المدرسة العليا للتكنولوجيا
École Supérieure de Technologie
ⵜⴰⵖⵡⵍⴰⵜ ⵏ ⵜⵓⵙⴷⴰⵡⵉⵜ ⴰⵏ ⵜⵓⵙⵙⵓⵔⵜ ⴰⵎ ⵓⵍⴰⵏ

جامعة محمد الأول بوجدة
UNIVERSITE MOHAMMED PREMIER OUJDA
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵎⵓⵃⴰⵎⵎⴻⴷ ⵜⴰⵎⴻⵣⵡⴰⵔⵜ ⵏ ⵡⵊⴷⴰ

Département : génie informatique

Filière : Informatique Décisionnelle et apprentissage automatique

STAGE TECHNIQUE

Sujet :

Protection de l'environnement à base du Deep Learning

Réalisé par :

Benyamna Mohammed

Makkour Israe

Encadré par :

Pr. Mounir GRARI

Soutenue le : 03/06/2025

Devant le jury : Pr. Mounir GRARI

Pr. Badaoui Mohammed

Année universitaire 2024 - 2025

Remerciement

Nous tenons à exprimer notre profonde gratitude envers Pr. Mounir GRARI, notre encadrant, pour sa guidance experte, ses précieux conseils et son soutien indéfectible tout au long de notre projet de fin d'études portant sur la protection de l'environnement à base du Deep Learning. Sa disponibilité, sa patience et son expertise ont été des atouts inestimables qui ont grandement contribué à la réussite de ce projet. Ses retours constructifs et sa capacité à nous pousser à donner le meilleur de nous-mêmes ont été d'une grande valeur, tant sur le plan académique que personnel. Nous aimerions également remercier chaleureusement Pr. Mohammed TAHRICHI, chef de filière, pour son soutien continu et ses encouragements. Sa confiance en notre travail nous a motivés à donner le meilleur de nous-mêmes.

Enfin, nos remerciements vont à Abdelhafid CHAFI, directeur de l'école, pour son appui et son intérêt pour notre projet. Sa vision et son soutien ont été des éléments essentiels de notre parcours académique. Sa confiance en notre capacité à mener à bien ce projet nous a donné la détermination nécessaire pour surmonter les obstacles et atteindre nos objectifs.

Nous sommes profondément reconnaissants envers ces personnes qui ont joué un rôle crucial dans notre réussite académique et professionnelle. Leur soutien inébranlable et leurs conseils avisés resteront gravés dans notre mémoire et continueront à nous inspirer dans nos futures entreprises.

Liste des Figures

Figure 1: Diagramme de Gantt 8
Figure 2: Logo de Git 9
Figure 3: Logo de GitHub 9
Figure 4: Logo de VSCode 9
Figure 5:Logo de Google Colab 10
Figure 6: Logo de Postman 10
Figure 7: Logo de Python 10
Figure 8: Logo de Flask 11
Figure 9: Logo de HTML 11
Figure 10: Logo de CSS 11
Figure 11: Logo de Java Script 11
Figure 12:Logo de React 11
Figure 13: Logo de TypeScript 12
Figure 14: Logo de Tailwind CSS 12
Figure 15: Logo de Next.js 12
Figure 16: Logo de SQLite 12
Figure 17: Logo de Framer Motion 13
Figure 18: Logo de Material-UI 13
Figure 19: Logo de MapTiler 13
Figure 20:Diagramme de l'architecture de ZoraVision 18

