République Tunisienne

Ministère de l'Enseignement Supérieur

et de la Recherche Scientifique

Université de Tunis El Manar

Institut Supérieur d'Informatique d'El Manar

Rapport de stage d'été

Spécialité : Ingénierie du développement logiciel

Par

GUEMBRI Lina

Développement d'un système de détection de véhicules

Encadrant professionnel : **ATTOUE Nivine**

Encadrant Académique : **ZAOUALI Sonia**

Réalisé au sein de ADDINN Tunisie

Innovate for change

Année Universitaire 2023 - 2024

République Tunisienne

Ministère de l'Enseignement Supérieur

et de la Recherche Scientifique

Université de Tunis El Manar

Institut Supérieur d'Informatique d'El Manar

Rapport de stage d'été

Spécialité : Ingénierie du développement logiciel

Par

GUEMBRI Lina

Développement d'un système de détection de véhicules

Encadrant professionnel : **ATTOUE Nivine**

Encadrant Académique : **ZAOUALI Sonia**

Réalisé au sein de ADDINN Tunisie

Innovate for change

Année Universitaire 2023 - 2024

J’autorise l’étudiant à faire le dépôt de son rapport.

Encadrant professionnel, **ATTOUE Nivine**

Signature et cachet

J’autorise l’étudiant à faire le dépôt de son rapport.

Encadrant académique, **ZAOUALI Sonia**

Signature et cachet

TABLE OF CONTENT

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES

— YOLO : *You Only Look Once*

— OpenCV : *Open Source Computer Vision Library*

GENERAL INTRODUCTION

L'industrie de l'assurance automobile est confrontée à une transformation profonde, stimulée par l'intégration de technologies avancées. L'un des défis majeurs dans ce domaine est la détection des fraudes, un problème qui nécessite des solutions innovantes pour améliorer la précision et l'efficacité des évaluations de sinistres. Ce projet se concentre sur le développement d'un système de détection de véhicules afin de répondre à ce besoin croissant.

Le contexte de ce projet s'inscrit dans l'effort pour moderniser les processus d'assurance automobile en utilisant des méthodes avancées de traitement d'images et de machine learning. Réalisé au sein d'ADDINN, une entreprise leader dans la fourniture de solutions technologiques, ce projet a pour objectif de concevoir un système capable de détecter avec précision les véhicules dans des conditions variées. En particulier, il se concentre sur la détection de signes potentiels de fraude en analysant les dommages signalés lors des réclamations.

Le système proposé repose sur des techniques de deep learning, choisies pour leur efficacité à traiter des images complexes et à fournir des résultats fiables dans des environnements difficiles, tels que des véhicules partiellement obscurcis ou dans des conditions de faible éclairage. Ces technologies permettent de tirer parti des capacités des modèles modernes pour améliorer les processus d'évaluation des sinistres.

Le rapport s'organisera comme suit : nous commencerons par présenter le contexte et les objectifs du projet, en justifiant le choix des méthodes et outils employés. Ensuite, nous décrirons les étapes de développement du système et les résultats obtenus. Enfin, nous analyserons les performances du modèle, discuterons des limitations rencontrées, et proposerons des recommandations pour des améliorations futures et des applications pratiques du système.

Cette structure vise à fournir une vue d'ensemble complète du projet, mettant en lumière les enjeux technologiques et les contributions apportées au domaine de l'assurance automobile.

CHAPITRE 1

CADRE DU PROJET

Introduction

Ce chapitre fournit un aperçu général du projet. Nous commencerons par exposer le contexte dans lequel le projet s'inscrit. Nous détaillerons ensuite l'étude de l'existant et les critiques des solutions actuelles, avant de proposer une solution adaptée aux besoins spécifiques de ce secteur. Enfin, la méthodologie de travail adoptée pour le développement du modèle de détection de voitures sera présentée, couvrant les principales étapes et approches utilisées.

1.1 Présentation de l'organisme d'acceuil

ADDINN est une entreprise spécialisée dans le développement de solutions numériques innovantes pour divers secteurs, y compris l'assurance. Elle se distingue par son engagement envers l'innovation technologique et la transformation digitale, visant à offrir des services avancés et adaptés aux besoins de ses clients. ADDINN s'appuie sur une équipe de développeurs, d'ingénieurs et d'experts en IA pour créer des solutions de pointe qui répondent aux défis spécifiques du marché.

FIGURE 1.1 – Logo de l'entreprise addinn

1.2 Cadre Général du Projet

Dans le domaine de l'assurance automobile, l'innovation technologique est essentielle pour améliorer la précision et l'efficacité des processus de gestion des réclamations et de détection de fraude. La capacité à détecter automatiquement les véhicules à partir d'images joue un rôle crucial en permettant une évaluation rapide et précise des dommages, ainsi qu'une vérification plus rigoureuse des réclamations. Cette approche facilite l'automatisation de l'analyse d'images, réduisant ainsi les erreurs humaines et accélérant les processus décisionnels. En intégrant des technologies de vision par ordinateur avancées, telles que les modèles de détection d'objets, le projet vise à renforcer les capacités d'analyse et à optimiser les processus d'assurance en fournissant des outils fiables pour détecter les véhicules et évaluer les réclamations de manière efficace.

1.3 Analyse de l'Existant

Actuellement, plusieurs solutions de détection de véhicules sont disponibles sur le marché. Ces solutions utilisent diverses approches et technologies pour identifier et localiser les véhicules dans les images. Les méthodes traditionnelles reposent souvent sur des techniques de traitement d'image et des algorithmes classiques, tandis que les approches modernes intègrent des modèles d'apprentissage automatique et des réseaux de neurones convolutifs (CNN). Ces technologies permettent une détection plus précise et rapide des véhicules, même dans des conditions variées. L'étude de ces solutions existantes révèle une diversité d'approches, allant des systèmes basés sur des règles simples aux modèles de deep learning sophistiqués.

1.4 Critique de l'Existant

Bien que les solutions actuelles offrent des fonctionnalités avancées, elles présentent également des limites notables. Certaines méthodes traditionnelles peuvent être limitées par leur capacité à gérer des variations importantes dans les conditions d'éclairage, les angles de vue et les types de véhicules. Les modèles plus modernes, bien que plus performants, peuvent nécessiter des ressources importantes pour l'entraînement et l'inférence. De plus, la plupart des solutions existantes ont encore des difficultés avec la détection dans des environnements complexes et les situations de forte densité de véhicules. Ces limitations soulignent la nécessité de solutions plus robustes et adaptées aux besoins spécifiques du secteur de l'assurance automobile.

1.5 Solution Proposée

Pour surmonter les limitations des solutions existantes, le projet propose le développement d'un modèle de détection de voitures basé sur YOLOv5. YOLOv5, une version avancée de la série YOLO, est particulièrement adaptée pour des tâches de détection d'objets en temps réel grâce à sa rapidité et à sa précision. Ce modèle sera spécifiquement ajusté aux besoins du système d'assurance automobile, en étant entraîné sur un ensemble de données riche et diversifié pour optimiser sa capacité à détecter les véhicules dans des conditions variées rencontrées dans le contexte des réclamations d'assurance.

Un aspect crucial du projet est le *fine-tuning* du modèle YOLOv5. Cette étape implique l'ajustement des hyperparamètres du modèle en fonction des caractéristiques spécifiques des données d'entraînement, afin de maximiser la précision et la robustesse des détections. Le *fine-*

tuning permet d'adapter le modèle de manière précise aux particularités du secteur de l'assurance automobile, en assurant une détection fiable même dans des situations complexes comme les collisions en chaîne ou les environnements urbains denses.

En personnalisant YOLOv5 pour le secteur de l'assurance, le projet vise à fournir une solution innovante et performante qui répond aux besoins critiques de ce domaine, notamment pour l'évaluation des réclamations et la détection de fraude. L'intégration de ce système dans les processus d'assurance permettra d'améliorer l'efficacité opérationnelle, de réduire les coûts liés aux fraudes et d'offrir une meilleure expérience utilisateur aux assurés.

1.6 Méthodologie de travail

La méthodologie suivante décrit les étapes clés pour le développement et la mise en œuvre du modèle de détection de voitures utilisant YOLOv5. Chaque étape est conçue pour garantir une approche rigoureuse et efficace, allant de la préparation des données à la création d'une API pour son intégration.

- **Acquisition et Préparation des Données**
  - **Collecte des Données :** Nous collecterons un ensemble de données varié contenant des images de véhicules dans différents contextes pour garantir la robustesse et la généralisation du modèle.
  - **Annotation des Images :** Les images seront annotées avec des boîtes englobantes pour permettre au modèle d'apprendre à identifier et localiser les véhicules.
  - **Prétraitement des Données :** Les images seront prétraitées par redimensionnement et normalisation, et divisées en ensembles d'entraînement, de validation, et de test.
- **Conception et Entraînement du Modèle**
  - **Configuration du Modèle :** Nous configurerons le modèle YOLOv5 en choisissant le modèle pré-entraîné approprié et en ajustant les paramètres pour la détection des véhicules.
  - **Entraînement du Modèle :** Le modèle sera entraîné sur les données annotées, avec surveillance des performances pour ajustements nécessaires.
  - **Évaluation du Modèle :** Le modèle sera évalué sur un ensemble de test pour mesurer sa précision et ses performances globales.
- **Optimisation et Ajustements**
  - **Affinement des Hyperparamètres :** Des ajustements des hyperparamètres seront effectués en fonction des résultats de l'évaluation pour améliorer les performances du

modèle.

- **Amélioration de la Généralisation :** Des techniques d'augmentation des données seront appliquées pour améliorer la capacité du modèle à généraliser sur de nouvelles données.

