Projet YOLO

Efkan TUREDI

Intro

L'objectif de ce projet est de réaliser une veille de recherche concernant une thématique de notre choix.

Notre goût prononcé pour les problématiques de computer vision nous a poussé à choisir la problématique suivante:

Mettre en place une solution de reconnaissance du port de masques sur des photos soumises par l'utilisateur

Nous devrons donc faire un tour des méthodes existantes, trouver les bons datasets, et implementer notre méthode

Préambule

Quelques questions préliminaires

- **Pourquoi les masques?** Etant donnée le contexte sanitaire, il nous a paru intéressant de traiter un sujet lié à cette actualité.

- **Pourquoi la détection d'objets?** Le projet précédent à propos de la reconnaissance des races de chiens était intéressant. De plus, c'était aussi une opportunité d'apprendre le framework Darknet

Qu’est ce que la détection d’objets?

- **Une détection d’objets** consiste à avoir une image avec plusieurs objets et de renvoyer un ou plusieurs conteneurs d’objets avec un label pour chaque conteneur
    - Elle implique l’utilisation de la classification d’image et de la localisation d’objets

- La base de données **ImageNet** est connue comme étant une référence dans les concours de détections d’objets

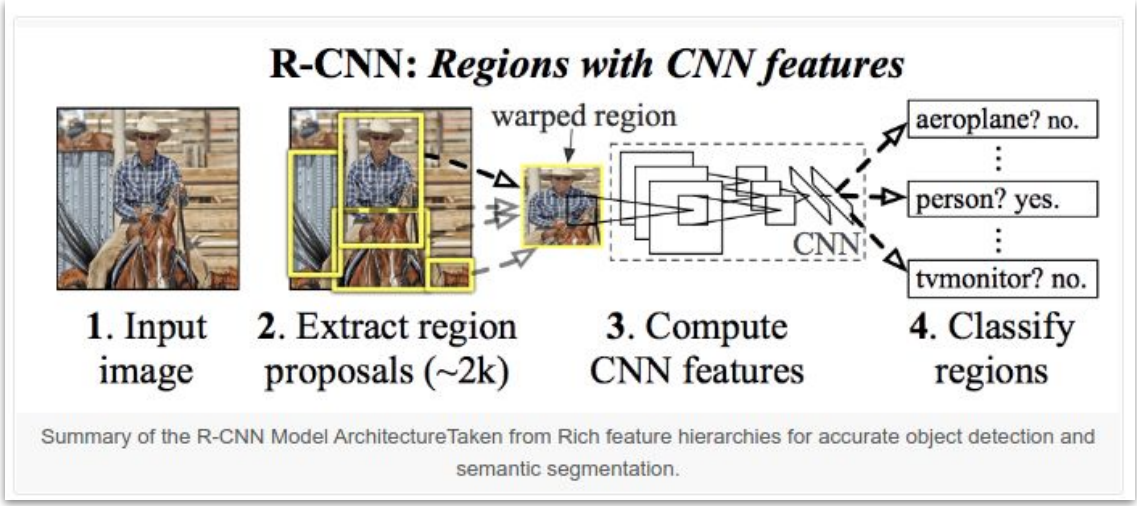
Revue des papiers existants

La famille R-CNN: R-CNN l'original (1/3)

La structure de cette méthode repose sur 3 modules qui s'enchaînent:

- Module 1: Proposition de régions. Génère et extrait des catégories indépendantes des régions c-a-d des contours candidats
- Module 2: Extraction de caractéristiques. Extraires des features de chaque région candidat à l'aide d'un CNN
- Module 3: Classification. Classifier les features obtenues dans l'une des classes disponibles à l'aide d'un softmax, logit ou SVM

