N° d'ordre : .....

Mémoire de fin d'études déposé à l'

UNIVERSITÉ MOHAMED BOUDIAF – MSILA



FACULTÉ DES MATHÉMATIQUES ET DE L'INFORMATIQUE
DÉPARTEMENT DE L'INFORMATIQUE

Pour la satisfaction partielle des exigences du diplôme de

Master en **Informatique**

Par

Oualid,Elmokhtar

Aissaoui, Somia

Titre du mémoire

Développement d'un Système Intelligent de Contrôle d'Accès par Reconnaissance Automatique des Plaques d'Immatriculation

---

Sous la direction de

Dr .Kamel Mohamed

Composition du jury

Dr .Barkat Abdelbasset	Université de M'sila	Président
Dr .Kamel Mohamed	Université de M'sila	Rapporteur
Dr .Benazi Makhlouf	Université de M'sila	Examinateur

Juin, 2026.

Dédicaces

"À mon père, qui m'a appris que le savoir est la plus grande des richesses, et à ma mère, dont la tendresse et les prières ont guidé chaque pas de mon chemin "

Somia Aissaoui

"À mes parents, source inépuisable de courage et de détermination, qui n'ont jamais cessé de croire en moi, et à tous ceux qui m'ont accompagné sur la voie de la connaissance "

Elmokhtar Oualid

Remerciements

Au terme de ce travail, nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce mémoire.

Nous adressons nos remerciements les plus sincères à notre directeur de mémoire, pour sa disponibilité, ses précieux conseils et son soutien constant tout au long de ce projet. Sa rigueur scientifique et son encadrement bienveillant ont été une source d'inspiration inestimable.

Nous remercions également les membres du jury qui nous ont fait l'honneur d'accepter d'évaluer ce travail, et dont les remarques constructives contribueront à son amélioration.

Nos sincères remerciements vont aussi à l'ensemble du corps enseignant du Département de l'Informatique de l'Université Mohamed Boudiaf – Msila, pour la qualité de la formation dispensée durant notre cursus.

Enfin, nous remercions chaleureusement nos familles et amis pour leur soutien moral indéfectible, leur patience et leurs encouragements tout au long de cette aventure académique.

Résumé

Ce mémoire présente la conception et l'implémentation du système Intelligent Gate un système ANPR (Automatic Number Plate Recognition) pour plaques algériennes basé sur YOLOv8n (détection) et PaddleOCR (reconnaissance des caractères). Le pipeline intègre l'acquisition vidéo, le prétraitement OpenCV, l'interface graphique Tkinter, un module de contrôle d'accès par liste blanche et une simulation de barrière de passage. Les résultats obtenus sont : mAP@0.5 = 100 %, Precision = 100 %, Recall = 100 %, IoU moyen = 0,9389 pour la détection ; Exact Match Accuracy = 88,46 % et Digit Accuracy = 96,17 % pour l'OCR, le tout sur CPU standard sans GPU.

Mots clés : ANPR ; YOLOv8 ; PaddleOCR ; Reconnaissance de plaques ; Détection d'objets ; Vision par ordinateur ; Contrôle d'accès

ملخص

تقدم هذه المذكرة تصميم وتنفيذ نظام Intelligent Gate — نظام التعرف التلقائي على لوحات الترقيم الجزائرية (ANPR)المبني على YOLOv8nللكشف و PaddleOCRللتعرف الضوئي على الأحرف .يدمج النظام اكتساب الفيديو، المعالجة المسبقة بـ OpenCV، واجهة رسومية Tkinter، وحدة التحكم في الوصول بالقائمة البيضاء، ومحاكاة حاجز المرور .النتائج المتحصل عليها100 = mAP@0.5 :٪، Precision = 100٪، Recall = 100٪، متوسط IoU = 0.9389للكشف؛ دقة المطابقة الكاملة 88.46 =٪ ودقة الأرقام = 96.17٪ للتعرف الضوئي، على معالج عادي دون .GPU

الكلمات المفتاحية: ANPR؛ YOLOv8؛ PaddleOCR؛ التعرف على لوحات الترقيم؛ كشف الأجسام؛ رؤية الحاسوب؛ التحكم في الوصول

Abstract

This thesis presents the design and implementation of Intelligent Gate — an ANPR system for Algerian license plates based on YOLOv8n (detection) and PaddleOCR (character recognition). The pipeline integrates video acquisition, OpenCV pre-processing, a Tkinter graphical interface, a whitelist-based access control module, and a gate simulation. Results: mAP@0.5 = 100 %, Precision = 100 %, Recall = 100 %,

mean IoU = 0.9389 for detection ; Exact Match Accuracy = 88.46 % and Digit Accuracy = 96.17 % for OCR, running on standard CPU without GPU

Key words: ANPR ; YOLOv8 ; PaddleOCR ; License Plate Recognition; Object Detection; Computer Vision; Access Control

Table des matières

List of figures xvi
Liste des tableaux xiv
Introduction générale 1
Chapitre1 État de l'art et concepts fondamentaux 2
1. Intelligence Artificielle et Apprentissage Automatique 2
1.1. Définition de l'Intelligence Artificielle 2
1.2. Définition du Machine Learning 3
1.3. Types d'apprentissage automatique 3
2. Apprentissage Profond (Deep Learning) 4
2.1. Définition et principes de base 4
2.2. Réseaux de neurones artificiels (ANN) 4
2.3. Réseaux de neurones convolutifs (CNN) 5
2.4. Applications du Deep Learning en vision par ordinateur 6
3. Systèmes ANPR et Plaque Algérienne 6
3.1. Définition et historique des systèmes ANPR 6
3.2. Pipeline général d'un système ANPR 6
3.3. La plaque algérienne — Caractéristiques et défis 7
3.4. État de l'art — Revue des travaux antérieurs en ANPR 7
4. Vision par Ordinateur 8
4.1. Définition et domaines d'application 8
4.2. Traitement numérique des images 8
4.3. Détection de mouvement et de fond (MOG2) 9
4.4. Reconnaissance Optique de Caractères (OCR) 9
5. Présentation des Outils et Bibliothèques Utilisés 9
5.1. Langage Python 10
5.2. YOLOv8 (Ultralytics) 10
5.3. PaddleOCR 10
5.4. OpenCV 10
5.5. Tkinter et NumPy 11
Conclusion 11
Chapitre 2 Conception et Modélisation du Système 12

Introduction 12
1. Analyse des Besoins 13
1.1 Besoins Fonctionnels 13
1.2 Besoins Non-Fonctionnels 14
1.3 Identification des Acteurs 15
2. Architecture Globale du Système 16
2.1 Vue d'Ensemble de l'Architecture 16
2.2 Description des Composants 16
2.3 Choix Architecturaux et Justifications 17
3. Modélisation UML 18
3.1 Diagramme de Cas d'Utilisation 18
3.2 Diagramme de Séquence — Flux Principal 19
3.3 Diagramme d'Activités — Pipeline de Traitement 20
3.4 Diagramme de Composants — Architecture Modulaire 22
4. Conception Détaillée des Modules 23
4.1 Module d'Acquisition et Détection de Mouvement 23
4.1.1 Chargement depuis fichier 23
4.1.2 Capture webcam avec détection de mouvement 23
4.2 Module de Détection YOLOv8n 23
4.2.1 Modèle et entraînement 23
4.2.2 Extraction de la ROI et transformation de perspective 24
4.3 Module de Reconnaissance OCR (PaddleOCR) 24
4.3.1 Pipeline de prétraitement 24
4.3.2 Reconnaissance et post-traitement 24
4.4 Module Interface et Journalisation 24
4.5 Module WhitelistManager — Gestion de la Liste Blanche 25
4.5.1 Structure du fichier de persistance 25
4.5.2 Algorithme de vérification d'autorisation 25
4.6 Module GateSimulator — Simulateur de Barrière de Passage 26
4.6.1 Machine à états de la barrière 26
4.6.2 Rendu graphique et éléments visuels 26
5. Conception de la Persistance des Données 27
5.1 Journal de Reconnaissance — plates_log.csv 27
5.2 Référentiel de Véhicules Autorisés — allowed_vehicles.json 27
Conclusion 28
Chapitre 3 Réalisation et Évaluation du Système Intelligent Gate 29
1. Environnement de Développement et de Test 29

2. Présentation des Interfaces Graphiques 29
3. Modèle de Détection — YOLOv8n 30
3.1 Architecture YOLOv8n — Présentation du Modèle 30
3.1.1 Pourquoi YOLOv8n ? 30
3.2 Dataset et Préparation des Données 31
3.2.1 Source et composition 31
3.2.2 Visualisation du dataset 32
3.3 Protocole et Paramètres d'Entraînement 34
3.4 Analyse des Courbes d'Entraînement 35
3.4.1 Phase initiale — Calibration (Époques 1–5) 37
3.4.2 Phase d'apprentissage principal (Époques 6–30) 37
3.4.3 Phase de raffinement post-Mosaic (Époques 31–40) 37
3.4.4 Absence de sur-apprentissage 37
3.5 Courbes d'Évaluation sur la Validation 38
4. Évaluation Indépendante — YOLOv8n sur le Jeu de Test 41
4.1 Protocole d'Évaluation 41
4.2 Résultats Globaux — Métriques de Détection 41
4.3 Robustesse au Seuil de Confiance 42
4.4 Qualité de Localisation — Distribution des IoU 42
4.5 Matrice de Confusion 43
4.6 Exemples Visuels de Détection 44
4.7 Performance Temporelle d'Inférence 48
5. Modèle de Reconnaissance OCR — PaddleOCR 49
5.1 Choix de PaddleOCR 49
5.2 Pipeline Interne de PaddleOCR 49
5.3 Pipeline de Prétraitement 50
6. Évaluation du Pipeline OCR sur le Jeu de Test 50
6.1 Protocole d'Évaluation Manuelle 50
6.2 Résultats Globaux 51
6.3 Distribution des Scores de Confiance OCR 52
6.4 Analyse des Erreurs par Longueur de Plaque 52
6.5 Analyse Qualitative des Erreurs Résiduelles 53
7. Synthèse Globale des Performances du Système ANPR 53
7.1 Tableau de Bord Complet des Performances 53
7.2 Performance End-to-End du Pipeline Complet 54
7.3 Comparaison avec l'État de l'Art 55
7.4 Points Clés pour la Commission d'Évaluation 56

Conclusion générale 57
Bibliographie 58

List of figures

Fig 1.1 — Relation hiérarchique : IA ⊃ Machine Learning ⊃ Deep Learning

Fig 1.2 — Structure d'un réseau de neurones artificiel (ANN) multicouche

Fig 1.3 — Illustration de l'opération de convolution 2D sur une image

Fig 1.4 — Pipeline général d'un système ANPR (flux de traitement horizontal)

Fig 1.5 — Exemples de plaques algériennes avec annotation du format

Fig 1.6 — Séquence de prétraitement d'une image de plaque avant reconnaissance OCR

Fig 1.7 — Architecture du moteur PaddleOCR : détection du texte et reconnaissance des caractères

Fig 1.8 — Architecture complète de YOLOv8 : Backbone, Neck (FPN-PAN) et tête découplée

Fig 2.1 — Diagramme de cas d'utilisation du système Intelligent Gate

Fig 2.2 — Diagramme de séquence — Flux de reconnaissance et contrôle d'accès

Fig 2.3 — Diagramme d'activités — Pipeline de traitement complet

Fig 2.4 — Diagramme de composants — Architecture modulaire du système

Fig 3.1 — Distribution des annotations dans le dataset

Fig 3.2 — Lot d'entraînement initial (batch 0) — annotations ground truth

Fig 3.3 — Lot d'entraînement final (batch 690) — diversité des conditions

Fig 3.4 — Courbes d'entraînement complètes — Losses et métriques sur 40 époques

Fig 3.5 — Courbe F1-Score vs seuil de confiance

Fig 3.6 — Courbe Precision-Recall (AP@0.5 = 0,995)

