RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ ABDELHAMID IBN BADIS – MOSTAGANEM

FACULTÉ DES SCIENCES ET DE LA TECHNOLOGIE

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

MÉMOIRE DE FIN D'ÉTUDES

Présenté en vue de l'obtention du diplôme de

MASTER

Filière : **Génie électrique**

Spécialité : **Électronique des systèmes embarqués**

THÈME

Conception et développement d'un système d'aide à la conduite automobile basé sur l'intelligence artificielle embarquée pour la détection et la reconnaissance des panneaux routiers algériens

Projet de startup soutenu dans le cadre de l'arrêté ministériel n° 1275

Réalisé par :

BRAKNI Mohamed Abdessamad

BEKHERASSANE Mohamed

Encadrement :

Encadrant : Pr. MERAH Mostefa – Université Abdelhamid Ibn Badis de Mostaganem

Co-encadrant : Dr. CHOUAI Mohamed – Université Technische d'Ingolstadt, Allemagne

Soutenu le 08 octobre 2026 devant le jury composé de :

Composé de	Grade	Fonction
BENAOUALI Mohamed	MCB	Président
ABDERRAHMANE Abdelkader	MCB	Examinateur
BENZIDANE Mohammed Ridha	MCB	Rapporteur
M.	–	Représentant de l'Incubateur de l'Université de Mostaganem
M.	–	Représentant de la Direction des Transports de Mostaganem

---

Année universitaire : 2025 – 2026

الملخص

تُعدّ السلامة المرورية تحديًا رئيسيًا، لا سيما بسبب الأخطاء البشرية. يقترح هذا المشروع نظامًا ذكيًا واقتصاديًا للكشف عن إشارات المرور وتصنيفها في الزمن الحقيقي، ومتكيفًا مع السياق الجزائري. تعتمد البنية على سلسلة معالجة متتابعة تجمع بين **YOLOv8n** للكشف و**MobileNetV2** للتصنيف. تم تدريب النموذجين باستخدام مجموعتي البيانات **GTSDB** و**GTSRB**، ثم تكييفهما مع السياق الجزائري باستخدام مجموعة بيانات محلية تم جمعها ووسمها باستخدام **Roboflow**. تم نشر النظام على منصة **Raspberry Pi 5** مجهزة بكاميرا **CSI**.

حققت النتائج قيمة **mAP@50** بلغت **97,1%** في الكشف، ودقة بلغت **95,24%** مع متوسط **F1-Score** قدره **95,01%** في التصنيف. وفي ظروف التشغيل الفعلية، يحقق النظام معدل معالجة يتراوح بين **22,5** و**26,0 إطارًا في الثانية**، مع **زمن تأخير إجمالي قدره 41,5 ملي ثانية لكل صورة**. وتُظهر هذه النتائج إمكانية تنفيذ حل مدمج للرؤية الحاسوبية يعمل في الزمن الحقيقي ضمن قيود الموارد الحاسوبية.

الكلمات المفتاحية : الذكاء الاصطناعي - الرؤية الحاسوبية - YOLOv8n - MobileNetV2 - التعلم العميق - الأنظمة المدمجة - السلامة المرورية.

Résumé

La sécurité routière constitue un enjeu majeur, notamment en raison des erreurs humaines. Ce projet propose un système intelligent et économique de détection et de classification des panneaux de signalisation en temps réel, adapté au contexte algérien. L'architecture repose sur un pipeline séquentiel combinant **YOLOv8n** pour la détection et **MobileNetV2** pour la classification. Les modèles sont initialement entraînés sur les datasets **GTSDB** et **GTSRB**, puis adaptés au contexte algérien à l'aide d'un dataset local collecté et annoté avec **Roboflow**. Le système est déployé sur une **Raspberry Pi 5** équipée d'une caméra **CSI**.

Les résultats obtenus atteignent un **mAP@50 de 97,1%** pour la détection et une **accuracy de 95,24%** avec un **F1-score moyen de 95,01%** pour la classification. En conditions réelles, le système atteint **22,5 à 26,0 FPS**, avec une **latence totale de 41,5 ms par image**. Ces résultats démontrent la faisabilité d'une solution embarquée de vision artificielle fonctionnant en temps réel sous contraintes de ressources.

Mots-clés : Intelligence artificielle – Vision par ordinateur – YOLOv8n – MobileNetV2 – Apprentissage profond – Systèmes embarqués – Sécurité routière.

Abstract

Road safety is a major challenge, particularly due to human error. This project proposes an intelligent and cost-effective system for real-time traffic sign detection and classification, adapted to the Algerian context. The architecture relies on a sequential pipeline combining **YOLOv8n** for detection and **MobileNetV2** for classification. The models are initially trained on the **GTSDB** and **GTSRB** datasets and then adapted to the Algerian context using a locally collected and **Roboflow**-annotated dataset. The system is deployed on a **Raspberry Pi 5** equipped with a **CSI** camera.

The obtained results reach a **mAP@50 of 97,1%** for detection and an **accuracy of 95,24%** with a **mean F1-score of 95,01%** for classification. Under real-world conditions, the system achieves **22,5 to 26,0 FPS**, with a **total latency of 41.5 ms per image**. These results demonstrate the feasibility of a real-time embedded computer vision solution under resource constraints.

Keywords : Artificial Intelligence – Computer Vision – YOLOv8n – MobileNetV2 – Deep Learning – Embedded Systems – Road Safety.

Remerciements

La réalisation de ce projet de fin d'études représente une étape importante de notre parcours universitaire. Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce travail.

Nous adressons nos plus sincères remerciements à notre encadrant pour son accompagnement, sa disponibilité, ses précieux conseils et la confiance qu'il nous a accordée tout au long de ce projet. Son expertise, sa rigueur scientifique et son soutien ont été essentiels à la réalisation de cette étude.

Nous exprimons également notre profonde reconnaissance à notre co-encadrant pour son implication, ses remarques constructives, sa disponibilité et ses orientations, qui ont contribué à améliorer la qualité de ce travail.

Nous remercions chaleureusement les membres du jury d'avoir accepté d'évaluer ce mémoire et de lui consacrer leur temps. Leurs remarques et leurs recommandations seront très appréciées.

Nous adressons également nos remerciements à l'ensemble des enseignants du département de Génie Électrique de l'université Abdelhamid Ibn Badis de Mostaganem pour la qualité de leur enseignement et les connaissances qu'ils nous ont transmises tout au long de notre formation.

Enfin, nous exprimons notre profonde gratitude à nos familles et à nos proches pour leur soutien, leur patience et leurs encouragements tout au long de notre parcours universitaire.

À toutes les personnes qui ont contribué, directement ou indirectement, à la réalisation de ce projet, nous adressons nos plus sincères remerciements.

BRAKNI Mohamed Abdessamad et BEKHERASSANE Mohamed.

Table des matières

Liste des tableaux **7**

Table des figures **8**

Liste des acronymes **10**

Introduction générale **13**

Chapitre 1: Généralités et état de l'art **14**
1.1 Introduction . . . 14
1.2 Contexte et problématique . . . 15
1.2.1 IA et sécurité routière . . . 15
1.2.2 Cas de l'Algérie . . . 16
1.2.3 Limites actuelles . . . 16
1.2.4 Objectif du projet . . . 17
1.3 État de l'art . . . 17
1.3.1 Approches classiques . . . 17
1.3.2 Approches basées sur l'apprentissage profond . . . 18
1.3.3 Synthèse comparative . . . 19
1.3.4 Conclusion . . . 20

Chapitre 2: Fondements des architectures IA et de l'environnement **21**
2.1 Introduction . . . 21
2.2 Environnement de développement logiciel (Software) . . . 21
2.2.1 Langage Python . . . 22
2.2.2 Bibliothèques utilisées . . . 23
2.3 Environnement matériel (hardware) . . . 24
2.3.1 Raspberry pi 5 8 Go . . . 24
2.4 Concepts de base détection/classification . . . 28
2.4.1 Détection d'objets . . . 28
2.4.2 Classification d'images . . . 28
2.4.3 Complémentarité détection et classification . . . 29
2.5 Les réseaux de neurones convolutifs . . . 29
2.5.1 Architectures de réseaux de neurones profonds . . . 30
2.5.2 Les couches de réseaux de neurones convolutionnels . . . 30
2.6 YOLO . . . 31
2.6.1 YOLO-V8 . . . 32
2.6.2 Architecture YOLO . . . 32
2.6.3 Principe de fonctionnement . . . 32
2.6.4 YOLOv8n pour la détection embarquée . . . 33
2.7 MobileNet . . . 34
2.7.1 Principe de fonctionnement . . . 34
2.7.2 Mécanisme des convolutions séparables en profondeur . . . 35
2.7.3 Adaptation à notre projet . . . 36

2.8 Conclusion 36

Chapitre 3: Conception et architecture du système proposé 37
3.1 Introduction 37
3.2 Analyse fonctionnelle du système 37
3.2.1 Besoin fonctionnel 38
3.2.2 Contraintes du système 38
3.3 Architecture globale du système 38
3.3.1 Vue d'ensemble 38
3.4 Description du pipeline de traitement 40
3.5 Description des modules 40
3.6 Datasets utilisés 43
3.6.1 German Traffic Sign Detection Benchmark 43
3.6.2 German Traffic Sign Recognition Benchmark 44
3.6.2.1 Synthèse des jeux de données 45
3.7 Conclusion 46

Chapitre 4: Développement et validation préliminaire du système 47
4.1 Introduction 47
4.2 Développement et validation du module de détection 47
4.2.1 Préparation de l'environnement de développement 47
4.2.2 Préparation du jeu de données GTSDB 49
4.2.3 Configuration du modèle et lancement de l'entraînement 50
4.2.4 Validation du modèle de détection 51
4.2.5 Notions importantes 53
4.2.5.1 Distinction entre *results* et *smooth* 53
4.2.5.2 Les métriques d'évaluation 53
4.3 Développement et validation du module de classification 54
4.3.1 Préparation du jeu de données GTSRB 54
4.3.2 Configuration et entraînement du modèle MobileNetV2 55
4.3.3 Validation du modèle de classification 56
4.4 Conclusion 57

Chapitre 5: Adaptation du système au contexte algérien et évaluation des performances 58
5.1 Introduction 58
5.2 Adaptation du système au contexte algérien 59
5.2.1 Présentation du jeu de données algérien 59
5.2.2 Adaptation du détecteur YOLOv8n 60
5.2.3 Fine-tuning du détecteur YOLOv8n 61
5.2.4 Résultats du fine-tuning du détecteur YOLOv8n 62
5.2.5 Résultats finaux du test de détection 63
5.2.5.1 Matrice de confusion 64
5.2.5.2 Matrice de confusion normalisée 65
5.2.5.3 Courbe Recall–Confidence 66
5.2.5.4 Courbe Precision–Recall 67
5.2.5.5 Courbe Precision–Confidence 68
5.2.5.6 Courbe F1–Confidence 69
5.2.5.7 Synthèse des résultats du test de détection 69
5.2.5.8 Discussion des résultats 70
5.2.6 Fine-tuning du classificateur MobileNetV2 71
5.2.7 Résultats du fine-tuning de MobileNetV2 72

5.2.8 Résultats finaux du test de classification . . . 74
5.2.8.1 Résultats globaux . . . 75
5.2.8.2 Matrice de confusion . . . 75
5.2.8.3 Matrice de confusion normalisée . . . 76
5.2.8.4 Analyse des performances par classe . . . 77
5.2.8.5 Résultats et discussion . . . 78
5.3 Évaluation du système complet et discussion des résultats . . . 78
5.3.1 Intégration des modèles . . . 78
5.3.2 Interprétation des résultats d'entraînement . . . 79
5.3.3 Résultats expérimentaux . . . 80
5.3.4 Discussion des résultats expérimentaux . . . 82
5.3.4.1 Panneau STOP . . . 82
5.3.4.2 Panneau « Dos d'âne » . . . 82
5.3.4.3 Panneau « Limitation de vitesse 30 » . . . 83
5.3.4.4 Panneau « Interdiction de tourner a gauche » . . . 83
5.3.5 Évaluation des performances temporelles . . . 84
5.4 Conclusion . . . 85

Conclusion générale **86**

Bibliographie **87**

Annexes **89**
A.1 Jeux de données utilisés . . . 89
A.1.1 GTSDB : German Traffic Sign Detection Benchmark . . . 89
A.1.2 GTSRB : German Traffic Sign Recognition Benchmark . . . 90
A.1.3 Dataset de panneaux algériens . . . 90
A.1.4 Rôle des différents jeux de données . . . 91
A.2 Environnement logiciel . . . 92
A.2.1 Création de l'environnement virtuel . . . 92
A.2.2 Installation des bibliothèques . . . 92
A.3 Configuration et entraînement de YOLOv8n . . . 92
A.3.1 Modèle initial . . . 93
A.3.2 Configuration du dataset . . . 93
A.3.3 Entraînement . . . 93
A.3.4 Sauvegarde du modèle . . . 93
A.4 Configuration et entraînement de MobileNetV2 . . . 94
A.4.1 Principe du transfert d'apprentissage . . . 94
A.4.2 Prétraitement des images . . . 94
A.4.3 Sauvegarde du modèle . . . 95
A.5 Tests et évaluation des modèles . . . 95
A.5.1 Validation . . . 95
A.5.2 Test du détecteur YOLOv8n . . . 95
A.5.3 Test de MobileNetV2 . . . 96
A.5.4 Pourquoi distinguer validation et test ? . . . 96
A.6 Intégration de YOLOv8n et MobileNetV2 . . . 96
A.6.1 Chargement des modèles . . . 97
A.6.2 Architecture du pipeline . . . 97
A.6.3 Extraction de la région d'intérêt . . . 98
A.6.4 Prétraitement du crop . . . 98
A.6.5 Classification . . . 98
A.6.6 Affichage . . . 98

A.7 Adaptation au contexte algérien . . . 99
A.7.1 Collecte et annotation . . . 99
A.7.2 Fine-tuning de YOLOv8n . . . 99
A.7.3 Préparation des crops algériens . . . 99
A.7.4 Fine-tuning de MobileNetV2 . . . 100
A.8 Acquisition vidéo avec OpenCV . . . 100
A.8.1 Importation d'OpenCV . . . 101
A.8.2 Ouverture de la caméra . . . 101
A.8.3 Acquisition d'une image . . . 101
A.8.4 Affichage . . . 101
A.8.5 Arrêt . . . 102
A.9 Déploiement sur Raspberry Pi 5 . . . 102
A.9.1 Composants . . . 102
A.9.2 Installation des dépendances . . . 102
A.9.3 Lancement du système . . . 102
A.9.4 Test du système complet . . . 103
A.10 Évaluation finale . . . 104
A.10.1 Évaluation de YOLOv8n . . . 104
A.10.2 Évaluation de MobileNetV2 . . . 104
A.10.3 Évaluation du système complet . . . 104
A.11 Fichiers principaux du projet . . . 105
A.12 Synthèse du fonctionnement . . . 105
A.13 Business Model Canvas (modèle d'affaire) . . . 106
A.14 Business Plan . . . 107

Liste des tableaux

1.1 Synthèse comparative des approches de détection de panneaux de signalisation . 20
1.2 Synthèse comparative des approches de classification de panneaux de signalisation 20

2.1 Problèmes de dépendances et intérêt de l'environnement virtuel . . . 23
2.2 Comparaison entre Raspberry Pi 4 8 Go et Raspberry Pi 5 8 Go . . . 26

3.1 Répartition des catégories dans le dataset GTSDB . . . 44
3.2 Exemples de distribution des classes dans le jeu de données GTSRB . . . 45
3.3 Comparaison des jeux de données GTSDB et GTSRB . . . 46

4.1 Environnement logiciel utilisé pour le développement . . . 48

5.1 Répartition du jeu de données utilisé pour le fine-tuning du détecteur YOLOv8n 61
5.2 Performances du détecteur YOLOv8n après fine-tuning . . . 62
5.3 Résultats finaux du test de détection du modèle **YOLOv8n** . . . 70
5.4 Répartition du jeu de données algérien pour la classification . . . 72
5.5 Résultats finaux du test de classification du modèle **MobileNetV2** . . . 75
5.6 Performances temporelles et matérielles du système embarqué . . . 85

Table des figures

1.1 Fonctionnement d'un système de reconnaissance de panneaux 15
1.2 Exemples de panneaux de signalisation dégradés sur les routes algériennes 16
1.3 Schéma fonctionnel du système de reconnaissance proposé 17

2.1 Logo Roboflow 22
2.2 Logo python 22
2.3 Logo Ultralytics 23
2.4 Logo OpenCV 23
2.5 Logo PyTorch. 24
2.6 Raspberry PI 5 8Go. 25
2.7 Refroidisseur de Raspberry Pi. 27
2.8 Caméra Raspberry Pi Module 2. 28
2.9 Distinction entre la détection et la classification dans le pipeline du système. . . 29
2.10 Les réseaux de neurones convolutifs. 30
2.11 Les couches de CNN. 31
2.12 YOLO-V8 comparaison avec les prédécesseurs. 32
2.13 Illustration graphique du fonctionnement du modèle YOLO 33
2.14 Comparaison entre une convolution classique et la convolution séparable en profondeur 35
2.15 Convolution séparable en profondeur. 36

3.1 Chaîne de traitement du système. 39
3.2 Flux de traitement d'une image. 40
3.3 Configuration matérielle pour l'acquisition des images 41
3.4 Visualisation des boîtes englobantes et scores de confiance générés par YOLO . . 42
3.5 Affichage en temps réel des résultats de détection et de classification des panneaux de signalisation. 43
3.6 Exemples d'images extraites du dataset GTSRB. 45

4.1 Environnement de développement du projet sous Visual Studio Code. 49
4.2 Organisation du jeu de données GTSDB. 50
4.3 Lancement de l'entraînement du modèle YOLOv8n. 51
4.4 Exemple de détection des panneaux de signalisation réalisée par le modèle YOLOv8n après entraînement. 52
4.5 Évolution des principales métriques d'entraînement du modèle YOLOv8n. 52
4.6 Organisation du jeu de données GTSRB utilisée pour l'entraînement du modèle MobileNetV2. 54
4.7 Arborescence du jeu de données GTSRB utilisée pour l'entraînement du modèle de classification. 55
4.8 Lancement de l'entraînement du modèle MobileNetV2 sur le jeu de données GTSRB. 56

5.1 Exemples d'images annotées du jeu de données sur Roboflow. 60
5.2 Évolution des performances du modèle YOLOv8n au cours du fine-tuning. . . . 63

5.3 Matrice de confusion du modèle **YOLOv8n** obtenue sur l'ensemble de test . . . 64
5.4 Matrice de confusion normalisée du modèle **YOLOv8n** obtenue sur l'ensemble de test . . . 65
5.5 Courbe Recall–Confidence du modèle **YOLOv8n** obtenue sur l'ensemble de test 66
5.6 Courbe Precision–Recall du modèle **YOLOv8n** obtenue sur l'ensemble de test . 67
5.7 Courbe Precision–Confidence du modèle **YOLOv8n** obtenue sur l'ensemble de test . . . 68
5.8 Courbe F1–Confidence du modèle **YOLOv8n** obtenue sur l'ensemble de test . . 69
5.9 Exemples de détection de panneaux de signalisation obtenus après le fine-tuning. 71
5.10 Résultats de classification du modèle **MobileNetV2** obtenus sur l'ensemble de test. . . . 72
5.11 Résultats du modèle MobileNetV2 fine-tuné sur des scènes routières algériennes 73
5.12 Résultats de classification des panneaux après fine-tuning de MobileNetV2 . . . 73
5.13 Matrice de Confusion (MobileNetV2) . . . 74
5.14 Évolution des métriques lors du test final de classification (MobileNetV2) 75
5.15 Matrice de confusion du modèle **MobileNetV2** obtenue sur l'ensemble de test. 76
5.16 Matrice de confusion normalisée du modèle **MobileNetV2** obtenue sur l'ensemble de test. . . . 77
5.17 Diagramme de fonctionnement du système complet . . . 79
5.18 Résultat de la détection et de la reconnaissance de plusieurs panneaux de signalisation. . . . 81
5.19 Exemple de reconnaissance d'un panneau de limitation de vitesse à 60 km/h. . . 81
5.20 Reconnaissance du panneau STOP . . . 82
5.21 Reconnaissance du panneau de danger (Dos d'âne) . . . 82
5.22 Reconnaissance du panneau de limitation de vitesse 30 . . . 83
5.23 Reconnaissance du panneau d'interdiction de tourner a gauche. . . . 84
5.24 Business Model Canvas . . . 106
5.25 Invesstisement . . . 107
5.26 Masse salariale . . . 107
5.27 Chiffre d'affaires . . . 107
5.28 Charges externes . . . 108
5.29 Besoin en Fonds de Roulement . . . 108
5.30 compte de résultat . . . 108
5.31 Flux de Trésorerie . . . 109

Liste des acronymes

- **NMS** : Non-Maximum Suppression
- **ROI** : Region of Interest
- **CNN** : Convolutional Neural Network
- **CPU** : Central Processing Unit
- **GTSRB** : German Traffic Sign Recognition Benchmark
- **GTSDB** : German Traffic Sign Detection Benchmark
- **DL** : Deep Learning
- **HOG** : Histogramme de Gradients Orientés
- **IA** : Intelligence Artificielle
- **R-CNN** : Region Based Convolutional Neural Networks
- **KNN** : K-Nearest Neighbors
- **ReLU** : Rectified Linear Unit
- **RGB** : Red Green Blue
- **HSV** : Hue, Saturation, Value
- **HSL** : Hue, Saturation, Lightness
- **SIFT** : Scale-Invariant Feature Transform
- **SVM** : Support Vector Machine
- **ADAS** : Advanced Driver Assistance Systems
- **YOLO** : You Only Look Once
- **mAP** : Mean Average Precision
- **OpenCV** : Open Source Computer Vision Library
- **MobileNetV2** : Mobile Network Version 2
- **FPS** : Frames Per Second

Introduction générale

Selon l'Organisation mondiale de la santé (OMS), les accidents de la route causent environ 1,19 million de décès chaque année à travers le monde [1], érigeant la sécurité routière au rang de défi majeur de santé publique. Face à l'augmentation du trafic et à la complexité des environnements urbains, l'Intelligence Artificielle s'impose désormais comme le levier le plus prometteur pour transformer le secteur des transports. En particulier, les systèmes d'aide à la conduite (ADAS) deviennent une solution indispensable pour pallier les erreurs humaines, qui demeurent la cause principale de la majorité des tragédies routières.

En Algérie, cette problématique est particulièrement criante. Malgré les efforts de sensibilisation, les statistiques d'accidents restent préoccupantes. L'introduction de technologies intelligentes se heurte toutefois à des obstacles spécifiques : le coût élevé des équipements sophistiqués, l'absence de bases de données routières locales exhaustives et des conditions climatiques variées qui mettent à l'épreuve la fiabilité des capteurs traditionnels.

C'est dans ce cadre que s'inscrit ce projet de fin d'études, dont l'objectif est de concevoir un système intelligent de reconnaissance des panneaux de signalisation routière basé sur les technologies d'intelligence artificielle et les systèmes embarqués. Le système repose sur un pipeline séquentiel combinant YOLO pour la détection des panneaux et MobileNetV2 pour leur classification, déployé sur une carte Raspberry Pi 5. Une contribution locale est également envisagée, visant à adapter le système aux spécificités du réseau routier algérien par la collecte de données réelles et le fine-tuning des modèles.

Ce mémoire est structuré en cinq chapitres de la manière suivante :

Chapitre I : : présente le contexte général, la problématique et un état de l'art comparatif des approches existantes en matière de détection et de classification de panneaux de signalisation.

Chapitre II : expose les fondements théoriques du projet, notamment les architectures YOLO et MobileNetV2, ainsi que l'environnement de développement utilisé.

Chapitre III : décrit l'architecture globale du système proposé, le pipeline séquentiel intégré et les datasets utilisés.

Chapitre IV : présente le développement et la validation préliminaire du système proposé. Il décrit la configuration de l'environnement de développement, la préparation des jeux de données ainsi que le développement et la validation des deux modules principaux : **YOLOv8n** pour la détection et **MobileNetV2** pour la classification. Cette étape permet de vérifier le fonctionnement du pipeline avant l'entraînement complet sur les jeux de données **GTSDB** et **GTSRB**.

Chapitre V : détaille l'entraînement et l'adaptation des modèles YOLOv8n et MobileNetV2 à partir des datasets GTSDB et GTSRB, ainsi que l'adaptation au contexte algérien par fine-tuning local. Il présente également l'intégration des modèles, et l'évaluation de ses performances.

Chapitre 1

Généralités et état de l'art

1.1 Introduction

De nos jours, le domaine de l'intelligence artificielle a connu une croissance fulgurante grâce à la collaboration entre la recherche académique et l'industrie, ainsi qu'à l'essor du monde open source. Cette révolution a permis de démocratiser des algorithmes complexes, autrefois réservés aux supercalculateurs, pour les rendre accessibles sur des plateformes de développement variées. On distingue aujourd'hui des cartes basiques comme les microcontrôleurs (Arduino, STM32) et des plateformes beaucoup plus puissantes capables de supporter du Deep Learning, telles que la Jetson Nano ou la Raspberry Pi.

En outre, ce développement matériel joue un rôle crucial dans l'expansion des systèmes [2]. L'intégration de la vision par ordinateur dans les véhicules n'est plus un luxe industriel, mais devient une base essentielle pour renforcer la sécurité routière. Cette montée en puissance du marché des systèmes embarqués encourage désormais la création de solutions sur mesure pour répondre à des problématiques locales spécifiques, comme la reconnaissance des panneaux de signalisation en conditions réelles.

En effet, l'architecture d'un système de reconnaissance d'images repose généralement sur trois piliers :

- **L'acquisition de données** : collecte et préparation des images.
- **Le traitement intelligent** : analyse via des modèles d'intelligence artificielle.
- **L'exécution matérielle** : déploiement sur des systèmes embarqués ou serveurs.

En addition, les exigences de sécurité routière imposent des propriétés de rapidité de traitement, de précision de détection et de robustesse. Ces contraintes nous ont guidés à choisir la carte Raspberry Pi comme coeur de notre système, étant donné qu'elle répond parfaitement au cahier des charges demandé pour l'implémentation de modèles d'IA légers.

Dans ce chapitre, nous allons d'abord poser le cadre conceptuel de la sécurité routière et les défis liés à son déploiement en Algérie. Nous présenterons ensuite les fondements théoriques des réseaux de neurones convolutionnels (CNN) et du deep learning (DL) et le rôle de la carte Raspberry Pi 5 dans notre architecture. Enfin, nous détaillerons l'étude du dataset, qui constitue la base d'entraînement de notre systeme embarquée intelligent.

1.2 Contexte et problématique

La reconnaissance automatique des panneaux de signalisation constitue aujourd'hui un défi majeur à l'intersection de l'intelligence artificielle et des systèmes embarqués. Si les architectures DL offrent des performances visuelles remarquables, elles se heurtent aux contraintes strictes de ces dispositifs en matière de mémoire, de puissance de calcul et de temps réel.

L'enjeu de ce projet est précisément de trouver le bon équilibre : exploiter la puissance des modèles d'apprentissage profond tout en restant deployable sur du matériel embarqué. À cela s'ajoute une dimension souvent négligée, mais déterminante l'adaptation aux spécificités locales des routes algériennes, qu'il s'agisse de l'état des panneaux, des conditions d'éclairage ou des standards de signalisation. C'est dans cette triple exigence de précision, d'efficacité et de pertinence contextuelle que s'inscrit pleinement ce travail.

1.2.1 IA et sécurité routière

Ces dernières années, l'intelligence artificielle a commencé à jouer un rôle concret dans l'amélioration de la sécurité routière. Grâce aux progrès de la vision par ordinateur, il est désormais possible de concevoir des systèmes capables de détecter et d'interpréter les éléments de la route, notamment les panneaux de signalisation. Cette tâche, qui peut paraître simple pour un conducteur expérimenté, reste techniquement complexe à automatiser, car elle implique de gérer des variations d'éclairage, d'angle de vue ou encore d'état de dégradation des panneaux. C'est précisément dans ce cadre que les approches basées sur l'apprentissage profond ont montré leur pertinence, en offrant une capacité de généralisation bien supérieure aux méthodes traditionnelles de traitement d'image.

La figure 1.1 illustre le principe de fonctionnement d'un tel système, où le panneau détecté est directement affiché sur l'écran de bord du véhicule.

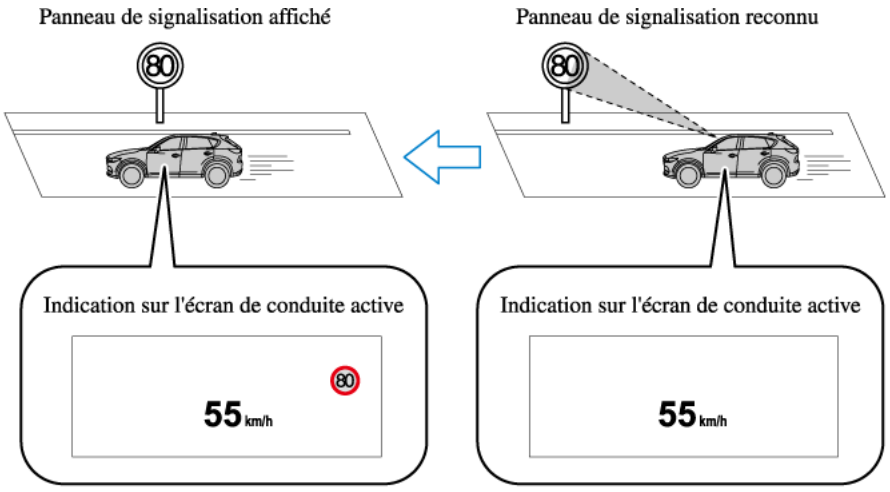


FIGURE 1.1 – Fonctionnement d'un système de reconnaissance de panneaux

1.2.2 Cas de l'Algérie

L'Algérie n'échappe pas à la problématique mondiale de l'insécurité routière. Bien au contraire, la situation y reste particulièrement préoccupante malgré les efforts déployés par les autorités compétentes. Selon le bilan 2025 de la Gendarmerie Nationale, qui assure la sécurisation de près de 85% du réseau routier national, pas moins de 7 735 accidents ont été recensés au cours de cette seule année, faisant 2 996 morts et 13 504 blessés [3].

Au-delà des chiffres, cette réalité s'explique en partie par des facteurs terrain difficiles à ignorer. Les panneaux de signalisation algériens souffrent fréquemment d'un manque d'entretien : décoloration, rouille, vandalisme ou encore occultation partielle par la végétation sont des situations courantes sur nos routes. À cela s'ajoutent des conditions climatiques variées, allant de la chaleur intense du sud aux brouillards du nord, qui mettent à rude épreuve la fiabilité de tout système de détection automatique.

C'est dans ce contexte que l'intérêt d'un système de reconnaissance de panneaux adapté aux spécificités locales prend tout son sens. Une solution générique, entraînée sur des données européennes ou américaines, ne saurait répondre efficacement aux réalités du terrain algérien. D'où la nécessité de concevoir un système pensé dès le départ pour ce contexte particulier.



FIGURE 1.2 – Exemples de panneaux de signalisation dégradés sur les routes algériennes

Source : reportage d'Ennahar TV.

1.2.3 Limites actuelles

Les systèmes de reconnaissance de panneaux de signalisation existants ont certes atteint des niveaux de performance remarquables sur des benchmarks publics.

Cependant, ces résultats sont obtenus dans des conditions contrôlées qui ne reflètent pas toujours la complexité du terrain réel, creusant ainsi un fossé notable entre les performances en laboratoire et le comportement effectif du système en situation réelle.

En outre, les bases de données disponibles ont été constituées dans un contexte purement européen ou américain, ce qui limite considérablement leur capacité de généralisation à d'autres environnements routiers. L'absence d'un dataset local adapté au contexte algérien constitue de ce fait un obstacle concret au développement de solutions véritablement efficaces sur le terrain [2].

Les solutions commerciales intégrant ces technologies demeurent largement inaccessibles, étant réservées aux véhicules haut de gamme à des coûts prohibitifs pour une adoption à grande échelle. En revanche, le recours à des plateformes embarquées open source et à faible coût, comme la Raspberry Pi, ouvre la voie à des alternatives viables et accessibles. C'est précisément

cette démarche que vise notre projet, en cherchant à concevoir un système performant tout en restant économiquement et techniquement déployable.

1.2.4 Objectif du projet

L'objectif principal de ce projet est de concevoir un système de reconnaissance de panneaux de signalisation autonome et économique, basé sur la vision par ordinateur. Le système repose sur un pipeline séquentiel en deux étapes complémentaires : une étape de détection et une étape de classification.

