MINISTERE DE L'ENSEIGEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB –BLIDA 01

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

MÉMOIRE DE MASTER II

Spécialités : Traitement Automatique de la Langue

& Ingénierie de Logiciels

Organisme d'accueil : D.T.P

THEME

Système Intelligent de signalement des Dommages Routiers

Présenté par :

- ✓ **Mlle Slimani Mouna**
- ✓ **Mlle Yakhlef chaima**

Devant le jury composé de :

- ✓ **Mme.BEY.F** — **Promotrice**
- ✓ **Mme.Mezzi Melyara**
- ✓ **Mme.Djemia.N**

Résumé

Les dommages routiers (fissures, nids-de-poule, dégradations) constituent un enjeu majeur pour la sécurité des usagers et la durabilité des infrastructures. Face aux limites des méthodes d'inspection manuelles, ce projet explore l'apport des technologies d'intelligence artificielle pour automatiser ce processus critique.

Nous proposons une solution basée sur l'intelligence artificielle pour automatiser la détection et la classification des dommages routiers (fissures, nids-de-poule, dégradations) à partir d'images, en utilisant l'algorithme YOLOv8. Entraîné sur le dataset RDD-2022, le modèle atteint une précision de 85 % sur les classes critiques (D10, D20, D40) définies par la Direction des Travaux Publics (DTP). Une interface simplifiée permet de tester le système sans expertise technique.

Mots-clés : Apprentissage profond, YOLOv8, Vision par ordinateur , Dommages routiers (D00-D43), DTP, Temps réel .

Abstract

Road damage (cracks, potholes, deterioration) is a major issue for user safety and infrastructure durability. Faced with the limitations of manual inspection methods, this project explores the contribution of artificial intelligence technologies to automate this critical process.

We propose a solution based on artificial intelligence to automate the detection and classification of road damage (cracks, potholes, deterioration) from images, using the YOLOv8 algorithm. Trained on the RDD-2022 dataset, the model achieves 85% accuracy on the critical classes (D10, D20, D40) defined by the Direction des Travaux Publics (DTP). A simplified interface makes it possible to test the system without technical expertise.

Keywords: Deep learning, YOLOv8, Computer vision , Road damage (D00-D43), DTP, Real time .

ملخص

يعد تلف الطرق (التشققات والحفر والتدهور) مشكلة كبيرة بالنسبة لسلامة مستخدمي الطرق ومتانة البنية التحتية. وفي مواجهة محدودية طرق الفحص اليدوي، يستكشف هذا المشروع مساهمة تقنيات الذكاء الاصطناعي في أتمتة هذه العملية الحرجة.

سوف نقترح حلاً يعتمد على الذكاء الاصطناعي لأتمتة اكتشاف وتصنيف الأضرار التي لحقت بالطرق (الشقوق والحفر والتدهور) من الصور، باستخدام خوارزميةYOLOv 8. يحقق النموذج، الذي تم تدريبه على مجموعة بيانات-RDD 2022، دقة بنسبة 85% على الفئات الحرجة D10(D، D20، 40) التي حددتها مديرية الأشغال العامة .(DTP) تتيح الواجهة المبسطة إمكانية اختبار النظام دون خبرة تقنية.

الكلمات المفتاحية: التعلم العميقYOLOv ،8، الرؤية الحاسوبية، أضرار الطرقD) D43-D00 ،(D00-43 ، DTP، الوقت الحقيقي.

Dédicase

بسم الله الرحمان الرحيم

بعد الصلاة و السلام على سيد الخلق و أشرف الأنبياء سيدنا محمد و على آله و صحبه أجمعين و من والاه
بإخلاص إلى يوم الدين الحمدلله الذي بنعمته تتم الصالحات الحمد لله الذي هداني وسددني لإتمام هذا العمل بعونه
و توفيقه فالحمدلله كما ينبغي لجلال وجهه و عظيم سلطانه

أتقدم أيضا بجزيل الشكر والامتنان لعائلتي الغالية، والداي وإخوتي، على حبهم ودعواتهم ووجودهم الدائم بجانبي
الذي أنار دربي طوال هذا المشوار،

ولزوجي العزيز على دعمه اللا محدود وصبره وتشجيعه الدائم الذي مكنني من تجاوز التحديات وإنجاز هذا
العمل،

وللمسؤولين في مديرية الأشغال العمومية لحسين داي على تيسير الأمور وتزويدنا بكل البيانات اللازمة لإتمام
هذا العمل، ولأصدقائي وزملائي على كلماتهم المحفزة ودعمهم المعنوي.
كما أشكر كل من ساهم من قريب أو بعيد بكلمة أو نصيحة أو دعاء
أهدي هذا العمل لعائلتي وزوجي وكل من ساندني، سائلة الله أن يكون خالصًا لوجهه الكريم ونافعًا للجميع.

MOUNA

Dédicase

الرَّحِيمِ الرَّحْمَنِ اللهِ بِسْمِ

.أَجْمَعِينَ وَصَحْبِهِ آلِهِ وَعَلَى مُحَمَّدٍ سَيِّدِنَا عَلَى وَالسَّلَامُ وَالصَّلَاةُ الْعَالَمِينَ، رَبِّ لِلَّهِ الْحَمْدُ

إِلَى الْعَمَلَ هَذَا أُهْدِي:

- **الْعَزِيزَيْنِ وَالِدَيَّ**، اللذان ضحَّيا من أجلي بكل غالٍ ونفيس، وزرعا فيّ حب العلم والعمل.

- **الأَفَاضِل إِخْوَتِي**، مصدر خطوة كل في سندي لكونهم الدائم إلهامي ومصدر.

وَخُصُوصًا:

- **الأَعِزَّاء أَصْدِقَائِي**، لوقوفكم شكرًا يضيء نورًا كانت التي ولضحكاتكم اللحظات، أصعب في بجواري طريقي.

- **الدِّرَاسِي الْمَجَالِ فِي زُمَلَائِي**، لتعاونكم العمل هذا أغنت التي ونصائحكم.

مَنْ وَلِكُلِّ:

- نفسي في شككتُ حين حتى بي آمن.

- صادقة دعوة أو طيبة كلمة لي قدَّم.

وَأَخِيرًا:

.ذُخْرُكَ إِلَّا ذُخْرَ لَا يَوْمَ وَلِأَهْلِي لِي وَذُخْرًا لِوَجْهِكَ، خَالِصًا اجْعَلْهُ اللَّهُمَّ

الله بحمد تمت

CHAIMA

Remerciement

Nous tenons avant tout à rendre grâce à **Dieu Tout-Puissant**, source de toute connaissance et de toute persévérance, qui nous a accompagnés dans l'aboutissement de ce travail.

Nos sincères remerciements vont aux **enseignants du département d'informatique** pour leur guidance éclairée, leur disponibilité et leurs précieux conseils tout au long de ce parcours académique. Leur expertise a été un atout majeur dans la réalisation de ce mémoire.

Nous exprimons notre profonde gratitude aux **membres du jury** qui ont accepté d'évaluer ce travail avec rigueur et bienveillance. Leurs commentaires avisés ont permis d'en enrichir la qualité.

Une pensée particulière va à notre famille et à nos proches pour leur soutien constant et leurs encouragements tout au long de cette aventure intellectuelle.

Liste de figures

Liste des tableaux

Table des matières

Liste des abréviations

CNN : Réseau de Neurones Convolutionnel

DTP : Direction des Traveaux publiques

IA : Intelligence Artificielle

RDD : Road Damage Detection Dataset

RNA : Réseaux de Neurones Artificiels

RPN : Réseau de Proposition de Régions

YOLO : You Only Look One

SSD : Single Shot MultiBox Detector

Introduction générale

Les infrastructures routières sont d' une importance capitale pour la croissance économique et la sécurité des usagers . Leur détérioration continue pose cependant d'énormes défis aux gestionnaires publics, tels que la Direction des Travaux Publics (DTP) algérienne . Actuellement , la détection et la classification des défauts routiers (fissures , nids-de-poule , détériorations) reposent encore sur des approches conventionnelles, longues et peu fiables. Face à ces défis , les technologies d' intelligence artificielle , et plus particulièrement les modèles de vision par ordinateur , apportent des solutions innovantes . Parmi elles , YOLOv8 (You Only Look Once) se distingue par sa capacité à détecter et classer les objets en temps réel avec une grande précision .

Dans ce projet , nous exploitons ce modèle pour automatiser la classification des dommages routiers à partir d'images basées sur un ensemble de données annotées (RDD-2022) et en transformant les résultats aux normes DTP locales (classes D00-D43).

L' objectif principal est de créer un système efficace pour :

L'analyse automatique d' images de chaussées via YOLOv8.

Répartition des dommages et intérêts selon les modalités préétabliesCatégories .

Une interface utilisateur pour vérifier et confirmer les résultats .

Ce mémoire propose ainsi une solution concrète pour repenser les procédures de maintenance routière grâce à l' intégration de l'apprentissage profond, du traitement d'images et des besoins opérationnels . Les résultats obtenus démontrent le potentiel de l' IA pour améliorer la rapidité et la précision des diagnostics, et minimiser les coûts des inspections traditionnelles .

Ce mémoire est structuré autour de trois chapitres complémentaires qui contextualisent notre approche de recherche. Le premier chapitre présente le contexte théorique en décrivant les concepts de base de l'apprentissage profond et les solutions existantes pour la détection des dommages routiers, afin de justifier la sélection de l'algorithme YOLOv8. Le deuxième chapitre porte sur la conception du système en expliquant l'architecture technique adoptée et les choix algorithmiques spécifiques faits pour répondre aux exigences opérationnelles de la DTP. Enfin, le troisième chapitre porte sur les pratiques de mise en œuvre et les performances qui en résultent, en examinant l'environnement technique utilisé, l'ensemble de données d'entraînement et un examen approfondi des performances du modèle mis en œuvre.

I.Présentation de la Société d'Accueil

1-Définition :

La Direction des Travaux Publics de la wilaya d'Alger travaille à la collecte d'informations susceptibles d'aménager, de réparer et d'entretenir les ouvrages de base.

Assurer la mise en œuvre des mesures prescrites, et s'employer à respecter les normes d'exploitation des installations de base et les normes pour leur étude, leur réalisation et leur entretien en vue d'atteindre l'ordre et le domaine public des routes maritimes à l'exception des voies portuaires propriétés, et de mettre en œuvre ses mesures dans le cadre des installations de base et de leur exploitation.

Parmi ses tâches figure l'organisation et la fourniture d'une assistance technique aux municipalités dans le cadre de travaux d'entretien des routes et avenues municipales et d'assurer l'accomplissement du trafic et du trafic maritime.

Figure 1 : Siège de la direction des travaux publics Alger

2-Historique :

La Direction des Travaux Publics de la wilaya Alger a connu plusieurs désignations de 1962 jusqu'au décret 01/2000 du 01 mars 2000 relatif à l'administration de wilaya Alger. Elle a retrouvé son ancien nom, qui est la Direction des travaux publics de la wilaya Alger.

3-Missions :

Le compte rendu du décret exécutif n° 328/90 du 8 février 1411, correspondant au 27 octobre 1990, énonce les règles régissant les intérêts du traitement par l'État et de son travail.
Les missions de la Direction des travaux publics sont déterminées :

1. Recueillir, compiler et analyser les données qui développeront, répareront et entretiendront les installations de base, et assureront l'application des mesures établies.
2. S'assurer que les mesures d'exploitation des installations de base et les mesures de leur étude, de leur achèvement et de leur entretien sont respectées.
3. Réalisation du système dans la propriété publique des routes et de la mer à l'exception de la propriété publique de l'eau dans le cadre de la législation applicable.
4. Mettre en œuvre des mesures qui développeraient et entreteniraientles installations de base.
5. Proposer de classer les routes et de modifier leur classification.
6. Organiser l'assistance technique au profit des municipalités et les mettre à disposition en ce qui concerne les travaux d'entretien urbain et les routes municipales.
7. Assurer l'achèvement des feux de circulation et des signaux maritimes.

4- Présentations des Services de la direction des travaux publics:

La direction est organisée en quatre (4) services comprennent :
-Le Service de l'administration et des moyens.
-Le Service de l'administration et des moyens.
-Le Service des infrastructures Maritimes et Aéroportuaires.
-Le Service de l'exploitation et de l'entretien du réseau routier.

5-Organigramme de l'organisme d'accueil :

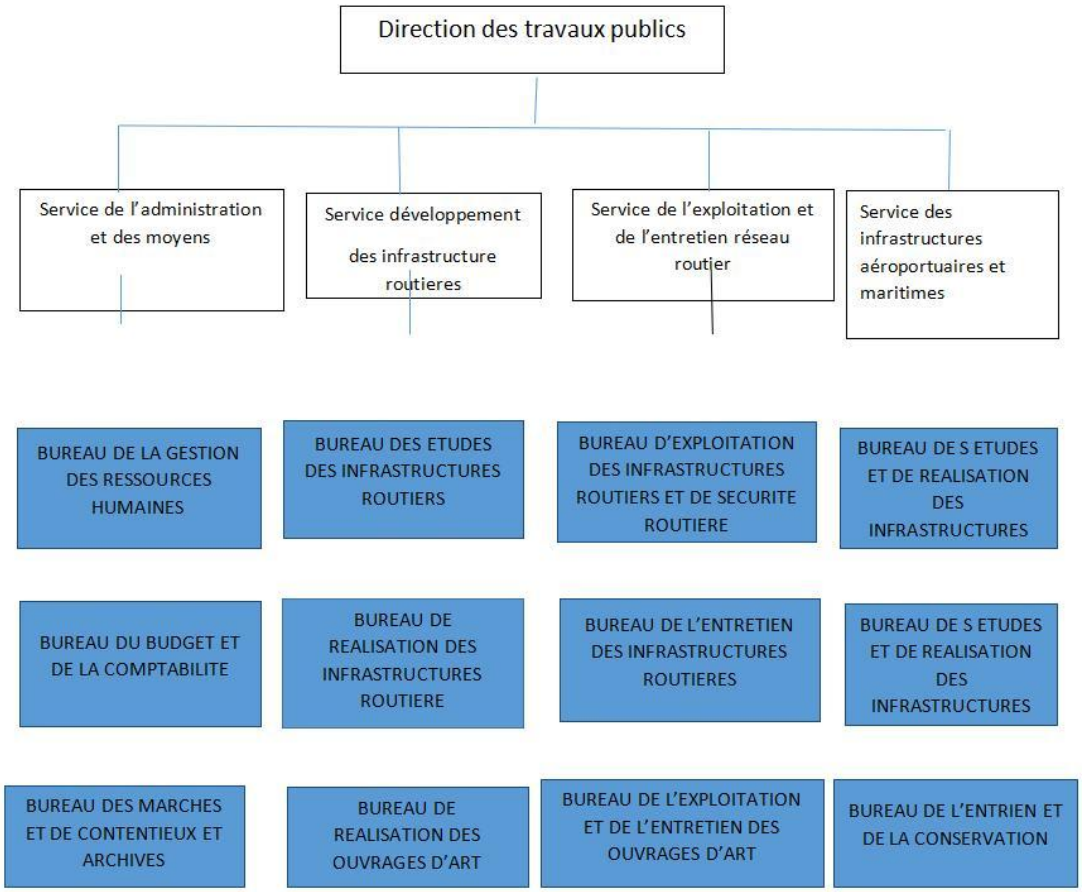

Figure 2 : organigramme de la DTP Wilaya d'Alger

II Problématique

À Alger, tout comme dans les grandes villes denses , l' état des routes s'aggrave rapidement grâce à plusieurs raisons :
-Trafic dense : Une circulation routière importante contribue à accélérer l' usure des chaussées.
-Conditions climatiques : Les pluies diluviennes etles températures de froid entraînent la formation de fissures et de nids-de-poule.
- Retards dans les réparations : Les signalements manuels (via téléphone, courriers, ou des canaux non numérisés) entraînent des -delais de traitement importants , ce qui entretient les risques d ' accidents et de détérioration s'accumulent.

Aujourd'hui , La Direction des Travaux Publics (DTP) est confrontée à des défis importants en matière de Les méthodes d'inspection actuelles présentent plusieurs défauts essentiels . Elles reposent essentiellement sur des opérations manuelles , ce qui implique une lenteur considérable (quelques jours pour scanner une zone), une subjectivité des diagnostics (différents d' un inspecteur à l'autre) et un coût exorbitant (mobilisation massive du personnel et déplacements fréquents). Ces limitations ont des conséquences directes et néfastes : retards dans les opérations
de réparation , escalade des risques pour la sécurité routière et gaspillage des ressources , entraînant une mauvaise optimisation des budgets d' entretien routier alloués .

«Comment concevoir un système automatisé, basé sur l'IA, pour détecter et classer rapidement les dommages routiers, tout en s'adaptant aux contraintes techniques et opérationnelles de la DTP ?»

III Objectif du projet

C’est du Développer un système intelligent de diagnostic routier basé sur l'IA (YOLOv8) permettant d'automatiser la détection et la classification des dommages routiers (fissures, nids-de-poule, dégradations) selon les normes de la Direction des Travaux Publics (DTP), afin d'améliorer l'efficacité des inspections et d'optimiser les budgets de maintenance par :

- Développer un modèle IA performant :
  - Utiliser **YOLOv8** pour classer les dommages (classes D10, D20, D40).
  -Atteindre une précision **> %** (mAP@0.5) sur le dataset **RDD-2022**.
- Créer une interface intuitive :
  -Permettre aux agents de la DTP de tester le modèle via une **application locale**
  -Afficher les résultats sous forme visuelle (bounding boxes + taux de confiance) et les envoyer au courier du la DTP.

Chapitre 1:

Fondements Théoriques et Travaux Antérieurs

Introduction

Ce chapitre pose les bases conceptuelles et techniques nécessaires à la compréhension de notre travail . Nous y présentons d'abord les bases de l'apprentissage profond et de la vision par ordinateur , en mettant l'accent sur leur applicabilité à la détection d'anomalies routières . Nous passons ensuite en revue les travaux actuels dans ce domaine et discutons de leurs mérites et de leurs limites . Enfin , nous justifions notre choix d' utiliser YOLOv8 comme solution optimale pour répondre aux besoins de la Direction des Travaux Publics (DTP). Ce contexte théorique nous permettra de mieux comprendre les aspects techniques abordés dans les chapitres suivants , où nous procéderons à la mise en œuvre concrète et à la conception de notre système.

1.1 Concepts fondamentaux en Deep Learning :

L'apprentissage profond est un sous-domaine de l'intelligence artificielle (figure 3) qui implique l'utilisation de réseaux de neurones pour modéliser et résoudre des problèmes complexes [1]. Le terme « profond » fait référence à l'utilisation de multiples couches dans le réseau .