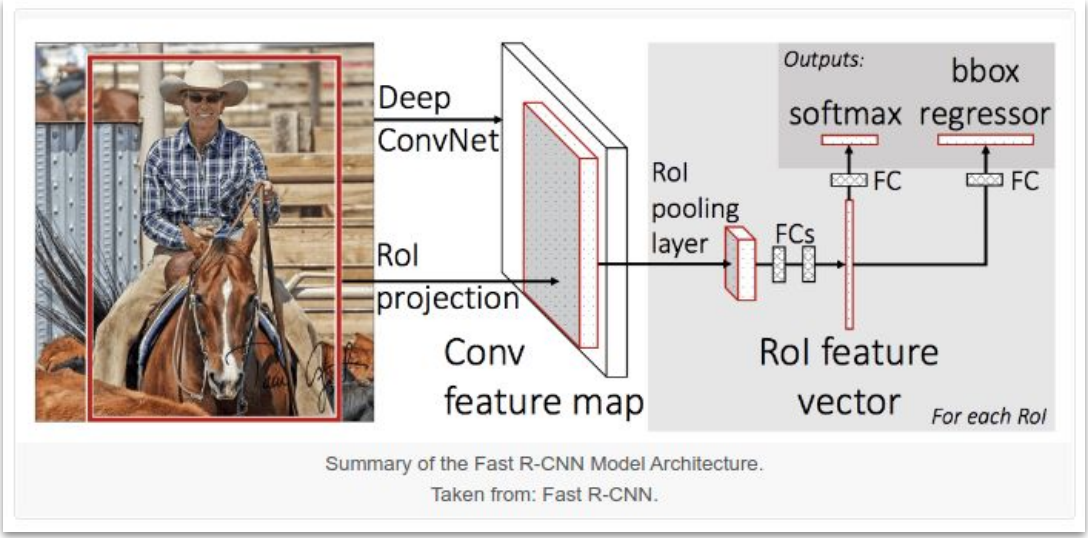
L'inconvénient majeur de cette méthode est qu'elle est très lente même avec de grosses ressources.



Summary of the R-CNN Model ArchitectureTaken from Rich feature hierarchies for accurate object detection and semantic segmentation.

La famille R-CNN: Fast R-CNN (2/3)

- Il reprend essentiellement la structure de l'algorithme en lui donnant des améliorations pour, comme son nom l'indique, augmenter rapidement sa vitesse d'entraînement. Son innovation principale consiste du fait que cette méthode demande des "Region of Interest" en input avec les images qui lui sont soumises. L'output de ces CNN est ensuite soumise à des couches connectés de 2 manières: un softmax pour classification, et un SVM linéaire pour la réalisation

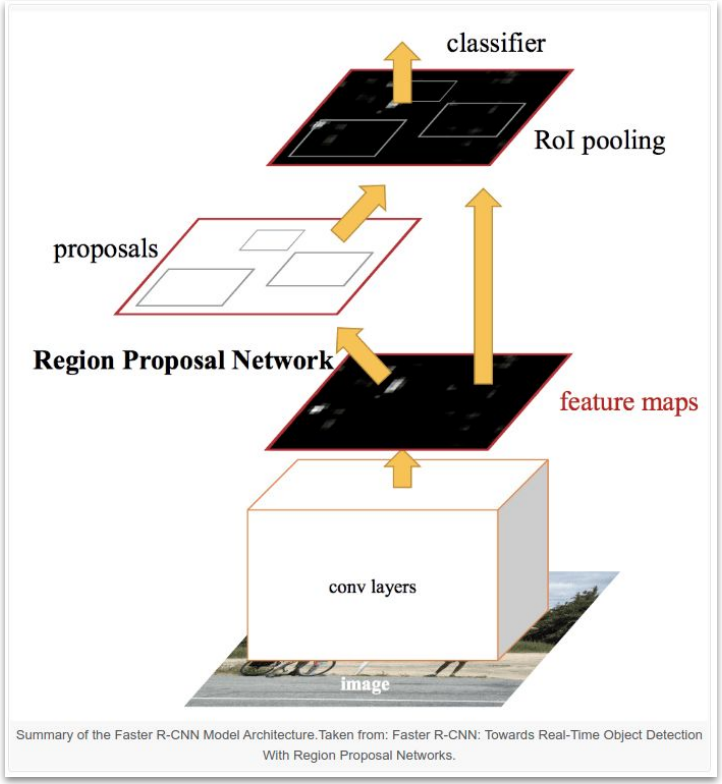


Summary of the Fast R-CNN Model Architecture.
Taken from: Fast R-CNN.