Sommaire

Introduction 4

I. Background du projet 5

1. Vision par ordinateur (Computer Vision) 5

2. Apprentissage profond (Deep Learning) 6

3. Détection d'objet (Object Detection) 6

4. YOLO (You Only Look Once) 7

5. Cartographie 7

6. Métadonnées du Drone DJI (DJI MetaData) 7

II. Planification du projet 8

1. Phases du projet 8

2. Diagramme de Gantt – Planification du projet 8

III. Méthodologie 9

1. Outils de développement 9

2. Technologies utilisées 10

IV. Jeu de données utilisé 14

1. Données locales – Région de Saïdia 14

2. Données Kaggle 14

3. Données SEANOE – Plages Japonaises 15

4. Annotation des images 15

5. Prétraitement des données (Voire Annexes) 16

V. Résultat et discussion 18

1. Architecture et flux de navigation de ZoraVision 18

VI. Difficultés rencontrées 20

1. Problèmes d'accès aux données 20

2. Contraintes matérielles 20

3. Limitations liées à Google Colab 20

VII. Améliorations possibles 21

Conclusion 23

Bibliographie 24

Annexes 26

Introduction

La protection de l'environnement s'impose aujourd'hui comme un défi majeur, à mesure que les effets du changement climatique, de la pollution et de la dégradation des écosystèmes deviennent de plus en plus visibles. Si les approches traditionnelles de gestion et de surveillance environnementale restent indispensables, elles montrent parfois leurs limites en termes de rapidité et d'efficacité face à l'ampleur des problématiques actuelles. Dans ce contexte, l'intelligence artificielle, et notamment le deep learning, émerge comme une solution prometteuse pour renforcer la prise de décision et améliorer les stratégies de préservation de l'environnement.

Ce projet repose sur l'utilisation du deep learning et de la vision par ordinateur pour analyser des images fixes, avec pour objectif l'identification et la classification automatique des déchets. En exploitant les capacités des réseaux de neurones profonds, il devient possible de détecter, de quantifier et d'évaluer la répartition des déchets dans différents contextes environnementaux. Cette capacité d'analyse constitue un levier important pour la sensibilisation, la planification d'actions ciblées, ainsi que l'optimisation des démarches de recyclage et de traitement des déchets.

Ce rapport présente les aspects techniques de cette approche, les solutions mises en œuvre, les difficultés rencontrées, ainsi que les résultats obtenus. Il propose également une réflexion sur les perspectives d'évolution de ce type de projet, en mettant en lumière le potentiel du deep learning et de la computer vision dans la lutte contre la pollution et pour une gestion plus efficace des déchets.

I. Background du projet

1. Vision par ordinateur (Computer Vision)

La vision par ordinateur (Computer Vision) est un domaine de l'intelligence artificielle qui permet aux machines d'analyser, d'interpréter et de comprendre des images et des vidéos. Elle s'appuie sur des techniques de traitement d'images et de deep learning pour extraire des informations visuelles. Ce domaine englobe plusieurs sous-domaines clés, notamment :

- La reconnaissance et classification d'images (identification de catégories d'objets ou de scènes),
- La segmentation d'images (partitionnement d'une image en régions significatives),
- La génération d'images (création de contenu visuel par des modèles),
- Le suivi et comptage d'objets (analyse de mouvement et dénombrement dans des séquences vidéo),
- La détection d'objets (localisation et identification d'objets dans une image).

Dans le cadre de notre projet, nous nous focalisons sur la détection d'objets, couplée à des techniques de suivi et comptage d'objets dans des flux vidéo. Cette approche permet d'automatiser l'identification et le traçage de déchets dans un environnement donné.

2. Apprentissage profond (Deep Learning)

L'apprentissage profond (Deep Learning) est une branche de l'intelligence artificielle qui repose sur des **réseaux de neurones artificiels** pour analyser et apprendre des modèles complexes à partir de grandes quantités de données. Il est particulièrement efficace pour les tâches liées aux images et vidéos, comme la classification, la reconnaissance et la détection d'objets.

Les réseaux de neurones convolutionnels (CNNs - Convolutional Neural Networks) sont couramment utilisés en vision par ordinateur, car ils permettent d'extraire automatiquement des caractéristiques pertinentes des images. Grâce à cette approche, les modèles de deep learning surpassent les méthodes classiques en termes de précision et d'adaptabilité, ce qui les rend essentiels pour des applications comme la détection et le suivi d'objets.

3. Détection d'objet (Object Detection)

La détection d'objets (Object Detection) consiste à identifier et localiser des objets spécifiques dans une image ou une vidéo. Contrairement à la simple classification, elle permet non seulement de reconnaître un objet, mais aussi de le situer précisément grâce à une boîte englobante (bounding box).

Il existe plusieurs méthodes pour la détection d'objets, parmi lesquelles :

- Les approches classiques utilisant des descripteurs de caractéristiques (HOG, SIFT).
- Les modèles basés sur le deep learning, comme Faster R-CNN, SSD et YOLO.

YOLO est particulièrement adapté aux applications nécessitant une analyse en temps réel, ce qui en fait un choix privilégié pour notre projet.

