ÉCOLE NATIONALE D'INGÉNIEURS DE TUNIS
Département Génie Electrique

Détection d'objets avec IA en périphérie (Edge AI)

Étudiants : Houssem-eddine Lahmar & Cheikh Brahim

Ahmed Jebbe

Encadrante : Dr Faten Ben Abdallah

3e année Génie Électrique – spécialité SMART

Année universitaire 2025/2026

Contexte et objectifs du projet

- **Edge AI = déployer les modèles d'IA directement sur des dispositifs embarqués / objets connectés (IoT).**
- **Passer d'une informatique centrée sur le cloud à une informatique centrée sur l'edge .**
- **Plateforme cible : NVIDIA Jetson Nano (GPU 128 cœurs, 4 Go de RAM).**
- **Application : détection d'objets en temps réel pour véhicules intelligents et robots.**
- **Objectif : démontrer qu'une détection d'objets à l'état de l'art est possible sur du matériel basse consommation, avec une latence et une précision acceptables.**

Problématique et motivations

Comment exécuter une détection d'objets précise en temps réel sur un dispositif edge aux ressources limitées ?

Défis :

- **Puissance de calcul limitée (472 GFLOPS en FP16 contre des GPU de data center beaucoup plus puissants).**
- **Seulement 4 Go de mémoire unifiée, partagée entre le CPU et le GPU.**
- **Nécessité d'une bonne précision (mAP ≥ 0,5–0,65) pour des tâches critiques en matière de sécurité.**
- **Contraintes énergétiques strictes (5–10 W) et limites thermiques.**

→ **Ce type de solution est essentiel pour les systèmes autonomes, les villes intelligentes, l'industrie et la santé, où faible latence, confidentialité des données et fiabilité sont indispensables.**

Pourquoi l'Edge AI plutôt que le Cloud ?

Limites d'une IA uniquement basée sur le cloud :

- **Latence réseau de 50 à 200 ms ou plus par image.**
- **Dépendance à la connectivité Internet (inacceptable pour les robots / voitures).**
- **Problèmes de confidentialité lorsque les images / vidéos brutes sont envoyées vers des serveurs externes.**
- **Coût élevé en bande passante pour le streaming continu de vidéo HD.**

Avantages de l'Edge AI sur Jetson Nano :

- **Traitement local : latence totale de 10 à 50 ms par image.**
- **Fonctionne même sans réseau, 24h/24 et 7j/7.**
- **Les images restent sur le dispositif → meilleure confidentialité et conformité plus simple aux réglementations.**
- **Seuls les événements détectés / métadonnées sont envoyés sur le réseau (énormes économies de bande passante).**

Pourquoi la NVIDIA Jetson Nano ?

- **GPU Maxwell 128 cœurs avec support CUDA : indispensable pour exécuter des CNN.**
- **Moteur TensorRT pour l'inférence optimisée (fusion de couches, FP16/INT8).**
- **Mémoire unifiée de 4 Go, avec zero-copy entre CPU et GPU.**
- **Pile logicielle complète (JetPack, CUDA, cuDNN, TensorRT, OpenCV).**
- **Faible consommation énergétique (environ 7–10 W pendant l'inférence).**
- **Carte abordable, facile à intégrer, avec des interfaces caméra pratiques.**

Détection d’objets avec YOLOv8

- **YOLOv8 : détecteur mono-étape, rapide et précis pour une utilisation en temps réel.**
- **Entrée : images RGB redimensionnées autour de 640×640 ou 416×416 pixels.**
- **Sortie : boîtes englobantes + labels de classe + scores de confiance.**
- **Classes cibles dans le projet : voiture, feu de signalisation, bicyclette, personne, etc.**
- **Entraînement sur un jeu de données de conduite urbaine avec objets annotés.**
- **Évaluation avec des métriques telles que mAP@0.5, précision, rappel et FPS.**

Approche unique de YOLO

L'innovation clé de YOLO réside dans son architecture à passage unique. L'algorithme traite l'image entière une seule fois à travers le réseau de neurones pour détecter les objets, d'où le nom « You Only Look Once ».

