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 التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de projet fin d'étude

Présenté par :

BAHAMID Lydia

Pour l'obtention du diplôme de Master en Electronique

Spécialité : Electronique des Systèmes Embarqués

Thème :

Système de Détection des Défauts et des Ralentisseurs sur les Chaussées par Vision Artificielle

Promoteur : Mr. KABIR Yacine

Président : Mme BENBLIDIA Nadjia

Examinateur : Mr.NAMANE Abderrahmane

Année Universitaire : 2023-2024

Dédicaces

Je dédie ce travail à la personne la plus chère à mes yeux, ma perle précieuse, ma mère, pour son amour, ses encouragements, ses sacrifices, et pour avoir pris soin de moi et m'avoir aimée chaque seconde. Je t'aime, ma mère, et je prie que Dieu te protège et te garde à mes côtés.

Je tiens également à exprimer ma profonde reconnaissance envers mon père pour son soutien et son amour.

Je tiens à exprimer mon amour profond et inconditionnel à mes chères sœurs, mes bras droits : Yasmine et Sarah.

À mon grand-père et à ma grand-mère que j'aime énormément, que Dieu les garde.

À mes chères tantes et oncles, dont l'influence bienveillante et positive a grandement contribué à mon développement.

À tous les membres de ma famille.

À tous mes amies proches pour leur soutien, leur amour inconditionnel et leur amitié sincère : Sara, Yasmine, Bahia et Amina.

À mes collègues de la « Classe ESE 2024 ».

Et à toutes les autres personnes qui m'ont apporté leur soutien et que j'ai oublié de mentionner.

BAHAMID Lydia

Remerciements

Tout d’abord, je rends grâce à Dieu, le Tout-Puissant, pour m’avoir donné la force, le courage et la volonté nécessaires pour accomplir ce travail.

Je tiens à exprimer ma profonde gratitude envers mon encadrant, M. *Kabir Yacine*, pour avoir supervisé ce travail avec diligence. Travailler sous sa direction a été un réel plaisir pour moi. Je tiens à le remercier pour ses conseils avisés, sa gentillesse, sa disponibilité et son soutien qui ont été essentiels pour mener ce travail à bien et atteindre notre objectif. J'espère avoir répondu à la confiance que vous m'avez témoignée et que ce travail soit à la hauteur de vos attentes.

J’exprime mes sincères remerciements aux membres du jury, Mme *Benblidia Nadjia* et *M. Naamane Abderrahmane*, pour avoir accepté de juger ce travail.

Je souhaite également exprimer ma reconnaissance et mon admiration envers notre chef de spécialité, Mme *Naceur Djamila*, pour sa pureté d'âme et son accueil chaleureux dans le domaine de l'électronique des systèmes embarqués. Sa confiance en nous et ses efforts ont grandement contribué à notre progression vers ce niveau d'études supérieur.

Je souhaite également exprimer ma gratitude envers nos professeurs pour leur gentillesse, leur aide précieuse, leurs efforts soutenus et leurs personnalités bienveillantes qui ont marqué mon parcours. De même, je remercie M. *Ait Saadi* pour son organisation impeccable et sa présence soutenue.

Je souhaite exprimer ma reconnaissance à tous les enseignants du cycle Licence et Master, ainsi qu’à toutes les personnes qui m’ont soutenu de près ou de loin dans la réalisation de ce travail. Leur partage de connaissances a été précieux.

Je souhaite également exprimer ma profonde gratitude envers mes parents et mes sœurs, qui ont été d’un soutien constant et une source d’encouragement tout au long de mon parcours académique. Leurs encouragements, leur confiance et leur présence ont constitué une source inestimable de motivation et de réconfort pour moi. J’ai pu mener ce projet à terme grâce à leur soutien.

Résumé

Le projet a pour objectif de développer un système de détection en temps réel des défauts de la chaussée et les ralentisseurs en utilisant des techniques de *Deep Learning*, dans le but d'améliorer la sécurité routière. Ce projet se déroule en deux phases distinctes : la première phase consiste à identifier initialement les obstacles présents sur la chaussée, tandis que la deuxième phase se concentre sur la détection en temps réel des défauts spécifiques de la chaussée en utilisant le modèle YOLOv8. Les résultats obtenus au cours de ce projet sont prometteurs, montrant une performance satisfaisante du modèle. De plus, le modèle a été testé sur un Raspberry Pi 3 modèle B, démontrant ainsi sa capacité à être déployé sur des dispositifs embarqués pour une utilisation pratique et efficace sur le terrain. Ces tests confirment le potentiel du système pour améliorer la sécurité de la conduite en détectant rapidement et précisément les défauts de la chaussée.

Les mots clés : Intelligence Artificielle, Deep Learning, défaut de la chaussée, les ralentisseurs, détection d'obstacle, Yolov8, temps réel, Raspberry Pi, ADAS.

Abstract

This project aims to develop a system for real-time detection of roadway defects and speedbumps using Deep Learning methods, with the aim of improving road safety. The project is being carried out in two distinct phases: the first phase involves the initial identification of obstacles present on the roadway, while the second phase focuses on the real-time detection of specific roadway defects using the YOLOv8 model. The results obtained during this project are promising, showing a sufficient level of performance from the model. In addition, the model was successfully tested on a Raspberry Pi 3 model B, demonstrating its ability to be deployed on embedded devices for practical and effective use in the field. These tests confirm the system's potential to improve driving safety by quickly and accurately detecting road defects.

Key words: Artificial intelligence, deep learning, road defects, speed bumps, obstacle detection, Yolov8, real-time, Raspberry Pi, ADAS.

الملخص

يهدف المشروع إلى تطوير نظام للكشف عن عيوب الطرق والممهلات في الوقت الحقيقي باستخدام تقنيات التعلم العميق، بهدف تحسين السلامة على الطرق. ويجري تنفيذ المشروع على مرحلتين متميزتين: تتضمن المرحلة الأولى التحديد الأولي للعوائق والممهلات الموجودة على الطريق، بينما تركز المرحلة الثانية على الكشف في الوقت الفعلي عن عيوب محددة على الطريق باستخدام نموذج YOLOv8. كانت النتائج التي تم الحصول عليها خلال هذا المشروع واعدة، حيث أظهرت أداءً مُرضياً للنموذج. وبالإضافة إلى ذلك، تم اختبار النموذج على جهاز Raspberry Pi 3 من الطراز B، مما يدل على إمكانية نشره على الأجهزة المدمجة للاستخدام العملي والفعال في الميدان. تؤكد هذه الاختبارات قدرة النظام على تحسين سلامة القيادة من خلال الكشف السريع والدقيق عن عيوب الطريق.

الكلمات المفتاحية: الذكاء الاصطناعي، التعلم العميق، عيوب الطريق، ممهلات، اكتشاف العوائق، Yolov8، الوقت .

Liste des Abréviations

ABS	Anti blocking system
ADAS	Advanced Driver Assistance Systems
AFU	assistance au freinage d'urgence
ANN	artificial neural network :réseaux de neurones artificiels
AP	Average precision
BAS	Brake assist system
B-BOX	Bounding box
CLI	Commande line interface
CNN	Convolutional neural network
conv	convolution
CPU	Central processing unit
CSP	Cross Stage partial
CV	Computer vision
DAW	Driver drownsiness and loss of attention warning
DL	Deep Learning
DTS	Dynamic steering torque
ESP	Electronic stability program
FCW	Forward-collision warning system
FPS	Frame per second
GPU	Graphical processing unit
HDMI	High-Definition Multimedia Interface
IA	l'intelligence artificielle
IDE	Integrated developpment environnement
IIHS	Insurance Institute for Highway Safety
IOT	Internet of things
IOU	Intersection over union
KNN	k nearest neighbors
LCA	Lane change assistant

LiDAR	Light Detection And Ranging
LSTM	réseaux de mémoire à long terme
MAE	Mean Absolute Error
MAP	Mean average precision
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLP	Traitement du langage naturel
NMS	Non maximum suppression
NVS	Nocturne vision system
PANet	Path Aggregation Network
PIP	Preferred installer program
RAM	Random access memory
R-CNN	Regional- Convolutional Neural Network
ReLU	*Rectified Linear Activation*
RNN	réseaux de neurones récurrents
ROI	Region of interest
RPN	Region proposition network
SAE	Society of Automotive Engineers
SDRAM	Dynamic random access memory
SoC	System-on-Chip
SPPF	Spacial pyramida pooling fast
SSD	Single shot multibox detection
SSH	Protocole Secure Shell
SVM	les machines à vecteurs de support
TanH	*Tangente hyperbolique*
VGG	Visual geometry group
VNC	(Virtual Network Computing).
YOLO	You Only Look Once

Table des matières

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

Aujourd'hui, de nombreux véhicules sont dotés de systèmes avancés d'aide à la conduite disponibles sur le marché. Ces systèmes offrent des fonctionnalités remarquables, telles que des alertes, l'ajustement de la vitesse, le maintien de la trajectoire et des interventions automatiques pour éviter les accidents, selon les situations rencontrées sur la route. Ces technologies améliorent la sécurité et le confort de conduite pour les utilisateurs. Cependant, ces systèmes ne sont pas encore pleinement développés, car chaque pays a ses propres normes et obstacles. En Algérie, les défauts de la chaussée posent un problème majeur, ce qui nécessite une adaptation des systèmes actuels pour gérer ces conditions spécifiques.

L'objectif de ce projet est d'intégrer le **Deep Learning** pour améliorer les systèmes **ADAS**, afin d'accroître la sécurité de la conduite en réduisant les accidents, les blessures et les dommages aux véhicules. Il s'agit d'optimiser ces systèmes pour minimiser les risques et les coûts liés aux accidents de la route.

Nous avons constaté une croissance rapide et fascinante de l'intelligence artificielle dans divers domaines, notamment la vision par ordinateur. Parmi les avancées significatives de cette discipline figure la technologie de détection d'objets. Cette technologie peut être utilisée pour identifier et localiser les obstacles sur une chaussée, en leur attribuant des étiquettes spécifiques pour une gestion efficace et sécurisée de la circulation routière.

Dans ce projet, nous avons pour ambition d'appliquer l'apprentissage profond aux défauts de la chaussée, en nous concentrant particulièrement sur le modèle **YOLOv8**, largement utilisé dans les applications en temps réel. Nous visons à améliorer les performances de détection et d'identification des obstacles sur la chaussée grâce à cette approche avancée.

Ce mémoire est structuré en quatre chapitres :

Chapitre 1 : Dans ce chapitre nous allons introduire les défauts de la chaussée routière. Nous y aborderons les concepts et problèmes liés à ce domaine, ainsi que les systèmes ADAS et les recherches antérieures dans ce domaine.

Chapitre 2 : Ce chapitre se concentrera sur l'exploration de *l'intelligence artificielle* et de ses applications, avec un focus sur le l'apprentissage automatique et l'apprentissage profond, et leurs utilisations dans les systèmes avancés d'aide à la conduite (*ADAS*).

Chapitre 3 : dans ce chapitre, nous aborderons l'étude de la vision par ordinateur, en particulier la détection d'objets, avec une présentation détaillée du modèle YOLOv8 utilisé dans ce domaine.

Chapitre 4 : Dans ce chapitre, nous mettrons l'accent sur la conception et la mise en œuvre de notre système proposé. Nous décrirons les différentes étapes de développement du système et discuterons les résultats obtenus.

CHAPITRE 1 : DEFAUT ET OBSTACLES DES CHAUSSES ROUTIERES

1.1 Introduction

Au quotidien, nos trajets dépendent largement des routes, véritables piliers de notre infrastructure. Cependant, ces voies de circulation sont souvent marquées par des imperfections comme les nids de poule et les dos d'âne. Les nids de poule, véritables pièges dans la chaussée, endommagent nos véhicules, tandis que les dos d'âne, bien que destinés à modérer la vitesse, peuvent se révéler gênants.

Dans ce chapitre, nous analyserons les ralentisseurs et les nids de poule, en mettant en lumière les problèmes provoqués par ces obstacles. Nous plongerons également dans les technologies existantes faite pour la détection de ces irrégularités routières.

1.2 Définitions

1.2.1 Nid de poule

Un nid-de-poule, connu en anglais sous le terme de « pothole », est une déformation ou un trou dans la surface de la route, généralement causé par l'usure, les conditions météorologiques, ou des défauts dans la construction ou l'entretien de la chaussée. Ces irrégularités peuvent varier en taille, en profondeur et en forme, allant de petites imperfections à un trou profond et large dans la route (comme le montre la Figure 1.1). Ils sont souvent caractérisés par leur forme irrégulière et leur bordure abrupte. Ces déformations apparaissent généralement lorsque l'eau pénètre sous la surface de la route, provoquant

l'affaiblissement et la rupture de la structure sous-jacente. Les causes incluent le gel et le dégel répétés, la pluie abondante, et la surcharge des véhicules.

Ces potholes contribuent fortement à l'endommagement des pneus et les suspensions des véhicules, augmentent les risques d'accidents et génèrent des coûts de réparation élevés pour les automobilistes. De plus, ils représentent un danger pour la sécurité routière, obligeant les conducteurs à effectuer des manœuvres brusques pour les éviter. [1]

Figure1.1 : Route endommagée avec nids-de-poule [1]

1.2.2 Ralentisseur

1.2.2.1 Définition

D'une manière générale, le terme ralentisseur est connu sous le nom de dos-d'âne, est une structure physique généralement construite sur une route ou une chaussée sous la forme d'une bosse perpendiculaire à la rue, qui sert à contraindre les conducteurs à ralentir leur véhicule et à contrôler la vitesse excessive des véhicules et de protéger les piétons(voir Figure 1.2). Certains peuvent être temporaires, comme ceux utilisés dans les zones de travaux routiers, tandis que d'autres sont permanents et font partie intégrante de la conception routière. Toutefois, ils peuvent entraîner des conséquences telles que l'inconfort pour les conducteurs et une usure accrue des suspensions des véhicules. De plus, une implantation mal conçue peut causer des freinages brusques, augmentant le risque d'accidents. [2]

Figure 1.2 : Ralentisseur

1.2.2.2 Types de ralentisseurs

Chaque type de ralentisseurs répond à des besoins spécifiques, réduisant les accidents et assurant la sécurité des piétons et des véhicules. [3]

a- Dos d'âne

C'est le ralentisseur le plus ancien et répandu, souvent appelé « gendarme couché ». Il se présente comme une bosse perpendiculaire à la route, obligeant les conducteurs à ralentir. Ce type est installé sur des routes accueillant entre 2000 et 3000 véhicules par jour. Ses caractéristiques incluent une hauteur de 100 mm (avec une tolérance de construction +10 mm) et une longueur de 4 m (avec une tolérance de construction +0,20 m) (voir Figure 1.3). [4]

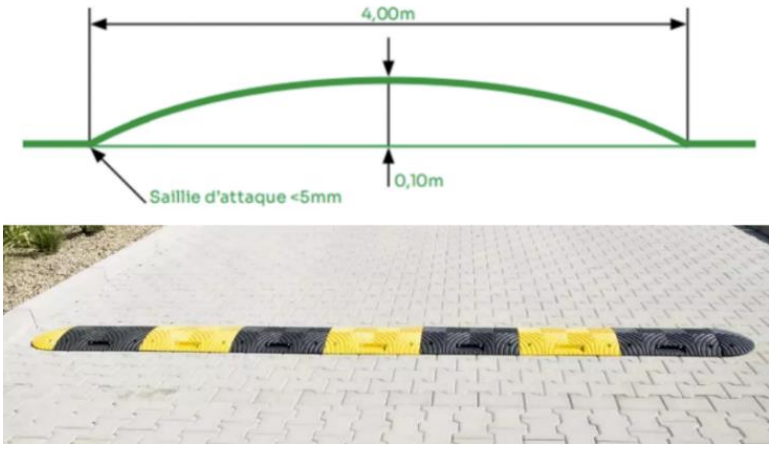

Figure 1.3 : Ralentisseur de type dos d'âne avec dimensions et spécifications [9]

b- Ralentisseur de type trapézoïdal

Les ralentisseurs de type trapézoïdal se distinguent des dos-d'âne traditionnels par leur configuration. Ils comportent un plateau surélevé et deux parties en pente, avec un

passage piéton obligatoire intégré. Leur inclinaison permet un ralentissement plus doux des véhicules, réduisant ainsi les secousses pour les passagers. Leurs caractéristiques comprennent une hauteur de 100 mm (+10 mm) et une longueur de plateau variant entre 2,50 m et 4 m (voir Figure 1.4). [4]

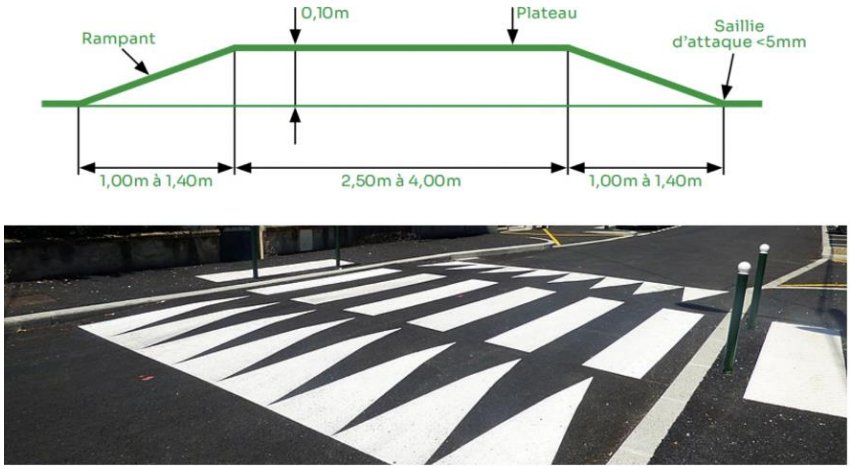

Figure 1.4: Ralentisseur de type trapézoïdal avec dimensions et spécifications [9]

c- Bande sonore ou rugueuse

Comme leur nom l'indique, ces ralentisseurs sont dotés de bandes rugueuses ou de marques spéciales sur la chaussée, qui ressemblent à des dos d'âne plus fins et moins hauts (voir Figure 1.5). Ils émettent un signal sonore lorsque les véhicules les franchissent et sont installés en groupes de trois à six pour alerter les conducteurs. [4]

Figure 1.5: Ralentisseur de type bande sonore

d- Les coussins

Ces coussins, de forme carrée et larges, sont placés par groupes de 2 ou 3 à travers toute la route, offrant un freinage progressif aux véhicules tout en minimisant les vibrations

pour les passagers. Leur disposition et leur taille permettent aux véhicules plus larges de les traverser sans réduire leur vitesse, assurant ainsi que les véhicules d'urgence tels que les ambulances et les camions de pompiers ne sont pas entravés dans leur progression (comme le montre la Figure 2.6). De plus, leur conception modulaire permet une flexibilité dans leur installation et leur retrait, ce qui les rend adaptés à différents environnements et situations de circulation. [4]

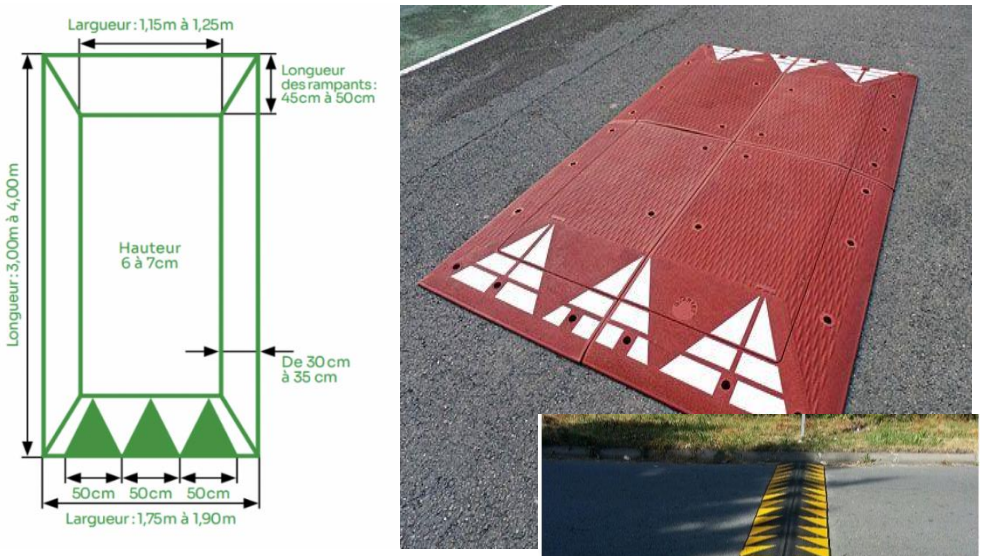

Figure 1.6: Ralentisseur de type coussins avec dimensions et spécifications [9] [11]

e- Ralentisseurs jaunes et noirs pour voies privées

Figure 1.7: Ralentisseur jaune et noir [12]

Fabriqué en caoutchouc ou en ciment, ces ralentisseurs régulent la circulation sur les voies privées tels que les hôpitaux, les centres commerciaux, les entrepôts, les stations essences (voir Figure 3.7). [4]

f- Les plateaux

Un plateau ralentisseur est une élévation de la chaussée qui s'étend sur toute sa largeur jusqu'aux bordures de trottoirs, généralement en béton ou en asphalte, offrant un freinage plus graduel en forçant les conducteurs à monter et descendre (voir Figure 1.8). Toutefois, leur usage est déconseillé sur les voies à forte circulation de transport en commun et sur les routes empruntées par les poids lourds pour limiter le bruit. [4]

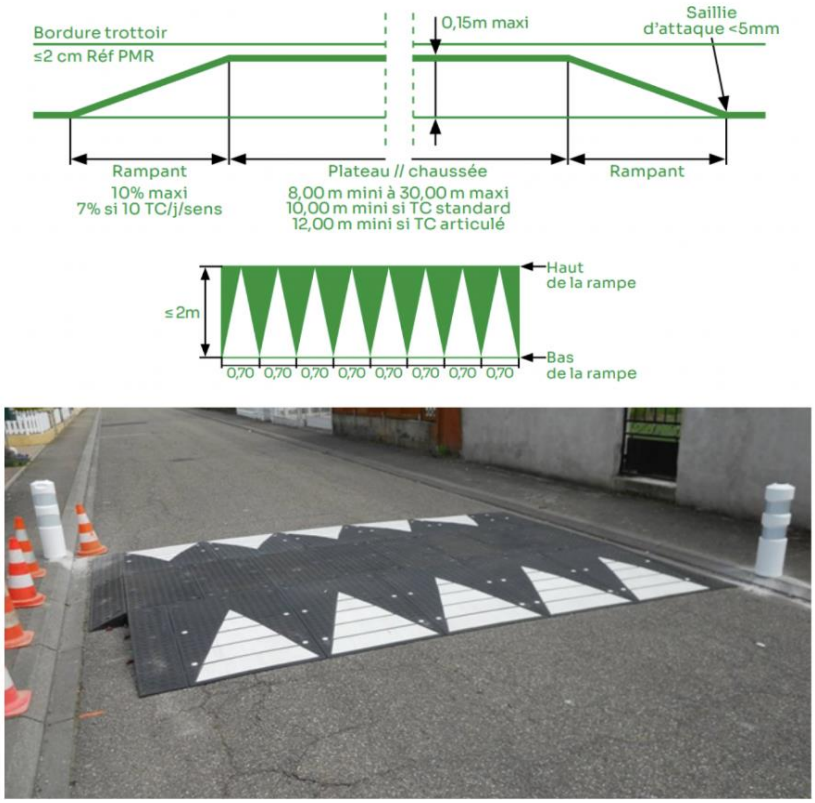

Figure 1.8 : Ralentisseur de type plateau avec dimensions et spécifications [9] [13]

g- Créneaux, chicanes et autres ralentisseurs

Le créneau implique de déformer intentionnellement la chaussée sur une courte distance, environ 50 mètres, avec deux virages serrés pour inciter les conducteurs à adapter leur conduite.

La chicane est une variante courante du créneau, présente des trottoirs alternés à droite et à gauche, séparés d'environ 40 mètres. L'écluse, quant à elle, réduit la largeur de la chaussée à environ 3,50 mètres, permettant le passage d'un seul véhicule à la fois, avec un espace dédié aux cyclistes (voir Figure 1.9). [4]

Figure 1.9 : Ralentisseur de créneaux

h- Ralentisseur circulaire

Le ralentisseur circulaire se caractérise par sa forme ronde (comme le montre la figure 1.10). Disponibles en différentes tailles et couleurs, principalement en noir, jaune et blanc, ces ralentisseurs ont une hauteur variant généralement entre 5 et 7 pouces. [5]

Figure 1.10 : Ralentisseur de type circulaire [5]

1.3 Méthodes traditionnelles de détection

Les méthodes traditionnelles de détection des irrégularités routières ont longtemps reposé sur des pratiques d'inspection visuelle et l'utilisation d'instruments de mesure conventionnels.

L'inspection visuelle, effectuée par des ingénieurs ou des techniciens sur le terrain, consiste à repérer visuellement les défauts de la chaussée tels que les dos d'âne, les fissures et les affaissements. Cette approche, bien que largement utilisée, présente des limites en termes de subjectivité et de fiabilité, en particulier dans des conditions météorologiques défavorables ou sur des routes à fort trafic. En complément, des instruments de mesure traditionnels tels que les niveaux et les inclinomètres ont été utilisés pour évaluer précisément les déformations de la chaussée. Cependant, ces méthodes sont souvent chronophages et nécessitent une expertise technique approfondie et une précision pour améliorer la fiabilité des résultats. [6]

1.4 Problématique

L'efficacité de la gestion des dommages causés par les nids-de-poule et les ralentisseurs sur les routes est au cœur des préoccupations des conducteurs et des autorités en charge des infrastructures routières. Les nids-de-poule, imprévisibles et omniprésents, représentent un danger pour les véhicules et la sécurité routière, tandis que les ralentisseurs, bien qu'utiles pour réduire la vitesse, peuvent également endommager les véhicules et incommoder les conducteurs.

La problématique inhérente à cette étude réside dans l'élaboration d'une solution fiable de détection des potholes, des ralentisseurs et autres anomalies de la chaussée. L'objectif étant d'atténuer les impacts sur les véhicules tout en préservant l'intégrité de la sécurité routière grâce à cette détection.

1.5 Systèmes avancés d'aide à la conduite

1.5.1 Définition

Advanced Driver Assistance Systems – ou ADAS regroupent des technologies de pointe développées tels que les ultrasons, caméras, radars et LiDAR pour analyser l'environnement du véhicule en temps réel.

Grâce à des algorithmes sophistiqués, ils peuvent fournir des informations au conducteur, l'alerter en cas de danger et même prendre des actions correctives pour éviter un accident, allant jusqu'à la prise de contrôle totale du véhicule dans certaines situations.

Les ADAS représentent une avancée majeure dans l'optimisation de l'expérience de conduite. Leur objectif principal est de renforcer la sécurité et d'améliorer le confort des conducteurs en adressant divers défis rencontrés sur la route. [16]

Ces systèmes offrent une multitude de fonctionnalités, notamment :

- Facilité de stationnement
- Repérage des piétons
- Signalement de franchissement de ligne
- Repérage des angles morts
- Système de freinage automatique d'urgence

En combinant ces fonctionnalités, les ADAS contribuent à rendre la conduite plus sûre et plus agréable pour tous les usagers de la route. Les systèmes ADAS sont désormais utilisés dans une variété de véhicules, y compris les voitures de tourisme, taxis et véhicules de covoiturage, véhicule de transport, ainsi que dans les véhicules agricoles, de construction et militaires. L'IIHS (Insurance Institute for Highway Safety) estime que les technologies ADAS disponibles actuellement pourraient prévenir ou atténuer les conséquences de 1,8 million d'accidents par an, sauvant ainsi jusqu'à 10 000 vies chaque année dans le monde. [15]

1.5.2 Les différentes technologies ADAS

On retrouve des grandes familles d'ADAS, ceux destinés à la sécurité des usagers de la route et ceux destinés au confort de route [15] :

A) Les systèmes ADAS destinés à la sécurité/protection des usagers de la route

- **L'ABS** (système anti blocage des roues) : empêche le blocage des roues lors des freinages brusques.
- **L'AFU** (assistance au freinage d'urgence) : aide le conducteur à freiner en urgence avec une puissance maximale. Certains constructeurs ajoutent l'activation automatique des feux de détresse pour avertir les autres usagers.
- **L'ESP** (programme électronique de stabilité) : corrige les écarts de trajectoire en agissant sur le système de freinage, surtout dans les virages.
- **FCW** (alerte d'anticollision frontale) : utilise un capteur/radar frontal pour surveiller la route et réduire le risque de collision entre véhicules. Il avertit le conducteur par des signaux visuels et sonores en cas de fort risque de collision.
- **LCA** (assistant de changement de voie) : aide le conducteur à changer de voie en toute sécurité en détectant les angles morts grâce à un radar. Il envoie un signal lumineux si un véhicule se rapproche trop, encourageant ainsi le conducteur à interrompre le changement de voie.
- **ISA** (assistant de vitesse intelligent) : diminue la puissance du moteur en cas de dépassement de la limite de vitesse autorisée. Utilise une caméra pour repérer les panneaux de limitation de vitesse à distance.
- **DAW** (avertisseur de somnolence et perte d'attention du conducteur) : utilise un dispositif de détection de signes de fatigue basé sur la capture vidéo du visage du

conducteur. Il alerte également en cas de mouvements brusques répétitifs pour encourager le conducteur à faire une pause.