La détection est assurée par le modèle YOLOv8n, entraîné sur le dataset GTSDB, dont le rôle est de localiser et d'extraire les panneaux présents dans l'image capturée par la caméra. La région d'intérêt extraite est ensuite transmise à un modèle de classification basé sur l'architecture MobileNetV2, entraîné sur le dataset GTSRB, qui se charge d'identifier le type de panneau détecté.

L'ensemble du système est déployé sur une carte Raspberry Pi 5, plateforme embarquée à faible coût, via une optimisation du modèle afin de garantir un traitement en temps réel malgré les ressources matérielles limitées.

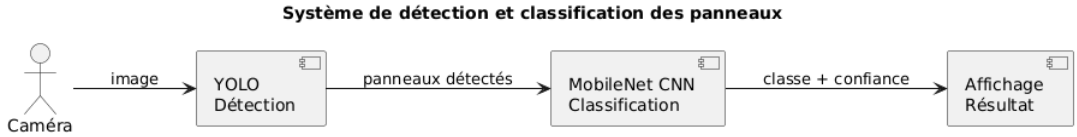


FIGURE 1.3 – Schéma fonctionnel du système de reconnaissance proposé

1.3 État de l'art

1.3.1 Approches classiques

Avant l'essor du Deep Learning, la reconnaissance des panneaux de signalisation reposait sur des méthodes de traitement d'images traditionnelles, organisées en deux étapes distinctes : la détection et la classification. La détection consistait à localiser les panneaux dans l'image à partir de leurs propriétés visuelles apparentes, telles que la couleur et la forme, tandis que la classification s'appuyait sur des descripteurs manuellement conçus pour identifier le type de panneau détecté [2].

Les approches basées sur la couleur exploitaient la nature standardisée des panneaux de signalisation, dont les couleurs dominantes (rouge, bleu, jaune) facilitent leur segmentation dans l'espace colorimétrique HSV ou HSI. Cependant, ces méthodes se révèlent peu robustes face aux variations d'éclairage et à la décoloration des panneaux due au vieillissement [2].

Les approches basées sur la forme, quant à elles, reposaient sur des descripteurs tels que le HOG ou le SIFT, combinés à des classificateurs classiques comme les SVM. Si ces méthodes ont produit des résultats acceptables dans des conditions contrôlées, elles présentent des limitations importantes face à la variabilité du monde réel, notamment les occlusions, le flou de mouvement et les variations de point de vue [2].

1.3.2 Approches basées sur l'apprentissage profond

L'introduction des architectures de Deep Learning a marqué un tournant décisif dans le domaine de la reconnaissance de panneaux de signalisation. Contrairement aux approches classiques qui nécessitent une extraction manuelle des caractéristiques, les réseaux de neurones profonds apprennent automatiquement des représentations hiérarchiques directement depuis les données brutes, offrant une capacité de généralisation bien supérieure [4].

Le concours GTSRB de 2011 a constitué un point de référence majeur pour évaluer ces nouvelles approches. Ciresan et al. [5] ont proposé un comité de réseaux de neurones convolutifs atteignant une précision de 98,98%, surpassant la performance humaine estimée à 98,81%. Sermanet et LeCun [6] ont quant à eux introduit une architecture CNN multi-échelles obtenant 99,17% de précision en utilisant des images en niveaux de gris.

Plus récemment, l'architecture YOLO (You Only Look Once) a révolutionné la détection d'objets en temps réel en traitant l'image en une seule passe, contrairement aux approches à deux étapes comme R-CNN. Cette propriété la rend particulièrement adaptée aux applications embarquées où la contrainte de temps réel est primordiale. Parallèlement, MobileNetV2 s'est imposé comme une architecture de classification légère et efficace, spécialement conçue pour les plateformes à ressources limitées, grâce à l'utilisation de convolutions séparables en profondeur qui réduisent considérablement le nombre de paramètres sans sacrifier la précision [7].

Les dernières avancées scientifiques dans le domaine de la détection et de la reconnaissance des panneaux de signalisation routière (TSDR) s'orientent vers des architectures plus performantes, plus rapides et mieux adaptées aux contraintes des systèmes embarqués. Parmi les principales orientations récentes, on peut citer :

1. **Vision Transformers et RT-DETR (Transformateurs visuels en temps réel)**

   Concept : Remplacement ou complément des réseaux convolutifs traditionnels par des architectures basées sur les transformateurs visuels, exploitant des mécanismes d'attention pour analyser les relations entre les différentes régions de l'image. Certaines architectures récentes, telles que RT-DETR, permettent également de réduire la dépendance aux étapes classiques de post-traitement, notamment la suppression des détections redondantes (*Non-Maximum Suppression – NMS*) [8].

   Avantage : Meilleure capacité à capturer les petits panneaux éloignés ainsi qu'à exploiter le contexte global de la scène routière, ce qui peut contribuer à améliorer la précision de la détection et à réduire les fausses alertes.
2. **Spiking Neural Networks – SNNs (Réseaux de neurones impulsionnels)**

   Concept : Architecture d'apprentissage inspirée du fonctionnement des neurones biologiques du cerveau humain, dans laquelle l'information est représentée et transmise sous forme d'impulsions. Ces réseaux peuvent être exécutés sur des architectures de calcul neuromorphique spécialement conçues pour ce type de traitement [9].

   Avantage : Réduction importante de la consommation énergétique par rapport à certaines architectures de calcul classiques, ce qui rend cette technologie particulièrement intéressante pour les véhicules électriques et les systèmes embarqués soumis à des contraintes énergétiques.
3. **Open-Vocabulary Detection et VLMs (Modèles Vision-Langage)**

   Concept : Fusion des informations provenant du texte et de l'image afin de permettre la

détection et la reconnaissance d'objets au-delà des catégories définies lors de l'entraînement classique. Des modèles tels que Grounding DINO et CLIP illustrent cette nouvelle approche combinant vision artificielle et langage naturel [10].

Avantage : Possibilité de reconnaître des panneaux rares, nouveaux ou non définis préalablement dans les catégories d'apprentissage. Cette approche ouvre notamment la voie à des systèmes capables d'identifier de nouvelles catégories sans nécessiter systématiquement un réentraînement préalable, grâce aux mécanismes d'apprentissage de type *Zero-Shot Learning*.

4. **Vision Mamba et SSMs (Modèles d'espace d'état)**

   Concept : Technologie de traitement alternative aux Transformers, basée sur les modèles d'espace d'état (*State Space Models – SSMs*). Ces architectures permettent notamment de traiter efficacement de longues séquences visuelles avec une complexité de calcul pouvant être linéaire $O(N)$, contrairement à la complexité quadratique $O(N^2)$ associée à certains mécanismes d'attention [7].

   Avantage : Vitesse d'exécution élevée et réduction des besoins en mémoire pour le traitement d'images haute résolution et de flux vidéo. Cette caractéristique est particulièrement intéressante pour les systèmes embarqués disposant de ressources limitées en calcul et en mémoire vive.

5. **Multi-Spectral Fusion (Fusion thermique et visuelle)**

   Concept : Fusion des données provenant de caméras infrarouges ou thermiques avec celles issues de caméras optiques classiques (RGB) au sein d'un même système de perception. Les différentes modalités permettent ainsi de fournir des informations complémentaires sur l'environnement routier [11].

   Avantage : Amélioration de la robustesse du système dans des conditions de visibilité difficiles, notamment durant la nuit, sous un brouillard dense ou en présence d'un éblouissement solaire important. Cette approche permet ainsi d'envisager une perception plus fiable lorsque les informations provenant d'une caméra RGB seule deviennent insuffisantes.

1.3.3 Synthèse comparative

Afin de répondre aux exigences méthodologiques, nous avons séparé l'étude comparative en deux étapes distinctes : la détection (Tableau 1.1) et la classification (Tableau 1.2). Les tableaux suivants résument les principales approches de la littérature scientifique en les comparant avec les résultats obtenus par notre système.

TABLE 1.1 – Synthèse comparative des approches de détection de panneaux de signalisation

Auteurs	**Méthode**	**Dataset**	**Performance (mAP)**
Lafuente-Arroyo et al.	HOG + SVM	GTSDB	96,89% (mAP)
Zhang et al. (2023)	Light-YOLOv8	MTSD	89,60% (mAP@0.5)
Zhu et al. (2022)	Improved YOLOv5	TT100K	91,20% (mAP@0.5)
Notre approche (Détection)	**YOLOv8n**	**Algerian Dataset**	**97,1% (mAP@0.5)**

TABLE 1.2 – Synthèse comparative des approches de classification de panneaux de signalisation

Auteurs	**Méthode**	**Dataset**	**Précision (Accuracy)**
Ciresan et al. [5]	CNN multi-colonnes	GTSRB	98.98%
Sermanet & LeCun [6]	CNN multi-échelles	GTSRB	99.17%
Haque et al. [12]	CNN léger	GTSRB	98.41%
Sultana et al. (2023)	MobileNetV2 + CBAM	GTSRB	99.10%
Notre approche (Classification)	**MobileNetV2**	**Algerian Dataset (48 classes)**	**95,24%**

1.3.4 Conclusion

Ce premier chapitre a été dédié à la cartographie de l'état de l'art dans le domaine de la reconnaissance automatique des panneaux de signalisation. L'analyse critique des travaux de la littérature a mis en évidence la transition des approches morphologiques et colorimétriques traditionnelles vers les modèles de Deep Learning, ces derniers offrant une résilience nettement supérieure face aux perturbations du monde réel, telles que les variations de luminosité, les occultations ou encore les aléas météorologiques.

S'inscrivant dans ce sillage, et pour pallier l'absence de datasets standardisés dédiés au contexte algérien ainsi que les barrières financières liées aux solutions commerciales, ce projet propose une alternative souveraine et optimisée. Notre approche repose sur le couplage de l'algorithme YOLO (pour la localisation) et du réseau léger MobileNetV2 (pour la pour la classification), instanciés sur une plateforme cible Raspberry Pi 5. Cette synergie permet d'atteindre un compromis optimal entre latence (temps réel) et puissance de calcul embarquée.

Le chapitre suivant sera consacré à l'exposé des concepts théoriques sous-jacents à ces modèles, ainsi qu'à la présentation de l'écosystème logiciel supportant nos développements.

Chapitre 2

Fondements des architectures IA et de l'environnement

2.1 Introduction

Ce deuxième chapitre est consacré à l'étude des fondements théoriques et des outils technologiques nécessaires à la réalisation de notre système. L'objectif est de poser les bases scientifiques qui justifient nos choix logiciels et architecturaux face aux contraintes du temps réel.

Dans un premier temps, nous présenterons l'environnement de développement ainsi que les bibliothèques logicielles indispensables au traitement d'images. Ensuite, nous détaillerons le fonctionnement de l'algorithme de détection YOLOv8n, chargé de localiser les panneaux dans leur environnement. Enfin, nous aborderons l'architecture de classification MobileNetV2, retenue pour sa légèreté et son efficacité. Cette étude théorique permettra de valider la viabilité d'un pipeline optimisé, capable de s'exécuter de manière fluide sur notre plateforme embarquée Raspberry Pi 5.

2.2 Environnement de développement logiciel (Software)

Raspberry OS

Pour faire tourner l'ensemble de notre application, le choix s'est porté sur Raspberry Pi OS Bookworm. Cette distribution Linux est construite sur la base stable de Debian 12 (Bookworm), ce qui nous garantit un système moderne, sécurisé et parfaitement optimisé pour le matériel de la carte.

Ce système d'exploitation tire pleinement parti de l'architecture des processeurs ARM. Basés sur un jeu d'instructions réduit (RISC), ces processeurs se distinguent par un excellent rapport performance/consommation, une caractéristique indispensable pour les applications embarquées et autonomes.

Côté interface, la distribution utilise un environnement de bureau graphique très léger, LXDE (Lightweight X11 Desktop Environment), associé au gestionnaire de fenêtres Openbox. Ce choix minimaliste est un vrai atout technique pour notre projet : en limitant l'empreinte mémoire du système sur la RAM, on libère un maximum de ressources pour les calculs intensifs exigés par nos modèles de traitement d'images (YOLOv8n et MobileNetV2). Enfin, le système embarque

nativement une suite d'outils standards comme le logiciel de calcul Wolfram Mathematica, le lecteur VLC et une version optimisée du navigateur Chromium.

Roboflow

Cet outil est devenu un standard pour simplifier les étapes critiques de préparation des données en vision par ordinateur, de l'annotation initiale jusqu'à l'export final. La plateforme prend en charge de multiples formats de données et combine des outils d'étiquetage manuel et assisté (automatisation), ce qui permet de générer rapidement des datasets structurés. L'un des grands atouts de Roboflow réside dans ses fonctionnalités d'augmentation de données. En appliquant des transformations géométriques contrôlées aux images existantes (comme le recadrage, la rotation ou le redimensionnement), le système génère artificiellement de nouvelles variations. Cette technique permet non seulement d'enrichir le volume du dataset, mais aussi d'améliorer la robustesse et la capacité de généralisation de nos modèles face aux conditions réelles.



FIGURE 2.1 – Logo Roboflow

2.2.1 Langage Python

Le choix de Python comme langage de programmation principal se justifie par sa position de référence dans le domaine de l'intelligence artificielle et de la vision par ordinateur. Sa syntaxe épurée et sa lisibilité facilitent le développement d'architectures complexes. De plus, la richesse de son écosystème de bibliothèques spécialisées et le dynamisme de sa communauté scientifique constituent des atouts majeurs pour ce projet.

Afin d'assurer la stabilité du système et la reproductibilité des résultats, le développement est encapsulé au sein d'un environnement virtuel isolé via (`venv`). L'isolation des packages et des dépendances ainsi obtenue assure une parfaite reproductibilité du code et optimise le processus de déploiement multiplateforme, notamment lors du transfert du code vers le système embarqué.



FIGURE 2.2 – Logo python

Sans environnement virtuel, tous nos projets Python partagent les mêmes packages installés. Voici ce qui peut mal tourner :

TABLE 2.1 – Problèmes de dépendances et intérêt de l'environnement virtuel

Situation	**Problème**	**Conséquence**
Raspberry Pi (PEP 668)	L'OS bloque l'installation globale de packages Python	Certaines bibliothèques peuvent ne pas fonctionner de la même manière
YOLO+MobileNetV2	Bibliothèques avec versions différentes	Erreurs d'incompatibilités et nécessité d'adapter le code existant
Déploiement sur une autre machine	Environnements logiciels différents	Comportement non identique du programme
Je collabore avec mon collègue	Ses versions diffèrent des miennes	L'un des deux projets casse

2.2.2 Bibliothèques utilisées

Plusieurs bibliothèques fondamentales ont été mobilisées pour la réalisation de ce projet.

Ultralytics est la bibliothèque officielle d'implémentation de YOLOv8, offrant une interface simple et efficace pour l'entraînement, la validation et le déploiement de modèles de détection d'objets. Elle intègre nativement des outils d'exportation vers des formats optimisés tels que ONNX et TensorFlow Lite, facilitant ainsi le déploiement sur systèmes embarqués.



FIGURE 2.3 – Logo Ultralytics

OpenCV est une bibliothèque open source de vision par ordinateur, utilisée pour la capture et le traitement des images en temps réel. Elle assure l'interface entre la caméra et le pipeline de traitement, ainsi que la restitution visuelle des résultats.

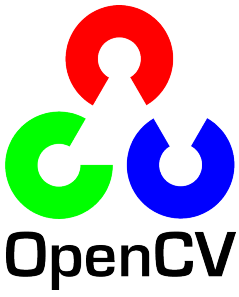


FIGURE 2.4 – Logo OpenCV

PyTorch et TorchVision constituent le framework d'apprentissage profond utilisé pour l'entraînement du modèle MobileNetV2. TorchVision fournit des modèles pré-entraînés et des uti-

litaires de prétraitement des données, facilitant la mise en œuvre du *transfer learning.* [1]



FIGURE 2.5 – Logo PyTorch.

2.3 Environnement matériel (hardware)

La phase initiale d'apprentissage et de validation des modèles a été exécutée sur un ordinateur équipé d'un processeur Intel Core i7-7700 (3, 60 GHz, 4 cœurs) et de 8 Go de RAM. Une fois les performances validées sur cette station de calcul, les modèles ont été déployés et évalués en conditions réelles sur la cible embarquée Raspberry Pi 5.

2.3.1 Raspberry pi 5 8 Go

Dernier-né de sa gamme, le Raspberry Pi 5 conserve le facteur de forme compact de son prédécesseur, le Raspberry Pi 4, tout en introduisant des ruptures technologiques majeures et un gain de puissance significatif. Cette évolution technique ouvre de nouvelles perspectives pour les applications exigeantes. La carte embarque un processeur ARM Cortex-A76 Quad-Core cadencé à 2, 4 GHz, associé à une connectivité élargie et à des performances multimédias accrues. Parfaitement adapté aux exigences actuelles du prototypage et des systèmes embarqués, ce nano-ordinateur garantit une exécution logicielle particulièrement fluide.

Caractéristiques principales

- Processeur : Broadcom BCM2712, Quad-core Cortex-A76 (ARM v8) 64 bits à 2,4 GHz.
- Mémoire RAM : 8 Go LPDDR4X-4267.
- GPU : VideoCore VII à 800 MHz (OpenGL ES 3.1, Vulkan 1.2).
- Connexion sans fil : Bluetooth 5.0 (BLE) et Wi-Fi AC bi-bande.
- Connexion filaire : Gigabit Ethernet 10/100/1000 Mbps (RJ45).
- Stockage : lecteur de carte microSD compatible avec le mode haute vitesse (SDR104).
- Interfaces MIPI : deux ports MIPI à 4 broches pouvant servir de connecteurs CSI/DSI.
- Ports : 2 ports USB 3.0, 2 ports USB 2.0, 1 port USB-C (alimentation uniquement), 1 connecteur GPIO à 40 broches et 2 ports micro-HDMI (jusqu'à 4K à 60 Hz).
- Interface PCIe : PCIe 2.0 x1 (500 Mo/s) avec connecteur FFC 16 broches (0,50 mm).
- Horloge temps réel : connecteur RTC dédié.

1. Le transfer learning est une méthode d'apprentissage profond consistant à réutiliser un modèle pré-entraîné sur une grande base de données, puis à l'adapter à une tâche spécifique en ajustant certaines couches. Cette approche permet de réduire le temps d'entraînement ainsi que la quantité de données nécessaires.

- Alimentation : 5V/5 A via USB-C (minimum 3 A), 5 V via le connecteur GPIO (minimum 3 A) et compatible Power over Ethernet (PoE) avec un HAT dédié.
- Bouton : Marche/Arrêt (ON/OFF).
- Dimensions : $86 \times 56 \times 16$ mm.
- Poids : 46 g.



FIGURE 2.6 – Raspberry PI 5 8Go.

TABLE 2.2 – Comparaison entre Raspberry Pi 4 8 Go et Raspberry Pi 5 8 Go

Caractéristique	**Raspberry Pi 4 Model B 8 Go**	**Raspberry Pi 5 8 Go**
Processeur	Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC à 1,5 GHz	Broadcom BCM2712 quad-core Cortex-A76 (ARM v8) 64-bit SoC à 2,4 GHz
RAM	8 Go LPDDR4	8 Go LPDDR4X
Stockage	Lecteur de carte micro-SD (non inclus)	Lecteur de carte micro-SD (non inclus)
Connectivité	2 × micro-HDMI, 2 × USB 3.0, 2 × USB 2.0, 1 × USB-C, Ethernet Gigabit, Wi-Fi 802.11b/g/n/ac, Bluetooth 5.0	2 × micro-HDMI, 2 × USB 3.0, 2 × USB 2.0, 1 × USB-C, Ethernet Gigabit, Wi-Fi 802.11b/g/n/ac, Bluetooth 5.0
Système d'exploitation	Raspberry Pi OS (recommandé), mais d'autres systèmes d'exploitation sont également compatibles	Raspberry Pi OS (recommandé), mais d'autres systèmes d'exploitation sont également compatibles
Prix (DA)	29 500,00	32 500,00
Points forts	Processeur puissant, plus de RAM, connectivité moderne, prix abordable	Processeur plus rapide, deux ports micro-HDMI permettant de connecter deux écrans 4K
Points faibles	Nécessite une carte micro-SD et un adaptateur secteur (non inclus)	Nécessite une carte micro-SD et un adaptateur secteur (non inclus)

Le refroidisseur actif Raspberry Pi

Pour répondre aux contraintes thermiques lors d'exécutions prolongées à vitesse maximale, le système intègre l'intellect de l'Active Cooler dédié au Raspberry Pi 5. Bien que la carte régule de manière autonome sa température pour éviter la surchauffe, l'apport de ce refroidisseur s'avère indispensable pour des applications lourdes et continues comme le machine learning.

Ce module comprend un dissipateur thermique en aluminium extrudé posé sur le processeur et un ventilateur radial conçu pour optimiser le flux d'air tout en maximisant la durée de vie du composant. Le dispositif garantit le maintien des températures bien en deçà du point d'accélération thermique, que la carte soit logée dans son boîtier officiel ou exploitée nue. Son intégration mécanique s'effectue via deux trous de montage, tandis que sa gestion logicielle et son alimentation s'appuient sur un connecteur JST standard à quatre broches.

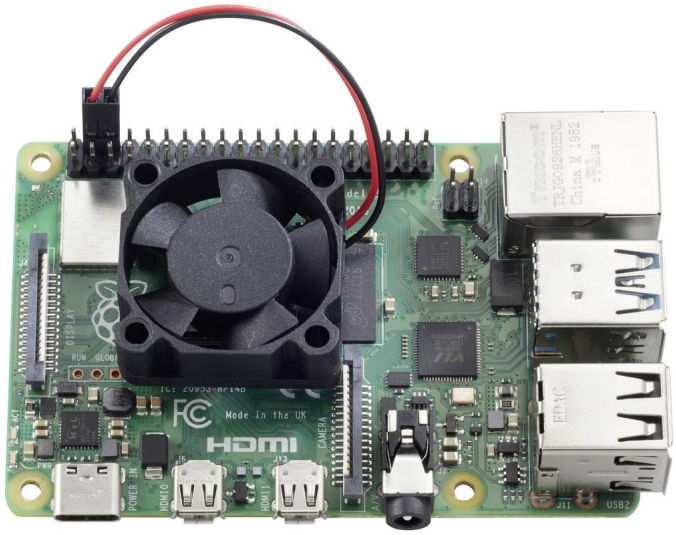


FIGURE 2.7 – Refroidisseur de Raspberry Pi.

Caractéristique

- Tension d'entrée : 5 V DC fournie via un connecteur de ventilateur à quatre broches sur le Raspberry Pi 5.
- Contrôle de la vitesse du ventilateur : modulation de largeur d'impulsion (PWM) avec tachymètre.
- Débit d'air maximal : 2,79 CFM.
- Vitesse maximale du ventilateur : 8 000 tr/min ± 15 %.
- Matériau du boîtier : ABS (base, cadre et couvercle) et PC (ventilateur).

Caméra Raspberry Pi V2

L'acquisition du flux vidéo en temps réel est assurée par le module officiel Raspberry Pi Camera v2. Équipé d'un capteur CMOS Sony IMX219 de 8 MP à focale fixe, ce dispositif permet la capture d'images statiques jusqu'à 3280 × 2464 pixels et prend en charge les formats vidéo standards, notamment le 1080p à 30 fps et le 720p à 60 fps.

Sur le plan matériel, ses dimensions réduites (25 × 23 × 9 mm) et sa masse de seulement 3 g répondent parfaitement aux contraintes d'intégration des systèmes embarqués. Le module s'interface nativement avec la carte de traitement via le bus dédié CSI (Camera Serial Interface), garantissant un transfert de données à haut débit et à faible latence, pleinement compatible avec la bibliothèque de vision par ordinateur OpenCV.

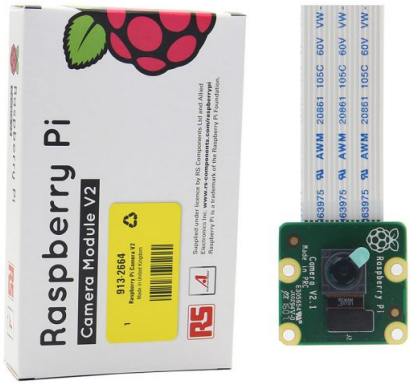


FIGURE 2.8 – Caméra Raspberry Pi Module 2.

Caractéristiques techniques

- Lentille à mise au point fixe (focus fixe).
- Résolution native : 8 MP.
- Résolution photo : 3280 × 2464 pixels.
- Résolution vidéo : 1080p à 30 ips, 720p à 60 ips et 640 × 480 à 90 ips.
- Dimensions : 25 mm × 23 mm × 9 mm.
- Poids : 3 g.
- Connexion au Raspberry Pi via une nappe flexible fournie.
- Compatible avec la dernière version de Raspbian.

2.4 Concepts de base détection/classification

2.4.1 Détection d'objets

La détection d'objets constitue une tâche fondamentale de la vision par ordinateur, combinant simultanément la localisation spatiale et la classification sémantique de multiples entités au sein d'une même scène visuelle. Contrairement à la classification globale, elle résout explicitement le problème de la position en générant des boîtes englobantes autour des objets d'intérêt. Chaque prédiction est assortie d'un score de confiance, reflétant le degré de certitude du modèle quant à la présence de la classe cible dans la région délimitée.

Dans le cadre de nos travaux, cette fonction critique est assignée à l'algorithme YOLO (You Only Look Once), dont le rôle est d'isoler en temps réel les panneaux de signalisation à partir du flux brut capturé par la caméra.

2.4.2 Classification d'images

La classification d'images consiste à assigner une étiquette unique à un contenu visuel parmi un ensemble de catégories prédéfinies. À la différence de la détection, cette tâche fait abstraction

de la localisation spatiale pour se concentrer exclusivement sur l'identification de la nature de l'objet, répondant ainsi à la problématique : « Quelle est la classe de cet objet ? » par exemple : « c'est un chien », « c'est une voiture ».

Au sein de notre architecture, cette mission est dévolue au modèle MobileNetV2. Ce réseau de neurones reçoit en entrée la *région d'intérêt*[2] préalablement extraite par YOLO. Il réalise ensuite l'inférence nécessaire pour catégoriser le panneau détecté parmi les 43 classes du jeu de données de référence GTSRB (German Traffic Sign Recognition Benchmark).

2.4.3 Complémentarité détection et classification

Ces deux tâches sont complémentaires et constituent ensemble le pipeline séquentiel de notre système. YOLOv8n assure la détection spatiale en identifiant la position du panneau dans l'image, tandis que MobileNetV2 assure la classification catégorielle en déterminant le type de panneau. Cette séparation des responsabilités permet d'optimiser chaque composant indépendamment et d'obtenir un système plus performant et plus flexible.

La figure 2.9 illustre la synergie entre les deux tâches constituant le pipeline de notre système.

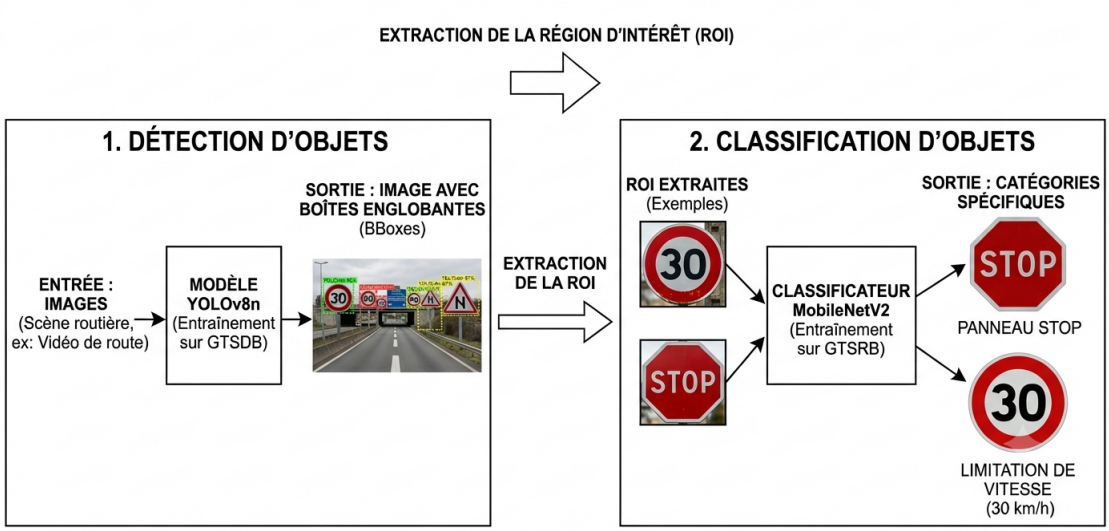


FIGURE 2.9 – Distinction entre la détection et la classification dans le pipeline du système.

2.5 Les réseaux de neurones convolutifs

Un type particulier de perceptron multicouche est le réseau de neurones convolutif (abrégé CNN, pour Convolutional Neural Network). Contrairement à une architecture d'apprentissage profond (Deep Learning), un réseau de neurones simple ne peut pas apprendre des caractéristiques complexes. Les CNN ont démontré d'excellentes performances dans de nombreuses applications, telles que la classification d'images, la détection d'objets et l'étiquetage de scènes. L'idée principale derrière un CNN est qu'il extrait d'abord des caractéristiques locales à partir des premières couches, avant de les transférer vers les couches suivantes pour former des caractéristiques plus

2. Une région d'intérêt (ou ROI, pour Region of Interest) est une zone spécifique délimitée dans un ensemble de données (image, vidéo, carte) ou d'un projet destinée à être analysée, modifiée ou mesurée en priorité, en ignorant le reste.

complexes. Sur le plan architectural, un CNN se compose de couches convolutionnelles, de couches de mise en commun (pooling) et de couches entièrement connectées (FC) [13].

2.5.1 Architectures de réseaux de neurones profonds

La vision par ordinateur connaît aujourd'hui une évolution rapide, portée en grande partie par le développement de l'apprentissage profond. Dans ce domaine, le réseau de neurones convolutif (désigné par l'acronyme CNN, pour Convolutional Neural Network) s'impose comme une architecture de référence. Un CNN se compose de deux parties bien distinctes :

La première est la partie convolutive à proprement parler, qui fait office d'extracteur de caractéristiques pour les images. Au cours de cette étape, l'image d'entrée traverse une succession de filtres, ou noyaux de convolution, qui génèrent de nouvelles images intermédiaires appelées cartes de caractéristiques (ou cartes de convolution). Ces cartes passent ensuite par une fonction d'activation non linéaire, généralement la fonction ReLU (Rectified Linear Unit), permettant au réseau d'apprendre des motifs complexes. De plus, certaines opérations intermédiaires réduisent la résolution spatiale de l'image grâce à un processus de maximum local (Max-Pooling). À la fin de ce bloc convolutif, les cartes de caractéristiques obtenues sont mises à plat (flatten) et concaténées pour former un unique vecteur de caractéristiques, souvent qualifié de code CNN.

Ce code CNN, généré en sortie de la partie convolutive, est ensuite injecté à l'entrée de la deuxième partie du réseau, constituée de couches entièrement connectées (Fully Connected). Le rôle de ce second bloc est de combiner les caractéristiques du code CNN afin de procéder à la classification de l'image. La sortie du réseau est alors représentée par une dernière couche comprenant un neurone par catégorie, dont les valeurs fournissent une distribution de probabilité sur les différentes classes cibles.

Une architecture CNN typique est illustrée par la Figure 2.10 [13].

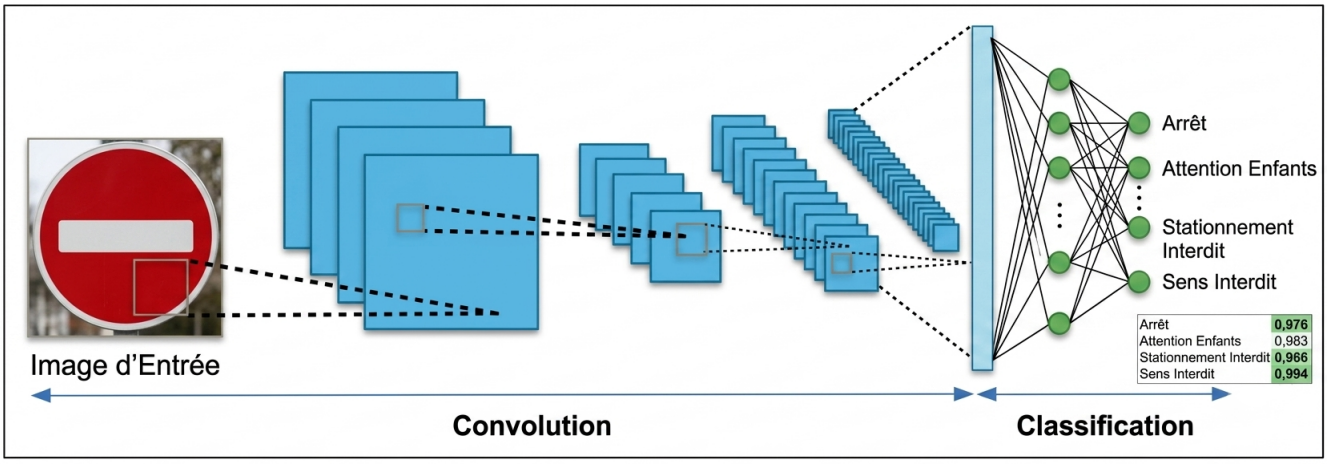


FIGURE 2.10 – Les réseaux de neurones convolutifs.

