Notes TP final robotique

**PROGRAMME DES TACHES A REALISER SUR LE PROJET**

**1 -** Programme capable de créer et de sauvegarder des modèles (avec data augmentation sur les données "panneau stop" du prof)

**2 -** Réaliser un max d'entrainements (en suivant un planning qu'on aurait mis en place par ex)

**3 -** Etre capable d'intégrer un modèle dans ROS (récupérable par le robot)

**4 -** Faire la documentation (10 pages pour expliquer ce qui a été fait, comparatif des modèles)

PLAN 17/01 – 19/01 🡺 Créer un programme capable de créer et sauvegarder des modèles (Pourra servir pour répartition tests lundi / mercredi)

**Intro** 🡺 Récupérer « code à compléter sur myges », l’extraire et exécuter « catkin\_make » (où se trouve le fichier README.md)

**A – Installation python dans robotic courses**

sudo apt-get install python3-pip python3-yaml

sudo pip3 install rospkg catkin\_pkg

//pip install ros-numpy 🡺 **NE FONCTIONNE PAS, IL VA FALLOIR VITE DEBUG**

sudo apt-get install python-catkin-tools python3-dev python3-numpy

//sudo apt-get install ros-melodic-ros-numpy 🡺 peut-être suffisant ? (je ne sais pas si la version est OK)

**B – Créer dossier spécial « robotics\_train\_sign\_data »** (<https://github.com/Dreamsplutox/robotics_train_sign_data>) récupérer les données de type « stop-sign » et les visualiser

**C – Créer un environnement python avec l’object detection API fonctionnelle**

(De mon côté j’ai clone mon environnement pour ne pas le perdre, et j’ai ajouté l’object detection)

conda create --clone <environment to clone> --name <new enviroment name>

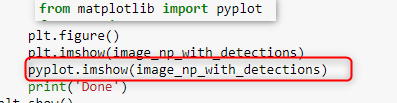
(install visual studio redis 2015 17 19 <https://support.microsoft.com/fr-fr/help/2977003/the-latest-supported-visual-c-downloads>)

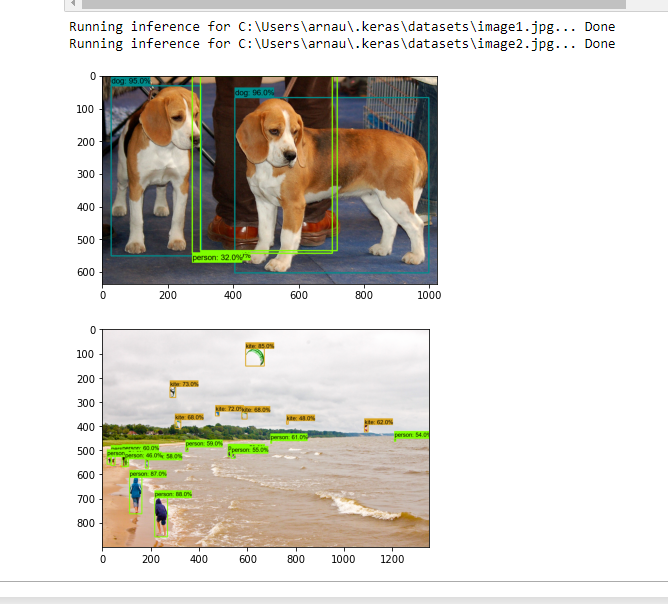
Installer CUDA 11.0 et CUDNN 11.1-windows-x64-v8.0.4.30 (utiliser word d’Antoine)

Commandes site du prof <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#tensorflow-object-detection-api-installation>

Tester notebook basique utilisant l’object detection :

<https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/auto_examples/plot_object_detection_saved_model.html#sphx-glr-auto-examples-plot-object-detection-saved-model-py>