- **Développement de l'API**
  - **Conception et Implémentation de l'API :** Une API sera développée avec FastAPI pour permettre l'interaction avec le modèle, la soumission des images, et la réception des résultats.

Conclusion

Ce chapitre a détaillé le cadre général du projet, l'étude des solutions existantes, ainsi que la méthodologie employée pour le développement du modèle de détection de voitures avec YOLOv5 et les étapes clés de notre approche.

Le prochain chapitre sera dédié à l'étude technologique et à la capture des besoins. Nous analyserons les technologies pertinentes pour notre projet et définirons les exigences spécifiques nécessaires.

CHAPITRE 2

SPÉCIFICATION DES BESOINS ET TECHNOLOGIES UTILISÉES

Introduction

Ce chapitre présente les éléments essentiels du projet, notamment les besoins spécifiques auxquels il répond, la description du jeu de données utilisé, et les technologies mises en œuvre. L'objectif est de fournir une vue d'ensemble complète des bases sur lesquelles repose le développement de la solution.

2.1 Analyse des besoins

Cette section décrit les exigences techniques pour le système de détection de véhicules, en précisant les objectifs fonctionnels et les détails des détections nécessaires pour répondre aux besoins du projet.

2.1.1 Détection des Véhicules :

- **Détection des Véhicules :** Le système doit détecter et localiser toutes les voitures présentes dans une image, indépendamment de leur position ou orientation.
- **Précision des Détections :** Pour chaque voiture détectée, le modèle doit fournir une probabilité ou une confiance associée, indiquant la certitude de la détection. Les résultats seront exprimés sous forme de coordonnées de boîtes englobantes et d'une mesure de précision (comme la confiance de YOLO).

2.1.2 Détails des Détections :

En plus des exigences fonctionnelles, cette section définit les besoins non fonctionnels qui caractérisent les propriétés souhaitées du système, telles que :

- **Nombre de Voitures :** Le modèle doit déterminer le nombre total de voitures présentes dans chaque image.
- **Localisation :** Le système doit fournir les coordonnées précises des boîtes englobantes pour chaque voiture détectée, permettant ainsi une analyse détaillée de la répartition des véhicules dans l'image.

2.2 Jeu de Données

Le jeu de données utilisé pour le projet comprend des images de véhicules annotées, fournissant ainsi un ensemble complet pour entraîner et évaluer le modèle de détection. Ce jeu de données est crucial pour garantir que le modèle peut identifier et localiser les véhicules dans divers contextes.

2.2.1 Collecte et Annotation des Données

Cette section décrit comment les images ont été collectées et annotées pour préparer les données nécessaires au développement du modèle.

- **Source des Données :** Les images proviennent de diverses sources pour assurer une couverture représentative des scénarios typiques rencontrés dans l'assurance automobile.
- **Annotation avec Roboflow :** Les images sont annotées à l'aide de Roboflow, qui permet de créer des boîtes englobantes autour des véhicules et d'exporter les annotations dans le format compatible avec YOLO.

2.2.2 Prétraitement des Données

Cette section décrit les étapes de prétraitement appliquées aux images pour les préparer à l'entraînement du modèle.

- **Prétraitement avec Roboflow :** Roboflow a été utilisé pour redimensionner les images à 640x640 pixels avec un étirement des données. Les données de pixels ont été automatiquement orientées en supprimant les informations EXIF.
- **Augmentation des Données :** Pour enrichir le jeu de données, des techniques d'augmentation ont été appliquées, créant ainsi trois versions de chaque image source. Ces techniques augmentent la diversité des données d'entraînement et améliorent la capacité de généralisation du modèle.

2.3 Technologies Utilisées

Ce projet repose sur l'utilisation de plusieurs technologies clés, chacune apportant des fonctionnalités spécifiques pour le traitement des données, l'entraînement du modèle, et la création d'une interface API. Ces technologies sont essentielles pour le succès de la détection automatique des véhicules.

- **OpenCV :**

  OpenCV est une bibliothèque open-source dédiée au traitement d'images et de vidéos. Elle offre des outils pour réaliser diverses opérations telles que la transformation, le filtrage, et la détection d'objets. Dans ce projet, OpenCV est utilisé pour le prétraitement des images, ce qui inclut des tâches comme le redimensionnement et l'amélioration des images.

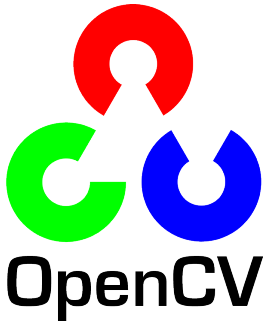

FIGURE 2.1 – Logo d'OpenCV

- **PyTorch :**

  PyTorch est une bibliothèque open-source pour le calcul scientifique et l'apprentissage profond, appréciée pour sa flexibilité et ses capacités de calcul dynamique. Elle est utilisée pour le développement et l'entraînement des modèles de machine learning. Dans ce projet, PyTorch permet la construction et l'optimisation du modèle de détection des véhicules.

FIGURE 2.2 – Logo de PyTorch

- **YOLOv5 :**

  YOLOv5 est une version améliorée de YOLO (You Only Look Once), un modèle de détection d'objets en temps réel réputé pour sa rapidité et sa précision. Il est employé dans ce projet pour la détection et la localisation des véhicules dans les images.

FIGURE 2.3 – Logo de YOLOv5

- **Python :**

  Python est un langage de programmation polyvalent largement utilisé pour le développement d'applications scientifiques, de machine learning et de traitement de données.

Sa simplicité et sa large gamme de bibliothèques en font un choix idéal pour ce projet, en particulier pour l'implémentation du modèle et la création de l'API.

FIGURE 2.4 – Logo de Python

- **FastAPI :**

  FastAPI est un framework moderne, rapide et performant pour la création d'APIs avec Python. Il facilite le développement d'interfaces web pour interagir avec le modèle de détection de véhicules, permettant ainsi une intégration fluide dans des systèmes de production.

FIGURE 2.5 – Logo de FastAPI

Conclusion

Ce chapitre a détaillé les besoins essentiels pour le projet, le jeu de données utilisé, et les technologies mises en œuvre. Nous avons abordé les exigences fonctionnelles du système, les spécificités des données collectées et annotées, ainsi que les outils technologiques employés.

Le prochain chapitre se concentrera sur l'architecture du système, en détaillant la structure et le fonctionnement de YOLOv5 ainsi que la conception et le flux de données de notre modèle de détection des véhicules.

CHAPITRE 3

ARCHITECTURE DU SYSTÈME

Introduction

Ce chapitre détaille le processus de conception et d'implémentation du système de détection de véhicules. Nous commencerons par une présentation de l'architecture de YOLOv5, le modèle de détection d'objets utilisé dans ce projet. Ensuite, nous aborderons l'architecture du système global, le flux de données, l'implémentation du modèle, et le développement de l'API.

3.1 Architecture de YOLOv5

Dans cette section, nous explorons la structure interne de YOLOv5, un modèle de détection d'objets en temps réel, en détaillant ses principaux composants et leur rôle dans le processus de détection.

3.1.1 Structure du Modèle

YOLOv5 (You Only Look Once, version 5) est un modèle de détection d'objets en temps réel. Il se compose de plusieurs composants clés :

- **Backbone :** Un réseau convolutionnel qui extrait des caractéristiques des images d'entrée.
- **Neck :** Des couches intermédiaires qui fusionnent les caractéristiques de différentes échelles pour renforcer les détails spatiaux.
- **Head :** Le composant final qui produit les prédictions, y compris les coordonnées des boîtes englobantes, les scores de confiance, et les classes prédictes.

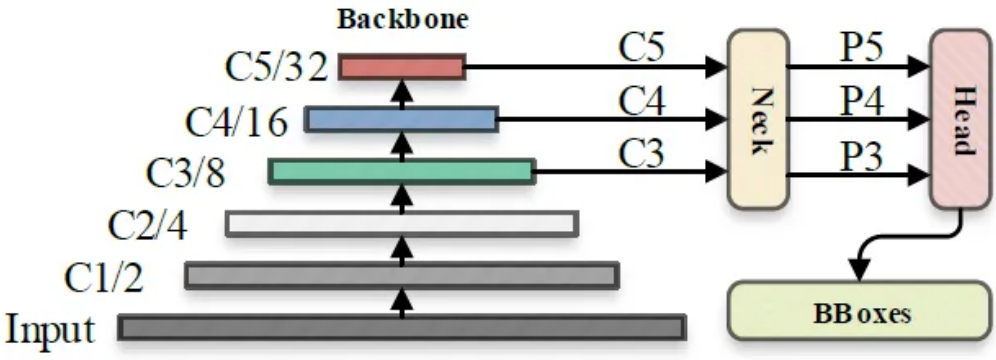

FIGURE 3.1 – Diagramme de l'architecture de YOLOv5.

[1]

3.1.2 Fonctionnement de YOLOv5

L'image est d'abord traitée par une couche d'entrée (*input*) puis envoyée au *backbone* pour l'extraction des caractéristiques. Le *backbone* obtient des cartes de caractéristiques de différentes tailles, qui sont ensuite fusionnées via le réseau de fusion de caractéristiques (*neck*) pour finalement générer trois cartes de caractéristiques : P3, P4, et P5 (dans YOLOv5, les dimensions sont respectivement de 80×80, 40×40 et 20×20). Ces cartes sont utilisées pour détecter des objets de petite, moyenne et grande tailles dans l'image.

Après que les trois cartes de caractéristiques ont été envoyées à la tête de prédiction (*head*), le calcul de la confiance et la régression des boîtes englobantes sont exécutés pour chaque pixel de la carte de caractéristiques en utilisant les ancres prédéfinies. Cela permet d'obtenir un tableau multidimensionnel (*BBoxes*) incluant la classe d'objet, la confiance de la classe, les coordonnées de la boîte, la largeur, et la hauteur.

