الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية العلوم
Faculté de Sciences

قسم الإعلام الآلي
Département d'Informatique

Mémoire de Projet de Fin d'Études

Présenté par

Camélia BADAOUI

Kenza BENYAKOUB

Pour l'obtention du diplôme de Master en Informatique

Spécialité : Ingénierie des Logiciels

Thème

Détection des véhicules immobilisés sur les rails d'un passage à niveau à l'aide de YOLO

Proposé par :

Fayçal YKHLEF, Maître de Recherche A, CDTA, Alger.

Co-promoteur

Hadjer YKHLEF, Maitre de Conférences B, Université de Blida 1, Blida

Année Universitaire 2024-2025

REMERCIEMENTS

Nous exprimons notre gratitude à Allah le tout puissant pour nous avoir accordé la santé et la volonté de mener à bien ce projet, El Hamdoullillah !

Ce projet a pu être mené à terme grâce à l'assistance de nombreuses personnes à qui nous adressons nos sincères remerciements.

Tout d'abord, nous remercions chaleureusement notre promoteur, le Dr. Fayçal YKHLEF, Maître de Recherche A, Directeur de la Division Architecture des Systèmes et Multimédia (ASM) au Centre de Développement des Technologies Avancées (CDTA), pour sa disponibilité et ses précieux conseils qui ont grandement contribué à nos avancées, pour son orientation, sa confiance en nous et son expertise dans le domaine. Grâce à ses encouragements et ses critiques constructives, nous avons pu repousser nos limites et trouver des solutions novatrices.

Nous tenons à exprimer notre gratitude envers à notre Co-promoteur, le Dr. Hadjer YKHLEF, Maître de Conférences B à l'université Saad Dahleb 1, pour sa disponibilité, sa précieuse contribution, Ses conseils et son expertise.

Nous souhaitons également remercier l'ensemble du personnel et des chercheurs du CDTA, en particulier ceux de la division ASM, pour leur accueil chaleureux et leur soutien.

Nos remerciements vont également aux membres du jury qui ont accepté d'évaluer notre travail et de l'enrichir de leurs connaissances et suggestions.

Nous tenons à exprimer notre reconnaissance envers le corps professoral et administratif de l'université Saad Dahleb 1, qui a contribué à notre réussite universitaire.

Nous tenons à remercier tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce projet. Merci du fond du cœur.

DEDICACE

Je dédie ce modeste travail à ceux qui m'ont donné la vie est on fait de moi celle que je suis aujourd'hui, mes chers parents FARIDA et NOUREDDINE.

A ma lumière, mon idole, celle qui a veillé à mes côtés depuis mon plus jeune âge jusqu'à ce jour. Maman tu n'es pas seulement une mère pour moi, mais aussi une meilleure amie, une sœur de cœur, mon pilier, mon refuge, mon tout. Sache que tous les mots du monde ne suffiraient pas à décrire à quel point je t'aime.

Au roi de mon cœur, mon seul et unique amour, PAPA, l'épaule solide sur laquelle je peux toujours compter, tu as su me guider tout au long de mon parcours et a su me soutenir, même quand on n'était pas du même avis, de ta petite chouchoute qui t'aime plus que tout au monde.

A toi RAYANE, mon frère de sang, celui qui partage mes rires, mes souvenirs. Pour toi j'avance, je grandis, je préserve. Ce travail, Je te le dédie, comme un pas de plus sur le chemin que je veux tracer pour nous deux.

A mes chères grand-mères, Anissa et Djamila, vous mains ont bercé mon enfance, vos prières ont protégé mes pas. Dans vos regards, j'ai vu l'amour pur, dans vos gestes la douceur qui rassure. Vous êtes mes racines, ma force, les fleurs vivantes de mon arbre de vie.

A mes défunts grands-pères, YOUCEF et BELKACEM, que la paix enveloppe vos âmes, comme vos bras m'ont enveloppée autrefois. Vos souvenirs vivent en moi, vos valeurs me guident à chaque pas, même si vos voix se sont tues, votre présence demeure douce et silencieuse.

A vous, cher Docteur TOUHAMI, il y a des gestes qui marquent une vie, le vôtre en fait partie. Lorsque mes pas étaient incertains, vous m'avez tendu la main. Par votre savoir, votre humanité, vous m'avez offert bien plus qu'une opération : vous m'avez rendu la liberté, celle de marcher, de me tenir droite, de vivre pleinement sans douleur et sans crainte. Ce que vous a fait pour moi dépasse les mots. Ce travail, ce parcours, ce souffle retrouvé, je vous le dédie du fond du cœur.

A mes meilleures amies, WAHIBA, ICHRAK, SARAH, FAIZA, celles qui ont séché mes larmes, encouragée et partagé mes plus beau souvenirs mes rêves

Camélia

DEDICACE

Au nom de Dieu, le Tout-Puissant, et de Son prophète Mohamed, que la paix soit sur lui.

Louange à Dieu, qui m'a permis d'atteindre ce jour tant attendu. Par ces quelques modestes mots, je souhaite exprimer ma profonde reconnaissance à tous ceux qui m'ont soutenue dans la réalisation de ce travail.

Je dédie ce modeste travail aux êtres les plus chers à mon cœur, qui m'ont accompagnée tout au long de mes études.

À l'homme à qui je dois ma vie, ma réussite et tout mon respect : mon très cher père, «Ben Yakoub Salah». Parmi tous les pères, tu es le meilleur. Grâce à toi, papa, j'ai appris la valeur du travail et de la responsabilité. Tes conseils ont toujours guidé mes pas vers la réussite. Tu es et resteras pour moi une référence, une lumière qui éclaire mon chemin. J'espère que tu trouveras dans ce travail le fruit de tes efforts et le témoignage de ma grande fierté de t'avoir comme père. Je te dois ce que je suis aujourd'hui et ce que je serai demain, et je ferai toujours de mon mieux pour rester ta fierté et ne jamais te décevoir.

À ma chère mère, «Ikache Fatiha», tu m'as comblée de ta tendresse et de ton affection tout au long de mon parcours. Tu n'as cessé de me soutenir et de m'encourager durant toutes mes années d'études. Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études et tout au long de ma vie. En ce jour mémorable, pour moi comme pour toi, reçois ce travail en signe de ma vive reconnaissance et de ma profonde estime. Puisse le Tout-Puissant te donner santé, bonheur et longue vie afin que je puisse, à mon tour, te combler. J'espère ne jamais te décevoir ni trahir ta confiance et tes sacrifices.

À ma belle-sœur, «Rania», aucune dédicace ne peut exprimer tout mon amour et ma gratitude. Tu es une sœur qui a su jouer son rôle comme il se doit, et tu as été à mes côtés pendant toutes les étapes de ce travail. Je t'en suis très reconnaissante. Je te dis merci.

Kenza

Résumé

Ce mémoire s'intéresse à la sécurisation des passages à niveau par la détection et le suivi en temps réel de deux catégories d'objets critiques : les piétons et les véhicules immobilisés sur les rails. Nous proposons une approche basée sur des variantes de l'algorithme YOLOv8 (You Only Look Once) pour la détection et la classification d'objets, combiné à DeepSORT (Deep Simple Online and Realtime Tracking) pour le suivi multi-objets. Notre approche a été évaluée sur la base de données COCO2017, répartie selon une proportion de 75% pour l'entraînement, 15% pour la validation et 15% pour les tests. Une augmentation des données a été appliquée dynamiquement durant l'apprentissage. Après optimisation des hyper-paramètres, le modèle a atteint les performances suivantes : Précision : 82,42%, Rappel : 75,27%, F1-score : 78,68%, mAP@50 : 82,61%, mAP@50-95 : 66,45%. Une interface graphique à base de python été conçue pour capturer le moindre mouvement.

Mot clé : Détection, classification, poursuite, YOLOv8, DeepSORT, apprentissage profond, COCO2017, en temps réel.

Abstract

This thesis focuses on securing level crossings by detecting and tracking in real time two categories of critical objects: pedestrians and vehicles immobilized on the rails. We propose an approach based on variants of the YOLOv8 (You Only Look Once) algorithm for object detection and classification, combined with DeepSORT (Deep Simple Online and Realtime Tracking) for multi-object tracking. Our approach was evaluated on the COCO2017 database, split into 75% for training, 15% for validation, and 15% for testing. Data augmentation was dynamically applied during training. After hyperparameter optimization, the model achieved the following performance: Precision: 82.42%, Recall: 75.27%, F1-score: 78.68%, mAP@50: 82.61%, mAP@50-95: 66.45%. A Python-based graphical interface was designed to capture even the slightest movement.

Keywords: Detection, classification, tracking, YOLOv8, DeepSORT, deep learning, COCO2017, real-time.

ملخص

تركز هذه الرسالة على تأمين معابر السكك الحديدية من خلال الكشف عن فئتين من الأجسام الحرجة وتتبعها في الوقت الفعلي: المشاة والمركبات المتوقفة على القضبان. نقترح نهجًا يعتمد على متغيرات خوارزمية YOLOv8 (You Only Look Once) للكشف عن الأجسام وتصنيفها، جنبًا إلى جنب مع DeepSORT (Deep Simple Online and Realtime Tracking) لتتبع العديد من الأجسام. تم تقييم نهجنا على قاعدة بياناتCOCO2017 ، وتقسيمه إلى 75% للتدريب، و15% للتحقق، و15% للاختبار. تم تطبيق زيادة البيانات ديناميكيًا أثناء التدريب. بعد تحسين المعاملات الفائقة، حقق النموذج الأداء التالي: الدقة: 82.42%، التذكر: 75.27%، درجةF1 78.68 %، درجةmAP@50 82.61 %، درجة mAP@50-95 66.44 % .تم تصميم واجهة رسومية قائمة على بايثون لالتقاط حتى أدنى حركة.

الكلمات المفتاحية: الكشف، التصنيف، التتبع، YOLOv8، DeepSORT، التعلم العميق، COCO2017، الوقت الحقيقي.

LISTE DES ACRONYMES ET ABREVIATIONS

ANN : *Artificial neural networks (Réseau de neurones artificiels)*

BCE : Entropie croisée binaire

CNN *: Convolutional neural network (Réseau de neurones convolutifs)*

COCO: *Common Objects in Context*

CPU *: Central processing unit (Unité centrale de traitement)*

CUDA : *Compute Unified Device Architecture*

DeepSORT: *Deep simple Online and Realtime Tracking (Algorithme de suivi multi-objets en temps réel)*

F1-Score : Mesure harmonique de la précision et du rappel

FN : *False Negative (Faux Négatif)*

FP : *False Positive (Faux Positif)*

FPN : *Feature Pyramid Networks*

FPS : *Frame per seconde (Images par seconde)*

GPU : *Graphics processing unit (Unité de traitement graphique)*

IA : Intelligence Artificielle

IoT : *Internet des Objets*

IoU : *Intersection over Union (Intersection sur Union)*

JSON : Format de données JavaScript

LiDAR : *Light detection and ranging*

mAP : Précision Moyenne

mAR : Rappel Moyen

ONNX : *Format d'échange de modèles de réseaux de neurones*

PANet : *Path Aggregation Network*

Radar : *Radio detection and ranging*

R-CNN : *Regions with CNN features (Réseau de propositions régionales)*

ROI : Régions d'intérêt

SIFT: *Scale-Invariant Feature Transform (Transformation de caractéristiques invariantes à l'échelle)*

SNCF : Société National de France Chemin de fer

SNTF : Société National des Transports Ferroviaires

SPP : *Spatial Pyramid Pooling*

SSD : *Single Shot MultiBox Detector (détecteur d'objets à tir unique)*

SURF : *Speeded-Up Robust Features (Fonctionnalités robustes accélérées)*

SVM : *Support Vector Machine*

TN : *True Negative (Vrai Négatif)*

TP : *True Positive (Vrai Positif)*

YAML : *Yet Another Markup Language (format de fichier de configuration)*

YOLO : *You Only Look Once (Détecteur d'objet en une seul Passage)*

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

Chapitre 1: Introduction

1.1 Motivations

Un passage à niveau est un croisement, au même niveau entre une voie ferrée et une voie routière ou piétonnière. Depuis de nombreuses années, ces intersections représentent des zones à grand risque, notamment en raison de l'absence de panneaux de signalisations, du non-respect de ces derniers, par des comportements humains imprudents de la part des usagers ; secondaires à du stress des automobilistes voire même de manière irréfléchie ou mal intentionnée.

D'autres risques peuvent être engendrés suite à des circonstances particulières, comme des défaillances techniques, propres à la signalisation, à la mécanique des automobiles ; mais aussi à l'encombrement des voies ferrées, secondaires à des embouteillages, bloquant la sortie du passage à niveau ou même de constructions illicites près des rails. Ces facteurs aggravent le risque des accidents.

Ces accidents ont des conséquences humaines telles que, perte de vies, de blessures graves, de chocs psychologiques et émotionnels ; de conséquences économiques telles que, des dégâts matériels.

En Algérie, la situation est préoccupante, d'après les informations fournies par la Société Nationale des Transports Ferroviaires (SNTF) et les responsables du transport ; qui ont établi 77 collisions en 2022, entraînant 6 morts et 47 personnes blessées. En 2023, une hausse significative a été constatée ; avec 85 accidents, causant 7 morts et 39 personnes blessées [1].

Sur le plan économique l'année 2022, a été particulièrement désastreuse ; puisque les dépenses totales pour la réparation des équipements ferroviaires endommagés (train, infrastructure et la gestion des retards causés par ces incidents) ont franchi la barre des onze millions de dinars [1].

Face à cette réalité, pour qu'il n'y ait pas de drames ; il serait souhaitable de créer, des technologies aptes à détecter ses situations dangereuses (voitures immobilisées, circulations embouteillées). Aussi, un développement performant de l'intelligence artificielle (IA) ainsi que de la vision par ordinateur, permettent d'acquérir des outils puissants pour engendrer des systèmes de surveillance intelligents. Parmi eux, l'algorithme You Only Look Once (YOLO), basé sur des réseaux de neurones convolutifs ; qui permet la détection et la classification d'objet, en temps réel, à partir du flux d'images.

De multiples travaux étrangers [2] [3] [4] [5] [6] ont été proposés dans le monde de la recherche, impliquant la détection de véhicules immobilisés, sur les rails d'un passage à niveau, à l'aide de YOLO. A notre connaissance, aucun système associant ces technologies, n'a été déployé en Algérie, jusqu'à présent.

Ce mémoire a pour but de répondre à cette problématique cruciale. Aussi notre ambition est de contribuer à la protection des vies humaines et de participer à l'amélioration des performances des technologies utilisées par l'IA, avec pour conséquence, un fort impact sociétal.

1.2 Contributions

L'intérêt de ce mémoire, est que nous puissions concevoir, un système de vidéosurveillance, capable de détecter, en temps réel, les véhicules immobilisés sur les rails d'un passage à niveau. Le choix d'utiliser l'algorithme YOLO, plus précisément la version 8, réputé pour sa rapidité d'exécution et sa précision [7] ; a été argumenté d'une part, en raison de sa performance dans la détection d'objets de petites taille ou partiellement visible [8] ; d'autre part, en raison d'une détection sans ancrage ; comparativement aux versions antérieures (YOLOv5, YOLOv6, YOLOv7) [7] [9].

Notre contribution se décline en plusieurs volets, en premier lieu, l'amélioration du modèle YOLOv8, en le spécialisant sur deux classes : « personne » et « voiture » ; optimisant ainsi sa précision et son rappel. Ce modèle est ainsi mieux adapté à la détection des véhicules immobilisés et des personnes aux passages à niveau.

Ensuite, un processus en fonction du temps a été mis en œuvre, pour mesurer la cinétique des véhicules, afin de déterminer s'ils sont en arrêt ou non ; dans le cas d'une immobilisation prolongée sur les voies, le système d'identification déclenche automatiquement une alerte par signal afin d'avertir le conducteur de la locomotive. Cela constitue un mécanisme de prévention fiable.

Par ailleurs, nous avons développé une interface graphique conviviale (GUI), en utilisant le langage Python ; qui permet de visualiser le trafic circulatoire automobile et de capturer le moindre mouvement suspect des sujets, évoluant dans le champ de vision de la caméra afin de générer des alertes de prévention. Ce logiciel, simple et efficace sera utilisé, de manière adéquate, par un personnel qualifié, facilitant au maximum la surveillance. Malgré des caractéristiques moyennes de nos PC, telles qu'un processeur Intel i5 à 2,60 GHz, une RAM de 8 Go, et l'absence de carte graphique dédiée, le système à bien fonctionné, assurant une

exécution fluide et efficace. L'ensemble de ce prototype fonctionnel représente une étape, vers l'implémentation d'un système déployable sur le terrain.

1.3 Impact du projet

La réalisation de notre projet de fin d'études, pourrait avoir des répercutions majeurs à divers niveaux. Sur le plan humain, la sauvegarde de vie humaine, en diminuant le nombre d'accidents mortels.

Sur le plan économique, une limitation des pertes matérielles, engendrées par des accidents et un coût financier amoindrie, pour les réparations d'équipements endommagés ainsi que des charges liées, pour la continuité du service ferroviaire et routier.

Sur le plan technologique et scientifique, ce projet permet d'adapter les performances de l'IA à notre problématique, concernant la sécurité des voies ferrées. Il met en évidence, l'utilité du Deep leaning comme YOLO ; tout en ouvrant la voie à d'autres applications similaires dans des lieux opérationnels. A noter, qu'une interface imagée conviviale, est beaucoup plus agréable et mieux tolérée par les utilisateurs, qui appréhendent quelque peu, les nouvelles technologies.

1.4 Organisation du mémoire

Ce mémoire est structuré en six chapitres, définit comme suit :

Le premier chapitre, comprend les motivations, les contributions ainsi que l'impact du projet lié à la sécurité ferroviaire en Algérie.

Le deuxième chapitre, évoque la problématique de notre sujet, les technologies en usages, et les synthèses d'études existantes, en rapport avec notre système.

Le troisième chapitre, est consacré aux techniques d'apprentissage pour la détection d'objets. Il traite les algorithmes de détection, allant des méthodes traditionnelles aux techniques récentes ; ainsi que les métriques d'évaluation liées à l'algorithme YOLO.

Le quatrième chapitre, décrit les particularités de l'ensemble de données. Il détaille les étapes de préparation de données, la procédure de détection et de suivi pour les objets d'intérêt de notre projet.

Le cinquième chapitre, présente les outils de développement, les résultats obtenus ainsi qu'une présentation de l'interface graphique créée.

Le sixième et dernier chapitre, conclut le mémoire, en évoquant les apports contributifs, en suggérant de nouvelles améliorations pour des recherches à l'avenir.

Chapitre 2 : Technologies de détection d'obstacles sur les rails d'un passage à niveau

2.1 Introduction

Dans ce chapitre, nous commencerons par présenter un passage à niveau, en décrivant ses caractéristiques et ses enjeux. Ensuite, nous allons explorer les différentes technologies de manière comparative, tels que les capteurs actifs et passifs, afin d'améliorer la sécurité dans ces lieux de croisement et de trouver la meilleure solution de détection d'obstacles, adaptée aux contraintes des passages à niveau. Le fonctionnement d'un système de détection sera également détaillé.

2.2 Passages à niveau

2.2.1 Définition

Les passages à niveau (figure 2-1) symbolisent des zones de croisement cruciales entre les routes et les rails de train. Ils sont visibles au quotidien, lorsqu'on entreprend des voyages et des déplacements entre plusieurs villes et métropoles. Ils ne possèdent pas d'infrastructures de séparation, comme des ponts ou des tunnels. Aussi, l'absence de séparation physique aboutit à une interaction directe entre trains, véhicules routiers et piétons ; ce qui rend ces lieux, particulièrement sensibles du point de vue sécuritaire [10]. Annuellement, ces croisements sont le théâtre de multiples accidents, fréquemment graves et parfois mortels.

Figure 2-1 : Passage à niveau.

Aussi, des systèmes de signalisation des passages à niveau ont été créés pour sécuriser ces lieux à risques [11] ; parmi eux ;

- **Feux tricolors et clignotants**

Les feux tricolores et les signaux lumineux, scintillants, au niveau des intersections afin d'avertir l'arrivée du train et alerter les conducteurs et piétons.

- **Alarmes sonores**

Les alarmes sonores sont un système d'avertissement qui lance automatiquement un signal sonore. Celui-ci est déclenché par des capteurs qui détectent le train à une certaine distance, pour informer les usagers.

- **Barrières automatiques**

Une barrière automatique est une barre horizontale, utilisée pour contrôler la circulation. Elle assure une sécurité et une gestion efficace du trafic aux entrées et sorties.

Dans les lieux à risque dominant, il est prévu l'aménagement de barbières automatiques barrant la route à l'approche d'un express.

- **Panneaux de signalisation**

Des panneaux règlementaires doivent être installés, pour informer les automobilistes et passants de la présence d'un passage à niveau. Ces panneaux incluent :

- Avertissement d'un passage à niveau.
- Arrêt obligatoire avant un passage à niveau.
- Feux à clignotants.

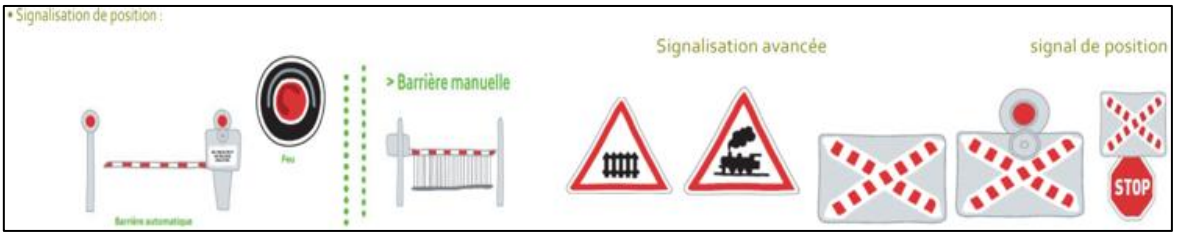


Figure 2-2 : Panneaux de signalisation *[11]*.

Les accidents aux passages à niveau découlent de nombreux éléments qui peuvent être regroupés en trois catégories majeures :

➢ **Facteurs liés à l'homme**

- Certains usagers ignorent les signaux automatiques et franchissent les barrières de sécurité par exemple, en Algérie, les causes des accidents sont en grande partie attribuées au comportement humain avec 90%.
- L'absence de respect des signaux de sécurité, comme le montre la figure (2-3) : par imprudence ou précipitations, de nombreux conducteurs et piétons ignorent les

feux rouges, les barrières fermées et alertes sonores ; 98% des accidents sont liés au non-respect du code de la route [1].

Figure 2-3 : Non-respect des signaux de sécurité *[12]*.

- Les erreurs humaines : l'origine de la plupart des accidents est due à l'inattention des chauffeurs et le fait d'être distrait (usage du téléphone, distractions, excès de vitesse) [13].

Figure 2-4 : Distraction d'un usager.

➢ **Facteurs techniques**

- Restriction liées aux trains : pour éviter la présence d'un obstacle sur la voie, il faut que le train qui circule à 80km/h, ait besoin de presque 800 mètres pour s'immobiliser. En comparaison, une automobile roulant à la même vitesse peut s'arrêter en moins de 100 mètres, ce qui illustre bien la vulnérabilité des passages à niveau [14].
- Arrêt des véhicules sur les voies ferrées : des embouteillages et des pannes techniques (pannes moteurs) peuvent immobiliser un véhicule sur la voie ; créant un risque direct et majeur de collision avec un train ; comme illustrée ci-dessous :

Figure 2-5 : Panne d'un véhicule.

➢ **Facteurs liés aux infrastructures**

- Infrastructures incomplètes : fautes de moyens financiers, quelques passages à niveau n'ont pas de système de signalisation, de barrières automatiques ou de détection modernes ; ce qui diminue leur performance en terme de prévention des accidents.
- Insuffisance de la signalisation efficace : une signalisation défectueuse peut induire les conducteurs en erreur et entraîner les accidents. ; comme le cas de cette figure.

Figure 2-6 : Passage à niveau dépourvu d'infrastructures *[15]*.

2.2.2 Causes des véhicules immobilisés sur les rails

➢ **Pannes mécaniques**

- des accidents étaient dus, à des pannes mécaniques (des pannes de moteurs, de batterie…etc.).

➢ **Embouteillages bloquant la sortie du passage à niveau**

- Les voies ferrées encombrées sont causées par des embouteillages et par une circulation anarchique.

➢ **Mauvaise visibilité et stress ou panique du conducteur**

- L'immobilisation des véhicules et des piétons, sur les rails du chemin de fer est causée par de la mauvaise visibilité, qui est due aux conditions climatiques (brouillard, pluie, obscurité). Un chauffeur inexpérimenté ou stressé, par l'arrivée soudaine d'un train, peut réagir de manière inappropriée [14].

2.2.3 Accidents aux passages à niveau en Algérie : enjeux et prévention

A. Cause des accidents aux passages à niveau : Analyse et cas concrets

En Algérie, les facteurs humains et infrastructurels sont, principalement responsables des accidents aux passages à niveau [1]. En voici, les causes les plus courantes :

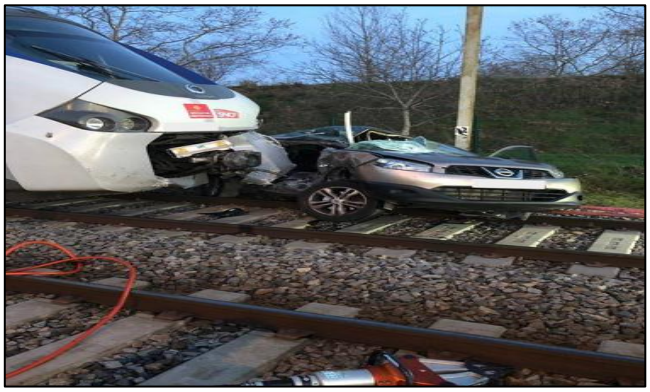


Figure 2-7 : Scène d'accident train contre voiture sur les rails.

❖ **Absence de respect des signaux et imprudence des usagers**

Certains automobilistes et piétons ne considèrent, aucune barrière baissée, ni les feux de stop et encore moins les signaux sonores ; mettant ainsi leur sécurité en péril. En mars 2024, à Hai Faoussi Ahmed (ex-Cité Ali Mondo), Réghaia, deux personnes ont été, mortellement fauchées par un train après avoir traversé, sans respecter la signalisation. Ce drame souligne l'importance de la sensibilisation des usagers [1].

❖ **Comportement à risque**

Parfois, certains conducteurs choisissent intentionnellement, d'accélérer pour traverser, avant l'arrivée du train, ce qui accroît le danger d'une collision.

❖ **Défaillance des équipements de sécurité**

Malheureusement, il existe certains passages à niveau, qui manquent de barrière automatiques ou d'une signalisation adéquate ; diminuant ainsi leur visibilité et leur degré de sécurité. A titre d'exemple : en avril 2024, à Réghaia, un père et son enfant autiste, ont perdu la vie après s'être trop approchés des rails. Ce cas tragique illustre l'urgence de mettre en place, des passerelles adaptées et des mesures de protection renforcées [1].

❖ **Encombrement des voies ferrées**

La défaillance mécanique ou l'engorgement, cause le blocage des véhicules sur les voies ferrées, ce qui augmente le risque d'accidents.

❖ **Construction illicites et mauvaise visibilité**

La réduction de la visibilité des conducteurs et des piétons est due, à des constructions d'habitations illicites et de commerces trop proches des voies, ce qui provoque des risques pour les usagers [1].

❖ **Infrastructure insuffisantes près des écoles**

L'absence de passages sécurisés, près des établissements d'enseignements (collèges et lycées), expose les enfants à des risques considérables. A Ain Defla et Khemis Miliana, plusieurs écoliers ont été victimes de collisions, faute d'infrastructures adaptées [1].

Ce problème structurel, souligne la nécessité d'une meilleure planification urbaine, pour sécuriser ces zones sensibles.

