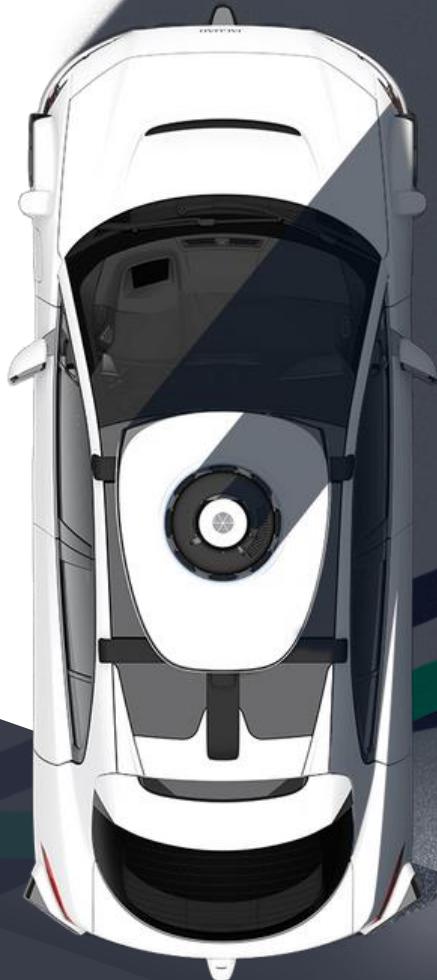


RAPPORT DE PROJET: **Détection D'Object**



Réalisé par :
Anouar Bouzhr
Aya Salim
Hamza Kholti
Ahlam El Gamani
Aya El Haoudar

Proposé par :
HACHCHANE Imane

Dédicaces

Nous dédions humblement ce travail à toutes les personnes qui occupent une place spéciale dans nos cœurs : nos parents, nos amis, et tous ceux qui ont été à nos côtés dans les moments de doute et de triomphe. À notre École, l'ENSA, qui a constamment été notre guide, nous ouvrant ses portes, nous encourageant et nous soutenant tout au long de ces années d'apprentissage. Enfin, notre dédicace particulière va à ceux qui ont joué un rôle significatif dans l'élaboration et le succès de ce modeste travail. Leur contribution a été inestimable, et c'est à eux que nous attribuons une part de cette réussite. Que cette dédicace soit l'expression de notre profonde gratitude envers ceux qui ont enrichi notre parcours académique et personnel.

Remerciement

Nous tenons tout d'abord à exprimer notre profonde gratitude à notre professeur, **HACHCHANE Imane**, qui nous a non seulement donné l'opportunité de réaliser ce passionnant projet de détection d'objets, mais nous a également accompagnés avec dévouement et expertise tout au long de notre période de réalisation. Son soutien constant et ses précieux conseils ont été des éléments clés de notre réussite.

Nous souhaitons également remercier chaleureusement notre équipe de projet, composée de : **Anouar BOUZHAR, Ahlam ELGAMANI, Hamza KHOLTI, Aya ELHAOUDAR et Aya SALIM**, pour avoir partagé cette aventure enrichissante. La coopération, l'entraide et la passion commune pour les défis technologiques nous ont permis de surmonter les obstacles et d'adapter nos méthodes de travail aux nouvelles exigences de notre projet. Notre engagement et notre esprit d'équipe ont été essentiels tout au long de cette expérience.

Nos familles méritent une mention spéciale pour leur patience et leur soutien indéfectible. Nous leur sommes infiniment reconnaissants pour leur compréhension, surtout lors des retours tardifs à la maison, et pour avoir cru en nous à chaque étape du projet.

Enfin, nous tenons à exprimer notre gratitude envers **l'École Nationale des Sciences Appliquées de Tétouan**, qui a mis à notre disposition les ressources nécessaires et un environnement propice à la recherche et à l'innovation.

Table des Matières

Contenu

| | |
|---|----|
| Liste des figure : | 6 |
| 1 Catalogue complet des données brutes : | 8 |
| 2 Prétraitement des données : | 10 |
| 2.1 Phase1: Renommage des classes et traitement des ensembles données | 10 |
| 2.1.1 Véhicule : | 10 |
| 2.1.2 Animaux : | 10 |
| 2.1.3 Moteur : | 12 |
| 2.1.4 Les panneaux de signalisation :..... | 12 |
| 2.1.5 Humain :..... | 13 |
| 2.2 Phase2 : Amélioration des Annotations d'Objets Non-Détectés à l'Aide de YOLO : | 13 |
| 2.3 Phase 3 : Centralisation des Ensembles de Données d'Objets en un Seul Dossier..... | 14 |
| 3 Les modèles d'entraînements : | 15 |
| 3.1 YOLO (You Only Look Once)..... | 15 |
| 3.1.1 Qu'est-ce que YOLO? | 15 |
| 3.1.2 Architecture originale de YOLO : | 15 |
| 3.1.3 Comment fonctionne YOLO : | 16 |
| 3.1.4 Le Transfert Learning | 17 |
| 3.2 Résultats de la première modèle d'entraînement : | 19 |
| 3.3 Les réseaux de neurones convolutif : | 20 |
| 3.3.1 Présentation Générale de CNN : | 20 |
| 3.3.2 L'utilisation de Model CNN Dans la Détection D'objets :..... | 21 |
| 3.3.3 Résultats et contraintes de l'entraînement de CNN : | 22 |
| 4 Les Mesures De Performances Pour L'évaluation Des Modèles. | 23 |
| 4.1 Introduction :..... | 23 |
| 4.2 Des Concepts Fondamentaux..... | 23 |
| 4.3 Métriques de détection d'objets | 24 |
| 4.4 Analyse Approfondie Des Métriques Pour Le Modèle YOLO.8 | 25 |
| 4.5 Interprétation des résultats | 26 |
| 4.5.1 Matrice De Confusion : | 26 |
| 4.5.2 Matrice De Confusion Normalisée : | 27 |
| 4.5.3 Les Métriques et les Pertes du Modèle YOLO :..... | 28 |
| 4.5.4 Courbe Précision-confiance PCC : | 31 |
| 4.5.5 Courbe Précision-Rappel PRC : | 32 |
| 4.5.6 Courbe confiance -Rappel RCC :..... | 33 |
| 4.5.7 Courbe F1 score-confidence : | 34 |
| 4.5.8 Analyse des annotations : | 35 |
| 4.5.9 Le corrélogramme : | 36 |
| 5 Le déploiement de notre projet dans Application web : | 39 |
| 6 Les contraintes rencontrées lors de la réalisation de ce projet : | 40 |

| | | |
|-----|--|----|
| 6.1 | Les contraintes liées à la phase de collection des données : | 40 |
| 6.2 | Les contraintes matérielles : | 40 |
| 7 | <i>Conclusion :</i> | 41 |
| 8 | <i>Annexes :</i> | 42 |

Liste des figure :

| | |
|---|----|
| Figure 1: data brute classe human | 9 |
| Figure 2: Data brut classe animaux | 10 |
| Figure 3: Annotation normalisée classe animaux | 11 |
| Figure 4: Annotation classe motor | 12 |
| Figure 5: image et annotation de classe de panneaux de signes | 13 |
| Figure 6 : Ameliorations des annotations | 14 |
| Figure 7 : Architecture originale de YOLO..... | 16 |
| Figure 8 : transfer learning | |
| Figure 9 : <i>Couche de convolution (CONV)</i> | 20 |
| Figure 10 : fonction d'activation RELU..... | 21 |
| Figure 11 : Fonctionnement d'un réseau neuronal à 2 couches cachées | 21 |
| Figure 12:Architecture de deuxième modèle d'entraînement. | 22 |
| Figure 13: résultat de deuxième modèle | 23 |
| Figure 14 : matrice de confusion | 26 |
| Figure 15 : <i>Matrice De Confusion Normalisée</i> | 28 |
| Figure 16: Les Métriques et les Pertes du Modèle YOLO | 28 |
| Figure 17: courbe Précision-confiance..... | 31 |
| Figure 18: Courbe precision-rappel..... | 32 |
| Figure 19: Courbe confiance -Rappel RCC..... | 33 |
| Figure 20: Courbe F1 score-confidence | 34 |
| Figure 21 : les courbes de l'annotation model YOLO | 35 |
| Figure 22 : Corrélogramme | |

Introduction : “H4A Driving”

« Dans les méandres du code et des pixels, là où la réalité se mêle à l'abstraction, notre voyage commence. »

Bienvenue dans le monde de “ **H4A Driving** ”, un projet qui transcende les limites de la vision artificielle et nous plonge dans un océan de pixels, de modèles et d’algorithmes. Imaginez-vous debout au sommet d’une montagne virtuelle, scrutant l’horizon numérique, prêt à explorer les secrets cachés dans chaque image.

Notre aventure a commencé par une simple question : **comment les machines voient-elles le monde ?** La **curiosité** nous a poussés à plonger tête la première dans le domaine du **computer vision**. Nous voulions comprendre comment les objets prennent forme dans l’univers des pixels, comment les contours se dessinent, comment les mouvements s’animent.

“**H4A Driving** ” tire son nom des membres de notre équipe: "H" pour Hamza et "4A" pour les quatre membres dont les noms commencent par "A" (Aya, Aya, Ahlam et Anouar). Ensemble, nous sommes une équipe passionnée par la création d'une technologie de détection d'objets pour la conduite autonome. Notre vision va au-delà de la simple détection d'objets, car nous aspirons à créer une technologie qui puisse être utilisée dans le domaine de la conduite autonome. Notre ambition est de rendre les routes plus sûres en développant une solution innovante et fiable. En combinant nos compétences et notre détermination, nous visons à repousser les limites de ce domaine en constante évolution.

«**H4A Driving** » est conçu pour identifier et classer précisément plusieurs types d'objets qui sont cruciaux pour la navigation et la sécurité des véhicules autonomes. Nos classes cibles comprennent les panneaux de signalisation, les véhicules, les motocyclettes, les humains, ainsi que les animaux urbains et ruraux. Chacune de ces catégories présente des défis uniques en termes de variabilité, de visibilité et de comportement, rendant la tâche de détection complexe et fascinante.

Nos armes secrètes? **YOLO (You Only Look Once)** et **CNN (Convolutional Neural Networks)**. Ces modèles sont reconnus pour leur efficacité et leur rapidité dans la détection d'objets en temps réel, ce qui en fait des choix parfaits pour l'application dans des systèmes embarqués sur des véhicules en mouvement.

La mise en œuvre de tels modèles nécessite une grande quantité de données précisément annotées. À cette fin, nous avons collecté environ **10 300 images** provenant de diverses sources fiables telles que **Kaggle**, **Roboflow** et **Open Images**. Chaque image a été soigneusement étiquetée pour refléter avec exactitude la présence et la classification des objets détectés. Ce processus rigoureux garantit que notre modèle apprend de manière efficace et minimise les erreurs potentielles en situation réelle.

Ainsi, en nous lançant dans ce projet ambitieux, nous espérons non seulement enrichir notre compréhension et notre expertise dans un champ technologique de pointe mais aussi apporter une contribution significative à la société en améliorant la sécurité et l'efficacité des technologies de conduite autonome. «**H4A Driving** » est plus qu'un projet ; c'est une promesse d'avenir, un pas vers un monde où la technologie et la sécurité se rencontrent pour créer des solutions durables et innovantes.

1 Catalogue complet des données brutes :

La phase de collecte de données pour notre projet de détection d'objets a été méthodiquement organisée afin d'obtenir un ensemble diversifié et représentatif de différentes catégories d'objets à détecter. Nous avons rassemblé cinq ensembles de données distincts, chacun ciblant une classe spécifique d'objets, provenant de sources variées pour garantir la représentativité et la généralisation de notre modèle de détection d'objets.

Chaque ensemble de données contient des images étiquetées avec l'objet correspondant à détecter, avec une attention particulière portée à la qualité et à la précision des étiquettes. Voici un aperçu des cinq ensembles de données que nous avons collectés :

- **Véhicules** : L'ensemble de données sur les véhicules comprend une variété d'images de voitures provenant de différentes marques, modèles et types. Ces images ont été collectées à partir de diverses sources, notamment des caméras de surveillance, des enregistrements vidéo et des bases de données publiques de véhicules. La source de cette collection provient de deux sites principaux : Roboflow, qui fournit 3496 images de voitures, et Kaggle, qui contribue avec 1254 images comprenant des catégories telles que 'Ambulance', 'Bus', 'Car', 'Motorcycle' et 'Truck'. Ces deux ensembles ont ensuite été combinés pour créer un ensemble de données plus diversifié et représentatif des différents types de véhicules.
- **Panneaux de signalisation routière** : La collection de panneaux de signalisation routière comprend des images de panneaux provenant de diverses régions géographiques, capturées à l'aide de caméras de circulation, de smartphones et d'autres sources provenant du site Kaggle. Cet ensemble de données a été capturé dans le pays Jordanie contient 10566 panneaux répartis dans 9246 images. Ce dataset comporte 7 classes, à savoir : "No Entry", "Hump", "Stop", "Pedestrian Cross", "No Stop", "Give Way", "Pass Either". Ces images ont été collectées à l'aide de la fonction Street View de Google Maps dans les rues de la capitale jordanienne, Amman. la fonction Street View remonte à 2017 mais offre une qualité élevée.
- **Humains**
Description : L'ensemble de données Humains contient 13652 images représentant des personnes dans une variété de contextes et de poses. Ces images ont été recueillies à partir de bases de données d'images publiques, de vidéos surveillées et d'autres sources accessibles au public. Chaque image est accompagnée d'étiquettes (labels) décrivant la présence et la position des personnes dans l'image. Ces annotations permettent d'identifier et de localiser les personnes dans les différentes scènes capturées. L'ensemble de données est accessible sur le site de ressources Roboflow.



Figure 1: data brute classe human

➤ Animaux

Le dataset contient une compilation d'images d'animaux provenant d'une variété d'espèces, de tailles et d'habitats différents. Ces images ont été collectées à partir de zoos, de réserves naturelles, de sites web spécialisés dans les animaux, et d'autres sources. Issu du site Kaggle, le dataset occupe une taille de 10 gigaoctets. Chaque espèce animale est associée à une classe distincte dans le dataset, représentée sous forme de chaîne de caractères dans les labels des images. Il comprend 21 types d'animaux tels que les chats, les chèvres, les moutons, les poulpes, les souris, les ânes, etc.