En définissant les seuils correspondants (*confthreshold*, *objthreshold*), les informations inutiles dans le tableau sont filtrées, et un processus de suppression des non-maxima (NMS) est exécuté pour obtenir les informations finales de détection.[1]

3.2 Architecture du Système

Dans cette section, nous décrivons l'architecture globale du système de détection de véhicules. Le système utilise le modèle YOLOv5, entraîné sur un dataset spécifique aux voitures, pour la détection d'objets dans les image.

3.2.1 Conception du Modèle de Détection

Le système de détection de véhicules est constitué de plusieurs composants clés :

- **Prétraitement des Images :** Les images sont préparées pour le modèle en étant redimensionnées, normalisées et, si nécessaire, augmentées. Ce prétraitement assure que les images sont adaptées au format d'entrée attendu par YOLOv5 et améliore ainsi la performance du modèle.
- **Entraînement du Modèle YOLOv5 :** Le modèle YOLOv5 est entraîné sur un jeu de données spécifique, comprenant des images de voitures annotées. L'entraînement produit un fichier de poids, best.pt, qui intègre les connaissances acquises par le modèle pour la détection des véhicules. Cette étape inclut l'optimisation des hyperparamètres afin de maximiser la précision de la détection.

- **Utilisation du Modèle Entraîné :** Le fichier de poids best.pt est utilisé pour la phase de détection. Lors de l'inférence, les images sont traitées par le modèle pour identifier et localiser les véhicules présents.

3.2.2 Flux de Données

Le flux de données dans le système suit le parcours suivant :

- **Entrée des Images :** Les images, fournies par l'utilisateur ou provenant d'une source externe, sont introduites dans le système.
- **Prétraitement :** Les images sont soit redimensionnées, soit converties au format RGB pour améliorer la performance du modèle YOLOv5. Ce prétraitement assure que les images sont dans un format compatible avec le modèle.
- **Détection avec le Modèle Entraîné :** Le modèle YOLOv5, avec les poids 'best.pt' issus de l'entraînement, analyse les images pour détecter les véhicules. Cette étape produit des prédictions sous forme de boîtes englobantes et de scores de confiance pour chaque détection.
- **Post-traitement :** Les résultats de détection sont affinés en appliquant des seuils de confiance pour éliminer les détections peu fiables.
- **Sortie des Résultats :** Les résultats finaux, comprenant les coordonnées des boîtes englobantes et les scores de confiance, sont renvoyés à l'utilisateur ou à un système client pour une utilisation ultérieure.

La figure ci-dessous décrit l'architecture de notre système utilisant YOLOv5. Ce diagramme illustre les étapes du processus de détection, depuis l'acquisition des images jusqu'à la sortie des résultats. Il montre les différentes phases du traitement des données, y compris le prétraitement, l'analyse avec le modèle YOLOv5, et le post-traitement. :

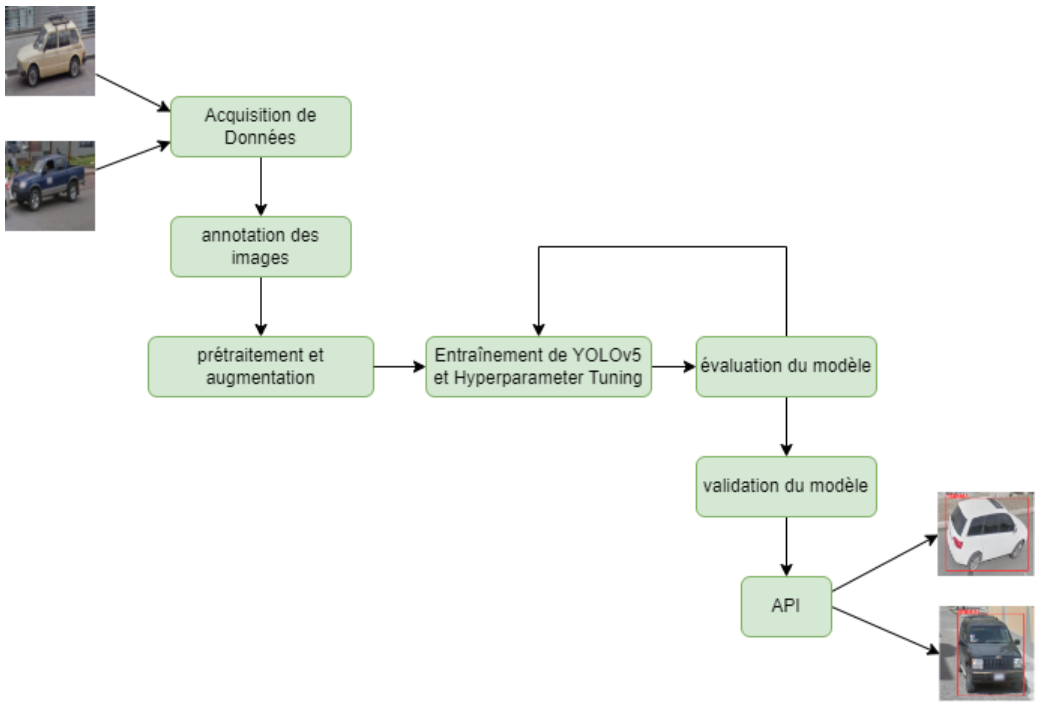

FIGURE 3.2 – Diagramme du Flux de Données.

Conclusion

Ce chapitre a présenté la conception du système de détection de véhicules basé sur YOLOv5, couvrant les étapes du prétraitement des images à la détection des véhicules. Nous avons détaillé l'architecture du modèle, le flux de données, et les composants principaux du système. Le prochain chapitre abordera la réalisation pratique du modèle, incluant l'entraînement, le tuning des hyperparamètres, et l'évaluation des performances.

CHAPITRE 4

RÉALISATION

Introduction

Ce chapitre présente les détails de la mise en œuvre et des méthodes utilisées pour la réalisation du système de détection de véhicules basé sur YOLOv5. Nous y décrivons les étapes pratiques suivies pour entraîner le modèle, effectuer le tuning des hyperparamètres, et évaluer les performances du modèle.

4.1 Réalisation du Modèle

Cette section aborde le processus de mise en œuvre du modèle YOLOv5, depuis l'entraînement initial jusqu'à l'optimisation des hyperparamètres.

4.1.1 Entraînement du Modèle

Le modèle YOLOv5 a été spécifiquement entraîné pour détecter des véhicules à l'aide d'un dataset personnalisé. Voici un aperçu détaillé de cette étape cruciale :

4.1.1.1 Préparation des Données

Pour optimiser la performance du modèle, les données ont été soigneusement préparées :

- **Redimensionnement :** Les images ont été redimensionnées à 640x640 pixels, ce qui a été réalisé en utilisant une méthode de redimensionnement par étirement pour correspondre aux spécifications d'entrée du modèle YOLOv5.
- **Augmentation des Données :** Pour améliorer la robustesse du modèle, chaque image source a été augmentée pour générer trois versions différentes. Cela a impliqué des modifications telles que des variations de luminosité, des rotations et des recadrages, afin de simuler diverses conditions de prise de vue et de renforcer la capacité du modèle à généraliser.

4.1.1.2 Configuration

Les paramètres et hyperparamètres du modèle YOLOv5 ont été soigneusement configurés pour s'adapter aux spécificités du projet :

- **Choix du Modèle :** La version `yolov5s` (small) a été sélectionnée pour son bon compromis entre la performance de détection et la vitesse de traitement, étant adaptée

pour un environnement de production efficace.

- **Fichier de Configuration YAML :** Le fichier `custom_yolov5s.yaml` a été modifié pour spécifier le nombre de classes et les noms des classes selon les besoins du projet. Dans ce cas, nous avons défini une seule classe, nommée `Car`.
- **Prétraitement :** Les images sont redimensionnées à une taille de 416x416, normalisées, et converties au format RGB pour garantir une compatibilité optimale avec le modèle YOLOv5.
- **Hyperparamètres :** Les hyperparamètres suivants ont été utilisés initialement pour entraîner le modèle YOLOv5 :
  - **Taux d'Apprentissage :** Le taux d'apprentissage détermine la vitesse à laquelle le modèle ajuste ses poids en réponse à l'erreur de prédiction. Un taux d'apprentissage initial de 0.01 a été utilisé.
  - **Nombre d'Époques :** Une époque est une passe complète sur l'ensemble du dataset d'entraînement. L'entraînement a été réalisé sur 25 époques.
  - **Taille des Mini-Batchs :** La taille des mini-batchs définit le nombre d'échantillons d'entraînement à utiliser avant de mettre à jour les poids du modèle. Les images ont été traitées en mini-batchs de 16.
  - **Taille des Images :** La taille des images détermine la résolution à laquelle les images sont présentées au modèle pendant l'entraînement. Les images ont été redimensionnées à une taille de 640x640 pixels.

4.1.1.3 Entraînement

Le processus d'entraînement a été structuré comme suit :

- **Initialisation :** Le modèle YOLOv5 a été initialisé avec des poids pré-entraînés pour bénéficier d'une bonne base pour le fine-tuning sur notre dataset spécifique.
- **Phase d'Entraînement :** Le modèle a été entraîné avec les données préparées, ajustant ses poids pour minimiser les erreurs de prédiction des boîtes englobantes, des scores de confiance, et des classes d'objets.
- **Suivi et Ajustement :** Les courbes de perte et de précision ont été surveillées en temps réel. Les ajustements nécessaires des hyperparamètres ont été effectués pour optimiser la performance du modèle.

