Detection d’objet - Yolov8

1/ Construction du Dataset

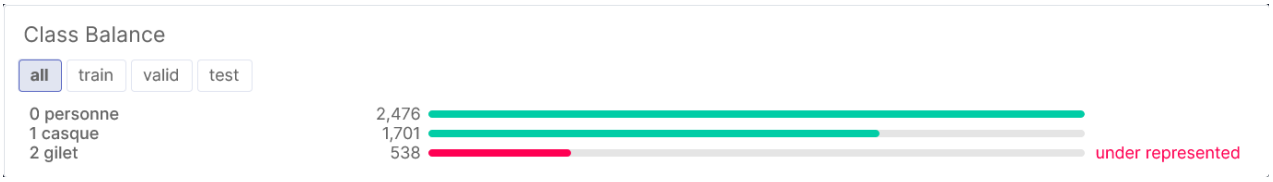
Nous voulons un modele capable de detecter, sur un flux video, si une personne porte bien un gilet de sécurite ET un casque.

Nous avons décider de créer un dataset contenant **3 classes** :

0 : personne
1 : casque
2 : gilet

from scrath

- web scrapping (01_scraping.py / 02_data_harvest.py)
- resizing (03_resize.py - YOLO prend des images en 640x640)
- labelisation au format YOLOV8 (pip install label-studio / makesense.ai)
- première version du Dataset :



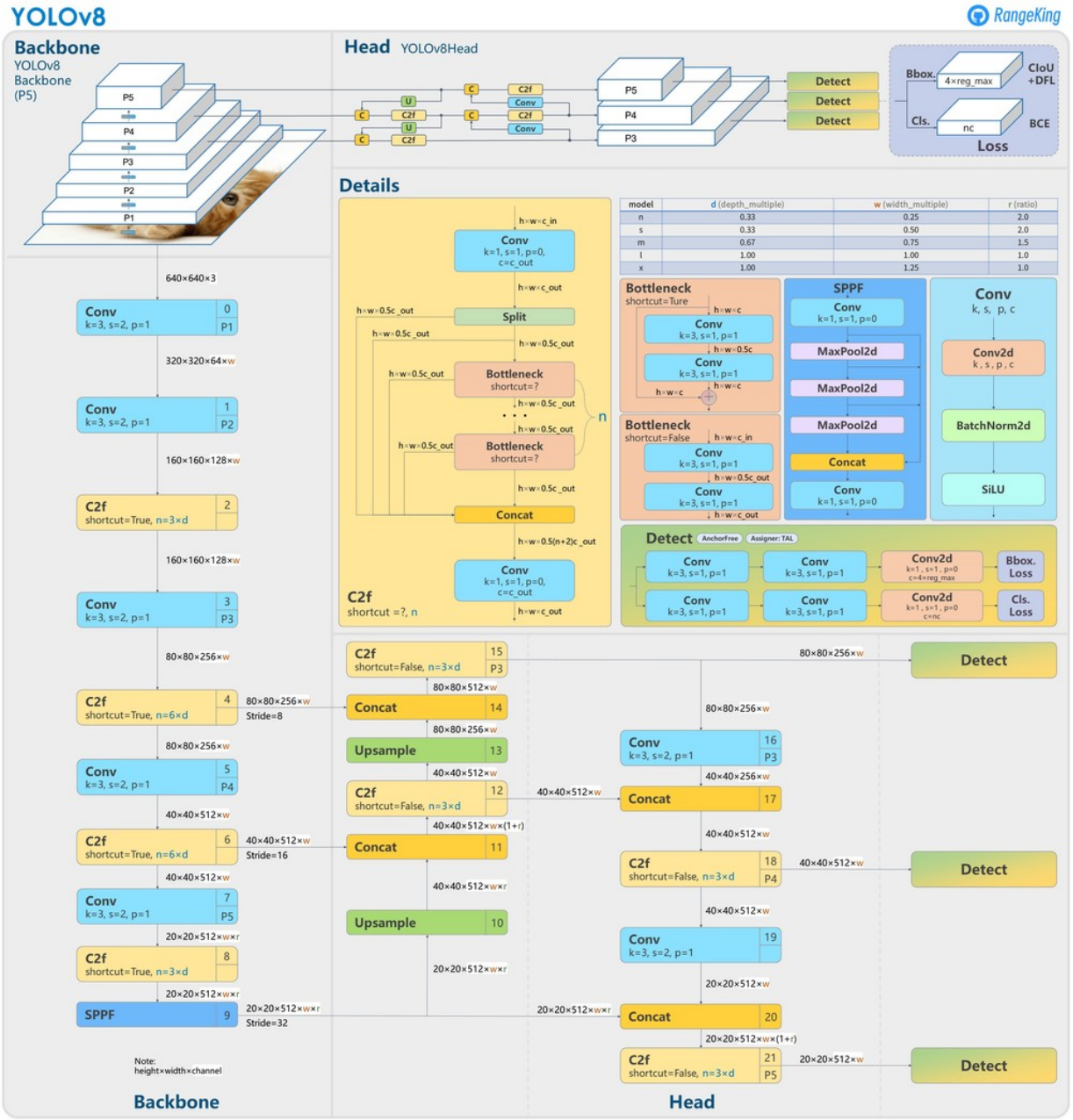
- ajout d’images contenant des gilets pour aider dans la répartition des classes
- upload, pre processing et train / eval / test split sur roboflow
- version finale : https://universe.roboflow.com/yolosafetygear/safety_gear_simplon/dataset/3

3/ Entrainement du modèle

choix du modèle : YOLOv8 de Ultralytics

YOLOv8 est disponible en tant que modèle gratuit et open source avec Python.

La biliothèque fournit des modèles pré-entraînés pour une variété de tâches de détection et de segmentation d'objets. Ces modèles peuvent être utilisés directement ou ils peuvent être ajustés sur un ensemble de données personnalisé.



Entrainement fait sur Google colab pour beneficier de la puissante de calcul necessaire

→ 05_train_yolov8.ipynb

4/ deployer le modèle dans une app Flask

→ 07_app.py

readme.md

```
├── 01_scraping.py
├── 02_data_harvest.py
├── 03_resize.py
├── 04_random_check.ipynb
├── 05_train_yolov8.ipynb
├── 06_test_display_results.py
├── 07_app.py

├── best_model
│   └── weights
│       └── best.pt

├── dataset_from_roboflow.yolov8
│   ├── test
│   ├── train
│   └── valid

├── requirements.txt

├── static
│   ├── css
│   ├── image_with_boxes.jpg
│   ├── test01.jpg
│   └── test02.jpg

├── templates
│   ├── results.html
│   └── upload.html
└── tree.py
```

Conclusion :

La doc de la bibliothèque python YOLO est vraiment très bien faite, le modèle est très simple à utiliser dans avec ses valeurs par défaut.