

Detection d'objet - YOLOv8

Plan :

1. Construction d'un dataset de 3 classes : personne, casque et gilet.
2. Entraînement d'un modèle YOLOv8 sur ce dataset.
3. Déploiement du modèle dans une application Flask.

1/ Construction du Dataset

Nous voulons un modèle capable de détecter, sur un flux vidéo, si une personne porte bien un gilet de sécurité ET un casque.

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

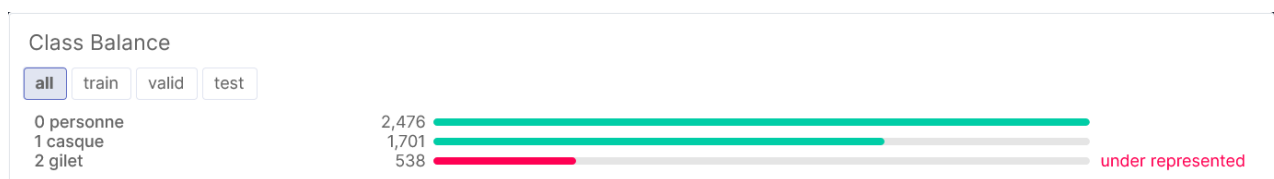
0 : personne

1 : casque

2 : gilet

from scratch

- web scraping (01_scraping.py / 02_data_harvest.py)
- resizing (03_resize.py - YOLO prend des images en 640x640)
- labélisation 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, pré traitement et train / eval / test split sur roboflow
- version finale :