Fig 3.7 — Courbe de Precision en fonction du seuil de confiance

Fig 3.8 — Courbe de Recall en fonction du seuil de confiance

Fig 3.9 — Matrice de confusion absolue — 52/52 True Positives

Fig 3.10 — Matrice de confusion normalisée — taux = 1,00

Fig 3.11 — Lot de validation 0 — Annotations ground truth

Fig 3.12 — Lot de validation 0 — Prédictions YOLOv8n

Fig 3.13 — Lot de validation 1 — Annotations ground truth

Fig 3.14 — Lot de validation 1 — Prédictions YOLOv8n

Liste des tableaux

Tableau 1.1 — Comparaison des types d'apprentissage automatique

Tableau 1.2 — Évolution des architectures CNN majeures

Tableau 1.4 — Comparaison des détecteurs d'objets modernes

Tableau 1.3 — Caractéristiques et défis de la plaque algérienne

Tableau 1.5 — Comparaison des moteurs OCR

Tableau 1.6 — Récapitulatif des bibliothèques et outils utilisés

Tableau 2.1 — Besoins fonctionnels du système Intelligent Gate

Tableau 2.2 — Besoins non-fonctionnels du système Intelligent Gate

Tableau 2.3 — Structure d'un enregistrement dans allowed_vehicles.json

Tableau 2.4 — Machine à états du module GateSimulator

Tableau 2.5 — Structure du fichier plates_log.csv

Tableau 3.1 — Caractéristiques architecturales du modèle YOLOv8n

Tableau 3.2 — Répartition du dataset par split

Tableau 3.3 — Hyperparamètres d'entraînement YOLOv8n

Tableau 3.4 — Évolution des métriques aux époques jalons

Tableau 3.5 — Résultats complets de l'évaluation YOLOv8n — jeu de test (52 images)

Tableau 3.6 — Analyse de robustesse par rapport au seuil de confiance

Tableau 3.7 — Distribution des valeurs IoU sur les 52 images

Tableau 3.8 — Statistiques du temps d'inférence YOLOv8n

Tableau 3.9 — Pipeline de prétraitement — étapes, paramètres et justifications

Tableau 3.10 — Résultats complets de l'évaluation OCR (52 images)

Tableau 3.11 — Histogramme des scores de confiance OCR

Tableau 3.12 — Exact Match Accuracy par longueur de plaque

Tableau 3.13 — Exemples représentatifs d'erreurs OCR et analyse causale

Tableau 3.14 — Tableau de bord des performances — Intelligent Gate ANPR

Tableau 3.15 — Métriques End-to-End du pipeline complet

Tableau 3.16 — Comparaison avec les systèmes ANPR existant

Introduction générale

L'identification automatique des véhicules constitue un défi technologique majeur dans les systèmes de gestion du trafic urbain et de la sécurité routière. En Algérie, l'essor du parc automobile et les besoins croissants en surveillance et en contrôle d'accès rendent nécessaire le développement de solutions localisées et performantes.

La reconnaissance automatique de plaques d'immatriculation (ANPR — Automatic Number Plate Recognition) est une technologie qui permet, à partir d'images ou de flux vidéo, d'identifier automatiquement les véhicules en lisant leurs plaques. Les défis spécifiques à la plaque algérienne — composée exclusivement de chiffres arabes selon un format structuré — justifient une approche dédiée exploitant les dernières avancées de l'intelligence artificielle.

Ce mémoire propose un système ANPR complet — nommé Intelligent Gate — reposant sur YOLOv8n pour la détection en temps réel et PaddleOCR pour la reconnaissance des caractères, intégré dans une application desktop développée en Python avec une interface graphique Tkinter. Le système inclut également un module de contrôle d'accès par liste blanche et une simulation de barrière de passage. Le travail est structuré en trois chapitres principaux :

— Le Chapitre 1 pose les bases théoriques et technologiques : intelligence artificielle, apprentissage profond, architectures CNN, systèmes ANPR, vision par ordinateur, et présentation des outils retenus.

— Le Chapitre 2 décrit la conception et la modélisation du système : analyse des besoins, architecture en pipeline à six étapes, modélisation UML, et conception détaillée de chaque module.

— Le Chapitre 3 présente la réalisation concrète : entraînement du modèle YOLOv8n, évaluation indépendante sur jeu de test, résultats du pipeline OCR, et synthèse des performances globales du système Intelligent Gate.

Chapitre1 État de l'art et concepts fondamentaux

L'essor fulgurant de l'intelligence artificielle au cours des deux dernières décennies a profondément reconfiguré le paysage technologique mondial. Dans ce contexte, la vision par ordinateur occupe une place de choix : elle permet aux machines de « voir » et d'interpréter le monde visuel, ouvrant la voie à des applications aussi concrètes que la surveillance de trafic, le contrôle qualité industriel, ou encore la reconnaissance automatique de plaques d'immatriculation (ANPR).

Le présent chapitre pose les bases théoriques et technologiques nécessaires à la compréhension du système ANPR développé dans ce mémoire. Nous introduisons les concepts de l'IA et du machine learning, l'apprentissage profond et les CNN, les systèmes ANPR et les spécificités de la plaque algérienne, la vision par ordinateur, puis les outils retenus : YOLOv8, PaddleOCR, OpenCV et Tkinter.

1. Intelligence Artificielle et Apprentissage Automatique

1.1. Définition de l'Intelligence Artificielle

L'intelligence artificielle (IA) est une discipline de l'informatique dont l'objet est de concevoir des systèmes capables d'exécuter des tâches qui, lorsqu'elles sont accomplies par un être humain, requièrent de l'intelligence. Cette définition, formulée dès 1956 lors de la conférence de Dartmouth par John McCarthy, a évolué au fil des décennies pour englober un spectre très large de techniques et d'applications [8]. L'IA se subdivise en IA « étroite » (systèmes spécialisés) et IA générale (encore largement théorique). Notre projet ANPR s'inscrit dans la première catégorie.

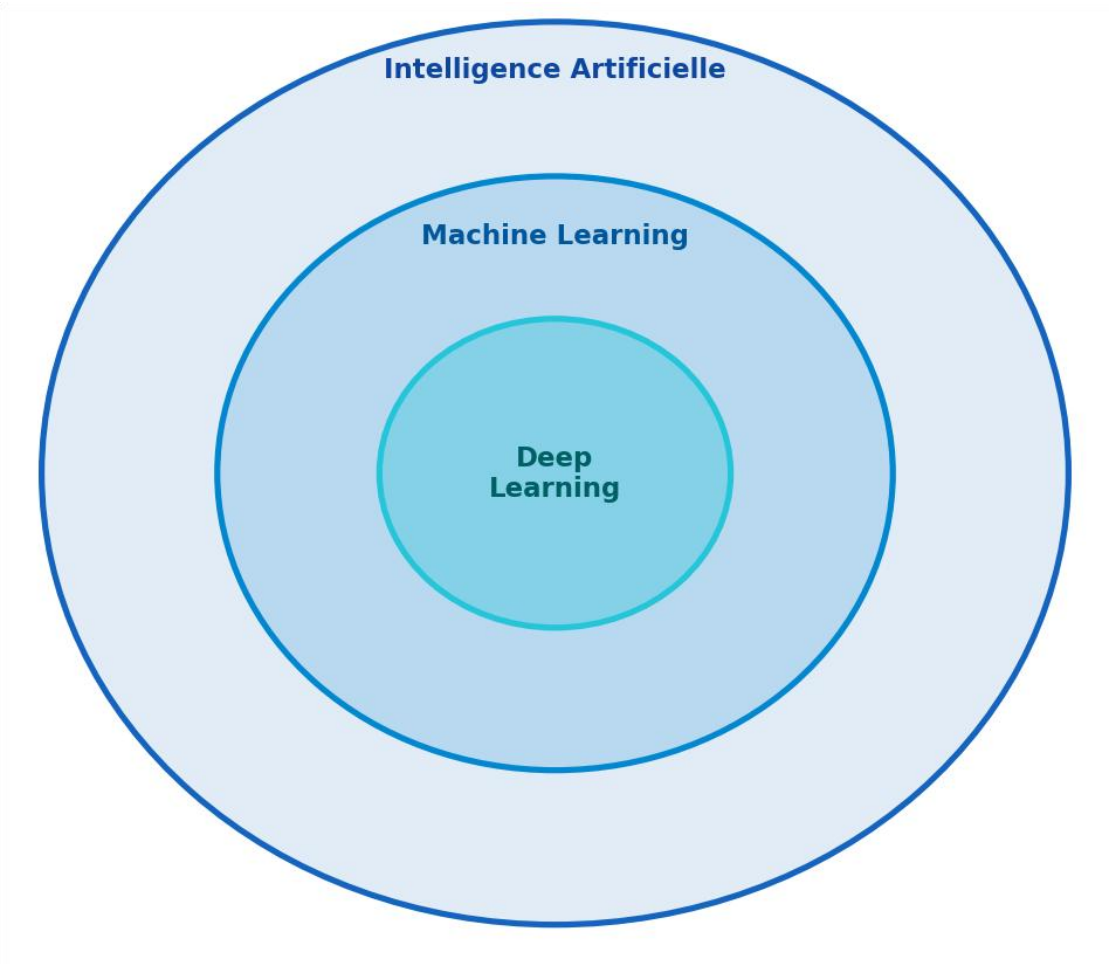


Fig 1.1 — Relation hiérarchique : Intelligence Artificielle ⊃ Machine Learning ⊃ Deep Learning

1.2. Définition du Machine Learning

Le machine learning (ML), ou apprentissage automatique, est une branche de l'IA intéressée aux algorithmes capables d'apprendre à partir de données. La définition canonique de Tom Mitchell (1997) : « un programme informatique est dit apprendre d'une expérience E par rapport à une tâche T et une mesure P, si sa performance à la tâche T, mesurée par P, s'améliore avec E » [9]. Dans notre système ANPR, YOLOv8n est le fruit d'un tel apprentissage.

1.3. Types d'apprentissage automatique

La littérature distingue trois grandes familles d'apprentissage automatique, différenciées par la nature des données et le signal d'apprentissage [10] :

Critère	Supervisé	Non supervisé	Par renforcement
Définition	Apprend à partir de données étiquetées	Découvre des structures dans données non étiquetées	Apprend par essais-erreurs
Données	Étiquetées (annotées)	Non étiquetées	Interaction avec un environnement
Exemple	Détection (YOLOv8)	K-means, Autoencodeur	AlphaGo, robotique
Dans le projet ?	✔ Oui (YOLOv8 + PaddleOCR)	Partiel (augmentation)	✘ Non

Tableau 1.1 — Comparaison des types d'apprentissage automatique

2. Apprentissage Profond (Deep Learning)

2.1. Définition et principes de base

L'apprentissage profond (deep learning) est une sous-discipline du ML reposant sur des réseaux de neurones à multiples couches. Chaque couche apprend à extraire des représentations de plus en plus abstraites à partir des données brutes, sans ingénierie manuelle des caractéristiques [11]. Le principe central est la rétropropagation du gradient : l'erreur du réseau est propagée en sens inverse pour ajuster les poids par descente de gradient (SGD, Adam, RMSProp).

2.2. Réseaux de neurones artificiels (ANN)

Un ANN est modélisé autour du neurone artificiel (perceptron) : il calcule une combinaison linéaire pondérée de ses entrées et applique une fonction d'activation non linéaire (ReLU, sigmoïde) [12]. Les ANN sont organisés en couches : entrée, couches cachées et couche de sortie. Dans YOLOv8, la tête de détection comporte des couches entièrement connectées produisant les prédictions de classe et de localisation.

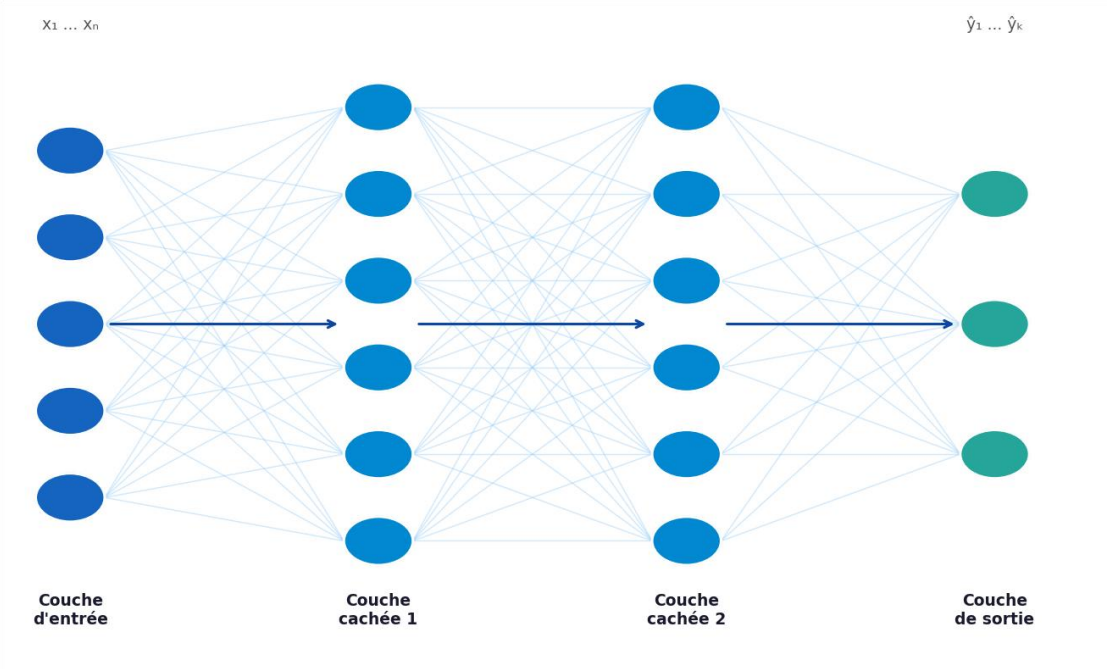


Fig 1.2 — Structure d'un réseau de neurones artificiel (ANN) multicouche

2.3. Réseaux de neurones convolutifs (CNN)

Les CNN sont spécifiquement conçus pour le traitement d'images. Introduits par LeCun et al. en 1998 [13], ils ont révolutionné la vision par ordinateur lors d'ImageNet 2012 avec AlexNet [12]. La convolution 2D avec partage de poids et invariance par translation est leur opération fondamentale [11].