B. Conséquence des accidents : impact humain, économique et psychologique

Les accidents aux passages à niveau ont des répercussions grave et multiple, qui vont bien au-delà des pertes humaines. Ces conséquences peuvent être classées en trois catégories principales : impact humain, économique et psychologique ; comme représenté dans la figure ci-dessous :

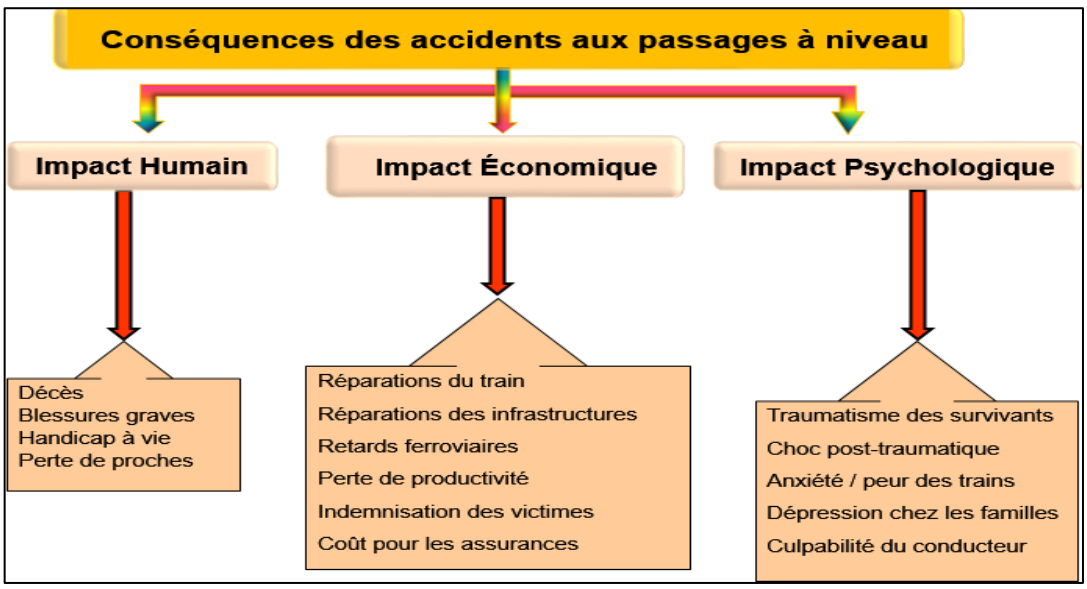


Figure 2-8 : Conséquence des accidents aux passages à niveau.

2.3 Détection d'obstacles

Face à ces enjeux, Il est important de mettre en avant la priorité, à la sécurisation de ces croisements ; afin de réduire au maximum les risques d'accidents, en lieu des passages à

niveau. Malgré, des compagnes de sensibilisation et de formations techniques, menées régulièrement, les incidents se produisent fréquemment.

La détection d'obstacles sur les rails, présente plusieurs défis majeurs. Tout d'abord, les systèmes doivent fonctionner en temps réel, afin de réagir rapidement, surtout que les trains lancés à grande vitesse, nécessitent une longue distance pour s'arrêter. Ensuite, les technologies employées doivent être fiables et précis, capable de distinguer un véritable obstacle d'un faux signal [3].

Aujourd'hui, l'amélioration de la surveillance et la détection de comportements dangereux, aux passages à niveau, d'une façon intemporelle, existent grâce à l'intelligence artificielle et à la vision informatique. En France, afin d'analyser, les vues en temps réel ; la National Société de France Chemin de fer (SNCF) utilise des algorithmes de vision numérique ; qui peuvent détecter automatiquement les dangers éventuels [16].

De même, l'intégration de capteurs intelligents et de système interconnectés, facilite bien l'identification des obstacles et donc de situations dangereuses. Par voie de conséquence, l'envoi d'alerte automatique, réduisant le temps de réaction, en cas d'accidents ; permet la sauvegarde de plusieurs vies humaines. Ces avancées ouvrent de nouvelles solutions, pour renforcer la sécurité et réduire significativement le nombre d'accidents.

Cependant, la situation est plus complexe dans des environnements, comme celui de l'Algérie, où plusieurs passages à niveau présentent un déficit d'équipements modernes, et où les ressources de technologies et financières peuvent être restreintes. Il faut alors penser à des solutions, qui soient à la fois technologiquement performants mais aussi financièrement acceptables.

Aussi, il serait souhaitable de concevoir un système de vidéosurveillance, pour la détection d'obstacles aux passages à niveau ; capable de fonctionner et de traiter les données, en temps réel, et surtout d'envoyer des alertes automatiques.

2.3.1 Technologies en usages

A l'heure actuelle, des systèmes de détection d'obstacles destinés à identifier en temps réel et signaler la présence de tout obstacle sur les voies, qu'il soit un véhicule un piéton ou tout autre objet.

a. Comparaison entre capteurs actifs et passifs

Une première étude, faite d'après l'article intitulé « An In-depth Comparison of LiDAR, Camera, and Radar Technologies », publié en février 2024, sur le site Edge AI and Vision

Alliance [17], offre une analyse approfondie des trois principaux capteurs de détection, utilisées dans les systèmes de surveillance des passages à niveau, il les a classés en deux catégories :

Les capteurs actifs émettent de l'énergie (par exemple des ondes radio ou de la lumière laser) et mesurent le signal réfléchi ou diffusé ; tandis que les capteurs passifs détectent le rayonnement naturel ou l'émission de la cible ou de l'environnement (par exemple la lumière du soleil ou la lumière artificielle pour les caméras).

2.3.2 Comparaison entre caméra, Radar et Lidar

Une seconde étude, faite d'après l'étude de Hesai Technology (l'entreprise chinoise spécialisé dans le développement et la fabrication du LiDAR) [18] ; qui compare les capteurs (Caméra, Radar à ondes millimétriques et LiDAR).

- **Caméra (capteur passif)**

Peut capturer des couleurs riches et des informations détaillées, comparable à l'œil humain, néanmoins peut avoir du mal, à visualiser clairement et perdre la trace des cibles, dans des environnements de faible luminosité ou à contre-jour, ce qui nuit à la fiabilité de détection [18].

Figure 2-9 : Voie ferrée équipée de systèmes de surveillance**.**

- **Radar à ondes millimétriques (capteur actif)**

Le Radar est un capteur actif (voir la figure en-dessus) qui utilise la bande des ondes millimétriques pour la télémétrie, la détection, le suivi et l'imagerie ; performant dans des environnements difficiles (brouillard, poussière), il détecte les obstacles mais souffre d'une résolution plus faible, limitant la précision de l'identification (sa faible résolution rend difficile, la détection précise de petits objets comme les piétons et les vélos) [18].

Figure 2-10 : Système radar utilisé pour la détection d'obstacles.

- **LiDAR (capteur actif très précis)**

Le LiDAR (voir la figure 2-11) est une technologie de télédétection qui utilise des faisceaux laser, pour mesurer des distances et détecter des mouvements précis, en temps réel ; contrairement, au radar, qui lui utilise des ondes radio. En émettant des millions de petits lasers par seconde, ces faisceaux laser impactent les objets environnants et reviennent vers la source émettrice, délimitant ainsi ses même objets comme les véhicules et les piétons ; avec une précision extrême et une résolution, comparable à celle d'une image parfaite. Cependant, ses performances peuvent être affectées dans une certaine mesure, par des conditions météorologiques extrêmes, comme de fortes pluies, de la neige ou du brouillard [18].

Figure 2-11 : Passage à niveau équipé d'un LiDAR.

Tableau 2-1 : Analyse comparative des capteurs passifs de détection *[17] [19] [20]*.

Capteurs	**Type**	**Avantages**	**Inconvénients**	**Coût**	**Domaine d'application**
Capteur Infrarouge (IR)	Passif	**.** Détection dans l'obscurité complète. **.** Sensible à la chaleur, utile pour différencier les êtres vivants.	**.** Sensibilité réduite en plein jour ou sous forte lumière. **.** Moins efficace pour les objets froids ou stationnaires. **.** Portée limitée, ce qui réduit son utilité pour la détection à longue distance.	Modéré	**.** Détection nocturne. **.** Surveillance thermique.
Fibre Optique	Passif	**.** Résistant aux conditions climatiques extrêmes. **.** Peu d'entretien, durable sur le long terme. **.** Détection des vibrations et changements de température.	**.** Ne détecte pas directement les objets, mais les vibrations ou changements. **.** Pas de retour visuel ou de reconnaissance d'objets.	Très élevé (Installation coûteuse pour de grandes zones).	Surveillance de vibrations.
Caméra	Passif	**.** Images claires pour une identification précise. **.** Intégration facile avec des systèmes de vision par ordinateur. **.** Analyse en temps réel possible.	**.** Sensible aux conditions météo (pluie, brouillard) et d'éclairage. **.** Maintenance élevée en cas de saleté ou d'obstructions. **.** Nécessite un traitement important des données. **. S**oulève des informations confidentielle (n'importe qui peut être détecté ou afficher).	Économique	Surveillance, reconnaissance d'objet.
Vision par ordinateur	Passif	**.** Analyse avancée en temps réel avec IA. **.** Apprentissage continu pour améliorer la détection. **.** Détection de comportements anormaux.	**.** Dépend de la qualité des images (sensibilité à la lumière et aux conditions météo). **.** Forte puissance de calcul requise. Formation des modèles coûteuse et exigeante en données.	Modéré à élevé (des algorithmes de traitement de données avancé sont nécessaires).	Reconnaissance d'objets, détection de comportements

Tableau 2-2 : Analyse comparative des capteurs actifs de détection *[17] [21]*.

Capteurs	Type	Avantages	Inconvénients	Coût	Domaine d'application
Radar	Actif	. Fonctionne dans toutes les conditions météo et d'éclairage. . Détection de la vitesse des objets en mouvement. . Moins cher que le LiDAR pour les applications à courte portée.	. Résolution inférieure, difficile de distinguer les types d'objets. . Interférences possibles avec d'autres dispositifs radar.	Modéré	Détection de vitesse, surveillance à distance
LiDAR	Actif	. Haute précision et résolution, cartographie 3D des obstacles. . Fonctionne dans diverses conditions de lumière. . Détection des objets stationnaires et en mouvement.	. Sensible aux conditions météo (pluie, neige, brouillard, poussière).	Coût élevé (des algorithmes de traitement de données avancé sont nécessaires).	. Cartographie 3D . Détection précise d'obstacles

2.3.3 Fonctionnement d'un système complet de détection

Voici les phases du processus d'un système de détection d'obstacles, qui fonctionne grâce à une caméra :

A. Phase d'acquisition

La caméra est positionnée à un endroit du passage à niveau, où elle peut visualiser la scène, afin de capturer le moindre mouvement suspect des sujets. Par la suite, ses données sont envoyées à l'unité de traitement (un ordinateur) [22].

B. Phase de prétraitement

Cette phase effectue des opérations, visant à préparer les données ; comme diminuer les bruits, qui parasitent la qualité d'image, le contraste ; perturbant l'environnement visuel ; rendant les images inexploitables. De plus, cette phase ajuste la luminosité et synchronise les données provenant d'une ou plusieurs caméras ; dans le but d'obtenir des images plus claires et exploitables pour la détection de ces obstacles [22].

C. Phase de détection et de classification

La détection d'objets représente à la fois, l'identification d'objets et leurs localisations, tandis que la classification se concentre sur l'étiquetage des caractéristiques particulières sur l'image [22]. Cette phase utilise des algorithmes de vision par ordinateur, pour analyser les images et détecter s'il y a un obstacle sur les voies de chemin de fer ; il reconnait également la nature structurale de cet obstacle ; piéton, animal, véhicule, ou train ….etc.

D. Phase d'alertement

Dès qu'un obstacle est identifié, le système s'assure de la situation. En cas de danger, il déclenche des alertes de signalisation et prévient, en même temps, le centre de contrôle ainsi que le conducteur de la locomotive, afin d'éviter toute incident.

2.4 Synthèse des travaux récents sur la détection d'obstacles

- **T. W. Wardhana et R. Jayadi** [2] ont fusionné les données du jeu Visdrone, fourni par Ultralytics, développeur de YOLOv8 avec des jeux de données personnalisés, ajoutant des objets non inclus dans le logiciel car aucune étude n'a utilisé YOLOv8, malgré que des travaux similaires aient été réalisés dans d'autres pays. Par exemple, en Indonésie (KAI Property, filiale de KAI Kereta Api), c'était la première étude avec YOLOv8, qui détecte les objets aux passages à niveau.

  Leur méthodologie se base sur des observations de terrain et des entretiens avec le service responsable de la vidéosurveillance aux passages à niveau. Ils ont programmé, via les outils d'interface, en ligne de commande d'AnacondaPrompt et l'outil Labellmg, a été utilisé pour annoter les jeux de données personnalisés ; leurs résultats ont atteint un taux de précision de 98.7%.

Figure 2-12 : Exemple de l'étude de T. W. Wardhana et R. Jayadi *[2]*.

- Paramètres d'entraînement : 9 classes d'objets, avec 45 images pour la validation et 135 images pour l'entraînement. 45 images ont été spécifiquement annotées pour l'entraînement.
- Spécification de YOLOv8 : 100 époques, un taux d'apprentissage de 0.01, la taille des images 640 px, la taille du lot 8, le nombre de couches 218 et les paramètres totaux 25 844 971. Le temps d'entraînement a pris 0.154 heures pour les 100 époques.

- **A. Amin, D. Chimba, K. Hasan, E. Samson** [3] ont étudié et ont proposé un système intelligent ; exploitant l'apprentissage automatique et la vision par ordinateur, pour renforcer la sécurité aux intersections autoroutes-rail ; car l'administration fédérale des chemins de fer (FRA) des États-Unis considère les accidents et les retards qui surviennent aux passages à niveau, comme une problématique majeure, en matière de sécurité.

  Le modèle d'ensemble développé, intègre plusieurs variantes de YOLO, pour identifier les rails à l'approche d'un train ; notamment YOLOv5s, YOLOv5m, et YOLOv5l pour la détection d'objets ainsi que des techniques de segmentation sémantique, issues de l'architecture UNet, qui ont été utilisées dans ce cas.

  Le système a été implémenté sur un Raspberry Pi, et utilise une caméra haute définition, installée au niveau des passages à niveau, pour surveiller la région d'intérêt, détecter l'approche des trains et de dégager la zone avant l'arrivée du train. Cette étude s'intéresse aux intersections autoroutes-rail ; un sujet similaire à notre propre recherche, axée sur la détection de véhicules immobilisés aux passages à niveau. L'objectif de ce travail est d'améliorer la sécurité à ces croisements, à l'aide d'un système intelligent reposant sur des techniques d'apprentissage automatique et de vision par ordinateur.
- **A. L. Amin, D. Chimba et K.Hassan** [4] ; leur étude scientifique met en place un système basé sur l'IA, pour détecter les accidents ferroviaires. Ce système repose sur une méthode, qui intègre une architecture UNet, pour identifier les trains venant en sens inverse dont il s'appuie aussi sur un modèle de YOLO, précisément YOLOv8, en utilisant trois variantes, afin de détecter les obstacles comme véhicules et piétons.

  Les chercheurs ont créé un jeu de données personnalisé, à partir d'enregistrement vidéo d'une durée de 50 heures, en utilisant une caméra Miovision Scout, placée sur différents passages à niveau à Nashville, dans le Tennessee ; un résultat de 5400 images ont été extraites et annotées. Afin d'enrichir le jeu de données, ils ont appliqué la méthode d'augmentation de données comme, la rotation, le retournement, la mosaïque, le recadrage et la mise à l'échelle, ce qui a abouti à une multitude d'image, au nombre 27000.

Les résultats obtenus, jusqu'à 96 % de précision, un F1-score de 95 % et un temps d'inférence, située entre un intervalle de 4 à 6.8 ms. Le modèle affiche un taux de vrais positifs de 100 % pour les piétons et de 95 % pour les voitures.

- **M. Sevi et İ. Aydın** [5] ont réalisé un système moderne de détection d'obstacles (personnes et véhicules), autour des lignes ferroviaires. Ils ont utilisé le jeu de données RailSem19, qui contient 8 500 images photographiées d'un point de vue du conducteur de train, présentées dans une variété de scènes et d'environnements. Ces images de personnes et de voitures, ont été annotées manuellement.

Figure 2-13 : Exemple de l'étude de M. Sevi et İ. Aydın *[5]*.

Leur jeu de données a été soumis à des tests de comparaison, en utilisant cinq variantes de YOLOv8 (n, s, m, l, x) ; les performances ont été évaluées par le détecteur de précision moyenne (mAP50). Les résultats prouvent que YOLOv8m, offre un équilibre entre vitesse et précision, avec un mAP de 88,8 %. Dans l'ensemble, cette recherche montre que YOLOv8, permet une détection efficace et en temps réel, dont le but d'éviter les accidents.

- **S.A. Fahim** [6] a écrit un article proposant, un détecteur d'objets basé sur le modèle YOLOv9, pour détecter les véhicules, en temps réel, du Bangladesh, à partir des images de la caméra. L'objectif étant de surveiller la circulation. Ce chercheur japonais a utilisé le jeu de données Poribihon-BD, car il est plus important en données, que les autres jeux disponibles au Bangladesh ; il contient 9058 images annotées, de 15 types de véhicules bangladais, photographiées sur des autoroutes.

Ces images montrent différents angles, positions multiples et conditions d'éclairages ; le jour ensoleillé, les endroits à faible luminosité et la nuit, ainsi que certains conditions météorologiques comme la pluie. De plus, le chercheur suggère le déploiement du modèle sur les systèmes de vidéosurveillance, dans toute la ville.

2.5 Conclusion

Devant le constat d'innombrables accidents survenant lors des lieux du passage à niveau, aux conséquences parfois dramatiques sur les plans humain, matériel et financier, nous avons souhaité répondre à la problématique de ce sujet en proposant, de manière comparative, certaines solutions technologiques. Dans le prochain chapitre, nous explorons les algorithmes de détection modernes et précis, dont le but d'éviter les accidents dans ces zones.

Chapitre 3 : Revue des techniques d’apprentissage pour la détection d'objets

3.1 Introduction

Pour la conception du système dans notre recherche, nous mettons en relief, les architectures de deep learning, tels que les réseaux de neurones convolutifs (Convolutional Neural Network-CNN), et les évolutions des algorithmes de détections ainsi les différentes versions de YOLO (You Only Look Once). Ensuite, nous allons évoquer les métriques d’évaluation pour la performance du modèle.

3.2 Machines d’apprentissage

Les machines d'apprentissage, sous-domaine de l’IA, visent à développer des algorithmes capables d'apprendre à partir de données sans programmation explicite ; ce qui permet aux systèmes de s'adapter et d'améliorer leurs performances avec l'expérience, contrairement aux méthodes traditionnelles. Dans le domaine de la détection d'objets, leur importance se manifeste à travers plusieurs avantages clés : une précision améliorée, l'automatisation de la détection, une grande flexibilité, la capacité à gérer des données complexes, ainsi qu’une évolutivité significative [23].

3.2.1 Types d’apprentissage automatique

A. Apprentissage supervisé

Apprentissage supervisé est une méthode d’apprentissage par induction, qui consiste à construire automatiquement un classifieur, à partir d’exemples déjà classés.

Ce type d’apprentissage est dit « supervisé » vu que la fonction de classification, s’entraîne sur les classes, ainsi que sur leurs caractéristiques ; Parmi ses applications, on y trouve : la détection d’objets et la reconnaissance de plaques d’immatriculation [24].

B. Apprentissage non supervisé

L’apprentissage non supervisé, aussi appelé apprentissage sans enseignant ; dans ce dernier, la tâche consiste, en la découverte des similarités entre les observations (entrées) dans une collection d’exemples, afin de regrouper celles-ci en sous-ensembles, appelés clusters ou classes [25].

C. Apprentissage par renforcement

L'apprentissage par renforcement est une méthode qui permet à un système d'apprendre à atteindre un objectif en essayant différentes actions et en observant les résultats. Au lieu de lui dire exactement quoi faire, on lui donne un but, et il apprend par essais et erreurs, en recevant des récompenses quand il fait bien et des pénalités quand il fait mal. Petit à petit, le système comprend quelles actions lui permettent d'atteindre son but de la meilleure façon possible [26].

3.2.2 Apprentissage profond et réseaux de neurones artificiels (ANN)

Les Réseaux de Neurones Artificiels (Artificial Neural Networks, ANN), s'inspirent du fonctionnement biologique du cerveau humain, pour résoudre des problèmes complexes, d'apprentissage automatique.

Ces modèles, particulièrement efficaces en vision par ordinateur, permettent de transformer des données brutes (comme des pixels d'image), en prédictions structurées, grâce à une architecture multicouche [27].

Tableau 3-1 : Structure des couches dans un réseau de neurones artificiels (ANN) *[28]*.

Composant	Description Améliorée
Neurone (Unité)	Élément fondamental du réseau : il reçoit des signaux d'entrée, applique des poids et un biais, puis passe le résultat à une fonction d'activation.
Couche d'entrée	Première couche du réseau, qui encode les données brutes en signaux exploitables par le modèle.
Couches cachées	Couches intermédiaires qui apprennent des représentations internes en combinant les signaux des couches précédentes ; plus leur nombre est grand, plus les représentations apprises sont abstraites.
Couche de sortie	Dernière couche du réseau, celle-ci génère le résultat final sous une forme interprétable (valeurs continues, probabilités, classes, etc.).

3.2.3 Apprentissage profond et apprentissage automatique

L'apprentissage automatique utilise des algorithmes pour apprendre à partir de données et faire des prédictions. Tandis que l'apprentissage profond, qui est une forme avancée d'apprentissage automatique, utilise des réseaux de neurones profonds pour traiter des données complexes. Il nécessite plus de données et de puissance de calcul, mais offre souvent de meilleures performances sur des tâches comme la reconnaissance d'images. Ainsi l'apprentissage profond permet de résoudre des problèmes plus complexes que l'apprentissage automatique classique [29].

Parmi les variantes des ANN, on trouve les CNNs, qui traitent les caractéristiques visuelles de l'image. Ils sont énormément utilisés en vison par ordinateur ; notamment pour des tâches, comme la détection d'objets, la classification d'images et la reconnaissance faciale.

3.3 Aperçu sur les techniques de détection d'objets

Avant tout, la détection d'objets est au centre des applications, en vision par ordinateur ; cela permet d'identifier et de localiser ces cibles, dans toute image ou vidéo, en précisant sa classe et son emplacement en même temps [30].

Néanmoins, malgré ses avancées, de multiples contraintes et difficultés, entravent la détection d'objets ; c'est pour cela que cette dernière doit être précise, rapide et efficace, en regard des conditions d'environnements particulières (éclairage, luminosité, présence d'une foule…etc.). Ces exigences entraînent des obstacles techniques, quelque peu difficiles à surmonter ; notamment dans le traitement, en temps réel et surtout à l'adaptation de divers milieux.

Voici les domaines dans lesquels la détection d'objets s'intègre [31] :

- ✓ La vidéosurveillance (détection d'intrusions, comportements suspects de la part d'individus).
- ✓ Les véhicules autonomes (détection des piétons, autres véhicules, panneaux de signalisation).
- ✓ La robotique (navigation, intersection avec des objets).
- ✓ La médecine (localisation de lésions ou tumeurs lors des imageries médicales)
- ✓ Le commerce de détails (comptage des personnes).

3.3.1 Méthodes traditionnelles de détection d'objets

A. Transformation de caractéristiques invariantes à l'échelle

Créé par David Lowe en 1999 [32], Transformation de caractéristiques invariantes à l'échelle, en anglais Scale-Invariant Feature Transform (SIFT), détecte et décrit des points clés invariants aux changements d'échelle, rotation, luminosité ou déformations. Elle utilise une pyramide de différences de Gaussiennes pour identifier des zones stables, puis filtre les points peu contrastés avant de générer des descripteurs de 128 dimensions.

B. Fonctionnalités robustes accélérées

Fonctionnalités robustes accélérées, en anglais Speeded-Up Robust Features (SURF), est une version optimisée de Transformation de caractéristiques invariantes à l'échelle SIFT

développée en 2006 par H. Bay et ses collaborateurs [33]. Il vise à accélérer significativement le traitement tout en maintenant une bonne robustesse aux transformations. Contrairement à SIFT qui utilise des différences de Gaussiennes, SURF emploie une approximation rapide de la transformée de Haar grâce à des images intégrales pour détecter efficacement les points clés.

C. Comparaison entre SIFT et SURF

Tableau 3-2 : Tableau comparatid des méthodes SIFT et SURF *[32] [33]*.

Critères	SIFT	SURF
Principe de la méthode	. Détection multi-échelle via différences de Gaussiennes. . Descripteurs basés sur des histogrammes de gradients.	. Détection basée sur le déterminant de la matrice de Hessienne. . Descripteurs de 64 ou 128 dimensions, calculées via des ondelettes de Haar.
Avantages	. Robustesse exceptionnelle aux transformations géométriques. . Idéal pour la reconnaissance d'objets ou la création de panoramas.	. Environ 3 fois plus rapide que SIFT. . Performant face aux changements d'échelle et de rotation. . Adapté aux applications en temps réel (navigation robotique).
Limites	. Calcul lent. . Brevetée jusqu'en 2020 (limitations commerciales).	. Moins précis que SIFT en cas de transformations complexes. . Sensible au bruit.

3.3.2 Détection d'objet par apprentissage profond

Dans cette section, nous présentons cinq architectures de détection d'objets, à savoir : R-CNN (Regions with CNN features), Fast R-CNN, Faster R-CNN**,** SSD (Single Shot MultiBox Detector) et YOLO (You Only Look Once). Ces architectures reposent sur les composants fondamentaux des CNN. Nous commençons donc par une brève présentation des CNN, avant de décrire en détail les différents détecteurs d'objets.

A. Principe des CNNs

Les CNNs, également appelés ConvNets, sont un type de réseaux de neurones à propagation directe ; particulièrement adaptés aux tâches liées à la vision par ordinateur, notamment à la classification d'image, la reconnaissance faciale ou encore la détection d'objets [34].

Comme les réseaux de neurones standards, ils comportent plusieurs couches séquentielles ; de sorte que les sorties d'une couche constituent les entrées de la couche suivante. La plupart des concepts des CNN, utilise les même principes que les ANN, comme la descente de gradient stochastique et rétro-propagation pour estimer les pondérations [34].

Une architecture CNN typique comprend, généralement des couches alternées de convolution et de pooling ; suivies d'une ou plusieurs couches entièrement connectées à la fin. Dans certains cas, une couche entièrement connectée est remplacée par une couche de pooling de la moyenne globale. Outre différentes unités de régulation, telles que, la normalisation par lots (batch normalization) et la technique de régularisation, par abandon aléatoire de neurones (dropout) ; sont également intégrées, pour optimiser les performances du CNN. La disposition de ces composants joue un rôle fondamental, dans la conception de nouvelles architectures et l'obtention des meilleures performances. Cette section, aborde brièvement le rôle de ces composants dans une architecture CNN [35]. Voir la figure ci-dessous :

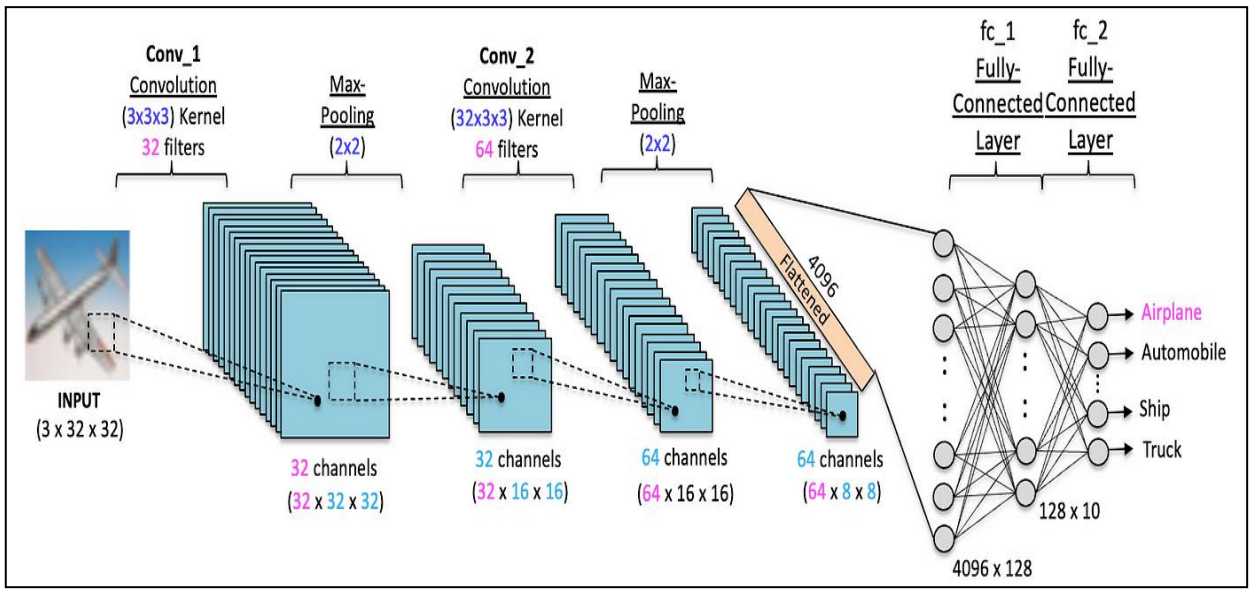


Figure 3-1 : Architecture d'un Réseaux de Neuronal Convolutif.