4. YOLO (You Only Look Once)

YOLO (You Only Look Once) est une architecture avancée de détection d'objets qui permet d'analyser une image en une seule passe à travers le réseau de neurones. Contrairement aux approches plus lentes qui segmentent l'image en plusieurs parties, YOLO traite l'ensemble de l'image simultanément, offrant ainsi une grande rapidité et une précision élevée.

Ses principaux avantages sont :

- Rapidité : traitement en temps réel des vidéos.
- Précision : bon équilibre entre vitesse et exactitude.
- Simplicité : détection efficace en une seule phase.

YOLO est largement utilisé pour des tâches comme la surveillance, la reconnaissance d'objets et, dans notre cas, l'identification et le suivi des déchets.

5. Cartographie

La cartographie est un outil essentiel de représentation visuelle permettant de traduire des données géographiques sous forme de cartes. Elle facilite la compréhension de phénomènes spatiaux, l'analyse territoriale et la prise de décision dans divers domaines comme l'environnement, l'urbanisme ou la gestion des ressources.

6. Métadonnées du Drone DJI (DJI MetaData)

Les drones DJI enregistrent, en plus des images ou vidéos capturées, un ensemble de métadonnées telles que la position GPS, l'altitude, l'orientation, la vitesse, et la date/heure de prise. Ces informations enrichissent les données visuelles et permettent une exploitation précise à des fins d'analyse, de cartographie ou de suivi spatio-temporel.

II. Planification du projet

1. Phases du projet

Phase	Tâches principales	Durée estimée	Dépendances
Phase 1 : Définition des besoins et objectifs	-Identifier les besoins fonctionnels et techniques	1 semaine	-
Phase 2 : Travail sue la version 2 de la Data	-Chercher la dataset sur internet (Kagel) -L'annotation de la dataset sur ZoraVision	1 semaine	Phase 1
Phase 3 : Entrainement du model	-Entrainer le model sur la dataset V1 et V2 -Tester le model -Visualiser les résultat	1 semaine	Phase1 - Phase 2
Phase 4 : Travail sur la version 3 de la Data	-Chercher la dataset sur internet (SEANOE) -L'annotation de la dataset sue ZoraVision	1 semaine	Phase 1
Phase 5 : Entrainement du model	-Entrainer le model sur la dataset V3 -Tester le model -Visualiser les résultats	1 semaine	Phase 1 – Phase 4
Phase 6 : Développement de l'application	-Développement du front-end et du back-end de l'application	1 – 2 semaines	Phase 1