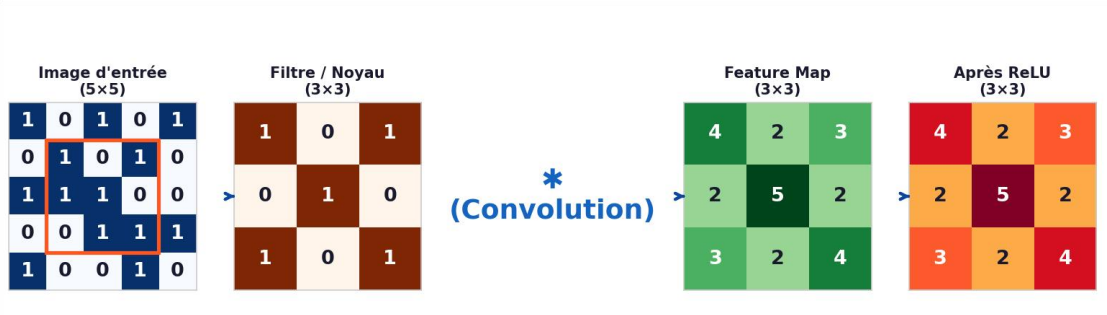


Fig 1.3 — Illustration de l'opération de convolution 2D sur une image

Architecture	Année	Couches	Innovation principale	Pertinence projet
LeNet-5	1998	7	Première CNN — MNIST	Base conceptuelle

AlexNet	2012	8	ReLU, Dropout, GPU	Fondement DL moderne
VGG-16	2014	16	Filtres 3×3 empilés	Référence pédagogique
ResNet-50	2015	50+	Connexions résiduelles	Backbone de détecteurs
YOLOv5	2020	Modulaire	Anchor-based, temps réel	Prédécesseur de YOLOv8
YOLOv8	2023	n/s/m/l/x	Anchor-free, TAL — SOTA	✔ Modèle utilisé

Tableau 1.2 — Évolution des architectures CNN majeures

2.4. Applications du Deep Learning en vision par ordinateur

Le deep learning a révolutionné la vision par ordinateur. La détection d'objets consiste à identifier et localiser des objets par des boîtes englobantes. Des architectures comme Faster R-CNN [15] et la famille YOLO ont successivement repoussé les limites de la discipline.

3. Systèmes ANPR et Plaque Algérienne

3.1. Définition et historique des systèmes ANPR

Un système ANPR (Automatic Number Plate Recognition) est un système de vision par ordinateur capable de lire automatiquement les plaques d'immatriculation à partir d'images ou de flux vidéo. Apparu dans les années 1970 au Royaume-Uni, l'ANPR s'est progressivement généralisé [17] : contrôle d'accès aux parkings, péages autoroutiers, surveillance du trafic, contrôle aux frontières, identification de véhicules recherchés.

3.2. Pipeline général d'un système ANPR

Un système ANPR complet s'articule autour d'un pipeline structuré : acquisition de l'image → détection de la plaque → prétraitement → OCR des caractères → consultation base de données.

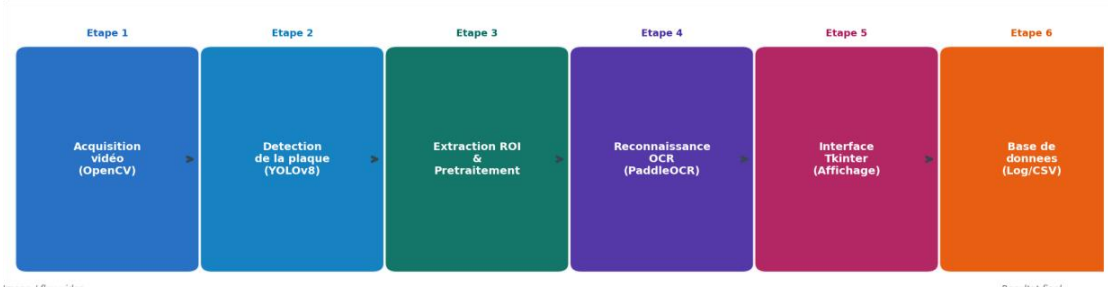


Fig 1.4 — Pipeline général d'un système ANPR (flux de traitement horizontal)

3.3. La plaque algérienne — Caractéristiques et défis

La plaque d'immatriculation algérienne présente des caractéristiques spécifiques qui influencent directement la conception du système ANPR. Contrairement aux plaques européennes, elle est composée exclusivement de chiffres arabes selon un format structuré en trois groupes séparés par des tirets.

Caractéristique	Description
Format des caractères	Chiffres arabes uniquement (ex : 123-45-678-16)
Structure	Numéro de série — Code wilaya — Code série (3 groupes séparés par tirets)
Couleurs (fond/texte)	Fond blanc/chiffres noirs (véhicules particuliers) ; Fond noir/chiffres blancs (plaques diplomatiques)
Dimensions standard	Environ 520 × 110 mm
Police de caractères	Police spécifique, chiffres épais, espacement régulier
Principaux défis	Éclairage variable (nuit, contre-jour) ; Angles obliques ; Plaque obstruée ou sale ; Variations de distance

Tableau 1.3 — Caractéristiques et défis de la plaque algérienne



Fig 1.5 — Exemples de plaques algériennes avec annotation du format

3.4. État de l'art — Revue des travaux antérieurs en ANPR

Les approches ANPR ont considérablement évolué. Les méthodes classiques souffraient d'une faible robustesse aux variations d'éclairage et d'angle [17]. Les approches modernes basées sur le deep learning combinent un détecteur d'objets (YOLO, SSD) avec un moteur OCR séquentiel (CRNN).

Détecteur	mAP COCO	Vitesse	Architecture	Adapté ANPR temps réel ?
Faster R-CNN	42.0 %	~140 ms	Two-stage (RPN)	✘ Trop lent
SSD300	25.1 %	~22 ms	Single-stage, ancres	✔ Acceptable
YOLOv5s	37.4 %	~6 ms	Single-stage, ancres	✔ Bon
YOLOv8s	44.9 %	~4 ms	Anchor-free, découplée	✔✔ Optimal — Choix

Tableau 1.4 — Comparaison des détecteurs d'objets modernes

4. Vision par Ordinateur

4.1. Définition et domaines d'application

La vision par ordinateur (Computer Vision) est une branche de l'IA qui cherche à doter les machines d'une capacité à interpréter le contenu visuel du monde [16]. Son spectre applicatif couvre : la sécurité et la surveillance (ANPR, reconnaissance faciale), le contrôle qualité industriel, l'imagerie médicale, et le véhicule autonome.

4.2. Traitement numérique des images

Le prétraitement vise à améliorer la qualité de l'image avant l'analyse algorithmique. Les opérations utilisées dans notre pipeline incluent : la conversion BGR → niveaux de gris, le lissage gaussien, l'égalisation d'histogramme adaptative (CLAHE) pour le contraste local, et la rectification de perspective [16].

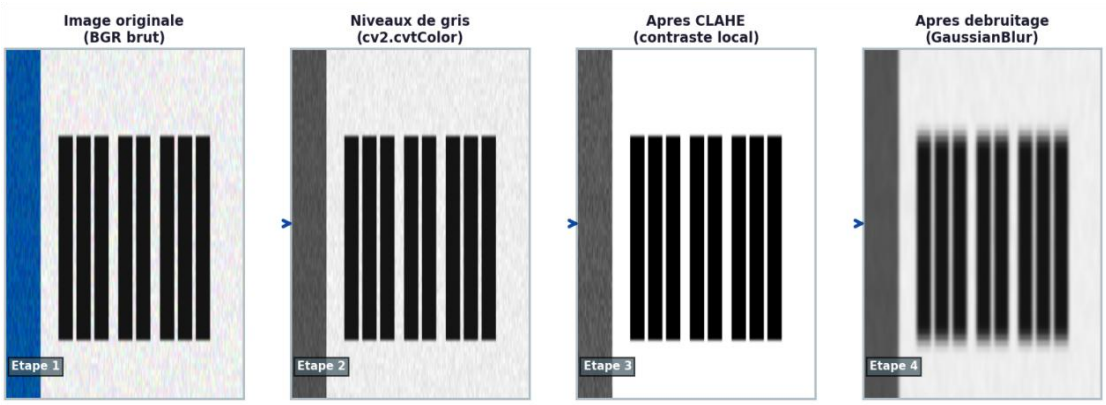


Fig 1.6 — Séquence de prétraitement d'une image de plaque avant reconnaissance OCR

4.3. Détection de mouvement et de fond (MOG2)

L'algorithme MOG2 (Mixture of Gaussians v2), disponible dans OpenCV via cv2.createBackgroundSubtractorMOG2(), modélise le fond par un mélange de distributions gaussiennes mis à jour dynamiquement, permettant de détecter les véhicules même en présence d'ombres [20].

4.4. Reconnaissance Optique de Caractères (OCR)

L'OCR permet de convertir des images de texte en texte machine. Dans le contexte ANPR, elle lit les chiffres composant la plaque à partir de la ROI extraite par le détecteur. Les moteurs OCR modernes utilisent des architectures CRNN (CNN + LSTM/CTC) pour la reconnaissance de séquences [17].

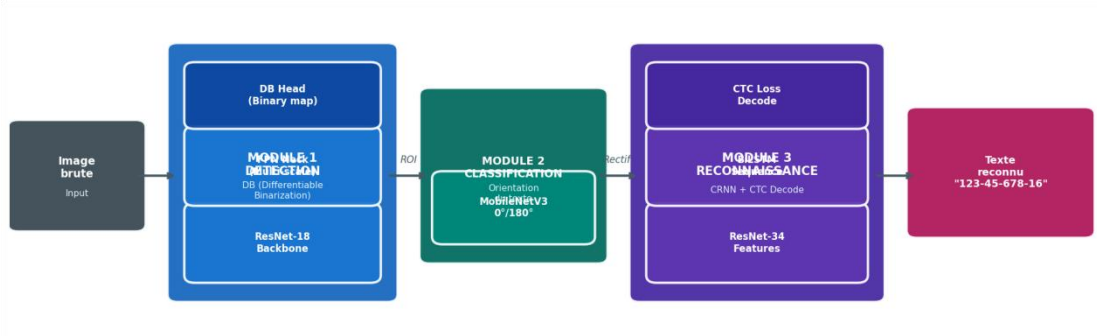


Fig 1.7 — Architecture du moteur PaddleOCR : détection du texte et reconnaissance des caractères

5. Présentation des Outils et Bibliothèques Utilisés

5.1. Langage Python

Python est un langage de programmation interprété dont la philosophie privilégie la lisibilité et la productivité. Lingua franca du machine learning, il bénéficie d'un écosystème riche (PyTorch, OpenCV, PaddlePaddle, Tkinter, NumPy) [18].

5.2. YOLOv8 (Ultralytics)

YOLOv8 est la huitième génération de la famille YOLO, développée par Ultralytics. C'est un détecteur anchor-free à un seul stade [1]. Son architecture : backbone CSPDarknet (modules C2f), neck PAN-FPN, tête de détection découplée. La variante nano (YOLOv8n) avec 3,01 M paramètres est retenue pour notre projet.

