Détection automatique de l'environnement d'un véhicule autonome à l'aide de l'intelligence artificielle

par

Quentin GUILLEMELLE

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE
M. Sc. A.

MONTRÉAL, LE 11 SEPTEMBRE 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Maarouf Saad, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Pier-Marc Comtois-Rivet, codirecteur du mémoire
Département de robotique et intelligence artificielle à l'Institut du véhicule innovant, St-Jérome

M. Christian Belleau, président du jury
Département de génie mécanique à l'École de technologie supérieure

M. Chakib Tadj, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 5 SEPTEMBRE 2023

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde gratitude envers Pier-Marc Comtois-Rivet, codirecteur de mémoire, pour son soutien inestimable concernant l'aspect technique et pratique de mon travail de recherche. Sa disponibilité pour répondre à mes questions a été d'une grande aide tout au long de cette étude. J’ai beaucoup appris à ses côtés.

Un grand merci également au Professeur Maarouf Saad, directeur de mémoire, pour m'avoir guidé avec bienveillance dans la planification et l'atteinte de mes objectifs. Ses conseils éclairés et son accompagnement ont été déterminants pour la réussite de ce mémoire.

De plus, je souhaite exprimer ma reconnaissance envers ces deux personnes pour leur active collaboration et leur réactivité. Leurs contributions à la révision de mon mémoire, en fournissant des avis critiques pour corriger les fautes, ont grandement amélioré la qualité de mon travail.

Je n'oublie pas de remercier chaleureusement les autres collaborateurs du projet : Anthony Courchesne, Simon Savard et Yassine Kali qui ont partagé leurs informations précieuses, leur expertise et mis à ma disposition des outils de travail de qualité pour mener à bien cette recherche.

Je remercie également aussi le jury pour l’examen de mon mémoire

En outre, je tiens à exprimer toute ma gratitude envers mes parents et mes amis pour leur soutien moral inconditionnel tout au long de ce projet. Leur encouragement et leur présence ont été essentiels pour me motiver à surmonter les défis et avancer sereinement dans la réalisation de ce mémoire.

Finalement, ce mémoire est la dernière étape de mes études supérieures qui ont commencé en France et qui se terminent à Montréal. Par conséquent je souhaite remercier toutes les personnes qui ont contribué de loin ou de près à mon parcours scolaire et à son aboutissement.

Détection automatique de l'environnement d'un véhicule autonome à l'aide de l'intelligence artificielle

Quentin GUILLEMELLE

RÉSUMÉ

Le Ministère des Transports du Québec (MTQ) souhaite préparer la province du Québec pour l'arrivée des véhicules autonomes. Ce projet a été confié à l'École de Technologie supérieure (ÉTS) et à l'Institut du Véhicule Innovant (IVI), dans le but de cartographier les routes du Québec à l'aide d'images provenant d'une caméra montée sur un véhicule. À partir de ces images, des objets seront détectés à l'aide d'algorithmes de vision artificielle. Ainsi, le MTQ souhaite faire la détection automatique des nids de poule, lampadaires brisés et glissières de sécurité accidentées défectueuses afin de rehausser le niveau de sécurité routière.

Cette étude met l'accent sur la récupération de données locales afin d'entrainer des modèles de détections pour les infrastructures routières. De ce fait, des modèles basés sur l'apprentissage profond et l'architecture YOLO ont été entrainés. Le meilleur résultat obtenu pour la détection de nids de poule est 0.729 mAP. Pour la détection de lampadaires brisés, il est de 0.840 mAP. Aussi, ce projet a permis de récupérer une base de données contenant de plus de 8000 images et 4 classes de détections.

Mots-clés : Détection d'objets, Infrastructures routières, Apprentissage profond

Automatic detection of an autonomous vehicle's environment using artificial intelligence

Quentin GUILLEMELLE

ABSTRACT

The Ministry of Transportation of Quebec (MTQ) is preparing the province of Quebec for the arrival of autonomous vehicles. The project has been entrusted to ÉTS and the Innovative Vehicle Institute (IVI), with the aim of mapping Quebec roads using images from a vehicle-mounted camera. From these images, objects will be detected using computer vision algorithms. In this way, the MTQ aims to automatically detect potholes, broken streetlights and defective guardrails, in order to improve road safety.

This study focuses on the collection of local data in order to train detection models for road infrastructure. As a result, models based on deep learning and the YOLO architecture were trained. The best result obtained for pothole detection is 0.729 mAP. For the detection of broken lampposts, it is 0.840 mAP. In addition, this project has produced a database containing over 8000 images and 5 detection classes.

Keywords: object detection, road infrastructure, deep learning

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

IA	Intelligence Artificielle
MTQ	Ministère des Transports du Québec
IVI	Institut du Véhicule Innovant
LiDAR	Light Detection And Ranging
V2V	Vehicule to Vehicule
V2P	Vehicule to Pedestrian
HOG	Histogramme des Gradients Orientés
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
DPM	Deformable Part Model
SVM	Support Vector Machines
AdaBoost	Adaptive Boosting
R-CNN	Region-based Convolutional Neural Network
CNN	Convolutional Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
VOC	Visual Object Classes
SSD	Single Shot Detector
YOLO	You Only Look Once
VGG	Visual Geometry Group
ResNet	Residual Neural Network
NLP	Natural Language Processing
DETR	DEtection Transformer
FFN	Feed Forward Network
ViT	Vision Transformer
CRDDC	Crowd sensing-based Road Damage Detection Challenge
RDD	Road Damage Dataset
AWS	Amazon Web Service

CVAT	Computer Vision Annotation Tool
GPS	Global Positioning System
CAN	Controller Area Network
ETS	École de Technologie Supérieure
RGB	Red Green Blue
ABS	Anti Blocker System
LTE	Long Term Evolution
SQL	Structured Query Language
API	Application Programming Interface
ROC	Receiving Operator Characteristic
FPN	Feature Pyramid Network
COCO	Common Objects in COntext
CmBN	Cross mini-Batch Normalization
NMS	Non-MaximumSuppression
PANet	Path Aggregation Network
SPP	Spatial Pyramid Pooling
CSPNet	Cross Stage Partial Network
SPPF	Spatial Pyramid Pooling Fusion
BCE	Binary Cross Entropy
GPU	GraphicProcessing Unit
mAP	mean Average Precision
FPS	Frames Per Second
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
AP	Average Precision
PR	Precision Recall
HSV	Hue Saturation Value

RGB	Red Green Blue

LISTE DES SYMBOLES ET UNITÉS DE MESURE

FPS	Frame per seconde
px	pixel
mAP	mean Average Precision

INTRODUCTION

Dans un monde de plus en plus interconnecté, l'automatisation et l'intelligence artificielle (IA) jouent un rôle de plus en plus important dans de nombreux domaines, y compris le transport. L'avènement des véhicules autonomes pourrait notamment permettre une gestion du trafic plus efficace tout en augmentant la sécurité sur la route. Cependant, pour que ces véhicules fonctionnent de manière efficace et sûre, ils doivent être capables de détecter, d'interpréter et d'interagir correctement avec leur environnement. D'autant plus, qu'ici au Québec, la météo ne facilite pas la conservation d'infrastructures correctes : la chaussée se dégrade rapidement à cause des grands écarts de température par exemple.

Ainsi, le Ministère des Transports du Québec (MTQ) fait face à plusieurs défis : maintenir l'intégrité de son réseau routier, assurer la sécurité de ses utilisateurs et préparer l'infrastructure pour l'avènement des véhicules autonomes. C'est dans ce contexte que s'inscrit ce mémoire, intitulé "Détection automatique de l'environnement d'un véhicule autonome à l'aide de l'intelligence artificielle".

On retrouve dans leur plan stratégique de 2019-2023, l'objectif premier du MTQ : le maintien des infrastructures routières en bon état. L'atteinte de cet objectif est mesurée selon plusieurs indicateurs comme :

- La proportion des structures du réseau routier supérieur en bon état selon l'indice d'état gouvernemental
- La proportion des chaussées du réseau routier supérieur en bon état selon l'indice d'état gouvernemental

À titre d'exemples, le rapport annuel de gestion 2021-2022 du MTQ confirme que les proportions cibles de ces deux objectifs (respectivement 78 % et 52 %) ont été respectées. Les objectifs pour 2023 sont fixés respectivement à 79 % et 53 %. C'est tout de même plus de 31000 kilomètres de chaussées du réseau routier supérieur qui sont sous la responsabilité du MTQ aujourd'hui.

D'ailleurs, en 2021, c'est 787 millions de dollars d'investis pour la conservation des routes du Québec (« Bilan de l'état des chaussées du réseau routier supérieur québécois, 2021 », 2021) (Figure 0.1.1). Et en 2021, le bon état des chaussées est estimé à 77,7 % (Figure 0.1.2).

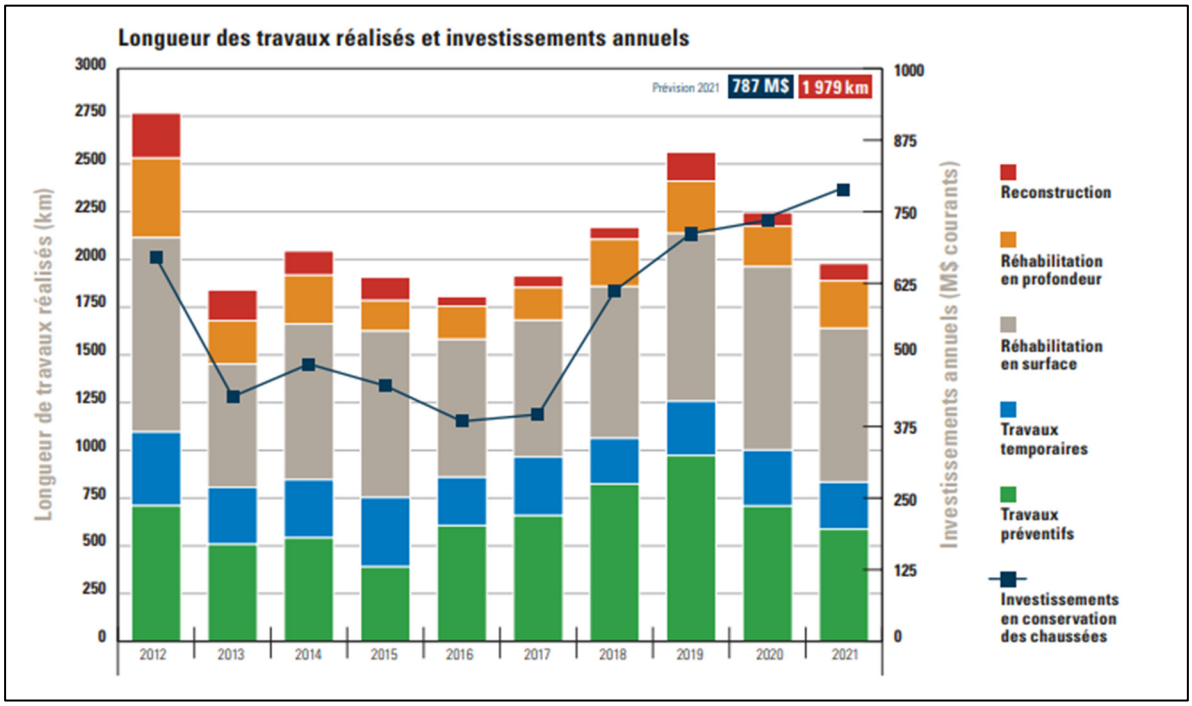

Figure 0.1.1 Évolution annuelle du coût investi par le MTQ (Ministère du Transport du Québec, 2021)

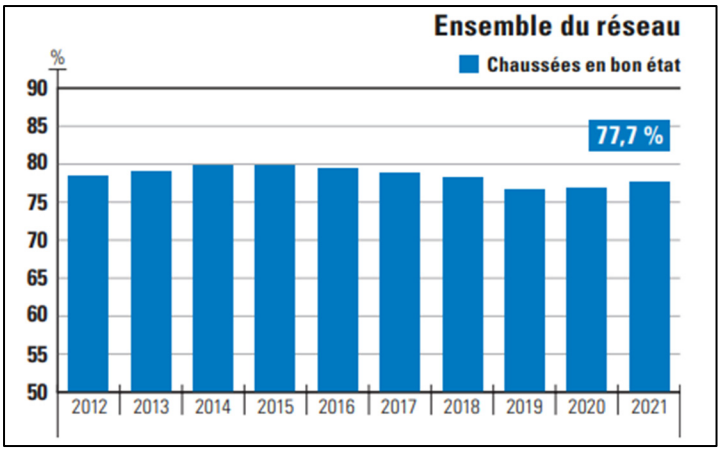

Figure 0.1.2 Bilan de l'état des chaussées selon l'indice de rugosité international (Ministère du Transport du Québec, 2021)

Dans cette dynamique, un projet a été confié à l'École de technologie supérieure (ÉTS) et à l'institut du Véhicule Innovant (IVI) pour cartographier les routes du Québec et prendre des images provenant d'une caméra montée sur un véhicule. À partir de ces images, des objets seront détectés à l'aide d'algorithmes de vision artificielle. Le MTQ souhaite faire la détection automatique des nids de poule, des lampadaires brisés et des glissières de sécurité endommagées.

L'objectif du projet est d'obtenir un algorithme de détection d'infrastructures défectueuses basé sur les images d'une caméra embarquée dans un véhicule. Pour l'atteindre, nous avons développé une architecture complète qui permet d'extraire des données d'intérêt sur les routes. Mais aussi, elle permet d'annoter ces données, afin d'être capable d'entrainer des modèles de détection d'objets basés sur l'apprentissage profond. Notre étude permettra un meilleur entretien du réseau routier pour augmenter considérablement la sécurité routière. De plus, les données récoltées permettront au MTQ d'effectuer des analyses pour une conception des routes plus adaptée et des interventions d'entretien plus efficaces.

Finalement, à travers 4 chapitres, ce rapport expose les étapes importantes du projet, ses résultats et son aboutissement pour y dégager des perspectives et recommandations :

- Le chapitre 1 est une revue de la littérature existante sur le sujet. Il commence par une introduction générale à la détection automatique et aux véhicules autonomes, avant de se concentrer sur les technologies de détection utilisées dans ces voitures. Il examine ensuite les techniques de détection d'objets basées sur les caméras, en mettant l'accent sur les techniques traditionnelles et celles basées sur l'apprentissage profond. Enfin, il aborde la détection des infrastructures routières défectueuses, en présentant des exemples de méthodes de détection de nids de poule et en discutant des ensembles de données utilisés.
- Le chapitre 2 décrit l'infrastructure et la méthodologie générale utilisées dans ce mémoire. Il détaille le processus d'acquisition des ensembles de données annotés et comment les réseaux neuronaux profonds sont utilisés dans le cas d'étude.
- Le chapitre 3 se concentre sur la construction des jeux de données utilisés. Il introduit le concept d'apprentissage actif et décrit comment il a été utilisé pour identifier et détecter les nids de poule. Ce chapitre aborde aussi la construction d'un jeu de donnée pour les lampadaires brisés et les glissières de sécurité endommagées.
- Le chapitre 4 présente les résultats expérimentaux obtenus. Il définit les métriques et les outils d'évaluation des performances utilisés, et discute des résultats obtenus pour la détection des infrastructures défectueuses.

CHAPITRE 1

REVUE DE LITTERATURE

1.1 Introduction

Par sa structure technologique et son domaine d'application, mon sujet d'étude concerne indirectement de nombreuses technologies et concepts. Nous les retrouverons tout au long de mon mémoire de manière logique.
Pour ce chapitre, l'objectif est de mettre en contexte le sujet de mon mémoire. Pour ceci, nous aborderons diverses notions utiles à ma recherche :

- La détection automatique sur les véhicules autonomes
- La détection d'objets par caméra
- La détection d'infrastructures sur les routes : modèles et jeux de données

1.2 Détection automatique et véhicules autonomes

Dans un monde de plus en plus axé sur la technologie, l'automatisation et l'intelligence artificielle sont devenues des éléments clés de notre quotidien. Un domaine où ces avancées ont un impact significatif est celui des transports, avec l'émergence des véhicules autonomes. Ces véhicules, capables de se déplacer sans intervention humaine, promettent de révolutionner notre façon de voyager, de réduire les accidents de la route et de rendre les déplacements plus efficaces. Cependant, la réalisation de ces promesses repose sur la capacité des véhicules à percevoir et à comprendre leur environnement, une tâche accomplie grâce à la détection automatique.

La détection automatique, qui fait référence à la capacité d'un système à identifier et à localiser des objets ou des caractéristiques spécifiques dans son environnement, est un élément essentiel des véhicules autonomes. Elle permet aux véhicules de "voir" et de réagir à leur environnement, ce qui est crucial pour la navigation et la sécurité. Cependant, malgré son

importance, la détection automatique dans le contexte des véhicules autonomes est un domaine complexe et en constante évolution, avec de nombreux défis à surmonter.

L'objectif de cette partie 1.2 est d'examiner les recherches existantes sur la détection automatique dans le contexte des véhicules autonomes. En fait, nous explorerons les différentes technologies de détection utilisées et leurs aboutissements.

1.2.1 Technologies de détection dans les véhicules autonomes

Dans le développement des voitures autonomes, les capteurs de détection jouent un rôle essentiel. Les progrès réalisés dans ce domaine ont permis de développer des systèmes sophistiqués capables de percevoir l'environnement. Ces technologies innovantes ouvrent de nouvelles perspectives pour la conduite autonome, allant de la sécurité routière accrue à une expérience de conduite plus agréable et efficiente.

Au cœur de ces avancées se trouve une multitude de technologies de détection comme le LiDAR (Laser Imaging Detection And Ranging), le radar, les caméras et les capteurs à ultrasons. Cette section 1.2.1 vise à explorer les différentes technologies de détection utilisées dans les véhicules autonomes, en expliquant leurs principes de fonctionnement et leurs applications (Hirz & Walzel, 2018) :

- La technologie LiDAR (détection et télémétrie par laser) est une technologie couramment utilisée dans les véhicules autonomes. Elle utilise des lasers pour émettre des impulsions lumineuses qui rebondissent sur les objets et retournent au capteur. En mesurant le temps mis par la lumière pour revenir, les systèmes LiDAR peuvent créer des cartes détaillées en 3D de l'environnement. Le LiDAR offre des données de nuages de points à haute résolution et est particulièrement utile pour la détection des obstacles, la localisation et la cartographie dans les voitures autonomes.

- Le radar, acronyme de détection et télémétrie par ondes radio, utilise des ondes radio pour détecter les objets et déterminer leur distance, leur vitesse et leur angle. Il s'agit d'une technologie mature et largement adoptée dans l'industrie automobile. Les systèmes radars peuvent fonctionner à différentes fréquences, offrant ainsi différents niveaux de précision et de portée. Ils excellent dans des conditions météorologiques difficiles et sont efficaces pour détecter des objets hors du champ de vision direct, ce qui les rend essentiels pour l'évitement des collisions et le régulateur de vitesse adaptatif.

- Les systèmes de vision, basés sur les caméras utilisant des algorithmes de traitement d'images, jouent un rôle essentiel dans la perception des véhicules autonomes. Les caméras captent des images, qui sont ensuite traitées pour extraire des informations significatives telles que la détection d'objets, les marquages de voie et la reconnaissance des panneaux de signalisation. Les récents progrès dans les algorithmes d'apprentissage profond ont considérablement amélioré la précision et la fiabilité des systèmes de vision basés sur les caméras. Ils ont permis des applications telles que la détection des piétons et l'assistance au maintien dans la voie.
  Les caméras offrent une grande richesse d'informations visuelles, mais peuvent être sensibles aux variations d'éclairage et aux conditions météorologiques défavorables.

- Les capteurs à ultrasons émettent des ondes sonores de haute fréquence et mesurent le temps mis par les ondes pour revenir après avoir heurté un objet. Le temps de parcours permet d'interpréter la distance entre l'objet et la voiture. Ces capteurs sont couramment utilisés pour la détection d'objets à courte portée et les systèmes d'aide au stationnement. Les capteurs ultrasoniques fournissent une détection de proximité fiable et sont particulièrement utiles pour les manœuvres à faible vitesse tout comme l'évitement des obstacles lors du stationnement.

Les technologies de détection sont des composantes essentielles du développement des véhicules autonomes, leur permettant de percevoir et de naviguer en toute sécurité dans leur environnement. Chaque technologie offre des capacités uniques et contribue à différents aspects de la conduite autonome, notamment la détection des objets, la cartographie, la localisation et le contrôle. D'ailleurs, ces technologies sont souvent combinées en perception pour tirer parti des avantages de chacune. Par exemple, les capteurs radar sont utilisés pour la détection des obstacles et la mesure de la vitesse relative, alors que les caméras permettent d'identifier le type de véhicule en mouvement ou l'obstacle.

En bref, voici un résumé d'une étude comparative de Hirz & Walzel (2018) dans le Tableau 1.1, de différentes technologies et leur niveau de qualité selon plusieurs critères comme le coût matériel ou la météo.

Tableau 1.1 Comparaison des différentes technologies de capteurs pour les fonctions de conduite automatisée
Tiré de Hirz & Walzel (2018)

Sensor	Bright light performance	Low light performance	Outdoor	Weather robustness	Vehicle classification	Vehicle adaptation	Material costs
Ultrasonic	Good	Good	Yes	Good	No	No	Low
Magnetic	Good	Good	Yes	Good	No	Yes	High
2D-camera	Good	Weak	Yes	Weak	Yes	No	Low
Laser and lidar	Good	Good	Yes	Good	Yes	No	Very High
3D-imaging technologies							
3D-camera	Good	Weak	Yes	Weak	Yes	No	Low
Structured light	Weak	Good	No	Weak	Yes	No	Medium
ToF-cameras	Good	Good	Yes	Good	Yes	No	High

Enfin, le partage d'information provenant des technologies évoquées plus haut est facilité par des réseaux connectés. Par exemple, le V2V (Vehicule-to-Vehicule) utilise des signaux sans fil pour échanger des informations entre véhicules sur les accidents, les conditions météorologiques, les barrages routiers et le trafic (Vinel, et al., 2018). Ou encore, le V2P (Vehicule-to-Pedestrian) qui communique sans fil avec les piétons (via leur téléphone intelligent par exemple) pour détecter leur direction et distance (Bauk et al., 2017).

Le partage d'information permet d'améliorer la compréhension globale de l'environnement des voitures autonomes.

1.3 La détection d'objets par caméra

De manière générale, on peut distinguer dans le domaine de la vision par ordinateur deux grandes familles :

- La segmentation est un processus qui divise une image en plusieurs segments, chacun correspondant à un objet ou à une partie d'un objet. Cette technique est particulièrement utile dans des applications telles que l'imagerie médicale, où elle peut aider à identifier et à isoler des structures spécifiques dans une image. Brièvement, on retrouve trois techniques différentes : la segmentation sémantique, d'instance et panoptique.
  La segmentation sémantique vise à classer chaque pixel dans une image en une catégorie d'objet. Par exemple, dans une image de rue, la segmentation sémantique identifierait les zones de l'image qui correspondent à des voitures, des personnes, des bâtiments, etc. (Long et al., 2015).
  La segmentation d'instance (He et al., 2018), non seulement identifie la présence d'objets spécifiques dans une image, mais distingue également entre différentes instances du même objet. Par exemple, si une image contient plusieurs voitures, la segmentation d'instance identifierait chaque voiture comme un objet séparé.
  Enfin, la segmentation panoptique, le concept relativement récent en vision par ordinateur, combine les forces de la segmentation sémantique et de la segmentation d'instance. Elle vise à fournir une compréhension unifiée et cohérente de la scène en étiquetant chaque pixel d'une image avec une étiquette de classe et une étiquette d'instance (Kirillov et al., 2019). Cette technique est particulièrement utile dans des scènes complexes où plusieurs objets se chevauchent et interagissent. La Figure 1.1 illustre pour une image (a) donnée, (b) la réalité du terrain pour la segmentation sémantique (étiquettes de classe par pixel), (c) la segmentation d'instance (masque et

étiquette de classe par objet) et (d) la tâche de segmentation panoptique proposée (étiquettes de classe et instance par pixel).

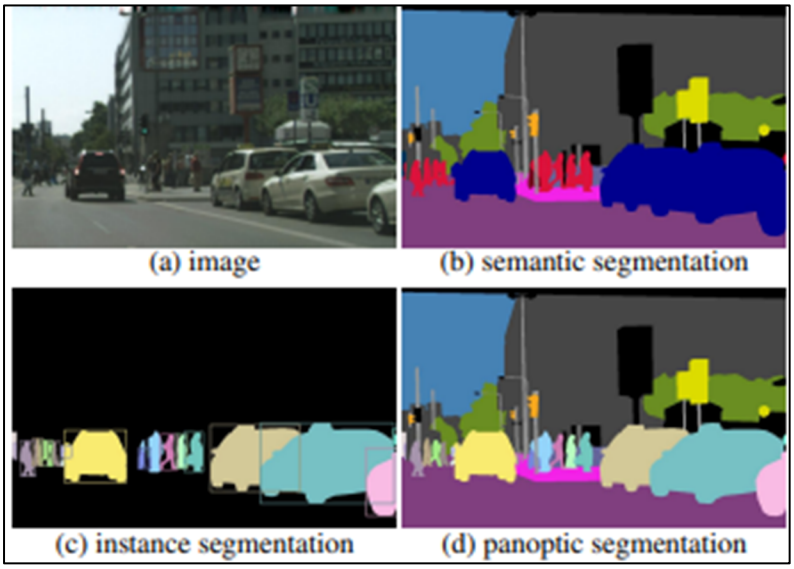

Figure 1.1 Comparaisons des techniques de segmentation
Tirée de Kirillov et al. (2019)

- La détection d'objets, quant à elle, est une tâche de vision par ordinateur qui consiste à identifier et à localiser des objets dans une image ou une vidéo. Il s'agit d'une tâche difficile, car les objets peuvent être de tailles, de formes et d'apparences différentes, et ils peuvent être situés dans n'importe quelle partie de l'image (Zou et al., 2023). Dans le cadre des routes, il peut s'agir d'objets comme des plaques d'immatriculation, véhicules, piétons, panneau, nids de poule ou encore des barrières de sécurité. Plusieurs des méthodes de détection d'objet sont abordées dans cette partie 1.3.

En conclusion, grâce à l'arrivé de la segmentation panoptique, les modèles de segmentation ont gain d'intérêt notamment pour la conduite autonome (Feng et al., 2021). Cependant, nous nous concentrerons dans la section 1.3 uniquement sur l'exploration des différentes techniques connues pour la détection d'objets basée sur une caméra.

Abordons donc les techniques traditionnelles et les techniques basées sur l'apprentissage profond.

1.3.1 Techniques traditionnelles de détection d'objets

Traditionnellement, les approches basées sur les caractéristiques étaient largement utilisées pour la détection d'objets. Ces techniques impliquaient l'extraction de caractéristiques conçues à la main à partir d'images et l'utilisation d'algorithmes d'apprentissage automatique pour la classification. Les caractéristiques couramment utilisées comprenaient l'histogramme des gradients orientés (HOG) (Dalal & Triggs, 2005), la transformation de caractéristiques invariantes à l'échelle (SIFT) (Lowe, 2004) et les caractéristiques robustes accélérées (SURF) (Bay et al., 2006). Ces méthodes ont montré des promesses dans certains scénarios, mais celles-ci ont rencontré des limites avec les variations d'échelle, les conditions d'éclairage et l'occlusion.

Les techniques de fenêtres glissantes et de recherche sélective étaient couramment employées conjointement avec des approches basées sur les caractéristiques. Les méthodes de fenêtres glissantes impliquaient le balayage d'une image à plusieurs échelles et positions, en appliquant un classificateur à chaque fenêtre. L'algorithme de Viola-Jones (Viola & Jones, 2001) est un exemple notable de cette approche. Les méthodes de recherche sélective visaient à générer des régions d'objets potentielles en regroupant des régions d'image similaires (Uijlings et al., 2013). Ces méthodes ont contribué à réduire la complexité computationnelle, mais ont rencontré des difficultés dans la localisation précise et la gestion des objets se chevauchant.

Les modèles de parties déformables (DPM) ont gagné en popularité en tant qu'extension des approches traditionnelles basées sur les caractéristiques. Les DPM visaient à gérer les variations d'apparence des objets et les déformations en décomposant les objets en plusieurs parties. Cette technique a permis de modéliser les relations spatiales entre les parties et l'objet dans son ensemble. Felzenszwalb et al. (2010) ont introduit la structure DPM, démontrant une

amélioration des performances de détection d'objets sur des ensembles de données difficiles. Cependant, les DPM nécessitaient souvent la conception manuelle de modèles de séparation et ont rencontré des limites dans la gestion des grandes variations de pose.

On retrouve également dans la littérature, la combinaison des caractéristiques de l'histogramme des gradients orientés (HOG) avec les machines à vecteurs de support (SVM) pour la détection d'objets. Le HOG, introduit par Dalal et Triggs (2005), capture les informations de forme de l'objet basées sur les gradients. Quant au SVM, il a été utilisé comme classificateur pour distinguer entre les régions d'objets et les régions non objet. Cette approche a obtenu un succès notable dans la détection des piétons et est devenue une référence pour les techniques ultérieures. Cependant, elle a rencontré des difficultés à traiter les catégories d'objets complexes et les arrière-plans encombrés.

Les classificateurs en cascade, popularisés par Viola et Jones (2001), ont fourni une approche efficace pour la détection d'objets. Les classificateurs en cascade utilisaient une séquence de classificateurs organisés en une structure en cascade, filtrant progressivement les régions non objet. AdaBoost (Freund & Schapire, 1999), un algorithme de boosting, a entrainé des classificateurs faibles de manière itérative pour en créer des classificateurs forts. Cette combinaison a permis la détection d'objets en temps réel dans des environnements à ressources limitées. Cependant, ces méthodes étaient sensibles aux variations d'échelle et nécessitaient un réglage précis des paramètres.

Les approches traditionnelles basées sur les caractéristiques, les méthodes de fenêtre glissante et de recherche sélective, les modèles de parties déformables, le HOG avec les SVM et les classifieurs en cascade avec AdaBoost ont contribué de manière significative au domaine. Bien que ces techniques aient démontré des avancées dans la détection d'objets, elles ont également rencontré des limites en matière de variations d'échelle, d'occlusions, de localisation précise et de catégories d'objets complexes. L'émergence des techniques d'apprentissage profond a

révolutionné le domaine de la détection d'objets, surmontant bon nombre de ces limitations et atteignant des performances de pointe.

1.3.2 La détection d'objets basée sur l'apprentissage profond

Voyons à présent plus en détail les différents modèles qui ont contribué au succès de la détection d'objets par apprentissage profond.

La famille R-CNN (Region-based Convolutional Neural Networks) a apporté des avancées significatives dans le domaine de la détection d'objets en vision par ordinateur. Ce sont des modèles basés sur des réseaux de neurones convolutifs (Bharati & Pramanik, 2020).

Le R-CNN (Girshick et al., 2014) est une méthode en deux étapes. Tout d'abord, elle génère des propositions de régions d'intérêt (region proposals) en utilisant des algorithmes tels que Selective Search présenté dans Uijlings *et al*. (2013) (étape 2 de la Figure 1.2.) Ensuite, chaque région est extraite et redimensionnée pour être entrée dans un réseau de neurones convolutifs (CNN) préentrainé, tel qu'AlexNet (étape 3, Figure 1.2). Les caractéristiques extraites sont ensuite utilisées pour classifier et localiser les objets dans chaque région (étape 4, Figure 1.2). Cette approche a montré des performances améliorées par rapport aux méthodes antérieures.

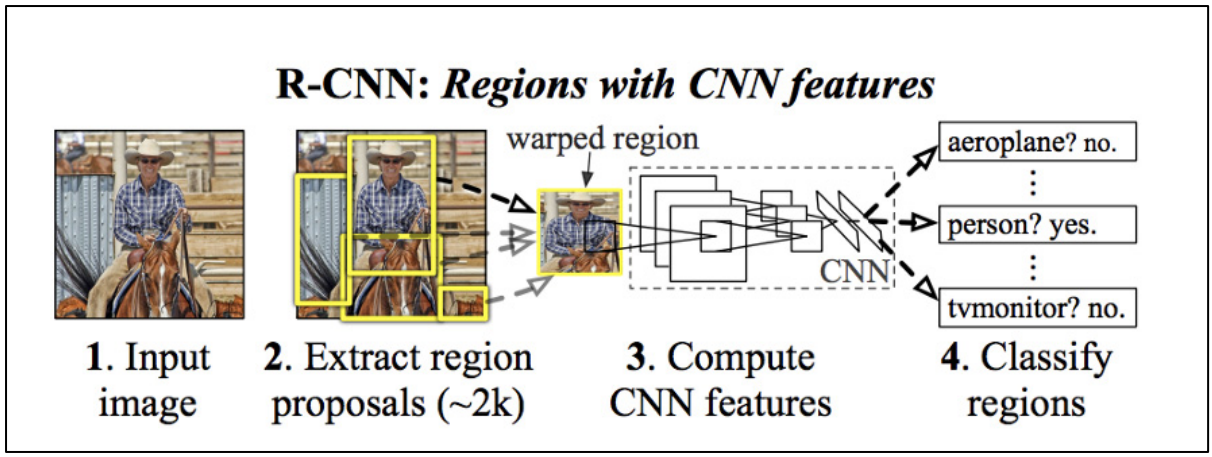

Figure 1.2 Structure schématique du R-CNN
Tirée de Girshick et al. (2014)

Le Fast R-CNN, présenté par Girshick (2015), a introduit des améliorations par rapport au R-CNN en termes de vitesse et d'efficacité. Sa structure générale est présentée en Figure 1.3. Dans cette méthode, l'image complète est passée dans le CNN (Convolutional Neural Networks) pour extraire les caractéristiques une seule fois. Ensuite, les régions d'intérêt sont alignées avec les caractéristiques en utilisant une couche de mise en correspondance (ROI pooling), ce qui permet de générer une représentation fixe de chaque région. Enfin, ces représentations sont utilisées pour la classification et la localisation des objets. Le Fast R-CNN a permis d'accélérer considérablement le processus de détection (Girshick, 2015).

