UNIVERSITÉ
CAEN
NORMANDIE

Université de caen

Rapport de projet

Développement d'un logiciel d'identification de véhicules

Rousseau Pierre-alexandre 21902963
Malick Sokhona 21910279

Semestre 2 – Année universitaire 2024-2025

May 13, 2025

Contents

Chapter 1

Introduction

Ce projet a pour objectif de concevoir un logiciel d'identification de véhicules à partir de la reconnaissance automatique des plaques d'immatriculation. Une première version réalisée au semestre 1 reposait sur un système simple : détection approximative d'une plaque, envoi à une API, puis affichage des données du véhicule. Ce second semestre a été consacré à une refonte complète du système, avec un moteur de détection fiable (YOLOv5), une interface web, un système de cache JSON et une interface administrateur.

Chapter 2

Semestre 1

2.1 Méthodologie

Pour réaliser la première version du projet, nous avons suivi une approche simple en plusieurs étapes :

1. **Traitement d'image** : utilisation d'OpenCV pour détecter les plaques d'immatriculation. Cela passe par la conversion en niveaux de gris, la réduction du bruit, la détection des contours, puis l'identification de la zone de la plaque.
2. **Reconnaissance de texte** : avec EasyOCR, nous avons extrait le numéro de plaque à partir de l'image détectée.
3. **Interrogation de l'API** : une fois le texte reconnu, on l'envoie à une API pour récupérer les informations du véhicule.
4. **Interface utilisateur** : développement d'une interface web simple avec Flask pour permettre l'envoi d'images et afficher les résultats.

2.2 Recherche et mise en place d'une solution simple

Nous avons effectué plusieurs recherches sur les méthodes de traitement d'image qui permettent de détecter et lire automatiquement les plaques d'immatriculation. Voici les principales étapes du processus :

1. **Prétraitement** : amélioration de la qualité de l'image, suppression du bruit et des effets de lumière. On utilise notamment la conversion en niveaux de gris et un flou gaussien.

2. **Transformations géométriques** : correction de la perspective ou de l'orientation si la plaque est inclinée ou vue sous un angle.

3. **Détection de contours** : repérage des bords à l'aide d'algorithmes comme Canny, pour isoler les zones qui ressemblent à une plaque.

4. **Segmentation** : une fois les contours détectés, on isole la région d'intérêt (ROI), c'est-à-dire la plaque, pour se concentrer uniquement sur cette zone.

5. **Traitements morphologiques** : nettoyage final pour améliorer l'image, combler les manques ou lisser les bords, grâce à des opérations comme l'érosion et la dilatation.

6. **Extraction de texte** : grâce à l'OCR (Tesseract), on extrait le numéro de la plaque sous forme de texte.

Après cela, nous envoyons le texte de la plaque à l'API suivante : https://apiplaqueimmatriculati
com/tester-lapi-plaque-immatriculation-siv/, qui retourne les informations du véhicule au format JSON.

À noter : l'API est gratuite pour un maximum de 50 requêtes. Pour économiser les appels pendant la phase de développement, nous avons simulé les réponses avec des données fixes, en ne modifiant que la valeur de l'immatriculation. Pour les derniers tests et la soutenance, nous avons prévu d'activer les vraies requêtes.

2.3 Partie visuelle réalisée avec Flask

L'interface web du projet a été construite à l'aide de Flask. Voici comment le projet est organisé :

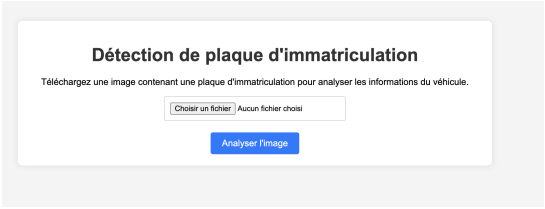
- **app.py** : le fichier principal qui lance l'application.
- **utils/** : contient des fonctions pour le traitement d'image, l'enregistrement et l'accès à l'API.
- **services/** : regroupe la logique métier, notamment la récupération d'informations.
- **templates/** : contient les fichiers HTML utilisés par Flask.
- **static/** : regroupe les fichiers statiques (CSS, images...).