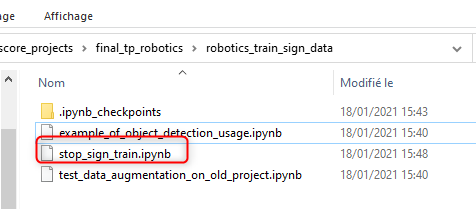


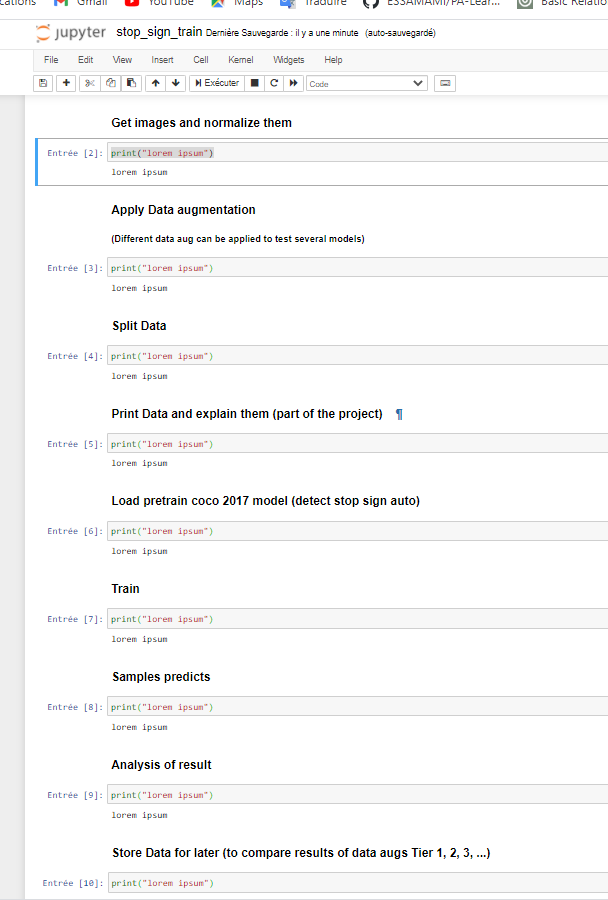


**D – Créer un notebook python (squelette)**

il sera utile pour réaliser l’ensemble des étapes (split, data aug, train, stock résultats dans des dossier pour un future analyse)

Notebook stop sign train :





**E – Récupération des données**

**Cloner ce projet Git**

<https://github.com/EscVM/OIDv4_ToolKit>

**Modifier fichier classes.txt**, supprimez tout et écrivez : **Stop sign**

**Téléchargez le dataset train des stop sign ainsi que le dataset validation puis mergez les 2 dans un dossier**

python3 main.py downloader --classes classes.txt --type\_csv train

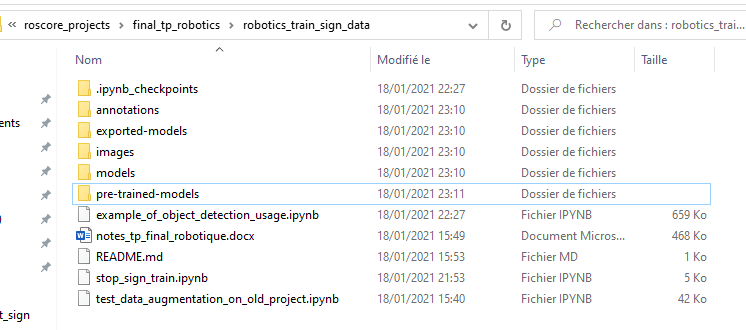
python3 main.py downloader --classes classes.txt --type\_csv test

Les images seront disponibles dans le dossier OID/dataset

*NB : Si vous voulez + de data, j’ai aussi récup 71 panneaux stop sur kaggle, demandez si vous êtes intéressés*

**F – Notebook en suivant les recommandations du tuto donné dans le TP** (<https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html#partition-the>)