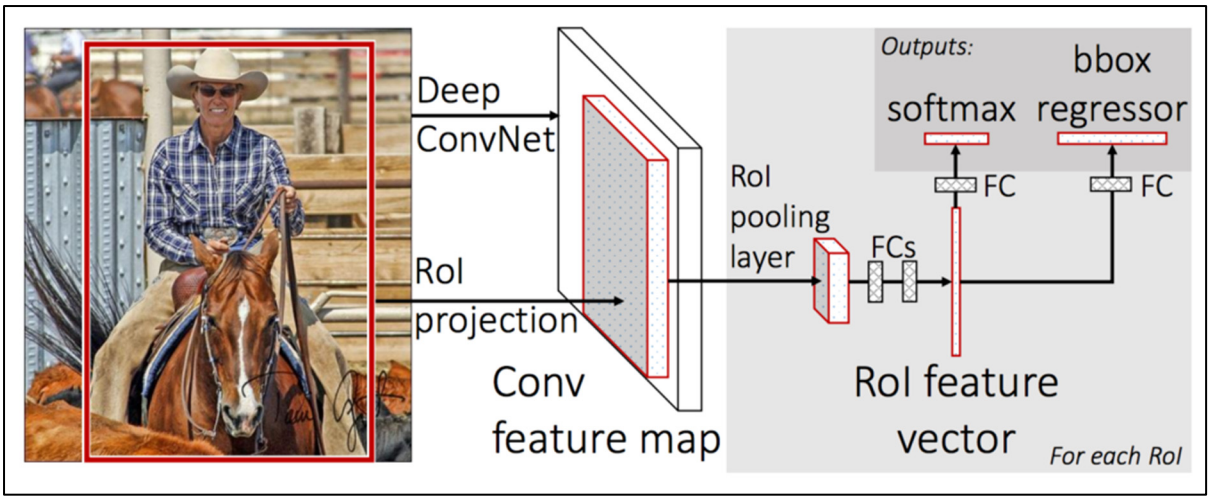

Figure 1.3 Structure schématique du Fast R-CNN
Tirée de Girshick (2015)

Présenté dans Ren et al. (2016), le Faster R-CNN introduit un module appelé Region Proposal Network (RPN) pour générer les propositions de régions d'intérêt directement à partir du CNN. Initialement on faisait appel à un algorithme différent pour les générer. Le RPN partage les caractéristiques du CNN et génère des ancres à échelles et ratios différents pour générer les propositions de régions. Ces propositions de régions sont ensuite utilisées par le reste du réseau pour effectuer la classification et la localisation des objets. Le Faster R-CNN a amélioré à la fois la précision et la vitesse de détection d'objets (Ren et al., 2016).

L'évolution des R-CNN via leur modification de structure, présentée dans le paragraphe précédent, est résumée dans la Figure 1.4 ci-dessous.

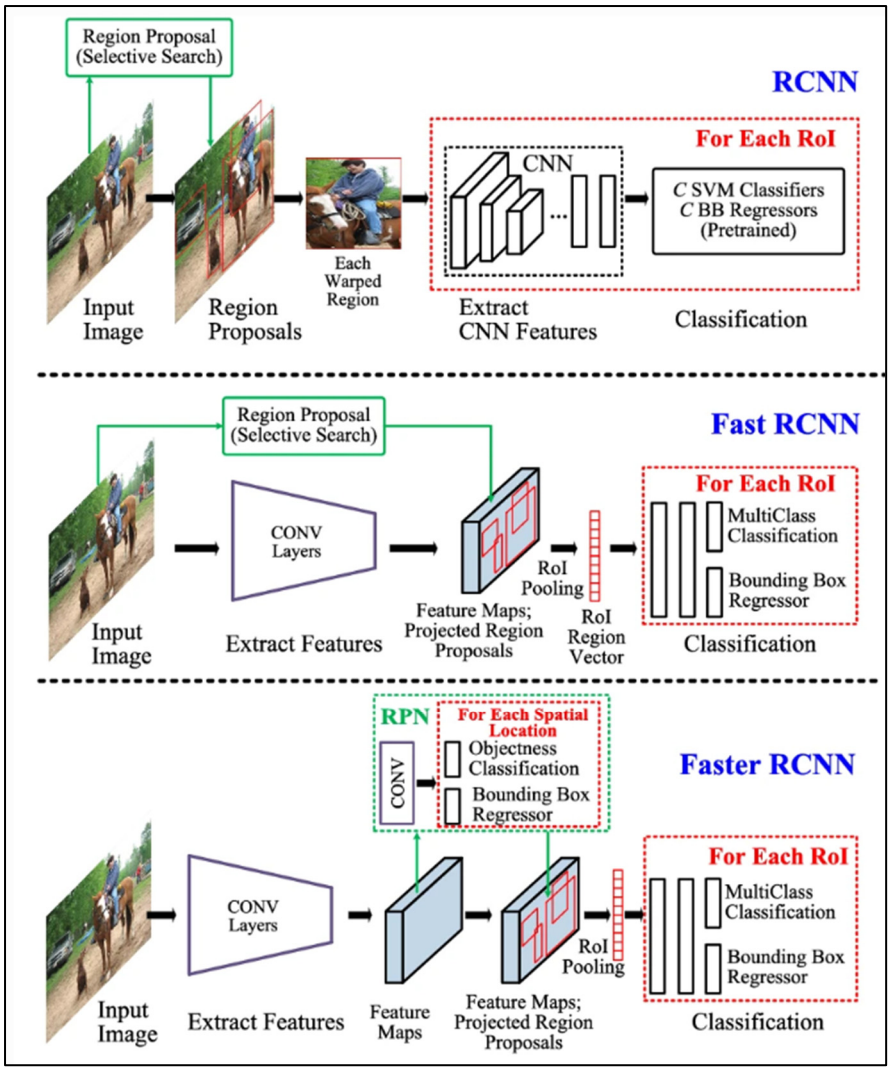

Figure 1.4 Résumé et comparaison des structures RCNN (en haut),
Fast RCNN (au milieu) et Faster RCNN (en bas)
Tirée de Liu *et al*. (2020)

Depuis l'introduction du RCNN (Girshick et al., 2014) puis la sortie du Faster RCNN (Ren et al., 2016) et ses excellents résultats sur la détection d'objets avec des jeux de données populaires comme le Pascal VOC, les détecteurs basés sur la suggestion de région sont leader dans la détection d'objet. Cependant, ces réseaux sont connus aussi pour leur coût de calcul élevé, qui se révèle problématique pour des équipements aux capacités de calcul et stockage limités. C'est pourquoi les chercheurs ont commencé à développer des modèles à stratégie de

détection unifiée comme le Single Shot Detector (SSD) ou You Only Look Once (YOLO) au lieu de continuer à essayer d'optimiser les composantes individuelles du modèle.

Ainsi, quant au détecteur en une étape, on retrouve tout d'abord le Single Shot Detector (SSD) proposé par Liu et al. (2016). Contrairement aux approches en deux étapes telles que R-CNN, le modèle SSD réalise la détection d'objets en une seule étape : il prédit directement les boîtes englobantes et les probabilités de classe pour les objets dans l'image. La Figure 1.5 illustre un exemple permettant de comprendre ce que représentent visuellement des boîtes englobantes.

Figure 1.5 Exemples de boîtes englobantes (en vert) et leur probabilité de classe associée

De plus, comme présenté à la Figure 1.6, ce modèle utilise un réseau CNN préentrainé, tel que VGG ou ResNet, comme base pour extraire des caractéristiques de l'image. Des couches supplémentaires, appelées couches de détection, sont ajoutées au réseau pour prédire les boîtes englobantes et les probabilités de classe. Aussi, il utilise plusieurs échelles de détection en ajoutant des couches de détection à différentes profondeurs du réseau CNN. Cela permet de détecter des objets de différentes tailles et de gérer les variations d'échelle dans l'image.

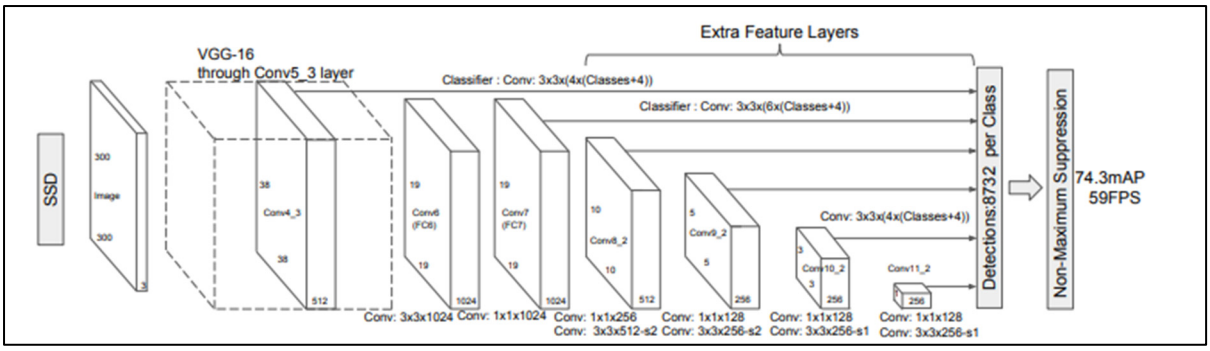

Figure 1.6 Structure par couches du SSD
Tirée de Liu *et al*. (2016)

Enfin, introduit par Redmon, *et al*. (2016), l'algorithme You Only Look Once (YOLO) divise l'image en une grille de cellules et prédit directement les boîtes englobantes et les probabilités de classe pour chaque cellule. La Figure 1.7 illustre bien cette différence d'architecture avec le SSD.

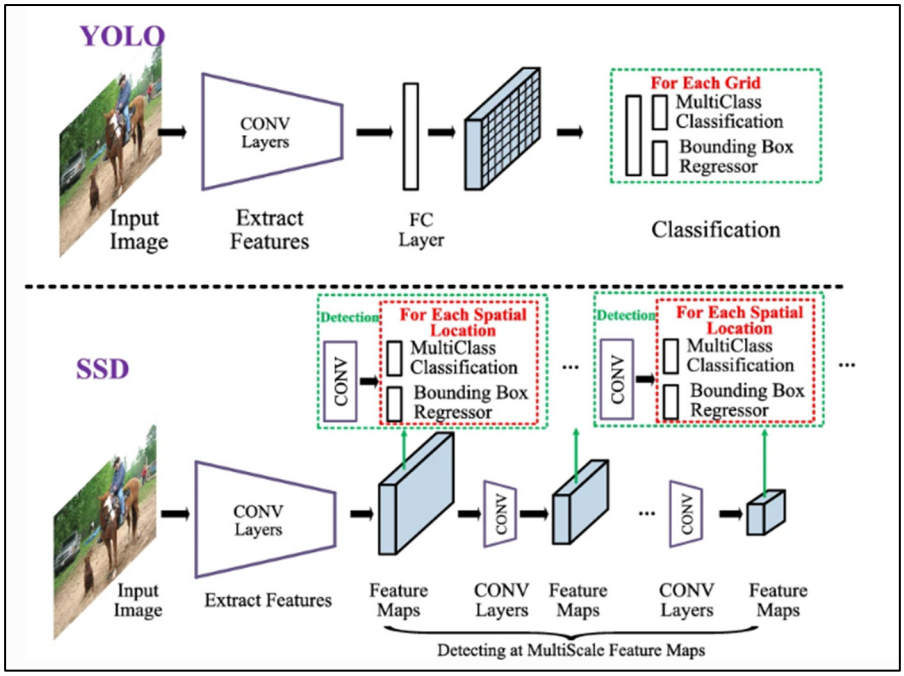

Figure 1.7 Résumé et comparaison des structures YOLO (en haut)
et SSD (en bas)
Tirée de Liu et al. (2020)

Le modèle utilise un réseau CNN, généralement avec plusieurs couches convolutives, pour extraire des caractéristiques de l'image. Ces caractéristiques sont ensuite utilisées pour prédire les boîtes englobantes et les probabilités de classe comme le montre son architecture disponible en Figure 1.8. Cette structure permet de traiter rapidement de grandes quantités d'images et de détecter efficacement différents types d'objets dans des scènes complexes.

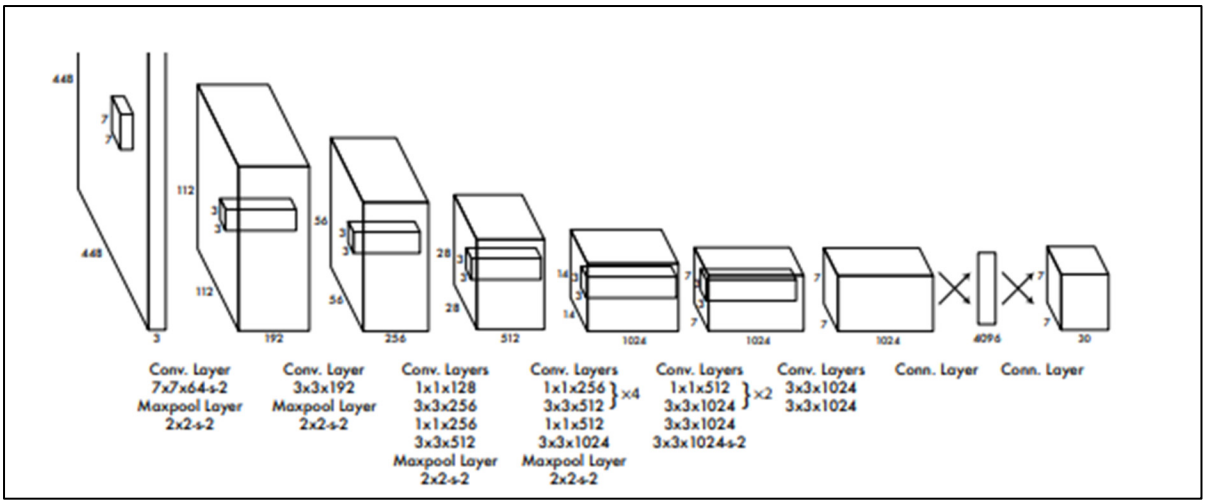

Figure 1.8 Architecture par couche de YOLO
Tirée de Redmon *et al.* (2016)

En bref, l'avantage majeur des algorithmes comme le SSD ou YOLO est leur vitesse d'exécution, tout en gardant de bonnes précisions. C'est pourquoi leur utilisation est très prisée dans des applications nécessitant une détection d'objets en temps réel, comme la surveillance vidéo ou la conduite autonome.

Néanmoins, ces dernières années, le domaine de l'apprentissage profond évolue rapidement. Les dernières tendances suggèrent l'utilisation d'autres méthodes comme les modèles basés sur des transformateurs.

Les transformateurs ont révolutionné le domaine de l'apprentissage automatique, en particulier dans le domaine du traitement du langage naturel (NLP) et de la détection d'images (Zaidi et al., 2022). Le modèle de transformateur, introduit par Vaswani et al. (2017), est basé sur des

mécanismes d'auto-attention et écarte totalement l'utilisation de la récurrence et des convolutions. Les auteurs soutiennent que ces dispositifs permettent au modèle de se concentrer sur différentes parties de la séquence d'entrée, offrant une compréhension plus nuancée et globale du contexte (Vaswani et al., 2017).

Dans le domaine de la détection d'images, Carion et al. (2020) ont introduit le modèle DETR (DEtection TRansformer). DETR utilise une architecture CNN pour apprendre une représentation 2D d'une image d'entrée. Le modèle l'aplatit (la structure est mise en forme unidimensionnelle) et ajoute un codage positionnel (permettant de conserver une certaine notion d'emplacement des caractéristiques dans l'image) avant de la passer dans le transformateur. À la sortie, un réseau de propagation avant partagé (FFN) prédit : soit une détection (classe et boîte englobante), soit une classe "aucun objet" (Figure 1.9).

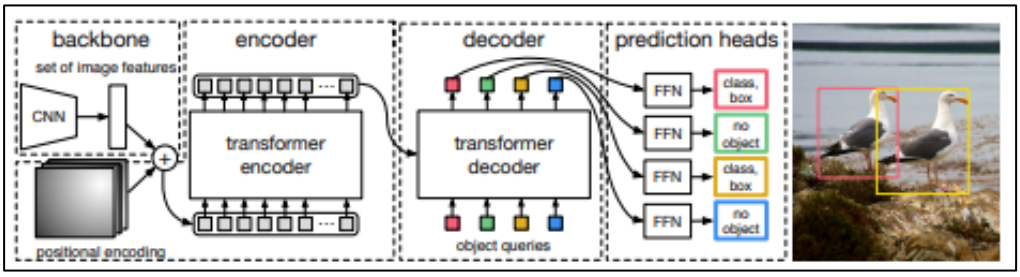

Figure 1.9 Architecture d'un DETR
Tirée de Carion et al. (2020)

Ce modèle est présenté comme simple, polyvalent et compétitif avec les détecteurs basés sur les CNN. Il démontre le potentiel des transformateurs dans la détection d'images (Carion et al., 2020).

De plus, Le Vision Transformer (ViT), introduit par Dosovitskiy et al. (2020), est un modèle qui repense l'application des transformateurs dans le domaine de la détection d'images. Contrairement au DETR qui combine les transformateurs avec les réseaux neuronaux de convolutions (CNN), le ViT applique directement l'architecture des transformateurs à des séquences de patches d'images. En effet, dans le modèle ViT, une image est d'abord divisée

en une série de petits carrés, appelés "patches". Ces patches sont ensuite traités comme une séquence par le transformateur (Figure 1.10). Cela signifie que le ViT traite les images de la même manière que les modèles de transformateurs traitent les séquences de mots dans le traitement du langage naturel.

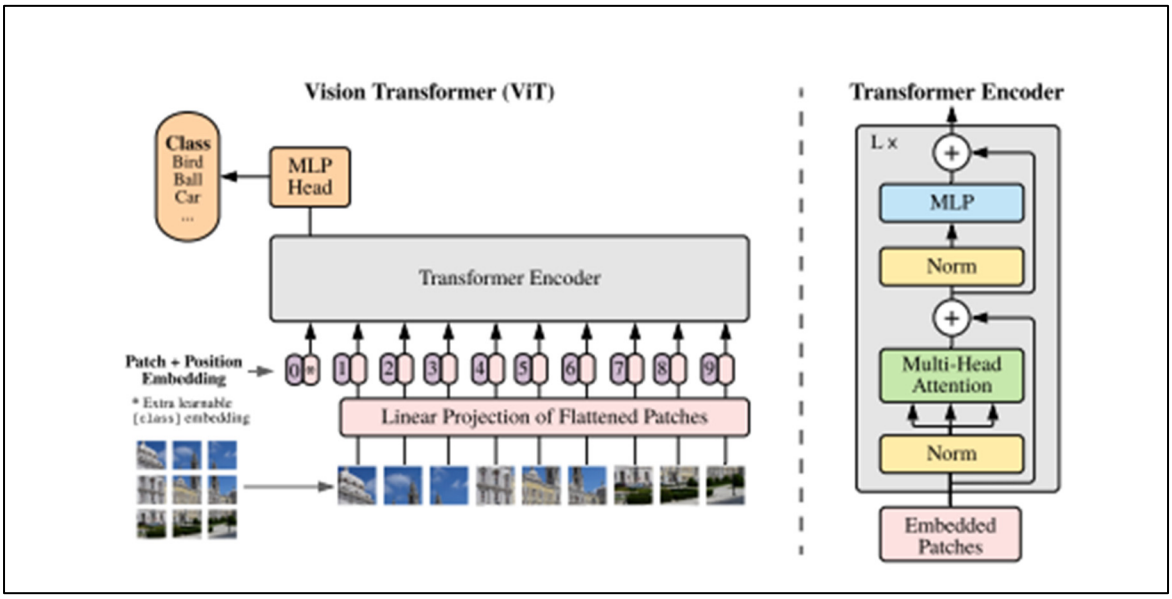

Figure 1.10 Aperçu du modèle ViT
Tirée de Dosovitskiy et al. (2021)

Les résultats obtenus par Dosovitskiy et al. (2020) ont été impressionnants. Ils ont mis en évidence que le modèle ViT était capable de rivaliser avec les modèles CNN les plus avancés sur plusieurs tâches de reconnaissance d'images. Cela suggère que l'architecture des transformateurs, initialement conçue pour le traitement du langage naturel, peut être efficacement utilisée comme une architecture autonome pour la détection d'images.

1.4 Détections d'infrastructures défectueuses sur les routes

Dans cette section 1.4, des exemples de modèles de détection appliqués à la détection d'infrastructures défectueuses sur les routes sont abordés.

1.4.1 Exemple de méthodes de détections de nids de poule

Assurer l'entretien et la sécurité des routes est un enjeu majeur présent dans de nombreux pays du monde. D'autant plus quand il s'agit de l'état de la chaussée en elle-même. C'est pourquoi de nombreuses études ont été consacrées à leur détection. Prenons un exemple commun au Québec, les nids de poule.

Dans la littérature, les travaux de Kim & Ryu. (2014) et Kim et al. (2022) s'accordent à distinguer 3 types d'approches développées récemment pour les nids de poule :

- Les méthodes basées sur les vibrations : plusieurs études comme celle de Yu *et al*. (2006) utilisent les données d'accéléromètre provenant de capteurs embarqués pour détecter des nids de poule. D'autres utilisent même des accéléromètres de téléphones intelligents (Mednis et al., 2011). Ces méthodes requièrent un espace de stockage réduit et peuvent être utilisées pour un traitement en temps réel. Cependant, il est possible que les principes basés sur les vibrations fournissent des résultats erronés. Notamment lorsqu'ils détectent les articulations et les jonctions de la route comme étant des creux. Tandis que les creux situés au centre d'une voie ne peuvent pas être détectés par les accéléromètres puisqu'ils ne sont pas heurtés par les roues du véhicule (Erikson *et al*., 2008).

- De nombreuses méthodes de reconstruction 3D existent aux sources technologiques variées. On retrouve quelques études basées sur l'utilisation des numériseurs 3D laser comme dans les études de Li *et al*. (2009). Cette méthode se révèle efficace pour la détection en temps réel de nids de poule, mais le coût reste élevé pour de l'utilisation embarquée comme sur des voitures. Dernièrement, plusieurs études se basent, plutôt sur la stéréovision. Cette technologie est la base de la photogrammétrie et de la reconstruction 3D, utilisant une paire d'images aux angles de vue distincts. Cette paire d'images permet, grâce à la notion de stéréovision, de reconstruire un modèle 3D de la scène. Cette technologie est associée notamment à des méthodes d'apprentissage

profond dans Dhiman *et al.* (2020) avec du transfert d'apprentissage sur un R-CNN et un YOLOv2. On retrouve aussi l'utilisation d'un réseau U-Net modifié et une analyse de composantes principales dans Chen et al. (2019). Même si ces méthodes s'avèrent performantes, elles exigent une capacité de calcul élevé pour la reconstruction 3D, ce qui rend difficile leur utilisation en temps réel. De plus, à cause des vibrations dans la voiture, l'utilisation de plusieurs caméras contraint la qualité des résultats si leur alignement n'est pas complètement maitrisé (Kim & Ryu, 2014).

- Les méthodes basées sur la vision : Ces méthodes sont parmi les plus répandues aujourd'hui, puisqu'elles utilisent des technologies plus légères, basées seulement sur des caméras. D'autant plus qu'avec la progression fulgurante de la vision par ordinateur et des méthodes d'apprentissage profond, toutes les attentions sont tournées vers ces méthodes basées sur la vision. On retrouve les méthodes classiques évoquées dans la partie 1.3.2 pour la détection de nids de poule. En effet, Kumar *et al*. (2021) utilise un CNN et Park *et al*. (2021) teste les versions 4 et 5 de YOLO sur la détection de nids de poule. Ces méthodes se révèlent plus performantes que celles basées sur la segmentation.

1.4.2 Jeux de données

En apprentissage profond, un bon jeu de données est primordial pour l'entrainement afin d'espérer atteindre les meilleures performances possibles avec le réseau utilisé. Encore faut-il que nous puissions avoir assez de données. D'ailleurs, c'est bien souvent un problème que nous rencontrons quand nous souhaitons développer des cas particuliers d'application, car les données peuvent être rares, l'annotation de données peut coûter cher, la qualité des jeux de données existants n'est peut-être pas adaptée ou pas assez bonne pour les utiliser. Même si dans de nombreux modèles et suivant la tâche souhaitée, une plus petite quantité de données d'entrainement peut suffire, grâce au « fine tuning » ou « affinage ». C'est-à-dire, un apprentissage par transfert où un modèle préentrainé est adapté à une nouvelle tâche spécifique.

Le constat reste que l'entrainement des réseaux nécessite de vastes ensembles de données annotées par des humains, et la préparation de tels ensembles de données est extrêmement exigeante et nécessite beaucoup de main-d'œuvre (Ma *et al*., 2022).
Reprenons notre cas d'étude sur la détection de nids de poule, voici quelques exemples de jeux de données en accès libre :

- Le « Pothole Image Data-Set » est un jeu de données d'environ 600 images de nids de poule non annotées collecté par un algorithme sur Google et aux résolutions variées. L'auteur notifie que le jeu de données peut contenir des redondances et des images bruitées (www.kaggle.com/datasets/sachinpatel21/pothole-image-dataset, 2019).
- Dans Nienaber et *al*. (2015), un jeu de données annoté de 2658 images à haute résolution (2760x3680 pixels) et capturé par caméra en Afrique du Sud. Il a été créé pour leur étude sur la détection des nids de poule à l'aide de techniques simples de traitement d'images (Kim *et al.,* 2022).
- Fan & Liu, (2020) introduisent un jeu de données annoté appelé « Pothole-600 » contenant 600 images RGB de nids de poule avec une résolution de 400x400 pixels (Kim *et al.,* 2022).

Aussi, dans le cadre du Crowd Road Damage Detection Challenge (CRDDC2022), le jeu de données RDD2022 a été créé. Il comprend 47 420 images de routes provenant de six pays : le Japon, l'Inde, la République tchèque, la Norvège, les États-Unis et la Chine. Le RDD2022 se compose de 55 000 cas de dommages routiers répartis en 4 grandes classes : les fissures transversales (i), les fissures longitudinales (ii), les fissures en forme d'alligator (iii) et les nids de poule (iv) que l'on retrouve en Figure 1.11 (Arya et *al*., 2022).

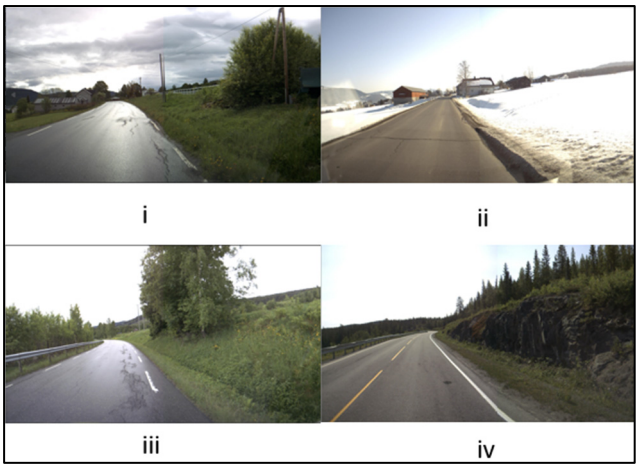

Figure 1.11 Exemples principaux des types de nids de poule
issus de Norvège dans le RDD2022
Tirée de Arya et *al*. (2022)

Finalement, c'est le jeu de données le plus complet en termes d'annotation, de quantité de données et d'informations que j'ai pu trouver dans la littérature et dont le défi CRDDC2022 invite les chercheurs du monde entier à proposer des solutions pour la détection automatique des dégâts routiers dans plusieurs pays.

Le RDD2022 est l'évolution du RDD2018 grâce à la contribution de chercheurs de plusieurs pays (voir la Figure 1.12 et la Figure 1.13).

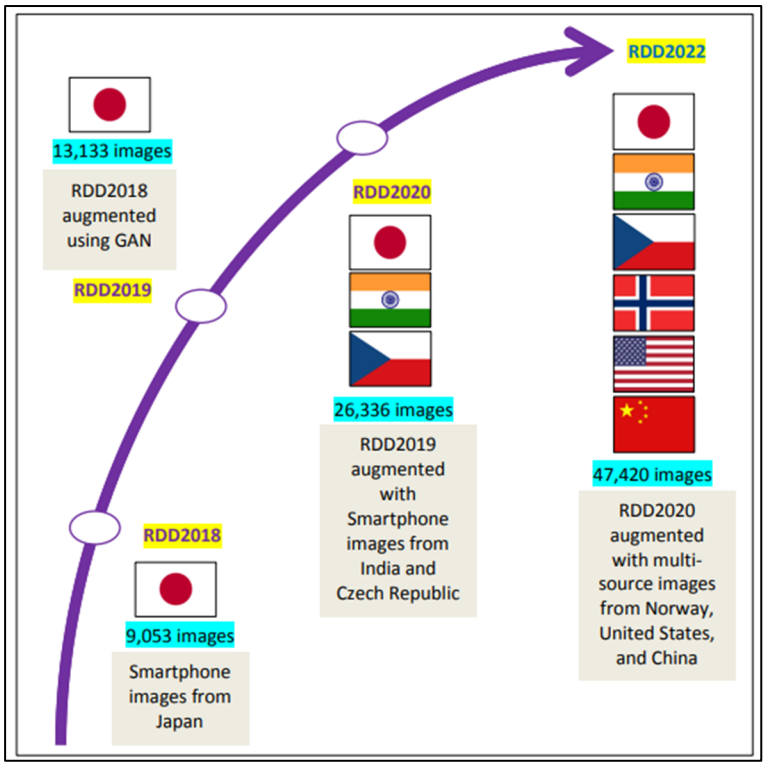

Figure 1.12 Évolution du jeu de données sur les dommages routiers de RDD2018 à RDD2022
Tirée de Arya et al. (2022)

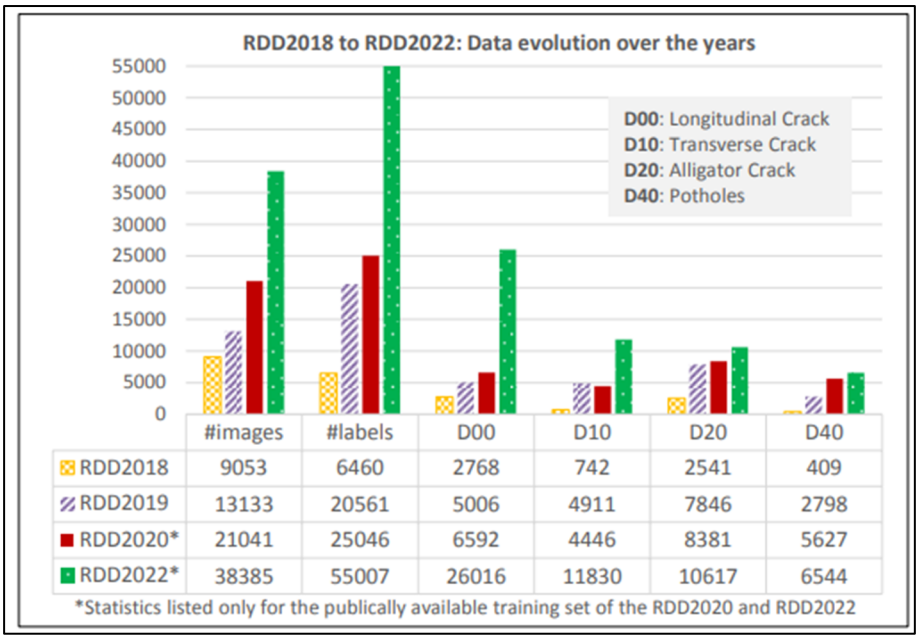

Figure 1.13 Comparaison statistique des ensembles de données
sur les dommages routiers de 2018 à 2022
Tirée de Arya et *al*. (2022)

De plus, dans ce jeu de données, les méthodes de capture d'images diffèrent d'un pays à l'autre (donc la qualité d'image aussi), on retrouve donc :

- Des téléphones intelligents montés sur véhicules au Japon, en Inde et en République tchèque. Ces données ont été uniformisées dans une résolution de 700x700 pixels.
- Des caméras à hautes résolutions sur véhicules en Norvège, permettant d'obtenir des images au format 3650x2044 pixels.
- Des images de Google Street View d'une résolution de 640x640 pixels pour les États-Unis
- En Chine, c'est des images 512x512 pixels récupérés grâce à une caméra embarquée sur une moto et un drone.

Le rapport du concours Arya *et al.* (2022) souligne que la majorité des équipes ont utilisé l'algorithme YOLO, avec des versions différentes. La meilleure équipe a utilisé un ensemble YOLO (pour des prédictions denses) et Faster-RCNN (pour obtenir des prédictions plus

précises grâce à sa structure en deux étapes). Par ailleurs, le rapport décrit leur dernière édition comme la plus haute en termes de difficulté compte tenu de la diversité géographique et de résolution des données.

1.5 Conclusion

Avec les années, beaucoup de modèles et techniques ont été proposés pour identifier les défauts de routes comme les nids de poule. Après l'analyse des différentes technologies à bord des voitures autonomes et les avancées dans le domaine de détections d'objets, les méthodes par vision se révèlent comme être les plus intéressantes, par leur structure peu coûteuse et légère, qui conviennent parfaitement aux systèmes embarqués. Mais aussi par leurs bons résultats.

Mon mémoire étant dédié à l'utilisation d'algorithmes de vision par ordinateur et d'apprentissage profond via l'utilisation d'une caméra embarquée. Je me suis naturellement tourné vers des algorithmes tels que YOLO. La littérature rend compte de l'efficacité et l'utilisation de ce type d'algorithmes pour de nombreux cas. Cependant, le grand défi est d'appliquer la détection automatique à une région particulière comme le Québec. C'est d'ailleurs ce que souligne le rapport du concours Arya *et al.* (2022) : le défis est d'évaluer et d'adapter les modèles à d'autres régions du monde.

Les algorithmes de détection par apprentissage profond requièrent un grand nombre de données d'entrainements (des exemples). Pour pouvoir espérer rendre la détection applicable à des infrastructures routières comme celles du Québec, il faudra des données locales.

CHAPITRE 2

INFRASTRUCTURE ET MÉTHODOLOGIE GÉNÉRALE

Le chapitre 2 aborde la structure et les méthodes générales utilisées dans ce projet. Il permet de comprendre l'utilité de chaque outil dans l'infrastructure déployée.

2.1 Acquisition des jeux de données

L'acquisition du jeu de données est le cœur de mon projet de maitrise. En effet, nous verrons par la suite que la qualité de ce dernier influe directement sur les performances des modèles de détections développés.
De plus, l'acquisition d'exemples annotés est nécessaire dans l'apprentissage supervisé en vision par ordinateur, lorsque l'on utilise des modèles neuronaux profonds. Car pour pouvoir entrainer le modèle, il nous faut des exemples annotés, et dans notre cas, il faut se donner les moyens d'acquérir des exemples puis de les annoter.
Ainsi, cette partie du mémoire aborde la chaine de traitement développée permettant d'acquérir des jeux de données annotés sur les infrastructures des routes au Québec.