Composants des CNN

➢ La couche de convolution (convolution layers)

La couche de convolution est parfois, appelée couche d'extraction de caractéristiques, car les caractéristiques de l'image sont extraites dans cette couche. Une partie de l'image est connectée à la couche convolution, pour effectuer une opération de convolution et calculer le produit scalaire, entre le champ récepteur (c'est une région locale de l'image d'entrée ayant la même taille que celle du filtre) et le filtre.

Le résultat de l'opération est un unique entier dans le volume de sortie. Ensuite, le filtre se glisse sur le champ récepteur suivant de la même image d'entrée par une foulée ; tout en répétant la même opération. Ce processus est effectué de manière répétitive jusqu'à ce que toute l'image soit parcourue [36]. Voir la figure ci-dessous :

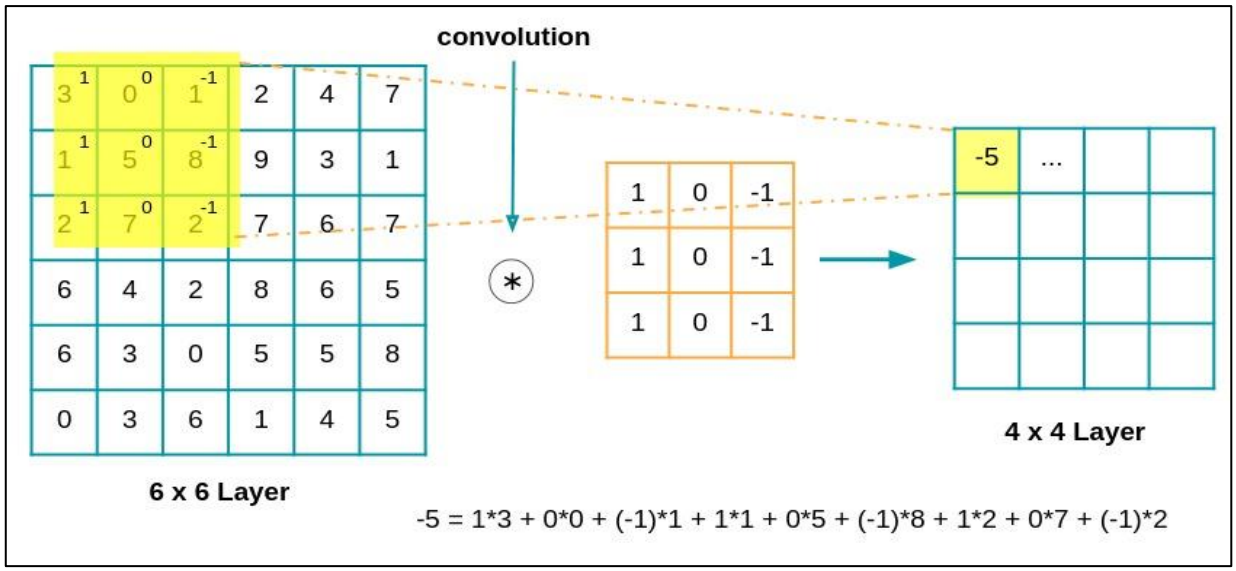


Figure 3-2 : Illustration de l'opération de convolution sur une image avec un filtre 3 × 3 *[37]*.

- **La couche de mise en commun (Pooling layers)**

La couche de pooling est une opération de sous-échantillonnage, utilisée pour réduire le volume spatial de l'image d'entrée, après la convolution. Elle est généralement appliquée, entre deux couches de convolution pour diminuer le nombre de paramètre et les calculs réseaux.

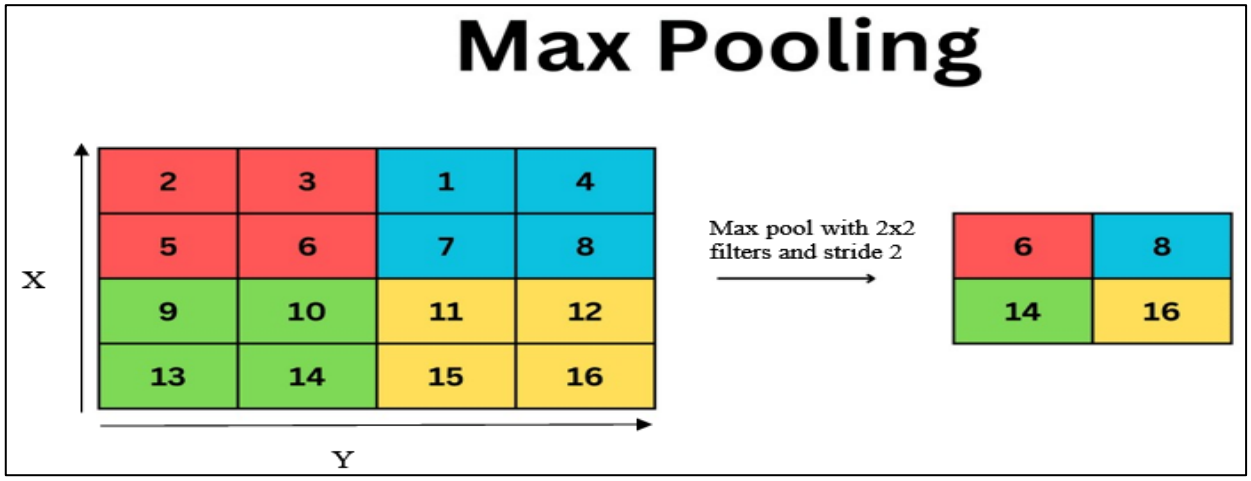


Figure 3-3 : Illustration du max pooling avec un filtre 2x2 et un pas de 2 *[38]*.

Sans pooling, l'application d'une couche complètement connectée après la convolution, entraine un coût de calcul élevé. Il existe plusieurs types de pooling (max, moyenne, somme), mais le plus courant est le max-pooling (voir la figure au-dessus), qui sélectionne la valeur maximale dans une région donnée (par exemple, une fenêtre 2 × 2 avec un pas de 2) ; ce qui réduit progressivement la taille de la carte des caractéristiques, tout en conservant les informations essentielles [36].

- **Couche entièrement connectées (Fully Connceted Layers)**

Après les couches de convolution et de pooling, le raisonnement de haut niveau dans un CNN, s'effectue via des couches entièrement connectées. Chaque couche du CNN agit comme un filtre de détection de caractéristiques ; les premières détectent des caractéristiques, qui peuvent être reconnues et interprétées facilement ; tandis que, les couches profondes détectent de plus en plus des caractéristiques plus abstraites. La dernière Couche entièrement connectée combine toutes ces informations pour produire une classification précise [39].

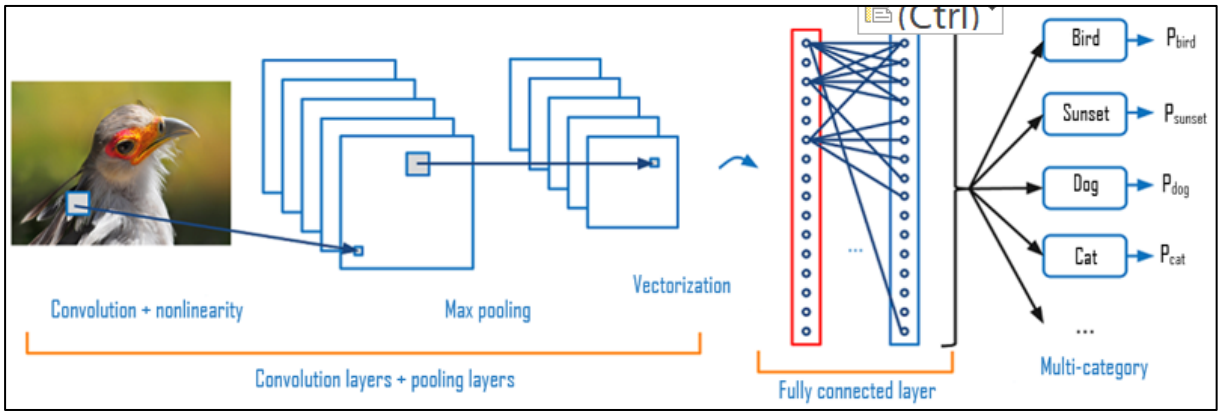


Figure 3-4 : La Couche entièrement connectées *[40]*.

Les CNNs représentent la base essentielle de nombreuses architectures modernes, de détection d'objets, car ils permettent d'extraire automatiquement des motifs dans les images. Ils sont capables de gérer plusieurs variations, comme la luminosité et l'orientation. Dans les tâches de classification, un CNN apprend à identifier les caractéristiques d'une image, directement à partir des données ; sans recourir à des descripteurs manuels, comme SIFT et SURF, où l'humain choisisse à l'avance quelles caractéristiques sont importantes (contours, angles…etc.) [41].

B. Méthodes de détection d'objets

1) Réseau de proposition régionale (Regions with CNN features)

Crée en 2014 par Ross Girshik et al [42], R-CNN, est l'un des premiers modèles à intégrer les CNN dans la détection d'objets.

- Il utilise la méthode recherche sélective (Selective Search), pour générer environ 2000 régions d'intérêt (ROIs) dans une image.
- Chaque ROI est redimensionnée à une taille fixe, elle est ensuite traitée par un CNN afin d'en extraire des caractéristiques (forme, textures couleur...etc.).
- Ces caractéristiques sont ensuite classifiées avec un SVM (un modèle de classification), pour identifier les objets (voiture, chat, personne…etc.) et affinées via une régression des boîtes englobantes.

Néanmoins, R-CNN est lent, car il applique un CNN séparément à chaque ROI, ce qui prend beaucoup de temps en calcul.

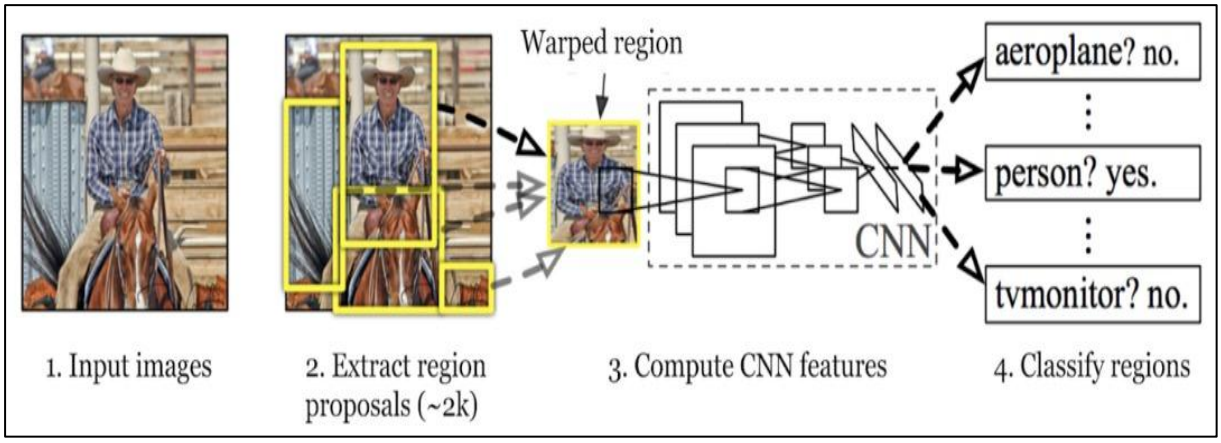


Figure 3-5 : Architecture du R-CNN pour la détection d'objets *[42]*.

2) Fast R-CNN

Proposé par Ross Girshick en 2015 [43], Fast R-CNN est une amélioration de R-CNN. Il applique un CNN une seule fois sur l'image entière, pour extraire une carte de caractéristique.

- L'image entière est traitée une seul fois par un CNN pour produire une carte de caractéristique globale.
- Il utilise encore recherche sélective pour générer les ROIs.
- Les ROIs sont sélectionnés et traitées par la couche mise en correspondance des ROI (ROI Pooling), pour ajuster leur taille avant classification.

- Ces régions sont ensuite classifiées via une couche Softmax, qui remplace le SVM utilisé dans R-CNN, intégrant directement la classification dans le réseau.
- Une régression des boîtes englobantes est également effectuée pour affiner la localisation.

Son avantage est qu'il est plus rapide que R-CNN, car il évite de recalculer les caractéristiques pour chaque ROI.

3) Faster R-CNN

Développé par Shaoqing Ren et al. en 2016 [44], Faster R-CNN améliore Fast R-CNN en intégrant un Réseau de Proposition de Région (RPN), qui remplace Selective Search.

Composé de trois parties principales :

- **Extracteur de caractéristiques (Feature Extractor) :** l'image est d'abord traitée par un CNN pour extraire les caractéristiques.
- **Réseau de Proposition de Région (Regional Proposal Network) :** génère les ROIs qui pourraient contenir des objets de manière plus rapide et efficace.
- **Réseau de détection (Detection Network) :** comme dans Fast R-CNN, les ROIs sont redimensionnées grâce à la couche ROI Pooling, puis classifiées et affinées.

4) Détecteur MultiBox à tir unique (Single Shot MultiBox Detector)

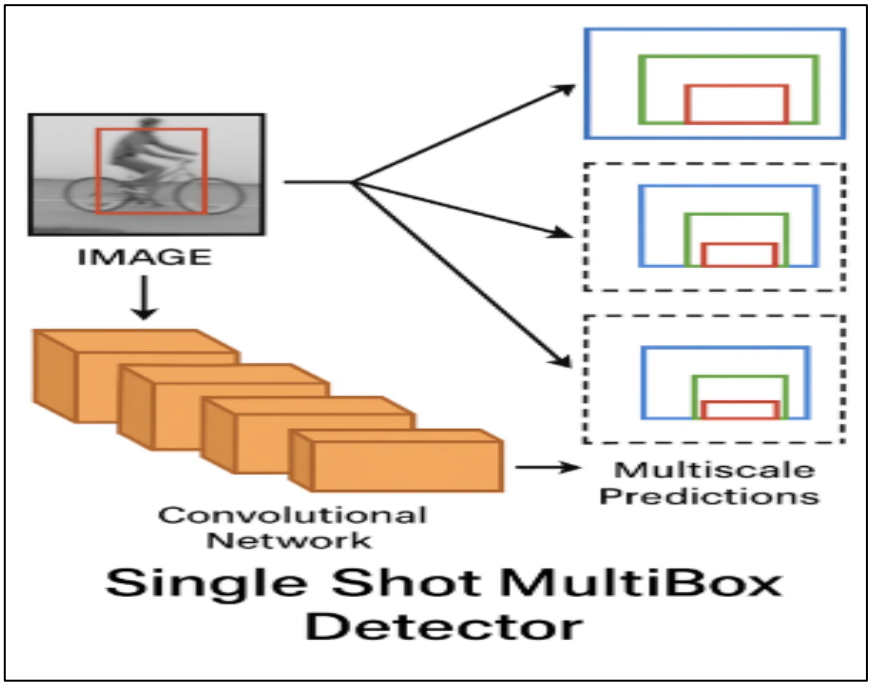


Figure 3-6 : Fonctionnement du modèle SSD.

Le modèle SSD traite l'image une seule fois, avec des prédictions à différents échelles, pour trouver et localiser tous les objets, même lorsque les images ont une faible résolution, contrairement à d'autres méthodes, qui analysent l'image en plusieurs étapes [45].

5) Détecteur d'objet en une seul Passage (You Only Look Once)

Le nom de YOLO indique qu'il constate l'image une seule fois pour la détection et la reconnaissance des objets, en temps réel, avec une grande précision et une rapidité d'exécution. Cet algorithme est Développé par J. Redmon, A. Farhadi, R. Girshick et S. Divvala [9].

YOLO présente la détection d'objets comme un problème de régression basé sur des boîtes englobantes et les probabilités de classe associées. A l'inverse d'autres algorithmes de détection, il examine l'image entière au moment du test, de sorte que ses prédictions sont basées sur le contexte global de l'image. Réputé pour sa rapidité, YOLO est idéal pour les applications temps réel.

Il se distingue par une approche triomphante ; la localisation et la classification des objets sont effectuées, à travers un CNN, ce qui permet de concilier vitesse d'exécution et précision des résultats.

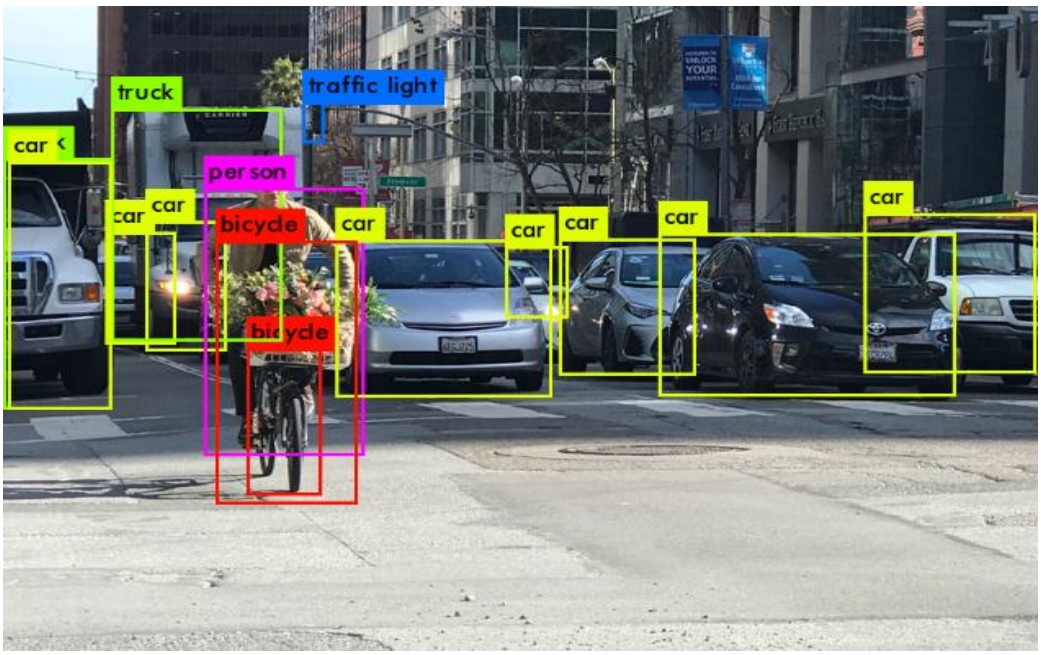


Figure 3-7 : Prédictions effectuées par le modèle YOLO *[9]*.

Tableau 3-3 : Comparaison des performances des principaux algorithmes de détection d'objets *[44] [9]*.

Critère	Faster RCNN	SSD	YOLO
Vitesse (FPS)	~5 à 10 FPS	~20 à 60 FPS	~30 à 120 FPS (selon les versions)
Précision (mAP)	Haute (~42-50%)	Moyenne (~22 à 30%)	Bonne (~45 à 65%)
Temps reel	Non adapté au temps réel à cause de sa lenteur.	Semi-temps reel	il est conçu pour le temps réel, avec des performances qui varie selon la version
Complexité	Élevée	Moyenne	Plus facile à entrainer et optimiser
Application	Adapté aux applications nécessitant une détection détaillée et une haute précision détaillée, comme la vidéo surveillance et médecine.	Convient aux applications mobiles, et aux drones en raison de son équilibre entre précision et vitesse.	Idéale pour les applications nécessitant une détection en temps réel, comme la conduite autonome et la surveillance en temps réel.

3.4 Détecteur et classification d'objets par YOLO

3.4.1 Architecture et fonctionnements de YOLO

L'architecture de ce réseau est inspirée du modèle GoogleNet, conçu pour la classification d'images. Ce réseau comporte 24 couches convolutives, suivies de 2 couches entièrement connectée ; au lieu des modules d'initialisation utilisée par GoogleNet, YOLO utilise simplement des couches de réduction 1 × 1 suivies de couches convolutives 3 × 3 [9].

L'algorithme YOLO suit trois étapes, comme le montre la figure 3-8 :

- **Division de l'image en une grille**

Divise l'image d'entrée en une grille de S × S cellules, ce qui donne N cellules au total ; si le centre d'un objet se trouve dans une cellule de la grille, celle-ci est chargée de détecter cet objet.

- **Prédiction des boîtes englobantes et scores de confiance**

Après la division, chaque cellule de la grille prédit B boîtes englobantes et leurs scores de confiance. Ces scores de confiance reflètent le degré de confiance du modèle, quant à la présence d'un objet dans la boîte et la précision de sa prédiction ; définie par :

$$Pr(Object) \times IoU(truth, pred) \qquad (1)$$

Pr(Object) : est la probabilité qu'un objet soit présent dans la boîte.

IoU : est l'intersection sur union qui mesure la qualité de localisation d'un objet détecté par YOLO, entre la boite réelle **(Truth)** et la boite prédit par le modèle **(Pred)**.

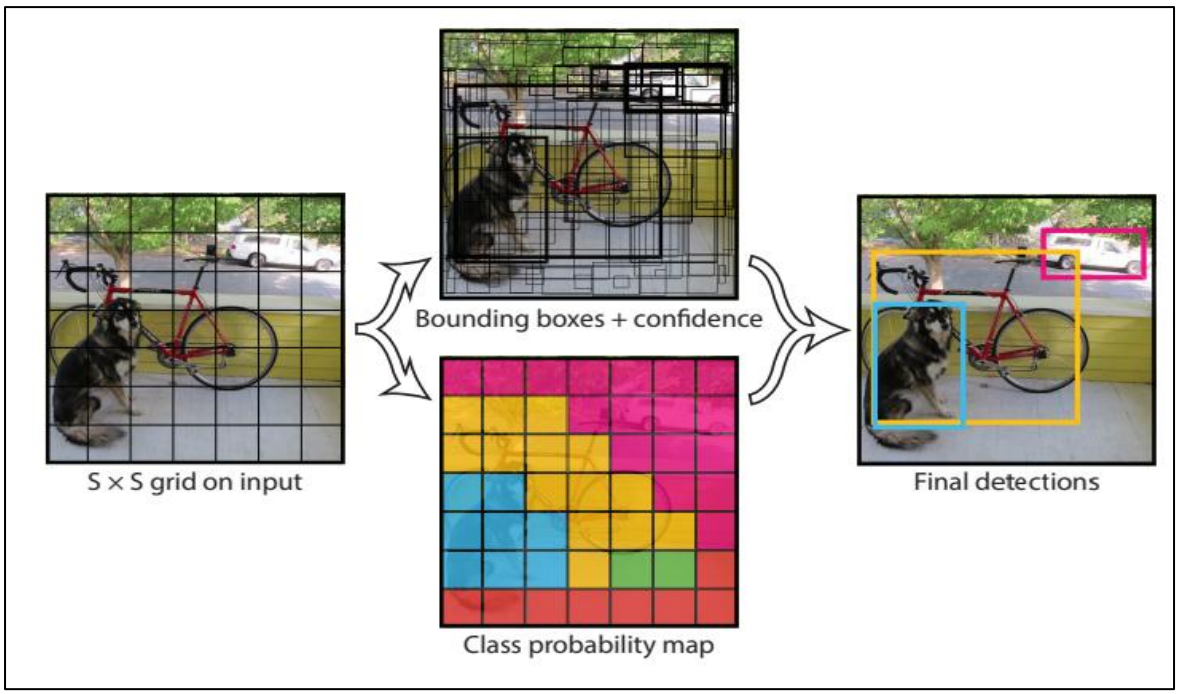
- Si aucun objet n'est présent dans cette cellule, le score de confiance doit être nul.
- Sinon, le score de confiance est égal à l'intersection sur union (IoU) entre la boîte prédite et la boîte de vérité terrain.

➢ **Prédiction des classes**

Chaque cellule de la grille, prédit également C probabilités de classes conditionnelles, partagées pour toutes les boîtes de la cellule. Ces probabilités indiquent la classe d'objet présent. Chaque boîte englobante est composée de cinq prédictions : x, y, w, h et un score de confiance.

- **x, y :** représentent les coordonnées du centre de la boîte, par rapport aux limites de la cellule de la grille.
- **w, h :** représentent la largeur et la hauteur de la boîte, par rapport à l'image entière.

Enfin, la prédiction de confiance représente l'IoU entre la boîte prédite et la boîte de vérité terrain.



. **Figure 3-8 :** Architecture et fonctionnement du modèle YOLO *[9]*.

3.4.2 Evolution des modèles de YOLO

Auparavant, et jusqu'à ces dernières années (voir la figure 3-9), il existait de multiples versions de YOLO, chacune d'elles évoluant de manière progressive, apportant son lot d'améliorations, par rapport à la précédente [46].

Voici une analyse comparative des différentes versions ; comme illustré dans ce tableau :

Tableau 3-4 : Analyse comparative des différentes versions *[47] [48] [49] [50] [51] [52]*.

Versions	Chercheurs	Activité
YOLOv2 (2016)	J. Redmon et A. Farhadi	**.**Détection et une multiple classification d'objets.
YOLOv3 (2018)	J. Redmon et A. Farhadi	**.**Détection multi-échelle, prédictions à trois niveaux.
YOLOv4 (2020)	A. Bochkovskiy, C.-Y. Wang et H.-Y.M. Liao	**.**Améliore la vitesse d'entraînement du modèle, détection et suivi d'objets.
YOLOv5 (2020)	Le Site officiel d'Ulratalytics	**.**Détection d'objets et segmentation.
YOLOv6(2022)	Meituan Inc	**.**Détection d'objets et segmentation pour applications industrielles (machine et robots).
YOLOv7(2022)	C.-Y. Wang et I.-H. Yeh	**.**Une meilleure précision et une vitesse de détection dans divers scénarios d'objets.
YOLOv8(2023)	Le Site officiel d'Ulratalytics	**.**Détection, segmentation d'objets, très rapide, Ajout du support des formats ONNX et TensorRT.
YOLOv9(2024)	C.-Y. Wang et I.-H. Yeh, H.-Y,-M, Liao	**.**Une meilleure précision, pour l'imagerie médicale et la conduite autonome.
YOLOv10(2024)	A. Wang, H. Chen, L. Liu, K.Chen, Z. Lin, J. Han, G. Ding	Détection extrêmement rapide avec précision, utilisé pour l'industrie.

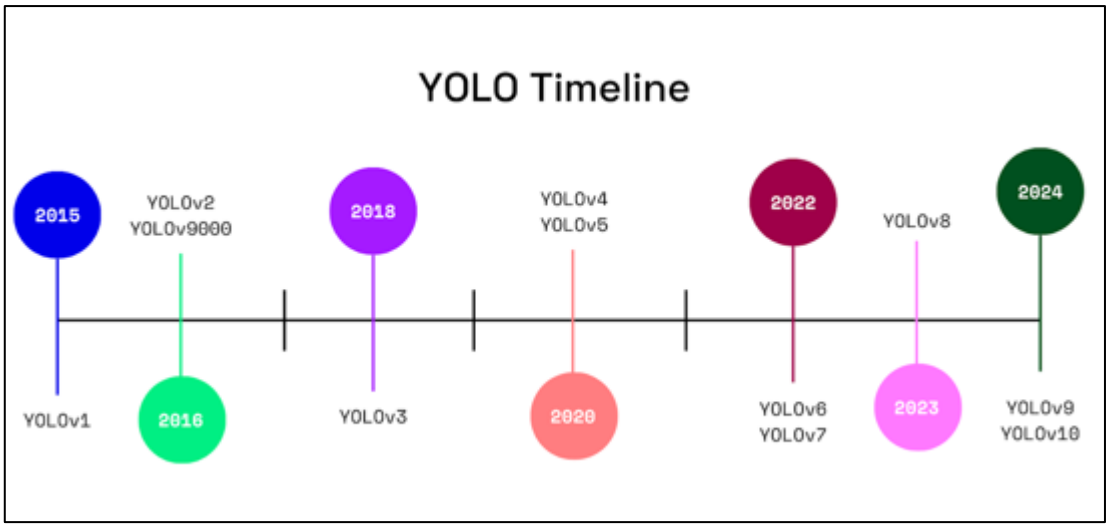


Figure 3-9 : Evolution des versions du modèles YOLO au fil de temps *[53]*.

