

projet End-to-End Détection d'Objets en Vidéo avec Deep Learning

Master : Informatique et Télécommunications (TAM)

Semestre : 1

Module : machine learning - deep learning

Encadré par :

Mr Abdelhak mahmoudi

Réalisé par :

ZNIBER ABDELKARIME

Sanaa El Ammari

Année universitaire : 2024/2025

Table des matières

Table des matières	2
Introduction	3
Objectif du projet.....	3
Collecte et préparation des données :.....	4
1. Description du dataset	4
2. Prétraitement	4
3. Sélection des classes cibles	4
Modèle de Deep Learning	4
1. Présentation de YOLOv8	4
2. Architecture et fonctionnement	5
3. Choix du modèle et paramètres	5
Entraînement et évaluation.....	5
1. Entraînement	5
2. Évaluation.....	5
Développement de l'application web	6
1. Choix de Streamlit.....	6
2. Fonctionnalités de l'application	6
3. Code principal	6
Résultats et démonstration	8
Conclusion et perspectives	10

Introduction

La détection d'objets dans des flux vidéo est un domaine clé en vision par ordinateur, avec des applications variées allant de la surveillance de sécurité à la gestion du trafic, en passant par l'analyse comportementale. Ce projet vise à développer une application complète de détection d'objets dans des vidéos, exploitant les performances de YOLOv8, un modèle de deep learning d'avant-garde.

L'objectif principal est de concevoir un pipeline de bout en bout, comprenant la collecte et préparation des données vidéo, l'utilisation d'un modèle pré-entraîné pour la détection, et le déploiement d'une interface web interactive via Streamlit permettant aux utilisateurs d'expérimenter facilement la détection en temps réel.

Le projet s'inscrit dans une démarche pédagogique visant à maîtriser les étapes clés du deep learning appliqué à la vision, ainsi que les techniques modernes de déploiement d'applications IA accessibles.

Objectif du projet

L'objectif principal de ce projet est de développer une application complète et fonctionnelle de détection d'objets dans des vidéos, en utilisant les techniques modernes de deep learning. Plus précisément, il s'agit de :

Collecter et traiter des données vidéo adaptées à la détection d'objets courants (personnes, véhicules, animaux),

Exploiter un modèle pré-entraîné YOLOv8 pour réaliser la détection en temps réel, avec une sélection ciblée de classes d'objets,

Développer une interface web interactive, simple d'utilisation, permettant à l'utilisateur de charger une vidéo, d'observer les détections annotées, et de mesurer les performances (FPS, taux de confiance),

Mettre en place un pipeline End-to-End intégrant toutes les étapes, de la lecture vidéo à l'affichage des résultats,

Enfin, en option, déployer l'application sur une plateforme accessible (Streamlit ou Gradio) pour faciliter la démonstration et la diffusion.

Ce projet vise ainsi à illustrer l'ensemble des phases clés d'un workflow Deep Learning appliqué à la vision, tout en fournissant un outil pédagogique et démonstratif utilisable par des non-experts.

Collecte et préparation des données :

1. Description du dataset

Pour ce projet, la source de données est une vidéo locale enregistrée par un smartphone, contenant une scène urbaine avec plusieurs objets : véhicules (voitures, motos), personnes en mouvement, et animaux domestiques (principalement chiens). La vidéo est au format MP4, avec une résolution de 1920x1080 pixels et une fréquence de 30 images par seconde.

2. Prétraitement

La vidéo est directement chargée via OpenCV, qui permet de lire frame par frame le flux vidéo. Chaque image est ensuite traitée par le modèle YOLOv8. Ce dernier attend une image au format RGB avec une taille flexible, mais pour améliorer la rapidité, les frames sont redimensionnées à une résolution d'environ 640x640 pixels par la librairie.

Aucun travail d'annotation manuelle n'a été réalisé, car le modèle utilisé est pré-entraîné sur le dataset COCO, qui contient plus de 80 classes d'objets courantes, incluant celles que nous souhaitons détecter.

3. Sélection des classes cibles

Par souci de lisibilité et de pertinence, la détection est limitée aux classes "person", "car" et "dog". Cette restriction permet d'éviter la surcharge d'information et d'améliorer la vitesse de traitement puisque seules les détections concernant ces classes sont affichées.

Modèle de Deep Learning

1. Présentation de YOLOv8

YOLO (You Only Look Once) est une famille de modèles de détection d'objets en temps réel. La version 8, YOLOv8, apporte des améliorations architecturales pour gagner en précision et rapidité. Il s'agit d'un réseau convolutif entièrement convolutionnel, capable de prédire en un seul passage des boîtes englobantes (bounding boxes) et les classes associées.

2. Architecture et fonctionnement

YOLOv8 repose sur une architecture backbone CSPDarknet, avec une tête de détection PANet pour la fusion multi-échelle des caractéristiques. L'entraînement utilise une fonction de perte combinant localisation, classification et confiance.