2.5.2 Les couches de réseaux de neurones convolutionnels

Il y a plusieurs couches différentes dans CNN comme le montre la figure 2.11.

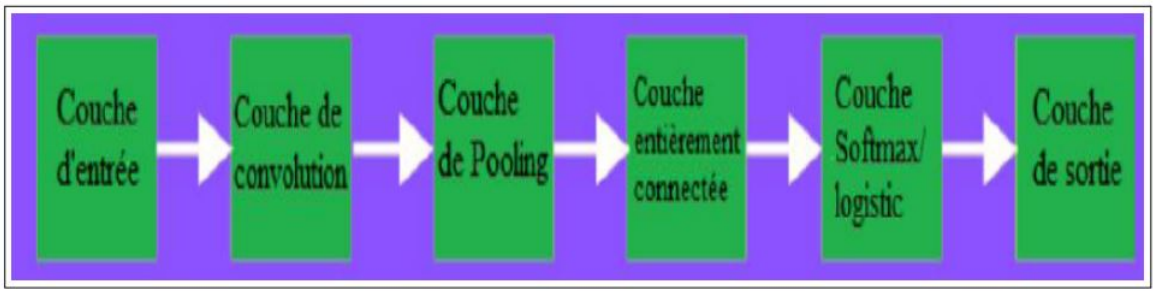


FIGURE 2.11 – Les couches de CNN.

- La couche de convolution (CONV) c'est la couche la plus importante et le cœur des éléments constitutifs du réseau convolutif, et c'est aussi elle qui effectue le plus de calculs lourds, car les caractéristiques de l'image sont extraites dans cette couche [13].
- La couche d'activation (ReLU) Placée immédiatement après la convolution, elle introduit de la non-linéarité dans le modèle pour lui permettre d'apprendre des formes complexes, comme les contours d'un panneau de signalisation. Concrètement, elle agit comme un filtre de tri : elle conserve toutes les valeurs positives telles quelles et remplace toutes les valeurs négatives par un zéro. Cela permet de nettoyer l'image en éliminant le bruit inutile, d'activer le réseau uniquement sur les détails pertinents et d'accélérer considérablement la vitesse d'apprentissage du modèle.
- La couche de Pooling est utilisée pour réduire le volume spatial de l'image d'entrée après la convolution. Sa fonction est de réduire progressivement la taille spatiale de la représentation pour réduire le nombre de paramètres et de calculs dans le réseau, et donc de contrôler également l'overfitting [13].
- Couche entièrement connectée (FC) après plusieurs couches de convolution et de max-pooling le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Il connecte les neurones d'une couche aux neurones d'une autre couche. Il est utilisé pour classer les images entre différentes catégories par apprentissage [13].
- Couche Logistique ou Softmax est la fin de la couche entièrement connectée. La logistique est utilisée pour la classification binaire et Softmax est pour la multi-classification [13].
- Couche de sortie (output layer) la couche de sortie contient l'étiquette qui est sous forme codée [13].

2.6 YOLO

You Only Look Once (YOLO) est un algorithme de pointe de détection d'objets en temps réel introduit en 2015 par Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi dans leur célèbre article de recherche You Only Look Once : Détection unifiée d'objets en temps réel [14].

Les auteurs présentent le problème de la détection d'objets comme une tâche de régression plutôt que de classification en séparant spatialement les boîtes de délimitation et en associant des probabilités à chaque image détectée à l'aide d'un seul réseau neuronal convolutionnel (CNN).

2.6.1 YOLO-V8

Introduite en janvier 2023 par Ultralytics, l'entité également à l'origine de YOLOv5, le modèle s'est rapidement imposé comme le nouvel état de l'art en matière de détection d'objets en temps réel. Les analyses comparatives initiales démontrent sa nette supériorité face à ses prédécesseurs, notamment YOLOv5 et YOLOv6. À résolution d'image égale (640 pixels), toutes les variantes de YOLOv8 affichent un meilleur débit et une efficacité computationnelle accrue pour un nombre de paramètres similaire, confirmant ainsi la pertinence de ses réformes architecturales au niveau matériel.

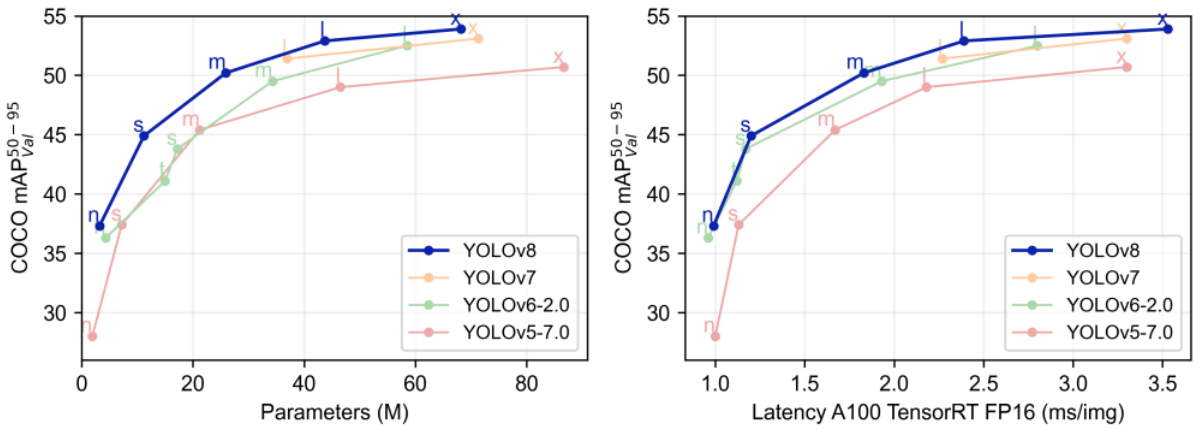


FIGURE 2.12 – YOLO-V8 comparaison avec les prédécesseurs.

Source : PyTorch, « YOLO Model Comparison ».

2.6.2 Architecture YOLO

YOLO est un modèle de détection d'objets dit« single-stage » (en une seule étape). Contrairement aux approches traditionnelles qui génèrent d'abord des régions candidates avant de les analyser, YOLO traite l'image entière en une seule passe à travers le réseau.

Le réseau divise l'image en une grille géométrique et prédit directement, pour chaque cellule, les boîtes englobantes (bounding boxes) ainsi que les scores de confiance associés. Les détections redondantes ou trop faibles sont ensuite éliminées par l'algorithme de suppression des non-maxima (NMS) [3]. Permettant d'éliminer les détections redondantes et de ne conserver que la boîte la plus représentative pour chaque objet détecté.

2.6.3 Principe de fonctionnement

L'algorithme YOLO se distingue des méthodes traditionnelles de détection d'objets par sa capacité à traiter la localisation et la classification en une seule et unique opération numérique. Au lieu d'isoler des régions d'intérêt potentielles pour les analyser séparément, il formule la

3. Le *Non-Maximum Suppression (NMS)* est une technique de post-traitement utilisée en détection d'objets afin d'affiner les prédictions d'un modèle. Lors de l'inférence, plusieurs boîtes englobantes (*bounding boxes*) peuvent être générées pour un même objet, chacune associée à un score de confiance. Ces redondances proviennent du fait que le modèle peut détecter un même objet à différentes positions ou échelles. Le NMS consiste à conserver uniquement la boîte ayant le score le plus élevé et à supprimer les autres boîtes fortement chevauchantes, ce qui permet d'obtenir une détection finale plus précise et sans doublons.

détection comme un problème de régression unique. Les coordonnées des boîtes englobantes et les probabilités des classes sont prédites directement à partir des pixels de l'image d'entrée en une seule passe à travers le réseau neuronal.

Pour mieux appréhender la fluidité de ce traitement, la démarche séquentielle menant à la détection est illustrée par la figure suivante :

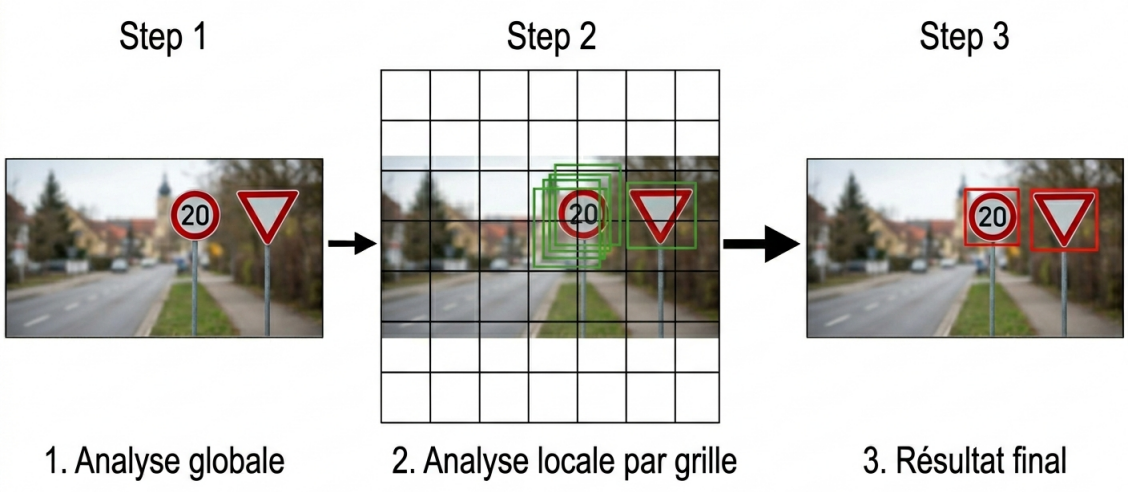


FIGURE 2.13 – Illustration graphique du fonctionnement du modèle YOLO

- **Analyse globale (Single Pass)** :Le processus débute par la capture de l'image, de la même manière qu'un appareil photo fige une scène. Le réseau neuronal convolutif réalise alors une unique passe avant pour analyser l'intégralité de l'image brute en une seule fois. Cette vue d'ensemble permet de capturer instantanément le contexte global de la scène (la route, le décor et la présence potentielle d'objets) avant toute analyse locale.
- **Analyse locale par grille (Grid-based Analysis)** : Dans l'étape suivante, l'algorithme divise structurellement l'image en un réseau de petits carrés, semblable à un damier, formant une grille de cellules (par exemple 6×6 à titre illustratif). C'est au sein de chaque carré que YOLO va rechercher des indices visuels et des caractéristiques géométriques (bords, formes, textures). Chaque cellule devient alors responsable de prédire simultanément si un objet s'y trouve, en générant des boîtes englobantes assorties d'un score de certitude et des probabilités de classe.
- **Résultat final** : Après application d'un seuil sur les scores de confiance et de l'algorithme NMS, les détections redondantes sont éliminées. Le système conserve uniquement les boîtes les plus précises, localisant correctement les objets détectés avec une forte confiance.

2.6.4 YOLOv8n pour la détection embarquée

YOLOv8 représente la huitième génération de l'architecture YOLO, développée par Ultralytics. La variante YOLOv8n constitue la version la plus légère de cette famille, spécialement conçue pour les plateformes à ressources limitées. Elle offre un compromis optimal entre précision de détection et efficacité computationnelle, avec un nombre réduit de paramètres permettant une inférence rapide sur des processeurs embarqués tels que celui de la Raspberry Pi 5.

2.7 MobileNet

MobileNetV2 est un réseau de neurones convolutif léger (Lightweight Convolutional Neural Network) développé par Google Research en 2018 pour les applications de vision par ordinateur sur les appareils mobiles et les systèmes embarqués. Cette architecture a été conçue pour répondre aux contraintes de calcul, de mémoire et de consommation énergétique imposées par les plateformes à ressources limitées.

En effet, les réseaux de neurones convolutifs classiques, tels que AlexNet, VGGNet et ResNet, ont démontré d'excellentes performances dans diverses tâches de vision par ordinateur. Cependant, ces architectures nécessitent un nombre important de paramètres ainsi qu'une grande puissance de calcul, ce qui entraîne une forte consommation de mémoire et un temps d'exécution élevé.

Suite au succès de la première version, plusieurs améliorations ont été introduites au sein de la famille MobileNet :

- **MobileNet (2017)** : introduction des convolutions séparables en profondeur (*Depthwise Separable Convolutions*) permettant de réduire considérablement le nombre de paramètres et le coût de calcul.
- **MobileNetV2 (2018)** : ajout des blocs résiduels inversés (*Inverted Residuals*) et des goulots d'étranglement linéaires (*Linear Bottlenecks*), améliorant l'efficacité du réseau tout en conservant de bonnes performances de classification.
- **MobileNetV3 (2019)** : optimisation automatique de l'architecture à l'aide de techniques de recherche neuronale (*Neural Architecture Search*) et intégration de mécanismes d'attention pour améliorer davantage le compromis entre précision et efficacité.

Parmi ces différentes versions, **MobileNetV2** est devenue l'une des architectures les plus utilisées dans les applications de vision par ordinateur embarquées grâce à son excellent compromis entre précision, rapidité d'exécution et faible consommation de ressources.

2.7.1 Principe de fonctionnement

Le pilier technologique garantissant la légèreté de MobileNetV2 réside dans l'exploitation exclusive des convolutions séparables en profondeur. Ce procédé permet de découpler l'extraction spatiale du regroupement des canaux, réduisant de manière significative la complexité mathématique des opérations.

Comme l'illustre la Figure 2.14, le traitement se décompose en deux phases distinctes, comparativement à une approche standard :

A. Convolution Classique : Dans un CNN traditionnel, un filtre tridimensionnel de taille $k \times k \times N$ est appliqué simultanément sur toute la profondeur du tenseur d'entrée ($M \times M \times 3$). Cette méthode traite toutes les dimensions en une seule opération pour générer le volume de produit ($M \times M \times M$), ce qui engendre un coût computationnel particulièrement élevé.

B. Convolution Séparable en Profondeur (MobileNetV2) : L'architecture fragmente cette opération en deux étapes consécutives, séparées par une phase de post-traitement :

1. **Convolution en Profondeur (Depthwise)** : Un ensemble de N filtres bidimensionnels de taille $k \times k$ est appliqué individuellement sur chaque canal de l'image d'entrée, de

manière isolée. Cette étape intermédiaire ($1 \times 1 \times N$) permet d'extraire les caractéristiques spatiales en allégeant considérablement le volume initial de multiplications.

2. **Convolution Ponctuelle (Pointwise)** : Une convolution de dimension $1 \times 1 \times N$ intervient ensuite pour fusionner de manière linéaire les sorties générées par l'étape précédente et reconstruire le tenseur de résultat final.

En adoptant cette structure séparable, MobileNetV2 réduit le nombre de paramètres et diminue le coût de calcul d'un facteur 8 à 9 par rapport aux convolutions standard, n'induisant qu'une baisse négligeable de la précision globale.

2.7.2 Mécanisme des convolutions séparables en profondeur

Comme l'illustre la Figure 2.15, MobileNetV2 fragmente cette opération en deux étapes consécutives :

- **Convolution en profondeur (Depthwise Convolution – Étape 1)** : Un filtre bidimensionnel indépendant est appliqué individuellement sur chaque canal de l'image d'entrée, sans les combiner. Cette isolation des canaux permet d'alléger considérablement le volume initial de multiplications.
- **Convolution ponctuelle (Pointwise Convolution – Étape 2)** : Une convolution de dimension 1×1 intervient ensuite pour fusionner de manière linéaire les sorties générées par l'étape précédente et reconstruire un nouvel espace de caractéristiques.

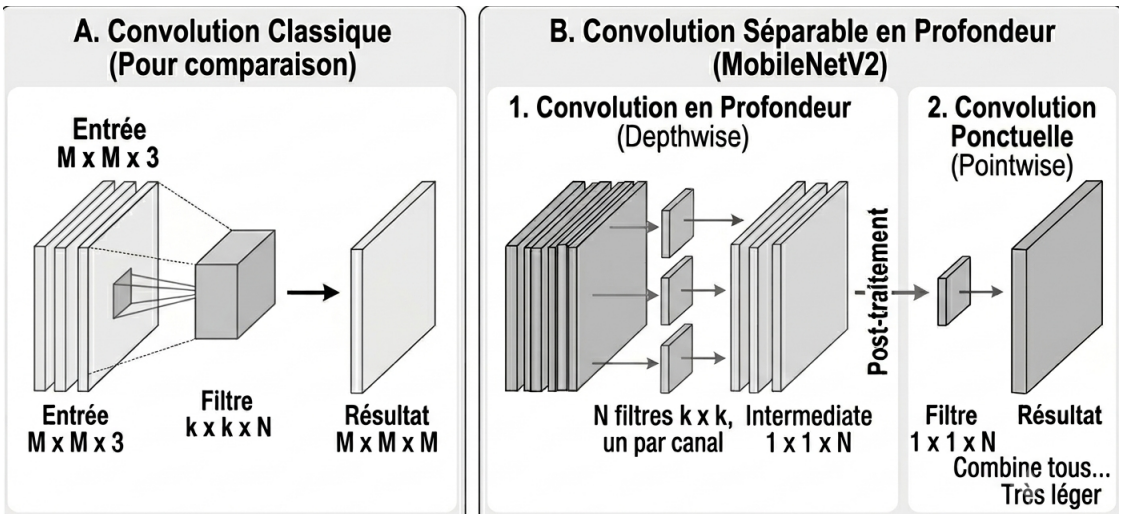


FIGURE 2.14 – Comparaison entre une convolution classique et la convolution séparable en profondeur

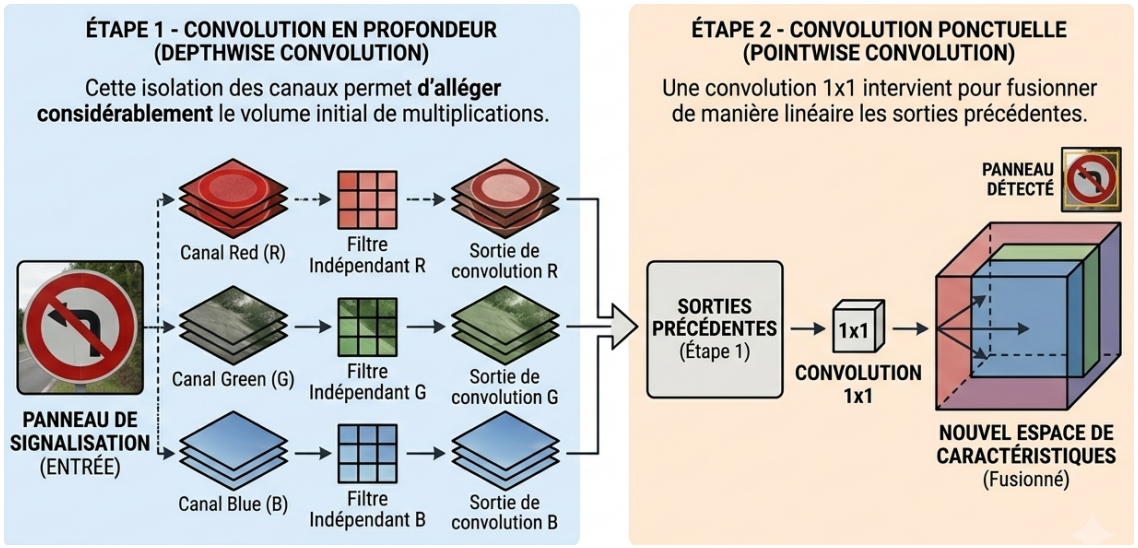


FIGURE 2.15 – Convolution séparable en profondeur.

2.7.3 Adaptation à notre projet

Dans notre système, MobileNetV2 joue le rôle de classificateur en aval de YOLOv8n. Il reçoit en entrée les régions d'intérêt extraites par le détecteur, redimensionnées à 224×224, et retourne la classe du panneau parmi les 43 classes du dataset GTSRB.

L'adaptation au projet repose sur le *transfer learning*. Le réseau exploite des poids pré-entraînés sur le jeu de données ImageNet, qui lui confèrent dès le départ une capacité d'extraction de caractéristiques visuelles génériques. Seule la couche de classification finale est remplacée et réentraînée pour correspondre aux 43 classes du GTSRB.

2.8 Conclusion

En somme, ce chapitre a posé les bases théoriques et logicielles nécessaires à la réalisation de notre système de reconnaissance de panneaux. L'approche retenue segmente le traitement en deux tâches complémentaires : la localisation spatiale d'une part, et l'identification de la signalisation parmi les 43 classes du GTSRB d'autre part.

La viabilité de ce pipeline repose sur l'association de deux architectures hautement optimisées :

YOLOv8n : assure une détection en temps réel grâce à sa structure en une seule passe.

MobileNetV2 : garantit une classification rapide et légère, un résultat rendu possible par l'usage de convolutions séparables en profondeur et d'un apprentissage par transfert (*transfer learning*).

Cette combinaison logicielle répond directement aux exigences de la plateforme embarquée Raspberry Pi 5, combinant ainsi sobriété matérielle, vitesse d'inférence et précision.

Le chapitre suivant détaillera la conception et l'implémentation concrète de cette solution globale.

Chapitre 3

Conception et architecture du système proposé

3.1 Introduction

Après avoir présenté les fondements théoriques relatifs à la détection d'objets, à la classification d'images ainsi qu'aux modèles YOLOv8n et MobileNetV2 dans le chapitre précédent, ce chapitre est consacré à la conception de la solution proposée.

L'objectif principal est de décrire l'organisation générale du système développé, les interactions entre ses différents modules ainsi que le déroulement du pipeline de traitement permettant la détection et la reconnaissance des panneaux de signalisation en temps réel.

Le système repose sur une architecture modulaire composée de plusieurs étapes successives. Dans un premier temps, une caméra embarquée capture en continu les images de l'environnement routier. Ces images sont ensuite transmises à un modèle de détection basé sur YOLOv8n afin de localiser les panneaux présents dans la scène. Les régions d'intérêt correspondant aux panneaux détectés sont ensuite extraites avant d'être analysées par un modèle MobileNetV2 chargé d'identifier précisément leur catégorie. Enfin, les résultats obtenus sont affichés en temps réel sur le flux vidéo afin d'assister le conducteur durant la conduite.

Ce chapitre présente également l'architecture globale de la solution, la description fonctionnelle des différents modules qui la composent ainsi que les jeux de données retenus pour l'entraînement des modèles de détection et de classification.

3.2 Analyse fonctionnelle du système

Avant de développer les différents modules de l'application, il est nécessaire de définir les besoins auxquels le système doit répondre ainsi que les principales contraintes liées à son fonctionnement. Cette analyse fonctionnelle permet d'identifier les fonctions essentielles assurées par chaque composant de l'architecture proposée.

3.2.1 Besoin fonctionnel

Le système développé a pour objectif d'assister le conducteur en détectant automatiquement les panneaux de signalisation présents dans son environnement puis en identifiant leur signification avec un niveau de précision élevé. L'ensemble des traitements est réalisé de manière autonome, sans intervention de l'utilisateur, afin de garantir une utilisation simple et adaptée à un contexte de conduite.

Pour atteindre cet objectif, plusieurs fonctions essentielles doivent être assurées :

1. Acquisition continue des images provenant de la caméra.
2. Détection automatique des panneaux de signalisation.
3. Extraction des régions d'intérêt correspondant aux panneaux détectés.
4. Classification précise de chaque panneau identifié.
5. Affichage des résultats.
6. Génération éventuelle d'alertes destinées au conducteur.

L'ensemble de ces fonctions contribue à fournir une assistance efficace au conducteur en améliorant sa perception des panneaux de signalisation tout en réduisant le risque d'erreur lié à une mauvaise visibilité ou à une inattention momentanée.

3.2.2 Contraintes du système

Outre les fonctionnalités précédemment décrites, le système doit satisfaire plusieurs contraintes techniques afin de garantir son bon fonctionnement sur une plateforme embarquée.

La première contrainte concerne l'exécution en temps réel. Le délai entre l'acquisition d'une image et l'affichage du résultat doit être suffisamment faible pour permettre au conducteur de réagir rapidement face aux informations détectées.

La deuxième contrainte est liée aux ressources matérielles disponibles sur le Raspberry Pi 5. Les modèles utilisés doivent présenter un compromis satisfaisant entre précision et complexité de calcul afin de limiter la consommation du processeur et de la mémoire tout en conservant des performances élevées.

Le système doit également rester robuste face aux différentes conditions rencontrées sur la route, telles que les variations d'éclairage, les changements de distance, les rotations des panneaux ou encore les occultations partielles. Enfin, l'architecture proposée doit être suffisamment modulaire pour faciliter l'intégration de nouveaux jeux de données ou l'ajout de nouvelles catégories de panneaux dans de futures évolutions.

3.3 Architecture globale du système

3.3.1 Vue d'ensemble

L'architecture que nous avons développée est conçue selon une approche modulaire, dans laquelle chaque module réalise une tâche spécifique au sein du processus de reconnaissance des

panneaux. Cette organisation facilite le développement, la maintenance ainsi que l'évolution du système, car elle garantit une exécution efficace sur le Raspberry Pi 5.

Comme l'illustre la Figure 3.1, le fonctionnement du système repose sur un pipeline séquentiel. À partir de l'acquisition des images par la caméra, les données transitent successivement par les modules de prétraitement, de détection, d'extraction des régions d'intérêt, de classification et d'affichage des résultats.

Chaque module reçoit les données produites par le module précédent, les traite, puis transmet les résultats au module suivant. Cette séparation des responsabilités permet d'améliorer la lisibilité de l'architecture, de simplifier les opérations de maintenance et de faciliter l'intégration d'améliorations futures. La figure 3.1 illustre l'ensemble du pipeline séquentiel du système proposé.

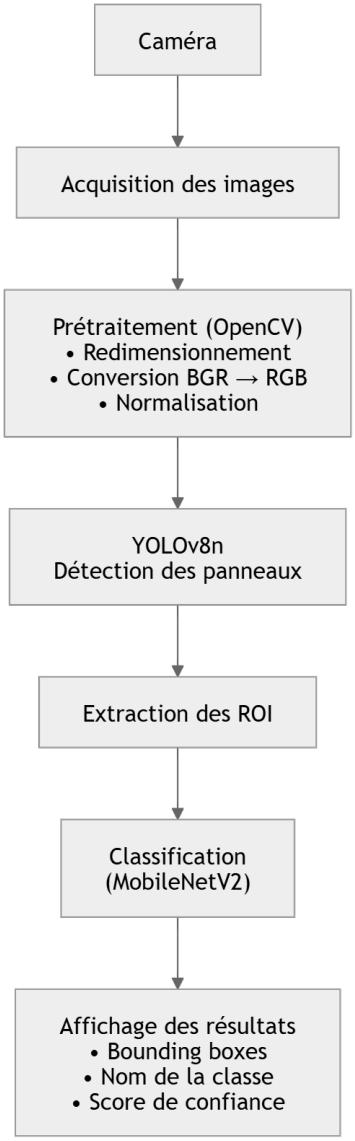


FIGURE 3.1 – Chaîne de traitement du système.

3.4 Description du pipeline de traitement

Le pipeline de traitement débute par l'acquisition continue des images à partir d'une caméra connectée au Raspberry Pi 5. Les images capturées représentent l'environnement routier et constituent les données d'entrée du système.

Avant leur analyse, les images subissent une étape de prétraitement visant à adapter leur format aux exigences des modèles d'intelligence artificielle. Cette étape comprend principalement le redimensionnement des images ainsi que les conversions nécessaires pour assurer leur compatibilité avec les bibliothèques utilisées.

Les images prétraitées sont ensuite transmises au modèle YOLOv8n, chargé de détecter les panneaux de signalisation présents dans la scène. Pour chaque panneau identifié, le modèle fournit les coordonnées de la boîte englobante ainsi qu'un score de confiance associé à la détection.

À partir des coordonnées obtenues, les régions d'intérêt sont extraites afin d'isoler les panneaux du reste de l'image. Cette opération permet de réduire la quantité d'informations à traiter par le classificateur et de limiter l'influence de l'arrière-plan sur les performances de reconnaissance. Ces dernières sont ensuite redimensionnées aux dimensions requises par MobileNetV2 avant d'être soumises au processus de classification. Le modèle identifie alors la catégorie correspondant à chaque panneau détecté et attribue une probabilité à la prédiction réalisée.

Enfin, les résultats de la détection et de la classification sont affichés sur le flux vidéo en temps réel. Chaque panneau reconnu est représenté par une boîte englobante accompagnée de son nom, ce qui permet au conducteur de visualiser rapidement les informations pertinentes.

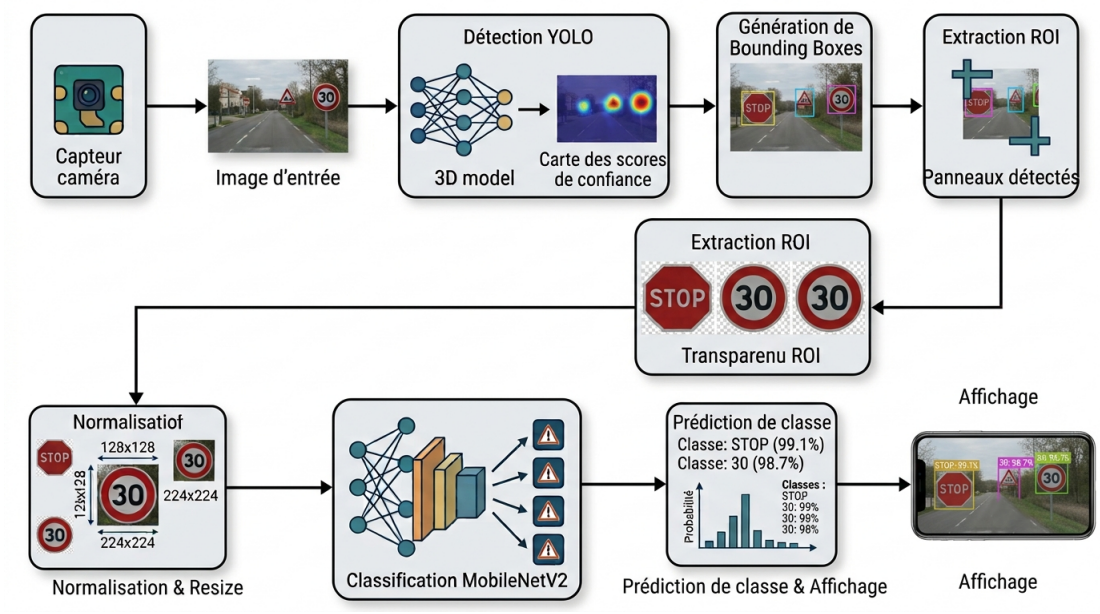


FIGURE 3.2 – Flux de traitement d'une image.

3.5 Description des modules

Cette section présente les différents modules constituant le système développé ainsi que leur rôle dans le pipeline de traitement.

Module d'acquisition

Le système débute par l'acquisition en continu d'un flux vidéo à l'aide d'une caméra connectée au Raspberry Pi 5. Contrairement à une approche basée sur des images statiques, le système traite successivement les images constituant la vidéo afin d'assurer une détection des panneaux en temps réel. Chaque image capturée est immédiatement transmise au pipeline de traitement.

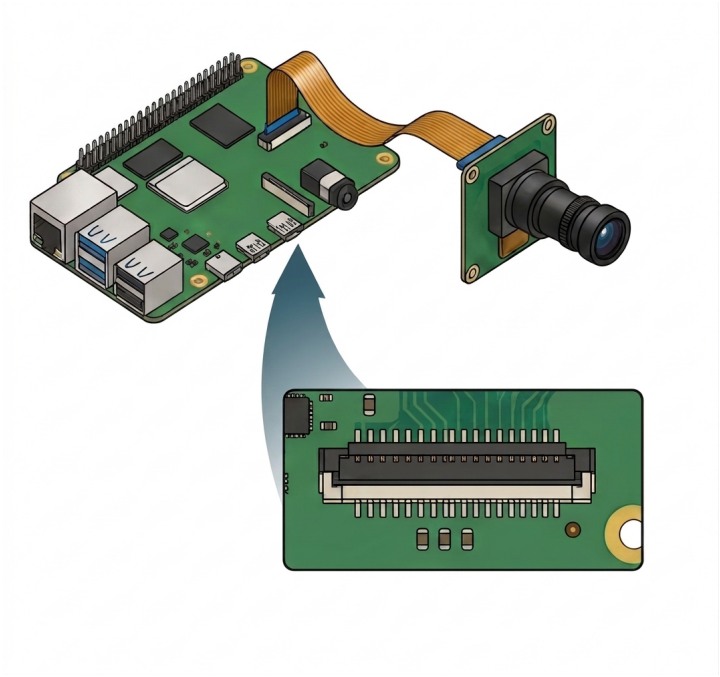


FIGURE 3.3 – Configuration matérielle pour l'acquisition des images

Le choix d'une acquisition continue permet d'assurer une surveillance permanente de l'environnement routier et de garantir un fonctionnement en temps réel.