Cette approche offre plusieurs avantages importants :

- **Vitesse : l'architecture à passage unique permet une détection d'objets en temps réel**
- **Efficacité : elle élimine les calculs redondants associés aux multiples passages**
- **Contexte global : elle prend en compte l'image entière simultanément, ce qui améliore la précision de détection**

Ces caractéristiques ont contribué à l'adoption massive de YOLO aussi bien dans la recherche que dans les applications industrielles.

Vue d’ensemble du jeu de données

Pour cette implémentation, nous avons utilisé le jeu de données Self-Driving Cars disponible sur Kaggle[1].

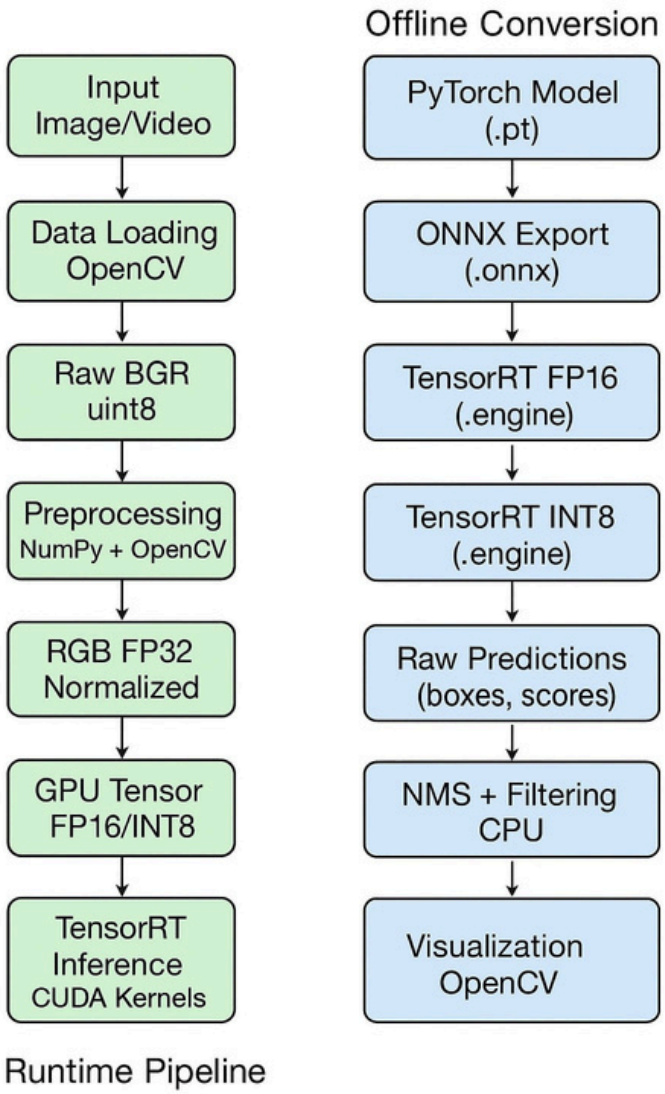
Ce jeu de données constitue une excellente ressource pour l’entraînement et la mise en pratique de la détection d’objets avec YOLO, en particulier dans le contexte de la conduite autonome.

Caractéristiques du jeu de données

Le jeu de données contient des images annotées avec cinq classes d’objets distinctes

