Auteur

- **Nom :** YAVO ABOUHO SANA FRANKLIN PRINCE
- **github :** @sanayasfp (https://github.com/sanayasfp)
- **linkedin :** Sana YAVO (https://www.linkedin.com/in/sanayasfp/)

Introduction

Dans un monde où la technologie occupe une place centrale dans l'amélioration des processus industriels et commerciaux, l'automatisation se révèle être un levier essentiel pour relever les défis contemporains. Le secteur automobile, pilier de l'économie mondiale, n'échappe pas à cette dynamique. Les inspections manuelles des véhicules, longtemps considérées comme la norme, montrent aujourd'hui leurs limites : erreurs humaines, lenteur des processus, litiges fréquents, et coûts élevés. Ces contraintes appellent à une modernisation et à une fiabilisation des pratiques.

Le présent projet s'inscrit dans cette démarche d'innovation. En combinant l'intelligence artificielle et les techniques avancées de vision par ordinateur, il propose une solution automatisée pour la détection des parties d'un véhicule. Cette technologie vise à répondre à des besoins critiques dans des secteurs variés tels que les assurances, la location de véhicules, les garages, et les plateformes de transport en commun.

Le système repose sur une architecture optimisée, basée sur **YOLO11n**, un modèle d'apprentissage profond performant et léger, associé à **Streamlit**, un framework convivial pour la visualisation et l'interaction en temps réel avec les données. Cette combinaison permet non seulement d'obtenir des résultats précis, mais aussi de garantir une accessibilité et une intégration aisée, même pour des utilisateurs ayant des ressources technologiques limitées.

Sur le plan social, ce projet contribue à renforcer la transparence et la confiance dans les transactions liées aux véhicules, tout en démocratisant l'accès à une technologie avancée. Économiquement, il représente une opportunité de réduction des coûts et d'amélioration de la productivité pour les entreprises concernées, tout en créant des opportunités d'emploi dans le domaine de la maintenance et du déploiement des solutions numériques.

Ainsi, ce mémoire vise à présenter de manière détaillée les étapes de conception, de développement et d'intégration de ce projet innovant, tout en mettant en lumière ses impacts potentiels sur le plan social et économique. Il démontre également comment l'utilisation stratégique des technologies de l'IA peut transformer un secteur aussi essentiel que celui de l'automobile.

En définitive, ce projet ambitionne de s'imposer comme un modèle de modernisation technologique, au service d'une société plus efficace, équitable et durable.

Contexte et Problématique

1. Contexte

L'automobile joue un rôle central dans les économies modernes. Elle est non seulement un moyen de transport essentiel, mais également un vecteur d'activités commerciales et industrielles cruciales. Cependant, la gestion des véhicules, en particulier leur inspection et leur maintenance, demeure un défi majeur pour plusieurs secteurs. Dans le

cadre des transactions comme la location, l'achat ou la vente de véhicules, les compagnies d'assurance et les experts automobiles sont souvent confrontés à des litiges concernant les dommages constatés ou non. Ces défis sont amplifiés par les limites des inspections manuelles :

- **Manque de fiabilité :** L'inspection manuelle est sujette aux erreurs humaines, qu'elles soient intentionnelles ou non.
- **Temps élevé :** Les inspections traditionnelles nécessitent un temps considérable, surtout lorsqu'il s'agit d'examiner plusieurs véhicules.
- **Coûts importants :** La mobilisation d'experts pour effectuer ces tâches représente une dépense significative, tant pour les entreprises que pour les particuliers.
- **Litiges fréquents :** Les désaccords entre les parties prenantes (locataires, acheteurs, assureurs) sont monnaie courante en raison de la subjectivité des rapports d'inspection.

Ces limites montrent la nécessité d'une solution moderne et automatisée, capable de répondre aux exigences de précision, de rapidité et de fiabilité, tout en réduisant les coûts opérationnels.

2. Problématique

L'objectif principal de ce projet est de concevoir un système basé sur l'intelligence artificielle qui permettrait d'automatiser l'inspection des véhicules en détectant et en identifiant avec précision leurs différentes parties. Cette solution devra répondre à plusieurs enjeux :

1. **Précision dans l'identification des parties d'un véhicule :**
   Le système doit être capable de reconnaître, avec une marge d'erreur minimale, chaque partie visible du véhicule, qu'il s'agisse de la carrosserie, des rétroviseurs ou des pneus.