3.4.3 Le modèle YOLOv8

Le modèle YOLOv8, apporte des fonctionnalités innovantes et des optimisations, qui en font un choix idéal pour diverses applications de détection d'objets ; grâce à ses améliorations significatives, en termes de précision et de rapidité par rapport aux versions précédentes [7].

YOLOv8 est un modèle sans ancres, c'est-à-dire qu'il n'utilise pas de boîte d'ancrage prédéfinie. Cette approche permet d'effectuer des prédictions denses, ce qui le rend possible de détecter des objets de tailles et formes variées et le rend aussi plus flexible pour d'autres tâches de vision par ordinateur [8]. Nous aborderons l'architecture ainsi que le choix du modèle YOLOv8 dans le prochain chapitre.

3.4.4 Analyse des variantes YOLOv8

La série YOLOv8 offre une gamme variée de modèles, chacun conçu pour des tâches particulières dans le domaine de la vision par ordinateur, ils répondent à une multitude de besoin ; la détection d'objets, la classification, la segmentation et la détection de points de clés. Chaque variante est fournie pour sa fonction spécifique, assurant une performance et une précision de haut niveau [54] [55].

a. YOLOv8n (Nano)

La conception de YOLOv8n (version ultralégère, environ 2 Mo en INT8), conçu pour les appareils à faible puissance (IoT, mobiles), facilite une détection rapide avec un impact réduit sur les performances. Compatible avec ONNX Runtime et TensorRT pour un déploiement optimisé efficace sur divers matériels [54].

b. YOLOv8s (Small)

Modèle équilibré (environ 9 millions de paramètres), conçu pour offrir un compromis entre vitesse et précision. Il intègre un Spatial Pyramid Pooling (SPP) [56] et un Path Aggregation Network (PANet) [57], améliorant la fusion des caractéristiques et la détections de petits objets. Cette variante fonctionne efficacement sur CPU [54].

c. YOLOv8m (Medium)

Grâce à l'architecture plus profonde de YOLOv8m (version intermédiaire, d'environ 25 millions de paramètres), il offre un compromis idéal entre performance de calcul et précision. Il permet à une variété de tâches de détection, comprenant des applications, en temps réel, nécessitant une grande précision, telles que la vidéo surveillance ou l'automatisation industrielle [54].

d. YOLOv8l (Large)

La conception de YOLOv8l (version améliorée, d'environ 55 millions de paramètres) pour les images à haute résolution et les objets complexes, permet d'intégrer des couches supplémentaires et un système d'attention amélioré, optimisant l'identification d'objets bien précis. Idéal pour des applications critiques telles que l'imagerie médicale et la conduite autonome, où une détection précise et fiable est essentielle [54].

e. YOLOv8x (Extra-large)

Pour rendre les applications exigeantes meilleures, telles que les systèmes de surveillance avancés, l'inspection industrielle de haute précision et les analyses médicales approfondies, il faut appliquer le YOLOv8x, qui est le modèle le plus étendu et le plus exact de la série YOLOv8, comptant environ 90 millions de paramètres. Cette performance nécessite une grande puissance de calcul avec une exécution idéale sur des GPU performants pour l'inférence en temps réel [54].

Ces modèles sont adaptés à diverses modes : entraînement, validation et inférence, ce qui facilite leur utilisation, à toutes les étapes du déploiement et du développement.

3.5 Métriques d'évaluation

Nous allons aborder les métriques essentielles, pour évaluer les performances de notre modèle de détection d'objets :

3.5.1 Mesures d'évaluation pour YOLO

A. Intersection sur Union (IoU)

L'IoU mesure la qualité de la localisation d'un objet détecté par YOLO. Elle est calculée en divisant la surface d'intersection, entre la boîte prédite et la boîte réelle, par la surface de leur union [58] ; comme illustrée sur la figure 3-10 :

$$IoU = \frac{Surface\ d'intersection}{Surface\ d'union} \qquad (2)$$

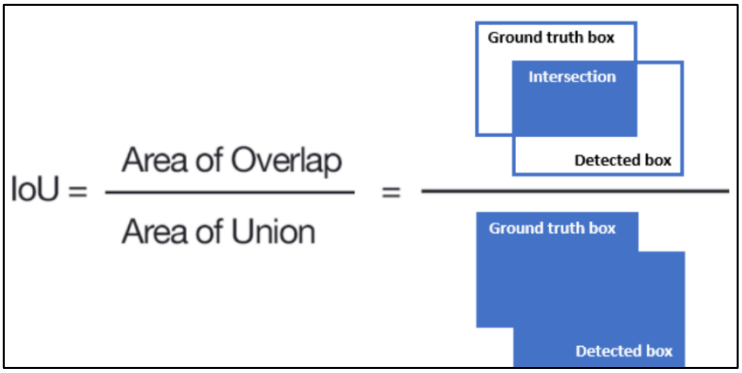


Figure 3-10 : Illustration du IoU pour l'évaluation de la détection d'objets *[59]*.

Un IoU proche de 1, signifie que la boîte prédite correspond très bien à la réalité. YOLO considère une détection correcte si IoU ≥ 0,5 ; comme celle représentée ci-dessous :

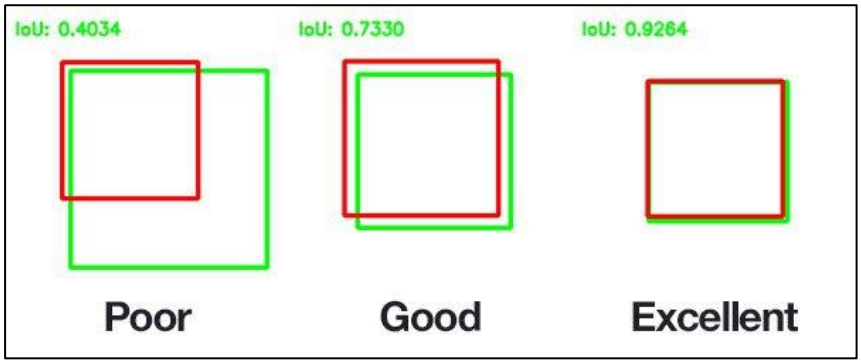


Figure 3-11 : Evaluation de la qualité des prédictions selon le IoU *[59]*.

B. Précision

La précision indique la proportion de détections correctes parmi toutes les détections, faites par le modèle [60].

$$Précision = \frac{TP}{(TP+FP)} \qquad (3)$$

Une précision élevée, signifie que le modèle fait peu d'erreurs, en détectant des objets qui n'existent pas (peu de faux positifs).

C. Rappel (Recall)

Le rappel mesure la capacité du modèle à détecter, tous les objets présents dans l'image [60].

$$Recall = \frac{TP}{(TP+FN)} \qquad (4)$$

Un rappel élevé signifie que le modèle ne manque pas beaucoup d'objets (peu de faux négatifs).

D. Score-F1

Le F1-Score combine la précision et le rappel, en une seule mesure équilibrée ; utile pour évaluer globalement la performance du modèle [61]. Un F1-Score élevé indique un bon compromis entre précision et rappel.

$$F1 = 2 \times \frac{(Précision \times Rappel)}{(Précision + Rappel)} \qquad (5)$$

E. Précision moyenne (mAP)

La précision moyenne est une mesure qui évalue la capacité d'un modèle YOLO, à détecter et classer correctement les objets dans une image [62].

Il existe deux variantes principales :

- mAP@0.5 utilise un seuil fixe d'IoU à 0,5, ce qui signifie qu'une détection est considérée correcte si la boîte prédite recouvre au moins 50 % de la boîte réelle.
- mAP@0.5:0.95 calcule la moyenne des mAP sur plusieurs seuils d'IoU, allant de 0,5 à 0,95, par pas de 0,05. Cette variante, utilisée notamment dans le jeu de données COCO.

3.5.2 Définitions fondamentales

- **Vrai Positif (TP) :** Le modèle détecte correctement, un objet présent [63].
- **Vrai Négatif (TN) :** Le modèle identifie correctement, l'absence d'objet (l'arrière-plan) [63].
- **Faux Positif (FP) :** Le modèle détecte un objet alors qu'il n'y en a pas (erreur de fausse alerte) [63].
- **Faux Négatif (FN) :** Le modèle ne détecte pas un objet alors qu'il est présent (erreur d'omission) [63].

3.5.3 Matrice de confusion pour YOLO

La matrice de confusion est un outil qui permet d'évaluer la performance d'un modèle de détection d'objets comme YOLO. Elle est construite en comparant les objets détectés avec les objets réels (vérité terrain) en tenant compte de :

- L'Intersection over Union (IoU) qui mesure le chevauchement entre la boîte prédite et la boîte réelle. Une détection est correcte si l'IoU dépasse un seuil (ex 0,5).
- Le score de confiance de chaque détection, qui indique sa fiabilité.
- Les classes des objets, y compris la classe 'arrière-plan' pour les zones sans objet.

La matrice indique les vrais positifs (détections correctes), les faux positifs (fausses alertes) et les faux négatifs (objets non détectés). Elle aide à comprendre où le modèle réussit ou fait des erreurs, pour mieux l'améliorer [64].

	Class 1	Class 2	...	Class M	Background
Class 1	(0,0)	(0,1)	...	...	...
Class 2	...	(1,1)	...	...	...
⋮	...	...	...	...	...
Class M	...	...	...	...	...
Background	...	...	...	...	(M,M)

Figure 3-12 : Matrice de confusion pour le modèle YOLO *[64]*.

3.5.4 Courbes de précision, rappel, PR et F1-score pour YOLO

A. Courbe de précision

La courbe de précision montre comment la précision du modèle YOLO, varie en fonction du seuil de confiance utilisé, pour accepter une détection. Contrairement à la mesure simple de précision, qui donne un seul chiffre, la courbe permet de voir la performance du modèle à différents niveaux de confiance [61].

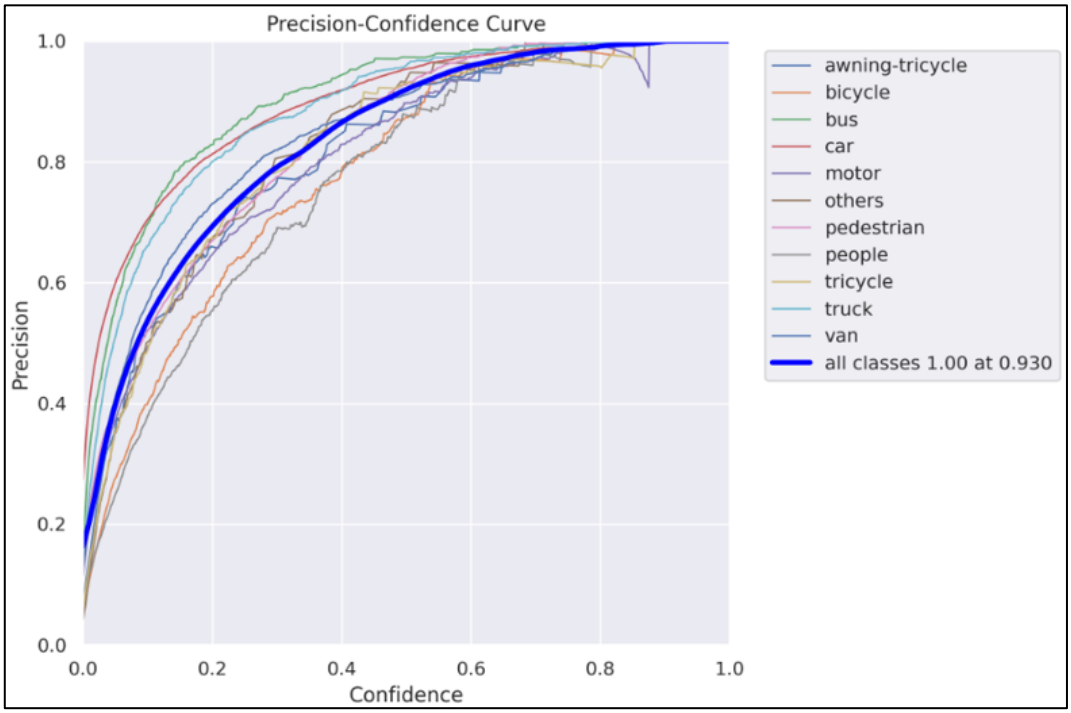


Figure 3-13 : Courbe de précision en fonction de la confiance *[65]*.

B. Courbe de rappel

La courbe de rappel indique la capacité de YOLO, à détecter tous les objets présents, également en fonction du seuil de confiance. Elle montre comment le rappel change lorsque le modèle devient, plus ou moins strict dans ses prédictions [61].

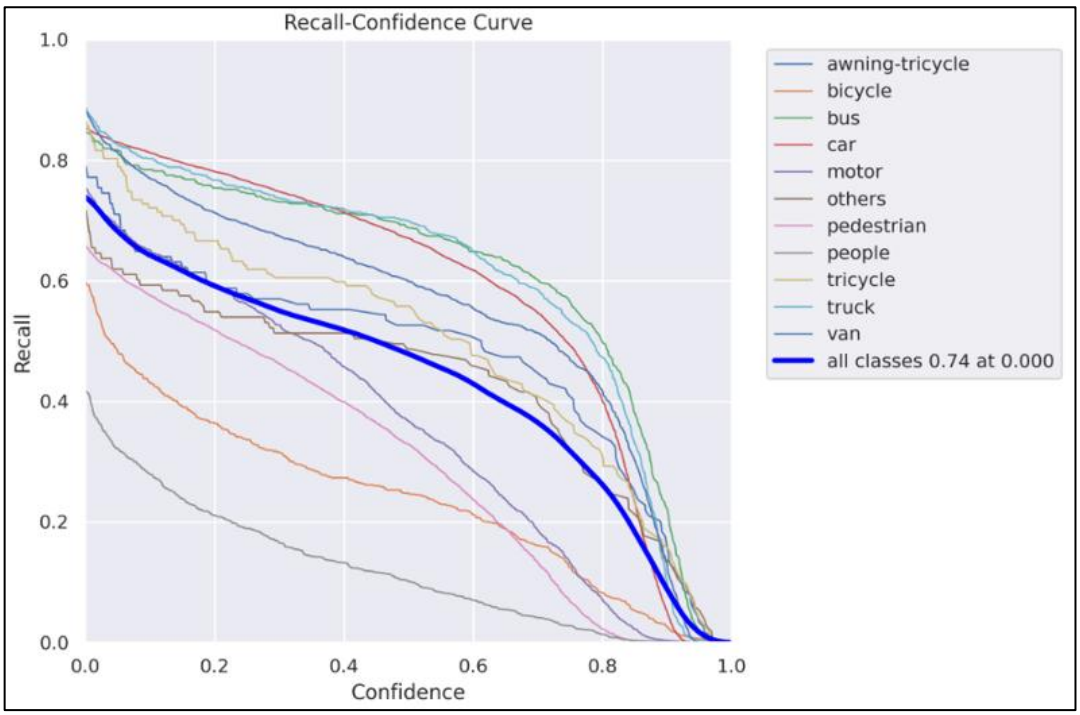


Figure 3-14 : Courbe de rappel en fonction de la confiance *[65]*.

C. Courbe précision et rappel (Précision-Rappel)

La courbe PR combine la précision et le rappel, en traçant la précision en fonction du rappel pour différents seuils de confiance. Elle permet de visualiser le compromis entre ces deux mesures et d'évaluer la performance globale du modèle [61].

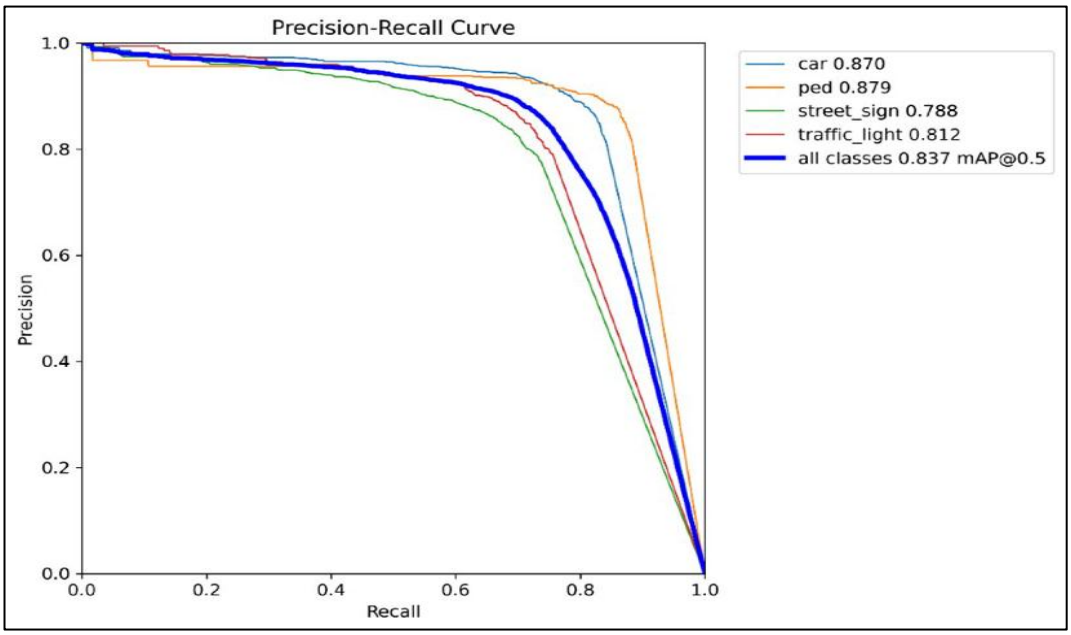


Figure 3-15 : illustration de la courbe de précision-rappel *[66]*.

D. Courbe F1-score

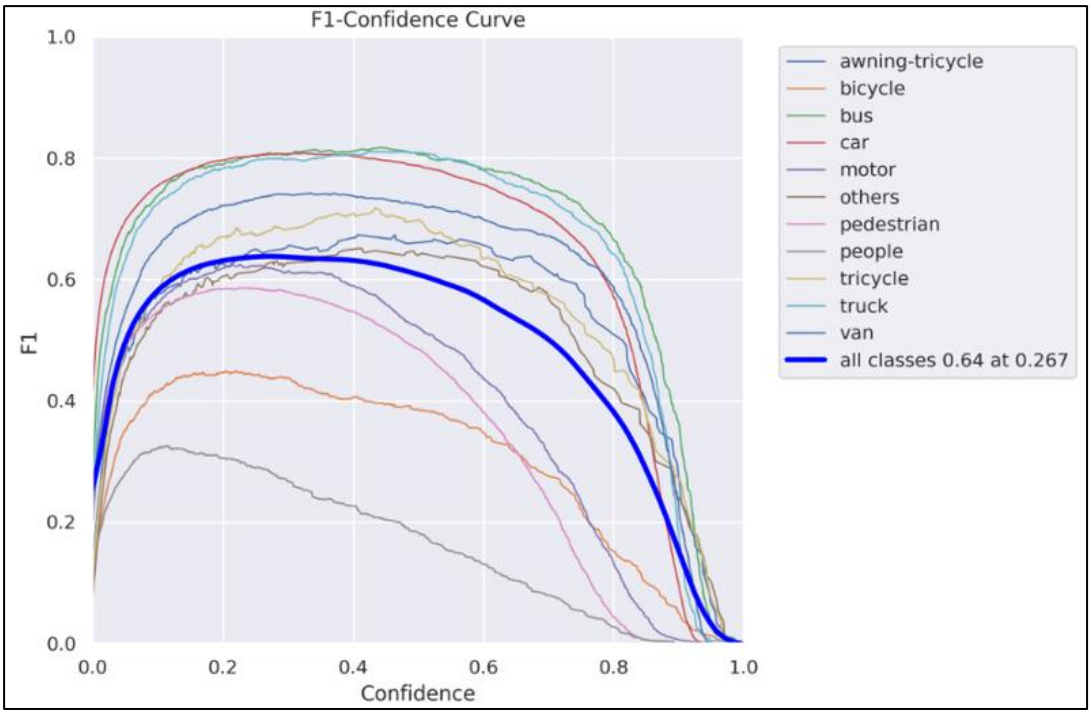


Figure 3-16 : Courbe F1 en fonction de la confiance *[65]*.

Le F1-score combine la précision et le rappel en une seule mesure équilibrée ; offrant un résumé global de la performance du modèle [61].

3.6 Conclusion

En résumé, les bases techniques abordées, permettent de valider la capacité des modèles, pour la pratique de système de détection d'objets. Tout d'abord, nous avons décrit les méthodes traditionnelles, jusqu'à l'évolution des méthodes modernes basées sur l'apprentissage profond.

Enfin, nous avons présenté les métriques essentielles afin d'analyser les performances de notre modèle de détection d'objets.

Chapitre 04 : Détection des véhicules immobilisés par YOLO version 8

4.1 Introduction

Dans ce chapitre, nous allons préconiser de faire l'entraînement du modèle YOLOv8 sur notre jeu de données. Tout d'abord, nous évoquerons la méthodologie globale, puis nous détaillerons les différentes étapes de préparation et de traitement de données. Ensuite, nous justifierons le choix du modèle YOLOv8, en analysant son architecture.

4.2 Etapes du travail

Figure 4-1 : Organigramme de notre système de détection d'objets.

4.3 Choix et traitement du jeu de données COCO2017

Parmi les nombreuses bases de données spécialisées dans la vision par ordinateur, principalement dans la détection d'objet, comme « PASCAL VOC » [67], « Open Image » [68] ou encore « ImageNet » [69], nous avons choisi d'utiliser Common Objects in Context (COCO2017) et ce, pour plusieurs raisons.

Ce jeu de données contient des images d'objets annotés, présents dans une variété de scènes et d'environnements ; allant des milieux urbains à des lieux naturels [70].

Par ailleurs, la qualité des annotations, réputée pour son exactitude ; offre un meilleur avantage d'entraînement et d'évaluation du modèle ; quel que soit le type d'annotations.

YOLOv8 se base sur des poids pré-entraînés obtenus, à partir d'un vaste ensemble de données, annotées pour des objets courants tels que des personnes, vélo, chaise…etc., et dont le modèle dispose déjà de connaissances générales, en détection d'objets ; ce qui permet d'identifier les classes cibles de notre projet, sans remise à niveau initial et donc, sans un apprentissage depuis un temps t0 [71].

De plus, grâce à son entraînement sur de nombreuses scènes, que ce soit des milieux urbains ou naturels, YOLOv8 reste performant, y compris avec de nouveaux contextes visuels, comme ceux des voies ferrées (des trains, des rails et des passages à niveau…etc.). De même, n'ayant pas visualisé un véhicule sur une voie ferrée, dans l'entraînement, YOLOv8 saura détecter un véhicule ou un piéton, de par ses connaissances antérieures d'autres environnements cumulés, auparavant.

Enfin, l'utilisation de COCO2017 permet de vérifier la capacité de détection du modèle, en mesurant les performances, à l'aide de mesures standardisées.

4.3.1 Présentation du jeu de données COCO2017

Le jeu de données COCO2017 est une base de données à grande échelle, il fournit des mesures d'évaluation standardisées, telles que la précision moyenne (mAP) pour la détection d'objets et le rappel moyen (mAR) pour les tâches de segmentation ; ce qui permet de comparer les performances des modèles d'un projet à l'autres [72].

A. Caractéristiques du jeu de données

- **Nombre total d'images :** environ 330 000 images, dont 200 000 sont annotées pour un total de plus de 1.5 millions d'objets reconnus.
- **Nombre de classes d'objets :** 80 classes utilisées pour la détection.

- **Segmentation d'objets :** annotations précises des contours de chaque objet, sous forme de polygones.
- **Reconnaissance en contexte :** les objets sont situés dans des scènes naturelles et urbaines.
- **Segmentation des éléments :** y compris les objets partiellement visibles ou présents dans une foule.
- **Annotations de points clé :** environ 250 000 personnes annotées avec des points clés.
- **légendes par images :** chaque image annotée est accompagnée de 5 descriptions textuelles.

Dans COCO les objets sont divisés en deux types :

- Les choses ce sont les objets individuels, exemple : personne, voiture, chien, chaise, tasse…etc.
- Les éléments qui occupent une grande partie de l'image, sans forme précise et qu'on ne peut pas compter, exemple : ciel, herbe, eau, mur, sol.

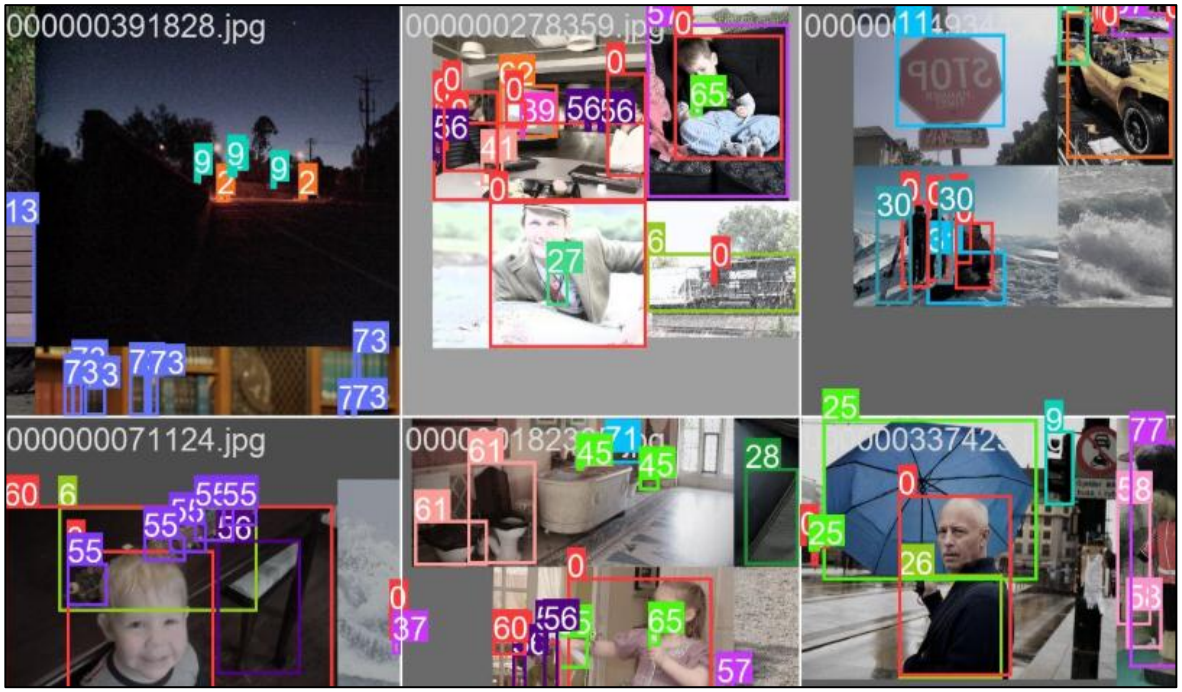


Figure 4-2 : Prédiction du modèle YOLO sur les images du jeu de données COCO2017 *[70]*.

B. Structure du jeu de données COCO

Ce jeu de données contient 118 287 images pour l'entraînement (train2017), 5000 images pour la validation (val2017) et 40 670 images pour les tests avec 80 classes d'objets annotés ;

présents dans une variété de scène et d'environnements, allant des milieux urbains à des espaces naturels.