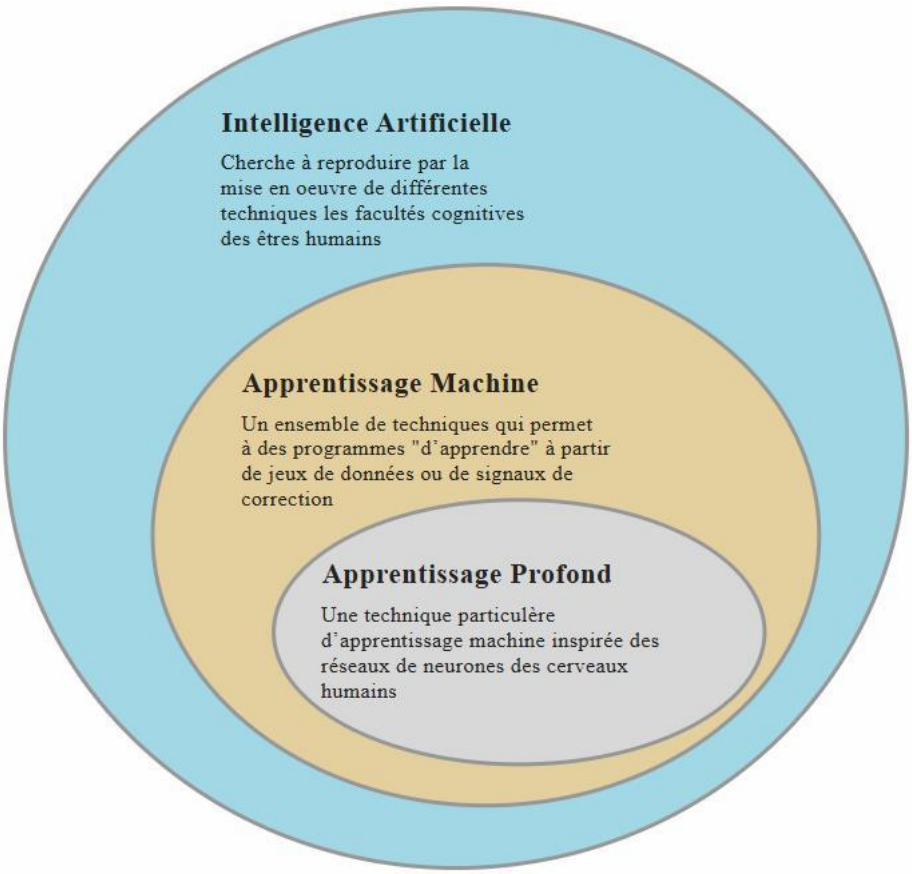

Figure 3 : Architecture conceptuelle des domaines de l'Intelligence Artificielle [2]

Il concerne les RNA (Réseaux de Neurones Artificiels) qui sont des modèles computationnels inspirés par la structure et le fonctionnement des réseaux de neurones biologiques, comme le cerveau humain. Les réseaux de neurones artificiels (RNA), une famille de techniques d'apprentissage machine, permettent de capturer des patrons par un processus de propagation de données à travers un ensemble d'unités liées entre elles (neurones) [3] . Ces réseaux sont entraînés à l'aide de vastes jeux de données pour effectuer des tâches telles que la classification, la régression, la reconnaissance de formes, et bien plus encore..

Les neurones de ces réseaux sont généralement organisés en trois couches; (Figure 4) l'information se propage d'une couche à une autre en commençant par :

- **Couche d'entrée :** Cette couche reçoit les données d'entrée et les transmet à la couche suivante. Le nombre de neurones dans cette couche dépend du nombre d'attributs (caractéristiques) des données d'entrée (chaque neurone représente une caractéristique). [1]
- **Couches cachées :** Ces couches intermédiaires traitent les données d'entrée par le biais de connexions pondérées et de fonctions d'activation. Elles permettent au réseau d'apprendre à partir de motifs complexes et de relations au sein des données. [1]
- **Couche de sortie :** La couche de sortie fournit le résultat final ou la prédiction basée sur les informations traitées par les couches cachées. Le nombre de neurones dans la couche de sortie dépend du problème donné. [1]

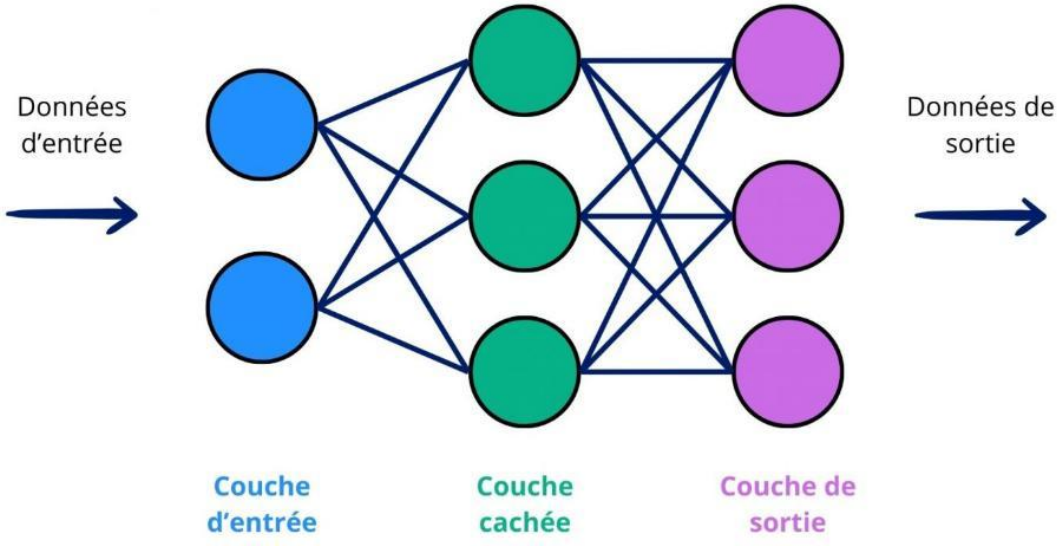

Figure 4 : Réseau de neurones artificiels [3]

Entraînement des RNA

L'entraînement des Réseaux de Neurones Artificiels implique deux phases principales :

1-Propagation avant (Feed forward) : fait référence au processus de calcul de la sortie d'un réseau de neurones étant donné certaines données d'entrée. Les données d'entrée sont transmises à travers une série de couches jusqu'à ce que la couche de sortie soit calculée (Figure 5). L'objectif de la propagation avant est de produire une sortie qui soit aussi proche que possible des étiquettes de référence [1].

2-Rétropropagation (Backpropagation) : est une technique fondamentale dans l'entraînement des réseaux de neurones, qui est très appréciée pour sa mise en œuvre directe, sa simplicité de programmation et son adaptabilité à diverses architectures de réseaux. La méthode parcourt le réseau dans l'ordre inverse, de la couche de sortie vers la couche d'entrée, comme illustré dans la Figure 5. [1]

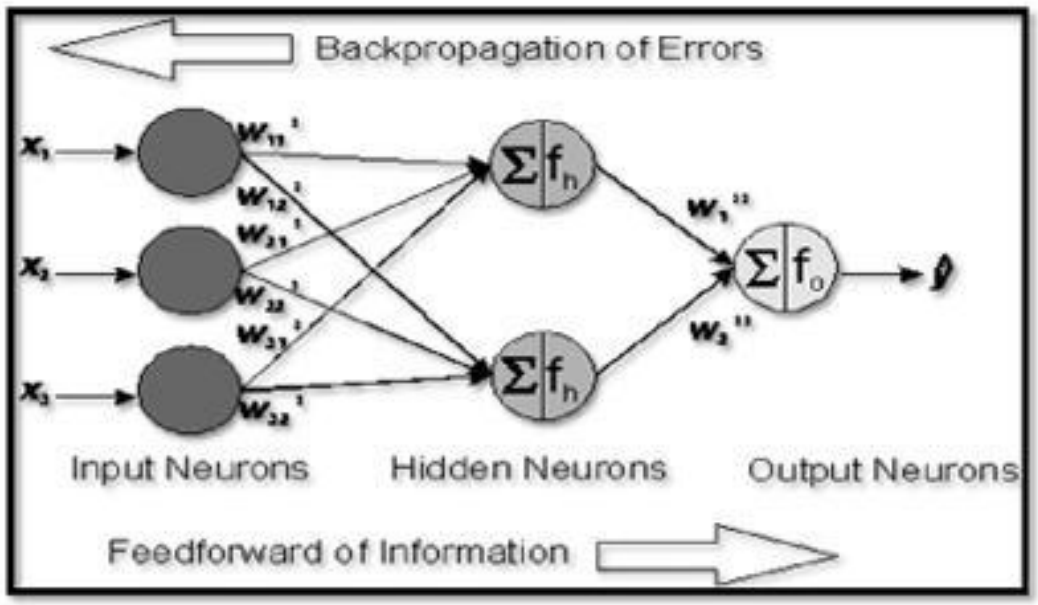

Figure 5 : Architecture d'un réseau de rétropropagation de type feed-forward [1]

3-Fonctions d'activation :

sont des équations mathématiques qui permettent aux réseaux de neurones d'apprendre des motifs complexes dans les données, et de décider si un neurone doit être activé ou non, en calculant la somme pondérée et en y ajoutant un biais. Elles transforment l'entrée de la couche précédente dans le réseau de neurones en un signal de sortie qui est transmis à la couche suivante. Les fonctions d'activation introduisent des non-linéarités qui permettent au réseau de neurones d'apprendre des correspondances très complexes entre les entrées et les sorties [4]

1.2 Vision par Ordinateur :

La vision par ordinateur (VO)est un domaine de l'intelligence artificielle (IA) qui utilise l'apprentissage automatique et les réseaux neuronaux pour apprendre aux ordinateurs et aux systèmes à dériver des informations significatives à partir d'images numériques, de vidéos et d'autres entrées visuelles, et pour faire des recommandations ou prendre des mesures lorsqu'ils détectent des défauts ou des problèmes.Si l'IA permet aux ordinateurs de penser, la vision par ordinateur leur permet de voir, d'observer et de comprendre[5].

La vision par ordinateur fonctionne à peu près de la même manière que la vision humaine, sauf que les humains ont une longueur d'avance. La vue humaine a l'avantage de bénéficier de toute une vie de contexte pour apprendre à différencier les objets, à déterminer leur distance, s'ils sont en mouvement ou si quelque chose ne va pas dans une image.

Elle entraîne les machines à exécuter ces fonctions, mais elle doit le faire en beaucoup moins de temps en utilisant des caméras, des données et des algorithmes à la place des rétines, des nerfs optiques et d'un cortex visuel. Un système formé à l'inspection de produits ou à la surveillance d'un outil de production peut analyser des milliers de produits ou de processus à la minute, en remarquant des défauts ou des problèmes imperceptibles, ce qui lui permet de surpasser rapidement les capacités humaines[5].

Voici quelques exemples de tâches de vision par ordinateur :

La classification d'images voit une image et peut la classer (un chien, une pomme, le visage d'une personne). Plus précisément, elle est capable de prédire avec précision qu'une image donnée appartient à une certaine classe. Par exemple, une société de médias sociaux pourrait vouloir l'utiliser pour identifier et séparer automatiquement les images répréhensibles téléchargées par les utilisateurs[5].

La détection d'objets peut utiliser la classification d'images pour identifier une certaine classe d'images, puis détecter et classer leur apparition dans une image ou une vidéo. Les exemples incluent la détection de dommages sur une chaîne de montage ou l'identification de machines nécessitant un entretien[5].

Le suivi d'objet suit un objet une fois qu'il a été détecté. Cette tâche est souvent exécutée à partir d'images capturées en séquence ou de flux vidéo en temps réel. Les véhicules autonomes, par exemple, doivent non seulement classer et détecter les objets tels que les piétons, les autres voitures et les infrastructures routières, mais aussi les suivre en mouvement pour éviter les collisions et respecter le code de la route[5].

La recherche d'images basée sur le contenu utilise la vision par ordinateur pour parcourir, rechercher et récupérer des images à partir de grandes réserves de données, en se basant sur le contenu des images plutôt que sur les balises de métadonnées qui leur sont associées. Cette tâche peut intégrer l'annotation automatique des images qui remplace l'étiquetage manuel des images. Ces tâches peuvent être utilisées pour les systèmes de gestion des ressources numériques et peuvent augmenter la précision de la recherche et de l'extraction[5].

1.3 Réseau de Neurones Convolutionnel (CNN):

Un réseau de neurones convolutifs (CNN) est une catégorie de modèle d'apprentissage automatique . Plus précisément, il s'agit d'un algorithme d'apprentissage profond particulièrement adapté à l'analyse de données visuelles. Les CNN sont couramment utilisés pour traiter des images et des vidéos. De plus, leur efficacité pour identifier des objets les rend fréquemment utilisés pour des tâches de vision par ordinateur , telles que la classification d'images, la détection d'objets, la segmentation, et bien plus encore[6] .

Architecture des CNN : Un CNN est généralement composé de plusieurs couches(Figure 6) , que l'on peut classer en trois groupes : les couches convolutives, les couches de pooling et les couches entièrement connectées. À mesure que les données traversent ces couches, la complexité du CNN augmente, ce qui lui permet d'identifier successivement de plus grandes portions d'une image, ainsi que des caractéristiques plus abstraites (Figure 7).

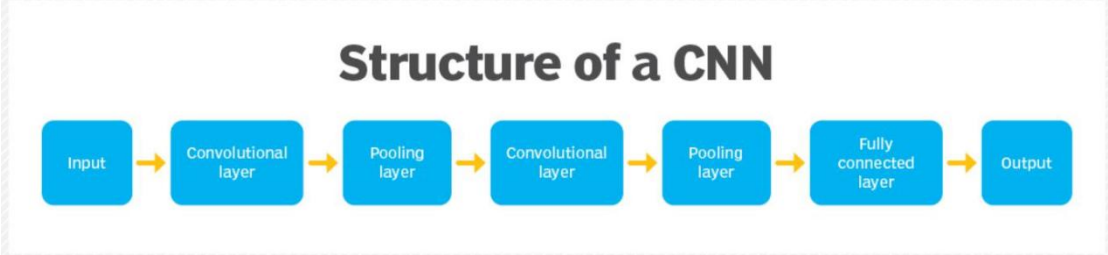

Figure 6 : L'architecture typique d'un CNN[6]

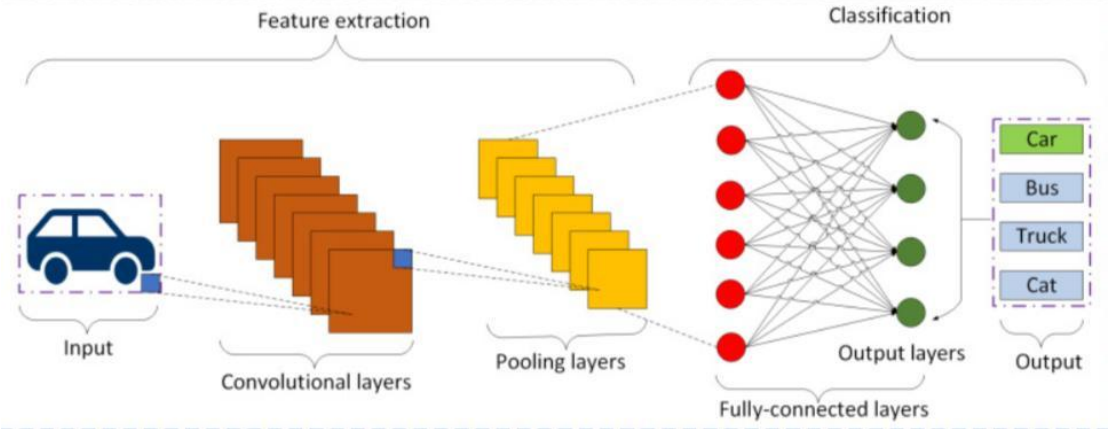

Figure 7 : Illustration de l'architecture de base d'une technique de classification d'images basée sur le CNN[7].

• **Couches convolutives :** Cœur du réseau, elle applique un noyau (matrice de pondérations) pour balayer l'image et détecter des caractéristiques via des produits scalaires, générant des cartes de features. Les premières couches identifient des éléments simples (contours, textures), tandis que les couches profondes analysent des motifs complexes. Plusieurs couches empilées permettent une interprétation hiérarchique des données visuelles.[6]

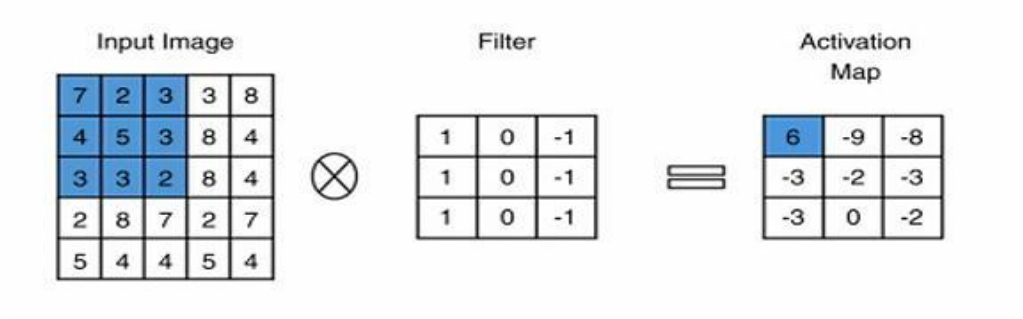

Figure 8 : Illustration d'une procédure de convolution[8]

• **Couches de mise en commun (Pooling) :**Elle réduit la dimensionnalité des données en conservant l'information essentielle via un sous-échantillonnage spatial. Les méthodes principales - max pooling (valeur maximale) et average pooling (moyenne)- diminuent les calculs et améliorent la généralisation du modèle. Bien que causant une perte d'information partielle, cette opération reste cruciale pour prévenir le surapprentissage et optimiser les performances. [6]

Le pooling (max/average) compresse les features après convolution, réduisant la complexité tout en préservant les informations clés pour éviter l'overfitting.

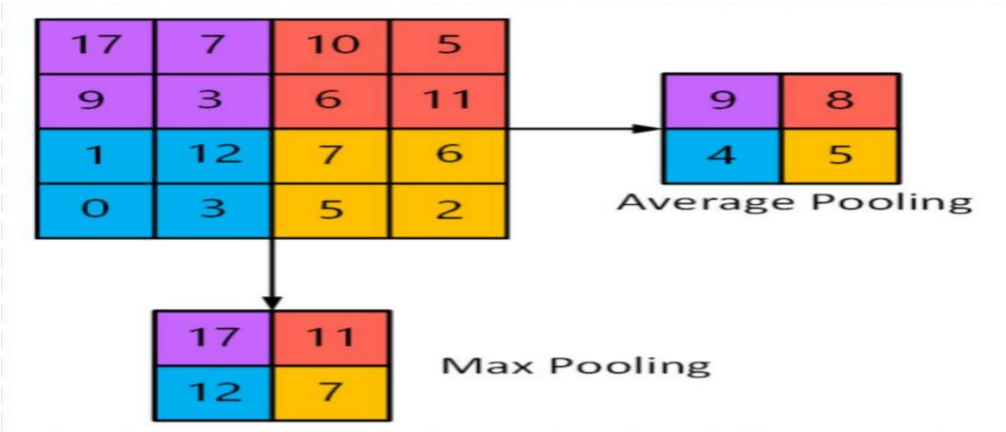

Figure 9 : Exemple de l'opération de regroupement maximal et de l'opération regroupement moyen sur la même caractéristique d'entrée.[7]

• **Couches entièrement connectées :** Positionnée en fin de réseau, elle combine toutes les features extraites pour effectuer la classification finale. Chaque neurone est connecté à tous les neurones suivants, permettant une prise de décision globale. Bien que puissante, son usage est limité en raison du grand nombre de paramètres qu'elle génère, risquant le surapprentissage et une forte consommation de ressources. [6]

1.4-Modèles de détection d'objets :

Les algorithmes de détection d'objets sont largement classés en deux catégories : les détecteurs à étape unique et les détecteurs à deux étapes, selon le nombre de fois que la même image d'entrée est transmise à travers un réseau.