- **DTS** (**dynamic steering torque**) : maintient la stabilité du véhicule sur les routes à faible adhérence en facilitant le maniement du volant pour un meilleur contrôle.
- **NVS** (système de vision nocturne) : améliore la visibilité lors de la conduite de nuit en offrant une vision étendue au-delà de la portée des phares. Les capteurs infrarouges permettent également de projeter la route à suivre sur l'écran de navigation.

B) ADAS axés sur le confort et la souplesse de conduite

- **Start & Stop :** économise le carburant en arrêtant le moteur lorsque le véhicule est à l'arrêt, par exemple, aux feux rouges ou au point mort. Le moteur redémarre automatiquement lorsque le conducteur débraye.
- **L'aide au démarrage en côte** : assiste le conducteur lors des démarrages en côte pour éviter que la voiture ne cale ou recule, assurant ainsi une manœuvre sécurisée.
- **Caméra de recul :** facilite les manœuvres, notamment en marche arrière, en offrant une vision à 360° grâce à une caméra qui retransmet sur l'écran de navigation tout ce qui se passe autour du véhicule.
- **Système de détection des places de stationnement** : facilite la recherche de places libres en détectant les espaces disponibles et en indiquant si la place détectée est suffisamment grande pour le véhicule.

C) Les systèmes ADAS actifs et passifs

- Systèmes actifs : le véhicule peut prendre des mesures autonomes pour prévenir les accidents.
- Systèmes passifs : des informations cruciales sont transmises au conducteur pour l'aider à prendre les bonnes décisions sur la route. Les méthodes d'avertissement comprennent des sons et des lumières clignotantes, et parfois un retour d'information physique (un volant qui tremble).

1.5.3 Fonctionnement des systèmes d'assistance à la conduite (ADAS)

1.5.3.1 Acquisition de données

Des capteurs variés, placés stratégiquement autour du véhicule, collectent en continu des informations sur l'environnement. Ces capteurs incluent :

- Caméras : Offrent une vision détaillée de l'environnement routier, permettant de détecter précisément les panneaux, marquages, véhicules, piétons et cyclistes.
- Radars à ondes millimétriques : Mesurent la distance et la vitesse relative des objets, assurant une détection efficace des véhicules même en conditions de faible visibilité.
- LiDAR (Light Detection And Ranging) : Produisent des cartes 3D précises de l'environnement, facilitant la navigation et les manœuvres complexes.
- Capteurs à ultrasons : Complètent les autres capteurs pour détecter les obstacles à courte portée, notamment lors des manœuvres de stationnement.

1.5.3.2 Traitement et analyse de l'information

Les données brutes captées par les capteurs sont envoyées à une unité centrale de traitement embarquée. Cette unité, utilisant des algorithmes avancés d'intelligence artificielle et de vision par ordinateur, réalise plusieurs tâches essentielles :

- **Fusion des données** : Combine les informations de tous les capteurs pour créer une représentation complète et cohérente de l'environnement.
- **Détection et classification des objets** : Identifie et catégorise les divers éléments environnants tels que les véhicules, piétons, cyclistes, panneaux de signalisation et marquages au sol.
- **Estimation des trajectoires et des comportements des objets** : Prédit les mouvements probables des objets détectés pour prévenir les situations à risque.
- **Analyse de la situation** : Évalue en temps réel le contexte de conduite, en tenant compte des conditions environnementales, de la vitesse du véhicule et des actions du conducteur.

1.5.3.3 Action ou information

En fonction de la situation analysée et du niveau d'autonomie de l'ADAS, le système peut adopter différentes actions, qu'elles soient actives ou passives.

1.5.3.4 Classification SAE des niveaux d'autonomie

La Society of Automotive Engineers (SAE) définit six niveaux d'autonomie pour les véhicules :

- **Niveau 0** : Surveillance totale par le conducteur de tous les aspects de la conduite.
- **Niveau 1** : Assistance à la conduite avec contrôle de la direction ou de l'accélération et de la décélération.
- **Niveau 2** : Automatisation partielle avec contrôle de la direction, de l'accélération et de la décélération par le système.
- **Niveau 3** : Contrôle complet de la conduite dynamique par le système, avec intervention du conducteur lorsque nécessaire.
- **Niveau 4** : Contrôle complet de la conduite dynamique, même sans intervention appropriée du conducteur.
- **Niveau 5** : Autonomie complète dans toutes les conditions routières sans besoin d'intervention humaine.

Les systèmes avancés d'aide à la conduite (ADAS) actuels se situent généralement entre les niveaux 1 et 3, offrant une assistance progressive au conducteur. [17]

1.6 Aperçu sur des travaux antérieurs

- **Advanced Driver Assistance System (ADAS) pour améliorer la sécurité et l'efficacité routières**

En 2016, Felipe Jiménez et ses collègues ont développé un système intégré d'évitement de collision utilisant la vision artificielle, un scanner laser 3D et des communications sans fil. Conçu pour les routes à chaussée unique, où les risques de sortie de route et de collisions sont élevés, ce système collecte des informations détaillées pour anticiper les dangers potentiels. Les communications sans fil permettent également aux véhicules de communiquer entre eux et avec leur environnement. Les premiers tests ont montré des résultats prometteurs, le système étant capable de proposer des comportements de conduite efficaces et de prendre le contrôle en cas de danger. Des essais supplémentaires sont en cours pour valider son efficacité et sa fiabilité(voir Figure 4.11). [18]

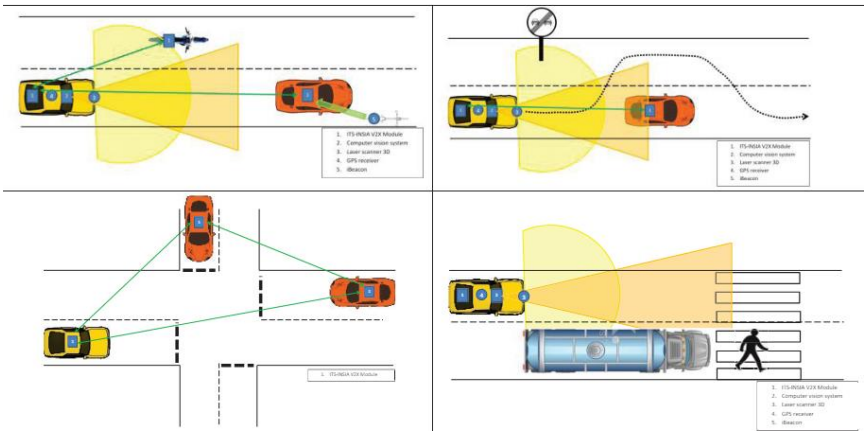

Figure 1.11 : Exemple ADAS

- **Identification des segments routiers à haut risque pour les accidents de conduite à contresens**

En 2022, Md Tanvir Ashraf, Kakan Dey et Sabyasachee Mishra ont étudié la conduite à contresens sur les autoroutes, un problème de sécurité majeur. Leur étude se distingue par trois contributions principales : [8]

1. Exploration des techniques de prétraitement des données pour traiter les ensembles de données déséquilibrés sur les accidents de conduite à contresens.
2. Utilisation de la modélisation des événements rares via des auto-encodeurs pour prédire les segments routiers à risque.
3. Analyse de l'importance et de la dépendance des caractéristiques à l'aide des valeurs SHAP.

Ils ont utilisé des données de trafic, de géométrie et de caractéristiques routières pour identifier les segments à risque. Les résultats montrent que les segments à haut risque se trouvent souvent près des zones densément développées et à fort volume de trafic, indiquant que la complexité géométrique et les comportements de conduite variés dans les zones urbaines sont des facteurs clés. Cette recherche fournit des informations précieuses aux organismes de transport pour améliorer la sécurité et l'efficacité des segments routiers à risque.

- **Reconnaissance des panneaux de signalisation routière basée sur l'apprentissage profond**

Yanzhao Zhu & Wei Qi Yan ont mené une étude en 2022 visant à développer un modèle de Reconnaissance des Panneaux de Signalisation Routière (TSR) basé sur l'apprentissage profond pour détecter rapidement et précisément les panneaux de signalisation routière en temps réel.

Utilisant le modèle YOLOv5, ils ont exploré différentes techniques de prétraitement des données pour améliorer les performances du modèle. Sur un ensemble de données comprenant 2182 images réparties en huit classes de panneaux : "Interdiction de faire demi-tour", "Dos d'âne", "Travaux routiers", etc. Le modèle YOLOv5 a démontré une précision élevée (97,70 %) pour toutes les classes, avec une vitesse de reconnaissance de 30 images par seconde (fps). Les résultats expérimentaux ont montré que le modèle YOLOv5 surpassait d'autres modèles comme SSD en termes de précision et de vitesse de reconnaissance, ce qui en fait un choix optimal pour la détection en temps réel des panneaux de signalisation routière. [19]

- **Détection et localisation des ralentisseurs par les véhicules**

En 2020, Charalambos Kyriakou, Symeon E. Christodoulou et Loukas Dimitriou ont mené une étude sur la capacité des véhicules à détecter et localiser les ralentisseurs. Leur approche multidisciplinaire a utilisé des données de divers capteurs embarqués, notamment des accéléromètres, des gyroscopes et des données GPS, intégrés dans des smartphones. Ils ont développé des modèles d'apprentissage automatique utilisant des algorithmes comme les réseaux de neurones artificiels (ANN), les k plus proches voisins (KNN) et les machines à vecteurs de support (SVM). [10]

Les modèles ont été entraînés avec des données sur les mouvements des véhicules et les coordonnées GPS en présence de ralentisseurs. Les performances des algorithmes ont été évaluées en termes de précision de détection et de localisation des ralentisseurs, les ANN ayant obtenu les meilleurs résultats, suivis de près par les KNN et les SVM. Cette approche combinant données de capteurs et techniques d'apprentissage automatique a permis de développer un système efficace pour détecter et localiser précisément les ralentisseurs sur la route.

1.7 Conclusion

La détection des ralentisseurs et des nids-de-poule sur les routes est essentielle pour la sécurité routière et la préservation des véhicules. Les systèmes avancés d'aide à la conduite (ADAS) représentent un domaine en constante évolution qui utilise des technologies de pointe pour améliorer la sécurité et le confort de conduite. Grâce à une combinaison de capteurs sophistiqués, de traitement de données performant et d'algorithmes intelligents, les ADAS

analysent l'environnement en temps réel et assistent le conducteur de manière graduelle. Des travaux ont montré qu'il est possible d'identifier rapidement et précisément des obstacles, permettant aux autorités responsables des infrastructures routières de prendre des mesures correctives appropriées. Cette approche offre une surveillance continue des routes, réduisant les risques d'accidents et prévenant les dommages matériels aux véhicules. En adoptant ces technologies de détection, nous pouvons significativement améliorer la sécurité et le confort des conducteurs sur les routes, tout en préservant l'intégrité des infrastructures routières.

Le prochain chapitre se concentrera sur l'intelligence artificielle, le machine learning et le deep learning, en mettant en lumière leurs applications dans la détection d'obstacles sur les routes.

CHAPITRE 2 : INTELLIGENCE ARTIFICIELLE ET SYSTEMES D'AIDE A LA CONDUITE

2.1 Introduction

L'intelligence artificielle a transformé de nombreux secteurs, y compris les transports, en améliorant la sécurité et l'efficacité de la conduite grâce aux systèmes avancés d'aide à la conduite. Ces technologies, comme la détection des obstacles et le stationnement automatique, ont révolutionné la conduite automobile. Dans ce chapitre, nous examinerons l'évolution de l'intelligence artificielle, du machine learning et du deep learning, ainsi que des architectures de réseaux de neurones convolutionnels. Nous analyserons en détail les applications des techniques de deep learning et de machine learning dans les systèmes d'aide à la conduite avancée (ADAS), illustrant ainsi leur rôle essentiel dans la fourniture de fonctionnalités de sécurité et de conduite sophistiquées.

2.2 L'intelligence artificielle

2.2.1 Définition

L'intelligence artificielle (IA) désigne la simulation de l'intelligence humaine dans des machines, leur permettant de penser et d'agir comme des humains. Elle se décompose en plusieurs compétences cognitives telles que la compréhension, la communication, la mémorisation, le raisonnement, l'adaptation et l'apprentissage autonome. Les algorithmes et programmes d'IA accomplissent des tâches nécessitant généralement l'intelligence humaine, comme la perception visuelle, la reconnaissance vocale, la prise de décision et la traduction de langues. Elle a le potentiel de révolutionner de nombreuses industries, des assistants personnels virtuels aux voitures autonomes. L'IA utilise divers outils tels que l'optimisation mathématique, la logique et les méthodes probabilistes, s'appuyant sur

des disciplines comme l'informatique et la psychologie. Son objectif est de comprendre et de reproduire le comportement humain, créant ainsi des ordinateurs intelligents capables de traitement du langage naturel, d'analyse faciale et de robotique. [20]

2.2.2 Origines de l'intelligence artificielle

L'histoire de l'intelligence artificielle (IA) remonte à l'Antiquité, mais le terme lui-même a été introduit en 1955 par John McCarthy. En 1956, McCarthy et ses collègues ont organisé la conférence « *Dartmouth Summer Research Project on Artificial Intelligence* », marquant ainsi le début officiel de l'IA et posant les bases de sous-domaines importants tels que le machine learning, le deep learning et les analyses prédictives. Parmi les jalons importants, on trouve le test de Turing en 1950, les débuts du machine learning et des réseaux de neurones dans les années 1950 et 1960, l'essor de l'IA symbolique dans les années 1980, et la révolution du deep learning dans les années 2010 (voir Figure 2.1). Ces événements ont établi les bases théoriques et pratiques de l'IA, conduisant à des progrès continus qui ont un impact sur des domaines variés tels que la santé, la finance, le commerce et la sécurité, rendant ainsi l'IA omniprésente et essentielle dans notre vie quotidienne. [21]

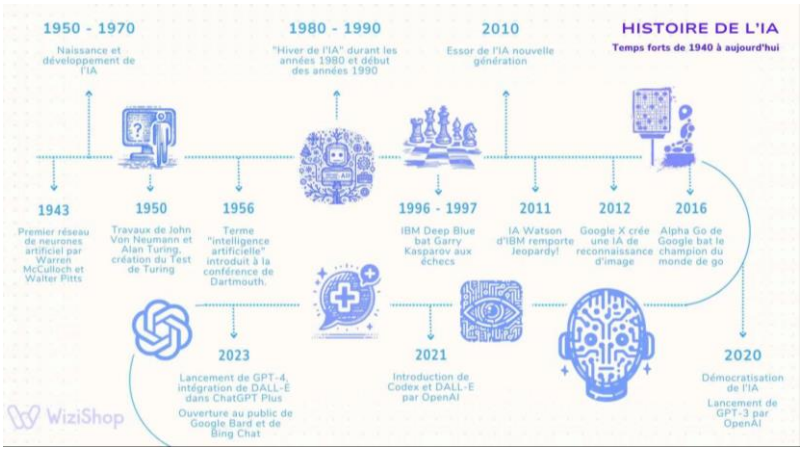

Figure 2.1: Histoire de l'IA [22]

2.2.3 Domaines de l'intelligence artificielle

L'intelligence artificielle (IA) a de nombreuses applications pratiques dans divers secteurs et domaines, notamment (comme le montre la figure 2.2) : [20]

- Santé : L'IA est utilisée pour le diagnostic médical, la découverte de médicaments et l'analyse prédictive des maladies, aidant à détecter les signes précoces de maladies comme le cancer à partir d'images médicales.
- Finance : L'IA aide à l'évaluation du crédit, la détection de la fraude et la prévision financière, permettant aux banques de repérer des activités suspectes en temps réel.
- Commerce de détail : L'IA est utilisée pour les recommandations de produits, l'optimisation des prix et la gestion de la chaîne d'approvisionnement, suggérant des produits en fonction des comportements d'achat des clients.
- Fabrication : L'IA aide au contrôle de la qualité, à la maintenance prédictive et à l'optimisation de la production, anticipant les pannes de machines pour minimiser les temps d'arrêt.
- Transport : L'IA est utilisée pour les véhicules autonomes, la prédiction du trafic et l'optimisation des itinéraires, permettant aux voitures autonomes de prendre des décisions de conduite en temps réel.
- Service client : Les chatbots alimentés par l'IA fournissent un support client 24/7, répondant aux questions fréquemment posées et traitant des demandes simples.
- Sécurité : L'IA est utilisée pour la reconnaissance faciale, la détection d'intrusions et l'analyse des menaces de cyber sécurité, identifiant des individus suspects et alertant les autorités.
- Marketing : L'IA est utilisée pour la publicité ciblée, la segmentation des clients et l'analyse des sentiments, créant des campagnes marketing personnalisées.
- Éducation : L'IA est utilisée pour l'apprentissage personnalisé, les tests adaptatifs et les systèmes de tutorat intelligent, adaptant les cours et exercices aux besoins de chaque étudiant.

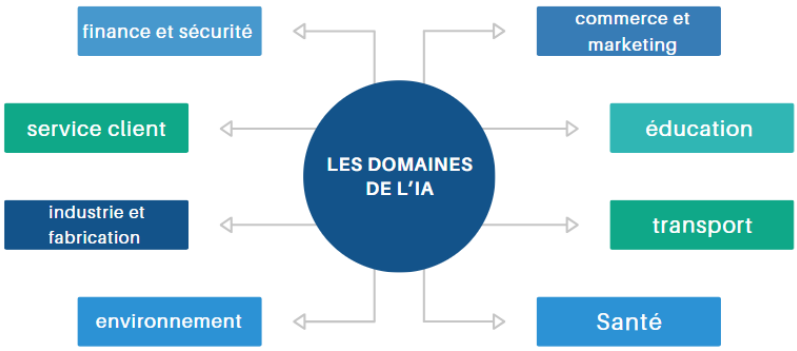

Figure 2.2 : Les Domaines d'Application de l'Intelligence Artificielle

2.2.4 Technologies basées sur l'intelligence artificielle

- Apprentissage automatique (Machine Learning) : Utilise des algorithmes pour permettre aux systèmes d'apprendre à partir de données et de faire des prédictions ou des décisions sans programmation explicite.
- Traitement du langage naturel (NLP) : Se concentre sur la capacité des ordinateurs à comprendre, interpréter et générer le langage humain.
- Vision par ordinateur (Computer Vision) : Traite du traitement et de l'analyse des informations visuelles avec des algorithmes informatiques.
- Robotique : Utilise des robots et des systèmes automatisés alimentés par l'IA pour accomplir des tâches dans la fabrication, la santé, le commerce de détail et d'autres industries.
- Réseaux de neurones (Neural Networks) : Algorithme d'apprentissage automatique modélisé d'après la structure et le fonctionnement du cerveau humain.
- Systèmes experts : Imite la capacité de prise de décision d'un expert humain dans un domaine spécifique.
- Chatbots : Assistants virtuels alimentés par l'IA qui interagissent avec les utilisateurs via des interfaces textuelles ou vocales.

Ces technologies représentent différentes facettes de l'IA et sont intégrées dans de nombreux produits et services pour améliorer leur efficacité et leur fonctionnalité (voir la figure 2.4). [23]

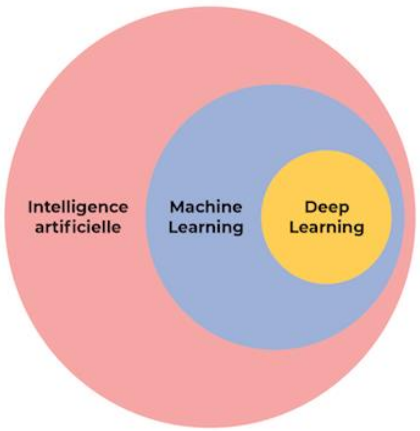

Figure 2.3 : Les Niveaux de l'Intelligence Artificielle [27]

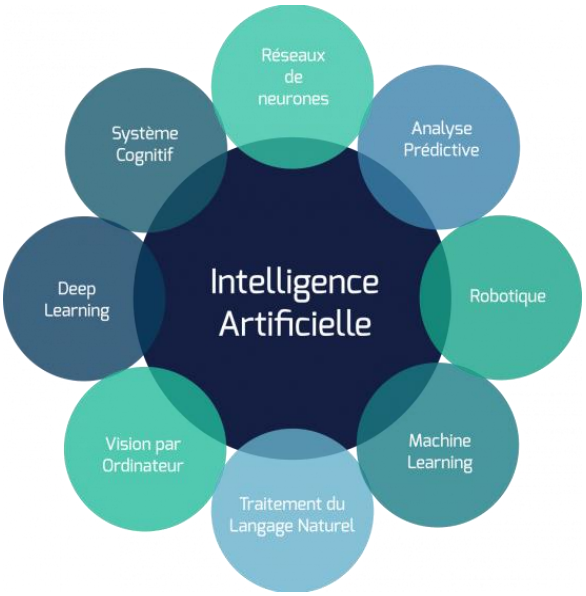

Figure 2.4 : les technologies basées sur l'Intelligence Artificielle [23]

2.3 Machine Learning

2.3.1 Définition

Le machine learning ou apprentissage automatique (ML) est une sous-catégorie de l'intelligence artificielle (IA) qui permet aux ordinateurs d'apprendre à partir de données sans programmation explicite (voir figure 2.3). En analysant de vastes ensembles de données, ces algorithmes identifient des patterns et des relations pour faire des prédictions ou prendre des décisions (voir figure 2.5) [24]. Leur efficacité dépend de la qualité des données et des ressources informatiques disponibles. Le ML est utilisé dans divers secteurs comme la santé, la finance et le commerce, et évolue rapidement grâce aux avancées technologiques. Il permet de répondre à des situations complexes, telles que la prévision des tendances ou la détection de fraudes. [25]

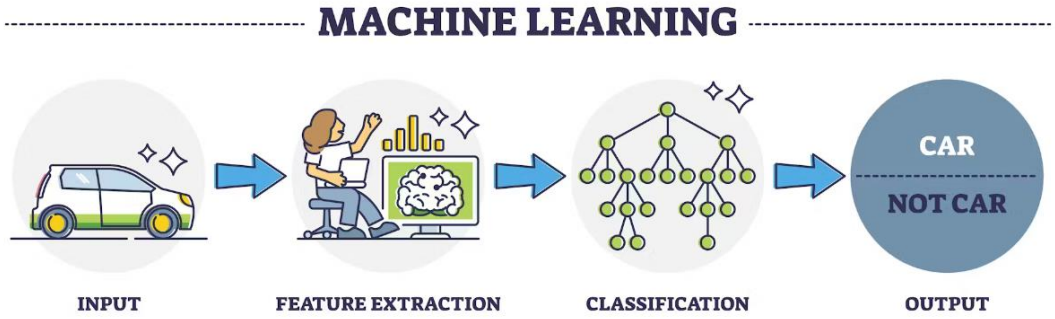

Figure 2.5 : Processus du Machine Learning [26]

2.3.2 Les catégories de l'apprentissage automatique

Le machine Learning peut être classé en trois grandes catégories : [28]

a. **Supervisé :** Le machine learning supervisé utilise des données étiquetées, fournissant à l'algorithme des entrées et des sorties souhaitées, ce qui lui permet de prédire des résultats futurs. Il englobe des tâches telles que la classification, où la sortie est une catégorie, et la régression, où la sortie est une valeur précise.

b. **Non supervisé :** Le machine learning non supervisé analyse des données non étiquetées pour découvrir des structures cachées ou des motifs. Il se concentre sur le Clustering, qui consiste à identifier des regroupements dans les données, et l'association, qui vise à trouver des règles dans de grands ensembles de données.

c. **Par renforcement :** Le machine learning par renforcement se caractérise par un programme qui apprend à atteindre des objectifs en interagissant avec un environnement dynamique. Ce programme reçoit des récompenses ou des pénalités en fonction de ses actions. Les types principaux incluent Monte Carlo, où les récompenses sont données à la fin d'un état terminal, et l'apprentissage par différence temporelle, où les récompenses sont attribuées à chaque étape du processus.

2.4 Deep Learning

2.4.1 Définition

L'apprentissage profond ou Deep Learning, abrégé DL, est une sous-catégorie du machine learning qui fonctionne de manière similaire mais avec des capacités et des approches distinctes (voir figure 2.3). Il tire son inspiration du fonctionnement des neurones du cerveau humain pour créer des réseaux neuronaux artificiels. Ces modèles utilisent plusieurs couches pour apprendre et extraire des informations à partir des données de manière automatique. Dans le DL, des applications comme les voitures autonomes, la traduction de langues et le traitement du langage naturel sont populaires. Par exemple, un modèle de deep learning peut identifier des objets comme des chats ou des chiens en analysant directement les images à travers différents niveaux de réseaux neuronaux, sans nécessiter d'extraction manuelle des caractéristiques (voir figure 2.6). Parmi les techniques couramment utilisées en Deep Learning, on trouve les réseaux de neurones convolutionnels, les réseaux neuronaux récurrents, les auto-encodeurs, ainsi que les réseaux neuronaux traditionnels. [27]

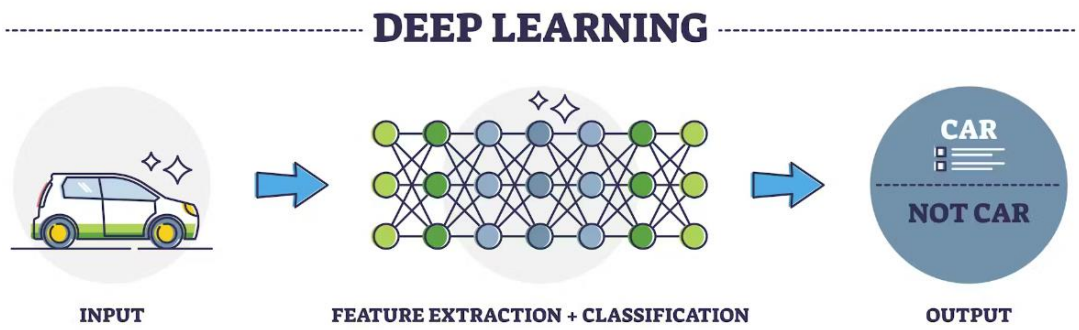

Figure 2.6 : Processus du Deep Learning [26]

2.4.2 Machine learning VS deep learning

Ci-dessous un tableau comparatif des principales différences entre le Machine Learning et le Deep Learning : [96]

Tableau 2. 1 : Comparaison ML vs DL

Caractéristique	Deep Learning (DL)	Machine Learning (ML)
Sous-ensemble de	Machine Learning (ML)	Intelligence Artificielle (IA)
Complexité	Plus complexe	Moins complexe
Architecture de base	Utilise des réseaux de neurones profonds pour apprendre à partir de données complexes	Utilise divers algorithmes pour apprendre à partir de données structurées ou non structurées
Données requises	Grandes quantités	Taille moyenne à grande
Ressources nécessaires	GPU spécialisé pour l'entraînement	CPU
Apprentissage	Automatique	Intervention humaine requise
durée de l'entraînement	Plus longue	Courte
Précision	Grande	Moins grande

2.5 Les Réseaux de neurones

2.5.1 Définition

Un réseau de neurones, également appelé réseau neuronal artificiel ou en anglais artificial neural network (ANN), est un modèle informatique inspiré du fonctionnement du cerveau humain. Il se compose de nœuds appelés neurones ou unités, disposés en couches où chacune joue un rôle crucial dans le traitement des données et dans le processus d'apprentissage. [30]

2.5.2 Les principales couches d'un réseau de neurones

Dans un réseau de neurones, il existe trois types principaux de couches : [30]

a. **Couche d'entrée (input layer)** : Cette couche reçoit les données initiales. Chaque neurone de cette couche correspond à une caractéristique des données d'entrée.
b. **Couches cachées (hidden layers)** : Situées entre la couche d'entrée et la couche de sortie, ces couches effectuent le traitement complexe des données et l'apprentissage. Chaque neurone reçoit des signaux des neurones de la couche précédente, applique des poids aux connexions et une fonction d'activation, puis transmet le signal transformé à la couche suivante. Les réseaux avec plusieurs couches cachées sont appelés "réseaux profonds" et sont utilisés dans le deep learning.
c. **Couche de sortie (output layer)** : Cette couche produit le résultat final du traitement ou de l'apprentissage du réseau. Le nombre de neurones dans cette couche dépend de la nature de la tâche à accomplir.

Chaque neurone est connecté aux neurones des couches adjacentes par des connexions pondérées. Pendant l'apprentissage, ces poids sont ajustés pour minimiser l'erreur de prédiction. Cette capacité d'apprentissage permet aux réseaux de neurones d'identifier des motifs complexes dans les données, les rendant efficaces pour des tâches telles que la reconnaissance d'images, la reconnaissance vocale et la traduction de langues.

2.5.3 Perceptron

a. **Définition :** Le perceptron, introduit par Frank Rosenblatt en 1958, est un modèle simple et fondamental de réseau de neurones. Il se compose d'une seule couche de neurones, la couche de sortie, directement connectée aux entrées. Chaque connexion est caractérisée par un poids ajusté lors de l'apprentissage (comme le montre la figure 2.7).