4.1.2 Ajustement des Hyperparamètres

L'ajustement des hyperparamètres a été une étape clé pour affiner les performances du modèle. Après la sélection initiale, la méthode de Recherche Aléatoire (Random Search) a été utilisée pour explorer de manière non exhaustive un espace défini d'hyperparamètres. Cette méthode implique :

- **Définition des Paramètres :** Un ensemble de valeurs possibles pour chaque hyperparamètre a été déterminé. Par exemple, des valeurs différentes pour le taux d'apprentissage, le nombre d'époques, et la taille des mini-batchs ont été sélectionnées.
- **Sélection Aléatoire :** Des combinaisons d'hyperparamètres ont été sélectionnées de manière aléatoire parmi les valeurs définies, plutôt que de tester toutes les combinaisons possibles.
- **Évaluation des Configurations :** Chaque combinaison sélectionnée a été testée en entraînant le modèle avec cette configuration spécifique et en évaluant sa performance sur un ensemble de validation. Les performances ont été mesurées en termes de précision, de rappel et de perte.
- **Sélection des Meilleurs Paramètres :** Les configurations ayant donné les meilleurs résultats de validation ont été sélectionnées comme étant les plus prometteuses pour l'entraînement final du modèle.

La figure 4.1 montre le code utilisé pour l'ajustement des hyperparamètres. Ce code illustre le processus de recherche aléatoire pour optimiser les paramètres du modèle.

```python
import os
import random
import subprocess

# Plages de valeurs pour les hyperparamètres
img_sizes = [416, 512, 640]
batch_sizes = [16, 32]
learning_rates = [0.01, 0.001, 0.0001]
epochs_range = [25, 50, 30]

def train_yolo(img_size, batch_size, learning_rate, epochs):
    command = f"""
    python train.py --img {img_size} --batch {batch_size} --epochs {epochs} \
    --data /content/drive/MyDrive/stage/yolov5/Cars-detecting-and-how-many-1/data.yaml --cfg ./models/custom_yolov5s.yaml --weights 'yolov5s.pt' \
    --name yolov5s_results_{img_size}_{batch_size}_{learning_rate}_{epochs} --cache
    """
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    while True:
        output = process.stdout.readline()
        if output == b'' and process.poll() is not None:
            break
        if output:
            print(output.decode().strip())
    rc = process.poll()
    return rc

# Nombre d'essais aléatoires
n_trials = 5

for _ in range(n_trials):
    img_size = random.choice(img_sizes)
    batch_size = random.choice(batch_sizes)
    learning_rate = random.choice(learning_rates)
    epochs = random.choice(epochs_range)

    print(f"Training with img_size={img_size}, batch_size={batch_size}, learning_rate={learning_rate}, epochs={epochs}")
    train_yolo(img_size, batch_size, learning_rate, epochs)
```

FIGURE 4.1 – Code de l'ajustement des hyperparametres.

À la fin de cette phase, le modèle entraîné a produit un fichier de poids `best.pt`, capturant les meilleurs paramètres pour la détection des véhicules dans les images.

4.2 Évaluation du Modèle

Cette section décrit les méthodes utilisées pour évaluer le modèle YOLOv5, y compris les métriques employées pour mesurer la performance du modèle.

4.2.1 Méthodologie d'Évaluation

L'évaluation du modèle a été réalisée en suivant plusieurs étapes pour obtenir une mesure précise de sa performance et de sa robustesse :

- **Validation :** Pendant l'entraînement, un sous-ensemble des données a été réservé pour la validation. Cette approche permet de surveiller les performances du modèle en temps réel, de détecter le surajustement (overfitting) et d'ajuster les hyperparamètres en conséquence. La validation permet également de suivre la convergence du modèle et d'assurer qu'il généralise bien sur des données non vues.
- **Test :** Un ensemble de test indépendant a été utilisé pour évaluer les performances finales du modèle après l'entraînement. Cet ensemble de test est distinct de l'ensemble de validation et des données d'entraînement, fournissant ainsi une mesure objective et impartiale de l'efficacité du modèle dans des conditions réelles. Cette étape est cruciale pour vérifier que le modèle n'est pas seulement performant sur les données d'entraînement mais aussi sur de nouvelles données.

4.2.2 Métriques d'Évaluation

Les performances du modèle ont été évaluées en utilisant les métriques suivantes :

- **Précision (Precision) :** La précision mesure la proportion de détections correctes parmi toutes les détections effectuées par le modèle. Une haute précision indique que le modèle fait peu d'erreurs dans ses prédictions.
- **Rappel (Recall) :** Le rappel mesure la proportion de détections correctes parmi tous les objets présents dans les images. Un rappel élevé indique que le modèle est efficace pour détecter la majorité des objets présents.
- **mAP@0.5 (mean Average Precision at IoU threshold 0.5) :** La mAP@0.5 est la moyenne des précisions pour différents seuils de rappel, calculée spécifiquement pour un seuil

d'IoU de 0.5. Elle indique la précision globale du modèle à un seuil d'IoU fixé, ce qui permet de mesurer la performance du modèle en termes de précision et de couverture des objets détectés.

- **Object Loss :** La perte liée à la détection des objets, qui mesure la différence entre les prédictions du modèle et les véritables annotations des objets. Elle est utilisée pour évaluer la capacité du modèle à détecter correctement les objets présents dans les images.
- **Box Loss :** La perte liée aux boîtes englobantes, qui évalue l'erreur dans la localisation des boîtes englobantes prédite par rapport aux véritables boîtes. Une faible box loss indique que les boîtes englobantes prédites sont proches des véritables positions des objets.
- **Accuracy (Précision Globale) :** L'accuracy mesure la proportion totale de prédictions correctes parmi toutes les prédictions effectuées par le modèle.

4.3 Mise en Production

Cette section décrit le processus d'intégration du modèle YOLOv5 dans un système fonctionnel via le développement d'une API en FastAPI, facilitant ainsi son utilisation pratique.

4.3.1 Intégration du Modèle dans le Système

Le modèle YOLOv5 a été intégré dans une API développée avec FastAPI. Cette API offre une interface pour interagir avec le modèle et permet les opérations suivantes :

- **Réception des Images :** L'API accepte les images envoyées par les utilisateurs via des requêtes HTTP POST. Les images peuvent être téléchargées directement à travers cette interface.
- **Prédictions :** Lorsqu'une image est reçue, l'API la transmet au modèle YOLOv5 pour effectuer des prédictions. Le modèle analyse l'image pour détecter les véhicules et génère des prédictions sous forme de boîtes englobantes et de scores de confiance.
- **Retour des Résultats :** Les résultats des détections sont renvoyés aux utilisateurs sous format JSON. Les données retournées incluent les coordonnées des boîtes englobantes, les scores de confiance, et les classes détectées.

4.3.2 Développement de l'API avec FastAPI

Le développement de l'API avec FastAPI a suivi plusieurs étapes clés :

- **Configuration de l'Environnement :** Installation de FastAPI et des bibliothèques nécessaires pour la gestion des requêtes HTTP et le traitement des images. Le modèle YOLOv5 a également été intégré dans l'environnement pour l'inférence.
- **Création des Points de Terminaison :** Mise en place des points de terminaison API pour :
  — Recevoir les images via HTTP POST.
  — Effectuer des prédictions en utilisant le modèle YOLOv5.
  — Retourner les résultats sous format JSON.
- **Tests et Validation :** Réalisation de tests pour garantir le bon fonctionnement de l'API. Les tests incluent la validation de la réception des images, la vérification de la précision des prédictions, et la confirmation de la réponse appropriée avec les résultats.

la figure 4.2 illustre l'interface de FastAPI pour la méthode POST.

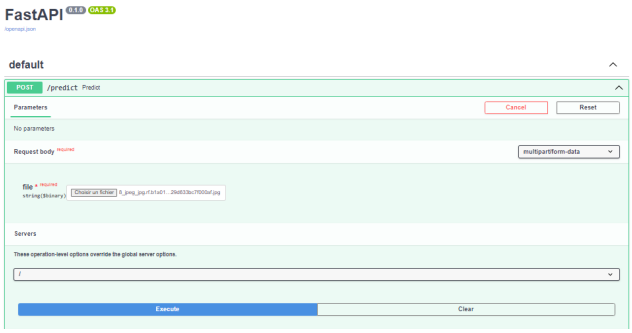

FIGURE 4.2 – Interface de l'api.

Conclusion

Ce chapitre a détaillé les processus de réalisation du modèle YOLOv5, y compris l'entraînement, le tuning des hyperparamètres, et l'évaluation des performances. Le chapitre suivant présentera les résultats obtenus, basés sur les métriques d'évaluation décrites dans ce chapitre.

CHAPITRE 5

RÉSULTATS ET ANALYSE

Introduction

Dans ce chapitre, nous présentons et analysons les résultats obtenus après l'entraînement et l'évaluation des différents modèles YOLOv5, en mettant l'accent sur les différentes configurations d'hyperparamètres testées. L'objectif est d'évaluer la précision, le rappel, et d'autres métriques de performance, afin de choisir le modèle le plus performant pour notre application de détection de véhicules. Des exemples de prédictions seront également fournis pour illustrer les performances du modèle.

5.1 Résultats des Tests

Cette section décrit les résultats des tests des différents modèles, basés sur les configurations d'hyperparamètres explorées.