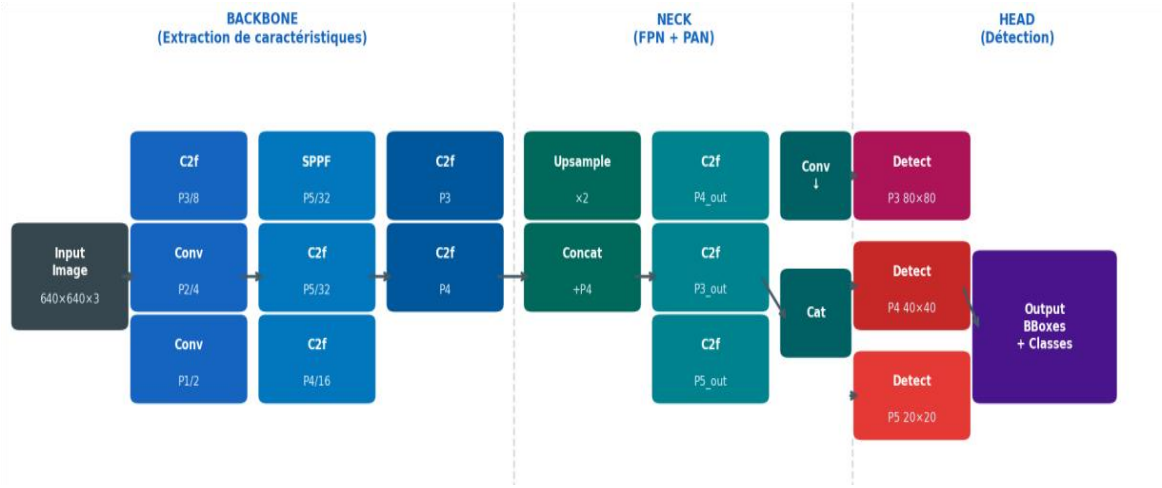


Fig 1.8 — Architecture complète de YOLOv8 : Backbone, Neck (FPN-PAN) et tête découplée

5.3. PaddleOCR

PaddleOCR est un moteur OCR open source développé par Baidu [2]. Il propose un pipeline complet : détecteur DB (Differentiable Binarization) + module CRNN avec attention. Ses avantages : précision >95 % (benchmarks), vitesse rapide, support de 80+ langues, API Python native.

5.4. OpenCV

OpenCV (Open Source Computer Vision Library, Apache 2.0) est la bibliothèque de référence pour le traitement d'images [19]. Elle gère le prétraitement des images de plaques (CLAHE, débruitage, rectification de perspective, binarisation adaptative) entre la détection YOLOv8n et la reconnaissance PaddleOCR.

5.5. Tkinter et NumPy

Tkinter est la bibliothèque standard de Python pour les interfaces graphiques (GUI). Elle constitue la couche de présentation de notre application ANPR. NumPy fournit les tableaux multidimensionnels servant de format d'échange standard entre toutes les bibliothèques du pipeline [21].

Moteur OCR	Précision	Vitesse	Langues	Intégration Python	Choix ?
Tesseract	Moyenne	Lente	100+	Via pytesseract	✘
EasyOCR	Bonne	Moyenne	80+	API Python native	—
PaddleOCR	>95 % (benchmarks)	Rapide	80+	API Python native	✔

Tableau 1.5 — Comparaison des moteurs OCR

Bibliothèque / Outil	Version	Rôle dans le projet
Python	3.10+	Langage principal — orchestration du pipeline
YOLOv8 (Ultralytics)	8.x	Détection et localisation des plaques en temps réel
PaddleOCR	2.7+	Reconnaissance optique des caractères
OpenCV	4.x	Prétraitement des images, manipulation ROI
Tkinter	Intégré Python	Interface graphique (GUI) de l'application
NumPy	1.24+	Manipulation des tableaux d'images
PyTorch	2.x	Backend de calcul pour YOLOv8

Tableau 1.6 — Récapitulatif des bibliothèques et outils utilisés

Conclusion

Ce premier chapitre a posé les fondements théoriques et technologiques sur lesquels repose notre système ANPR. Nous avons introduit les concepts de l'IA et de l'apprentissage automatique supervisé, l'apprentissage profond et les CNN qui constituent le socle de YOLOv8n, ainsi que les systèmes ANPR et la plaque algérienne avec ses spécificités. La revue des approches de l'état de l'art a motivé nos choix technologiques. Les outils retenus forment un écosystème technologique cohérent et performant.

Chapitre 2 Conception et Modélisation du Système

Introduction

Le chapitre précédent a posé les fondements théoriques de notre système en présentant les concepts clés de l'apprentissage automatique, de la vision par ordinateur et des outils logiciels retenus. Fort de ces bases, le présent chapitre aborde la phase de conception et de modélisation du système Intelligent Gate — Système Intelligent de Reconnaissance Automatique des Plaques d'Immatriculation Algériennes. Cette étape constitue le pont essentiel entre la réflexion théorique et l'implémentation concrète.

Nous débutons par une analyse rigoureuse des besoins fonctionnels et non-fonctionnels du système, avant d'en présenter l'architecture globale fondée sur un pipeline de traitement en six étapes. La modélisation est ensuite formalisée à l'aide du langage UML, au travers de diagrammes de cas d'utilisation, de séquence, d'activités et de composants. Enfin, nous détaillons la conception de chacun des modules fonctionnels, incluant les deux nouveaux modules clés du système : le gestionnaire de liste blanche (WhitelistManager) et le simulateur de barrière de passage (GateSimulator).

1. Analyse des Besoins

L'analyse des besoins est la première étape incontournable de toute démarche de conception logicielle. Elle permet de clarifier ce que le système doit réaliser (besoins fonctionnels) et sous quelles contraintes il doit opérer (besoins non-fonctionnels), tout en identifiant les acteurs qui interagissent avec lui.

1.1 Besoins Fonctionnels

Les besoins fonctionnels décrivent l'ensemble des fonctions que le système Intelligent Gate doit offrir à ses utilisateurs. Ils couvrent désormais six modules fonctionnels, incluant le contrôle d'accès par liste blanche et la simulation de la barrière physique de passage.

ID	Module	Description du besoin
BF01	**Acquisition**	Le système doit permettre le chargement d'une image depuis le système de fichiers (formats JPG, PNG, BMP).
BF02	**Acquisition**	Le système doit permettre la capture d'image en temps réel via la webcam de l'ordinateur.
BF03	**Acquisition**	Le système doit détecter automatiquement tout mouvement d'objet devant la caméra grâce à la soustraction de fond (MOG2).
BF04	**Acquisition**	L'utilisateur doit pouvoir ajuster la sensibilité de la détection de mouvement via un curseur gradué de 1 à 20.
BF05	**Détection**	Le système doit localiser et extraire automatiquement la région de la plaque d'immatriculation dans l'image en utilisant le modèle YOLOv8n.
BF06	**Détection**	Le système doit redresser géométriquement la plaque extraite par transformation de perspective pour corriger les distorsions de prise de vue.
BF07	**OCR**	Le système doit reconnaître les caractères de la plaque via PaddleOCR après une séquence de prétraitements d'image.
BF08	**OCR**	Le système doit filtrer le résultat OCR pour ne conserver que les chiffres (0–9) et corriger les substitutions courantes (O→0, I→1).
BF09	**Liste blanche**	Le système doit vérifier automatiquement si la plaque reconnue figure dans la liste blanche (allowed_vehicles.json).
BF10	**Liste blanche**	L'opérateur doit pouvoir ajouter, supprimer et rechercher des véhicules autorisés dans la liste blanche (plaque, propriétaire, véhicule, couleur).
BF11	**Barrière**	Le système doit ouvrir automatiquement la barrière de passage (animation Canvas) pour tout véhicule autorisé, et déclencher une alerte visuelle de refus pour tout véhicule non autorisé.

BF12	**Barrière**	La barrière doit se fermer automatiquement après 5 secondes d'ouverture et doit pouvoir être commandée manuellement par l'opérateur.
BF13	**Journalisation**	Le système doit enregistrer automatiquement chaque reconnaissance dans plates_log.csv avec : ID, numéro de plaque, date, heure et chemin de l'image.
BF14	**Interface**	Le système doit afficher un tableau de l'historique avec filtres de recherche par numéro et par date, ainsi qu'une horloge en temps réel et un compteur total.

Tableau 2.1 : Besoins fonctionnels du système Intelligent Gate

1.2 Besoins Non-Fonctionnels

Les besoins non-fonctionnels définissent les critères de qualité que le système doit respecter, indépendamment des fonctionnalités qu'il offre. Ils couvrent les aspects de performance, de fiabilité, d'ergonomie et de maintenabilité, tels que détaillés dans le Tableau 2.2.&

Catégorie	Indicateur	Exigence
Performance	Temps de réponse	Le délai total du pipeline (acquisition → affichage résultat) ne doit pas dépasser 3 secondes sur un CPU standard.
Performance	Inférence YOLO	Le modèle YOLOv8n doit traiter chaque image en moins de 250 ms sur CPU Intel Core i5.
Performance	Animation barrière	La transition d'ouverture/fermeture (20 pas) doit être fluide et s'effectuer en 600 ms maximum.
Précision	Détection	Le modèle doit atteindre un mAP@0.5 d'au moins 95% sur le jeu de test dédié aux plaques algériennes.
Précision	OCR	Le taux de reconnaissance correct des chiffres doit dépasser 80% sur des images de qualité standard.
Fiabilité	Gestion des erreurs	Le système doit gérer sans interruption les cas d'absence de plaque, de plaque illisible, de caméra déconnectée et de liste blanche vide.
Ergonomie	Interface utilisateur	L'interface doit être utilisable sans formation préalable, avec des retours visuels clairs et codés par couleur (vert/rouge/orange).
Maintenabilité	Modularité	Le code doit être structuré en modules indépendants, permettant le remplacement ou l'amélioration de chaque composant isolément.
Portabilité	Environnement	Le système doit fonctionner sous Windows 10/11 avec Python 3.10 et les dépendances listées, sans GPU dédié.
Sécurité	Données locales	Toutes les données (CSV, JSON) sont stockées

		localement, sans transmission réseau externe.

Tableau 2.2 : Besoins non-fonctionnels du système Intelligent Gate

1.3 Identification des Acteurs

L'analyse du système a permis d'identifier trois acteurs interagissant avec le système Intelligent Gate :

- L'Opérateur (utilisateur humain) : acteur principal. Il interagit avec l'interface graphique pour charger des images, déclencher la capture webcam, gérer la liste blanche des véhicules autorisés, contrôler manuellement la barrière et consulter l'historique des reconnaissances ;
- La Barrière de Passage (acteur système physique) : acteur secondaire représentant le dispositif de contrôle d'accès simulé. Elle reçoit des commandes d'ouverture/fermeture depuis le module GateSimulator en fonction du résultat de la vérification d'autorisation ;
- Le Système de Fichiers (acteur secondaire, environnement) : support de persistance des données — images source, fichier plates_log.csv et base de données JSON allowed_vehicles.json. Il interagit de façon transparente avec les modules d'acquisition, de journalisation et de gestion de la liste blanche.

2. Architecture Globale du Système

L'architecture du système Intelligent Gate est fondée sur un pipeline de traitement séquentiel en six étapes, depuis l'acquisition de l'image jusqu'au contrôle physique de la barrière. Cette organisation modulaire assure une séparation claire des responsabilités et facilite la maintenance et l'évolution du système.

2.1 Vue d'Ensemble de l'Architecture

Le système est organisé en trois couches fonctionnelles : la couche de Présentation (interface Tkinter et simulateur Canvas), la couche de Traitement (modules métier), et la couche de Données (fichiers de persistance et modèles IA). Ces couches communiquent de façon unidirectionnelle selon le flux d'exécution du pipeline ci-dessous.

Le pipeline de traitement intégral se déroule en six étapes séquentielles :

- Étape 1 — Acquisition : capture de l'image source (fichier ou webcam avec détection de mouvement MOG2);
- Étape 2 — Détection : localisation de la plaque par YOLOv8n (bounding box, IoU ≥ 0.5);
- Étape 3 — Extraction : découpage de la ROI et correction géométrique par transformation de perspective ;
- Étape 4 — Reconnaissance OCR : prétraitement (Lanczos ×4, CLAHE, débruitage) puis lecture par PaddleOCR ;
- Étape 5 — Vérification d'autorisation : consultation de la liste blanche (allowed_vehicles.json) par WhitelistManager;
- Étape 6 — Contrôle de la barrière : ouverture animée (véhicule autorisé) ou alerte de refus (véhicule non autorisé) via GateSimulator, suivie de la journalisation CSV.