2. Diagramme de Gantt – Planification du projet

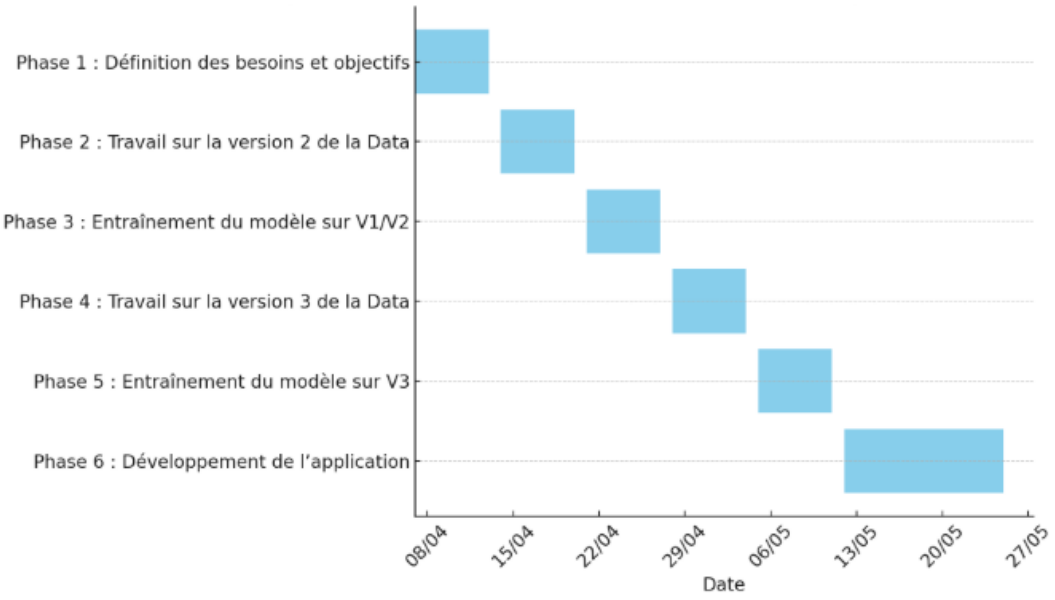

Figure 1: Diagramme de Gantt

III. Méthodologie

1. Outils de développement

Figure 2: Logo de Git

Figure 3: Logo de GitHub

- **Git / GitHub** : Git est un système de gestion de versions permettant de suivre les modifications du code. GitHub est une plateforme qui facilite la collaboration et l'hébergement des projets Git.
- **Visual Studio Code** : Un éditeur de code puissant et léger, avec des extensions pour divers langages comme Python, JavaScript, HTML, etc.

Figure 4: Logo de VSCode

- **Google Colab** : Un environnement de développement basé sur le cloud, permettant d'exécuter du code Python directement dans le navigateur. Il est particulièrement adapté aux projets de data science et de Deep Learning, grâce à son accès gratuit à des ressources matérielles puissantes comme les GPU. Colab facilite le partage de notebooks, l'expérimentation rapide et la visualisation des résultats dans un format interactif.

Figure 5:Logo de Google Colab

- **Postman** : Un outil de test d'API qui permet d'envoyer facilement des requêtes HTTP et d'analyser les réponses. Il est largement utilisé pour vérifier le bon fonctionnement des endpoints d'une API, tester différents scénarios, et automatiser des tests. Postman offre une interface conviviale pour simuler les interactions entre le front-end et le back-end sans avoir à coder manuellement ces appels.

Figure 6: Logo de Postman

2. Technologies utilisées

- **Python** : Langage de programmation utilisé pour le backend de l'application. Il est simple, puissant et dispose de nombreuses bibliothèques.

Figure 7: Logo de Python

- **Flask** : Un micro-framework Python permettant de créer des applications web rapidement et efficacement.

Figure 8: Logo de Flask

- **HTML / CSS / JavaScript** : Les langages essentiels du développement web. HTML structure les pages, CSS les stylise et JavaScript les rend interactives.

Figure 9: Logo de HTML

Figure 10: Logo de CSS

Figure 11: Logo de Java Script

- **React** : Bibliothèque JavaScript dédiée à la création d'interfaces utilisateur interactives et modulaires, permettant un développement efficace basé sur des composants réutilisables.

Figure 12:Logo de React

- **TypeScript** : Langage basé sur JavaScript, avec un système de typage statique, qui améliore la fiabilité du code, facilite la détection des erreurs et rend les projets plus maintenables à long terme.

Figure 13: Logo de TypeScript

- **Tailwind CSS** : Framework CSS utilitaire qui permet de styliser rapidement des interfaces en utilisant des classes prédéfinies directement dans le HTML ou le JSX, favorisant une grande flexibilité et rapidité de développement.

Figure 14: Logo de Tailwind CSS

- **Next.js** : Framework basé sur React qui facilite la création d'applications web performantes avec des fonctionnalités comme le rendu côté serveur (SSR), le pré-rendu statique (SSG) et le routage simplifié.

Figure 15: Logo de Next.js

- **SQLite** : Base de données relationnelle légère et intégrée, idéale pour des projets de petite à moyenne envergure. Elle ne nécessite pas de serveur indépendant et stocke les données dans un simple fichier.

Figure 16: Logo de SQLite

- **Framer Motion** : Bibliothèque d'animation pour React qui permet de créer des transitions fluides et interactives avec une syntaxe simple et expressive.

Figure 17: Logo de Framer Motion

- **Material UI** : Bibliothèque de composants React basée sur les principes du design Material de Google. Elle permet de créer rapidement des interfaces utilisateur modernes et cohérentes.

Figure 18: Logo de Material-UI

- **MapTiler** : Plateforme qui permet de créer, personnaliser et héberger des cartes interactives. Elle est utilisée pour intégrer facilement des cartes dans des sites web ou des applications.

Figure 19: Logo de MapTiler

IV. Jeu de données utilisé

Au cours de notre projet, nous avons utilisé trois versions de jeux de données différentes. Chacune d'entre elles a présenté des avantages mais aussi des contraintes spécifiques, ce qui nous a permis d'explorer plusieurs situations réelles de détection de déchets.