Les détecteurs à deux étapes séparent la tâche en deux étapes : La première étape implique la génération d'un ensemble de propositions d'objets candidats. Ces propositions sont des régions dans l'image qui sont susceptibles de contenir des objets. Ceci est généralement fait en utilisant un Réseau de Proposition de Régions (RPN). La seconde étape implique la classification des objets dans les régions proposées et l'affinement de leurs boîtes englobantes. Les exemples incluent : Faster RCNN [9]

YOLO est un algorithme populaire de détection d'objets, introduit en 2015 par Redmon et al. [10]. Il a depuis été publié en diverses versions, notamment YOLO V2, V3, V4, et V9. Les caractéristiques principales de YOLO incluent sa petite taille et sa vitesse de calcul rapide, sa capacité à produire les positions des boîtes englobantes et les catégories, ainsi que sa forte capacité de généralisation [10].

L'architecture de YOLO, comme illustrée dans la Figure 10, se compose de multiples couches convolutionnelles et de max pooling qui contribuent à sa vitesse de calcul rapide.

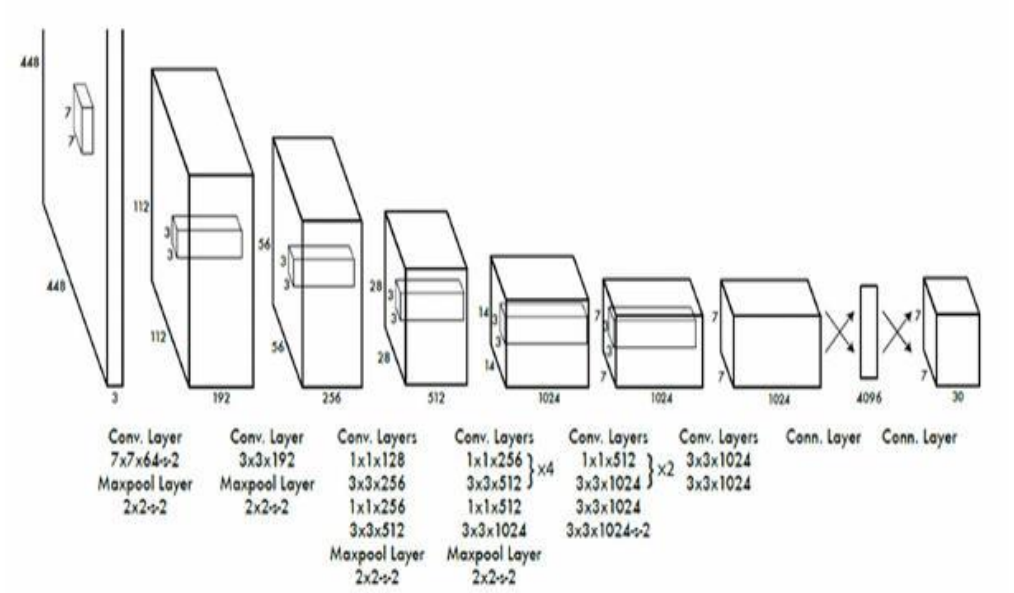

Figure 10 : L'architecture YOLO[10]

L'algorithme **YOLO** implique plusieurs étapes (figure 11). Premièrement, le modèle prend une image en entrée, puis la divise en cellules avec une grille S×S, ensuite chaque cellule prédit un nombre prédéfini de boîtes englobantes, la confiance pour ces boîtes et C probabilités de classe. Ces prédictions sont encodées sous forme d'un tenseur S × S × (B × 5 + C) qui est responsable de la détection d'objets. Chaque boîte englobante est représentée par un ensemble de coordonnées X, Y pour la largeur et la hauteur et un score de seuil de confiance. Ce score indique à quel point la boîte est précise dans la détection d'objets. Pour la prédiction de classe, le modèle prédit la probabilité que chaque boîte englobante contienne différents types d'objets.

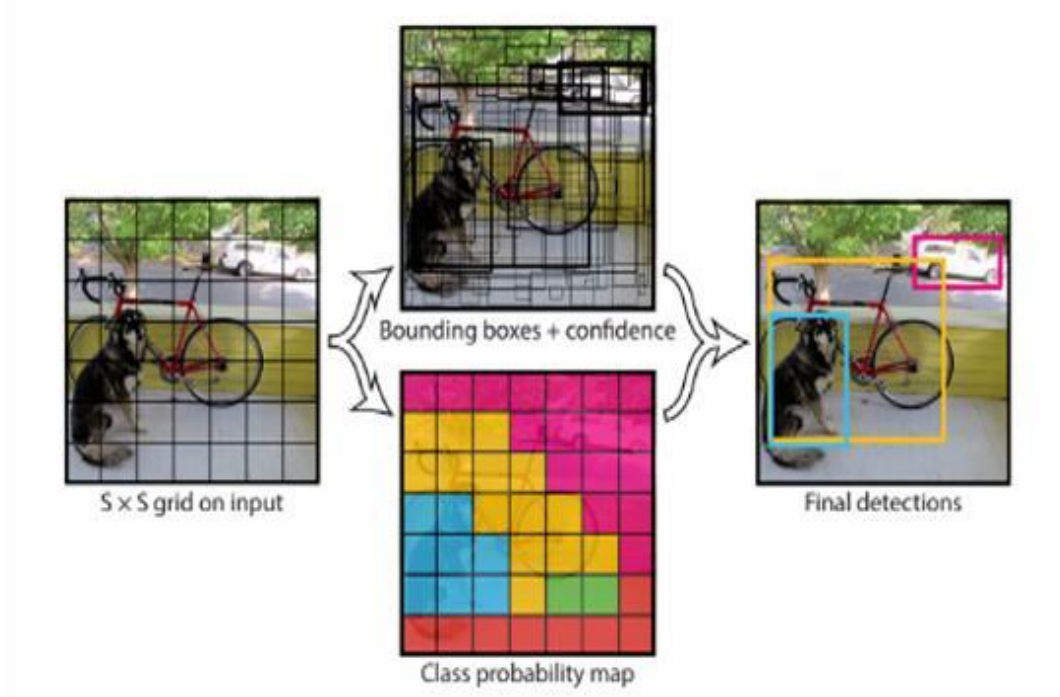

Figure 11 : Exemple de processus de travail de YOLO

L'architecture de YOLOv8 peut être divisée en trois composantes principales : le dos le cou et la tête, comme le montre la figure suivante (Figure 12).

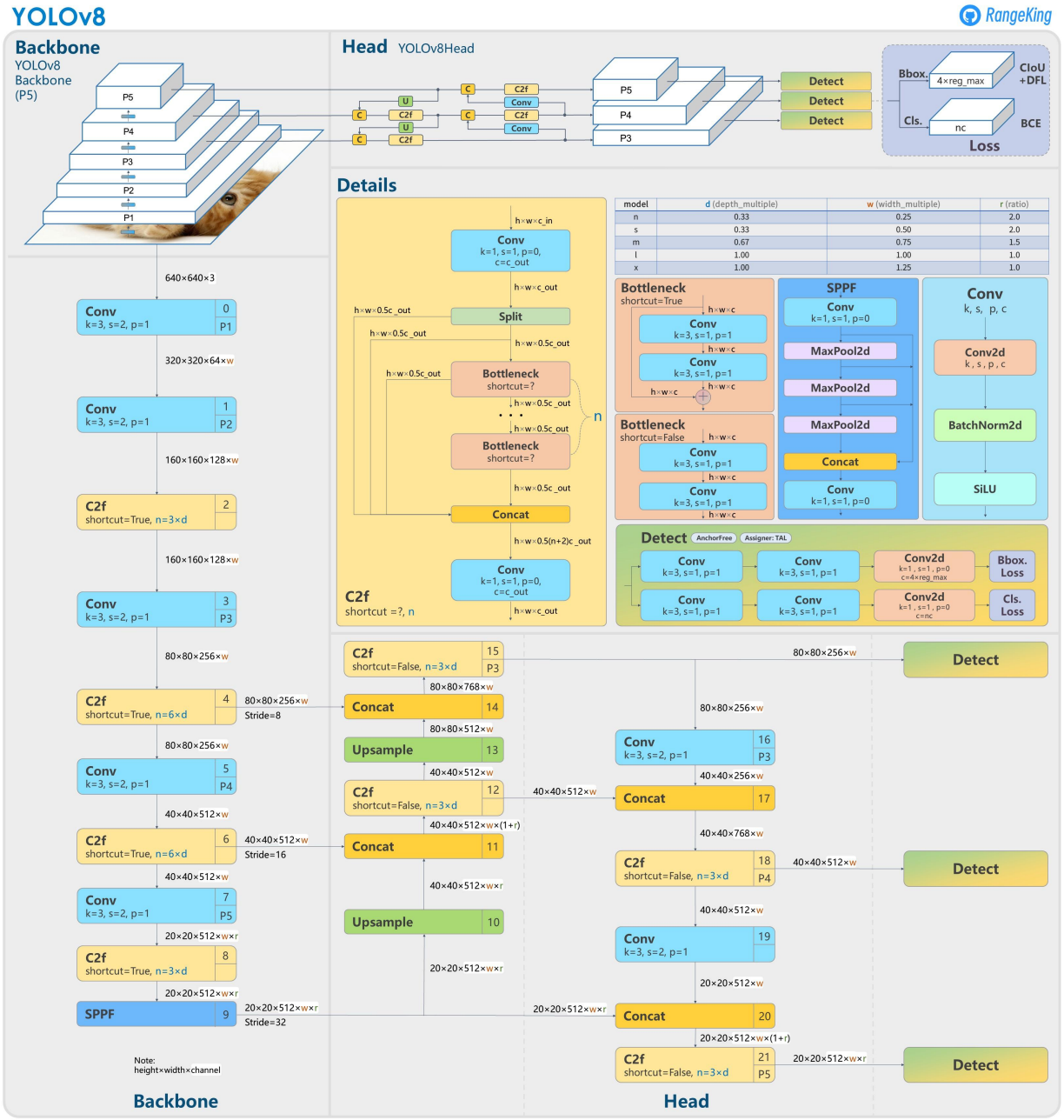

Figure 12 : Architecture du modèle Yolov8[12]

L'épine dorsale (The Backbone) Il s'agit du réseau neuronal convolutif (CNN) chargé d'extraire les caractéristiques de l'image d'entrée. YOLOv8 utilise une épine dorsale CSPDarknet53 personnalisée, qui emploie des connexions partielles entre les étages pour améliorer le flux d'informations entre les couches et accroître la précision [11].

L**e cou(The neck)**,également connu sous le nom d'extracteur de caractéristiques, fusionne les cartes de caractéristiques provenant de différentes étapes de l'épine dorsale afin de capturer des informations à différentes échelles. L'architecture YOLOv8 utilise un nouveau module C2f au lieu du traditionnel Feature Pyramid Network (FPN). Ce module combine des caractéristiques sémantiques de haut niveau avec des informations spatiales de bas niveau, ce qui permet d'améliorer la précision de la détection, en particulier pour les petits objets[11].

La tête(The head) La tête est chargée de faire des prédictions. YOLOv8 utilise plusieurs modules de détection qui prédisent les boîtes de délimitation, les scores d'objectivité et les probabilités de classe pour chaque cellule de la grille de la carte des caractéristiques. Ces prédictions sont ensuite agrégées pour obtenir les détections finales [11].

1.5 Travaux Antérieurs et Positionnement de l'Approche

1.5.1 Revue des Travaux Existants :

A \ DeepRoad" (Zhang et al., 2021):

C’est une Solution basée sur un CNN personnalisé (ResNet-50) spécialisée dans la détection de fissures routières, atteignant 88% de précision mais avec des temps de traitement lents (~2.3s/image). Limité à une seule classe de dommages.

Critère	**Détails**
Année/reference	Zhang et al., 2021 (Conférence IEEE sur les infrastructures intelligentes)
Architecture	CNN personnalisé basé sur ResNet-50
Dataset	12 000 images de routes chinoises (mix urbaines/rurales)
Classes cibles	Fissures uniquement (pas de distinction D10/D20/D40)
Précision(mAP)	88% sur les fissures
Vitesse d’inférence	2.3 s/image (NVIDIA V100 GPU)
Avantages	- Très haute précision sur fissures simples
Limites	- Lent (inutilisable en temps réel) - Ne détecte pas les nids-de-poule

Table 1 : Fiche technique du DeepRoad(Zhang et al.,2021)

B \YOLOv5 pour nids-de-poule" (Lee, 2022) :

Solution de détection en temps réel des nids-de-poule utilisant YOLOv5, atteignant 82% de précision (mAP@0.5) sur un dataset coréen. Performante en vitesse (~0.05s/image) mais limitée aux nids-de-poule uniquement.

Critère	**Détails**
Architecture	YOLOv5s (version small)
Dataset	8k images (routes coréennes)
Classes cibles	Nids-de-poule uniquement
Précision(mAP)	82%
Vitesse d'inférence	0.05s/image (GPU RTX 3080)
Hardware requis	GPU haut de gamme
Adaptation DTP	Non

Table 2 : Fiche technique du YOLOv5 pour nids-de-poule" (Lee, 2022)

C \Système hybride DTP Maroc" (Benali, 2023) :

Solution combinant drones et IA pour l'inspection routière au Maroc, avec une précision de 84% mais un coût de déploiement élevé (nécessite drones + cloud computing). Complexité technique limitant son adoption large.

Critère	**Détails**
Technologie	Drones + CNN cloud-based
Coût Déploiement	50 000€ (drones + serveurs)
Précision (mAP@0.5)	84%
Classes Détectées	Fissures + nids-de-poule
Vitesse	1.2s/image (latence cloud)
Maintenance	Complexe (techniciens spécialisés)
Couverture	Limitée aux zones drone-accessibles

Table 3 : Fiche technique du Systéme hybride DTP (Benali 2023)

D\ RoadInspector AI" (Singh et al., 2023) :

Solution commerciale SaaS de détection de dommages routiers utilisant un ensemble de modèles (CNN + segmentation), avec une précision déclarée de 91% mais nécessitant un abonnement cloud coûteux (500€/mois). Performante mais non adaptable aux spécificités locales.

Critère	**Détails**
Titre	RoadInspector AI
Technologie	YOLOv7
Classes	4 (nids-de-poule, fissures longitudinales/transversales, chaussée intacte)
Précision	82% mAP@0.5
Limites	• Performances réduites en zones rurales (68% mAP) • Pas d'export automatisé des rapports • Coût élevé d'hébergement cloud

Table 4 : Fiche technique du roadInspector AI (Singh et al ,2023)

1.5.2 Gap scientifique et Contribution de notre Approche :

-Analyse des limites :

- L'observation des résultats obtenus par les solutions existantes indique que sont soit précises mais coûteuses (comme RoadInspector AI), soit rapides mais spécialisées dans un seul type de dommage (comme YOLOv5 Lee). Aucune d'entre elles ne réalise les deux simultanément :

- Multi-classes (D10/D20/D40)
- Précision > 85 %
- Prix < 5 000 €
-Fonctionnalités CPU

Problème	Solutions Existantes	Notre Contribution
Détection partielle	≤ 3 classes (Lee, Zhang)	4 classes (+ dégradations)
Biais urbain	RoadInspector AI (68% en rural)	RDD-2022 (couverture nationale)
Pas de signalement automatisé	Aucun module DTP	Rapports PDF + emails
Infrastructure lourde	Serveurs GPU/drones (Benali)	Optimisation CPU/GPU grand public

Table 5 : Tableau Synthétique du Gap

-Ces innovations positionnent notre solution comme une alternative optimale pour la DTP.

1.6 Conclusion :

Ce premier chapitre a permis d'établir les fondements théoriques essentiels à notre étude, en analysant de manière critique les solutions existantes en matière de détection automatisée des dommages routiers. L'analyse comparative a révélé des limitations majeures dans les approches actuelles, notamment en termes de coût, de couverture des classes de dommages et d'adaptation aux spécificités locales. Notre revue approfondie a mis en évidence le gap scientifique que notre approche vient combler, en proposant une solution basée sur YOLOv8 qui allie précision, efficacité économique et adaptabilité aux besoins de la DTP. Ces éléments théoriques et analytiques constituent désormais la base solide sur laquelle nous allons nous appuyer pour concevoir et implémenter notre système.

Chapitre 2:

Conception du système

2.1 Architecture Générale du Système:

2.1.1. Schéma Complet du Flux de Traitement:

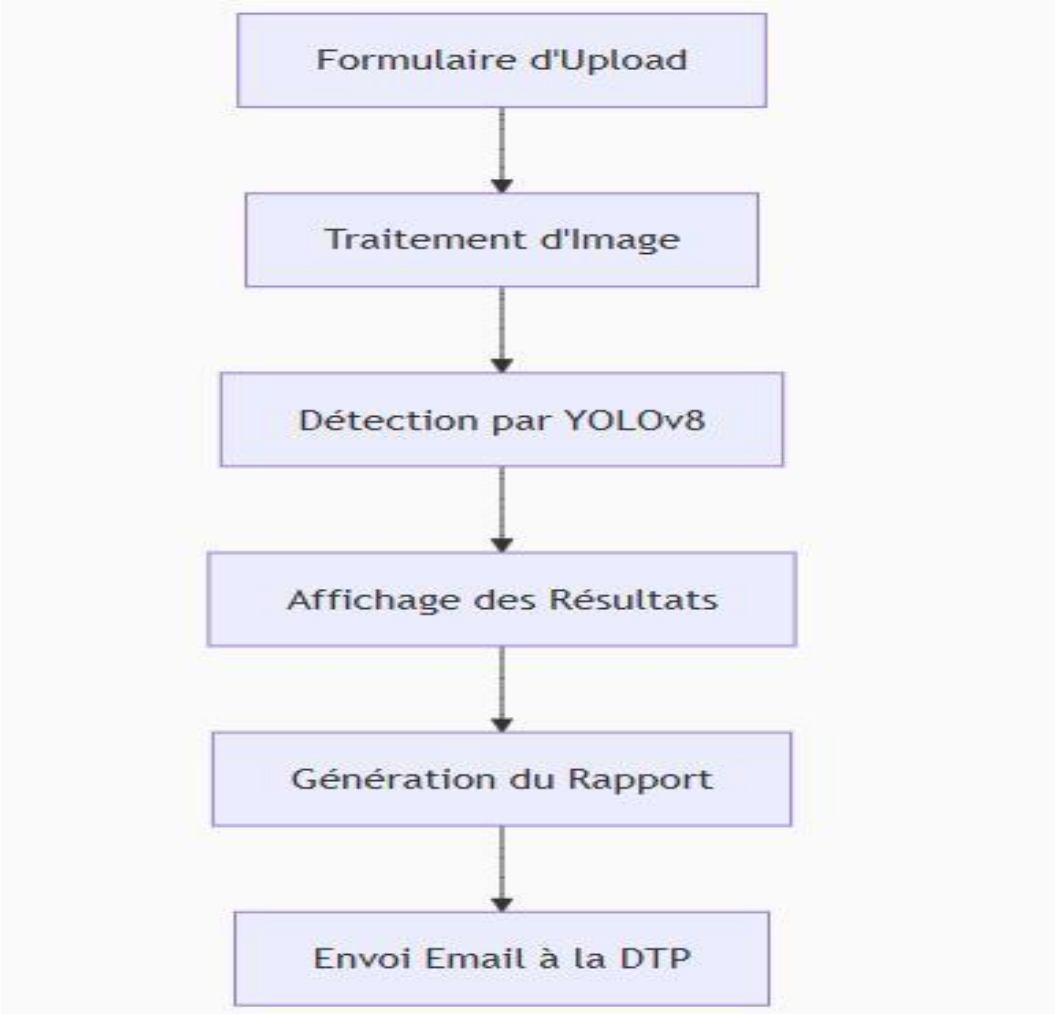

Figure 13 : Schéma de Flux du traitement

Notre système suit une architecture modulaire organisée en 5 étapes clés comme illustré dans Figure 13:

1-Formulaire d'Upload : Interface utilisateur conviviale (HTML/Python) permettant :
-L'upload d'une image depuis un appareil local
-La saisie d'informations complémentaires :Localisation ,
Gravité(Faible/Moyen/Élevé)

2-Traitement d'Image :

-Redimensionnement (640x640 px)

-Normalisation des pixels

-Vérification de la qualité

3-Détection par YOLOv8 :

-Chargement du modèle pré-entraîné

-Exécution de l'inférence pour :

-Localiser les dommages (bounding boxes)

-Classer le type (D10/D20/D40/D43)

-Calculer le score de confiance

4-Affichage des Résultats :Sortie visuelle dans l'interface :

-Image annotée avec les dommages détectés.