2.2 Description des Composants

Le système compte six modules logiciels principaux à responsabilité unique :

- Module d'Acquisition : gestion du flux webcam, analyse de mouvement (MOG2), sauvegarde du frame capturé (camera_capture.jpg);

- Module de Détection (YOLOv8n) : chargement du modèle best.pt, inférence, extraction de la ROI, transformation de perspective ;
- Module de Reconnaissance (PaddleOCR) : prétraitement de la ROI, reconnaissance des caractères, filtrage regex et substitutions de caractères ambigus ;
- WhitelistManager : lecture/écriture du fichier allowed_vehicles.json, vérification d'autorisation par comparaison de chiffres, gestion CRUD des véhicules autorisés ;
- GateSimulator : animation Canvas de la barrière (ouverture/fermeture en 20 pas de 30 ms), gestion des états (CLOSED/OPENING/OPEN/CLOSING), statistiques journalières, fermeture automatique à 5 secondes ;
- Module de Journalisation & Interface (Tkinter) : orchestration du pipeline, affichage des résultats, écriture CSV, Treeview avec filtres dynamiques.

2.3 Choix Architecturaux et Justifications

Le choix d'une architecture en pipeline séquentiel est justifié par la nature dépendante du traitement : chaque étape requiert impérativement le résultat de l'étape précédente, rendant tout parallélisme inutile et potentiellement source d'incohérence. Le format JSON pour la liste blanche (allowed_vehicles.json) a été retenu pour sa lisibilité, sa flexibilité de schéma et sa facilité de modification manuelle par l'opérateur, en complément de l'interface graphique dédiée.

L'animation Canvas du GateSimulator, bien que simulée, reproduit fidèlement la logique d'une barrière physique réelle (état machine, temporisation, signaux visuels codés par couleur) permettant ainsi de valider le comportement du système de contrôle d'accès sans matériel physique. Cette approche est conforme aux pratiques de prototypage rapide préconisées dans le domaine des systèmes embarqués.

3. Modélisation UML

La modélisation du système Intelligent Gate est réalisée à l'aide du langage UML (Unified Modeling Language) [23]. Nous présentons quatre diagrammes complémentaires — cas d'utilisation, séquence, activités et composants — offrant une vue complète du système sous ses dimensions statique et dynamique.

3.1 Diagramme de Cas d'Utilisation

Le diagramme de cas d'utilisation (Figure 2.1) représente les interactions entre les trois acteurs identifiés et les quatorze cas d'utilisation du système. Les nouveaux cas d'utilisation liés au contrôle d'accès — UC5 (vérification liste blanche), UC6 (contrôle barrière), UC10 (gestion liste blanche), UC11 (ajout/suppression de véhicule), UC12 (contrôle manuel) — s'intègrent naturellement dans la chaîne de traitement existante.

Le flux principal (UC1/UC2 → UC3 → UC4 → UC5 → UC6 → UC7) est entièrement automatisé et suit le pipeline en six étapes. Les cas d'utilisation UC10 à UC13 sont des fonctions de gestion et de supervision accessibles à l'opérateur indépendamment du flux de reconnaissance.

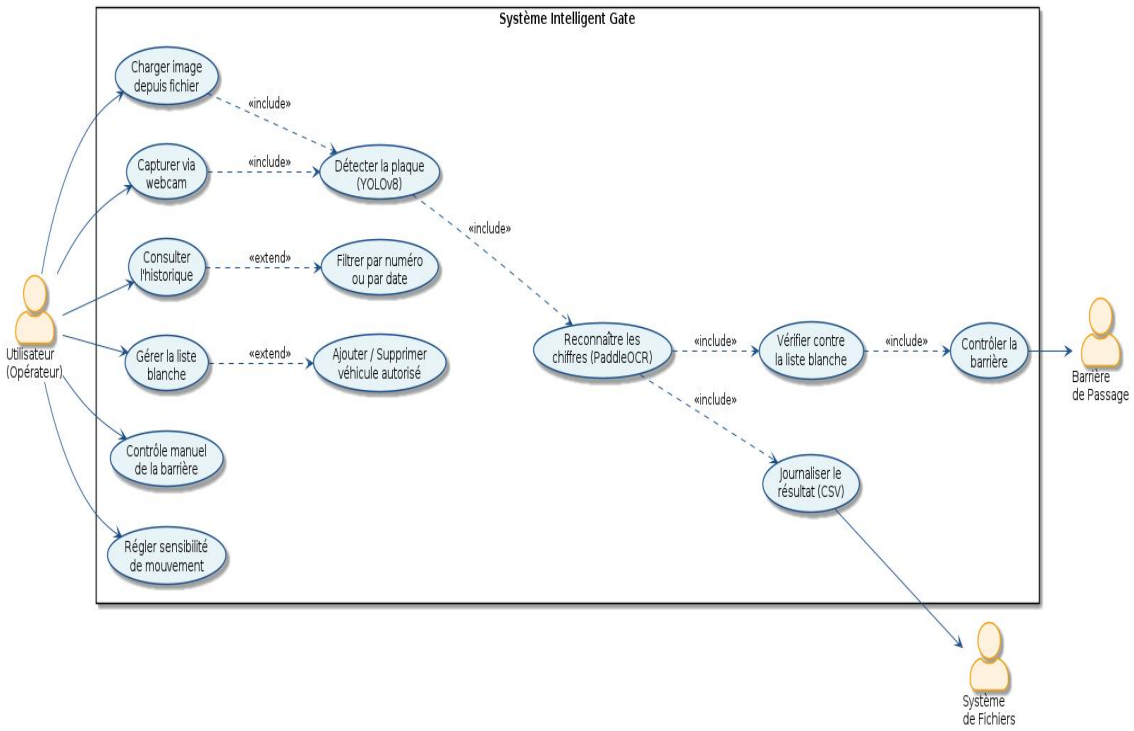


Fig 2.1 : Diagramme de cas d'utilisation du système Intelligent Gate

3.2 Diagramme de Séquence — Flux Principal

Le diagramme de séquence (Figure 2.2) détaille le déroulement temporel du flux principal impliquant sept composants logiciels. Par rapport à l'architecture initiale à quatre étapes, deux nouveaux participants apparaissent : WhitelistManager et GateSimulator, intercalés entre la sortie OCR et la journalisation CSV.

Le point de décision central est la réponse de WhitelistManager.is_authorized() : si le véhicule est autorisé, GateSimulator déclenche animate_open() puis planifie la fermeture automatique après 5 000 ms. Dans le cas contraire, _flash_denied() provoque 6 itérations de 200 ms (3 ON et 3 OFF, soit 1 200 ms) de clignotements rouges accompagnés d'un signal sonore système (root.bell()), signalant clairement le refus à l'opérateur.

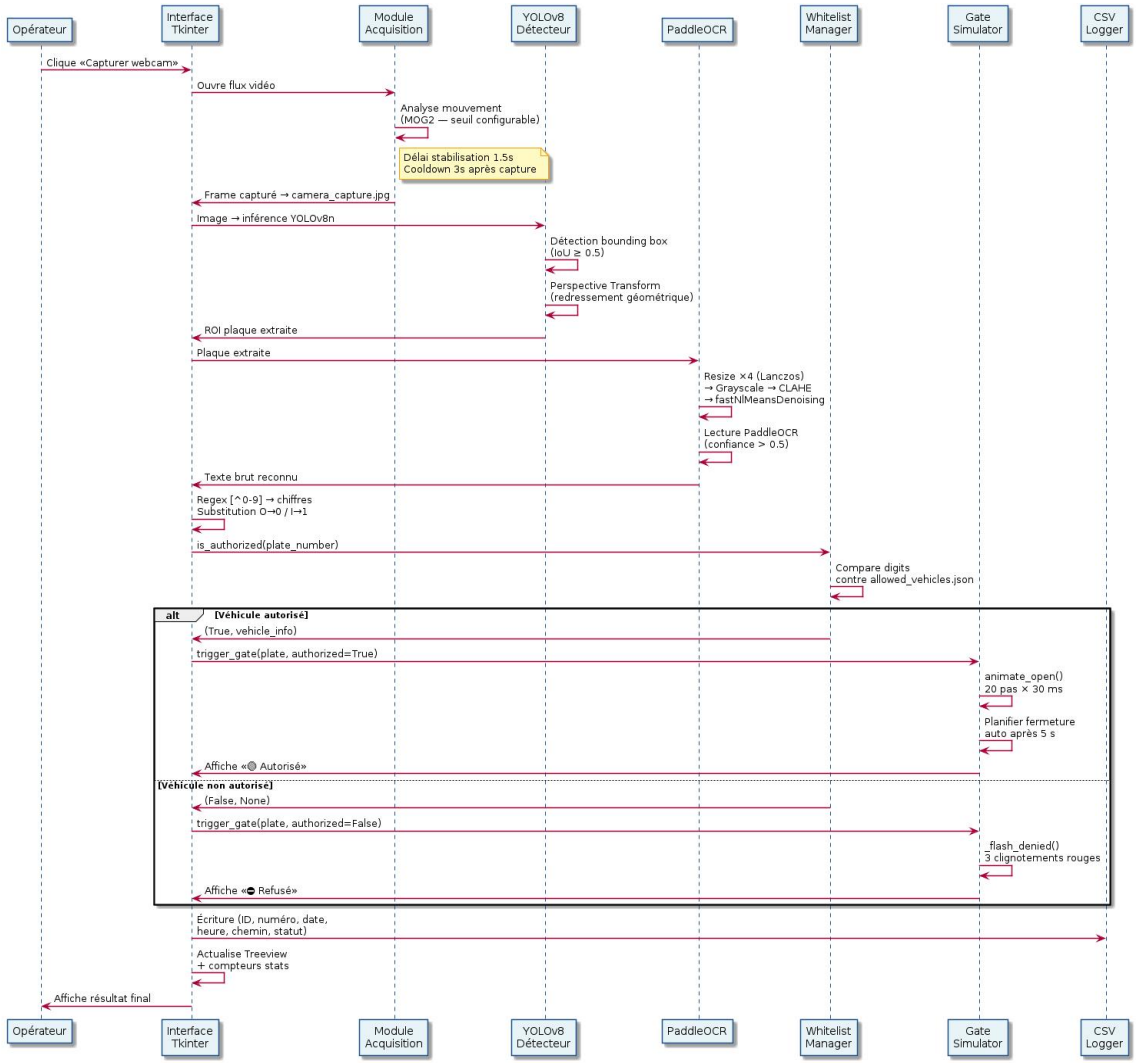


Fig 2.2 : Diagramme de séquence — Flux de reconnaissance et contrôle d'accès

3.3 Diagramme d'Activités — Pipeline de Traitement

Le diagramme d'activités (Figure 2.3) décrit le flux de contrôle complet du système, de l'acquisition jusqu'au contrôle de la barrière. Il met en évidence cinq points de décision conditionnels successifs, dont la nouvelle branche d'autorisation (« Véhicule autorisé ? ») qui constitue le cœur logique du système de contrôle d'accès.

Les deux branches de ce point de décision produisent des comportements visuellement distincts : l'animation d'ouverture verte (animate_open) pour les véhicules autorisés et le clignotement rouge de refus (_flash_denied) pour les véhicules non autorisés. Dans les deux cas, les statistiques journalières (total / autorisés / refusés) sont mises à jour et le Treeview est actualisé.