1. Données locales – Région de Saïdia

Ce jeu de données a été produit dans le cadre de notre projet, à l'aide d'un drone DJI ayant survolé plusieurs plages de la région de Saïdia.

- Nombre d'images : plus de 300.
- Méthode de capture : les images ont été prises à une hauteur d'environ 10 mètres.
- Avantages : données réelles et contextualisées, capturées spécifiquement pour ce projet.
- Inconvénients : la hauteur d'acquisition a entraîné une perte de qualité visuelle, rendant la détection des déchets parfois difficile, en particulier pour les petits objets.

2. Données Kaggle

Ce dataset a été téléchargé depuis la plateforme Kaggle.

- Taille : moyenne.
- Contenu : images représentant plusieurs types de déchets.
- Avantages : accessible facilement.

- Inconvénients :
    - La qualité globale des images était moyenne, avec parfois une faible résolution.
    - Plusieurs images contenaient des zones totalement noires (pixels noirs), ce qui a perturbé l'étape d'annotation automatique et réduit la qualité de certains apprentissages.

3. Données SEANOE – Plages Japonaises

Ce dataset a été récupéré à partir de la plateforme scientifique SEANOE, qui fournit des données ouvertes sur les environnements marins.

- Nombre d'images : plus de 5000.
- Origine : plages du Japon.
- Traitement : deux dossiers séparés ont été fusionnés pour obtenir un seul jeu de données complet.
- Avantages : très bonne diversité des images, grand volume de données.
- Inconvénients : la différence de nature entre les plages japonaises et celles de notre contexte local a parfois affecté la cohérence de la classification, mais cet écart est resté relativement gérable.

4. Annotation des images

Lors de la phase de préparation des données, l'annotation des images a représenté une étape déterminante dans le développement de notre projet. En effet, la qualité de l'annotation joue un rôle fondamental dans la performance des modèles d'apprentissage automatique, notamment pour les tâches de classification et de détection. Pour répondre à ce besoin, nous avons opté pour l'application ZoraVision, un outil d'annotation simple, intuitif et adapté à notre contexte.

Afin de faciliter son accès à l'ensemble de l'équipe, nous avons procédé à la création d'une version exécutable et téléchargeable de l'application. Cette initiative a permis aux sept membres de l'équipe de stage de l'utiliser facilement, quel que soit leur poste de travail, sans avoir à configurer manuellement l'environnement de développement. Ainsi, chaque membre pouvait contribuer efficacement à l'annotation des images, tout en respectant une nomenclature et une structuration de données communes, établies en amont pour garantir la cohérence des annotations.

Cette étape collaborative a non seulement amélioré la productivité de l'équipe, mais elle a également assuré une qualité homogène des données annotées, condition indispensable pour l'entraînement et l'évaluation rigoureuse des modèles.

5. Prétraitement des données (Voire Annexes)

Nettoyage des images invalides : Une première étape essentielle du prétraitement a consisté à nettoyer le jeu de données en supprimant les images corrompues ou invalides, ainsi que leurs annotations associées. Un script a été utilisé pour parcourir toutes les images listées dans le fichier COCO, en vérifiant leur accessibilité et la validité de leurs dimensions (largeur et hauteur supérieures à zéro). Les images ne respectant pas ces critères ont été automatiquement supprimées du dossier, et leurs identifiants ont été enregistrés afin d'exclure également les annotations correspondantes. À la fin de ce processus, un nouveau fichier COCO nettoyé, intitulé cocofinal_cleaned.json, a été généré, garantissant ainsi l'intégrité et la cohérence des données d'entrée utilisées pour l'entraînement.

Renommage des images et mise à jour des annotations : Dans une deuxième phase du prétraitement, un script a été mis en place pour uniformiser le nommage des images et mettre à jour les annotations COCO en conséquence. Chaque image a été renommée de manière systématique selon un format standard (image_001.jpg, image_002.jpg, etc.) afin de garantir une meilleure organisation

et d’éviter tout conflit de nommage ultérieur. Les noms des fichiers dans le dictionnaire images du fichier JSON ont également été modifiés pour correspondre aux nouveaux noms. Ce processus a permis de générer un fichier d’annotations à jour (updated_annotations.json) contenant les mêmes données, mais avec des noms d’images cohérents et normalisés.