2.4 Résultats

Nous avons testé notre application sur plusieurs images de plaques. Les résultats sont globalement satisfaisants :

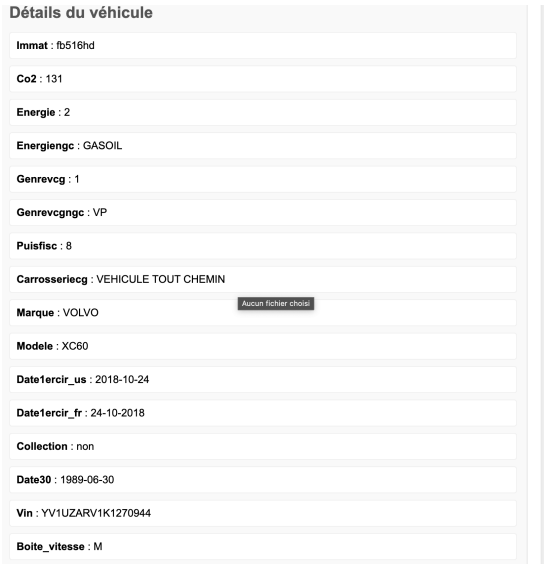
- La détection de la plaque fonctionne bien dans la majorité des cas.
- L'OCR arrive correctement à lire le numéro de plaque.
- L'API renvoie les bonnes informations de façon fiable.

Voici un exemple de résultat obtenu :

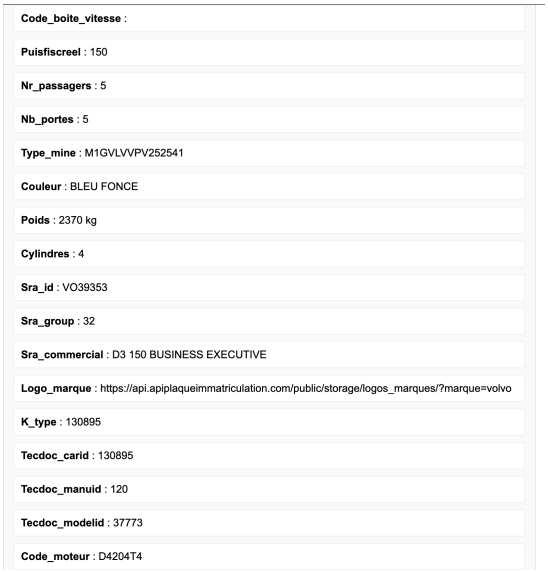


(a) Envoie de l'image

(b) Image Traité



(c) Infortaion du véhicule (1)



(d) Infortaion du véhicule (1)

Figure 2.1: Résultat avec flask

Chapter 3

Semestre 2

3.1 Amélioration de la détection des plaques

3.1.1 Limites de la version précédente

Le système de détection du semestre 1 était peu fiable et ne permettait pas une extraction robuste de la plaque. Il reposait sur une méthode simple non entraînée, sensible au bruit, et dépendait entièrement d'un appel API sans vérification locale.

3.1.2 Fonctionnement de YOLOv5

YOLOv5 (You Only Look Once version 5) est un algorithme d'apprentissage profond destiné à la détection d'objets. Il fonctionne en analysant une image en un seul passage, ce qui lui permet de détecter plusieurs objets simultanément avec une grande rapidité.

Principe général

YOLOv5 divise l'image d'entrée en une grille. Chaque cellule de cette grille est responsable de prédire un certain nombre de boîtes englobantes et la probabilité que ces boîtes contiennent un objet d'une certaine classe (ici : une plaque d'immatriculation grâce à notre entraînement).

Chaque boîte retournée contient :

- Les coordonnées (x, y, w, h) du rectangle englobant,
- Un score de confiance (probabilité qu'un objet soit présent),
- La probabilité pour chaque classe (ici : plaque, mais cela pourrait être une voiture, un feu rouge, etc.).

Avantages de YOLOv5

YOLOv5 est léger, rapide et précis. Contrairement à d'autres modèles plus lourds (comme Faster R-CNN), il est adapté aux applications temps réel comme les vidéos ou les flux webcam. YOLOv5 a aussi l'avantage d'être facile à entraîner et à déployer, ce qui nous a permis de le personnaliser pour notre propre cas.

Entraînement pour notre projet

Nous avons entraîné YOLOv5 sur un jeu de données composé d'images de plaques d'immatriculation, avec des annotations au format `YOLO`. Chaque annotation décrit la position de la plaque sur l'image. L'entraînement a été réalisé sur GPU afin de réduire le temps de calcul.

Après l'entraînement, nous avons obtenu un fichier `best.pt` contenant les poids du modèle, que nous utilisons ensuite pour détecter les plaques dans les images, vidéos et flux webcam via l'interface Flask.