La famille R-CNN: Faster R-CNN (3/3)

Encore une fois une amélioration du modèle précédent. Il se compose en fait d'un Fast R-CNN précédé d'un module supplémentaire concernant les "Region of interest".

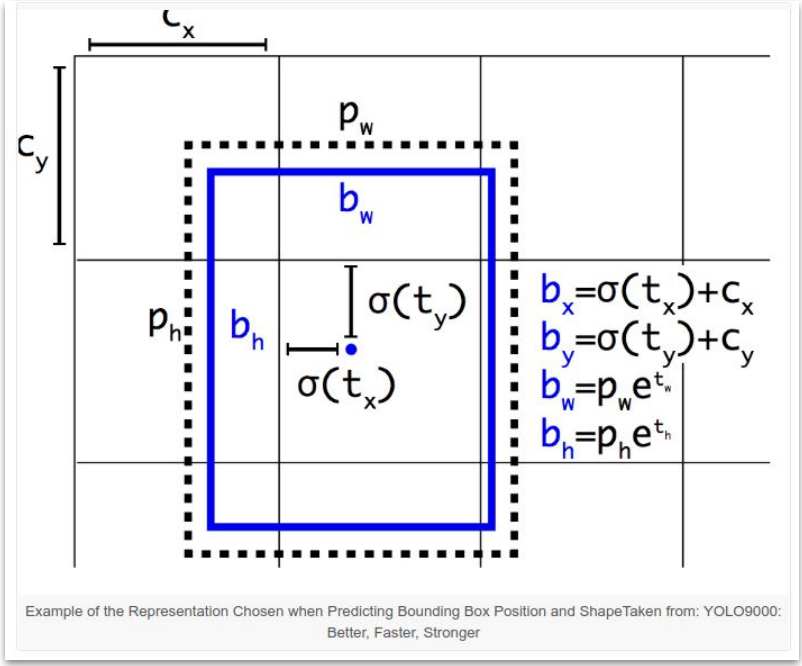
- Module 1: CNN pour proposer des régions intéressantes.
- Module 2: Fast R-CNN pour extraction des features et dessiner des contours



Summary of the Faster R-CNN Model Architecture.Taken from: Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks.

La famille YOLO: de v1 à v4

Summary of Predictions made by YOLO Model.Taken from: You Only Look Once: Unified, Real-Time Object Detection



Example of the Representation Chosen when Predicting Bounding Box Position and ShapeTaken from: YOLO9000: Better, Faster, Stronger

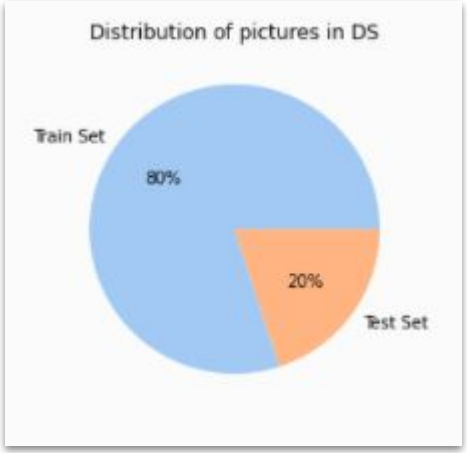
Dataset

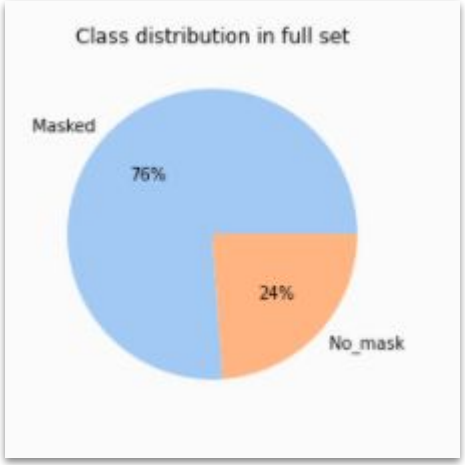
Dataset Mask au format YOLO

- Nous avons choisi une base de données présentes sur Kaggle compatible avec le format YOLO
- Nous avons une classification avec deux classes: mask et no_mask. Les test et training set sont déjà mis en place