Conversion en format YOLO : Enfin, une étape de conversion a été réalisée afin de transformer les annotations du format COCO vers le format YOLOv8, requis pour l'entraînement du modèle avec Ultralytics YOLO. Ce processus a impliqué l’extraction des informations de bounding boxes et leur conversion vers le format attendu par YOLO (classe, x_centre, y_centre, largeur, hauteur), toutes exprimées sous forme normalisée (valeurs entre 0 et 1). Un répertoire a été créé pour chaque image contenant un fichier .txt décrivant ses annotations, ainsi qu’un fichier classes.txt listant toutes les classes du jeu de données. Cette conversion a permis d’adapter efficacement les données à la structure attendue par le modèle YOLOv8.

V. Résultat et discussion

L'application web développée, intitulée ZoraVision, permet l'analyse automatisée d'une zone spécifique à partir d'images importées par l'utilisateur. Grâce à un modèle d'intelligence artificielle entraîné préalablement, l'application est en mesure d'identifier et de localiser les déchets présents dans les images fournies.

Par la suite, des cartes thermiques sont générées afin de visualiser les zones de forte concentration de déchets. Cette représentation facilite le processus de collecte en le rendant plus ciblé et plus efficace, tout en soutenant la prise de décision pour les actions de nettoyage.

ZoraVision constitue ainsi un outil pertinent pour améliorer la gestion des déchets dans des environnements côtiers, en combinant automatisation, précision et accessibilité.

1. Architecture et flux de navigation de ZoraVision

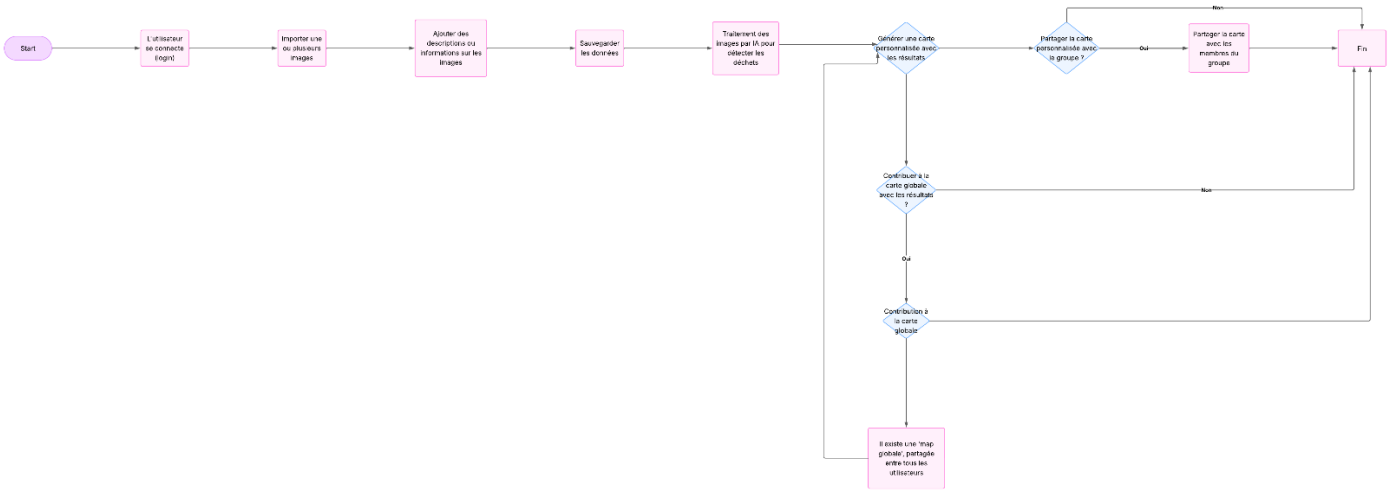

Figure 20:Diagramme de l'architecture de ZoraVision

Ce diagramme illustre le parcours typique d'un utilisateur au sein de l'application de détection de déchets. Après s'être connecté, l'utilisateur importe des images, y ajoute des informations pertinentes, puis sauvegarde l'ensemble. Un modèle d'intelligence artificielle se déclenche automatiquement pour analyser les images et détecter les déchets. À la fin du traitement, une carte

personnalisée est générée pour l'utilisateur, avec possibilité de la partager avec son groupe. En parallèle, l'utilisateur peut également contribuer à une carte globale, accessible à l'ensemble de la communauté de l'application.