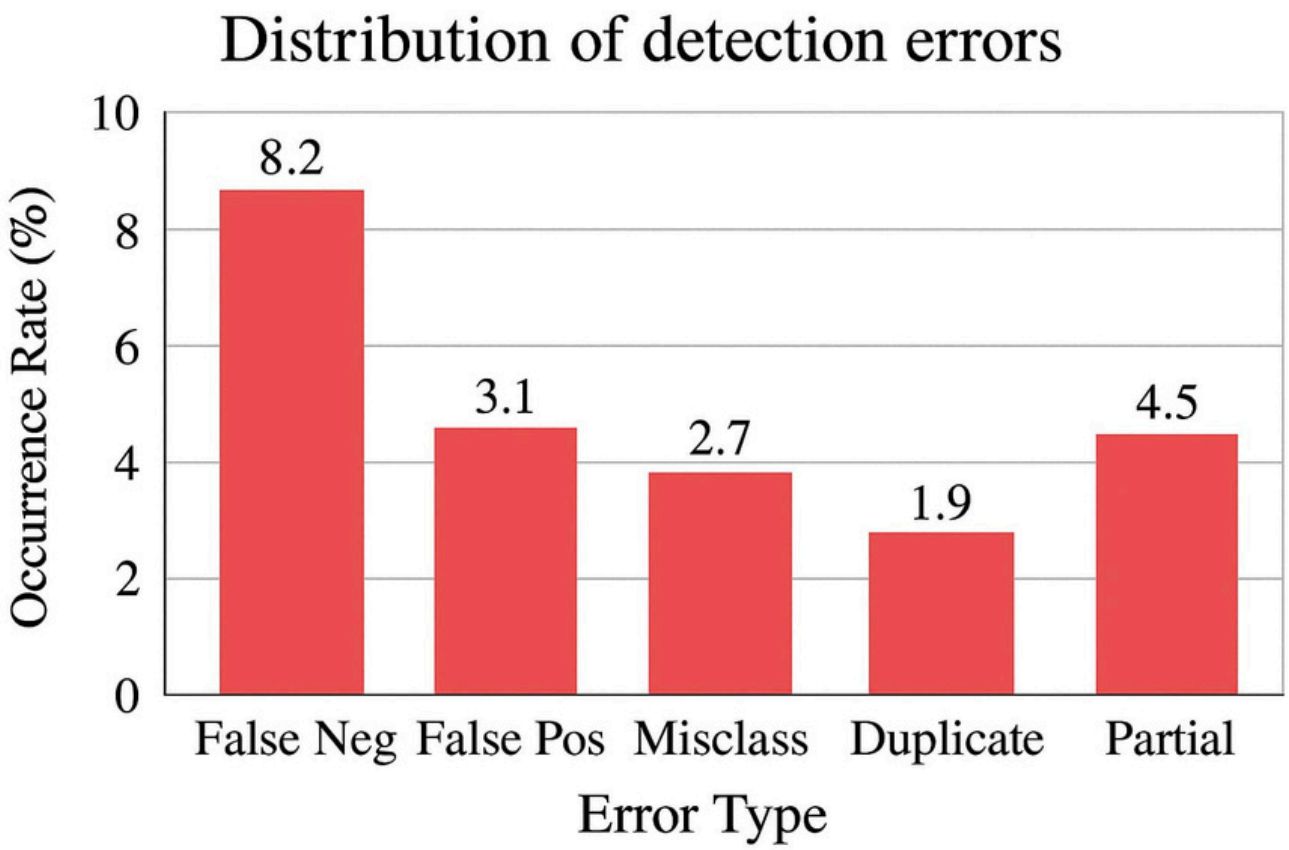
- **voiture**
- **camion**
- **piéton**
- **cycliste**
- **feu (tricolore / signal lumineux)**

Pipeline d’inférence et de conversion (TensorRT)