Le format d'origine, des annotations COCO est en JSON, organisées selon les coordonnées exactes des boîtes englobantes [70]. Ces dernières sont exprimées sous la forme suivante :

$$(x_{min}, y_{min}, largeur, hauteur) \qquad (6)$$

4.3.2 Préparation et intégration du jeu de données COCO2017

Dans cette section, nous allons décrire les différentes étapes de préparation et d'intégration du jeu de données COCO2017, utilisées pour entraîner notre modèle de détection ; comme illustrée ci-dessous :

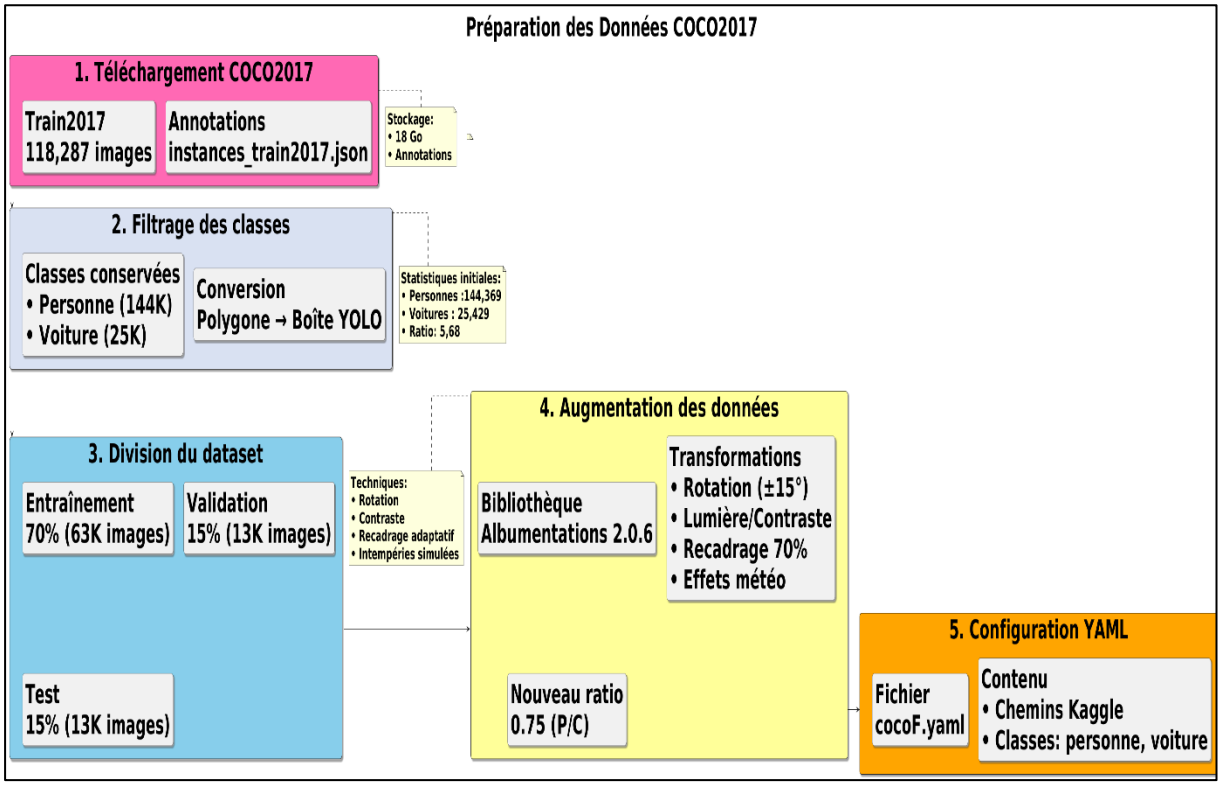


Figure 4-3 : Processus de préparation de données.

A. Téléchargement et filtrage du jeu de données

Le jeu de données COCO2017 contient 80 classes d'objets annotés, mais seulement deux d'entre-elles, ont un intérêt pour ce projet : car (voitures) et person (personnes). Nous avons effectué une opération de filtrage pour ne garder que les annotations, correspondant à ces deux catégories ; en éliminant tous les objets appartenant aux autres classes, non nécessaires à l'entraînement du modèle.

Les étapes de cette opération sont :

Dans un premier temps, cela a nécessité le téléchargement des données COCO2017, via le site d'Ultralytics, incluant les images d'apprentissage (train2017) et le fichier d'annotations (instances_train2017.json), associé à ces images.

Cette figure illustre les instructions données, pour télécharger le jeu de données COCO2017.

```yaml
# Download script/URL (optional)
download: |
  from pathlib import Path
  from ultralytics.utils.downloads import download
  # Download labels
  segments = True  # segment or box labels
  dir = Path(yaml["path"])  # dataset root dir
  url = "https://github.com/ultralytics/assets/releases/download/v0.0.0/"
  urls = [url + ("coco2017labels-segments.zip" if segments else "coco2017labels.zip")]  # labels
  download(urls, dir=dir.parent)
  # Download data
  urls = [
      "http://images.cocodataset.org/zips/train2017.zip",  # 19G, 118k images
      "http://images.cocodataset.org/zips/val2017.zip",  # 1G, 5k images
      "http://images.cocodataset.org/zips/test2017.zip",  # 7G, 41k images (optional)
  ]
  download(urls, dir=dir / "images", threads=3)
```

Figure 4-4 : Téléchargement du jeu de données COCO2017 depuis la documentation Ultralytics.

Cette figure montre l'espace de stockage requis par le jeu de données COCO2017, après l'extraction des fichiers, en local, sur ordinateur.

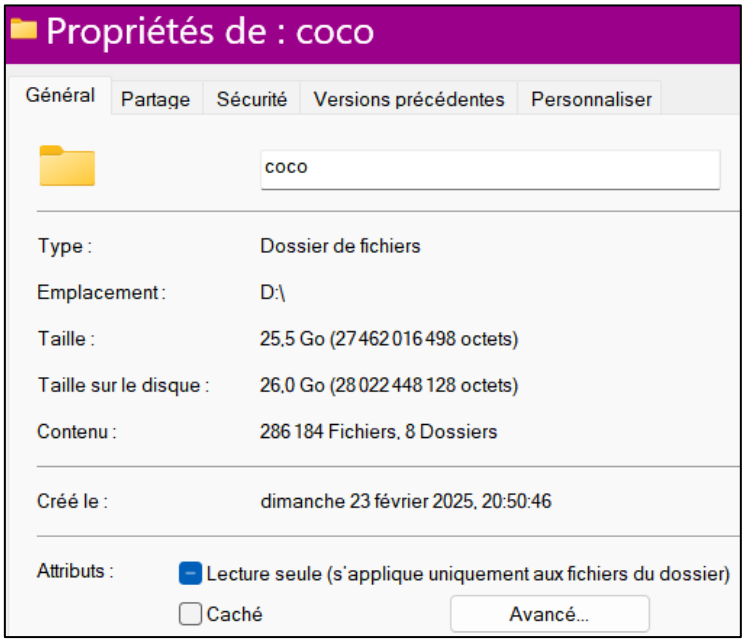


Figure 4-5 : Taille du jeu de données COCO2017 sur disque.

Dans un second temps, nous avons créé un script Python, pour lire et filtrer les annotations, ne gardant que celles qui appartiennent à ces deux classes. Afin d'adapter précisément le modèle YOLOv8 à notre problématique.

Cependant, d'autres méthodes existent, comme le fine-tuning d'un modèle pré-entrainé sur un jeu personnalisé, la modification des annotations, ou le mappage de classes. Lorsqu'on utilise un modèle pré-entrainé « best.pt » sans filtrage, cela ne permet pas d'optimiser la détection sur ces deux classes cibles ; car le modèle est entrainé sur 80 classes différentes, ce qui provoquera du « bruit » et réduira la performance sur les classes d'intérêt. Cependant le filtrage reste la méthode la plus simple et directe pour améliorer l'entraînement du modèle.

Enfin, dans un troisième temps, afin d'être en conformité avec les exigences du format d'entraînement YOLOv8. Nous avons procédé à une transformation pour les convertir en boîtes englobantes, en utilisant le script Python car les annotations dans COCO, fournies au format polygone, étaient incompatibles avec le format requis par YOLOv8.

B. Séparation des données

La division d'un ensemble de données, en trois parties est une étape essentielle, dans tout projet d'apprentissage supervisé. Elle permet :

- d'entraîner le modèle sur un ensemble riche et varié.
- d'évaluer ses performances, de façon continue pendant l'entraînement.
- de mesurer sa capacité de généralisation.

Nous avons réparti les données en trois sous-ensembles distincts :

- 70% pour l'entraînement : 63447 images.
- 15% pour la validation : 13596 images.
- 15% pour le test : 13596 images.

Cette répartition a été réalisée, de manière aléatoire ; tout en veillant à maintenir un certain équilibre entre les classes. L'objectif était de garantir, que chacun des sous-ensembles contient des occurrences représentatives des deux classes ciblées, à savoir « personne » et « voiture ».

C. Augmentation des données

Suite, à l'opération de filtrage, de conversion et de séparation du jeu de données COCO2017, nous avons constaté, un déséquilibre significatif entre ces deux classes, surnommés : « personne » et « voiture ».

L'analyse statistique initiale a révélé les éléments suivants :

- **Nombre total d'images :** 38 035.
- **Nombre d'annotations :**

- **Personnes :** 144 369.
- **Voitures :** 25 429.

➢ **Ratio Personnes et Voitures :** 5.68.

Ce déséquilibre pourrait biaiser l'entraînement du modèle YOLOv8, en faveur de la classe majoritaire, à savoir « personne». Pour y remédier, nous avons décidé, d'augmenter artificiellement, le nombre d'annotations pour la classe « voitures », afin d'atteindre un ratio cible de 1.0. Cela implique la création de 118 940 annotations supplémentaires, pour la classe « voiture».

Pour résoudre ce problème, nous avons conçu un script Python, en utilisant la bibliothèque Albumentations (version 2.0.6), qui applique la méthode d'augmentation de données sur les images de voiture.

Tableau 4-1 : Techniques d'augmentation de données appliquées aux images de voitures.

Transformation	Description
Rotation contrôlée	Les images de voitures sont tournées (entre -15 et +15) pour que le modèle les reconnaisse sous différents angles.
Ajustement photométrique	la luminosité (entre 0.8 et 1.2) et le contraste (entre 0.9 et 1.1) des images ont été modifiés pour que le modèle s'adapte à différents conditions de lumière, comme le soleil, l'ombre ou les jours nuageux.
Recadrage adaptatif	Nous avons appliqué un zoom, en gardant au moins 70% de la taille d'origine des voitures, pour ne pas trop les couper.
Perturbation réalistes	Nous avons ajouté du bruit et des effets comme la pluie ou la neige pour que le modèle apprenne à détecter les objets, même dans un mauvais temps.

Voici les différentes figures du jeu de données COCO2017, après avoir appliqué des techniques d'augmentation de données sur la classe « voiture » :

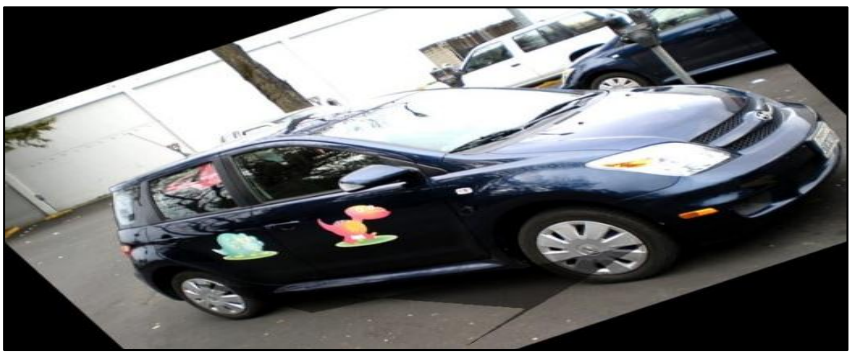


Figure 4-6 : Rotation et ajustements photométriques d'une scène urbaine.

Figure 4-7 : Simulation complète d'intempéries sur une scène sportive.

Figure 4-8 : Variation de luminosité sur une scène routière.

Après avoir appliqué les techniques d'augmentation de données, nous avons obtenu les résultats suivants :

Tableau 4-2 : Statistiques des annotations par catégorie après augmentation des données.

Jeu de données	Personne	Voiture	Ratio P/V
Entraînement	103 676	146 005	0.71
Validation	26 109	28 031	0.93
Total	129 785	174 036	0.75

D. Configuration du jeu de données via le fichier YAML

Afin de permettre au modèle YOLOv8, de reconnaître uniquement, les deux classes ciblées ; nous avons modifié le fichier de configuration correspondant coco.yaml. Ce fichier, au format Yet Another Markup Language (YAML), contient des informations sur :

- les chemins d'accès aux images d'entraînement et de validation.
- le nombre de classes.
- les noms des classes.

Nous avons adapté ce fichier en :

- remplaçant les chemins d'origine, par ceux de notre environnement de travail Kaggle : /kaggle/input/cocofiltreaugment/filtragaugment.
- ne conservant que les deux classes : « personne » et « voiture », en excluant les 78 autres classes non nécessaires pour notre projet.

```yaml
path: /kaggle/input/cocofiltreaugment/filtragaugment
train: /kaggle/input/cocofiltreaugment/filtragaugment/images/train2017
val: /kaggle/input/cocofiltreaugment/filtragaugment/images/val2017
nc: 2
names:
  0: person
  1: car
```

Figure 4-9 : Fichier cocoF.yaml ajusté pour notre jeu de données filtré.

4.4 Conception de détection et suivi du modèle

4.4.1 Choix du modèle YOLOv8

YOLOv8 s'avère judicieux, pour les activités de détection d'objets, en temps réel ; dans plusieurs domaines d'application, et cela grâce à une position harmonieuse entre précision et vitesse [7].

L'architecture du modèle YOLOv8 apporte de nouvelles améliorations, par apport à ses prédécesseurs (YOLOv5, YOLOv6, YOLOv7) ; c'est pour cela que ce modèle atteint une précision plus élevée, y compris pour la détection d'objets de petites taille ou à moitié visible [8].

Grâce à sa conception YOLOv8 offre une vitesse d'inférence plus rapide, ce qui est intéressant et primordial, pour les systèmes embarqués ainsi qu'aux applications de surveillance nécessitant un traitement, en temps réel.

A noter que, le modèle est conçu pour être facilement intégrer, à des Framework populaires comme PyTorch et OpenCV, facilitant ainsi le traitement de flux vidéo en directe. Pour améliorer ses performances, en matière d'extraction, de caractéristiques et de détection d'objets ; YOLOv8 repose sur une architecture cervicale et dorsale de haute technologie.

Par ailleurs, YOLOv8 utilise une tête de détection sans ancrage, qui améliore la précision ainsi que l'efficacité du processus de détection ; et ce comparativement, aux méthodes basées sur l'ancrage [7].

YOLOv8 est entièrement intégré dans l'environnement Ultralytics, ce qui facilite les tâches d'entraînement, de transfert d'apprentissage et de déploiement [7]. En outre, il possède plusieurs variantes ; chaque variante est fournie pour sa fonction spécifique, assurant une performance et une précision de haut niveau, selon les besoins.

Le choix s'est porté sur YOLOv8-medium, comparativement aux variantes (nano ou small) ; du fait de sa précision dans des conditions difficiles de météorologie, de luminosité (pénombre), ou de distances difficilement appréciables. Il nécessite moins de mémoire GPU, moins de puissance de calcul et il possède moins de paramètres que les variantes (large et extra-large), déjà cité dans le chapitre précédent ; ce qui réduit le temps d'entraînement avec une bonne précision.

YOLOv8 est bien adapté, avec une intégration des algorithmes de suivi, tels que DeepSORT, ByteTrack et OC-SORT. Il fournit des identifiants de détection cohérents ainsi que des cadres précis ; formant ainsi une base fiable, pour le suivi multi-objets, image par image, dans une séquence vidéo [73].

Pour mémoire, de nouvelles versions ont vu le jour récemment, telles que YOLOv9 et YOLOv10 ; néanmoins nous n'avons pas pu les exploiter de manière continue, malgré plusieurs essais, en raison d'une capacité insuffisante de l'environnement employé.

4.4.2 Architecture du modèle de détection YOLOv8

L'architecture de YOLOv8 repose sur un réseau de neurones convolutifs profond, optimisé pour la détection d'objets, en temps réel. Elle se divise en trois grandes parties :

- **Épine dorsale (Backbone)**

C'est la partie qui extrait les informations importantes de l'image ; YOLOv8 utilise une version améliorée de CSPDarknet, qui réduit les calculs, tout en capturant bien les détails à différentes résolutions (8, 16, 32) [74]. Elle comprend :

- des couches de convolution, pour détecter les motifs locaux.
- une normalisation par lot (Batch Normalization), pour stabiliser l'apprentissage.
- une fonction d'activation SiLU (Swish) qui aide le modèle à mieux apprendre.

- **Cou (Neck)**

Cette partie améliore et combine les informations extraites par le Backbone ; YOLOv8 utilise un PANet optimisé, qui fusionne les détails fins et les informations globales, grâce à des connexions dans les deux sens ; cela permet de mieux détecter les petits objets, tout en gardant les détails importants [74].

➢ **Tête (Head)**

La tête s'occupe de reconnaître les objets et de localiser précisément leurs positions dans l'image. Contrairement aux versions précédentes, YOLOv8 n'utilise pas de boîtes d'ancrage prédéfinies ; ce qui simplifie le processus [74].

Le modèle prédit directement les coordonnées des boîtes englobantes (centre, largeur, hauteur), ce qui améliore la détection d'objets de différentes tailles.

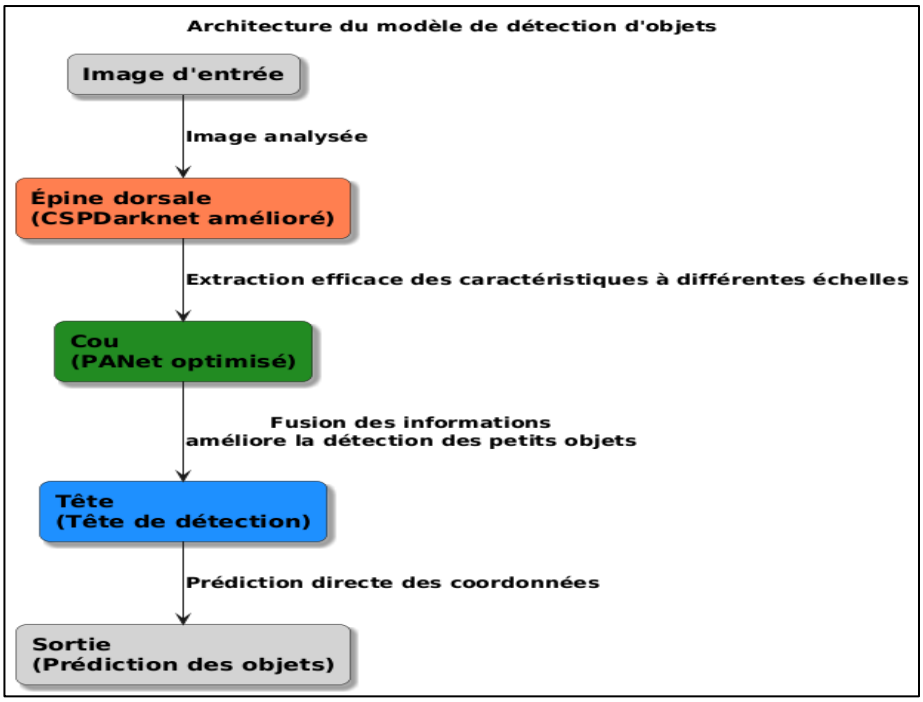


Figure 4-10 : Architecture du modèle de détection d'objets YOLOv8.

4.4.3 Mécanisme de détection et suivi avec YOLOv8 et DeepSort

A. Système de détection des objets avec YOLOv8

YOLOv8 offre une détection rapide, précise, recevant des images redimensionnées et produit pour chacune d'elles, des prédictions sous forme de cadres entourés, associées aux classes : « personne » et « voiture » ; comme illustrée ci-dessous :

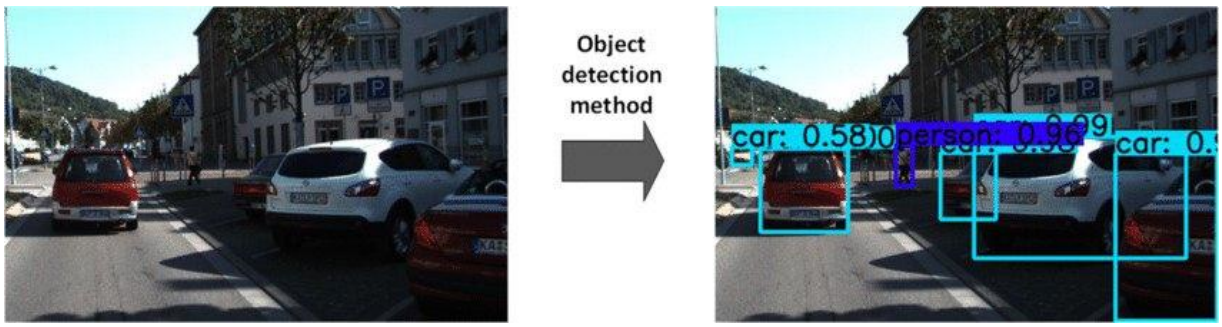


Figure 4-11 : Détection et classification des voitures sur une scène à l'aide de YOLO *[75]*.

Cette méthode de détection est à la fois rapide et précise, ce qui permet une reconnaissance simultanée de plusieurs objets, en temps réel. Chaque détection est caractérisée par sa classe, sa position, ses dimensions et un score de confiance. L'algorithme multi-objets, appelé en anglais Deep simple Online and Realtime Tracking (DeepSORT) exploite les résultats de détection de YOLOv8 et il assure un suivi précis et continu des objets au fil du temps.

B. Système de suivi des objets avec DeepSORT

Une fois, les véhicules et les personnes détectés sur les rails, il est nécessaire de pouvoir les suivre dans le temps ; pour analyser les comportements des personnes et l'arrêt soudain des véhicules ; afin d'éviter des collisions avec le train. Cette tâche est menée, par l'algorithme de suivi DeepSORT) car il permet d'associer une identité unique, à chaque cible détectée ; facilitant la compréhension de leur déplacement par un suivi, image par image, tout au long de la vidéo.

Grâce à ce système de suivi intelligent, il est possible de reconnaitre chaque véhicule ou personne, individuellement et de les suivre ; même s'il y en a plusieurs, à l'image, en même temps. Ce suivi fonctionne même s'ils se croisent ou apparaissent partiellement ; il est essentiel, pour vérifier si un véhicule est arrêté sur les rails ou s'il continue à avancer normalement afin d'éviter les accidents.

DeepSORT est une extension de l'algorithme Simple Online and Realtime Tracking (SORT), il s'appuie sur des techniques avancées, particulièrement un filtre de Kalman ; afin d'anticiper la position future des cibles et d'assurer une association correcte, entre les détections successives, même en cas d'occlusion temporaire [76].

Ainsi, l'alliance entre YOLOv8 pour la détection et DeepSort pour le suivi ; permet de surveiller les objets en mouvement, tout en conservant une trace précise de chacun, y compris dans n'importe quel environnement.

DeepSORT combine deux types d'informations [76] :

- Les caractéristiques dynamiques, comme la position, la vitesse et la taille de la boîte englobante.
- Les caractéristiques d'apparence visuelles, extraites à l'aide d'un CNN.

C. Détection de l'immobilisation des véhicules

Cette étape détecte la stabilité de la position d'un véhicule ou d'une personne, au cours du temps ; elle consiste à mesurer, si un même identifiant reste dans la même position, pendant un intervalle de temps défini, plus de deux minutes ; c'est alors que le système considère qu'il

est immobilisé. Dès qu'une immobilisation prolongée est détectée une alerte est générer automatiquement.

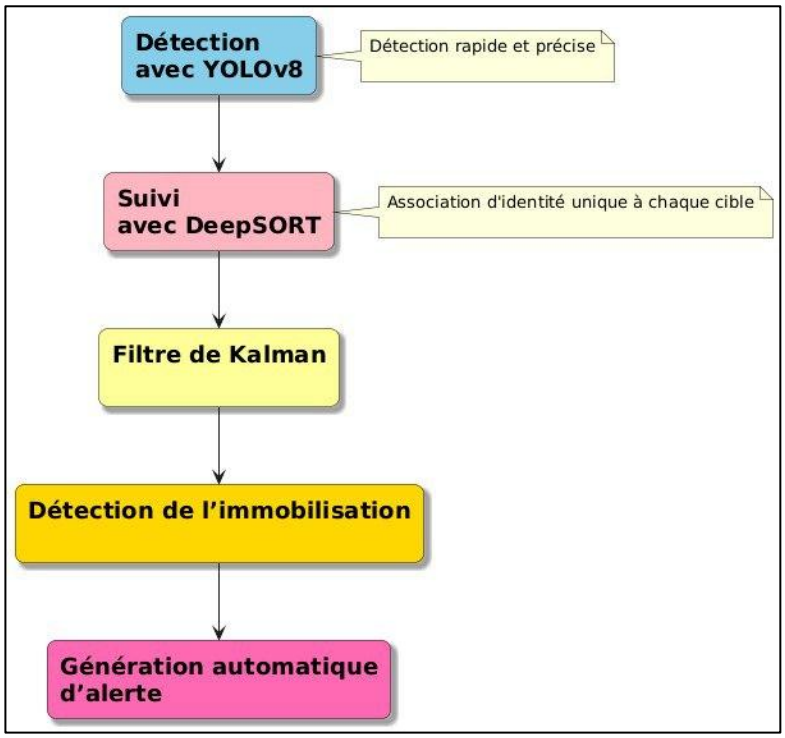


Figure 4-12 : Mécanisme de détection et de suivi avec YOLOv8 et DeepSort.

4.5 Entraînement du modèle YOLOv8 sur le jeu de données

L'entraînement du modèle représente une étape centrale de notre système de détection et de suivi d'objets. Son objectif est de développer un modèle fiable, efficace et capable de s'adapter à des scènes réelles.

Ce processus, pour lequel nous avons réalisé, avec un script Python, exécuté dans une plate-forme Kaggle ; englobe toutes les étapes essentielles de l'entraînement du modèle : la préparation des données, la configuration des hyper-paramètres, l'initialisation du système de suivi ainsi que l'emplacement et la sauvegarde structurée des résultats. Chaque étape a été pensée pour maximiser les performances, face aux contraintes de la plateforme Kaggle ; où des coupures d'électricités, de connexion d'internet ou les limites de la durée des sessions ; peuvent interrompre l'exécution.

4.5.1 Configuration des hyper paramètres

Le modèle YOLOv8m est distribué sur deux GPU grâce à **torch.nn.DataParallel.** Afin d'accélérer l'entraînement, garantissant un équilibre de charge satisfaisant. L'entraînement est

lancé avec les hyper-paramètres (voir la figure 4-13), qui garantissent un bon équilibre entre performance et vitesse :

```
train_args = {
    "data": "/kaggle/working/data.yaml",
    "epochs":200,
    "imgsz": 640,
    "batch": 16,
    "device": "0,1",
    "project": project_dir,
    "name": exp_name,
    "exist_ok": True,
    "save": True,
    "verbose": True,
    "workers": 4,
}
```

Figure 4-13 : Paramètres utilisés pour l'entraînement du modèle.

4.5.2 Surveillance d'entraînement avancées

Nous avons pensé à un système de surveillance et de reprise pour assurer la robustesse face aux interruptions fréquentes (coupures d'électricités, pertes de connexion, limites de la durée des sessions Kaggle).

A. **Sauvegarde structurée des résultats**

A la fin de chaque époque, les métriques suivantes sont automatiquement sauvegardées dans un fichier «results.csv» :

- La précision (metrics/ precision).
- Le rappel (metrics/recall).
- La moyenne de précision (mAP50, mAP50-95).

```
last_pt_path = f"{input_dataset}/last.pt"

if os.path.exists(last_pt_path):
    print(f"🔄 Reprise de l'entraînement depuis {last_pt_path}")
```

Figure 4-14 : Mécanisme de reprise d'entraînement depuis last.pt.

4.5.3 Structure de projet

Pour garantir une traçabilité, la gestion des versions, et l'export des résultats complète du projet, le système une organisation rigoureuse des répertoires :

- Répertoire principal de ce projet : **/kaggle/working/yolo_training.**
- Nom du projet : **filtred_coco_car_person.**
- Dossier de sortie des poids :
  /kaggle/working/yolo_training/filtred_coco_car_person/weights/

4.6 Conclusion

Ce chapitre a permis de mettre en place, un modèle YOLOv8 performant, pour la détection et le suivi des cibles immobilisés. Les étapes de préparation des données, d'entraînement et d'intégration avec DeepSORT ont été réalisées avec succès ; garantissant une détection fiable, en temps réel. Ce système constitue une base solide, pour la prévention des accidents aux passages à niveau.

