

# Projet YOLO

**Efkan TUREDİ**

# 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 devons 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**

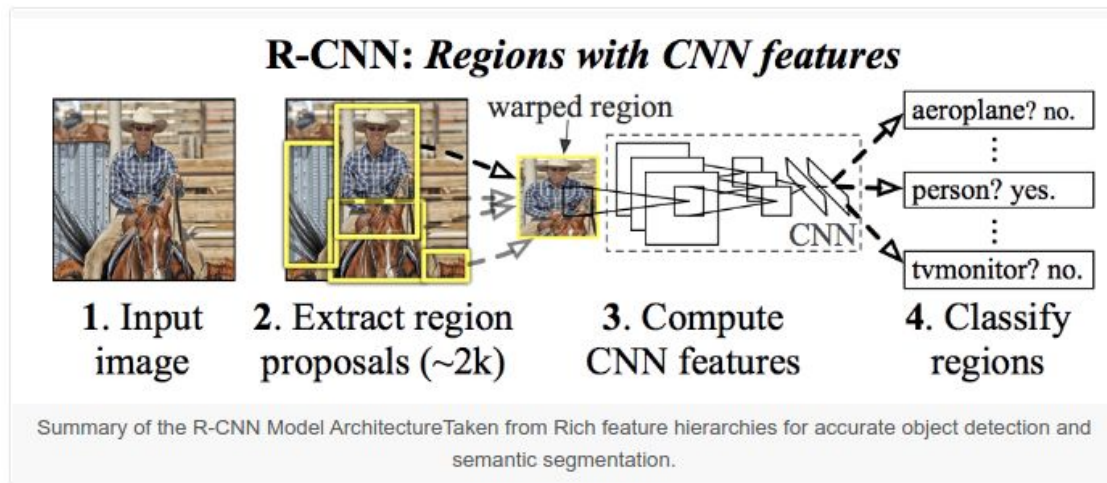
---

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

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

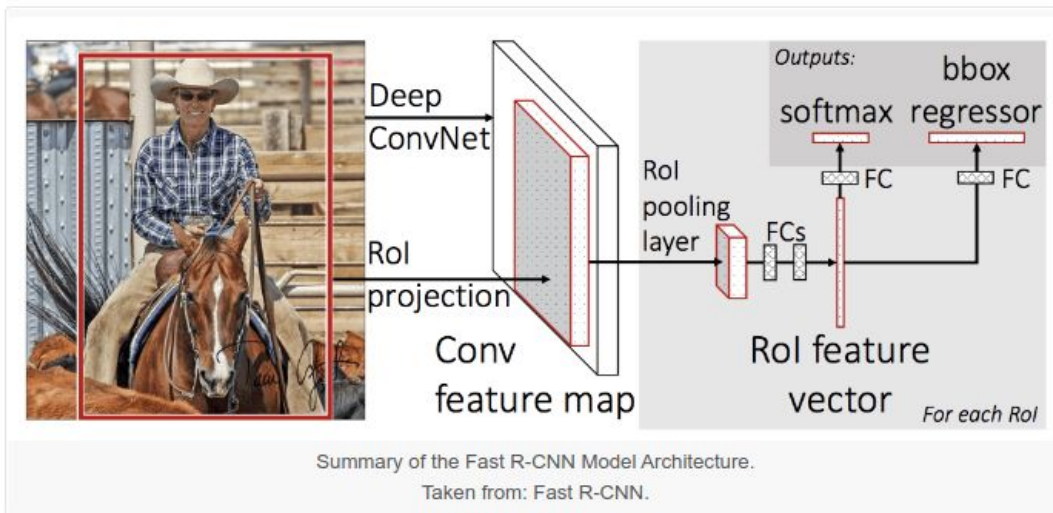
- Module 1: Proposition de régions. Génère et extrait des catégories indépendantes des régions c-a-d des contours candidats
- Module 2: Extraction de caractéristiques. Extraire des features de chaque région candidat à l'aide d'un CNN
- Module 3: Classification. Classifier les features obtenues dans l'une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