2.1.1 Aperçu général

Avant de rentrer dans le détail du processus. La Figure 2.1 permet d'introduire une vue globale de celui-ci, afin de comprendre ce qui a été mis en place et les outils utilisés. Chaque outil composant ce diagramme sera abordé plus en détail dans les parties suivantes.
Voici un cours descriptif du rôle de chaque outil :

- AWS : c'est un service infonuagique permettant notamment le stockage de données.
- CVAT : c'est un outil en ligne d'annotation d'images et de vidéos open source.
- Fiftyone : c'est un outil open source dédié à la construction d'ensemble de données et de modèle de vision par ordinateur.

- Grafana : c'est un outil de visualisation de données.
- PostgreSQL : c'est un système de base de données relationnel-objet open source.
- Module de filtrage : script python développé dans ce projet pour une sélection rapide des images et vidéos d'intérêt.

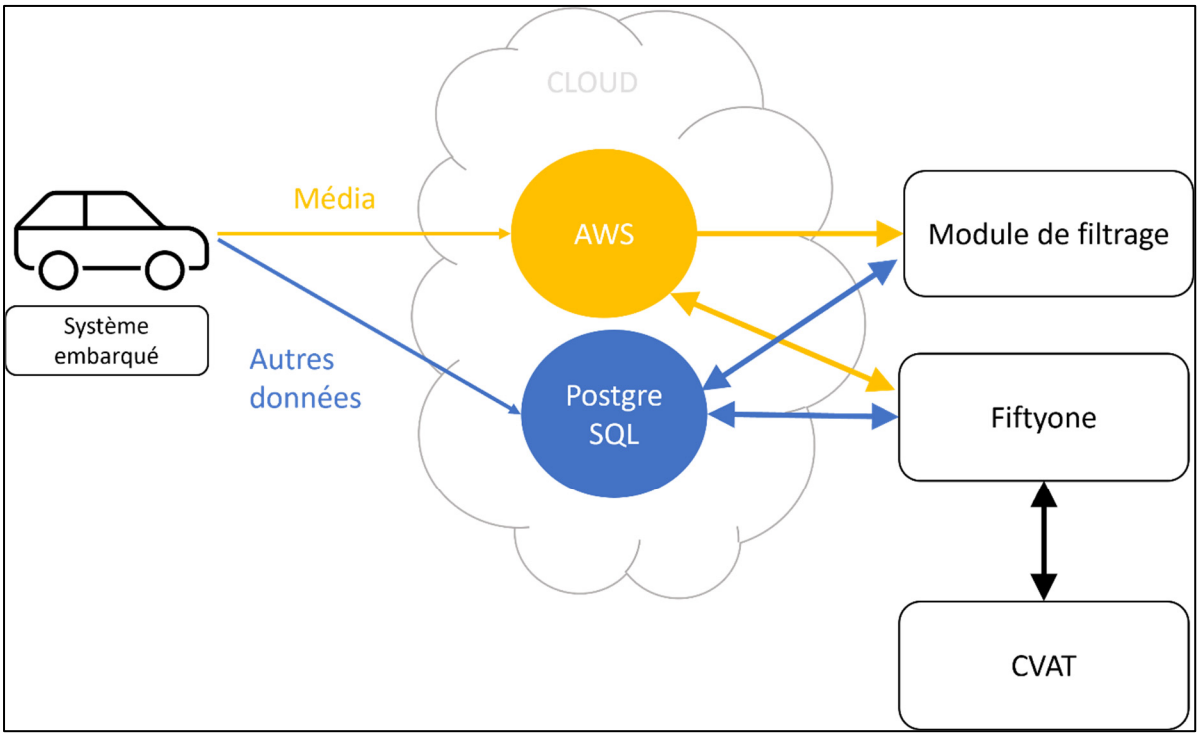

Figure 2.1 Outils et canaux principaux d'interactions

2.1.2 Acquisition des données par système embarqué

Cette section aborde l'équipement du système embarqué et les modes de déclenchement pour la capture de média par la caméra monoculaire.

2.1.2.1 Le système embarqué

Les données recueillies sur les routes du Québec sont prises via les voitures du Ministère du Transport du Québec. Dans le cadre de ce projet, 45 de ces voitures sont utilisées par les opérateurs pour un entretien continu des voiries et de leurs infrastructures.

Dans ces voitures, un système embarqué a été installé. Il comprend 4 composantes essentielles (Figure 2.2) :

- Un ordinateur embarqué NVIDIA Jeston Orin aux capacités de calcul adapté pour l'IA.
- Une caméra SONY STARVIS IMX291 installée dans le pare-brise qui observe la route
- Un modem cellulaire qui permet d'envoyer les données vers l'infonuagique et de recevoir les mises à jour logiciel.
- Un système GPS de haute précision afin de localiser avec fidélité les événements rapportés.

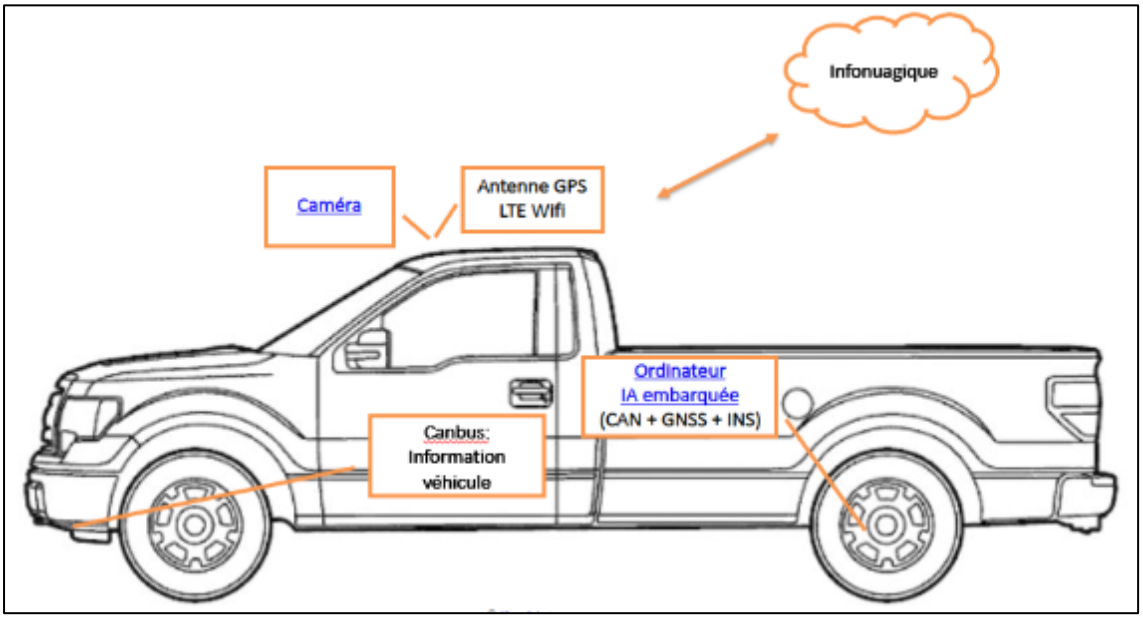

Figure 2.2 Système embarqué dans une voiture du MTQ

La collecte de média via la caméra embarquée est déclenchée par différents modes de capture disponibles qui sont présentés dans la partie 2.1.2.2 suivante. Les médias collectés sont de deux types et dépendent du mode de capture, il en existe deux :

- Des images
- Des vidéos contenues dans un Rosbag, un type de fichier contenant divers types d'information horodatée. Ces vidéos peuvent aussi être aussi convertis en une série d'images que l'on exploite.

Ces médias sont stockés sur l'infonuagique AWS Amazon.

Aussi, plusieurs informations sont récupérées en parallèle des médias pour décrire l'événement, elles sont conservées dans une base de données dont la Figure 2.3 résume la structure.

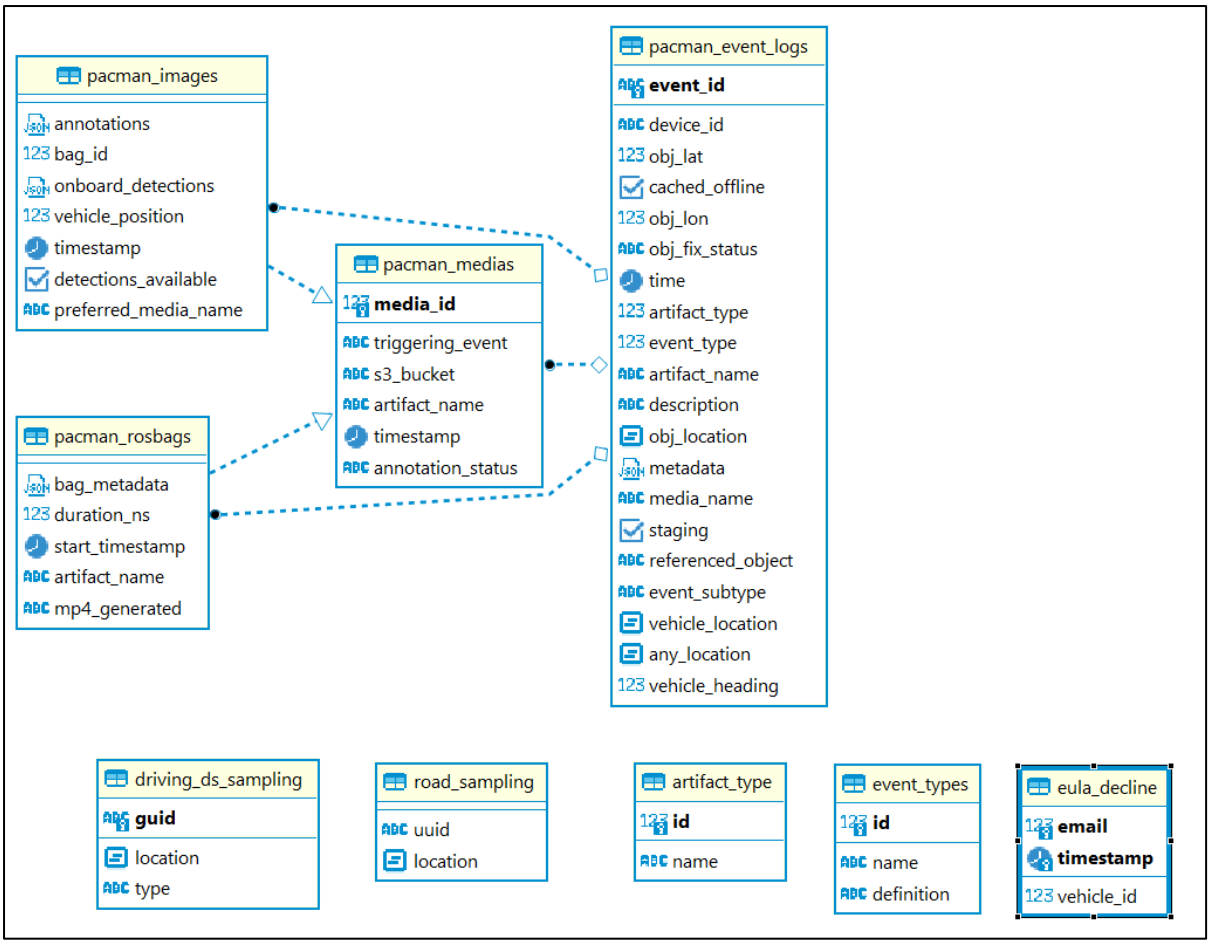

Figure 2.3 Structure et relation des tables des tables de la base de données

Chaque événement capturé est une ligne de la table « pacman_event_log ». On retrouve de nombreuses informations sur l'événement comme :

- La date de capture (colonne « time »)
- La localisation GPS de la voiture qui a capturé le média (colonne « vehicule_location »
- Etc.

Et dans le cas d'utilisation de modèles de détection comme mode de capture, d'autres informations s'ajoutent et que l'on retrouve dans la colonne « metadata » :

- Le taux de confiance de la détection

- La position de l'objet détecté dans l'image (boîte englobante).

De plus, il est aussi important d'introduire les états que nous attribuons aux données dans son cheminement à l'intérieur de la structure en Figure 2.1. Ces états sont appelés « status » dont la colonne de la base de données se trouve dans la table « pacman_event_log ». Ils permettent de connaitre sa position dans la chaine de traitement des données. Nous en définissons cinq :

- « New » : la donnée a été acquise par le système embarqué, mais n'est pas encore traitée.
- « To_do » : la donnée a été sélectionnée et est en attente d'annotation
- « Processing » : la donnée est en cours d'annotations dans l'outils CVAT
- « Dataset » : la donnée est annotée et a rejoint le jeu de données Fiftyone
- « Ignored » : la donnée n'a pas été retenue pour une quelconque raison au cours de la chaine de traitement de données.

2.1.2.2 Modes de captures

Plusieurs modes de déclenchement (ou capture) ont été développés et correspondent à la façon dont nous souhaitons acquérir les données. Ces modes régissent le déclenchement de la caméra pour la capture d'un événement (photo ou vidéo). Les modes de déclenchement principaux disponibles sur les voitures sont :

- Déclenchement par CAN (Controller Area Network): Afin de s'assurer de ne pas interférer avec la communication des différents équipements à bord du véhicule, le système d'acquisition est connecté par un lecteur sans contact au bus CAN du véhicule. De cette façon, il est possible de savoir lorsque des manœuvres intéressantes se produisent, telles que des arrêts brusques et l'activation des freins ABS (Système Anti-Blocage). On y retrouve tous les messages décodés du bus CAN, l'information du GPS ainsi que le flux vidéo. Voici une liste non exhaustive de données transportée par le bus CAN :

  - Accélération longitudinale, latérale et verticale
  - Vitesse du véhicule
  - Vitesse des roues
  - Pression sur le frein
  - Position de la pédale d'accélération
  - ABS
  - Température
  - Clignotants
  - Angle du volant
  - Traction
  - Portes et frein à main
  - Lampes des freins/feux de stops

  Ainsi, toute information brute ou interprétée du bus CAN peut être un motif de capture vidéo, par exemple un coup de frein à une vitesse supérieure à 110 km/h.
- Déclenchement par intervalle de distance
- Déclenchement par intervalle de temps
- Déclenchement par point d'intérêt GPS
- Déclenchement par activation manuelle
- Déclenchement par modèle de détection d'objets développé (nids de poule, lampadaires) (Figure 2.4).

Figure 2.4 Cliché pris par le déclenchement du modèle de détection de lampadaires brisés

2.1.3 Sélection des données

Actuellement, il y a 35 véhicules équipés qui circulent aujourd'hui sur les routes du Québec. Cette flotte combinée aux multiples méthodes de captures disponibles permet d'acquérir une grande quantité d'information. Cependant, pour chaque cas d'usage, seulement une partie des données est intéressante. Par exemple pour les nids de poule, seules les images de jour sont utiles, car la nuit la luminosité n'est pas assez haute pour distinguer les formes au sol facilement. Aussi, dans le souci de former des jeux de données qualitatifs, certaines données paraissent moins pertinentes à prendre en compte que d'autres. Par exemple, comme le montre la Figure 2.5, rien ne peut être interprété à cause de la pluie.

Figure 2.5 Image issue de production

Ainsi, il a fallu préfiltrer nos données avant de les annoter, car elles ne sont pas toute exploitables en fonction de ce que l'on recherche. Un outil que nous avons appelé le module de préfiltrage a été développé pour trier nos données par centre d'intérêt. C'est grâce à des requêtes SQL sur les métadonnées, en relation avec les médias capturés, que nous pouvons facilement filtrer les données (évoqué en 2.1.2).

Voici un exemple en Figure 2.6 de l'interface de l'application. La requête SQL visible sur la figure permet de sélectionner 50 images capturées par le modèle de détection des nids de poule ordonné par date croissante. Après la requête chargée via les bases de données PostgreSQL, une interface comme celle de la Figure 2.7 apparait et permet de choisir les images à garder ou non en les regardant une par une.

En fait, l'application va actualiser son statut dans la base de données : « to_do » si la donnée est retenue et doit continuer le processus de traitement ; « ignored » sinon.

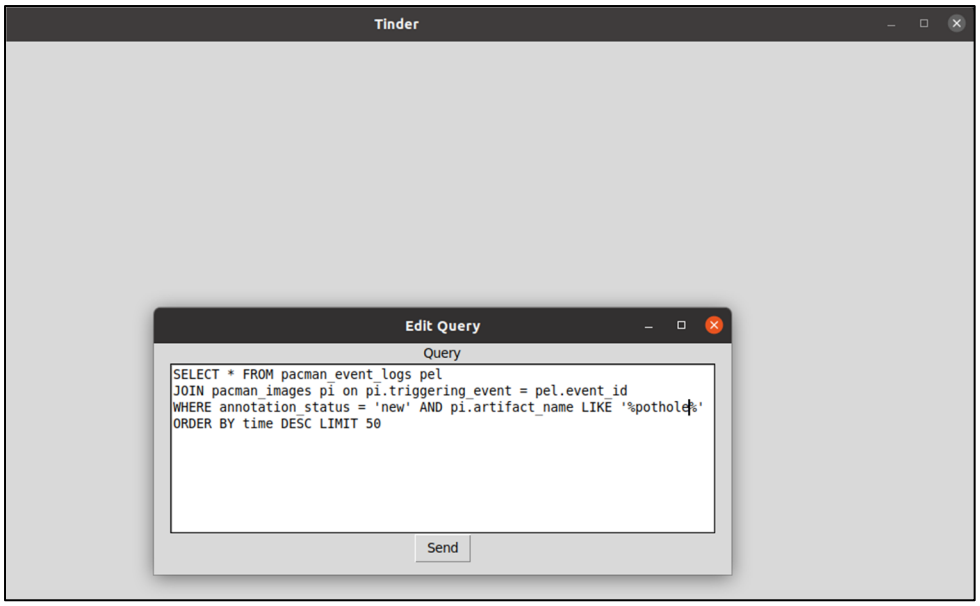

Figure 2.6 Interface requête SQL du module de pré filtrage

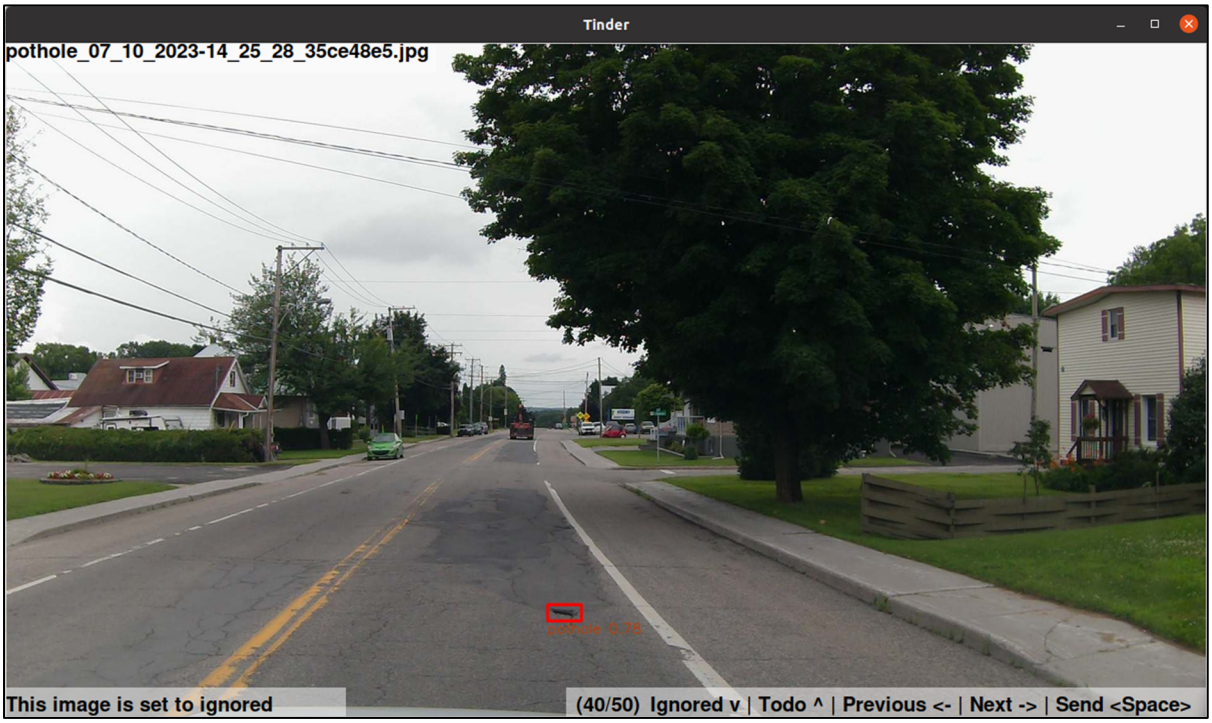

Figure 2.7 Extrait du module de préfiltrage pour la sélection de données

Aussi, la capture résultante par le modèle de détection est sujette à une sélection avant d'être collectée sur le service d'infonuagique. C'est grâce à la bibliothèque python Norfair qui permet un suivi en temps réel des objets détectés. Basé sur un filtre de Kalman (Kalman, 1960), Norfair utilise des techniques de correspondance de caractéristiques (permettant de trouver des points ou des descripteurs communs entre deux images, ce qui facilite l'estimation du mouvement des objets) et d'association d'objets (permettant d'attribuer un identifiant unique à chaque objet et de suivre leur trajectoire de manière cohérente). Les outils proposés par le module sont personnalisables et influent directement sur le choix des données.

En effet, le suivi d'objets permet de sélectionner la meilleure capture du lot, de limiter le nombre de détections emmagasiné et d'éviter les redondances. Dans notre projet, c'est au minimum 3 détections de suite du même objet, pour que nous considérions la détection fiable. Comme nous avons le suivi de l'objet, on peut aussi choisir de prendre comme capture résultante (finale), la première ou la dernière détection faite, mais aussi celle avec le meilleur taux de confiance du modèle par exemple. Je ne maitrisais pas directement ces facteurs dans mon projet, mais il faut bien comprendre qu'ils ont un impact direct sur ce que nous voyons et obtenons du modèle en production.

2.1.4 Annotation des données

Considérons à présent seulement les données que nous avons sélectionnées pour poursuivre le processus. Nous souhaitons les annoter afin de les ajouter à un jeu de données existant ou bien en créer un à partir d'elles.

En vision par ordinateur, annoter signifie délimiter dans chaque image les objets que l'on souhaite détecter. On associe à région d'intérêt délimitée, une classe prédéfinie comme « nid de poule » ou « débris », mais aussi « lampadaire éteint » et « lampadaire allumé » par exemple. Plus généralement, on attribue autant de classes que l'on souhaite distinguer d'objets de nature différente dans l'image.

L'annotation se fait, dans la plupart des cas, à l'aide d'un outil dédié qui permet à l'utilisateur d'ajouter de l'information sur l'image de manière visuelle pour les retransmettre dans le format requis par le modèle.

C'est avec CVAT (Computer Vision Annotation Tool) que nous annotons nos données. Il présente plusieurs avantages :

- Open source: CVAT est un outil librement accessible et modifiable.
- Support multimodal : CVAT prend en charge l'annotation à la fois d'images et de vidéos. Il permet de travailler avec des séquences vidéo et d'annoter des objets dans des cadres individuels ou de manière continue sur plusieurs images clés.
- Annotations multiples : CVAT permet d'annoter de plusieurs façons, par des boîtes englobantes, des masques, des points clés, des contours, etc. Cette flexibilité permet de couvrir une large gamme de tâches de vision par ordinateur.
- Auto-annotation : CVAT permet l'intégration de modèle de détection et classification d'apprentissage profond pour faciliter l'utilisateur dans l'annotation grâce à des l'intelligence artificielle. Par exemple dans notre cas, l'algorithme de détection YOLO a été utilisé pour nous aider à annoter
- Collaboration facilitée : CVAT offre des fonctionnalités de collaboration avancées, permettant à plusieurs annotateurs de travailler simultanément sur un même projet. Les utilisateurs peuvent partager des tâches, réviser les annotations des autres annotateurs et communiquer efficacement au sein de la plateforme.
- Contrôle de la qualité : CVAT propose des mécanismes pour vérifier et valider la qualité des annotations. Les modèles d'annotation personnalisés peuvent être définis pour maintenir la cohérence et la précision des annotations.
- Extensibilité : Il offre une API qui permet aux développeurs d'intégrer CVAT avec d'autres architectures, c'est le cas pour Fiftyone par exemple.
- Formats d'exportation flexibles : CVAT prend en charge l'exportation des données annotées dans plusieurs formats, ce qui facilite l'intégration avec d'autres outils (https://www.cvat.ai/).

Dans le cadre du projet, l'annotation a été essentiellement assurée par mes soins. À la suite des premiers modèles développés, j'ai eu recours aussi à l'auto-annotation à l'aide de nos modèles YOLO développés, pour faire une prédétection des points d'intérêt dans l'image. Cela nous a permis de gagner du temps précieux, puisque j'estime à plus de 10000, le nombre d'images que j'ai dû annoter/réannoter dans le cadre de ce projet sur CVAT.
Cet outil aide à l'annotation sans pour autant être une source très fiable. Puisque ses performances dépendent directement du modèle utilisé. Cependant, cet outil reste très intéressant puisqu'il permet de tester nos propres modèles sur de futures données qui serviront pour l'entrainer.

2.1.5 Jeux de données et visualisation

Après avoir été annotées, les données doivent être organisées dans des jeux qui permettront d'entrainer des modèles de détections. C'est là qu'intervient le dernier outil Fiftyone.

FiftyOne a été développé par la société Voxel51, une entreprise spécialisée dans l'analyse des données visuelles et la vision par ordinateur. Fondée en 2017, Voxel51 s'est concentrée sur la création de technologies et de solutions avancées pour exploiter le potentiel des données visuelles. Fiftyone est une plateforme d'exploration et d'analyse de données en vision par ordinateur qui vise à simplifier et à améliorer le processus d'entrainement des modèles de détection. Elle offre des fonctionnalités avancées pour l'annotation, l'exploration des données, l'évaluation des modèles et la collaboration, ce qui en fait un outil puissant et polyvalent pour les chercheurs et les praticiens de la vision par ordinateur (Figure 2.8) :

- Annotation précise et flexible : FiftyOne permet d'annoter des images avec différents types d'annotations, tels que des boîtes englobantes, des masques, des points clés, etc. L'interface d'annotation intuitive facilite le processus d'annotation en garantissant la précision et la cohérence des annotations. De plus, il offre une flexibilité pour ajouter des annotations personnalisées afin de répondre aux besoins spécifiques de votre projet.

- Exploration des données : La plateforme FiftyOne propose des outils puissants pour explorer et visualiser les données annotées. Il est facile de naviguer à travers les échantillons, visualiser les annotations, filtrer les données en fonction de critères spécifiques et obtenir des statistiques détaillées sur votre ensemble de données. Cela facilite l'analyse et la compréhension des données, permettant de prendre des décisions lors de l'entrainement des modèles.
- Évaluation des modèles : FiftyOne propose des métriques d'évaluation prédéfinies et personnalisables pour mesurer les performances de vos modèles de détection. Nous pouvons évaluer nos modèles sur des ensembles de données de test, visualiser les résultats d'évaluation sous forme de matrices de confusion, de courbes ROC, etc. Cette fonctionnalité permet de suivre et de comparer les performances de différents modèles, d'identifier les points faibles et de prendre des mesures pour les améliorer.
- Collaboration et partage des résultats : FiftyOne facilite la collaboration entre les membres de votre équipe. Il permet de partager des ensembles de données annotées, des modèles entrainés et des résultats d'évaluation avec des collègues, ce qui favorise la reproductibilité des expériences et la collaboration efficace.
- Intégration avec les outils existants : FiftyOne s'intègre facilement avec d'autres outils et bibliothèques populaires de vision par ordinateur, tels que TensorFlow, PyTorch, OpenCV, etc.
- Suivi des expériences et reproductibilité : FiftyOne permet de suivre les expériences d'entrainement des modèles, y compris les hyperparamètres, les métriques d'évaluation et les configurations de données. Cela facilite la reproductibilité des expériences et permet de revenir sur des configurations précédentes pour analyser et comparer les résultats.

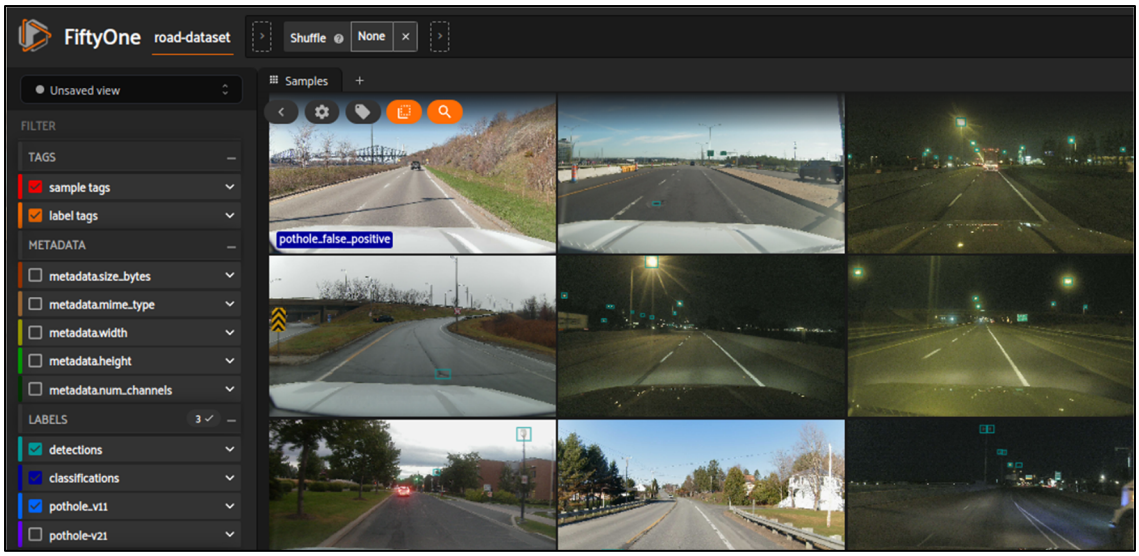

Figure 2.8 Aperçu de l'interface Fiftyone sur nos jeux de données

2.2 Vision par ordinateur : détection d'infrastructures routières défectueuses grâce aux réseaux neuronaux profonds

Cette section, a pour but d'aborder les modèles d'apprentissage profond choisi dans le projet, tout en se plongeant un peu plus en détail dans leur histoire et leur fonctionnement. L'algorithme de base que nous utilisons pour la détection est You Only Look Once (YOLO) introduit pour la première fois dans Redmon et al. (2016). Actuellement, cet algorithme est reconnu pour être l'un des plus performants en termes de détection d'objets. En effet, il se distingue par son équilibre entre vitesse et précision permettant une détection rapide et fiable d'objets dans l'image.

2.2.1 Base de l'algorithme de détection YOLO

L'algorithme YOLO est présenté en 2016 par Redmon et al. comme une nouvelle approche dans le domaine de la détection d'objets. En effet, sa composition et son principe de fonctionnement sont novateurs.

De manière générale, l'architecture YOLO se divise en trois parties principales : la colonne vertébrale (backbone), cou (neck) et tête (head) (voir Figure 2.9).

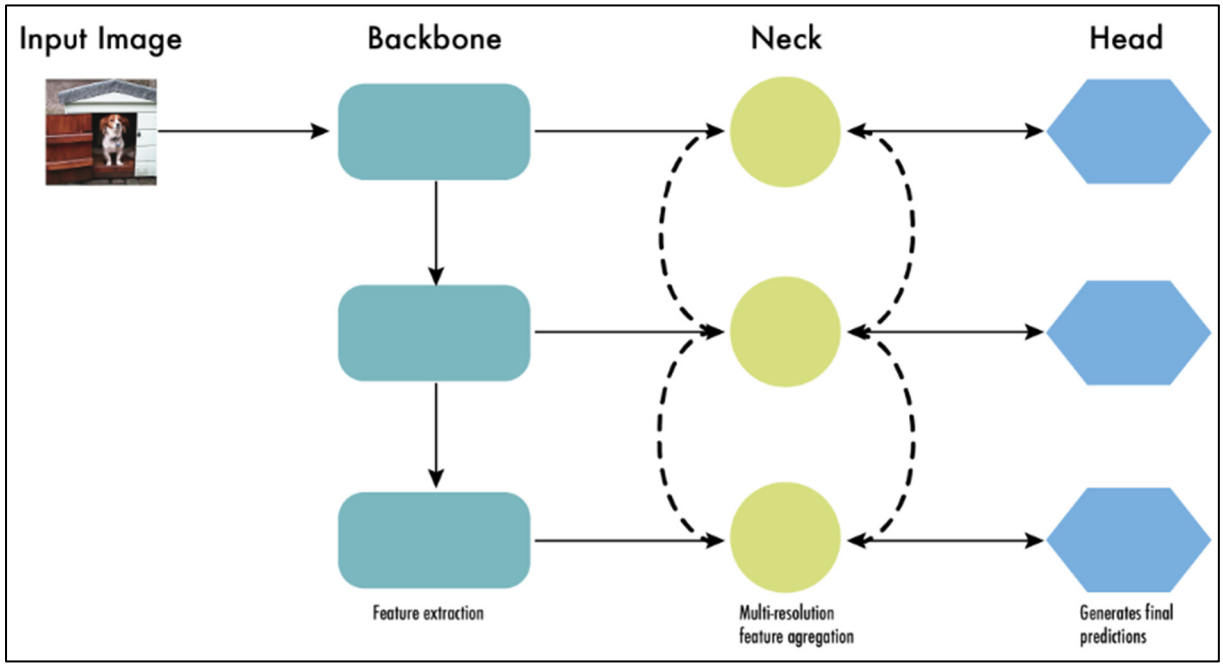

Figure 2.9 Description de l'architecture d'un détecteur d'objets moderne en tant que colonne backbone, neck et head
Tirée de Terven & Cordova-Esparza (2023)

Le réseau backbone constitue la base du modèle de détection d'objets. Il est responsable de l'extraction de caractéristiques à partir des images d'entrée. En général, ces réseaux sont pré-entrainés sur des tâches de classification d'images à grande échelle, telles que ImageNet (Deng et al., 2009). Ces réseaux backbone jouent un rôle essentiel dans les systèmes de détection d'objets, en fournissant des représentations de caractéristiques riches pour les étapes de traitement ultérieures (Terven & Cordova-Esparza, 2023).