5.1.1 Description des Données de Test

Le dataset de test, utilisé pour évaluer les performances des modèles, est constitué de 105 images soigneusement sélectionnées pour représenter une diversité de conditions de véhicules, permettant d'évaluer la capacité du modèle à détecter différents véhicules. Les images sont capturées dans divers environnements, ainsi que dans différentes conditions d'éclairage, telles que la lumière du jour, l'éclairage nocturne, et des conditions de faible luminosité. En outre, le dataset inclut des images montrant des véhicules en mouvement, stationnés, et partiellement visibles, afin de tester la robustesse du modèle dans des situations variées et complexes. L'objectif principal de ce dataset de test est d'évaluer la généralisation du modèle YOLOv5, c'est-à-dire sa capacité à effectuer des détections précises et fiables sur des images qu'il n'a pas vues pendant l'entraînement, assurant ainsi une évaluation représentative de la performance du modèle en conditions réelles.

5.1.2 Entraînement Initial

Lors de l'entraînement initial, le modèle YOLOv5 a été configuré avec des hyperparamètres de base pour établir une ligne de base des performances. Les valeurs spécifiques des hyperparamètres utilisés sont les suivantes :

- **Taux d'apprentissage** : 0.01
- **Nombre d'époques** : 25

- **Taille du batch** : 16
- **Taille des images** : 640

L'entraînement a été réalisé avec ces paramètres initiaux pour obtenir une première évaluation des performances du modèle. Les courbes suivantes montrent l'évolution des performances du modèle durant cette phase initiale :

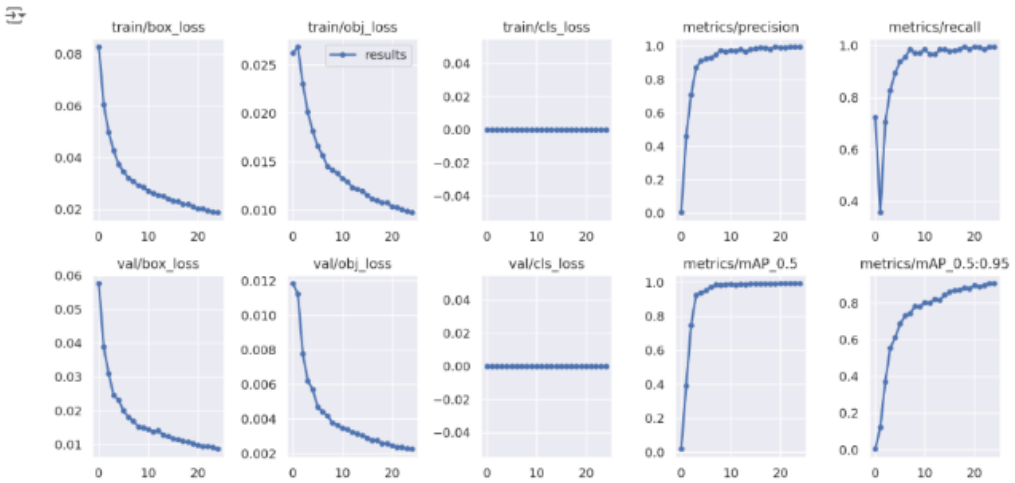

FIGURE 5.1 – Courbes d'evolution des performances pour le modèle initial.

Les courbes de perte montrent comment la perte d'entraînement et de validation évolue au fil des époques. On peut observer si le modèle commence à surapprendre ou si les performances de validation se stabilisent.

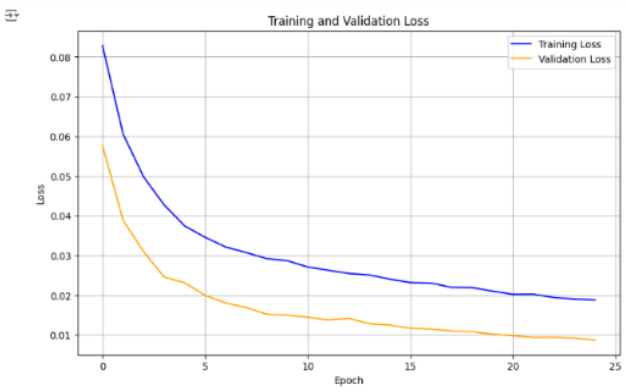

FIGURE 5.2 – Courbes de perte d'entraînement et de validation pour le modèle initial.

Les courbes de précision et de rappel montrent l'évolution de ces métriques au cours des époques. Elles permettent d'évaluer la capacité du modèle à équilibrer précision et rappel.

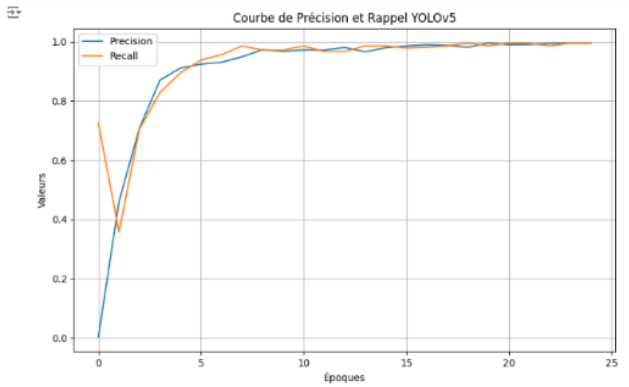

FIGURE 5.3 – Courbes de précision et de rappel pour le modèle initial.

La courbe d'accuracy illustre l'évolution de l'exactitude globale du modèle pendant l'entraînement.

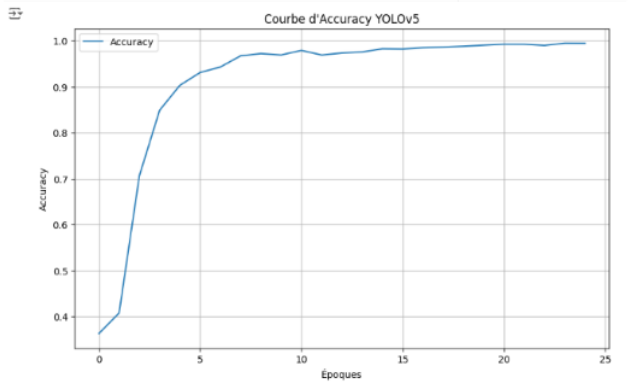

FIGURE 5.4 – Courbe d'accuracy pour le modèle initial.

La figure 5.5 montre le résultat de détection sur une image de la base de test :

FIGURE 5.5 – Résultats de détection sur données de test pour le Modèle initial.

Les résultats de l'entraînement initial de votre modèle YOLOv5 pour la détection de véhicules montrent des progrès prometteurs, notamment une réduction régulière des pertes

de localisation et de détection des objets, ainsi qu'une amélioration des métriques de précision et de rappel. Cependant l'entraînement initial a révélé des signes d'overfitting, comme le montre la divergence entre les courbes de perte d'entraînement et de validation. En synthèse, bien que les résultats initiaux soient encourageants, des ajustements ciblés sur les hyperparamètres et la régularisation sont nécessaires pour optimiser le modèle.

5.1.3 Comparaison des Hyperparamètres

L'ajustement des hyperparamètres a été réalisé en utilisant la méthode de recherche aléatoire (Random Search) en raison des ressources limitées. Les configurations testées et leurs résultats sont décrits ci-dessous :

5.1.3.1 Modèle 1

- **Image Size** : 416
- **Batch Size** : 16
- **Nombre d'époques** : 50

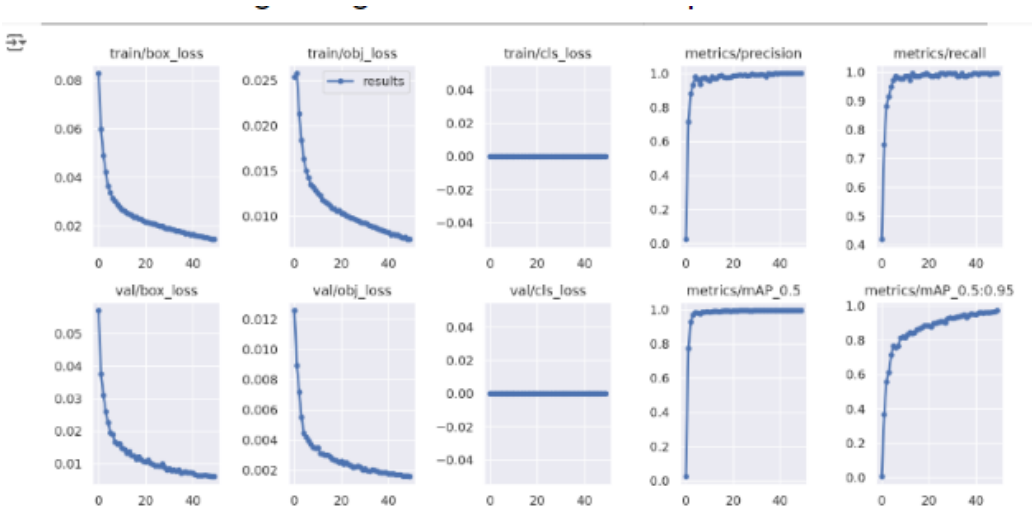

FIGURE 5.6 – Courbes d'evolution des performances pour le modèle 1.

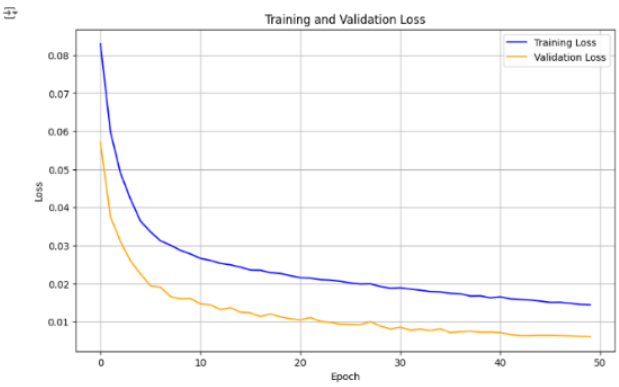

FIGURE 5.7 – Courbes de perte d'entraînement et de validation pour le modèle 1.