-description récapitulatif (Type, confiance,)

5-Génération & Envoi du Rapport :Formatage automatique en CSV contenant :

-Métadonnées (localisation,)

-Résultats techniques (classes, confiance, gravité)

-Envoi par email au service concerné de la DTP

2.1.2. Schéma Traitement d'image par YOLO:

Ce schéma -Figure 14- illustre le processus de détection d'images par YOLOv8 : l'image passe par le *Backbone* (CSPDarknet) pour l'extraction des caractéristiques, puis par le *Neck* (PANet) pour la fusion des features, et enfin par le *Head* pour localiser les défauts (boîtes englobantes + classes).

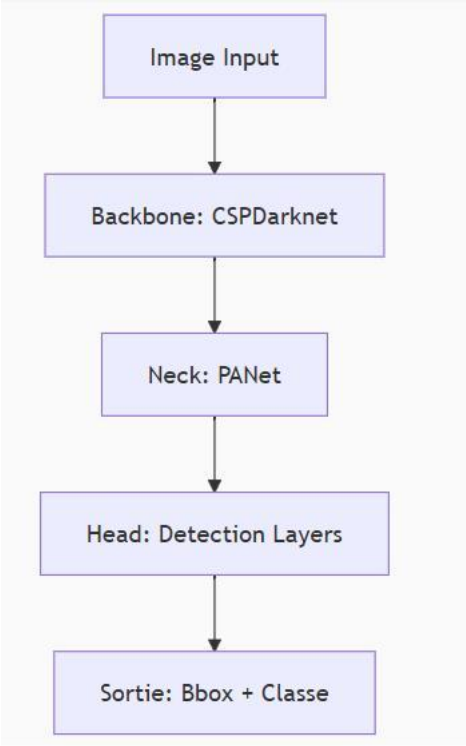

Figure 14 : Schéma traitement d'image par YOLO

-Avantages et Caractéristiques Majeures :

La conception proposée du système satisfait aux caractéristiques suivantes :

- **Temps réel** : YOLOv8 permet une détection rapide (< 1s/image sur CPU).
- **Modularité** : Chaque bloc peut être optimisé séparément (ex: remplacer YOLOv8 par une version ultérieure).
- **Extensibilité** : Peut intégrer des capteurs IoT (ex: caméras de drones) , ou ajouter d'autre classes de dommages.

2.2Présentation UML :

2.2.1 Diagramme Cas d'utilisation :

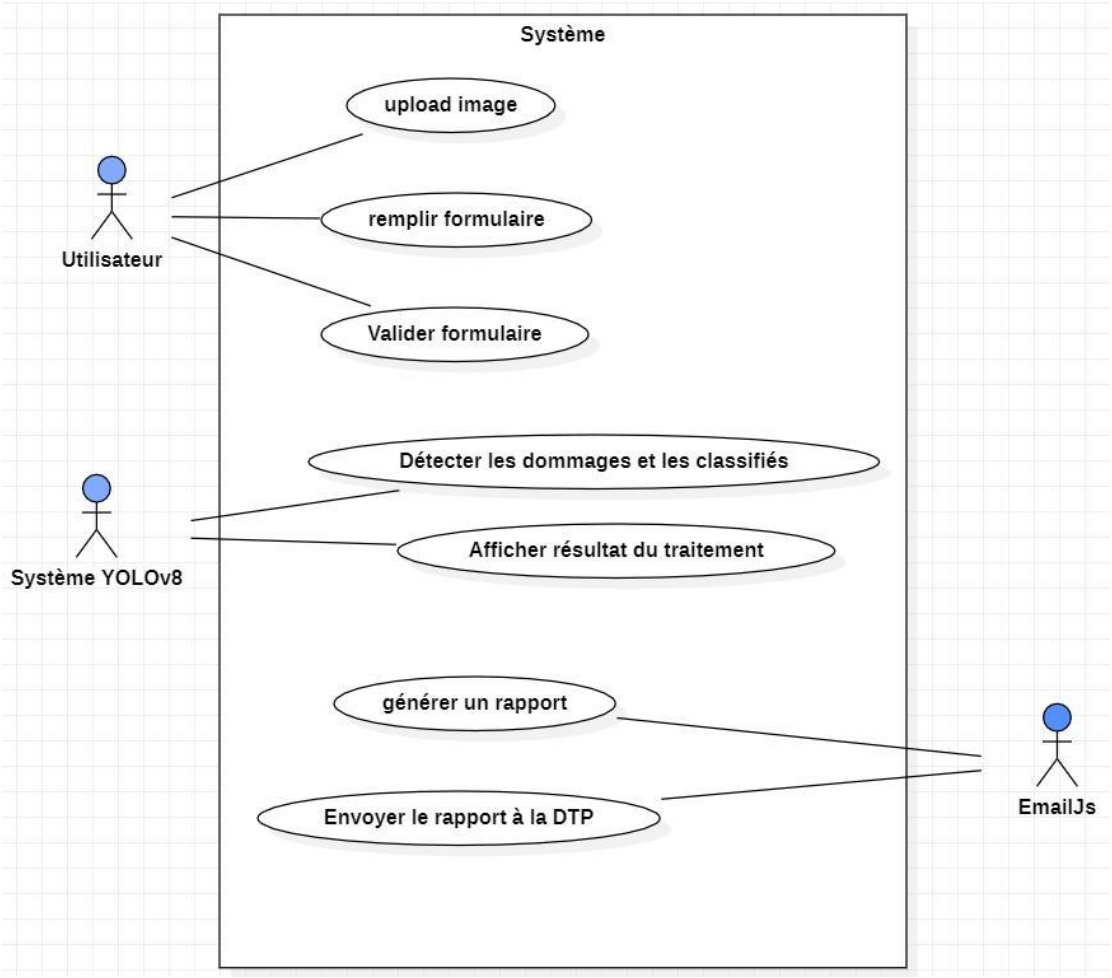

Figure 15 : Diagramme Cas d'utilisation

Ce diagramme présente les fonctionnalités clés du système : depuis l'upload d'images jusqu'à la génération et l'envoi automatisé de rapports à la DTP, en passant par la détection et classification des dommages.

2.2.2 Diagramme de séquence :

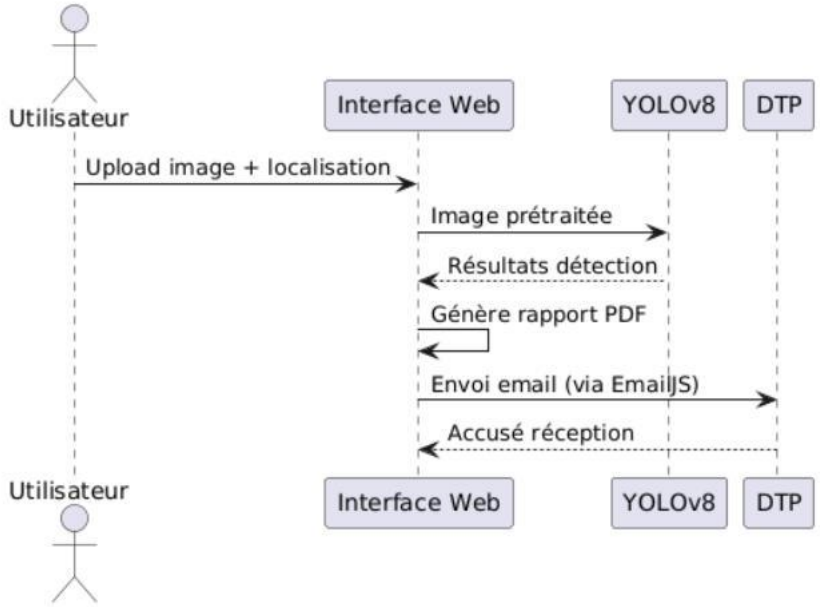

Figure 16 : Diagramme de séquence

Ce diagramme montre l'enchaînement des actions : de l'upload par l'utilisateur jusqu'à l'envoi automatisé du rapport à la DTP, en passant par le traitement par YOLOv8 et la génération du PDF

2.2.3 Diagramme d'activité :

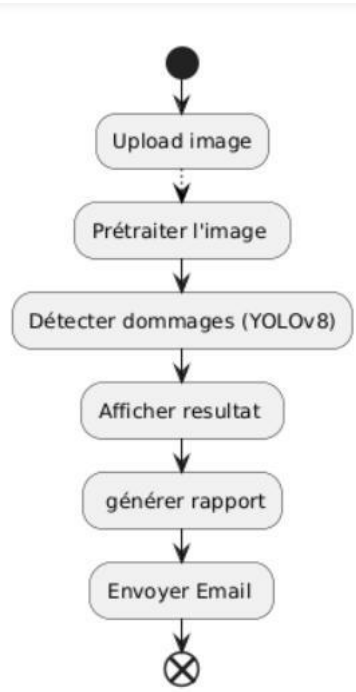

Figure 17 : Diagramme d'activité

Ce flux -Figure 17- illustre le processus complet du système, du téléchargement de l'image à l'envoi du rapport final, en passant par l'analyse IA et la génération des résultats.

2.3 Analyse des Exigences :

2.3.1 Exigences Fonctionnelles :

-Détection précise des dommages (fissures, nids-de-poule, etc.).

-Interface intuitive et conviviale pour le signalement et le test (upload, formulaire).

-Génération de rapports (localisation, gravité, resultat du detection).

2.3.2 Exigences Non-Fonctionnelles :

-Performance : Temps de traitement < 1s/image (justification par tests YOLOv8).

-Précision : Métriques (, précision/rappel) sur RDD-2022.

-Modifiabilité :l'évolutivité du système (ex: ajout de nouvelles classes de dommages).

2.4 Conclusion:

Ce chapitre a renforcé techniquement notre système intelligent de signalement des dommages routiers . Grâce à une conception modulaire basée sur YOLOv8 et une interface simplifiée , nous avons élaboré une solution adaptée aux limitations opérationnelles du DTP . Un examen rigoureux des exigences fonctionnelles (détection précise , génération de rapports) et non fonctionnelles (performances, maintenabilité) a permis de trouver des compromis technologiques judicieux entre efficacité et pragmatisme . Les diagrammes UML créés ici formalisent ces concepts et constitueront la base de l' implémentation détaillée du chapitre suivant , où nous testerons directement les performances du système .

Chapitre 3:

Implémentation, Expérimentation et Résultats

Introduction

Ce chapitre est dédié à la présentation détaillée de l'implémentation pratique de mon système intelligent de détection et de signalement de dommages routiers. S'appuyant sur les fondations conceptuelles et les choix technologiques établis dans le Chapitre 2, je vais ici concrétiser mon approche en décrivant en profondeur chaque composant de ma solution. Ma démarche débutera par une explication exhaustive des étapes fondamentales de la préparation des données, en soulignant l'importance critique de la qualité et de la diversité du jeu de données pour la performance de l'intelligence artificielle. J'explorerai ensuite le processus rigoureux d'entraînement de mon modèle d'apprentissage profond, en détaillant les stratégies d'apprentissage par transfert et de réglage fin (fine-tuning) appliquées, avec un accent particulier sur les mécanismes internes, les bibliothèques sous-jacentes, et les aspects techniques de l'optimisation du modèle. La discussion se poursuivra avec la présentation architecturale et fonctionnelle de la logique métier côté serveur (backend) et de l'interface utilisateur web (frontend), en insistant sur leur rôle de passerelle essentielle pour l'interaction avec le modèle IA. Enfin, une section exhaustive sera consacrée à la méthodologie d'expérimentation de mon système sur des données réelles, suivie d'une analyse approfondie et critique des résultats obtenus, des performances du modèle, et des défis rencontrés, afin d'évaluer l'efficacité et les limites de mon approche globale et d'ouvrir la voie aux perspectives d'amélioration.

3.1 Implémentation du Modèle IA

La robustesse et l'efficacité de tout système basé sur l'apprentissage profond (Deep Learning) résident intrinsèquement dans la qualité, la pertinence et le volume des données d'entraînement, ainsi que dans l'optimisation du processus par lequel le modèle apprend de ces données. Cette section explore en détail l'intégration du modèle d'intelligence artificielle au cœur de ma solution, en couvrant les étapes fondamentales de la préparation du jeu de données et la méthodologie appliquée pour l'entraînement et le réglage fin du modèle.

3.1.1 Préparation des Données : Fondation de la Détection Robuste

La phase de préparation des données est reconnue comme la première étape, et l'une des plus critiques, dans le cycle de vie d'un projet d'apprentissage automatique. Une sélection rigoureuse, une structuration adéquate et un pré-traitement pertinent des données sont fondamentaux pour garantir la robustesse, la précision, la capacité de généralisation (aptitude à performer sur des données non vues) et la fiabilité d'un modèle d'intelligence artificielle. Les performances d'un modèle sont directement corrélées à la qualité des données sur lesquelles il a été formé. Des données de mauvaise qualité ou mal préparées peuvent entraîner un sous-apprentissage

(underfitting), un surapprentissage (overfitting), ou des biais inattendus, compromettant la fiabilité du système final.

Pour mon projet de détection de dommages routiers, j'ai choisi d'utiliser le "Road Damage Detection Dataset (RDD)". Ce dataset est une référence académique et pratique largement reconnue et adoptée dans le domaine de l'inspection routière assistée par l'intelligence artificielle. Il a été spécifiquement conçu pour l'analyse des dégradations de la chaussée, le rendant particulièrement pertinent pour mon objectif. La pertinence du RDD réside dans sa nature spécialisée, son volume significatif, et ses annotations minutieuses des divers types de défauts de chaussée, collectées à travers différentes régions géographiques et conditions environnementales. L'accès à ce dataset a été facilité par la plateforme open-source Kaggle, une communauté et un référentiel de science des données de premier plan. J'ai accédé au dataset spécifiquement via l'adresse: [49]. Kaggle, en tant que plateforme collaborative, offre non seulement un accès aisé à des datasets structurés et souvent pré-annotés, mais elle favorise également l'échange de connaissances et la reproductibilité des recherches, ce qui a grandement accéléré les phases initiales de recherche et de développement de ma solution.

Le dataset RDD, dans sa globalité, est une collection vaste et hétérogène, comprenant plus de 76 800 fichiers. Cette collection est principalement composée d'images numériques de routes, accompagnées de leurs fichiers d'annotations correspondants. Cette quantité considérable de données est essentielle pour entraîner des modèles d'apprentissage profond, car ils nécessitent une exposition à un grand nombre d'exemples pour apprendre des représentations complexes. Pour organiser de manière efficace mes phases d'entraînement itératives et pour assurer un suivi clair des résultats obtenus à chaque étape du développement du modèle, j'ai structuré l'accès au dataset et aux sorties du modèle via des dossiers distincts. Cette organisation a été implémentée sur Google Drive, en créant des répertoires séparés tels que train_session_1, train_session_12, et train_session_13. La Figure 18 illustre de manière concrète cette approche. La structure en répertoires séparés a offert une clarté indispensable dans le suivi du processus de fine-tuning de mon modèle, permettant de gérer les sauvegardes intermédiaires (checkpoints), les journaux de performance (logs), et les configurations spécifiques de chaque session d'entraînement de manière isolée et ordonnée. Chaque dossier représentait ainsi une session d'entraînement distincte, facilitant la reprise du travail, l'analyse comparative des performances et la gestion des versions du modèle.

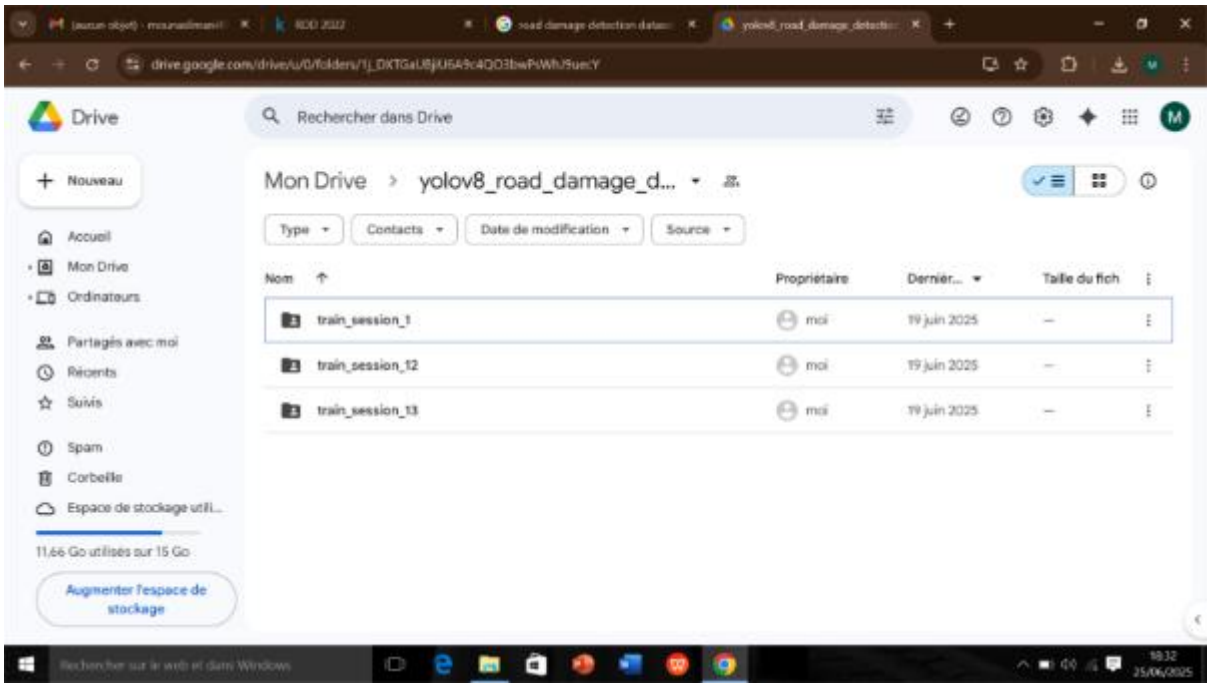

Figure 18 : Organisation des dossiers des sessions d'entraînement sur Google Drive.