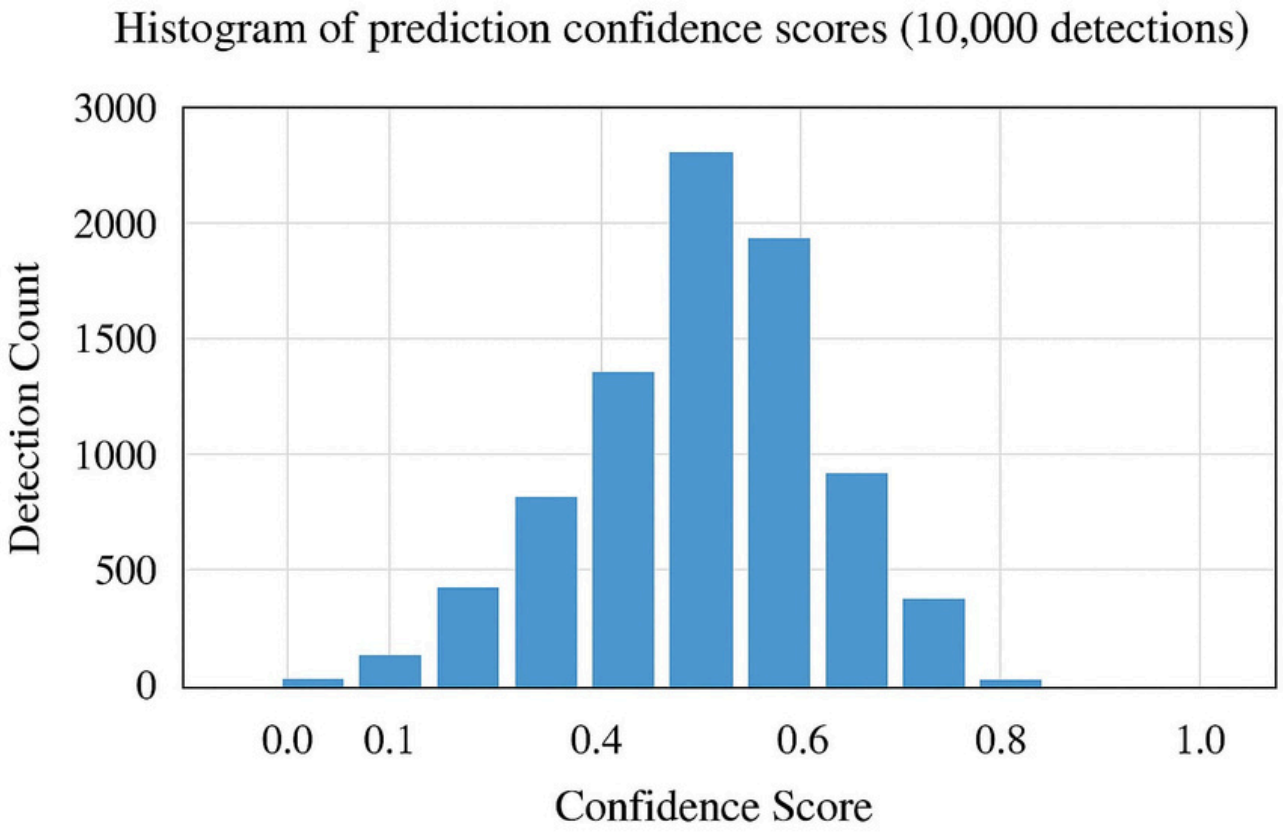
Ce schéma résume toute la chaîne : conversion hors ligne du modèle PyTorch vers TensorRT (FP16/INT8), puis pipeline temps réel sur Jetson Nano (acquisition, pré-traitement, inférence GPU, NMS et visualisation) pour obtenir une détection rapide et optimisée en edge.