# 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

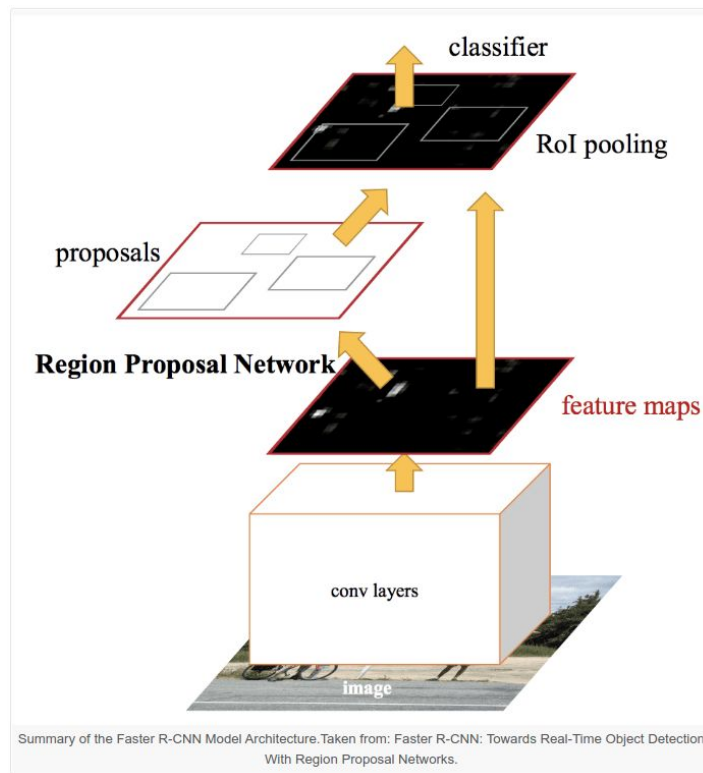




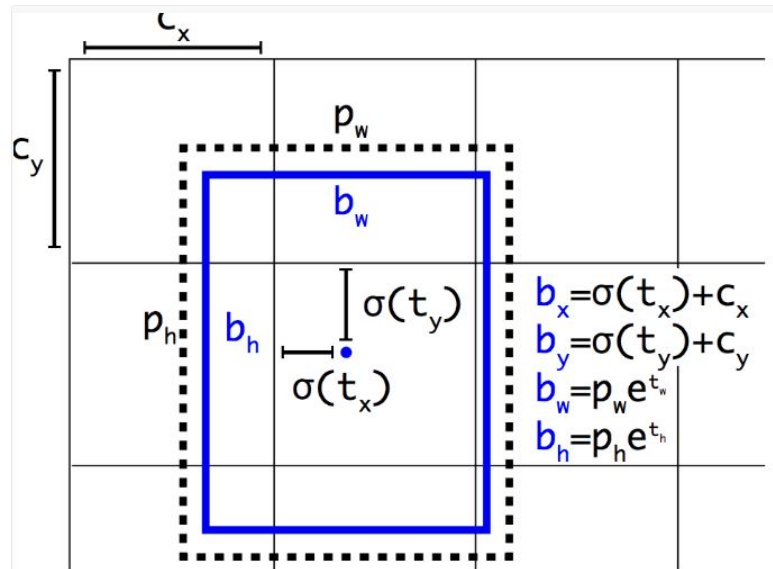
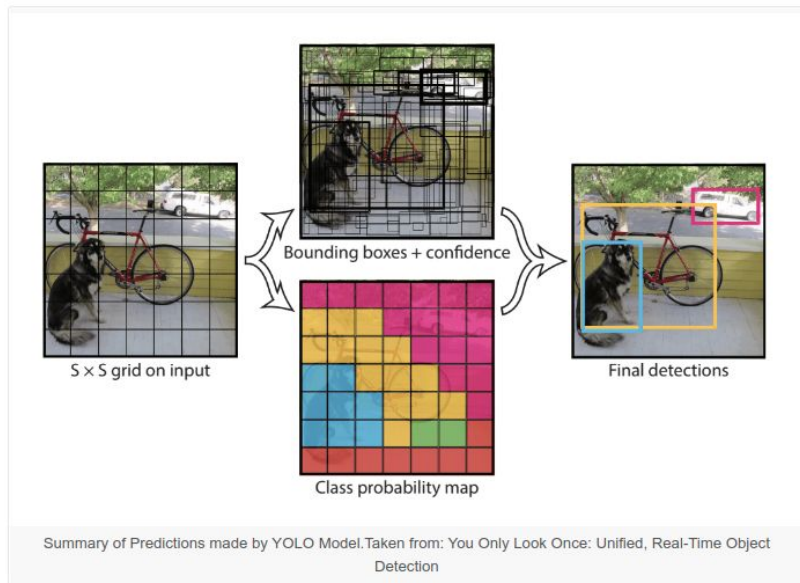
# La famille R-CNN: Faster R-CNN (3/3)