Le cou est un composant intermédiaire qui relie la colonne vertébrale à la tête, facilitant la fusion de caractéristiques à différents niveaux. Le Feature Pyramid Network (FPN) est un modèle couramment utilisé. Le cou permet d'améliorer la représentation des caractéristiques (Terven & Cordova-Esparza, 2023).

La tête de détection d'objets est responsable de la génération de prédictions de boîtes englobantes et de probabilités de classe. Les prédictions sont basées sur les caractéristiques fusionnées provenant du cou.

2.2.2 L'évolution de YOLO

YOLO a connu un développement incrémental par la publication de plusieurs versions. Ses différentes versions ont largement contribué à sa progression, mais aussi au succès grandissant de l'algorithme et à son utilisation populaire aujourd'hui.

2.2.2.1 Jeu de données

Les performances des déclinaisons du modèle YOLO ont toujours été basées sur un jeu de données en particulier : le jeu de données COCO (Common Objects in Context) développé par Microsoft (Lin et al., 2015). Ce jeu de données COCO est une collection couramment utilisée dans le domaine de la vision par ordinateur pour la détection d'objets et la segmentation. Il contient plus de 200 000 images annotées avec 80 catégories d'objets différents, ainsi que des annotations détaillées telles que les boîtes englobantes des objets, les masques de segmentation et les points clés. En voici un extrait en Figure 2.10.

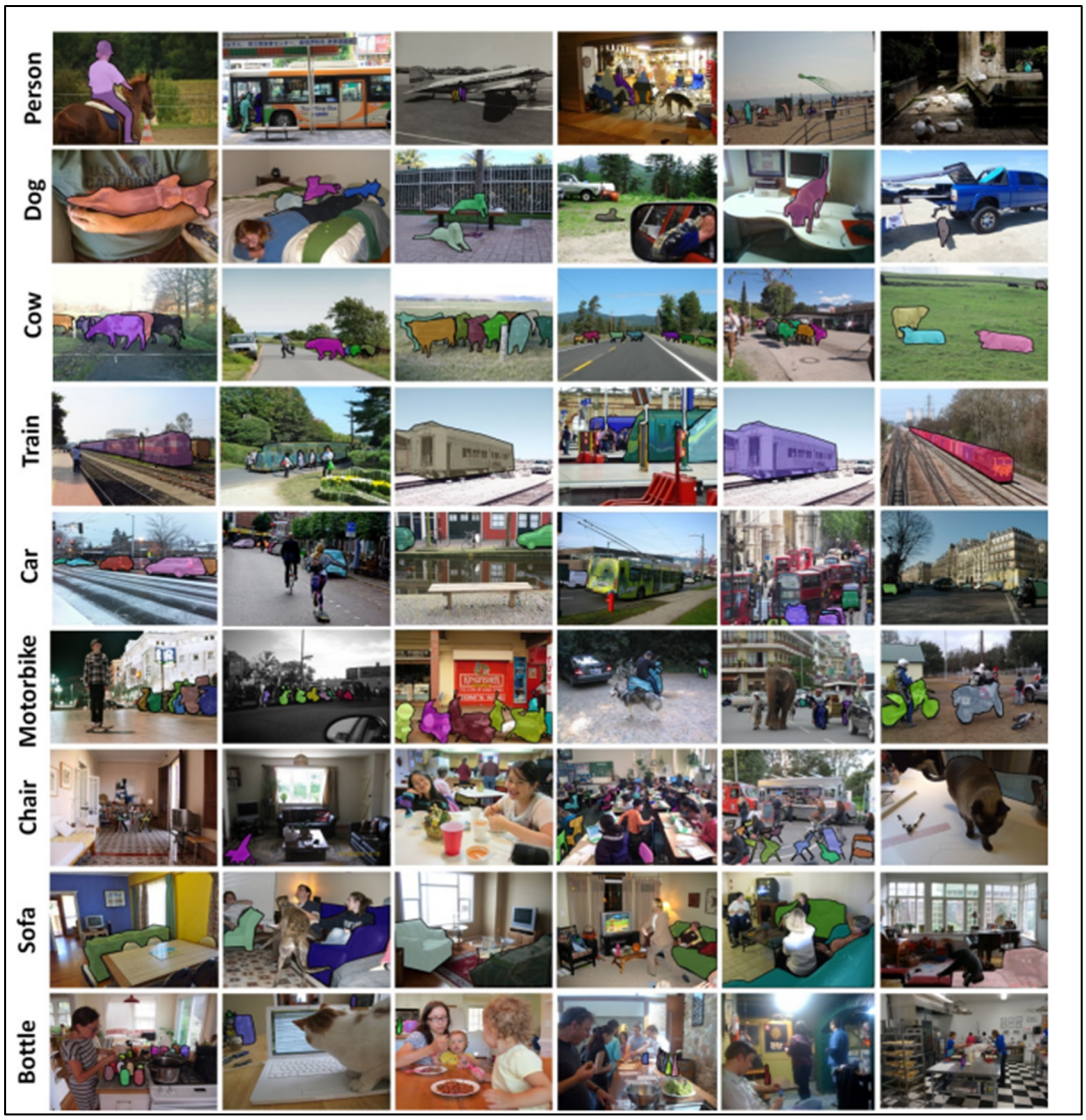

Figure 2.10 Exemples d'images annotées dans l'ensemble de données MS COCO
Tirée de Lin et al. (2015)

2.2.2.2 Versions de YOLO

Redmon et al. (2016) ont été les premiers à présenter le modèle YOLO. Ce modèle révolutionne alors l'approche de la détection d'objets en proposant une approche unifiée qui combine la localisation et la classification des objets en une seule opération. La grosse

difficulté soulignée est sa capacité à détecter de petits objets et des objets proches les uns des autres.

C'est alors qu'en fin d'année 2016, une deuxième version de YOLO est publiée sous le nom de YOLOv2 ou YOLO900 (version renforcée de YOLOv2) (Redmon & Farhadi, 2016). Cette version vise à diminuer le nombre d'erreurs de localisation (trop important par rapport au R-CNN) de YOLOV1. Aussi, elle vise à augmenter le rappel (la capacité du modèle à prédire toutes les boîtes dans une image) tout en maintenant la précision de classification acquise dans l'ancien modèle. Trois des grands changements résident dans :

- L'utilisation de boîtes d'ancrage pour prédire les boîtes englobantes
- L'introduction d'un entrainement à multi-échelles.
- Un nouveau modèle de classification comme backbone de 19 couches de convolution appelé Darknet-19.

En 2018, Redmon et Farhadi présentent une troisième version appelée YOLOv3 : elle est constituée d'une architecture élargie avec des couches de convolution supplémentaires (Darknet-51) ; et utilise trois échelles de détection différentes pour mieux gérer les objets de différentes tailles.

En 2020, Bochkovskiy et al. apportent avec YOLOv4, de nouvelles améliorations qui permettent notamment d'améliorer grandement la vitesse et la précision. On retrouve l'ajout de plusieurs techniques comme celle d'augmentation de données intitulée Mosaïque qui combine 4 images en une pour ajouter du contexte inhabituel aux exemples. D'autres techniques modifiées pour l'architecture sont présentées dans Bochkovskiy et al. (2020) comme:

- L'utilisation d'un réseau modifié PANet (Liu et al., 2018) dans le coup de l'architecture générale.
- Une architecture du backbone complexifié avec un Darknet-53. Elle est combinée à une technique qu'ils appellent CmBN (Cross mini-Batch Normalization).

Peu de temps après le modèle YOLOv5 sort. Aucun papier scientifique ne l'accompagne, même encore aujourd'hui. Cependant, la société Ultralytics qui l'a publié indique que ce modèle est en fait une version implémentée sur Pytorch de YOLOv4 (originalement Darknet) (Jocher et al., 2020). Grossièrement, YOLOv5 reprend les ajouts de YOLOv4 en y intégrant en plus un algorithme appelé AutoAnchor : cet outil de pré-entrainement vérifie les boîtes d'ancrage. Si elles sont mal adaptées à l'ensemble de données et aux paramètres d'entrainement (tels que la taille de l'image), l'algorithme les ajuste (Terven & Cordova-Esparza, 2023).

Par la suite, Terven & Cordova-Esparza (2023) recense 11 versions différentes sorties après YOLOv5 entre 2020 et 2023 (Figure 2.11). On comprend que les changements entre les architectures tournent autour de trois aspects, le type de framework, le réseau utilisé en backbone et l'utilisation d'ancres (anchors).
Alors, concentrons-nous dorénavant sur deux des versions suivantes, YOLOX et YOLOv8, puisqu'elles sont importantes pour la compréhension de mon mémoire.

Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	Darknet	Darknet24	*63.4*
YOLOv2	2016	Yes	Darknet	Darknet24	*63.4*
YOLOv3	2018	Yes	Darknet	Darknet53	36.2
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	YOLOv5CSPDarknet	55.8
PP-YOLO	2020	Yes	PaddlePaddle	ResNet50-vd	45.9
Scaled-YOLOv4	2021	Yes	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	Yes	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	Yes	Pytorch	CSPDarknet	55.4
YOLOX	2021	No	Pytorch	YOLOXCSPDarknet	51.2
PP-YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	YOLOv7Backbone	56.8
DAMO-YOLO	2022	No	Pytorch	MAE-NAS	50.0
YOLOv8	2023	No	Pytorch	YOLOv8CSPDarknet	53.9
YOLO-NAS	2023	No	Pytorch	NAS	52.2

Figure 2.11 Résumé des architectures YOLO
Tirée de Terven & Cordova-Esparza (2023)

YOLOX est introduit dans les travaux de Ge & al. (2021) et prend comme point de départ YOLOv3. Cet algorithme permet de gagner en précision, en rapidité et en simplicité, car il vient notamment casser deux des méthodes utilisées depuis YOLOv2 :

- La détection basée sur les ancres est révolue. En effet, le modèle retourne à une architecture plus simple en se passant de cette technique : le nombre de paramètres à régler est largement diminué.
- Le pré entrainement du classificateur dans le backbone avec ImageNet (Deng et al., 2009). En effet, il est prouvé inutile grâce à l'amélioration du processus d'augmentation de données : l'algorithme MixUp est implémenté (Zhang et al., 2018), en plus de l'algorithme Mosaïque déjà présent dans YOLOv5.

Enfin, Ultralytics dévoile en 2023 le modèle YOLOv8 (Jocher & al., 2023). Ce modèle, supporte d'autres tâches de vision par ordinateur : classification, détection, segmentation, suivi et l'estimation de pose. Il utilise comme point de départ l'architecture backbone du YOLOv5 avec quelques modifications. Cependant il n'utilise pas de la notion d'ancre. De plus, YOLOv8 propose plusieurs améliorations en utilisant de nouvelles fonctions d'estimation de perte (Terven & Cordova-Esparza, 2023). Globalement, le modèle améliore la détection de plus petits objets et donc ses performances en termes de détection d'objets. Toutefois, comme la Figure 2.12 l'illustre, il superforme tous les anciens modèles.

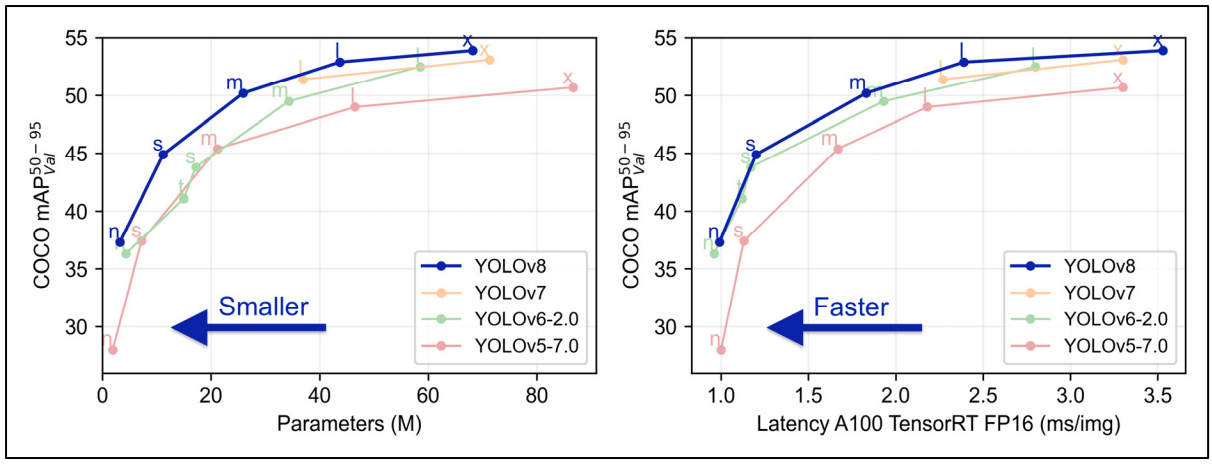

Figure 2.12 Comparaisons des performances pour les derniers modèles YOLO
Tirée de Jocher & al. (2023)

2.2.3 Du principe de YOLO aux méthodes de YOLOv5 et YOLOv8

Le fonctionnement de l'algorithme YOLO peut se décomposer en plusieurs étapes :

- **Prétraitement** : les données sont normalisées pour être compatibles avec l'entrée du modèle.
- **Division de l'image en grille** : Tout d'abord, l'image est divisée en une grille de cellules. Chaque cellule de cette grille est responsable de la détection des objets dont leur centre tombe à l'intérieur de la cellule.
- **Extraction des caractéristiques** : YOLO prédit alors les attributs des boîtes à l'aide d'un module de régression dans le format général suivant dans le cas de détection à une classe :

$$Y = [p_c, b_x, b_y, b_w, b_h, c] \quad (2.1)$$

  Où Y est la représentation vectorielle finale de chaque boîte de délimitation.
  Pour chaque cellule, l'algorithme prédit B boîtes englobantes. Une boîte englobante est un rectangle qui met en évidence un objet dans une image. Chaque boîte englobante comprend les attributs suivants :
  - Le centre de la boîte englobante par ses coordonnées b_x et b_y,
  - sa largeur b_w et hauteur b_h,
  - La classe de l'objet prédit représentée par la lettre c,
  - Le score de probabilité que la cellule contienne l'objet p_c.

Par exemple, la Figure 2.13 illustre un modèle YOLO simplifié avec une grille de trois par trois, trois classes et pour produire un vecteur de huit valeurs.

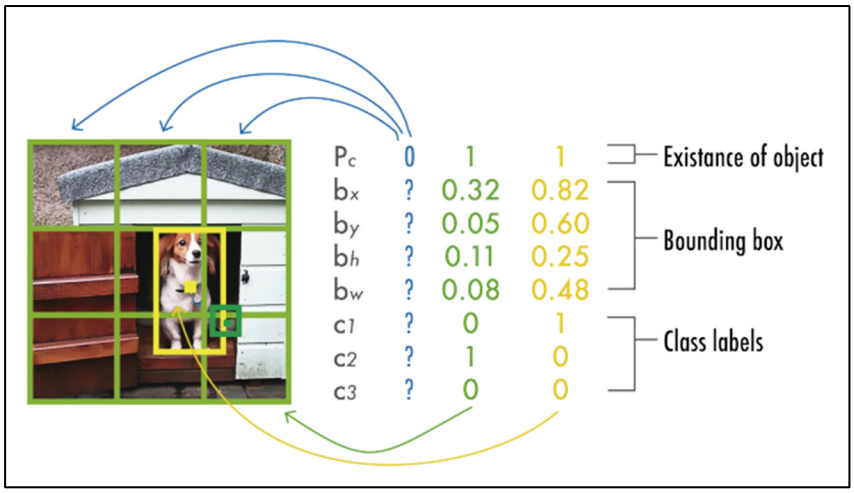

Figure 2.13 Exemple de prédiction simplifié d'un modèle YOLO
Tirée de Terven & Cordova-Esparza (2023)

- **Suppression non maximale** : YOLO utilise la suppression non maximale (NMS) pour ne conserver que la meilleure boîte englobante et éviter les bruits ou doublons comme la Figure 2.14 l'illustre.

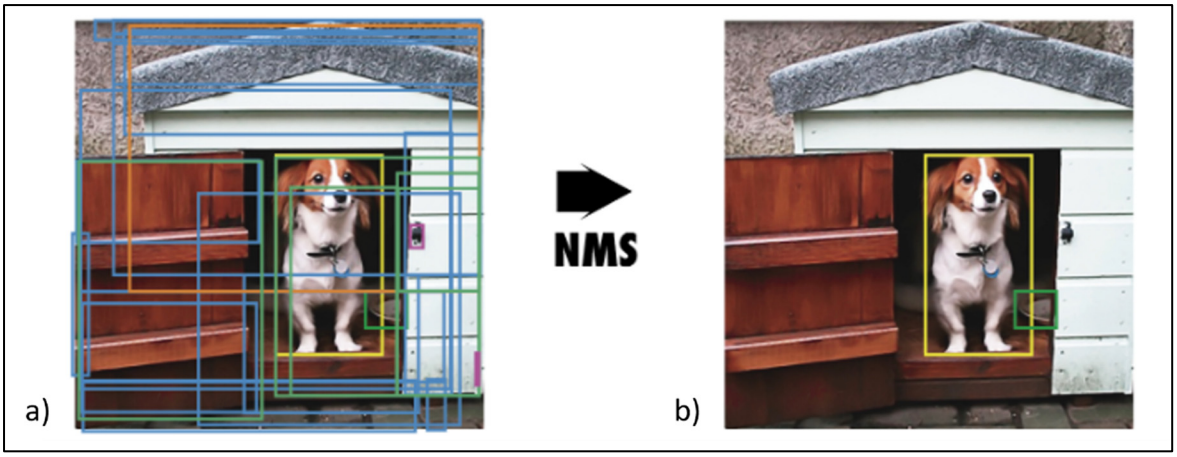

Figure 2.14 Suppression non maximale (NMS). a) montre la sortie typique d'un modèle de détection d'objet contenant plusieurs boîtes se chevauchant. b) montre la sortie après NMS
Tirée de Terven & Cordova-Esparza (2023)

La première étape de la NMS consiste à supprimer toutes les boîtes englobantes prédites dont la probabilité de détection est inférieure à un seuil NMS donné. La deuxième étape utilise l'IoU

pour supprimer les dernières boîtes englobantes qui ne sont pas intéressantes. Un seuil minimum est fixé selon le cas d'utilisation. Toutes les boîtes à l'IoU inférieur à ce seuil ne sont pas gardées.

L'IoU signifie l'intersection sur l'union, c'est une mesure de l'exactitude de la boîte englobante prédite. Le score IoU est un ratio (compris entre 0 et 1). Plus la boîte prédite correspond à la vérité terrain, plus le score est proche de 1 et inversement. Schématiquement, le calcul de l'IoU correspond à la Figure 2.15, dont des cas de figure simplifiés sont représentés pour la compréhension d'un bon score IoU en Figure 2.16.

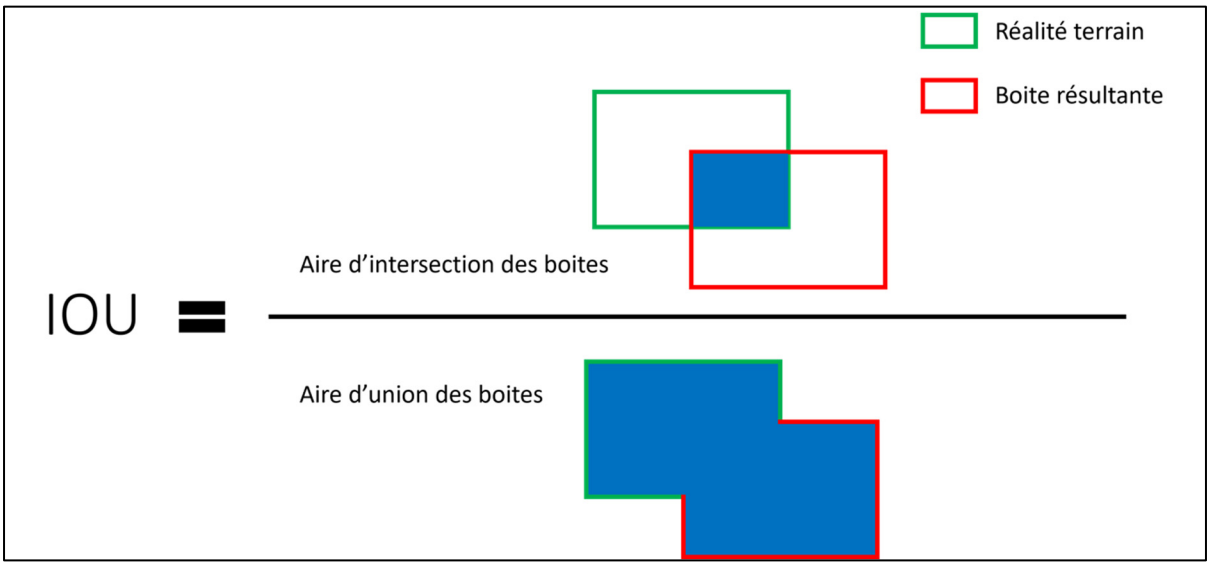

Figure 2.15 Calcul schématisé de l'intersection sur l'union IoU

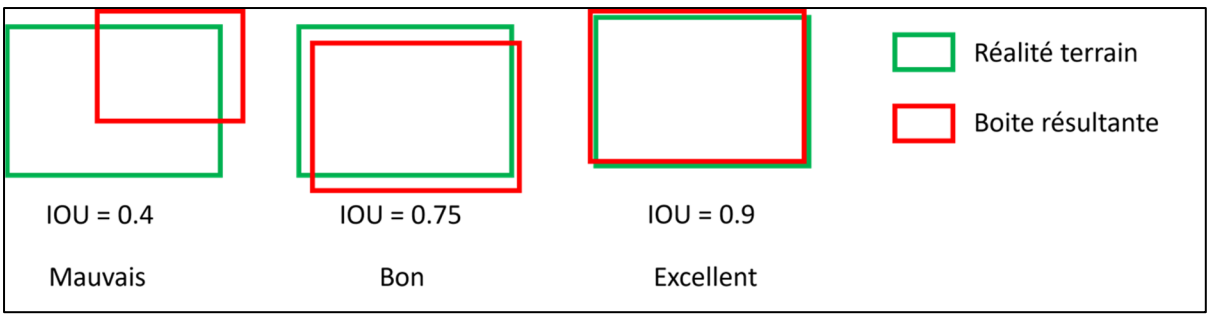

Figure 2.16 Différents cas de figure d'IoU

Pour comprendre comment le calcul de l'IoU fonctionne en pratique, considérons deux boîtes englobantes : une boîte prédite par l'algorithme de détection d'objets (P) et la véritable boîte englobante (V) qui délimite l'emplacement réel de l'objet dans l'image.

Chaque boîte englobante est définie par :

- Les coordonnées de son coin supérieur gauche : (x_1,y_1) pour P et (x_3,y_3) pour V
- Les coordonnées de son coin inférieur droit : (x_2,y_2) pour P et (x_4,y_4) pour V

De même, la boîte définissant l'intersection s'exprime comme un couple de deux coordonnées (x_{inter1},y_{inter1}) et (x_{inter2},y_{inter2}) qui sont respectivement les coordonnées du coin supérieur gauche et du coin inférieur droit. On obtient la configuration de la Figure 2.17 :

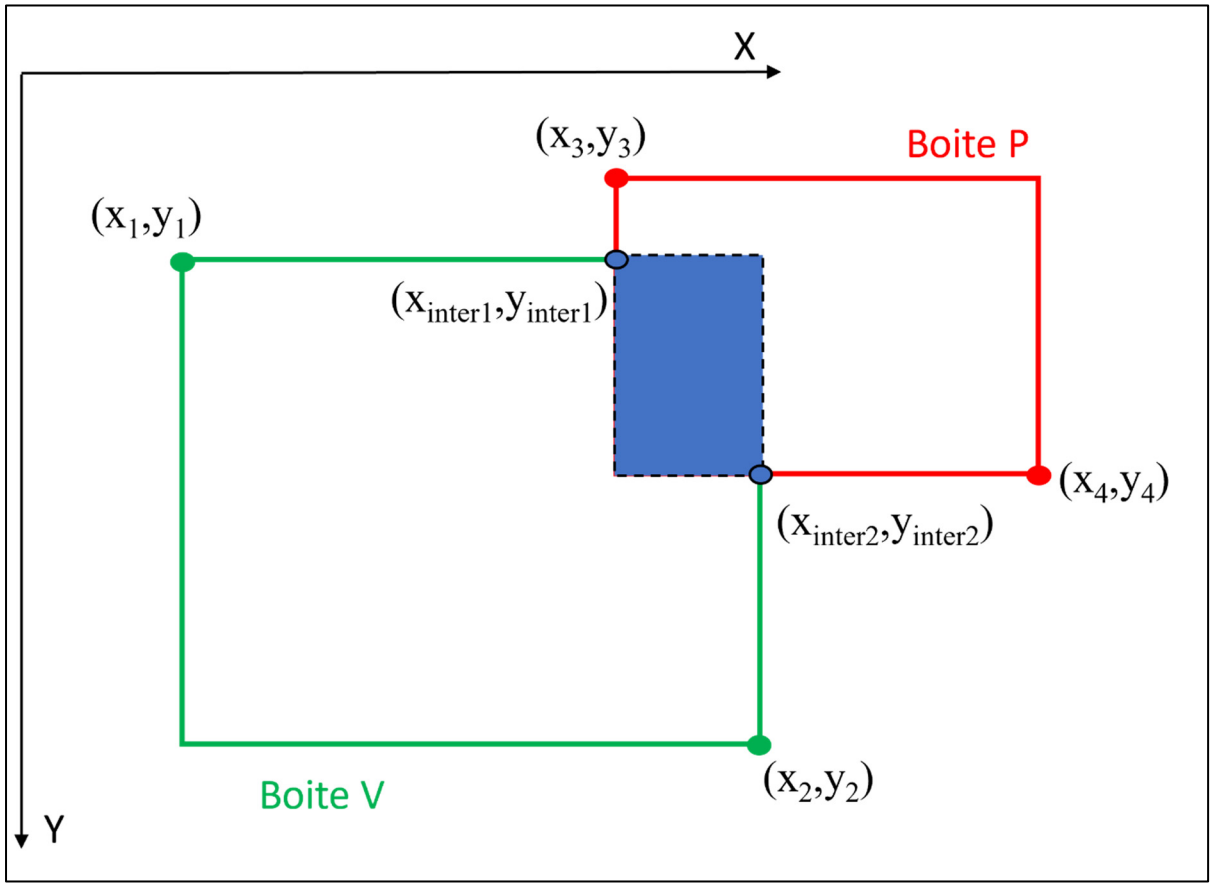

Figure 2.17 Exemple de chevauchement de boîtes

Ainsi, dans le cas où les boîtes se chevaucheraient, les coordonnées d'intersection sont déterminées comme suit :

$$\begin{cases} x_{inter1} = \max(x1, x3) \\ y_{inter1} = \max(y1, y3) \end{cases}$$
$$\begin{cases} x_{inter2} = \max(x2, x4) \\ y_{inter2} = \max(y2, y4) \end{cases} \quad (2.2)$$

L'aire $A_{P\cap V}$ de l'intersection des deux boîtes est alors :

$$A_{P\cap V} = (x_{inter2} - x_{inter1}) * (y_{inter2} - y_{inter1}) \quad (2.3)$$

Puis, l'aire de l'union $A_{P\cup V}$ se calcule à partir de l'aire des boîtes $\mathrm{A_p}$ et A_V:

$$A_{P\cup V} = \mathrm{A_p} + A_V - A_{P\cap V} \quad (2.4)$$

Où :

$$\begin{cases} A_V = (x_2 - x_1) * (y_2 - y_1) \\ A_P = (x_4 - x_3) * (y_4 - y_3) \end{cases} \quad (2.5)$$

Finalement, l'intersection IoU est donc :

$$IoU = \frac{A_{P\cap V}}{A_{P\cup V}} \quad (2.6)$$

2.2.3.1 YOLOv5

À présent, rentrons dans les méthodes principales employées par YOLOv5. Son architecture est décomposée en trois grandes parties : le backbone, le coup (représenté par le PANnet) et la sortie (la tête). Dans la Figure 2.18, nous retrouvons les concepts principaux utilisés : Le BottleNeckCSP, le Spatial Pyramid Pooling (SPP) et le Path Aggregation Network (PANet).

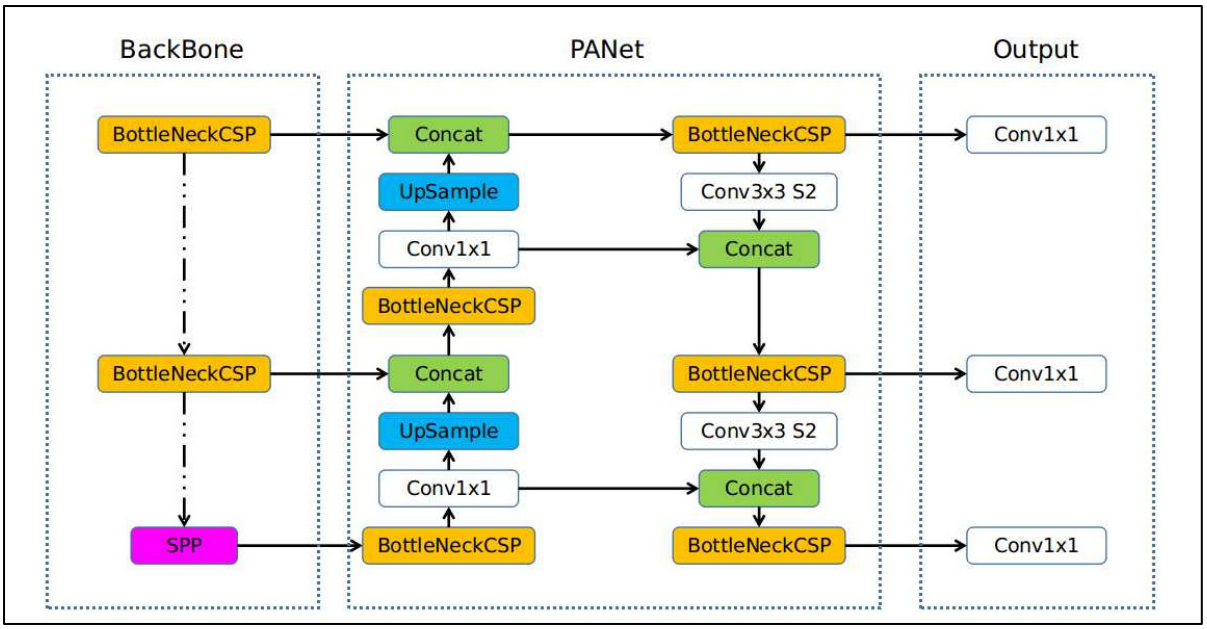

Figure 2.18 Aperçu de l'architecture YOLOv5
Tirée de Ultralytics (2020)

Le CSP-Darknet53 : YOLOv5 a utilisé un réseau Cross Stage Partial (CSPNet) (Wang et al., 2020) pour l'intégrer au Draket53 déjà existant, créant ainsi le CSPDarknet en tant que base de son fonctionnement. L'objectif principal de la conception de CSPNet est de permettre à cette architecture d'obtenir une combinaison de gradients plus riche tout en réduisant la quantité de calcul. Pour ceci, le CSPNet sépare la carte des fonctionnalités de la couche de base en deux parties, pour les fusionner grâce à une stratégie de fusion des caractéristiques (Figure 2.19). Une partie passe par un bloc dense et une couche de transition, tandis que l'autre partie est ensuite combinée avec la carte des fonctionnalités transmise à l'étape suivante. Ainsi, le CSPNet permet de réduire la quantité de calcul, améliorer la vitesse d'inférence ainsi que la précision.

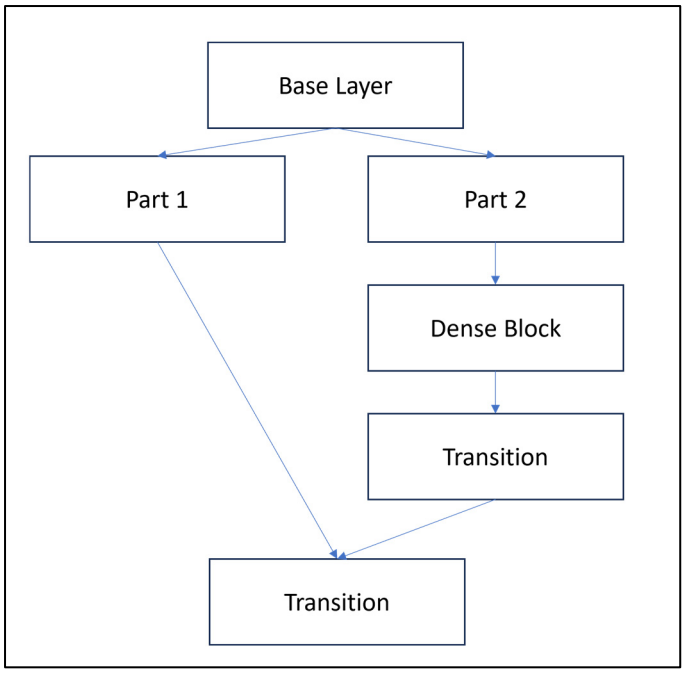

Figure 2.19 Stratégie de fusion des caractéristiques
Adaptée de Wang et al. (2020)

Le Path Aggregation Network (PANet) : Introduit dans (Wang et al., 2019), le PANet est le premier grand changement dans l'architecture du coup (« neck ») de YOLOv5. Son utilisation permet de stimuler le flux d'information et la tâche de segmentation. YOLOv5 modifie le PANet en y intégrant BottleNeckCSP à l'intérieur (voir Figure 2.18) et en adoptant une nouvelle structure de réseau de pyramide de caractéristiques (FPN) avec un chemin descendant. Ainsi, la propagation des caractéristiques de bas niveau du modèle est améliorée.

Le Spatial Pyramid Pooling (SPP) : Introduit dans He et al. (2014)**,** il permet aux réseaux de traiter des images de tailles variables et de générer des caractéristiques de manière invariante à l'échelle. Le SPP fonctionne en divisant la carte des fonctionnalités en plusieurs régions de tailles différentes, puis en appliquant une opération de pooling (réduction des données d'entrée) sur chaque région (Figure 2.20). Cependant, au lieu de spécifier une taille fixe pour le pooling, le SPP utilise des binaires qui s'adaptent à la taille de la région. Cela permet d'extraire des caractéristiques indépendamment de la taille de l'image.

