respectifs. Pour certains d'entre nous, le projet a aussi permis de découvrir la bibliothèque graphique OpenCV très utilisée.