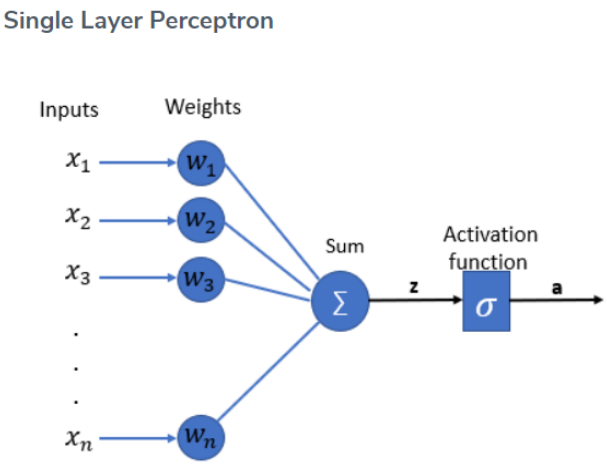

Figure 2.7 : Perceptron monocouche [29]

Le perceptron excelle dans la résolution de problèmes de classification linéaire, où les données peuvent être séparées par une ligne droite ou un hyperplan. Cependant, il présente des limitations pour traiter des problèmes plus complexes qui ne sont pas linéairement séparables. Pour surmonter ces limitations, des réseaux de neurones multicouches ont été développés, capables de modéliser des relations plus complexes au sein des données. [29]

b. Multi-layer perceptron : Contrairement au perceptron simple limité à la classification linéaire, le MLP est un type de réseau de neurones artificiels qui intègre des couches cachées qui effectuent des transformations non linéaires des données à l'aide de fonctions d'activation (voir figure 2.8). Cette capacité permet au réseau d'apprendre à partir de données complexes et de capturer des modèles plus complexes. Il s'agit donc d'une architecture fondamentale en apprentissage profond, souvent utilisée comme base pour des modèles plus avancés comme les réseaux de neurones convolutifs (CNN). [31]

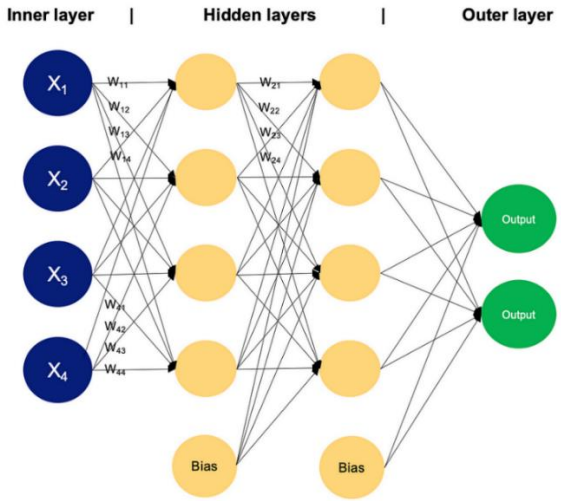

Figure 2.8 : Perceptron multicouche [31]

c. Composants de base d'un perceptron : Les caractéristiques principales du perceptron incluent (voir figure 2.9) :

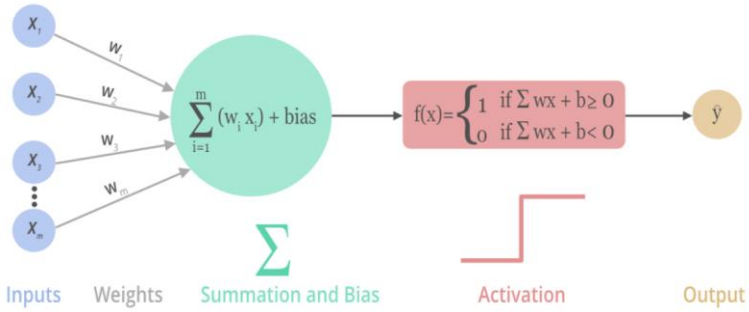

Figure 2.9 : Composants De Base D'un Perceptron [33]

- **Les données d'entrée :** également appelées features en anglais, sont les points de données initiaux utilisés par un perceptron pour le traitement de l'information. Elles constituent les données essentielles permettant au modèle d'apprendre et de faire des

prédictions. Chaque feature représente une valeur spécifique, comme l'intensité d'un pixel dans une image ou une mesure particulière dans un jeu de données tabulaire.

- **Poids :** Les poids sont des paramètres clés dans un perceptron, influençant l'importance de chaque caractéristique d'entrée (xi) pour la sortie finale. Chaque xi est multiplié par un poids (Wi), ajusté itérativement pendant l'entraînement pour optimiser la précision et minimiser l'erreur entre la prédiction et la sortie réelle. Les poids déterminent la contribution de chaque caractéristique à la sortie : un poids élevé indique une forte influence, un poids faible une influence moindre. Cruciaux en apprentissage automatique, les poids permettent au perceptron de s'adapter aux données, améliorant ainsi la précision de ses prédictions.
- **Biais :** Le biais (b) est un paramètre additionnel du perceptron, ajustant la fonction d'activation pour offrir plus de souplesse au modèle. Il agit comme une constante ajoutée à la somme pondérée des entrées, permettant au perceptron de s'adapter indépendamment des entrées et d'améliorer sa capacité à modéliser les données. En l'absence de biais, le perceptron serait restreint à des fonctions linéaires passant par l'origine, limitant sa capacité à représenter des relations complexes entre les données.
- **La somme pondérée :** Les produits des caractéristiques d'entrée et des poids sont additionnés plus le biais, comme montré sur la fonction suivante :

$$S = \sum_0^n (wi \times xi) + b \qquad (2.1)$$

- **La phase d'activation :** La phase d'activation dans un perceptron, souvent appelée seuil (Y), est essentielle car elle détermine la sortie finale. Après le calcul de la somme pondérée des entrées et l'ajout du biais, cette valeur est soumise à une fonction d'activation (le choix de la fonction appropriée peut améliorer les performances du réseau pour différentes tâches d'apprentissage automatique). Le perceptron utilise typiquement une fonction de seuil (fonction échelon de Heaviside) qui compare cette somme à un seuil spécifique.

$$Y = \begin{cases} 0 & si \quad S < 0 \\ 1 & si \quad S \geq 0 \end{cases} \qquad (2.2)$$

- **Les fonctions d'activation :** La fonction d'activation dans un réseau de neurones détermine la sortie d'un neurone en fonction de la somme pondérée de ses entrées. Elle introduit une non-linéarité essentielle, permettant au modèle de saisir des relations complexes dans les données. Il existe plusieurs fonctions d'activation comme ReLU,

TanH et sigmoïde, et le choix de la fonction appropriée peut améliorer les performances du réseau pour différentes tâches d'apprentissage automatique (voir figure 2.10). [33]

Voici des détails sur chacune des fonctions d'activation : [34]

- La fonction *Sigmoïde* comprime les valeurs dans un intervalle de 0 à 1, ce qui est utile pour représenter des probabilités. Elle est souvent utilisée dans les couches de sortie des réseaux de neurones pour la classification binaire, mais elle n'est pas limitée à ces couches ; elle est appliquée à toutes les couches du réseau.

$$S(x) = \frac{1}{1+e^{-\alpha x}} \qquad (2.3)$$

- La fonction *ReLU (Rectified Linear Activation)* attribue zéro aux valeurs négatives et conserve les valeurs positives. Elle est souvent utilisée dans les réseaux de neurones profonds pour sa simplicité, son efficacité et sa rapidité de calcul, aidant à surmonter le problème de la disparition du gradient.

$$R(x) = \max(0, x) \qquad (2.4)$$

- La fonction *Tangente hyperbolique (Tanh*) comprime les valeurs entre -1 et 1, ce qui la rend utile pour les couches cachées où les valeurs sont centrées autour de zéro. Son rôle est de régulariser les valeurs de sortie.

$$T(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}} = 2\frac{1}{1+e^{-2x}} - 1 = 2 * logistic(2x) - 1 \qquad (2.5)$$

- La fonction *Softmax* est utilisée dans la couche de sortie pour la classification multi-classe, normalise les sorties en une distribution de probabilités dont la somme est égale à 1. Elle est couramment employée pour prédire la probabilité d'appartenance à chaque classe dans les tâches de classification avec plusieurs classes.

$$M(x_i) = \frac{e^{x_j}}{\sum_{j=1}^{N} e^{x_j}} \quad pour\ i = 1,2,\dots N \qquad (2.6)$$

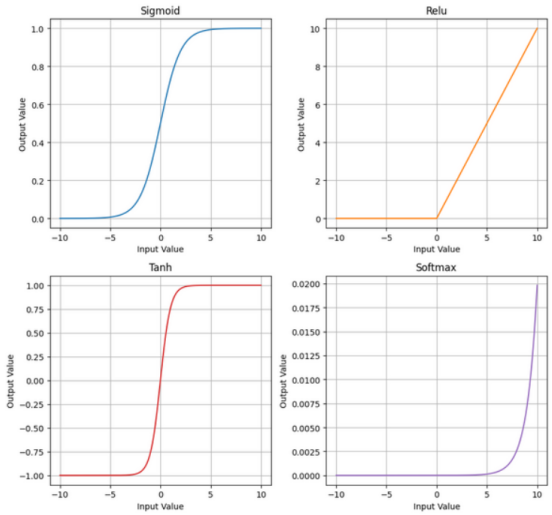

Figure 2.10 : Graphe des Fonctions d'activation [32]

- **La sortie** : La couche de sortie d'un MLP génère les prédictions finales en appliquant une fonction d'activation aux données de la dernière couche cachée. Le réseau ajuste les poids pour réduire l'erreur entre les prédictions et les cibles, ce qui lui permet d'apprendre des motifs complexes et de faire des prédictions précises sur de nouvelles données. [36]

2.5.4 Processus d'apprentissage

a. La Descente de gradient

La descente de gradient est une méthode d'optimisation en apprentissage automatique qui vise à minimiser la fonction de perte d'un modèle (comme le montre la figure 2.11). Elle ajuste les paramètres du modèle de manière itérative en suivant la direction opposée au gradient de la fonction de perte. L'objectif est de trouver le minimum de cette fonction en se déplaçant dans la direction où elle augmente le plus, ajustant ainsi les paramètres jusqu'à atteindre une convergence satisfaisante. [35]

La procédure de descente de gradient comprend deux phases principales : [41]

- **Propagation avant (Forward propagation) :** Dans cette phase, les données sont introduites dans le réseau de neurones et traversent les couches, de l'entrée à la sortie, en passant par les couches cachées. Les activations de chaque neurone sont calculées à l'aide des poids, des biais et des fonctions d'activation. Ce processus unidirectionnel génère des prédictions ou des classifications basées sur les connaissances acquises lors de l'apprentissage, permettant au réseau de réaliser des tâches comme la reconnaissance d'images ou la classification de texte (voir figure 2.12).
- **Rétro propagation (Backward propagation) :** La rétropropagation est un processus crucial pour entraîner les réseaux de neurones, consistant à calculer le gradient de la fonction de perte par rapport aux poids en se déplaçant de la sortie vers l'entrée du réseau. D'abord, lors de la propagation avant, les données sont introduites et les sorties calculées. Ensuite, la rétropropagation calcule l'erreur entre les sorties prédites et réelles et la propage en arrière, en utilisant la règle de dérivation en chaîne pour obtenir les gradients locaux à chaque couche.

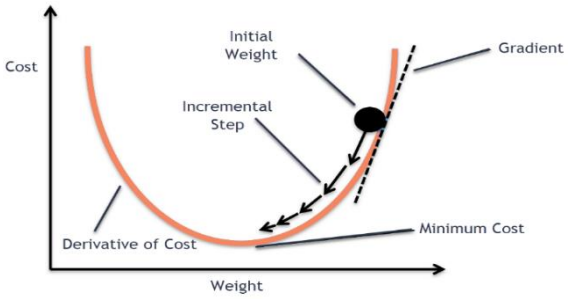

Figure 2.11 : Descente de Gradient pour Minimiser la Fonction de Coût [39]

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{[l-1]}} = \left[\boldsymbol{W}^{[l]}\right]^{\mathrm{T}} \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{[l]}} * g^{[l-1]\prime}\left(\boldsymbol{z}^{[l-1]}\right)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{[l]}} \cdot \left[\boldsymbol{a}^{[l-1]}\right]^{\mathrm{T}}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{[l]}}$$

Figure 2.12 : Équations de Rétropropagation dans un Réseau de Neurones Multicouches [96]

Ces gradients permettent de mettre à jour les poids du réseau via des méthodes d'optimisation comme la descente de gradient (voir la figure 2.12). Ce processus est répété sur plusieurs itérations (appelées époques) jusqu'à ce que les poids minimisent la fonction de perte, améliorant ainsi les prédictions du réseau (voir figure 2.13). [38]

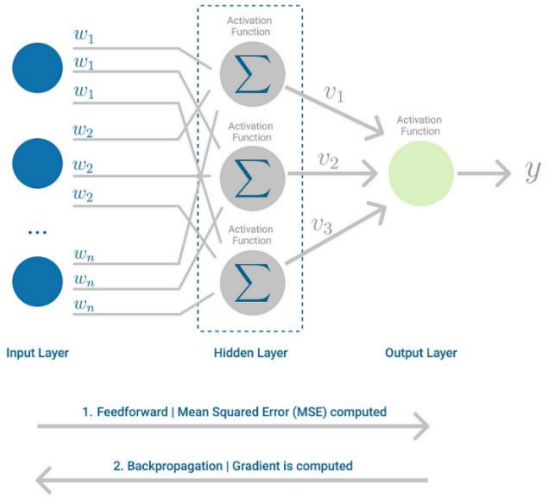

Figure 2.13: Forward and Backward Propagation [41]

b. Loss function

La fonction de perte, ou fonction de coût, est une mesure en apprentissage automatique qui évalue la performance d'un modèle en quantifiant l'écart entre ses prédictions et les valeurs réelles des données d'entraînement. Un chiffre unique représente cette erreur, et l'objectif de l'entraînement est de minimiser cette fonction de perte. Une perte plus faible indique un modèle plus précis. Essentielle pour les algorithmes d'optimisation comme la descente de gradient, la fonction de perte guide l'ajustement des paramètres du modèle pour réduire l'erreur globale. Voici des types de fonction de perte : [37]

- **Erreur quadratique moyenne (Mean Squared Error, MSE)** : Utilisée pour la régression, elle calcule la moyenne des carrés des écarts entre les valeurs réelles et prédites. Réduire le MSE améliore la précision du modèle.
- **Erreur absolue moyenne (Mean Absolute Error, MAE)** : Utilisée en régression, elle mesure la moyenne des écarts absolus entre les valeurs réelles et prédites, étant plus robuste aux valeurs aberrantes que le MSE.
- **Entropie croisée binaire (Binary Cross-Entropy)** : Utilisée pour la classification binaire, elle mesure la différence entre les distributions de probabilité des prédictions et des vraies valeurs.
- **Log loss** : Évalue la précision des probabilités prédites pour différents résultats en classification binaire, pénalisant fortement les prédictions incorrectes confiantes.
- **Hinge loss** : Utilisée pour les SVM en classification, elle pénalise les erreurs de prédiction avec un changement de signe, offrant de meilleures performances que la cross-entropy.

2.6 Convolutional neural network

2.6.1 Définition

Les réseaux de neurones convolutifs (CNN), aussi appelés *ConvNets*, sont des algorithmes de deep learning spécialisés dans l'analyse d'images. Inspirés par le fonctionnement du cortex visuel humain, les CNN utilisent une architecture hiérarchique pour extraire automatiquement des caractéristiques des images à différentes échelles. Ils comprennent des couches convolutionnelles qui appliquent des filtres pour détecter des motifs locaux, des couches de pooling qui réduisent la taille des caractéristiques extraites, et des couches entièrement connectées pour la classification finale. Les CNN sont largement utilisés dans des domaines tel que la reconnaissance d'objets, la segmentation d'images, et sont cruciaux pour des applications pratiques comme les véhicules autonomes et les systèmes de sécurité vidéo. [40]

2.6.2 Composants du CNN :

Le réseau de neurones convolutifs est composé de quatre parties principales (comme le montre la figure 2.14):

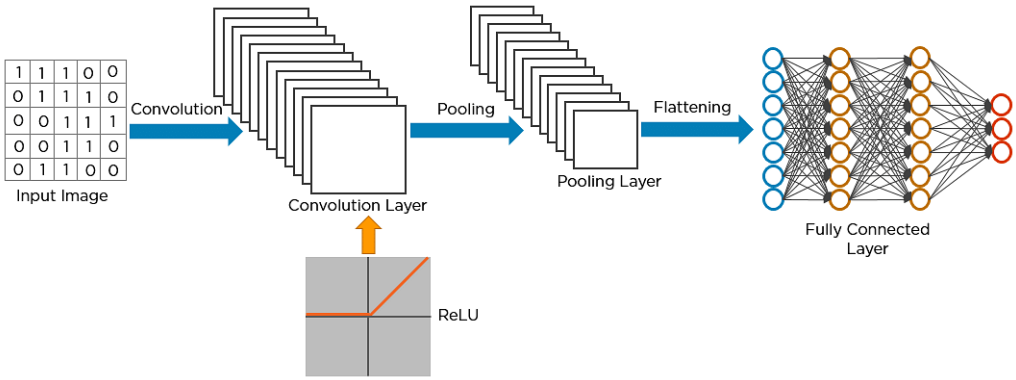

Figure 2.14 : Architecture d'un Réseau de Neurones Convolutionnel (CNN) pour la Classification d'Images [44]

a- Convolutional layer :

La couche de convolution est une étape cruciale dans l'extraction de caractéristiques précieuses d'une image. Elle utilise plusieurs filtres pour effectuer des convolutions sur une image, considérée comme une matrice de valeurs de pixels. Une image RVB a trois plans de pixels, tandis qu'une image en niveaux de gris n'en a qu'un seul (voir figure 2.15). Les couches convolutionnelles détectent des caractéristiques spécifiques dans les données d'entrée, qu'il s'agisse de l'image initiale ou de la sortie d'une couche précédente.

Par exemple, dans le modèle VGG-16, l'entrée est une image couleur de 224x224x3, et la sortie est une carte de caractéristiques appelée "*feature map*".[42]

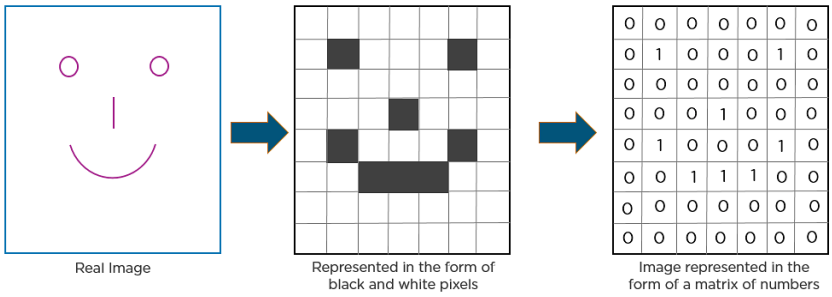

Figure 2.15: Transformation d'une Image Réelle en Matrice de Pixels [42]

Un filtre, ou noyau, est une petite matrice de taille N*N qui glisse sur l'entrée pour effectuer des convolutions, comme le noyau *Sobel* utilisé pour détecter les bords verticaux (comme le montre la figure 2.16). Le processus implique le calcul du produit scalaire entre les valeurs du filtre et celles de l'entrée, générant une valeur unique. Cette opération est répétée en déplaçant le filtre sur l'image, couvrant chaque section de l'entrée, appelée champ réceptif. Cela produit une *feature map* convoluée, souvent suivie d'une fonction d'activation *ReLU*. Les couches convolutionnelles et les filtres sont essentiels pour

extraire des caractéristiques pertinentes des images, facilitant la reconnaissance et la classification dans les réseaux de neurones convolutifs (CNN), ce qui les rend efficaces pour l'analyse et la compréhension des images. [43]

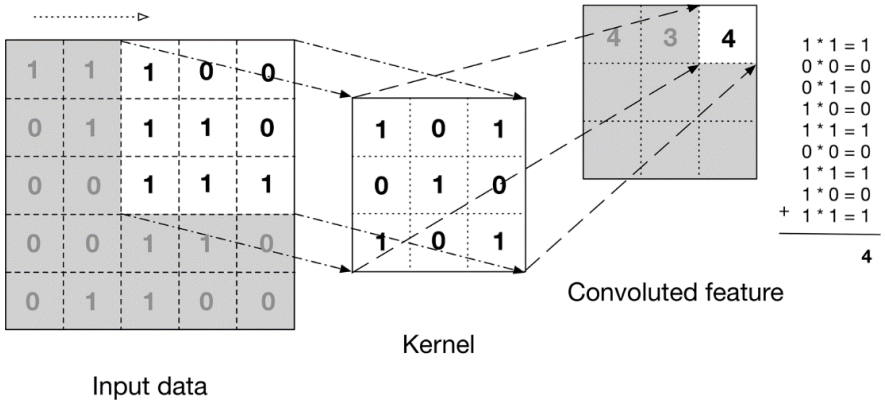

Figure 2.16 : Processus de Convolution dans les Réseaux de Neurones Convolutionnels [42]

b- La Fonction ReLU :

La fonction d'activation *ReLU (Rectified Linear Unit)* intervient après l'extraction des cartes de caractéristiques en remplaçant les valeurs négatives par 0 et en conservant les valeurs positives. Elle accélère l'apprentissage tout en maintenant les caractéristiques importantes et introduit de la non-linéarité, ce qui permet de mieux modéliser les relations complexes entre les caractéristiques.

c- Pooling layer

Le pooling est une opération de sous-échantillonnage dans les réseaux de neurones convolutifs (CNN) qui réduit la dimensionnalité des *feature maps*. Utilisant un filtre coulissant 2D (souvent de taille 2x2) et un pas configurable, le pooling peut être moyen ou max, ce dernier étant le plus couramment utilisé (voir figure 2.17).

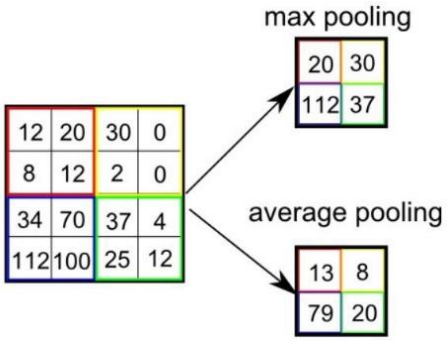

Figure 2.17 : Comparaison entre Max Pooling et Average Pooling dans les Réseaux de Neurones Convolutionnels [44]

Pour un emplacement de filtre donné, les valeurs de la tranche d'entrée sont soumises à une opération *max ()*. La valeur maximale est ensuite enregistrée en sortie, ce qui agrège les informations locales en une seule valeur. Cela réduit la taille spatiale des données tout en conservant les caractéristiques principales. Cette technique aide à réduire le sur apprentissage en diminuant le nombre de paramètres du réseau et rend les caractéristiques détectées plus invariantes aux petites translations ou déformations de l'image. [44]

d- Fully connected layer

*FLATTENING***,** ou aplatissement, transforme les matrices résultantes des cartes de caractéristiques regroupées en un seul vecteur linéaire unidimensionnel pour qu'elles puissent être passées dans une couche entièrement connectée pour la catégorisation ou la régression. Les couches *fully connected* (FC), situées vers la fin de l'architecture d'un réseau de neurones convolutif (CNN), viennent après les couches convolutionnelles et de pooling. Chaque neurone dans une couche *FC* est relié à tous les neurones de la couche précédente, permettant une combinaison dense de toutes les caractéristiques extraites. [46]

L'entrée d'une couche FC est ce vecteur unidimensionnel aplati, résultant des cartes de caractéristiques traitées par les couches précédentes. Chaque neurone de la couche FC applique une combinaison linéaire à ses entrées, calculant une somme pondérée, ajoutant un biais, puis appliquant une fonction d'activation comme ReLU pour introduire de la non-linéarité. [47]

La dernière couche *FC* utilise souvent une fonction d'activation softmax pour produire un vecteur de probabilités, chaque élément représentant la probabilité que l'image d'entrée appartienne à une classe spécifique. Les couches *FC* sont essentielles pour combiner et interpréter les caractéristiques complexes extraites par les couches précédentes, transformant ces informations en décisions finales pour les tâches de classification (voir figure 2.18).

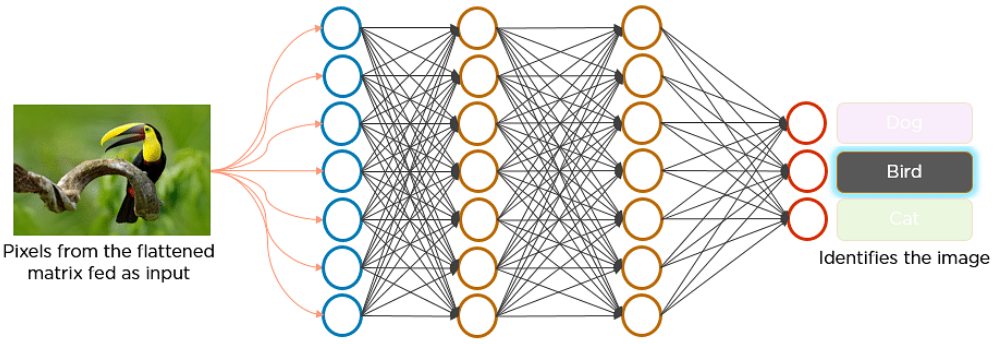

Figure 2.18 : Reconnaissance d'Image par Réseau Neuronal Profond [44]

2.7 Application de ml et dl dans les systèmes ADAS

Le Machine Learning (ML) et le Deep Learning (DL) sont cruciaux pour améliorer les systèmes ADAS en permettant l'apprentissage à partir des données, l'adaptation à de nouvelles situations et la prise de décisions éclairées. Ils renforcent les capacités des ADAS de la manière suivante :

2.7.1 Détection et suivi d'objet :

Les algorithmes de machine learning et de deep learning sont essentiels dans les systèmes ADAS pour la détection et le suivi des objets. Des techniques telles que les réseaux de neurones convolutifs (CNN), YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) et les réseaux de neurones convolutifs régionaux (R-CNN) analysent en temps réel les données des capteurs pour identifier des éléments tels que les véhicules en mouvement, les piétons et les obstacles. Ces approches permettent aux systèmes ADAS de prendre des décisions critiques comme le freinage d'urgence ou la correction de trajectoire, améliorant ainsi la sécurité routière. Grâce à la détection des voies et à la reconnaissance des piétons, ces algorithmes jouent un rôle déterminant dans l'optimisation de la sécurité des systèmes de conduite assistée. [49]

2.7.2 Diagnostics et réparations

Les diagnostics et réparations automobiles impliquent l'identification et la résolution des problèmes des véhicules. Les techniciens utilisent des outils avancés pour évaluer les systèmes des véhicules et appliquent leur expertise pour résoudre les problèmes détectés. Ils interagissent avec les clients pour expliquer les problèmes identifiés, les

solutions proposées ainsi que les coûts associés, tout en documentant rigoureusement les procédures pour assurer un suivi précis.

L'apprentissage continu est crucial pour se maintenir à jour sur les dernières techniques et technologies de réparation. Pour les diagnostics, des algorithmes de machine learning comme les réseaux de neurones artificiels (ANN), les machines à vecteurs de support (SVM) et les forêts aléatoires sont souvent employés. Ces algorithmes permettent de détecter des anomalies, de prédire des pannes potentielles et d'analyser les données des capteurs afin d'identifier les problèmes mécaniques et électroniques des véhicules. [50]

2.7.3 Systèmes d'info divertissement

Les systèmes d'info-divertissement intègrent des algorithmes d'apprentissage automatique, incluant des méthodes de recommandation basées sur le filtrage collaboratif et des techniques de traitement du langage naturel (NLP). Ces algorithmes prévoient et suggèrent des contenus, ajustent automatiquement les paramètres et facilitent une interaction intuitive avec les conducteurs.

2.7.4 Confort et plaisir de conduite

Les applications de confort des systèmes ADAS incluent des fonctionnalités comme le régulateur de vitesse adaptatif, le stationnement automatique et la reconnaissance des panneaux de signalisation. Ces systèmes font souvent appel à des algorithmes d'apprentissage automatique comme les SVM, les arbres de décision et les méthodes de classification probabilistes pour améliorer les performances et garantir une expérience de conduite plus agréable et pratique.

2.7.5 Modélisation des dépendances temporelles dans les systèmes ADAS

Les réseaux de neurones récurrents (RNN) et les réseaux de mémoire à long terme (LSTM) sont fréquemment intégrés aux systèmes ADAS pour analyser des données séquentielles et modéliser des dépendances temporelles. Ils sont essentiels pour des tâches comme la prédiction du comportement des conducteurs, la détection de la somnolence et la prévision des mouvements des véhicules environnants dans le cadre de la conduite autonome.

2.8 Conclusion

Ce chapitre a présenté les bases de l'IA appliquée aux systèmes ADAS, en examinant les différences entre le ML et le DL ainsi que les principes des réseaux de neurones artificiels, avec un focus sur les couches et les architectures, en particulier les réseaux convolutionnels (CNN). Nous avons également discuté des applications spécifiques du ML et du DL dans l'aide à la conduite.

Le prochain chapitre se concentrera sur les modèles de détection d'objets par vision par ordinateur, en mettant en avant le modèle YOLOv8 utilisé dans notre projet.

Chapitre 3 : ALGORITHMES DE VISION PAR ORDINATEUR POUR LA DETECTION D'OBJETS

3.1 Introduction

La vision par ordinateur a transformé notre capacité à interpréter les images numériques, notamment dans le domaine crucial de la détection d'objets. Des algorithmes avancés comme SSD (Single Shot MultiBox Detector), Fast R-CNN, et plus récemment YOLO (You Only Look Once), sont largement employés pour localiser avec précision des objets dans des images et des vidéos.