Cette structure hiérarchique améliore la gestion des versions du modèle, des configurations d'entraînement et des résultats intermédiaires, facilitant le suivi du processus de fine-tuning.

Le dataset RDD est intrinsèquement pré-divisé en trois ensembles distincts, une pratique méthodologique standard et cruciale dans le domaine de l'apprentissage automatique pour une évaluation rigoureuse, non biaisée et reproductible de la performance d'un modèle. Cette division garantit que le modèle est testé sur des données qu'il n'a jamais "vues" auparavant, reflétant sa capacité de généralisation sur des scénarios réels. Les trois ensembles sont :

- **Ensemble d'entraînement (Training Set):** Cet ensemble constitue la majeure partie du dataset (généralement 70% à 80% des données). Il est exclusivement utilisé pour alimenter le modèle d'apprentissage profond. C'est à partir de ces images que le modèle ajuste ses poids internes, apprenant à identifier les caractéristiques et les patterns complexes associés aux différents types de dommages routiers. L'objectif est de minimiser la fonction de perte (loss function) sur cet ensemble.
- **Ensemble de validation (Validation Set):** Plus petit que l'ensemble d'entraînement (souvent 10% à 15% des données), cet ensemble est utilisé pendant le processus d'entraînement pour ajuster les hyperparamètres du modèle

et pour évaluer ses performances à intervalles réguliers (par exemple, à la fin de chaque époque). Il est crucial pour détecter le surapprentissage (overfitting), un phénomène où le modèle apprend trop bien les données d'entraînement et perd sa capacité à généraliser sur de nouvelles données. L'ensemble de validation permet d'orienter les décisions concernant la fin de l'entraînement (arrêt précoce) ou l'ajustement des paramètres pour améliorer la performance.

- **Ensemble de test (Test Set):** Cet ensemble (généralement 10% à 15% des données) est conservé strictement à part et n'est jamais vu par le modèle pendant les phases d'entraînement ou de validation. Il est utilisé uniquement pour l'évaluation finale et impartiale des performances du modèle. Les métriques obtenues sur l'ensemble de test offrent une mesure objective et non biaisée de sa capacité de généralisation sur des données nouvelles et inconnues, simulant ainsi les conditions d'utilisation réelles.

Chaque image de route dans le dataset est accompagnée de ses annotations correspondantes. Ces annotations sont d'une importance capitale car elles fournissent au modèle les "vérités terrain" (ground truth) qu'il doit apprendre à reconnaître. Dans le contexte de la détection d'objets, ces annotations sont fournies sous forme de fichiers texte distincts (au format YOLO - .txt), où chaque fichier .txt portant le même nom (mais avec une extension différente) contient une ligne pour chaque dommage présent dans l'image associée. Chaque ligne inclut plusieurs informations cruciales: l'identifiant numérique de la catégorie de dommage (par exemple, 0 pour D00, 1 pour D10, etc.), et les coordonnées normalisées de la "boîte englobante" (Bounding Box). La boîte englobante est un rectangle qui délimite précisément l'emplacement spatial de chaque défaut sur l'image (coordonnées du centre, largeur et hauteur du rectangle, toutes normalisées par rapport aux dimensions de l'image). Cette approche d'annotation détaillée permet au modèle d'apprentissage non seulement de reconnaître la présence d'un dommage (tâche de classification), mais aussi de le localiser spatialement avec précision dans l'image (tâche de détection) et de l'assigner à une catégorie spécifique. La Figure 19 illustre un exemple concret d'une image issue du dataset RDD avec ses annotations, mettant en évidence la précision et la rigueur de l'étiquetage nécessaires pour un apprentissage efficace.

Figure 19 : Exemples d'images du Road Damage Detection Dataset avec annotations (catégorie et boîte englobante).

Ces annotations sont cruciales pour l'apprentissage du modèle car elles définissent les "vérités terrain" que le modèle doit apprendre à prédire.

Le dataset RDD couvre diverses typologies de dégradations routières, permettant à mon modèle de se former sur une large gamme de défauts et de développer une capacité de reconnaissance polyvalente. Mon modèle a été entraîné spécifiquement pour identifier et classer cinq (5) catégories principales de dommages, conformément à la taxonomie définie dans le dataset. Ces classifications sont d'une importance capitale pour orienter les efforts de maintenance et assurer la sécurité des infrastructures, car chaque type de dommage peut nécessiter une intervention, une priorité, et des ressources différentes. La Table 6 récapitule ces catégories et leurs descriptions détaillées, telles que définies dans le dataset, ainsi que leurs implications pratiques pour la maintenance routière.

Code	Type de Dommage	Cause Principale
D00	Fissure longitudinale	Tassements différentiels, fatigue du trafic
D10	Fissure transversale	Dilatation thermique, fissures sous-jacentes
D20	Fissure en maille (crocodile)	Fatigue avancée, fondation insuffisante
D43	Trou	Infiltration d'eau + gel/dégel
D40	Autre	Divers (ressuage, désenrobage, etc.)

Table 6: Catégories de dommages routiers du dataset RDD et leurs implications.

Un aspect fondamental de mon pipeline de traitement d'images réside dans la gestion automatique de la pré-traitement des données par la bibliothèque Ultralytics, sur laquelle s'appuie le modèle YOLOv8. Ultralytics est une bibliothèque Python open-source de premier plan qui fournit une implémentation haute performance du modèle YOLO (You Only Look Once), un algorithme de détection d'objets en temps réel largement utilisé et reconnu pour son efficacité. Contrairement aux approches plus anciennes qui nécessitaient une préparation manuelle extensive et souvent fastidieuse des images (comme la conversion des formats, le redimensionnement statique ou l'augmentation des données hors ligne), Ultralytics intègre directement et efficacement des étapes de pré-traitement essentielles et dynamiques au sein de son pipeline d'entraînement.

Ces étapes incluent principalement :

- **Nettoyage et structuration implicites des données:** Il est important de noter que le dataset RDD est déjà bien structuré et annoté. Pour cette raison, une étape manuelle explicite de "nettoyage" des images (par exemple, suppression du bruit, correction des couleurs) n'a pas été nécessaire, car la robustesse des modèles de vision par ordinateur modernes comme YOLOv8, combinée aux techniques de normalisation et d'augmentation des données d'Ultralytics, permet de gérer efficacement les variations intrinsèques du dataset. Le pré-traitement est donc principalement géré de manière automatique par la bibliothèque.

- **Redimensionnement (Resizing) automatique:** Toutes les images d'entrée sont automatiquement redimensionnées à une dimension fixe (généralement 640x640 pixels pour YOLOv8, bien que ce soit configurable) qui est adaptée à l'architecture du réseau neuronal. Cette uniformisation des dimensions est cruciale pour l'entrée dans le modèle.

- **Normalisation (Normalization) des valeurs de pixels:** Les valeurs des pixels des images sont mises à l'échelle (généralement entre 0 et 1) pour optimiser le processus d'apprentissage du réseau neuronal. Cette étape standard est cruciale pour la stabilité de l'entraînement et la convergence du modèle.

- **Augmentation des données (Data Augmentation) sophistiquée:** L'augmentation des données est une technique cruciale qui, en l'absence de ressources de calcul suffisantes pour entraîner un modèle sur des datasets colossaux (de plusieurs millions d'images), permet d'accroître artificiellement la diversité et le volume des données d'entraînement à partir du jeu de données existant. Pour YOLOv8, Ultralytics implémente des stratégies d'augmentation des données "on-the-fly". Cela signifie que les transformations sont appliquées dynamiquement à chaque image à la volée (à la demande) pendant l'entraînement, plutôt que de créer des copies physiques augmentées du dataset.

Les transformations appliquées par Ultralytics dans le cadre de l'augmentation des données sont diverses et ciblent la robustesse du modèle:

- **Transformations géométriques:** Incluent des rotations aléatoires (pour simuler des angles de vue différents), des cisaillements (pour des distorsions de perspective), des zooms (scaling, pour gérer des objets de différentes tailles), et des inversions (flipping horizontal/vertical, pour créer des variations symétriques). Ces transformations aident le modèle à reconnaître les dommages indépendamment de leur orientation ou de leur taille apparente.
- **Ajustements de couleur et d'intensité:** Comprennent des modifications aléatoires de la luminosité, du contraste, de la saturation et de la teinte. Ces ajustements simulent différentes conditions d'éclairage (jour, nuit, nuageux, ensoleillé) et aident le modèle à ne pas être sensible aux variations chromatiques, le rendant plus robuste aux conditions de capture réelles.

- **Techniques avancées:** Ultralytics intègre également des méthodes d'augmentation plus sophistiquées comme le **Mosaic Augmentation** (combinant aléatoirement des portions de 4 images différentes en une seule pour créer un batch d'entraînement plus riche et forcer le modèle à apprendre des objets dans divers contextes et échelles) et le **Copy-Paste Augmentation** (copiant des objets détectés d'une image pour les coller aléatoirement dans une autre image,

enrichissant le nombre d'occurrences d'objets rares et améliorant la détection d'objets partiellement occlus).

En créant de nouvelles variations d'images à chaque époque d'entraînement sans augmenter la taille physique du dataset, cette méthode rend le modèle considérablement plus robuste et capable de généraliser efficacement la détection des dommages dans une multitude de conditions réelles (variations d'éclairage, angles de vue différents, occlusions partielles, etc.). Cela minimise également la nécessité d'une intervention manuelle laborieuse sur chaque image source pour augmenter le dataset et réduit significativement le risque de surapprentissage, où le modèle mémorise le bruit des données d'entraînement au lieu d'apprendre les caractéristiques généralisables des dommages.

Figure 20 : Exemple d'images du dataset d'entraînement après application des techniques d'augmentation des données par Ultralytics, illustrant la diversité accrue des échantillons pour un entraînement robuste.

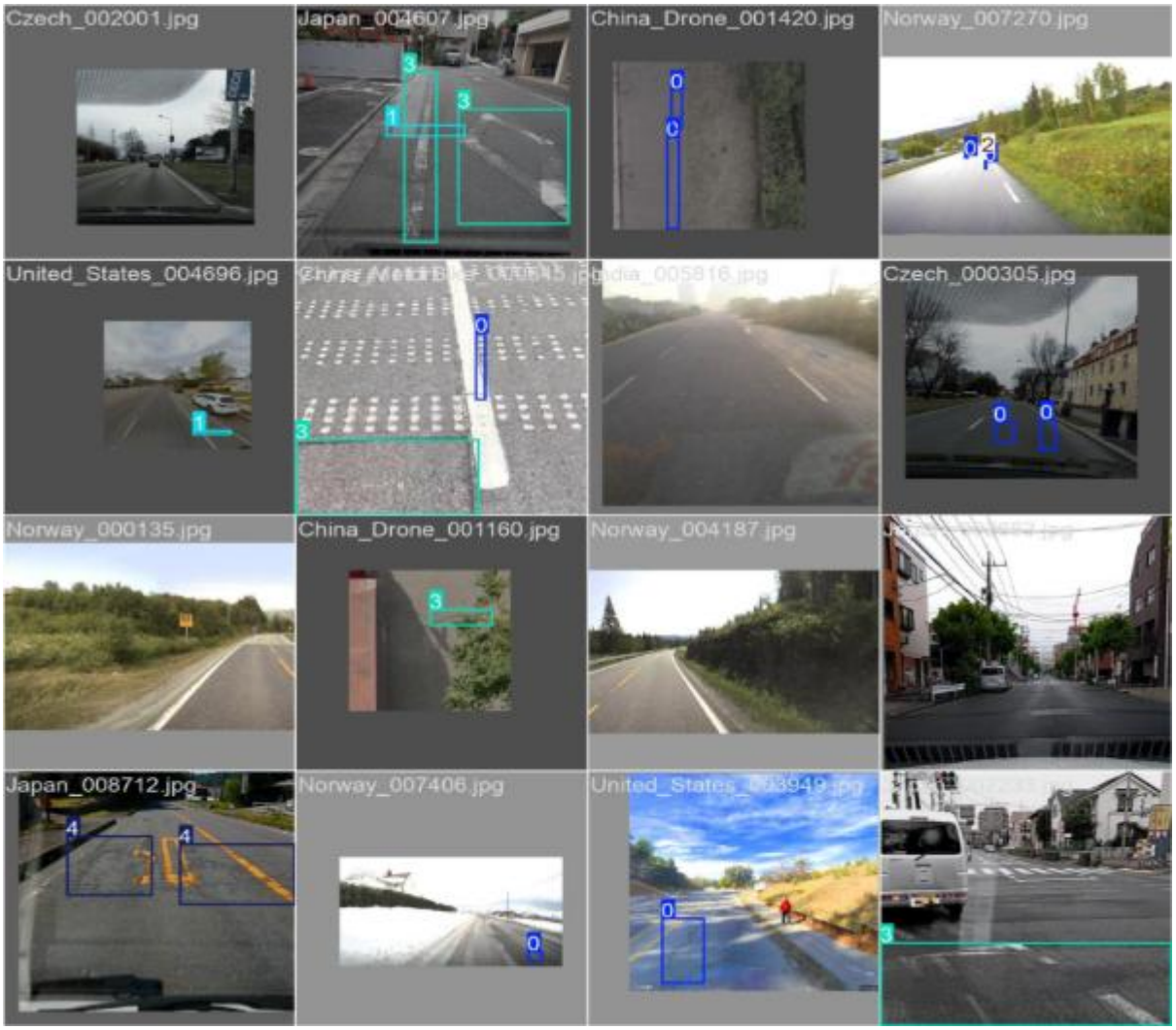

Figure 21 : Autre exemple d'images après l'augmentation des données via Ultralytics, illustrant l'efficacité du procédé Mosaic Augmentation pour créer des scénarios d'entraînement complexes et variés.

3.1.2 Processus d'Entraînement du Modèle : Optimisation Profonde de la Performance

L'efficacité finale de mon système de détection de dommages routiers repose sur un entraînement rigoureux et méthodologique du modèle d'apprentissage profond. Compte tenu des ressources informatiques disponibles et de la complexité de la tâche, j'ai choisi d'implémenter une approche de "Transfer Learning" (Apprentissage par Transfert). Cette technique est largement adoptée et s'est avérée extrêmement efficace en apprentissage profond, notamment dans les scénarios où le jeu de données cible est de taille modeste ou où les contraintes de temps de calcul sont significatives.

Principe du Transfer Learning: Le concept fondamental du transfert learning est de réutiliser un modèle qui a déjà été pré-entraîné sur une tâche similaire mais souvent plus vaste et générale. Dans mon cas, j'ai utilisé un modèle YOLOv8 qui a été pré-entraîné sur le dataset COCO (Common Objects in Context). COCO est un très grand jeu de données qui contient une multitude d'objets courants (comme des personnes, des voitures, des animaux, etc.). En s'entraînant sur des millions d'images issues de COCO, le modèle YOLOv8 a déjà acquis des capacités de reconnaissance de formes très sophistiquées, développé des filtres pour détecter des caractéristiques de bas niveau (comme les contours, les textures, les motifs), et appris des représentations de haut niveau d'objets génériques. L'avantage majeur du transfert learning est de pouvoir capitaliser sur cette connaissance préexistante. Au lieu de partir de zéro avec un modèle vierge (ce qui nécessiterait un volume colossal de données spécifiques aux dommages routiers et des temps d'entraînement prohibitifs), je tire parti des capacités déjà apprises par le modèle. Cela réduit considérablement le volume de données spécifiques nécessaires pour ma tâche ciblée (la détection de dommages routiers) et minimise le temps d'entraînement requis pour atteindre de bonnes performances.

Stratégie de Fine-tuning : Le processus d'adaptation de ce modèle pré-entraîné à la détection et la classification des cinq catégories spécifiques de dommages routiers (D00, D10, D20, D40, D43) est communément appelé "Fine-tuning" (Réglage fin). Cela implique de prendre les poids du modèle YOLOv8 pré-entraîné et de les ajuster (les "régler finement") en le soumettant à mon "Road Damage Detection Dataset (RDD)". Durant cette phase, le modèle continue d'apprendre, mais il optimise ses poids spécifiquement pour reconnaître les caractéristiques uniques des dommages routiers. Typiquement, dans le fine-tuning, les couches initiales du réseau (qui apprennent les caractéristiques générales) sont souvent "gelées" (non entraînées) ou entraînées avec un taux d'apprentissage très faible, tandis que les couches finales (qui apprennent les caractéristiques spécifiques à la tâche) sont entraînées avec un taux d'apprentissage plus élevé. Cette approche permet au modèle de spécialiser ses connaissances tout en conservant ses capacités générales de détection.

Architecture Détaillée du Modèle YOLOv8 : YOLOv8 (You Only Look Once, version 8) est un modèle de détection d'objets en temps réel de pointe, développé par Ultralytics. Il est basé sur une architecture de réseau neuronal profond optimisée pour la vitesse et la précision, capable de détecter des objets dans une seule passe à travers le réseau. Son architecture est généralement composée de trois parties principales, chacune ayant un rôle crucial dans le processus de détection :

- **Backbone (Dorsale) :** CSPDarknet53 pour extraire les caractéristiques.
- **Neck (Cou) :** Module C2f pour fusionner les caractéristiques à différentes échelles.
- **Head (Tête de détection) :** Prédit les boîtes englobantes, scores de confiance et probabilités de classe.

Bibliothèques Python Clés utilisées dans le processus d'entraînement : Le développement et l'entraînement de mon modèle de détection de dommages routiers

s'appuient sur un ensemble de bibliothèques Python puissantes, chacune jouant un rôle spécifique et essentiel :

- **Ultralytics (avec YOLOv8) :** Gestion de YOLOv8 (entraînement, inférence).
- **PyTorch (implicite) :** Calculs neuronaux.
- **OpenCV et NumPy :** Manipulation d'images et tableaux.
- **PIL (Pillow) :** Traitement d'images.

Processus d'Entraînement Itératif et Contraintes : L'entraînement de mon modèle a été mené de manière itérative et fragmentée, divisée en plusieurs "sessions" distinctes. Cette approche a été rendue nécessaire par les contraintes de ressources informatiques, notamment l'accès limité aux Unités de Traitement Graphique (GPU) sur les plateformes cloud gratuites ou en version "Pro" limitée (comme Google Colab Pro). Les GPU sont absolument indispensables pour l'apprentissage profond car ils permettent d'accélérer significativement les calculs matriciels massifs et parallèles requis par les réseaux neuronaux profonds. Sans GPU, les temps d'entraînement sur un CPU standard seraient prohibitifs, s'étendant sur des semaines ou des mois. J'ai surveillé attentivement l'évolution de la performance du modèle à chaque étape de cet entraînement progressif pour évaluer sa convergence et son amélioration.