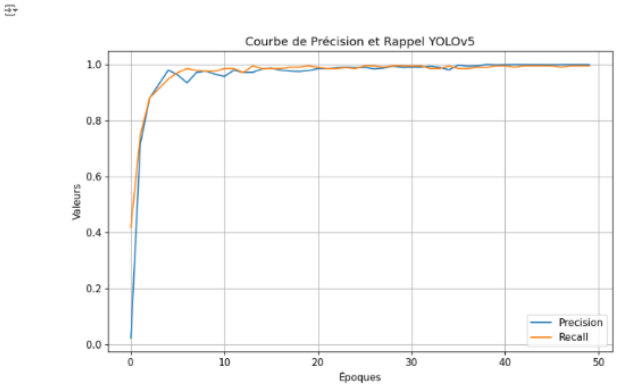

FIGURE 5.8 – Courbes de précision et de rappel pour le modèle 1.

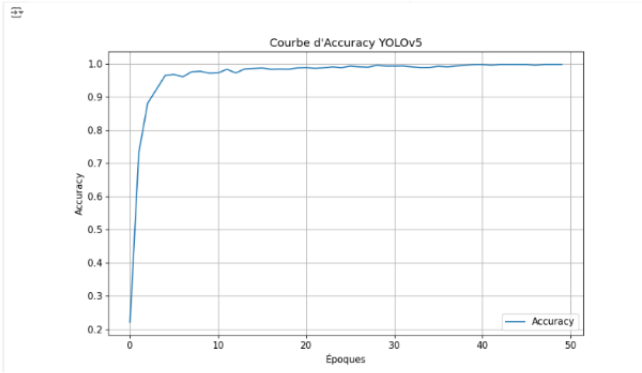

FIGURE 5.9 – Courbe d'accuracy pour le modèle 1.

FIGURE 5.10 – Résultats de détection sur données de test pour le modèle 1.

5.1.3.2 Modèle 2

- **Image Size** : 512
- **Batch Size** : 32
- **Nombre d'époques** : 30

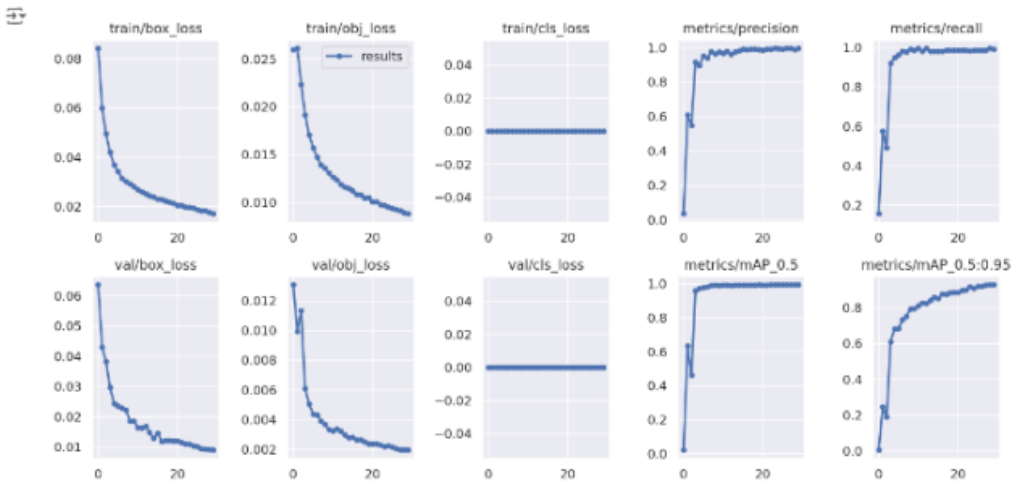

FIGURE 5.11 – Courbes d'evolution des performances pour le modèle 2.

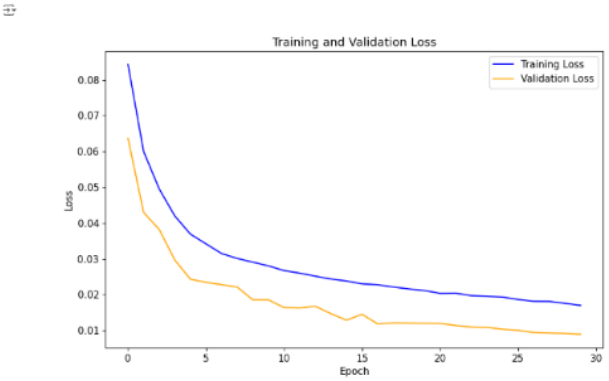

FIGURE 5.12 – Courbes de perte d'entraînement et de validation pour le modèle 2.

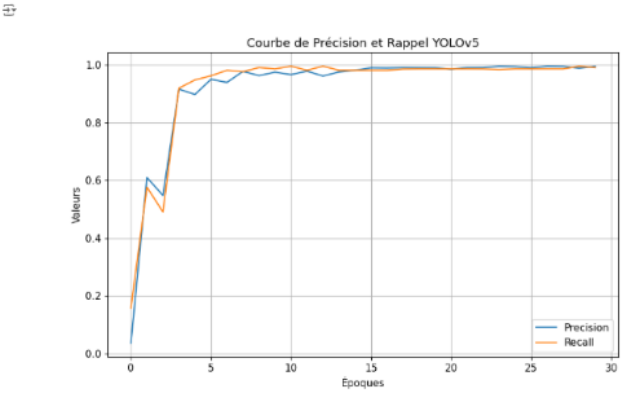

FIGURE 5.13 – Courbes de précision et de rappel pour le modèle 2.

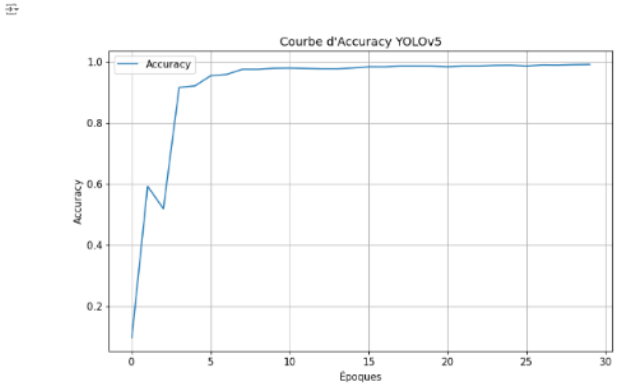

FIGURE 5.14 – Courbe d'accuracy pour le modèle 2.

FIGURE 5.15 – Résultats de détection sur données de test pour le modèle 2.

5.1.3.3 Modèle 3

- **Image Size** : 512
- **Batch Size** : 32
- **Nombre d'époques** : 50

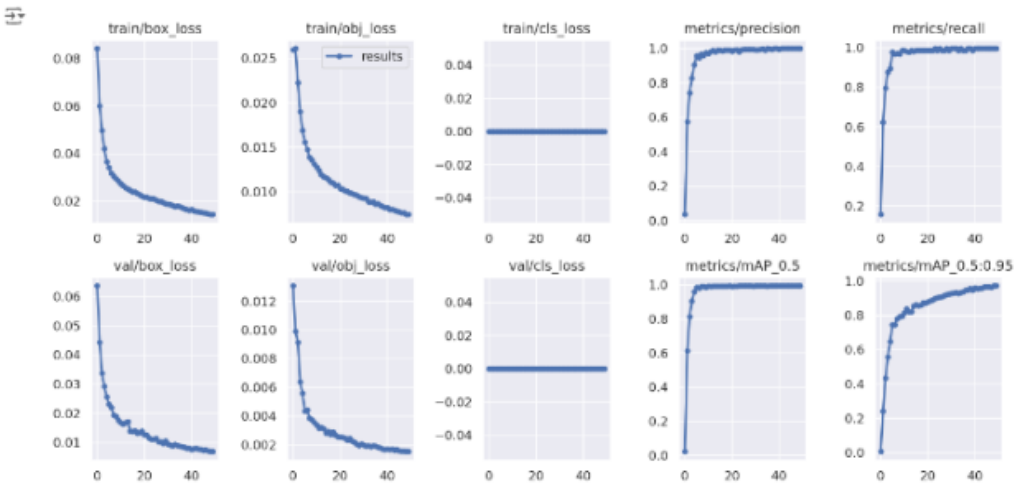

FIGURE 5.16 – Courbes d'evolution des performances pour le modèle 3.

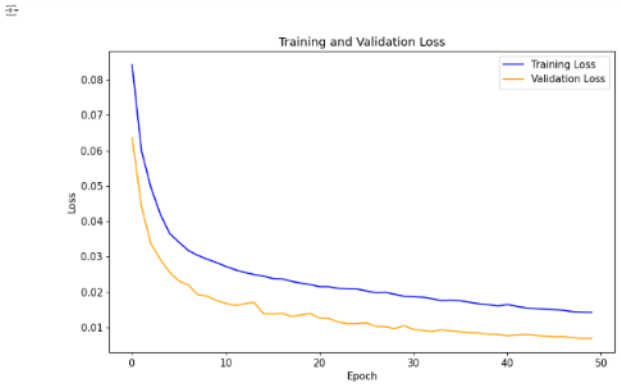

FIGURE 5.17 – Courbes de perte d'entraînement et de validation pour le modèle 3.