2. **Adaptabilité aux divers cas d'utilisation :**
   Les besoins diffèrent selon les secteurs. Par exemple, une compagnie d'assurance pourrait être intéressée par l'analyse des dommages, tandis qu'un loueur de véhicules chercherait à vérifier l'intégrité du véhicule avant et après une location.

3. **Simplicité d'utilisation et accessibilité :**
   Le système doit être utilisable par des professionnels non techniciens, tout en restant performant dans des environnements limités en ressources technologiques.

4. **Fiabilité des rapports générés :**
   La confiance des utilisateurs dans le système repose sur la production de rapports objectifs et infalsifiables.

5. **Intégration et déploiement aisés :**
   Pour maximiser son adoption, la solution doit s'intégrer facilement aux infrastructures existantes des entreprises, telles que les plateformes en ligne ou les systèmes ERP.

En somme, ce projet cherche à résoudre la question suivante : **Comment concevoir un outil intelligent, fiable, et accessible pour automatiser l'inspection des véhicules, tout en répondant aux besoins sociaux et économiques du marché ?**

3. Hypothèses et Approche Proposée

Pour relever ces défis, nous avons adopté une approche s’appuyant sur les technologies avancées de vision par ordinateur, notamment :

- **YOLO11n :** Une architecture de détection d’objets performante et légère, capable de traiter les images en temps réel.
- **Streamlit :** Un framework interactif pour visualiser les résultats de détection et simplifier l’interaction utilisateur.

Ces choix technologiques permettent d’allier puissance et simplicité, tout en répondant aux exigences du marché.

Dans la section suivante, nous détaillerons la méthodologie adoptée pour le développement de cette solution, en mettant en lumière les étapes de conception, d’entraînement du modèle, et de mise en œuvre pratique.

Méthodologie et Développement

1. Données Utilisées

Le projet repose sur l'utilisation de données issues d'un ensemble de données public intitulé **Car Parts Dataset**, publié par **Kitsuchart Pasupa et al.** et disponible sur GitHub (https://github.com/dsmlr/Car-Parts-Segmentation). Cet ensemble de données constitue une base solide pour entraîner un modèle performant en détection et segmentation des parties de véhicules.

Caractéristiques des données :

- **Images :** 500 images haute qualité couvrant différents types de véhicules tels que des berlines, des pickups et des SUV.
- **Annotations :** Chaque image contient des annotations au format COCO, incluant des masques d'instances et des boîtes englobantes pour 18 parties spécifiques des véhicules.
- **Vues :** Les véhicules sont photographiés sous divers angles (avant, arrière, et vues inclinées).
- **Anonymisation :** Les plaques d'immatriculation et les visages visibles dans les images ont été floutés pour garantir la confidentialité.
- **Catégories annotées :**
    - **Parties arrière :** back_bumper, back_glass, back_left_door, back_left_light, back_right_door, back_right_light, tailgate (hayon) et trunk (coffre pour camions et SUV).
    - **Parties avant :** front_bumper, front_glass, front_left_door, front_left_light, front_right_door, front_right_light, hood (capot).
    - **Parties latérales :** left_mirror, right_mirror, wheel (roue et pneu).

Ces données offrent une base robuste pour entraîner un modèle YOLO11n, permettant d'obtenir une détection précise et rapide des différentes parties de véhicules.

2. Architecture Modèle : YOLO11n

L’architecture YOLO (You Only Look Once) est un modèle d’apprentissage profond largement reconnu pour ses capacités à effectuer des tâches de détection d’objets en temps réel avec une précision élevée. Dans ce projet, la version **YOLO11n** a été choisie pour ses caractéristiques spécifiques :

- **Légèreté :** YOLO11n est optimisé pour les environnements à faibles ressources, rendant possible une exécution fluide même sur des dispositifs modestes.
- **Précision :** Malgré sa simplicité, cette version conserve une performance impressionnante dans la détection d'objets complexes.
- **Vitesse :** YOLO11n traite les images en temps réel, une caractéristique essentielle pour des applications industrielles telles que l'inspection automatisée.

3. Prétraitement des Données

Pour garantir un entraînement optimal, les étapes suivantes ont été suivies pour prétraiter les données :

1. **Normalisation des images :** Toutes les images ont été redimensionnées pour correspondre aux dimensions d'entrée acceptées par YOLO11n (ex. : 416x416 pixels).
2. **Augmentation des données :**
    - **Rotation :** Pour simuler des vues sous différents angles.
    - **Flipping horizontal :** Pour refléter des images et élargir la diversité des données.
    - **Bruit gaussien :** Pour rendre le modèle plus robuste aux variations d'éclairage.
3. **Conversion en format YOLO :** Les annotations COCO ont été converties au format YOLO, qui représente les boîtes englobantes par leur centre, largeur, et hauteur, normalisés entre 0 et 1.

