Institut National de Statistique et d'Economie Appliquée

Détection Automatique des Espaces de Stationnement avec YOLOv8
Rapport

Computer Vision

Élèves :
EL AMRI Youssef

31 octobre 2024

Table des matières

Table des figures

1 Introduction

La gestion des espaces de stationnement dans les zones urbaines est devenue un enjeu majeur, nécessitant des solutions innovantes pour améliorer l'expérience des conducteurs. Ce projet vise à développer un système de détection d'espaces de stationnement utilisant le modèle **YOLOv8**, capable d'identifier en temps réel les places disponibles et occupées. Le système affichera des informations cruciales sur un contrôleur, telles que l'état de la caméra, le nombre de places de stationnement disponibles, et des alertes en temps réel. En fournissant ces données, notre solution contribuera à réduire le temps de recherche de stationnement, optimisant ainsi l'utilisation de l'espace urbain.

2 Préparation de l'Environnement

Pour débuter le projet, il est crucial de préparer l'environnement de travail. Cela inclut l'installation des dépendances nécessaires, telles que **PyTorch, Ultralytics YOLOv8, et OpenCV**. Il est recommandé de créer un environnement virtuel pour isoler les bibliothèques et éviter les conflits de versions. Une fois les dépendances installées, il convient de vérifier que toutes les bibliothèques fonctionnent correctement en exécutant des scripts simples qui testent les versions installées.

- **PyTorch** : Framework open-source de deep learning développé par Facebook, utilisé pour créer et entraîner des modèles de réseaux de neurones grâce à ses capacités de calcul tensoriel et d'auto-différentiation.
- **Ultralytics YOLOv8** : Implémentation de la dernière version de YOLO (You Only Look Once), un modèle de détection d'objets en temps réel, optimisé pour sa rapidité et précision.
- **OpenCV** : Bibliothèque open-source de vision par ordinateur, utilisée pour le traitement d'images et de vidéos, ainsi que pour diverses tâches de reconnaissance et détection d'objets.

3 Collecte et Préparation des Données

Dans tout projet de détection d'objets, comme celui de détection d'espaces de stationnement, collecter et préparer les données est une étape cruciale. Elle consiste à rassembler des images pertinentes, les organiser, et les labéliser (annoter) correctement afin que le modèle de détection, comme YOLOv8, puisse être entraîné efficacement. Cette étape garantit que le modèle comprend et identifie correctement les objets d'intérêt, comme les voitures et les espaces de stationnement.

3.1 Processus de Labélisation

- Collectez des images variées du parking.
- Utilisez un outil comme **LabelImg ou robflow** pour labéliser les voitures et les espaces de stationnement.
- Enregistrez les annotations dans des fichiers texte au format YOLO.
- Organisez les images et annotations dans des dossiers spécifiques.
- Vérifiez les annotations pour garantir leur exactitude.

```
0 0.8515625 0.28671875 0.0734375 0.11796875
0 0.5296875 0.28984375 0.08046875 0.0984375
0 0.7734375 0.28984375 0.0828125 0.1171875
0 0.61015625 0.2921875 0.08046875 0.12109375
0 0.6921875 0.2921875 0.0890625 0.12578125
0 0.92734375 0.2921875 0.07421875 0.12578125
0 0.44765625 0.29921875 0.0796875 0.0984375
0 0.1953125 0.303125 0.0828125 0.109375
1 0.11484375 0.303125 0.08125 0.10078125
```

FIGURE 1 – labelisation

FIGURE 2 – dataset

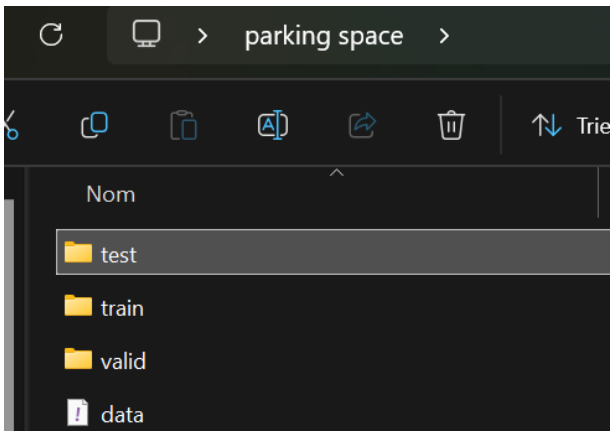

FIGURE 3 – structure de dataset

4 L'architecture de YOLOv8

YOLOv8 est un modèle de détection d'objets optimisé pour la précision et la rapidité, conçu pour s'adapter à une variété d'applications de vision par ordinateur nécessitant une reconnaissance d'objets rapide et efficace.