3.2 Interface graphique et fonctionnalités

3.2.1 Structure générale avec Flask

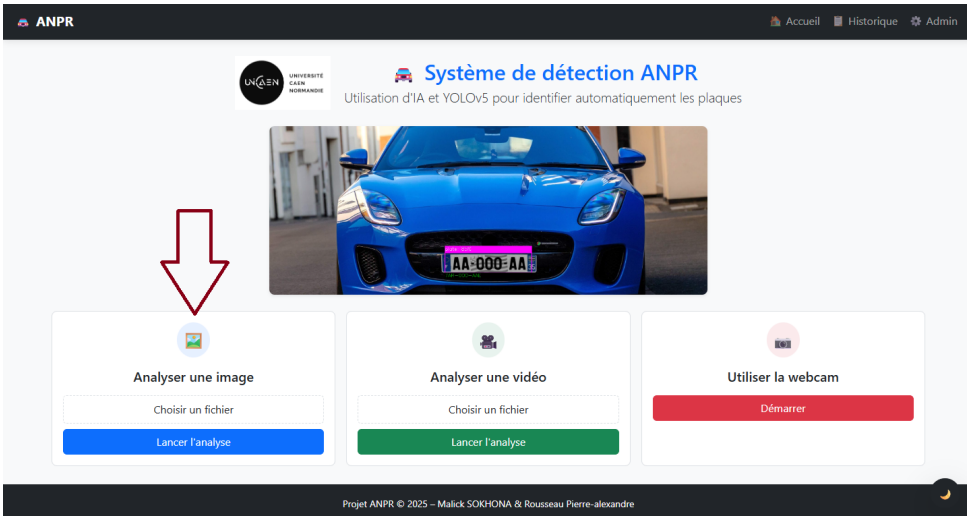
Nous avons utilisé Flask comme framework web pour centraliser les trois fonctionnalités principales du projet dans une interface unique et intuitive.

3.2.2 Fonctionnalité image avec Yolo

L'utilisateur peut envoyer une image via l'interface Flask. Le système détecte la plaque avec YOLOv5. Les résultats sont ensuite affichés à l'écran avec un lien pour plus de détails.

Exemple

Lors de l'arrivée sur la page principale, tout à gauche, il faut envoyer une image dans un format accepté (.png ou .jpg).



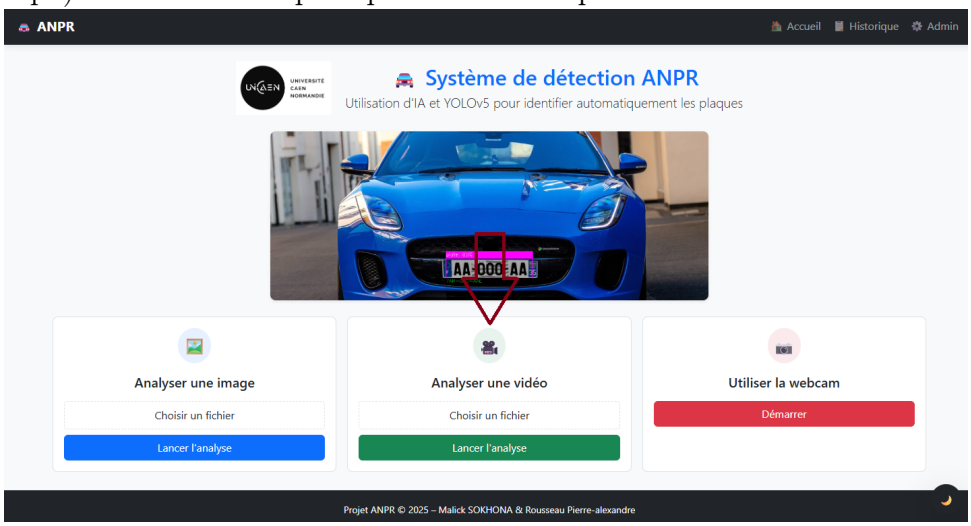
Ensuite, le système utilise YOLOv5 pour traiter l'image et redirige vers une page où elle est annotée (rectangle violet sur la plaque détectée). En dessous, une liste des plaques détectées est affichée. Un clic sur une plaque redirige vers la page d'information.