Distribution des erreurs de détection



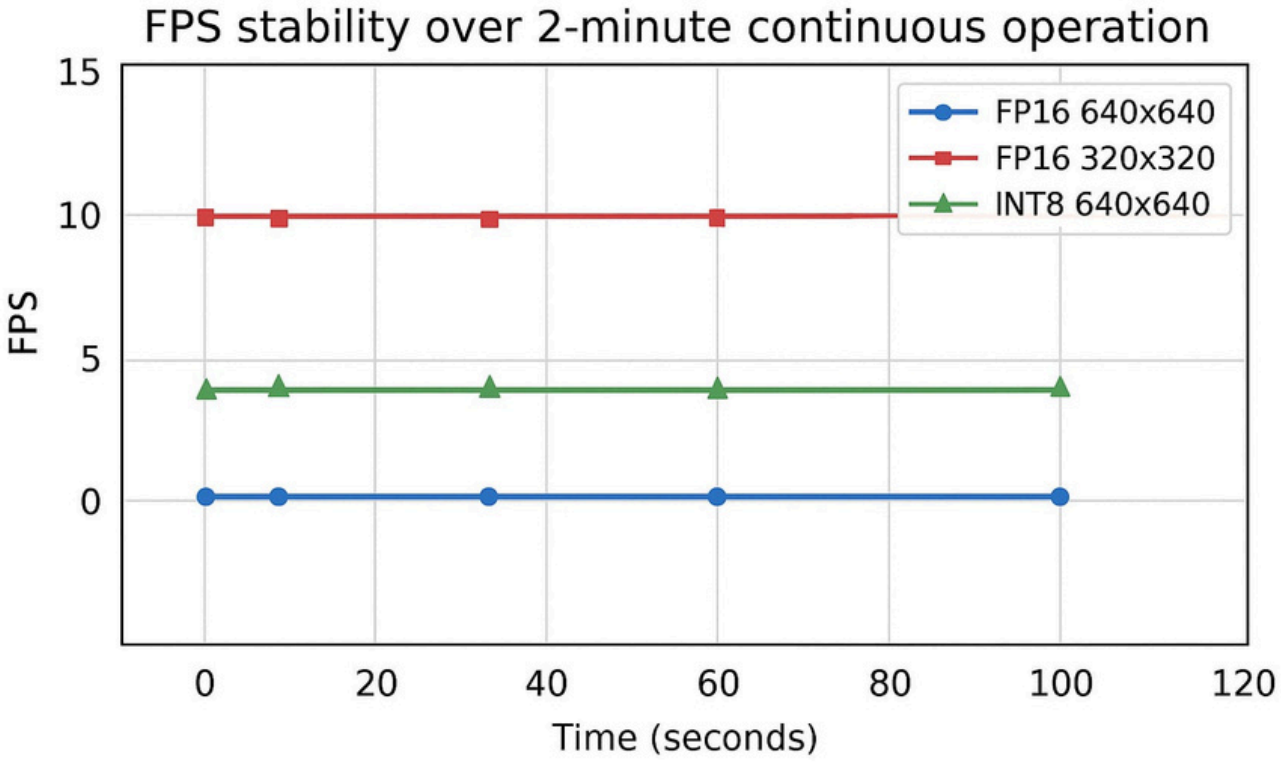
Les erreurs les plus fréquentes sont les faux négatifs, suivis des détections partielles et des faux positifs, ce qui montre que le modèle manque encore certains objets ou ne les cadre pas parfaitement, surtout dans les scènes complexes.

Histogramme des scores de confiance des prédictions



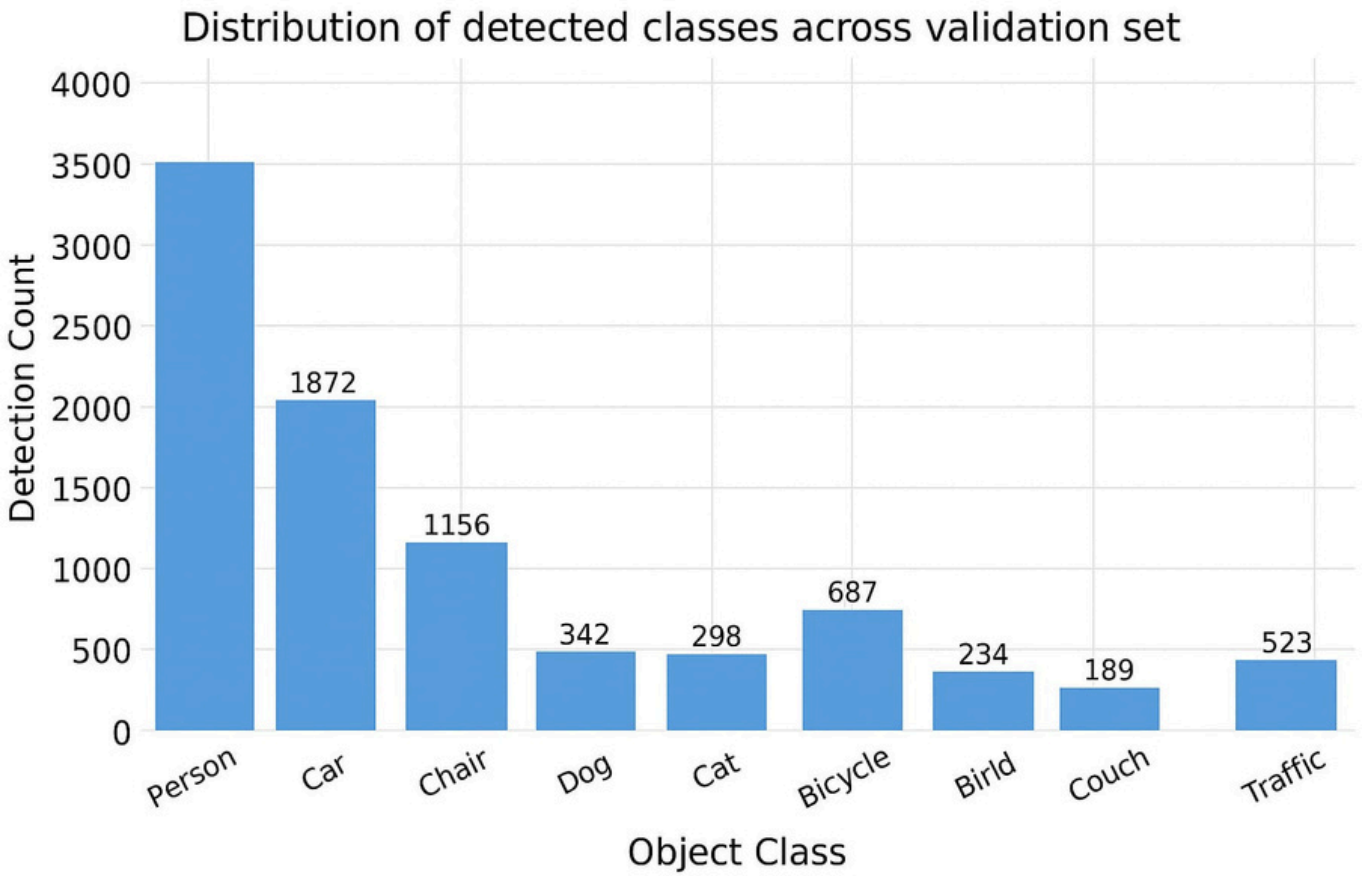
La plupart des prédictions ont un score de confiance compris entre 0,4 et 0,6, ce qui indique que le modèle produit globalement des détections sûres, tout en gardant une certaine marge d'incertitude sur les cas difficiles.