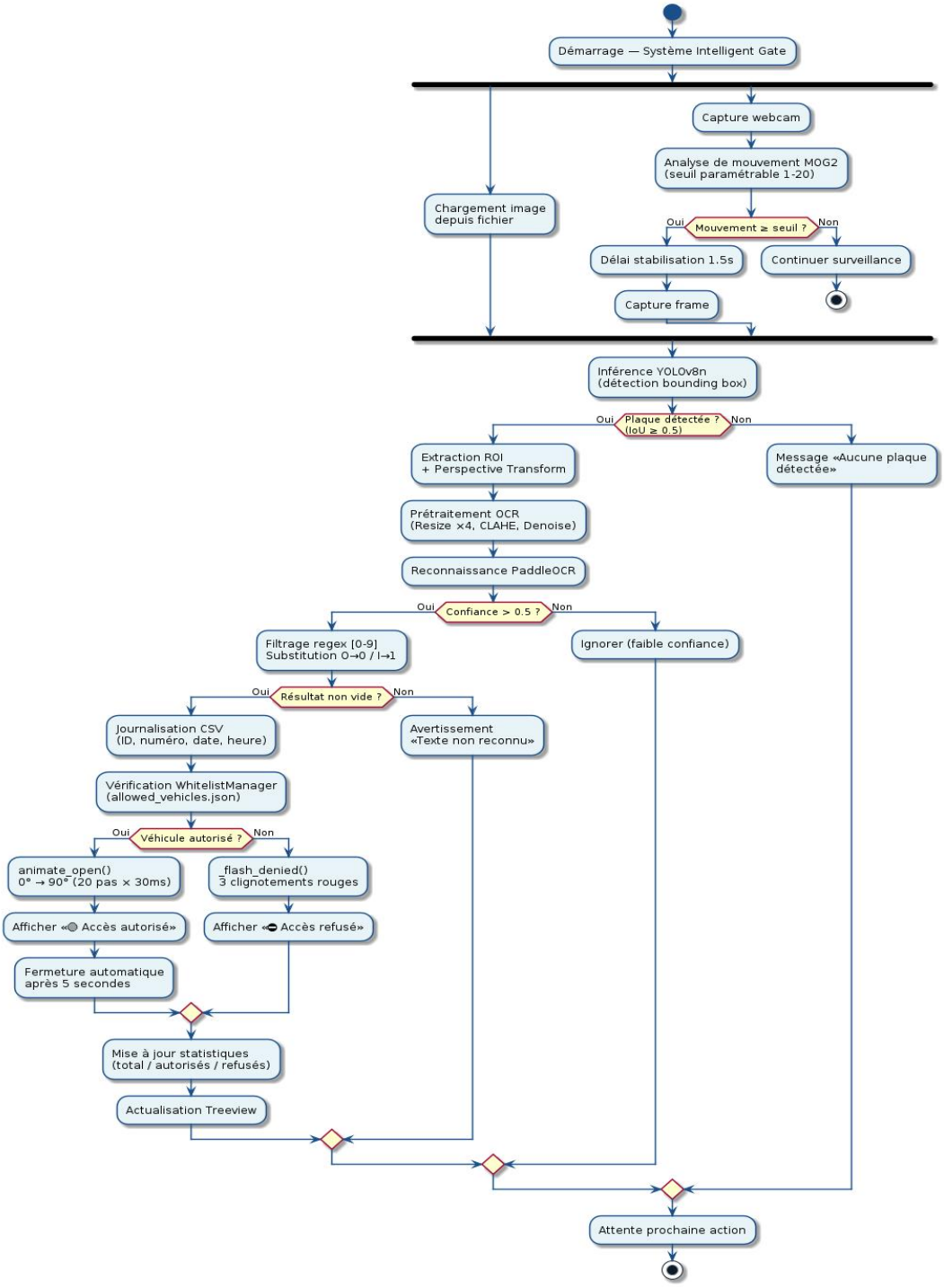


Fig 2.3 *:* Diagramme d'activités — Pipeline de traitement complet

3.4 Diagramme de Composants — Architecture Modulaire

Le diagramme de composants (Figure 2.4) offre une vue architecturale de haut niveau représentant les six composants logiciels et leurs dépendances, organisés en trois couches. Par rapport à la version initiale, deux nouveaux composants ont été ajoutés à la couche Présentation et à la couche Traitement : GateSimulator et WhitelistManager.

WhitelistManager dépend directement de la ressource JSON allowed_vehicles.json et expose un résultat d'autorisation binaire vers GateSimulator. Ce dernier est positionné dans la couche Présentation car son rôle principal est l'affichage animé de l'état de la barrière. Cette séparation garantit que la logique de vérification (WhitelistManager) est indépendante de la couche de présentation, conformément au principe de séparation des préoccupations [23].

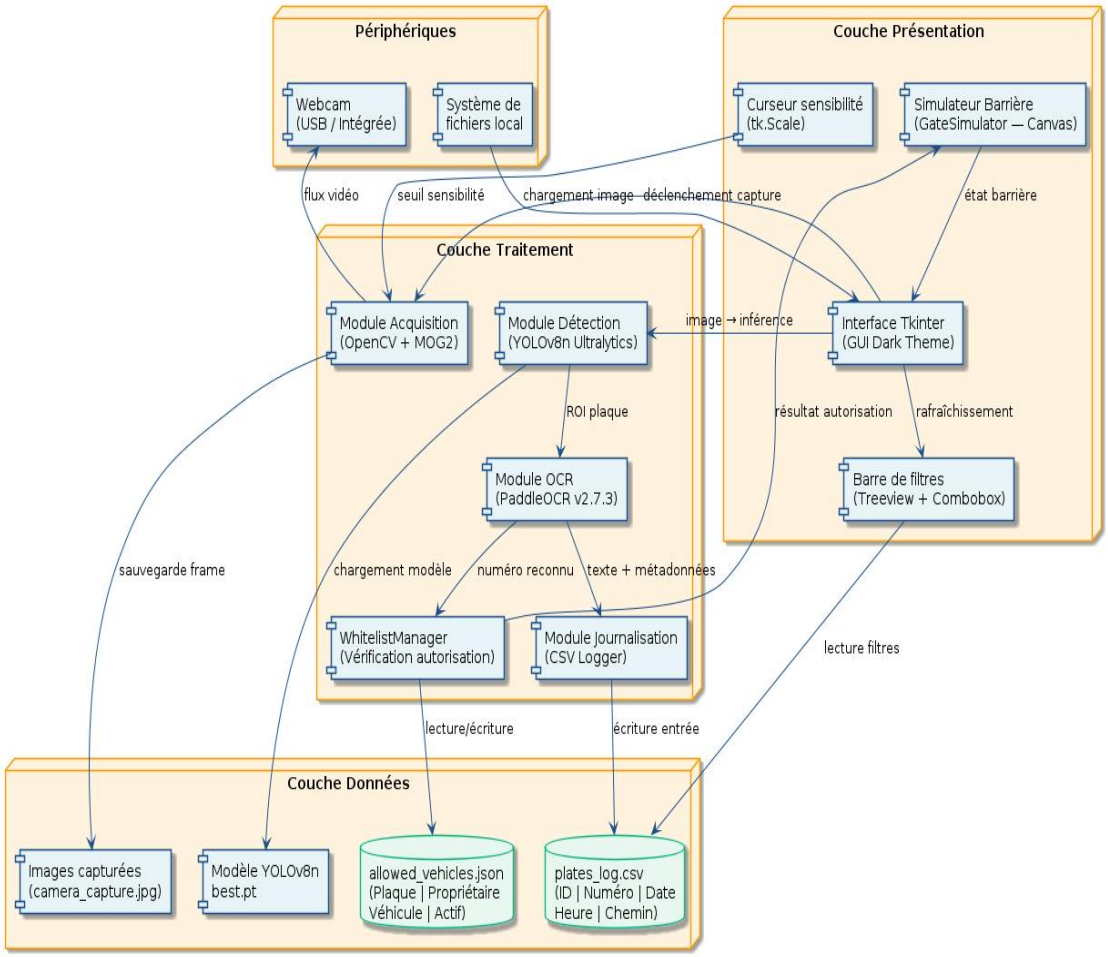


Fig 2.4 : Diagramme de composants — Architecture modulaire du système

4. Conception Détaillée des Modules

Cette section présente la conception interne de chacun des six modules fonctionnels du système, en précisant les algorithmes utilisés, les paramètres retenus et les justifications des choix de conception.

4.1 Module d'Acquisition et Détection de Mouvement

4.1.1 Chargement depuis fichier

L'opérateur sélectionne un fichier image via tkinter.filedialog.askopenfilename(). L'image est chargée par OpenCV (cv2.imread), validée, puis transmise directement au Module de Détection. Les formats supportés sont JPG, JPEG, PNG et BMP.

4.1.2 Capture webcam avec détection de mouvement

La détection de mouvement repose sur l'algorithme MOG2 (cv2.createBackgroundSubtractorMOG2, history=500, varThreshold=50), appliqué à chaque frame du flux webcam. Le masque de premier plan est affiné par une ouverture morphologique (noyau elliptique 5×5) puis une dilatation (iterations=2). Le pourcentage de pixels en mouvement est calculé comme : motion_percent = (NonZeroPixels / TotalPixels) × 100.

Le seuil de déclenchement est paramétré par un curseur tk.Scale (plage 1–20, valeur par défaut 5) stocké dans un tk.IntVar partagé avec le thread de capture. Une temporisation de 1,5 seconde est appliquée avant la capture pour stabiliser la scène, suivie d'un cooldown de 3 secondes pour éviter les captures multiples d'un même véhicule.

4.2 Module de Détection YOLOv8n

4.2.1 Modèle et entraînement

Le modèle YOLOv8n (nano) a été entraîné par transfer learning depuis les poids pré-entraînés yolov8n.pt (COCO) sur un dataset de 516 images de plaques algériennes annotées (Roboflow, licence CC BY 4.0). La répartition est : 361 images pour l'entraînement (70%), 103 pour la validation (20%) et 52 pour le test (10%). L'entraînement a été réalisé sur 40 époques avec batch=16 et imgsz=640, sur CPU Intel Core i5-7300U. Le modèle résultant (best.pt) atteint un mAP@0.5 de 100% sur le jeu de test [1].

4.2.2 Extraction de la ROI et transformation de perspective

Pour chaque image, le modèle retourne la boîte englobante de confiance maximale. La fonction order_points() trie les quatre coins (haut-gauche, haut-droit, bas-droit, bas-gauche), puis cv2.getPerspectiveTransform() calcule la matrice homographique et cv2.warpPerspective() produit une image rectangulaire rectifiée de la plaque.

4.3 Module de Reconnaissance OCR (PaddleOCR)

4.3.1 Pipeline de prétraitement

L'image de la plaque extraite subit quatre traitements successifs : (1) redimensionnement ×4 par interpolation Lanczos (cv2.INTER_LANCZOS4) pour augmenter la résolution effective des caractères ; (2) conversion en niveaux de gris (BGR→GRAY) ; (3) CLAHE (clipLimit=3.0, tileGridSize=(8,8)) pour corriger les variations d'éclairage locales ; (4) débruitage par cv2.fastNlMeansDenoising(h=10) pour réduire le bruit de capteur sans estomper les contours [3].

4.3.2 Reconnaissance et post-traitement

PaddleOCR v2.7.3 (use_angle_cls=True, lang='en') est appelé sur l'image prétraitée. Seuls les résultats de confiance supérieure à 0,5 sont retenus. Le post-traitement applique un double filtre : (1) filtrage regex re.sub(r'[^0-9]', '', text) pour ne conserver que les chiffres, et (2) substitutions de caractères ambigus (O→0, I→1) fréquemment confondus par les moteurs OCR sur les plaques d'immatriculation.

4.4 Module Interface et Journalisation

L'interface Tkinter orchestre l'ensemble du pipeline. Elle est structurée en quatre onglets (via ttk.Notebook) : 'Reconnaissance' pour le flux principal, 'Simulateur' pour la barrière, 'Historique' pour le journal d'accès et 'Paramètres' pour la liste blanche. La capture webcam s'exécute dans un thread daemon séparé, communiquant avec l'interface via root.after() pour garantir la thread-safety. La journalisation dans plates_log.csv inclut : ID auto-incrémenté, numéro de plaque, date (YYYY-MM-DD), heure (HH:MM:SS) et chemin de l'image source.

4.5 Module WhitelistManager — Gestion de la Liste Blanche

Le module WhitelistManager est responsable de la gestion de l'ensemble des véhicules autorisés à accéder au site surveillé. Il encapsule toutes les opérations de lecture et d'écriture sur le fichier de persistance allowed_vehicles.json, offrant une interface programmatique claire à l'interface graphique principale.

4.5.1 Structure du fichier de persistance

Les données sont stockées dans un fichier JSON structuré en deux sections distinctes : le tableau allowed_vehicles contenant les enregistrements des véhicules autorisés, et l'objet settings regroupant les paramètres de configuration globaux du système (délai de fermeture automatique, sensibilité par défaut, confirmation requise). Le Tableau 2.3 décrit le schéma de chaque enregistrement de véhicule.

Champ	Type	Description
plate	String	Numéro de plaque d'immatriculation (format libre, comparaison par chiffres extraits).
owner	String	Nom du propriétaire du véhicule (optionnel).
vehicle	String	Désignation du véhicule (marque, modèle — optionnel).
color	String	Couleur du véhicule (optionnel).
added_date	String	Date d'ajout au format YYYY-MM-DD (générée automatiquement).
active	Boolean	Indicateur d'activation : False désactive le véhicule sans le supprimer.

Tableau 2.3 : Structure d'un enregistrement dans allowed_vehicles.json

4.5.2 Algorithme de vérification d'autorisation

La méthode centrale is_authorized(plate) implémente une comparaison robuste aux variations de formatage : plutôt que de comparer directement les chaînes de caractères, elle extrait les chiffres des deux membres (plaque reconnue et entrées de la liste) via re.sub(r'[^0-9]', '', plate), puis compare les séquences de chiffres obtenues. Cette approche rend la vérification insensible aux espaces, tirets ou caractères alphabétiques parasites potentiellement introduits par l'OCR. La méthode retourne un tuple (authorized: bool, vehicle_info: dict | None) exploité directement par le module GateSimulator.