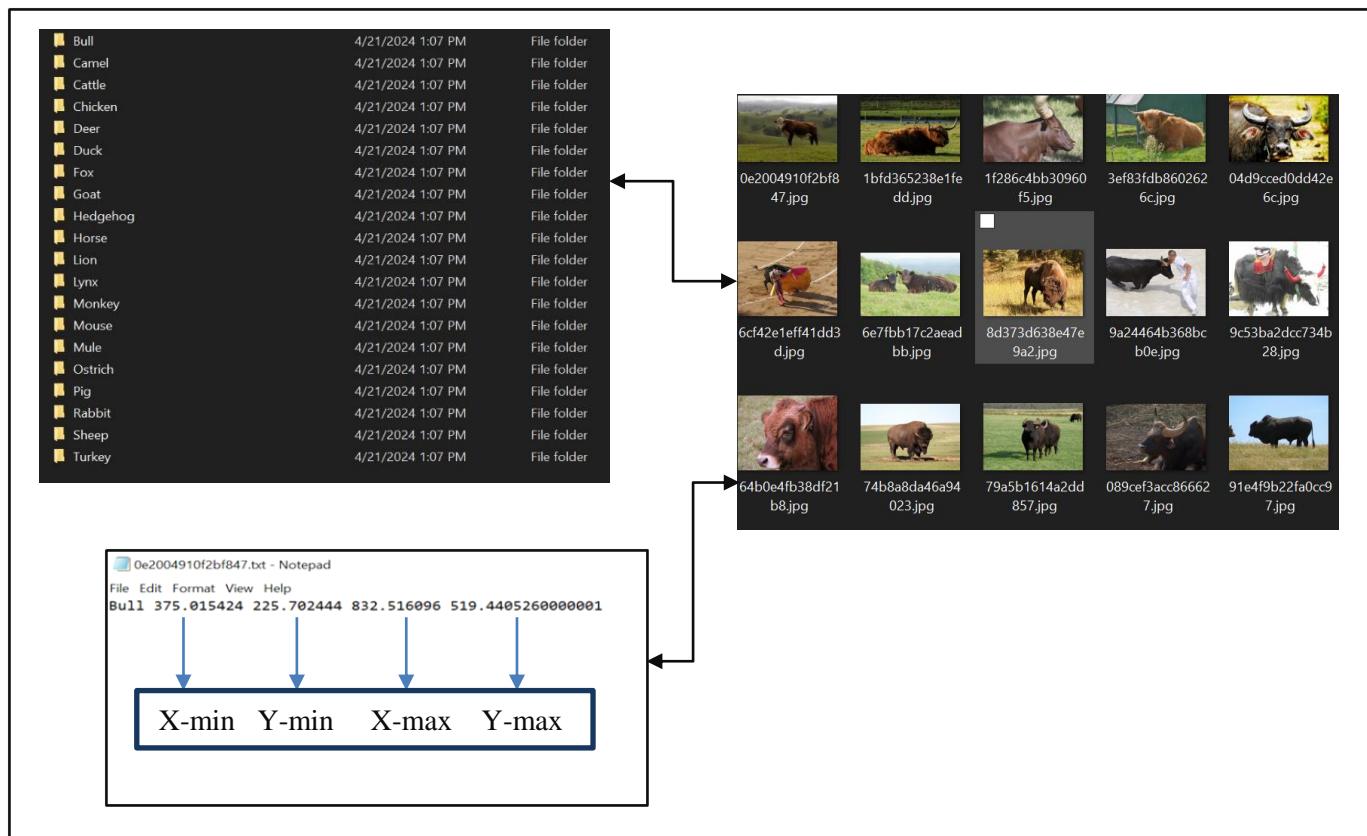


Figure 2: Data brut classe animaux

- **Motos :** Cet ensemble de données se concentre sur les motocyclettes, avec des images capturées dans des environnements urbains, suburbains et ruraux, provenant de vidéos, de photos de rue et d'autres sources similaires.

Chaque ensemble de données a été soigneusement vérifié pour garantir la diversité des exemples et la qualité des annotations, assurant ainsi la robustesse et la fiabilité de notre modèle de détection d'objets.

Aperçu

2 Prétraitement des données :

2.1 Phase1: Renommage des classes et traitement des ensembles données

Après avoir collecté les données brutes, nous avons entrepris de standardiser les noms des classes selon l'annotation que nous avions sélectionnée. Il arrive parfois que les annotations dans les fichiers texte des images diffèrent de celles que nous avions choisies au préalable. De plus, plusieurs classes peuvent être assignées à différents objets, mais nous avons opté pour les regrouper en une seule classe. Notre choix d'annotation est le suivant : 0 pour les voitures, 1 pour les humains, 2 pour les motocyclettes, 3 pour les panneaux de signalisation et 4 pour les animaux.

2.1.1 Véhicule :

Dans le jeu de données sur les voitures, nous avons distingué deux ensembles de données distincts. Le premier ensemble ne comporte qu'une seule classe, que nous avons renombrée pour être représentée par le numéro 0. Quant au deuxième ensemble de données, il inclut plusieurs catégories telles que 'Ambulance', 'Bus', 'Voiture', 'Moto' et 'Camion'. Afin de simplifier notre analyse, nous avons choisi de regrouper l'ensemble de ces catégories en une seule classe représentant les véhicules, à laquelle nous avons également attribué le numéro 0. Enfin, nous avons fusionné les deux ensembles de données en un seul jeu de données unifié.

2.1.2 Animaux :

Au départ, notre dataset brut était incroyablement diversifié, regroupant une multitude d'animaux de diverses catégories, incluant des espèces sauvages, des animaux forestiers, ainsi que des animaux marins et terrestres de différents habitats. Cependant, pour notre projet de détection d'objets, nous avons jugé nécessaire de restreindre notre sélection aux animaux susceptibles de se trouver sur les routes urbaines ou dans des environnements ruraux.

Ainsi, après une analyse minutieuse, nous avons retenu une liste d'animaux pertinents comprenant le bœuf, le chameau, le bétail, le poulet, le cerf, le canard, le renard, la chèvre, le hérisson, le cheval, le lion, le lynx, le singe, la souris, la mule, l'autruche, le cochon, le lapin, le mouton et la dinde.

La dataset brute était déjà divisée en deux parties distinctes : les données d'entraînement et les données de test, ce qui nous a permis de procéder directement au prétraitement des données pour chacune de ces sous-ensembles. Dans chaque partie, les données étaient organisées par classe d'animal, avec un dossier dédié pour les images et un autre pour les labels correspondants.

Pour effectuer le nettoyage et la manipulation de ces données, nous avons développé un script en

Python. Ce script a été conçu pour rassembler tous les sous-dossiers contenant les différentes catégories d'animaux dans un unique dossier regroupant les images de tous les animaux, et un autre dossier pour leurs labels associés

En ce qui concerne les labels des animaux, ceux-ci étaient initialement représentés sous forme de noms de classe (chaînes de caractères) correspondant au type d'animal. Nous avons donc pris la décision de donner la même classe à tous les animaux, simplifiant ainsi la représentation des objets détectés dans l'étape suivante. Tous les animaux ont ainsi reçu la classe 4.

Lorsque nous avons examiné les annotations des labels pour les données d'animaux, nous avons constaté que les coordonnées des boîtes englobantes n'étaient pas normalisées comme celles des autres classes. Afin d'assurer la cohérence des données et de faciliter le traitement ultérieur, nous avons décidé de normaliser ces coordonnées en utilisant les formules spécifiques suivantes :

$$X_{center} = (x_{min} + x_{max}) / (2 * w)$$

$$Y_{center} = (y_{min} + y_{max}) / (2 * h)$$

$$Width = (x_{max} - x_{min}) / w$$

$$Height = (y_{max} - y_{min}) / h$$

En normalisant les coordonnées de cette manière, nous avons pu rendre les annotations des animaux cohérentes avec celles des autres classes, ce qui facilite le traitement et l'entraînement de notre modèle. Cette normalisation contribue également à améliorer la précision et la fiabilité de notre système d'annotation et de détection d'objets.

Après avoir effectué ces transformations sur les données d'entraînement et de test, nous avons construit un ensemble de validation à partir de l'ensemble d'entraînement et de l'ensemble de test . Enfin, pour réduire la taille globale de la dataset tout en préservant sa représentativité, nous avons légèrement réduit sa taille, tout en veillant à maintenir un équilibre entre les différentes classes d'animaux.

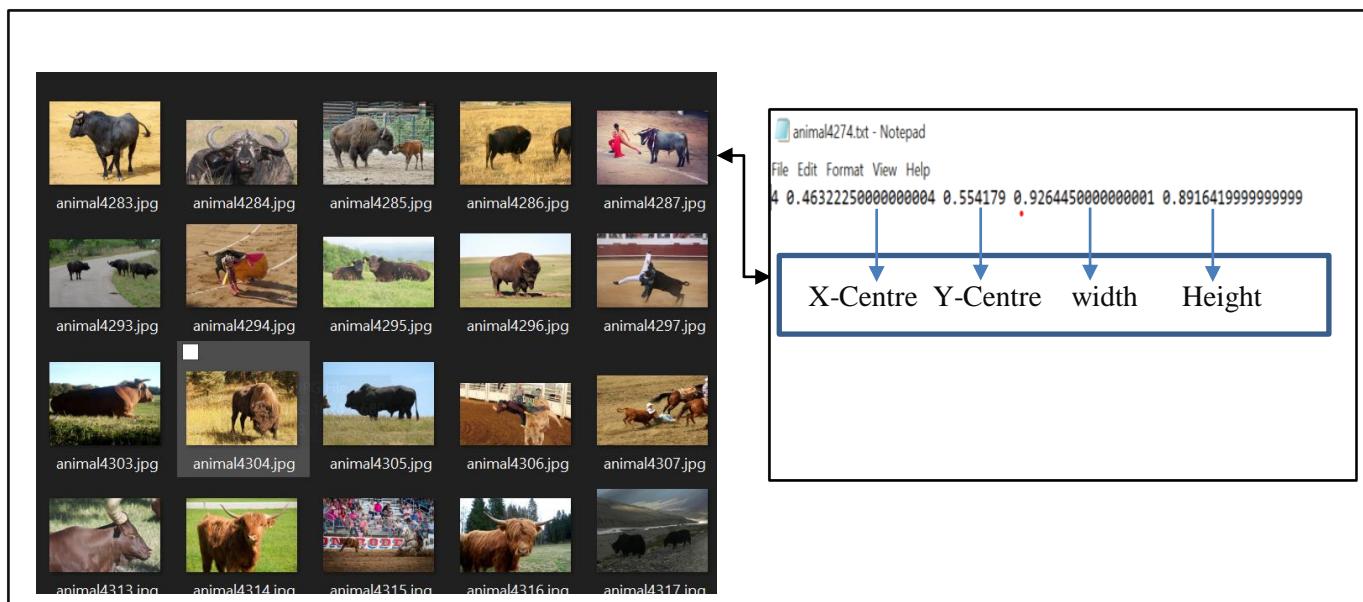


Figure 3: Annotation normalisée classe animaux

2.1.3 Moteur :

Dans le jeu de données sur les motos que nous avons examiné, nous avons identifié deux classes distinctes : "moto avec casque" et "moto sans casque". Pour simplifier notre analyse, nous avons choisi de regrouper ces deux classes en une seule classe représentant les motos, attribuée avec le numéro 2.

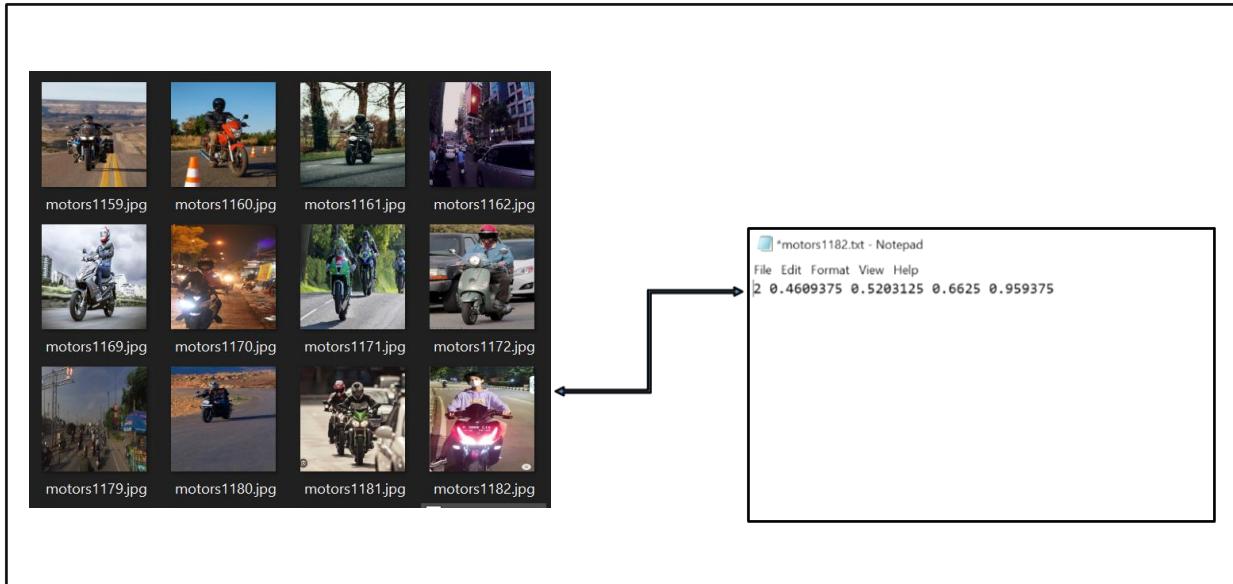


Figure 4: Annotation classe motor

2.1.4 Les panneaux de signalisation :

Dans le jeu de données sur les panneaux de signalisation, nous avons répertorié plusieurs classes distinctes, comprenant des panneaux tels que "Interdiction d'entrée", "Bosse", "Arrêt", "Passage piéton", "Pas d'arrêt", "Cédez le passage" et "Passage des deux côtés". Afin de maintenir la cohérence avec notre démarche, nous avons décidé de regrouper l'ensemble de ces classes en une seule catégorie représentant les panneaux de signalisation, désignée par le numéro 3.



Image 1952 contient de 4 panneaux.
Chaque panneaux design un objet qui est représenté par une ligne dans les étiquettes.

Figure 5: image et annotation de classe de panneaux de signes

2.1.5 Humain :

Dans le jeu de données sur les humains, nous avons suivi le même processus que celui appliqué dans les autres ensembles de données. La seule différence réside dans le fait que nous avons ajusté les étiquettes de classe pour correspondre à la catégorie des humains, et nous les avons attribuées avec le numéro 1.

Pour ce faire, nous avons développé un script Python qui parcourt les dossiers contenant les images et les annotations, utilisant une boucle pour renommer et compter les fichiers. Dans ce script, nous avons inclus des instructions pour mettre à jour les noms des classes dans chaque fichier texte dans le dossier des annotations. Cela garantit que les annotations correspondent fidèlement aux classes que nous avons définies.

En résumé, notre processus vise à harmoniser les annotations avec nos classes sélectionnées en modifiant les noms des fichiers dans le dossier des annotations. Cette approche assure une organisation cohérente des données et nous permet de préparer efficacement notre ensemble de données pour les étapes ultérieures de notre projet.

2.2 Phase2 : Amélioration des Annotations d'Objets Non-Détectés à l'Aide de YOLO :

Après avoir finalisé la mise à jour des noms de classes conformément à nos annotations sélectionnées, nous avons constaté la présence d'objets non annotés dans les images de chaque classe. Cette lacune était due à la disposition de dossiers distincts pour chaque classe, contenant à la fois les annotations et les images spécifiques à cette classe. Il est essentiel de veiller à inclure toutes les annotations nécessaires pour chaque image afin d'éviter toute omission pouvant affecter les performances de notre modèle.

Pour remédier à cette situation, nous avons opté pour l'utilisation d'un modèle pré-entraîné YOLO, configuré spécifiquement pour détecter les classes que nous avions sélectionnées parmi celles prises en charge par YOLO. Parmi les classes détectables par YOLO, nous avons choisi de conserver uniquement celles qui ont l'annotation dans YOLO. Par exemple, pour les panneaux de signalisation, nous avons conservé uniquement deux types (feu de circulation, panneau stop) parmi les nombreux disponibles dans les annotations précédentes. De même, pour la catégorie des animaux, nous avons sélectionné neuf types spécifiques (chat, chien, cheval, mouton, vache, éléphant, ours, zèbre, girafe), et pour les véhicules, nous nous sommes concentrés sur trois principaux types (bus, camion, voiture). Nous avons également ajouté deux sous-classes pour les motos, qui sont vélo et moto.