Dans ce chapitre, nous explorerons les principes fondamentaux de la vision par ordinateur en nous concentrant sur la détection d'objets. Nous aborderons les concepts clés de plusieurs algorithmes avancés populaires tels que SSD, Fast R-CNN et YOLO. Nous porterons une attention particulière à YOLO v8, en analysant ses capacités et ses applications dans divers contextes.

3.2 La vision par ordinateur

3.2.1 Définition

La vision par ordinateur, ou computer vision (CV), est une branche de l'intelligence artificielle visant à donner aux ordinateurs la capacité de "voir" et d'analyser des images et des vidéos de manière similaire aux humains. [51] Ce domaine a émergé dans les universités pionnières de l'IA à la fin des années 1960. La vision par ordinateur utilise des algorithmes mathématiques et statistiques pour extraire et interpréter des informations à partir de données visuelles. Ces algorithmes reposent souvent sur des réseaux de neurones, en particulier les réseaux de neurones convolutifs (CNN), qui s'inspirent du fonctionnement du cortex visuel humain. Les principaux types d'algorithmes en vision par ordinateur sont la détection d'objets, la segmentation d'objets et la classification (comme le montre la figure 3.1). Dans notre étude, nous nous concentrerons sur la détection d'objets. [52]

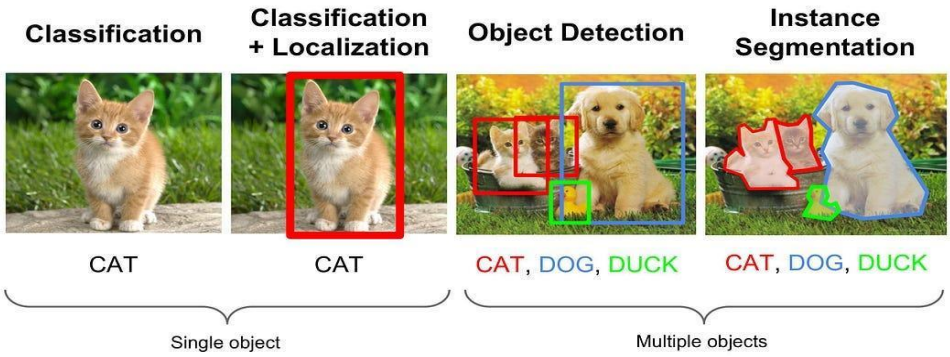

Figure 3.1 : Différents Types de Reconnaissance Visuelle en Vision par Ordinateur [52]

3.2.2 Domaine d'utilisation de computer vision

La vision par ordinateur est couramment appliquée dans de nombreux domaines tels que : [53]

- **Surveillance et sécurité** : Utilisée pour surveiller les zones de sécurité, comme les aéroports, les stades, et les centres commerciaux.
- **Vision industrielle** : Employée dans les processus de fabrication pour inspecter les pièces et détecter les défauts.
- **Conduite autonome** : Les véhicules autonomes utilisent la vision par ordinateur pour interpréter leur environnement et prendre des décisions de conduite.
- **Reconnaissance faciale** : Utilisée dans les smartphones et les applications pour identifier les visages sur les photos.
- **Médecine** : Aide au diagnostic et à l'analyse des images médicales, comme les scanners ou les IRM.
- **Tri des déchets** : Améliore le contrôle de qualité en identifiant et triant automatiquement les déchets.

3.3 Object détection en vision par ordinateur

3.3.1 Définition

La détection d'objets est une branche de la vision par ordinateur qui combine identification, classification et localisation d'objets dans des images ou vidéos. Elle dessine des cadres de délimitation et classe les objets détectés. Les méthodes basées sur les CNN se divisent en deux catégories (voir figure 3.2) : [54]

- **Algorithmes de détection à deux étapes** : Prédictions en plusieurs étapes, incluant R-CNN, Fast R-CNN, Faster R-CNN, et Mask R-CNN.

- **Algorithmes de détection à une seule étape** : Incluent SSD et YOLO.

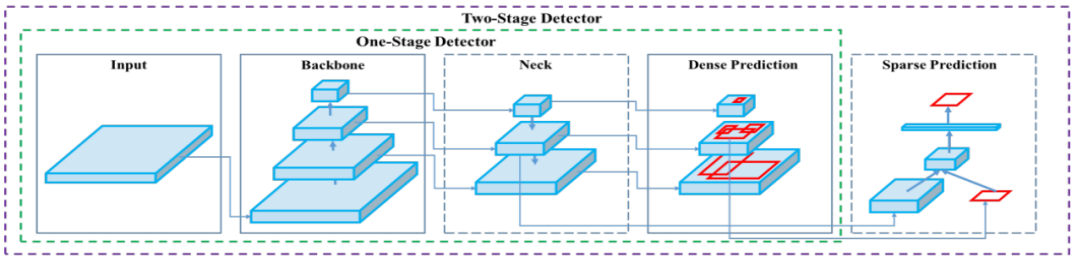

Figure 3.2 : Architecture générale des détecteurs d'objets basés sur des réseaux de neurones convolutifs a une étape et a deux étapes

3.3.2 Les Fondements de détection d'objet

Un réseau de détection d'objets à une double mission : identifier les objets présents dans une image et les classer selon leur catégorie. Cette technologie s'appuie sur des concepts et techniques essentiels qui permettent aux systèmes de vision par ordinateur de localiser avec précision et d'identifier les objets spécifiques dans des images ou des vidéos. Voici les principaux principes de la détection d'objets :

3.3.2.1 Traitement d'image

Le traitement des images, étape cruciale en vision artificielle, améliore la qualité des images et extrait des caractéristiques pertinentes à travers des opérations telles que le filtrage, la segmentation et la normalisation.

- **Le filtrage**

Le filtrage en traitement d'images consiste à modifier les valeurs des pixels pour améliorer l'image ou en extraire des caractéristiques spécifiques (voir figure 3.3). Cela inclut le lissage pour réduire le bruit, la correction de contraste, l'accentuation des contours pour améliorer la netteté, et la détection de bords pour préparer l'image à des analyses ultérieures. [83]

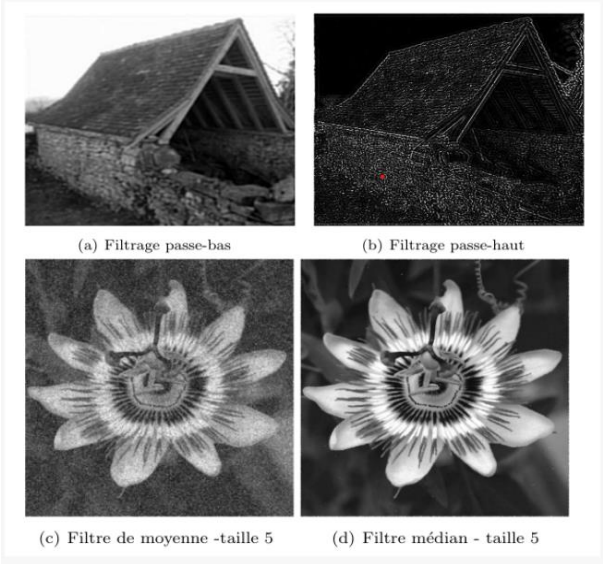

Figure 3.3 : exemple sur les filtres existants [84]

- **La normalisation**

La normalisation des pixels en deep learning pour l'analyse d'images standardise chaque pixel selon sa valeur maximale possible (255 pour les images 8 bits, 4095 pour les images 12 bits, 65535 pour les images 16 bits). Cela accélère l'apprentissage et améliore la comparabilité entre les images, facilitant ainsi un apprentissage efficace des modèles. [85]

3.3.2.2 Techniques d'ancrage et de sélection de régions

- **Les régions d'intérêt**

Les régions d'intérêt (RoI) sont des zones de l'image où des objets peuvent être présents, générées à partir de méthodes comme la recherche sélective ou la segmentation pour identifier des candidates à la détection d'objets. En se concentrant sur ces zones spécifiques, les RoI réduisent l'espace de recherche. Contrairement à Faster R-CNN qui utilise explicitement les RoI pour extraire des caractéristiques, YOLO prédit directement les boîtes englobantes et les classes pour chaque cellule de grille de l'image, simplifiant ainsi la détection d'objets avec une bonne précision et rapidité.

- **Anchor box**

Les boîtes d'ancrage, également appelées ancres ou boîtes par défaut, sont des cadres prédéfinis avec des tailles, des rapports d'aspect et des positions spécifiques utilisés comme références lors de la détection d'objets (comme le montre la figure 3.4). Elles sont disposées en grilles sur l'image pour capturer des objets de diverses tailles et formes. Ces boîtes jouent un rôle crucial

dans les algorithmes de détection d'objets, où elles sont utilisées pour prédire les emplacements et les formes des objets lors de l'entraînement et de l'inférence. Les dimensions et le nombre des boîtes d'ancrage dépendent de l'algorithme de détection d'objets et sont généralement définis en fonction des caractéristiques des données d'entraînement. [86]

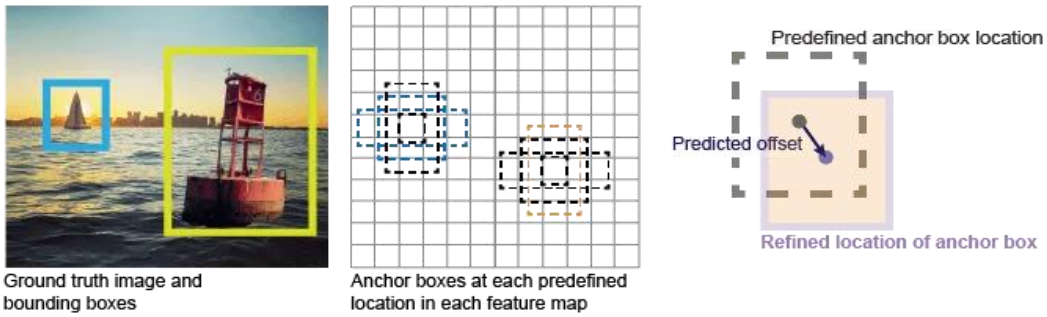

Figure 3.4 : Anchor boxes [86]

- **Bounding box**

Les boîtes de délimitation, aussi appelées ground truth boxes, sont des annotations essentielles pour évaluer la précision des modèles dans la localisation des objets lors de l'entraînement et des tests. Elles définissent des cadres rectangulaires qui entourent les objets ou zones d'intérêt spécifiques dans une image, délimités par les coordonnées (x_min, y_min) du coin supérieur gauche et (x_max, y_max) du coin inférieur droit. Ces boîtes servent à localiser précisément les objets dans les images et sont souvent annotées dans les ensembles de données pour la détection d'objets, en utilisant les contours des objets pour déterminer leur emplacement dans l'image (voir figure 3.5). [85]

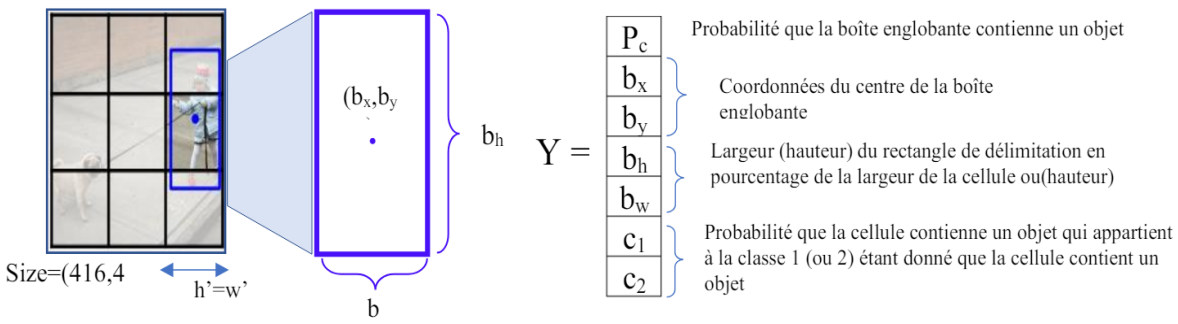

Figure 3.5 : le vecteur prédit dans le cas d'une seule boîte

- **Suppression non maximale**

La suppression non maximale (NMS) est une technique cruciale utilisée dans la détection d'objet pour sélectionner la meilleure boîte de délimitation pour chaque objet détecté, tout en

éliminant les boîtes redondantes (voir figure 3.6). L'algorithme NMS fonctionne en itérant sur toutes les boîtes détectées, en commençant par celle avec le score de confiance le plus élevé. Il compare ensuite les chevauchements entre cette boîte et les autres, supprimant progressivement les boîtes redondantes jusqu'à ce que seules les boîtes les plus pertinentes et non chevauchantes restent. Ce processus assure une détection précise et efficace des objets dans l'image. [59]

Figure 3.6 : Application de la suppression non maximale (Non Max Suppression) pour la détection d'objet

- **Intersection over union (IoU)**

L'Intersection over Union (IoU), également appelée indice de Jaccard, est une mesure couramment utilisée pour évaluer la précision de la localisation dans les modèles de détection d'objets. [71]

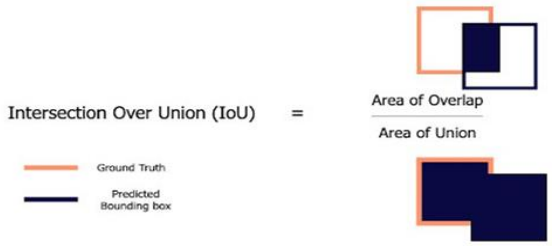

Figure 3.7 : IoU

L'IoU est définie comme le rapport entre l'aire d'intersection et l'aire d'union des boîtes de délimitation prédite et réelle (comme illustré dans la figure 3.7). Une valeur IoU élevée indique une meilleure correspondance entre les boîtes de délimitation prédite et réelle, reflétant une localisation plus précise. [72] :

- **Zone d'intersection :** La zone commune partagée par les deux boîtes englobantes (chevauchement).

- **Superficie d'union :** La zone totale couverte par les deux boîtes englobantes combinées.

La formule mathématique pour l'Intersection over Union (IoU) est :

$$IoU = \frac{TP}{(TP+FP+FN)} \tag{3.1}$$

Cette formule produit des valeurs d'IoU allant de 0 à 1, où 0 signifie aucun chevauchement et 1 signifie une correspondance parfaite entre la boîte prédite et la boîte de référence (voir figure 3.8).

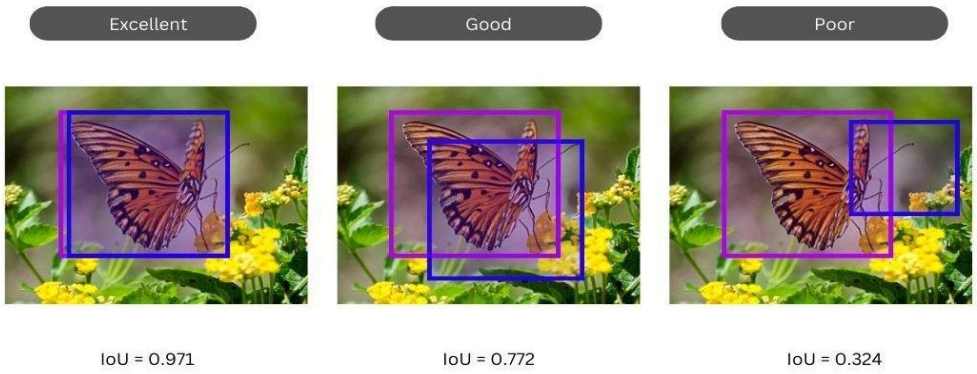

Figure 3.8 : Évaluation de la précision des détections d'objets basée sur l'Intersection over Union (IoU)

3.4 YOLO

3.4.1 Définition

YOLO, pour “You Only Look Once", est un algorithme de vision par ordinateur populaire pour la détection d'objets en temps réel et la segmentation d'image. Conçu par Joseph Redmon et Ali Farhadi en 2016, il est implémenté dans le framework Darknet.

Ce modèle à une seule étape utilise un réseau de neurones convolutionnel (CNN) pour prédire simultanément les boîtes de délimitation et les probabilités de classe des objets dans les images. Cela le rend rapide, précis et idéal pour les applications en temps réel. Depuis sa création, plusieurs versions de YOLO ont été développées : YOLOv1 à YOLOv9, YOLO-NAS, YOLO-World, PP-YOLO, et plus récemment YOLOv10. Chaque version améliore la précédente avec une précision accrue, des temps de traitement réduits et une meilleure détection des petits objets. [55]

3.4.2 Architecture Yolo

Ce modèle comporte 24 couches de convolution et 4 couches de max-pooling, suivies de 2 couches entièrement connectées (comme illustré dans la figure 3.9). Pour réduire le nombre de

canaux, une convolution 1x1 est utilisée, suivie d'une convolution 3x3. L'image est d'abord redimensionnée et du padding est ajouté si nécessaire. Ensuite, l'image traverse un réseau CNN où la fonction d'activation Leaky ReLU est utilisée partout, sauf dans la dernière couche où une fonction d'activation linéaire est appliquée. Enfin, la technique de dropout est employée pour éviter le surapprentissage. [56]

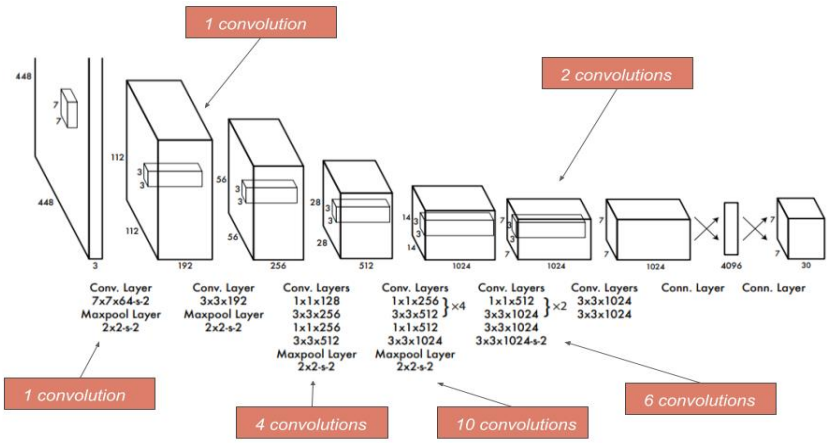

Figure 3.9 : Architecture YOLO

Cette architecture prend une image de dimension S×S et la divise en une grille de taille N×N (comme le montre la figure 3.10). Chaque cellule de cette grille représente un petit segment de l'image et prédit des cadres de délimitation ainsi que des classes avec un score de confiance.

Chaque boîte englobante prédit cinq valeurs : *(x,y,w,h)* et un score de confiance. Les coordonnées *(x,y)* indiquent le centre de la boîte par rapport aux limites de la cellule de la grille. Les dimensions *(w,h)* représentent la largeur et la hauteur de la boîte englobante. Le score de confiance reflète la probabilité qu'un objet soit présent dans cette boîte. [57]

En plus des cadres de délimitation, chaque cellule de la grille prédit un seul ensemble de probabilités de classe C, indépendamment du nombre de boîtes. Ces prédictions sont encodées dans un tenseur 3D de *taille N×N×(5B+C).*

Cela génère des scores de confiance spécifiques à chaque classe pour chaque boîte, représentant à la fois la probabilité de présence de la classe dans la boîte et la précision de la correspondance entre la boîte et l'objet réel. Une suppression non maximale est ensuite appliquée pour éliminer les prédictions redondantes, générant ainsi les prédictions finales.

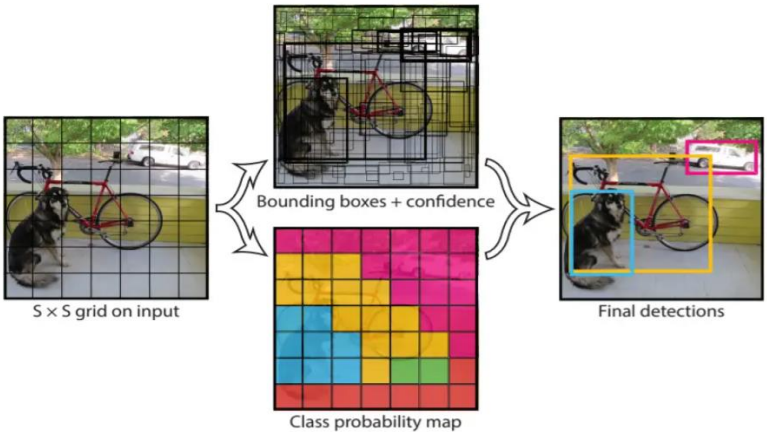

Figure 3.10 : processus d'identification d'objet avec Yolo [58]

3.4.3 Les indicateurs de performance YOLO

Les performances des modèles de détection d'objets sont évaluées à l'aide de plusieurs métriques standard : [71]

3.4.3.1 Matrice de confusion et précision/rappel

La matrice de confusion permet de déterminer les valeurs suivantes :

- **Vrai Positif (TP)** : Nombre de détections correctes d'objets.
- **Faux Positif (FP)** : Nombre de détections incorrectes d'objets.
- **Faux Négatif (FN)** : Nombre d'objets non détectés.
- **Vrai Négatif (TN)** : Nombre de fois où l'absence d'objet est correctement identifiée (rarement utilisée dans la détection d'objets).

À partir de ces valeurs, on calcule :

- **Précision :** Définie comme le nombre de vrais positifs divisé par la somme des vrais positifs et des faux positifs, la précision quantifie la justesse des prédictions positives par rapport à toutes les prédictions positives effectuées par le modèle. (ie : proportion de détections correctes parmi toutes les détections effectuées)

$$Précision = \frac{Vrai\ Positifs}{(Vrai\ Positifs + Faux\ Positifs)} \quad (3.2)$$

- **Rappel :** Le rappel est défini comme le nombre de vrais positifs divisé par la somme des vrais positifs et des faux négatifs. (ie : proportion d'objets correctement détectés parmi tous les objets présents.)

$$Rappel = \frac{Vrai\ Positifs}{(Vrai\ Positifs + Faux\ négatifs)} \quad (3.3)$$

- **F1 Score :** Le score F1 est la moyenne harmonique de la précision et du rappel. Il identifie le seuil de score de confiance optimal où précision et rappel atteignent leur meilleur équilibre. Un score F1 élevé indique que la précision et le rappel sont tous deux élevés, et inversement. [73]

$$F1\ Score = \frac{2 \times (Précision \times Rappel)}{Précision + Rappel} \quad (3.4)$$

3.4.3.2 Précision moyenne (AP)

La précision moyenne (AP) est une mesure qui combine précision et rappel pour différents seuils de confiance. Voici comment elle est liée à la matrice de confusion :

- **Courbe précision-rappel** : En variant le seuil de confiance du modèle, on obtient différents ensembles de TP, FP, et FN, permettant de tracer une courbe précision-rappel.
- **Calcul de l'AP** : L'AP est l'aire sous cette courbe précision-rappel. Pour chaque point de la courbe, la précision et le rappel sont calculés à partir des valeurs de la matrice de confusion correspondantes.

3.4.3.3 Précision moyenne moyenne (mAP)

La précision moyenne moyenne (mAP) est la moyenne des AP pour chaque classe d'objet détectée par le modèle. Elle est calculée comme suit :

- **AP pour chaque classe** : Pour chaque classe d'objet, on trace une courbe précision-rappel et calcule l'AP.
- **Calcul de la mAP** : La mAP est la moyenne de ces AP sur toutes les classes.

Ainsi, la matrice de confusion fournit les données de base nécessaires pour évaluer les performances des modèles de détection d'objets via les métriques AP et mAP.

3.5 YOLOv8

3.5.1 Définition

YOLO v8, développé par l'équipe Ultralytics et sorti en janvier 2023, est un algorithme de détection d'objet de pointe. Il identifie et localise les objets dans les images et les vidéos en un seul passage, offrant une vitesse et une précision remarquables (voir figure3.11). [60]

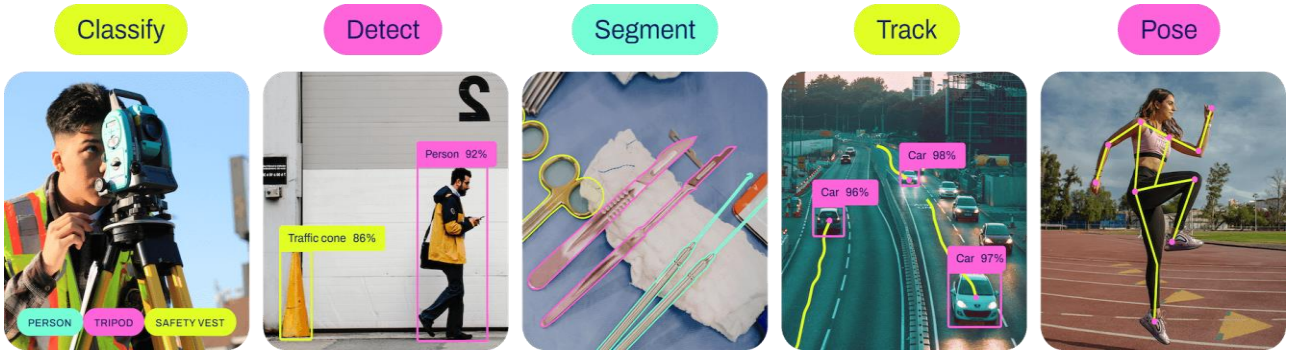

Figure 3.11 : les modes YOLOv8 [60]

3.5.2 Les variantes YOLOv8

YOLOv8 propose plusieurs variantes offrant des performances différentes : small (s), medium (m), large (l) et extra large (x). Le tableau 3.2 présente les performances de toutes les versions de YOLOv8. Il dispose également d'un package Python et d'une implémentation en ligne de commande (CLI), ce qui facilite son utilisation et son développement. [59]

Tableau 3.2 : STATE-OF-THE-ART YOLOv8 models [60]

model	size	mAP 50-95	speed CPU ONNX(ms)	speed A100 tensorRT (ms)	Speed (fps)	parameters
Yolov8n	640	37.3	80.4	0.99	40 fps	3.2
Yolov8s	640	44.9	128.4	1.20	30 fps	11.2
Yolov8m	640	50.2	234.7	1.83	20 fps	25.9
Yolov8l	640	52.9	375.2	2.39	15fps	43.7
Yolov8x	640	63.9	479.1	3.53	10 fps	68.2

3.5.3 L'architecture YOLO 8

L'architecture de YOLOv8, fondée sur les versions précédentes des algorithmes YOLO utilisant des réseaux CNN, se compose de trois blocs essentiels : le Backbone, le Neck et le Head (comme illustré dans la figure 3.12). [62]

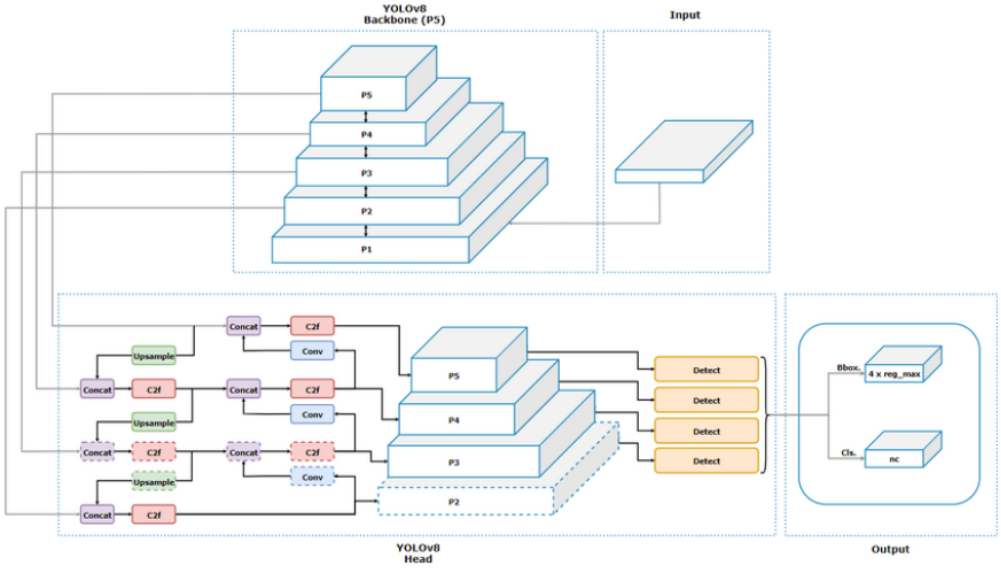

Figure 3.12 : Architecture YOLOv8

A) Backbone

YOLOv8 utilise un réseau backbone performant, le CSPDarknet53, qui sert de base pour extraire les caractéristiques significatives de l'image d'entrée. Ce backbone, une version améliorée de Darknet avec 53 couches convolutives et des connexions partielles inter-étages (CSP), améliore le flux d'informations entre les couches, augmentant ainsi la capacité d'apprentissage et la précision. Les principales fonctionnalités de ce backbone incluent : [63]

- Capturer des motifs simples dans les couches initiales, tels que les bords et les textures.
- Offrir plusieurs échelles de représentation, capturant des caractéristiques à différents niveaux d'abstraction.
- Fournir une représentation riche et hiérarchique de l'entrée.

B) Neck

Le "neck" ou cou agit comme un lien entre le backbone et la tête du modèle, permettant la fusion des caractéristiques et l'intégration des informations contextuelles. En collectant des

"feature maps" à différentes étapes du backbone, il combine des informations à diverses échelles, ce qui améliore la capacité du modèle à détecter des objets de tailles variées.