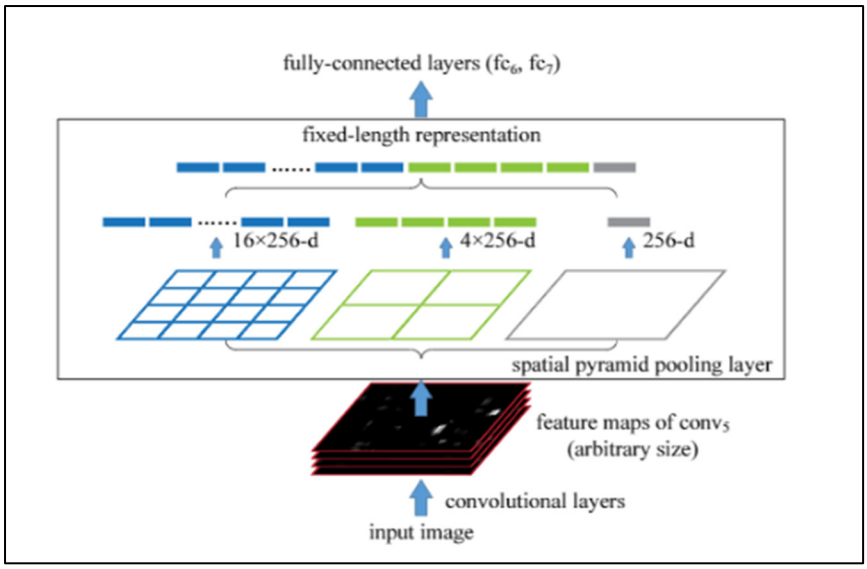

Figure 2.20 Structure d'un réseau avec SPP
Tirée de He et al. (2014)

Après avoir appliqué le pooling sur chaque région, les caractéristiques extraites sont concaténées pour former une représentation globale de l'image. Cette représentation intègre les informations spatiales à différentes échelles, ce qui permet au réseau de capter des motifs de différentes tailles présents dans l'image. YOLOv5 modifie le SPP qu'il renomme SPPF (Spatial Pyramid Pooling Fusion) : contrairement au SPP qui utilise une seule carte des fonctionnalités SPP, le SPPF fusionne les informations de plusieurs cartes des fonctionnalités SPP pour une représentation encore plus riche et complète de l'image.

Pour la dernière partie de sortie, communément appelée la tête, trois couches de convolution sont utilisées. Elles prédisent l'emplacement des boîtes de délimitation, les scores et les classes d'objets. Alors, afin de mesurer l'écart en prédictions et la vérité terrain, YOLOV5 utilise une fonction de perte. En réalité, la perte finale est calculée grâce à la combinaison de trois fonctions de pertes telles que :

$$Loss = \lambda_1 * L_{cls} + \lambda_2 * L_{obj} + \lambda_3 * L_{loc} \quad (2.7)$$

Où :

- L_{cls} est la fonction de perte de classes, c'est une perte d'entropie croisée binaire (BCE loss) qui mesure l'erreur pour la tâche de classification. Dans le cas d'une classification à plusieurs classes :

$$L_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=0}^{C-1}\left[y_{ij}\log\left(p_{ij}\right) + \left(1 - y_{ij}\right)\log\left(1 - p_{ij}\right)\right] \quad (2.8)$$

Où : N est le nombre total de boîtes englobantes, C le nombre de classes d'objets, y_{ij} représente l'étiquette de vérité terrain pour la i-ème boîte englobante et la j-ème classe et p_{ij} représente la probabilité prédite que la i-ème boîte englobante appartienne à la j-ème classe.

- L_{obj} est la fonction de perte d'objectivité, c'est une autre perte d'entropie croisée binaire qui calcule l'erreur de détection de la présence ou non d'un objet dans une cellule particulière de la grille.

$$L_{obj} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i\log(p_i) + (1 - y_i)\log(1 - p_i)\right] \quad (2.9)$$

Où : N est le nombre total de boîtes englobantes, y_i représente le score objet de la vérité terrain pour la boîte englobante et p_i représente le score objet prédit.

- L_{loc} est la fonction de perte de localisation ou la fonction de perte IoU complète, qui mesure l'erreur de localisation de l'objet à l'intérieur de la cellule de la grille.

$$L_{\text{loc}} = 1 - \text{IoU} + \left(c_x^p - c_x^t\right)^2 + \left(c_y^p - c_y^t\right)^2 + (w^p - w^t)^2 + (h^p - h^t)^2 \quad (2.10)$$

Où c_x^p, c_y^p, c_x^t, c_y^t sont respectivement les coordonnées du centre de la boîte prédite et vérité terrain, IoU représente l'Intersection over Union (IoU) entre les boîtes englobantes prédites et de vérité terrain. w^p, h^p, w^t, h^t sont respectivement les largeurs et hauteurs de la boîte englobante prédite et de la boîte vérité terrain.

- λ_1, λ_2 et λ_3 sont des constantes qui permettent de prendre en compte, de manière plus ou moins importante, un aspect de la fonction de perte globale.

L'augmentation de données est un concept permettant d'améliorer la capacité du modèle à se généraliser et à réduire le sur-apprentissage. En général, l'augmentation de données agit directement sur le jeu de données d'entrainement. Dans le cas de la vision par ordinateur, ces algorithmes s'appliquent sur les images. YOLOv5 utilise différentes techniques d'augmentation de données dans son processus d'entrainement, il est d'ailleurs possible de régler l'utilisation de ces techniques via les hyperparamètres du modèle (Figure 2.21).

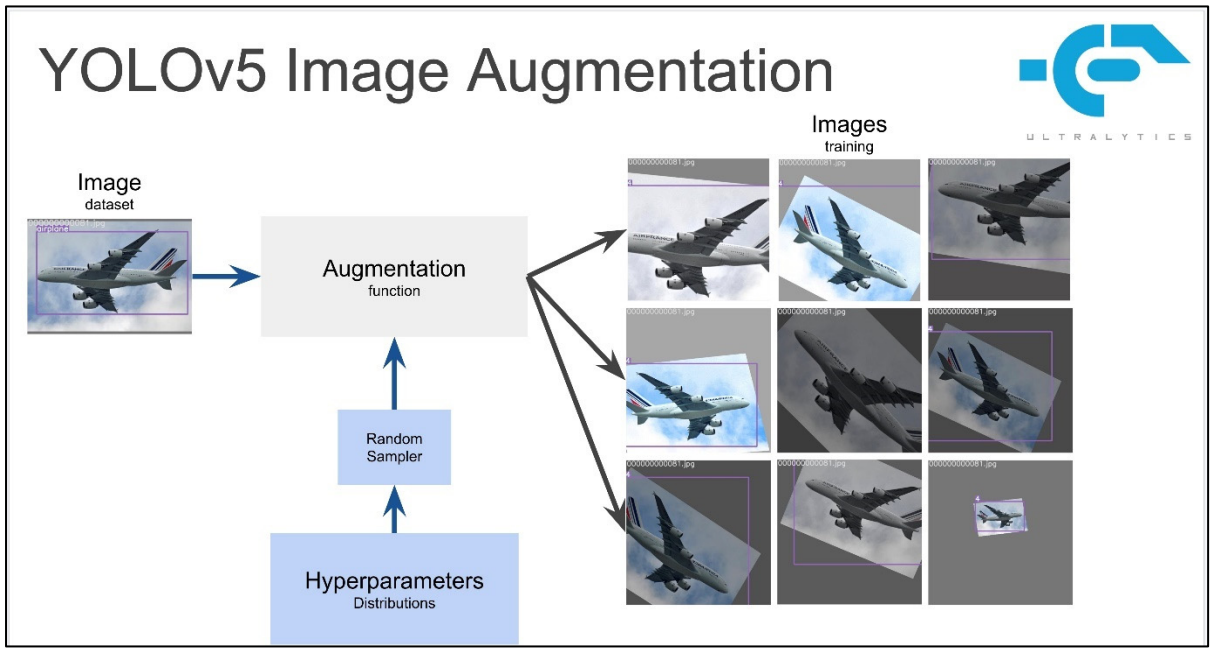

Figure 2.21 Processus d'augmentation des données dans YOLOv5
Tirée de Ultralytics (2022)

Survolons les techniques utilisées avec ce modèle :

- L'augmentation mosaïque combine quatre images du jeu de données pour n'en former qu'une. Elle encourage le modèle à mieux gérer les exemples d'objets à différentes échelles et qui peuvent se retrouver à différentes positions dans l'image (Figure 2.22).

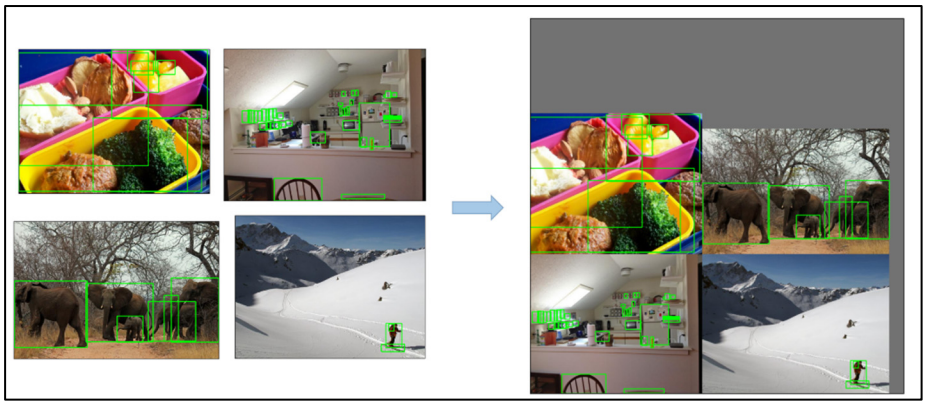

Figure 2.22 Augmentation mosaïque d'un quatuor d'images
Tirée de Ultralytics (2023)

- YOLOV5 utilise aussi les transformations affines, elles peuvent être de plusieurs types : rotation de l'image (Figure 2.23), changement d'échelle (Figure 2.24), décalage (Figure 2.25) et cisaillement de l'image (Figure 2.26).

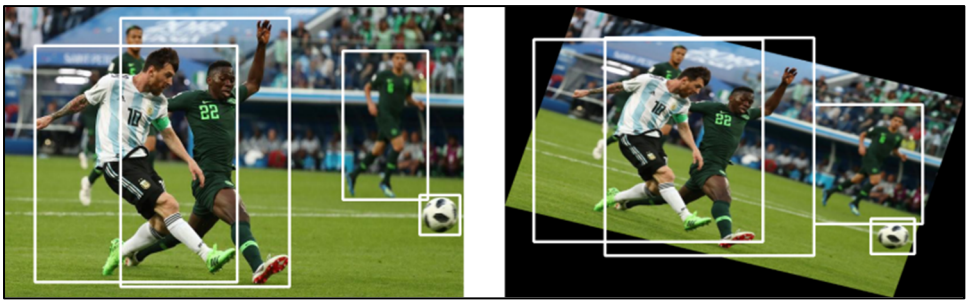

Figure 2.23 Rotation d'une image (gauche : originale, droite : transformée)
Tirée de Kathuria (2018)

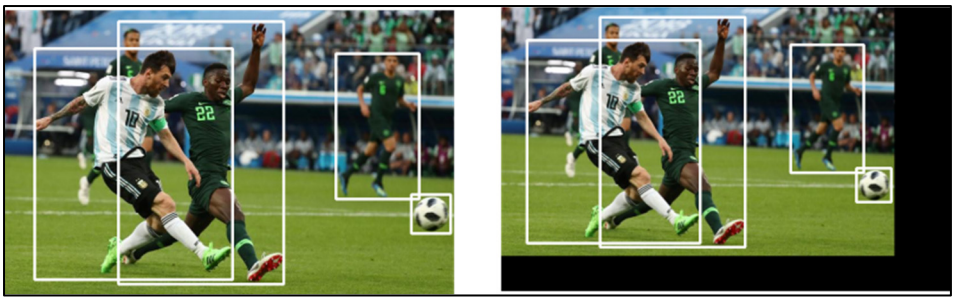

Figure 2.24 Changement d'échelle sur une image
(gauche : originale, droite : transformée)
Tirée de Kathuria (2018)

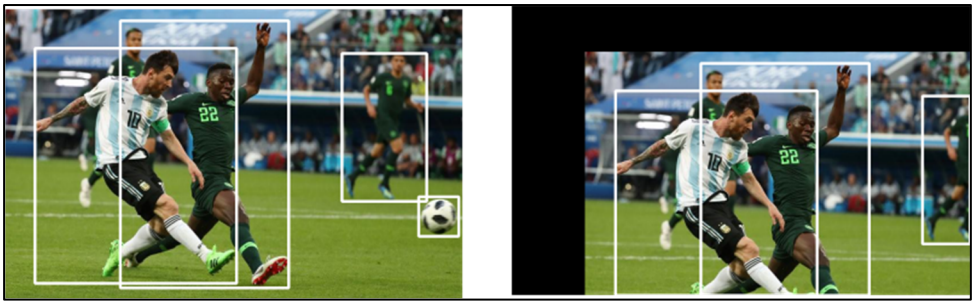

Figure 2.25 Translation/décalage d'une image
(gauche : originale, droite : transformée)
Tirée de Kathuria (2018)

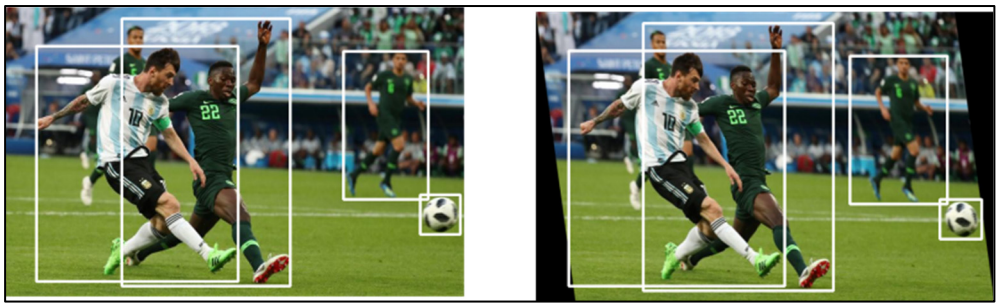

Figure 2.26 Cisaillement d'une image
(gauche : originale, droite : transformée)
Tirée de Kathuria (2018)

- L'augmentation MixUp (Zhang & al, 2018) créée des images en associant deux images et leurs étiquettes associées par l'intermédiaire d'une combinaison linéaire telles que :

$$\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j \qquad (2.11)$$

Où : $\tilde{x}$ représente l'exemple d'entrainement synthétique obtenu grâce au MixUp, x_i et x_j sont deux exemples d'entrainement choisis au hasard dans le jeu de données, λ est un paramètre (compris entre 0 et 1) déterminant le degré de mélange entre les deux exemples (Figure 2.27).

Cette technique d'augmentation aide à agrandir la diversité des données d'entrainement et peut améliorer la capacité du modèle à généraliser, tout comme à traiter des cas difficiles.

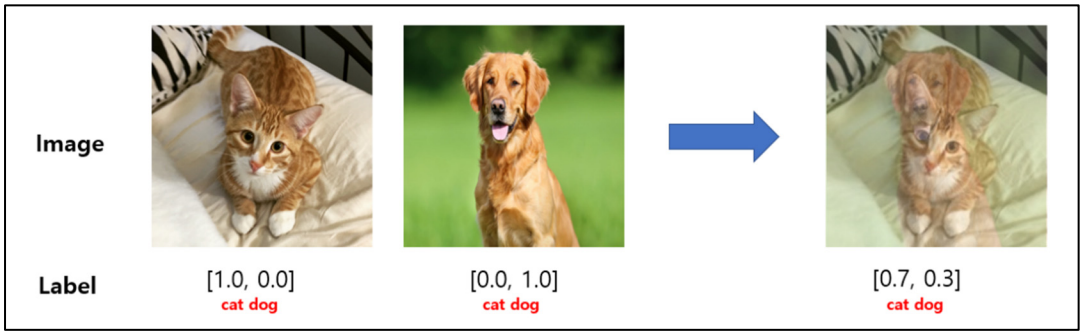

Figure 2.27 MixUp de deux images pour $\lambda = 0.7$
Tirée de Kaggle (2020)

- L'augmentation HSV modifie aléatoirement les couleurs (teinte), la saturation et la valeur des couleurs de l'image (la clarté ou l'obscurité relative d'une couleur) (Figure 2.28).

L'échelle HSV (pour Hue Saturation Value) est une alternative au RGB fournissant une lecture numérique pour chaque pixel de l'image. La teinte est mesurée en degrés, de 0 à 360. Par exemple, le cyan se situe entre 181 et 240 degrés, et le magenta entre 301 et 360 degrés. La valeur et la saturation d'une couleur sont toutes deux analysées sur une échelle de 0 à 100 %. Cette technique d'augmentation peut aider le réseau à devenir invariant en termes de couleurs.

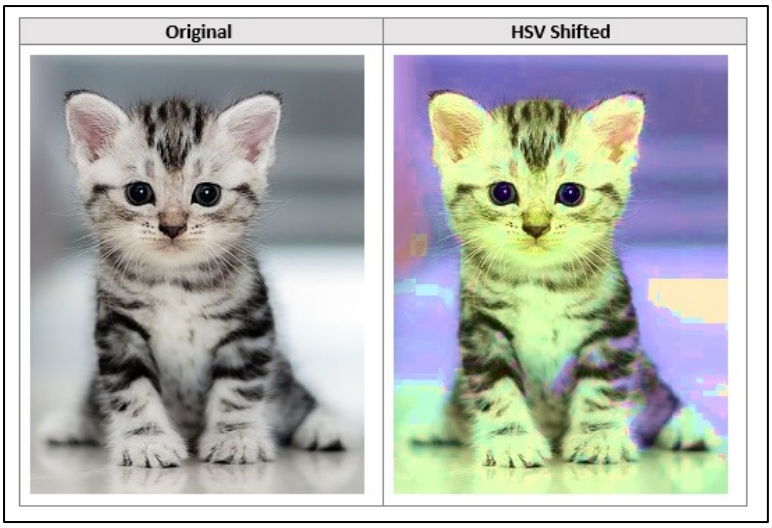

Figure 2.28 Exemple augmentation HSV d'une image
(gauche : originale, droite : transformée)
Tirée d'Affine (2021)

- Le retournement horizontal aléatoire procède à la mise en miroir d'une image. Il tend à rendre le modèle invariant au retournement. Comme cette méthode contient le mot « aléatoire », elle suit une loi de probabilité p. En d'autres termes, l'image transformée peut être identique à l'image originale même si le retournement horizontal aléatoire a été appliqué (Figure 2.29).

Figure 2.29 Utilisation du retournement horizontal aléatoire 5 fois
Tirée de Pytorch (2017)

2.2.3.2 YOLOv8

Comme YOLOv5, YOLOv8 n'a pas de publication scientifique associée. En fait YOLOv8 est considéré comme une série d'amélioration et d'extension apportées par Ultralytics à YOLOv5, qui permet d'atteindre une meilleure vitesse et une meilleure précision. Aujourd'hui, il supporte un large panel de tâches de vision basées sur l'intelligence artificielle : détection, segmentation, estimation de la pose, suivi et classification.

En comparaison à YOLOv5, certaines couches de convolutions dans le backbone et dans le coup ont été modifiées (OpenMMLab, 2023).
Cependant, les changements majeurs se trouvent dans la tête de l'architecture du modèle. La tête est passée de la structure de couplage d'origine à la structure de découplage (Figure 2.30) pour traiter indépendamment les tâches mesurant l'existence d'un objet, de classification et de régression.
Cette conception permet à chaque branche de se concentrer sur sa tâche afin de simplifier et d'améliorer la précision globale du modèle.
Aussi, la détection n'est plus basée sur les boîtes l'ancrage de YOLOv5. En effet, YOLOV8 n'utilise plus e principe, on le dit donc sans ancrage (introduit pour la première fois dans une structure YOLO avec YOLOX de Ge & al. (2021)).
Plus en détail, les modèles de détection basés sur les ancres comme YOLOv5 utilisent des boîtes de délimitation prédéfinies (ancres) comme propositions. C'est-à-dire que ces boîtes ont des tailles prédéfinies auxquelles le modèle prédit des étiquettes de classes et la manière dont l'ancre doit être ajustée pour la faire correspondre aux objets de la vérité terrain ((le cas a) de la Figure 2.31). Ils calculent la boîte englobante d'un objet par régression des décalages par rapport à une boîte d'ancrage. Alors que des modèles comme YOLOX et YOLOv8 prédisent directement des valeurs correspondantes à la position et à la taille d'un objet à partir d'un point donné ((le cas b) de la Figure 2.31).

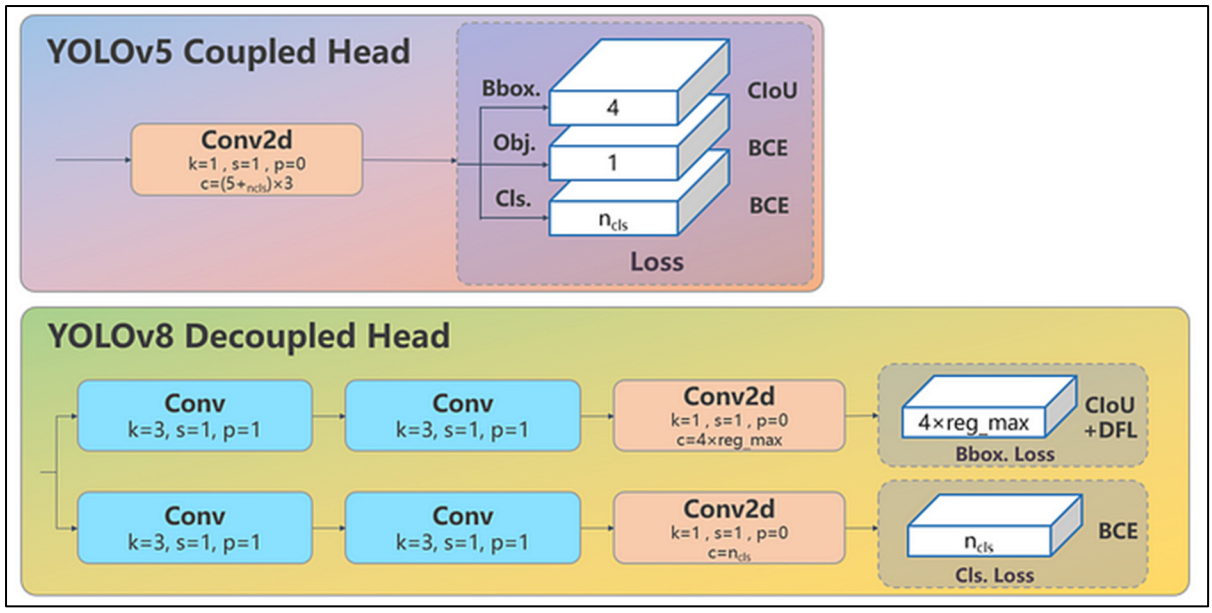

Figure 2.30 Tête couplée de YOLOv5 et tête découplée de YOLOv8
Tirée de OpenMMLab (2023)

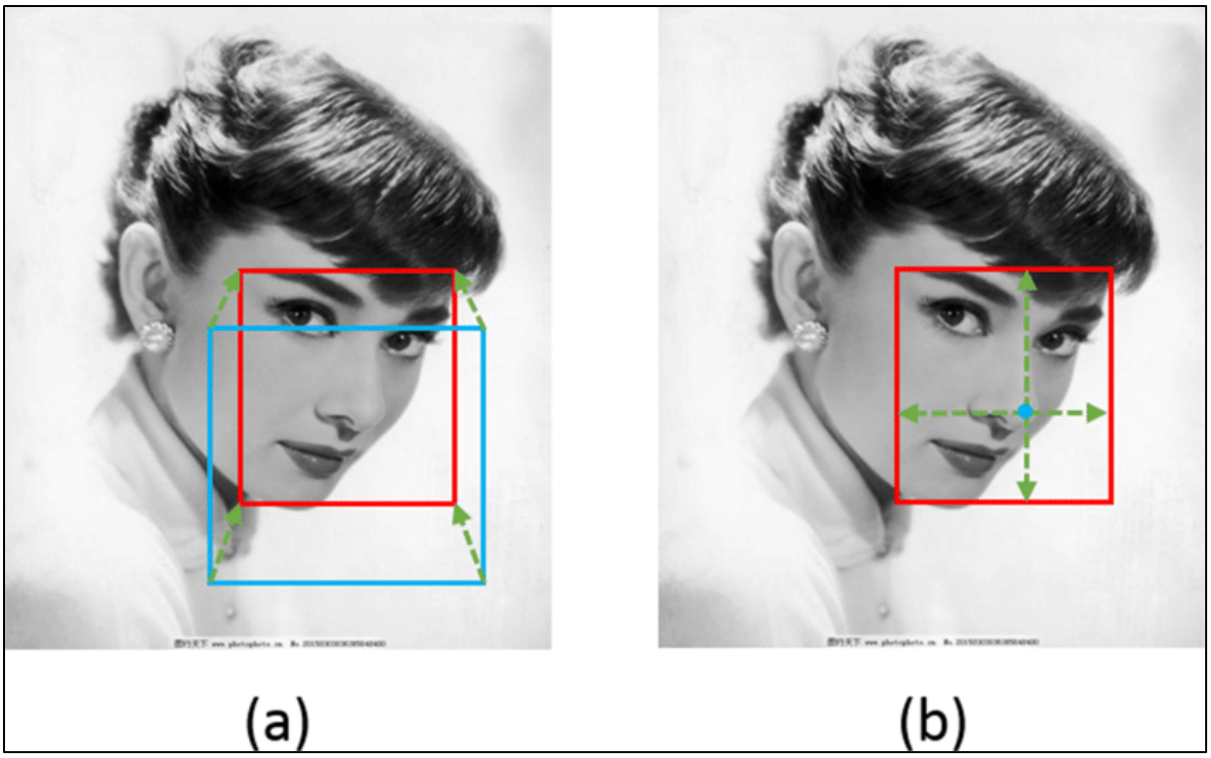

Figure 2.31 Modèle avec ancre (a) et sans ancre (b)
Tirée de Wang et al. (2021)

2.2.3.3 Notre configuration

Dans le cadre de notre projet, c'est particulièrement deux modèles de détection qui ont été utilisés. Ces deux modèles ont été développés par l'entreprise Ultralytics. Le premier YOLOv5 est celui avec lequel la majeure partie de ma recherche a été faite. En effet, le deuxième YOLOv8 n'a été sorti qu'en 2023.

Da manière générale, le choix de ces modèles s'explique par plusieurs raisons :
YOLOv5 et YOLOv8 sont flexibles et personnalisables. Ils proposent cinq tailles de réseaux différents : nano, small, medium, large et Xlarge. Chaque modèle offre un équilibre différent entre vitesse et précision comme présenté en Figure 2.32 pour le modèle YOLOv5 et en

Figure 2.33 pour YOLOv8 (avec plus de détail).

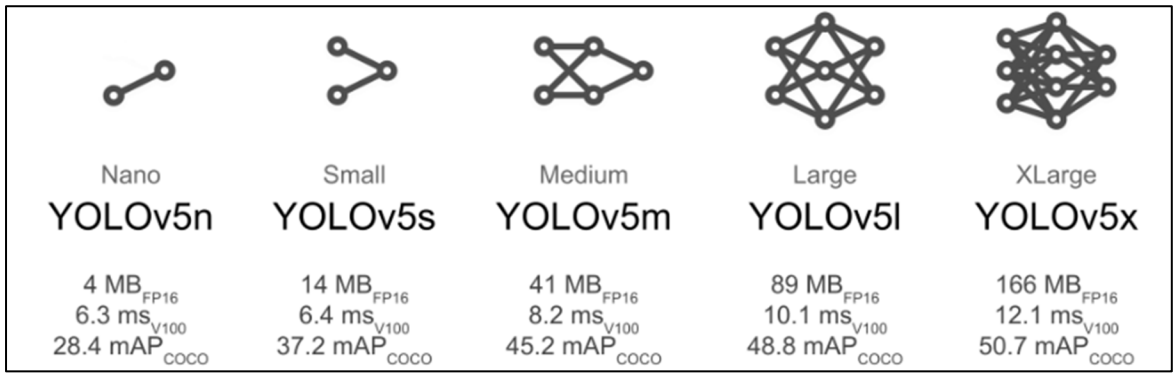

Figure 2.32 Différentes tailles de modèles YOLOV5, où FP16 représente la demi-précision en virgule flottante, V100 est un temps d'inférence en millisecondes sur le GPU Nvidia V100, et la mAP basée sur le set de validation COCO 2017
Tirée de Ultralytics (2023)

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 2.33 Différentes tailles de modèles YOLOv8
Tirée de Ultralytics (2023)

- Ultralytics offre de nombreux avantages via ses répertoires pour l'utilisation des modèles. En effet, le répertoire Ultralytics est conçu pour être facile à utiliser : il offre une documentation complète sur l'utilisation de ses modèles avec des scripts pour l'entrainement et l'inférence, ainsi que pour l'évaluation des modèles. On retrouve aussi l'intégration des outils d'automatisation et de gestions d'expérience comme ClearMl facilitant le processus de mise en œuvre.

- YOLO, par ses versions 5 puis 8, est reconnu comme un des algorithmes les plus performants dans le domaine de la détection d'objets en temps réel, alliant rapidité, légèreté et précision.
- Les répertoires sont des sources ouvertes et une grande communauté contribue à leurs mises à jour via GitHub.
- L'utilisation du « fine-tuning » ou « réglage fin » est rendue possible via l'accès libre des modèles pré-entrainés YOLOv5 et YOLOv8. Cette méthode de transfert d'apprentissage consiste à prendre les poids d'un réseau neuronal entrainé et à les utiliser comme initialisation pour un nouveau modèle de détection. Les avantages sont nombreux :
    - L'utilisation d'un modèle pré-entrainé signifie que le nouveau modèle n'est pas à entrainer de zéro. Cela peut économiser beaucoup de temps et de ressources de calcul.
    - Le réglage fin permet au modèle de s'adapter à de nouvelles tâches qui peuvent être légèrement différentes de la tâche pour laquelle le modèle a été initialement formé.
    - Le réglage fin peut aider à améliorer les performances du modèle sur un ensemble de données spécifique. En ajustant les poids du modèle pré-entrainé sur un nouvel ensemble de données, le modèle peut apprendre à mieux généraliser à partir de ces données.

Les choix de configurations sont multiples et dépendent aussi des équipements, notamment lorsque l'on utilise les modèles en temps réel (i.e. en production). Voici la configuration de base pour l'inférence en production :

- Modèle de détection en production : YOLOv5 médium puis YOLOv8 médium.
- Résolution d'image en entrée du modèle : 576 px par 1024 px.
- Fréquence d'image : 15 FPS.
- Framework utilisé : TensortRT.

Les configurations utilisées en production sont limitées par les capacités des équipements du système embarqué. Alors qu'à l'entrainement, nous avons une plus grande liberté dans les choix des paramètres, car nos capacités de calcul sont notamment plus grandes.

2.3 Conclusion

Dans ce chapitre nous avons pu discuter de ce que nous avons mis en place pour organiser la structure générale du projet. Finalement, il a fallu mettre en place beaucoup d'outils pour gérer toutes les données entourant l'entrainement du modèle et son suivi. Aussi, le processus en amont de l'entrainement est assez important, puisque c'est lui qui permet d'obtenir les données annotées d'intérêt pour l'entrainement. Pour trouver ces données intéressantes, il a fallu établir des stratégies, c'est l'enjeu du prochain chapitre qui s'intitule : CHAPITRE 3 CONSTRUCTION DES JEUX DE DONNÉES.

CHAPITRE 3

CONSTRUCTION DES JEUX DE DONNÉES

3.1.1 Introduction

Dans le cadre de mon projet, il a fallu convenir d'une stratégie pour trouver des exemples sur les objets de mon étude : les nids de poule, les lampadaires brisés et les barrières de sécurités endommagées. En effet, tout au long du projet, l'enjeu a été de trouver les données intéressantes. Ces dernières se trouvent dans une masse d'images et vidéos prises par les véhicules, afin d'entrainer nos modèles de détection. D'autant plus qu'au début, nous n'avions pas de premiers modèles développés capables de détecter une quantité raisonnable d'objets intéressants. Pour commencer à extraire des données intéressantes, il a fallu donc utiliser les méthodes de déclenchement (présentées dans la section 2.1.2.2) avec des méthodes de filtrage post acquisition abordées dans ce chapitre.
Ces données intéressantes seront ensuite annotées puis ajoutées au jeu de données d'entrainement ou de test.

3.1.2 Apprentissage actif

Dans de nombreux problèmes d'apprentissage automatique, les données d'entrainement sont considérées comme une partie fixe dans de la définition du problème. C'est-à-dire que l'on part d'un jeu de données construit en amont, le jeu est complet et n'est pas sujet à de grandes variations d'exemples ou de contexte dans le temps. En fait, le jeu de données dépend directement de la tâche de détection souhaitée et des sources de données accessibles. Cependant, dans notre cas d'étude, nous devons construire ce jeu de données pour l'entrainement du détecteur. Pour remédier à ceci, on choisit donc d'entrainer itérativement un détecteur avec les données que l'on accumule progressivement. Les premières versions de celui-ci permettent de commencer à détecter des exemples intéressants, mais ne font pas de lui un détecteur fiable. En fait, le jeu de données évolue avec le détecteur : l'apprenant à l'occasion

de jouer un rôle dans la décision des données qui seront acquises pour l'entrainement. Ce processus est généralement appelé "apprentissage actif", où l'apprenant est un participant actif dans le processus d'entrainement. (Cohn, 2010)

Dans le cas de ce projet, l'infrastructure des routes change selon les régions et pays, mais aussi pendant les saisons. Aussi, dans une même région, cette infrastructure peut évoluer et se moderniser. Ces facteurs de changements font des routes un environnement très varié. Ainsi, il faut être capable d'obtenir des jeux de données variés, tout en leur assurant un bon équilibre d'exemples, afin d'obtenir de bonnes performances de détection. Cela en tout temps et dans le plus de configurations réelles possible.