3.2.3 Fonctionnalité vidéo

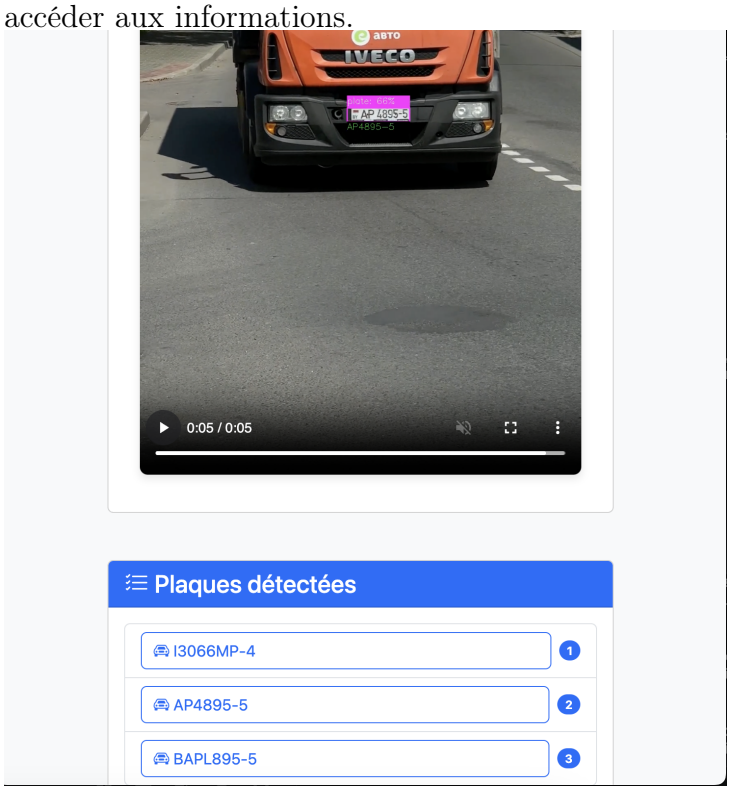
L'utilisateur peut envoyer une vidéo. Le système la traite image par image pour détecter les plaques, interroger l'API si besoin, puis afficher les résultats.

Exemple

Sur la page principale (au centre), il faut envoyer une vidéo dans un format supporté (ex. mp4). Le traitement peut prendre du temps selon la taille du fichier.



Le système utilise YOLOv5 pour analyser la vidéo. Ensuite, il affiche une page avec la vidéo annotée et les plaques détectées en dessous. Chaque plaque est cliquable pour accéder aux informations.