Dans YOLOv8, l'introduction de PANet (Path Aggregation Network) enrichit le flux d'informations à travers des échelles variées grâce à un réseau de pyramide de caractéristiques obtenu par le backbone. Cela renforce la capacité du modèle à gérer efficacement des objets de tailles différentes. Ces principales fonctionnalités incluent :

- Intégration d'informations contextuelles pour améliorer la précision de la détection en prenant en compte le contexte global de la scène.
- Réduction de la résolution spatiale et de la dimensionnalité des ressources pour améliorer l'efficacité de calcul, ce qui peut accélérer le modèle tout en potentiellement affectant la qualité. [63]

C) Head

La tête de détection de YOLOv8 est la dernière étape du réseau responsable de générer les sorties. Elle comprend plusieurs couches convolutionnelles suivies de couches entièrement connectées. Ces dernières prédisent les cadres de délimitation, les probabilités de classe pour chaque cellule de la grille de caractéristiques extraite du backbone et du neck, ainsi que les scores de confiance à différentes échelles pour la détection d'objets.
YOLOv8 améliore la tête YOLO en intégrant une affectation d'ancrage dynamique et une nouvelle fonction de perte IoU, ce qui permet de prédire efficacement des objets de différentes formes et tailles à l'aide de boîtes d'ancrage.
Une caractéristique distinctive de YOLOv8 est l'incorporation d'un mécanisme d'auto-attention dans sa tête. Ce mécanisme permet au modèle de se concentrer sur différentes parties de l'image et d'ajuster l'importance des caractéristiques en fonction de leur pertinence pour la tâche de détection d'objets. [63]

3.5.4 Comparaison de performance et amélioration entre les versions YOLO

Le tableau ci-dessous (voir tableau3.3) présente l'innovation et les améliorations des versions précédentes de YOLO. [64]

Tableau 3.3 : Évolution et Progrès des Versions Antérieures de YOLO

Version	**Améliorations**	**Vitesse/Précision**	**Application**
V1	Prédiction par cellule de grille, méthode a tir unique.	Rapide mais moins précis	Détection en temps réel (fondamentale (recherche)
V2 et V3	Boites d'ancrage, normalisation par lots (batch normalisation)	Plus rapide et plus précis	Diverses applications en temps réel
V4 et V5	Pooling pyramidal spatial, optimisations.	Equilibre entre vitesse et précision	Environnements exigeants, comme le transport
V6 àV8	Optimisations ciblées, architectures améliorées	Très précis et en temps réel	Applications spécialisées, comme la surveillance
V9	Amélioration de la détection des petits objets, intégration avec d'autres modèles d'IA et IA explicable	Précision et vitesse accrues	Application telles que l'imagerie médicale, la conduite autonome ou la détection de défauts industriels

Le modèle YOLOv8 surpasse les versions précédentes, telles que YOLOv5 et YOLOv7, en termes de performance de classification, comme illustré dans la figure 3.13 suivante :

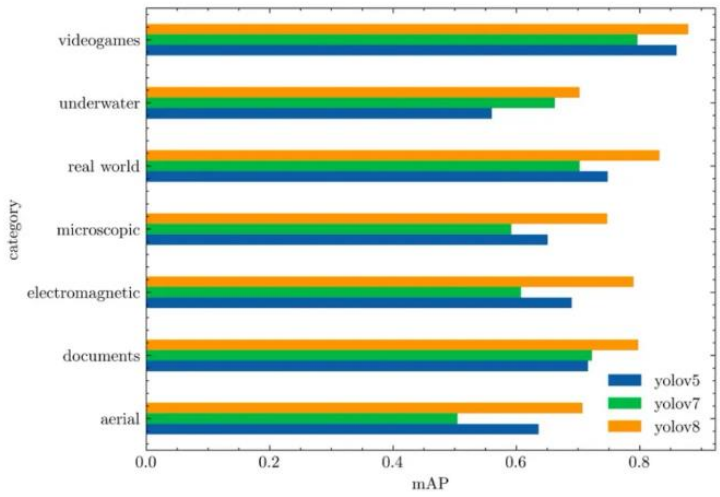

Figure 3.13: Moyenne de mAP de YOLO par catégoriesRF100[65]

Le graphique comparatif ci-dessous (voir figure 3.14) met en évidence l'évolution de la précision moyenne (mAP) entre YOLOv8 et les versions précédentes d'Ultralytics (YOLOv5, v6, et v7), démontrant une amélioration progressive jusqu'à ce que YOLOv8 atteigne les performances les plus élevées. [65]

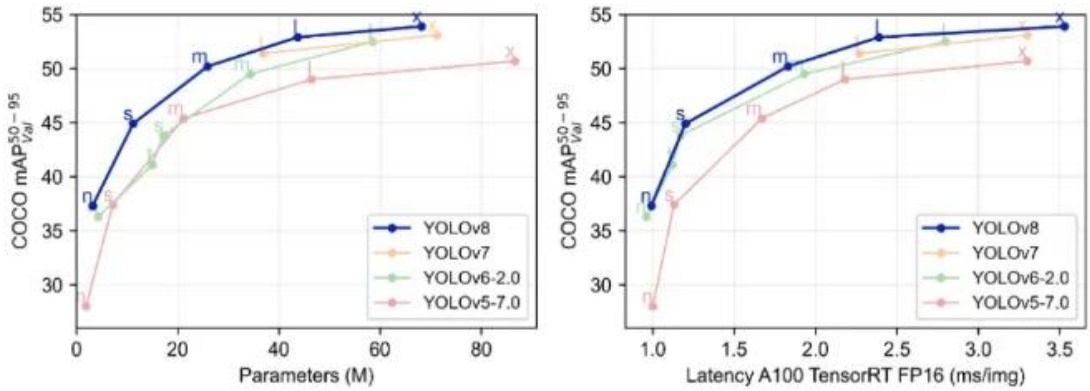

Figure 3.14 : La précision moyenne (mAP) des différentes versions de YOLO [65]

Le tableau ci-dessous (voir tableau 3.4) illustre les performances initiales des Ultralytics de première génération (YOLOv5 et ses dérivés), en comparaison avec la version améliorée (YOLOv8), et met en évidence leur différence pour une taille d'image de 640 pixels.

Tableau 3.4 : Comparaison entre la détection d'objet avec YOLOv8 vs YOLOv5 [66]

Taille du Modèle	YOLOv5	YOLOv8	Différence
Nano	28	37,3	+33,2%
small	37,4	44,9	+20,05%
medium	45,4	50,2	+10,57%
Large	49	52,9	+7,96%
Extra large	50,7	53,9	+6,31%

3.5.5 Les avantages de yolo v8

YOLOv8 excelle dans plusieurs domaines clés :

- **Vitesse en temps réel et précision élevée :** Idéal pour les applications comme les véhicules autonomes et la robotique, il offre des vitesses d'inférence impressionnantes sans compromettre la précision.

- **Polyvalence et compatibilité matérielle :** Intègre des fonctionnalités variées telles que la classification, la détection, la segmentation d'instance et l'estimation de la pose. Il est compatible avec une gamme étendue de matériels, des processeurs aux GPU.
- **Modèles pré-entraînés :** Facilite l'utilisation et le transfert d'apprentissage sur divers ensembles de données grâce à ses modèles pré-entraînés.
- **Légèreté et efficacité des ressources :** Conçu pour être léger, il optimise l'utilisation des ressources matérielles, ce qui le rend approprié pour les appareils avec des capacités limitées, tout en maintenant des performances élevées.
- **Open source et communauté active :** En tant que projet open source, YOLOv8 bénéficie du soutien continu d'une communauté dynamique, favorisant le développement collaboratif et l'intégration de nouvelles fonctionnalités.
- **Optimisation de l'architecture :** Développé avec des techniques d'optimisation avancées, tant au niveau de l'architecture du réseau de neurones que des algorithmes d'entraînement, il assure des performances maximales avec une utilisation minimale des ressources.

3.6 Autres algorithmes de détection d'objet

- **R-CNN :** R-CNN, ou Region-based Convolutional Neural Network, est un modèle CNN spécialement conçu pour la détection d'objets. Il introduit le concept de régions d'intérêt (RoI) pour améliorer la précision de la détection.

  En segmentant l'image en régions potentielles contenant des objets, R-CNN utilise un réseau de neurones convolutifs pour extraire les caractéristiques de ces régions. Ensuite, un classifieur est utilisé pour déterminer la présence d'objets et les classifier. Cependant, cette approche souffre d'une lenteur relative en raison du traitement séparé de chaque région d'intérêt. [67]

- **Fast R-CNN :** Fast R-CNN améliore le modèle R-CNN en intégrant plusieurs optimisations pour accroître à la fois la vitesse et la précision de la détection d'objets. Au lieu de traiter individuellement chaque région proposée, Fast R-CNN analyse l'image entière en une seule passe. Son architecture repose sur trois composants clés : un CNN de base pour extraire les caractéristiques de l'image, une couche de pooling spatial pour ajuster la taille des régions d'intérêt, et un réseau de détection pour classifier les objets et raffiner les boîtes englobantes. Cette approche réduit la redondance computationnelle et améliore l'efficacité globale du processus de détection.

- **Faster R-CNN :** Faster R-CNN améliore et unifie le processus de détection d'objets par rapport à son prédécesseur R-CNN. Intégrant la génération de propositions d'objets et la détection dans un seul réseau, il offre des gains significatifs en vitesse et précision.

  Son architecture se compose de trois éléments principaux : un backbone CNN comme VGG16 ou ResNet pour extraire les caractéristiques de l'image, un réseau de proposition de région (RPN) qui identifie les zones potentielles d'objets en parcourant la carte de caractéristiques du CNN, et enfin un réseau de détection qui classe ces propositions et affine leurs boîtes englobantes (voir figure 3.15). [67]

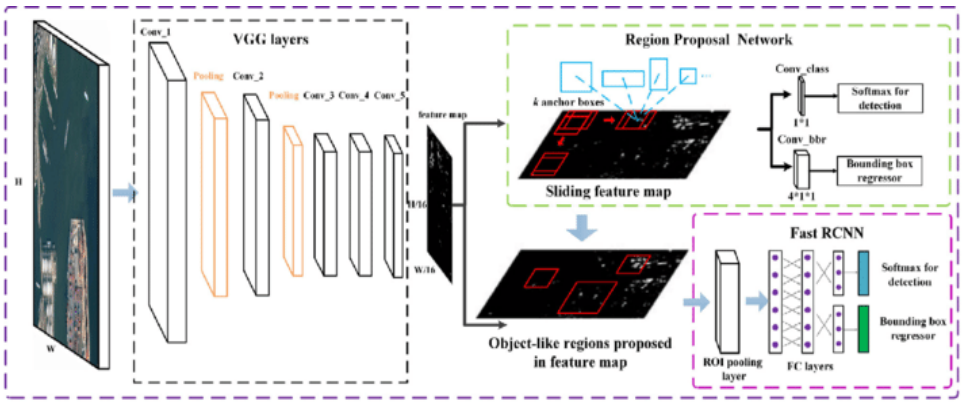

Figure 3.15 : Architecture du Faster R-CNN [68]

- **MASK R-CNN :** Mask R-CNN est une extension avancée du modèle Faster R-CNN qui combine la détection d'objets et la segmentation d'instances. Intégrant une architecture enrichie par une branche dédiée appelée "tête de masque", ce modèle prédit des masques de segmentation pixel par pixel pour chaque objet identifié. En utilisant des régions d'intérêt (RoI), il extrait des caractéristiques spécifiques à chaque objet, permettant ainsi la prédiction simultanée des classes, des boîtes englobantes et des masques de segmentation pour chaque instance détectée. Cette approche offre une segmentation sémantique précise, idéale pour des applications nécessitant une compréhension détaillée et précise de la scène visuelle. [69]
- **SSD :** Le Single Shot Multibox Detector (SSD) est un algorithme de détection d'objets en temps réel réputé pour sa précision élevée. Utilisant un réseau de neurones convolutifs profonds (CNN), il analyse une image en une seule passe directe pour générer des boîtes de délimitation et des prédictions de classe pour tous les objets détectés. Contrairement à d'autres méthodes, le SSD combine la localisation et la classification en une étape unique, ce qui lui confère son nom « Single Shot ». [70]

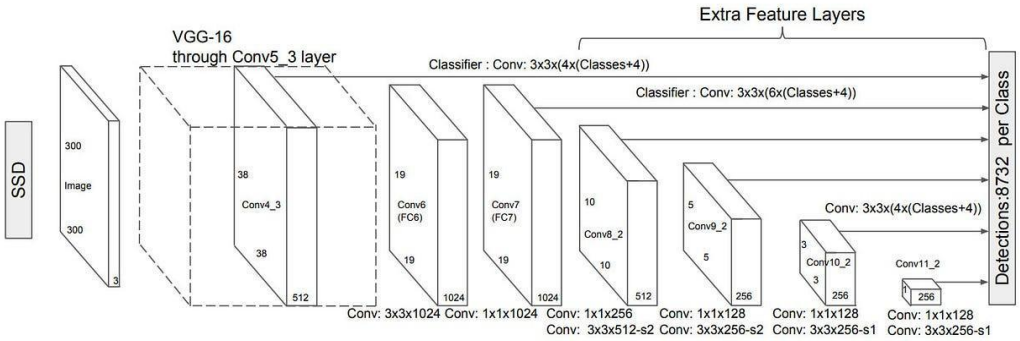

Figure 3.16 : Architecture du SSD

Le SSD utilise l'architecture VGG-16 pour extraire des cartes de caractéristiques. Initialement, il détecte les objets en utilisant la couche Conv4_3, puis applique des filtres de convolution 3 * 3 sur ces cartes pour évaluer les scores de localisation et de classe (comme le montre la figure 3.16). Pour détecter efficacement des objets de différentes tailles, le SSD exploite plusieurs couches de caractéristiques à différentes échelles. Cependant, à mesure que les couches convolutionnelles réduisent la dimension spatiale, la résolution des cartes de caractéristiques diminue. Cela peut poser problème pour la détection d'objets très petits.

Ainsi, le SSD utilise des couches de résolution plus basse pour détecter les objets de grande taille. Pendant l'entraînement, chaque boîte prédite est évaluée par rapport aux boîtes réelles pour calculer une perte composite, combinant la perte de localisation et la perte de confiance.

3.7 Conclusion

En conclusion, la détection d'objets en vision par ordinateur repose sur des algorithmes avancés comme YOLO. Cette méthode assure une localisation précise en temps réel, répondant aux standards actuels. YOLOv8, avec sa capacité de détection en une seule passe, ouvre de nouvelles perspectives d'application, de la sécurité routière à la surveillance vidéo.

CHAPITRE 4 : CONCEPTION ET IMPLEMENTATION

4.1 Introduction

Ce chapitre abordera tous les aspects de la mise en œuvre de notre système, en commençant par la configuration de l'environnement de travail. Ensuite, nous traiterons de l'entraînement du modèle YOLOv8 pour la détection des nids-de-poule et des ralentisseurs, ainsi que de l'utilisation du site web 'Makesense' pour l'étiquetage des images. Nous présenterons ensuite les résultats de l'analyse effectuée par ce système. Enfin, nous explorerons la mise en œuvre de notre système pour la détection en temps réel via le streaming direct ainsi que l'implémentation dans le RASPBERRY PI

4.2 Environnement de travail

4.2.1 Matériel :

Marque	**DELL Inc**
Model	Inspiron 3576
Type Système	SE 64bit, Processuer x64
SE	Windows 10 Professional
Processeur	Intel(R) Core(TM) i7-8550U CPU
Mémoire ram installé	8,00 Go

4.2.2 Logiciels :

- **Redimensionnement d'image** : Le prétraitement des images est essentiel pour assurer des performances optimales du modèle et éviter des problèmes durant l'entraînement. Pour notre modèle, il est recommandé de redimensionner les images à une taille de 640x640 pixels.
- **Makesense.ai :** une plateforme open-source pour l'étiquetage manuel des objets, supportant divers types d'étiquettes comme le rectangle, la ligne, le point et le polygone.

Elle prend en charge plusieurs formats de sortie, dont YOLO, VOC XML, VGG JSON et CSV. [89]

- **Google Colaboratory** : ou Google Colab, est une plateforme de calcul en cloud fournie par Google. Elle permet d'exécuter du code Python et des notebooks Jupyter directement dans le navigateur sans nécessiter de configuration locale. Il offre un accès gratuit aux GPU et TPU, accélérant les calculs et les modèles d'apprentissage profond, et facilite la collaboration grâce à des fonctionnalités de partage et de commentaires intégrées. [90]
- **Google Drive** : une solution de stockage en ligne de Google, permettant de sauvegarder, stocker et partager des fichiers dans le cloud. Les utilisateurs peuvent accéder à leurs fichiers depuis n'importe quel appareil connecté à Internet et collaborer en temps réel. Dans notre projet, les images stockées sur Google Drive ont été utilisées dans Google Colab.
- **PyCharm** : un environnement de développement intégré (IDE) développé par JetBrains, spécialisé dans le langage Python. Il propose une gamme complète d'outils pour le développement, incluant l'édition de code, la navigation de projet, le débogage et la gestion des tests, ce qui le rend populaire pour la programmation en Python grâce à sa richesse en fonctionnalités et sa convivialité. [91]

4.2.3 Langages de programmation

- **Python (v3.12) :** Python est un langage de programmation largement utilisé, réputé pour sa simplicité et sa polyvalence. Grâce à sa syntaxe claire et concise, il est adapté à divers domaines, notamment le développement web, l'analyse de données et l'intelligence artificielle. Python bénéficie du soutien d'une communauté active de développeurs et prend en charge plusieurs paradigmes de programmation. De plus, il offre une vaste bibliothèque standard, facilitant le développement d'applications variées. [92]

4.2.4 Bibliothèques [93]

4.2.4.1 L'entrainement sur Google colab

- **Ultralytics** : Les principales bibliothèques associées au package Ultralytics incluent matplotlib, opencv-python, pillow, pyyaml, requests, scipy, torch, torchvision, tqdm, psutil, pandas, seaborn, contourpy, et numpy.
- **Tqdm** : Il s'agit d'une bibliothèque Python utilisée pour afficher des barres de progression lors de l'exécution de boucles ou d'opérations longues.

- **Module** : Un fichier contenant du code Python, incluant des définitions de fonctions, des variables et d'autres éléments, permettant une organisation et une structuration logique du code.

  En Python, les modules peuvent être importés dans d'autres scripts à l'aide de l'instruction `import`. Dans notre modèle, nous avons implémenté les modules suivants :

  - Os : Ce module fournit des fonctionnalités pour interagir avec le système d'exploitation, telles que la création de dossiers, la manipulation de chemins de fichiers, etc.
  - Shutil : Ce module offre des opérations de haut niveau sur les fichiers et dossiers, telles que la copie, le déplacement et la suppression.
  - Random : Ce module permet de générer des nombres aléatoires.

4.2.4.2 Prédiction

- **Cvzone** : Une bibliothèque Python dédiée aux applications de vision par ordinateur, fournissant des outils pour la détection d'objets, le suivi et la réalité augmentée.
- **Math** : Un module standard de Python pour effectuer des opérations mathématiques avancées, incluant les fonctions trigonométriques, logarithmiques et algébriques.
- **Numpy** : Une bibliothèque Python conçue pour le calcul numérique, offrant des structures de données efficaces et des opérations sur des tableaux multidimensionnels.
- **Cv2** : OpenCV (Open Source Computer Vision) est une bibliothèque largement utilisée pour le traitement d'images et les applications de vision par ordinateur.
- **Time** : Un module standard de Python pour la gestion du temps, permettant de mesurer les temps d'exécution et de manipuler les dates et heures.
- **Pandas** : Une bibliothèque Python pour la manipulation et l'analyse des données, offrant des structures de données et des outils performants pour le traitement des données tabulaires.

4.3 Réalisation

Le schéma ci-dessous (comme illustré dans la figure 4.1) présente les étapes principales de notre projet de détection des ralentisseurs et des défauts de chaussée, allant de la collecte des données à l'implémentation finale. Il détaille le processus de préparation et d'entraînement du modèle, les ajustements nécessaires pour optimiser les performances,

ainsi que l'intégration du système sur un dispositif embarqué, le Raspberry Pi. Avant de commencer l'entraînement de notre modèle YOLOv8, la création d'une base de données est l'étape la plus cruciale. Il est impératif que cette base de données soit correctement étiquetée en fonction des classes d'obstacles, car nous traitons deux classes dans notre cas : les ralentisseurs et les nids-de-poule. De plus, il est essentiel que les fichiers d'annotations (ground truth) soient au format YOLO.

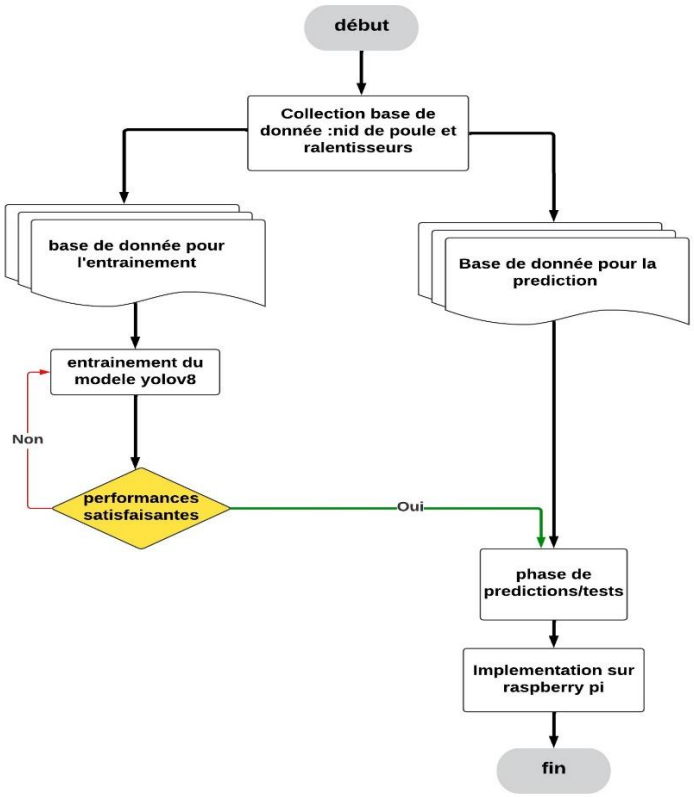

Figure 4.1 : Processus du système de détection des ralentisseurs et des défauts de chaussées par vision artificielle.

4.3.1 La création de la base de données

4.3.1.1 Collection d'images pour dataset : (les images)

Dans l'ensemble des données recueillies, nous avons collecté environ 1735 images sur la plateforme Roboflow [7] [81] et 1728 images de dos d'âne et de nids-de-poule dans divers endroits de l'université et dans les espaces publics (rues, routes de Blida). Ces images ont été prises pendant la journée afin de capturer une variété de conditions de visualisation et d'éclairage. Elles ont été prises avec un téléphone Samsung M20 Android équipé d'une

caméra de 13 MP. Les images capturées par téléphone ont été prises par beau temps ensoleillé : certaines le matin à 10h, d'autres le soir à 16h.

Les images capturées par le téléphone proviennent de vidéos converties en images à l'aide du site web « VIDEO TO JPG CONVERTER » [82] La plupart des ralentisseurs étaient mal entretenus, avec des peintures décolorées. L'ensemble de données recueillies comprend des images de ralentisseurs marqués, de ralentisseurs avec des peintures fanées, de nids-de-poule secs et de nids-de-poule remplis d'eau de pluie.

Ces images ont été divisées en deux parties : une pour la phase d'entraînement du modèle et une autre pour la phase de prédiction. Les images de prédiction sont des images que le modèle n'a jamais vues ni utilisées durant l'entraînement. Nous avons étiqueté uniquement les images d'entraînement et non les images de test.

4.3.1.2 Etiquetage des images (annotation/ labelling)

L'étiquetage des images est une opération qui consiste à marquer ou annoter des éléments spécifiques dans une image avec des informations descriptives, utilisé pour la détection d'objets. Ces informations peuvent inclure des catégories, des régions d'intérêt, des coordonnées, etc. Dans notre cas, nous avons utilisé la plateforme Makesense.ai. Chaque objet d'intérêt dans une image est généralement encadré par un rectangle de délimitation et associé à une classe (par exemple : "pothole" ou "speedbump") (comme le montre la figure 4.2). Ces rectangles de délimitation et leurs classes associées sont souvent enregistrés dans un format spécifique, tel que le format YOLO.

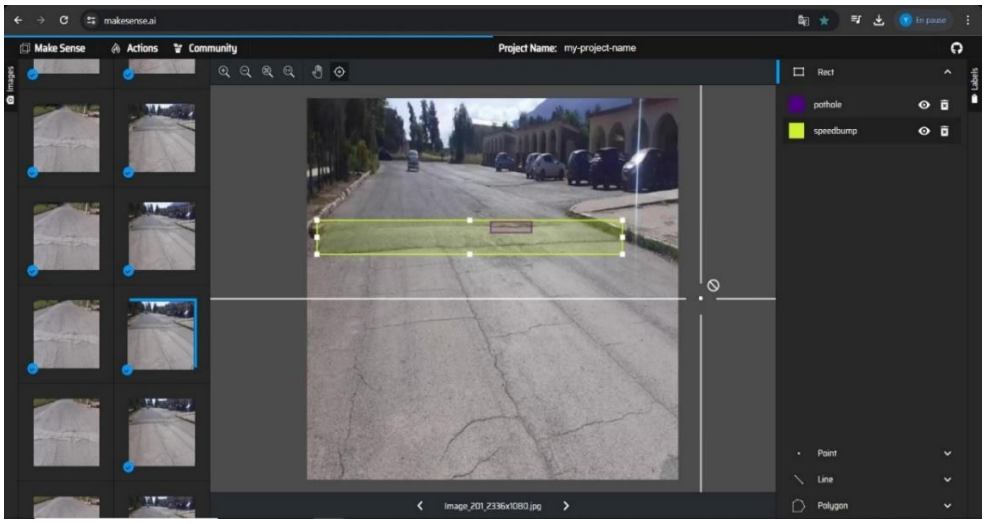

Figure 4.2: Annotation de données sur la plateforme web Make Sense.ai

Le processus d'étiquetage des images est réalisé manuellement. À la fin de cette étape, nous obtenons pour chaque image des fichiers texte (.txt).

Chaque fichier indique les coordonnées (x, y, h, l) des rectangles englobants de chaque classe d'objet. Le fichier est structuré comme suit : <Classe d'objet> <centre-x> <centre-y> <largeur> <hauteur>.

Pour les deux classes d'objets : le chiffre 0 représente "pothole" et le chiffre 1 représente "speedbump".

Dans la figure ci-dessous :

- Le premier segment « 1/0 » représente la classe : ralentisseur/nid-de-poule : il a identifié un seul ralentisseur et deux nids-de-poule.
- Les quatre autres segments (ex : 0.489340, 0.387479, 0.671743, 0.0077843) représentent les coordonnées du cadre d'étiquetage (voir figure 4.3).

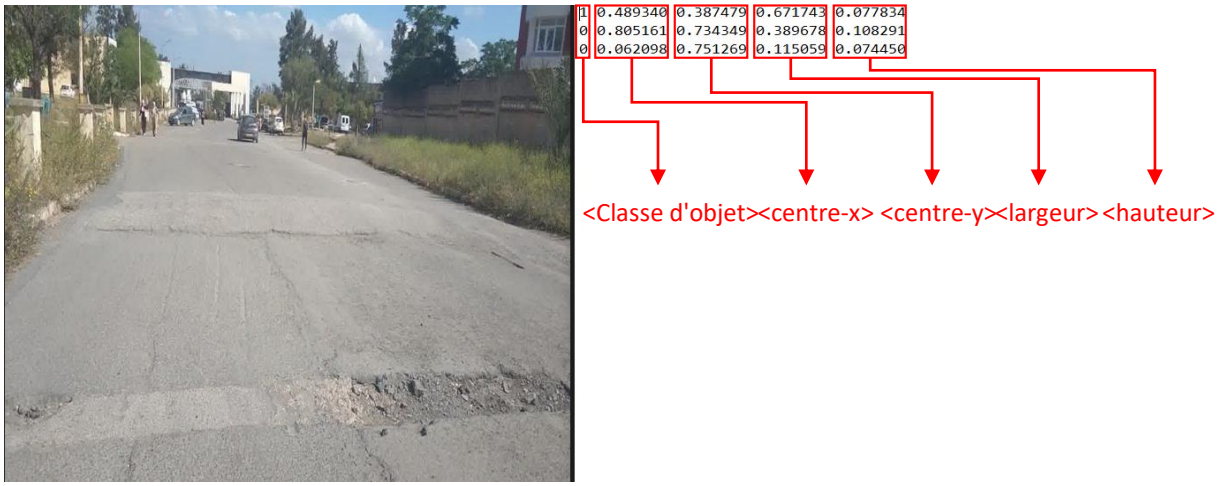

Figure 4.3: Images accompagnées de leur fichier texte contenant les coordonnées des boîtes englobantes et des classes correspondantes.

4.3.2 Entrainement du modèle YOLOv8

Le processus d'entraînement d'un modèle comprend plusieurs étapes essentielles : la préparation des données, la sélection et la configuration du modèle, l'entraînement avec les données, la validation pour ajuster les paramètres, le test pour évaluer la performance et, enfin, le déploiement pour des prédictions réelles. Chaque étape contribue à créer un modèle précis et efficace pour des applications concrètes.

4.3.2.1 Préparation à l'entrainement

Avant de commencer l'entraînement, nous avons créé un dossier nommé **yolov8** sur Google Drive contenant notre base de données (**data_4**) et le fichier YAML (**dataset.yaml**) (voir figure 4.4).