Module de prétraitement

Avant d'être analysée par le modèle de détection, chaque image subit une phase de prétraitement destinée à assurer la compatibilité avec les modèles d'intelligence artificielle utilisés. Cette étape comprend notamment le redimensionnement des images, leur conversion de format ainsi que les opérations de normalisation nécessaires.

Dans notre système, ces opérations sont réalisées à l'aide de la bibliothèque OpenCV, cette étape contribue à uniformiser les données d'entrée et à améliorer la stabilité du pipeline de traitement.

Module de détection

Le module de détection s'appuie sur le modèle YOLOv8n afin de localiser les panneaux de signalisation présents dans chaque image. Les résultats produits comprennent les coordonnées des boîtes englobantes ainsi qu'un score de confiance associé à chaque détection.

Ces informations sont ensuite transmises au module chargé de l'extraction des régions d'intérêt.



FIGURE 3.4 – Visualisation des boîtes englobantes et scores de confiance générés par YOLO
Source : Sichkar et Kolyubin [15].

Module d'extraction des régions d'intérêt

Après la détection, les coordonnées des boîtes englobantes sont utilisées afin d'extraire les régions d'intérêt, comme présenté à la Figure 3.4. Cette opération permet d'isoler uniquement les panneaux détectés tout en supprimant les informations inutiles provenant de l'arrière-plan. Les régions extraites sont ensuite redimensionnées selon les dimensions attendues par le classificateur MobileNetV2.

Cette opération permet de fournir au module de classification une image centrée uniquement sur le panneau détecté.

Module de classification

Le module de classification reçoit les régions d'intérêt extraites puis les analyse à l'aide du modèle MobileNetV2. Son objectif consiste à déterminer précisément la catégorie du panneau détecté parmi les différentes classes apprises durant la phase d'entraînement.

Le résultat de cette étape correspond au nom du panneau ainsi qu'à la probabilité associée à la prédiction.

Module d'affichage

Le module d'affichage constitue la dernière étape du pipeline. Il superpose les résultats obtenus directement sur le flux vidéo en affichant les boîtes englobantes, le nom des panneaux reconnus ainsi que, si nécessaire, des indications complémentaires destinées à assister le conducteur.

Cette restitution en temps réel permet de rendre le système facilement exploitable dans un contexte de conduite.

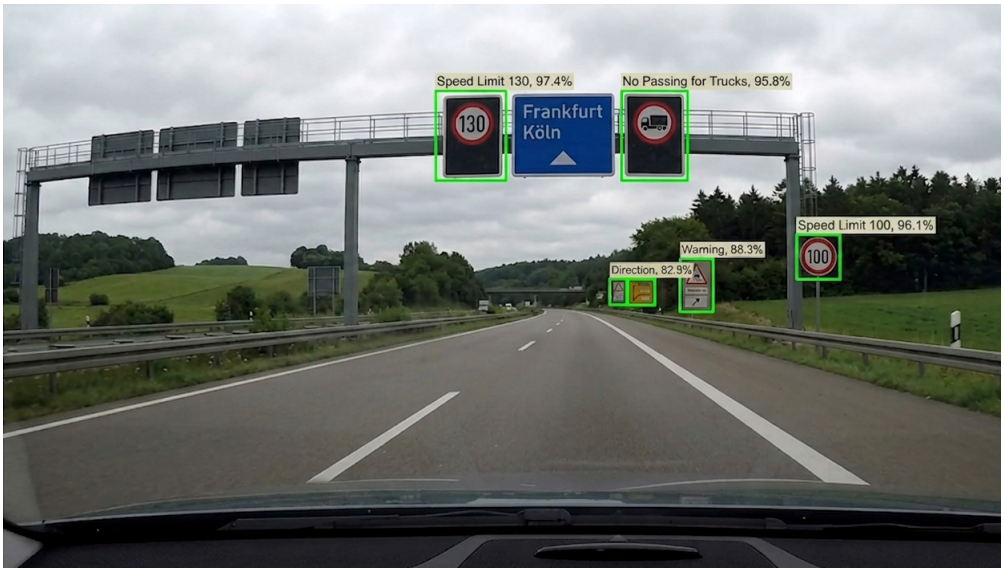


FIGURE 3.5 – Affichage en temps réel des résultats de détection et de classification des panneaux de signalisation.

Source : German Traffic Sign Detection Benchmark (GTSDB), disponible sur Kaggle.

3.6 Datasets utilisés

Avant d'entraîner les modèles de détection et de classification, il est indispensable de disposer de jeux de données représentatifs des panneaux de signalisation. Dans le cadre de ce projet, deux jeux de données publics ont été retenus. Le premier, German Traffic Sign Detection Benchmark (GTSDB), est utilisé pour entraîner le modèle de détection YOLOv8n, tandis que le second, German Traffic Sign Recognition Benchmark (GTSRB), est utilisé pour entraîner le modèle de classification MobileNetV2. Ces deux bases de données sont largement utilisées dans la littérature scientifique et constituent une référence pour l'évaluation des systèmes de reconnaissance de panneaux de signalisation.

3.6.1 German Traffic Sign Detection Benchmark

Le GTSDB (German Traffic Sign Detection Benchmark) est un dataset standardisé conçu pour valider les performances de détection de panneaux au sein d'images réelles [16]. Sa structure se distingue nettement de celle du GTSRB ; alors que ce dernier fournit des images déjà détourées, le GTSDB impose de localiser précisément les cibles dans leur contexte routier global au moyen de boîtes englobantes.

Ce corpus se compose de 900 images de 1360×800 pixels, scindées en un sous-ensemble d'entraînement (600 images) et un sous-ensemble de test (300 images) [16]. Les prises de vue couvrent une grande variété de configurations spatiales (milieux urbains, ruraux et autoroutiers) et temporelles (variations de luminosité diurne). Les scénarios comptabilisent plus de 1 200 instances de panneaux, classées selon quatre catégories majeures : les signaux d'interdiction, de danger, d'obligation et les autres types de signalisation.

TABLE 3.1 – Répartition des catégories dans le dataset GTSDB

Catégorie	**Nombre d'instances**	**Pourcentage**
Interdiction	396	46,81 %
Danger	156	18,44 %
Obligation	114	13,48 %
Autres	180	21,28 %
Total	**846**	**100 %**

3.6.2 German Traffic Sign Recognition Benchmark

Le GTSRB (German Traffic Sign Recognition Benchmark) constitue la référence mondiale pour l'évaluation des algorithmes de classification de panneaux de signalisation. Introduit lors de la compétition IJCNN 2011 [16], il se distingue fondamentalement du GTSDB par la nature de ses données : les images fournies se présentent sous la forme de vignettes (ou crops) préalablement recadrées autour des panneaux et directement prêtes pour la phase d'identification.

Ce jeu de données regroupe 39 209 images d'apprentissage et 12 630 images de test [5], réparties à travers 43 classes distinctes conformes à la Convention de Vienne sur la signalisation routière [2]. Les résolutions des images sont hétérogènes, variant de 15×15 à 250×250 pixels, et affichent une forte variabilité visuelle inhérente aux conditions réelles de capture. Le modèle est ainsi confronté à des contraintes complexes telles que des fluctuations d'éclairage, des angles de prise de vue variés, du flou de mouvement, des occlusions partielles ou encore la dégradation physique des infrastructures [6]. C'est précisément sur ce corpus que repose le pré-entraînement du modèle MobileNetV2, chargé de classifier les régions d'intérêt préalablement localisées par YOLOv8n.



FIGURE 3.6 – Exemples d'images extraites du dataset GTSRB.

Source : dataset GTSRB disponible sur Kaggle.

La distribution des classes au sein du dataset est fortement déséquilibrée, certaines classes étant nettement surreprésentées par rapport à d'autres, comme illustré dans le tableau suivant :

TABLE 3.2 – Exemples de distribution des classes dans le jeu de données GTSRB

ID	Classe	Nombre d'images
0	Limitation 20 km/h	210
1	Limitation 30 km/h	2 220
2	Limitation 50 km/h	2 250
3	Limitation 60 km/h	1 410
4	Limitation 70 km/h	1 980
14	Stop	780
17	Sens interdit	1 110

3.6.2.1 Synthèse des jeux de données

Les deux jeux de données utilisés dans ce projet répondent à des objectifs complémentaires. Le GTSDB fournit les informations nécessaires à la localisation des panneaux de signalisation dans une scène routière, tandis que le GTSRB permet d'identifier précisément la catégorie de chaque panneau détecté.

Cette complémentarité est cohérente avec l'architecture proposée, qui repose sur deux modèles

spécialisés : un modèle de détection chargé de localiser les panneaux et un modèle de classification chargé de reconnaître leur signification. Cette approche modulaire améliore la précision du système tout en facilitant son évolution vers de nouveaux jeux de données ou de nouvelles catégories de panneaux.

TABLE 3.3 – Comparaison des jeux de données GTSDB et GTSRB

Critère	**GTSDB**	**GTSRB**
Fonction principale	Détection des panneaux	Classification des panneaux
Type d'entrée	Image de scène routière	Image d'un panneau isolé
Type d'annotation	Boîtes englobantes	Classe du panneau
Nombre de classes	4 catégories	43 classes
Nombre d'images	$\approx$ 900	Plus de 50 000
Modèle utilisé	YOLOv8n	MobileNetV2
Objectif dans notre projet	Localiser les panneaux	Identifier leur catégorie

3.7 Conclusion

Au cours de ce chapitre, la conception et l'architecture globale du système de reconnaissance des panneaux de signalisation proposé ont été présentées. Après une analyse des besoins fonctionnels et des contraintes techniques, une architecture modulaire a été définie afin de répondre aux exigences d'une application embarquée fonctionnant en temps réel sur une plateforme Raspberry Pi 5.

Le pipeline de traitement retenu repose sur une succession d'étapes complémentaires, allant de l'acquisition du flux vidéo jusqu'à l'affichage des résultats. Chaque module a été décrit en précisant son rôle dans le fonctionnement global du système. Le modèle YOLOv8n est chargé de détecter les panneaux de signalisation présents dans les images, tandis que les régions d'intérêt extraites sont ensuite analysées par le modèle MobileNetV2 afin d'identifier précisément la catégorie de chaque panneau. Enfin, les informations obtenues sont restituées directement sur le flux vidéo, permettant ainsi une assistance visuelle en temps réel.

Ce chapitre a également présenté les deux jeux de données retenus pour le développement de la solution. Le GTSDB est utilisé pour la phase de détection, alors que le GTSRB est consacré à l'entraînement du modèle de classification. Leur complémentarité constitue un élément essentiel de l'architecture proposée.

Le chapitre suivant sera consacré aux expérimentations préliminaires. Il présentera la mise en place de l'environnement de développement, ainsi que les premiers essais d'entraînement et de validation des modèles de détection et de classification avant leur apprentissage sur les jeux de données complets.

Chapitre 4

Développement et validation préliminaire du système

4.1 Introduction

Après avoir présenté l'architecture générale du système ainsi que les différentes technologies retenues, il est nécessaire de procéder à une première phase de validation avant l'entraînement des modèles sur les jeux de données de référence. Cette étape a pour objectif de vérifier le bon fonctionnement des différents composants du pipeline proposé et de s'assurer que l'ensemble des outils logiciels et matériels interagissent correctement.

Les expérimentations réalisées dans ce chapitre reposent sur un mini-jeu de données constitué de panneaux de signalisation algériens. Contrairement aux bases de données de référence utilisées par la suite, ce jeu de données a été créé dans le but de valider progressivement le fonctionnement des modules de détection et de classification dans un contexte proche des conditions réelles d'utilisation. Cette approche permet notamment de contrôler les différentes étapes du traitement, depuis l'acquisition des images jusqu'à la production des premiers résultats.

Dans un premier temps, le module de détection basé sur YOLOv8n est mis en oeuvre afin de vérifier sa capacité à localiser les panneaux de signalisation présents dans les images. Les panneaux détectés sont ensuite extraits sous forme de régions d'intérêt et utilisés pour valider le fonctionnement du module de classification reposant sur MobileNetV2. Cette démarche progressive permet de confirmer le bon fonctionnement du pipeline complet avant de réaliser l'entraînement définitif sur les jeux de données GTSDB et GTSRB, présenté dans le chapitre suivant.

4.2 Développement et validation du module de détection

4.2.1 Préparation de l'environnement de développement

La première étape de cette phase expérimentale a consisté à mettre en place un environnement de développement stable et reproductible. L'ensemble du projet a été développé en langage Python, reconnu pour son vaste écosystème de bibliothèques dédiées à l'intelligence artificielle, à la vision par ordinateur et au calcul scientifique.

Afin d'assurer une gestion rigoureuse des dépendances logicielles, un environnement virtuel Py-

thon a été créé avant l'installation des différentes bibliothèques. Cette approche permet d'isoler les packages spécifiques au projet des autres applications installées sur le système, limitant ainsi les risques de conflits de versions et garantissant la reproductibilité des expérimentations sur une autre machine. L'utilisation d'un environnement virtuel facilite également la maintenance et l'évolution du projet tout au long de son développement.

Après la création de cet environnement, les bibliothèques nécessaires au développement ont été installées. La bibliothèque Ultralytics a été utilisée pour l'entraînement, la validation et l'inférence du modèle YOLOv8n. Le framework PyTorch constitue le moteur d'apprentissage profond sur lequel repose le modèle de détection. La bibliothèque OpenCV a été employée pour l'acquisition du flux vidéo en temps réel, le traitement des images ainsi que l'affichage des résultats. Les bibliothèques NumPy et Pandas ont été utilisées respectivement pour les opérations numériques sur les matrices et la manipulation des données tabulaires. Enfin, l'annotation des images a été réalisée à l'aide de l'outil Roboflow, qui permet de produire directement des annotations compatibles avec le format attendu par YOLO.

Une série de vérifications a ensuite été effectuée afin de s'assurer du bon fonctionnement de l'environnement de développement. Ces essais ont notamment consisté à charger le modèle YOLOv8n pré-entraîné, exécuter une première inférence sur des images de test et vérifier le bon fonctionnement des principales bibliothèques utilisées. Cette étape a permis de confirmer que l'environnement était correctement configuré avant d'entreprendre les expérimentations sur les panneaux de signalisation.

Table 4.1 – Environnement logiciel utilisé pour le développement

Composant	**Description**
Langage de programmation	Python 3.11
Environnement virtuel	`venv`
Environnement de développement (IDE)	Visual Studio Code
Framework de Deep Learning	PyTorch
Détection d'objets	Ultralytics YOLOv8
Vision par ordinateur	OpenCV
Calcul numérique	NumPy
Manipulation des données	Pandas
Outil d'annotation	Roboflow

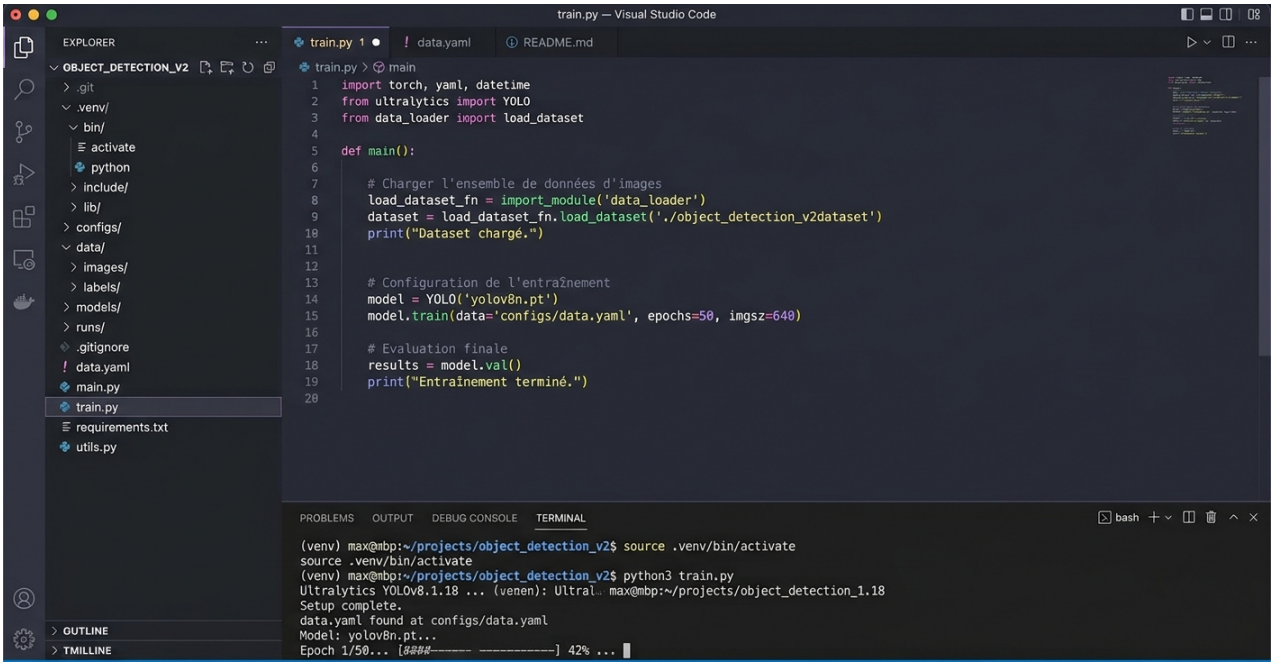


FIGURE 4.1 – Environnement de développement du projet sous Visual Studio Code.

4.2.2 Préparation du jeu de données GTSDB

Après la mise en place de l'environnement de développement, la phase suivante a consisté à préparer le jeu de données destiné à l'apprentissage du modèle de détection. Pour cette étape, le jeu de données German Traffic Sign Detection Benchmark (GTSDB) a été retenu en raison de sa large utilisation dans les travaux de recherche consacrés à la détection des panneaux de signalisation. Il constitue une référence dans ce domaine grâce à la diversité des scènes routières qu'il contient ainsi qu'à la qualité de ses annotations.

Le jeu de données GTSDB regroupe des images acquises dans des conditions réelles de circulation. Chaque image contient un ou plusieurs panneaux de signalisation accompagnés de leurs annotations, représentées sous forme de boîtes englobantes (bounding boxes). Ces annotations permettent d'indiquer avec précision la position des panneaux et servent de vérité terrain (ground truth) lors de l'entraînement du modèle.

Avant de lancer l'apprentissage, le jeu de données a été organisé conformément à la structure attendue par la bibliothèque Ultralytics. Les images et leurs fichiers d'annotations ont été répartis dans les ensembles d'entraînement, de validation et de test. Cette organisation facilite le chargement automatique des données tout au long du processus d'apprentissage et permet d'évaluer les performances du modèle sur des données indépendantes.

Enfin, un fichier de configuration data.yaml a été créé afin de spécifier les chemins d'accès aux différents ensembles de données ainsi que le nombre de classes à détecter. Ce fichier joue un rôle essentiel puisqu'il permet au framework Ultralytics de configurer automatiquement le processus d'entraînement du modèle YOLOv8n.

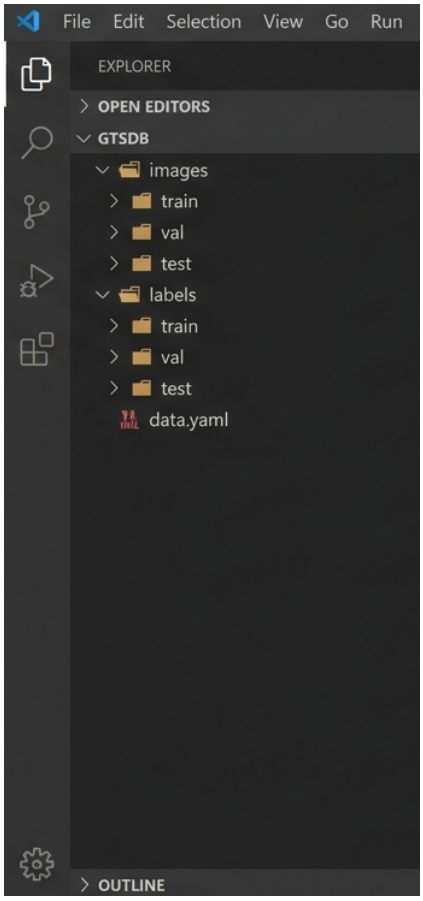


FIGURE 4.2 – Organisation du jeu de données GTSDB.

4.2.3 Configuration du modèle et lancement de l'entraînement

Une fois le jeu de données préparé, la phase suivante a consisté à configurer le modèle de détection avant de lancer les premières expérimentations d'entraînement. Le modèle YOLOv8n, proposé par la bibliothèque Ultralytics, a été retenu en raison de son architecture légère et de sa rapidité d'exécution, deux caractéristiques particulièrement adaptées à une future implémentation sur un système embarqué tel que le Raspberry Pi 5.

La configuration du modèle repose principalement sur le fichier data.yaml, qui permet d'indiquer l'emplacement des ensembles d'entraînement, de validation et de test, ainsi que le nombre de classes à détecter. Ce fichier est automatiquement lu par la bibliothèque Ultralytics au moment du lancement de l'entraînement et constitue un élément indispensable au bon déroulement de l'apprentissage.

Après la configuration du modèle, une première phase d'entraînement a été lancée afin de vérifier le bon fonctionnement de l'ensemble du processus. Cette étape a permis de contrôler le chargement des données, la lecture des annotations, l'initialisation du modèle ainsi que la génération automatique des fichiers produits par Ultralytics, notamment les poids du modèle et les courbes de suivi de l'entraînement.

Les principaux paramètres d'apprentissage, tels que le nombre d'epoch, la taille des images d'entrée, la taille des lots ainsi que le taux d'apprentissage, ont été définis avant l'exécution de l'entraînement. Les valeurs retenues seront présentées et justifiées de manière détaillée dans le chapitre suivant, consacré à l'entraînement complet du modèle sur le jeu de données GTSDB.

À l'issue de cette étape, le framework Ultralytics génère automatiquement plusieurs fichiers permettant de suivre l'évolution de l'apprentissage, parmi lesquels les poids du meilleur modèle (best.pt), les poids de la dernière époque (last.pt) ainsi que les courbes représentant l'évolution des principales métriques d'entraînement.

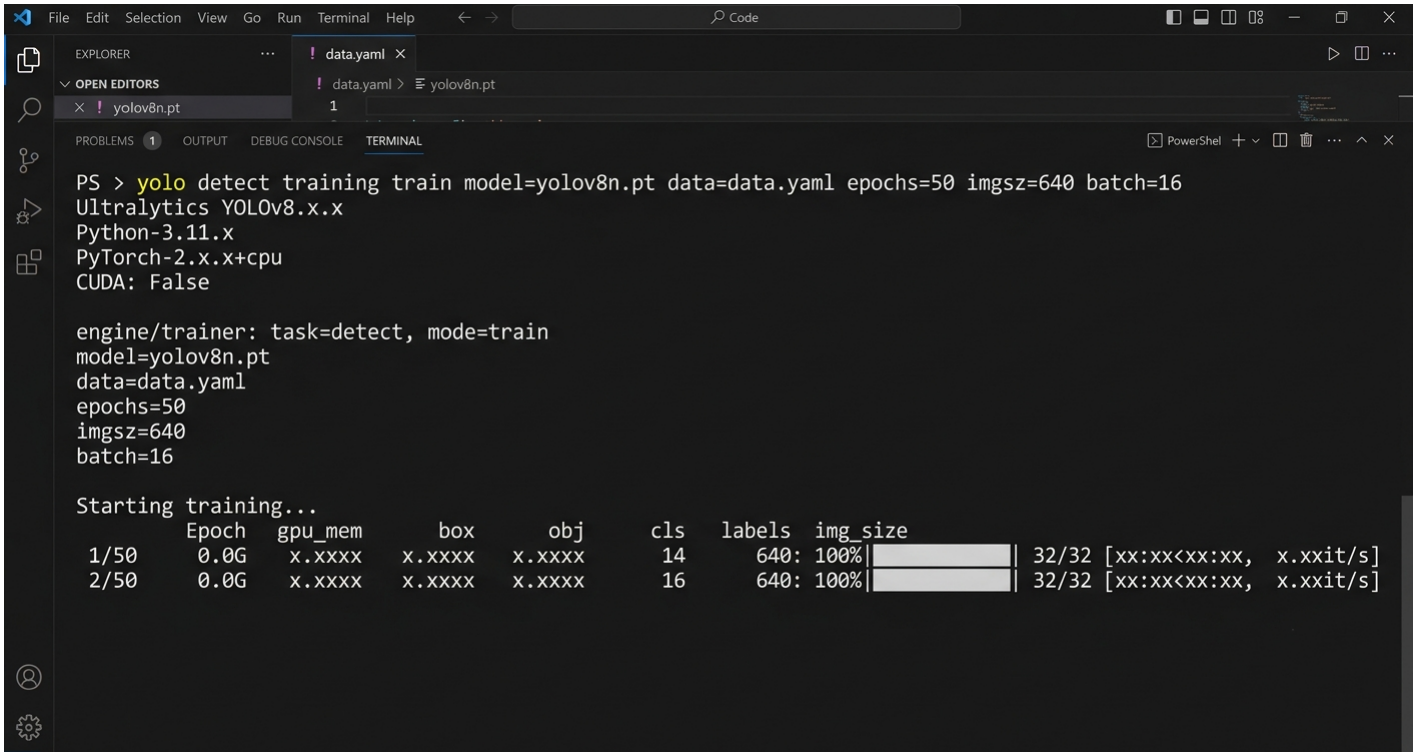


FIGURE 4.3 – Lancement de l'entraînement du modèle YOLOv8n.

4.2.4 Validation du modèle de détection

À l'issue de l'entraînement, une première phase de validation a été réalisée afin d'évaluer le comportement du modèle de détection sur des images n'ayant pas participé à l'apprentissage. Cette étape consiste à appliquer le modèle YOLOv8n sur des images de test afin de vérifier sa capacité à localiser correctement les panneaux de signalisation.

Les résultats obtenus montrent que le modèle est capable d'identifier les panneaux présents dans les scènes routières en générant automatiquement une boîte englobante (bounding box) autour de chaque objet détecté. Pour chaque détection, un score de confiance est également calculé, permettant d'estimer la probabilité que l'objet détecté corresponde effectivement à un panneau de signalisation.

En parallèle des résultats visuels, la bibliothèque Ultralytics génère automatiquement plusieurs indicateurs permettant de suivre le déroulement de l'entraînement. Parmi ceux-ci figurent notamment les courbes d'évolution des fonctions de perte (box loss, classification loss et distribution focal loss) ainsi que les métriques de performance telles que la précision (Precision), le rappel (Recall) et le mAP (mean Average Precision). Ces indicateurs permettent de vérifier que le processus d'apprentissage s'est déroulé correctement et que le modèle converge progressivement au cours des différentes époques.

Les premières expérimentations ont permis de confirmer le bon fonctionnement du module de détection ainsi que la cohérence des résultats obtenus. Les performances détaillées du modèle,

ainsi que l'analyse complète des métriques de détection, seront présentées dans le chapitre 5, consacré à l'entraînement définitif du détecteur sur le jeu de données GTSDB.



FIGURE 4.4 – Exemple de détection des panneaux de signalisation réalisée par le modèle YOLOv8n après entraînement.

Source : German Traffic Sign Detection Benchmark (GTSDB).

La figure 4.5 illustre les principales courbes de performance générées automatiquement par Ultralytics au cours de l'entraînement du modèle.

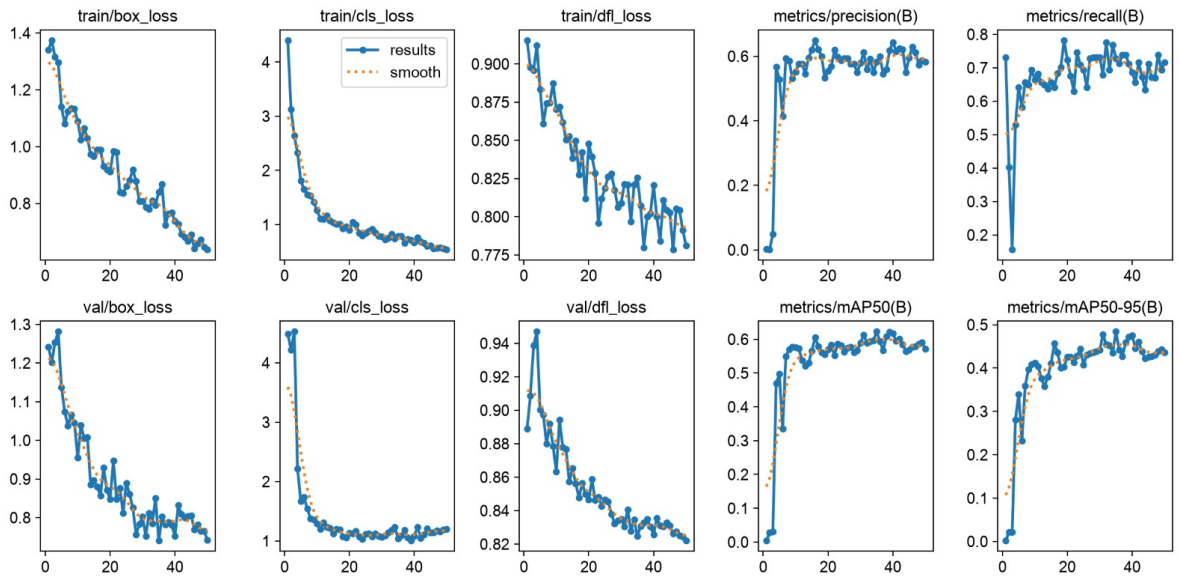


FIGURE 4.5 – Évolution des principales métriques d'entraînement du modèle YOLOv8n.

4.2.5 Notions importantes

4.2.5.1 Distinction entre *results* et *smooth*

Lors du suivi de l'entraînement d'un modèle, deux types de courbes apparaissent généralement sur les graphiques :

- **La courbe *results* (données brutes)** correspond aux valeurs mesurées directement à chaque *epoch*, sans aucun traitement. Elle reflète fidèlement les variations réelles de la perte ou des métriques, ce qui la rend parfois irrégulière. Ces oscillations sont normales : elles proviennent du fait que chaque lot de données (*batch*) traité par le modèle diffère légèrement, empêchant une progression parfaitement linéaire.
- **La courbe *smooth* (données lissées)** est obtenue en appliquant un traitement statistique (par exemple une moyenne glissante ou un lissage exponentiel) sur la courbe brute. L'objectif est de supprimer le bruit visuel afin de ne conserver que la tendance générale, facilitant ainsi l'interprétation de l'évolution globale du modèle.

En pratique, la courbe brute permet de vérifier le comportement exact du modèle à chaque *epoch*, tandis que la version lissée offre une vision plus synthétique de son évolution, particulièrement utile lorsque les fluctuations rendent l'analyse plus difficile.

4.2.5.2 Les métriques d'évaluation

Une métrique est un indicateur chiffré permettant d'évaluer la qualité des prédictions d'un modèle. Elle sert notamment à comparer plusieurs modèles entre eux ou à ajuster leurs hyperparamètres.

- **Précision (Precision)** : correspond à la proportion de détections positives qui sont réellement correctes. Une précision élevée traduit un faible nombre de faux positifs, ce qui est particulièrement important lorsque les fausses alarmes doivent être limitées.
- **Rappel (Recall)** : représente la proportion des objets réellement présents que le modèle parvient à détecter. Un rappel élevé indique un faible nombre de faux négatifs.
- **F1-score** : combine la précision et le rappel à l'aide de leur moyenne harmonique. Il fournit un indicateur unique qui pénalise les modèles présentant un déséquilibre entre ces deux métriques.
- **mAP (Mean Average Precision)** : constitue la métrique de référence en détection d'objets. Elle repose sur l'IoU (*Intersection over Union*), qui mesure le degré de recouvrement entre une boîte prédite et la boîte de vérité terrain (1 correspondant à une correspondance parfaite et 0 à une absence totale de recouvrement).
    - **mAP@50** : précision moyenne calculée avec un seuil d'IoU fixé à 50 %.
    - **mAP@50–95** : précision moyenne calculée sur plusieurs seuils d'IoU compris entre 50 % et 95 %, offrant une évaluation plus rigoureuse des performances du modèle.
- **Perte (Loss)** : mesure l'écart entre les prédictions du modèle et la vérité terrain durant l'entraînement. En détection d'objets, elle est généralement composée de trois éléments :
    - **Box Loss** : évalue la précision de la localisation des boîtes englobantes.
    - **Classification Loss (Cls Loss)** : mesure la qualité de la classification des objets détectés.