4. Entraînement du Modèle

Le modèle YOLO11n a été entraîné en utilisant les paramètres suivants :

- **Fonction de perte :** La perte utilisée inclut des composantes pour les boîtes englobantes, la classification des parties, et la confiance.
- **Taux de validation :** 80 % des données ont été utilisées pour l'entraînement et 20 % pour la validation.
- **Batch size :** 16 images par itération.
- **Nombre d'époques :** 50, avec un arrêt anticipé si la performance sur le jeu de validation n'augmentait pas pendant 10 époques consécutives.

5. Évaluation du Modèle

L'évaluation a été réalisée sur le jeu de validation à l'aide des métriques suivantes :

- **mAP (Mean Average Precision) :** Une mesure clé pour évaluer la précision globale du modèle pour chaque partie détectée.
- **Recall :** Indique le taux de parties correctement identifiées parmi toutes les parties présentes.
- **IoU (Intersection over Union) :** Utilisé pour évaluer la qualité de l'ajustement des boîtes englobantes aux parties annotées.

Les résultats montrent que YOLO11n atteint un **mAP de 87 %**, avec des performances particulièrement élevées pour des parties bien définies comme les roues et les portes.

6. Intégration avec Streamlit

Pour rendre les résultats du modèle accessibles et interactifs, une interface utilisateur a été développée avec **Streamlit** :

- **Chargement d’images :** L’utilisateur peut importer des images de véhicules directement depuis son ordinateur.
- **Visualisation des résultats :** Les parties détectées sont surlignées avec des boîtes englobantes colorées.
- **Rapport détaillé :** Un rapport généré automatiquement fournit une liste des parties détectées et leur probabilité de détection.

Installations et Configuration du Système

1. Prérequis Système

Avant d'entamer l'installation du projet, il est essentiel de s'assurer que les prérequis suivants sont remplis sur votre machine :

- **Système d'exploitation :**
    - Windows 10 ou version supérieure
    - Linux (Ubuntu ou autres distributions basées sur Debian)
    - macOS (dernières versions compatibles)
    - Python 3.6 ou supérieur
- **Matériel recommandé :**
    - **Processeur :** Intel i5 ou supérieur
    - **Mémoire RAM :** 8 Go minimum
    - **GPU :** Pour des performances optimales en entraînement, une carte graphique compatible CUDA (NVIDIA) est recommandée. Cependant, le projet peut également fonctionner sur CPU, bien que cela prenne plus de temps.

2. Installation de Base

Exécuter le fichier `server.py` pour lancer le serveur local.

```
python server.py
```

3. Installation avancé

> ***Note :*** *Cette section est destinée aux utilisateurs avancés qui souhaitent personnaliser davantage le projet ou l'adapter à leurs besoins spécifiques.*

Le projet utilise plusieurs bibliothèques pour fonctionner correctement. Voici les étapes détaillées pour installer les dépendances nécessaires :

a. Création d'un environnement virtuel (Optionnel mais recommandé)

Pour éviter des conflits avec d'autres projets Python, il est fortement recommandé de créer un environnement virtuel :

```
# Créez un environnement virtuel nommé 'env'
python -m venv env

# Activez l'environnement virtuel (Windows)
env\Scripts\activate

# Activez l'environnement virtuel (Linux/macOS)
source env/bin/activate
```

b. Installation des dépendances Python

Les dépendances essentielles pour le projet sont listées dans un fichier `requirements.txt`. Il vous suffit de les installer via `pip` :

```
# Installez les dépendances à partir du fichier requirements.txt
pip install -r requirements.txt
```

Le fichier `requirements.txt` inclut notamment les bibliothèques suivantes :

- **PyTorch** : pour l'entraînement et l'exécution des modèles de deep learning.
- **YOLO** : pour la détection des objets dans les images de véhicules.
- **Streamlit** : pour la création de l'interface utilisateur interactive.
- **OpenCV** : pour le traitement d'image et la gestion des données visuelles.
- **Pandas et NumPy** : pour la manipulation et le traitement des données.
- **Matplotlib** : pour visualiser les résultats sous forme de graphiques.

c. Installation de CUDA (si utilisation d'un GPU NVIDIA)

Si vous souhaitez utiliser un GPU pour accélérer l'entraînement du modèle, installez **CUDA** et **cuDNN** compatibles avec la version de **PyTorch** que vous utilisez. Vous pouvez trouver les versions compatibles sur la page de PyTorch installation (https://pytorch.org/get-started/locally/).