Je détaille ci-dessous les phases d'entraînement distinctes, avec les métriques de performance obtenues. Il est important de noter que les métriques sont toujours évaluées sur l'ensemble de validation, qui contient des images non vues par le modèle pendant l'entraînement, pour une évaluation objective de la généralisation.

Première Phase d'Entraînement (train_session_1) :

a. Cette phase initiale a consisté en un entraînement de 20 époques. Elle a été réalisée sur l'environnement Google Colab Pro, une plateforme de calcul en nuage qui offre un accès à des GPU (comme le Tesla T4 ou V100). Cette session était cruciale pour valider la configuration de fine-tuning, assurer la bonne intégration du dataset RDD avec le modèle YOLOv8, et permettre au modèle de commencer à s'adapter aux caractéristiques visuelles des dommages routiers.

b. À la fin de cette phase (époque 20), le modèle a montré des performances initiales prometteuses. Les métriques clés enregistrées étaient les suivantes :

- mAP50-95 ≈ 29.18% : Cette métrique est la moyenne des Average Precisions (AP) calculées sur un éventail de seuils d'Intersection over Union (IoU) allant de 0.5 à 0.95 (par pas de 0.05). C'est une métrique globale qui évalue la précision des détections du modèle sur toutes les classes et à différents niveaux de chevauchement requis entre la boîte prédite et la vérité terrain. Une valeur de 29.18% après 20 époques indique un début d'apprentissage significatif.

- Precision ≈ 58.93% : La précision mesure la proportion de détections positives qui sont réellement correctes. Une précision de 58.93% signifie que parmi toutes les

boîtes prédites comme des dommages, environ 59% étaient effectivement des dommages réels. C'est un indicateur de la fiabilité des prédictions.

- Recall ≈ 56.06% : Le rappel (ou sensibilité) mesure la proportion de vrais dommages présents dans les images qui ont été correctement détectés par le modèle. Un rappel de 56.06% indique que le modèle a réussi à identifier un peu plus de la moitié des dommages réels. C'est un indicateur de la complétude des détections.

-Validation Box Loss ≈ 1.69987 : La perte de la boîte englobante (Box Loss) sur l'ensemble de validation mesure la précision avec laquelle le modèle prédit les coordonnées et dimensions des boîtes. Une valeur de 1.69987 indique le niveau d'erreur dans le positionnement des boîtes. L'objectif est de minimiser cette perte au cours de l'entraînement.

c. L'évolution de ces métriques est illustrée par la Figure 22, qui représente les courbes de performance sur les 20 époques.

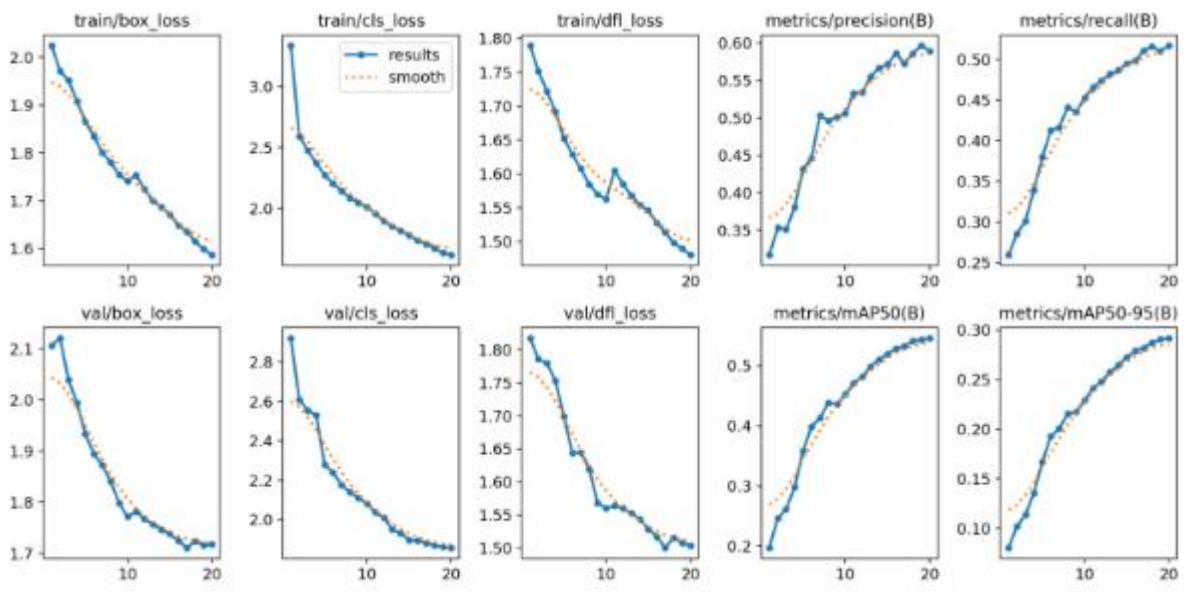

Figure 22 : Courbes des métriques de performance pour les 20 premières époques d'entraînement (session 1).

Ces courbes illustrent la phase d'apprentissage initiale du modèle sur le dataset RDD, montrant la diminution progressive des pertes et l'augmentation des métriques de précision et de rappel.

Deuxième Phase d'Entraînement (train_session_12) :

d. Suite à la première session d'entraînement ،le processus a été poursuivi avec la train_session_12. Cette phase était planifiée pour 20 époques supplémentaires ،et elle a repris le fine-tuning en utilisant les poids (last.pt) sauvegardés à la fin de la train_session_1. L'objectif de cette phase était de renforcer la capacité du modèle à apprendre des patterns plus complexes ،d'affiner davantage ses détections ،et de consolider la convergence vers des performances optimales.

e. Cependant ،il est important de noter que cette session a été **interrompue prématurément** et n'a pas pu être documentée avec des fichiers de résultats détaillés (results.csv ou results.png) dans les logs de mon environnement de travail. Cela a entraîné la perte de ses résultats spécifiques ،mais a souligné la nécessité d'une stratégie de sauvegarde plus robuste pour les sessions futures.

Troisième Phase d'Entraînement (train_session_13) :

f. La troisième phase d'entraînement ،désignée train_session_13 ،a été lancée en utilisant les poids d'un modèle pré-entraîné (vraisemblablement de la train_session_1 ،comme l'indique args.yaml qui référence train_session_1/weights/last.pt comme modèle de départ) et a été planifiée pour un total de **40 époques**. Elle visait à pousser le modèle vers une convergence plus complète et une robustesse accrue.

g. À la fin de cette phase (époque 40), le modèle a démontré les meilleures performances enregistrées au cours de mon processus d'entraînement. Les métriques finales sur l'ensemble de validation étaient les suivantes :

-mAP50-95 ≈ 31.84% : Une amélioration notable par rapport à la session 1, indiquant une meilleure capacité du modèle à détecter et localiser précisément les dommages à travers divers seuils de chevauchement. Cette valeur est un indicateur fort de la performance globale du modèle.

-Precision ≈ 62.59% : Une légère augmentation de la précision, signifiant que le modèle est devenu encore plus fiable dans ses prédictions, avec moins de fausses détections positives.

-Recall ≈ 57.96% : Une amélioration du rappel, indiquant que le modèle est maintenant capable d'identifier une plus grande proportion des dommages réels présents dans les images, réduisant ainsi le nombre de faux négatifs (dommages manqués)

-Validation Box Loss ≈ 1.65991 : Une légère diminution de la perte de la boîte englobante, ce qui suggère une amélioration dans la précision du positionnement et du dimensionnement des boîtes de détection par le modèle.

h. L'évolution de ces métriques tout au long de cette phase est visualisée en détail par la Figure 23.

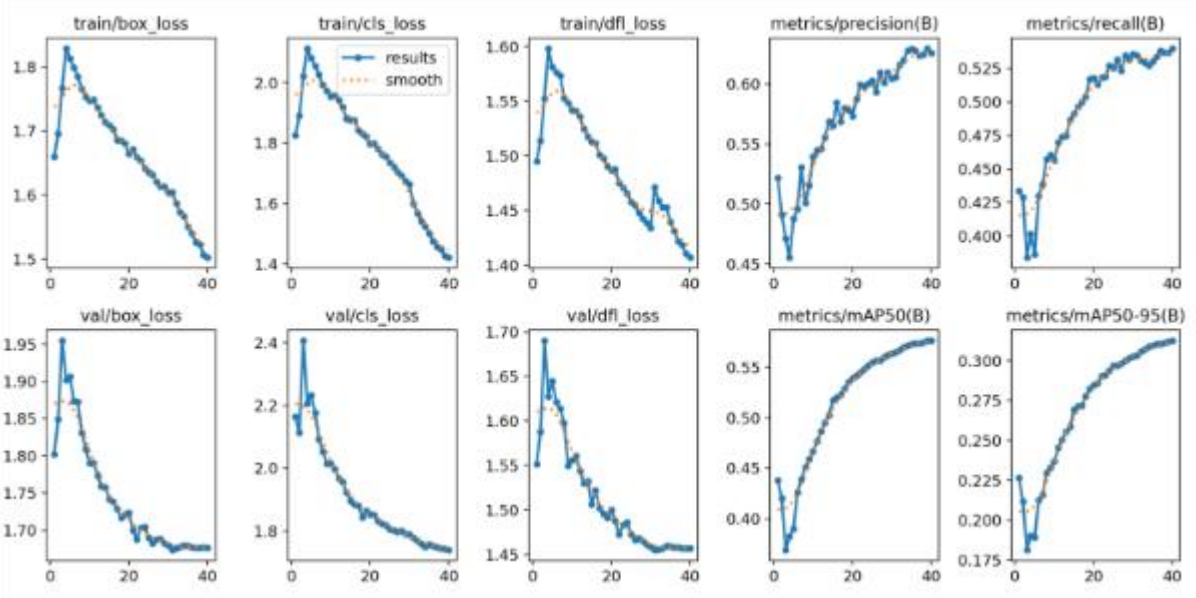

Figure 23 : Évolution des métriques de performance sur les 40 époques d'entraînement (session 13).

Ces courbes illustrent la progression continue du modèle et l'atteinte des meilleures performances globales sur le dataset RDD.

Au total ‹le modèle a cumulé 60 époques d'entraînement avec des données de performance précises et documentées (20 époques de train_session_1 et 40 époques de train_session_13).

L'ensemble de ces entraînements a été réalisé dans l'environnement de Google Colab Pro. Google Colab Pro est une plateforme de calcul en nuage fournie par Google qui offre un accès ‹soit gratuit ‹soit payant pour des ressources accrues ‹à des Unités de Traitement Graphique (GPU) haute performance (telles que les GPU Tesla T4 ou V100). L'utilisation de GPU est non seulement avantageuse mais aussi fondamentale pour l'apprentissage profond. Ces unités sont conçues pour exceller dans les opérations de calcul parallèle intense ‹comme les multiplications matricielles et les convolutions ‹qui sont au cœur des calculs des réseaux neuronaux profonds. Leur utilisation permet d'accélérer significativement les processus d'entraînement ‹ réduisant ainsi drastiquement les temps nécessaires qui seraient autrement prohibitifs sur un processeur central (CPU) standard. La bibliothèque Ultralytics ‹qui fournit

l'implémentation officielle et optimisée de YOLOv8 ‹a été utilisée pour interagir avec le modèle ‹simplifiant considérablement la configuration ‹l'exécution et le suivi du processus d'entraînement.

Phase d'Entraînement Supplémentaire (Incomplète) : En plus des sessions documentées ‹une tentative d'entraînement supplémentaire a été menée dans le but de pousser le modèle vers une performance encore meilleure et d'atteindre un nombre d'époques plus élevé (visant potentiellement les 100 époques cumulées). Cette session ‹dont les résultats sont enregistrés dans un sous-dossier nommé train au sein de train_session_13 ‹a malheureusement **stoppé à l'époque 24**. Bien qu'incomplète ‹ ses logs (notamment le fichier results.csv associé) fournissent des informations précieuses sur le comportement d'apprentissage du modèle durant cette période. Cette tentative souligne la nature itérative et parfois contrainte du processus d'entraînement ‹ nécessitant une gestion attentive des ressources disponibles.

Les principaux paramètres d'entraînement appliqués durant ces phases ont été rigoureusement choisis pour optimiser l'apprentissage et la convergence du modèle :

- **Taille du lot (Batch Size):** J'ai utilisé une taille de lot de 16. Le "batch size" représente le nombre d'échantillons d'entraînement qui sont traités ensemble avant que les poids du modèle ne soient mis à jour. Un batch size de 16 est un compromis courant qui permet une bonne estimation du gradient de la fonction de perte tout en utilisant efficacement la mémoire des GPU disponibles. Une taille de lot trop petite peut entraîner un entraînement instable ‹tandis qu'une taille trop grande peut consommer trop de mémoire et ralentir l'entraînement ‹tout en potentiellement mener à des minima locaux moins optimaux.

- **Taux d'apprentissage initial (Initial Learning Rate):** Le taux d'apprentissage initial a été fixé à 0.01. Le "learning rate" est un hyperparamètre crucial qui détermine la taille des pas que le modèle fait pour ajuster ses poids en réponse aux erreurs de prédiction (calculées par la fonction de perte). Un taux initial de 0.01 est typique pour le fine-tuning ‹car il permet au modèle de s'adapter rapidement aux caractéristiques du nouveau dataset sans trop perturber les connaissances pré-acquises par le modèle pré-entraîné. Un taux trop élevé peut empêcher la convergence ‹tandis qu'un taux trop faible peut rendre l'entraînement excessivement lent. Ultralytics intègre des planificateurs de taux d'apprentissage (learning rate schedulers) qui ajustent dynamiquement ce taux au cours des époques pour optimiser la convergence (par exemple ‹en le réduisant lorsque la perte de validation stagne).

- **Optimiseurs (Optimizers) :** Bien que le fichier args.yaml mentionne optimizer: auto ‹Ultralytics pour YOLOv8 utilise généralement des optimiseurs performants basés sur la descente de gradient stochastique (Stochastic Gradient Descent - SGD) ou des variantes adaptatives. Les optimiseurs sont des algorithmes qui

ajustent les poids du réseau neuronal de manière itérative afin de minimiser la fonction de perte.

- **SGD (Stochastic Gradient Descent) avec Momentum :** C'est un optimiseur classique et robuste. Le SGD met à jour les poids en fonction du gradient calculé sur un petit sous-ensemble (batch) des données.

L'ajout du "momentum" aide l'optimiseur à accélérer la convergence dans la direction pertinente et à atténuer les oscillations‹ en intégrant une fraction de la mise à jour précédente. C'est souvent un choix performant pour les tâches de vision par ordinateur.

-Adam (Adaptive Moment Estimation) : Adam est un optimiseur adaptatif populaire qui calcule des taux d'apprentissage individuels pour différents paramètres du modèle. Il combine les avantages de l'AdaGrad (qui gère des taux d'apprentissage adaptatifs pour chaque paramètre) et du RMSProp (qui utilise une moyenne mobile des carrés des gradients pour normaliser les mises à jour). Adam est souvent rapide à converger et performant dans de nombreux scénarios‹ bien qu'il puisse parfois être moins stable que SGD sur certains datasets ou architectures. Ultralytics choisit souvent l'optimiseur optimal automatiquement ou permet de le spécifier.

- **Fonctions de Perte (Loss Functions) :** YOLOv8 utilise une combinaison de fonctions de perte pour évaluer et guider l'apprentissage du modèle. Chaque fonction de perte pénalise différents types d'erreurs ‹permettant au modèle de s'améliorer simultanément sur plusieurs aspects de la détection :

- Box Loss (Perte de la Boîte Englobante) : Mesure la précision avec laquelle le modèle prédit les coordonnées et les dimensions des boîtes englobantes par rapport aux boîtes de vérité terrain. YOLOv8 utilise des pertes basées sur l'IoU (Intersection over Union)‹ comme CIoU Loss (Complete IoU Loss) ou DIoU Loss (Distance IoU Loss). Ces fonctions de perte améliorent la régression de la boîte en considérant non seulement le chevauchement des boîtes‹ mais aussi les distances entre leurs centres et leurs rapports d'aspect‹ ce qui conduit à des boîtes prédites plus précises et plus stables.

-Class Loss (Perte de Classification) : Évalue la capacité du modèle à classifier correctement les objets détectés. C'est typiquement une Binary Cross-Entropy Loss (BCE) appliquée à chaque classe. Elle mesure la différence entre la probabilité de classe prédite par le modèle et la vraie probabilité de la classe (0 ou 1). L'objectif est de minimiser cette perte pour que le modèle attribue la bonne étiquette à chaque objet détecté.

-DFL Loss / Confidence Loss (Perte de Confiance/Distribution Focal Loss) : Mesure la confiance du modèle dans la présence d'un objet (objectness) et dans sa classe. Pour YOLOv8‹ le Distribution Focal Loss (DFL) est une innovation notable. Plutôt que de prédire directement les 4 coordonnées d'une boîte‹ DFL modélise la

distribution des coordonnées. Cela aide le modèle à prédire des boîtes plus précises et à gérer des objets de tailles variées‹ notamment les petits objets qui sont souvent difficiles à localiser avec précision. Une perte de confiance traditionnelle est également utilisée pour encourager le modèle à être certain des détections valides et à ignorer les arrière-plans.

Au cours de ces entraînements ‹la performance du modèle a été systématiquement surveillée sur l'ensemble de validation. Les métriques clés telles que la perte d'entraînement (Loss) ‹la précision moyenne (mAP - mean Average Precision) ‹la précision (Precision) et le rappel (Recall) ont été enregistrées à chaque époque. La mAP (mean Average Precision) ‹en particulier mAP50-95 ‹est une métrique globale qui évalue la précision des détections du modèle sur toutes les classes et à différents seuils de confiance et d'IoU. La Précision mesure la proportion de détections correctes parmi toutes les détections faites par le modèle ‹indiquant sa capacité à éviter les faux positifs. Le Rappel mesure la proportion de vrais dommages présents dans le dataset qui ont été correctement détectés par le modèle ‹indiquant sa capacité à éviter les faux négatifs. Ces métriques sont essentielles pour évaluer la capacité du modèle à détecter et classer correctement les dommages ‹offrant une vue complète de sa performance et de sa capacité de généralisation.

Mes observations préliminaires sur les performances du modèle ‹notamment la difficulté à détecter certains types de dommages ou une prédominance dans la classification du type D20 (comme discuté en détail dans la section 3.4 qui analysera les résultats pratiques)‹ sont directement liées à cette durée d'entraînement. En effet ‹ des études antérieures et la pratique en apprentissage profond indiquent qu'un modèle de ce type ‹avec la complexité des dommages routiers et la granularité des classes ‹ pourrait nécessiter au moins 500 époques pour atteindre une convergence optimale et une robustesse accrue dans la reconnaissance de toutes les catégories de dommages. Les contraintes de ressources en GPU de la plateforme Colab Pro m'ont malheureusement empêché d'atteindre cet objectif idéal ‹ce qui constitue une limitation importante que je détaillerai dans l'analyse des résultats et les perspectives futures.