- **Distribution Focal Loss (DFL Loss)** : améliore la précision de la localisation en affinant l'estimation des frontières des boîtes englobantes.

4.3 Développement et validation du module de classification

Après avoir validé le fonctionnement du module de détection, une seconde phase d'expérimentation a été consacrée au module de classification. Cette étape vise à vérifier que le réseau de neurones MobileNetV2 est capable d'identifier correctement la catégorie de chaque panneau détecté avant son intégration au pipeline complet de reconnaissance.

Contrairement au module de détection, dont l'objectif est de localiser les panneaux de signalisation dans une image, le module de classification intervient uniquement sur les ROI extraites par YOLOv8n. Chaque panneau détecté est ainsi recadré, prétraité puis transmis au réseau MobileNetV2 afin d'en déterminer la classe correspondante.

Comme pour la phase de détection, une validation préliminaire a été réalisée afin de vérifier le bon déroulement des différentes étapes du processus de classification avant l'entraînement définitif sur le jeu de données GTSRB.

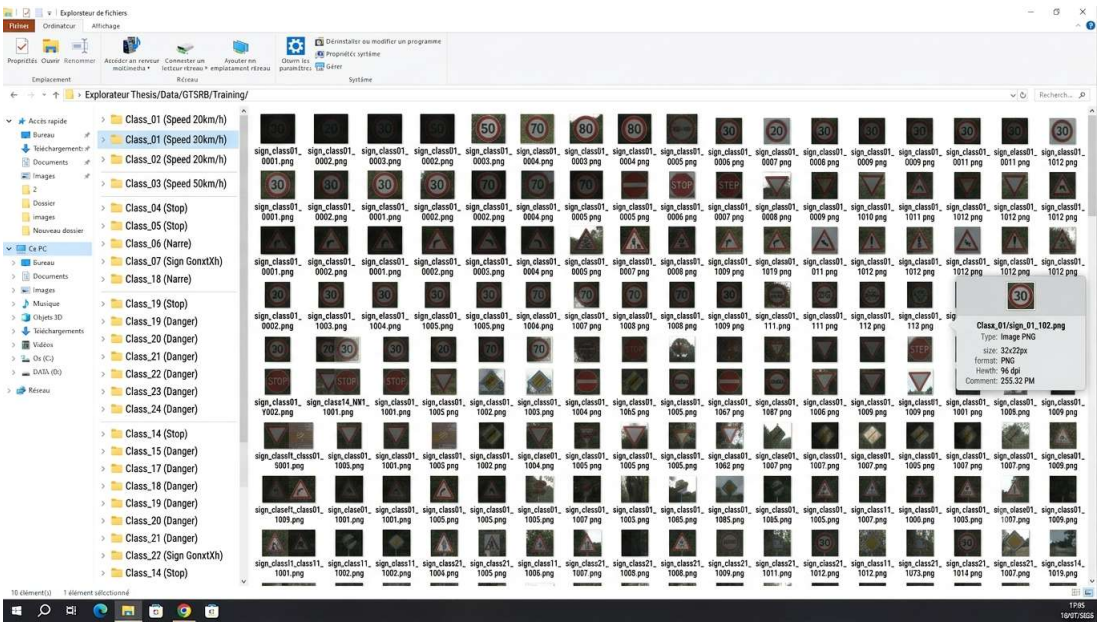


FIGURE 4.6 – Organisation du jeu de données GTSRB utilisée pour l'entraînement du modèle MobileNetV2.

4.3.1 Préparation du jeu de données GTSRB

Le jeu de données German Traffic Sign Recognition Benchmark (GTSRB) a été retenu pour l'apprentissage du modèle de classification. Contrairement au jeu de données GTSDB, qui contient des scènes routières complètes, GTSRB est constitué d'images déjà centrées sur les panneaux de signalisation. Chaque image appartient à une classe spécifique représentant un type de panneau.

Avant l'entraînement, les données ont été organisées selon une arborescence adaptée aux modèles de classification, dans laquelle chaque dossier correspond à une classe de panneaux. Cette organisation permet au framework d'identifier automatiquement les différentes catégories et de charger les images correspondantes lors de l'apprentissage.

Afin d'assurer une homogénéité des données, toutes les images ont été redimensionnées à la taille attendue par MobileNetV2 puis normalisées avant leur utilisation. Ces opérations garantissent une représentation uniforme des données d'entrée et contribuent à la stabilité de l'apprentissage.

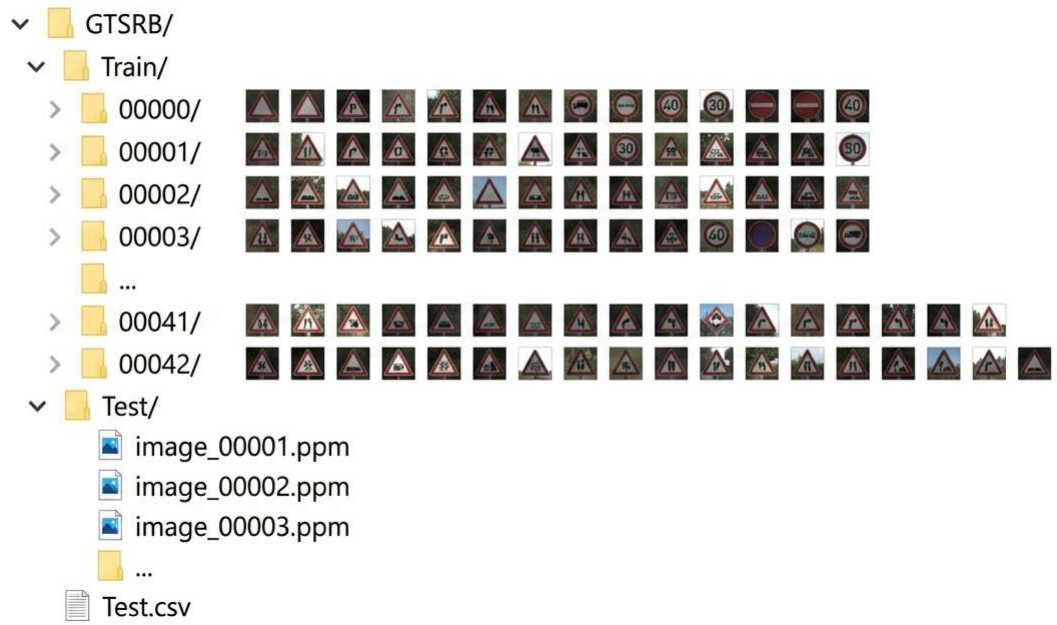


FIGURE 4.7 – Arborescence du jeu de données GTSRB utilisée pour l'entraînement du modèle de classification.

4.3.2 Configuration et entraînement du modèle MobileNetV2

Après la préparation du jeu de données GTSRB, le modèle MobileNetV2 a été configuré afin de réaliser les premières expérimentations de classification des panneaux de signalisation. Ce modèle a été retenu en raison de sa légèreté, de sa rapidité d'exécution et de ses bonnes performances pour les tâches de classification d'images.

L'entraînement a été effectué à partir du jeu de données GTSRB, dont les images ont été préalablement redimensionnées et prétraitées afin de répondre aux exigences du réseau de neurones. Les principaux paramètres d'entraînement ont ensuite été définis avant le lancement de l'apprentissage.

Au cours de cette phase, le modèle a progressivement appris à reconnaître les différentes catégories de panneaux de signalisation présentes dans le jeu de données. Les poids obtenus à l'issue de l'entraînement ont été sauvegardés afin d'être utilisés dans les étapes suivantes du projet, notamment lors de l'intégration du module de classification avec le détecteur YOLOv8n.

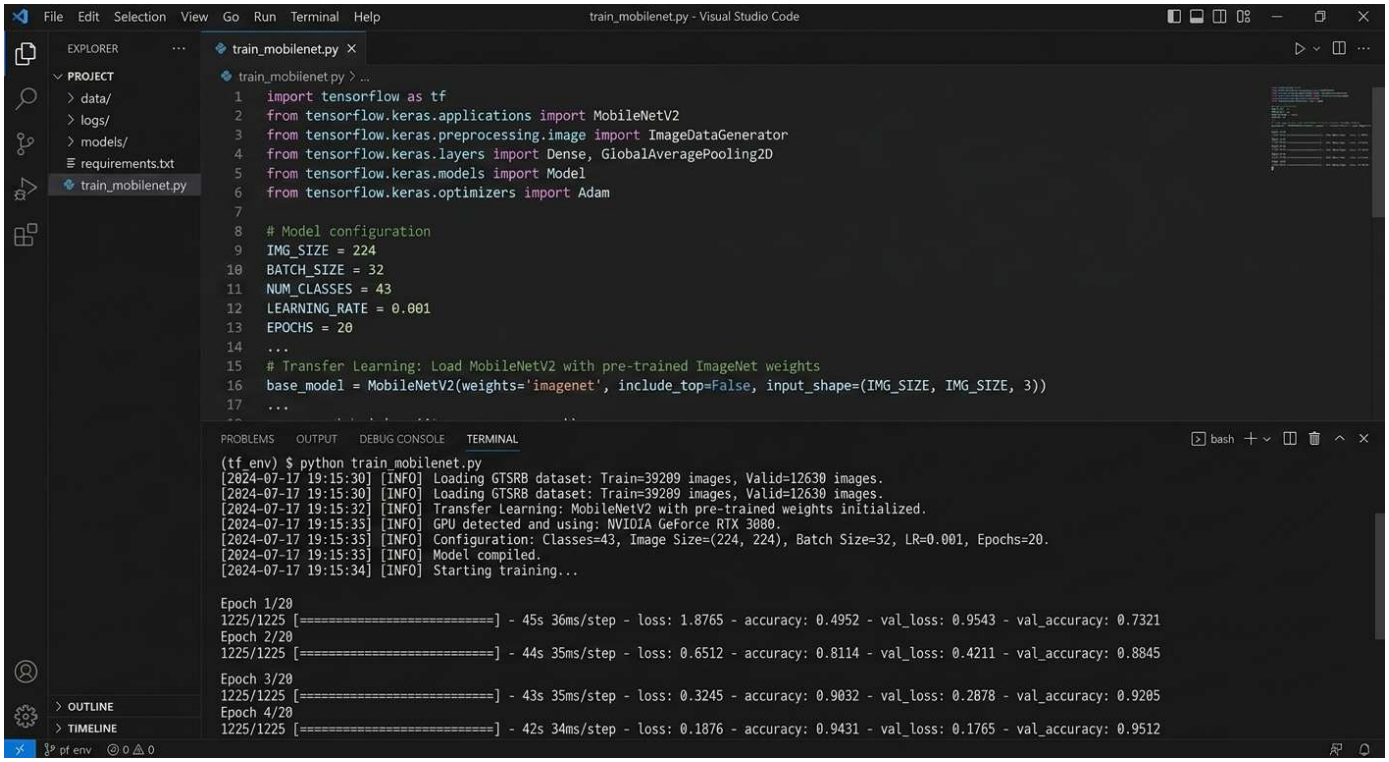


FIGURE 4.8 – Lancement de l'entraînement du modèle MobileNetV2 sur le jeu de données GTSRB.

4.3.3 Validation du modèle de classification

À l'issue de l'entraînement, une première phase de validation a été réalisée afin d'évaluer le comportement du modèle de classification sur des images n'ayant pas participé à l'apprentissage. Cette étape consiste à présenter au réseau des images de panneaux de signalisation et à comparer la classe prédite avec la classe réelle.

Les premiers résultats obtenus montrent que le modèle est capable d'identifier correctement les différentes catégories de panneaux présentes dans le jeu de données. Les prédictions générées confirment le bon fonctionnement de la chaîne de classification ainsi que la capacité du réseau MobileNetV2 à extraire les caractéristiques visuelles nécessaires à la reconnaissance des panneaux.

En complément des prédictions individuelles, les principales métriques d'apprentissage, telles que l'évolution de la précision (Accuracy) et de la fonction de perte (Loss), ont été analysées afin de vérifier la stabilité du processus d'entraînement. L'étude détaillée des performances du modèle ainsi que la comparaison des résultats obtenus seront présentées dans le chapitre suivant, consacré à l'entraînement complet du classificateur sur le jeu de données GTSRB.

Cette phase de validation a permis de confirmer le bon fonctionnement du module de classification avant son intégration au pipeline complet de reconnaissance des panneaux de signalisation.

4.4 Conclusion

Ce chapitre a permis de présenter les différentes étapes de développement ainsi que la validation préliminaire des deux principaux modules constituant le système proposé. Dans un premier temps, l'environnement de développement a été mis en place et configuré afin de garantir un cadre de travail stable et reproductible. Les bibliothèques nécessaires au développement ont été installées et leur bon fonctionnement a été vérifié avant le lancement des expérimentations.

La validation du module de détection a ensuite permis de confirmer le bon fonctionnement du modèle YOLOv8n, depuis la préparation des données jusqu'à la génération des premières détections. De la même manière, la validation du module de classification a permis de vérifier le comportement du modèle MobileNetV2 et sa capacité à reconnaître les différentes catégories de panneaux de signalisation.

Les résultats obtenus au cours de cette phase montrent que l'ensemble du pipeline fonctionne correctement et que les différents modules interagissent de manière cohérente. Cette étape constitue une validation technique indispensable avant de réaliser les entraînements complets des modèles.

Le chapitre suivant sera consacré à l'entraînement approfondi des modèles sur les jeux de données de référence GTSDB et GTSRB. Une phase de fine-tuning sera ensuite réalisée à l'aide d'un jeu de données représentatif du contexte routier algérien afin d'adapter les modèles aux spécificités de la signalisation nationale et d'améliorer leurs performances dans les conditions réelles d'utilisation.

Chapitre 5

Adaptation du système au contexte algérien et évaluation des performances

5.1 Introduction

Le chapitre précédent a permis de présenter le développement du système proposé ainsi que la validation préliminaire des modules de détection et de classification. Les premières expérimentations réalisées sur les jeux de données de référence ont confirmé le bon fonctionnement des différentes étapes du pipeline, depuis la détection des panneaux de signalisation jusqu'à leur classification.

Le présent chapitre est consacré à l'adaptation du système au contexte routier algérien ainsi qu'à l'analyse des performances obtenues. Bien que les jeux de données GTSDB et GTSRB constituent des références largement utilisées pour l'entraînement des modèles de détection et de classification, ils sont principalement composés de panneaux de signalisation allemands et ne représentent pas entièrement les caractéristiques des panneaux rencontrés sur le réseau routier algérien.

Afin d'améliorer la capacité de généralisation du système dans son environnement d'utilisation, une étape de fine-tuning [1] a été réalisée à l'aide de notre propre jeu de données contenant des panneaux de signalisation algériens. Cette adaptation permet au détecteur YOLOv8n et au classificateur MobileNetV2 de mieux prendre en compte les spécificités locales, notamment les différences liées à l'apparence des panneaux, aux conditions d'éclairage, aux angles de prise de vue et à l'environnement routier.

Ce chapitre présente tout d'abord notre propre jeu de données utilisé pour cette adaptation, puis décrit le processus de fine-tuning appliqué aux deux modèles. Enfin, les performances obtenues sont analysées afin d'évaluer l'apport de cette adaptation sur la qualité de la détection et de la classification des panneaux de signalisation algériens.

1. Le fine-tuning (ou ajustement fin) est une technique d'apprentissage par transfert consistant à poursuivre l'entraînement d'un modèle pré-entraîné sur un nouveau jeu de données afin de l'adapter à une tâche spécifique, généralement en utilisant un faible taux d'apprentissage pour préserver les connaissances déjà acquises.

5.2 Adaptation du système au contexte algérien

5.2.1 Présentation du jeu de données algérien

Afin d'adapter le système de détection et de reconnaissance des panneaux de signalisation au contexte routier algérien, un jeu de données spécifique a été conçu dans le cadre de ce projet. Contrairement aux bases de données internationales telles que GTSDB et GTSRB, qui reposent principalement sur des panneaux provenant d'autres environnements routiers, l'objectif de ce travail était de constituer une base de données davantage représentative des conditions rencontrées sur les routes algériennes.

La conception de ce jeu de données a constitué une étape importante du projet, notamment en raison de l'absence d'une base de données algérienne suffisamment adaptée aux besoins spécifiques du système développé. Les images collectées sur le terrain ont été prises dans la ville de Mostaganem à l'aide de smartphones, dans différents endroits et environnements routiers afin d'obtenir une diversité suffisante en termes de conditions de prise de vue, d'environnement routier, d'éclairage et de positionnement des panneaux. Cette diversité vise à améliorer la capacité de généralisation des modèles lors de leur utilisation dans des conditions réelles.

Le jeu de données conçu comprend **48 classes de panneaux de signalisation**, réparties selon quatre grandes catégories :

- **Att** : panneaux d'avertissement ou de danger ;
- **Inter** : panneaux d'interdiction ;
- **Oblig** : panneaux d'obligation ;
- **Indic** : panneaux d'indication.

Les différentes catégories ont été organisées afin de couvrir les principaux types de panneaux de signalisation nécessaires au fonctionnement du système de reconnaissance développé dans le cadre de ce projet. Cette organisation facilite également la préparation des données et l'analyse des résultats obtenus par les modèles.

Pour préparer les données à l'entraînement du modèle de détection (YOLOv8), nous avons effectué un processus de classification manuel et semi-automatisé sur la plateforme Roboflow. Ce processus consistait à placer chaque panneau de signalisation dans un cadre et à le classer dans la catégorie appropriée. Ces classifications ont ensuite été exportées dans un format compatible avec le modèle YOLOv8n.

Afin de répondre aux besoins des deux phases du système proposé, l'ensemble de données a été organisé en deux ensembles complémentaires. Le premier ensemble est dédié à la détection des panneaux de signalisation et contient 609 images accompagnées de 609 fichiers de classification, répartis entre les phases d'entraînement, de vérification et de test. Ces données permettent au modèle YOLOv8n d'apprendre à localiser les panneaux sur la route.

Le second ensemble est dédié à la classification. Après la détection, les zones représentant les panneaux sont extraites des images par recadrage. Outre le processus d'amélioration automatisé via la plateforme Roboflow (rotation, ajustement de la luminosité, zoom, etc.) qui garantit la préservation des informations essentielles de l'image, ces images constituent les données utilisées pour entraîner et adapter le modèle MobileNetV2. L'ensemble de données de classification comprend 2 081 images réparties entre les ensembles d'entraînement, de validation et de test.

Cette séparation permet ainsi de respecter l'architecture en deux étapes adoptée dans ce projet :

le modèle YOLOv8n assure d'abord la **localisation** du panneau dans la scène, tandis que MobileNetV2 assure ensuite son **identification parmi les 48 classes**. Les deux modèles utilisent donc des données préparées spécifiquement pour leur fonction respective.

Pour garantir une évaluation objective, les données ont été réparties en trois sous-ensembles : un ensemble d'entraînement (*Training*), utilisé pour l'apprentissage des modèles, un ensemble de validation (*Validation*), utilisé pour suivre leur évolution au cours de l'entraînement et ajuster les paramètres, et un ensemble de test (*Test*), conservé pour l'évaluation finale sur des données non utilisées lors de l'apprentissage.

La préparation du jeu de données constitue ainsi une étape essentielle de l'adaptation du système au contexte algérien. En combinant des images issues de scènes routières variées, une annotation adaptée à la détection et une préparation spécifique des images destinées à la classification, le jeu de données développé fournit une base commune aux deux modèles constituant la chaîne de reconnaissance proposée.

Afin d'illustrer le contenu du jeu de données utilisé, la Figure 5.1 présente quelques exemples d'images annotées sur la plateforme Roboflow Universe.

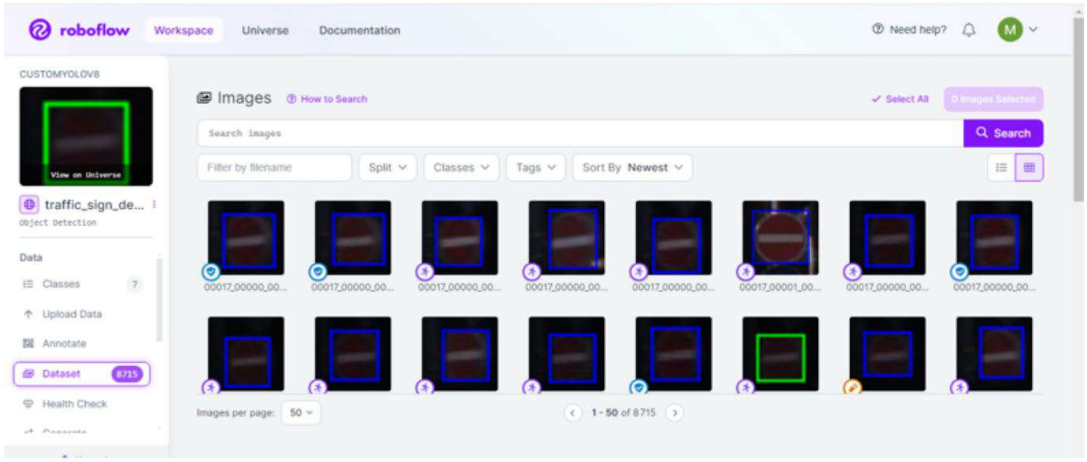


FIGURE 5.1 – Exemples d'images annotées du jeu de données sur Roboflow.

5.2.2 Adaptation du détecteur YOLOv8n

Après l'entraînement initial du modèle YOLOv8n sur le jeu de données GTSDB, une étape de fine-tuning a été réalisée afin d'améliorer ses performances dans un contexte proche des conditions de circulation algériennes. Cette approche permet de conserver les connaissances acquises lors du premier entraînement tout en adaptant progressivement le modèle aux caractéristiques des panneaux de signalisation présents sur le réseau routier algérien.

Pour cette phase d'adaptation, nous avons importé le jeu de données sur la plateforme Roboflow afin de procéder à sa préparation et à son utilisation pour l'entraînement du modèle au format YOLOv8n, directement compatible avec le framework Ultralytics. Avant le lancement du fine-tuning, les images ont été soumises à un prétraitement automatique comprenant la correction de leur orientation (Auto-Orient) ainsi qu'un redimensionnement uniforme à une résolution de 640×640 pixels, correspondant à la taille d'entrée utilisée par le modèle YOLOv8n. Aucune augmentation artificielle des données n'a été appliquée lors de cette version du jeu de données.

Le jeu de données a ensuite été réparti en trois ensembles distincts afin de garantir un apprentis-

sage fiable et une évaluation objective des performances du modèle. L'ensemble d'entraînement a été utilisé pour ajuster les paramètres du réseau, tandis que l'ensemble de validation a permis de suivre les performances du modèle au cours du fine-tuning et d'ajuster les différents paramètres du processus d'apprentissage. Enfin, l'ensemble de test, conservé à l'écart durant l'entraînement, a été utilisé pour évaluer les performances finales et la capacité de généralisation du détecteur sur des images inédites.

Le Tableau 5.1 présente la répartition des images utilisées lors de cette étape d'adaptation.

Ensemble	**Nombre d'images**	**Nombre de labels**
Entraînement (Training)	426	426
Validation	91	91
Test	92	92
Total	**609**	**609**

TABLE 5.1 – Répartition du jeu de données utilisé pour le fine-tuning du détecteur YOLOv8n

Note explicative : L'analyse du tableau ci-dessus révèle une stricte égalité entre le nombre total d'images et le nombre de labels (soit un ratio unitaire de 1 : 1 pour chaque sous-ensemble). Ce choix méthodologique s'explique par la nature de la constitution de notre propre jeu de données de terrain : chaque image capturée a été spécifiquement cadrée, sélectionnée ou découpée (*croppée*) autour d'une unique instance de panneau de signalisation.

Contrairement à des bases de données globales de scènes routières denses où une seule image peut regrouper de multiples objets hétérogènes, notre approche garantit que chaque échantillon d'apprentissage correspond à un cas cible unique et isolé. Cette normalisation permet d'éviter l'ambiguïté spatiale des superpositions multiples et simplifie l'évaluation de la performance brute des modèles de détection et de classification sur notre contexte local.

5.2.3 Fine-tuning du détecteur YOLOv8n

À l'issue de l'entraînement initial réalisé sur le jeu de données GTSDB, une phase de fine-tuning a été entreprise afin d'adapter le détecteur YOLOv8n aux caractéristiques des panneaux de signalisation rencontrés sur le réseau routier algérien. Cette approche consiste à poursuivre l'apprentissage d'un modèle déjà entraîné en utilisant un nouveau jeu de données plus représentatif de l'environnement d'utilisation visé.

Le modèle obtenu à partir de GTSDB a servi de point de départ pour cette étape d'adaptation. Les connaissances acquises lors du premier entraînement, notamment la capacité à localiser les panneaux de signalisation, ont ainsi été conservées, tandis que les paramètres du réseau ont été progressivement ajustés afin de mieux prendre en compte les spécificités des panneaux présents dans le jeu de données algérien.

Le fine-tuning du modèle YOLOv8n a été réalisé dans l'environnement Google Colab à l'aide du framework *Ultralytics*. L'entraînement a été effectué sur le jeu de données algérien de détection présenté précédemment. Les paramètres d'entraînement ont été choisis de manière à assurer une convergence stable du modèle tout en limitant les risques de surapprentissage.

À l'issue de cette phase d'adaptation, un nouveau modèle de détection optimisé pour le contexte routier algérien a été obtenu. Ce modèle est ensuite utilisé dans le système complet afin de localiser les panneaux de signalisation avant leur classification par le réseau MobileNetV2.

5.2.4 Résultats du fine-tuning du détecteur YOLOv8n

À l'issue du fine-tuning, les performances du détecteur YOLOv8n ont été évaluées à l'aide des métriques couramment utilisées pour les tâches de détection d'objets. L'objectif de cette évaluation est de vérifier la capacité du modèle à localiser correctement les panneaux de signalisation présents dans les images du jeu de données algérien.

La Figure 5.2 présente l'évolution des métriques du détecteur au cours des 30 époques d'entraînement et de validation. On observe une diminution progressive et continue des trois fonctions de perte de détection (Box Loss, Classification Loss et Distribution Focal Loss) tant en train qu'en val, traduisant une amélioration constante de la localisation des boîtes englobantes et de la classification des objets détectés. Parallèlement, les métriques de performance, notamment la précision (Precision), le rappel (Recall) ainsi que les indicateurs mAP@50 et mAP@50-95, augmentent progressivement avant de se stabiliser en fin d'entraînement. Cette évolution témoigne d'une convergence satisfaisante du modèle sans aucun signe de surapprentissage.

Les performances finales obtenues sur l'ensemble de test sont résumées dans le Tableau 5.2.

Table 5.2 – Performances du détecteur YOLOv8n après fine-tuning

Métrique	**Valeur**	**Valeur (%)**
Precision (P)	0,863	86,3 %
Recall (R)	0,810	81,0 %
mAP@50	0,971	97,1 %
mAP@50–95	0,761	76,1 %

Les résultats montrent que le détecteur est capable de localiser efficacement les panneaux de signalisation dans différentes conditions d'acquisition, ce qui confirme l'intérêt du fine-tuning réalisé à partir du modèle initial entraîné sur GTSDB.

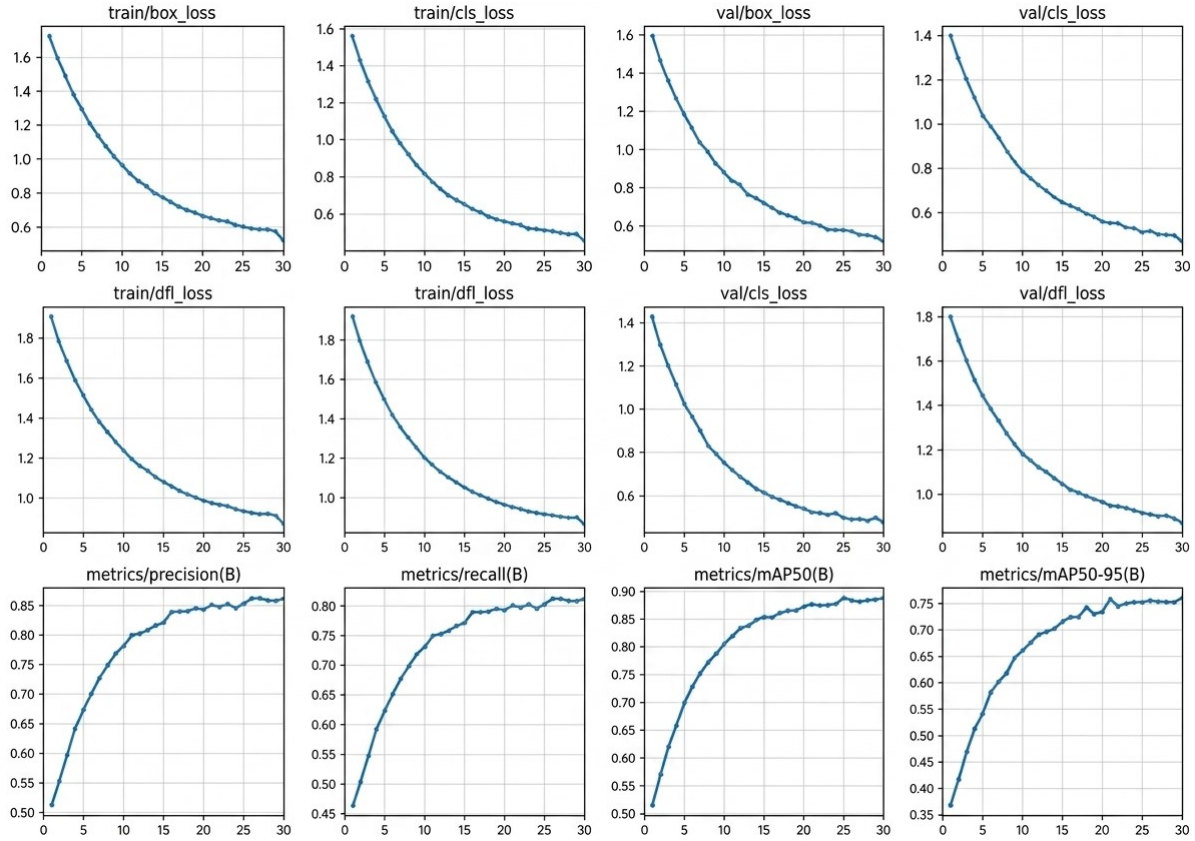


FIGURE 5.2 – Évolution des performances du modèle YOLOv8n au cours du fine-tuning.

5.2.5 Résultats finaux du test de détection

Un test final de détection a été réalisé sur les images contenues dans le dossier **test** du jeu de données de détection. Cette étape permet d'évaluer les performances du détecteur sur des données distinctes de celles utilisées lors de son apprentissage.

L'objectif de cette évaluation est de mesurer la capacité de **YOLOv8n** à localiser correctement les panneaux de signalisation dans des images routières, tout en limitant les fausses détections et les panneaux non détectés. Les résultats sont analysés à travers la matrice de confusion, la matrice de confusion normalisée et différentes courbes permettant d'étudier l'influence du seuil de confiance sur les performances du modèle.