4.6 Module GateSimulator — Simulateur de Barrière de Passage

Le module GateSimulator implémente une simulation visuelle réaliste d'une barrière de contrôle d'accès, développée avec le widget Canvas de Tkinter. Son rôle est double : fournir un retour visuel immédiat à l'opérateur sur le résultat du contrôle d'accès et démontrer la faisabilité du couplage entre la reconnaissance de plaque et un système physique d'ouverture de barrière.

4.6.1 Machine à états de la barrière

La barrière est modélisée par une machine à quatre états distincts, dont les transitions sont pilotées par les résultats de WhitelistManager et les actions de l'opérateur. Le Tableau 2.4 décrit les transitions d'états possibles.

État source	État cible	Déclencheur	Action associée
CLOSED	**OPENING**	Véhicule autorisé détecté	Lancement de animate_open() — angle 0° → 90° en 20 pas de 30 ms.
OPENING	**OPEN**	Fin animation (step > 20)	Planification fermeture auto (root.after, 5 000 ms).
OPEN	**CLOSING**	Timeout 5 s ou contrôle manuel	Lancement de animate_close() — angle 90° → 0° en 20 pas de 30 ms.
CLOSING	**CLOSED**	Fin animation (step > 20)	Mise à jour de l'indicateur LED et du label de statut.
CLOSED	**CLOSED**	Véhicule non autorisé	Déclenchement de _flash_denied() — 6 itérations de 200 ms (3 ON et 3 OFF, soit 1 200 ms) + root.bell().

Tableau 2.4 : Machine à états du module GateSimulator

4.6.2 Rendu graphique et éléments visuels

L'animation est rendue par la méthode _draw_gate() appelée à chaque pas d'animation. Le rendu comprend : une route (rectangle sombre avec bandes de marquage orange), deux piliers de barrière, un bras de barrière dont la couleur est interpolée dynamiquement entre rouge (#FF5252) et vert (#00E676) selon l'angle courant, des points de marquage blancs réguliers sur le bras, et un indicateur LED tricolore (rouge/orange/vert) reflétant l'état courant. Un tableau de bord intégré affiche les statistiques journalières (total de véhicules, autorisés, refusés) ainsi que le numéro de la dernière plaque reconnue et l'horodatage de la dernière interaction.

5. Conception de la Persistance des Données

Le système Intelligent Gate utilise deux mécanismes de persistance complémentaires : un fichier CSV pour le journal des reconnaissances et un fichier JSON pour la gestion des véhicules autorisés. Cette dualité répond à des besoins distincts : le CSV est optimisé pour l'écriture séquentielle et la consultation chronologique, tandis que le JSON offre la flexibilité structurelle nécessaire à la gestion d'un référentiel de véhicules avec attributs multiples.

5.1 Journal de Reconnaissance — plates_log.csv

Le fichier plates_log.csv enregistre chaque événement de reconnaissance dans l'ordre chronologique. Sa structure (Tableau 2.5) est optimisée pour l'affichage dans le Treeview et le filtrage en mémoire. L'identifiant est auto-incrémenté en lisant la valeur maximale des entrées existantes avant chaque écriture. Le fichier est créé automatiquement avec ses en-têtes lors de la première reconnaissance réussie.

Colonne	Type	Format	Description
ID	Entier	Auto-incrémenté	Identifiant unique de l'enregistrement.
Plate_Number	Chaîne	Chiffres uniquement	Numéro reconnu après filtrage regex et substitutions.
Date	Chaîne	YYYY-MM-DD	Date de la reconnaissance (norme ISO 8601).
Time	Chaîne	HH:MM:SS	Heure précise de la reconnaissance.
Image_Path	Chaîne	Chemin absolu	Chemin vers l'image source utilisée pour la reconnaissance.

Tableau 2.5 : Structure du fichier plates_log.csv

5.2 Référentiel de Véhicules Autorisés — allowed_vehicles.json

Contrairement au CSV à écriture séquentielle, le fichier JSON est entièrement rechargé et réécrit à chaque opération de modification (ajout, suppression, mise à jour), ce qui est acceptable pour un référentiel de taille modeste (quelques centaines de véhicules). La méthode is_authorized() compare exclusivement les séquences de chiffres extraites des numéros de plaque, garantissant la robustesse aux variations de formatage introduites par l'OCR. Le champ active permet de suspendre

temporairement l'autorisation d'un véhicule sans le supprimer du référentiel, facilitant la gestion des accès temporaires.

Conclusion

Ce chapitre a présenté la conception complète du système Intelligent Gate dans sa version étendue incluant le contrôle d'accès intelligent. Nous avons établi un référentiel de quatorze besoins fonctionnels et dix besoins non-fonctionnels, puis défini les trois acteurs du système et leurs interactions. L'architecture en pipeline à six étapes — Acquisition, Détection, Extraction, Reconnaissance OCR, Vérification d'autorisation, Contrôle de la barrière — a été exposée et justifiée.

La modélisation UML réalisée en quatre diagrammes (cas d'utilisation, séquence, activités, composants) offre une représentation formelle du système sous ses dimensions statique et dynamique. La conception détaillée des six modules fonctionnels — en particulier WhitelistManager et GateSimulator, composants clés du système de contrôle d'accès — précise les algorithmes, les structures de données et les comportements attendus. La double persistance (CSV pour le journal, JSON pour la liste blanche) répond aux besoins spécifiques de chaque type de données.

L'ensemble de ces éléments de conception constitue le cahier des charges technique qui a guidé l'implémentation du système. Le chapitre suivant documente la réalisation concrète : environnement de développement, implémentation des modules, résultats expérimentaux et évaluation des performances globales du système Intelligent Gate.

Chapitre 3 Réalisation et Évaluation du Système Intelligent Gate

Les chapitres précédents ont posé les fondements théoriques et la conception architecturale du système Intelligent Gate. Le présent chapitre constitue la phase de validation expérimentale. Il documente l'architecture du modèle YOLOv8n retenu, son entraînement et l'analyse des courbes de convergence (Sections 3–3.5), puis présente l'évaluation indépendante sur le jeu de test (Section 4). Les Sections 5 et 6 couvrent le pipeline PaddleOCR et son évaluation OCR. La Section 7 centralise la synthèse globale des performances End-to-End.

1. Environnement de Développement et de Test

L'ensemble du développement et des tests a été réalisé sur une machine équipée d'un processeur Intel Core i5-7300U @ 2.60 GHz (2 cœurs / 4 threads), 8 Go DDR4, carte graphique intégrée Intel HD Graphics 620 (mode CPU uniquement), SSD 256 Go, sous Windows 10 (build 22621). La stack logicielle comprend : Python 3.10.0, Ultralytics YOLOv8 8.3.228, PaddleOCR 2.7.3, PaddlePaddle 2.6.2, OpenCV 4.6.0, PyTorch 2.9.1+cpu.

2. Présentation des Interfaces Graphiques

L'ensemble des interfaces adopte un thème sombre personnalisé (couleur de fond #0f1117, couleur d'accent #6c63ff). L'interface principale (ocrinterface.py) intègre : une zone d'affichage de l'image source et de la plaque détectée, des boutons d'action (chargement image statique, démarrage/arrêt webcam), un curseur de sensibilité de détection de mouvement (plage 1–20), un tableau historique (ttk.Treeview), une barre de statut avec horloge temps réel.

Figure — Interface principale — résultat de reconnaissance, véhicule autorisé (bandeau vert)

Le module gate_simulator.py implémente une animation Canvas simulant une barrière de passage. Le module whitelist_manager.py gère la liste blanche. Les

onglets « Journal d'accès » et « Paramètres » permettent respectivement la consultation de l'historique et la gestion des véhicules autorisés.

3. Modèle de Détection — YOLOv8n

Cette section présente l'architecture du modèle de détection de plaques d'immatriculation retenu et son processus d'entraînement. Conformément à la rigueur académique, la phase d'entraînement et l'évaluation sur le jeu de test sont traitées dans deux sections distinctes (Sections 3–4) afin de garantir une mesure de performance parfaitement objective.

3.1 Architecture YOLOv8n — Présentation du Modèle

3.1.1 Pourquoi YOLOv8n ?

YOLO (You Only Look Once) est une famille d'architectures de détection d'objets en temps réel, dont la version 8 (YOLOv8), développée par Ultralytics [1], représente l'état de l'art en matière de compromis précision/vitesse. Nous avons retenu la variante nano (YOLOv8n) pour trois raisons fondamentales :

- **Légèreté computationnelle :** 3 011 043 paramètres (3,01 M), ce qui est imposé par notre environnement CPU-only (Intel Core i5-7300U sans GPU dédié). Les variantes plus lourdes produiraient des temps d'inférence incompatibles avec le contrôle d'accès ;
- **Adaptation mono-classe :** le système ne détecte qu'une seule classe (license_plate), rendant les grandes architectures surdimensionnées et susceptibles de sur-apprentissage sur un dataset de 516 images ;
- **Efficacité du transfer learning :** YOLOv8n pré-entraîné sur COCO (330 000 images, 80 classes) possède déjà une capacité de détection de rectangles texturés, caractéristique des plaques d'immatriculation, permettant une convergence rapide.

.3.1.2 Structure de l'architecture

YOLOv8n adopte une architecture en trois composants : (1) le Backbone CSPDarknet avec mécanisme C2f extrait les features multi-échelles de l'image 640×640 px, (2) le Cou FPN+PAN fusionne les features de différentes résolutions pour améliorer la détection des petits objets, et (3) la Tête Decoupled Head sépare les

prédictions de localisation et de classification, améliorant la précision du recadrage par rapport aux architectures couplées.

Le Tableau 3.1 présente les caractéristiques architecturales précises du modèle utilisé, obtenues par inspection directe du fichier best.pt.

Caractéristique	Valeur	Remarque
Architecture	YOLOv8n	Variante nano — la plus légère de la famille YOLOv8
Nb. de paramètres	3 011 043 (≈ 3,01 M)	Confirmé par inspection directe de best.pt
Nombre de classes	1	Classe unique : license_plate
Résolution d'entrée	640 × 640 pixels	Compromis standard précision/vitesse YOLOv8
Backbone	CSPDarknet + C2f	Cross Stage Partial — optimise le flux des gradients
Cou (Neck)	FPN + PAN	Fusion multi-échelles pour objets petits et distants
Tête	Decoupled Head	Localisation et classification séparées
Pré-entraînement	yolov8n.pt (COCO)	Transfer learning depuis 330 000 images, 80 classes
Framework	Ultralytics 8.3.228	Backend PyTorch 2.9.1+cpu

Tableau 3.1 : Caractéristiques architecturales du modèle YOLOv8n utilisé

3.2 Dataset et Préparation des Données

3.2.1 Source et composition

Le dataset utilisé est le « Algerian License Plate 500 » (v1, format YOLOv5 PyTorch), obtenu via Roboflow sous licence CC BY 4.0 [7]. Il contient 516 images photographiques de plaques algériennes en conditions réelles, annotées au format YOLO. La diversité des conditions (angles, distances, éclairages) assure une bonne représentativité des cas opérationnels réels.

Le Tableau 3.2 présente la répartition des données selon les trois splits strictement disjoints. Les 52 images test n'ont jamais été exposées au modèle avant l'évaluation finale.