Par ailleurs, dans le chapitre suivant, nous allons décrire les outils utilisés, tout au long, de notre projet et présenter les résultats obtenus.

Chapitre 05: Résultats et Interprétation

5.1 Introduction

Ce chapitre présente les résultats obtenus, à partir du système de détection créé ; il inclut des mesures de performances telles que les courbes de perte, les matrices de confusion et des représentations graphiques. Une analyse comparative entre les versions YOLOv8n et YOLOv8m est réalisée pour évaluer la précision de détection et les taux d'erreurs. Ces résultats offrent un aperçu précieux, de l'efficacité du système et de ses axes d'amélioration.

5.2 Outils de développement

5.2.1 Langage de programmation

Python : c'est un langage de programmation interprété, de haut niveau, orienté objet, reconnu pour sa syntaxe simple, sa facilité d'apprentissage et sa large extensibilité. Python est largement dans tous les domaines d'IA grâce à ses nombreuses bibliothèques spécialisées [77].

5.2.2 Bibliothèques

Open CV : c'est une bibliothèque open source, de traitement d'images et de la vision par ordinateur. Elle traite la capture d'images vidéo, dessine des boites englobantes autour des cibles détectés et affiche des textes informatifs sur les images [78].

Ultralytics : est une bibliothèque de détection d'objets basés sur YOLOv8. Il facilite l'entraînement du modèle avec une ligne de code (entraîner un modèle, faire des prédictions, évaluer les performances…etc.) [79].

PyQt5 : est une bibliothèque, spécialisée dans la création d'interfaces graphiques, en Python. Nous avons été motivés, du fait qu'elle soit capable de construire des fenêtres, qu'un utilisateur pourrait interagir avec ; d'afficher un champ de vision, et de communiquer entre les composants via des signaux et slots [80].

Numpy : Numpy est utilisé pour manipuler les coordonnées des boîtes englobantes, calculer les distances et gérer les matrices nécessaires aux algorithmes de suivi [81].

SQLite : est une bibliothèque légère de gestion de bases de données relationnelles intégrées. Son intérêt est de stocker les informations de l'utilisateur et l'historique de détections d'objets immobilisés [82].

FilterPy (Kalman) : est une bibliothèque Python afin d'intégrer les filtres de Kalman ; qui permet dans notre application, de prédire et de suivre la position des objets détectés, dans les images vidéos ; améliorant la performance du suivi [83].

Pygama : est une bibliothèque Python pour le développement des jeux, nous l'avons exploité pour la gestion audio, en déclenchant un son d'alarme lorsque l'objet détecté est arrêté [84].

5.2.3 Environnement de développement logiciel

Visual Studio Code : est un environnement de développement intégré développé par Microsoft pour écrire et gérer les codes, dans le but de créé des applications web, applications bureau, jeux vidéo ou d'autre types de projets [85].

Google Colab : est un service cloud, offert par Google, basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner les modèles directement dans cloud [86].

Kaggle : est une plateforme communautaire en ligne, elle offre un accès à des jeux de données publiques, des environnements de travail collaboratifs, des GPU gratuits pendant une durée de 30 heures, idéal pour les projets d'apprentissage [87].

5.2.4 Environnement Matériel

Tableau 5-1 : Configuration matérielles des ordinateurs utilisés.

Matériel	Processeur	Mémoire RAM	Carte graphique
DELL-Desktop	12th Gen Intel(R) Core (TM) i7-1255U 1.70 GHz	16 Go, 3200 MHz	128 MB Intel(R) Iris(R) Xe Graphique
DESKTOP-FI1RE3S	Intel(R) Core (TM) i5-7300 CPU 2.60 GHz	8 Go, 2133 MHz	Aucune VRAM dédiée

5.3 Résultats d'entraînement du modèle YOLOv8m

Dans cette section, nous représentons les résultats obtenus lors de l'entraînement du modèle YOLOv8m, sur l'ensemble de données. Le suivi de l'apprentissage a été effectué sur plusieurs époques, en enregistrant les courbes de perte, pour les jeux d'entraînement et de validation. Les performances du modèle sont également évaluées à l'aide de courbes de précision, de rappel, ainsi que des métriques mAP0.5 et mAP@0.5:0.95.

5.3.1 Perte de localisation

Cette métrique mesure, à quel point les boîtes détectées correspondent aux emplacements réels des objets ; comme illustrée ci-dessous :

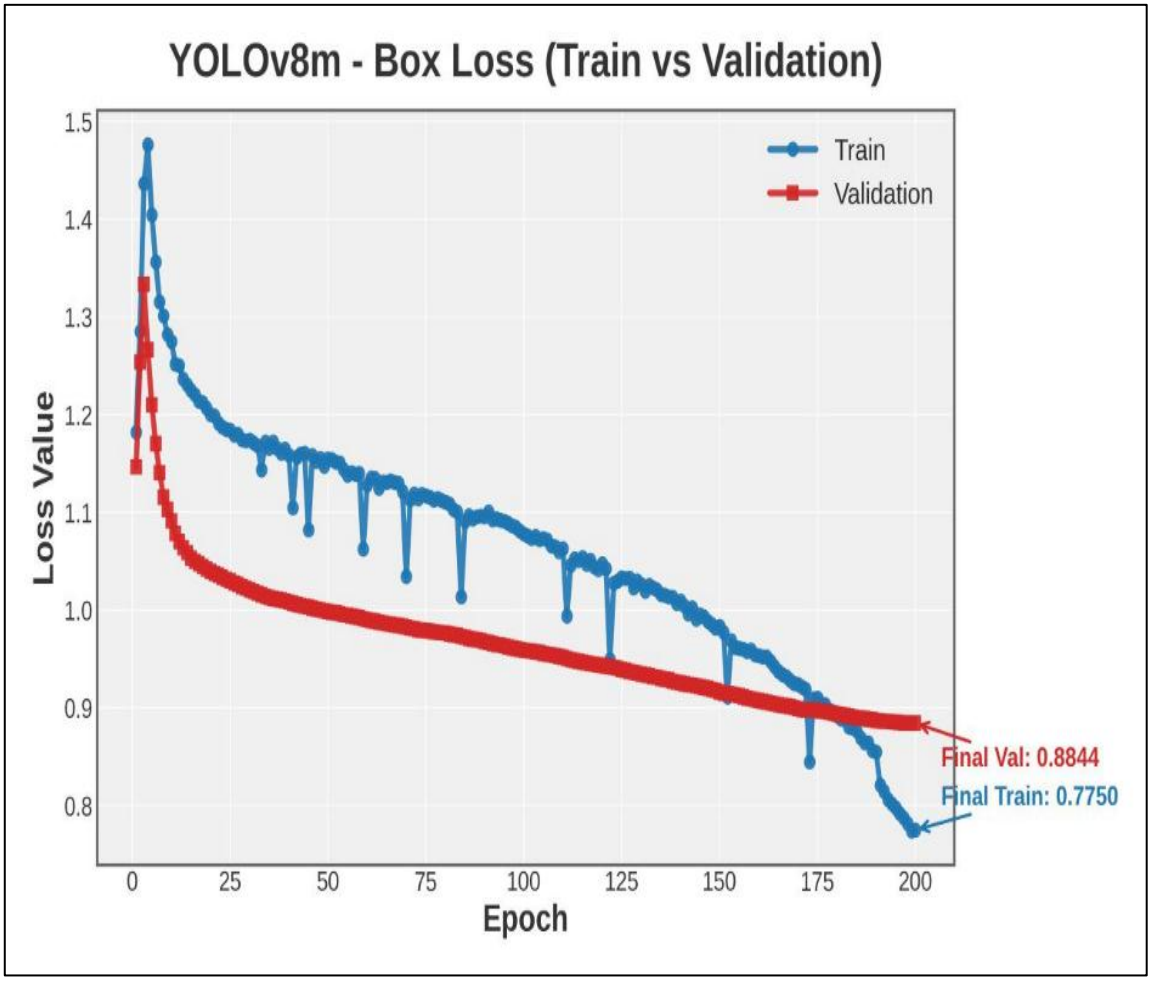


Figure 5-1 : Courbe de perte de localisation du modèle YOLOv8m pour les ensembles d'entrainement et de validation.

A. La courbe « Bleue » (Train/Box_loss)

Elle représente la progression de la perte d'entraînement, liée à la précision des boîtes englobantes prédites ; comme celle représentée au-dessus :

Au début, la perte commence aux alentours de 1.2, à l'époque 0, puis augmente légèrement jusqu'à dépasser 1.4 dans les premières époques ; car le modèle n'a pas encore bien appris, à localiser les objets dans les images.

A partir de l'époque 10, nous observons une diminution rapide de la perte ; cela signifie que le modèle commence à apprendre, à ajuster les boîtes de détection, autour des « voitures » et « personnes », et que celles-ci deviennent de plus en plus précises.

Après l'époque 25, la perte continue à descendre, mais de manière plus modérée ; indiquant une amélioration progressive de la précision de localisation. Nous constatons, à ce moment-là, l'existence de discontinuités, dues à la reprise de l'entraînement après l'interruption.

A noter, par ailleurs, que la courbe se stabilise aux alentours de 0.77, lorsque elle siège aux époques 175 à 200 ; ce qui explique que le modèle atteint un équilibre, dans sa capacité, à détecter les voitures et les personnes, avec une bonne précision. Cette stabilisation indique que le modèle a atteint le niveau de réussite le plus élevée sur ce jeu de données.

B. La courbe « Rouge » (Val/Box_loss)

Elle désigne la progression de la perte de validation, liée à la localisation des objets ; elle est légèrement plus basse que celle de l'entraînement (courbe Bleue) ; comme celle illustrée sur la figure 5-1 :

Durant les premières époques de 0 à 25, nous remarquons une hausse, suivie d'une chute rapide de la perte (1,15 à 1,05). Ceci signifie que le modèle commence à ajuster ses prédictions et améliore très vite ses performances de localisation.

Entre les époques 25 et 175, la courbe continue à diminuer progressivement (1,05 à 0,88), ceci prouve que le modèle devient de plus en plus précis, ajustant la position des boites autour des objets cibles «voitures» et « personnes».

Au-delà, de l'époque 175, la courbe se stabilise autour de la valeur 0.8844, ce qui prouve que le modèle s'est amélioré beaucoup plus, en stabilité et en fiabilité (pas de sur-apprentissage).

5.3.2 Perte de classification

La perte de classification pendant l'entraînement du modèle, évalue la capacité du modèle à différencier les classes des objets détectés ; comme celle représentée sur la figure 5-2 :

A. La courbe « Bleue » (Train/Cls_loss)

Cette courbe symbolise l'évolution de la perte de classification sur les données d'entraînement ; comme celle illustrée sur la figure 5-2 :

La hausse de cette courbe, qui va de 1.2 jusqu'à dépasser 1.5 ; est dû au faite que le modèle ne sait pas encore distinguer les classes. Ce qui induit quelques erreurs de classification. À partir de l'époque 25, la courbe baisse de manière régulière, ce qui indique que le modèle apprend progressivement, à mieux classer les « voitures » et les « personnes ».

Au-delà, de l'époque 50, cette diminution progressive, néanmoins constante de la perte, montre que le modèle comprend de mieux en mieux les objets à détecter ; ce qui est essentiel pour des systèmes, en temps réel. Des discontinuités persistent de manière comparable à celle de la courbe « Bleue » (**Train/Box_loss**).

Vers les époques 175 à 200, la courbe se stabilise, autour de 0.48. Cette stabilisation signifie que le modèle n'apprend plus, de nouvelles informations ; cela prouve qu'il a atteint une performance maximale, en classification, pour ce jeu de données.

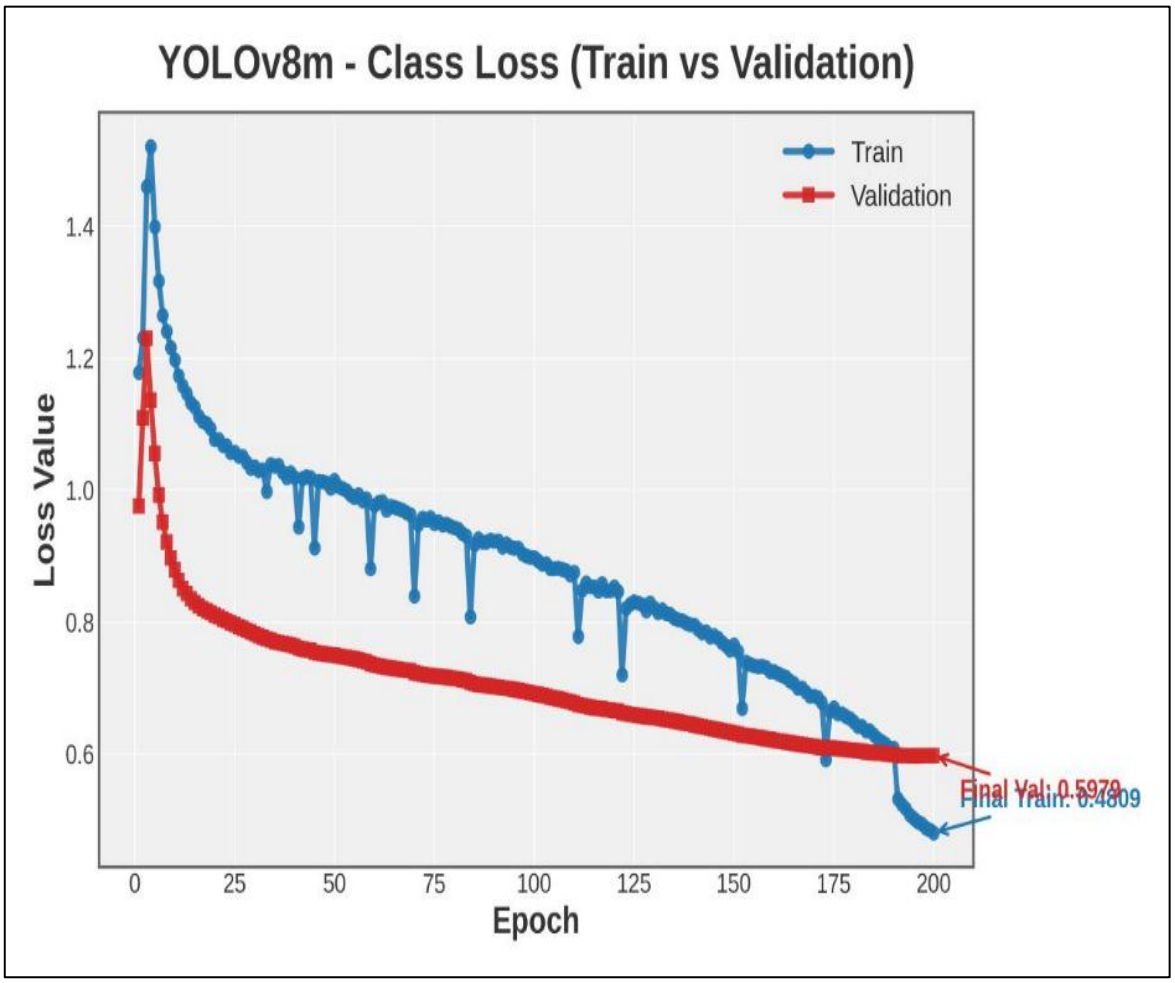


Figure 5-2 : Courbe de perte de classification du modèle YOLOv8m pour les ensembles d'entrainement et de validation.

B. La courbe « Rouge » (Val/Cls_loss)

Cette courbe reflète l'évolution de la perte de classification sur les données de validation pendant l'entraînement ; comme celle illustrée sur la figure 5-2 :

Nous observons de l'époque 0 à 10, une légère augmentation de la perte (0,99 à 1,23), en raison des prédictions aléatoires. Ensuite, entre les époques 10 et 25, la courbe affiche une

forte baisse, indiquant que le modèle commence à bien reconnaître les classes des objets.

De l'époque 30 à 175, la diminution devient plus lente mais régulière ; le modèle continue à apprendre progressivement.

Enfin, après l'époque 175, la perte se stabilise autour de 0.59, ce qui signifie que le modèle commet peu d'erreurs de classification sur les données de validation. Cette stabilisation signifie que le modèle a bien appris à reconnaître les objets sans erreurs.

Dans notre entraînement, les croisements observés entre nos courbes de perte d'entraînement et de validation reflètent une légère variation naturelle dans la dynamique d'apprentissage entre nos données d'entraînement et de validation. Ce phénomène est courant et ne signifie pas forcément un problème. Tant que nos courbes restent proches, cela montre que notre modèle s'améliore de manière stable et généralise bien. Les fluctuations et croisements mineurs sont normaux dans notre processus d'optimisation et n'affectent pas négativement la performance globale.

5.3.3 Métrique de précision

La courbe de précision de la figure 5-3, débute par une légère baisse arrivant jusqu'à 0.65, puis remonte rapidement dès les premières époques, comprise entre 5 et 25, indiquant que le modèle apprend à éviter les fausses détections. Par la suite, la courbe avance de façon constante et modérée, pour se stabiliser aux alentours de 0.82. Cette progression suggère que le modèle devient fiable au fil du temps dans ses prédictions.

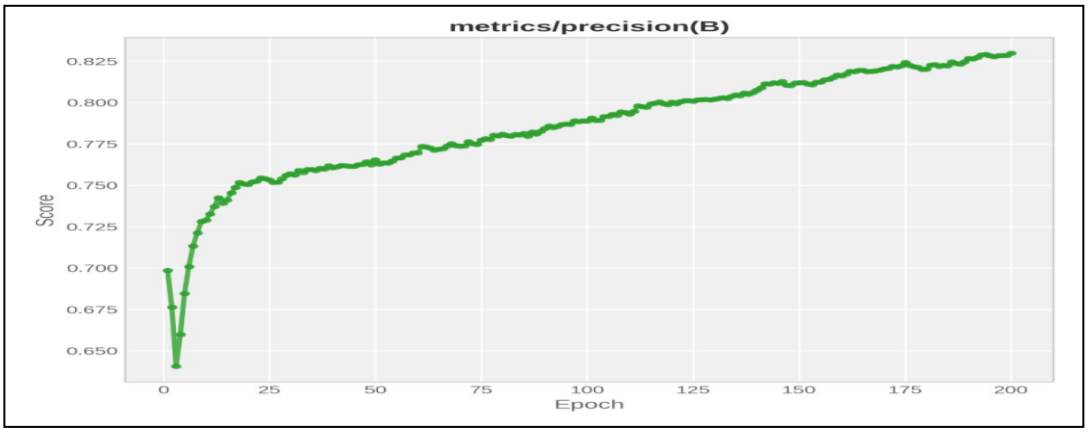


Figure 5-3 : Evolution du score de la précision.

5.3.4 Métrique de rappel

Dans le graphisme de la figure 5-4, la courbe est comparable à celle de la courbe précédente, avec des chiffres du score différentes ; à l'époque 30, la montée devient plus lente mais régulière, jusqu'à ce qu'elle se stabilise aux alentours de 0.7353, cela signifie que le modèle repère la majorité des **«** voitures » et « personnes » présents dans les images.

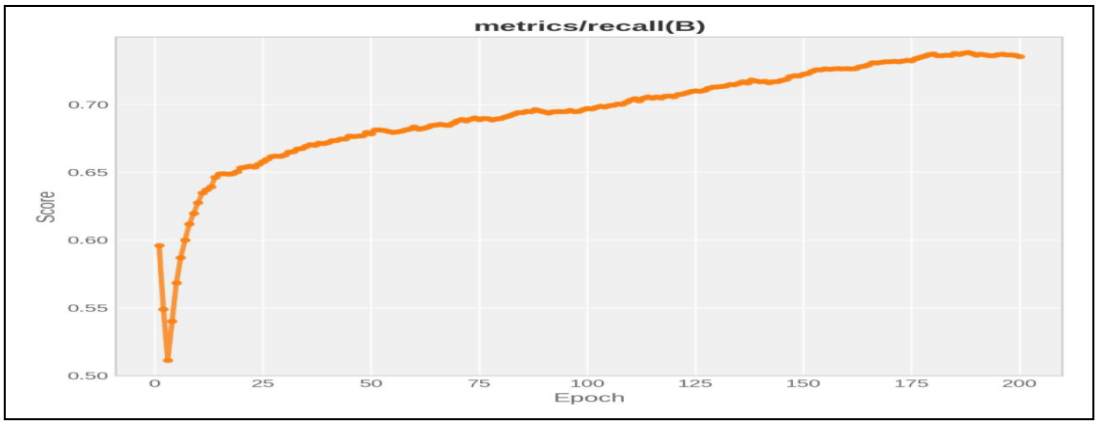


Figure 5-4 : Evolution du score du rappel.

5.3.5 Précision moyenne à un seuil

On note des perturbations visibles, vraisemblablement en rapport avec un apprentissage instable, en début d'étude. Par la suite, de 10 à 75 la courbe remonte très rapidement, ce qui déduit un apprentissage rapide du modèle. Au delà de 75 la progression se fait de manière lente ; finalement le mAP50 passe de 0.55 à 0.82, ce qui prouve un très bon niveau de précision atteint, en fin d'entraînement. La courbe finit par être plate à la fin, ce qui suggère que le modèle est arrivé à une solution stable ; comme le montre cette figure :

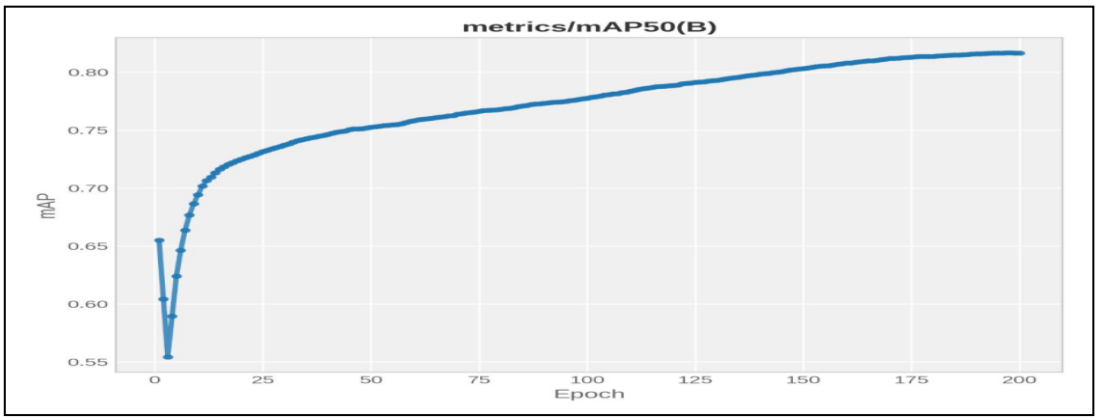


Figure 5-5 : Evolution de la courbe mAP@50.

5.3.6 Précision moyenne à plusieurs seuils

On constate dans la figure 5-6, une progression régulière des performances du modèle, tout au long, de l'entraînement. Elle est comme la courbe précédente, dont la seule différence est que l'autre utilise IoU de 50% ; par contre celle representée en-bas est de 50 à 95 % ; ce qui atteste qu'elle donne l'idée la plus complète de la performance du modèle. Sa progression atteint une mAP d'environ 0.60 à partir de 175 époques.

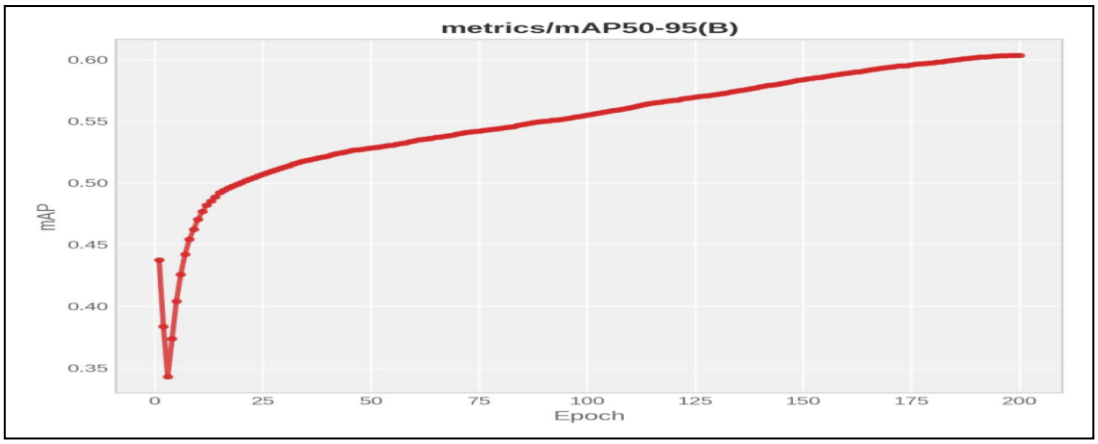


Figure 5-6 : Evolution de la courbe mAP@50-95.

les résultats finaux, du modèle YOLOv8m, avec une précision de 83%, un rappel de 73%, un mAP@50 de 82% et un mAP@50-95 de 60% après 200 époques d'entraînement ; indiquent une très bonne capacité de détection et de localisation car ils montrent une convergence stable sans sur-apprentissage avec une progression ameliorée des performances.

5.4 Analyses des résultats d'évaluation sur l'ensemble test

5.4.1 Matrice de confusion

Les valeurs situées sur la diagonale principale de la figure 5-7, correspondent aux prédictions correctes, c'est-à-dire que la classe prédite, pour chaque objet, est identique, à celles qui sont réelles.

- ✓ Le modèle a correctement détecté 18 297 personnes.
- ✓ Il a très bien identifié 23 428 voitures.

Ces chiffres, témoignent, le fait que le modèle fonctionne bien ; car il fait beaucoup plus de prédictions justes que d'erreurs.

Nommions, les valeurs siégeant en dehors de la diagonale, sont des erreurs de classification :

- ✓ 6245 fois, qu'une personne réelle n'a pas été détectée ; le modèle l'a confondu avec l'arrière-plan (faux négatif).
- ✓ 3061 fois, qu'une voiture réelle a été ignorée ; il l'a confondu avec l'arrière-plan (faux négatif).

Le modèle a également détecté des objets là où il n'y en avait pas :

- ✓ 7061 fois, l'arrière fond a été pris pour une personne.
- ✓ 4969 fois, l'arrière fond a été considéré comme une voiture.

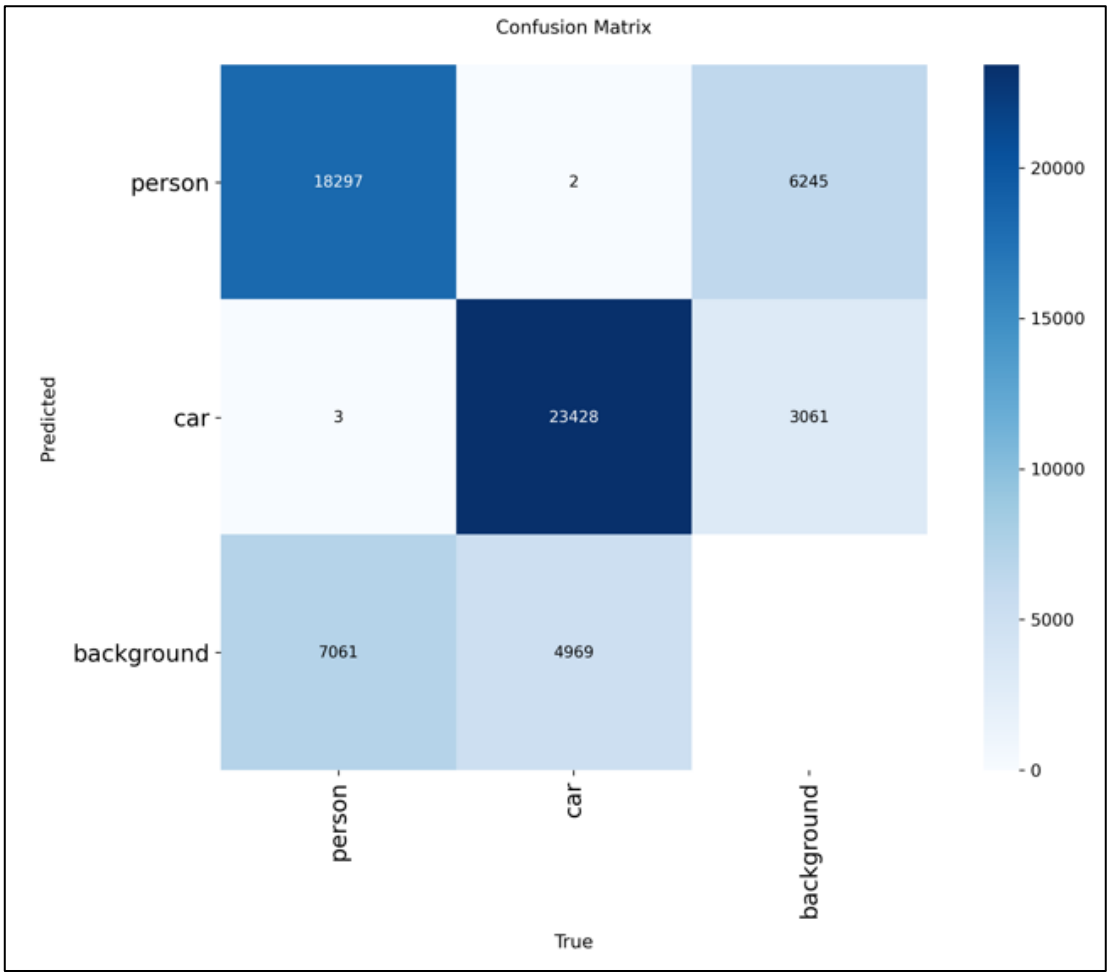


Figure 5-7 : Matrice de confusion du modèle de classification.