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



# La famille YOLO: de v1 à v4



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

# Dataset

---

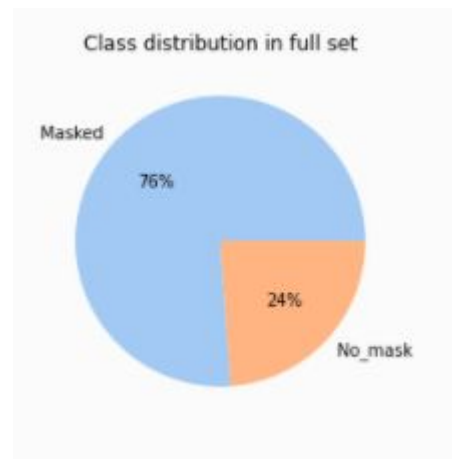
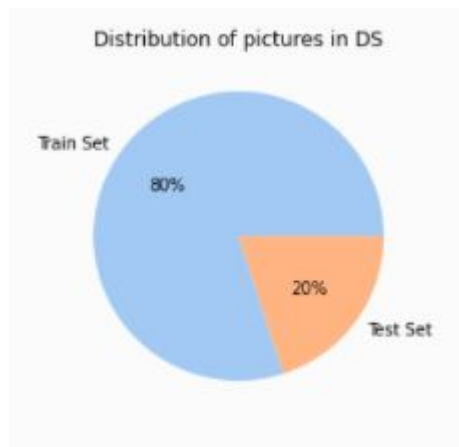
# Dataset Mask au format YOLO

- 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

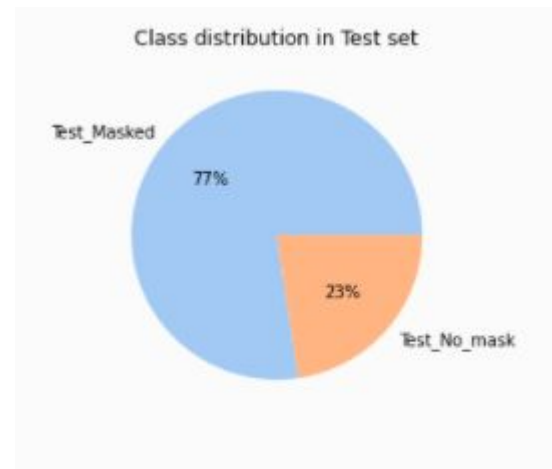
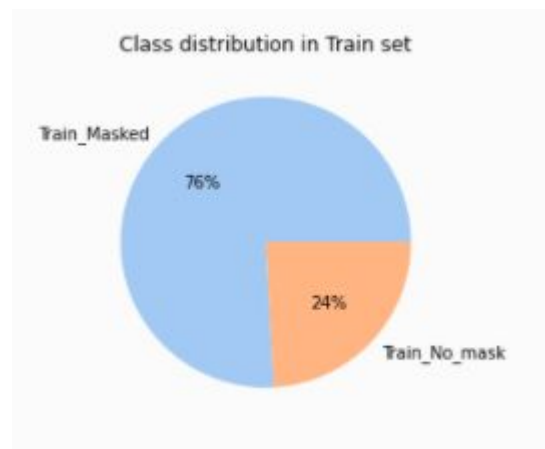


# Quelques graphiques sur le dataset

	Set_type	Masked_truth	No_mask_truth	Total_truth	#_of_samples
0	train	1511	480	1991	1208
1	test	445	130	577	302
2	full_dataset	1956	612	2568	1510



# Quelques details sur le Train et Test Set



# **Implémentation de notre 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/labellled_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/labellled_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:
  - $\text{Max\_batches} = \# \text{ de classes} * 2000 = 2 * 2000 = 4000$
  - $\text{Steps} = 80\%, 90\% * \text{max\_batches} = 3200, 3600$
  - Classes = 2 dans chaque couche YOLO
  - Filter = 24 (masks + ) \* dans chaque couche juste avant chaque YOLO

# Une dummy baseline

- Nous utilisons un dummy classifieur de scikit-learn que nous utilisons comme baseline de classification. Ce classifieur renvoie toujours la classe unmasked quelque soit l'entrée.

	mAP	Precision	Recall	F1-score
YOLOv4	[0.93]	[0.91]	[0.96]	[0.93]
YOLOv3	[0.88]	[0.94]	[0.90]	[0.92]

# Les performances globales

- YOLOv4 fournit une performance supérieure à YOLOv3, mais le temps d'entraînement est aussi très long: 8h contre 2h30 avec YOLOv3. Malgré tout nous retenons YOLOv4 car l'excédent de performance est non-négligeable.

	mAP	Precision	Recall	F1-score	Temps d'entraînement
YOLOv4	[0.93]	[0.91]	[0.96]	[0.93]	[8h]
YOLOv3	[0.88]	[0.94]	[0.90]	[0.92]	[2h30]

# Les performances par au niveau de la détection

- Comme montré dans la slide [XX], nous avons [XX] visage à détecter dans le test set. Notre modèle YOLOv4 en détecte au total [313], cela fait donc un ratio total de [XX] % de détecté.
- Concernant le split par classes nous avons les chiffres suivants
  - Notre test set possède au total: [XX] individus masqués et [xx] individus non masqués
  - L'algorithme détecte [71] personnes masqués, dont [50] TP et [21] FP. Prec moy = 88%
  - L'algorithme détecte [242] personnes masqués, dont [234] TP et [8] FP. Prec moy = 99%

# Script de prediction

- Une fois les poids entraînés nous pouvons sortir du notebook pour écrire des scripts pour faire des prédictions.
- Cette modularité est aussi 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!**