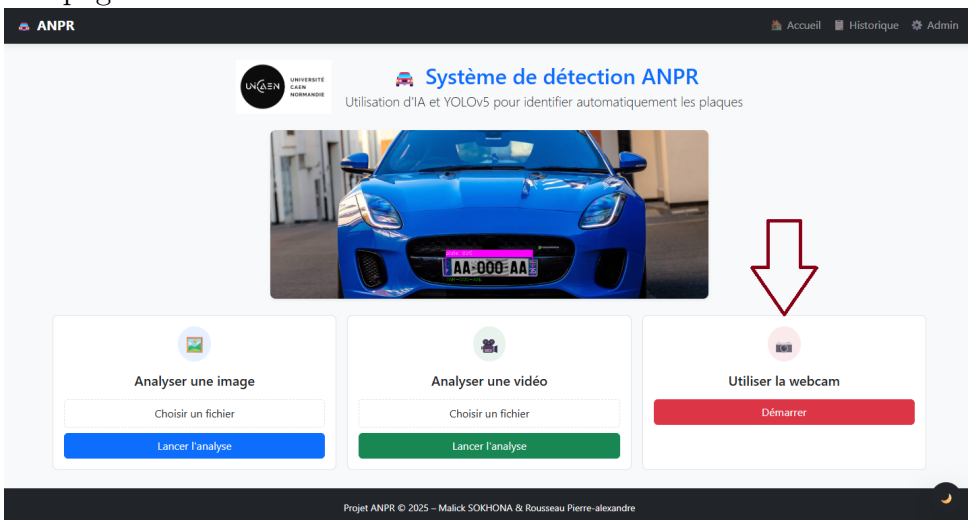


3.2.4 Fonctionnalité webcam

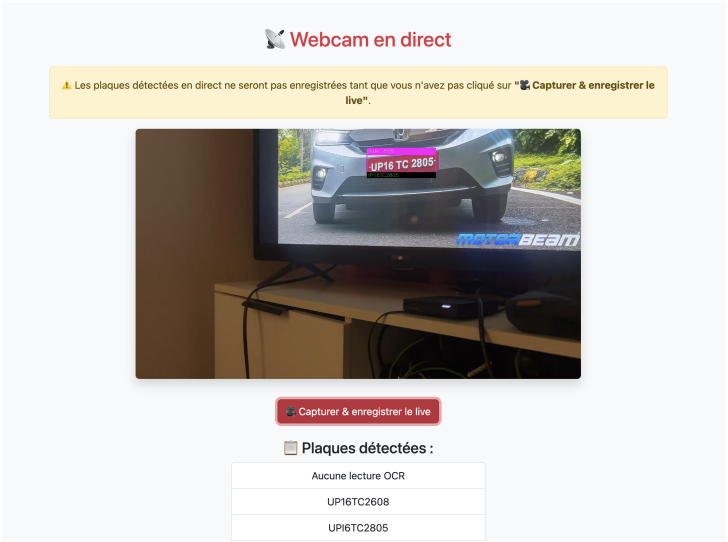
Une détection en direct via webcam a été ajoutée. Le système traite en temps réel l'image capturée avec la même méthode que pour la vidéo.

Exemple

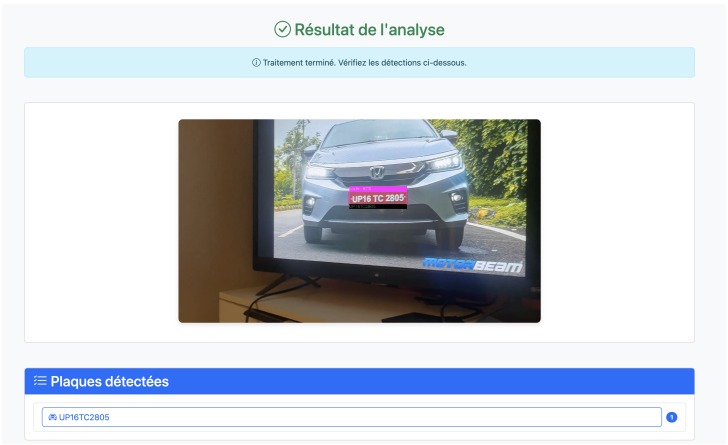
Depuis la page principale, cliquer sur "Webcam" pour lancer la détection. Cela ouvre une page affichant en direct le flux webcam actif.



YOLOv5 détecte les plaques et affiche des rectangles violets. Les plaques sont listées et cliquables pour accéder aux détails.



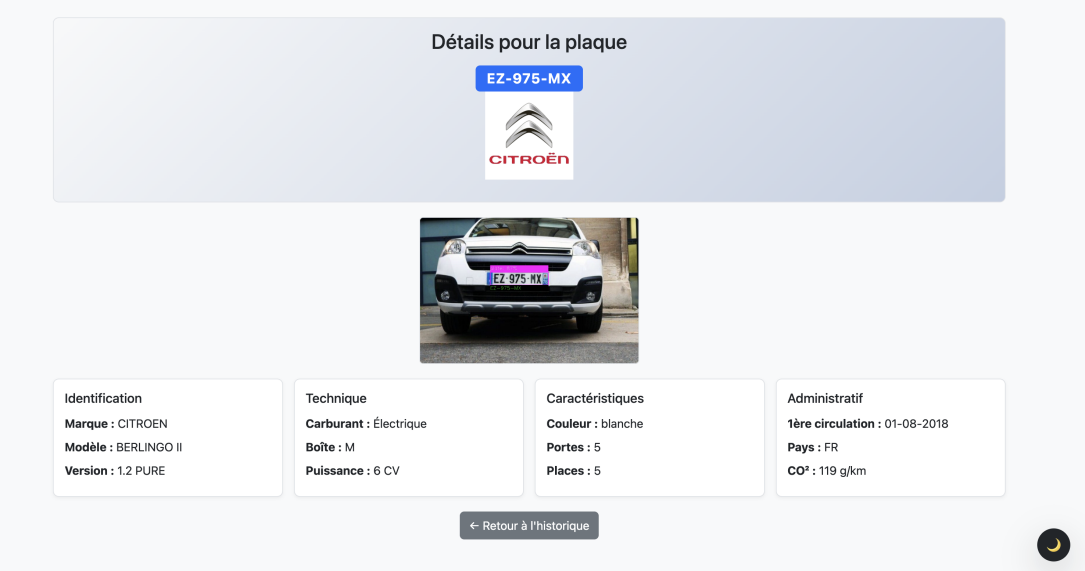
(a) Pendant l'enregistrement



(b) Après l'enregistrement

3.2.5 Page détail

Cette page affiche les informations détaillées d'une plaque. Elle vérifie d'abord si la plaque est déjà dans un fichier JSON local. Sinon, elle interroge l'API.