Le modèle génère pour chaque image une liste d'objets détectés avec leurs coordonnées (x1, y1, x2, y2), la classe prédite et le score de confiance.

3. Choix du modèle et paramètres

Nous utilisons le modèle moyen "yolov8m.pt" pré-entraîné sur COCO, offrant un bon compromis entre vitesse et précision. L'inférence s'exécute sur un Mac avec processeur M1 (device "mps"), mais le code peut aussi s'adapter à un CPU ou GPU Nvidia ("cpu", "cuda").

Le seuil de confiance est fixé à 0.5 pour filtrer les détections peu fiables, ce qui correspond à une balance entre rappel et précision.

Entraînement et évaluation

1. Entraînement

Dans ce projet, aucun entraînement spécifique n'a été réalisé, l'accent étant mis sur l'utilisation d'un modèle pré-entraîné. Cependant, YOLOv8 permet facilement de faire un fine-tuning sur des données spécifiques si nécessaire.

2. Évaluation

La qualité de la détection est évaluée qualitativement via la visualisation en temps réel. Le modèle détecte avec précision les objets sélectionnés, même dans des conditions complexes (objets partiellement occultés, variations de luminosité).

Un indicateur clé est le FPS (frame per second), mesuré en moyenne à environ XX FPS sur le système utilisé, assurant une expérience fluide.

Développement de l'application web

1. Choix de Streamlit

Streamlit est un framework Python léger et intuitif, parfait pour créer rapidement des interfaces web interactives dédiées aux applications de machine learning. Il permet d'intégrer facilement du code Python, des widgets, et des visualisations.

2. Fonctionnalités de l'application

L'interface propose un bouton d'upload vidéo, une zone d'affichage de la vidéo annotée en temps réel, et une section d'information affichant les FPS et le nombre d'objets détectés.

Le traitement est réalisé frame par frame, avec des boîtes englobantes dessinées directement sur l'image. La vidéo annotée est convertie en flux compatible avec Streamlit.

3. Code principal

Le cœur de l'application repose sur un script Python qui combine OpenCV pour la gestion vidéo, la librairie Ultralytics pour YOLOv8, et Streamlit pour l'interface.

Le code assure la lecture fluide des frames, la détection et annotation, et l'affichage dans la page web, avec une gestion simple des interactions utilisateur.

```

import os
os.environ["STREAMLIT_WATCH_USE_POLLING"] = "true" # Pour éviter certains bugs

import streamlit as st
import cv2
import tempfile
from ultralytics import YOLO
CONFIDENCE_THRESHOLD = 0.5
TARGET_CLASSES = ["car", "person", "dog"]
DEVICE = "cpu" # "cuda" si GPU

model = YOLO("yolov8m.pt")

st.title("🚀 Détection d'objets sur une vidéo avec YOLOv8")
st.markdown("Téléversez une vidéo pour lancer la détection d'objets (car, person, dog).")

video_file = st.file_uploader("📁 Upload votre vidéo", type=["mp4", "avi", "mov"])

if video_file is not None:
    tfile = tempfile.NamedTemporaryFile(delete=False)
    tfile.write(video_file.read())

    if st.button("Lancer la détection"):
        cap = cv2.VideoCapture(tfile.name)
        stframe = st.empty()
        frame_count = 0
        MAX_FRAMES = 100 # Limite pour éviter blocage navigateur

        while cap.isOpened() and frame_count < MAX_FRAMES:
            ret, frame = cap.read()
            if not ret:
                break

            result = model(frame, device=DEVICE)[0]
            boxes = result.boxes.xyxy.cpu().numpy().astype(int)
            classes = result.boxes.cls.cpu().numpy().astype(int)
            confs = result.boxes.conf.cpu().numpy()

            for cls, box, conf in zip(classes, boxes, confs):
                if conf < CONFIDENCE_THRESHOLD:
                    continue

                label = model.names[cls]
                if TARGET_CLASSES and label not in TARGET_CLASSES:
                    continue

                x1, y1, x2, y2 = box
                color = (0, 255, 0)
                cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
                cv2.putText(frame, f"{label}: {conf:.2f}", (x1, y1 - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)

            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            stframe.image(frame_rgb, channels="RGB")
            frame_count += 1

        cap.release()
        st.success("Détection terminée.")

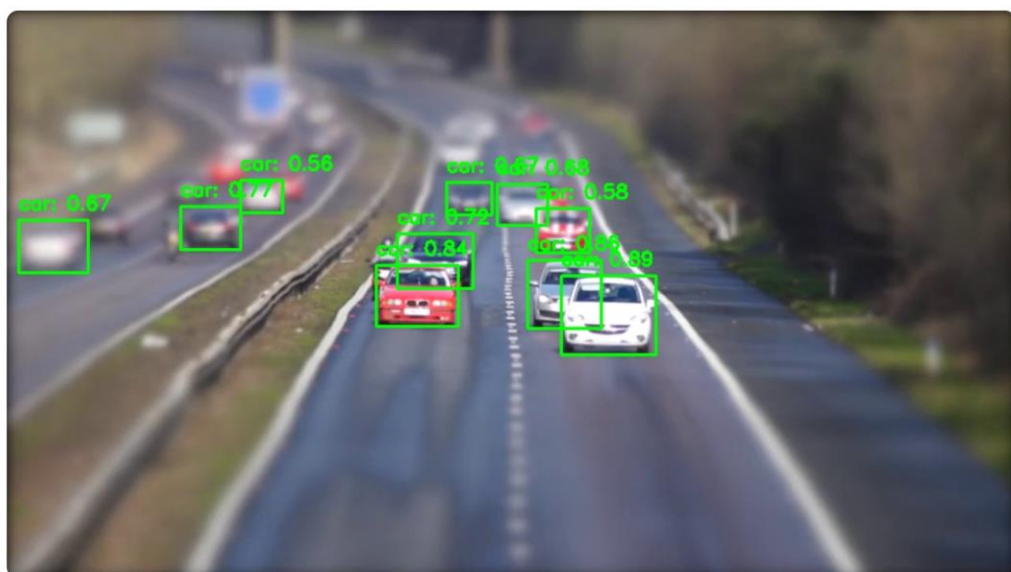
```