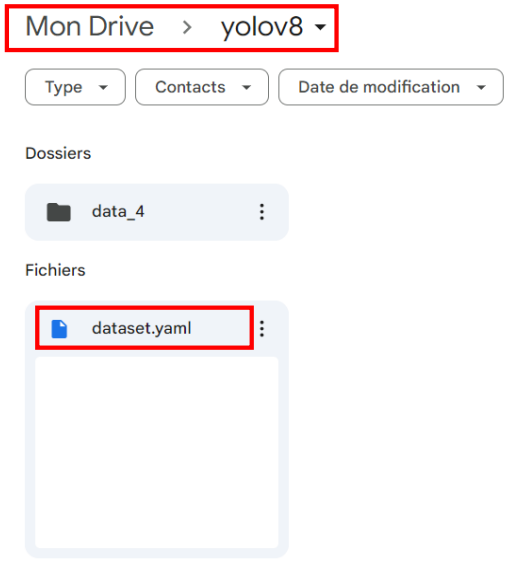

Figure 4.4 : Interface de Google Drive affichant les fichiers nécessaires pour l'apprentissage.

Le fichier YAML sert de carte de référence en fournissant une vue d'ensemble claire et bien organisée des paramètres et des configurations nécessaires pour entraîner, évaluer et déployer ce modèle, ce qui en fait un outil précieux pour la gestion et la documentation des expériences de l'apprentissage (voir figure 4.5).

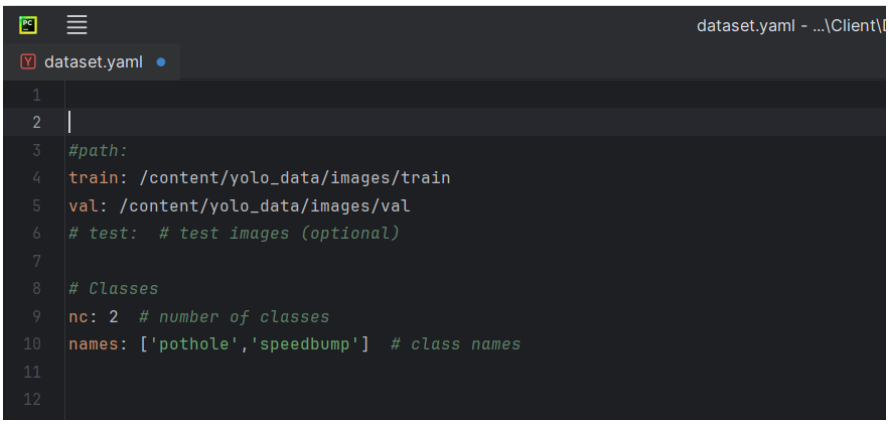

Figure 4.5: Contenu du fichier YAML.

Après avoir préparé, traité, nettoyé et organisé les données, nous passerons à l'exécution du code.

4.3.2.2 Exécution de l'algorithme/modèle :

Nous avons choisi Google Colab comme environnement d'entraînement en raison de sa capacité à fournir des GPU, ce qui accélère considérablement le processus d'entraînement.

1) La première étape consiste à configurer les paramètres d'exécution pour connecter l'environnement à un GPU T4. Ensuite, nous installons le package Ultralytics (comme le montre la figure 4.6), qui inclut les frameworks et bibliothèques développés par l'équipe Ultralytics, les créateurs de YOLOv8.

```
[ ]  !pip install ultralytics
```

Figure 4.6: package Ultralytics installé

2) L'étape suivante consiste à installer les bibliothèques nécessaires pour la suite du processus (voir figure 4.7).

```
[ ]  !pip install tqdm --upgrade
     from tqdm.notebook import tqdm

[ ]  import os
     import shutil
     import random
```

Figure 4.7: bibliothèques installés

3) Après cela, nous accédons à Google Drive pour importer les fichiers et les placer dans des dossiers sur Colab (comme illustré dans la figure 4.8) :

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

Figure 4.8: accès à Google drive

4) Les chemins ci-dessous (voir figure 4.9) sont utilisés pour organiser les données nécessaires à l'entraînement et à la validation du modèle YOLO.

```
[ ]  train_path_img = "./yolo_data/images/train/"
     train_path_label = "./yolo_data/labels/train/"
     val_path_img = "./yolo_data/images/val/"
     val_path_label = "./yolo_data/labels/val/"
     test_path = "./yolo_data/test"
```

Figure 4.9: chemin data.

5) Le code ci-dessous (comme le montre la figure 4.10) permet de diviser l'ensemble de données en deux parties distinctes :
   - Les images pour l'entraînement du modèle (training set)
   - Les images pour la validation du modèle (validation set).

```python
# Cette fonction divise les données d'un répertoire en ensembles d'entraînement et de validation.
def train_test_split(path,neg_path=None, split = 0.2):
    print("------ PROCESS STARTED -------")

    # Récupérer les noms de fichiers sans extensions (.jpg, .txt) et les convertir en un ensemble unique
    files = list(set([name[:-4] for name in os.listdir(path)]))

    print (f"--- This folder has a total number of {len(files)} images---")

# Fixer la graine aléatoire pour la reproductibilité et mélanger les fichiers
    random.seed(42)
    random.shuffle(files)

# Calcul les tailles des ensembles de test et d'entraînement :

    test_size = int(len(files) * split)
    train_size = len(files) - test_size

    ## Création des répertoires nécessaires pour les ensembles d'entraînement et de validation :
    os.makedirs(train_path_img, exist_ok = True)
    os.makedirs(train_path_label, exist_ok = True)
    os.makedirs(val_path_img, exist_ok = True)
    os.makedirs(val_path_label, exist_ok = True)

    # Copier les fichiers pour l'ensemble d'entraînement
    for filex in tqdm(files[:train_size]):
      if filex == 'classes':
          continue
      shutil.copy2(path + filex + '.jpg',f"{train_path_img}/" + filex + '.jpg' )
      shutil.copy2(path + filex + '.txt', f"{train_path_label}/" + filex + '.txt')
```

```python
  #Afficher un message de confirmation pour les données d'entraînement :
   print(f"------ Training data created with 80% split {len(files[:train_size])} images -------")

  #Ajouter les images négatives si spécifiées :
   if neg_path:
       neg_images = list(set([name[:-4] for name in os.listdir(neg_path)]))
       for filex in tqdm(neg_images):
           shutil.copy2(neg_path+filex+ ".jpg", f"{train_path_img}/" + filex + '.jpg')

       print(f"------ Total  {len(neg_images)} negative images added to the training data -------")

       print(f"------ TOTAL Training data created with {len(files[:train_size]) + len(neg_images)} images -------")

   #Copier les fichiers pour l'ensemble de validation :
   for filex in tqdm(files[train_size:]):
     if filex == 'classes':
         continue
     # print("running")
     shutil.copy2(path + filex + '.jpg', f"{val_path_img}/" + filex + '.jpg' )
     shutil.copy2(path + filex + '.txt', f"{val_path_label}/" + filex + '.txt')

   print(f"------ Validating data created with a total of {len(files[train_size:])} images ----------")

   print("------ TASK COMPLETED -------")

# Exemple d'utilisation de la fonction ( fonction utilisé sur le fichier yolov8 situé dans le drive )
train_test_split('/content/drive/MyDrive/yolov8/data_4/')
## sans image negative
```

Figure 4.10 : code de tri et division de dataset.[94]

6) Ci-dessous (voir figure 4.11) se trouve le résultat de l'algorithme de tri et de division. Les images du dataset ont été réparties en 80% pour l'entraînement et 20% pour la validation.

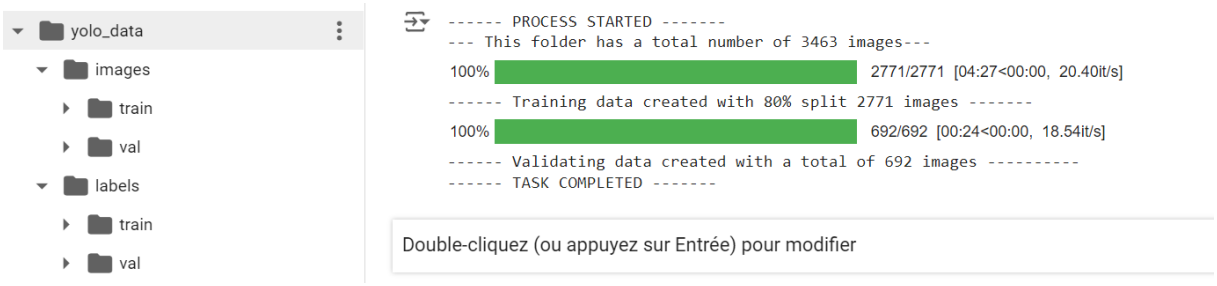

Figure 4.11: Trajectoire des images avec répartition en pourcentage.

7) Faire appel à la fonction Checks() de ultralytics comme illustré dans la figure 4.12 :

```
[ ]  import ultralytics
     ultralytics.checks()
```

Figure 4.12 : Appellation à la fonction checks() de ultralytics

8) Avant de commencer l'entraînement, nous avons configuré le mode de travail en 'train' et opté pour la variante YOLOv8 'n'(nano) avec une résolution d'image de 640 pixels. En raison des contraintes de la GPU sur Google Colab, nous avons ajusté les paramètres d'entraînement du modèle pour inclure 146 epochs avec une taille de batch de 35. L'entraînement complet a duré 3 heures et 20 minutes.

```
!yolo task=detect mode=train model=yolov8n.pt data=/content/drive/MyDrive/yolov8/dataset.yaml
epochs=146 imgsz=640 batch=35 project=/content/drive/MyDrive/yolov8/training_results
name=pfe
```

9) Lorsque l'opération est lancée, l'architecture YOLO s'affiche (voir figure 4.13) :

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.block.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.block.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.block.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.block.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.block.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.conv.Concat	[1]
21	-1	1	493056	ultralytics.nn.modules.block.C2f	[384, 256, 1]
22	[15, 18, 21]	1	751702	ultralytics.nn.modules.head.Detect	[2, [64, 128, 256]]

Figure 4.13 :L'architecture lors de l'entraînement

- De *Conv* à *SPPF* : Cette section correspond au backbone, qui extrait diverses caractéristiques à différentes résolutions.
  - Les blocs *Conv* se réfèrent aux couches de convolution où la carte des caractéristiques est traitée. Ces blocs sont responsables de l'extraction des caractéristiques de base de l'image telles que les bords, les textures et les motifs.
  - SPPF (Spatial Pyramid Pooling Fast) est un bloc utilisé à la fin du backbone pour produire des caractéristiques fixes qui représentent des objets de différentes tailles, sans redimensionner les images ni perdre d'information spatiale. SPPF combine les caractéristiques extraites à différentes échelles pour obtenir une représentation plus robuste des objets présents dans l'image.
- De Upsample à Detect : Cela correspond à la tête (head) du modèle.
  - La couche d'upsampling est utilisée pour augmenter la résolution de la carte des caractéristiques issue de SPPF, afin de la rendre compatible avec la carte des caractéristiques du bloc **C2F**. Cela permet d'intégrer les informations provenant de différentes couches pour améliorer la précision de la détection.

10) La détection représente la phase ultime où les objets sont repérés et localisés dans l'image. En utilisant les caractéristiques combinées, cette étape prédit les boîtes englobantes ainsi que les classes des objets détectés, assurant une localisation précise et une détection fiable.

11) Une fois l'entraînement terminé, les résultats sont enregistrés dans un dossier nommé "training_results" sur Google Drive sous le titre "pfe". Les poids du modèle sont sauvegardés dans un répertoire appelé "weights", qui contient deux fichiers : "best.pt" et "last.pt".
Voici ce que l'on trouve dans ces résultats (comme le montre la figure 4.14):

Figure 4.14 : train Batch

12) La prédiction des images de validations (voir figure 4.15 et figure 4.16) :

Figure 4.15:Validation batch prédiction

Figure 4.16 : images de validation avec étiquettes.

4.3.3 Evaluation des performances :

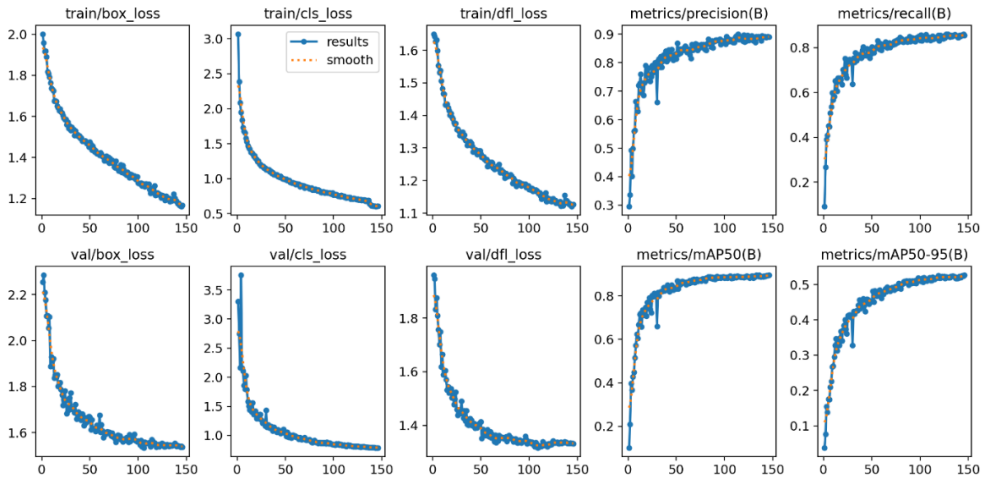

Figure 4.17 : graphe montrant les résultats du modèle YOLOv8n après l'entraînement.

Discussion :

Les graphiques de performance montrent des résultats satisfaisants (illustrés dans la figure 4.17) : les pertes liées aux boîtes englobantes, à la classification et à la distribution focale diminuent,

ce qui témoigne d'une réduction des erreurs au cours des epochs. En parallèle, la précision, le rappel, ainsi que les mAP50 et mAP 50-95 montrent une amélioration progressive au fil de l'entraînement.

Ces observations indiquent que notre modèle a appris de manière équilibrée, évitant à la fois le sur-apprentissage et le sous-apprentissage, ce qui est crucial pour des performances optimales en détection d'objets.

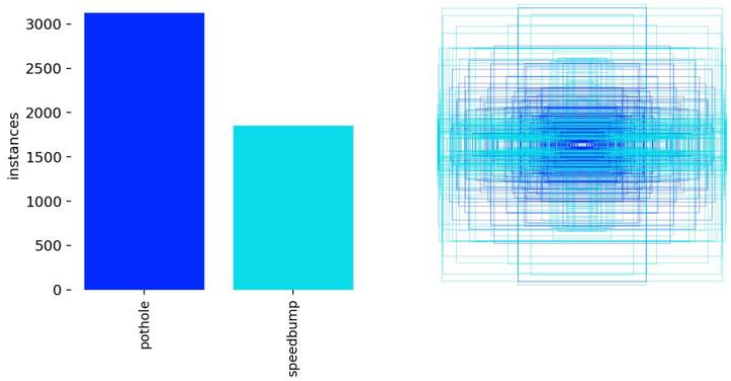

Figure 4.18: Des statistiques montrant le nombre d'étiquettes pour les nids-de-poule et les ralentisseurs.

- Lorsque la précision augmente, la confiance augmente (voir figure 4.19).

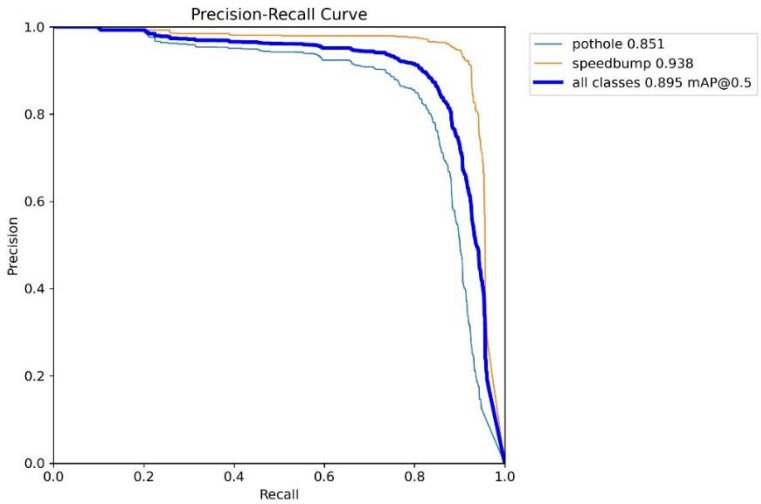

Figure 4.19: précision en fonction du rappel.

- Lorsque le rappel augmente, la confiance diminue et vice versa (voir figure 4.20 et figure 4.21).

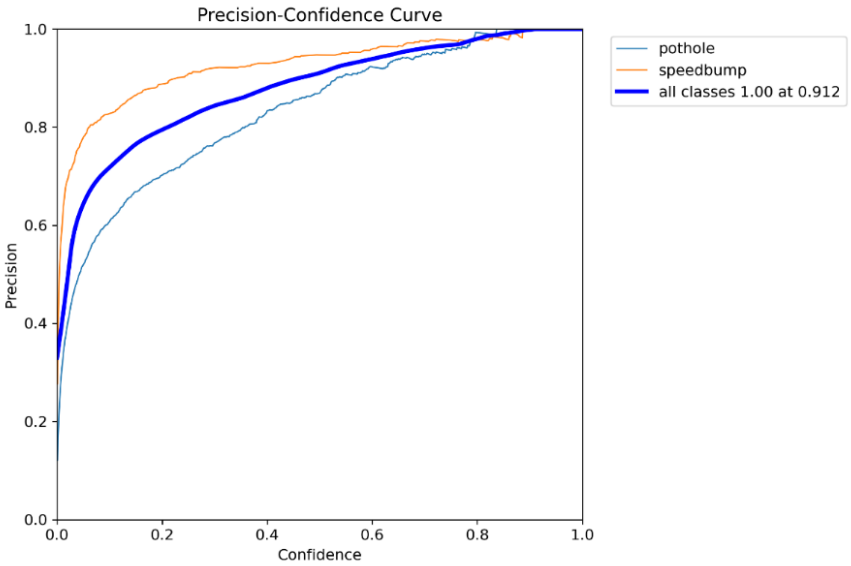

Figure 4.20 augmentation de précision en fonction de la confiance.

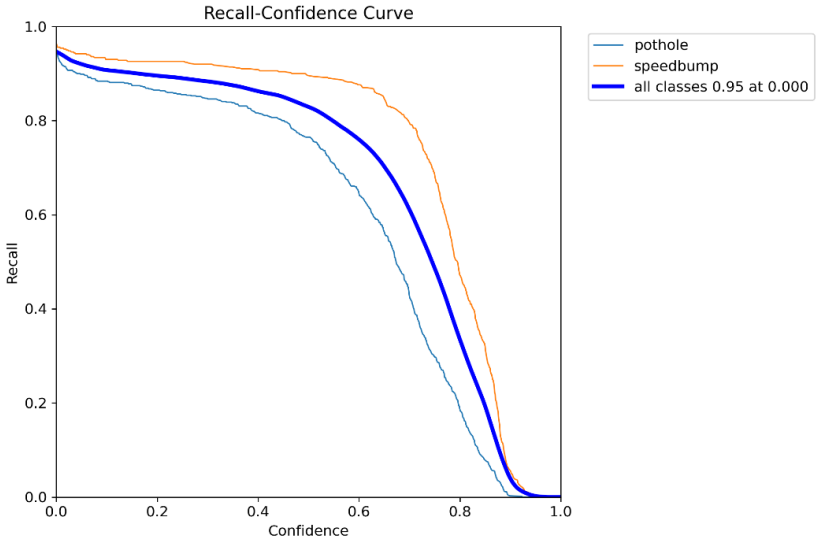

Figure 4.21 : diminution du rappel en fonction de la confiance.

- La matrice ci-dessous (comme illustré dans la figure 4.22) illustre le nombre de prédictions exactes et incorrectes générées par le modèle de classification sur un ensemble de données contenant des annotations pour les nids-de-poule et les ralentisseurs. Elle confronte les prédictions du modèle aux valeurs réelles (étiquettes) de l'ensemble de données. Une matrice de confusion typique présente les nombres bruts de vrais positifs, faux positifs, vrais négatifs et faux négatifs.

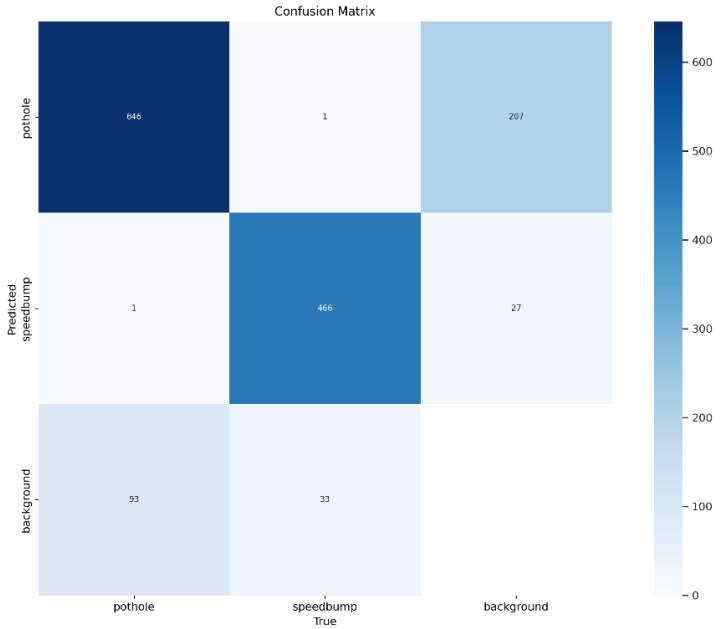

Figure 4.22 : matrice de confusion

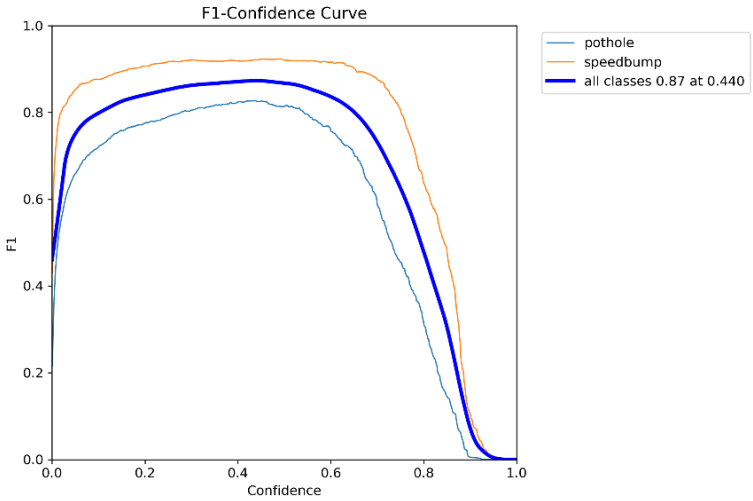

Figure 4.23 :F1 score

- Voici ce que nous constatons Lorsque nous entraînons les dernières couches du réseau de neurones YOLOv8 avec notre jeu de données (voir tableau 4.5) :

Tableau 4.5: perfomances des 2 classes : speedbump et pothole.

Performance	Précision	recall	mAP 50	mAP 50-90
all	0.89	0.856	0.895	0.53
pothole	0.847	0.807	0.851	0.417
speed bump	0.934	0.906	0.938	0.636

D'après les données du tableau ci-dessus, il est observé que le nombre d'images étiquetées comme nids-de-poule est supérieur à celui étiqueté comme ralentisseurs. Pourtant, le mAP des ralentisseurs est plus élevé que celui des nids-de-poule. Le mAP, une métrique utilisée pour évaluer la performance des modèles de détection d'objets, indique une meilleure précision et rappel du modèle pour une catégorie spécifique lorsqu'il est plus élevé. Ainsi, malgré un nombre inférieur d'images étiquetées, le modèle détecte mieux les ralentisseurs que les nids-de-poule.

4.3.4 Phase des tests :

Une fois l'entraînement terminé, nous procéderons à des tests sur des images qui n'ont jamais été utilisées dans le processus d'entraînement. Nous avons effectué des prédictions sur quelques images et présentons ci-dessous les résultats obtenus (comme le montre les figures 4.24, figure 4.25 et figure 4.26).

- Code utilisée pour la prédiction sur Colab

```
!yolo task=detect mode=predict
model=/content/drive/MyDrive/yolov8/training_results/pfe/weights/best.pt conf=0.50
source=/content/drive/MyDrive/yolov8/test_images
```

Figure 4.24 :images tests de détection des ralentisseurs non marqués et de peinture fanée.

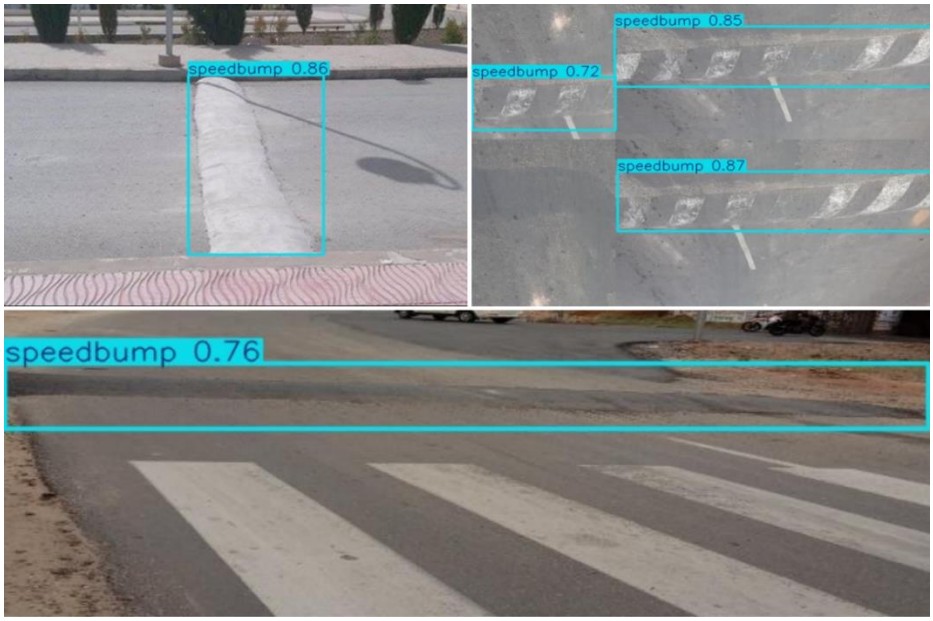

Figure 4.25:images tests de détection des ralentisseurs non marqués et de peinture fanée.

Figure 4.26 : images tests de détection des nids-de-poule remplis d'eau de pluie et secs.

Après avoir obtenu les résultats ci-dessus, nous avons effectué d'autres entraînements avec différentes variantes du modèle YOLOv8, telles que le modèle Medium et le modèle Large (voir tableau 4.6 et tableau 4.7). Les tableaux suivants montrent les performances de chaque variante :

- **YOLOv8L (large)** :

Tableau 4.6: performance de la variante Large du yolov8

performance	Précision	recall	mAP 50	mAP 50-90
all	0.869	0.833	0.876	0.576
pothole	0.83	0.748	0.81	0.389
speed bump	0.91	0.917	0.94	0.644

- **YOLOv8m (medium)** :

Tableau 4.7 : performance de la variante medium du yolov8

performance	Précision	recall	mAP 50	mAP 50-90
all	0.903	0.838	0.892	0.546
pothole	0.855	0.77	0.831	0.412
speed bump	0.95	0.906	0.954	0.68

- **Comparaison de performance au niveau de frames :**

La comparaison se concentrera sur le nombre d'images traitées par seconde (FPS) (voir tableau 4.8), étant donné que chaque version du modèle est entraînée avec des configurations distinctes en termes d'epochs et de taille de batch. Les données des FPS sont prises du tableau (Tableau 3.8 : STATE-OF-THE-ART YOLOv8 models)

Tableau 4.8 : Comparaison des performances des variantes au niveau de vitesse

paramètre / Variante	Epochs	Batchs	vitesse
nano	146	35	40 fps
medium	100	30	20 fps
large	70	20	15 fps

D'après les tableaux ci-dessus, il est notable que la variante medium se distingue par ses performances supérieures parmi les versions (nano et large). Toutefois, sa vitesse en images par seconde est notablement plus lente pour une utilisation en temps réel. YOLOv8n présente des

performances équilibrées en termes de précision et de mAP, avec une vitesse en images par seconde nettement supérieure à celle des autres variantes, ce qui en fait le choix optimal pour une utilisation en temps réel.

4.3.5 Test sur des vidéos streaming (simulation temps réels)

Dans cette section, nous avons évalué nos résultats en appliquant notre modèle à des vidéos en streaming à l'aide de PyCharm. Pour ce faire, nous avons organisé nos fichiers dans un répertoire nommé « compV », comprenant des scripts Python ainsi qu'un sous-dossier appelé « video » contenant les vidéos de test (voir figure 4.27).