5.2.5.1 Matrice de confusion

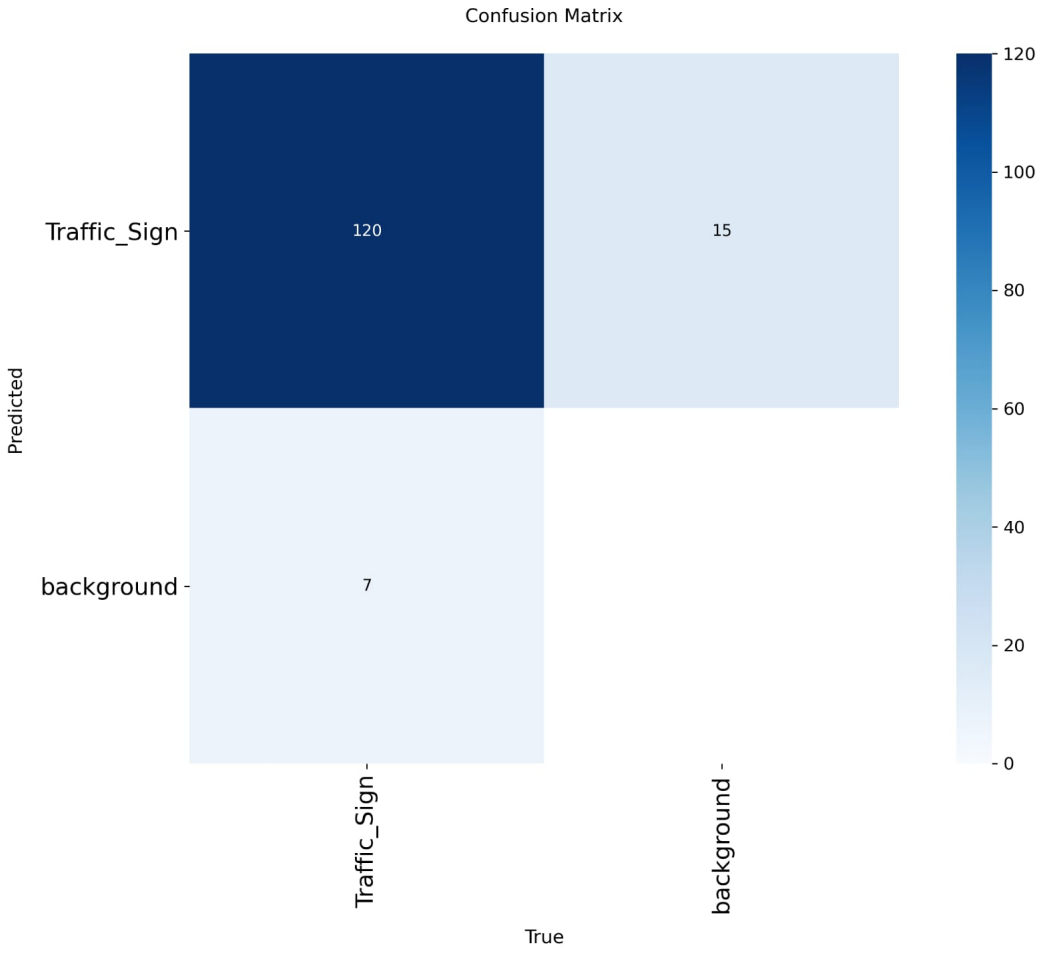


FIGURE 5.3 – Matrice de confusion du modèle **YOLOv8n** obtenue sur l'ensemble de test

La matrice de confusion obtenue lors du test final permet d'analyser les différentes catégories de prédictions produites par le modèle. Les valeurs brutes obtenues sont présentées ci-dessous :

- **Vrais positifs (TP = 120)** : le modèle a correctement détecté 120 panneaux de signalisation présents dans les images du dossier **test**.
- **Faux positifs (FP = 15)** : le modèle a généré 15 détections dans des zones ne contenant pas réellement de panneau de signalisation.
- **Faux négatifs (FN = 7)** : 7 panneaux présents dans les images du dossier **test** n'ont pas été détectés par le modèle.
- **Vrais négatifs (TN = 0)** : cette valeur est liée à la nature de la détection d'objets. Contrairement à une classification binaire classique, l'arrière-plan n'est pas considéré comme une classe indépendante faisant l'objet d'une prédiction explicite.

Ainsi, la majorité des panneaux présents dans les images du dossier **test** ont été correctement détectés. Le nombre relativement limité de faux positifs et de faux négatifs montre également que le modèle parvient à distinguer efficacement les panneaux de signalisation de l'arrière-plan.

5.2.5.2 Matrice de confusion normalisée

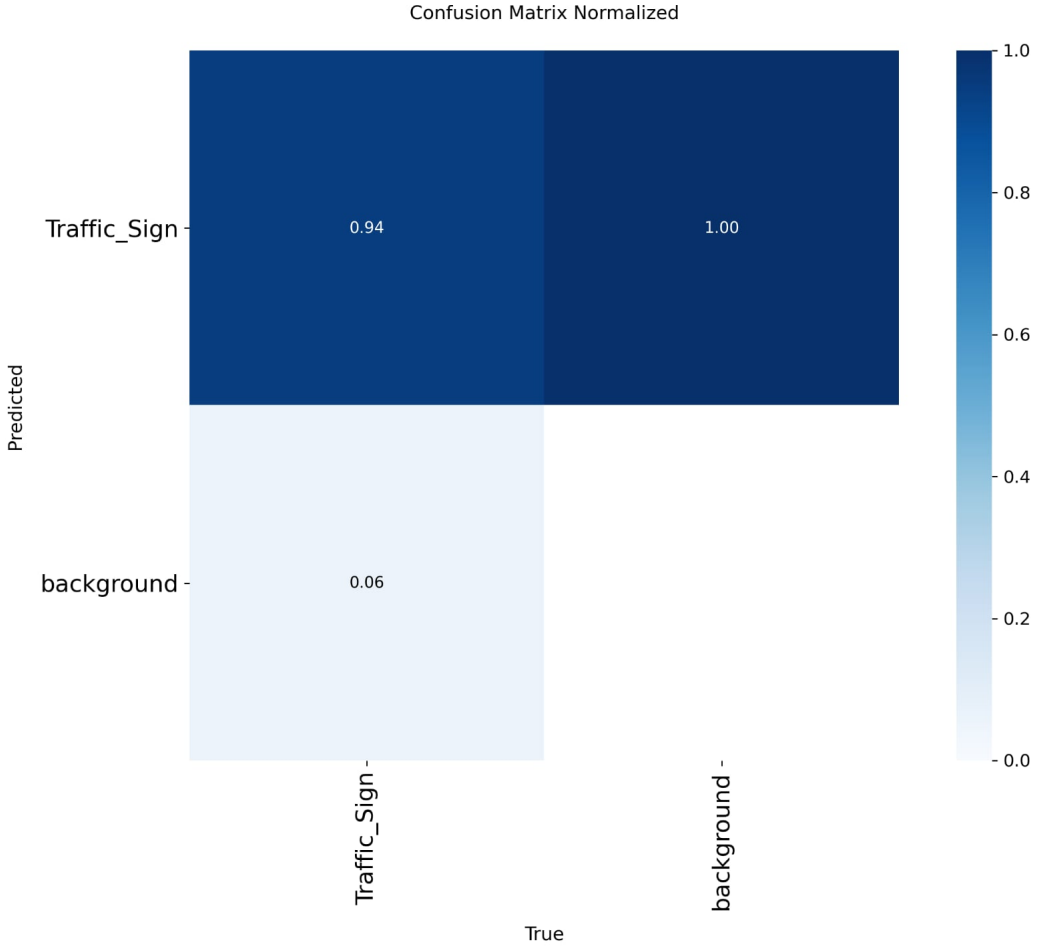


FIGURE 5.4 – Matrice de confusion normalisée du modèle **YOLOv8n** obtenue sur l'ensemble de test

La matrice de confusion normalisée permet d'exprimer les résultats sous forme de proportions afin de faciliter leur interprétation.

Les résultats indiquent un taux de détection d'environ **94%** des panneaux présents dans le dossier **test**. Autrement dit, environ 94% des objets réels appartenant à la classe *Traffic Sign* ont été correctement détectés, tandis qu'environ 6% n'ont pas été détectés.

Cette représentation permet de mettre en évidence la capacité du modèle à retrouver les panneaux présents dans les images tout en donnant une vision proportionnelle des erreurs de détection.

5.2.5.3 Courbe Recall–Confidence

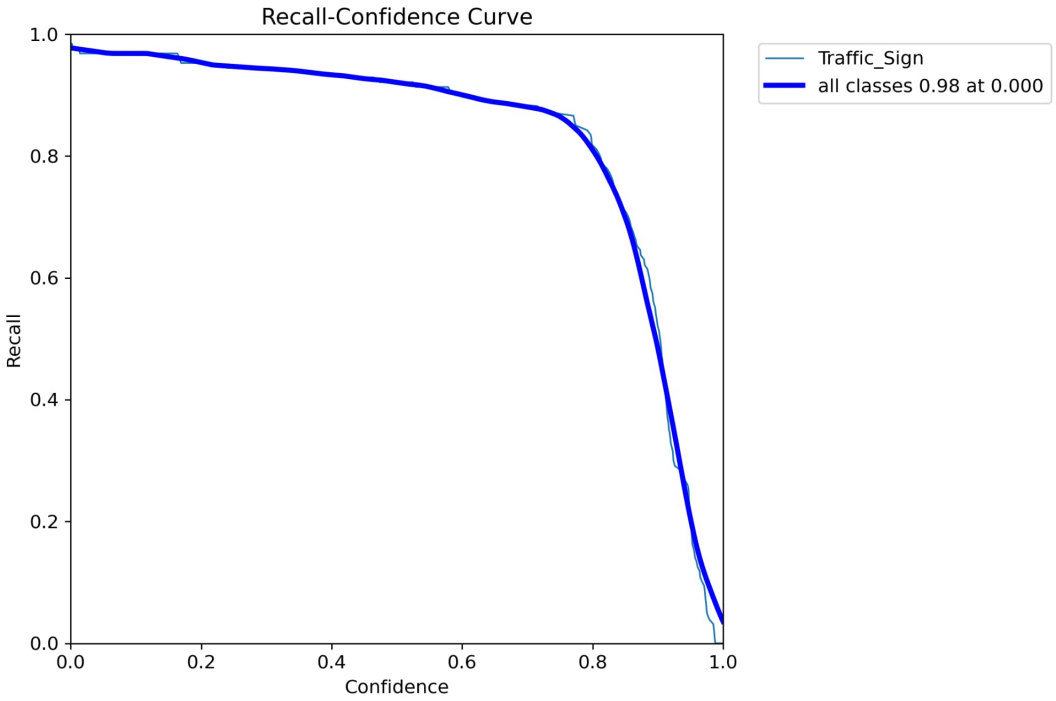


FIGURE 5.5 – Courbe Recall–Confidence du modèle **YOLOv8n** obtenue sur l'ensemble de test

La courbe *Recall–Confidence* représente l'évolution du rappel en fonction du seuil de confiance appliqué aux prédictions du modèle.

Le rappel maximal atteint environ **0,98** lorsque le seuil de confiance est fixé à **0,000**. Dans cette configuration, le modèle conserve pratiquement toutes les prédictions possibles, ce qui lui permet de retrouver jusqu'à 98 % des panneaux présents dans les données de test.

Lorsque le seuil de confiance augmente, certaines prédictions présentant une confiance plus faible sont progressivement éliminées. Le rappel diminue alors progressivement. Une diminution plus marquée est observée pour les seuils élevés, notamment à partir d'environ **0,75**.

Cette courbe montre donc le compromis existant entre la conservation d'un nombre important de détections et l'augmentation du niveau de confiance exigé pour accepter une prédiction.

5.2.5.4 Courbe Precision–Recall

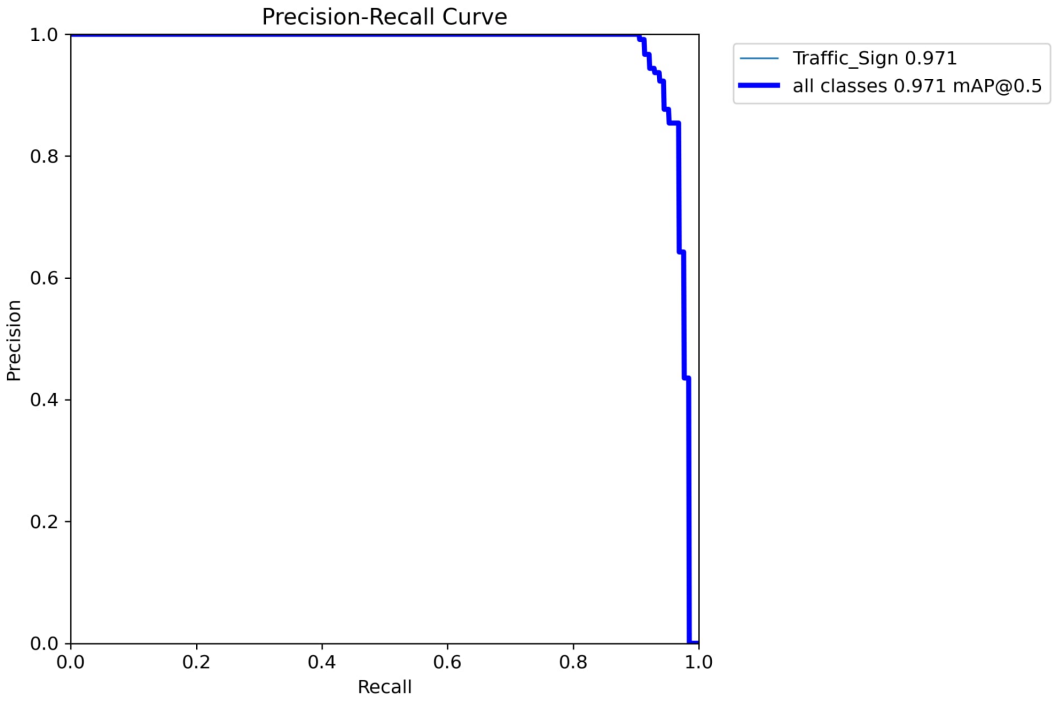


FIGURE 5.6 – Courbe Precision–Recall du modèle **YOLOv8n** obtenue sur l'ensemble de test

La courbe *Precision–Recall* permet d'étudier le compromis entre la précision des prédictions et le rappel du modèle.

L'évaluation finale réalisée sur le dossier **test** donne un **mAP@0.5 de 0,971**, soit **97,1%**. Cette métrique mesure la performance moyenne de détection lorsque le seuil d'Intersection over Union (IoU) considéré est égal à 0,5.

La courbe montre que la précision reste proche de 1,0 pour une large plage de valeurs du rappel, jusqu'à environ 0,90. Une diminution de la précision apparaît ensuite lorsque le modèle cherche à récupérer les derniers objets, correspondant généralement aux panneaux les plus difficiles à détecter.

Le **mAP@0.5 de 97,1%** obtenu sur le dossier **test** constitue ainsi un indicateur important de la qualité de localisation et de détection du modèle **YOLOv8n**.

5.2.5.5 Courbe Precision–Confidence

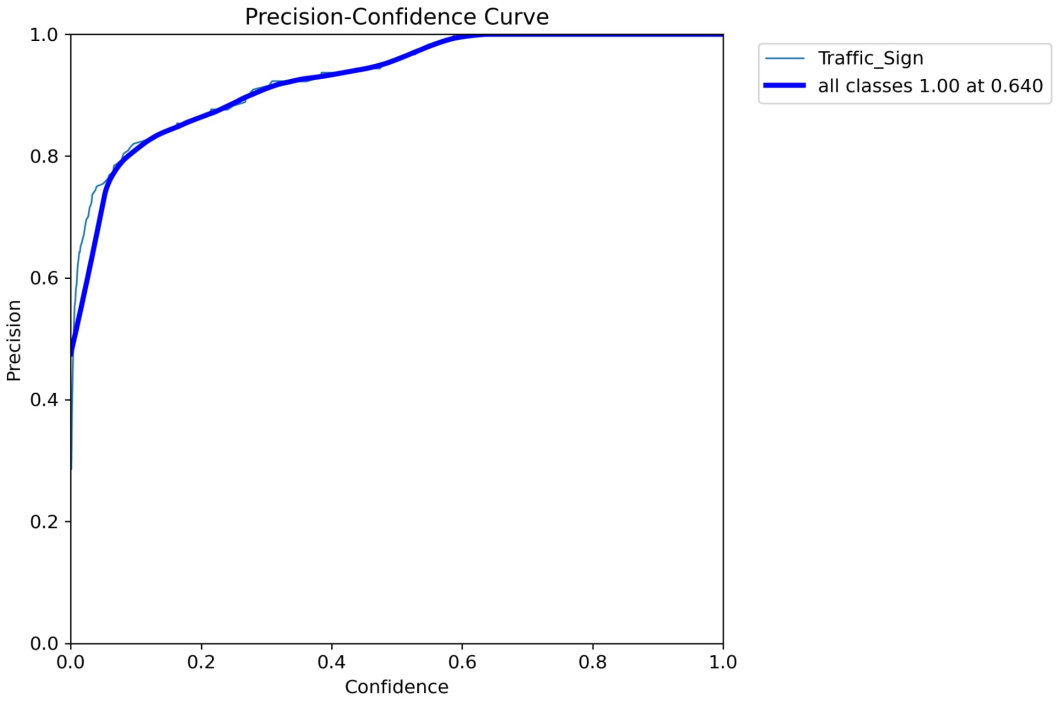


FIGURE 5.7 – Courbe Precision–Confidence du modèle **YOLOv8n** obtenue sur l'ensemble de test

La courbe *Precision–Confidence* représente l'évolution de la précision en fonction du seuil de confiance appliqué aux prédictions.

La précision atteint une valeur de **1,00** pour un seuil de confiance d'environ **0,640**. Dans l'évaluation réalisée sur le dossier **test**, cela signifie qu'à ce niveau de confiance, les prédictions conservées ne présentent pas de faux positifs dans les résultats considérés.

Cependant, cette valeur doit être interprétée conjointement avec le rappel. En effet, l'augmentation du seuil de confiance permet de réduire les fausses détections, mais peut également supprimer certaines détections correctes présentant une confiance inférieure.

La précision maximale ne constitue donc pas, à elle seule, le meilleur réglage pour le fonctionnement global du système.

5.2.5.6 Courbe F1–Confidence

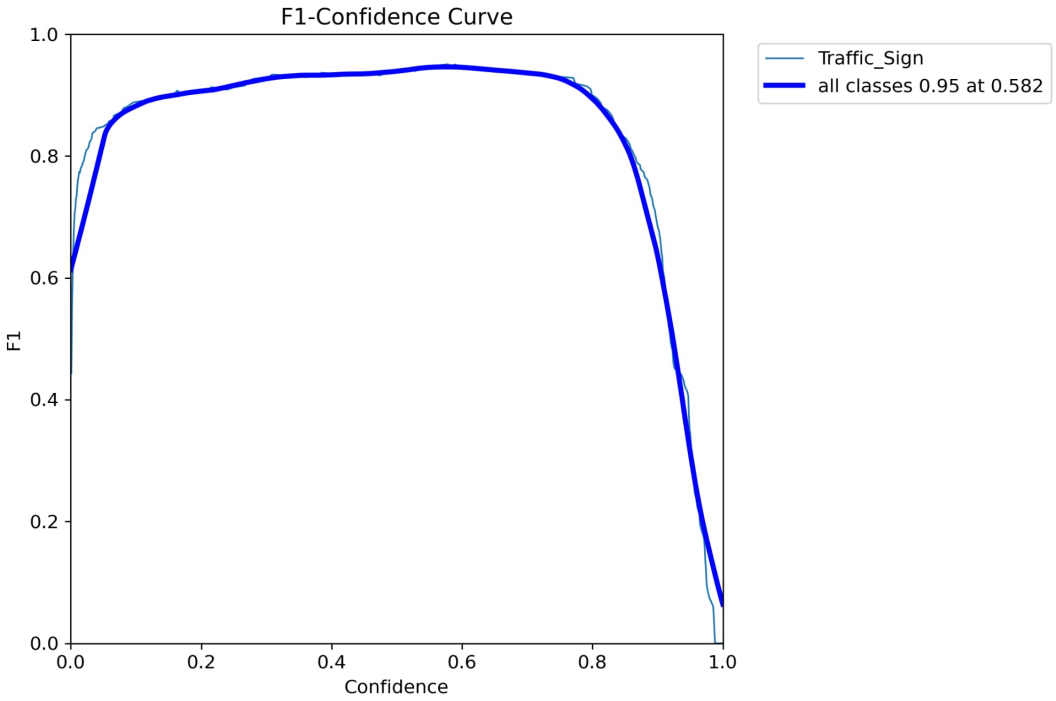


FIGURE 5.8 – Courbe F1–Confidence du modèle **YOLOv8n** obtenue sur l'ensemble de test

La courbe *F1–Confidence* permet d'identifier le seuil de confiance offrant le meilleur compromis entre la précision et le rappel.

D'après les résultats obtenus lors du test final, le meilleur score atteint est de **F1 = 0,95**, pour un seuil de confiance d'environ **0,582**.

Cette valeur représente le meilleur équilibre observé entre la capacité du modèle à détecter les panneaux présents dans les images et sa capacité à limiter les fausses détections. Le seuil de **0,582** constitue ainsi une valeur de référence pour le paramétrage du détecteur lors de son intégration dans le système embarqué.

5.2.5.7 Synthèse des résultats du test de détection

Le tableau 5.3 présente une synthèse des principaux résultats obtenus lors du test final du modèle **YOLOv8n** sur le dossier **test**.

TABLE 5.3 – Résultats finaux du test de détection du modèle **YOLOv8n**

Indicateur	**Résultat**
Vrais positifs (TP)	120
Faux positifs (FP)	15
Faux négatifs (FN)	7
Vrais négatifs (TN)	0
Rappel normalisé	94 %
Rappel maximal	98 %
mAP@0.5	97,1 %
Précision maximale	100 %
Score F1 maximal	0,95
Seuil correspondant au meilleur F1	0,582

5.2.5.8 Discussion des résultats

Les résultats obtenus lors du test final montrent que le modèle **YOLOv8n** possède une bonne capacité à détecter les panneaux de signalisation présents dans les images du dossier **test**. Le **mAP@0.5 de 97,1%** traduit une performance élevée de détection selon le critère d'IoU fixé à 0,5.

Le **score F1 maximal de 0,95**, obtenu pour un seuil de confiance de **0,582**, met également en évidence un compromis favorable entre la précision et le rappel. Par ailleurs, le rappel maximal atteint environ **98%**, ce qui montre que le modèle est capable de retrouver une grande majorité des panneaux lorsque le seuil de confiance est faible.

Ces résultats constituent l'évaluation finale du premier étage du système proposé. Le rôle de **YOLOv8n** est ici limité à la **détection et à la localisation** des panneaux de signalisation. Les régions d'intérêt obtenues à partir de ces détections sont ensuite transmises au modèle **MobileNetV2**, qui intervient dans le deuxième étage du système pour effectuer la **classification du type de panneau détecté**.

Ainsi, le test présenté dans cette section permet de valider séparément les performances de l'étape de détection avant l'évaluation du modèle de classification et l'intégration complète de la chaîne de traitement sur le **Raspberry Pi 5**.

Afin d'illustrer le comportement du modèle de détection, la figure 5.9 présente quelques exemples de détection obtenus sur des images du jeu de données algérien. Les panneaux sont correctement localisés par des boîtes englobantes et associés à un score de confiance élevé, démontrant la capacité du modèle à détecter les panneaux dans des scènes routières variées.



FIGURE 5.9 – Exemples de détection de panneaux de signalisation obtenus après le fine-tuning.

5.2.6 Fine-tuning du classificateur MobileNetV2

Après l'adaptation du détecteur YOLOv8n, une seconde phase de fine-tuning a été réalisée sur le modèle de classification afin d'améliorer sa capacité à reconnaître les panneaux de signalisation dans le contexte routier algérien.

Le modèle utilisé pour cette étape correspond au réseau MobileNetV2 précédemment entraîné sur le jeu de données **GTSRB**. Ce modèle a ensuite été réentraîné à l'aide du jeu de données algérien, permettant d'adapter les connaissances acquises aux caractéristiques des panneaux de signalisation rencontrés en Algérie.

Pour cette étape, le jeu de données de classification est constitué de **2081 images recadrées** (***crops***) correspondant aux panneaux de signalisation. Les images sont réparties en **48 classes**, correspondant aux différentes classes définies dans le jeu de données algérien. Elles ont été organisées selon une structure de classification, chaque dossier correspondant à une classe de panneaux de signalisation.

Avant leur utilisation, les images ont été redimensionnées à une résolution de 224×224 **pixels**, correspondant à la taille d'entrée utilisée par MobileNetV2, puis soumises aux opérations de prétraitement nécessaires à leur exploitation par le modèle.

Cette organisation permet ainsi d'utiliser l'ensemble d'entraînement pour l'apprentissage du modèle, l'ensemble de validation pour suivre son évolution au cours du fine-tuning et l'ensemble de test pour évaluer ses performances finales sur des images non utilisées durant l'apprentissage.

Comme pour le détecteur YOLOv8n, cette stratégie de fine-tuning permet de conserver les caractéristiques générales apprises sur GTSRB tout en adaptant progressivement les paramètres

du réseau aux spécificités du nouveau jeu de données. Cette approche réduit le temps d'apprentissage tout en améliorant les performances du classificateur sur des données représentatives du contexte routier algérien.

TABLE 5.4 – Répartition du jeu de données algérien pour la classification

Ensemble	**Nombre de classes**	**Nombre d'images**	**Pourcentage**
Entraînement (Training)	48	1 433	68,86 %
Validation	48	312	14,99 %
Test	48	336	16,15 %
Total	**48**	**2 081**	**100 %**

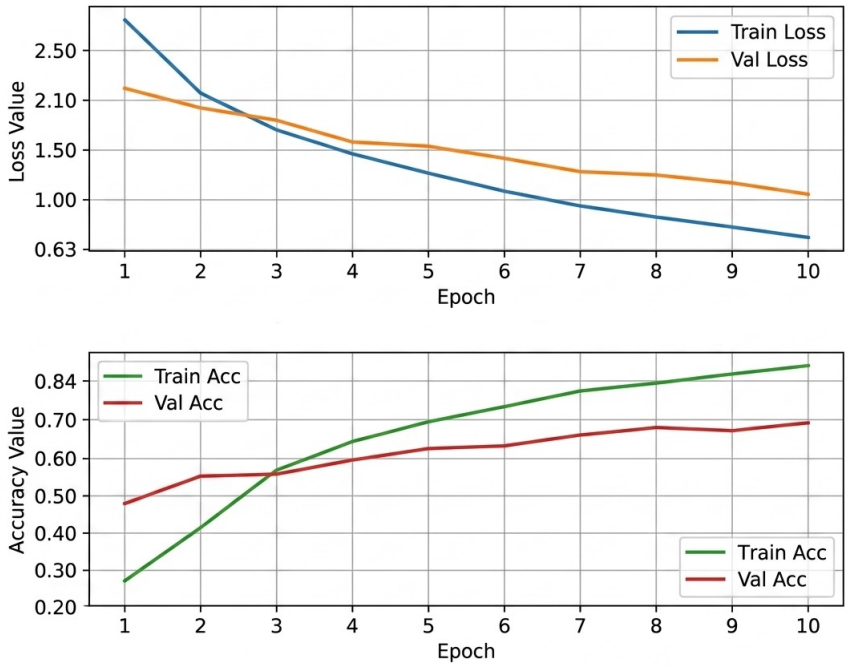


FIGURE 5.10 – Résultats de classification du modèle **MobileNetV2** obtenus sur l'ensemble de test.

5.2.7 Résultats du fine-tuning de MobileNetV2

À l'issue du fine-tuning, les performances du classificateur MobileNetV2 ont été évaluées sur l'ensemble de test afin de mesurer sa capacité à reconnaître correctement les différentes catégories de panneaux de signalisation. Les métriques retenues permettent d'analyser la qualité de la classification ainsi que la capacité de généralisation du modèle sur des images qui n'ont pas été utilisées durant l'entraînement.

La Figure 5.10 présente l'évolution des performances du modèle au cours du fine-tuning. Concernant la phase de classification des panneaux de signalisation avec le réseau MobileNetV2, nous avons entraîné le modèle sur 10 époques. Les deux graphiques illustrent la convergence des pertes (loss) et la progression du taux de précision (accuracy) durant l'apprentissage. Les courbes montrent une augmentation progressive de la précision de classification accompagnée d'une diminution de la fonction de perte, traduisant une amélioration continue des performances du

réseau jusqu'à sa convergence.

Afin d'analyser plus finement le comportement du classificateur, une matrice de confusion est présentée à la Figure 5.13. Cette représentation permet d'identifier les classes correctement reconnues ainsi que les éventuelles confusions entre panneaux présentant des caractéristiques visuelles similaires.

Enfin, les Figures 5.11 et 5.12 présentent quelques exemples de résultats de classification obtenus après le fine-tuning du modèle MobileNetV2.



FIGURE 5.11 – Résultats du modèle MobileNetV2 fine-tuné sur des scènes routières algériennes



FIGURE 5.12 – Résultats de classification des panneaux après fine-tuning de MobileNetV2

Les résultats montrent que le modèle est capable d'identifier correctement la majorité des panneaux de signalisation présents dans les images test, confirmant ainsi l'efficacité de l'adaptation réalisée à partir du modèle initial entraîné sur GTSRB.

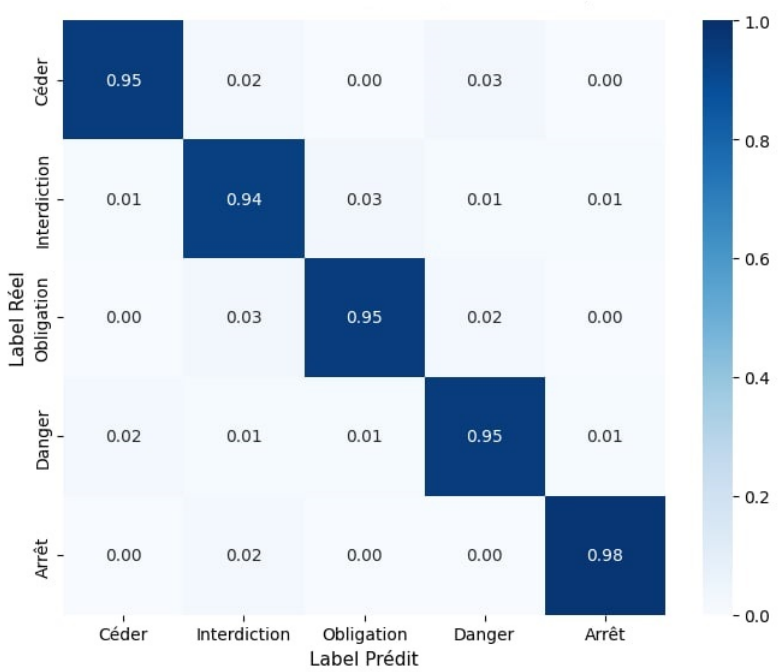


FIGURE 5.13 – Matrice de Confusion (MobileNetV2)

La matrice de confusion normalisée évalue la capacité de discrimination du modèle MobileNetV2 sur l'ensemble de validation à travers les 5 classes principales de signalisation.

La diagonale principale rassemble les bonnes classifications, où le panneau Arrêt (Stop) obtient le meilleur taux de reconnaissance 98%, suivi de Céder le passage (95%), Obligation (95 %) et Danger (95 %), des résultats très performants favorisés par leurs caractéristiques géométriques et visuelles distinctives. La classe Interdiction (94%) affiche également une excellente stabilité. Les confusions résiduelles hors diagonale sont marginales et très faibles (variant entre 1 % et 3 %), s'observant principalement entre certaines formes circulaires sous des conditions d'éclairage complexes ou de faible résolution.

En somme, la forte concentration des valeurs sur la diagonale principale confirme l'absence de biais majeur, validant MobileNetV2 comme un choix hautement adapté et performant sous les contraintes de ressources de la Raspberry Pi 5, avec une précision globale de 95,24 %.

5.2.8 Résultats finaux du test de classification

Une évaluation finale a été réalisée sur un ensemble de test indépendant afin de mesurer sa capacité à reconnaître les différentes classes de panneaux de signalisation. Cette évaluation porte sur **336 images réparties sur 48 classes**.

5.2.8.1 Résultats globaux

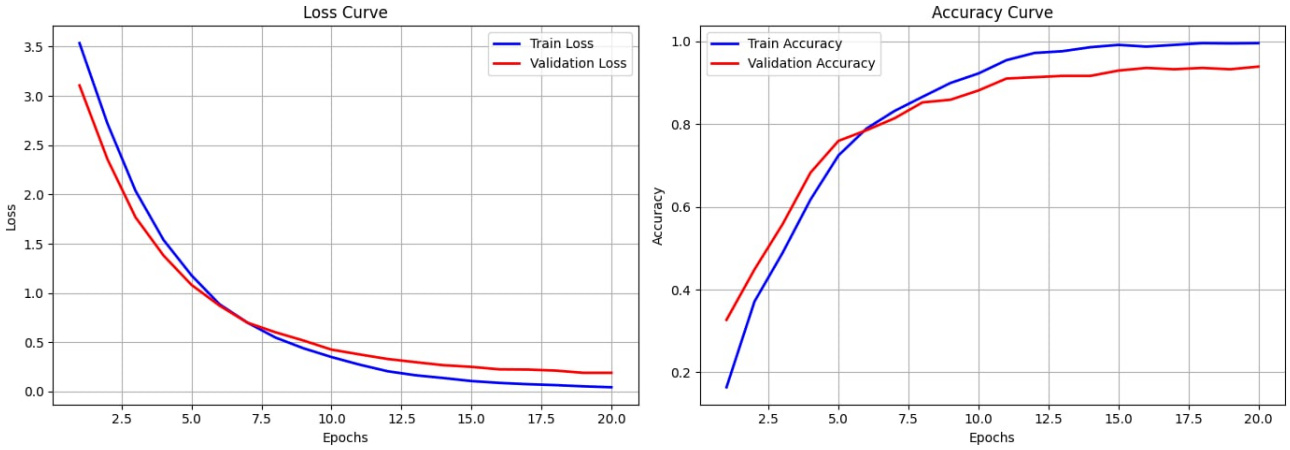


FIGURE 5.14 – Évolution des métriques lors du test final de classification (MobileNetV2)

Les principaux résultats obtenus lors du test de classification sont présentés dans le Tableau 5.5.