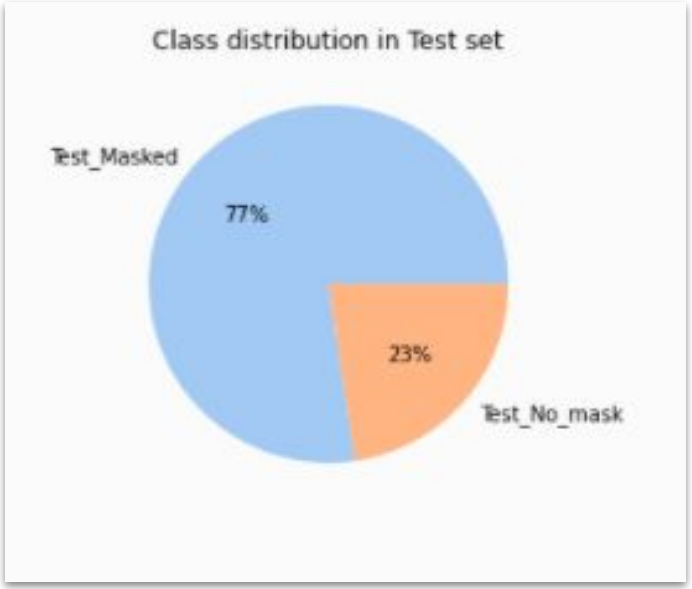
Quelques graphiques sur le dataset

	Set_type	Masked_truth	No_mask_truth	Total_truth	#_of_samples
0	train	1511	480	1991	1208
1	test	445	132	577	302
2	full_dataset	1956	612	2568	1510





Quelques details sur le Train et Test Set



Implémentation de motre méthode

Des instructions de terminal avant tout

- Le framework Darknet est assez particulier comparé à Keras. Nos instructions pour lancer nos modèles seront avant tout des instructions avec bash
- Pour lancer l'entraînement:

```
!./darknet detector train cfg/labelled_data.data cfg/yolov4-custom_train.cfg
weights/yolov4.conv.137 -dont_show
```

- Pour évaluer la performance sur le test set:

```
!./darknet detector map data/labelled_data.data cfg/yolov4-custom_test.cfg
weights/yolov4-custom_train_last.weights -points 0
```

Les configurations

- Nous allons largement nous baser sur les fichiers de configurations initiales de YOLOv4 et v3, et adapter quelques paramètres selon les conventions.

- Les paramètres que nous changeons sont:
  - Max_batches = # de classes * 2000 = 2*2000 = 4000 (6000 pour YOLOv4)
  - Steps = 80%, 90% * max_batches = 3200,3600 (4800,5400 pour YOLOv4)
  - Classes = 2 dans chaque couche YOLO
  - Filter = 24 (masks +)*dans chaque couche juste avant chaque YOLO

Une dummy baseline

- Nous utilisons un dummy classifier de scikit-learn que nous utilisons comme baseline de classification. Ce classifier renvoie toujours la classe masked (la classe la plus fréquente) quelque soit l'input.

	Precision	Recall	Count
Masked	77%	100%	445
No_mask	0%	0%	132

Les performances YOLOv4 / YOLOv3 (1/2)

- Darknet rend très facile l'évaluation de ses méthodes en permettant l'utilisation de la commande "map" qui permet d'afficher mAP, precision et recall pour chacun de nos algorithmes

	mAP	Precision	Recall	F1-score	Temps d'entrainement
YOLOv4	89%	84%	89%	86%	7h15
YOLOv3	96%	95%	90%	92%	2h30

Les performances YOLOv4 / YOLOv3 (2/2)

YOLOv4	mAP	TP	FP
Masked	84%	387	73
Unmasked	95%	126	27

YOLOv3	mAP	TP	FP
Masked	96%	400	23
Unmasked	97%	118	6

Script de prediction

- Une fois les poids entraînés nous pouvons sortir du notebook pour écrire des scripts pour faire des prédictions.

- Cette modularité est intéressante car elle nous permet de tester nos modèles facilement sur des données

- A noter que nous pouvons gérer 3 types d’input: des images, des vidéos et mêmes des flux vidéos live avec la webcam.

Merci de votre attention!