Pour ceci, la stratégie d'entrainement en tant qu'apprentissage actif choisi et disponible en Figure 3.1 permet deux choses :

- Un entrainement visé sur les faiblesses du modèle à l'instant t
- Une obtention de données annotée ou à réannoter permettant de faire grossir le jeu de données d'entrainement. En effet, le problème au début de mon projet a été le peu de données disponibles et accessibles simplement, notamment dû au peu de véhicules équipés au départ. En fait, ici le jeu de données s'est construit parallèlement à l'entrainement.

Finalement, cette boucle permet d'espérer que le modèle soit efficace toute l'année, mais aussi qu'il soit alerte de derniers changements en l'entrainant avec des données récentes et les plus proches de la réalité possible.

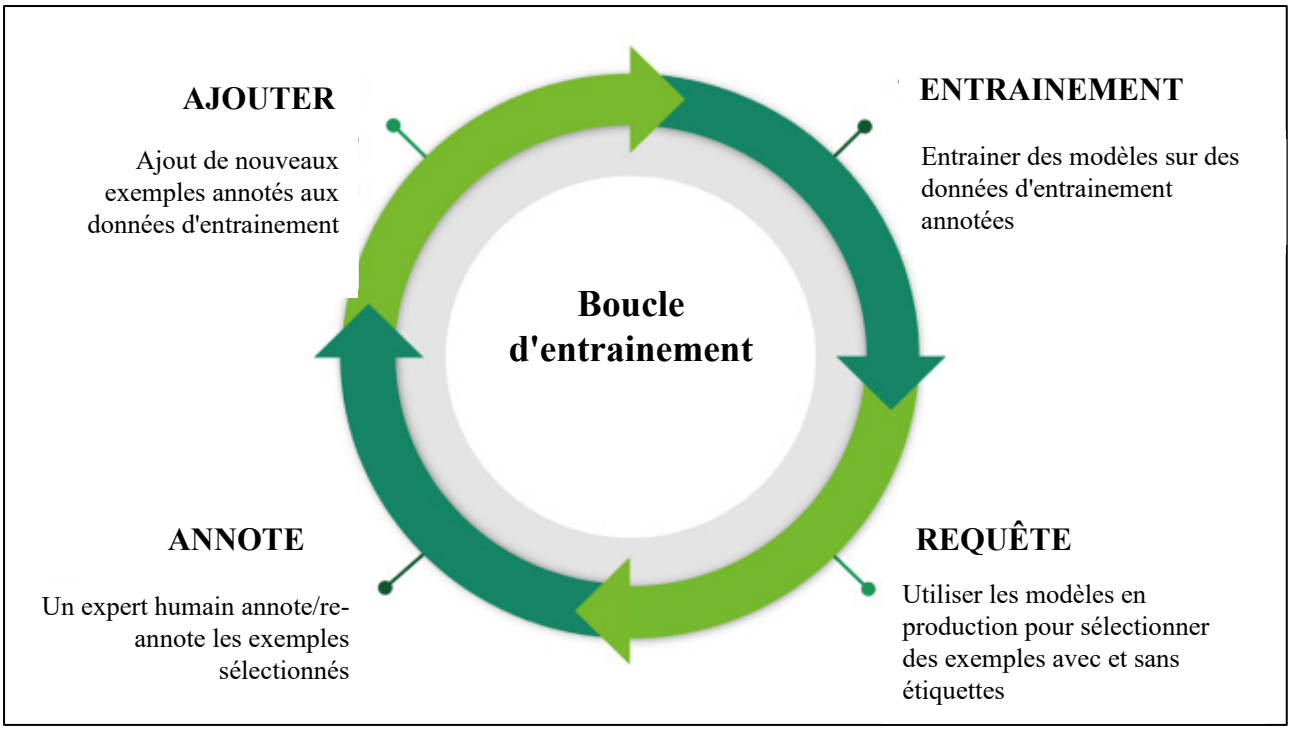

Figure 3.1 Boucle d'entrainement

3.1.3 Les nids de poule

Un modèle de détection d'objets avec YOLOv5 puis YOLOv8 a été entrainé afin de détecter les nids de poule dont la classe associée est « pothole ». A partir de maintenant, nous considérons dans ce mémoire, le terme « nids de poule » comme terme général illustrant toute dégradation de la chaussée.

3.1.3.1 Récolte des données sources

Dans un premier temps, la récolte des données s'est faite par déclenchement manuel de la caméra avec les véhicules des développeurs. Une fois suffisamment d'images de nids de poule accumulées, un premier modèle a été entrainé. Par la suite, la mise en production du premier modèle d'entrainement a permis de récolter les premières détections. Des ajouts réguliers de

données récoltées manuellement permettent de réduire le biais des réseaux entrainés (Figure 3.2). Après un nouvel entrainement, si le modèle valide les tests, il peut être déployé en production.

L'ajout de données s'est fait essentiellement par les images détectées de nos modèles en production, et cela ne parait pas la meilleure solution, au risque d'introduire un biais ou de manquer de diversité d'exemple. Pourtant, il était trop difficile de trouver aléatoirement des images avec des exemples de nids de poule. Il aurait fallu dédier quelqu'un à la recherche manuelle de nids de poule sur le terrain. D'où l'importance de nos choix dans la partie suivante : 4.1.3.2 Extraction des données significatives.

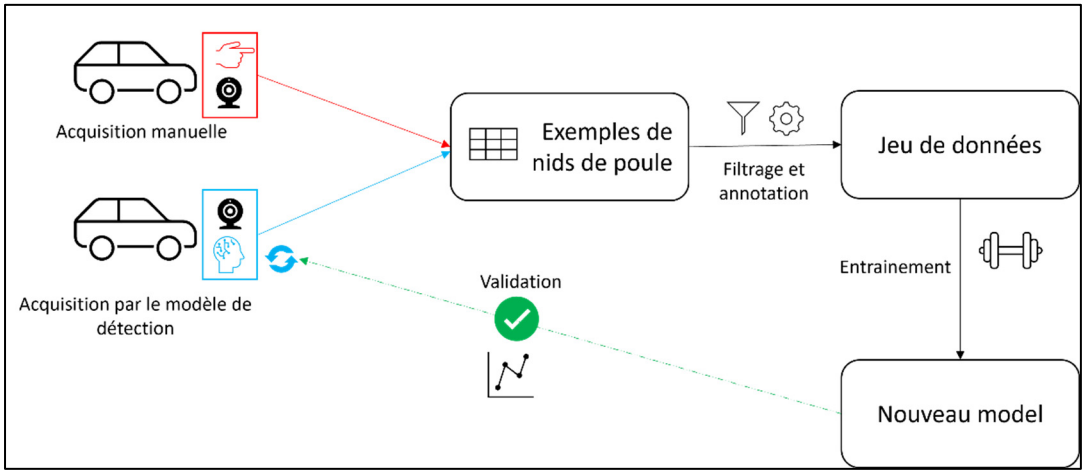

Figure 3.2 Schéma d'acquisition des nids de poule

3.1.3.2 Extraction des données significatives

Chaque nid de poule croisé sur la route est par essence unique, c'est pourquoi sa détection n'est pas triviale. De plus, définir un nid de poule en tant que tel n'est pas aisé non plus, cela peut être défini par un trou d'un certain diamètre ou une fissure d'une largeur minimum par exemple. Aussi, la présence ou non d'un nid de poule n'est pas binaire. La différence entre un inconfort de la route et un nid de poule n'est pas bien définie. Tout ceci rend la sélection des images difficiles. C'est pourquoi nous avons choisi dans un premier temps de mettre des exemples de nids de poule évidents dans le jeu de données d'entrainement.

Le modèle est implémenté dans une portion de la flotte de véhicules de patrouille depuis plusieurs mois et permet un entrainement périodique du modèle. En fait, un premier modèle a été entrainé en amont pour détecter les nids de poule les plus évidents. Toutefois, le contraste, la luminosité, l'imperfection de la chaussée et d'autres paramètres environnementaux augmentent considérablement les fausses détections, aussi appelés « faux positifs ». Nous arrivons à réduire le nombre de faux positifs en réentrainant le modèle au fur et à mesure que notre jeu de données de nids de poule augmente en variété.

Aussi, on exploite les faux positifs pour entrainer le modèle sur ses erreurs. En effet, si le modèle détecte un nid de poule alors qu'il s'agit d'une bouche d'égout, il est possible d'intégrer cette image à la base de données. Le modèle apprendra donc à distinguer les nids de poule des bouches d'égout. Cela fait partie de la boucle d'apprentissage actif mentionné en 3.1.2.

Grâce à l'architecture d'acquisition de données présentées dans la partie 2.1, on peut aussi facilement choisir des données selon le taux de confiance que le modèle attribue à la détection. En effet, un exemple évident de nid de poule à un taux de confiance bas (inférieur à 0.6 par exemple) est potentiellement intéressant à introduire dans notre jeu de données. Car ceci peut tendre à améliorer nos modèles sur des exemples auxquels le modèle est peu confiant.

Finalement, en pointant spécifiquement sur certaines faiblesses du modèle interprété grâce à sa mise en production, on peut rendre le modèle plus performant et plus robuste.

3.1.3.3 Présentation du jeu de données final

À la fin de mon projet, c'est au total 6212 images que nous avons récoltées dans le jeu de données des nids de poule : 5236 sont des images de nids de poule, soit 7914 occurrences (nombre d'objet dans les images), et 976 sont des images de faux positifs (des mauvaises détections du modèle introduites comme des exemples d'arrière-plans seuls, ie aucune annotation n'accompagne ces données). La distribution du jeu de données est résumée en Figure 3.3.

Traditionnellement, un jeu de données est composé :

- D'un lot d'entrainement : c'est le lot sur lequel le modèle est entrainé. Il est classiquement reparti en deux sous parties. L'une sera considérée comme le lot d'entrainement final et l'autre comme le lot de validation. Classiquement, on attribue environ 80 % des données au premier et 10 % au deuxième. Le lot de validation est en fait un lot de test, à l'intérieur de l'entrainement, qui permet au modèle de s'évaluer en s'y confrontant. Les poids du modèle sont affectés en fonction des résultats.
- D'un lot de test : ce lot permet d'évaluer le modèle entrainé sur des données qui lui sont inconnues. Ici, c'est 683 images du jeu de données qui sont réservées exclusivement au lot de test. Cela représente un peu plus de 10 % du jeu de données dans sa globalité.

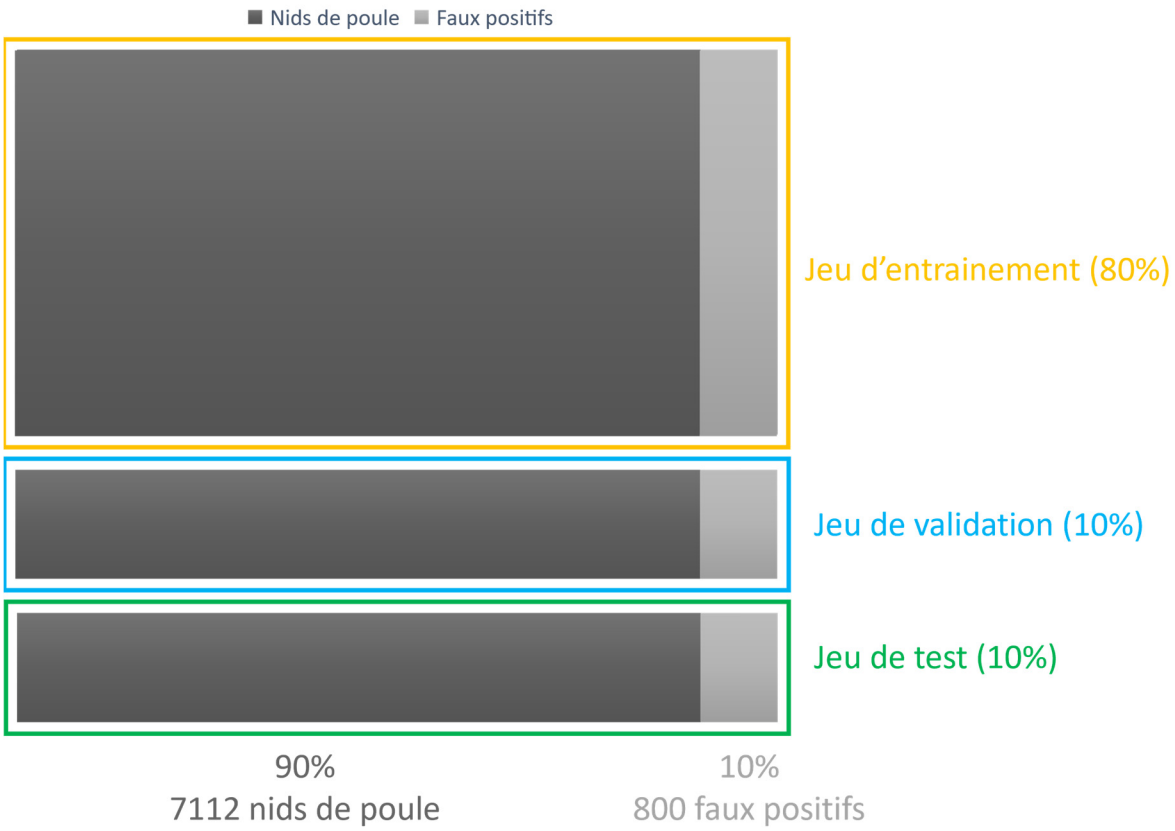

Figure 3.3 Distribution du jeu de données des nids de poule

Dans notre cas, le jeu de test a été trié de façon à essayer d'identifier les performances sur certaines catégories. On retrouve :

- Les évidents ou « obvious » : Ces nids de poule sont considérés comme facilement identifiable par l'humain. Ils peuvent être profonds, larges, à fort contraste sur la route ou identifiable dans un contexte simple (Figure 3.4).

Figure 3.4 Cas d'un nid de poule identifié comme évident dans le jeu de données

- Les petits ou « small » : Ces nids de poule sont considérés comme des petits trous de route (Figure 3.5).

Figure 3.5 Exemple des petits nids de poule

- Les faux positifs : Il y a aussi des exemples dans le lot de test qui sont des erreurs récurrentes d'anciennes versions du réseau. Ils permettent d'interpréter si le nouveau modèle ne relève plus les cas de mauvaises détections des anciens modèles. Aussi, ces exemples jaugent sa sensibilité. En Figure 3.6, une version de notre modèle détectait par erreur ce correctif de route comme un nid de poule.

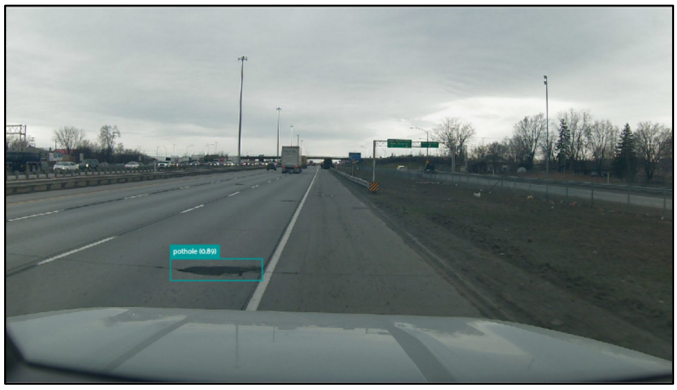

Figure 3.6 Exemples de faux positif

- Les cracks : Ce sont des trous fins et étendus, comme des fentes dans la route, ils sont souvent perpendiculaires ou verticaux (Figure 3.7) à la route et horizontaux (Figure 3.8).

Figure 3.7 Exemples de cracks verticaux

Figure 3.8 Exemples d'un crack horizontal

- Les autres : il existe une quasi-infinité de nids de poule, beaucoup ne sont pas facilement catégorisables, mais ils retrouvent quand même leur place dans le lot de test. En effet, ce lot doit être le plus représentatif de notre jeu de données au complet.

Finalement, ces sous-catégories permettent de tester les performances du modèle sur des cas de figure précis. Nous essayons de dresser un bilan moins binaire et donc plus détaillé des performances du modèle, car l'identification d'un nid de poule n'est pas une tâche aisée.

Détailler ses forces et ses faiblesses permet de comprendre comment il peut être amélioré (quelles données peuvent être ajoutées au jeu de données pour l'entrainement).

3.1.4 Lampadaires

Un modèle de détection d'objets YOLOv5 a été entrainé afin de détecter les lampadaires éteints la nuit. Ce modèle est entrainé sur deux classes:

- Les lampadaires en état de marche (allumé) dont la classe correspondante est « lighting ». La détection de ce cas se justifie par la facilité à les identifier et par ses potentielles utilités pour le MTQ comme vérifier l'emplacement de ses lampadaires, modèles par exemple.
- Les lampadaires éteints dont la classe correspondante est « ligthoff ». La nuit, en contraste avec les lampadaires allumés, les lampadaires éteints sont considérés comme défectueux.

3.1.4.1 Récolte des données sources

Les données sources ont été récoltées en capturant des images aléatoires de nuit. Puisque les lampadaires défectueux sont nombreux sur les routes surveillées par les patrouilleurs, il était facile d'obtenir des exemples de lampadaires éteints avec des images capturées aléatoirement (mode de déclenchement par intervalle de temps ou de distance).

En parallèle de l'entrainement, le dernier modèle entrainé effectue des inférences de nuit dans les véhicules de patrouille ce qui nous permet de valider son comportement sur les routes.

Afin d'éviter des cas ambigus au crépuscule et à l'aube comme sur la Figure 3.9, lorsqu'on ignore si certains lampadaires devraient être allumés ou éteint, la détection est seulement active après le coucher du soleil. En effet, un algorithme vérifie la position géographique et valide

l'heure du coucher du soleil afin de démarrer la détection de nuit, peu importe le temps de l'année.

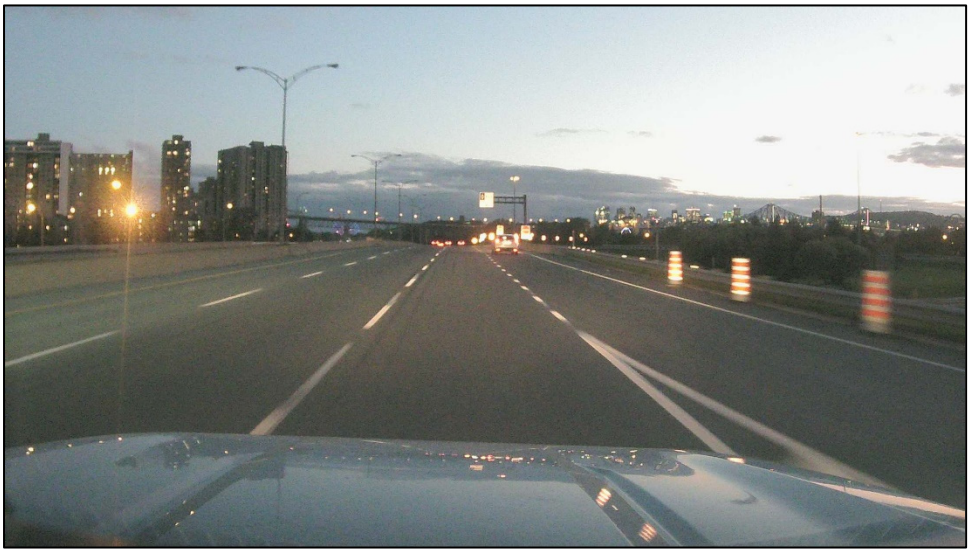

Figure 3.9 Exemples d'un cas ambigu au coucher de soleil

3.1.4.2 Extraction des données significatives

A contrario du cas sur les nids de poule, l'extraction de données significatives est plus aisée. En effet, les lampadaires sont omniprésents dans les milieux urbains. De plus, le cas de lampadaires brisés n'est pas un phénomène rare. Donc obtenir des exemples la nuit de manière aléatoire, sans un premier modèle de détection, est assez facile. Au-delà des différents modèles de lampadaires installés au Québec, un lampadaire brisé reste une simple tâche à détecter avec peu de variances dans les cas de figure. Jusqu'à lors nous avons identifié trois types de lampadaires dans la région que nous nommerons modèle A en Figure 3.10, modèle B en Figure 3.11 et modèle C en Figure 3.12.

Figure 3.10 Lampadaire
double avec potence
courbée (modèle A)

Figure 3.11 Lampadaire double
avec potence droite (modèle B)

Figure 3.12 Tour d'éclairage
(modèle C)

3.1.4.3 Présentation du jeu de données final

À la fin de mon projet, c'est au total 2305 images que nous avons récoltées dans le jeu de données des lampadaires : 12722 occurrences concernent les lampadaires en état de marche et 2464 concernent ceux hors service (voir Figure 3.13).

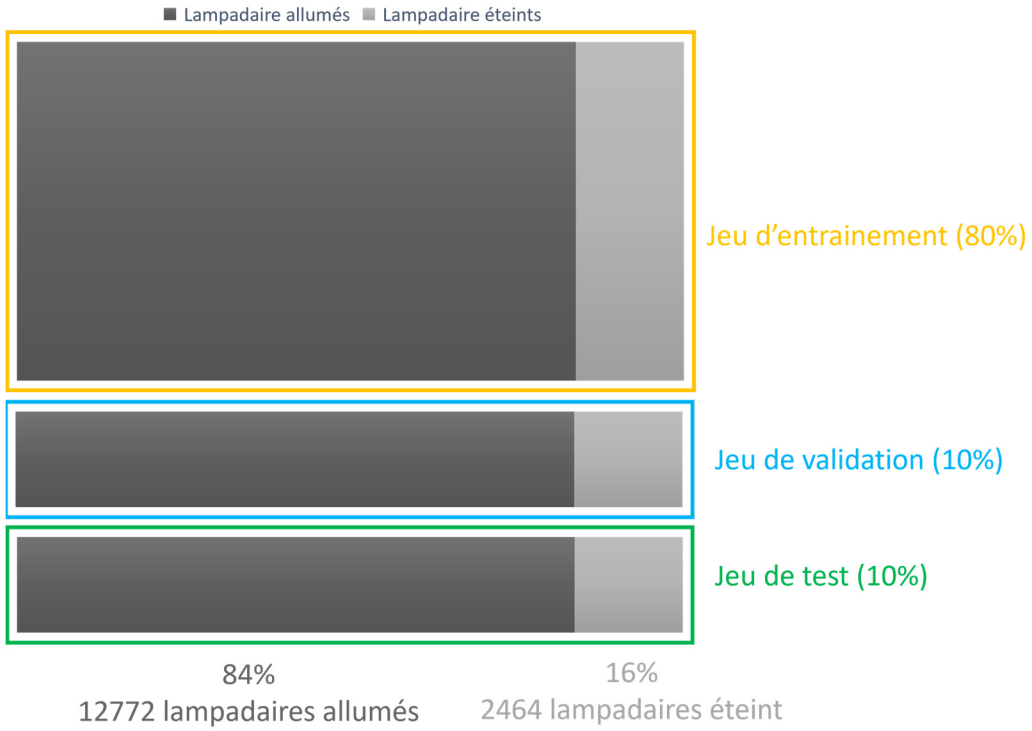

Figure 3.13 Distribution du jeu de données sur les lampadaires

Ici, le lot de test représentant 10 % du jeu de donnée est sous catégorisé en deux parties :

- Les images où les lampadaires brisés sont situés au premier plan (appelées « foreground »): il est difficile de définir une distance à laquelle nous souhaitons que le modèle détecte des lampadaires éteints. D'autant plus que la visibilité dans la profondeur de l'image dépend de la luminosité et de la météo au moment de la capture. Cependant, un lampadaire loin dans l'image arrivera au premier plan de la voiture

quelques centaines de mètres plus tard. On estime alors que le détecteur doit être le plus performant possible pour les lumières brisées au premier plan (Figure 3.14).

Figure 3.14 Images du jeu de données catégorisées comme des exemples de premier plan

- Les autres : toutes les images où les lampadaires ne sont pas au premier plan exclusivement et dont la visibilité est plus difficile (Figure 3.15)

Figure 3.15 Exemples d'image de lampadaires brisés non catégorisés

3.1.5 Détection des glissières de sécurité endommagées

Étudions le cas particulier du jeu de données des glissières de sécurité endommagées.

3.1.5.1 Récolte des données sources

Dans le cas des glissières brisées, la récolte des données en sélectionnant des vidéos/photos aléatoires de la route est particulièrement difficile puisque son occurrence est faible. Une partie peut être récoltée en activant manuellement les caméras sur le véhicule du développeur. Cependant, cela ne suffit pas à constituer un premier jeu de données afin d'envisager un premier détecteur qui pourrait détecter quelques barrières.

3.1.5.2 Extraction des données significatives

On peut décrire les glissières endommagées comme un événement rare en comparaison avec les événements détectés dans les parties précédentes. En effet, les glissières sont présentes sur quasiment tous les grands axes routiers du Québec, on a donc d'innombrables exemples de glissières en bon état, mais ce n'est pas le cas pour celles endommagées. Donc, ces données étant rares, la capture d'images aléatoires n'est pas viable pour la collecte de données. Devoir regarder tous les clichés pris en espérant trouver une glissière serait inefficace (comme le cas des nids de poule). Ainsi, il a fallu trouver une stratégie qui nous garantirait une plus grande probabilité de trouver des glissières brisées sur les routes. Le Ministère des Transports possède un inventaire des glissières accidentées au Québec avec leurs coordonnées géographiques.
La Figure 3.16 est un aperçu de la base de données de glissières endommagées fournie par le MTQ.

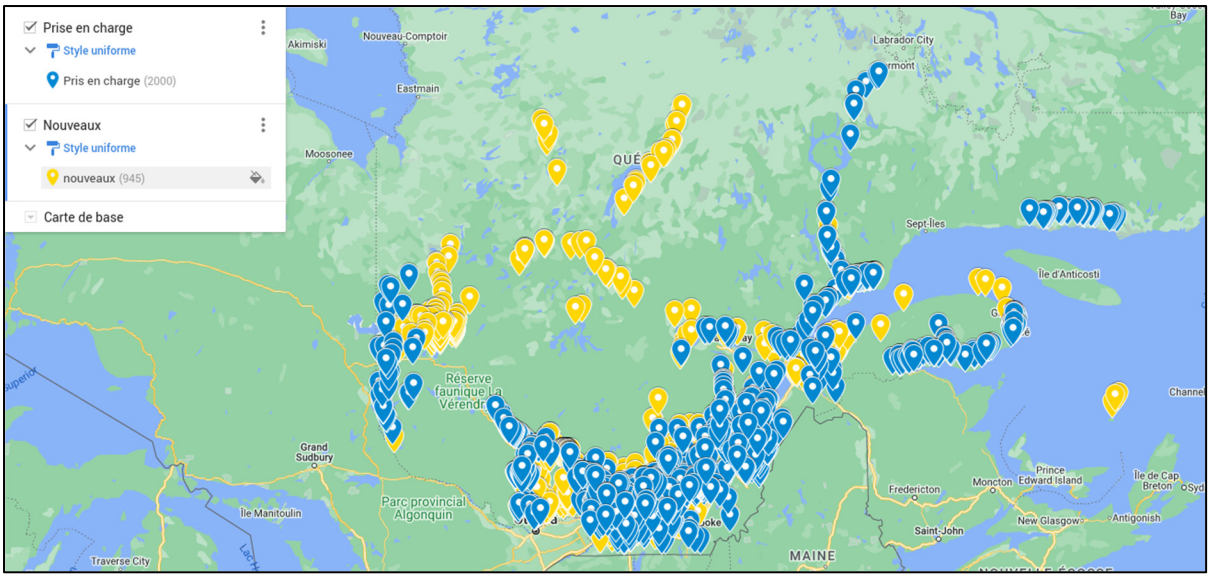

Figure 3.16 Points répertoriés des glissières endommagés par le MTQ

Avec ces données, nous avons mis en place un système d'acquisition qui permet de capturer une série d'images autour du point d'intérêt géographique. Effectivement, dès qu'un véhicule de patrouille équipé du système observe l'un des points d'intérêt, il enregistre la vidéo. Un algorithme de projection sur plan d'image roule en permanence sur le système et vérifie si un objet d'intérêt (par coordonnées GPS) est observable dans le champ de vue de la caméra.
Pour s'assurer de la validité de l'information, un curseur rouge est intégré aux vidéos après l'acquisition. Cela nous permet de valider les informations de l'inventaire du MTQ, mais aussi de localiser rapidement l'emplacement de la potentielle glissière brisée dans l'image. On peut observer une barrière endommagée sur la Figure 3.17 et une autre qui a été réparée depuis sur la Figure 3.18.

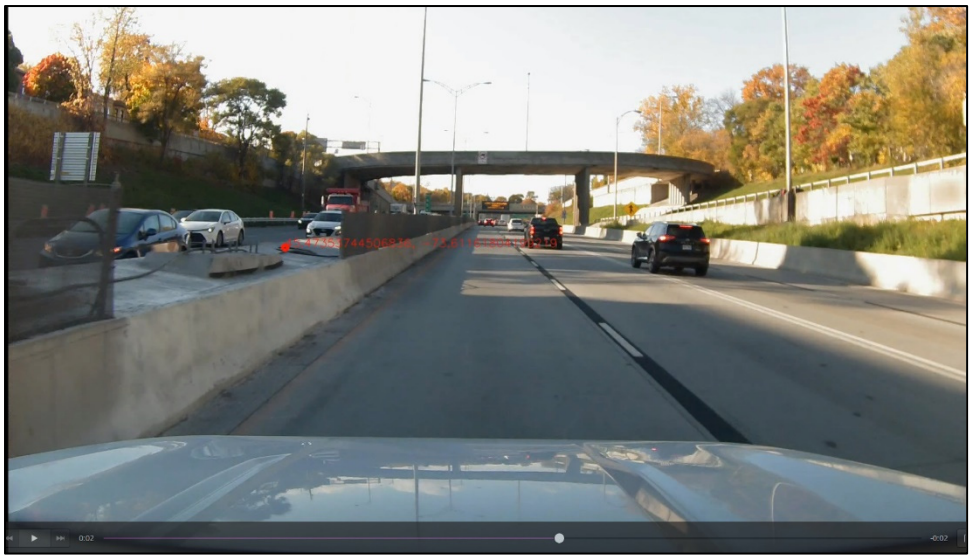

Figure 3.17 Pointeur sur barrière endommagée

Figure 3.18 Pointeur sur barrière réparée

Par exemple, d'après la dernière mise à jour fournie par le Ministère, nous avons 2945 points d'intérêt au Québec (Figure 3.16 Points répertoriés des glissières endommagés par le MTQ). Ils répartissent les barrières avec des anomalies en deux catégories: celles prises en charge (en bleu) et non prise en charge (nouveau, en jaune).

Dans un premier temps, nous voulions obtenir le plus de données possible pour évaluer rapidement le fonctionnement des captures d'images autour d'un point d'intérêt géographique donné. Nous avons donc pris en compte l'intégralité des données fournies. D'autant plus que le parc de voiture en service en ce moment ne couvre qu'une petite partie des routes, nous avons essayé de les exploiter au maximum.

Une fois le système d'acquisition validé, nous avons pu constater que beaucoup de ces points d'intérêts n'étaient en fait plus d'actualité. Ainsi, l'acquisition devenait contreproductive puisque nous obtenions un nombre important de données à visualiser sans intérêt. Donc nous avons restreint l'acquisition aux points d'intérêts dits "nouveaux". Nous avons aussi la possibilité de mettre à jour notre base de données de points d'intérêt, afin d'éliminer ceux qui ont déjà été détectés ou réparés. Cependant, la faible couverture du réseau par les patrouilleurs équipés du système embarqué à ce moment-là, ne nous permettait pas d'obtenir assez d'exemples différents. D'autant plus que la récolte d'une centaine de glissières n'aurait pas été suffisante pour entrainer un modèle de détection.

CHAPITRE 4

RÉSULTATS EXPÉRIMENTAUX

4.1 Introduction

Ce chapitre est le dernier du mémoire. Après vous avoir présenté l'architecture globale et la méthodologie au CHAPITRE 2 puis l'acquisition des jeux de données au CHAPITRE 3, je vous présente ici les résultats des modèles. Ce chapitre aborde, dans un premier temps, les métriques utilisées pour l'évaluation des performances de modèles de détection. Puis les résultats par domaine d'application seront exposés et discutés.

4.2 Métriques et outils d'évaluations des performances

Pour évaluer de manière adéquate la performance de ces modèles, il est essentiel de comprendre et d'appliquer correctement un ensemble de métriques spécifiques.

Avant de plonger dans ces métriques, il est important de comprendre certains concepts clés :

- Vrais positifs (TP) : les objets qui ont été correctement identifiés comme appartenant à la classe.
- Faux positifs (FP) : les objets qui ont été incorrectement identifiés comme appartenant à la classe.
- Vrais négatifs (TN) : les objets qui ont été correctement identifiés comme n'appartenant pas à la classe.
- Faux négatifs (FN) : les objets qui ont été incorrectement identifiés comme n'appartenant pas à la classe.

Ces quatre termes sont essentiels pour comprendre comment les performances d'un modèle de détection d'objets sont évaluées. Par exemple, si nous utilisons un modèle YOLO pour détecter

les nids de poule dans une image, c'est un cas de figure où il y a une seule classe nommée nid de poule, alors les termes introduits deviennent :

- TP : un nid de poule est correctement identifié par le modèle (Figure 4.1).
- FP : un objet est prédit comme un nid de poule, mais n'en est pas un en réalité (Figure 4.2).
- TN : aucun nid de poule n'est détecté dans l'image et il n'y a effectivement pas de nid de poule dans l'image en réalité.
- FN : le modèle n'a pas réussi à détecter la présence d'un nid de poule dans l'image, pourtant il y en a.