VI. Difficultés rencontrées

1. Problèmes d'accès aux données

L'un des défis majeurs du projet a été l'obtention de données adaptées. Le projet nécessitait des images prises spécifiquement sur les plages de la région de Saïdia afin d'assurer une cohérence avec le contexte environnemental local. Cependant, l'accès à un volume suffisant d'images représentatives s'est révélé difficile. Le manque de diversité et la qualité parfois réduite des images disponibles ont limité l'efficacité de certaines phases du traitement et de l'entraînement du modèle

2. Contraintes matérielles

L'entraînement d'un modèle d'apprentissage profond requiert des ressources matérielles importantes, notamment des GPU performants. Dans le cadre de ce projet, les machines disponibles localement ne disposaient pas de la puissance nécessaire pour exécuter les tâches de calcul intensif liées à l'apprentissage du modèle. Cette contrainte matérielle a limité les possibilités de développement en local, nécessitant le recours à des plateformes cloud.

3. Limitations liées à Google Colab

Pour pallier l'absence d'infrastructure locale adaptée, l'environnement Google Colab a été utilisé pour l'entraînement du modèle. Bien que cette solution ait permis d'accéder à des ressources GPU gratuitement, elle présentait des limitations non négligeables. Les sessions Colab étant limitées dans le temps, de nombreuses interruptions ont été constatées, rendant le processus d'entraînement instable et nécessitant de multiples relances. Cela a entraîné une perte de temps et une complexification du suivi des expérimentations.

VII. Améliorations possibles

Pour améliorer davantage les fonctionnalités de l'application et favoriser une utilisation plus large et collaborative, plusieurs évolutions peuvent être envisagées. Tout d'abord, l'ajout de niveaux d'accès utilisateurs permettrait une gestion plus fine des droits et des responsabilités. En définissant des rôles tels qu'administrateur, collaborateur et lecteur, il serait possible de sécuriser les contributions et d'organiser les interactions. Par exemple, seuls les administrateurs pourraient valider ou modifier les données de la carte globale, tandis que les collaborateurs pourraient proposer des annotations ou des ajouts, et les lecteurs se limiteraient à la consultation.

Ensuite, l'intégration d'un système de commentaires ou de messagerie interne enrichirait l'aspect collaboratif de l'application. Lors du partage de cartes entre plusieurs membres ou groupes, il serait utile de pouvoir échanger directement sur les observations : poser des questions, proposer des modifications, ou simplement discuter des détections effectuées. Ce type de fonctionnalité favoriserait la communication entre les utilisateurs et renforcerait l'aspect communautaire du projet.

Par ailleurs, l'amélioration du système d'annotation grâce à l'utilisation de la segmentation sémantique ou instance à la place des simples boîtes englobantes (bounding boxes) représenterait un réel gain en précision. En effet, la segmentation permet de délimiter plus finement les contours des objets annotés (par exemple les déchets), ce qui est particulièrement utile dans des environnements complexes ou encombrés. Cela offrirait non seulement une annotation plus fidèle à la réalité, mais aussi des données de meilleure qualité pour l'entraînement des modèles.

Enfin, le développement d'un tableau de bord statistique permettrait de visualiser et d'analyser les données collectées de manière plus synthétique et

parlante. On pourrait y suivre l'évolution du volume de déchets détectés dans le temps, identifier les zones les plus touchées, ou encore observer la répartition des types de déchets. Ces représentations graphiques offriraient une meilleure compréhension de la situation environnementale et pourraient servir de base à des actions de sensibilisation ou de planification d'interventions ciblées.

Conclusion

Le développement de ce projet ne s'est pas limité à la simple création d'un outil technique : il a été le reflet d'une démarche humaine, citoyenne et profondément engagée envers les défis environnementaux actuels. En combinant les avancées de l'intelligence artificielle avec la puissance de la géolocalisation et de la visualisation cartographique, nous avons conçu une application capable non seulement de détecter automatiquement les déchets dans les images, mais aussi de les situer précisément dans l'espace. Cela ouvre la voie à une nouvelle manière de sensibiliser, de mobiliser et de passer à l'action sur le terrain.