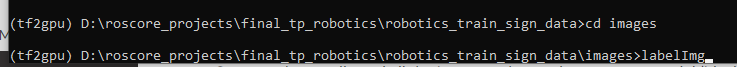
Architecture :



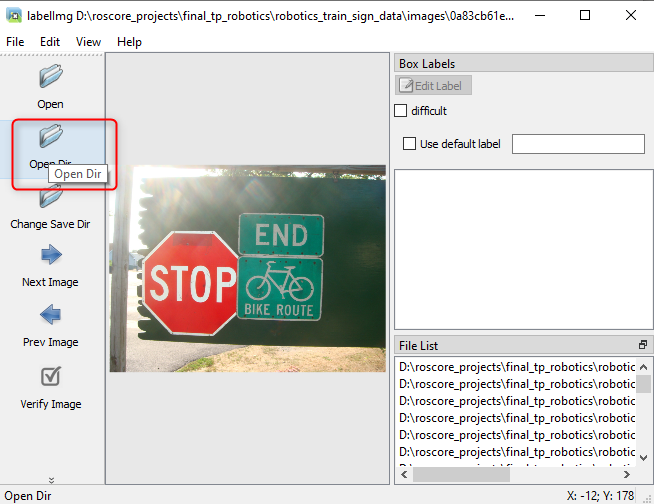
Installez LableImg en version 1.8.3

Placer les images dans le dossier « images »

Démarrez labelImg en tapant « labelImg » dans une CMD (dans le dossier images)



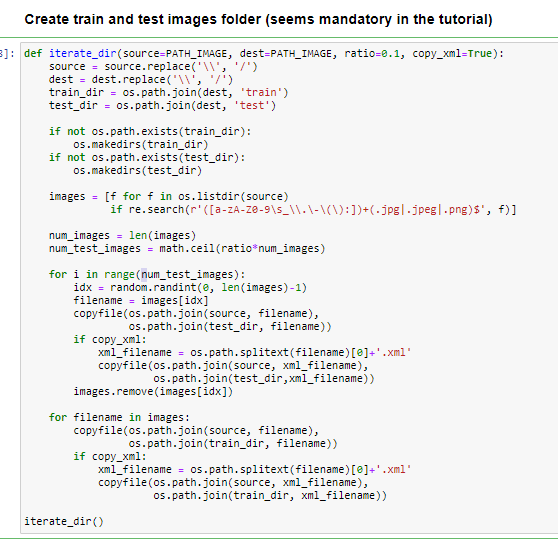
Cliquez sur Open Dir :

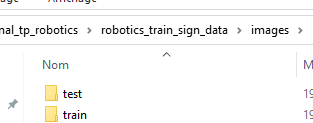


Attribuez des classes pour chaque image en sélectionnant le panneau stop (touche w)

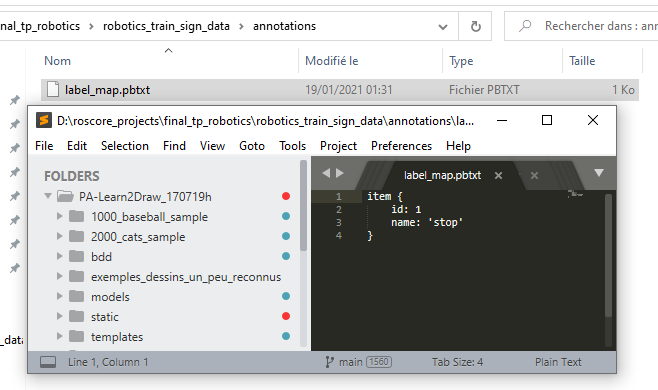
Je l’ai fait pour la partie OIDV4 (388 images)

Créer ensuite une fonction python qui va mettre en place les dossier d’image train et test (Tuto sans les inputs CMD)





Créer fichier label map qui associe chaque label à un nombre, ici on a qu’un seul label 🡺 stop



Création de tensorflow records à partir des données test et train