Afin de compléter nos données et de garantir une détection optimale des objets dans nos images, nous avons développé un script Python capable de parcourir chaque image et d'utiliser YOLO pour identifier tous les objets présents. Ensuite, le script a été conçu pour ajouter uniquement les classes précédemment non détectées, avec leurs coordonnées correspondantes, au fichier texte associé à chaque image.

Cette approche vise à améliorer la qualité de nos données en ajoutant des annotations pour les objets précédemment non détectés. Nous anticipons ainsi une amélioration des performances de notre modèle lors de l'entraînement et de l'inférence.

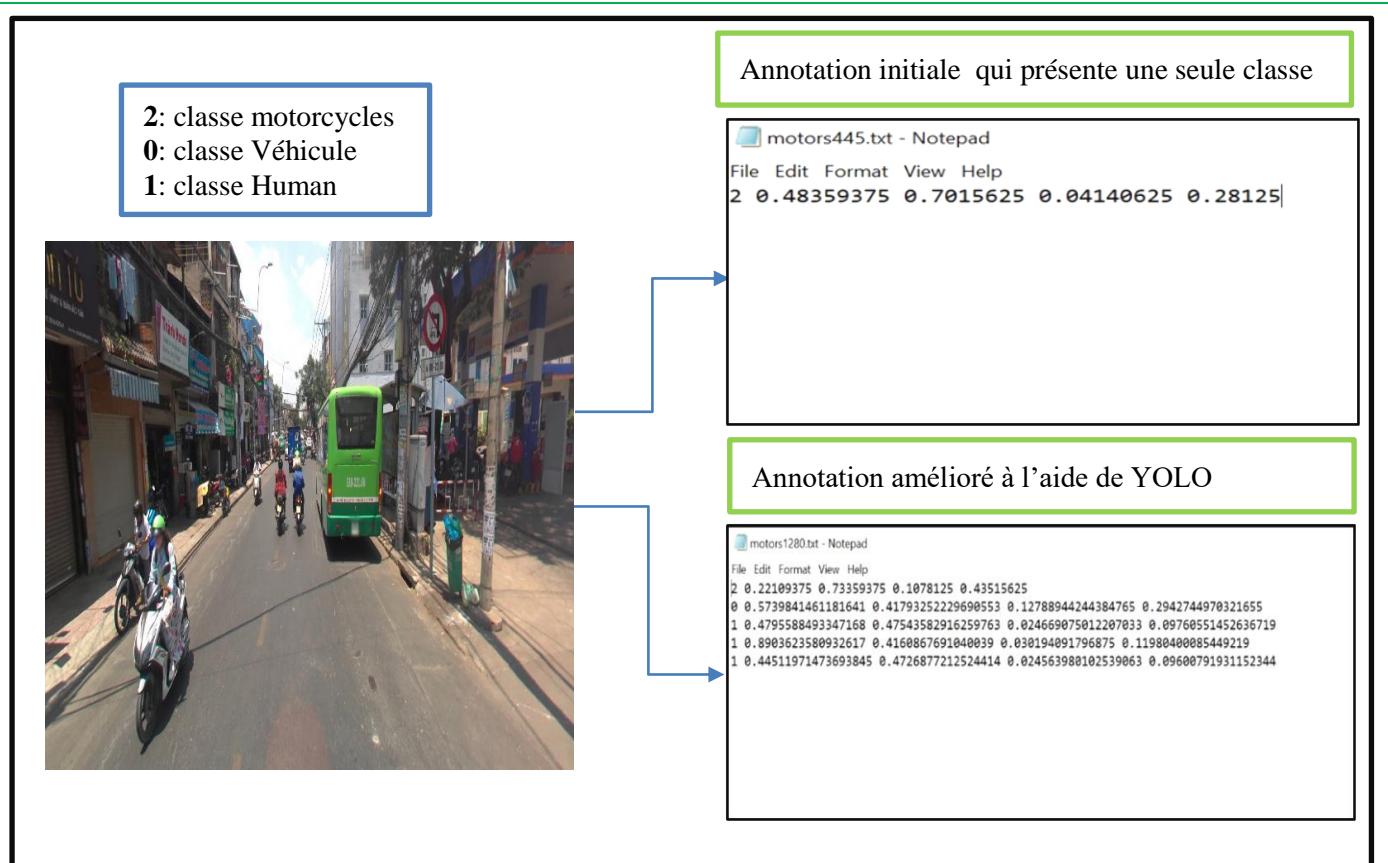


Figure 6 : Améliorations des annotations

2.3 Phase 3 : Centralisation des Ensembles de Données d'Objets en un Seul Dossier

Après avoir terminé le traitement de nos données, nous sommes arrivés à la phase de regroupement des données dans un seul dossier. Initialement, nos données étaient réparties dans des dossiers distincts pour chaque classe, comprenant des sous-dossiers pour la validation, l'entraînement et les tests. Chacun de ces sous-dossiers contenait à son tour deux autres dossiers, l'un pour les labels et l'autre pour les images.

Pour réaliser cette tâche, nous avons développé un script Python capable de parcourir chaque classe et chaque type de données (entraînement, validation, test). Ce script rassemble les images et les fichiers de labels de chaque classe dans un seul dossier correspondant à son type de données (entraînement, validation, test).

En regroupant ainsi toutes les données dans des dossiers uniques, nous simplifions la gestion et l'accès aux données pour les étapes ultérieures de notre projet, telles que l'entraînement et l'évaluation de notre modèle. Ce processus permet également d'assurer une cohérence dans la structure et l'organisation de nos données, ce qui facilite leur manipulation et leur utilisation.

3 Les modèles d'entrainements :

3.1 YOLO (You Only Look Once)

3.1.1 Qu'est-ce que YOLO?

YOLO, qui est une abréviation de You Only Look Once, est un algorithme de détection d'objets basé sur un réseau neuronal profond. Il a été inventé en 2015 par [Joseph Redmon](#), [Santosh Divvala](#), [Ross Girshick](#) et [Ali Farhadi](#). Contrairement à ses prédecesseurs tels que les R-CNN (Region-based Convolutional Neural Networks) qui détectent des objets en parcourant l'image en différentes étapes, YOLO quant à lui ne regarde l'image qu'une seule fois (d'où son nom) ce qui accélère de manière conséquente tout le processus de détection et de localisation d'un objet sur cette même image. Les R-CNN étant composés de plusieurs éléments affectés à une tâche dédiée sont ainsi plus difficiles à optimiser car chacun de ses composants doit alors être entraîné séparément ce qui peut rendre la tâche fastidieuse. Dans le cas de YOLO, toutes les opérations s'effectuent au sein d'un seul et même Réseau de Neurones.

[YOLO a été développé en plusieurs versions](#), telles que YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8 et YOLOv9. Chaque version a été construite au-dessus de la version précédente avec des fonctionnalités améliorées telles qu'une précision améliorée, un traitement plus rapide et une meilleure manipulation des petits objets.

Pour notre projet, nous avons choisi d'utiliser YOLOv8 d'Ultralytics, un modèle de pointe dans le domaine de la détection d'objets.

- Ultralytics YOLOv8 est un modèle de pointe (SOTA) qui s'appuie sur le succès des versions précédentes de YOLO et introduit de nouvelles fonctionnalités et améliorations pour améliorer encore les performances et la flexibilité. YOLOv8 est conçu pour être rapide, précis et facile à utiliser, ce qui en fait un excellent choix pour un large éventail de tâches de détection et de suivi d'objets, de segmentation d'instances, de classification d'images et d'estimation de pose.
- Techniquement, YOLOv8 est un groupe de modèles de réseaux neuronaux convolutifs, créés et entraînés à l'aide du framework PyTorch.

3.1.2 Architecture originale de YOLO :

L'architecture YOLO est similaire à [GoogleNet](#). Comme illustré ci-dessous, il comporte au total 24 couches convolutives, quatre couches de poolage max et deux couches entièrement connectées.

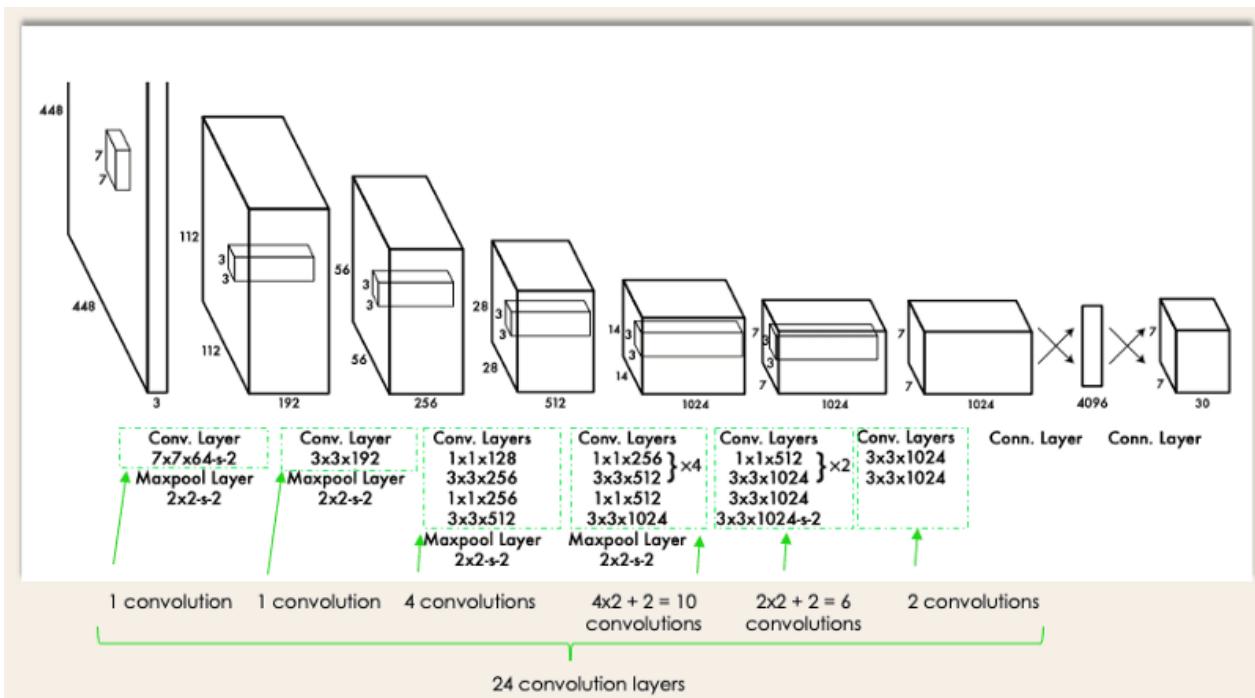


Figure 7 : Architecture originale de YOLO

L'architecture fonctionne comme suit:

- Redimensionne l'image d'entrée en 448x448 avant de passer par le réseau convolutif.
- Une convolution 1x1 est d'abord appliquée pour réduire le nombre de canaux, qui est ensuite suivi d'une convolution 3x3 pour générer une sortie cubique.
- La fonction d'activation sous le capot est ReLU, à l'exception de la dernière couche, qui utilise une fonction d'activation linéaire.
- Certaines techniques supplémentaires, telles que la normalisation des lots et l'abandon scolaire, régularisent respectivement le modèle et l'empêchent de s'immobiliser

3.1.3 Comment fonctionne YOLO :

L'idée de base derrière YOLO est de diviser l'image d'entrée en une grille de cellules et, pour chaque cellule, de prédire la probabilité de la présence d'un objet et des coordonnées de la zone de délimitation de l'objet. Le processus de YOLO peut être décomposé en plusieurs étapes:

1. L'image d'entrée est passée à travers un CNN pour extraire des caractéristiques de l'image.
2. Les caractéristiques sont ensuite passées à travers une série de couches entièrement connectées, qui prédisent les probabilités de classe et les coordonnées de la zone de délimitation.
3. L'image est divisée en une grille de cellules, et chaque cellule est responsable de la prédiction d'un ensemble de boîtes de délimitation et de probabilités de classe.

4. La sortie du réseau est un ensemble de zones de délimitation et de probabilités de classe pour chaque cellule.
5. Les cases de délimitation sont ensuite filtrées à l'aide d'un algorithme de post-traitement appelé suppression non max pour supprimer les cases se chevauchant et choisir la boîte avec la plus forte probabilité.
6. La sortie finale est un ensemble de boîtes de délimitation prédictes et d'étiquettes de classe pour chaque objet dans l'image.

3.1.4 Le Transfert Learning

3.1.4.1 Concept Général De La Technique De Transfert Learning

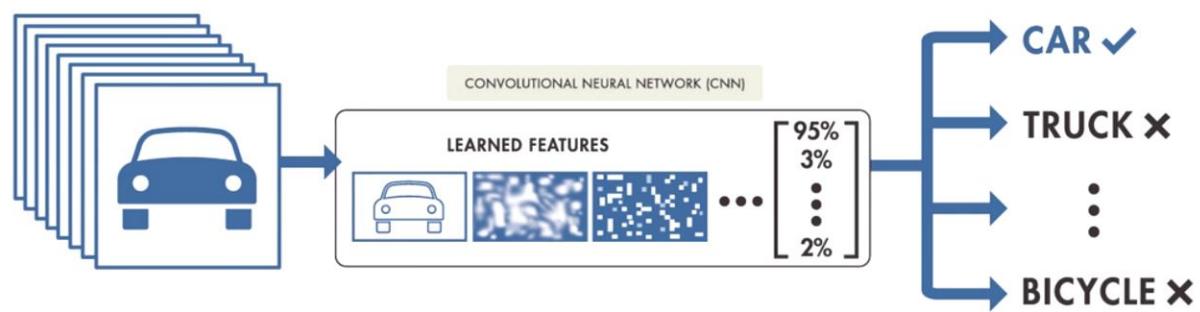
Le transfert Learning est une technique fondamentale dans le domaine de l'apprentissage automatique, qui permet de transférer des connaissances acquises lors de l'entraînement d'un modèle sur une tâche spécifique à une nouvelle tâche ou un nouveau domaine. Plutôt que de créer un modèle à partir de zéro et d'entraîner chaque paramètre à partir de données brutes, le transfert learning capitalise sur les connaissances préalablement acquises par un modèle déjà entraîné sur une tâche similaire ou une tâche plus générale. Cette approche présente de nombreux avantages, notamment une réduction du temps et des ressources nécessaires à l'entraînement d'un nouveau modèle, ainsi qu'une amélioration des performances grâce à l'utilisation de connaissances préalables.

3.1.4.2 Utilisation de Technique de Transfert Learning Dans La Détection d'objet :

En pratique, le transfert Learning implique généralement de réutiliser les couches basses ou intermédiaires d'un modèle pré-entraîné, souvent sur un grand ensemble de données, et d'adapter ces couches à la nouvelle tâche ou au nouveau domaine spécifique. Cette adaptation peut se faire en ré-entraînant uniquement quelques couches du modèle pré-entraîné, en ajustant les poids des couches existantes, ou en ajoutant de nouvelles couches pour répondre aux exigences de la nouvelle tâche.