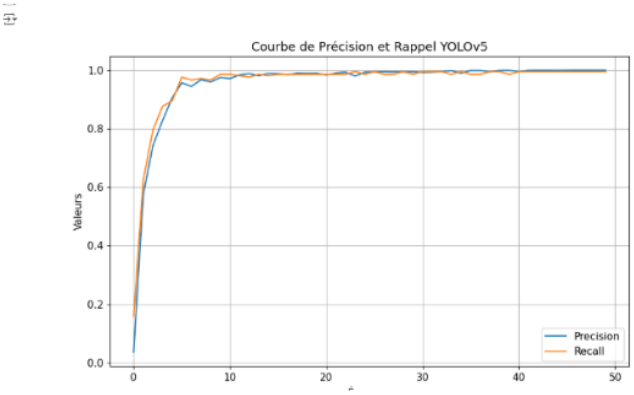

FIGURE 5.18 – Courbes de précision et de rappel pour le modèle 3.

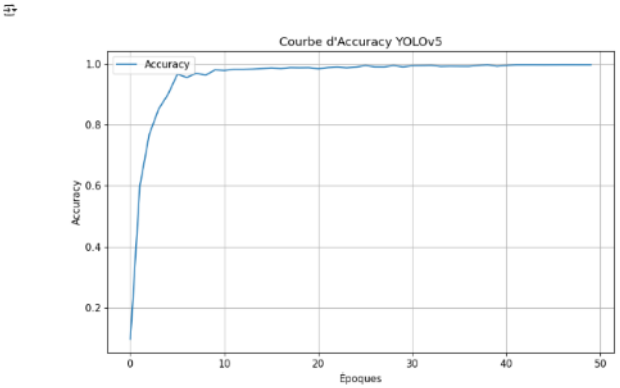

FIGURE 5.19 – Courbe d'accuracy pour le modèle 3.

FIGURE 5.20 – Résultats de détection sur données de test pour le modèle 3.

5.1.3.4 Modèle 4

- **Image Size** : 416
- **Batch Size** : 16
- **Nombre d'époques** : 25

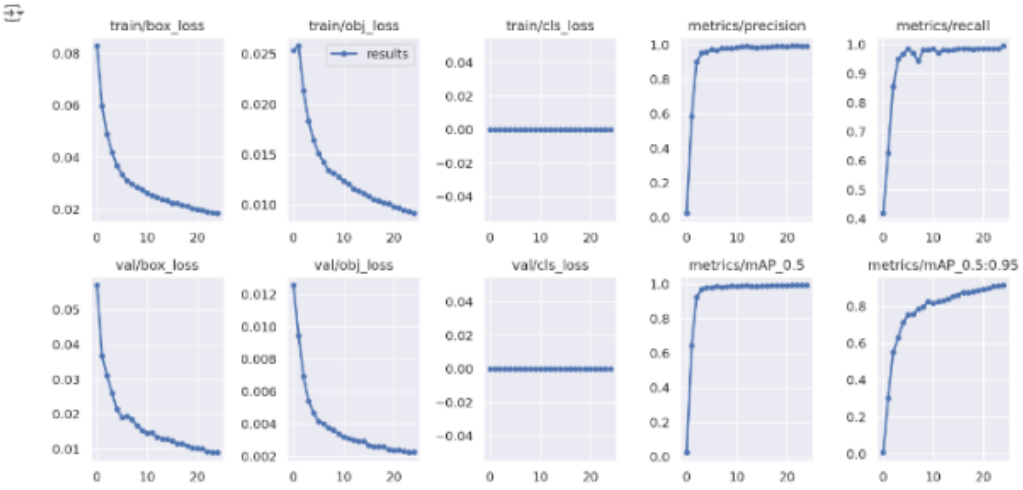

FIGURE 5.21 – Courbes d'evolution des performances pour le modèle 4.

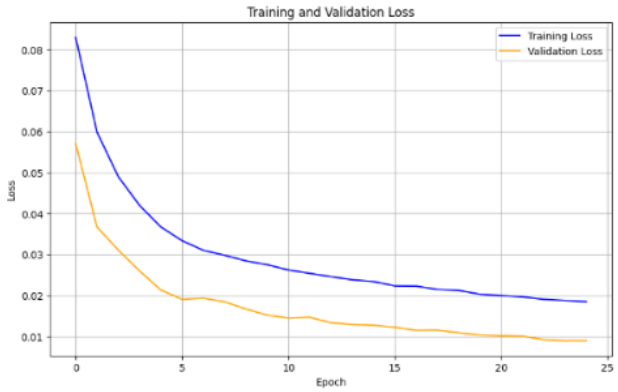

FIGURE 5.22 – Courbes de perte d'entraînement et de validation pour le modèle 4.

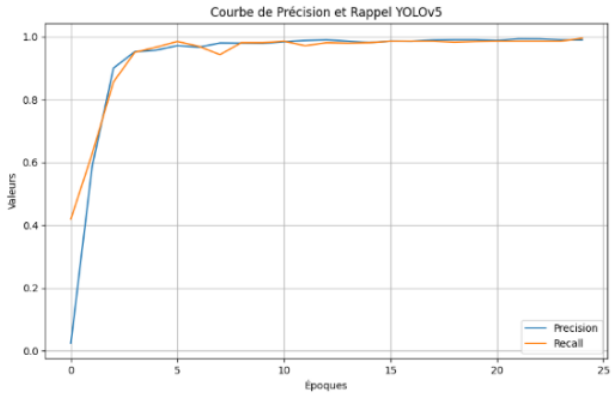

FIGURE 5.23 – Courbes de précision et de rappel pour le modèle 4.

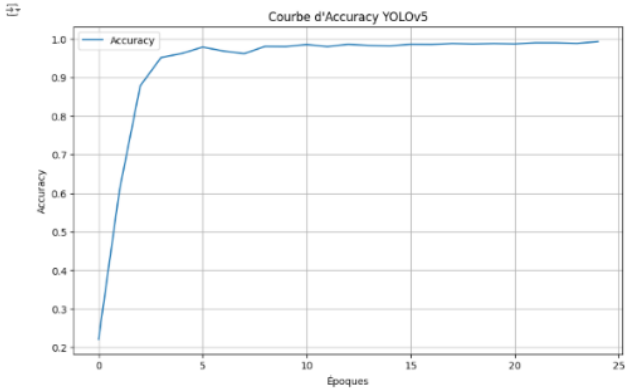

FIGURE 5.24 – Courbe d'accuracy pour le modèle 4.

FIGURE 5.25 – Résultats de détection sur données de test pour le modèle 4.

5.1.3.5 Modèle 5

- **Image Size** : 640
- **Batch Size** : 16
- **Nombre d'époques** : 30

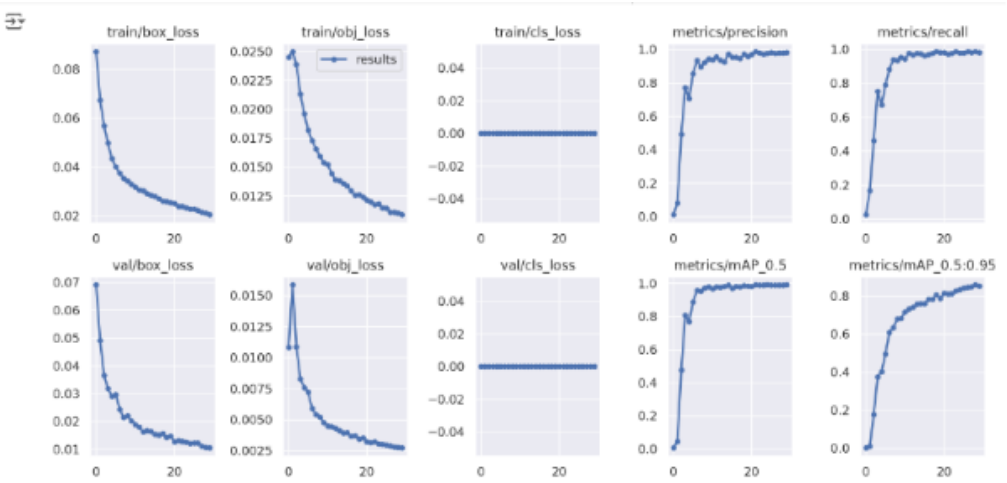

FIGURE 5.26 – Courbes d'evolution des performances pour le modèle 5.

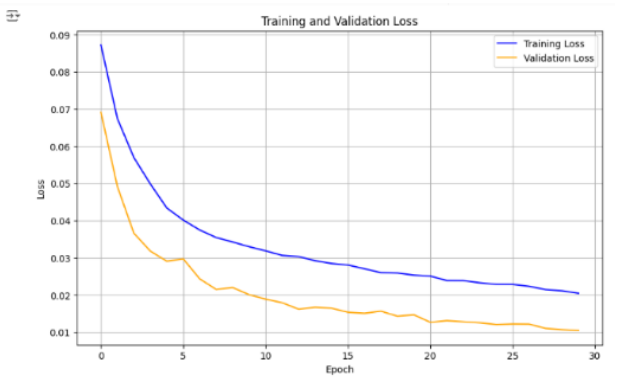

FIGURE 5.27 – Courbes de perte d'entraînement et de validation pour le modèle 5.

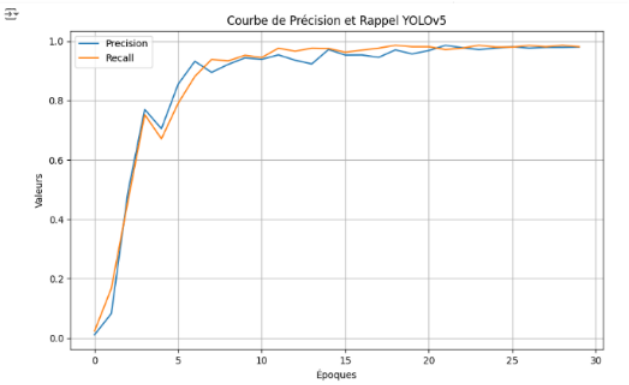

FIGURE 5.28 – Courbes de précision et de rappel pour le modèle 5.