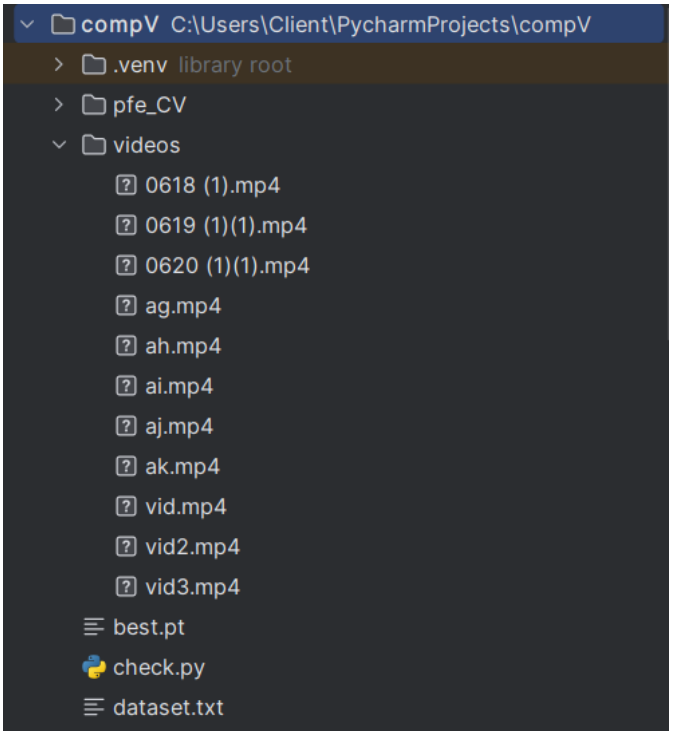

Figure 4. 27 : Fenêtre du projet avec les fichiers créés

L'illustration ci-dessous (figure 4.28) présente la bibliothèque utilisée pour effectuer les prédictions dans notre projet. Elle met en évidence les différents composants et modules intégrés, notamment `cvzone` et `YOLO` pour la détection d'objets, ainsi que `OpenCV` pour le traitement vidéo.

```
from ultralytics import YOLO
import cv2
import math
import pandas as pd
import time
import cvzone
```

Figure 4.28: Bibliothèque utilisée dans la prédiction

Notre code de prédiction est optimisé en traitant un frame sur trois (comme le montre la figure 4.29).

```python
def resize_video(input_path, output_path, new_width, new_height):
    cap = cv2.VideoCapture(input_path)
    if not cap.isOpened():
        print(f"Erreur: Impossible de lire la vidéo {input_path}.")
        return
    # Récupère le nombre de frames par seconde
    fps = cap.get(cv2.CAP_PROP_FPS)
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, (new_width, new_height))
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        resized_frame = cv2.resize(frame, dsize: (new_width, new_height), interpolation=cv2.INTER_AREA)
        out.write(resized_frame)
    cap.release()
    out.release()
    cv2.destroyAllWindows()
    print(f"Vidéo redimensionnée sauvegardée à {output_path}")
```

Figure 4.29 : code de prédiction (1)

```python
def process_video(input_path):
    # Initialisation de la capture vidéo (vidéo locale)
    cap = cv2.VideoCapture(input_path)
    if not cap.isOpened():
        print(f"Erreur: Impossible de lire la vidéo {input_path}.")
        return
    # Chargement du modèle YOLOv8
    model = YOLO("best.pt")
    # Liste des noms de classes du modèle pré-entraîné sur COCO
    classNames = ["pothole", "speedbump"]
    # Initialisation des variables pour le calcul des FPS
    prev_frame_time = 0
    # Compteur pour sauter des frames
    frame_counter = 0
    skip_frames = 3  # Par exemple, traiter une frame sur deux
    while True:
        # Capture d'une frame de la vidéo
        success, img = cap.read()
        if not success:
            print(f"Erreur: Impossible de lire la frame de la vidéo {input_path}.")
            break
        # Skipping frames
        frame_counter += 1
```

Figure 4.30 : code de prédiction (2)

```
        if frame_counter % skip_frames != 0:
            continue
        # Réduire la taille de l'image pour accélérer la prédiction
        img_resized = cv2.resize(img, dsize: (640, 480))
        # Effectuer la détection d'objets avec YOLOv8
        results = model(img_resized, stream=True)
        for r in results:
            boxes = r.boxes
            for box in boxes:
                x1, y1, x2, y2 = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                w, h = x2 - x1, y2 - y1
                # Dessiner le rectangle autour de l'objet détecté en jaune
                cv2.rectangle(img_resized, (x1, y1), (x2, y2), (0, 255, 255), 2)

                # Extraire la confiance et la classe
                conf = math.ceil((box.conf[0] * 100)) / 100
                cls = int(box.cls[0])
                if cls < len(classNames):
                    # Afficher le nom de la classe et la confiance
                    cv2.putText(img_resized, text: f'{classNames[cls]} {conf}', org: (max(0, x1), max(35, y1)),
                                cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.9, color: (0, 255, 255), thickness: 2)
```

Figure 4.31 : code de prédiction (3)

```
        # Calculer les FPS
        new_frame_time = time.time()
        fps = 1 / (new_frame_time - prev_frame_time)
        prev_frame_time = new_frame_time
        print(f"FPS: {fps:.2f}")
        # Afficher l'image avec les détections
        cv2.imshow( winname: "Vidéo", img_resized)
        # Réduire le délai d'attente pour augmenter la vitesse de lecture de la vidéo
        if cv2.waitKey(10) & 0xFF == ord('q'):  # Ajustez la valeur ici pour contrôler la vitesse de lecture
            break
    # Libérer les ressources
    cap.release()
    cv2.destroyAllWindows()
# Chemin vers la vidéo d'entrée
input_video_path = "videos/0618(1).mp4"
# Redimensionner la vidéo
resized_video_path = "videos/resized_video.mp4"
new_width, new_height = 640, 480  # Dimensions souhaitées
resize_video(input_video_path, resized_video_path, new_width, new_height)
# Traiter la vidéo redimensionnée
process_video(resized_video_path)
```

Figure 4.32 : code de prédiction (4)

Dans l’image ci-dessous, on peut observer la détection d'un ralentisseur (speedbump) avec une précision de 67% (comme illustré dans la figure 4.33).

L'objet détecté est encadré par un rectangle jaune qui met en évidence la zone identifiée, et le pourcentage de confiance est indiqué à côté de l'objet détecté.

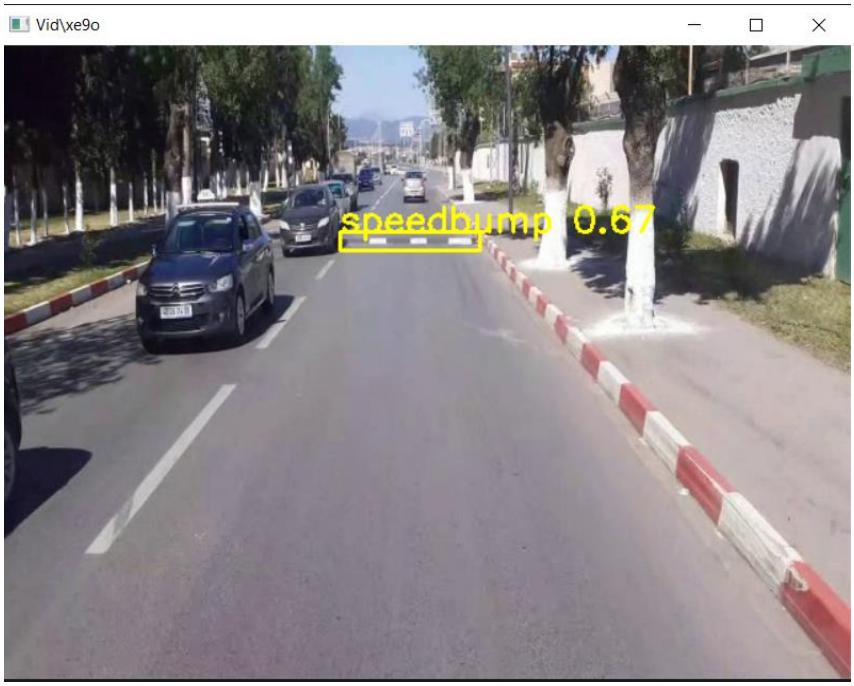

Figure 4.33 : Détection de ralentisseur avec une confiance de 67%

```
0: 480x640 2 speedbumps, 166.4ms
Speed: 2.1ms preprocess, 166.4ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
FPS: 5.40

0: 480x640 1 speedbump, 173.8ms
Speed: 2.0ms preprocess, 173.8ms inference, 1.1ms postprocess per image at shape (1, 3, 480, 640)
FPS: 5.25

0: 480x640 1 speedbump, 190.3ms
Speed: 3.0ms preprocess, 190.3ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
FPS: 4.54

0: 480x640 1 speedbump, 185.6ms
Speed: 1.0ms preprocess, 185.6ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
FPS: 4.73

0: 480x640 1 speedbump, 208.7ms
Speed: 2.0ms preprocess, 208.7ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
FPS: 4.37
```

Figure 4.34 : Affichage des performances de la vidéo sur le terminal, montrant les FPS et les autres statistiques de traitement en temps réel.

Dans la deuxième vidéo en streaming, les nids-de-poule sont détectés et 2 instances de nids-de-poule sont identifiées à une cadence variant entre 3 et 6 images par seconde (FPS) (voir figure 4.35).

Figure 4.35 : deux nids-de-poule encadrés avec une confiance de 54% et 32%

4.4 Implémentation du système sur raspberry pi

4.4.1 Historique

Le nom "Raspberry Pi" combine le mot anglais pour framboise (fruit souvent utilisé dans les noms de marques technologiques) avec "Pi" faisant référence à "Python interpreter", un langage de programmation essentiel. Les fondateurs, incluant David Braben, Jack Lang, Pete Lomas, Alan Mycroft, Robert Mullins et Eben Upton, ont lancé ce projet pour résoudre le déclin des candidats en informatique et le coût élevé des ordinateurs [75]. En 2009, la Fondation Raspberry Pi a été créée pour rendre l'informatique et la création numérique accessibles à tous, notamment en aidant les jeunes à apprendre la programmation à moindre coût [76]. L'idée a donné naissance au Raspberry Pi, lancé officiellement le 29 février 2012 avec l'ambition initiale de vendre 10 000 unités, un objectif largement dépassé puisque plus de 6 millions d'unités ont été vendues dans le monde.

4.4.2 Définition

Le Raspberry Pi est un ordinateur mono carte à processeur ARM, conçu par des enseignants de l'université de Cambridge. Il est vendu sous forme de carte mère nue, nécessitant l'ajout de périphériques tels que clavier, souris, câble d'alimentation et écran

pour fonctionner comme un mini-ordinateur personnel. Programmable principalement en Python, il offre des broches GPIO pour connecter des extensions et d'autres composants électroniques pour des projets variés (voir figure 4.36).

Polyvalent et abordable, le Raspberry Pi dépasse largement ses attentes initiales de popularité. Il est utilisé dans l'éducation, les projets DIY comme la domotique et les médiacenters, ainsi que pour le développement de prototypes IoT, serveurs légers, jeux rétro, recherche scientifique, automatisation industrielle, traitement d'images/vidéos en temps réel, applications IoT et robotiques [75].

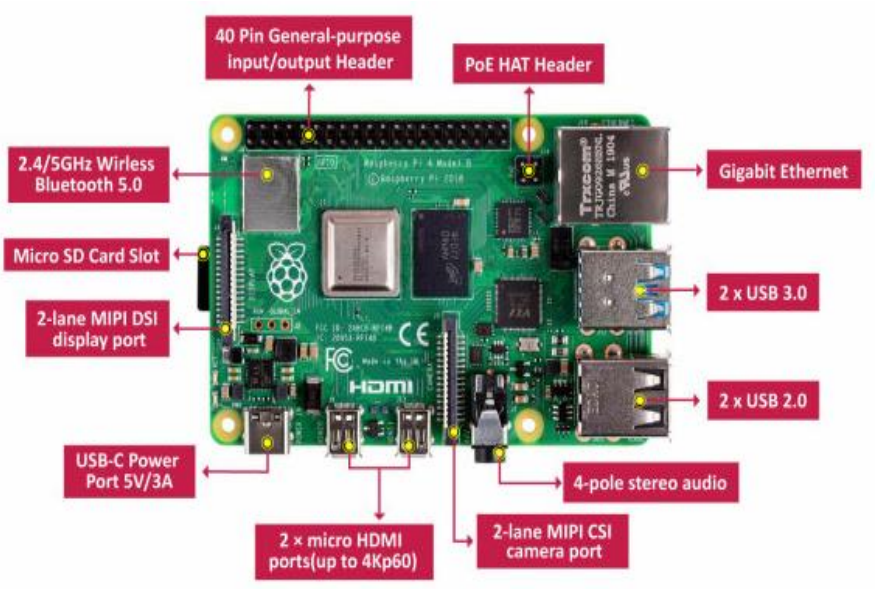

Figure 4.36 : Carte Raspberry Pi 4 model B [77]

4.4.3 Modèles carte raspberry pi

Voici les différents modèles de cartes Raspberry Pi disponibles [79] :

- Raspberry Pi 5 : modèle 4 Go, modèle 8 Go
- Raspberry Pi 4 : modèle B 1 Go, modèle B 2 Go, modèle B 4 Go, modèle B 8 Go
- Raspberry Pi 3 : modèle B+, modèle A+, modèle B
- Raspberry Pi 2 : modèle
- Raspberry Pi 1 : modèle B+
- Raspberry Pi 1 : modèle A+
- Raspberry Pi Zéro : version 1.3
- Raspberry Pi Zero W
- Raspberry Pi Zero WH

4.4.4 Exploration du raspberry Pi

4.4.4.1 Processeur paspberry pi

Le Raspberry Pi est équipé d'un SoC (System-on-Chip) Broadcom à processeur ARM, incluant un GPU intégré pour le traitement graphique. Sa vitesse de processeur varie entre 700 MHz et 1,2 GHz, et il est doté de SDRAM intégrée allant de 256 Mo à 1 Go. De plus, il offre des modules intégrés SPI, I2C, I2S et UART pour une connectivité étendue avec d'autres composants électroniques. [78]

4.4.4.2 Système d'exploitation pour raspberry pi

La Raspberry Pi Foundation propose le système d'exploitation Raspbian, dérivé de Debian, ainsi que l'outil d'installation NOOBS pour les Raspberry Pi. En plus des options officielles, il est possible d'installer d'autres systèmes d'exploitation tiers comme Ubuntu, Archlinux, RISC OS et Windows 10 IoT Core.

Raspbian, spécialement optimisé pour Raspberry Pi, offre une interface graphique complète avec des outils pour la navigation, la programmation en Python, la productivité bureautique et les jeux. Pour fonctionner, il est recommandé d'utiliser une carte SD d'au moins 8 Go pour stocker le système d'exploitation Linux et ses logiciels associés, compatibles avec diverses distributions GNU/Linux.

Une fois le système d'exploitation installé, le Raspberry Pi peut être connecté à des périphériques de sortie tels que des moniteurs ou des téléviseurs via une interface HDMI, permettant une utilisation flexible et polyvalente dans de nombreux projets. [78]

4.4.4.3 A propos de la carte raspberry pi 3 modèle B :

Pour ce projet, nous avons opté pour la carte Raspberry Pi 3 Modèle B (voir figure 4.37). Voici ses principales caractéristiques : [80]

Tableau 4.5 : Carte Raspberry Pi 3 Modèle B

Caractéristique	Détail
Date de sortie	2016
Cadencement	1,2 GHz
Puce (SoC)	Broadcom BCM2837
Processeur	ARM Cortex-A53 64 bits quatre cœurs
Processeur graphique	Broadcom VideoCore IV double coeur (OpenGL ES 2.0, H.264 Full HD à 30 ips)
Mémoire (SDRAM)	1GB LPDDR2
Nombre de ports USB 2.0	4
Port extension	GPIO 40 pin
Sorties vidéos	HDMI et RCA, plus 1 connecteur de caméra CSI, Interface d'affichage DSI
LED d'alimentation	une LED ROUGE, connecté à 5V directement et commencera à clignoter chaque fois que la tension d'alimentation descend en dessous de 4,63 V.
ACT PWR	ACT PWR est une LED verte qui montre l'activité de la carte SD.
Sorties audio	Stéréo Jack 3,5mm ou HDMI
stockage	Carte MicroSD
Connexion réseau	10/100 Ethernet, WiFi 802.11n et Bluetooth 4.1 (BLE - Low Energy)
Alimentation	5v 2.5A via micro-USB
Dimensions et poids	85,60 mm × 53,98 mm × 17 mm avec un poids de 45 g

Malgré ses ressources limitées, l'exploration des capacités de la carte Raspberry Pi 3 demeure cruciale. Elle permet d'étendre la vision par ordinateur aux environnements embarqués et à l'IoT, enrichissant ainsi notre compréhension visuelle grâce à cette technologie

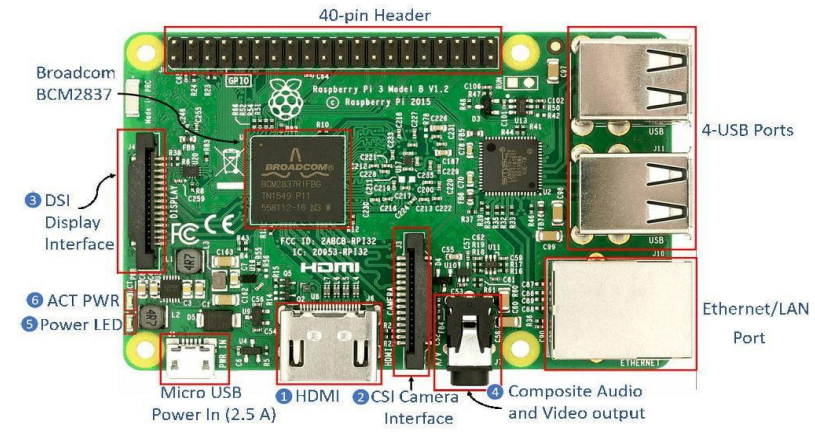

Figure 4.37 : Carte Raspberry Pi 3 model B [78]

4.4.5 Configuration de la carte raspberry pi

Voici les étapes suivies pour configurer cette carte :

1) Connectez un clavier, une souris, et un écran à la carte via un câble HDMI. Assurez-vous d'avoir également une source d'alimentation.
2) Téléchargez et installez Raspberry Pi OS à l'aide de Raspberry Pi Imager, disponible sur le site officiel : raspberrypi.com.
3) Sélectionnez le modèle de la carte Raspberry Pi (3 modèle B), le système d'exploitation (OS), et préparez la carte mémoire (comme le montre la figure 4.38).

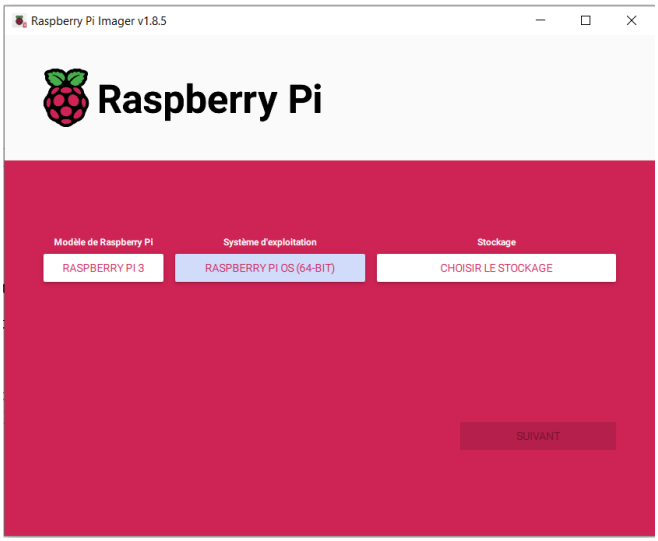

Figure 4.38 : Interface principale de Raspberry Pi Imager.

4) Une fois ces étapes réalisées, nous procédons à la configuration du serveur SSH, du Wi-Fi, du mot de passe et du serveur VNC (comme illustré dans la figure 4.39).

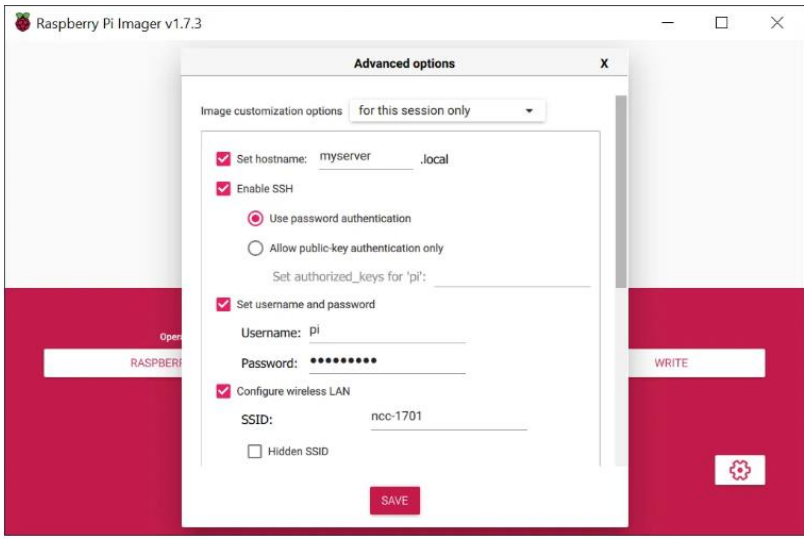

Figure 4.39 : Interface des paramètres dans Raspberry Pi Imager

5) Une fois l'OS installé sur la carte SD, nous pourrons y accéder en utilisant la carte Raspberry Pi.
6) Installez Advanced IP Scanner : Advanced IP Scanner permet de trouver l'adresse IP des appareils connectés au même réseau internet (voir figure 4.40).

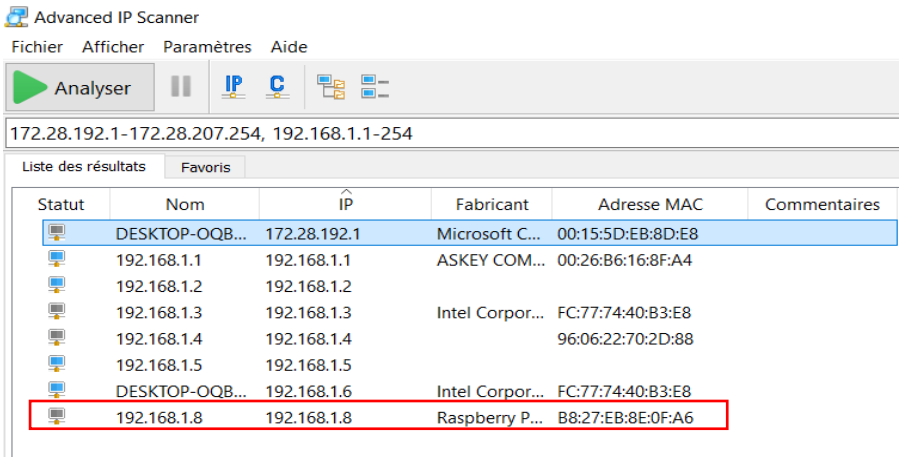

Figure 4.40 : Interface d'Advanced IP Scanner affichant les adresses IP des appareils connectés

7) Installez PuTTY (64-bit) et procédez à la configuration avec l'adresse IP de Raspberry Pi : 192.168.1.8 (voir figure 4.41) : PuTTY est un logiciel qui permet de se connecter à distance à des ordinateurs via un réseau. Il est principalement utilisé pour exécuter des commandes et gérer des systèmes à distance de manière sécurisée grâce au protocole SSH (Secure Shell))

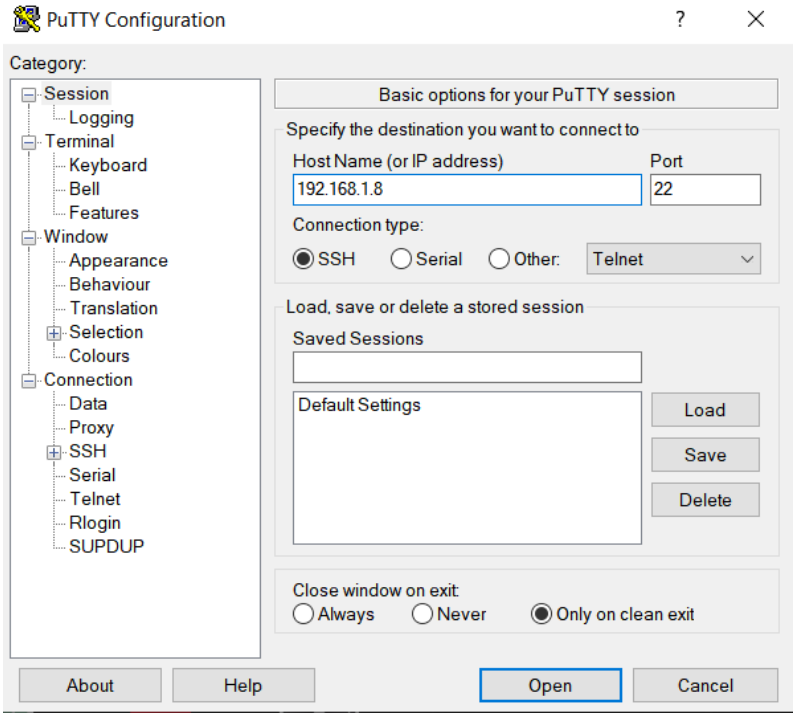

Figure 4.41 : Interface de configuration de PuTTY

8) Installez Real VNC Viewer (voir figure 4.42).
   - Real VNC Viewer permet d'accéder à distance à l'interface graphique d'un ordinateur ou d'un appareil comme le Raspberry Pi via le protocole VNC (Virtual Network Computing).
   - SSH est un protocole sécurisé pour se connecter à distance à des machines informatiques, assurant une communication cryptée et sécurisée.

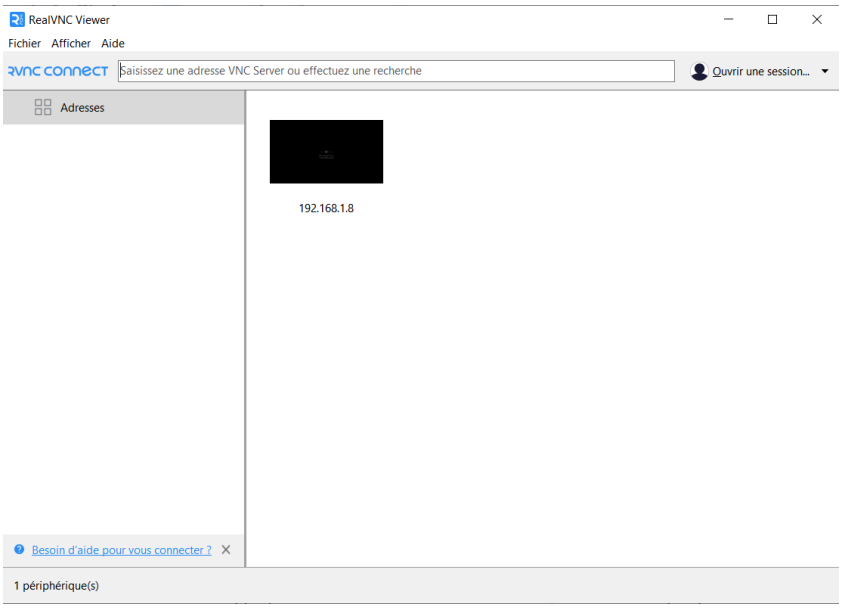

Figure 4.42 : Interface de VNC Viewer

9) Dans l'environnement de développement intégré (IDE) Thonny Python, une partie du code est implémentée (comme le montre la figure (4.43).

   Thonny offre une interface conviviale pour développer et déboguer des programmes Python, particulièrement adaptée aux applications embarquées telles que celles sur Raspberry Pi. Cet IDE permet de surveiller en temps réel les performances vidéo, y compris les FPS (images par seconde) et d'autres statistiques de traitement. Cette fonctionnalité facilite le développement et l'optimisation des algorithmes de détection d'objets comme les dos d'âne.

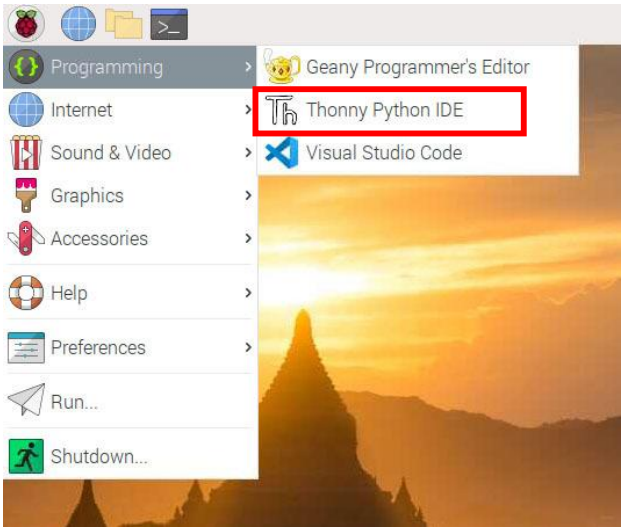

Figure 4.43 : Fenêtre environnement de développement Thonny [45]

10) Nous avons utilisé la commande suivante dans le terminal pour installer les bibliothèques nécessaires : "sudo pip3 install <nom de la bibliothèque>".

4.4.6 Implémentation du code sur raspberry pi

Nous avons utilisé le même code mentionné précédemment et les vidéos de prédictions sont désormais sur le bureau (le résultat est illustré dans la figure 4.44). Cependant, nous avons remarqué une baisse du FPS par rapport aux tests effectués sur le PC (comme le montre la figure 4.45). Cette diminution de performance peut être due aux différences matérielles entre la carte Raspberry Pi et le PC portable.