Après avoir **collecté, identifié, annoté et traité les données**, l'étape la plus ardue de notre projet a été l'entraînement de YOLOv8 sur ce jeu de données en utilisant la technique de **Transfer Learning**. Cette méthode implique l'adaptation d'un modèle pré-entraîné à notre ensemble de données spécifique. Plutôt que de partir de zéro, nous avons utilisé ce modèle pré-entraîné comme point de départ, tirant parti des connaissances déjà acquises sur une tâche générale ou plus large pour accélérer l'apprentissage sur notre propre tâche ou ensemble de données. Ce processus nous a permis de personnaliser le modèle pour qu'il réponde mieux à nos besoins spécifiques, tout en réduisant le temps et les ressources nécessaires pour atteindre des performances satisfaisantes.

TRAINING FROM SCRATCH



TRANSFER LEARNING

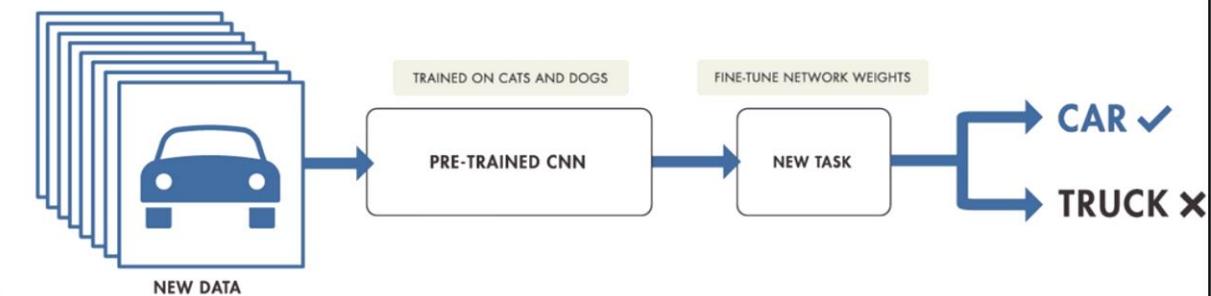
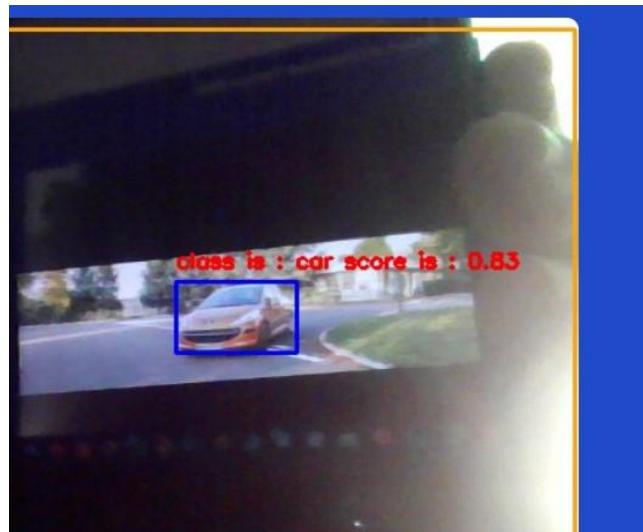
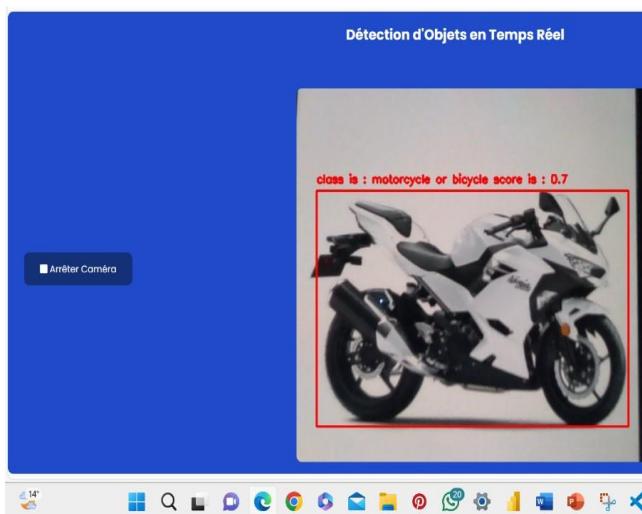


Figure 8 : transfer learning

3.2 Résultats de la première modèle d'entraînement :



3.3 Les réseaux de neurones convolutif :

3.3.1 Présentation Générale de CNN :

Les réseaux de neurones convolutifs ont été créés dans les années 1980. Le principe d'une convolution consiste à "résumer" les données d'entrée par des convolutions, afin de les analyser par des neurones entièrement connectés. Le principe des neurones convolutifs est composé de 4 parties :

- Convolution ;
- Pooling ;
- Fonction d'activation ReLU ;
- Couche FC (fully-connected).

Tous les réseaux de neurones ont été définis selon l'architecte, c'est-à-dire que le nombre de filtres par convolution, la taille du noyau et le "stride" ne varie pas et ne sont pas modifiables, sauf si l'utilisateur modifie les fichiers de configurations de l'architecture.

Généralement, l'architecture d'un **Convolutional Neural Network** est sensiblement la même :

- **Couche de convolution (CONV)** : Le rôle de cette première couche est d'**analyser les images fournies en entrée** et de **détecter la présence d'un ensemble de features**. On obtient en sortie de cette couche un ensemble de feature maps.

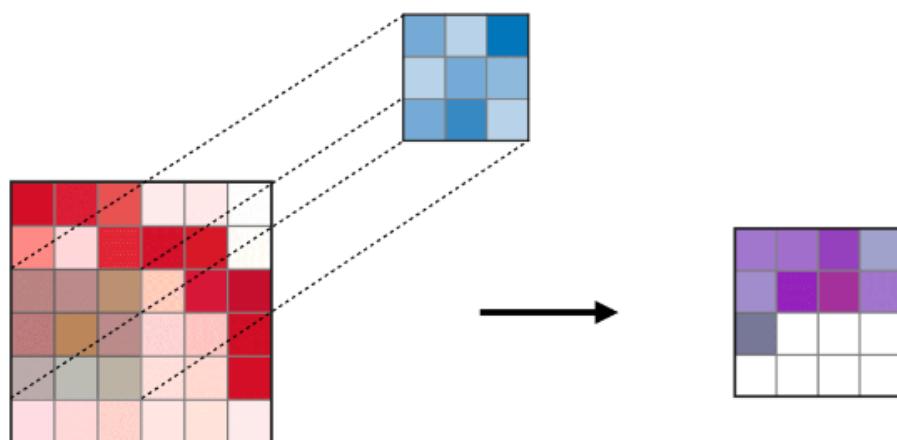


Figure 9 : Couche de convolution (CONV)

- **Couche de Pooling (POOL)** : La couche de Pooling est une opération généralement appliquée entre deux couches de convolution. Celle-ci reçoit en entrée les features maps formées en sortie de la couche de convolution et son rôle est de **réduire la taille des images, tout en préservant leurs caractéristiques les plus essentielles**. Parmi les plus utilisés, on retrouve le **max-pooling** mentionné précédemment ou encore l'**average pooling** dont l'opération consiste à conserver à chaque pas, la valeur moyenne de la fenêtre de filtre.

Finalement, on obtient en sortie de cette couche de Pooling, le même nombre de feature maps qu'en entrée mais considérablement compressées.

- **La couche d'activation ReLU (Rectified Linear Units) :** Cette couche **remplace toutes les valeurs négatives reçues en entrées par des zéros**. L'intérêt de ces couches d'activation est de rendre le modèle non linéaire et de ce fait plus complexe.

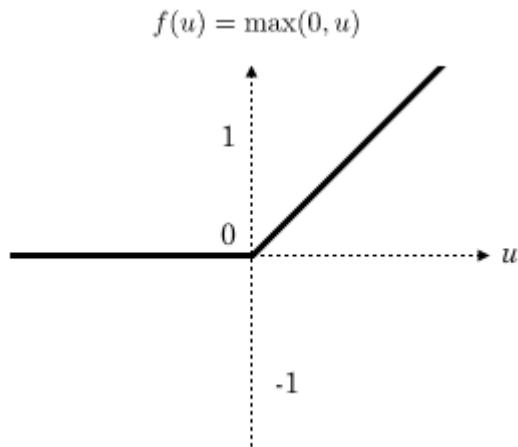


Figure 10 : fonction d'activation RELU

- **Couche Fully Connected (FC) :** Ces couches sont placées en fin d'architecture de CNN et sont **entièrement connectées à tous les neurones de sorties** (d'où le terme fully-connected). Après avoir reçu un vecteur en entrée, la couche FC applique successivement une **combinaison linéaire** puis une **fonction d'activation** dans le but final de **classifier l'input image** (*voir schéma suivant*). Elle renvoie enfin en sortie un **vecteur de taille d'correspondant au nombre de classes** dans lequel chaque composante représente la probabilité pour l'input image d'appartenir à une classe.

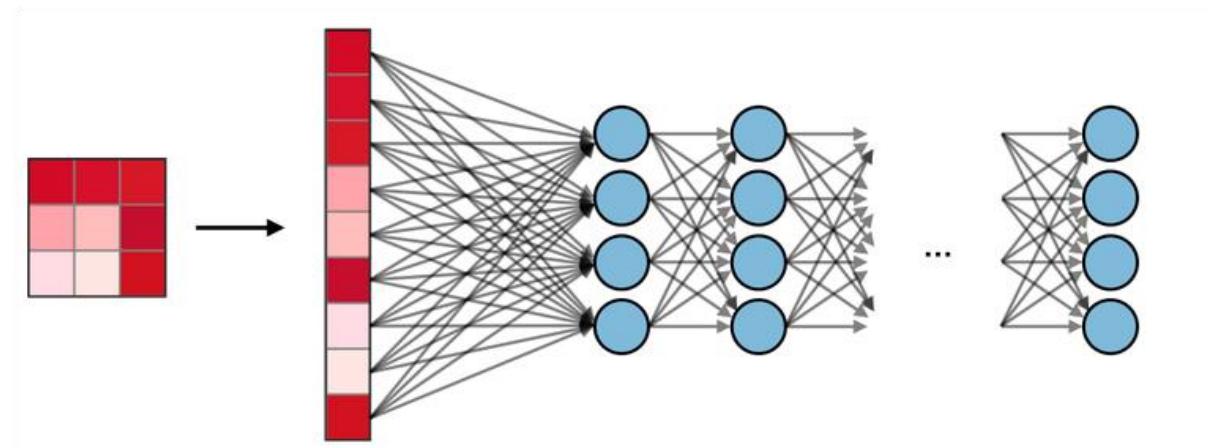


Figure 11 : Fonctionnement d'un réseau neuronal à 2 couches cachées

3.3.2 L'utilisation de Model CNN Dans la Détection D'objets :

3.3.2.1 Idée générale sur ResNet50:

ResNet50 est un modèle de réseau de neurones convolutifs (CNN) profondément architecturé, faisant partie de la famille des réseaux résiduels (ResNet). Il a été développé par des chercheurs de Microsoft Research Asia.

Le concept principal derrière ResNet50 est l'utilisation de "blocs résiduels" qui permettent au réseau d'apprendre des représentations plus précises et plus profondes. Ces blocs résiduels sont constitués de connexions directes (skip connections) qui contournent une ou plusieurs couches, facilitant le flux d'information et aidant à éviter le problème de disparition du gradient lors de l'entraînement de réseaux profonds.

ResNet50 est particulièrement connu pour sa capacité à atteindre des performances exceptionnelles dans la classification d'images et d'autres tâches de vision par ordinateur. Il a été pré-entraîné sur de vastes ensembles de données d'images, comme ImageNet, ce qui lui permet d'extraire des caractéristiques pertinentes à partir d'images complexes. En raison de ses performances élevées et de sa disponibilité dans des frameworks de Deep Learning populaires tels que PyTorch et TensorFlow, ResNet50 est largement utilisé dans de nombreux domaines, notamment la reconnaissance d'objets, la reconnaissance faciale et l'analyse d'images médicales.

3.3.2.2 L'utilisation de ResNet50 dans notre projet :

Dans notre approche, nous avons utilisé ResNet50, un modèle de classification, pour effectuer des tâches de détection d'objets en ajoutant des couches supplémentaires. Pour ce faire, nous avons intégré un modèle ResNet50 pré-entraîné comme base et ajouté des couches personnalisées par-dessus. Dans cette configuration, le modèle de base ResNet50 est figé, c'est-à-dire qu'il n'est pas entraînable, et seules les couches personnalisées sont sujettes à l'entraînement.

Les couches supplémentaires que nous avons ajoutées comprennent une couche de pooling, une couche Dense avec 256 unités et une activation ReLU, ainsi que deux couches de sortie pour les prédictions de classes et les prédictions de bounding boxes.

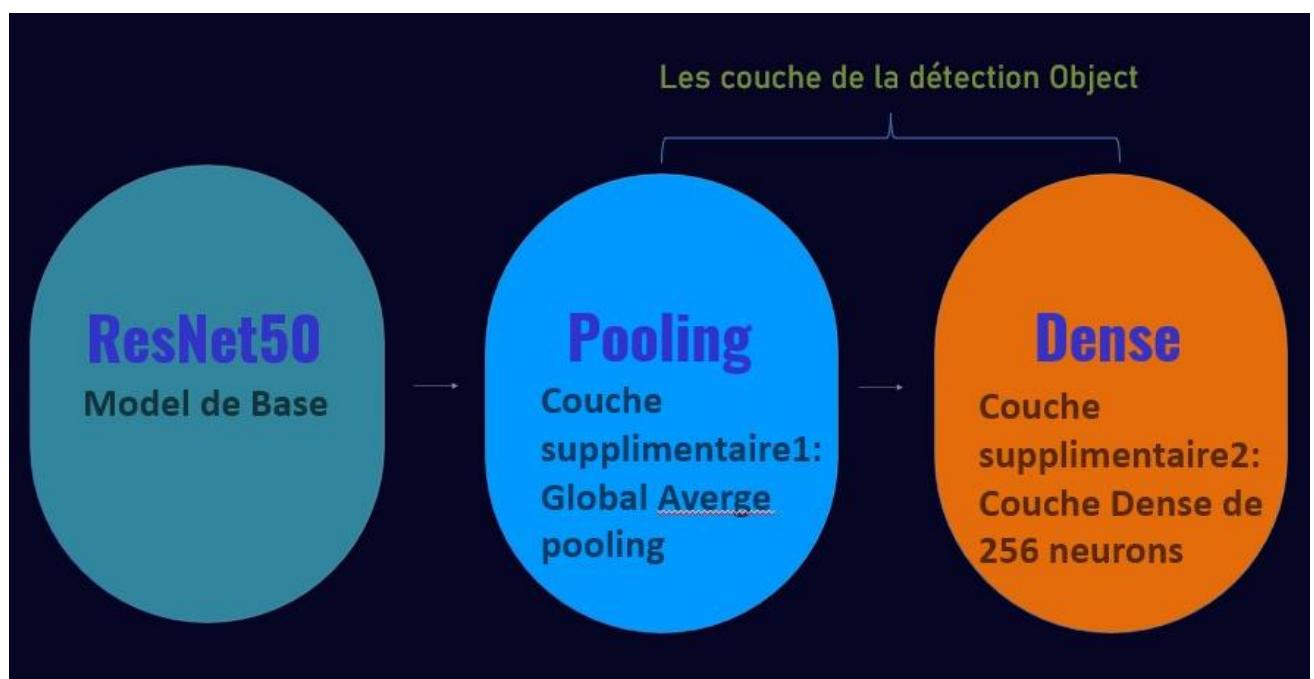


Figure 12: Architecture de deuxième modèle d'entraînement.