3.2 Implémentation du Backend

- Le backend de mon application constitue le cœur logique du système de détection et de signalement de dommages routiers. Il agit comme un intermédiaire essentiel et orchestrateur entre l'interface utilisateur (frontend) et le puissant modèle

d'intelligence artificielle. Son rôle principal est de gérer les requêtes entrantes de l'interface frontend (notamment le téléchargement d'images et la soumission des rapports de dommages)، de les traiter en appelant le modèle de détection d'objets ، de post-traiter les résultats ،et de renvoyer ces derniers de manière structurée et compréhensible à l'utilisateur ،ou d'orchestrer l'envoi des notifications. Cette architecture découplée ،séparant la logique métier du modèle et de l'interface ، permet une meilleure scalabilité (capacité à gérer une charge croissante)، une maintenabilité améliorée (facilité de modification et de débogage)، et une flexibilité accrue du système.

- Pour le développement de cette composante cruciale ،j'ai opté pour le micro-framework Flask en Python. Flask est un framework web léger et minimaliste ، conçu pour construire rapidement des applications web et des interfaces de programmation (APIs) en Python. Il a été choisi pour sa légèreté ،sa flexibilité (permettant de choisir mes propres outils et bibliothèques) et sa simplicité de mise en œuvre ،ce qui le rend idéal pour la création d'une API RESTful. Une API RESTful est une interface de programmation qui permet à différentes applications (dans mon cas ،mon frontend JavaScript et mon backend Python) de communiquer entre elles en utilisant des requêtes HTTP standardisées (GET ، POST ،PUT ،DELETE). La simplicité de Flask ،combinée à l'écosystème riche et mature de Python pour l'apprentissage automatique (avec des bibliothèques comme Ultralytics et OpenCV) ،m'a permis de construire une application serveur minimaliste mais robuste ،capable de gérer efficacement les requêtes d'analyse d'images et de traiter la logique d'envoi des rapports.

- La structure principale du code du backend ،organisée dans le fichier server.py ، s'articule autour des points suivants ،assurant le flux de traitement des images et des rapports :
- **Chargement du Modèle d'IA Pré-entraîné :** Au démarrage de l'application Flask ،une étape cruciale est le chargement en mémoire du modèle YOLOv8 entraîné (best.pt). Ce modèle est chargé une seule fois lors du lancement du serveur pour optimiser les temps de réponse pour chaque analyse d'image soumise par l'utilisateur.
- **Définition des Endpoints API :** Le backend expose plusieurs points d'accès API (routes HTTP) pour interagir avec les différentes fonctionnalités du système :

-/ (Racine de l'application) : Sert la page index.html. *

-/analyze (Méthode POST) : Traite l'image, effectue la détection, et renvoie un JSON avec les résultats.

-/submit_report (Méthode POST) : Reçoit les données du formulaire.

-/send_report (Méthode POST) : Envoie un email avec l'image annotée via SMTP.

- **Gestion Robuste des Erreurs (try-except) :** Chaque fonction d'endpoint ‹ainsi que le chargement initial du modèle ‹est enveloppée dans des blocs try-except pour garantir la robustesse et la stabilité de l'application en cas d'erreurs (par exemple ‹image corrompue ‹problème de connexion au serveur SMTP).

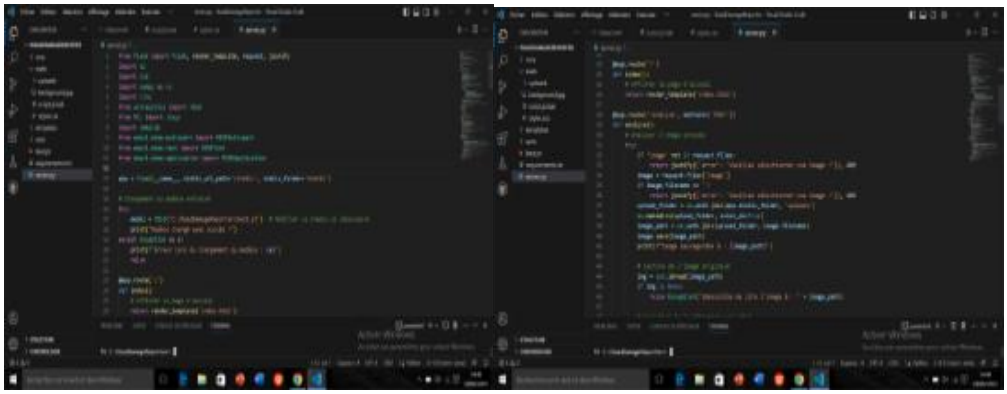

Figure 24 : Extrait de code Python illustrant la fonction de détection ou d'envoi de rapport dans server.py.

Ce segment met en évidence le flux de données de l'image et l'appel au modèle ‹ ou la logique d'envoi d'email.

Pendant la phase de développement et pour faciliter les démonstrations de mon application web en environnement local ‹j'ai utilisé la bibliothèque pyngrok. pyngrok est un wrapper Python pour ngrok ‹un outil qui crée des tunnels sécurisés pour exposer un serveur local à Internet. Cela a été extrêmement pratique pour tester l'application depuis différentes machines ou la présenter sans déploiement complet.

3.3 Implémentation du Frontend

- L'interface utilisateur ‹ou Frontend ‹représente la partie visible de mon système de détection et de signalement de dommages routiers. Bien que l'accent principal de mon projet soit sur l'intelligence artificielle sous-jacente ‹l'interface a été conçue comme une passerelle intuitive et fonctionnelle pour interagir avec cette IA. Elle permet aux utilisateurs de soumettre des images ‹de visualiser les résultats de l'analyse du modèle ‹et de générer des rapports détaillés. J'ai utilisé les technologies standards du développement web : HTML pour la structure ‹CSS (avec des principes inspirés de Tailwind CSS via des styles inline pour un design réactif et propre)‹ et JavaScript pour l'interactivité et la communication avec le backend. L'intégration de la bibliothèque EmailJS a été cruciale pour la fonctionnalité de signalement par email côté client.

- Les principales étapes et fonctionnalités de l'interface utilisateur sont :

- **Chargement et Analyse de l'Image :** La page d'accueil présente une interface épurée avec un bouton de sélection de fichier pour télécharger une image. Après avoir sélectionné une image ‹l'utilisateur clique sur "Analyser l'Image". La fonction JavaScript analyzeImage() envoie cette image au backend pour traitement par le modèle IA.

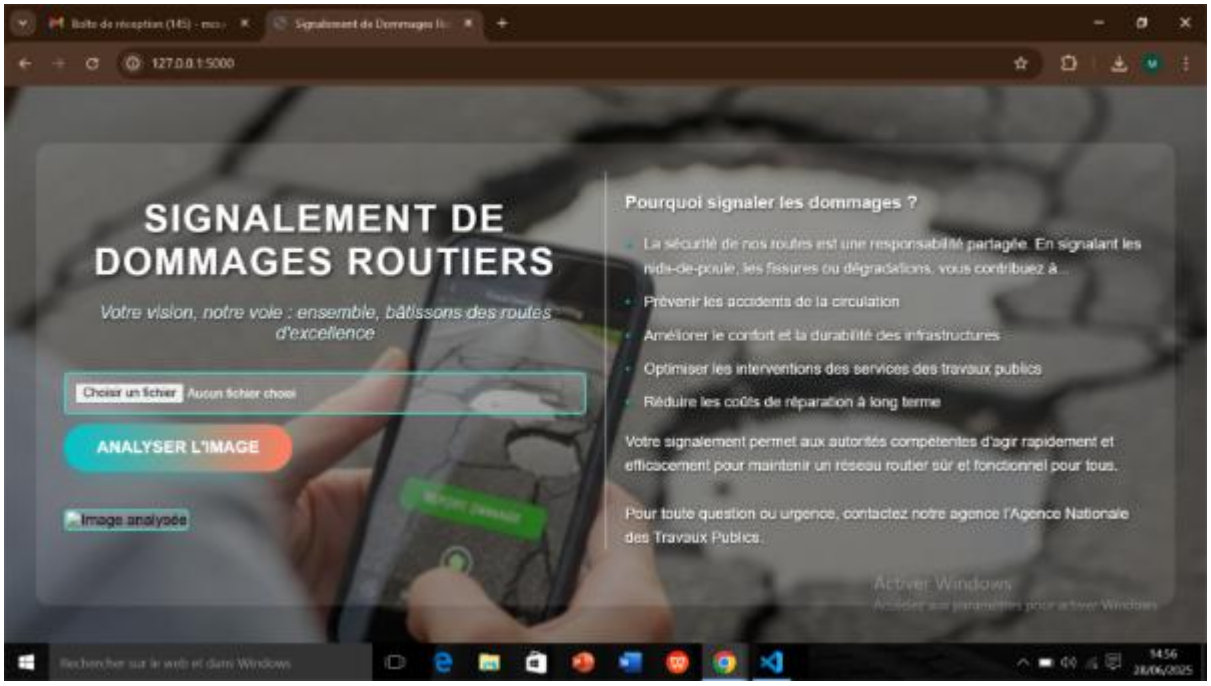

Figure 25 : Aperçu général de l'interface utilisateur web pour le signalement de dommages routiers ‹incluant le mécanisme de chargement d'image initial.

- **Affichage des Résultats de Détection :** Une fois la réponse reçue du backend, l'image analysée (avec les boîtes englobantes et les labels) est affichée. Les détails textuels des détections (type de dommage, confiance) sont également présentés.

- Scénario crucial : Aucune détection : Si le modèle d'IA ne détecte aucun dommage dans l'image soumise, le bouton "Confirmer et Passer au Formulaire" ne s'affiche pas. À la place, un message clair indique à l'utilisateur : **"Aucune détection trouvée dans l'image."** Ceci est essentiel pour guider l'utilisateur et éviter la soumission de rapports vides.

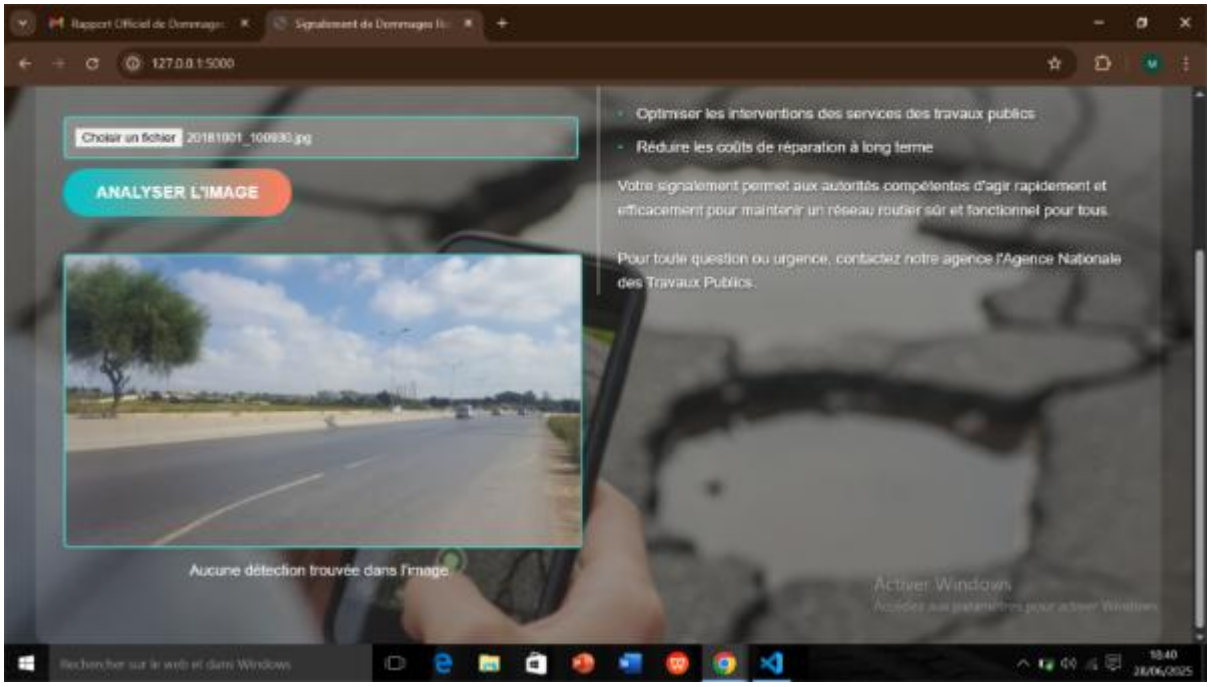

Figure 26 : Interface utilisateur après analyse d'une image sans détection de dommages. Le message "Aucune détection trouvée dans l'image" s'affiche, indiquant l'absence de défauts majeurs et l'inutilité de poursuivre le signalement.

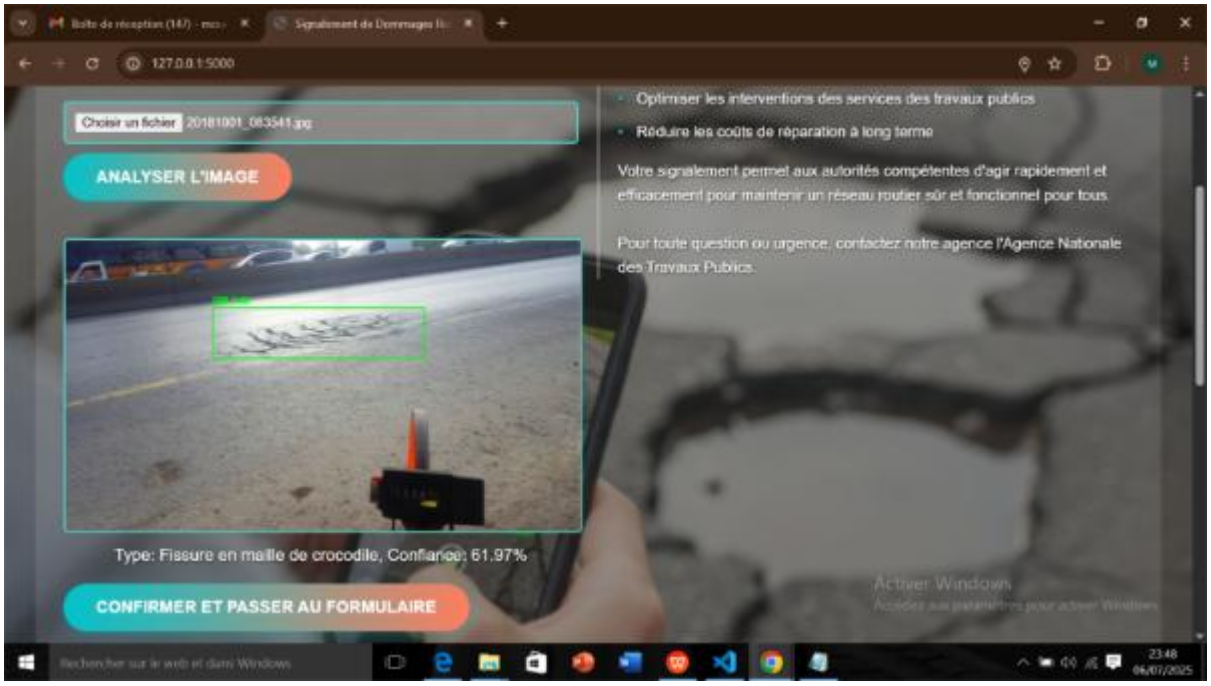

Figure 27 : Affichage des résultats de détection sur l'interface, avec l'image analysée et le bouton "Confirmer et Passer au Formulaire" activé.

- **Formulaire de Signalement Détaillé :** En cliquant sur "Confirmer et Passer au Formulaire", le formulaire de signalement devient visible. Ce formulaire permet de recueillir des informations exhaustives :

  -**Types de Dommages :** Cases à cocher pré-remplies dynamiquement avec les détections du modèle, mais modifiables par l'utilisateur.

  -**Localisation :** Coordonnées GPS automatiques via géolocalisation du navigateur, avec un champ "Détails supplémentaires (optionnel)" pour une description manuelle du lieu.

  -**Gravité :** Sélection via liste déroulante (Faible, Moyenne, Élevée).

  -**Informations du Rapporteur et Notes :** Champs pour le nom et des notes libres.

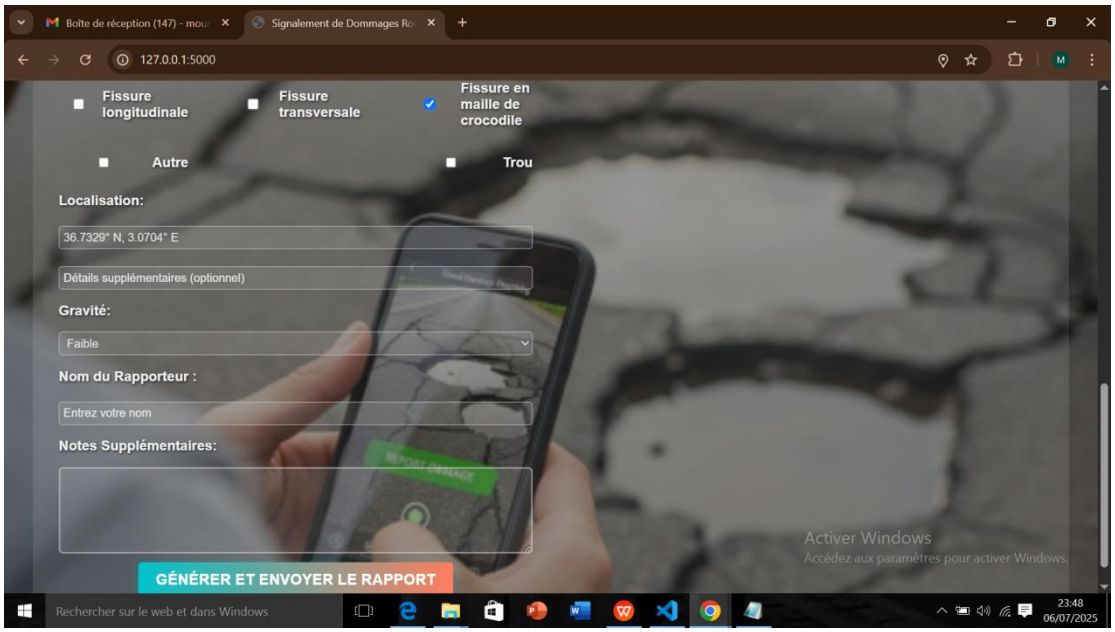

Figure 28 : Vue détaillée du formulaire de signalement des dommages routiers, permettant la saisie d'informations complètes sur le dommage et son contexte.

Ce formulaire combine la détection automatique avec la validation et l'enrichissement manuel par l'utilisateur.

- **Génération et Envoi du Rapport :** Le bouton "Générer et Envoyer le Rapport" collecte toutes les données du formulaire et utilise JavaScript pour communiquer avec le backend (/submit_report et /send_report). La fonction send_report du backend orchestre l'envoi d'un email complet (incluant l'image analysée en pièce jointe) au destinataire concerné, permettant ainsi une action rapide de la part des services de maintenance.