Stabilité du débit (FPS) sur 2 minutes



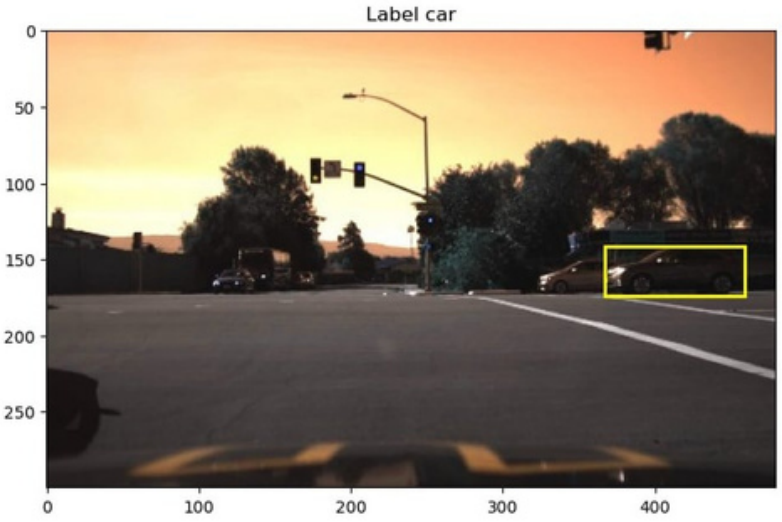
Les FPS restent quasiment constants pendant 2 minutes, ce qui prouve que le système est stable dans le temps et que la Jetson Nano supporte un fonctionnement continu sans chute de performances.

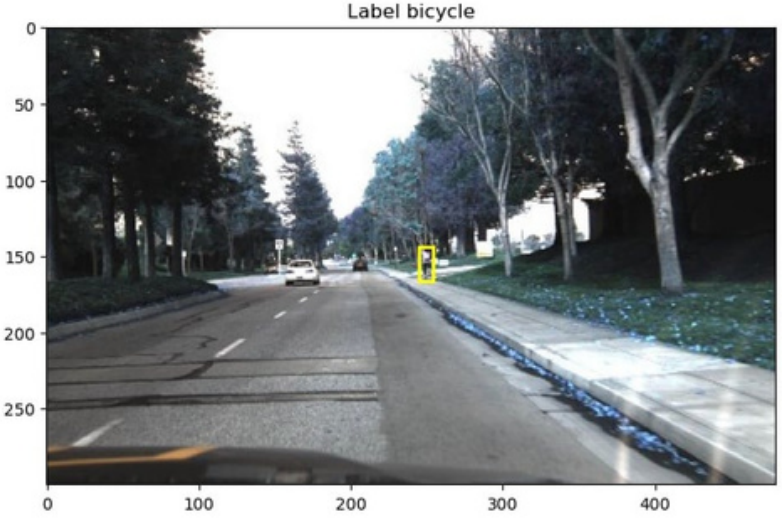
Distribution des classes détectées sur le jeu de validation