3.3.3 Résultats et contraintes de l'entraînement de CNN :

Finalement, Malgré nos efforts et notre entraînement sur 20 époques, nous avons rencontré des défis dans la prédiction des bounding boxes, comme illustré sur la figure. Ces défis peuvent provenir de diverses sources telles que des données d'entraînement insuffisantes, des paramètres de modèle inappropriés ou des difficultés spécifiques liées à la nature des objets détectés.

```

2024-05-06 15:18:30.031296: I tensorflow/core/util/port.cc:113] oneDNN custom op rs. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-06 15:18:32.594352: I tensorflow/core/util/port.cc:113] oneDNN custom op rs. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-06 15:18:39.597653: I tensorflow/core/platform/cpu_feature_guard.cc:141] To enable the following instructions: AVX2 FMA, in other operations, rebuild.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be run.
1/1 2s 2s/step
[[[1.2588653e-07 4.2198462e-05 2.7675301e-07 2.9259528e-09 1.4205454e-01]
 [5.7252483e-03 1.6266632e-01 1.2460706e-07 2.5787005e-08 3.0369472e-04]
 [2.4120902e-01 4.5571280e-05 7.4190335e-05 4.6129654e-07 3.6527219e-04]
 [4.4928305e-02 9.0634484e-07 4.2925905e-07 1.6343342e-10 1.6171297e-08]
 [4.0258306e-01 2.3532372e-08 3.3279819e-09 1.6295841e-18 1.3678778e-07]]
 [5.8116748e-11 9.0248582e-09 2.9964908e-09 5.0607496e-09]
 [2.7007435e-10 6.3795208e-10 4.2266963e-09 4.9407574e-12]
 [2.5464575e-09 3.7758061e-10 4.8890719e-10 4.2032714e-12]
 [8.9142010e-10 1.4546817e-10 1.7238738e-09 2.5546712e-11]]]

```

Figure 13: résultat de deuxième modèle .

4 Les Mesures De Performances Pour L'évaluation Des Modelés.

4.1 Introduction :

Les mesures de performance sont des outils clés pour évaluer la précision et l'efficacité des modèles de détection d'objets. Elles mettent en lumière l'efficacité avec laquelle un modèle peut identifier et localiser des objets dans des images. En outre, elles permettent de comprendre comment le modèle gère les faux positifs et les faux négatifs. Ces informations sont cruciales pour évaluer et améliorer les performances du modèle. Dans notre rapport , nous allons explorer les différentes mesures de performance associées à YOLOv8, leur signification et la façon de les interpréter.

4.2 Des Concepts Fondamentaux.

Dans le processus d'évaluation des performances d'un modèle, il est crucial de comprendre les concepts de vrais positifs, faux positifs, vrais négatifs et faux négatifs.

Les vrais positifs (VP) : représentent les cas où le modèle a correctement identifié une instance positive dans les données. Par exemple, si un modèle de détection d'objets prédit la présence d'un chien dans une image qui contient effectivement un chien, alors cette prédiction est considérée comme un vrai positif.

Les faux positifs (FP) : à l'opposé sont des cas où le modèle a incorrectement prédit la présence d'une instance positive qui en réalité n'existe pas dans les données. Par exemple, si un modèle de détection d'objets prédit la présence d'un chien dans une image où il n'y a pas de chien, cette prédiction est considérée comme un faux positif.

Les vrais négatifs (VN): quant à eux, représentent les cas où le modèle a correctement identifié l'absence d'une instance positive dans les données. Par exemple, si un modèle de détection d'objets ne détecte pas la présence d'un chien dans une image où il n'y a effectivement pas de chien, alors cette non-détection est considérée comme un vrai négatif.

Les faux négatifs (FN) : sont des cas où le modèle a manqué de détecter une instance positive qui est réellement présente dans les données. Par exemple, si un modèle de détection d'objets ne parvient pas à détecter la présence d'un chien dans une image où un chien est clairement visible, alors cette non-détection est considérée comme un faux négatif.

4.3 Métriques de détection d'objets

Commençons par discuter de certaines mesures qui ne sont pas seulement importantes pour YOLOv8 mais qui sont largement applicables à différents modèles de détection d'objets.

Intersection sur Union (IoU) : L'IoU est une mesure qui quantifie le chevauchement entre une boîte de délimitation prédite et une boîte de délimitation de vérité terrain. Elle joue un rôle fondamental dans l'évaluation de la précision de la localisation des objets.

Précision moyenne (AP) : AP calcule la surface sous la courbe de précision-rappel, fournissant une valeur unique qui englobe les performances de précision et de rappel du modèle.

Précision et rappel : La précision quantifie la proportion de vrais positifs parmi toutes les prédictions positives, évaluant la capacité du modèle à éviter les faux positifs. D'autre part, le rappel calcule la proportion de vrais positifs parmi tous les positifs réels, mesurant ainsi la capacité du modèle à détecter toutes les instances d'une classe.

$$\text{Précision} = \frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux positifs}}$$

$$\text{Rappel} = \frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux négatifs}}$$

Précision moyenne (mAP) : mAP étend le concept de PA en calculant les valeurs moyennes de PA sur plusieurs classes d'objets. Cette méthode est utile dans les scénarios de détection d'objets multi-classes pour fournir une évaluation complète des performances du modèle.

Score F1 : Le score F1 est la moyenne harmonique de la précision et du rappel, fournissant une évaluation équilibrée des performances d'un modèle tout en tenant compte des faux positifs et des faux négatifs.

$$F1 = \frac{2 * \text{Précision} * \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

4.4 Analyse Approfondie Des Métriques Pour Le Modèle YOLO.8

La validation est une étape critique du pipeline d'apprentissage automatique, qui te permet d'évaluer la qualité de tes modèles entraînés. Le mode Val du site **Ultralytics YOLOv8** fournit une suite robuste d'outils et de mesures pour évaluer les performances de tes modèles de détection d'objets. L'utilisation du mode de validation est simple. Une fois que tu as un modèle entraîné, tu peux invoquer la fonction **model.Val()**. Cette fonction traitera alors l'ensemble de données de validation et renverra diverses mesures de performance. Mais que signifient ces mesures ? Et comment dois-tu les interpréter ?

la sortie de la fonction **model.val()** se décompose sur plusieurs sorties , comprenons chaque segment de la sortie.

➤ Métriques par classe

L'une des sections de la sortie est la répartition par classe des mesures de performance. Ces informations granulaires sont utiles lorsque tu cherches à comprendre les performances du modèle pour chaque classe spécifique, en particulier dans les ensembles de données comportant une gamme variée de catégories d'objets. Pour chaque classe de l'ensemble de données, les informations suivantes sont fournies :

Classe: Elle désigne le nom de la classe d'objets, par exemple "personne", "voiture" ou "chien".

Images: Cette métrique t'indique le nombre d'images de l'ensemble de validation qui contiennent la classe d'objets.

Instances: Ce chiffre indique le nombre de fois où la classe apparaît sur toutes les images de l'ensemble de validation.

Box (P, R, mAP50, mAP50-95): Cette métrique donne un aperçu des performances du modèle en matière de détection d'objets :

- P (Précision): La précision des objets détectés, indiquant combien de détections étaient correctes.
- R (Recall): La capacité du modèle à identifier toutes les instances d'objets dans les images.
- mAP50: précision moyenne calculée à un seuil d'intersection sur union (IoU) de 0,50. C'est une mesure de la précision du modèle qui ne prend en compte que les détections "faciles".
- mAP50-95: La moyenne de la précision moyenne calculée à différents seuils d'IoU, allant de 0,50 à 0,95. Elle donne une vue d'ensemble des performances du modèle à différents niveaux de difficulté de détection.

• Mesures de la vitesse

La vitesse d'inférence peut être aussi critique que la précision, en particulier dans les scénarios de détection d'objets en temps réel. Cette section décompose le temps nécessaire aux différentes étapes du processus de validation, du prétraitement au post-traitement.

4.5 Interprétation des résultats

4.5.1 Matrice De Confusion :

Elle permet de visualiser la performance d'un algorithme en comparant les valeurs prédites par le modèle aux valeurs réelles de l'ensemble de données. La matrice de confusion est généralement de taille $n \times n$, où n est le nombre de classes dans le problème de classification. Les lignes (l'axe x) de la matrice représentent les valeurs réelles, tandis que les colonnes (l'axe y) représentent les valeurs prédites par le modèle.

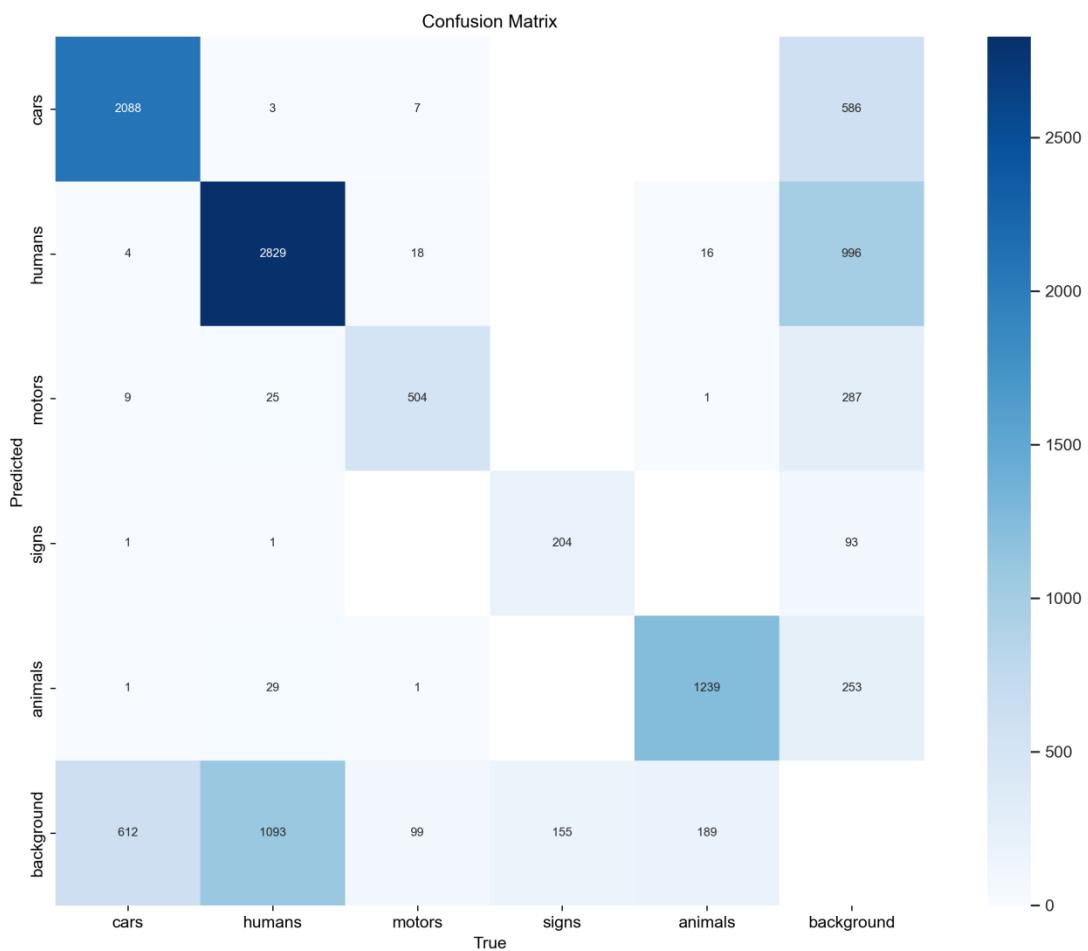


Figure 14 : matrice de confusion

Dans notre cas, le nombre de classes n est égale à 5 plus le background qui est considéré comme une classe. Les catégories présentes sont : Voitures, Humains, Moteurs, Signes, Animaux, Fond

❖ Interprétation et Évaluation :

Les performances du modèle varient dans la classification des différentes catégories d'objets. Il parvient à bien classifier les voitures avec 2088 prédictions correctes, ainsi que 2829 humains et 1239 animaux correctement identifiés. Cependant, des confusions persistent,

notamment dans la distinction entre humains et moteurs. Le modèle présente 25 faux positifs et identifie à tort 18 objets comme des moteurs alors qu'ils sont considérés comme humains. De plus, il confond 29 objets humains avec des animaux et 16 objets animaux avec des humains.

Les erreurs persistent également dans la classification du fond, qui est souvent confondu avec d'autres catégories. Par exemple, le modèle a classé 155 objets de type panneaux de signalisation comme faisant partie du fond. Cette confusion découle d'un processus d'annotation limité lors de la création du jeu de données. Les annotations initiales ont été réalisées en utilisant l'outil YOLO, qui ne comprenait que deux types de panneaux de signalisation (stop et autres panneaux de signalisation) parmi les catégories. Par conséquent, toutes les autres catégories de panneaux n'ont pas été annotées, ce qui a conduit le modèle à considérer ces objets comme faisant partie du fond.

Ainsi, bien que les annotations aient été effectuées avec soin pour certains types d'objets, la limitation de l'outil d'annotation utilisé a entraîné des confusions dans la classification des autres catégories, nécessitant une révision et une extension de l'ensemble de données annotées pour améliorer les performances du modèle.

4.5.2 Matrice De Confusion Normalisée :

Cette visualisation est une version normalisée de la matrice de confusion. Elle représente les données en proportions plutôt qu'en nombres bruts. Ce format permet de comparer plus simplement les performances entre les classes.

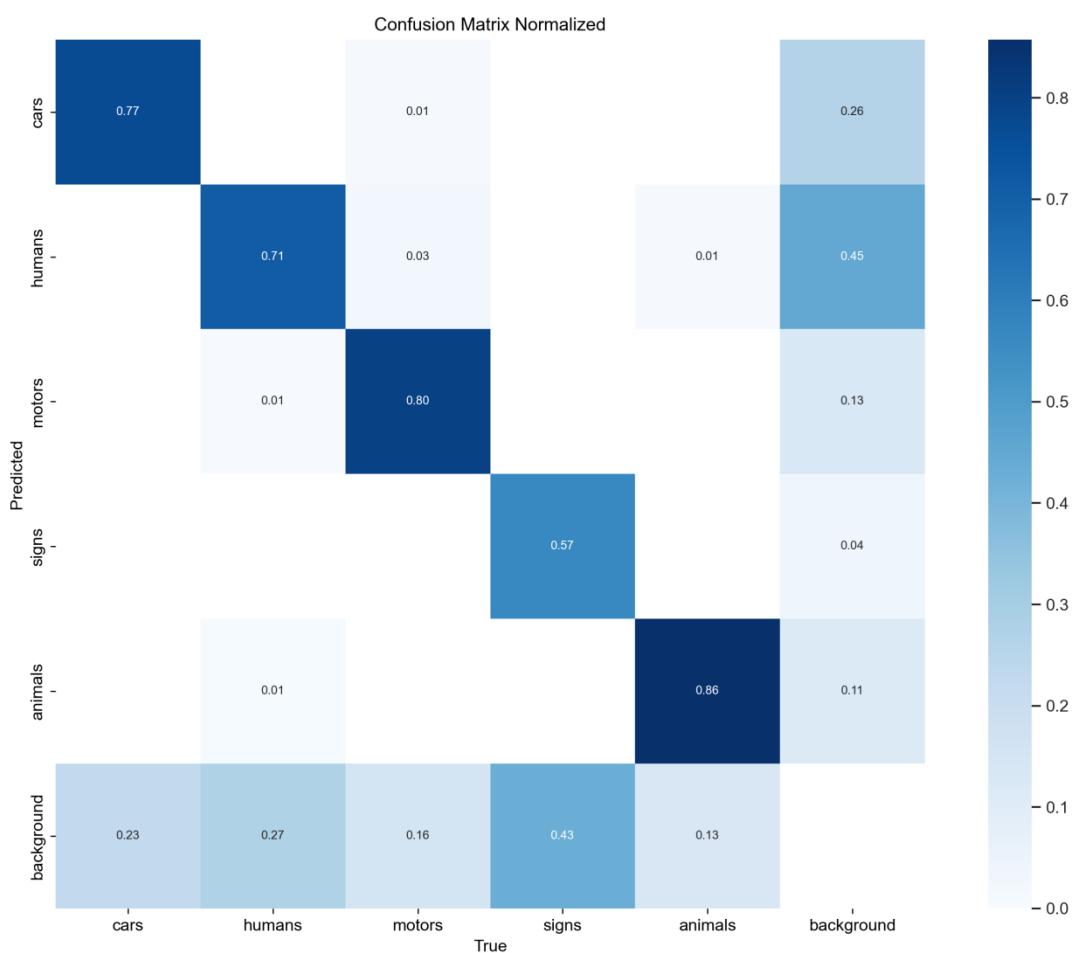


Figure 15 : Matrice De Confusion Normalisée

4.5.3 Les Métriques et les Pertes du Modèle YOLO :

Les pertes d'entraînement ainsi dans le contexte du modèle YOLO (You Only Look Once) sont essentielles pour évaluer la performance et le processus d'apprentissage du modèle pendant la phase d'entraînement qui contient 10 épisodes. Ces pertes sont généralement calculées à chaque itération (ou épisode) d'entraînement et fournissent des indications sur la capacité du modèle à ajuster ses poids et ses biais pour mieux s'adapter aux données d'entraînement.