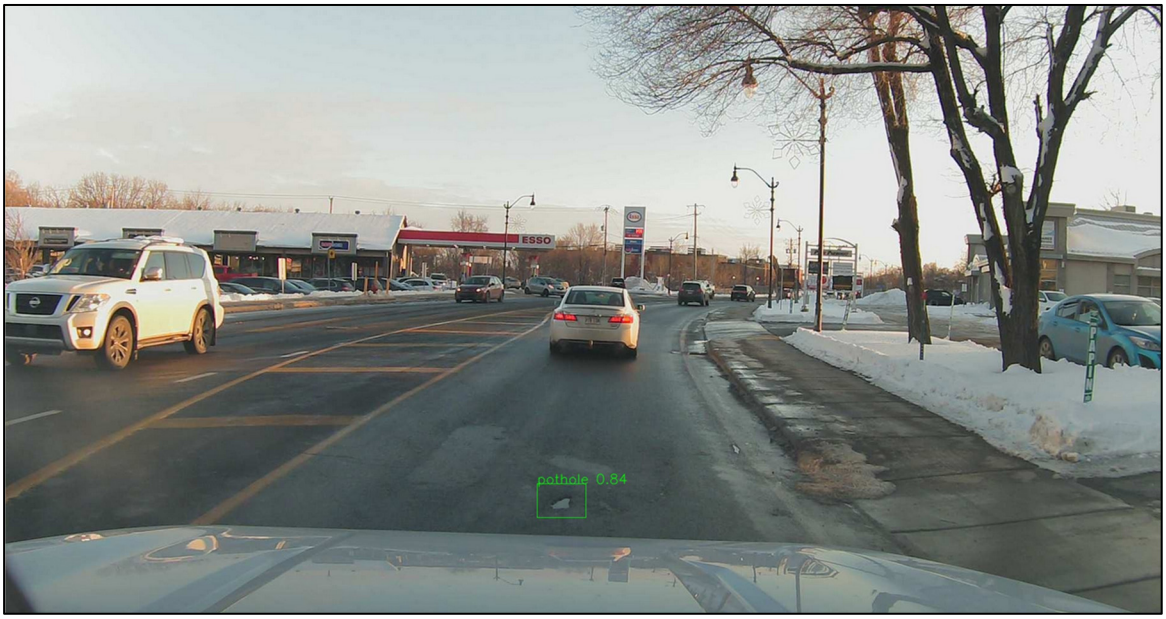

Figure 4.1 Exemple de nid de poule vrai positif (TP)

Figure 4.2 Exemple de nid de poule faux positif (FP)

A présent nous allons explorer les métriques clés utilisées pour évaluer la performance des modèles de détection d'objets en vision par ordinateur, notamment la matrice de confusion, la précision, le rappel et le score F1. Chacune de ces métriques offre un aperçu unique de la performance du modèle, et ensemble, elles fournissent une image complète qui peut aider à identifier les forces et les faiblesses du modèle, et à orienter les efforts d'amélioration.

4.2.1 Matrice de confusion

La matrice de confusion est un tableau de dimension deux qui permet de visualiser la performance d'un algorithme de détection ou de classification. Chaque ligne de la matrice représente les instances d'une classe prédite, tandis que chaque colonne représente les instances d'une classe réelle. Les quatre termes clés introduits dans la section précédente composent la matrice de confusion de la Figure 4.3. En fait, elle fournit un résumé visuel des prédictions correctes et incorrectes faites par le modèle.

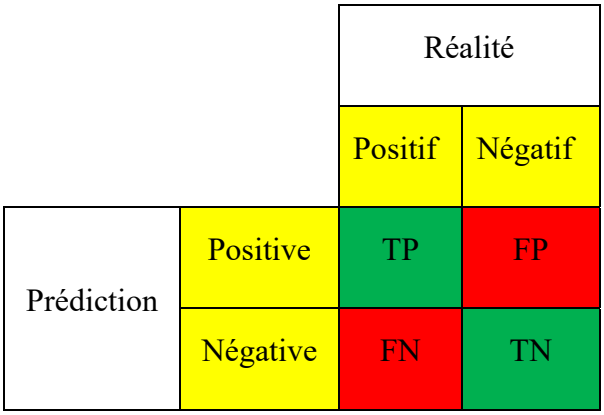

Figure 4.3 Forme d'une matrice de confusion

4.2.2 La précision

La précision est une métrique qui mesure la proportion de prédictions positives qui sont réellement correctes. En d'autres termes, parmi toutes les prédictions que le modèle a identifiées comme positives, combien sont réellement positives. La précision est calculée comme suit :

$$Précision = \frac{TP}{TP + FP} \tag{4.1}$$

Toujours dans le cas de figure de la détection de nids de poule et de sa classe associée, la précision représente la proportion de nids de poule réellement positifs sur le nombre total de nids de poule détecté par le modèle. Donc plus la précision est élevée (proche de 1), plus le modèle est fiable sur la détection d'un nid de poule dans l'image.

En revanche, comme la précision est calculée à partir des instances positives, elle ne prend pas en compte les instances négatives (faux négatif et vrai positif). C'est-à-dire qu'un modèle peut être précis tout en manquant énormément d'objets à détecter (faux négatifs). C'est pourquoi la précision est un indicateur de performance, mais ne peut être utilisée seule. Ainsi, pour évaluer

de manière plus complète la performance d'un modèle, il est recommandé de considérer la précision en conjonction avec d'autres métriques telles que le rappel et le score F1.

4.2.3 Le rappel

Le rappel, également connu sous le nom de sensibilité, mesure la proportion de véritables positifs qui sont correctement identifiés par le modèle. En d'autres termes, parmi tous les véritables positifs, combien le modèle a-t-il réussi à capturer d'objets. Le rappel est calculé comme suit :

$$Rappel = \frac{TP}{TP + FN} \tag{4.2}$$

Toujours dans le cas de figure de la détection de nids de poule, le rappel représente le nombre de bons nids de poule détectés sur le nombre total de nids de poule en réalité à détecter dans le(s) image(s). Donc plus le rappel est grand (proche de 1), plus le modèle est capable de détecter les nids de poule dans leur globalité, plus il est sensible.

En revanche, le rappel ne prend pas en compte les faux positifs dans son calcul. C'est-à-dire qu'un modèle peut avoir un très bon score de rappel, mais détecter beaucoup d'autres choses, que l'objet de détection, par erreur. Ainsi, tout comme la précision, il est nécessaire de combiner cet indicateur à d'autres métriques comme la précision ou le score F1.

4.2.4 Score de confiance

Le score de confiance indique la probabilité qu'une prédiction soit correcte. Par exemple, lorsqu'un modèle essaie d'identifier un objet dans une image, il peut classer l'objet avec une certaine étiquette et attribuer un score de confiance à cette étiquette. En fait, ce score reflète la confiance dans la classification et la localisation de l'objet. La formule pour le score de confiance de classe :

$$C = Pr(class) * IoU \tag{4.3}$$

Où : $Pr(class)$ est la probabilité d'appartenance de l'objet à cette classe et IoU le score d'intersection sur l'Union entre la boîte prédite et la boîte de vérité terrain.

Ce score est compris entre 0 et 1. Plus le score de confiance est élevé, plus l'algorithme est sûr que sa prédiction est correcte.

Cependant, il convient de noter que même si un score de confiance est très élevé, cela ne garantit pas que la prédiction soit correcte. C'est simplement une indication de la certitude de l'algorithme à propos de sa prédiction. Les scores de confiance peuvent donc être utiles pour comprendre le niveau de certitude de l'algorithme, mais ils ne doivent pas être interprétés comme une preuve définitive de l'exactitude d'une prédiction.

4.2.5 Le score F1

Le score F1 est une moyenne harmonique de la précision et du rappel. Il donne une mesure globale de la performance du modèle, en tenant compte à la fois de la précision et du rappel :

$$\text{Score F1} = \frac{2 * Précison * Rappel}{Précision + Rappel} \quad (4.4)$$

Par conséquent, un modèle ne peut obtenir un score F1 élevé qu'en ayant à la fois une bonne précision et un bon rappel. Il est donc utile pour interpréter les performances globales avec un seul indicateur.

Par exemple, si notre modèle YOLO a une précision de 0.9 et un rappel de 0.8 au score de confiance k, le score F1 serait 2 * (0.9 * 0.8)/(0.9 + 0.8) = 0.847. Cela signifie que le modèle a une bonne performance globale en termes de précision et de rappel au score de confiance k.

4.2.6 Précision moyenne AP

Alors que la précision, le rappel et le score F1 sont calculés pour un seuil de confiance donné, la précision est définie comme l'aire sous la courbe de précision-rappel (courbe PR). On obtient cette courbe en modifiant le seuil de confiance k évoqué dans la section précédente. Toute détection avec un score inférieur au seuil est considérée comme un faux positif. À chaque niveau unique de score de classification présent dans les résultats de détection, nous calculons la précision et le rappel à ce point. Après avoir parcouru toutes les paires de valeurs de précision-rappel correspondant à chaque seuil de score unique, nous obtenons une courbe de précision-rappel qui peut ressembler à la Figure 4.4 par exemple.

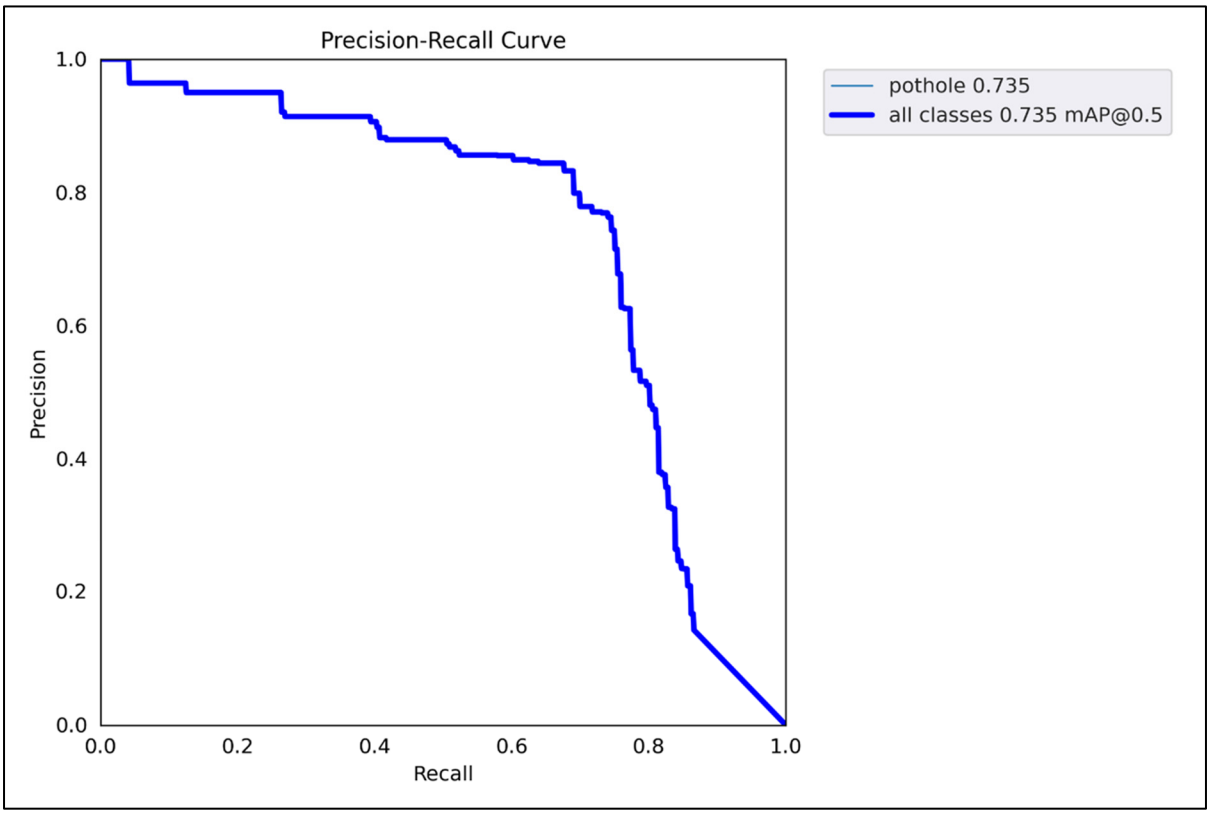

Figure 4.4 Courbe précision-rappel sur des données tests pour notre modèle de détection de nids de poule

Souvent, on introduit la mAP pour « mean Average Precision », car l'AP est calculée individuellement pour chaque classe. Donc la mAP est la moyenne de l'AP de n classes :

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{4.5}$$

Où α désigne le seuil d'IoU dont le concept est introduit dans 2.2.1.

L'AP est calculée en fonction de ce seuil. Dans des modèles comme YOLO, il est souvent question de mAP@0.5 (usuellement appelée mAP) ou mAP@0.95 qui traduisent respectivement le calcul d'une mAP pour un seuil IoU égal à 0.5 ou à 0.95. Plus explicitement, cela signifie que pour qu'une prédiction soit considérée comme correcte, elle doit avoir un chevauchement avec la boîte englobante réelle de 50 %, respectivement 95 %. On considère mAP@0.95 comme une mesure très exigeante.

4.3 Détections des infrastructures défectueuses

Cette partie est dédiée à l'analyse des performances des modèles de détection de l'entrainement à la production.

4.3.1 Détection des nids de poule

Dans cette section, nous allons aborder les résultats du modèle de détection des nids de poule par l'apprentissage actif. Dans un premier temps nous aborderons les performances issues de l'entrainement, puis nous verrons le comportement en production. Enfin, nous discuterons des résultats.

4.3.1.1 Entrainement du modèle

L'entrainement du modèle de détection s'est fait de manière itérative. Globalement, une itération correspond à un changement dans le jeu de données, puisque nous avons utilisé

l'entrainement actif pour faire grandir notre jeu de données, tout en faisant évoluer notre modèle.

Pour la version YOLOv5, les meilleurs résultats sont présentés dans le Tableau 4.1 et leurs courbes associées en Figure 4.5, Figure 4.6 et Figure 4.7. Ils ont été obtenus avec un jeu de données contenant 2394 images et 3381 occurrences pour un lot de test représentant 10 % de ce jeu. On obtient en général des résultats moyens que ce soit en termes de précision ou de rappel. Le modèle semble performer moins bien pour des petits nids de poule ou ceux en crack. Plus précisément, la plus grande faiblesse du modèle se situe dans sa capacité à détecter tous les exemples dans une image même à basse confiance, notre modèle est peu sensible.

Tableau 4.1 Résultats du modèle YOLOv5

Catégorie du lot de test	Précision P	Rappel R	MAP50	MAP95
Cracks	0.625	0.457	0.432	0.174
Petits	0.769	0.678	0.729	0.276
Evidents	0.806	0.683	0.723	0.301
Test (lot entier)	0.708	0.615	0.609	0.26

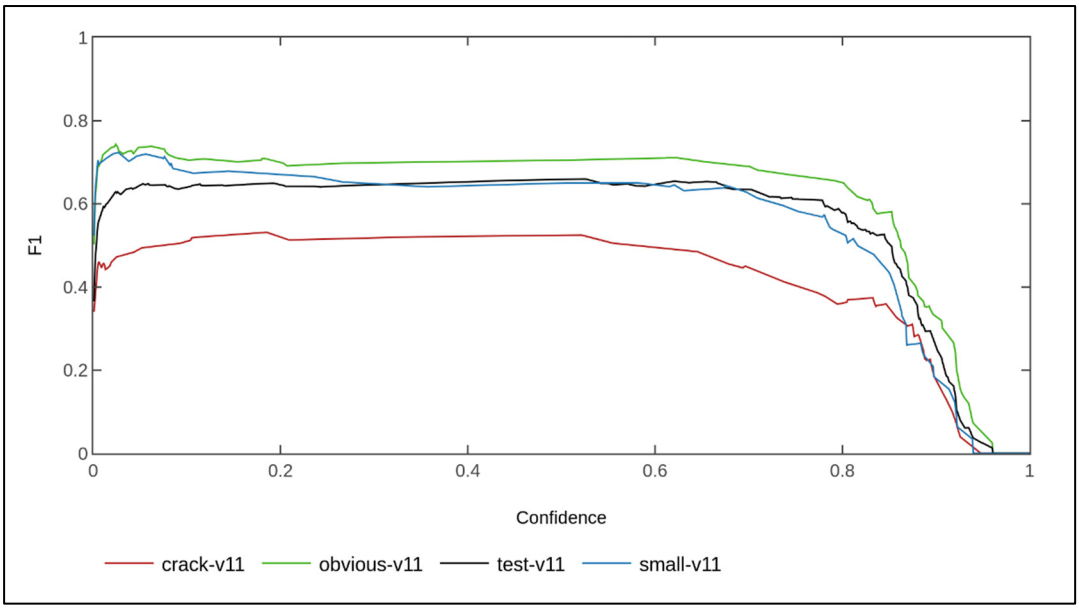

Figure 4.5 Courbes du score F1 du modèle YOLOv5 pour chaque catégorie de nids de poule définie dans le lot de test (test-v11)

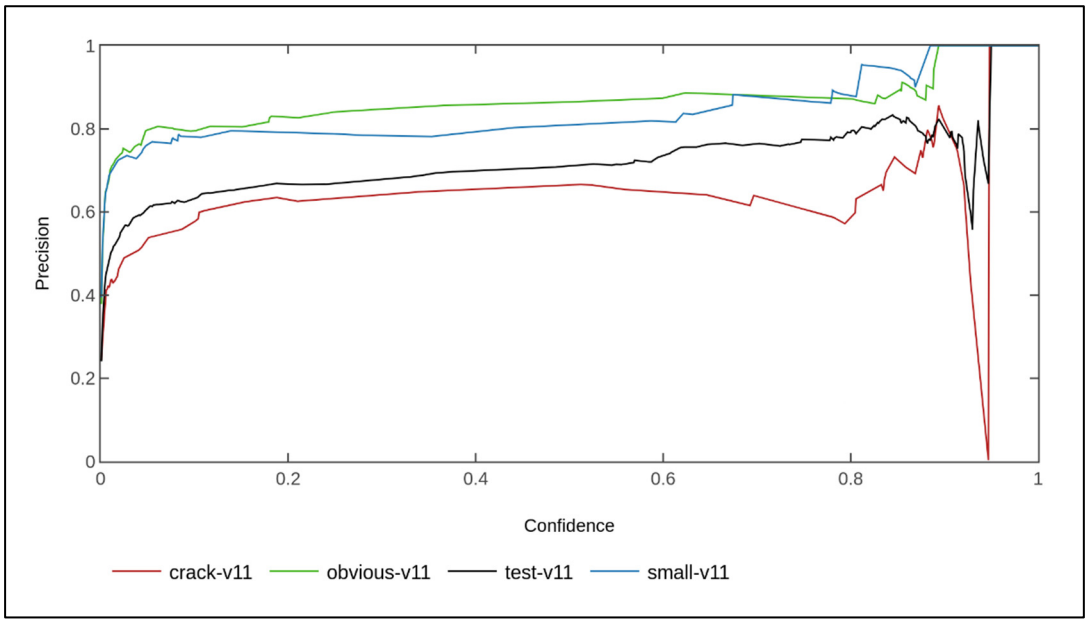

Figure 4.6 Courbes de précision en fonction de la confiance du modèle YOLOv5 pour chaque catégorie de nids de poule définie dans le lot de test (test-v11)

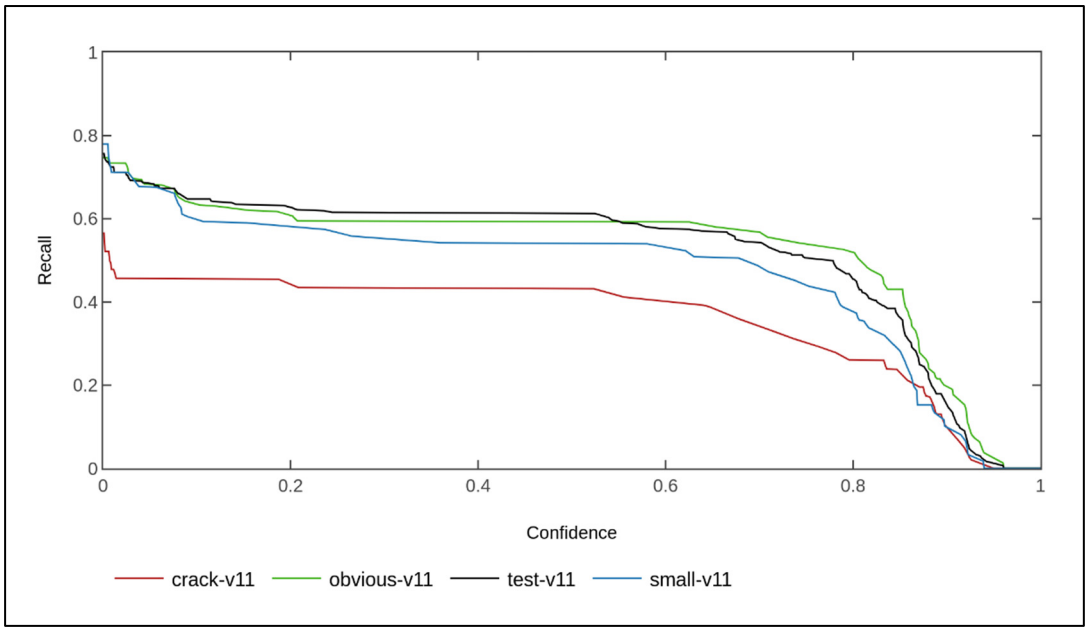

Figure 4.7 Courbes de rappel en fonction de la confiance du modèle YOLOv5 pour chaque catégorie de nids de poule définie dans le lot de test (test-v11)

4.3.1.2 Résultats en production

Les performances terrain sont parfois décalées de ce que nous pouvons observer à l'entrainement. Ceci s'explique par le fait que le jeu de données d'entrainement n'est qu'une représentation très partielle de la réalité. En effet, les cas de nids de poule sont très variés, tout comme la luminosité ou le temps durant l'année.

Afin d'évaluer et de présenter les performances de nos modèles sur le terrain, nous avons choisi de réunir 1000 images de détections pour chaque modèle en production. Ces données sont prises aléatoirement entre la mise en production du modèle et aujourd'hui.

En production, chaque détection est associée à un taux de confiance qui permet de juger l'assurance du modèle quant à la détection d'un objet dans l'image. Il est compris strictement entre 0 et 1, plus il est élevé, plus le modèle est certain de sa détection. En réalité, nous ne retenons en productions que les détections avec un taux de confiance supérieure à 0.4. En effet, celles en dessous, n'ont pas d'intérêt à être prise en compte, puisqu'elles sont par définition très peu fiables.

Il est important de noter que le taux de confiance ne garantit pas que la détection est toujours correcte. En effet, il peut y avoir des situations où le modèle n'est pas capable de détecter avec précision certaines caractéristiques ou nuances de l'image, ou bien que les données sur lesquelles il a été entrainé ne reflètent pas la diversité des situations réelles. C'est pourquoi, même avec un taux de confiance élevé, il est toujours important d'effectuer une vérification manuelle pour confirmer la prédiction ou la détection du modèle.
Ainsi, nous avons dû annoter ces données pour juger les performances de notre modèle. Nous attribuons donc manuellement à chaque détection le caractère "vrai positif" si elle est correcte et "faux positif" sinon.

Le diagramme circulaire de la Figure 4.8 représente les proportions de mauvaises détections "pothole_fp" et bonnes détections "pothole_tp" sur les 1211 détections sorties par le modèle (sur 1000 images issues de production). Nous obtenons donc environ 75 % de bonnes détections pour 25 % de mauvaises.
A priori, si nous arrêtons l'analyse ici, nous pourrions faire l'hypothèse qu'une détection sur quatre est mauvaise. C'est en réalité plus nuancé que cela.

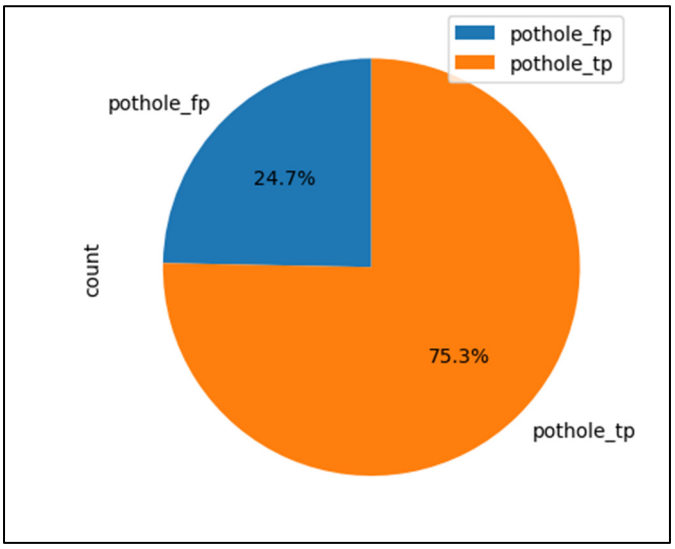

Figure 4.8 Proportion de faux et vrais positifs sur 1211 détections de nids de poule issues de production

C'est pourquoi nous retrouvons en Figure 4.9 un histogramme des bonnes et mauvaises détections. Cet histogramme traduit leur nombre d'occurrences par intervalle de confiance. À titre d'exemple, avec une confiance comprise entre 0.4 et 0.425, sur les 1000 images et 1211 détections faites aux totales, le modèle a donné 15 mauvaises détections contre 23 bonnes. Alors qu'entre 0.875 et 0.9, c'est 8 mauvaises détections pour 85 bonnes. En fait, l'histogramme permet de comprendre que la distribution n'est pas constante, mais variable selon l'intervalle de confiance. En d'autres termes, l'hypothèse évoquée plus haut est complètement erronée.

La Figure 4.9 permet de comprendre que le modèle commet des erreurs de détection pour tout intervalle de confiance confondue, mais qu'il existe des erreurs plus importantes que d'autres. En effet, l'attention se porte sur les détections erronées avec un taux de confiance haut car elles traduisent que le modèle se trompe, mais qu'il est confiant dans sa détection. De plus, l'histogramme permet de montrer qu'en plaçant un seuil de confiance à plus de 0.75,

nous enlevons plus de la moitié des mauvaises détections tout en gardant la grande majorité des bonnes détections. Ainsi, en ajustant le seuil de confiance du modèle de détection, on peut améliorer la précision et la fiabilité des résultats, car nous diminuons le nombre de faux positifs.

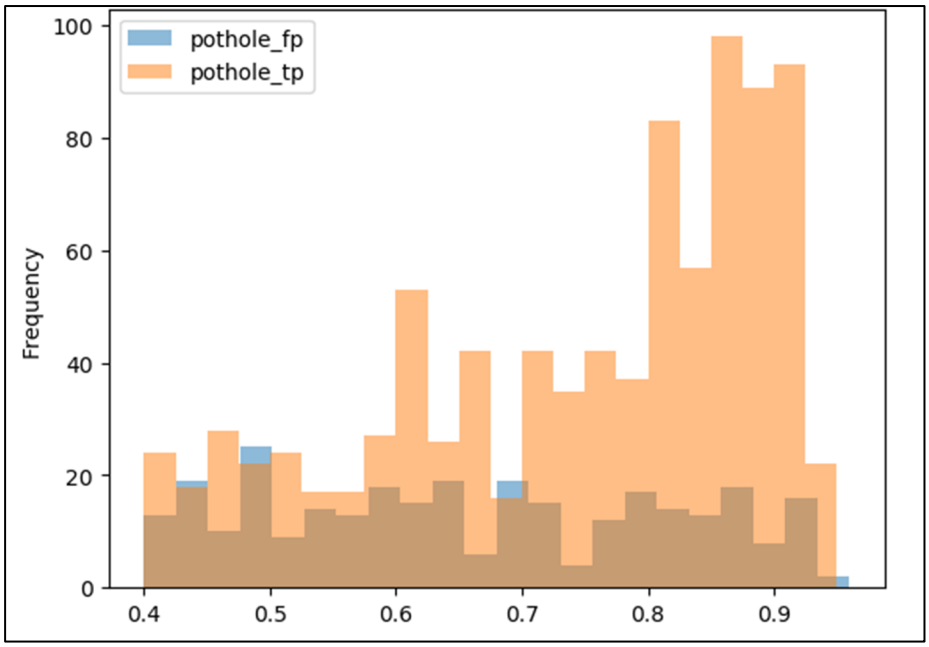

Figure 4.9 Histogramme de faux et vrais positifs selon leur confiance sur 1211 détections de nids de poule issues de production

À présent, si nous parcourons les détections les plus problématiques, c'est à dire pour lesquelles le taux de confiance est élevé, mais la détection est mauvaise, nous pouvons comprendre quels sont les types d'exemples sur la route qui posent encore des problèmes. En voici des exemples en Figure 4.10 où l'on retrouve une plaque d'égout, un débris, une flaque d'eau et un animal mort sur la route comme des objets détectés par le modèle en tant que nid de poule.

Figure 4.10 Exemples de faux positif du modèle en production

4.3.1.3 Discussion

Au départ, nous avions trop peu de données, car la mise en service des systèmes embarqués a pris du temps. Cependant, le processus d'annotation aussi prend du temps. D'autant plus qu'au début, l'auto-annotation n'était pas utile puisque notre modèle n'était pas assez performant, pour nous aider à identifier quelques bonnes caractéristiques à annoter dans l'image. Ainsi, l'évolution du modèle d'une itération à l'autre n'était pas très interprétable, car la quantité de données n'augmentait pas drastiquement (un petit ajout dans le jeu de données peut aussi bouleverser les résultats car trop peu d'exemples pour généraliser). Dans le cas idéal, la documentation d'Ultralytics recommande plus de 1500 images par classe et au moins 10000 instances (objets étiquetés) par classe.

Au fil du temps, avec l'objectif de tendre vers un jeu de données de l'ordre de grandeur recommandé, nous sommes parvenus à obtenir la meilleure version du projet à ce jour. En fait, le jeu d'entrainement a simplement doublé en une itération. Cette augmentation soudaine est

corrélée à l'augmentation du nombre de voitures équipées d'un système embarqué. Aussi, nous avions opté pour une stratégie d'extraction d'exemples significatifs : ajouter tous les nids de poule détectés en production avec un taux de confiance inférieur à 0,6. Cette stratégie a pu être mise en place grâce à une nouvelle mise à jour du module de filtrage, qui a permis de sélectionner par taux de confiance les détections en production.
Cependant, malgré nos résultats en demi-teinte, après cette version, l'ajout de nouvelles données pour arriver au jeu de données comme il est aujourd'hui (7914 occurrences de nids de poule) n'a pas permis d'obtenir un meilleur modèle. Bien que nous sommes plus que jamais proches des 10000 instances recommandées.

Le passage à l'architecture YOLOv8 n'a pas non plus amélioré significativement les résultats. On obtient sur le test set une mAP de 0. 697. Nous pensons donc que nous sommes arrivés à un palier où le nombre de données n'est plus le problème, mais plutôt la qualité du jeu de données. En effet, l'une des difficultés de l'augmentation du jeu de données sur quasi 8 mois est la perte de fidélité dans notre annotation. Nous l'avons observé en parcourant le jeu de données : par rapport à aujourd'hui, nous annotions différemment au début du projet. C'est bien normal, nous sommes influencés par les résultats obtenus au cours des entrainements itératifs et de ce qui était observé en production. Inconsciemment nous ajoutons un biais dans le jeu de données qui se répercute dans le modèle, sa politique de sélection peut changer, d'une itération à l'autre à cause de notre façon d'annoter qui fluctue. Par exemple en Figure 4.11, le modèle détecte deux nids de poule (en rouge) et nous n'en avions annoté qu'un. Ces détections pénalisent le modèle alors qu'en réalité les prédictions sont bonnes. L'inconsistance d'annotation de ce type d'exemple dans le jeu de données contribue à pénaliser les performances. Puisque dans un autre cas en Figure 4.12, nous avions bien décomposé le crack en 2 nids de poule identifiables et distinctifs. De plus, beaucoup de cas de figures sur les routes sont ambiguës et donc difficile à annoter en gardant une stricte cohérence.

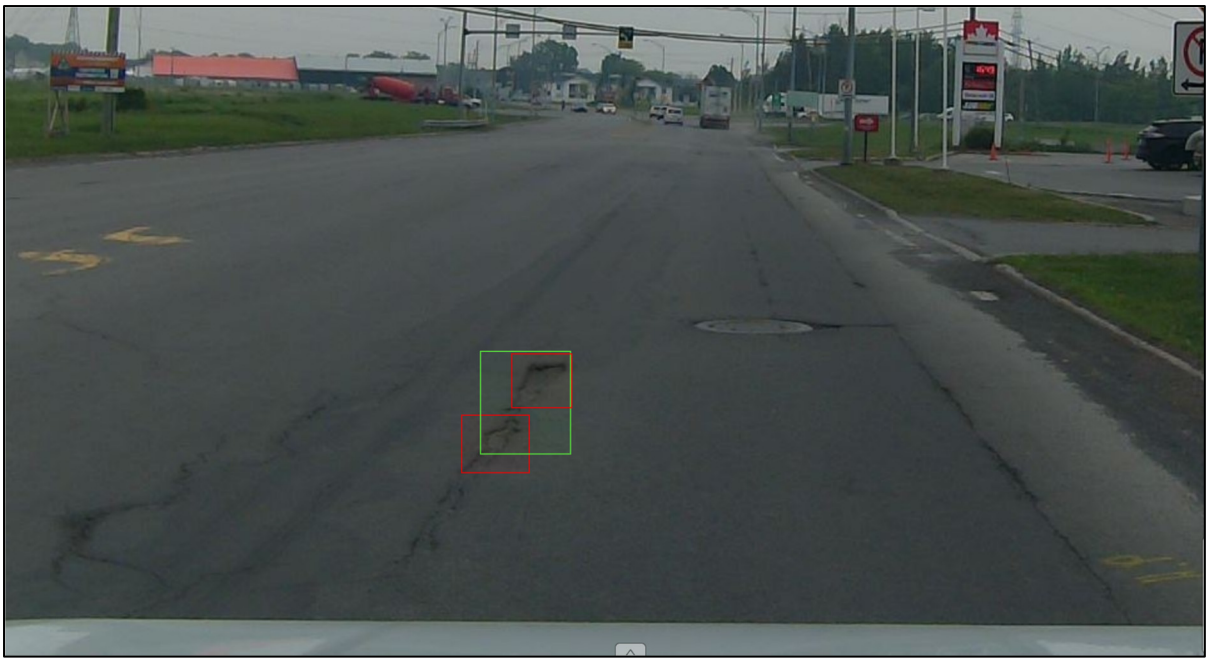

Figure 4.11 Exemple de détection de nids de poule (prédiction du modèle en rouge et notre annotation en vert)

Figure 4.12 Exemple d'annotation où deux nids de poule sont identifiés

Ainsi, pour améliorer les résultats, nous pensons que l'accent devrait être mis sur la révision du jeu de données pour en améliorer sa qualité et la fidélité conjointe des exemples. De plus, nous n'avons pas d'idée des nids de poule et de leur nombre que nous manquons en production, il faudrait développer des indicateurs dans ce sens.

4.3.2 Détection des lampadaires brisés

Dans cette partie, nous allons aborder les résultats du modèle de détection de lampadaires. Dans un premier temps, nous aborderons les performances issues de l'entrainement, puis nous verrons le comportement en production. Enfin, nous discuterons les résultats.