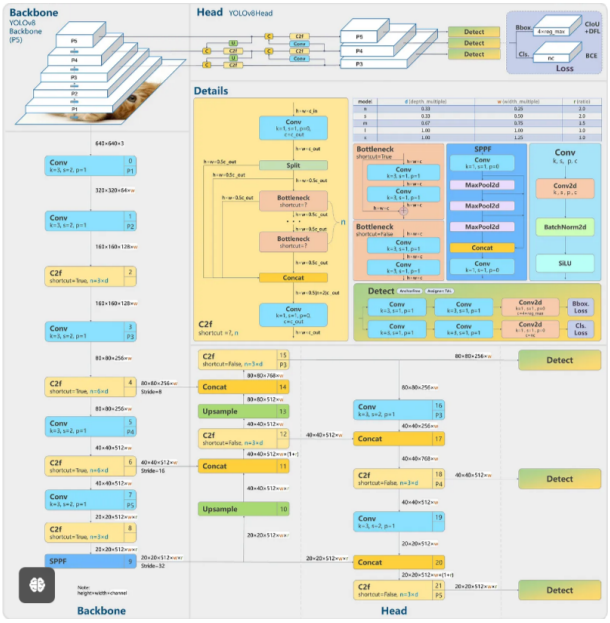

FIGURE 4 – architecture de YOLOv8

4.1 Backbone

Le **backbone** est responsable de l'extraction des caractéristiques à partir de l'image d'entrée. Il réduit progressivement la taille spatiale (hauteur et largeur) de l'image tout en augmentant la profondeur (canaux) pour capturer des caractéristiques riches. Il se compose de :

- **Conv** : Couches de convolution pour extraire les caractéristiques à différents niveaux.
- **C2f** : Bloc de convolution avancé avec des connexions de raccourci (*skip connections*) pour une meilleure transmission des gradients.
- **SPPF** : Bloc de pooling pyramidal, qui extrait des informations de contexte globales à plusieurs échelles pour améliorer la capacité à détecter des objets de différentes tailles.

4.2 Head

Le **head** transforme les caractéristiques extraites par le *backbone* en prédictions finales (boîtes englobantes, scores de confiance et classes d'objets). Il intègre des techniques comme :

- **Upsample** : Augmente la résolution des caractéristiques pour les combiner avec celles provenant de différentes couches.
- **Concat** : Combine les informations provenant de différentes échelles pour mieux gérer la détection multi-échelle.
- **Conv et C2f** : Couches de convolution supplémentaires pour affiner les caractéristiques avant de produire les prédictions finales.

4.3 Details

Cette section donne des informations sur les blocs et les modules utilisés dans le *backbone* et le *head* :

- **Bottleneck** : Bloc utilisé pour extraire des caractéristiques tout en maintenant des connexions résiduelles pour une meilleure propagation des informations.
- **SPPF** : Exploite plusieurs couches de pooling pour extraire des caractéristiques globales.
- **Conv2d**, **BatchNorm2d**, **SiLU** : Couches standards pour la convolution, la normalisation et l'activation dans les réseaux de neurones profonds.

4.4 Predictions et Losses

Le réseau génère des prédictions à différentes échelles :

- **Boîtes englobantes (Bbox)** : Prédictions des positions et dimensions des objets.
- **Classes (Cls)** : Prédictions des catégories des objets détectés.
- **Losses** : Utilise des fonctions de perte comme *BCE* et *CIOU* pour optimiser la précision des prédictions et la localisation des boîtes.

5 Fine-Tuning du Modèle YOLOv8

5.1 Concept de Fine-Tuning

Le fine-tuning implique l'ajustement d'un modèle pré-entraîné sur une tâche connexe pour le rendre plus performant sur votre tâche spécifique (dans ce cas, la détection d'espaces de stationnement). En utilisant un modèle pré-entraîné, vous bénéficiez des caractéristiques déjà apprises sur un large ensemble de données, ce qui permet d'accélérer le processus d'entraînement et d'améliorer la précision, surtout si vous disposez d'un ensemble de données limité.

5.2 résultat du fine-tuning

5.2.1 matrice de confusion

La matrice de confusion est un outil utilisé pour évaluer les performances d'un modèle de classification dans votre projet de détection de places de parking. Elle représente la relation entre les classes prédites et les classes réelles (ou vraies).