Pour les pertes de validation sont utilisées pour évaluer la performance du modèle sur un jeu de données de validation distinct de celui utilisé pour l'entraînement. Ces pertes fournissent une indication de la capacité du modèle à généraliser à de nouvelles données, c'est-à-dire à faire des prédictions précises sur des données qu'il n'a pas vu pendant l'entraînement

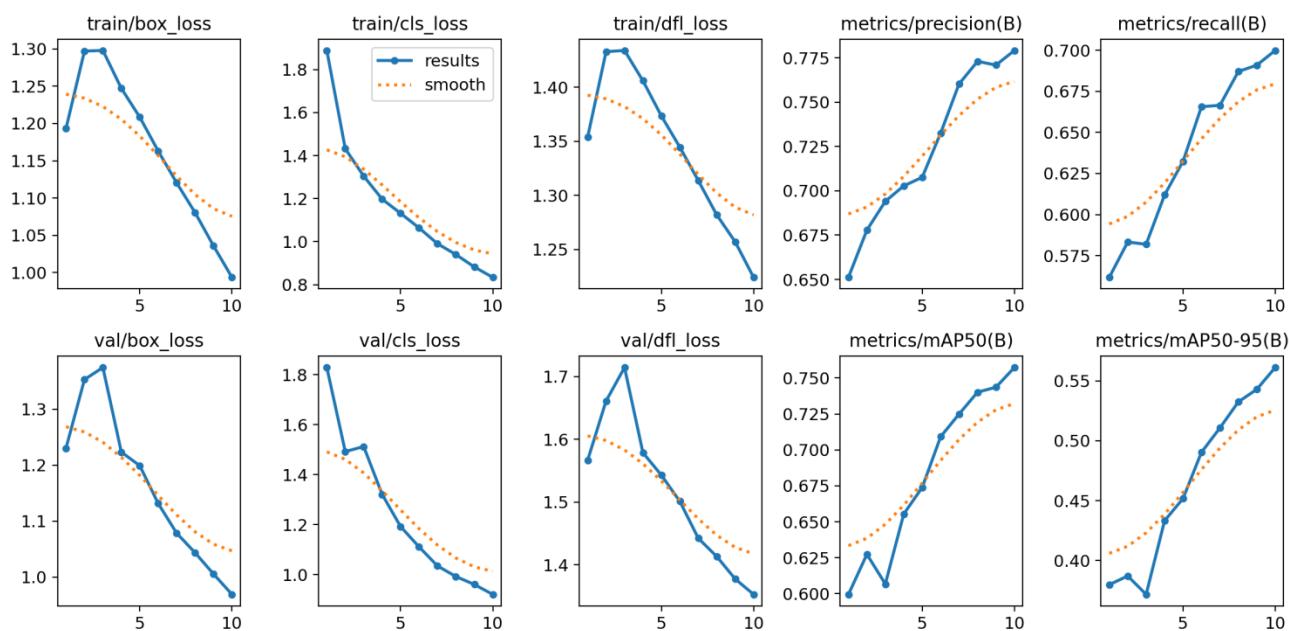


Figure 16: Les Métriques et les Pertes du Modèle YOLO

➤ Les pertes du modèle YOLO :

✚ Courbe train/box_loss (perte de boîte) :

Cette perte mesure à quel point les boîtes englobantes prédites par le modèle correspondent aux boîtes englobantes réelles des objets dans l'image. Une diminution de cette perte indique que le modèle apprend à mieux localiser les objets dans l'image. Plus précisément, cette perte est calculée en comparant les coordonnées prédites des boîtes (comme les coordonnées x et y du centre, la largeur et la hauteur) avec les coordonnées réelles des boîtes dans les données d'entraînement. Une diminution de cette perte indique que le modèle améliore sa capacité à prédire avec précision la localisation des objets.

Courbe Val/box_loss (perte de boîte) :

Pendant la validation. Une baisse constante de cette perte indique que le modèle généralise bien à de nouvelles données en termes de localisation précise des objets. Une perte de boîte stable indique que le modèle est capable de localiser correctement les objets dans des scènes qu'il n'a pas vues pendant l'entraînement.

Courbe train/cls_loss (perte de classification) :

Cette perte mesure à quel point les prédictions de classe du modèle correspondent aux classes réelles des objets dans l'image. Une diminution de cette perte indique que le modèle apprend à mieux classifier les objets détectés. Plus précisément, cette perte est calculée en comparant les probabilités prédites des classes avec les classes réelles des objets dans les données d'entraînement. Une diminution de cette perte indique que le modèle améliore sa capacité à classifier correctement les objets détectés.

Courbe Val/cls_loss (perte de classification) :

Pour la phase de validation. Une baisse constante de cette perte indique que le modèle généralise bien à de nouvelles données en termes de classification précise des objets. Une perte de classification stable indique que le modèle est capable de classifier correctement les objets dans des scènes qu'il n'a pas vues pendant l'entraînement.

Courbe train/df1_loss (perte totale):

Cette perte totale combine à la fois la perte de boîte et la perte de classification, généralement pondérées par des coefficients pour donner plus d'importance à l'un ou l'autre aspect selon les besoins. Une diminution de cette perte indique une amélioration globale de la performance du modèle dans la détection et la classification des objets dans l'image. Cette perte est souvent utilisée pour l'optimisation globale du modèle pendant l'entraînement.

Courbe Val/df1_loss (perte totale):

Pour la phase de validation. Une baisse constante de cette perte indique une bonne généralisation du modèle à de nouvelles données en termes de détection et de classification précises des objets. Une perte totale stable indique que le modèle est capable de généraliser correctement à de nouvelles données et n'est pas surajusté (overfitting) aux données d'entraînement.

En synthèse, la réduction des pertes, qu'il s'agisse de la perte de boîte, de la perte de classification ou de la perte totale, reflète l'amélioration progressive du modèle YOLO dans sa capacité à localiser et à classifier les objets dans les images d'entraînement lors de la phase d'entraînement, ainsi que lors de la phase de validation. Cette amélioration suggère que le modèle peut généraliser à de nouvelles données et produire des prédictions précises pour des scènes qu'il n'a pas rencontrées durant l'entraînement initial. Les pertes de validation revêtent donc une importance capitale pour évaluer la performance réelle du modèle et garantir son efficacité dans des contextes du monde réel.

➤ **Les métriques de modèle YOLO :**

✚ **Courbe (metrics/precision(B))**

Une augmentation de la précision pendant les 10 épisodes d'entraînement indique que le modèle YOLO devient progressivement plus précis dans sa capacité à identifier correctement les objets dans les images. L'augmentation de la précision au fil des épisodes peut être due à plusieurs facteurs. Le modèle peut apprendre à mieux discriminer les caractéristiques des objets d'intérêt, ce qui conduit à des prédictions plus précises. De plus, l'optimisation continue des poids du modèle pendant l'entraînement peut également contribuer à cette amélioration.

✚ **Courbe (metrics/recall(B))**

Une augmentation du rappel pendant les 10 épisodes d'entraînement indique que le modèle YOLO devient progressivement plus efficace pour détecter un plus grand nombre d'objets positifs dans les images. Cela est généralement considéré comme une amélioration de la performance du modèle, car il devient plus complet dans sa capacité à identifier correctement les objets d'intérêt.

✚ **Courbe Metrics/mAP@50(B):**

mAP@50(B) évalue la précision moyenne du modèle à une intersection sur union (IoU) de 0,5, considérant ainsi les détections avec un seuil IoU plus bas, souvent appelées détections "faciles". L'augmentation de mAP@50(B) indique que le modèle devient plus précis dans la localisation des objets avec un IoU de 0,5, ce qui signifie qu'il est capable de détecter plus efficacement les objets même avec un chevauchement moins important entre la prédiction et la vérité terrain.

✚ **Courbe Metrics/mAP@50-95(B):**

mAP@50:95(B) étend cette évaluation sur une plage d'IoU de 0,5 à 0,95, couvrant ainsi une gamme plus large de seuils d'IoU. Une augmentation de mAP@50:95(B) montre que le modèle devient plus robuste dans la détection d'objets sur une gamme variée de seuils d'IoU, ce qui signifie qu'il peut détecter avec précision les objets même dans des conditions où le chevauchement entre la prédiction et la vérité terrain peut varier.

4.5.4 Courbe Précision-confiance PCC :

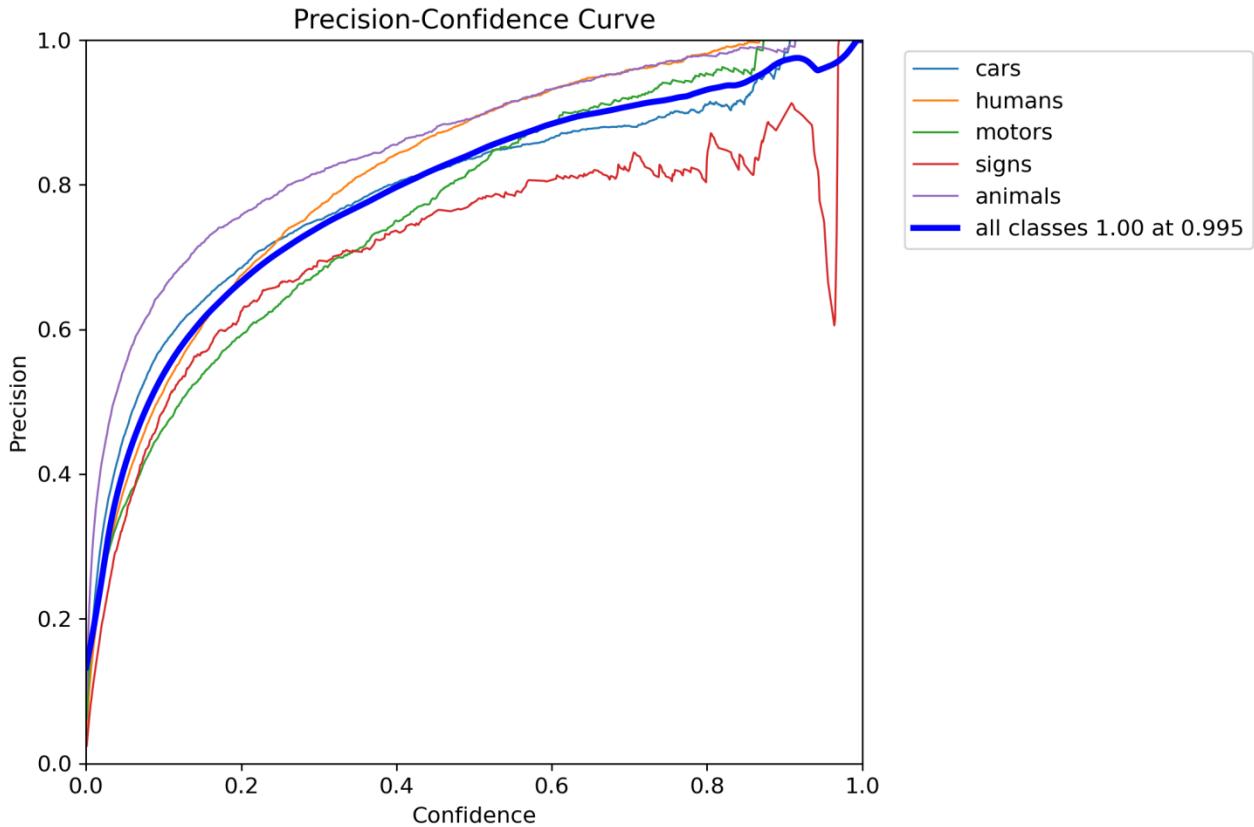


Figure 17: courbe Précision-confiance

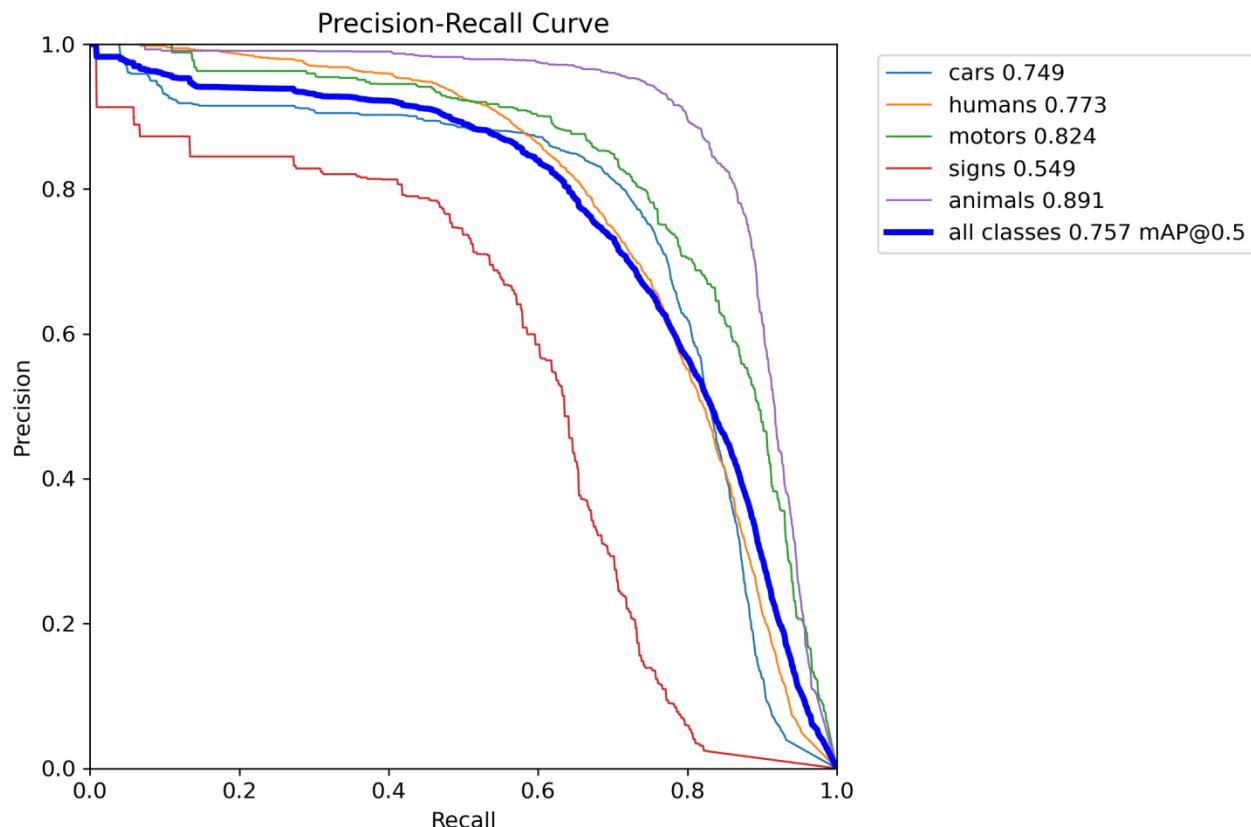
Cette courbe représente la relation entre la précision du modèle et le seuil de confiance utilisé pour filtrer les prédictions.