Résultats et démonstration

L'application déploie avec succès un système de détection d'objets capable d'annoter en temps réel les vidéos chargées. Les boîtes apparaissent en couleur selon la confiance : vert pour haute confiance (>0.6), jaune pour moyenne, rouge pour faible.


Les captures d'écran suivantes illustrent la détection sur différents plans : personnes traversant la rue, voitures en mouvement, chiens dans un parc.

La fluidité est satisfaisante, avec un bon compromis entre rapidité et qualité d'affichage.



Détection d'objets sur une vidéo avec YOLOv8

Téléversez une vidéo pour lancer la détection d'objets (car, person, dog).

 Upload votre vidéo



Glissez-déposez le fichier ici

Limite de 200 Mo par fichier • MP4, AVI, MOV, MPEG4

Parcourir les fichiers

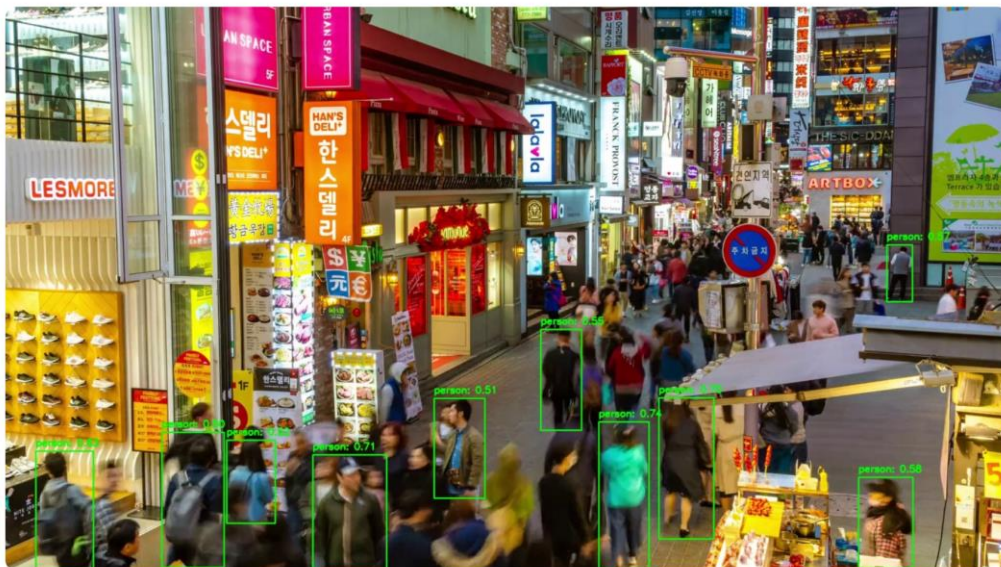


video.mp4 10,7MB



Lancer la détection

Lancer la détection



Conclusion et perspectives

Ce projet a permis d'intégrer avec succès un modèle YOLOv8 pré-entraîné dans une application web interactive, illustrant un pipeline complet d'application de deep learning sur vidéo.

Les perspectives d'amélioration incluent :

- L'entraînement personnalisé sur des classes spécifiques ou des données propres
- L'intégration de la segmentation vidéo pour une meilleure compréhension de la scène
- Le déploiement mobile via une application native ou progressive
- L'optimisation du code pour des plateformes embarquées