5.4.2 Matrice de confusion normalisée

Chaque surface d'un carrée évalue les performances du modèle, en indiquant, le pourcentage de correspondance ; voici une analyse de la figure 5-8 :

- ✓ 72 % des personnes sont correctement, classées comme personne.
- ✓ 82 % des voitures sont correctement, classées comme voitures.

Ces pourcentages prouvent que le modèle reconnaît bien les classes voitures et personnes, mais a un peu de mal, à identifier correctement, les arrière fond.

- ✓ 67 % des personnes réelles ont été prises pour l'arrière-plan (faux négatif).
- ✓ 33 % des voitures réelles ont été ignorées et classées comme l'arrière-plan (faux négatif).

Le modèle a également détecté des objets là où il n'y en avait pas :

- ✓ 28 % des cas, le modèle confond le fond avec une personne réelle (faux positif).
- ✓ 17 % des cas, le modèle confond le fond avec une voiture réelle (faux positif).

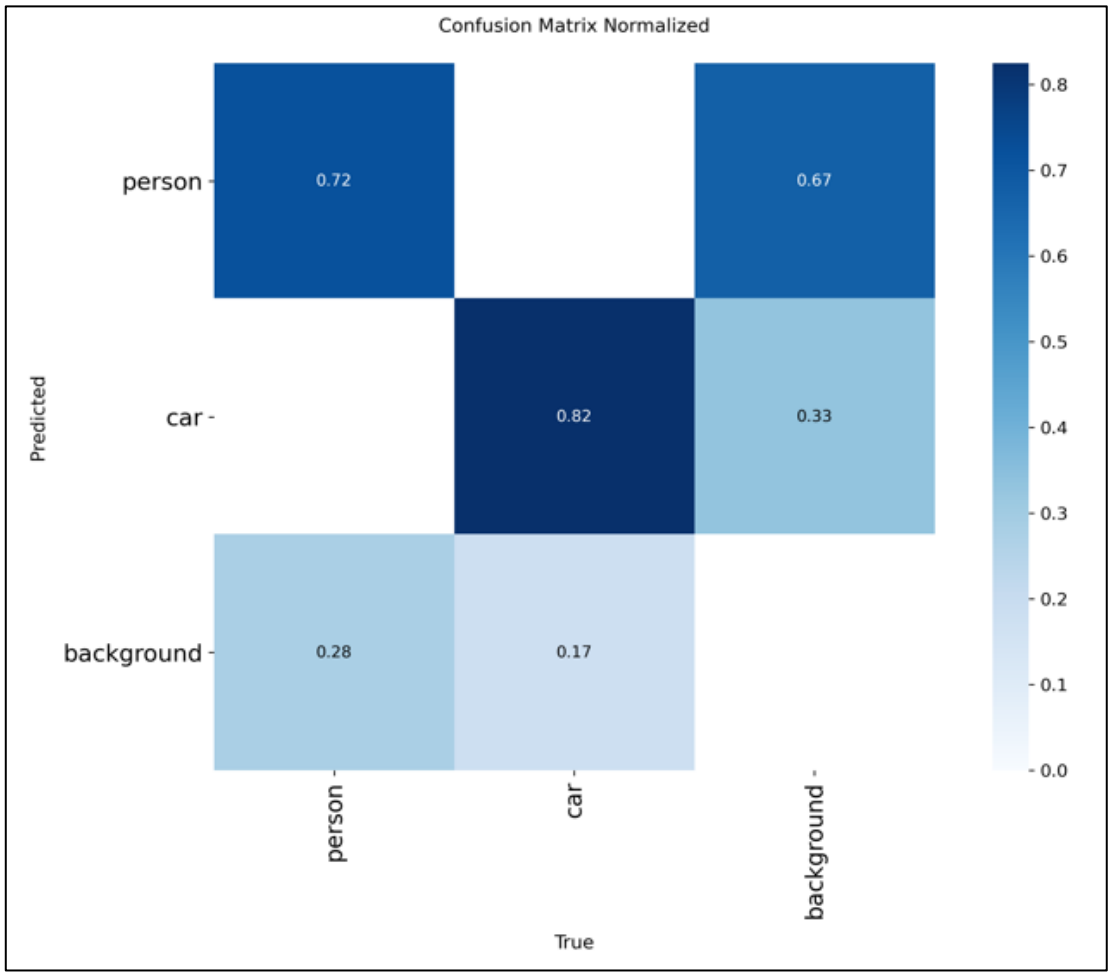


Figure 5-8 : Matrice de confusion normalisée du modèle de classification.

5.4.3 Courbe F1-score

Notre meilleur score F1 est 0,79 (voir figure 5-9) ; atteint pour un seuil de confiance de 0,284, qui équilibre bien la précision et le rappel. La courbe est stable entre les seuils 0,1 et 0,4 ; ce qui nous laisse une marge pour ajuster le seuil selon ce qu'on veut faire.

Lorsque le seuil de confiance est inferieur ou égal à 0,4 ; une confiance faible correspond à un rappel élevé mais une précision moindre ; tandis qu'une confiance plus élevé améliore la precision au détriment du rappel.

Après le seuil de confiance de 0,4 ; la courbe diminue rapidement, ce qui signifie que lorsque le seuil de confiance augmente au-delà de 0,4 ; le nombre d'objets détectés (rappel) diminue énormement, car le modèle devient strict et n'accepte que les détections, dont il est très sur. En meme temps, la précision n'augemente plus beaucoup, car la plupart des erreurs ont déjà été éliminées avant ce seuil.

Il faut donc choisir un seuil de confiance, qui équilibre bien précision et rappel, ici à 0,284, pour avoir la méilleure performance.

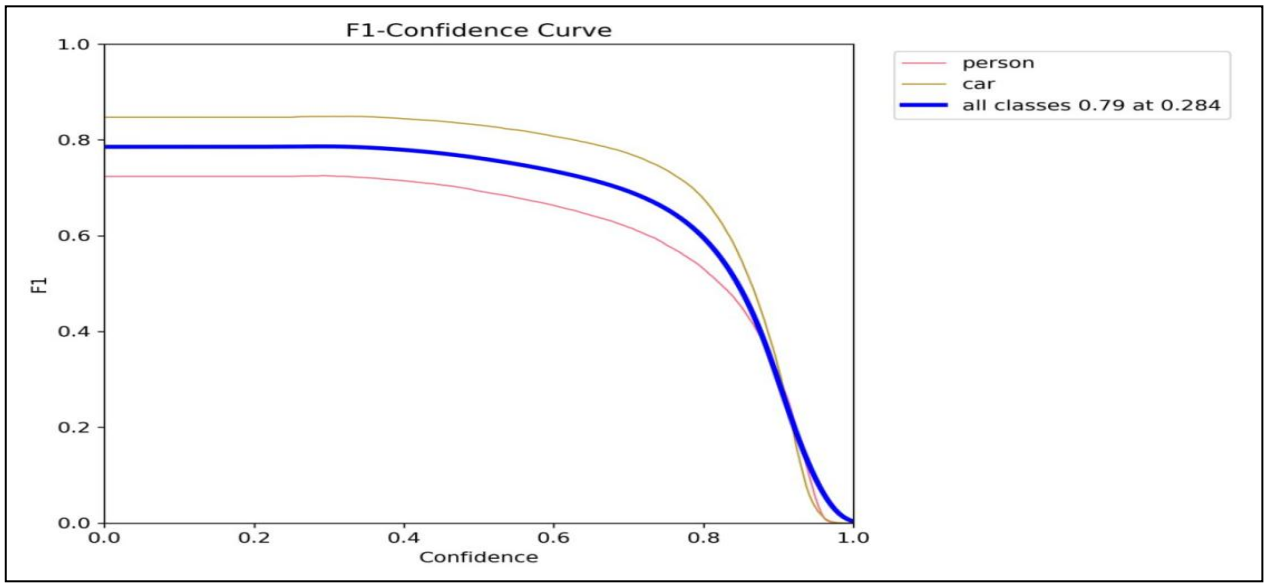


Figure 5-9 : Courbe F1-score du seuil de confiance pour chaque classe.

5.4.4 Courbe précision et rappel

Nos courbes de la figure 5-10, montrent que la précision est très haute (preseque 1), quand le rappel est faible (jusqu'à environ 0,3-0,4). Cela veut dire que lorsque notre modèle est sûr de ses détections, il fait peu d'erreurs. La précision baisse doucement, quand le rappel augmente ; ce qui montre un bon équilibre entre la fiabilité des détections et la quantité d'objets trouvés. La classe « voiture » a un score mAP de 0,887, ce qui est très bon, tandis que la classe « personne » a un score un peu plus bas 0,766.

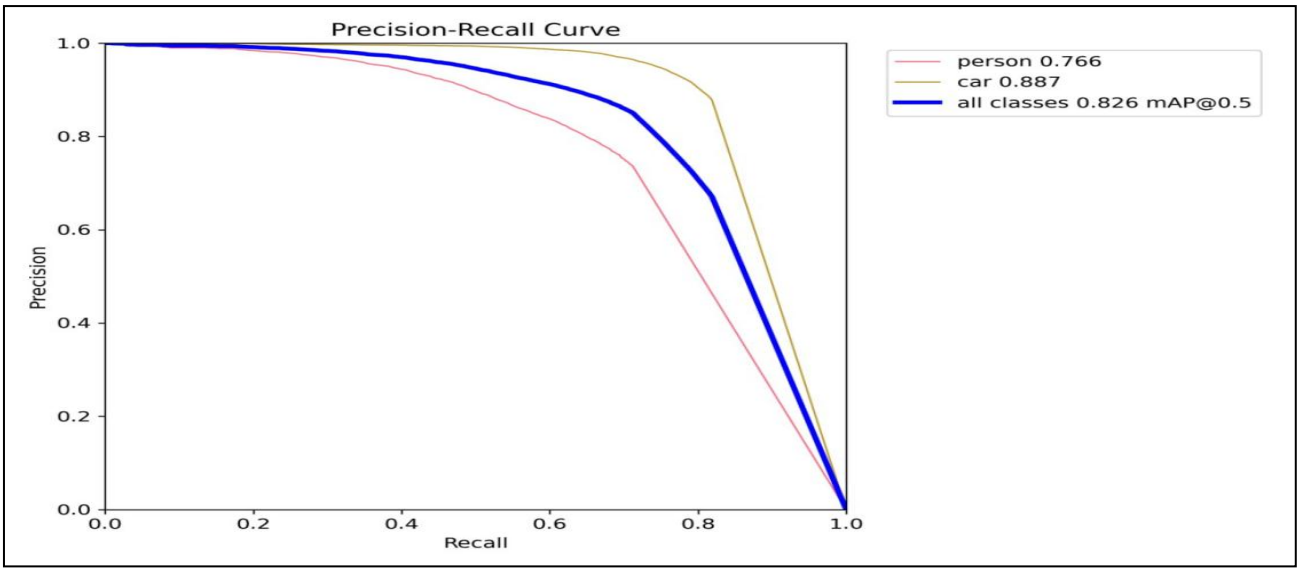


Figure 5-10 : Courbe précision-rappel pour les classes personne et voiture avec mAP global à 0.5.

5.4.5 Courbe confiance de précision

Nos courbes de la figure 5-11, montrent que la précision augmente régulièrement avec la confiance. La «confiance » est un score ; que notre modèle donne à chaque détection, afin

d'indiquer qu'il est sûr de sa prédiction, en commençant par de 0 (pas du tout sûr) à 1 (totalement sûr) ; à confiance zéro, la précision est déjà aux alentours de 0,8 ; ce qui prouve que même avec des détections les moins sûres, sont souvent correctes. La précision atteint 1,0 (parfaite) quand la confiance est au maximum. Cela veut dire que plus notre modèle est sûr de sa détection, plus celle-ci est fiable.

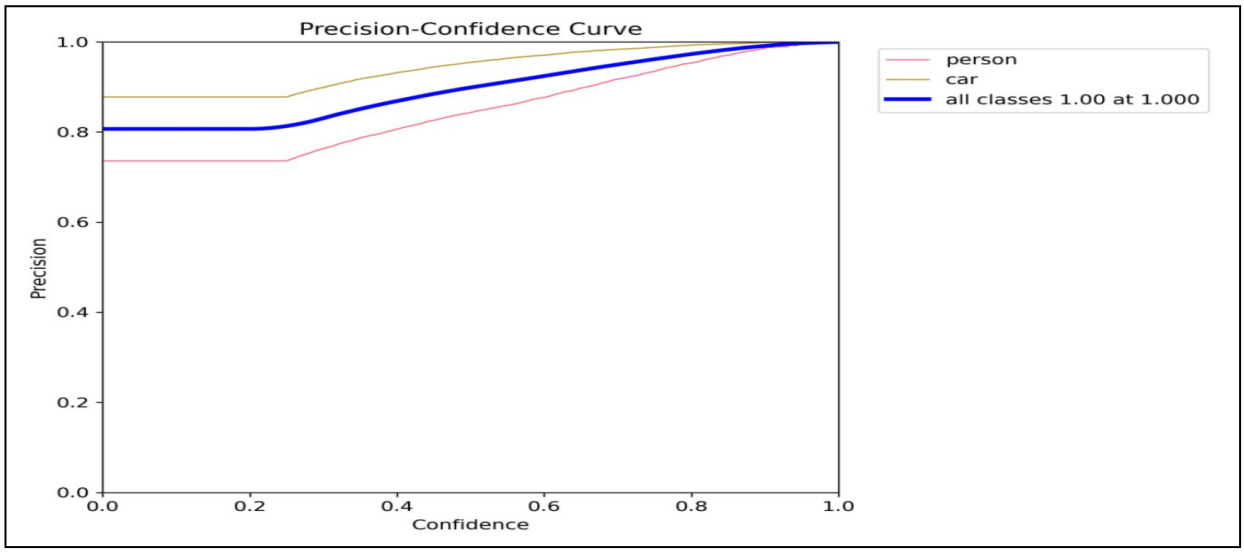


Figure 5-11 : Courbe de précision en fonction du seuil de confiance pour chaque classe.

5.4.6 Courbe confiance de rappel

Le rappel commence à 0,77 quand la confiance est zéro, puis baisse doucement jusqu'à 0, quand la confiance est la plus elevée. Ici aussi, la « confiance » est le score de certitude du modèle, de 0 à 1. Cette baisse régulière montre que les scores de confiance sont bien répartis ; ce qui nous aide à choisir un bon seuil de confiance sans perdre trop d'objets détectés.

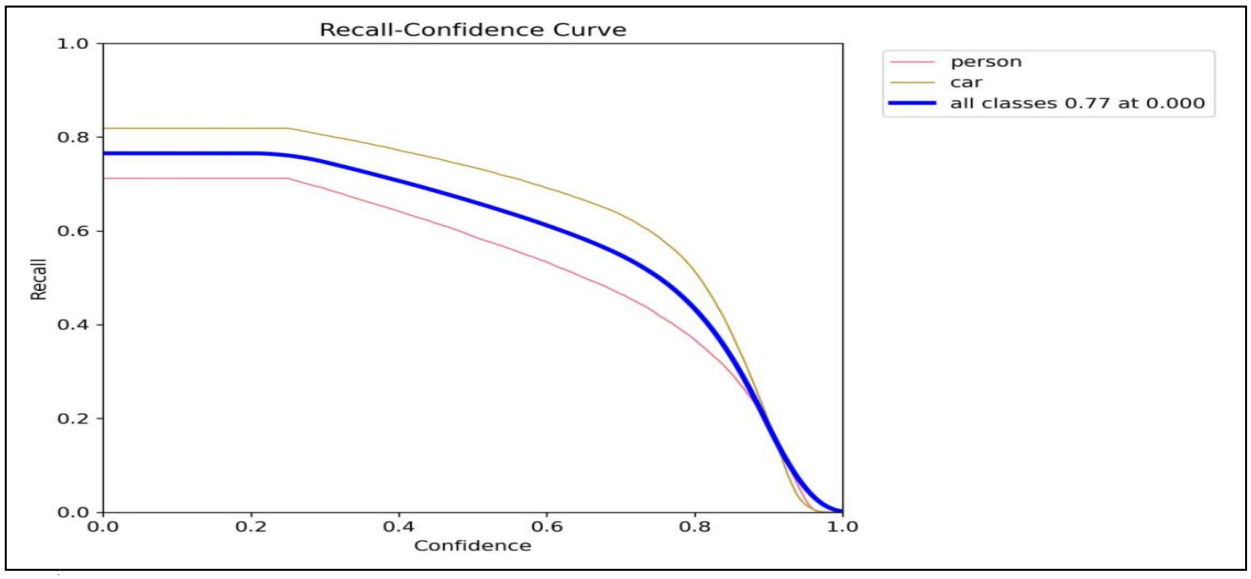


Figure 5-12 : Courbe de rappel en fonction du seuil de confiance pour chaque classe.

Selon ces courbes étudiées, Les résultats obtenus, lors de cette évaluation ; font que le modèle donne de bon résultats, une précision 82,42 %, un rappel 75,27 %, un F1-score 78,68 %, un mAP50 82,61 %, un mAP50-95 66,44 % et une mIoU 82,61 %. Cela veut dire, que notre modèle, trouve correctement environ 82,6 % des objets quand on utilise un seuil de confiance de 0.5. C'est une bonne performance pour une application, en temps réel.

Voici un tableau qui montre les résultats de chaque classe :

Tableau 5-2 : mesure de performance pour chaque classe.

mesures de performance	Personnes	Voitures
Précision	75,52 %	89,32 %
Rappel	69,65 %	80,88 %
F1-score	72,47 %	84,89 %
IoU	76,56 %	88,65 %

5.5 Comparaison entre les variantes

Nous avons effectué une comparaison entre les deux variantes du modèle YOLOv8, dans le but de savoir leurs performances, qui ont été justifié selon plusieurs critères comme la précision, la vitesse d'exécution et la détection dans de multiples conditions (luminosité, noirceur, la distance lointaine). Les résultats ont montré que l'une excelle dans la rapidité de détection et l'autre dans sa précision.

La variante « nano » est plus rapide lors de l'entraînement, mais elle est moins précise. En revanche, l'autre variante « medium » offre une meilleure précision, dans la détection et la localisation, mais prends beaucoup plus de temps lors de l'entraînement (environ 30 minutes par époque).

Tableau 5-3 : Comparaison de la précision et du rappel entre YOLOv8-nano et YOLOv8- medium.

YOLOv8-nano		
Classes	**Précision**	**Rappel**
Tous	80,8%	63,3%
Personne	82,2%	67,7%
Voiture	79,4%	59%
YOLOv8-medium		
Classes	**Précision**	**Rappel**
Tous	82,42 %,	75,27 %,
Personne	75,52 %	69,65 %
Voiture	89,32 %	80,88 %

Tableau 5-4 : Comparaison des performances mAP entre YOLOv8-nano et YOLOv8- medium.

YOLOv8-nano		
Classes	**mAP50**	**mAP50-95**
Tous	72,8%	51,6%
YOLOv8-medium		
Classes	**mAP50**	**mAP50-95**
Tous	82,61 %,	66,44 %

5.6 Présentation de l'application bureau

Nous avons créé une application du bureau, qui permet de visualiser le trafic circulatoire automobile et de capturer le moindre mouvement des sujets, évoluant dans le champ de vision de la caméra ; notamment les passant sur les rails. Cette application permet aussi, de déclencher des alertes d'avertissement au conducteur de la locomotive, devant des comportements humains irréfléchis parfois mal intentionnés ; ainsi que des circonstances particulières de la mécanique des automobiles, pouvant survenir en toute circonstance.

L'objectif final est d'éviter les catastrophes entre trains, piétons et voitures. Ce système minimise les risques de blessures, les chocs psychologiques et les pertes matérielles, liés à ces accidents. L'application, contient plusieurs interfaces parmi lesquelles :

L'interface de connexion présentée dans la figure suivante, permet au responsable de la vidéo surveillance, de saisir ses informations personnelles (nom d'utilisateur et un mot de passe).

Figure 5-13 : Interface de connexion

Une fois l'utilisateur connecté à son compte via la page de connexion, l'interface principale de notre système s'affiche. Celle-ci possède un bouton qui permet d'activer le système de détection et de suivi.

Une fois la détection activée, l'utilisateur accède à une vue de l'environnement capté par la caméra, qui permet de visualiser, en temps réel, les voitures et les piétons ; passant sur les rails d'un passage à niveau ; de les détecter en utilisant l'algorithme YOLOv8 sous forme d'un cadrage ; et de les suivre grâce à l'algorithme DeepSORT, en leur attribuant un identifiant unique ; comme l'indique la figure 5-15 :

Figure 5-14 : Interface de la vidéosurveillance pour la détection d'obstacles.

Lorsqu'un véhicule s'immobilise, pendant plus de deux minutes, le statut s'affiche en rouge et le système déclenche une alerte. Dans le cas contraire, le statut s'affiche en vert, cela prouve que la situation est normale ; comme illustrée ci-dessous :

Figure 5-15: Interface d'alerte en temps réel pour objets immobilisés

Chaque immobilisation est enregistrée dans l'interface « historique de détections » qui marque la date, l'heure, l'identifiant de chaque objets arrêté et son taux de confiance. Cette interface permet d'effacer un ou tous les enregistrements (voir figure 5-16).

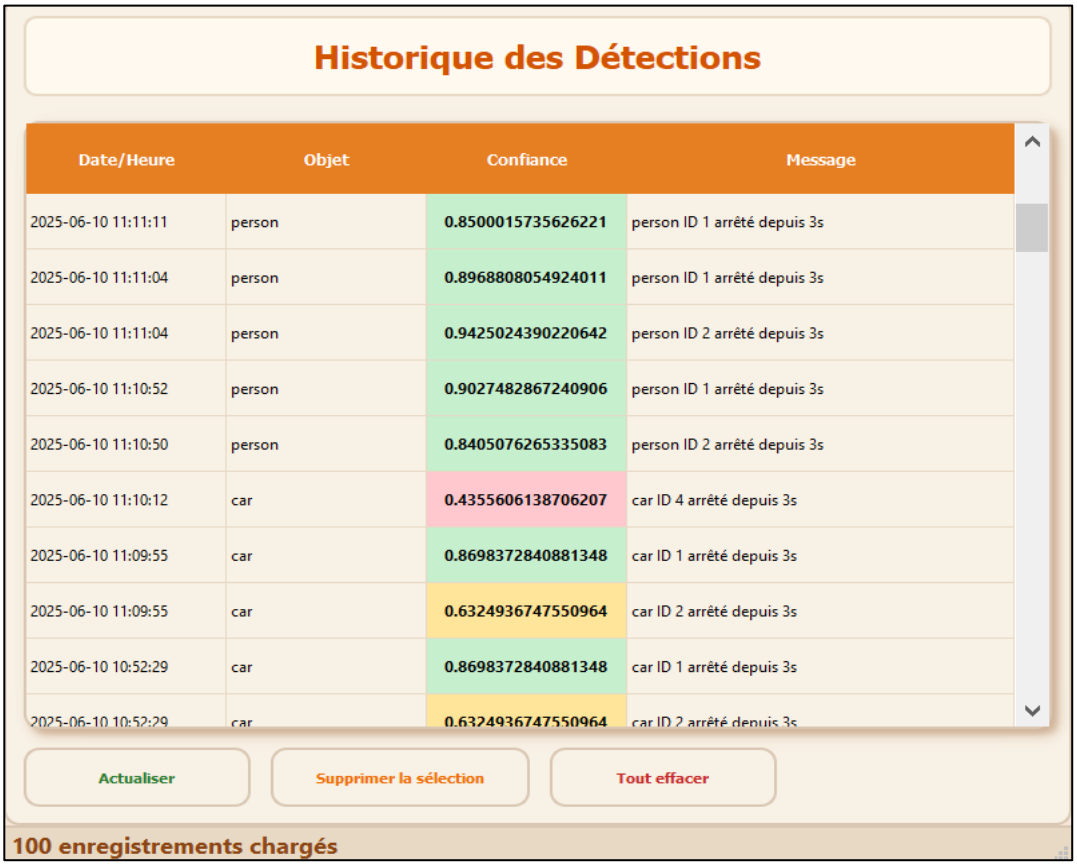


Figure 5-16 : Interface de l'historique des détections.

5.7 Tests effectués par l'interface graphique

Nous avons effectué plusieurs tests grâce à notre interface graphique dans des conditions variées. Comme le montre ses figures suivantes :

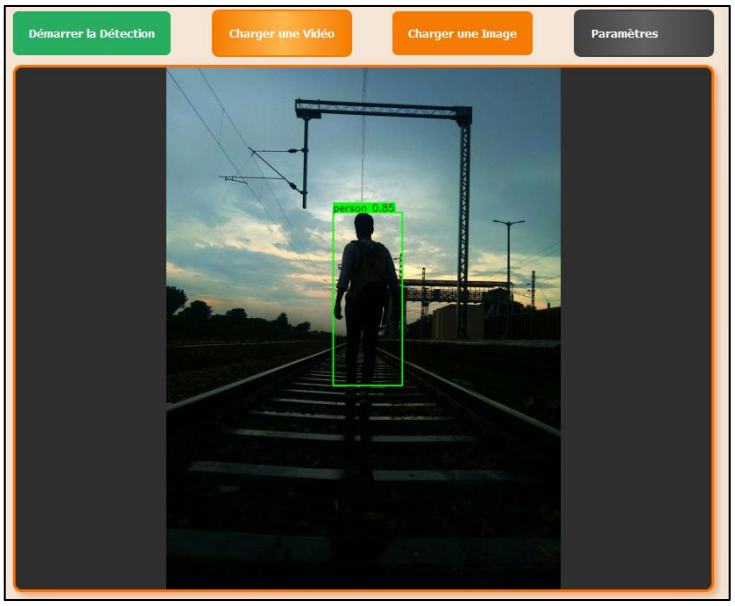


Figure 5-17 : Détection d'objets en état noirceur.