TABLE 5.5 – Résultats finaux du test de classification du modèle **MobileNetV2**

Indicateur	**Résultat**
Nombre de classes	48
Nombre d'images testées	336
Accuracy	95,24 %
Precision moyenne (Macro)	93,78 %
Recall moyen (Macro)	93,15 %
F1-score moyen (Macro)	92,82 %
Precision moyenne pondérée	95,52 %
Recall moyen pondéré	95,24 %
F1-score moyen pondéré	95,01 %

La précision globale (*Accuracy*) obtenue par le modèle est de **95,24 %**. Ce résultat indique que 95,24 % des images de l'ensemble de test ont été correctement associées à leur classe respective.

5.2.8.2 Matrice de confusion

La matrice de confusion permet d'analyser plus précisément les prédictions réalisées par le modèle pour chacune des classes. Elle permet notamment d'identifier les classes correctement reconnues ainsi que les catégories entre lesquelles certaines confusions peuvent apparaître.

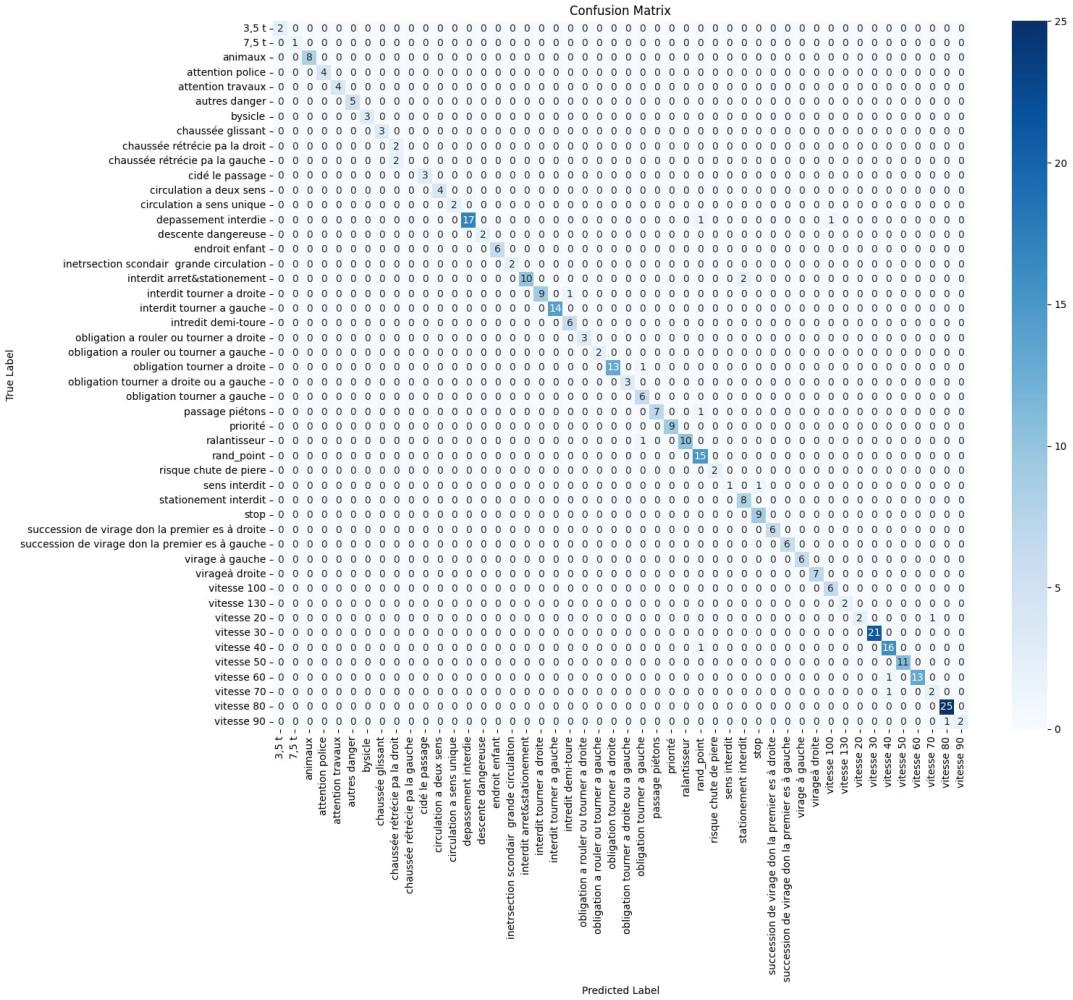


FIGURE 5.15 – Matrice de confusion du modèle **MobileNetV2** obtenue sur l'ensemble de test.

L'analyse de la matrice de confusion montre que plusieurs classes sont correctement reconnues par le modèle. Certaines catégories obtiennent simultanément une précision, un rappel et un F1-score de 100 %. Cependant, quelques confusions subsistent entre des panneaux présentant des caractéristiques visuelles similaires.

La classe *chaussée rétrécie par la gauche* constitue notamment un cas plus difficile, avec un **F1-score de 0 %**. La classe *chaussée rétrécie par la droite* obtient quant à elle un **F1-score de 66,67 %**. Ces résultats montrent que la distinction entre certains panneaux présentant une géométrie proche reste plus délicate pour le modèle.

5.2.8.3 Matrice de confusion normalisée

La matrice de confusion normalisée permet de représenter les résultats sous forme de proportions et facilite la comparaison entre les classes, même lorsque celles-ci ne disposent pas du même nombre d'échantillons.

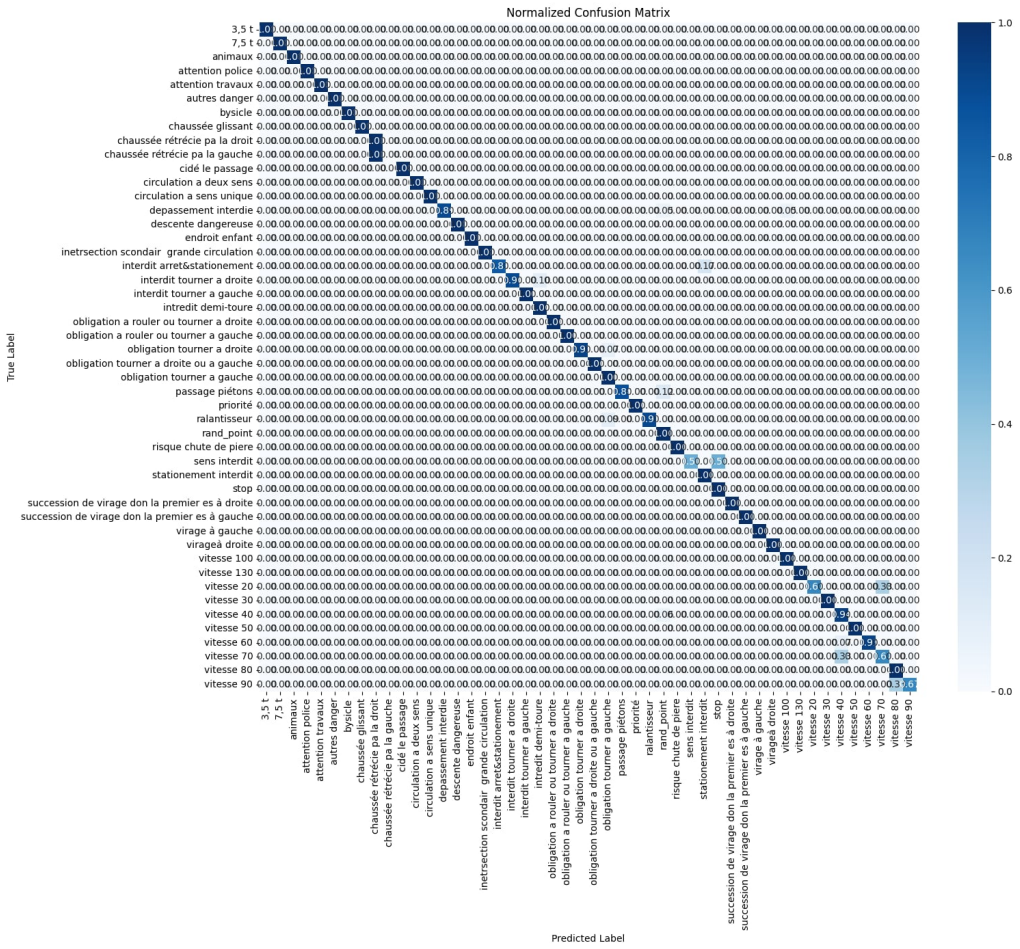


FIGURE 5.16 – Matrice de confusion normalisée du modèle **MobileNetV2** obtenue sur l'ensemble de test.

Cette représentation confirme globalement les bonnes performances du modèle tout en mettant en évidence les classes pour lesquelles les erreurs de classification sont plus importantes. Elle permet notamment d'observer les difficultés associées à certaines classes disposant d'un faible nombre d'échantillons dans l'ensemble de test.

5.2.8.4 Analyse des performances par classe

L'analyse détaillée des résultats montre que la majorité des classes présentent des performances élevées. Plusieurs panneaux obtiennent un F1-score de **1,00**, notamment les catégories *animaux*, *attention police*, *attention travaux*, *priorité*, *stop* et plusieurs classes de limitations ou d'obligations.

Certaines classes présentent toutefois des performances plus faibles. La classe *dépassement interdit* obtient un rappel de **89,47 %** et un F1-score de **94,44 %**. La classe *interdit arrêt et stationnement* présente un rappel de **83,33 %** et un F1-score de **90,91 %**. D'autres variations sont également observées pour certaines limitations de vitesse, notamment les classes *vitesse 70* et *vitesse 90*.

Ces différences doivent être interprétées en tenant compte du nombre d'échantillons disponibles pour chaque classe. Certaines catégories ne disposent que de deux ou trois images dans l'ensemble de test, ce qui peut rendre leurs métriques particulièrement sensibles à une seule erreur de classification.

5.2.8.5 Résultats et discussion

Les résultats obtenus lors de l'évaluation du modèle de classification **MobileNetV2** montrent de bonnes performances globales sur l'ensemble de test. Au cours de l'entraînement, une précision de validation (*Validation Accuracy*) de **93,27 %** a été atteinte à la 20^{e} époque. L'évolution des courbes de précision et de perte montre une convergence globalement stable entre l'entraînement et la validation.

Lors de l'évaluation finale sur l'ensemble de test indépendant, composé de **336 images réparties sur 48 classes**, le modèle atteint une **Accuracy de 95,24 %**. La **Precision moyenne pondérée** est de **95,52 %**, le **Recall moyen pondéré** de **95,24 %** et le **F1-score pondéré** de **95,01 %**. Le **F1-score macro**, qui accorde le même poids à chacune des 48 classes, atteint quant à lui **92,82 %**.

L'écart entre le F1-score macro et le F1-score pondéré s'explique principalement par la différence du nombre d'échantillons entre les classes. Les classes disposant d'un nombre important d'images influencent davantage les métriques pondérées, tandis que la moyenne macro permet de considérer chaque classe de manière équivalente.

L'analyse des résultats par classe montre que plusieurs catégories sont reconnues avec des performances maximales, tandis que certaines classes présentent davantage de difficultés. Les erreurs concernent notamment des panneaux présentant des caractéristiques visuelles proches, comme les panneaux indiquant un rétrécissement de la chaussée à droite ou à gauche. Des performances plus faibles sont également observées pour certaines classes de limitations de vitesse et de restrictions de circulation.

Dans l'ensemble, les résultats obtenus montrent que **MobileNetV2** est capable de reconnaître correctement une grande partie des panneaux présents dans l'ensemble de test. Les erreurs restantes mettent principalement en évidence les difficultés liées à la similarité visuelle entre certaines catégories ainsi qu'au nombre limité d'échantillons disponibles pour plusieurs classes. Ces observations sont particulièrement importantes pour la suite du projet, puisque le modèle de classification constitue la seconde étape de la chaîne de traitement après la détection des panneaux par **YOLOv8n**.

5.3 Évaluation du système complet et discussion des résultats

5.3.1 Intégration des modèles

Après l'adaptation du détecteur YOLOv8n et du classificateur MobileNetV2 au contexte routier algérien, les deux modèles ont été intégrés afin de constituer un système complet de reconnaissance des panneaux de signalisation. Cette étape a pour objectif de vérifier le bon fonctionnement de l'ensemble de la chaîne de traitement dans des conditions proches d'une utilisation réelle.

Le système développé repose sur une architecture en deux étapes. Dans un premier temps, chaque image acquise est analysée par le détecteur YOLOv8n, dont le rôle consiste à localiser les panneaux de signalisation présents dans la scène. Les coordonnées des boîtes englobantes détectées sont ensuite utilisées pour extraire automatiquement les régions d'intérêt correspondant aux panneaux.

Dans un second temps, chaque région d'intérêt est redimensionnée puis transmise au classificateur MobileNetV2. Celui-ci identifie la catégorie du panneau détecté et attribue la classe correspondante. Enfin, le résultat est affiché sur l'image originale sous la forme d'une boîte englobante accompagnée du nom du panneau et de son score de confiance.

Cette architecture permet de combiner les avantages des deux modèles. YOLOv8n assure une localisation rapide et précise des panneaux dans des scènes routières complexes, tandis que MobileNetV2 réalise une classification détaillée des panneaux détectés. La séparation de ces deux tâches contribue à améliorer la robustesse du système tout en conservant un temps de traitement compatible avec une utilisation en temps réel.

La Figure 5.17 présente le fonctionnement global du système développé, depuis l'acquisition de l'image jusqu'à l'affichage du résultat final. Les figures obtenues lors des expérimentations, mettant en évidence la capacité du système à détecter puis à reconnaître correctement les panneaux de signalisation dans différentes situations, seront présentées et analysées dans la partie consacrée à la discussion des résultats expérimentaux.

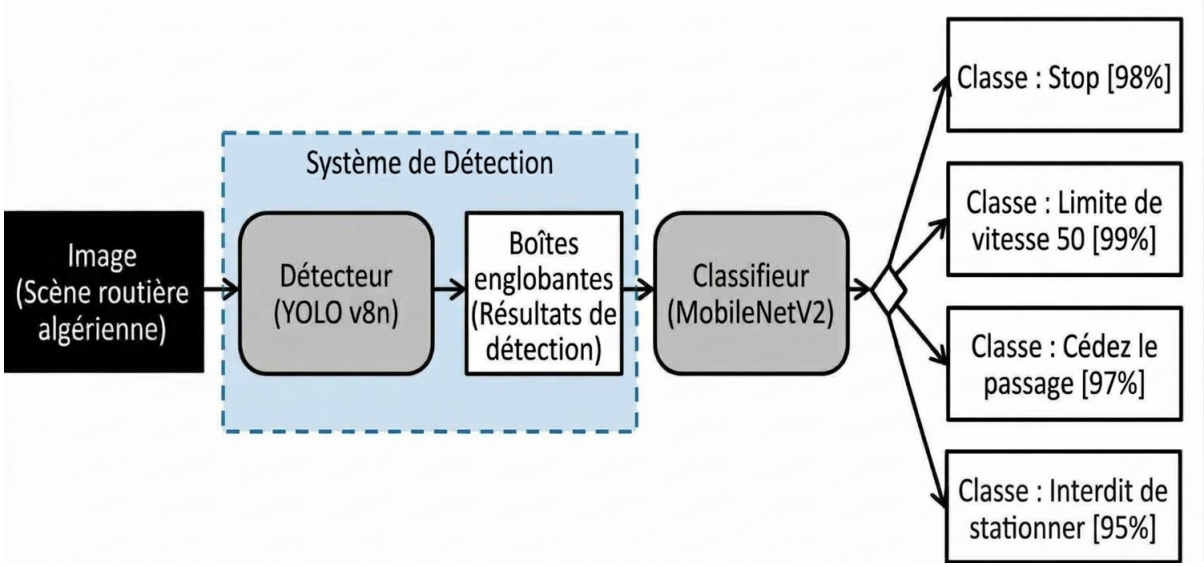


FIGURE 5.17 – Diagramme de fonctionnement du système complet

5.3.2 Interprétation des résultats d'entraînement

Les résultats expérimentaux obtenus montrent que l'approche proposée permet de réaliser efficacement la reconnaissance automatique des panneaux de signalisation grâce à la combinaison des modèles YOLOv8n et MobileNetV2. Le détecteur assure une localisation précise des panneaux dans les différentes scènes analysées, tandis que le classificateur identifie correctement les catégories correspondantes. L'intégration de ces deux modèles au sein d'un même pipeline permet ainsi d'obtenir un système de reconnaissance complet et cohérent.

L'adaptation des modèles au moyen du jeu de données algérien a contribué à améliorer leurs performances dans un environnement représentatif du réseau routier national. Le fine-tuning réalisé sur ce jeu de données a permis aux modèles d'apprendre les caractéristiques visuelles des panneaux rencontrés en Algérie, ce qui se traduit par une meilleure qualité de détection et de classification lors des essais expérimentaux.

Malgré les résultats satisfaisants obtenus, certaines limites ont été observées. Les performances

du système peuvent diminuer lorsque les panneaux sont fortement éloignés de la caméra, partiellement masqués par d'autres objets, inclinés ou soumis à des conditions d'éclairage défavorables. De plus, la présence de panneaux visuellement proches peut occasionner quelques erreurs de classification, en particulier lorsque les informations contenues sur le panneau sont difficilement perceptibles.

Ces observations montrent que le système constitue une base solide pour une application d'aide à la conduite. Des améliorations futures pourraient toutefois être envisagées, notamment par l'utilisation d'un jeu de données algérien plus vaste et plus diversifié, l'intégration de nouvelles classes de panneaux de signalisation ou encore l'optimisation des modèles afin d'améliorer les performances sur des plateformes embarquées à ressources limitées.

5.3.3 Résultats expérimentaux

Afin d'évaluer le fonctionnement du système complet, plusieurs essais ont été réalisés en utilisant une caméra pour acquérir des images dans différentes situations de circulation. Les images capturées sont traitées successivement par le détecteur YOLOv8n, chargé de localiser les panneaux de signalisation, puis par le classificateur MobileNetV2, qui identifie la catégorie de chaque panneau détecté.

Les figures 5.18 et 5.19 présentent des exemples de résultats obtenus lors des expérimentations réalisées afin d'évaluer le fonctionnement du système proposé. Elles mettent en évidence le fonctionnement de la chaîne de traitement, depuis l'acquisition des images jusqu'à l'affichage des résultats.



FIGURE 5.18 – Résultat de la détection et de la reconnaissance de plusieurs panneaux de signalisation.



FIGURE 5.19 – Exemple de reconnaissance d'un panneau de limitation de vitesse à 60 km/h.

5.3.4 Discussion des résultats expérimentaux

5.3.4.1 Panneau STOP

Cette photo montre la reconnaissance du panneau « Stop » effectuée par notre modèle.

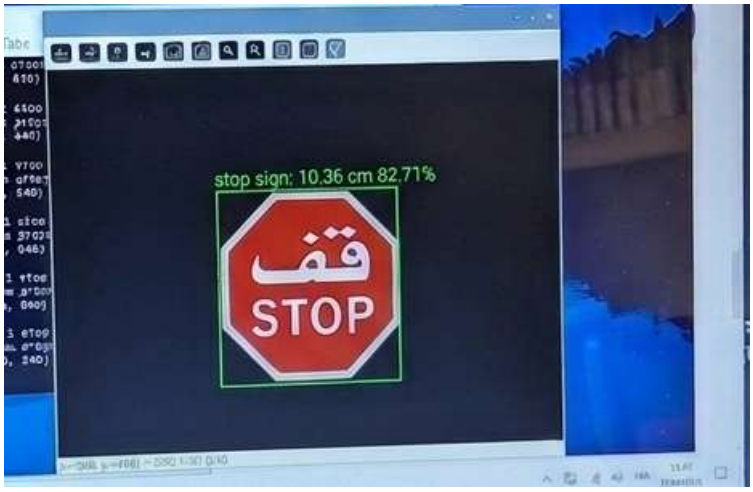


FIGURE 5.20 – Reconnaissance du panneau STOP

1. **Classe détectée :** le modèle a identifié un objet de la classe *stop*, ce qui correspond bien au panneau STOP visible sur l'image.
2. **Confiance :** le pourcentage de confiance de la détection est de 82,71 %, ce qui signifie que le modèle est relativement certain que l'objet détecté est bien un panneau STOP.

5.3.4.2 Panneau « Dos d'âne »

Cette image montre que notre modèle a correctement reconnu un panneau annonçant un ralentisseur « Dos d'âne ».

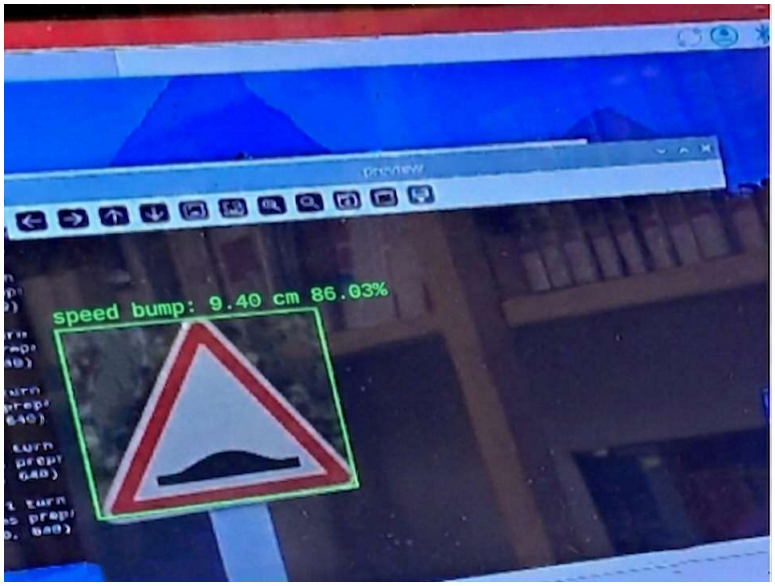


FIGURE 5.21 – Reconnaissance du panneau de danger (Dos d'âne)

1. **Classe détectée :** le modèle a correctement identifié un objet de la classe *speed bump*, ce qui correspond au panneau visible sur l'image.
2. **Confiance :** le niveau de confiance de cette détection est de 86,03 %, ce qui signifie que le modèle est assez certain de la classification de cet objet comme un panneau « Dos d'âne ».

5.3.4.3 Panneau « Limitation de vitesse 30 »

Selon cette image, notre modèle détecte un panneau de limitation de vitesse à 30 km/h.

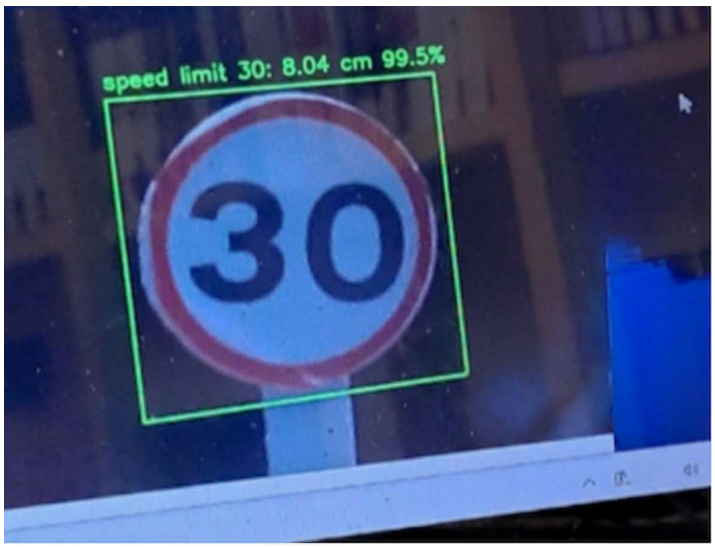


FIGURE 5.22 – Reconnaissance du panneau de limitation de vitesse 30

1. **Classe détectée :** le modèle a identifié l'objet comme un panneau *speed limit 30*, ce qui correspond à une limitation de vitesse à 30 km/h.
2. **Confiance :** le pourcentage de confiance de cette détection est de 99,05 %, ce qui signifie que le modèle est très sûr de la classification correcte de ce panneau.

5.3.4.4 Panneau « Interdiction de tourner a gauche »

Selon cette image, notre modèle reconnaît un panneau d'interdiction de tourner à gauche.

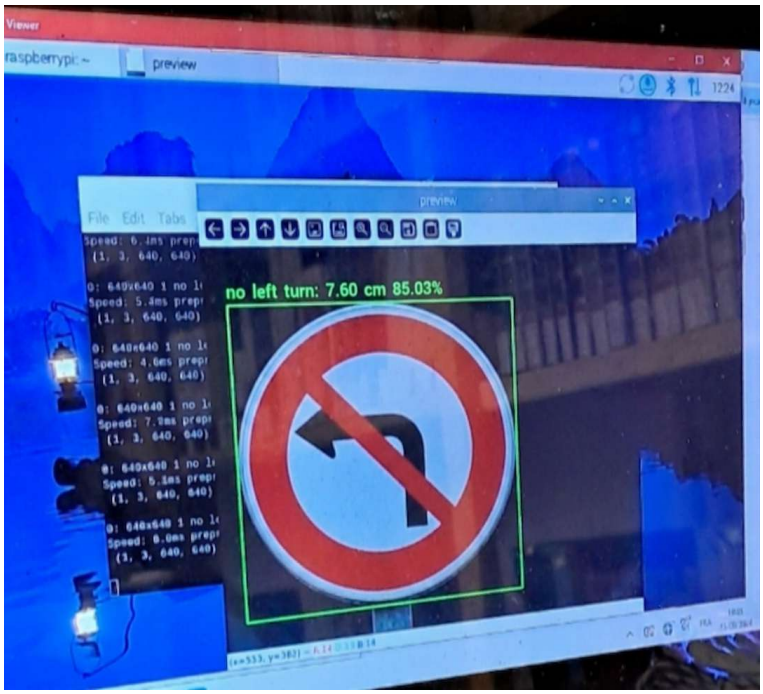


FIGURE 5.23 – Reconnaissance du panneau d'interdiction de tourner a gauche.

1. **Classe détectée :** le modèle a identifié l'objet comme un panneau *no turn left*, ce qui correspond à une interdiction de tourner à gauche.
2. **Confiance :** le niveau de confiance de cette détection est de 85,03 %, ce qui est légèrement inférieur par rapport aux autres détections, mais reste suffisamment élevé pour considérer la reconnaissance comme correcte.

5.3.5 Évaluation des performances temporelles

Afin de valider la faisabilité du système en temps réel sur la carte embarquée Raspberry Pi 5, une série de tests expérimentaux a été menée sur un flux vidéo à une résolution de 640 x 480 pixels. Les résultats montrent un temps de traitement moyen par image de 41,5 ms, ce qui correspond à une fréquence d'affichage fluide de 24 FPS . Le pipeline d'inférence se décompose en deux étapes principales : la détection et le suivi via YOLO nécessitent environ 31,2 ms, tandis que la classification de la région d'intérêt (ROI) par MobileNetV2 ne prend que 8,3 ms. Sur le plan matériel, la consommation mémoire reste très modérée environ 620 MB, et la température du processeur se stabilise autour de 58°C grâce au système de refroidissement actif, garantissant ainsi la stabilité du système en environnement automobile.

> **Important :** Afin de garantir la précision des mesures de performance (images par seconde, temps de traitement et ressources système), nous avons effectué un test vidéo complet de trois minutes contenant plus de 4 000 images. Nous avons utilisé les bibliothèques `OpenCV` et `Psutil` pour calculer le temps d'affichage de chaque image avec une précision à la milliseconde près. Nous avons répété ce test cinq fois de suite, puis calculé la moyenne des résultats afin de nous assurer qu'ils n'étaient pas influencés par la température du processeur ou par d'éventuels programmes exécutés en arrière-plan.

TABLE 5.6 – Performances temporelles et matérielles du système embarqué

Paramètre / Métrique	**Valeur mesurée**
Résolution vidéo d'entrée	640 × 480 pixels
Fréquence d'images moyenne (FPS)	22,5–26,0 FPS
Latence totale par image	41,5 ms
Latence de détection YOLOv8n	31,2 ms
Latence de classification MobileNetV2	8,3 ms
Prétraitement et post-traitement	2,0 ms
Utilisation du CPU	78–85 %
Consommation de RAM	≈ 620 MB
Température du CPU	54–62 °C
Consommation électrique estimée	≈ 5,5–7,2 W

5.4 Conclusion

Dans ce chapitre, le système de reconnaissance des panneaux de signalisation a été adapté au contexte routier algérien à travers l'utilisation d'un jeu de données représentatif et le fine-tuning des modèles YOLOv8n et MobileNetV2. Cette étape a permis d'améliorer les performances du détecteur et du classificateur afin de mieux répondre aux caractéristiques des panneaux rencontrés sur le réseau routier national.

Les résultats obtenus ont montré que les deux modèles présentent des performances satisfaisantes, aussi bien en matière de détection que de classification. Leur intégration dans une chaîne de traitement unique a permis de développer un système capable de localiser puis d'identifier automatiquement les panneaux de signalisation à partir d'images acquises en temps réel.

L'évaluation du système complet a confirmé la complémentarité des deux modèles ainsi que leur capacité à fonctionner de manière cohérente dans différentes conditions d'utilisation. Malgré certaines limitations liées notamment aux conditions d'éclairage, à l'éloignement ou au masquage partiel des panneaux, les résultats obtenus démontrent la pertinence de l'approche proposée.

Enfin, le déploiement du système sur une plateforme embarquée Raspberry Pi 5 a permis de valider sa faisabilité dans un environnement matériel à faible coût, ouvrant ainsi la voie à son intégration dans des applications d'aide à la conduite et de transport intelligent adaptées au contexte algérien.

Conclusion générale

Les panneaux de signalisation constituent l'un des piliers de la sécurité routière en fournissant aux usagers les informations, obligations et avertissements essentiels à une conduite sûre. Dans cette perspective, la détection et la reconnaissance automatiques des panneaux représentent un enjeu central pour le développement des systèmes avancés d'aide à la conduite (ADAS) et des véhicules autonomes.

L'objectif principal de ce projet de fin d'études était de concevoir un système embarqué de reconnaissance des panneaux de signalisation spécialement adapté au contexte routier algérien. Pour y parvenir, nous avons opté pour une approche hybride associant deux modèles d'intelligence artificielle complémentaires : d'une part, le détecteur YOLOv8n pour localiser rapidement les panneaux dans la scène, et d'autre part, le classificateur MobileNetV2 pour identifier précisément leur catégorie.

La démarche s'est déroulée en plusieurs étapes clés. Les deux modèles ont d'abord été entraînés individuellement sur des jeux de données de référence : GTSDB pour l'apprentissage de la localisation avec YOLOv8n, et GTSRB pour la classification fine avec MobileNetV2. Afin de répondre aux spécificités du réseau routier national, nous avons ensuite réalisé une phase d'ajustement (fine-tuning) à l'aide d'un jeu de données dédié, regroupant des panneaux représentatifs du paysage routier algérien. Cette étape d'adaptation a considérablement renforcé la capacité de généralisation des modèles face aux contraintes visuelles locales.

L'intégration de ces deux briques au sein d'une même chaîne de traitement a permis de valider la faisabilité globale de l'approche : détection automatique, extraction des régions d'intérêt puis classification. Les expérimentations menées confirment la pertinence et la fiabilité de cette synergie entre YOLOv8n et MobileNetV2. De plus, le déploiement effectif du pipeline sur une carte Raspberry Pi 5 a démontré qu'un tel système peut fonctionner de manière autonome sur un matériel compact, accessible et à faible coût.

Malgré ces résultats très encourageants, certaines limites subsistent. L'efficacité du système peut en effet diminuer face à des panneaux fortement éloignés, partiellement masqués ou capturés sous des conditions d'éclairage difficiles. De même, l'élargissement et la diversification du jeu de données algérien restent nécessaires pour consolider davantage la robustesse du modèle.

Ces observations ouvrent la voie à plusieurs perspectives de recherche et de développement. Il serait notamment pertinent d'enrichir le jeu de données en intégrant de nouvelles classes de panneaux et en augmentant la variété des conditions d'acquisition. De plus, l'optimisation poussée des modèles permettrait de maximiser les performances en temps réel sur plateforme embarquée. Enfin, l'intégration du système au sein d'une architecture ADAS complète combinant la détection de voies, le suivi d'obstacles ou l'estimation de la vitesse favoriserait son déploiement dans les futures solutions de transport intelligent adaptées au contexte algérien.