Derrière chaque script exécuté et chaque modèle entraîné, il y avait une volonté forte : rendre l'invisible visible, transformer des images en informations exploitables, puis ces informations en outils de prise de décision. Ce projet a représenté bien plus qu'un simple travail académique ; il a été une expérience complète, qui nous a confrontés à la réalité des données imparfaites, aux contraintes matérielles, et aux limites techniques d'un environnement comme Google Colab. Chaque difficulté rencontrée a été une opportunité d'apprentissage, nous poussant à adapter nos méthodes, à hiérarchiser nos priorités et à rester concentrés sur l'essentiel.

Au-delà de l'aspect technique, ce projet a renforcé notre compréhension de l'impact que peuvent avoir les technologies bien utilisées. Il nous a permis de mettre en pratique des compétences variées — traitement d'images, annotation, conception d'interface, structuration de données — tout en développant notre autonomie, notre capacité à résoudre des problèmes concrets et notre esprit critique face aux choix techniques.

Cette application constitue une première étape solide dans une démarche d'amélioration continue. Nous avons d'ores et déjà identifié plusieurs pistes d'évolution, telles que la gestion des rôles utilisateurs, l'intégration d'une messagerie collaborative et la mise en place d'un tableau de bord statistique pour mieux analyser les données collectées. En ce sens, ce projet ne marque pas une fin, mais bien le commencement d'une aventure plus vaste, tournée vers l'innovation, l'impact et la contribution active à la protection de notre environnement.

Nous en sortons enrichis, fiers du chemin parcouru et conscients du potentiel immense que recèle la technologie lorsqu'elle est mise au service d'une cause juste.

Bibliographie

Technologies utilisées :

• **Python :** Langage de programmation utilisé pour le développement du back-end et des scripts d'analyse.
→ https://www.python.org/

• **Flask :** Micro-framework Python utilisé pour créer des API REST et des applications web légères.
→ https://flask.palletsprojects.com/

• **React :** Bibliothèque JavaScript dédiée au développement d'interfaces utilisateurs dynamiques.
→ https://react.dev/

• **TypeScript :** Surensemble typé de JavaScript, utilisé pour améliorer la robustesse et la maintenabilité du code.
→ https://www.typescriptlang.org/

• **Next.js :** Framework basé sur React, facilitant le rendu côté serveur et la génération de sites statiques.
→ https://nextjs.org/

• **Tailwind CSS :** Framework CSS utilitaire pour créer des interfaces modernes rapidement.
→ https://tailwindcss.com/

• **HTML / CSS / JavaScript :** Technologies de base du développement web, utilisées pour la structure, le style et l'interactivité des pages.
→ https://developer.mozilla.org/fr/docs/Web

• **SQLite :** Système de gestion de base de données relationnelle intégré, léger et simple à utiliser.
→ https://www.sqlite.org/

• **Framer Motion :** Bibliothèque d'animations pour React, utilisée pour des transitions fluides et interactives.
→ https://www.framer.com/motion/

• **Google Colab :** Environnement de développement cloud permettant d'exécuter des notebooks Python avec GPU/TPU.

→ https://colab.research.google.com/

• **Postman :** Outil de test et de simulation d'API pour vérifier les échanges entre client et serveur.

→ https://www.postman.com/

• **Git :** Système de gestion de versions distribué, utilisé pour suivre les modifications du code source.

→ https://git-scm.com/

• **GitHub :** Plateforme collaborative pour héberger, gérer et partager du code avec Git.

→ https://github.com/

• **Visual Studio Code (VS Code) :** Éditeur de code multiplateforme utilisé pour développer, tester et déboguer le projet.

→ https://code.visualstudio.com/

- Dépôt de code : Ensemble de codes constituants l'application Zoravision

→ https://github.com/NEREUScode/zoravision-react.git

Annexes

Introduction…… 4

I. Background du projet……..5

II. Planification du projet……8

III. Méthodologie……9

IV. Jeu de données utilisées……Error! Bookmark not defined.

V. Résultat et discussion……Error! Bookmark not defined.**8**

VI. Difficultés encontrées……Error! Bookmark not defined.

VII. Améliorations possibles ……..21

Conclusion……..23

Rapport technique de ZoraAnnotator

Rapport des métriques des jeu de données