3.3 Optimisation et gestion des données

3.3.1 Recherche dans fichier JSON

Pour éviter les appels redondants à l'API (limitée à 50 requêtes/jour par PC), les informations sont enregistrées dans un fichier '.json'. Avant chaque requête, le fichier est consulté.

3.3.2 Historique administrateur

Une interface admin permet de consulter l'historique des recherches, par type (image, vidéo, webcam), et d'effectuer des corrections manuelles si une plaque est mal détectée — fonctionnalité utile pour des améliorations futures.

Chapter 4

Finalité du projet

4.1 Difficultés rencontrées

Nous avons rencontré plusieurs défis techniques durant ce projet :

4.1.1 Reconnaissance optique de caractères

- **Confusions alphanumériques** : L'OCR confondait fréquemment :
  - Le chiffre `0` avec la lettre `O`
  - Le chiffre `1` avec la lettre `I`
  - Le chiffre `8` avec la lettre `B`

  Solution : Implémentation d'un dictionnaire de correction et de règles de validation spécifiques aux plaques françaises.

- **Artéfacts parasites** : Ajouts intempestifs de caractères spéciaux (`!`, `-`) en début/fin de plaque. *Solution* : Filtrage par expressions régulières et validation via la fonction `is_valid_plate()`.

4.1.2 Limitations matérielles

- **Temps de traitement vidéo** : Latence importante sur les vidéos HD (>720p). *Solution* : Optimisation du modèle YOLOv5 exporté en ONNX et réduction de la résolution d'analyse à 640x640 pixels.

- **Compatibilité multi-OS** : Chemins différents pour Tesseract OCR selon les systèmes. *Solution* : Détection automatique du système via `platform.system()`.

4.1.3 Interface utilisateur

- **Synchronisation webcam** : Délais entre l'acquisition et l'analyse en temps réel. *Solution* : Implémentation d'un buffer circulaire et optimisation du flux MJPEG.
- **Expérience mobile** : Incompatibilité partielle avec les appareils tactiles. *Solution* : Ajout de media-queries CSS et d'éléments interactifs adaptés.

4.1.4 Gestion de projet

- La différence d'horaires entre membres du groupe et avec les encadrants, due aux emplois du temps respectifs, a complexifié les réunions de synchronisation.
- L'entraînement de YOLOv5 a nécessité une phase importante de documentation et d'expérimentation pour adapter le modèle à notre cas d'usage spécifique.

4.2 Résultats obtenus

Le système actuel est stable et permet une détection fiable de plaques en images, vidéos et flux directs. Les performances du modèle sont satisfaisantes avec un taux de détection élevé dans des conditions standards. L'interface Flask rend l'utilisation intuitive, et l'administrateur peut suivre toutes les recherches via un historique centralisé.

4.3 Partage des tâches

Pour se projet du a notre petit nombre on a peu touche a tout cependant on peut voir un partage comme ci-dessous:
-Malick : Code, graphisme
-Rousseau : Code, Documentation

4.4 Conclusions

Ce second semestre a permis de passer d'un prototype expérimental à un logiciel fonctionnel et complet. L'intégration de l'intelligence artificielle via YOLOv5, la gestion locale des données, et la conception d'une interface claire sont les points forts de cette version. De futures améliorations pourraient viser .

Appendix A

Annexes

A.1 Instructions pour lancer le projet

1. git clone https://github.com/malikiskn/Identification-de-vehicule.git
2. Installer les dépendances :

```
pip install -r requirements.txt
```

3. Lancer l'application Flask :

```
cd plate_recognition
python3 app.py
```

4. Accéder à l'application via : `http://127.0.0.1:5000`

Bibliography

[1] Glenn Jocher et al., *YOLOv5 - Ultralytics*, GitHub Repository, 2020. https://github.com/ultralytics/yolov5

[2] : Video explicatif de yolov5: https://www.youtube.com/watch?v=cXDzzefyGPs

[3] Pallets Projects, *Flask – The Python micro framework*, https://flask.palletsprojects.com/