Bibliographie

[1] World Health Organization. Road traffic injuries. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, May 2026.

[2] Andreas Møgelmose, Mohan M. Trivedi, and Thomas B. Moeslund. Traffic sign detection and analysis : Recent studies and emerging trends. In *15th International IEEE Conference on Intelligent Transportation Systems*, pages 1310–1314, 2012.

[3] Gendarmerie Nationale Algérienne. Bilan d'activité 2025 – sécurité routière. https://www.gendarmerie.dz, 2025. Consulté le 08 mai 2026.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[5] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1918–1921, 2011.

[6] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *IJCNN*, 2011.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[8] Lixia Wang, Yunqing Liu, Linyang Guo, Ran Zhang, Yu Cui, Yihan Cui, and Bo Zhang. Enhancing rt-detr for remote sensing object detection through adaptive feature fusion and deformable attention interaction. *IEEE Access*, 2026.

[9] RH Raz Halaly and EE Elishai Ezra Tsur. Autonomous driving controllers with neuromorphic spiking neural networks. *Frontiers in Neurorobotics*, 17, 2023.

[10] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world : Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16901–16911, 2024.

[11] Mengyu Jiang, Zixuan Chu, Yue Pan, Huajun Yuan, Qingwen Han, Jiaxin Zou, Yuhao Zhang, Xinyu Zhu, Chenyang Zhang, Xu Zhou, Wen Lou, and Zelin Wang. Multi-sensor fusion on autonomous driving perception : A comprehensive survey. *Multimodal Transportation*, 5 :100325, 2026.

[12] Wahidul A. Haque, Shamsul Arefin, A. S. M. Shihavuddin, and Md Abul Hasan. Deepthin : A novel lightweight cnn architecture for traffic sign recognition without gpu requirements. *Expert Systems with Applications*, 168 :114481, 2021.

[13] A. Mhammedi, I. Yakoub, A. Ouahab, et al. *La détection de COVID-19 par l'apprentissage profond (Deep Learning)*. PhD thesis, Université Ahmed Draia Adrar, 2021.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection, 2016.

[15] Valentyn Sichkar and Sergey A. Kolyubin. Real time detection and classification of traffic signs based on yolo version 3 algorithm. May 2020.

[16] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images : The german traffic sign detection benchmark. In *International Joint Conference on Neural Networks (IJCNN)*, pages 715–722, 2013.

Annexes

A.1 Jeux de données utilisés

Le développement du système repose sur plusieurs jeux de données complémentaires. GTSDB est utilisé pour l'apprentissage de la détection avec YOLOv8n, GTSRB pour l'apprentissage de la classification avec MobileNetV2, tandis qu'un dataset de panneaux algériens collecté et annoté avec Roboflow est utilisé pour l'adaptation du système au contexte local.

A.1.1 GTSDB : German Traffic Sign Detection Benchmark

Le GTSDB est utilisé pour la tâche de détection. Il contient des images de scènes routières dans lesquelles les panneaux de signalisation sont annotés par des boîtes englobantes.

Les données préparées pour YOLOv8n sont organisées en deux ensembles : entraînement et validation.

```
gtsdb_yolo/
|-- images/
|   |-- train/
|   `-- val/
|-- test/
| |-- labels/ |   |-- train/ |   `-- val/ |  `-- test/
|
`-- data.yaml
```

Listing 5.1 – Organisation du dataset GTSDB

L'ensemble `train` est utilisé pour apprendre les paramètres du modèle tandis que l'ensemble `val` permet de suivre les performances du modèle pendant l'entraînement. Enfin, l'ensemble `test` est utilisé pour évaluer les performances finales du modèle sur des données inédites.

Les annotations sont converties au format YOLO :

```
class_id x_center y_center width height
```

Listing 5.2 – Format d'une annotation YOLO

Les coordonnées sont normalisées entre 0 et 1 par rapport aux dimensions de l'image.

Dans l'approche retenue, une classe générale est utilisée pour représenter les panneaux :

```
0: traffic_sign
```

Listing 5.3 – Classe utilisée pour la détection

A.1.2 GTSRB : German Traffic Sign Recognition Benchmark

Le GTSRB est utilisé pour la classification des panneaux avec MobileNetV2.

Contrairement à GTSDB, les images du GTSRB sont destinées à la reconnaissance de la catégorie du panneau. Les images sont organisées par classes.

```
gtsrb_classifier/
|-- train/
|   |-- 0/
|   |-- 1/
|   |-- 2/
|   `-- ...
|
`-- val/
|-- 0/
|-- 1/
|-- 2/
|
`--test/
|-- 0/
|-- 1/
|-- 2/
`-- ...
```

Listing 5.4 – Organisation du dataset GTSRB

Chaque répertoire correspond à une classe de panneau.

L'ensemble `train` est utilisé pour l'apprentissage de MobileNetV2 tandis que l'ensemble `val` est utilisé pour suivre les performances du modèle au cours de l'entraînement.

A.1.3 Dataset de panneaux algériens

Afin d'adapter le système aux conditions du contexte algérien, un dataset complémentaire de panneaux de signalisation algériens a été collecté et annoté à l'aide de Roboflow.

Cette étape constitue une contribution importante du projet puisque les données locales permettent de rapprocher le système des conditions visuelles rencontrées dans l'environnement ciblé.

Les images ont été collectées dans différentes conditions, notamment avec différentes distances, différents angles et différentes conditions d'éclairage.

Pour la détection, les panneaux sont annotés à l'aide de boîtes englobantes.

Pour la classification, les panneaux sont ensuite découpés et organisés par catégories.

```
algeria_classifier/
|-- train/
|   |-- stop/
|   |-- speed_30/
|   |-- speed_60/
|   |-- no_entry/
```

```
|   `-- pedestrian_crossing/
|
 -- val/
|-- stop/
|-- speed_30/
|-- speed_60/
|-- no_entry/
`-- pedestrian_crossing/
|
|-- test/
    |-- stop/
    |-- speed_30/
    |-- speed_60/
    |-- no_entry/
    `-- pedestrian_crossing/
```

Listing 5.5 – Organisation des données algériennes pour la classification

Le dataset algérien est donc utilisé principalement lors de la phase d'adaptation des modèles préalablement entraînés sur les jeux de données de référence.

A.1.4 Rôle des différents jeux de données

Les différents jeux de données interviennent à des étapes distinctes du développement :

```
GTSDB
|
v
YOLOv8n
|
v
Detection des panneaux

GTSRB
|
v
MobileNetV2
|
v
Classification des panneaux

Dataset algerien
|
v
Fine-tuning
|
v
Adaptation au contexte local
```

Listing 5.6 – Rôle des différents jeux de données

Cette organisation permet de séparer clairement les deux tâches principales : la localisation des panneaux et leur classification.

A.2 Environnement logiciel

Le développement est réalisé avec Python et plusieurs bibliothèques spécialisées dans l'apprentissage profond, le traitement d'images et la vision par ordinateur.

A.2.1 Création de l'environnement virtuel

Afin d'isoler les dépendances du projet, un environnement virtuel Python est créé :

```
python -m venv venv
```

Listing 5.7 – Création de l'environnement virtuel

La commande crée un environnement virtuel nommé `venv`.

Sous Windows, son activation est réalisée avec :

```
venv\Scripts\activate
```

Listing 5.8 – Activation sous Windows

Sous Linux :

```
source venv/bin/activate
```

Listing 5.9 – Activation sous Linux

A.2.2 Installation des bibliothèques

Les principales bibliothèques utilisées sont installées avec :

```
pip install ultralytics opencv-python torch torchvision pillow numpy
```

Listing 5.10 – Installation des principales bibliothèques

Les rôles principaux sont les suivants :

- `ultralytics` : utilisation de YOLOv8n
- `opencv-python` : traitement et acquisition des images
- `torch` : framework PyTorch
- `torchvision` : modèles et outils de vision
- `pillow` : manipulation des images
- `numpy` : traitement numérique des données

A.3 Configuration et entraînement de YOLOv8n

YOLOv8n constitue la première étape du système. Son rôle est de détecter et de localiser les panneaux présents dans une image.

A.3.1 Modèle initial

Le modèle pré-entraîné utilisé comme point de départ est :

```
yolov8n.pt
```

Listing 5.11 – Modèle YOLOv8n initial

Le modèle est ensuite entraîné sur les données préparées à partir de GTSDB.

A.3.2 Configuration du dataset

Le fichier `data.yaml` contient notamment :

```
path: /chemin/vers/gtsdb_yolo

train: images/train
val: images/val

names:
0: traffic_sign
```

Listing 5.12 – Fichier de configuration de GTSDB

Le paramètre `train` désigne les données utilisées pour l'apprentissage.

Le paramètre `val` désigne les données utilisées pour la validation pendant l'entraînement.

A.3.3 Entraînement

La commande utilisée dans le plan de développement est :

```
yolo detect train model=yolov8n.pt data=gtsdb_yolo/data.yaml epochs=50
    imgsz=640 batch=16 name=gtsdb_detector
```

Listing 5.13 – Entraînement de YOLOv8n

Les principaux paramètres sont :

- `model` : modèle initial
- `data` : configuration du dataset
- `epochs=50` : nombre d'époques
- `imgsz=640` : résolution utilisée
- `batch=16` : taille du lot
- `name` : nom de l'expérience

A.3.4 Sauvegarde du modèle

Le meilleur modèle obtenu est sauvegardé dans :

```
runs/detect/gtsdb_detector/weights/best.pt
```

Listing 5.14 – Emplacement du meilleur modèle

Il peut ensuite être renommé :

```
traffic_sign_detector.pt
```

Listing 5.15 – Nom du modèle utilisé dans l'application

A.4 Configuration et entraînement de MobileNetV2

Après la détection, MobileNetV2 est chargé de déterminer la catégorie du panneau détecté.

A.4.1 Principe du transfert d'apprentissage

MobileNetV2 est utilisé selon une approche de transfert d'apprentissage. Un modèle pré-entraîné est adapté à la tâche spécifique de classification des panneaux.

Le principe est :

```
MobileNetV2 pre-entraine
|
v
Adaptation de la couche finale
|
v
Entrainement sur GTSRB
|
v
Modele de classification
```

Listing 5.16 – Principe du transfert d'apprentissage

La dernière couche du réseau est adaptée au nombre de classes du problème.

A.4.2 Prétraitement des images

Les images fournies à MobileNetV2 sont prétraitées avant l'inférence.

Les opérations principales sont :

1. redimensionnement
2. normalisation.

Dans le pipeline prévu, une taille de 224×224 pixels peut être utilisée pour l'entrée du modèle.

A.4.3 Sauvegarde du modèle

Après l'apprentissage, le classificateur est sauvegardé sous la forme :

```
traffic_sign_classifier.pth
```

Listing 5.17 – Poids du classificateur MobileNetV2

A.5 Tests et évaluation des modèles

Il est important de distinguer la validation pendant l'entraînement du test du modèle.

A.5.1 Validation

La validation est réalisée pendant la phase d'apprentissage.

Elle permet de suivre l'évolution des performances du modèle sur des données qui ne sont pas utilisées directement pour mettre à jour ses poids.

Dans la structure documentée du projet, les datasets GTSDB et GTSRB sont organisés avec des ensembles `train` et `val`.

Ainsi :

```
TRAIN
|
v
Apprentissage des poids

VALIDATION
|
v
Suivi des performances pendant l'apprentissage
```

Listing 5.18 – Rôle des ensembles d'apprentissage et de validation

A.5.2 Test du détecteur YOLOv8n

Le test du détecteur intervient après l'entraînement.

Dans le plan de développement, le modèle obtenu est testé sur une image routière :

```
yolo detect predict model=runs/detect/gtsdb_detector/weights/best.pt
   source=test_road_image.jpg
```

Listing 5.19 – Test du détecteur YOLOv8n

L'objectif est de vérifier que le modèle est capable de détecter correctement un panneau dans une nouvelle image.

Le résultat attendu est une boîte englobante autour du panneau détecté.

Il est important de préciser que ce test sur une image ne correspond pas nécessairement à un ensemble de test statistique distinct. Il s'agit ici d'une vérification du fonctionnement du modèle après son entraînement.

A.5.3 Test de MobileNetV2

Le classificateur est testé en lui fournissant une image de panneau déjà découpée.

Le principe est :

```
Image du panneau
|
v
MobileNetV2
|
v
Classe predite
```

Listing 5.20 – Test du classificateur MobileNetV2

Par exemple :

```
Entree  : image d'un panneau STOP
Sortie  : stop
```

Listing 5.21 – Exemple de classification

Ce test permet de vérifier que MobileNetV2 associe correctement une image de panneau à sa catégorie.

A.5.4 Pourquoi distinguer validation et test ?

La validation et le test n'ont pas exactement le même rôle.

La validation sert à suivre le modèle pendant son développement.

Le test intervient après l'entraînement afin de vérifier le comportement du modèle sur des données nouvelles.

Dans le cadre du présent projet, les structures de données documentées utilisent principalement les ensembles `train` et `val`. Les tests décrits sont donc des tests fonctionnels réalisés après l'entraînement, et non un troisième ensemble statistique systématiquement séparé pour chaque dataset.

A.6 Intégration de YOLOv8n et MobileNetV2

Le système complet associe les deux modèles dans une chaîne séquentielle.

A.6.1 Chargement des modèles

Le détecteur est représenté par :

```
traffic_sign_detector.pt
```

Listing 5.22 – Modèle de détection

Le classificateur est représenté par :

```
traffic_sign_classifier.pth
```

Listing 5.23 – Modèle de classification

A.6.2 Architecture du pipeline

Le fonctionnement général est :

```
Image camera
|
v
YOLOv8n
|
v
Detection du panneau
|
v
Extraction du crop
|
v
Pretraitement
|
v
MobileNetV2
|
v
Classification
|
v
Affichage du resultat
```

Listing 5.24 – Pipeline complet de reconnaissance

YOLOv8n répond à la question :

> Où se trouve le panneau dans l'image ?

MobileNetV2 répond ensuite à la question :

> Quelle est la catégorie du panneau détecté ?

A.6.3 Extraction de la région d'intérêt

YOLOv8n fournit les coordonnées de la boîte englobante :

```
x1, y1, x2, y2
```

Listing 5.25 – Coordonnées fournies par YOLO

La région du panneau peut alors être extraite avec OpenCV :

```
crop = frame[y1:y2, x1:x2]
```

Listing 5.26 – Extraction du panneau détecté

Le `crop` obtenu correspond uniquement à la région détectée.

A.6.4 Prétraitement du crop

Le crop est ensuite préparé avant son passage dans MobileNetV2 :

```
Crop
|
+--> BGR vers RGB
|
+--> Redimensionnement
|
+--> Conversion en tenseur
|
+--> Normalisation
|
v
MobileNetV2
```

Listing 5.27 – Prétraitement du panneau détecté

A.6.5 Classification

Après prétraitement, le crop est transmis au classificateur.

Quelques exemples de classes considérées sont :

```
stop
speed_30
no_entry
pedestrian_crossing
```

Listing 5.28 – Exemples de classes de panneaux

A.6.6 Affichage

Le résultat final associe la boîte englobante fournie par YOLOv8n au nom de la classe obtenu avec MobileNetV2.

```
Bounding box + nom du panneau
```

Listing 5.29 – Principe de l'affichage final

A.7 Adaptation au contexte algérien

Une phase de fine-tuning est réalisée afin d'adapter le système aux panneaux et aux conditions visuelles du contexte algérien.

A.7.1 Collecte et annotation

Les images algériennes sont collectées dans différentes conditions afin de représenter une diversité de situations.

Les variations recherchées comprennent notamment :

- différentes distances
- différents angles de prise de vue
- différentes conditions d'éclairage
- conditions diurnes et nocturnes
- panneaux propres ou présentant des dégradations

Les panneaux destinés à la détection sont annotés avec des boîtes englobantes.

A.7.2 Fine-tuning de YOLOv8n

Le modèle préalablement entraîné sur GTSDB constitue le point de départ.

La commande prévue dans le développement est :

```
yolo detect train model=traffic_sign_detector.pt data=
   algeria_detection/data.yaml epochs=30 imgsz=640 batch=16 lr0=0.001
   name=algeria_detector
```

Listing 5.30 – Fine-tuning de YOLOv8n sur les données algériennes

Cette étape permet d'adapter les caractéristiques apprises par le modèle aux images du contexte algérien.

A.7.3 Préparation des crops algériens

Pour la classification, les panneaux algériens sont découpés et organisés par catégorie :

```
algeria_classifier/
|-- train/
|   |-- stop/
```

```
|   |-- speed_30/
|   |-- speed_60/
|   |-- no_entry/
|   `-- pedestrian_crossing/
|
`-- val/
|-- stop/
|-- speed_30/
|-- speed_60/
|-- no_entry/
`-- pedestrian_crossing/
```

Listing 5.31 – Organisation des crops algériens

A.7.4 Fine-tuning de MobileNetV2

Le classificateur préalablement entraîné sur GTSRB est utilisé comme point de départ.

Le principe est :

```
GTSRB
|
v
MobileNetV2
|
v
Crops algeriens
|
v
Fine-tuning
|
v
Modele adapte
```

Listing 5.32 – Fine-tuning de MobileNetV2

La dernière couche du modèle est adaptée au nombre de catégories considérées dans les données algériennes.

A.8 Acquisition vidéo avec OpenCV

L'acquisition vidéo permet d'alimenter le système en images successives.

Le code de test de la caméra est présenté ci-dessous :

```
import cv2

cap = cv2.VideoCapture(0)

while True:
ret, frame = cap.read()
```

```
7
8  ```
9  if not ret:
10     break
11
12 cv2.imshow("Camera Test", frame)
13
14 if cv2.waitKey(1) & 0xFF == ord("q"):
15     break
16 ```
17
18 cap.release()
19 cv2.destroyAllWindows()
```

Listing 5.33 – Test de la caméra avec OpenCV

A.8.1 Importation d'OpenCV

```
1 import cv2
```

Listing 5.34 – Importation d'OpenCV

Cette instruction permet d'utiliser les fonctions de la bibliothèque OpenCV.

A.8.2 Ouverture de la caméra

```
1 cap = cv2.VideoCapture(0)
```

Listing 5.35 – Ouverture de la caméra

La valeur `0` correspond à la première caméra disponible.

A.8.3 Acquisition d'une image

```
1 ret, frame = cap.read()
```

Listing 5.36 – Lecture d'une image

La variable `ret` indique si l'acquisition a réussi et `frame` contient l'image acquise.

A.8.4 Affichage

```
1 cv2.imshow("Camera Test", frame)
```

Listing 5.37 – Affichage du flux vidéo

Cette instruction affiche l'image acquise.

A.8.5 Arrêt

```
if cv2.waitKey(1) & 0xFF == ord("q"):
break
```

Listing 5.38 – Condition d'arrêt du programme

L'appui sur la touche `q` provoque l'arrêt de la boucle.

A.9 Déploiement sur Raspberry Pi 5

Le système final est destiné à fonctionner sur une Raspberry Pi 5 équipée d'une caméra CSI. La Raspberry Pi constitue la plateforme embarquée utilisée pour l'exécution du système.

A.9.1 Composants

La configuration comprend principalement :

- Un Raspberry Pi 5
- une caméra
- une alimentation adaptée
- un dispositif d'affichage

A.9.2 Installation des dépendances

Les bibliothèques nécessaires peuvent être installées sur la plateforme avec :

```
pip install ultralytics opencv-python torch torchvision pillow numpy
```

Listing 5.39 – Installation des bibliothèques sur Raspberry Pi 5

A.9.3 Lancement du système

Après l'installation des dépendances et la préparation de l'environnement d'exécution, le système peut être lancé directement depuis le terminal de la Raspberry Pi 5. Les commandes utilisées sont les suivantes :

```
cd ~/Desktop/Traffic_Sign_Models
source venv/bin/activate
python pi_pipeline.py
```

Listing 5.40 – Lancement du système de détection et de classification

La première commande permet de se positionner dans le répertoire contenant les fichiers du projet :

```
cd ~/Desktop/Traffic_Sign_Models
```

Listing 5.41 – Accès au répertoire du projet

La deuxième commande active l'environnement virtuel Python utilisé par le projet :

```
source venv/bin/activate
```

Listing 5.42 – Activation de l'environnement virtuel

Cette activation permet d'utiliser les bibliothèques et dépendances installées dans l'environnement virtuel `venv`.

Enfin, la commande suivante exécute le script principal du système :

```
python pi_pipeline.py
```

Listing 5.43 – Exécution du pipeline embarqué

Le fichier `pi_pipeline.py` assure l'exécution du pipeline de traitement, depuis l'acquisition des images par la caméra jusqu'à la détection et à la classification des panneaux de signalisation. Le lancement de ce script permet ainsi d'exécuter le système complet sur la plateforme Raspberry Pi 5.

A.9.4 Test du système complet

Après le test individuel de la caméra, le système complet est exécuté selon la chaîne :

```
Camera
|
v
YOLOv8n
|
v
Crop
|
v
MobileNetV2
|
v
Resultat
```

Listing 5.44 – Exécution du système complet

Le test du système complet permet de vérifier :

- la détection du panneau
- l'extraction correcte de la région d'intérêt
- la classification
- le fonctionnement en temps réel
- le nombre d'images traitées par seconde

A.10 Évaluation finale

L'évaluation est réalisée à plusieurs niveaux afin de distinguer les performances de chaque modèle des performances du système complet.

A.10.1 Évaluation de YOLOv8n

Pour le détecteur, les indicateurs considérés sont notamment :

- Precision
- Recall
- mAP
- FPS

Ces indicateurs permettent d'évaluer la capacité du modèle à localiser correctement les panneaux.

A.10.2 Évaluation de MobileNetV2

Pour le classificateur, les indicateurs comprennent :

- Accuracy
- Precision
- Recall
- F1-Score

Ils permettent d'évaluer la capacité du modèle à attribuer la bonne catégorie au panneau détecté.

A.10.3 Évaluation du système complet

Enfin, le système complet est évalué à partir de la chaîne :

```
Camera
|
v
YOLOv8n
|
v
Detection
|
v
Crop
|
v
MobileNetV2
|
```

```
v
Classification
|
v
Resultat final
```

Listing 5.45 – Évaluation du système complet

Cette évaluation permet notamment de vérifier le fonctionnement en conditions proches de l'utilisation réelle.

Sur Raspberry Pi 5, les paramètres suivants peuvent également être observés :

- FPS
- temps de traitement
- température
- consommation énergétique

A.11 Fichiers principaux du projet

Les principaux fichiers utilisés dans la chaîne de traitement sont :

```
yolov8n.pt
data.yaml
best.pt
traffic_sign_detector.pt
traffic_sign_classifier.pth
```

Listing 5.46 – Principaux fichiers du projet

- `yolov8n.pt` : modèle YOLOv8n initial
- `data.yaml` : configuration du dataset de détection
- `best.pt` : poids du meilleur modèle obtenu après entraînement
- `traffic_sign_detector.pt` : modèle YOLOv8n utilisé pour la détection
- `traffic_sign_classifier.pth` : modèle MobileNetV2 utilisé pour la classification.

A.12 Synthèse du fonctionnement

Le fonctionnement global du système peut être résumé comme suit :

```
CAMERA
|
v
Acquisition image
|
v
YOLOv8n
|
v
```

```
Detection panneau
|
v
Extraction crop
|
v
Pretraitement
|
v
MobileNetV2
|
v
Classification
|
v
Resultat final
```

Listing 5.47 – Architecture finale du système

A.13 Business Model Canvas (modèle d'affaire)



FIGURE 5.24 – Business Model Canvas

A.14 Business Plan

Mensuel

	à intégrer ↓ Salaire Base Mensuel	à intégrer ↓ Indemnité Mensuel	Calcul automatique Retenue Sécurité	Calcul automatique Retenue Sécurité	à intégrer ↓ Prime Panier & Transport	Calcul automatique Salaire imposable	Calcul automatique IRG	Calcul automatique Salaire Net Mensuel	Calcul automatique Salaire chargé Annuel
Directeur Général / CEO	0	0	0	0	0	0	0	0	0
Secretaire	$35 000,00	5000	3 150	10 400	5 000	41 850	2 070	39 780	689 640
Commercial	45 000	8000	4 050	13 780	5 000	53 950	4 450	49 500	914 760
delegue	45 000	8000	4 050	13 780	5 000	53 950	4 450	49 500	914 760
ingenieurs	90 000	15000	8 100	27 300	10 000	106 900	16 900	90 000	1 910 400
techniciens	50 000	8000	4 500	15 080	10 000	63 500	5 800	57 700	1 066 560
	0		0	0		0	0	0	0
	0		0	0		0	0	0	0
	0		0	0		0	0	0	0
	0		0	0		0	0	0	0

FIGURE 5.25 – Invesstisement

ASF ALGERIAN STARTUP FUND

Durée d'amortissement	à intégrer ↓ 5 années

	à intégrer ↓ Désignation du matériel	à intégrer ↓ Fonctionnalité du matériel	à intégrer ↓ Prix unitaire	à intégrer ↓ Année 01	à intégrer ↓ Année 02	à intégrer ↓ Année 03	à intégrer ↓ Année 04	à intégrer ↓ Année 05
Matériel N°01	Raspberry Pi 5 (8 Go)	Exécution du système embarqué	77 000	250 000	125 000	125 000		
Matériel N°02	Alimentation, câbles, boîtiers, SD	Assemblage du prototype	10 000	100 000	60 000	60 000		
Matériel N°03	Station de développement (PC)	Entraînement des modèles IA	350 000	350 000	0	0		
Matériel N°04	Caméra HD pour Raspberry Pi	Acquisition des images	8 000	80 000	40 000	40 000		
Matériel N°06	Serveur de stockage (NAS)	Stockage, sécurisation des données	400 000	400 000	0	0		

FIGURE 5.26 – Masse salariale

	N-1 (historique)	Année 01	Année 02	Année 03	Année 04	Année 05
DZD	FY22	FY23	FY24	FY25	FY26	FY27
Chiffre d'affaires	-	7 220 000	11 960 000	14 760 000	20 020 000	25 050 000
Achats consommés	-	(99 000)	(137 500)	(176 000)	(220 000)	(275 000)
Marge brute	**-**	**7 121 000**	**11 822 500**	**14 584 000**	**19 800 000**	**24 775 000**
Charges externes	-	(59 200)	(64 200)	(70 200)	(72 700)	(39 200)
Salaires et charges sociales	-	(5 802 120)	(10 383 480)	(14 275 200)	(15 189 960)	(18 166 920)
EBITDA	**-**	**1 259 680**	**1 374 820**	**238 600**	**4 537 340**	**6 568 880**
Amortissements et provisions		(236 000)	(281 000)	(326 000)	(326 000)	(326 000)
Reprise sur provisions		-	-	-	-	-
EBIT		**1 023 680**	**1 093 820**	**(87 400)**	**4 211 340**	**6 242 880**
Charges financières						
Résultat avant impôts		**1 023 680**	**1 093 820**	**(87 400)**	**4 211 340**	**6 242 880**
Impôts sur les sociétés		-	-	-	-	-
Résultat net		**1 023 680**	**1 093 820**	**(87 400)**	**4 211 340**	**6 242 880**
Evolution du chiffre d'affaires		**-**	***66%***	***23%***	***36%***	***25%***
Tx de marge brute	-	*99%*	*99%*	*99%*	*99%*	*99%*
Tx d'EBITDA	-	*17%*	*11%*	*2%*	*23%*	*26%*

FIGURE 5.27 – Chiffre d'affaires

Charges externes

	à intégrer ↓ Historique (n-1	à intégrer ↓ Année 01	à intégrer ↓ Année 02	à intégrer ↓ Année 03	à intégrer ↓ Année 04	à intégrer ↓ Année 05
DZD	FY22	FY23	FY24	FY25	FY26	FY27
Sous-traitance						
Loyers						
Electricité		30 000	30 000	36 000	36 000	42 000
Frais Marketing		10 000	15 000	15 000	17 500	20 000
Honoraires d'avocat						
Honoraires du Notaire						
Honoraires d'expert-comptable						
Honoraires Commissaire aux Comptes						
Frais du transit						
Frais télécom		1 200	1 200	1 200	1 200	1 200
Divers fournitures		8 000	8 000	8 000	8 000	8 000
Frais de formation						
R&D		10 000	10 000	10 000	10 000	10 000
Autre 1						
Autre 2						
Total	0	59 200	64 200	70 200	72 700	39 200

FIGURE 5.28 – Charges externes

	à intégrer ↓	Année 01	Année 02	Année 03	Année 04	Année 05
Clients						
DZD	DSO (délai de paiement clients)	FY23	FY24	FY25	FY26	FY27
Chiffre d'affaires		7 220 000	11 960 000	14 760 000	20 020 000	25 050 000
Clients en jours de CA	0	0	0	0	0	0
Clients		-	-	-	-	-
Fournisseurs	DSO (délai fournisseur)					
DZD		FY23	FY24	FY25	FY26	FY27
Consommation matières		99 000	137 500	176 000	220 000	275 000
Achats		59 200	64 200	70 200	72 700	39 200
Fournisseurs en jours d'achats	0	0	0	0	0	0
Fournisseurs		-	-	-	-	-
Stocks						
DZD		FY23	FY24	FY25	FY26	FY27
Consommation matières		99 000	137 500	176 000	220 000	275 000
Stocks de MP en jours d'achats consom	60	60	60	60	60	60
Stocks		16 500	22 917	29 333,33	36 666,67	45 833,33

FIGURE 5.29 – Besoin en Fonds de Roulement

	N-1 (historique)	Année 01	Année 02	Année 03	Année 04	Année 05
DZD	FY22	FY23	FY24	FY25	FY26	FY27
Chiffre d'affaires	-	7 220 000	11 960 000	14 760 000	20 020 000	25 050 000
Achats consommés	-	(99 000)	(137 500)	(176 000)	(220 000)	(275 000)
Marge brute	**-**	**7 121 000**	**11 822 500**	**14 584 000**	**19 800 000**	**24 775 000**
Charges externes	-	(59 200)	(64 200)	(70 200)	(72 700)	(39 200)
Salaires et charges sociales	-	(5 802 120)	(10 383 480)	(14 275 200)	(15 189 960)	(18 166 920)
EBITDA	**-**	**1 259 680**	**1 374 820**	**238 600**	**4 537 340**	**6 568 880**
Amortissements et provisions		(236 000)	(281 000)	(326 000)	(326 000)	(326 000)
Reprise sur provisions		-	-	-	-	-
EBIT		**1 023 680**	**1 093 820**	**(87 400)**	**4 211 340**	**6 242 880**
Charges financières						
Résultat avant impôts		**1 023 680**	**1 093 820**	**(87 400)**	**4 211 340**	**6 242 880**
Impôts sur les sociétés		-	-	-	-	-
Résultat net		**1 023 680**	**1 093 820**	**(87 400)**	**4 211 340**	**6 242 880**
Evolution du chiffre d'affaires		***-***	***66%***	***23%***	***36%***	***25%***
Tx de marge brute	-	*99%*	*99%*	*99%*	*99%*	*99%*
Tx d'EBITDA	-	*17%*	*11%*	*2%*	*23%*	*26%*

FIGURE 5.30 – compte de résultat

	Année 01	Année 02	Année 03	Année 04	Année 05
DZD	**FY23**	**FY24**	**FY25**	**FY26**	**FY27**
EBITDA	1 259 680	1 374 820	238 600	4 537 340	6 568 880
Variation de BFR	**(16 500)**	**(6 417)**	**(6 417)**	**(7 333)**	**(9 167)**
BFR d'exploitation	16 500	22 917	29 333	36 667	45 833
BFR hors exploitation					
BFR	16 500	22 917	29 333	36 667	45 833
IBS	-	-	-	-	-
Flux de trésorerie provenant de l'exploitation	**1 243 180**	**1 368 403**	**232 183**	**4 530 007**	**6 559 713**
CAPEX (Investissements)	(1 180 000)	(225 000)	(225 000)	-	-
Flux de trésorerie provenant de l'investissement	**(1 180 000)**	**(225 000)**	**(225 000)**	**-**	**-**
Free cash flow	**63 180**	**1 143 403**	**7 183**	**4 530 007**	**6 559 713**
Emprunt bancaire					
Apport en capital					
Charges financières	-	-	-	-	-
Dividendes					
Flux de trésorerie provenant du financement	**-**	**-**	**-**	**-**	**-**
Net Cash flow	63 180	1 143 403	7 183	4 530 007	6 559 713
Solde initial	**-**	**63 180**	**1 206 583**	**1 213 767**	**5 743 773**
Solde final	**63 180**	**1 206 583**	**1 213 767**	**5 743 773**	**12 303 487**

FIGURE 5.31 – Flux de Trésorerie