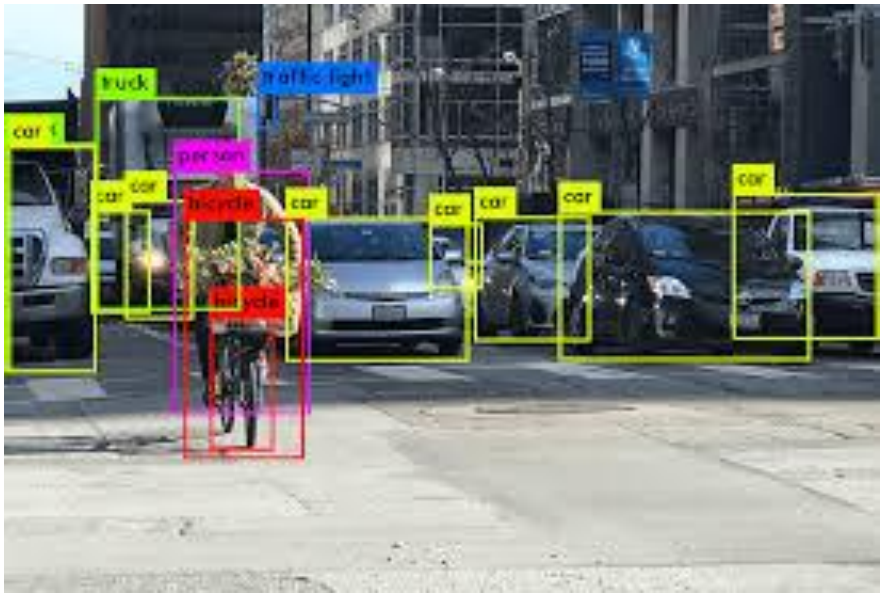
ATELIER Détection d'objets avec YOLOv8

Dataset : coco128

1) Introduction : Architecture de YOLO

YOLO signifie *You Only Look Once*.
Il s'agit d'une famille de modèles de détection d'objets dite « one-stage », car la prédiction des boîtes englobantes et des classes est réalisée en une seule passe.
Contrairement aux méthodes « two-stage » (ex : Faster R-CNN) qui passent par une étape de propositions de régions puis de classification, YOLO réalise directement la détection.

YOLO est donc particulièrement adapté aux applications temps réel.