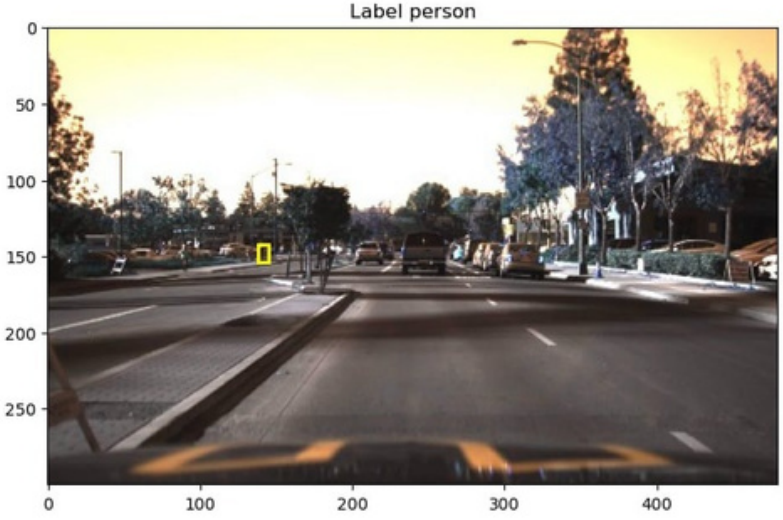


La majorité des détections concerne la classe personne, suivie des voitures et des chaises, ce qui reflète la composition urbaine du jeu de validation et montre que le modèle voit surtout des scènes avec des piétons et du trafic.

Exemples d’images annotées







Exemples d’annotations pour les classes voiture, feu de signalisation, bicyclette et personne.

Optimisation et déploiement Edge AI

- **Objectif : exécuter YOLOv8 en temps réel sur Jetson Nano pour la détection d'objets.**
- **Passer du modèle PyTorch de base à une version optimisée pour l'embarqué.**
- **Réduire la latence et la consommation tout en gardant une bonne précision.**
- **Valider que la plateforme peut supporter une application continue 24/7.**

Déploiement et évaluation (aperçu)

- Plateforme de déploiement : système embarqué sur Jetson Nano avec un modèle YOLO optimisé par TensorRT.
- Évaluation temps réel : tests sur un flux caméra pour mesurer la latence, le nombre de FPS, l'utilisation des ressources (CPU/GPU/RAM) et la température.
- Métriques de performance : évaluation de la qualité de détection avec la mAP, la précision et le rappel.
- Résultat principal : l'Edge AI permet une détection d'objets en temps réel sur un matériel peu coûteux.
- Condition de réussite : nécessité d'optimisations (FP16/INT8, TensorRT, pré-traitement soigné) pour atteindre de bonnes performances.
- Domaines d'application : solution adaptée à la robotique, aux véhicules intelligents et aux applications de smart city.

Conclusion & Perspectives

Conclusion :

- **Nous avons implémenté une chaîne de détection d'objets en edge avec YOLOv8 sur NVIDIA Jetson Nano.**
- **Le système respecte les contraintes de temps réel tout en conservant une bonne précision.**
- **L'Edge AI constitue une alternative solide aux systèmes de vision basés uniquement sur le cloud.**

Perspectives / travaux futurs :

- **Ajouter le suivi d'objets (object tracking) et la fusion multi-caméras.**
- **Explorer des modèles plus légers ou quantifiés pour augmenter encore les FPS.**
- **Intégrer la détection dans une pile complète de robot autonome.**
- **Étendre le jeu de données à davantage de classes et à des scénarios plus complexes.**