En raison de ses ressources limitées en termes de RAM et de puissance de calcul, le Raspberry Pi pourrait ne pas être capable de charger et d'exécuter le modèle aussi efficacement que le PC. Cela peut entraîner des ralentissements, des temps de traitement plus longs voire une

incapacité à exécuter le modèle si celui-ci est trop complexe pour les capacités du Raspberry Pi.

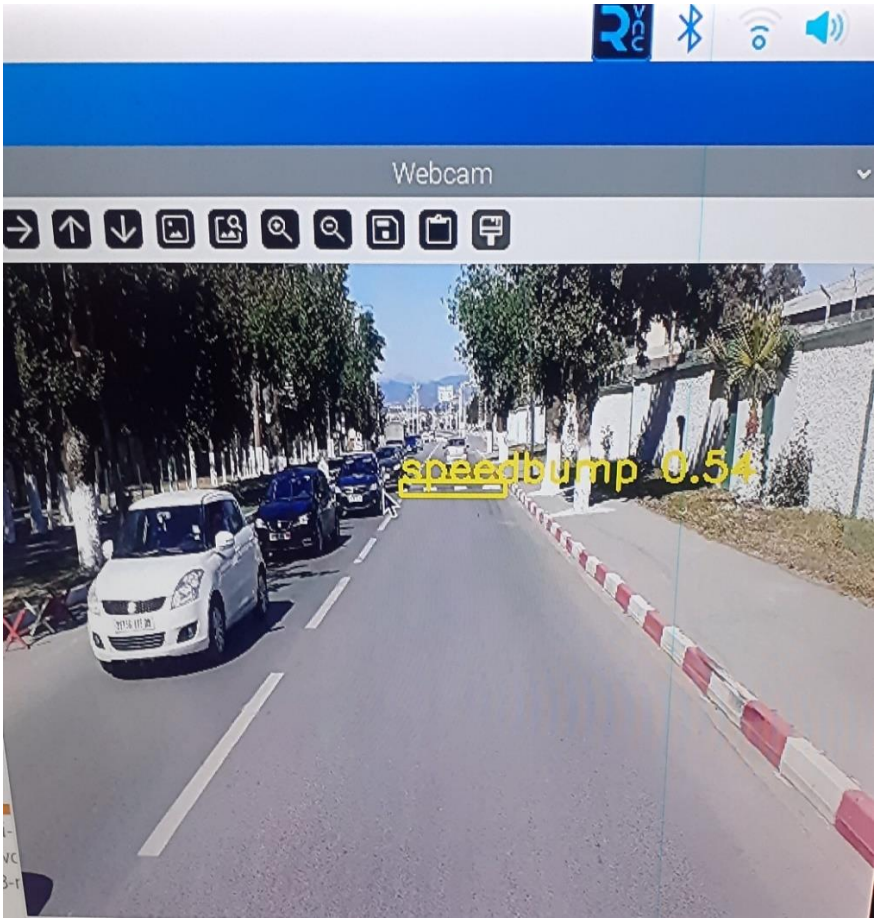

Figure 4.44 : Détection de dos d'âne sur Raspberry Pi

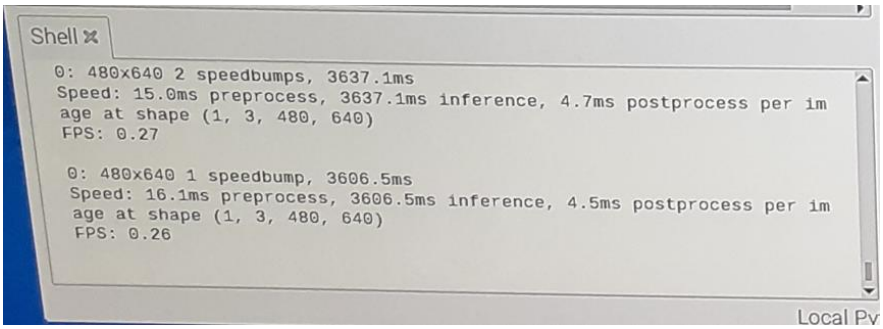

Figure 4.45 : Affichage des performances de la vidéo sur le terminal de Thonny Python IDE, présentant les FPS et d'autres statistiques de traitement.

4.5 Conclusion

Dans ce chapitre, nous avons détaillé les étapes suivies pour la réalisation de notre projet. Nos résultats montrent que notre système est capable de résoudre les problèmes liés aux dommages causés aux véhicules et de réduire les accidents. Nous avons atteint de bons résultats en utilisant le modèle YOLO v8 et ses variantes pour identifier et détecter efficacement les dos d'âne et les nids de poule, avec une précision élevée. De plus, nous avons opté pour la variante Nano pour le streaming afin de simuler un environnement proche du temps réel.

Cependant, des ajustements supplémentaires des paramètres sont nécessaires pour améliorer encore les performances du système. De plus, l'accès à des ressources GPU serait bénéfique pour accélérer le processus d'apprentissage et optimiser les opérations.

CONCLUSION GÉNÉRALE

Dans ce projet, nous avons développé un système de détection des ralentisseurs et des défauts de la chaussée routière en utilisant la vision par ordinateur.
Le projet était structuré en deux phases principales : la première consistait à entraîner un modèle pour détecter les dos d'âne et les nids-de-poule, tandis que la seconde phase se concentrait sur la prédiction en temps réel via un flux vidéo, implémentée sur un Raspberry Pi 3.

Nous avons choisi le modèle YOLO v8n et configuré notre dataset avec deux classes. Le modèle a été entraîné avec des paramètres spécifiques tels que 146 epochs et 35 batchs, sur des images de 640 pixels. Nous avons exploré plusieurs variantes du modèle YOLO v8, à savoir Nano, Medium et Large, toutes offrant des performances satisfaisantes avec une précision élevée pour les classes "pothole" et "speed bump".

Après analyse des résultats, nous avons déterminé que YOLO v8n était la meilleure option pour la prédiction en temps réel. Nous l'avons intégré à l'environnement de développement intégré PyCharm IDE en utilisant les poids optimaux issus de l'entraînement sur des vidéos en streaming.

Cette approche nous a permis d'obtenir des prédictions précises et rapides directement dans l'IDE, optimisant ainsi l'efficacité opérationnelle. Nous avons également porté notre code sur un Raspberry Pi, réalisant ainsi la détection en streaming vidéo.
Ce processus incluait l'adaptation du modèle YOLO v8n pour fonctionner efficacement sur le Raspberry Pi, en tenant compte de ses limitations matérielles. Nous avons optimisé notre implémentation pour garantir une détection en temps réel des défauts de chaussée avec une faible latence et une haute précision.

Ce projet nous a ouvert de nouvelles perspectives dans le domaine des systèmes ADAS et de la vision par ordinateur, notamment en matière de détection d'obstacles. Nous avons acquis des connaissances approfondies en intelligence artificielle et en deep learning, ainsi qu'une compréhension pratique des technologies applicables dans ce contexte.

Nous avons également surmonté plusieurs défis, notamment l'optimisation de la base de données et la gestion des ressources GPU limitées sur Google Colab pour l'entraînement du modèle YOLO. Malgré ces défis, notre approche a abouti à un système robuste capable de détecter efficacement les défauts de chaussée en conditions réelles.

Pour l'avenir, nous recommandons l'amélioration continue du système en utilisant des caméras panoramiques, en optimisant la qualité des données d'entraînement, et en augmentant les

performances du modèle grâce à des ressources matérielles plus puissantes comme les GPU. L'intégration de technologies avancées telles que le LIDAR ou les caméras infrarouges pourrait également renforcer les capacités du système.

En conclusion, ce projet vise à contribuer à l'amélioration continue des technologies de sécurité routière, en intégrant des innovations pour rendre la conduite plus sûre et confortable pour tous les usagers de la route.

Références bibliographiques :

[1]Lindeke B, Lindeke B. Four reasons Minnesota's roads are so riddled with potholes, and what we could be doing about it [Internet]. MinnPost. 2024. Disponible sur : https://www.minnpost.com/cityscape/2023/03/four-reasons-minnesotas-roads-are-so-riddled-with-potholes-and-what-we-could-be-doing-about-it/ Consulté le : 10/04/2024

[2]RALENTISSEUR DE VITESSE [Internet]. Cj Créations. Disponible sur : https://www.cj-creations.fr/securite/1675-element-ralentisseur-jaune-5cm-v.html Consulté le : 12/04/2024

[3]Daphnée. The complete list of speed bump types [Internet]. Sino Concept. 2024. Disponible sur : https://www.sinoconcept.co.uk/purchasing-guides/everything-about-speed-bumps/list-speed-bump-types/ Consulté le : 18/04/2024

[4]Bizouard F. Les différents types de ralentisseurs - Sino Concept Fabricant [Internet]. Sino Concept. 2024. Disponible sur : https://www.sinoconcept.fr/guides-achat/guide-ralentisseur-routier/differents-types-ralentisseurs-routiers/ Consulté le : 10/04/2024

[5]Daphnée. What are circular speed bumps ? [Internet]. Sino Concept. 2024. Disponible sur : https://www.sinoconcept.co.uk/purchasing-guides/everything-about-speed-bumps/list-speed-bump-types/circular-speed-bumps/ Consulté le : 12/04/2024

[6]Smith, J. (2020). "State of the Art in Road Surface Inspection: A Review of Traditional Methods and Challenges." Transportation Research Part B: Methodological, 105, 82-95.Consulté le : 12/04/2024

[7]potholes | Roboflow Universe Search [Internet]. Roboflow. Disponible sur : https://universe.roboflow.com/search?q=potholes Consulté le : 13/04/2024

[8]Ashraf MT, Dey K, Mishra S. Identification of high-risk roadway segments for wrong-way driving crash using rare event modeling and data augmentation techniques. Accident Analysis And Prevention [Internet]. 1 mars 2023 ; 181 : 106933. Disponible sur : https://www.sciencedirect.com/science/article/abs/pii/S0001457522003682Consulté le : 13/04/2024

[9] Ma. Les normes d& # 39 ; implantation des dos d& # 39 ; âne et des ralentisseurs [Internet]. 40 Millions D'automobilistes. 2023. Disponible sur : https://www.40millionsdautomobilistes.com/fiche-pratique/les-normes-d-implantation-des-dos-d-ane-et-des-ralentisseurs. Consulté le : 18/04/2024

[10] : Kyriakou C, Christodoulou SE, Dimitriou L. Spatial Roadway Condition-Assessment Mapping Utilizing Smartphones and Machine Learning Algorithms. Transportation Research Record [Internet]. 9 avr 2021 ; 2675(9) : 1118-26. Disponible sur : https://doi.org/10.1177/03611981211006105

[11]Coussin berlinois en caoutchouc recycl - Virages [Internet]. Disponible sur : https://www.virages.com/Ralentisseurs-de-Vitesse/Coussins-Berlinois

[12]Ralentisseur de vitesse pour voie privée | Ralentisseur en élastomère [Internet]. Sol Direct. Disponible sur : https://sol-direct.fr/ralentisseurs-routiers/ralentisseur-routier-pour-voies-privees-p-435-2778.html . Consulté le : 12/04/2024

[13] PLATEAU RALENTISSEUR SURÉLEVÉ [Internet]. Cj Créations. Disponible sur : https://www.cj-creations.fr/securite/1674-plateau-ralentisseur-sureleve-nouveau-modele-au-m.html . Consulté le : 17/04/2024

[14]Giorgi JP. Sécurité : Les ralentisseurs de vitesse # 4 - Cyclotourisme Mag [Internet]. Cyclotourisme Mag. 2020. Disponible sur : https://cyclotourisme-mag.com/conseil/securite-les-ralentisseurs-de-vitesse-4/. Consulté le : 12/04/2024

[15]Aurore L. Systèmes ADAS : définition, avantages et formation [Internet]. SNECI. 2022. Disponible sur : https://www.sneci.com/fr/blog/systemes-adas-definition-avantages-et-formation/. Consulté le : 13/04/2024

[16]Valeo. Aide à la conduite | Système ADAS | Valeo [Internet]. Valeo. 2024. Disponible sur : https://www.valeo.com/fr/aides-a-la-conduite/ . Consulté le : 09/04/2024

[17]Charland M. Ce qu& # 8217 ; il faut savoir sur les systèmes ADAS pour les conseillers aux opérations [Internet]. Cours de Mecanique Automobile Montreal. 2024. Disponible sur : https://www.ecoleauto.com/blog/ce-quil-faut-savoir-sur-les-systemes-adas-pour-les-conseillers-aux-operations/. Consulté le : 12/04/2024

[18]Jiménez F, Naranjo JE, Anaya JJ, García F, Ponz A, Armingol JM. Advanced Driver Assistance System for Road Environments to Improve Safety and Efficiency. Transportation Research Procedia [Internet]. 1 janv 2016 ; 14 : 2245-54. Disponible sur : https://www.sciencedirect.com/science/article/pii/S2352146516302460. Consulté le : 1/04/2024

[19] Zhu Y, Yan WQ. Traffic sign recognition based on deep learning. Multimedia Tools And Applications [Internet]. 7 mars 2022 ; 81(13) : 17779-91. Disponible sur : https://doi.org/10.1007/s11042-022-12163-0 . Consulté le : 18/04/2024

[20]GeeksforGeeks. Artificial Intelligence An introduction [Internet]. GeeksforGeeks. 2024. Disponible sur : https://www.geeksforgeeks.org/artificial-intelligence-an-introduction/?ref=gcse

[21]https://www.netapp.com/fr/artificial-intelligence/what-is-artificial-intelligence/. Consulté le : 12/04/2024

[22]Sansonetti J. Histoire de l& # 39 ; intelligence artificielle (IA) : création et évolutions [1943 à 2024] [Internet]. 2022. Disponible sur : https://www.wizishop.fr/blog/histoire-intelligence-artificielle. Consulté le : 19/04/2024

[23]Neovision. L'IA c'est quoi ? Découvrez la réponse de nos experts [Internet]. Neovision. 2024. Disponible sur : https://neovision.fr/definition-intelligence-artificielle/. Consulté le : 16/04/2024

[24]Julien. Quelles différences entre intelligence artificielle et machine learning ? [Internet]. Ryax Technologies. 2021. Disponible sur : https://ryax.tech/fr/differences-intelligence-artificielle-machine-learning/. Consulté le : 09/04/2024

[25]Machine learning, qu'est-ce que c'est ? - IA School [Internet]. IA School. 2023. Disponible sur : https://www.intelligence-artificielle-school.com/ecole/technologies/machine-learning-quest-ce-que. Consulté le : 13/05/2024

[26]Deep Learning vs Machine Learning : The Ultimate Battle [Internet]. 2022. Disponible sur : https://www.turing.com/kb/ultimate-battle-between-deep-learning-and-machine-learning

[27] Appréhendez le Deep Learning ou l'apprentissage profond [Internet]. OpenClassrooms. Disponible sur : https://openclassrooms.com/fr/courses/6417031-objectif-ia-initiez-vous-a-lintelligence-artificielle/6823506-apprehendez-le-deep-learning-ou-lapprentissage-profond Consulté le : 8/05/2024

[29] Christopher A. What is Perceptron : A BeginnersTutorial For Perceptron. Medium [Internet]. 6 janv 2022 ; Disponible sur : https://antoblog.medium.com/what-is-perceptron-a-beginnerstutorial-for-perceptron-632539884146 .Consulté le : 13/05/2024

[30] Bento C. Multilayer Perceptron Explained with a Real-Life Example and Python Code : Sentiment Analysis. Medium [Internet]. 5 janv 2022 ; Disponible sur : https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141. Consulté le : 7/05/2024

[31] Lunge N. A deep architecture : Multi-Layer Perceptron - Ninad Lunge - Medium. Medium [Internet]. 24 mars 2024 ; Disponible sur : https://medium.com/@nlunge786/a-deep-architecture-multi-layer-perceptron-164bc5ff3842 Consulté le : 23/04/2024

[32] GeeksforGeeks. Understanding Activation Functions in Depth [Internet]. GeeksforGeeks. 2022. Disponible sur : https://www.geeksforgeeks.org/understanding-activation-functions-in-depth/ Consulté le : 27/05/2024

[33] Pandey B. How Do Neural Networks Make Decisions ? A Look at Activation Functions [Internet]. Goglides Dev 🌱. 2023. Disponible sur : https://www.goglides.dev/bkpandey/how-do-neural-networks-make-decisions-a-look-at-activation-functions-141e . Consulté le : 5/05/2024

[34] J LC. A new activation function for Neural Networks - logmoid [Internet]. 2022. Disponible sur : https://www.linkedin.com/pulse/activation-functions-neural-networks-leonardo-calderon-j-/ Consulté le : 13/05/2024

[35] Banoula M. What is Cost Function in Machine Learning [Internet]. Simplilearn.com. 2023. Disponible sur : https://www.simplilearn.com/tutorials/machine-learning-tutorial/cost-function-in-machine-learning Consulté le : 25/05/2024

[36] Lunge N. A deep architecture : Multi-Layer Perceptron - Ninad Lunge - Medium. Medium [Internet]. 24 mars 2024 ; Disponible sur : https://medium.com/@nlunge786/a-deep-architecture-multi-layer-perceptron-164bc5ff3842.Consulté le : 7/05/2024

[37] Sharma P. Basic Introduction to Loss Functions [Internet]. Analytics Vidhya. 2023. Disponible sur : https://www.analyticsvidhya.com/blog/2022/08/basic-introduction-to-loss-functions/

[38] What is Forward Propagation ? | H2O.ai [Internet]. Disponible sur : https://h2o.ai/wiki/forward-propagation/#:~:text=Backward%20Propagation%20is%20the%20process,of%20connected%20input%2Foutput%20nodes. Consulté le : 19/05/2024

[39] Villard P. Deep learning [Internet]. Disponible sur : https://seshat.gitlabpages.inria.fr/deeplearning/docHtml/IA1.html Consulté le : 8/05/2024

[40] GeeksforGeeks. Convolutional Neural Network (CNN) in Machine Learning [Internet]. GeeksforGeeks. 2024. Disponible sur : https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/ Consulté le : 20/05/2024

[41]Coursesteach. Deep Learning (Part 34)-Forward and Backward Propagation. Medium [Internet]. 13 avr 2024 ; Disponible sur : https://medium.com/@Coursesteach/deep-learning-part-34-forward-and-backward-propagation-e91892b6dfef .Consulté le : 16/05/2024

[42] Biswal A. Convolutional Neural Network Tutorial [Internet]. Simplilearn.com. 2023. Disponible sur : https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network .Consulté le : 16/05/2024

[43] Kromydas B, Kromydas B. Convolutional Neural Network : A complete guide [Internet]. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Code, & Tutorials. 2024. Disponible sur : https://learnopencv.com/understanding-convolutional-neural-networks-cnn/. Consulté le : 13/06/2024

[44] Mandal M. Introduction to Convolutional Neural Networks (CNN) [Internet]. Analytics Vidhya. 2024. Disponible sur : https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ .Consulté le : 23/05/2024

[46]GeeksforGeeks. Introduction to Convolution Neural Network [Internet]. GeeksforGeeks. 2024. Disponible sur : https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[47]https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[48]Biswal A. Convolutional Neural Network Tutorial [Internet]. Simplilearn.com. 2023. Disponible sur : https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network .Consulté le : 16/05/2024

[49] Artificial Intelligence (AI) utilizing deep learning techniques to enhance ADAS [Internet]. Design And Reuse. Disponible sur : https://www.design-reuse.com/articles/54914/artificial-intelligence-ai-deep-learning-adas.html#:~:text=ADAS%20relies%20heavily%20on%20deep,are%20examples%20of%20these%20sensors. .Consulté le : 26/05/2024

[50]https://www.way.com/fr/blog/revolutionizing-car-services-the-impact-of-ai-and-ml/.Consulté le : 18/05/2024

[51] LePont. LePont : formations en data science & IA - Le Pont [Internet]. LePont. 2024. Disponible sur : https://www.lepont-learning.com/fr/computer%20vision/#:~:text=La%20vision%20par%20ordinateur%2C%20ou,le%20font%20les%20%C3%AAtres%20humains. .Consulté le : 23/05/2024

[52] Ark A. YOLO : Algorithm for Object Detection. - Arulkumar ARK - Medium. Medium [Internet]. 12 août 2023 ; Disponible sur : https://medium.com/@arulkumarark1924/yolo-algorithm-for-object-detection-2925453013e .Consulté le : 13/05/2024

[53] Marie-Laurence. Computer vision [Internet]. LePont. 2024. Disponible sur : https://www.lepont-learning.com/fr/computer-vision/#:~:text=La%20vision%20par%20ordinateur%2C%20ou,le%20font%20les%20%C3%AAtres%20humains .Consulté le : 24/04/2024

[54]Boesch G. A Guide to YOLOv8 in 2024 [Internet]. viso.ai. 2024. Disponible sur : https://viso.ai/deep-learning/yolov8-guide .Consulté le : 19/05/2024

[55] YOLO Algorithm : Real-Time Object Detection from A to Z [Internet]. Kili-website. Disponible sur : https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z#what-is-yolo? .Consulté le : 28/05/2024

[56] Belaidi N. YOLO : détection sur les images avec TensorFlow [Internet]. Formation Tech et Data en ligne | Blent.ai. Disponible sur : https://blent.ai/blog/a/detection-images-yolo-tensorflow .Consulté le : 23/05/2024

[57] GeeksforGeeks. YOLO you only look once Real time object detection [Internet]. GeeksforGeeks. 2022. Disponible sur : https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/ .Consulté le : 26/05/2024

[58] Boesch G. A Guide to YOLOv8 in 2024 [Internet]. viso.ai. 2024. Disponible sur : https://viso.ai/deep-learning/yolov8-guide/ .Consulté le : 03/06/2024

[59] Zambare S. Object detection : using non-max supression over YOLOv2. Medium [Internet]. 8 déc 2021 ; Disponible sur : https://medium.com/@sarangz/object-detection-using-non-max-supression-over-yolov2-382a90212b51.Consulté le : 03/06/2024

[60]Ultralytics. GitHub - ultralytics/ultralytics : NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite [Internet]. GitHub. Disponible sur : https://github.com/ultralytics/ultralytics

[61] Abdullah M. YOLO Working principle, difference between its ddifferent Variants and Versions. Medium [Internet]. 17 oct 2023 ; Disponible sur : https://medium.com/@muhabd51/yolo-working-principle-difference-between-its-ddifferent-variants-and-versions-95b8ad7b95ab .Consulté le : 05/06/2024

[62] Toward Accurate Fused Deposition Modeling 3D Printer Fault Detection Using Improved YOLOv8 With Hyperparameter Optimization - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-improved-YOLOv8-network-architecture-includes-an-additional-module-for-the-head_fig2_372207753 .Consulté le : 05/06/2024

[63] Pedro J. Detailed Explanation of YOLOv8 Architecture — Part 1. Medium [Internet]. 11 déc 2023 ; Disponible sur : https://medium.com/@juanpedro.bc22/detailed-explanation-of-yolov8-architecture-part-1-6da9296b954e .Consulté le : 03/06/2024

[64] YOLO : un véritable détecteur d'objets en temps réel [Internet]. Disponible sur : https://www.innovatiana.com/post/what-is-yolo-in-ai .Consulté le : 04/06/2024

[65] Spodarets D. A Guide to the YOLO Family of Computer Vision Models [Internet]. Data Phoenix. 2024. Disponible sur : https://dataphoenix.info/a-guide-to-the-yolo-family-of-computer-vision-models/ .Consulté le : 13/06/2024

[66]Rath S, Rath S. YOLOv8 : Comprehensive Guide to State of the Art Object Detection [Internet]. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Code, & Tutorials. 2024. Disponible sur : https://learnopencv.com/ultralytics-yolov8/.Consulté le : 03/06/2024

[67]https://fastercapital.com/fr/contenu/Precision-dans-la-reconnaissance-d-images---avancees-dans-la-detection-d-objets.html .Consulté le : 09/06/2024

[68]Fig. 2. The architecture of Faster R-CNN. [Internet]. ResearchGate. Disponible sur : https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN_fig2_324903264

[69] Potrimba P. What is Mask R-CNN ? The Ultimate Guide. [Internet]. Roboflow Blog. 2024. Disponible sur : https://blog.roboflow.com/mask-rcnn/ .Consulté le : 03/06/2024

[70] Hui J. SSD object detection : Single Shot MultiBox Detector for real-time processing. Medium [Internet]. 15 déc 2020 ; Disponible sur : https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06 .Consulté le : 03/06/2024

[71] YOLO Algorithm : Real-Time Object Detection from A to Z [Internet]. Kili-website. Disponible sur : https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z#what-is-yolo? .Consulté le : 03/06/2024

[72]Boesch G. What is Intersection over Union (IoU) ? [Internet]. viso.ai. 2024. Disponible sur : https://viso.ai/computer-vision/intersection-over-union-iou/ .Consulté le : 03/06/2024

[73] Shah D. Mean Average Precision (mAP) Explained : Everything You Need to Know [Internet]. V7. 2024. Disponible sur : https://www.v7labs.com/blog/mean-average-precision .Consulté le : 03/06/2024

[74] Mean Average Precision (MAP) : A complete guide [Internet]. Kili-website. Disponible sur : https://kili-technology.com/data-labeling/machine-learning/mean-average-precision-map-a-complete-guide .Consulté le : 03/06/2024

[75] L'histoire du Raspberry Pi [Internet]. Disponible surhttp://www.blewando.fr/elv/Promo2020/th8/pag3.html .Consulté le : 20/06/2024

[76]Fromaget P. L'Histoire incroyable du Raspberry Pi – RaspberryTips [Internet]. Disponible sur : https://raspberrytips.fr/histoire-du-raspberry-pi/ .Consulté le : 18/06/2024

[77] Présentation de la carte Raspberry Pi : [Découverte de Linux et de la carte Raspberry] [Internet]. Disponible sur : https://dautrylimoges.scenari-community.org/1STI2D/raspberry1_web/co/Presentation_de_la_carte_Raspberry.html.Consulté le : 17/06/2024

[78] Raspberry Pi Introduction | Raspberry Pi [Internet]. © 2018 ElectronicWings. Disponible sur : https://www.electronicwings.com/raspberry-pi/raspberry-pi-introduction

[79] Qu'est-ce que le Raspberry Pi ? [Internet]. Disponible sur : https://www.kubii.com/fr/content/72-qu-est-ce-que-le-raspberry-pi

[80] Raspberry Pi 3 modèle B [Internet]. Génération Robots. Disponible sur : https://www.generationrobots.com/fr/402366-raspberry-pi-3-modele-b.html.Consulté le : 20/06/2024

[81] speed breaker | Roboflow Universe Search [Internet]. Roboflow. Disponible sur : https://universe.roboflow.com/search?q=speed%20breaker .Consulté le : 20/06/2024

[82] Video to JPG (image sequence) converter [Internet]. Disponible sur : https://ezgif.com/video-to-jpg .Consulté le : 20/06/2024

[83] Philippeau X. Traitement d'images - Les filtres usuels [Internet]. Developpez.com. Disponible sur : https://xphilipp.developpez.com/articles/filtres/?page=page_3

[84]http://wcours.gel.ulaval.ca/2017/a/GIF4100/default/5notes/A2017TraitementImagesPartie1PageWeb.pdf .Consulté le : 20/06/2024

[85] Madeleine S. Normalisation, centrage and standardisation des images TDM [Internet]. IMAIOS. 2024. Disponible sur : https://www.imaios.com/fr/ressources/blog/images-tdm-normalisation-centrage-et-standardisation .Consulté le : 20/06/2024

[86] Malviya N. Object Detection — Anchor Box VS Bounding Box - Nikita Malviya - Medium. Medium [Internet]. 7 août 2023 ; Disponible sur : https://medium.com/@nikitamalviya/object-detection-anchor-box-vs-bounding-box-bf1261f98f12 .Consulté le : 20/06/2024

[87] Anchor Boxes for Object Detection - MATLAB & Simulink [Internet]. Disponible sur : https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html

[88] Mesbah F. Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel. [Internet]. 2021. Disponible sur : https://dspace.univ-guelma.dz/jspui/handle/123456789/11668 .Consulté le : 19/06/2024

[89] Make sense [Internet]. Disponible sur : https://www.makesense.ai/ .Consulté le : 19/06/2024

[90] Google Colab. Colab.google [Internet]. colab.google. Disponible sur : https://colab.google/ .Consulté le : 19/06/2024

[91]Wikipedia contributors. PyCharm [Internet]. Wikipedia. 2024. Disponible sur : https://en.wikipedia.org/wiki/PyCharm .Consulté le : 19/06/2024

[92] What's new in Python 3.12 [Internet]. Python Documentation. Disponible sur : https://docs.python.org/3/whatsnew/3.12.html.Consulté le : 19/06/2024

[93] The Python Standard Library [Internet]. Python Documentation. Disponible sur : https://docs.python.org/3/library/.Consulté le : 19/06/2024

[94]Deepakat. yolov8/yolov8_custom_training.ipynb at main · deepakat002/yolov8 [Internet]. GitHub. Disponible sur : https://github.com/deepakat002/yolov8/blob/main/yolov8_custom_training.ipynb.Consulté le : 19/06/2024

[95] Running Visual Studio Code on Raspberry Pi OS [Internet]. 2021. Disponible sur : https://code.visualstudio.com/docs/setup/raspberry-pi.Consulté le : 19/06/2024

[96] Kurbiel T. Deriving the Backpropagation Equations from Scratch (Part 1). Medium [Internet]. 16 déc 2021 ; Disponible sur : https://towardsdatascience.com/deriving-the-backpropagation-equations-from-scratch-part-1-343b300c585a consulté le : 22/06/2024