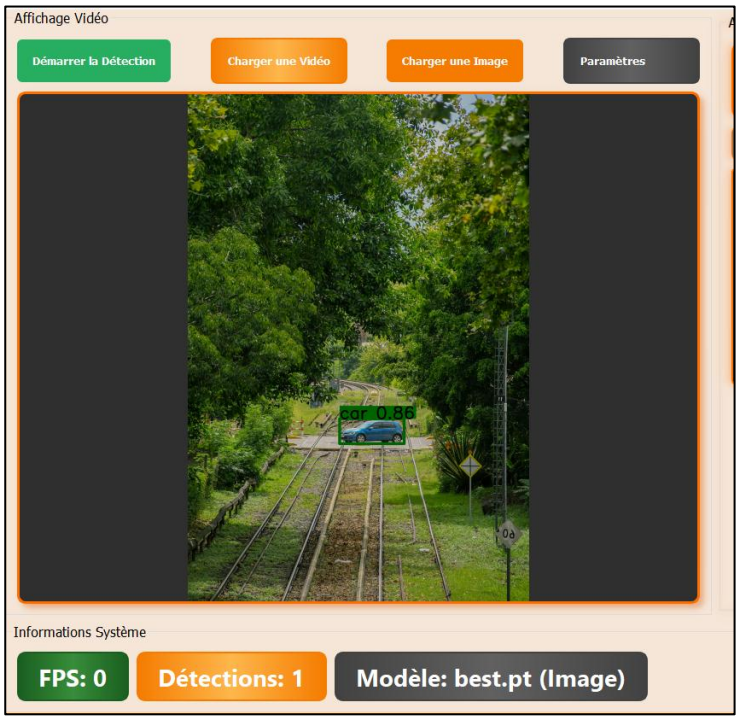


Figure 5-18 : Détection d'objets à longue distance.

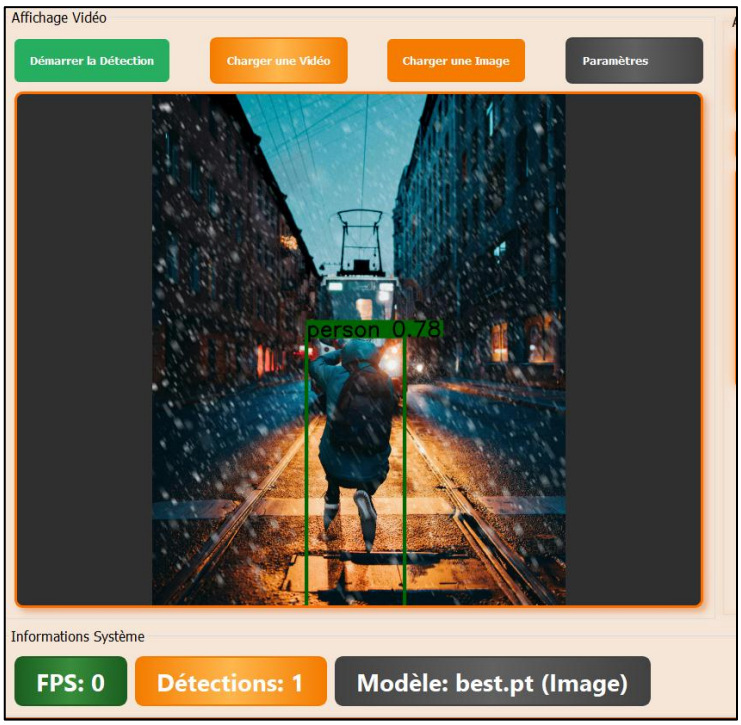


Figure 5-19 : Détection d'objets selon les conditions météorologique.

5.8 Contrainte de notre étude

Premièrement, nous avons utilisé des plateformes en ligne, comme Kaggle ; malgré cela, cette solution présentait plusieurs inconvénients ; des coupures, des blocages et des limites dans la durée des sessions, ce qui perturbe l'entraînement du modèle ; mais nous les avons surmonté, en créant des scripts de codes puissant.

Des interruptions survenaient à tout moment, durant l'entraînement. A chaque fois que c'était le cas, nous étions obligé de télécharger les fichiers sauvegardés, comme le fichier «best.pt», « last.pt » et « results.csv» afin de poursuivre l'entraînement sans revenir en arrière, c'est-à-dire sans commencer depuis l'époque 0. De plus, chaque époque prenait plus de 30 minutes à s'exécuter, ce qui ralentit le processus d'entraînement, prenant en tout 100 heures (COCO2017 filtré est très volumineux avec 18 Go d'espace de stockage).

Deuxièmement, le Google Drive ne permettait pas de télécharger tout l'ensemble de données pour entraîner notre modèle sur la plate-forme Google Colab, nous étions obligé de télécharger des parties de celui-ci car son espace de stockage, était limité à seulement à 15 Go ; alors que le jeu de donnée est 18 Go. Par ailleurs, le fait que nous ne possédions pas de connexion par fibre optique, cela rendait le téléchargement excessivement lent.

5.9 Conclusion

Les résultats mettent en relief, les points forts et les points faibles, du système de détection, à travers une évaluation complète des performances. La comparaison entre YOLOv8n et YOLOv8m, atteste des différences, du point de vue, précision et taux d'erreurs.

Ces observations sont essentielles pour améliorer la détection et guider les développements futurs. Ce chapitre valide les capacités du système afin de pouvoir déployer, sur le terrain des améliorations à venir.

Chapitre 06: Conclusion et travaux futures

6.1 Conclusion

Durant ces dernières années, nous avons constaté une problématique majeure, concernant la sécurité ferroviaire et routière ; qui se traduit par des accidents graves, aux conséquences dramatiques ; se produisant suite à des véhicules immobilisés sur des rails aux passages à niveau ; dues à des embouteillages, des pannes et des comportements irréfléchis des usagers, en Algérie.

Avec l'avènement récent des technologies d'intelligence artificielle et de vision par ordinateur, des systèmes performants pour la détection et le suivi d'objets dans divers environnements ; ont vu le jour, afin de renforcer la sécurité dans ces zones critiques.

C'est dans ce contexte précis, que ce mémoire a pour objectif, de décrire la création d'un système optimisé, de détection et de suivi, en temps réel, des cibles immobilisées, sur les rails d'un passage à niveau ; en utilisant l'algorithme YOLO, plus précisément sa version huit pour la détection ; qui offre un équilibre avantageux et harmonieux, entre précision et rapidité [7] ; combiné à DeepSORT pour le suivi.

Dans un premier temps, nous avons déployé des efforts considerables, pour collecter les informations nécessaires et mener des études approfondies ; afin de bien cerner le contexte et les enjeux liés à la problématique.

Dans un second temps, nous avons analysé les algorithmes de détection traditionnelles et modernes ainsi que les diffrentes évolutions du modèle YOLO.

Enfin, dans un troisième temps, nous avons détaillé la méthodologie de notre système, afin de répondre à la problématique du sujet ; en commencant par une préparation minutieuse du jeu de données, en le filtrant. Une augmentaion des données a été appliquée dynamiquement, durant l'apprentissage afin d'équilibrer les classes cibles (voitures et personnes).

Par ailleurs, nous avons fait une étude comparative, avec deux variantes YOLOv8n et YOLOv8m ; dans le but d'améliorer la précision, preuve à l'appui ; avec la variante « medium » qui s'est révelée plus performante.

Une interface graphique conviviale ; tolérée par les utilisateurs, peu ou pas familiers, avec les nouvelles technologies ; a été conçue en utilisant le langage Python ; permettant de visionner les lieux de passage et de surveiller les mouvements suspects des sujets, présents

dans le champ de vision de la caméra, afin de générer des alertes de prévention en cas d'immobilisation prolongée.

Les résultats finaux, qui en découlent, font que le modèle YOLOv8 s'est amélioré en performance, comme la précision 82,42 %, le rappel 75,27 %, le F1-score 78,68 %, le mAP50 82,61 % et mAP50-95 66,44 %. Ces indicateurs attestent, de la capacité du modèle à détecter efficacement les véhicules et à minimiser les erreurs ; même dans des conditions difficiles, de luminosité (pénombre), de météorologie ou de distances mesurées et cela grâce à une augmentation de données.

6.2 Travaux futurs

Nous souhaitons dans un futur proche, d'appliquer notre modèle sur terrain ; sous différentes conditions météorologiques, de jour comme de nuit, afin de vérifier sa faisabilité, sa puissance, son efficacité et sa robustesse.

Aussi, nous souhaitons, dans notre étude, intégrer, avec l'algorithme YOLOv8 ; un Radar à ondes millimétriques et un Lidar, pour une meilleure détection précise.

Mettre en œuvre une fusion multi-capteurs (Caméra RVB, Radar, Lidar) via des techniques de deep learning.

Mieux encore, l'algorithme YOLOv10, développé récemment, nous serait plus approprié pour notre recherche sur des mesures de distances lointaines. Ainsi, il serait souhaitable, dans notre pays, d'exploiter ce système, même dans d'autres environnements, comme celui des réseaux autoroutiers.

Nous envisageons également de renforcer la résilience de notre système, face aux attaques adversaires, en intégrant des mécanismes de défenses capables de détecter et de neutraliser les perturbations visuelles.

Bibliographie

[1] K. Benelkadi, «« Pour 2023, on compte 85 heurts, sept morts et 39 blessés : La SNTF sensibilise sur les risques ferroviaires », El Watan, 29 avril 2024.,» [En ligne]. Available: https://elwatan-dz.com/pour-2023-on-compte-85-heurts-sept-morts-et-39-blesses-la-sntf-sensibilise-sur-les-risques-ferroviaires.

[2] T. W. Wardhana et R. Jayadi, *« Object Detection at Railway Level Crossing to Improve Public Safety », Journal of Theoretical and Applied Information Technology, vol. 102, no. 20, pp. 7567–7573, oct. 2024. [En ligne]. Disponible sur : https://www.jatit.org/volumes/Vol102No20/24Vol102N.*

[3] D. C. K. H. e. E. S. A. Amin, *« Intelligent Railroad Grade Crossing: Leveraging Semantic Segmentation and Object Detection for Enhanced Safety », arXiv preprint arXiv:2403.11060, mars 2024. [En ligne]. Disponible sur : https://arxiv.org/pdf/2403.11060..*

[4] *A. L. Amin, D. Chimba, and K. Hasan, « Integrating AI and edge computing for advanced safety at railroad grade crossings,» J. Rail Transp. Plan. Manag., vol. 33, p. 100501, 2025, doi: 10.1016/j.jrtpm.2024.100501.*

[5] *M. Sevi et İ. Aydın, « Detection of Foreign Objects Around the Railway Line with YOLOv8 », Journal of Computer Science, vol. IDAP-2023, pp. 19–23, 2023. [En ligne]. Disponible : https://dergipark.org.tr/en/download/article-file/3346537.*

[6] *S. A. Fahim, « Finetuning YOLOv9 for Vehicle Detection: Deep Learning for Intelligent Transportation Systems in Dhaka, Bangladesh », arXiv preprint arXiv:2410.08230v2, 2024. [En ligne]. Disponible :https://arxiv.org/pdf/2410.08230.*

[7] *Ultralytics, « Explorer Ultralytics YOLOv8 », Ultralytics Documentation, 2023. [En ligne]. Disponible : https://docs.ultralytics.com/fr/models/yolov8/.*

[8] *YOLOv8.org, « YOLOv8 Architecture Explained », YOLOv8.org, 2024. [En ligne]. Disponible : https://yolov8.org/yolov8-architecture-explained/.*

[9] *J. Redmon, S. Divvala, R. Girshick et A. Farhadi, « You Only Look Once: Unified, Real-Time Object Detection », arXiv, juin 2015. [En ligne]. Disponible sur : https://arxiv.org/pdf/1506.02640.*

[10] *M. A. B. Fayyaz et C. Johnson, « Object Detection at Level Crossing Using Deep Learning », Micromachines, vol. 11, no. 12, art. 1055, 2020. [En ligne]. Disponible : https://doi.org/10.3390/mi11121055,* vol. 11.

[11] *SNTF (Société Nationale des Transports Ferroviaires). Passages à niveau. [PDF]. Disponible sur : https://www.sntf.dz/documents/Passages_à_niveau.pdf.*

[12] *Sécurité routière, "Passage à niveau," Sécurité routière, [En ligne]. Disponible sur : https://www.securite-routiere.gouv.fr/chaque-situation-sa-conduite/conduire-sur-routes-*

specifiques/passage-niveau.

[13] *Secrétariat Général du Gouvernement, « Journal Officiel de la République Algérienne Démocratique et Populaire, n° 44 », 2000. [En ligne]. Disponible : https://www.joradp.dz/FTP/jo-francais/2000/F2000044.PDF.*

[14] *Vanberg Prévention, « Les dangers des passages à niveau », Formation Sécurité Routière, 2024. [En ligne]. Disponible : https://formation-securite-routiere.fr/guide-risques-routiers/sensibilisation-conducteurs/les-dangers-des-passage-sa-niveau/.*

[15] *july, "Urbex France," Pinterest, Aug. 28, 2019. https://www.pinterest.com/pin/70439181660420960/ .*

[16] SNCF Numérique, *« Sécurité ferroviaire augmentée : des détecteurs d'obstacles en expérimentation », SNCF Numérique, 19 juin 2017. [En ligne]. Disponible : https://numerique.sncf.com/actualites/securite-ferroviaire-augmentee-des-detecteurs-dobstacles-en-experimentation/.*

[17] *Outsight, « An In-depth Comparison of LiDAR, Camera and Radar Technologies », Edge AI and Vision Alliance, 21 février 2024. [En ligne]. Disponible : https://www.edge-ai-vision.com/2024/02/an-in-depth-comparison-of-lidar-camera-and-radar-technologies/.*

[18] H. Technology, «Things You Need to Know About LiDAR: What Is LiDAR?,» 21 mars 2023. [En ligne]. Available: https://www.hesaitech.com/what-you-need-to-know-about-lidar-the-strengths-and-limitations-of-camera-radar-and-lidar/.

[19] *BJUltrasonic, Détecteurs infrarouges : avantages et inconvénients. [En ligne]. Disponible sur : https://www.bjultrasonic.com/fr/pros-and-cons-of-infrared-detectors.*

[20] *Université de Béchar, Les applications, les avantages et les inconvénients de la fibre optique. [En ligne]. Disponible sur : https://fr.scribd.com/document/478581765/Les-Applications-Les-Avantages-Et-Les-Inconvenients-de-La-Fibre-Optique.*

[21] H. Technology, «Things You Need to Know About LiDAR: What Is LiDAR?,» 21 mars 2023. [En ligne]. Available: https://www.hesaitech.com/what-you-need-to-know-about-lidar-the-strengths-and-limitations-of-camera-radar-and-lidar/. [Accès le 02 mars 2025].

[22] *S. Yadav and S. Sharma, "A review of vision-based on-board obstacle detection and distance estimation in railways," Sensors, vol. 21, no. 10, p. 3452, May 2021. [En ligne]. Disponible: https://www.mdpi.com/1424-8220/21/10/3452.*

[23] *T. Mitchell, "Machine Learning," 1997. Disponible: https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.*

[24] *A. Azencott, "Introduction à l'apprentissage automatique," [En ligne]. Disponible sur: https://cazencott.info/dotclear/public/lectures/IntroML_Azencott.pdf.*

[25] *«L'apprentissage non supervisé,» Mathworks.com, 2025. https://fr.mathworks.com/discovery/unsupervised-learning.html.*

[26] *R. S. Sutton et A. G. Barto, Apprentissage par renforcement : une introduction, 2e éd.,*

Cambridge, MA, USA : MIT Press, 2018. [En ligne]. Disponible : https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[27] *Exocoeur, "Introduction aux réseaux de neuronnes ANN," ExoCo-LMD, Feb. 18, 2023. https://www.exoco-lmd.com/machine-learing/introduction-aux-reseaux-de-neuronnes-ann/.*

[28] O. B. E. M. a. S. L. O. Kharroubi, *"Application du réseau des neurones artificiels à la prévision des débits horaires: Cas du bassin versant de l'Eure, France," Hydrological Sciences Journal, vol. 61, no. 3, pp. 541–550, Jan. 2016, doi: https://doi.org/10.1080/02626667.2014.933225.*

[29] A. V. A. P. e. M. T. A. K. Singh, *« Machine Learning v/s Deep Learning », International Research Journal of Engineering and Technology (IRJET), vol. 6, no 2, p. 239–242, févr. 2019. [En ligne]. Disponible : https://www.irjet.net/archives/V6/i2/IRJET-V6I286.pdf.*

[30] *Ultralytics, «Détection d'objets» Ultralytics Glossaire, [En ligne]. Disponible sur : https://www.ultralytics.com/fr/glossary/object-detection.*

[31] *"Real-World Applications of Object Detection," GeeksforGeeks, 2024. [En ligne]. Disponible sur : https://www.geeksforgeeks.org/real-world-applications-of-object-detection.*

[32] *D. G. Lowe, « Distinctive Image Features from Scale-Invariant Keypoints », International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004. [En ligne]. Disponible : https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf.*

[33] *H. Bay, T. Tuytelaars, et L. Van Gool, « SURF: Speeded Up Robust Features », Computer Vision – ECCV 2006, Lecture Notes in Computer Science, vol. 3951, pp. 404–417, 2006. [En ligne]. Disponible : https://people.ee.ethz.ch/~surf/eccv06.pdf.*

[34] *H. Hakim et A. Fadhil, « Survey: Convolution Neural Networks in Object Detection », Journal of Physics: Conference Series, vol. 1804, no. 1, p. 012095, févr. 2021. [En ligne]. Disponible : https://iopscience.iop.org/article/10.1088/1742-6596/1804/1/012095.*

[35] A. S. U. Z. e. A. S. Q. A. Khan, *« A Survey of the Recent Architectures of Deep Convolutional Neural Networks », Artificial Intelligence Review, vol. 53, no. 8, pp. 5455–5516, avr. 2020. [En ligne]. Disponible : https://arxiv.org/pdf/1901.06032.*

[36] *A. B. Louam, «Deep Learning basé sur les méthodes de réduction pour la reconnaissance de visage», Mémoire de Master, Université Mohamed Khider de Biskra, Faculté des Sciences et Technologies, Département Télécommunication, 2019, p. 14.*

[37] *Gaël, "3 Deep Learning Architectures explained in Human Language - Datakeen," Datakeen, Feb. 25, 2018. https://www.datakeen.co/en/3-deep-learning-architectures-explained-in-human-language/.*

[38] *M. Arham, « Diving into the Pool: Unraveling the Magic of CNN Pooling Layers », KDnuggets, 28 septembre 2023. [En ligne]. Disponible sur :*

https://www.kdnuggets.com/diving-into-the-pool-unraveling-the-magic-of-cnn-pooling-layers.

[39] *H. E. Trad, «La détection d'objet avec OpenCV et deep learning», Mémoire de Master, Université Mohamed Khider de Biskra, Faculté des Sciences et Technologies, Département Electronique Réseaux Télécommunication, 30 septembre 2020, p. 14..*

[40] *Rutger Ruizendaal, "Deep Learning #2: Convolutional Neural Networks - TDS Archive - Medium," Medium, May 04, 2017. https://medium.com/data-science/deep-learning-2-f81ebe632d5c .*

[41] *W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," Neural Computation, vol. 29, no. 9, pp. 2352–2449, 2017. doi: 10.1162/neco_a_00990.*

[42] J. D. T. D. e. J. M. R. Girshick, *« Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation », in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 juin 2014, pp. 580–587.*

[43] *R. Girshick, « Fast R-CNN », arXiv preprint arXiv:1504.08083, sept. 2015. [En ligne]. Disponible sur : https://arxiv.org/pdf/1504.08083. [Consulté le : 29 mars 2025]..*

[44] *S. Ren, K. He, R. Girshick et J. Sun, « Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks », *arXiv preprint arXiv:1506.01497*, juin 2015. [En ligne]. Disponible sur : https://arxiv.org/pdf/1506.01497..*

[45] Q. X. S. A. M. M. M. Z. a. A. V. M. Bahaghighat, *"Estimation of Wind Turbine Angular Velocity Remotely Found on Video Mining and Convolutional Neural Network," Applied Sciences, vol. 10, no. 10, p. 3544, May 2020, doi: https://doi.org/10.3390/app10103544.*

[46] *J. Terven et D. Cordova-Esparza, « A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS », arXiv, 2024. [En ligne]. Disponible : https://arxiv.org/pdf/2304.00501v7.*

[47] *J. Redmon et A. Farhadi, « YOLOv3: An Incremental Improvement », avril 2018. [En ligne]. Disponible sur : https://arxiv.org/pdf/1804.02767.*

[48] *A. Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao, « YOLOv4: Optimal Speed and Accuracy of Object Detection », avril 2020. [En ligne]. Disponible sur : https://arxiv.org/pdf/2004.10934v1. [Consulté le : 02 avril 2025]..*

[49] A. S. H. F. M. Z. A. A. B. A. H. B. E. a. A. A. N. I. M. Yusof, *"Assessing the performance of YOLOv5, YOLOv6, and YOLOv7 in road defect detection and classification: a comparative study," Bulletin of Electrical Engineering and Informatics, vol. 13, no. 1, pp. 350–360, Feb. 2024, doi: 10.11591/eei.v13i1.6317.*

[50] *D. Reis, J. Kupec, J. Hong, et A. Daoudi, « Détection en temps réel d'objets volants avec YOLOv8 », arXiv preprint arXiv:2305.09972, mai 2023. [En ligne]. Disponible : https://arxiv.org/pdf/2305.09972.*

[51] *C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," arXiv preprint arXiv:2402.13616, Feb. 2024. [Online]. Available: https://arxiv.org/pdf/2402.13616.*

[52] *A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "YOLOv10: Real-Time End-to-End Object Detection," arXiv, May 2024. [En ligne]. Disponible : https://arxiv.org/abs/2405.14458.*

[53] *"YOLO model for real-time object detection: A full guide | Viam," Viam.com, 2024. https://www.viam.com/post/guide-yolo-model-real-time-object-detection-with-examples.*

[54] *M. Yaseen, "What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," arXiv preprint arXiv:2408.15857, 2024. [En ligne]. Disponible : https://arxiv.org/abs/2408.15857.*

[55] *Y. Zhang, « Chronologie des variantes de You Only Look Once (YOLO) », ResearchGate, mars 2023. [En ligne]. Disponible sur : https://www.researchgate.net/figure/Timeline-of-You-Only-Look-Once-YOLO-variants_fig1_369379818.*

[56] *K. He, X. Zhang, S. Ren et J. Sun, « Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition », arXiv preprint arXiv:1406.4729, juin 2014. [En ligne]. Disponible sur : https://arxiv.org/pdf/1406.4729.*

[57] *S. Liu, L. Qi, H. Qin, J. Shi et J. Jia, « Path Aggregation Network for Instance Segmentation », arXiv preprint arXiv:1803.01534, mars 2018. [En ligne]. Disponible : https://arxiv.org/pdf/1803.01534.*

[58] Ultralytics, *"Intersection over Union (IoU) - Learn what Intersection over Union (IoU) is, how it's calculated, and its critical role in object detection and AI model evaluation.," Ultralytics.com, 2025. https://www.ultralytics.com/fr/glossary/intersection-over-union-.*

[59] *"Explication des paramètres de l'évaluation COCO — Picsellia," Picsellia.fr, 2023. https://www.picsellia.fr/post/explication-parametres-evaluation-coco#5-Intersection-sur-Union.*

[60] *T. Keldenich, "Recall, Precision, F1 Score - Simple Metric Explanation Machine Learning," Inside Machine Learning, Sep. 02, 2021. https://inside-machinelearning.com/en/recall-precision-f1-score-simple-metric-.*

[61] *Ultralytics, "F1-Score - Discover the importance of the F1-score in machine learning! Learn how it balances precision and recall for optimal model evaluation.," Ultralytics.com, 2025. https://www.ultralytics.com/fr/glossary/f1-score.*

[62] *Ultralytics, "YOLO Performance Metrics," Ultralytics.com, 2023. https://docs.ultralytics.com/fr/guides/yolo-performance-metrics/#class-wise-metrics.*

[63] *GeeksforGeeks, "Essential Metrics for Model Assessment: TP, TN, FP, FN in Machine Learning," GeeksforGeeks, Jun. 28, 2024. https://www.geeksforgeeks.org/essential-metrics-for-model-assessment-tp-tn-fp-fn-in-machine-learning/.*

[64] *Ann, "Confusion matrix of YOLOv8 - Ann - Medium," Medium, Jul. 26, 2023. https://medium.com/@a0922/confusion-matrix-of-yolov8-97fd7ff0074e.*

[65] *"La Matrice de Confusion en Computer Vision — Picsellia," Picsellia.fr, 2023. https://www.picsellia.fr/post/matrice-confusion-computer-vision#3-Prcision--la-mesure-dvaluation-la-plus-lmentaire.*

[66] *A. E. Abbasi, A. M. Mangini, and M. P. Fanti, "Object and Pedestrian Detection on Road in Foggy Weather Conditions by Hyperparameterized YOLOv8 Model," Electronics, vol. 13, no. 18, pp. 3661–3661, Sep. 2024, doi: https://doi.org/10.3390/electronics1318366.*

[67] *M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," Int. J. Comput. Vis., vol. 88, no. 2, pp. 303–338, 2010. doi: 10.1007/s11263-009-0275-4..*

[68] *A. Kuznetsova et al., "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale," Int. J. Comput. Vis., vol. 128, pp. 1956–1981, 2020. doi: 10.1007/s11263-020-01316-z.*

[69] *J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Miami, FL, USA, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848..*

[70] Ultralytics, "COCO," Ultralytics.com, 2023. https://docs.ultralytics.com/fr/datasets/detect/coco/#sample-images-and-annotations.

[71] T.-Y. M. M. B. S. H. J. P. P. R. D. D. P. &. Z. C. L. Lin, *(2014). Microsoft COCO: Common Objects in Context. European Conference on Computer Vision (ECCV), 740–755. Springer.https://doi.org/10.1007/978-3-319-10602-1_48.*

[72] *T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick et P. Dollár, « Microsoft COCO: Common Objects in Context », arXiv preprint, arXiv:1405.0312, 2014. [En ligne]. Disponible sur : https://cocodataset.o.*

[73] *J. Jingh-ai, ultralytics-YOLO-DeepSort-ByteTrack-PyQt-GUI, GitHub repository, 2024. [En ligne]. Disponible sur : https://github.com/jingh-ai/ultralytics-YOLO-DeepSort-ByteTrack-PyQt-GUI.*

[74] *H. Li, Y. Li, L. Xiao, Y. Zhang, L. Cao, and D. Wu, "RLRD-YOLO: An Improved YOLOv8 Algorithm for Small Object Detection from an Unmanned Aerial Vehicle (UAV) Perspective," Drones, vol. 9, no. 4, p. 293, Apr. 2025, doi: https://doi.org/10.3390/drones904029.*

[75] *Road object detection: a comparative study of deep learning-based algorithms - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/left-Objects-before-detection-right-Objects-after-detection-showing-the-bounding-box_fig1.*

[76] *D. Guo, Z. Li, H. Shuai, and F. Zhou, "Algorithme de suivi de véhicules multi-cibles basé sur DeepSORT amélioré," Sensors, vol. 24, no. 21, Art. no. 7014, Oct. 2024. [En ligne]. Disponible : https://doi.org/10.3390/s24217014.*

[77] *M. Lutz, Learning Python, O. Media, Éd., Sebastopol, CA, 2013.*

[78] *DataScientest, « OpenCV : tout savoir sur le principal outil de Computer Vision », DataScientest, 1er septembre 2022. [En ligne]. Disponible : https://datascientest.com/opencv.*

[79] *Ultralytics, « Ultralytics: YOLOv5, YOLOv8, YOLO11 », GitHub, 2025. [En ligne]. Disponible : https://github.com/ultralytics/ultralytics.*

[80] *Riverbank Computing Ltd., « PyQt5 Reference Guide », Riverbank Computing, 2023. [En ligne]. Disponible : https://www.riverbankcomputing.com/static/Docs/PyQt5/.*

[81] *C. Harris et a. et, «Array programming with NumPy,» Nature, vol. 585, p. 357–362,.*

[82] *DataScientest, « SQLite : Tout savoir sur cette base de données embarquée », DataScientest, 2025. [En ligne]. Disponible : https://datascientest.com/sqlite-tout-savoir.*

[83] *G. Welch et G. Bishop,« An Introduction to the Kalman Filter »,University of North Carolina at Chapel Hill, Department of Computer Science, Technical Report TR 95-041, 1995. [En ligne]. Disponible : https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.*

[84] *Riptutorial.com, Learning Pygame, 2019. [En ligne]. Disponible : https://riptutorial.com/Download/pygame.pdf.*

[85] *Microsoft, "Visual Studio Code Documentation," 2024. [En ligne]. Disponible : https://code.visualstudio.com/.*

[86] *Google, «Google Colaboratory Documentation,» 2024. [En ligne]. Disponible : https://research.google.com/colaboratory/.*

[87] *Kaggle, «Kaggle: Your Machine Learning and Data Science Community,» [En ligne]. Disponible : https://www.kaggle.com.*