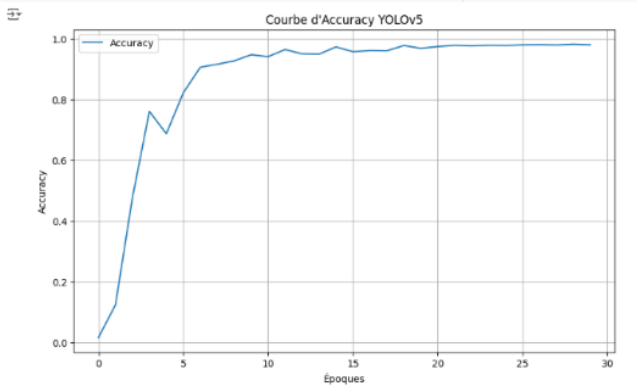

FIGURE 5.29 – Courbe d'accuracy pour le modèle 5.

FIGURE 5.30 – Résultats de détection sur données de test pour le modèle 5.

5.2 Choix du modèle

Le modèle final sélectionné est le modèle avec image size 416, batch size 16, learning rate 0.01, et 25 époques. Ce modèle a été choisi pour sa capacité à détecter efficacement les véhicules, même dans des conditions complexes ainsi qu'une meilleure allure des courbes de precision et rappel.

Le modèle sélectionné se distingue par plusieurs caractéristiques clés :

- **Précision élevée** : Le modèle a montré une capacité exceptionnelle à identifier correctement les voitures, avec un faible taux de fausses détections. Cela signifie que les boîtes englobantes générées autour des voitures détectées sont généralement précises et bien ajustées.
- **Rappel optimisé** : Le modèle a réussi à détecter une grande proportion des voitures présentes dans les images, même dans des scénarios difficiles où les véhicules étaient partiellement obscurcis ou en faible contraste. Cela indique une bonne capacité du modèle à identifier les objets pertinents même dans des conditions moins idéales.
- **Robustesse** : Le modèle a démontré une robustesse accrue face à des variations telles que la taille, l'orientation, et l'éclairage des véhicules. Cette capacité à maintenir des performances solides malgré les variations environnementales est cruciale pour des applications pratiques dans des contextes réels.

L'optimisation des hyperparamètres a été cruciale pour atteindre ces résultats. Les hyperparamètres ajustés, comme le taux d'apprentissage et la taille du lot, ont été sélectionnés après une évaluation minutieuse des performances sur des ensembles de validation. Cette sélection rigoureuse a permis au modèle de généraliser efficacement sur de nou-

velles données tout en évitant le sur-apprentissage, garantissant ainsi une bonne performance sur des images non vues pendant l'entraînement.

5.3 Visualisation des Résultats

Pour illustrer les performances du modèle en conditions réelles, nous présentons des exemples de détections effectuées via l'API. Les images ci-dessous montrent comment le modèle détecte les voitures et fournit des prédictions en termes de boîtes englobantes et de scores de confiance.

les figure 5.31 et 5.32 montrent les boîtes englobantes autour des véhicules détectés, ainsi que les scores de confiance associés à chaque détection et le log qui contient le nombre de voitures détectés.

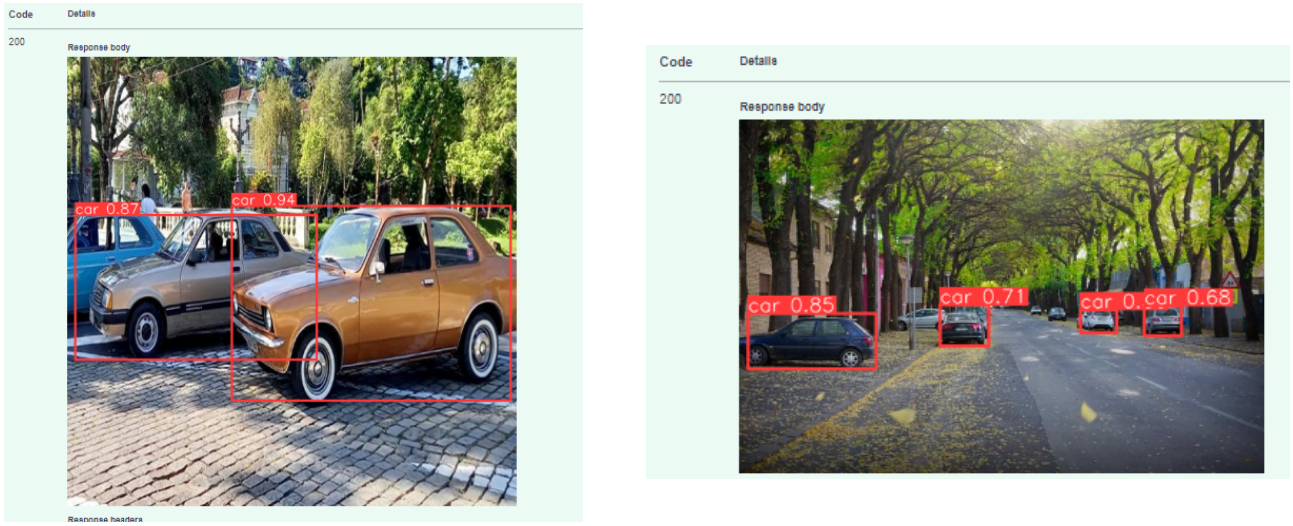

FIGURE 5.31 – Détections réalisées par le modèle via l'API.

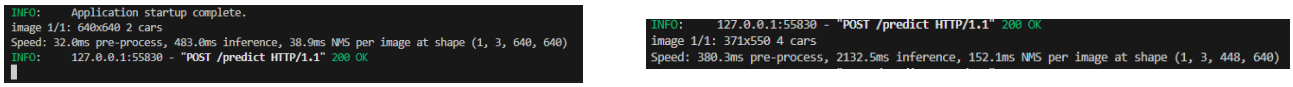

FIGURE 5.32 – Nombre de vehicules détectés.

Les résultats démontrent que le modèle est capable de localiser les voitures avec une grande précision, même dans des environnements variés et des situations d'éclairage différentes. Les scores de confiance associés à chaque détection fournissent une indication sur la certitude du modèle quant à la présence d'un véhicule dans chaque boîte englobante.

5.4 Limitations

Cette section aborde les limitations rencontrées durant le projet, en mettant l'accent sur les ressources GPU limitées et les défis liés à l'acquisition d'un dataset varié.

- **Ressources GPU Limitées** : Les ressources GPU disponibles étaient insuffisantes pour permettre l'entraînement de modèles très profonds ou pour utiliser des tailles de lot plus grandes. Cette contrainte a limité la capacité à tester des configurations plus complexes et a potentiellement restreint les performances du modèle. L'utilisation de GPU plus puissants pourrait permettre des expérimentations plus approfondies et une amélioration des performances globales du modèle.
- **Acquisition de Dataset Varié** : Le dataset utilisé pour l'entraînement était limité en termes de diversité. Le manque de variété dans les images, telles que les conditions d'éclairage, les types de véhicules, et les contextes de scène, a pu influencer la capacité du modèle à généraliser sur des données du monde réel. Enrichir le dataset avec une plus grande diversité pourrait améliorer la performance et la robustesse du modèle en permettant au modèle de mieux apprendre à partir de variations plus représentatives des conditions réelles.

Ces limitations ont eu un impact sur les résultats du modèle et offrent des opportunités pour des améliorations futures. L'augmentation des ressources de calcul et l'enrichissement du dataset sont des axes potentiels pour des développements ultérieurs. En surmontant ces limitations, il serait possible d'atteindre des performances encore meilleures et d'assurer une meilleure généralisation du modèle dans des contextes diversifiés.

Conclusion

Ce chapitre a présenté une analyse détaillée des performances du modèle, en mettant l'accent sur l'impact des ajustements des hyperparamètres et la sélection du modèle final. Les visualisations des résultats fournissent un aperçu pratique des capacités du modèle, tandis que les limitations discutées offrent des perspectives pour des améliorations futures. Les résultats et les analyses fournissent une base solide pour les recommandations et les applications pratiques du modèle dans des contextes réels.

CONCLUSION GÉNÉRALE

En conclusion, le projet de développement d'un système de détection de véhicules a réussi à atteindre ses objectifs principaux en proposant une solution innovante pour améliorer la gestion des sinistres automobiles. En utilisant des réseaux de neurones convolutifs et des techniques de deep learning, nous avons conçu un modèle capable de détecter les véhicules avec une grande précision, même dans des conditions difficiles telles que les environnements mal éclairés ou les véhicules partiellement obstrués.

Cependant, plusieurs limitations ont été identifiées au cours du projet. Les contraintes liées aux ressources GPU disponibles ont restreint la capacité à tester des configurations plus complexes, tandis que la diversité limitée du dataset a pu influencer la capacité du modèle à généraliser efficacement sur des données variées. Ces défis ont été des opportunités d'apprentissage importantes et ont permis d'affiner les approches et les méthodes utilisées.

Cette expérience a enrichi mes compétences techniques et m'a offert une perspective approfondie sur les défis liés à la gestion des données et des ressources. Pour l'avenir, il serait bénéfique d'améliorer les ressources de calcul et d'élargir le dataset afin de renforcer la robustesse et la précision du modèle. En somme, le projet a jeté des bases solides pour une solution avancée dans le domaine de l'assurance automobile et ouvre la voie à des développements futurs pour optimiser la détection des fraudes et améliorer la gestion des sinistres.

BIBLIOGRAPHIE

[1] Sik-Ho Tsang. Yolov5. <https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a/>, 2023. [En ligne; accés 17-Juillet-2024].