https://universe.roboflow.com/yolosafetygear/safety_gear_simplon/dataset/3

2/ Entraînement du modèle

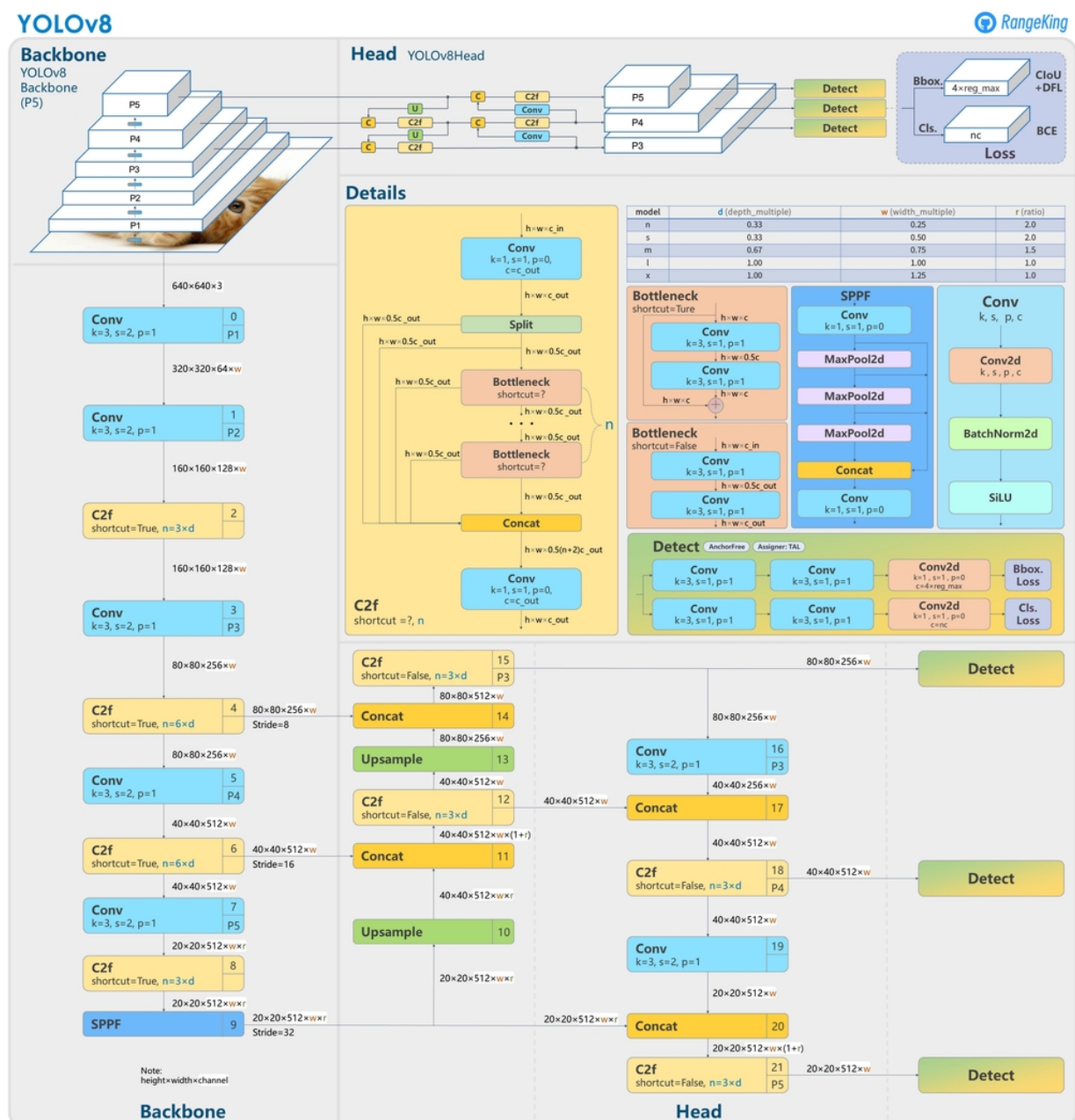
choix du modèle : YOLOv8 de Ultralytics

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

La bibliothèque 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.

La version 8 est sorti en 2023

Architecture :



Entrainement fait sur Google colab pour beneficier de la puissance de calcul
nécessaire <https://colab.research.google.com/drive/1haVoMxOHWRqEHvxCUKGXXVGnhcuMjPEo>

→ 05_train_yolov8.ipynb

(mAP50 = 0.826)

La métrique mAP50 signifie "Mean Average Precision" à un seuil IoU (Intersection over Union) de 50 %. Il s'agit d'une métrique d'évaluation largement utilisée dans les tâches de détection d'objets. La mAP50 mesure la précision du modèle pour détecter et localiser des objets dans une image.

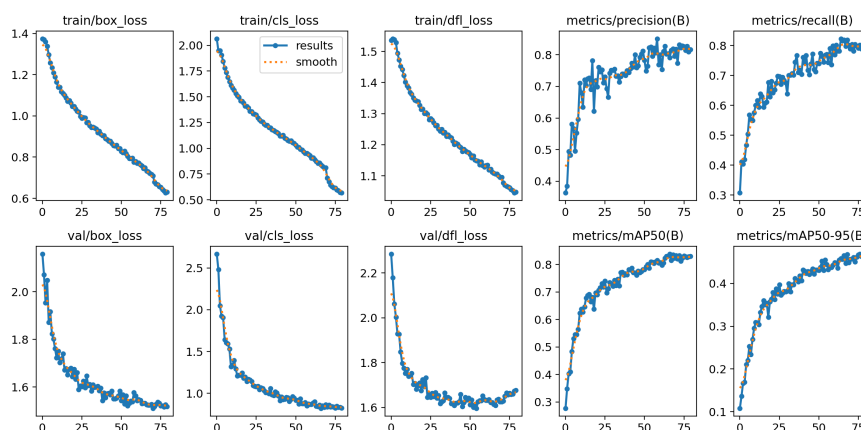
- La mAP50 varie de 0 à 1, où 0 indique une mauvaise performance et 1 une performance parfaite.
- Une mAP50 de 0,826 signifie qu'en moyenne, le modèle obtient une précision de 82,6% dans la détection et la localisation des objets dans l'ensemble de test en utilisant un seuil IoU de 50 %.

Matrice de confusion normalisée :



Malgrès un manque de gilet dans
notre dataset, le modèle s'en sort
plutôt bien.

Par contre, il a tendance à confondre
les casque avec le fond de l'image.



best_model/
weights/
results.png

3/ deployer le modèle dans une app Flask

→ app.py


Safety Gear Detection

with Ultralytics YOLOv8.

Object Detection Results

Real-time prediction of safety vest detection

Start Webcam Feed Stop Webcam Feed



0 personne : 0.78 casque : 0.75
2 gilet: 0.72

Wearing Safety Vest

0 personne - 0.78
1 casque - 0.75
2 gilet - 0.72



Powered by [Hatchi-Kin](#)

OBJECT_DETECTION_YOLOV8

```
|
|— app.py
|— tree.txt
|— rapport.pdf
|— readme.md
|— requirements.txt
|
|— best_model
|   |— confusion_matrix_normalized.png
|   |— weights
|   |   |— best.pt
|
|— model
|   |— 01_scraping.py
|   |— 02_data_harvest.py
|   |— 03_resize.py
|   |— 04_random_check.ipynb
|   |— 05_train_yolov8.ipynb
|   |— 06_test_display_results.py
|
|— static
|   |— image_with_boxes.jpg
|   |— test01.jpg
|   |— test02.jpg
|
|— templates
|   |— upload.html
```

Conclusion :

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

Les performances du modèle fine-tuned sur notre propre dataset sont plutôt bonnes (mAP50 = 0.826)

Et un utilisateur peut tester les capacités du modèle via sa webcam grace à l'application flask.

Reste à implémenter :

- Script python pour afficher les boites entourants les personnes en rouge si ces boites ne contiennent pas un casque ET un gilet, et toutes les autres boites en bleu.

~~- Trouver le moyen d'afficher la webcam et les predictions dans le navigateur.~~