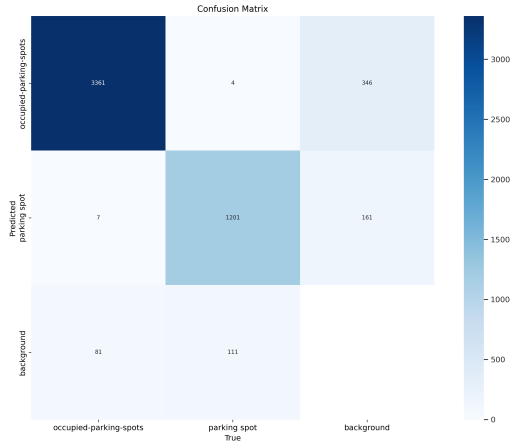

FIGURE 5 – matrice de confusion

La matrice de confusion montre que le modèle de détection des places de parking fonctionne efficacement pour identifier les places occupées, avec 3361 prédictions correctes. Toutefois, quelques erreurs apparaissent, notamment 81 places occupées mal classées comme arrière-plan. Concernant les places de parking disponibles, le modèle a correctement identifié 1201 d'entre elles, mais a commis des erreurs, classant 111 places disponibles comme arrière-plan et 7 comme des places occupées. En ce qui concerne l'arrière-plan, bien que 161 arrière-plans aient été correctement identifiés, le modèle a confondu 346 arrière-plans avec des places occupées et 4 avec des places disponibles.

5.2.2 courbe Precision-Recall

La courbe Precision-Recall montre que le modèle de détection des places de parking affiche de très bonnes performances. Pour la catégorie des places de parking occupées, la précision est de 0,988, ce qui indique une très haute fiabilité dans la détection des places réellement occupées. La catégorie des places disponibles a une précision légèrement inférieure à 0,966, mais reste globalement performante. En moyenne, le modèle obtient un mAP@0.5 de 0,977, ce qui reflète une excellente précision et un rappel global sur toutes les classes. Cela montre que le modèle est très efficace dans la détection des places de parking, avec peu d'erreurs.

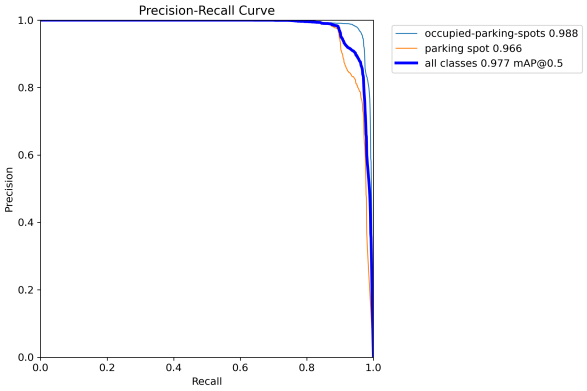

FIGURE 6 – La courbe Precision-Recall

5.2.3 résultats

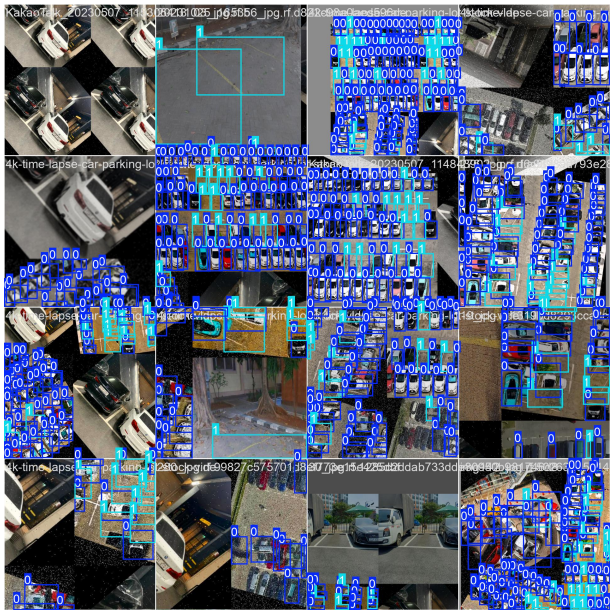

FIGURE 7 – le résultats du fine-tuning

6 Conclusion

Ce projet a démontré l'efficacité de YOLOv8 pour la détection en temps réel des places de stationnement disponibles et occupées. Grâce à un processus de collecte, labélisation, et fine-tuning des données, le système peut améliorer la gestion des parkings en réduisant le temps de recherche pour les conducteurs et en optimisant l'utilisation des infrastructures. Cette solution pose les bases pour des applications futures dans les systèmes de parkings intelligents et les environnements urbains connectés.