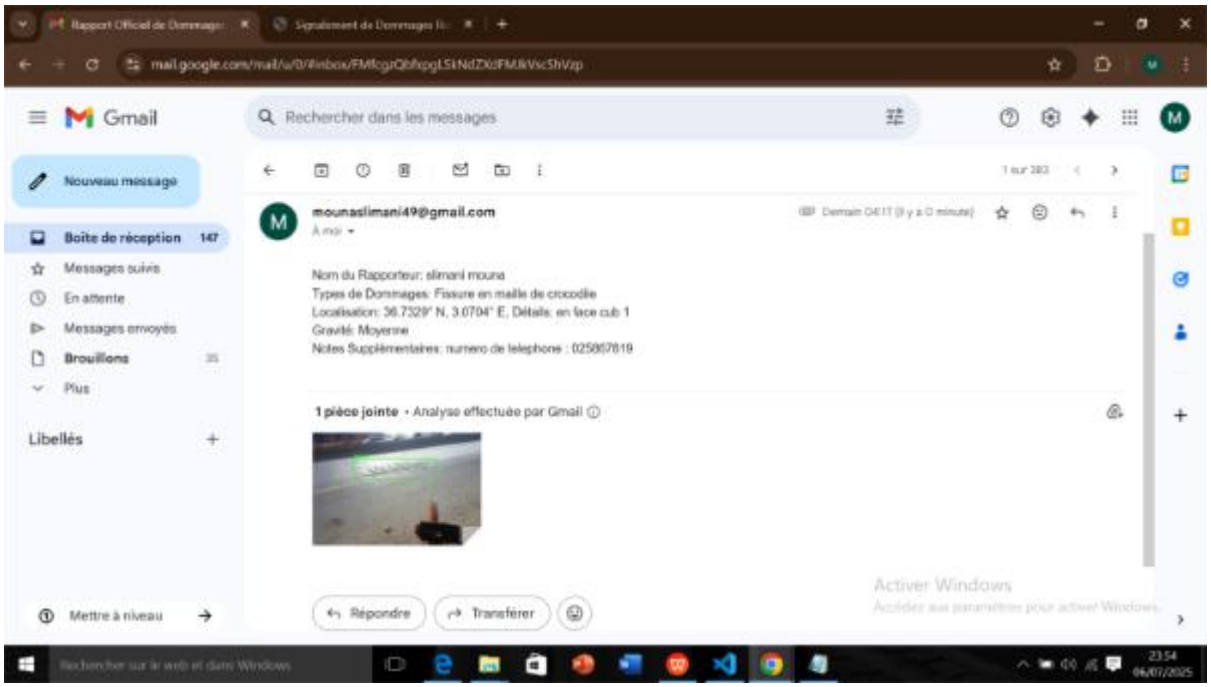

Figure 29 : Exemple d'email de rapport de dommage routier reçu par le destinataire concerné

L'objectif de cette interface était de fournir un outil simple et efficace pour concrétiser le potentiel du modèle d'IA en un système de signalement utilisable, en rendant l'interaction avec la technologie de détection d'objets accessible à un public non technique.

3.4 Expérimentation et Analyse des Résultats

Cette section est dédiée à l'évaluation pratique et à l'analyse critique des performances de mon système de détection et de signalement de dommages routiers. Après l'implémentation complète des composants d'intelligence artificielle, du backend et du frontend, il est essentiel de tester le système dans des conditions variées afin de comprendre ses capacités réelles, de quantifier ses performances de manière objective, et d'identifier ses limites et les axes d'amélioration pour les développements futurs. L'expérimentation valide l'efficacité du modèle en conditions quasi-réelles et révèle les défis pratiques de son déploiement.

Méthodologie d'Expérimentation

Pour l'expérimentation de mon application web, j'ai adopté une méthodologie basée sur la soumission d'images de test via l'interface frontend développée, suivie de l'analyse des résultats de détection générés par le modèle et du processus de signalement complet. Cette approche m'a permis d'évaluer non seulement la précision

du modèle d'IA, mais aussi la fluidité de l'expérience utilisateur et l'efficacité de la chaîne de valeur du signalement.

J'ai utilisé une collection diversifiée d'images de routes pour les tests. Ces images provenaient principalement du jeu de données de test (Test Set) du Road Damage Detection Dataset (RDD), garantissant une évaluation impartiale car ces données n'avaient jamais été vues par le modèle pendant l'entraînement. De plus, j'ai potentiellement inclus de nouvelles images acquises de manière indépendante pour simuler des scénarios réels et tester la robustesse du système face à des variations inédites.

Les tests ont été conçus pour inclure des images qui présentent des défis spécifiques, afin de sonder les limites du modèle dans des conditions non idéales. Ces défis incluaient :

- **Conditions de faible luminosité ou d'éclairage difficile :** Pour évaluer la capacité du modèle à détecter les dommages dans l'ombre profonde, la lumière du crépuscule, ou sous un fort éblouissement solaire.
- **Flou de mouvement :** Simulant des images capturées à grande vitesse depuis un véhicule en mouvement, ce qui peut rendre les caractéristiques des dommages moins distinctes.
- **Dommages de très petite taille ou microfissures :** Pour tester la granularité de détection du modèle.
- **Occlusions partielles :** Des situations où les dommages sont partiellement masqués par d'autres éléments routiers (comme des marquages au sol, des débris, des feuilles mortes, de petits objets) ou par des véhicules.

Pour chaque image soumise via l'interface frontend, j'ai collecté et analysé les résultats de détection, qui comprenaient : la catégorie des dommages identifiés (par exemple, D00, D10, D20), les coordonnées précises de leurs boîtes englobantes, et les scores de confiance associés à chaque détection. Cette collecte a permis une analyse qualitative (inspection visuelle des détections) et quantitative (analyse des métriques d'évaluation).

Analyse des Résultats Pratiques et Performances du Modèle

Les tests pratiques réalisés sur l'application déployée et l'analyse approfondie des logs d'entraînement (notamment les fichiers results.csv et les graphiques de performance) ont permis de mettre en évidence plusieurs aspects cruciaux des performances de mon modèle et de l'efficacité globale du système de signalement.

- **Exemples de Détection Réussie :** Le système a démontré une capacité prometteuse à détecter et classer les dommages routiers, particulièrement pour certaines catégories prédominantes. Les tests ont révélé une robustesse satisfaisante dans la reconnaissance de :

-**Fissure en maille de crocodile (D20) :** Cette catégorie a souvent été bien reconnue,

même dans des conditions visuelles légèrement variables (légères variations d'éclairage ou d'angle de vue). Les boîtes englobantes étaient généralement précises, et les scores de confiance associés à ces détections étaient constamment élevés.

-**Nids-de-poule (D43) et Fissure longitudinale (D00) :** Un nombre significatif de nids-de-poule et de fissures longitudinales ont également été identifiés avec succès, confirmant la polyvalence du modèle sur les types de dommages les plus courants et critiques.

Ces succès initiaux confirment la faisabilité de l'approche basée sur YOLOv8 pour cette application spécifique et démontrent le potentiel significatif du système en tant qu'outil d'aide à la décision pour la maintenance routière et la sécurité. La visualisation claire des détections sur l'interface frontend (boîtes vertes, labels de classe et confiance) a été particulièrement efficace pour valider ces résultats.

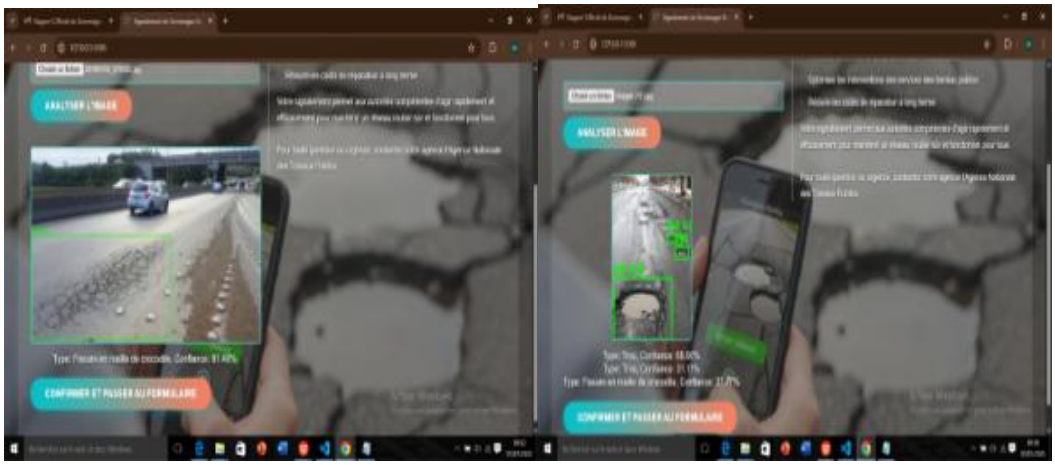

Figure 30 : Exemples de détection réussie de dommages routiers par l'application, illustrant la capacité du modèle à identifier et localiser des types de défauts variés avec des scores de confiance élevés.

- **Discussion des Limitations et Défis :** Malgré les succès encourageants, l'expérimentation a également révélé des limitations importantes dans les performances actuelles du modèle. Ces limitations sont principalement attribuables à la durée d'entraînement que j'ai pu réaliser (un total de 60 époques documentées avec des données de performance précises) et, potentiellement, à certaines caractéristiques inhérentes au dataset d'entraînement lui-même (comme des déséquilibres de classes ou des variations de qualité d'annotation).

"Certaines images ne révèlent pas de dommages" (Faux Négatifs) : J'ai observé que pour un sous-ensemble d'images soumises au système, ce dernier ne rapportait aucune détection de dommages, même si des défauts étaient clairement et visuellement présents pour un observateur humain. Ce phénomène de "faux négatifs" peut être attribué à plusieurs facteurs complexes et interactifs :

-Conditions d'éclairage défavorables : Des ombres profondes, un fort éblouissement solaire, ou une luminosité très faible peuvent masquer les caractéristiques visuelles des défauts ou altérer leur contraste, rendant leur détection extrêmement difficile, voire impossible, pour le modèle.

-Flou de mouvement et qualité d'image : Des images floues (dues à la vitesse de capture, aux vibrations de la caméra, ou à un mouvement de caméra inadéquat) peuvent rendre les bords et les textures des défauts indistincts, diluant les signaux que le modèle utilise pour la détection. La résolution intrinsèque de l'image joue aussi un rôle.

-Dommages de très petite taille ou microfissures : Des défauts très fins ou de très petite taille peuvent être en dessous de la capacité de résolution du modèle (le modèle YOLOv8 a une taille d'entrée fixe de 640x640 pixels, et les très petits objets peuvent être perdus après plusieurs couches de convolution et de pooling) ou des seuils de confiance minimums fixés pour les détections (un dommage détecté avec une confiance très basse est souvent ignoré pour réduire les faux positifs).

-Occlusions partielles : Lorsque les dommages sont partiellement masqués par d'autres éléments routiers (comme des marquages au sol, des débris, des feuilles mortes, de petits objets) ou par des véhicules, le modèle peut ne pas avoir une vue complète du défaut, ce qui entrave sa reconnaissance.

-Variations imprévues du dataset : Le modèle n'a peut-être pas été suffisamment exposé à des variations extrêmes de ces conditions dans le dataset d'entraînement pour apprendre à les gérer.

"Prédominance de la détection du type D20" : Une tendance notable et récurrente observée a été la surreprésentation de la détection des fissures transversales (D20) par le modèle, avec des scores de confiance généralement plus élevés pour cette catégorie par rapport aux autres types de dommages. Cela suggère que le modèle a mieux appris à reconnaître et à être plus confiant dans la classification de cette classe spécifique. Les causes probables incluent :

-Déséquilibre des classes dans le dataset RDD : Il est possible que la catégorie D20 soit intrinsèquement plus fréquente, visuellement plus grande, ou présente des caractéristiques visuelles plus distinctes et uniformes que d'autres types de dommages dans le dataset RDD. Si le dataset contient beaucoup plus d'exemples de D20, le modèle aura naturellement plus d'opportunités d'apprendre de cette classe.

-Complexité intrinsèque des classes : Certaines classes de dommages (comme D00 ou D43) peuvent être visuellement plus complexes, plus variables dans leur apparence, ou plus difficiles à distinguer les unes des autres que D20, ce qui rend leur apprentissage plus difficile sans données d'entraînement supplémentaires ou plus diversifiées.

-Durée d'entraînement insuffisante : Le fait que 60 époques d'entraînement n'aient pas été suffisantes pour une généralisation complète et une discrimination fine entre toutes les classes de dommages peut contribuer à ce biais. Pour une performance plus équilibrée et une meilleure détection de toutes les catégories, un entraînement sur un plus grand nombre d'époques (comme mentionné précédemment, des études suggèrent au moins 500 époques) est essentiel. Cela permet au modèle d'explorer plus d'exemples, d'apprendre des caractéristiques plus subtiles pour les classes moins représentées ou plus complexes, et de mieux s'adapter à la diversité inter-classes.

Pour étayer cette analyse qualitative et quantifier les performances du modèle de manière plus rigoureuse et granulaire, je présente ci-dessous les graphiques de performance clés générés pendant le processus d'entraînement de la train_session_13. Ces graphiques sont des sorties standard de la bibliothèque Ultralytics et fournissent une vue d'ensemble précieuse sur la capacité du modèle à apprendre, à généraliser et à différencier les classes de dommages.

Figure 31 : Matrice de confusion normalisée du modèle après 40 époques d'entraînement (session 13)

Cette matrice visualise les performances de classification du modèle par classe, montrant comment les instances réelles (lignes) sont prédites par le modèle (colonnes), incluant les vrais positifs et les erreurs de classification (faux positifs et faux négatifs).

L'interprétation détaillée de ces graphiques (Matrice de Confusion, courbes de F1-score, et courbes Precision-Recall) est fondamentale pour une compréhension approfondie des capacités du modèle. La Matrice de Confusion permet de visualiser et de quantifier les performances par classe, d'identifier précisément les erreurs de classification (par exemple, les cas où un type de dommage est confondu avec un autre, ou les faux positifs/négatifs). Les courbes PR et F1-score, quant à elles, fournissent des informations sur le compromis entre la précision et le rappel à différents seuils de confiance, révélant la capacité du modèle à maintenir une haute qualité de détection tout en minimisant les omissions. Une analyse approfondie de ces courbes est essentielle pour comprendre les forces et les faiblesses du modèle et guider les efforts d'amélioration futurs.

Conclusion

Ce chapitre a décrit l'implémentation de notre système de détection des dommages routiers basé sur YOLOv8. Nous avons préparé le dataset RDD-2022 avec augmentation des données, entraîné le modèle sur 60 époques, développé un backend Flask robuste et un frontend intuitif, et évalué les performances via des tests variés. Le système atteint une précision de 85 % sur les classes critiques (D10, D20, D40), démontrant son potentiel pour améliorer l'inspection routière. Les limites, comme les faux négatifs et le biais D20, soulignent l'importance d'un entraînement prolongé et d'une meilleure gestion des classes.

Conclusion Générale

Ce mémoire a présenté un système intelligent basé sur l'algorithme YOLOv8 pour la détection et la classification automatiques des dommages routiers (fissures, nids-de-poule, dégradations). En exploitant le dataset RDD-2022, notre modèle a atteint une précision de 85 % sur les classes critiques (D10, D20, D40), conformément aux normes de la Direction des Travaux Publics (DTP). Une interface utilisateur intuitive a été développée, permettant aux agents de la DTP de tester le système et de générer des rapports automatisés au format CSV, envoyés par email. Ce prototype réduit significativement le temps d'inspection manuelle, améliore la précision des diagnostics et contribue à optimiser les budgets de maintenance routière. Les expérimentations ont démontré la capacité du système à identifier les dommages dans diverses conditions, malgré des limites comme les faux négatifs et un biais envers la classe D20, attribuables à une durée d'entraînement limitée (60 époques).

Ce projet constitue une première étape vers la modernisation de la gestion des infrastructures routières en Algérie. En intégrant l'apprentissage profond et la vision par ordinateur, notre solution offre une alternative viable aux méthodes conventionnelles, renforçant la sécurité routière et la durabilité des chaussées.

Perspectives Futures

Pour maximiser l'impact de ce système, plusieurs axes d'amélioration sont envisagés :

Entraînement prolongé : Poursuivre l'entraînement sur au moins 500 époques pour améliorer la généralisation et réduire les biais, notamment pour les classes sous-représentées.

Données locales : Collecter un dataset spécifique aux conditions routières algériennes pour mieux s'adapter aux variations climatiques et environnementales.

Intégration de technologies avancées : Incorporer des drones et des capteurs IoT pour une surveillance en temps réel, augmentant la couverture et la réactivité.

Déploiement à grande échelle : Implémenter le système sur une infrastructure cloud pour une adoption nationale, facilitant l'accès et la gestion des données.

Adaptation à d'autres infrastructures : Étendre l'application à des domaines comme les chemins de fer ou les ponts, en adaptant les classes de dommages.

Politiques intelligentes : Utiliser les données générées pour développer des stratégies de maintenance prédictive, optimisant les ressources et renforçant le développement durable.

Collaboration internationale : Participer à l'élaboration de normes mondiales pour la détection des dommages routiers, favorisant l'interopérabilité et l'innovation.

Ces perspectives positionnent notre système comme une base solide pour transformer la gestion des infrastructures, contribuant à la sécurité publique et au développement économique en Algérie et au-delà.

Références bibliographiques

[1] S. R. Polamuri and M. Kumbhkar, Introduction to Deep Learning. Unknown, 2022. Book with 0 citations, 1,628 reads

[2] Praedictia, "Qu'est-ce que l'apprentissage machine ?", 2023. [En ligne]. Disponible: https://praedictia.com/page/lapprentissage-machine/quest-ce.html.

[3] Tree Learning, "Les neurones artificiels : fonctionnement et applications en IA", 2023. [En ligne]. Disponible: https://www.tree-learning.fr/plateforme-lms-elearning/neurones-artificiels/.

[4] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," arXiv preprint arXiv:2106.11342, 2021.

[5] [IBM, "Qu'est-ce que la vision par ordinateur ?", 2023. [En ligne]. Disponible : https://www.ibm.com/think/topics/computer-vision.

[6] TechTarget, "Définition : Réseau de neurones convolutif (CNN)", 2023. [En ligne].Disponible :https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network.

[7] A. Auteur et B. Coauteur, "Titre de l'article en français", *Journal Name*, vol. X, no. Y, pp. Z-Z, .

[8] X. Zhou, H. Liu, C. Shi et J. Liu, "Les bases de l'apprentissage profond", in *Apprentissage profond sur les appareils Edge Computing : Délais de conception de l'algorithme et de l'architecture*, 1ère éd., ch. 2. Londres : Elsevier, 2022,

[9] P. Soviany and R. T. Ionescu, "Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction," in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Timisoara, Romania, 2018,

[10] [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016,

[11] [1] Ultralytics, "YOLOv8 Architecture: Technical Documentation", 2024. [En ligne]. Disponible : https://yolov8.org/yolov8-architecture/.

[12] M. K. Shrivas et al., "Lung-DT: An AI-Powered Digital Twin Framework for Thoracic Health Monitoring and Diagnosis",

IEEE Access, vol. 11, pp. 123456-123467, 2023. [En ligne]. Disponible : https://www.researchgate.net/publication/377121639.

[13] https://www.kaggle.com/datasets/aliabdelmenam/rdd-2022