1.1. Pipeline général

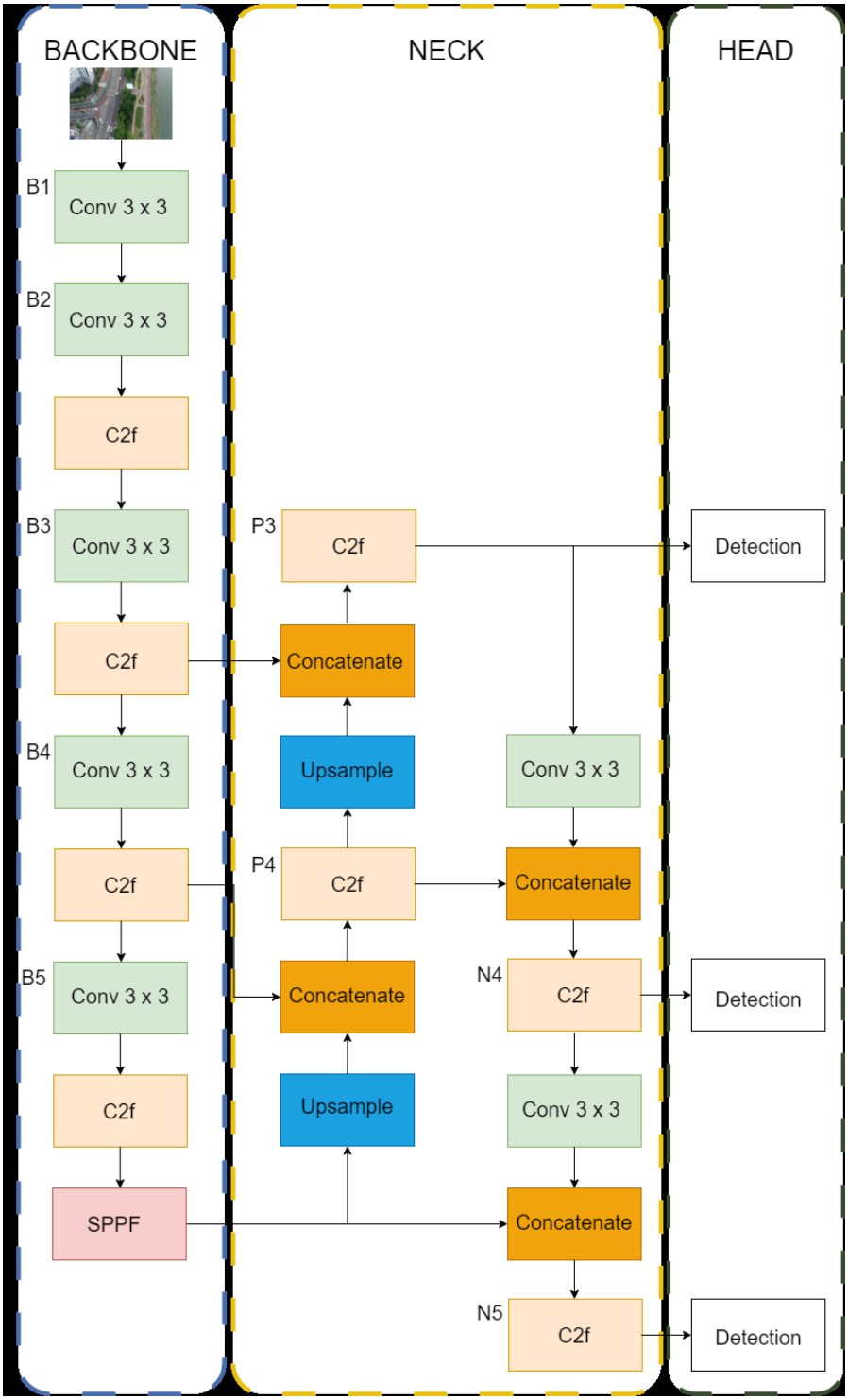


Figure 1: Architecture du modèle YoloV8

YOLOv8 est un modèle de détection d'objets qui fonctionne en une seule étape.
Il prend une image en entrée et renvoie directement les objets détectés, leurs classes et leurs positions.
Le pipeline de YOLOv8 se compose de plusieurs parties.

1. Premièrement, l'image d'entrée est redimensionnée et normalisée pour être prête à être traitée.

2. Deuxièmement, le *backbone* extrait des caractéristiques visuelles importantes, comme les formes, les textures et les contours. C'est un ensemble de couches de convolution qui transforme l'image en cartes de caractéristiques.
3. Ensuite, le *neck* combine les informations venant de différents niveaux du backbone. Cette fusion multi-échelle est utile pour détecter aussi bien les petits objets que les grands.
4. Puis la *head*, qui est de type anchor-free, prédit directement les boîtes englobantes, les scores de confiance et les classes des objets. C'est elle qui génère les détections brutes.
5. Pendant l'entraînement, des fonctions de perte comparent les prédictions du modèle avec la vérité terrain, afin d'ajuster ses paramètres.
6. Enfin, lors de l'inférence, une étape de post-traitement appelée NMS permet d'éliminer les doublons et de garder les meilleures prédictions.
7. En sortie, YOLOv8 renvoie une liste d'objets détectés avec leur catégorie, leur score de confiance et la position de leur boîte.

Résumé

- Backbone : extraction de caractéristiques
- Neck : fusion multi-échelles
- Head : prédictions (bbox + classe)
- NMS : suppression des doublons

Comparé aux méthodes two-stage, YOLO est plus rapide tout en offrant une bonne précision.

1) Objectifs pédagogiques

1. Comprendre le principe de la détection d'objets.
2. Utiliser Google Colab avec GPU.
3. Entraîner un modèle YOLOv8 sur coco128.
4. Évaluer les performances (mAP, précision, rappel).
5. Effectuer l'inférence sur images / webcam.
6. Exporter le modèle.
7. Créer une mini-démo interactive.

2) Prérequis

- Compte Google
- Bases Python
- Notions de base en apprentissage supervisé

3) Atelier Google Colab — Étapes

3.1. Préparer l'environnement

```
# Vérifier le GPU
import torch
print("Torch:", torch.__version__)
print("CUDA dispo:", torch.cuda.is_available())
if torch.cuda.is_available():
    print("GPU:", torch.cuda.get_device_name(0))

# Installer YOLOv8
!pip -q install ultralytics opencv-python matplotlib

from ultralytics import YOLO
YOLO
```

3.2. Dataset coco128

Dataset léger (128 images), multi-classes.
Téléchargé automatiquement par Ultralytics.

```
import os
print("Dataset coco128 sera téléchargé automatiquement lors de l'entraînement")
```

3.3. Entraînement (CLI)

```
!yolo detect train model=yolov8n.pt data=coco128.yaml epochs=20 imgsz=640 batch=16 device=0
```