Chaque courbe sur le graphique représente une classe d'objets détectés par le modèle. On observe que la courbe correspondant aux animaux présente une précision élevée même à des niveaux de confiance relativement bas, tandis que pour les autres classes telles que les moteurs, les panneaux de signalisation, les voitures et les humains, la précision est généralement faible lorsque la confiance est basse, sauf pour les panneaux de signalisation où la précision diminue à partir d'une confiance de 0.8.

Cette observation suggère une relation significative entre la précision des classes détectées et le niveau de confiance associé à ces détections. En effet, lorsque la confiance du modèle est élevée, la précision des classes détectées tend également à être élevée, témoignant ainsi d'une forte fiabilité dans les prédictions. En revanche, une baisse de la confiance du modèle est souvent accompagnée d'une diminution de la précision des classes détectées, signalant une moindre fiabilité dans les prédictions. Ainsi, la confiance du modèle semble jouer un rôle crucial en tant qu'indicateur de la précision de ses prédictions, et une confiance élevée est généralement associée à une précision accrue dans la détection des différentes classes d'objets.

4.5.5 Courbe Précision-Rappel PRC :

L'image illustre la courbe de précision-rappel pour différentes classes : voitures, humains, moteurs, panneaux, animaux et toutes les classes. Généralement tous les courbes de classes



montrent une contradiction entre rappel et précision.

Figure 18: Courbe precision-rappel

La Contradiction entre précision et confidence :

Lorsque la précision est maximale, le rappel est nul, et vice versa. Cela peut se produire dans des cas où le modèle est très conservateur dans ses prédictions. Lorsque le modèle est très strict sur ce qu'il prédit comme étant un objet, il peut avoir une précision élevée, car les prédictions sont généralement correctes, mais il manquera probablement beaucoup d'instances réelles d'objets (faux négatifs), ce qui conduira à un rappel faible. D'autre part, lorsque le modèle est très généreux dans ses prédictions, il peut avoir un rappel élevé car il détecte la plupart des objets réels (vrais positifs), mais il peut aussi inclure beaucoup de fausses prédictions (faux positifs), ce qui réduit la précision.

4.5.6 Courbe confiance -Rappel RCC :

La courbe RCC idéale maintiendrait un taux de rappel élevé à tous les niveaux de confiance, indiquant que le modèle détecte les objets avec précision et un minimum de faux négatif. Cependant, dans la pratique, il existe souvent un compromis entre la précision et le rappel à différents niveaux de confiance.

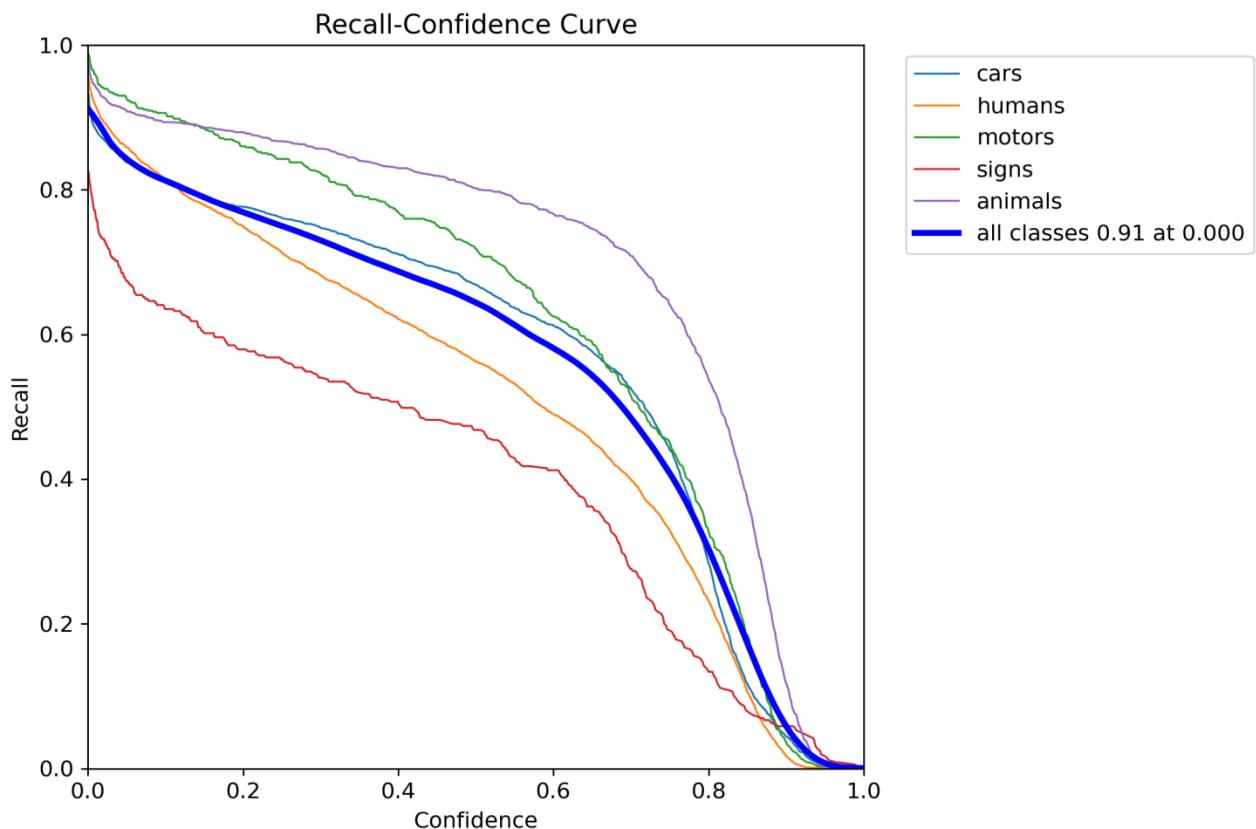


Figure 19: Courbe confiance -Rappel RCC

La Contradiction entre Rappel et confidence :

Si le rappel diminue lorsque la confiance augmente, cela peut indiquer que le modèle devient plus sélectif dans ses prédictions à mesure que la confiance augmente. En d'autres termes, à des seuils de confiance plus élevés, le modèle ne prédit que les détections pour lesquelles il est très sûr, ce qui peut conduire à manquer certaines détections plus subtiles mais moins certaines. Cela peut se traduire par une baisse du rappel.

4.5.7 Courbe F1 score-confiance :

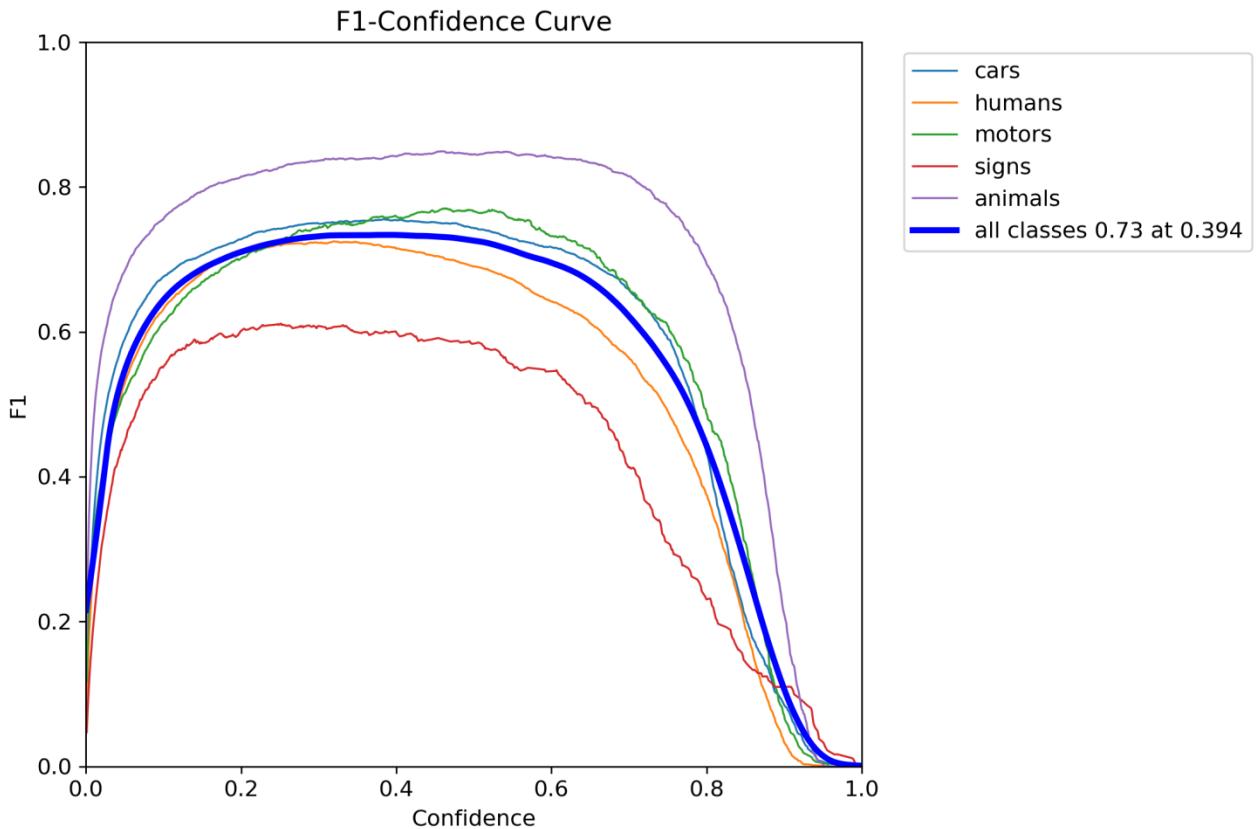


Figure 20: Courbe F1 score-confiance

L'axe des abscisses, qui représente le seuil de confiance, mesure la certitude du modèle dans ses prédictions, variant de 0 à 1. Un seuil de confiance plus élevé indique que le modèle est plus sûr de ses détections, ne considérant que les prédictions jugées très fiables, tandis qu'un seuil de confiance plus bas peut inclure des détections moins fiables. À l'opposé, l'axe des ordonnées représente le score F1, une métrique combinant à la fois la précision et le rappel pour évaluer la performance globale du modèle.

En examinant les courbes spécifiques, on remarque que pour les voitures, le modèle maintient un score F1 élevé et constant autour d'une confiance comprise entre 0.2 et 0.8, suggérant ainsi une bonne capacité à détecter avec précision les voitures. De même, pour les humains, la courbe révèle une performance élevée avec un score F1 stable autour du même niveau de confiance.

En revanche, la courbe des panneaux indique un score F1 plus bas, soulignant ainsi une difficulté accrue dans la détection de cette classe pour le modèle, avec un maximum atteint à environ 0.5 de confiance. Bien que les panneaux présentent une précision relativement élevée, leur rappel est légèrement inférieur, atteignant leur score F1 maximum à une confiance d'environ 0.7.

En ce qui concerne les animaux, leur courbe affiche le score F1 le plus élevé de toutes les classes, avec une valeur de 0.8 à la fois à 0.2 et à 0.8 de confiance. Pour les moteurs, une performance satisfaisante est observée dans la même plage de confiance, atteignant un score F1 d'environ 0.7.

La courbe moyenne (bleu foncé) représente la performance globale du modèle pour toutes les classes, avec un score F1 moyen de 0.76 à une confiance allant de 0.2 à 0.8. Cette valeur indique une bonne performance globale du modèle dans la détection d'objets sur l'ensemble des classes à des niveaux de confiance relativement élevés.

4.5.8 Analyse des annotations :

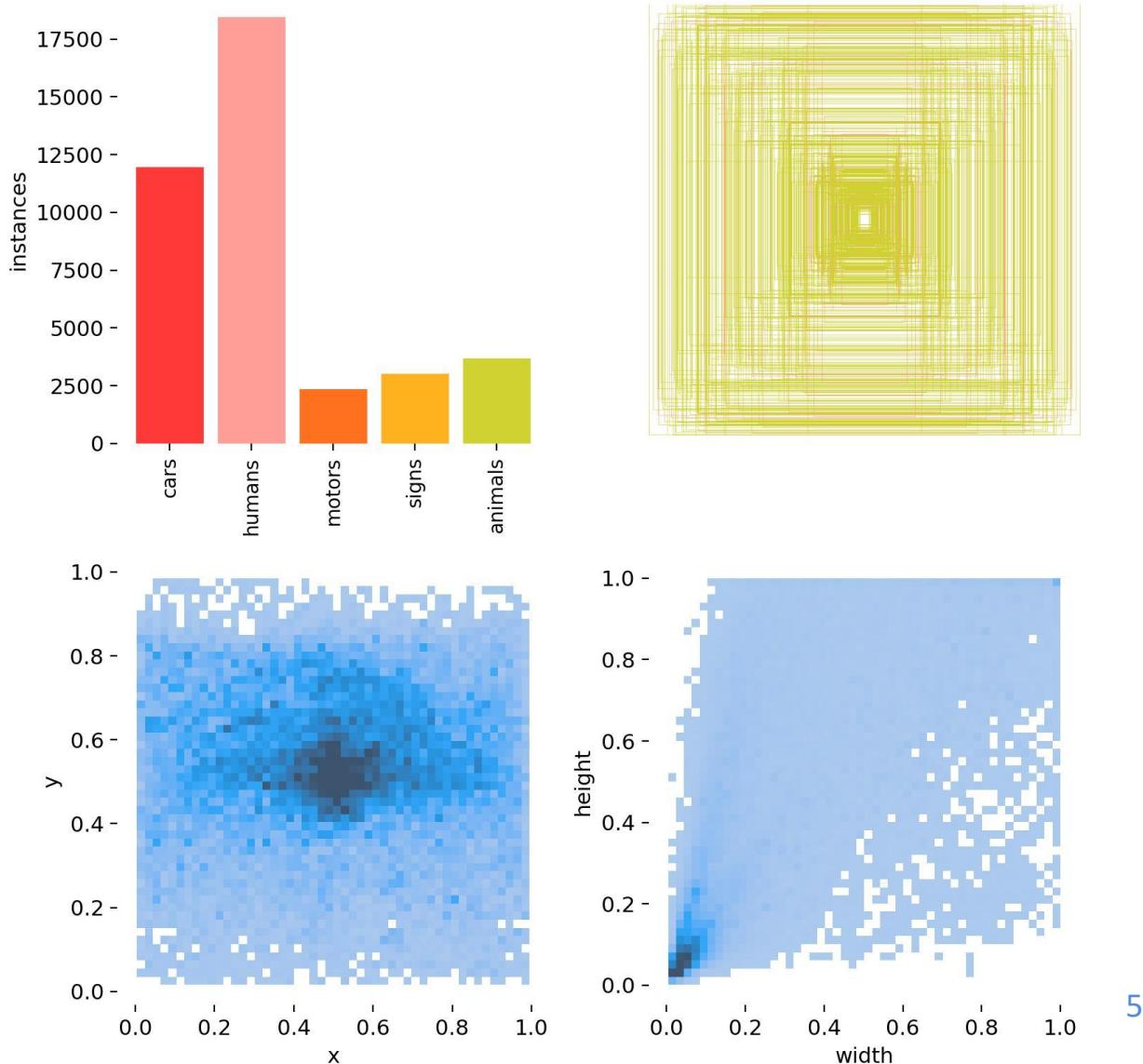


Figure 21 : les courbes de l'annotation model YOLO

➤ Histogramme des instances par catégorie :

L'histogramme représente le nombre d'instances pour cinq catégories distinctes : voitures, humains, motos, panneaux et animaux. Les instances indiquent le nombre de fois où chaque classe apparaît dans l'ensemble de validation. Les humains sont la catégorie la plus fréquente, c'est-à-dire cette classe est la classe la plus apparue dans les instances, suivies par les voitures , tandis que les motos, les panneaux et les animaux ont nettement moins d'instances.