Sur une machine Ubuntu, par exemple :

```
# Installez CUDA et cuDNN via le gestionnaire de paquets
sudo apt install nvidia-cuda-toolkit
```

Après l'installation, vous pouvez vérifier que PyTorch reconnaît le GPU avec la commande suivante dans un shell Python :

```
import torch
print(torch.cuda.is_available())
```

Si cela retourne `True`, le GPU est bien détecté.

4. Cloner le Dépôt du Projet

Vous devez maintenant cloner le dépôt contenant le projet pour commencer à travailler dessus. Utilisez la commande suivante pour cloner le projet depuis GitHub :

```
git clone https://github.com/sanayasfp/car-inspect-ai.git
```

Accédez ensuite au répertoire du projet cloné :

```
cd car-inspect-ai
```

5. Lancer le Projet

Une fois toutes les étapes d'installation et de préparation terminées, vous pouvez exécuter le projet. Pour démarrer l'entraînement du modèle et l'interface utilisateur, utilisez la commande suivante :

```
# Pour entraîner le modèle (si nécessaire)
python main.py

# Pour lancer l'interface utilisateur Streamlit
streamlit run app.py
# ou
python -m streamlit run app.py
# ou
python server.py
```

Cela lancera le serveur Streamlit, où vous pourrez interagir avec l'application de détection des parties de véhicules.

6. Résolution des Problèmes Communes

- **Erreur `ModuleNotFoundError` :**
  Si un module est manquant, assurez-vous d'avoir exécuté `pip install -r requirements.txt` correctement. Si nécessaire, installez manuellement les modules manquants via `pip install nom_du_module`.

- **Problèmes de GPU :**
  Si le modèle ne reconnaît pas votre GPU, assurez-vous que **CUDA** est correctement installé et que PyTorch utilise la bonne version compatible avec votre GPU.

7. Conclusion

Cette section vous a guidé à travers le processus d'installation et de configuration du système pour travailler sur ce projet de détection de parties de véhicules. Vous êtes maintenant prêt à entraîner le modèle, à tester ses performances, et à intégrer les résultats dans une interface interactive Streamlit.

Conclusion

Le projet de détection des parties de véhicules à l'aide de la méthode de segmentation d'images représente une avancée significative dans le domaine de l'intelligence artificielle appliquée à l'automobile. En utilisant des technologies modernes telles que **YOLOv4** et **Streamlit**, il est possible d'identifier et de localiser précisément des parties spécifiques des véhicules dans des images, ce qui peut avoir des applications variées, allant de l'inspection automobile à l'analyse de la sécurité et de l'assurance.

La mise en place de ce système repose sur une architecture robuste et flexible, qui permet une adaptation facile aux besoins spécifiques du domaine. Les données d'entraînement, fournies par le **Car Parts Dataset**, ont permis de former un modèle capable de détecter efficacement 18 parties du véhicule dans différentes perspectives (avant, arrière et vues angulaires). En outre, l'utilisation de techniques avancées de **segmentation sémantique** permet de distinguer avec précision chaque composant du véhicule, contribuant à la précision du système.

L'intégration de **Streamlit** pour l'interface utilisateur permet d'offrir une expérience interactive et accessible, permettant à n'importe quel utilisateur de charger des images, d'effectuer des prédictions et de visualiser les résultats de manière intuitive. Ce projet ouvre ainsi la voie à des applications futures dans plusieurs secteurs, notamment l'entretien automobile, l'assurance, et même la gestion des flottes de véhicules.

Enfin, les défis rencontrés lors de l'entraînement du modèle, ainsi que les solutions mises en place pour améliorer les performances et garantir une interface fluide, témoignent de l'importance de l'optimisation dans des projets utilisant des modèles de deep learning pour des tâches en temps réel.

Ce projet montre que les technologies de segmentation d'images peuvent offrir des solutions puissantes et pratiques aux industries nécessitant une analyse visuelle fine, tout en étant un excellent exemple de l'utilisation des techniques d'IA dans un contexte industriel spécifique. Il est désormais possible d'envisager des applications plus vastes dans la reconnaissance des objets et dans d'autres domaines du secteur automobile grâce à des approches similaires.

Brouillon

Jupyter Notebook : car_part_detection.ipynb
(https://colab.research.google.com/drive/1axucSB5na0mQx7ojfdWiNEkftTdC66n3?usp=sharing)