4.3.2.1 Entrainement

Contrairement au cas des nids de poule présenté en 4.3.1, le modèle de détection a connu beaucoup moins d'itération avant d'être performant. En effet, les exemples de lampadaires ne manquent pas et ne nécessitent pas de développer le début d'un modèle de détection pour trouver des exemples. Il a suffi de prendre des clichés de nuits sur les routes. D'autant plus que les lampadaires brisés ne sont pas des événements rares et s'identifient très bien la nuit (voir 3.1.4). Nous avons seulement utilisé le détecteur en production pour peaufiner ces quelques erreurs et non pour augmenter drastiquement le jeu de données.

Le modèle YOLOv5 entrainé obtient une mAP de 0.775 pour les lampadaires brisés (Tableau 4.2). Ce score est correct. Néanmoins, si on le regarde les résultats sur les lampadaires brisés au premier plan, on observe un score mAP50 de 0.84. C'est un bon score et une bonne indication sur les performances de notre modèle qui nous intéresse. En effet, un lampadaire brisé non détecté au second plan (loin sur la route) passera forcément au premier plan quand la voiture avancera.

Tableau 4.2 Résultats du modèle de détection de lampadaires

Catégorie du lot de test	Classe	Précision P	Rappel R	MAP50	MAP95
Premier plan (foreground)	lighting	0.857	0.959	0.965	0.756
	lightoff	0.893	0.785	0.840	0.402
	all	0.875	0.872	0.902	0.579
Test (lot entier)	lighting	0.841	0.811	0.873	0.474
	lightoff	0.710	0.841	0.775	0.332
	all	0.775	0.826	0.824	0.403

Les courbes en Figure 4.13, Figure 4.14 et Figure 4.15 illustrent les bons résultats obtenus en précision et rappel. On observe un plateau du score F1 à 0.8 sur l'intervalle 0.05 à 0.70 qui traduit le bon équilibre entre la précision et le rappel pour les lampadaires brisés au premier plan.

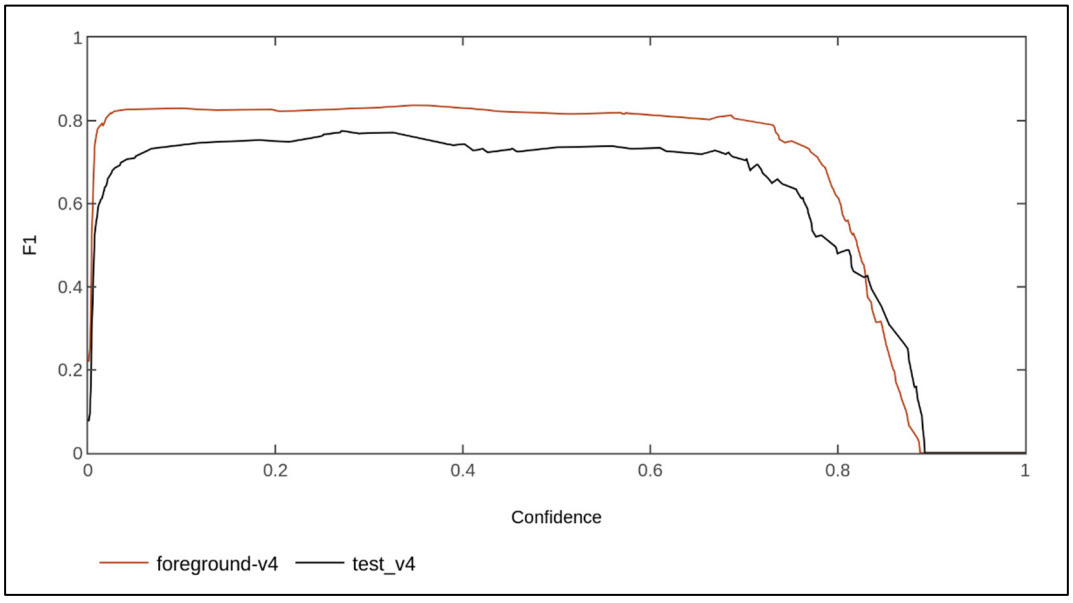

Figure 4.13 Courbes des scores F1 du modèle pour les lampadaires brisés au premier plan (foregroud-v4) et sur le lot de test entier (test_v4)

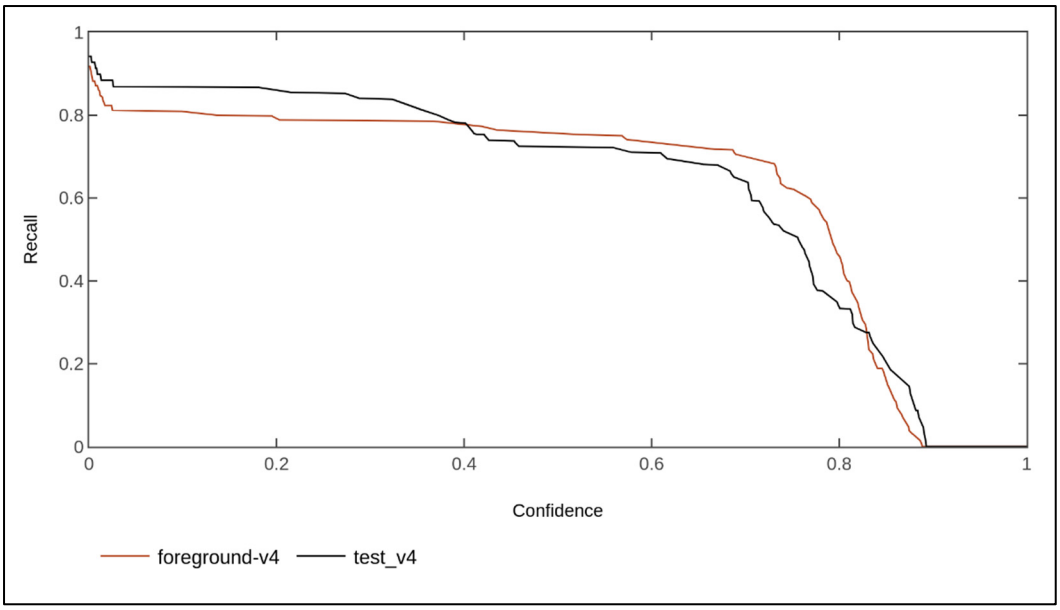

Figure 4.14 Courbes de rappel du modèle pour les lampadaires brisés au premier plan (foregroud-v4) et sur le lot de test entier (test_v4)

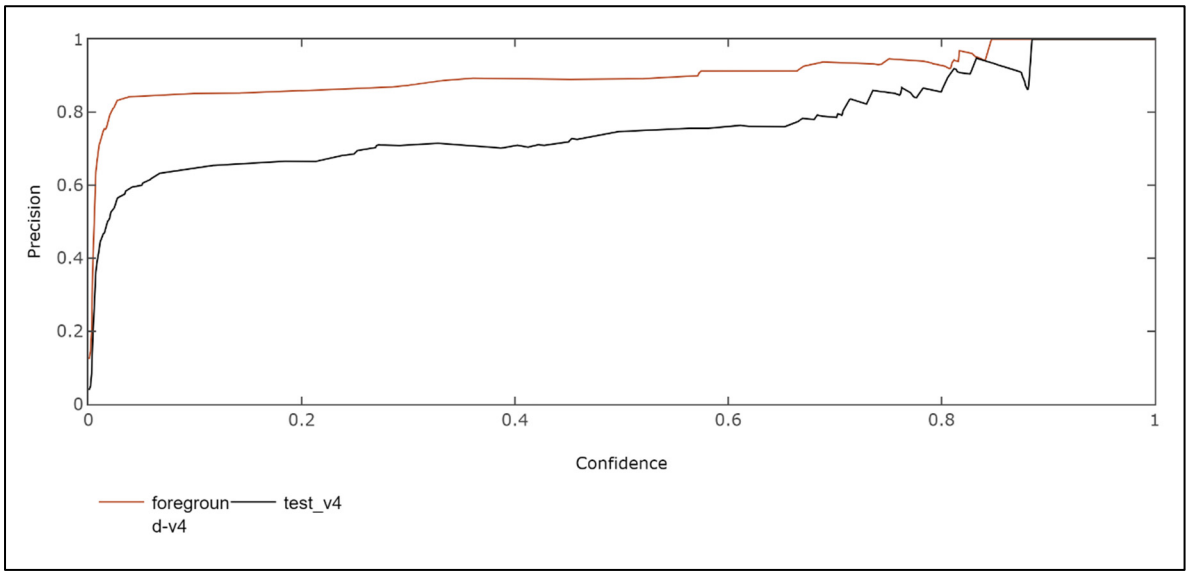

Figure 4.15 Courbes de précision du modèle pour les lampadaires brisés au premier plan (foregroud-v4) et sur le lot de test entier (test_v4)

4.3.2.2 Production

Ici, nous retrouvons un modèle où la proportion de faux positif sur 1958 détections de lampadaires brisés en production est très basse (2,2 %). Le modèle semble très fiable dans ses détections.

La distribution des faux positifs sur la Figure 4.16 permet de comprendre que pour ce modèle, un seuil de confiance plus bas que celui du modèle des nids de poule peut être choisi. En effet, même en nous plaçant au seuil le plus bas à 0.4, nous obtenons des objets détectés en production, 100 % des vrais positifs avec seulement 2 % des détections totales qui sont des faux positifs. D'autant plus qu'en nous plaçant au-dessus de 0.8, plus aucun faux positif n'apparait.

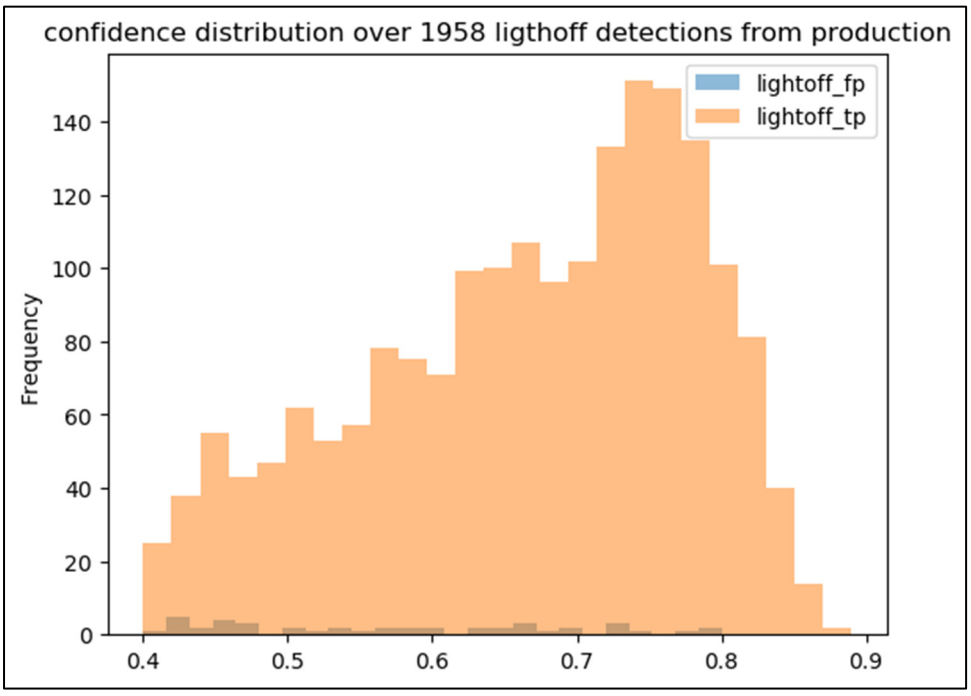

Figure 4.16 Histogramme de faux et vrais positifs selon leur confiance sur 1958 détections de lampadaires brisés, issues de production.

Quand même, il reste intéressant d'examiner les faux positifs avec un taux de confiance assez haut, même s'ils sont peu nombreux, pour essayer de perfectionner notre modèle lors du

prochain entrainement en essayant de diminuer encore le nombre de faux positifs problématiques.

Voici quelques exemples qu'il faudrait retrouver idéalement dans le jeu de données d'entrainement pour diminuer la probabilité de les détecter avec un taux de confiance élevé en Figure 4.17 et Figure 4.18 :

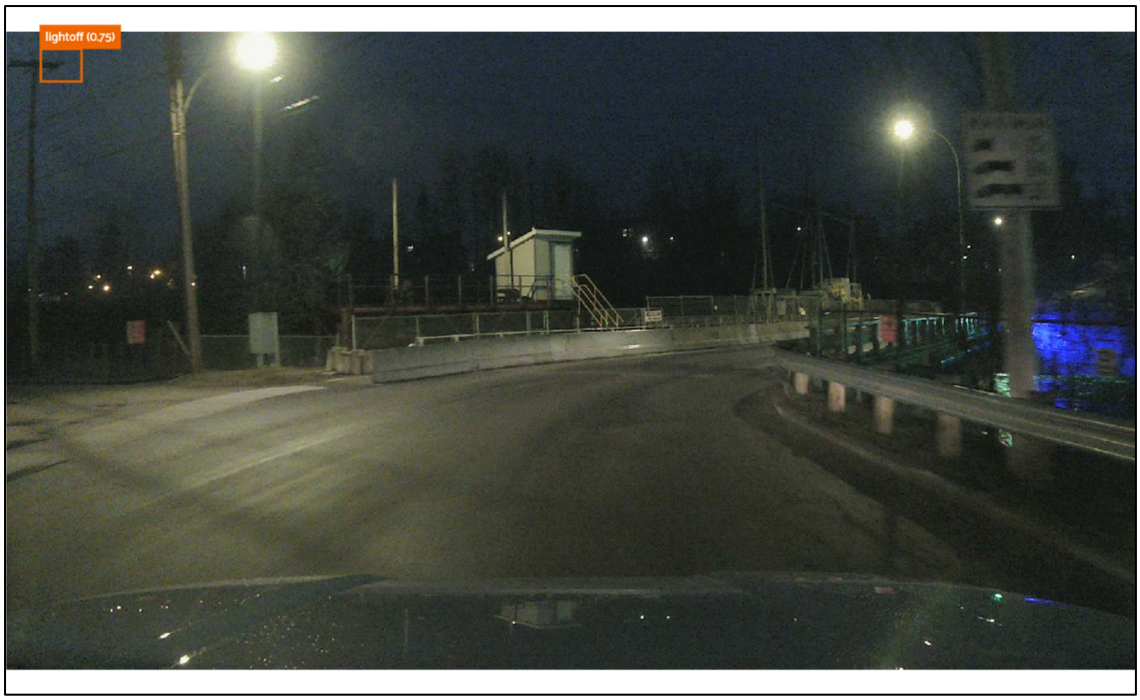

Figure 4.17 Haut de poteau électrique comme faux positif

Figure 4.18 Une boîte électrique comme faux positif

Attention, la mauvaise détection d'un objet du même type n'est pas systématique, car sa détection dépend de beaucoup de facteurs comme la luminosité ou l'angle de vue de la caméra. Par exemple, en Figure 4.18, le haut d'un poteau électrique a été détecté avec un taux de confiance de 0.75. Or c'est le seul exemple « problématique » détecté sur les 1000 images sélectionnées. Alors qu'il y a bon nombre de poteaux électriques sur le bord des routes dans ce lot d'images.

4.3.2.3 Discussion

Ce modèle est prometteur et satisfaisant d'après les résultats en production et à l'entrainement, il est assez fiable et précis. Cependant, une piste d'amélioration serait l'augmentation des performances en termes de rappel du modèle. En effet, c'est la métrique la plus basse d'après le modèle d'entrainement. Même si en production, nous observons un modèle fiable en termes de prédiction, rien ne nous indique la quantité potentielle de lampadaires non détectés par le modèle. Par conséquent, il serait intéressant d'augmenter les exemples de lampadaires brisés pour tendre aux 10000 occurrences (contre environ 2000 occurrences actuellement) préconisées dans la documentation.

CONCLUSION ET RECOMMANDATIONS

Ce mémoire s'inscrit dans une problématique générale entourant la détection automatique de l'environnement d'un véhicule autonome à l'aide de l'intelligence artificielle. Plus particulièrement, il contribue au développement de modèles de détection d'objet basés sur l'apprentissage profond et dédiés à la détection d'infrastructures défectueuses par caméra à bord d'un système embarqué. Ce rapport concerne trois cas d'infrastructures défectueuses au Québec : les nids de poule, les glissières de sécurité brisées et les lampadaires hors service.

L'objectif du projet est d'obtenir des modèles de détection d'infrastructures défectueuses basés sur les images d'une caméra sur un véhicule en mouvement. Ces modèles doivent être capables de détecter trois cas de figure au Québec : les nids de poule, les glissières de sécurité brisées et les lampadaires hors service. Ces modèles embarqués permettront in fine un meilleur entretien de notre réseau routier en augmentant considérablement la sécurité routière. De plus, les données récoltées permettront au MTQ de cartographier les routes du Québec et d'effectuer des analyses qui permettront une conception routière plus adaptée.

Afin d'atteindre notre objectif, nous utilisons le modèle YOLO comme détecteur d'objet basé sur l'apprentissage profond. Ses versions YOLOv5 et YOLOv8 sont reconnus comme des techniques performantes et adaptés aux infrastructures légères et limitées des systèmes embarqués.

Afin d'entrainer nos modèles de détections, nous avons construit des jeux de données sur les trois cas d'étude mentionnés, grâce au développement d'une infrastructure entière capable d'acquérir, filtrer et annoter des données. Cette infrastructure repose sur un système embarqué muni d'une caméra déployée dans plusieurs véhicules du MTQ, qui parcourent les routes au Québec. Au final, nous avons obtenu une base de données contenant de plus de 8000 images et 4 classes de détections.

Quant aux modèles de détection, les résultats sont contrastés. Nous n'avons pas été dans la capacité d'en développer un pour la détection de glissières de sécurité brisées. En effet, la flotte de voitures mise à disposition ne couvrait pas assez de terrain pour exploiter les coordonnées mis à disposition par le MTQ. D'autant plus que ces données n'étaient pas toujours assez fiables. Pour la détection de nids de poule, les résultats sont plus encourageants, le meilleur modèle obtient 0.729 mAP. Toutefois, dans l'objectif d'améliorer ces performances (qui restent largement perfectibles en production), le choix d'augmentation intensif de notre jeu de données par l'intermédiaire de l'apprentissage actif a perturbé les performances de notre modèle. La quasi-totalité des données provient de détection du modèle en production, ce qui peut être aussi pénalisant pour la généralisation du modèle. Pour les lampadaires, malgré un jeu de données loin des recommandations suggérées par Ultralytics (10000 instances par classe contre moins de 3000 pour les lampadaires brisés), nous avons obtenu des résultats très satisfaisants à l'entrainement (mAP50 de 0.84 sur les lampadaires brisés au premier plan) comme en production. Mais encore aucune étude n'a été menée sur ce que l'on pourrait manquer en production même si le jeu de données tests affiche un bon rappel de 0.841.

Malgré les résultats en demi-teinte, les modèles de détection développés portant sur les nids de poule et lampadaires font figure de bonnes bases pour la phase deux du projet. Le modèle pour les lampadaires brisés, grâce à sa fidélité dans ses détections, est déjà en production et permet de commencer à cartographier le Québec. Un accent pourrait être mis encore sur l'augmentation du nombre de données contribuant à son entrainement, pour essayer d'atteindre des performances encore meilleures. On pourrait aussi commencer à distinguer les types de forme de lampadaires et la technologie d'éclairage (LED ou incandescente) pour donner encore plus d'information au MTQ pour l'intervention. Quant à la détection des nids de poule, nous avons mis en évidence un problème de fidélité dans nos annotations. Nous pourrions utiliser le concept connu de k-fold Cross Validation afin de comprendre le comportement du modèle vis-à-vis du jeu de données d'entrainement et réannoter le jeu de données en conséquence. Ces modèles seraient entrainés sur une partie de notre jeu de données excluant la partie que nous

réannotons. En fait la méthode pourrait permettre de comparer les annotations à ce que des modèles entrainés détectent. Cela pourrait permettre, notamment, de souligner les problèmes de non-consistance dans l'annotation du jeu de données et les exemples ambiguës qui pénalisent le réseau. Aussi, On pourrait aussi à l'avenir sous catégoriser notre jeu de données entier sur les nids de poule par type (crack, petits, gros…) pour le balancer au mieux. La phase 2 permettra aussi de se concentrer plus sur la construction d'un jeu de données pour les glissières brisées, car la flotte de véhicules parcourant le Québec augmente rapidement ces derniers mois.

Enfin, les solutions de détection d'objets par apprentissage profond évoluent très rapidement. Nous pouvons penser à l'utilisation d'autres modèles comme YOLO-NAS (Deci, 2023) qui s'appuie sur la technologie de recherche d'architecture neuronale ou encore le RT-DETR (Lv et al., 2023) basé sur les transformateurs. Ces modèles ont prouvé leur grande performance sur des jeux de données comme COCO et viennent proposer des alternatives à l'architecture YOLOv8.

BIBLIOGRAPHIE

Affine (2021, 27 octobre). Data Augmentation For Deep Learning Algorithms. Repéré à https://affine.ai/data-augmentation-for-deep-learning-algorithms/

Arya, D., Maeda, H., Ghosh, S., Toshniwal, D., Omata, H., Kashiyama, T., & Sekimoto, Y. (2022). *Crowdsensing-based Road Damage Detection Challenge (CRDDC-2022)*. https://doi.org/10.48550/arXiv.2211.11362

Arya, D., Maeda, H., Ghosh, S., Toshniwal, D., & Sekimoto, Y. (2022). *RDD2022: A multi-national image dataset for automatic Road Damage Detection*. (S.l.) : (s.n.).

Bauk, S., Calvo, J. A. L., Mathar, R., & Schmeink, A. (2017). V2P/I communication for increasing occupational safety at a seaport. Dans *2017 International Symposium ELMAR* (pp. 79-82). https://doi.org/10.23919/ELMAR.2017.8124439

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. Dans A. Leonardis, H. Bischof, & A. Pinz (Éds), *Computer Vision – ECCV 2006* (pp. 404-417). Berlin, Heidelberg: Springer. https://doi.org/10.1007/11744023_32

Bharati, P., & Pramanik, A. (2020). Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey. Dans A. K. Das, J. Nayak, B. Naik, S. K. Pati, & D. Pelusi (Éds), *Computational Intelligence in Pattern Recognition* (pp. 657-668). Singapore: Springer. https://doi.org/10.1007/978-981-13-9042-5_56

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020, 22 avril). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv. https://doi.org/10.48550/arXiv.2004.10934

Bouraya, S., & Belangour, A. (2021). Deep Learning based Neck Models for Object Detection: A Review and a Benchmarking Study. *International Journal of Advanced Computer Science and Applications*, *12*(11)

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. Dans A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Éds), *Computer Vision – ECCV 2020* (pp. 213-229). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-58452-8_13

Chen, J., Liu, G., & Chen, X. (2019). *Road Crack Image Segmentation Using Global Context U-net* (p. 185). (S.l.): (s.n.). https://doi.org/10.1145/3374587.3374602

Chong, J. (2016). Véhicules autonomes et connectés : état d'avancement de la technologie et principaux enjeux stratégiques pour les pouvoirs publics au Canada.

Cohn, D. (2010). Active Learning. Dans C. Sammut & G. I. Webb (Éds), *Encyclopedia of Machine Learning* (pp. 10-14). Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-30164-8_6

CVAT. Repéré à https://www.cvat.ai/

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. Dans 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). San Diego, CA, USA : IEEE. https://doi.org/10.1109/CVPR.2005.177

Deci. (2023, 3 mai). YOLO-NAS by Deci Achieves State-of-the-Art Performance on Object Detection Using Neural Architecture Search. *Deci*. Repéré à https://deci.ai/blog/yolo-nas-object-detection-foundation-model/

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. Dans *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248-255). https://doi.org/10.1109/CVPR.2009.5206848

Dewangan, D., & Sahu, S. P. (2020). PotNet: Pothole detection for autonomous vehicle system using convolutional neural network. *Electronics Letters*, *57*. https://doi.org/10.1049/ell2.12062

Dhiman, A., & Klette, R. (2020). Pothole Detection Using Computer Vision and Learning. *IEEE Transactions on Intelligent Transportation Systems*, *21*(8), 3536-3550. https://doi.org/10.1109/TITS.2019.2931297

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … Houlsby, N. (2021, 3 juin). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv. Repéré à http://arxiv.org/abs/2010.11929

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., & Balakrishnan, H. (2008). The pothole patrol: using a mobile sensor network for road surface monitoring. Dans *Proceedings of the 6th international conference on Mobile systems, applications, and services* (pp. 29-39). Breckenridge CO USA : ACM. https://doi.org/10.1145/1378600.1378605

Fan, R., & Liu, M. (2020). Road Damage Detection Based on Unsupervised Disparity Map Segmentation. *IEEE Transactions on Intelligent Transportation Systems*, *21*(11), 4906-4911. https://doi.org/10.1109/TITS.2019.2947206

Fan, R., Ozgunalp, U., Hosking, B., Liu, M., & Pitas, I. (2020). Pothole Detection Based on Disparity Transformation and Road Surface Modeling. *IEEE Transactions on Image Processing*, *29*, 897-908. https://doi.org/10.1109/TIP.2019.2933750

Felzenszwalb, P., Girshick, R., Mcallester, D., & Ramanan, D. (2010). Object Detection with Discriminatively Trained Part-Based Models. *IEEE transactions on pattern analysis and machine intelligence*, *32*, 1627-45. https://doi.org/10.1109/TPAMI.2009.167

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Gläser, C., Timm, F., … Dietmayer, K. (2021). Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, *22*(3), 1341-1360. https://doi.org/10.1109/TITS.2020.2972974

Freund, Y., & Schapire, R. E. (1999). A Short Introduction to Boosting.

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021, 5 août). YOLOX: Exceeding YOLO Series in 2021. arXiv. Repéré à http://arxiv.org/abs/2107.08430

Girshick, R. (2015, 27 septembre). Fast R-CNN. arXiv. https://doi.org/10.48550/arXiv.1504.08083

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014, 22 octobre). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv. https://doi.org/10.48550/arXiv.1311.2524

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018, 24 janvier). Mask R-CNN. arXiv. https://doi.org/10.48550/arXiv.1703.06870

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*. https://doi.org/10.1109/TPAMI.2015.2389824

Hirz, M., & Walzel, B. (2018). Sensor and object recognition technologies for self-driving cars. *Computer-Aided Design and Applications*, *15*, 1-8. https://doi.org/10.1080/16864360.2017.1419638

Jocher, G. (2020, mai). YOLOv5 by Ultralytics. [Python]. https://doi.org/10.5281/zenodo.3908559

Kaggle (2020). Data Augmentation Tutorial: Basic, Cutout, Mixup. (2020). Repéré à https://kaggle.com/code/kaushal2896/data-augmentation-tutorial-basic-cutout-mixup

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35-45. https://doi.org/10.1115/1.3662552

Kathuria, Data Augmentation for Bounding Boxes: Scaling and Translation. (2018, 12 septembre). *Paperspace Blog*. Repéré à https://blog.paperspace.com/data-augmentation-bounding-boxes-scaling-translation/

Kim, T., & Ryu, S. (2014). Review and Analysis of Pothole Detection Methods. Repéré à https://www.semanticscholar.org/paper/Review-and-Analysis-of-Pothole-Detection-Methods-Kim-Ryu/41d8189111cc320a2fd8444b0bf13a6263f45807

Kim, Y.-M., Kim, Y.-G., Son, S.-Y., Lim, S.-Y., Choi, B.-Y., & Choi, D.-H. (2022). Review of Recent Automated Pothole-Detection Methods. *Applied Sciences*, *12*(11), 5320. https://doi.org/10.3390/app12115320

Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019, 10 avril). Panoptic Segmentation. arXiv. https://doi.org/10.48550/arXiv.1801.00868

Li, Q., Yao, M., Yao, X., & Xu, B. (2009). A real-time 3D scanning system for pavement distortion inspection. *Measurement Science and Technology*, *21*(1), 015702. https://doi.org/10.1088/0957-0233/21/1/015702

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., … Dollár, P. (2015, 20 février). Microsoft COCO: Common Objects in Context. arXiv. https://doi.org/10.48550/arXiv.1405.0312

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, *128*(2), 261-318. https://doi.org/10.1007/s11263-019-01247-4

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8759-8768. https://doi.org/10.1109/CVPR.2018.00913

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector (Vol. 9905, pp. 21-37). (S.l.): (s.n.). (arXiv:1512.02325 [cs]). https://doi.org/10.1007/978-3-319-46448-0_2

Long, J., Shelhamer, E., & Darrell, T. (2015, 8 mars). Fully Convolutional Networks for Semantic Segmentation. arXiv. https://doi.org/10.48550/arXiv.1411.4038

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, *60*(2), 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Lv, W., Zhao, Y., Xu, S., Wei, J., Wang, G., Cui, C., … Liu, Y. (2023, 6 juillet). DETRs Beat YOLOs on Real-time Object Detection. arXiv. https://doi.org/10.48550/arXiv.2304.08069

Ma, N., Fan, J., Wang, W., Wu, J., Jiang, Y., Xie, L., & Fan, R. (2022). Computer vision for road imaging and pothole detection: a state-of-the-art review of systems and algorithms. *Transportation Safety and Environment*, *4*(4), tdac026. https://doi.org/10.1093/tse/tdac026

Mednis, A., Strazdins, G., Zviedris, R., Kanonirs, G., & Selavo, L. (2011). *Real Time Pothole Detection Using Android Smartphones with Accelerometers* (p. 6). https://doi.org/10.1109/DCOSS.2011.5982206

Ministère des Transports du Québec. (2021). Bilan de l'état des chaussées du réseau routier supérieur québécois 2021. Repéré à https://www.transports.gouv.qc.ca/fr/entreprises-partenaires/entreprises-reseaux-routier/chaussees/Documents/bilan-chaussees.pdf

Ministère du Transport du Québec (2022). Rapport annuel de gestion 2021-2022. Repéré à https://cdn-contenu.quebec.ca/cdn-contenu/adm/min/transports/ministere-des-transports/publications-amd/rapport-annuel-de-gestion/RA_rapport_annuel_2021-2022_MTQ.pdf

Nienaber, S., Booysen, M. J. (Thinus), & Kroon, S. (2015, 13 octobre). Dataset of images used for pothole detection.

OpenMMLab. (2023, 17 janvier). Dive into YOLOv8: How does this state-of-the-art model work? *Medium*. Repéré à https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

Park, S.-S., Tran, V.-T., & Lee, D.-E. (2021). Application of Various YOLO Models for Computer Vision-Based Real-Time Pothole Detection. *Applied Sciences*, *11*(23), 11229. https://doi.org/10.3390/app112311229P

Pothole Image Data-Set (2019). Repéré à https://www.kaggle.com/datasets/sachinpatel21/pothole-image-dataset

Pytorch (2017). Illustration of transforms — Torchvision 0.10.0 documentation. Repéré à https://pytorch.org/vision/0.10/auto_examples/plot_transforms.html#sphx-glr-auto-examples-plot-transforms-py

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, 9 mai). You Only Look Once: Unified, Real-Time Object Detection. arXiv. https://doi.org/10.48550/arXiv.1506.02640

Redmon, J., & Farhadi, A. (2016, 25 décembre). YOLO9000: Better, Faster, Stronger. arXiv. https://doi.org/10.48550/arXiv.1612.08242

Redmon, J., & Farhadi, A. (2018, 8 avril). YOLOv3: An Incremental Improvement. arXiv. https://doi.org/10.48550/arXiv.1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2016, 6 janvier). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv. https://doi.org/10.48550/arXiv.1506.01497

RoadDamageDetector. (2023, 5 juillet). [Jupyter Notebook], sekilab. Repéré à https://github.com/sekilab/RoadDamageDetector (Ouvrage original publié en 9 janvier 2018).

Terven, J., & Cordova-Esparza, D. (2023, 9 juin). A Comprehensive Review of YOLO: From YOLOv1 and Beyond. arXiv. https://doi.org/10.48550/arXiv.2304.00501

tryolabs/norfair. (2023, 17 juillet). [Python], Tryolabs. Repéré à https://github.com/tryolabs/ norfair (Ouvrage original publié en 1 juillet 2020).

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, *104*(2), 154-171. https://doi.org/10.1007/s11263-013-0620-5

Ultralytics. Hyperparameter evolution. Repéré à https://docs.ultralytics.com/yolov5/tutorials/hyperparameter_evolution

Ultralytics (2022). Overview of model structure about YOLOv5 Issue #280 ultralytics/yolov5. *GitHub*. Repéré à https://github.com/ultralytics/yolov5/issues/280

Ultralytics (2023). Architecture Summary. Repéré à https://docs.ultralytics.com/yolov5/tutorials/architecture_description

Ultralytics (2023). Tips for Best Training Results. Repéré à https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results

Ultralytics (2023). Ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > CoreML > TFLite. Repéré à https://github.com/ultralytics/ultralytics

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is All you Need. Dans *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc. Repéré à https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Vinel, A., Lyamin, N., & Isachenkov, P. (2018). Modeling of V2V Communications for C-ITS Safety Applications: A CPS Perspective. *IEEE Communications Letters*, *22*(8), 1600-1603. https://doi.org/10.1109/LCOMM.2018.2835484

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Dans *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (Vol. 1, p. I-511-I-518). Kauai, HI, USA: IEEE Comput. Soc. https://doi.org/10.1109/CVPR.2001.990517

Wang, C., Luo, Z., Zhong, Z., & Li, S. (2021). SAFD: single shot anchor free face detector. *Multimedia Tools and Applications*, *80*. https://doi.org/10.1007/s11042-020-10401-x

Wang, C.-Y., Liao, H.-Y. M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., & Hsieh, J.-W. (2019, 26 novembre). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. arXiv. https://doi.org/10.48550/arXiv.1911.11929

Wang, K., Liew, J. H., Zou, Y., Zhou, D., & Feng, J. (2019). PANet: Few-Shot Image Semantic Segmentation With Prototype Alignment. Dans *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 9196-9205). Seoul, Korea (South): IEEE. https://doi.org/10.1109/ICCV.2019.00929

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., & Lee, B. (2022). A survey of modern deep learning based object detection models. *Digital Signal Processing*, *126*, 103514. https://doi.org/10.1016/j.dsp.2022.103514

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018, 27 avril). mixup: Beyond Empirical Risk Minimization. arXiv. https://doi.org/10.48550/arXiv.1710.09412

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*, *111*(3), 257-276. https://doi.org/10.1109/JPROC.2023.3238524