➤ **Représentation des boîtes englobantes (Bounding Boxes) :**

Cette visualisation présente les boîtes englobantes générées par le modèle pour les objets détectés, offrant ainsi une représentation visuelle précise de l'emplacement et de la taille de chaque objet dans l'image. Elle pourrait illustrer la superposition des boîtes englobantes autour des objets détectés, permettant ainsi une compréhension claire de leur position relative et de leur dimension. Les structures formées par les lignes jaunes pourraient révéler des relations spatiales significatives entre les objets détectés, fournissant ainsi des informations supplémentaires sur leur disposition dans l'image

➤ **NUAGE DE POINTS (Distribution spatiale) :**

Le troisième graphique présente un nuage de points bleu foncé sur fond bleu clair, illustrant la distribution spatiale des objets détectés dans les images. La concentration la plus élevée se trouve au centre de l'image, ce qui indique une zone d'intérêt. Cette forte concentration de points suggère que la plupart des objets ont été détectés dans cette zone centrale de l'image. Ces résultats indiquent que le modèle se comporte bien en localisant les objets dans le cadre de l'image.

➤ **NUAGE DE POINTS (Hauteur et largeur relative) :**

Le quatrième graphique présente également un nuage de points qui représente la hauteur et la largeur relatives des objets détectés. La concentration principale de points dans le coin inférieur gauche du graphique indique que la plupart des objets ont une petite hauteur et largeur relatives. Cette observation suggère que le modèle pourrait être plus précis pour détecter les petits objets par rapport aux plus grands.

➤ **Interprétation globale**

Sur la base de l'analyse de ces quatre graphiques, on peut conclure que le modèle YOLO a obtenu des résultats prometteurs dans la détection d'objets de différentes catégories. Le modèle montre de bonnes performances dans la localisation des objets et l'identification des classes d'objets courantes. Cependant, il y a une marge d'amélioration dans la gestion des objets moins fréquemment rencontrés et dans l'amélioration de la précision de la détection des objets plus petits.

4.5.9 Le corrélogramme :

Le corrélogramme, également appelé pair plot, est un outil puissant pour analyser les relations et les distributions entre les paires de variables dans un ensemble de données. Voici ce que chaque partie de ce graphique signifie :

1. Scatter plots (Graphiques de dispersion) :

- Chaque petit graphique représente un scatter plot (nuage de points) entre deux variables.
- Les axes x et y de chaque scatter plot correspondent à deux variables différentes dans l'ensemble de données.

- Les points dans chaque scatter plot représentent les valeurs de ces deux variables pour chaque échantillon de données.
- L'agencement des points permet d'observer visuellement les relations entre les variables.

2. Histogrammes (ou kernel density estimates) :

- Le long de la diagonale du graphique, on trouve des histogrammes (ou des estimations de densité de noyau) pour chaque variable individuelle.
- Ces histogrammes montrent la distribution des valeurs pour chaque variable.

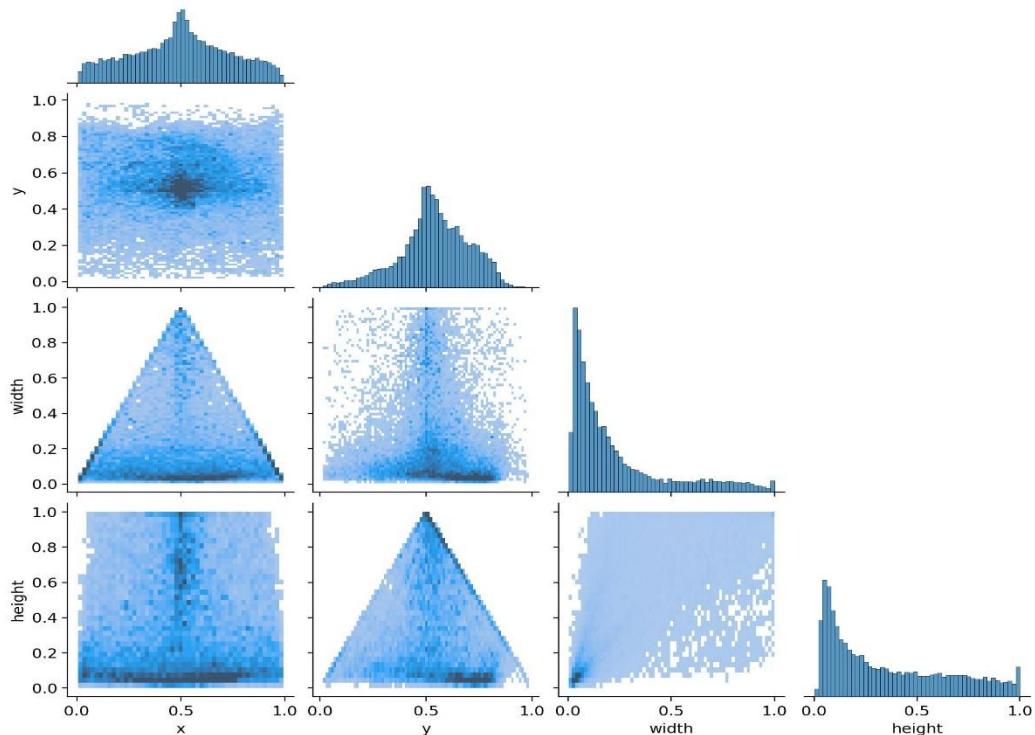


Figure 22 : Corrélogramme

○ Interprétations :

- La première ligne et la première colonne montrent la distribution des coordonnées x et y respectivement. La concentration élevée au centre indique que la plupart des objets détectés sont centrés.
- La troisième ligne et colonne représentent la distribution de la largeur des boîtes englobantes. Il y a une concentration vers le côté gauche, indiquant que beaucoup d'objets ont une petite largeur.

- La dernière ligne et colonne montrent la distribution de la hauteur avec également une concentration sur le côté gauche, suggérant que beaucoup d'objets ont une petite hauteur.
- Les graphiques hors diagonale illustrent les corrélations entre deux variables différentes sous forme de nuages de points. Cherche des tendances linéaires ou des relations spécifiques entre les variables. Prenant l'exemple de deux graphes :
 - Les scatter plots représentant les coordonnées x et y indiquent la répartition spatiale des objets détectés, mettant en évidence leur position dans l'image. Lorsque la plupart des points sont alignés vers le centre, cela signifie que la majorité des objets sont situés près du centre de l'image.
 - Quant aux scatter plots de la largeur et de la hauteur des boîtes englobantes, leur concentration dans le coin inférieur gauche suggère plusieurs choses. D'une part, cela indique que de nombreux objets détectés ont des dimensions relativement petites, ce qui pourrait signifier que le modèle tend à prédire des boîtes englobantes plus petites pour ces objets. D'autre part, cette disposition des points suggère que les objets représentés sont compacts ou petits dans l'image, comme des panneaux de signalisation ou des feux de circulation

5 Le déploiement de notre projet dans Application web :

Notre application web est une plateforme polyvalente qui offre une gamme de fonctionnalités pour les utilisateurs. Dotée d'une interface utilisateur conviviale et intuitive, elle permet aux utilisateurs de télécharger des images, de les analyser en temps réel pour détecter les objets présents, et de télécharger les images annotées résultantes. L'application intègre également une fonctionnalité de détection d'objets en temps réel à partir d'une caméra en direct, offrant ainsi une expérience immersive aux utilisateurs.

Cette application a été développée en utilisant une combinaison de technologies modernes. Le langage de balisage HTML (HyperText Markup Language) a été utilisé pour structurer le contenu de la page web, tandis que le CSS (Cascading Style Sheets) a été employé pour styliser et mettre en forme l'apparence visuelle. JavaScript a été utilisé pour rendre l'application interactive, en permettant notamment la détection d'événements tels que les clics sur les boutons et la manipulation dynamique des éléments de la page.

Pour la partie backend de l'application, nous avons utilisé Flask, un framework web léger pour Python. Flask nous a permis de développer rapidement des routes et des fonctions pour gérer les requêtes HTTP, traiter les téléchargements d'images, effectuer la détection d'objets à l'aide du modèle YOLO, et servir les résultats aux utilisateurs. L'intégration du modèle YOLO (You Only Look Once) a été cruciale pour la détection d'objets. YOLO est un modèle de vision par ordinateur qui permet de détecter et de localiser simultanément des objets dans une image en temps réel, avec une grande précision et une grande rapidité.

Grâce à l'utilisation de ces technologies et outils, nous avons pu créer une application réactive, dynamique et conviviale, offrant une expérience utilisateur optimale sur une variété de dispositifs. Que ce soit pour la détection d'objets à partir d'images téléchargées ou en temps réel à partir d'une caméra, notre application fournit une solution efficace et conviviale pour répondre aux besoins des utilisateurs.

6 Les contraintes rencontrées lors de la réalisation de ce projet :

6.1 Les contraintes liées à la phase de collection des données :

- La première contrainte rencontrée était l'indisponibilité de données personnalisées couvrant l'ensemble des classes d'objets que nous avions sélectionnées pour la détection. Cette contrainte nous a obligés à rechercher et traiter individuellement des ensembles de données pour chaque classe d'objet, en traitant les problèmes spécifiques rencontrés pour chaque ensemble de données. Ensuite, nous avons procédé à l'assemblage de ces ensembles de données individuels pour former un jeu de données complet, incluant toutes les classes cibles.
- Même avec la diversité des ressources pour les classes choisies pour la détection d'objets, nous avons rencontré un problème de format (.txt) des étiquettes de ces données, ce qui a restreint notre recherche dans les ressources trouvées.
- Aussi, lorsque nous trouvons des données pour une classe avec leurs étiquettes, ces étiquettes ne présentent que les objets de cette classe et non toutes les autres classes, ce qui a constitué un défi pour nous.

6.2 Les contraintes matérielles :

- Notre ensemble de données comprend plus de 15 000 images, nécessitant ainsi des ressources de calcul plus puissantes pour entraîner nos modèles dans des délais raisonnables, ou bien une patience accrue si nous optons pour des ressources plus modestes.
- Initialement, nous avons transféré notre ensemble de données sur Google Drive afin de procéder à l'entraînement de notre modèle sur Google Colaboratory, en exploitant les capacités de traitement offertes par les GPU de Google. Cependant, nous avons rapidement rencontré un obstacle : le temps d'expiration est intervenu dès la phase de préparation des données, ne permettant pas le démarrage effectif de l'entraînement.
- Malgré nos efforts, nous n'avons pas réussi à configurer nos propres ordinateurs pour tirer parti de leurs unités de traitement graphique (GPU). Par conséquent, nous avons dû nous contenter de l'entraînement sur les processeurs centraux (CPU) disponibles. Bien que nous ayons réussi à former le modèle sur 20 époques, cette solution a naturellement prolongé le temps nécessaire pour obtenir des résultats significatifs.
- Malgré une précision généralement satisfaisante, nous avons identifié un problème notable : dans certaines images où des individus sont présents à proximité, le modèle a tendance à classer ces instances comme des "animaux". Cette observation soulève des questions sur la capacité du modèle à discriminer précisément entre différentes classes d'objets, en particulier dans des contextes où plusieurs catégories peuvent coexister.
- L'indisponibilité d'une connexion fibre optique dans la salle de lecture de notre école ou chez nous complique les choses lors de l'utilisation du WiFi. Par exemple, pour l'importation de nos données sur le Drive a pris environ 6 heures pour s'implémenter.

7 Conclusion :

Dans notre projet, nous avons entrepris une analyse approfondie de la détection d'objets en temps réel, soulignant l'importance du prétraitement des données, du choix des modèles d'entraînement et de l'évaluation rigoureuse des performances pour obtenir des résultats efficaces.

Nous avons débuté par un prétraitement méticuleux des données, regroupant les données brutes et améliorant les annotations d'objets non détectés avec YOLO, ce qui a fourni une base solide pour l'entraînement des modèles. Dans notre exploration des modèles, nous avons examiné en détail YOLO et ResNet50, mettant en lumière leur architecture, leur utilisation dans le transfert learning, et leur intégration dans notre projet.

L'évaluation des performances des modèles a été une étape essentielle, où nous avons présenté une gamme de métriques de détection d'objets, avec une analyse approfondie de celles-ci, en mettant particulièrement l'accent sur YOLO.

Malgré le succès rencontré par YOLO dans la détection précise de toutes les classes d'objets choisies, notre expérience avec ResNet50 a souligné les défis qui persistent dans ce domaine. Ces résultats soulignent la nécessité d'une approche méthodique et adaptative dans le développement de solutions de détection d'objets en temps réel, en tenant compte des spécificités de chaque modèle et des contraintes rencontrées tout au long du processus.

En conclusion, notre projet fournit des perspectives précieuses pour des développements futurs dans le domaine de la détection d'objets en temps réel, en identifiant les défis et en proposant des pistes pour les surmonter. Ce travail jettera les bases pour des systèmes de détection d'objets plus performants et fiables, répondant aux exigences de diverses applications telles que la surveillance, l'automatisation industrielle et la conduite autonome.

8 Annexes :

Dataset animals :

<https://www.kaggle.com/datasets/antoreepjana/animals-detection-images-dataset>.

motorcycles:

<https://universe.roboflow.com/karabuk-university-hqtax/motorbike-and-helmet-detect/browse>

Data humans :

Le lien ne rend plus accessible .

Dataset traffic signs:

<https://www.kaggle.com/datasets/khaledhweij/jordanian-traffic-signs>

Dataset cars:

<https://universe.roboflow.com/cars-fjcrk/cars-detecting-and-how-many>

Data finale:

<https://drive.google.com/drive/folders/1WEnw7vNTwdTWkjRwQpLsjmhskvbgZnlt?usp=sharing>