3.4. Entraînement (Python)

```
from ultralytics import YOLO

model = YOLO("yolov8n.pt")
```

```
results = model.train(
   data="coco128.yaml",
   epochs=20,
   imgsz=640,
   batch=16,
   device=0 if torch.cuda.is_available() else 'cpu',
   patience=10,
   workers=2
)

print(results)
```

3.5. Évaluation + visualisation

```
import IPython, os, glob

runs = sorted(glob.glob("runs/detect/train*"))
run_dir = runs[-1]
print("Dernier run:", run_dir)

for img_name in ["results.png", "confusion_matrix.png", "PR_curve.png", "F1_curve.png"]:
   path = os.path.join(run_dir, img_name)
   if os.path.exists(path):
      display(IPython.display.Image(filename=path, width=800))
```

3.6. Validation (mAP)

```
!yolo detect val model={run_dir}/weights/best.pt data=coco128.yaml imgsz=640
```

3.7. Inférence sur images

```
import urllib.request, os, glob
from IPython.display import Image, display

os.makedirs("test_images", exist_ok=True)
url = "https://ultralytics.com/images/zidane.jpg"
urllib.request.urlretrieve(url, "test_images/zidane.jpg")

!yolo detect predict model={run_dir}/weights/best.pt source=test_images/ conf=0.25 save=True

pred_dirs = sorted(glob.glob("runs/detect/predict*"))
pred_dir = pred_dirs[-1]
pred_images = glob.glob(os.path.join(pred_dir, "*.jpg"))

for p in pred_images[:3]:
   display(Image(p, width=800))
```

3.8. Webcam

print("Webcam optionnelle, peut ne pas fonctionner sous Colab")

3.9. Mini-demo Gradio

```
!pip -q install gradio

import gradio as gr
from ultralytics import YOLO
import tempfile
import cv2, os

model = YOLO(f"{run_dir}/weights/best.pt")

def detect_objects(img):
    tmp = tempfile.NamedTemporaryFile(suffix=".jpg", delete=False)
    cv2.imwrite(tmp.name, cv2.cvtColor(img, cv2.COLOR_RGB2BGR))
    res = model.predict(source=tmp.name, conf=0.25, save=True,
project="runs/detect/gradio", name="pred", verbose=False)
    out_dir = res[0].save_dir
    outs = [p for p in os.listdir(out_dir) if p.lower().endswith((".jpg",".png",".jpeg"))]
    out_img = cv2.imread(os.path.join(out_dir, outs[0]))
    return cv2.cvtColor(out_img, cv2.COLOR_BGR2RGB)

demo = gr.Interface(fn=detect_objects, inputs=gr.Image(type="numpy"),
outputs=gr.Image(), title="YOLOv8 Demo")
demo.launch(share=True)
```

3.10. Export du modèle

```
!yolo export model={run_dir}/weights/best.pt format=onnx opset=12
!yolo export model={run_dir}/weights/best.pt format=torchscript
```

3.11. Sauvegarde Google Drive

```
from google.colab import drive
drive.mount('/content/drive')

!mkdir -p "/content/drive/MyDrive/YOLO_TP"
!cp -r {run_dir} "/content/drive/MyDrive/YOLO_TP/"

print("Run transféré dans Google Drive")
```

4) Rappels théoriques

Boîte (x, y, w, h)
Confiance = probabilité que la région contienne un objet
mAP = moyenne de la précision selon plusieurs seuils d'IoU
NMS = supprime les doublons

5) Exercices à rendre

1. Résultats d'entraînement
    - Fournir results.png, confusion_matrix.png
    - Analyse courte (5 à 7 lignes)
2. Effet de la taille d'image
    - imgsz = 512, 768
    - Comparer mAP, temps, VRAM
3. Comparaison de modèles
    - yolov8n vs yolov8s
    - Tableau comparatif
4. Variation du seuil
    - conf=0.25 vs 0.50
    - Observer FP / FN
5. Généralisation
    - Tester sur 3 images réelles
    - Analyse des erreurs

6) Livrables

`PDF ≤ 5 pages`
`Contenu :`

- Résultats

- Comparaisons
- Captures inférence

Fichiers :

- best.pt
- (optionnel) ONNX

7) Astuces

Problème : Mémoire
Solutions : réduire batch, imgsz, utiliser yolov8n

Problème : Pas de détection
Solutions : baisser conf

Problème : mAP faible
Solutions : augmenter epochs, modèle plus grand

Webcam ne fonctionne pas
Utiliser images ou Gradio

8) Extensions possibles

- Utiliser son propre dataset
- Ajouter suivi multi-objets
- Test vidéo