Split	Images	Proportion	Rôle
Train	361	69,9 %	Mise à jour des poids (rétropropagation)
Validation	103	20,0 %	Suivi des métriques — détection du sur-apprentissage
Test	**52**	10,1 %	Évaluation finale indépendante (jamais vue pendant l'entraînement)
Total	516	**100 %**	—

Tableau 3.2 : Répartition du dataset par split

3.2.2 Visualisation du dataset

La figure suivante illustre la distribution spatiale et dimensionnelle des annotations, confirmant la diversité effective des conditions photographiques intégrées dans le dataset

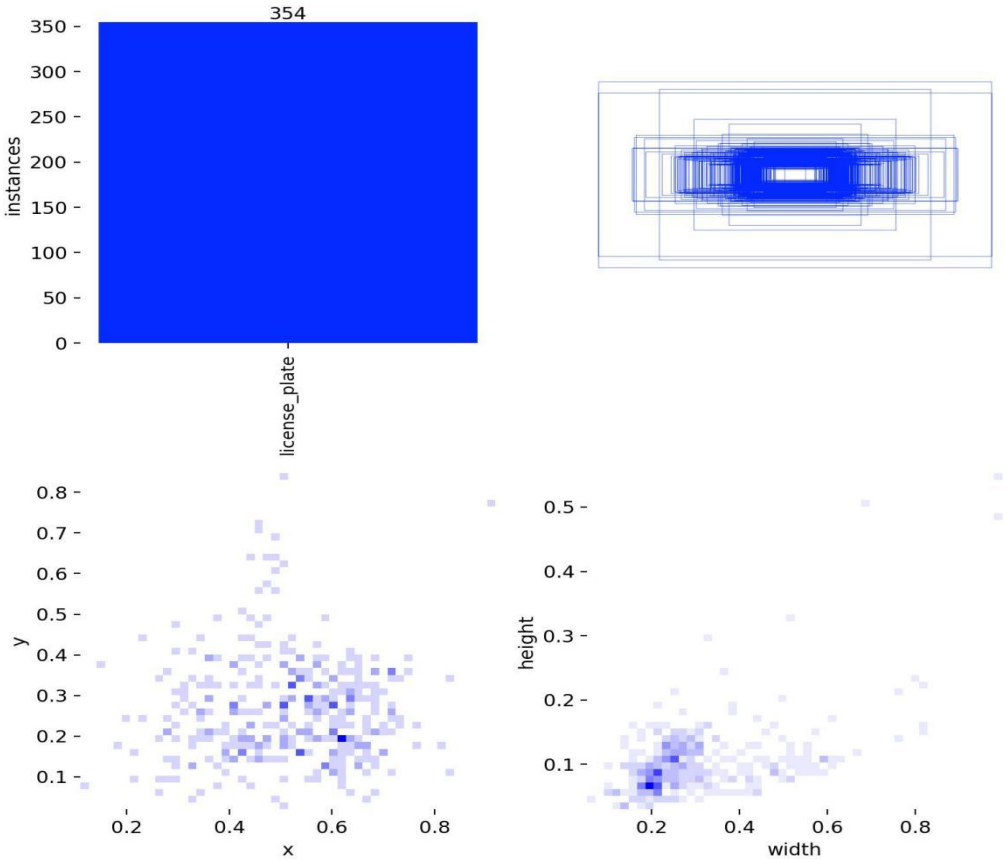


Fig3.1 : Distribution des annotations — taille et position des boîtes englobantes dans le dataset

Les deux lots ci-dessous montrent respectivement les premières et dernières images d'entraînement avec leurs annotations ground truth superposées, illustrant la cohérence du dataset du début à la fin de l'entraînement.

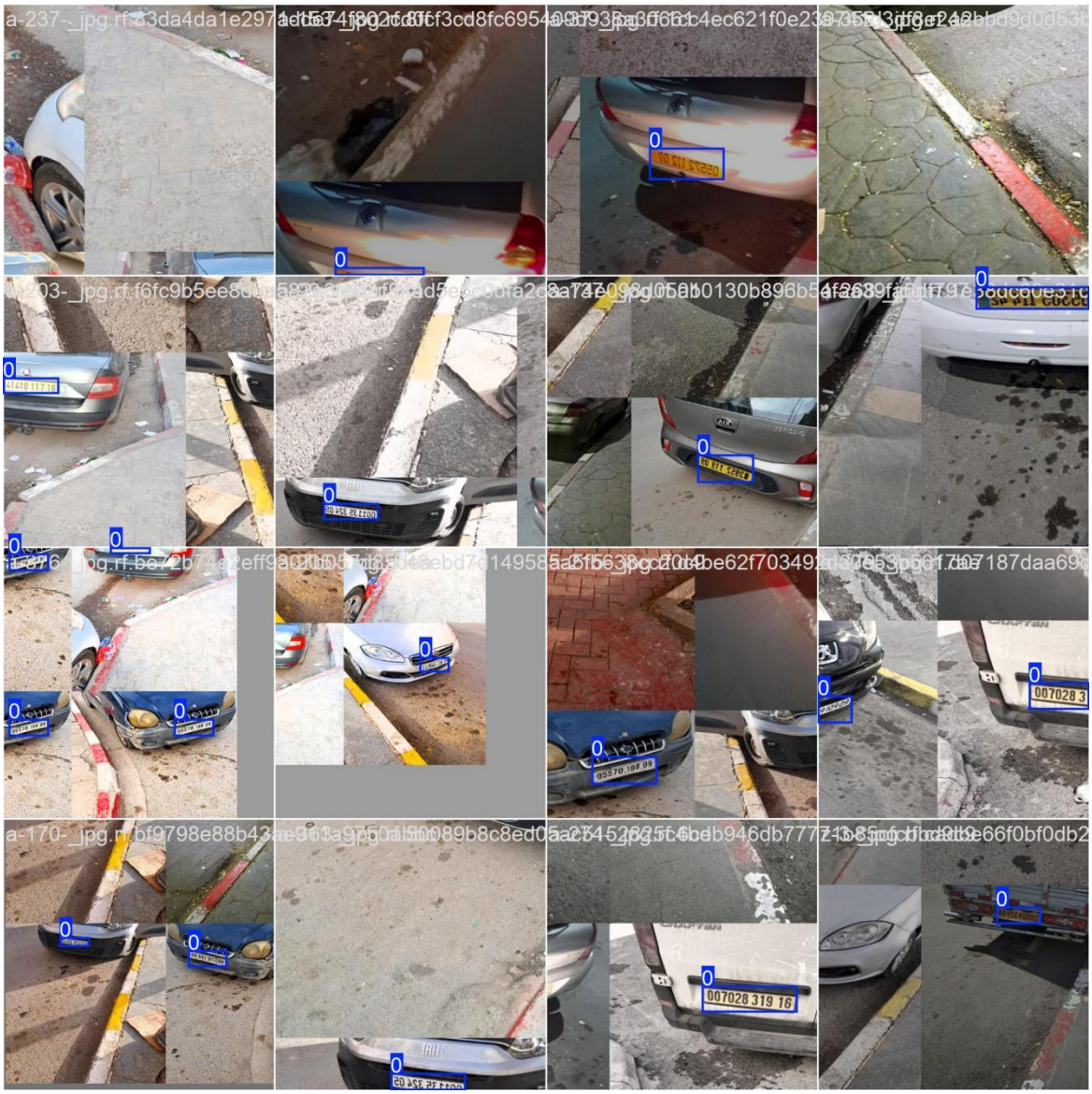


Fig 3.2 : Lot d'entraînement initial (batch 0) — images avec annotations ground truth superposées

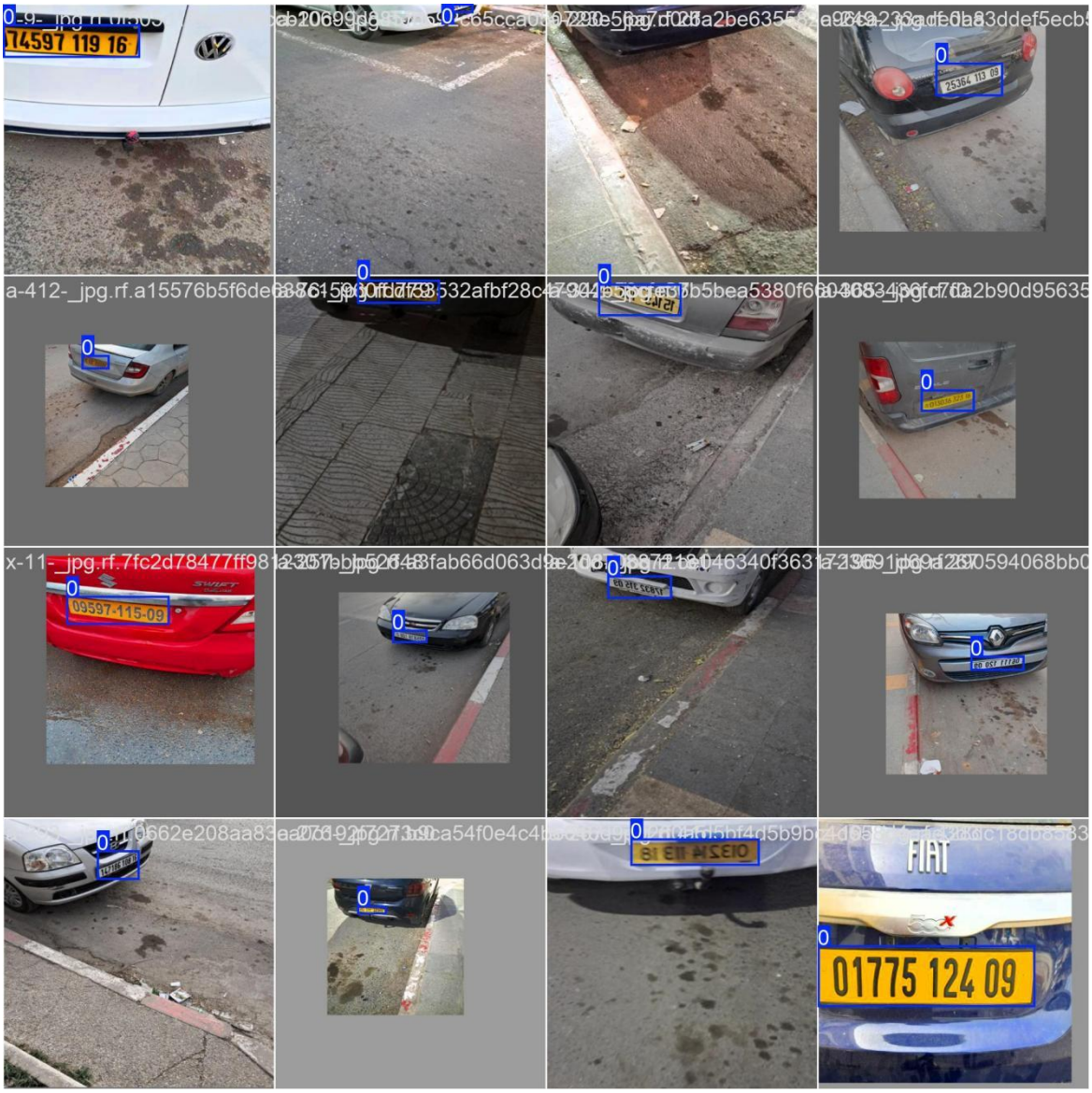


Fig 3.3 : Lot d'entraînement final (batch 690) — diversité des conditions en fin d'entraînement

3.3 Protocole et Paramètres d'Entraînement

L'entraînement a été réalisé par transfer learning à partir des poids pré-entraînés yolov8n.pt (COCO). Cette approche est justifiée pour un dataset de taille modeste (361 images) : les features bas-niveau apprises sur des millions d'images COCO sont directement transférables à la détection de plaques, permettant une convergence rapide et une meilleure généralisation.

Le Tableau 3.3 présente les hyperparamètres retenus avec leurs justifications. Ils ont été fixés après analyse des contraintes matérielles disponibles (CPU uniquement, 8 Go RAM).

Paramètre	Valeur retenue	Justification
Modèle de base	YOLOv8n	Variante nano — optimisée CPU, 3,01 M paramètres
Pré-entraînement	yolov8n.pt (COCO)	Transfer learning — convergence accélérée sur dataset limité
Époques	40	Convergence complète observée dès l'époque 36 — 40 époques évitent la sous-adaptation
Batch size	16	Adapté aux 8 Go de RAM disponibles en mode CPU
Taille d'image	640 × 640 px	Résolution standard YOLOv8 — précision/vitesse optimale
Optimiseur	AdamW (auto)	Sélection automatique Ultralytics — convergence stable sur petit dataset
Learning rate (lr0)	0,01	Taux de départ standard, décroissance cosinus jusqu'à 0,001
Warmup epochs	3,0	Montée progressive du lr — stabilise l'initialisation
Momentum (β)	0,937	Paramètre AdamW — mémoire des gradients passés
Weight decay	0,0005	Régularisation L2 — prévient le sur-apprentissage
Mosaic augmentation	1,0 (actif)	Assemblage de 4 images — multiplie la diversité effective
Flip horizontal	0,5 (50 %)	Symétrie gauche/droite — diversité des orientations
Close mosaic	10 dernières époques	Désactivation en fin d'entraînement pour affiner la localisation
Dispositif	CPU	Intel Core i5-7300U 2,60 GHz — aucun GPU dédié
AMP	Activé	Précision mixte automatique — réduit la mémoire RAM
Graine (seed)	0	Reproductibilité complète des résultats

Tableau 3.3 : Hyperparamètres d'entraînement YOLOv8n

3.4 Analyse des Courbes d'Entraînement

Les courbes d'entraînement sont un outil diagnostique essentiel pour vérifier la convergence, détecter le sur-apprentissage et valider la qualité globale de l'apprentissage. La Figure 3.4 synthétise l'évolution des losses et des métriques sur l'ensemble des 40 époques.

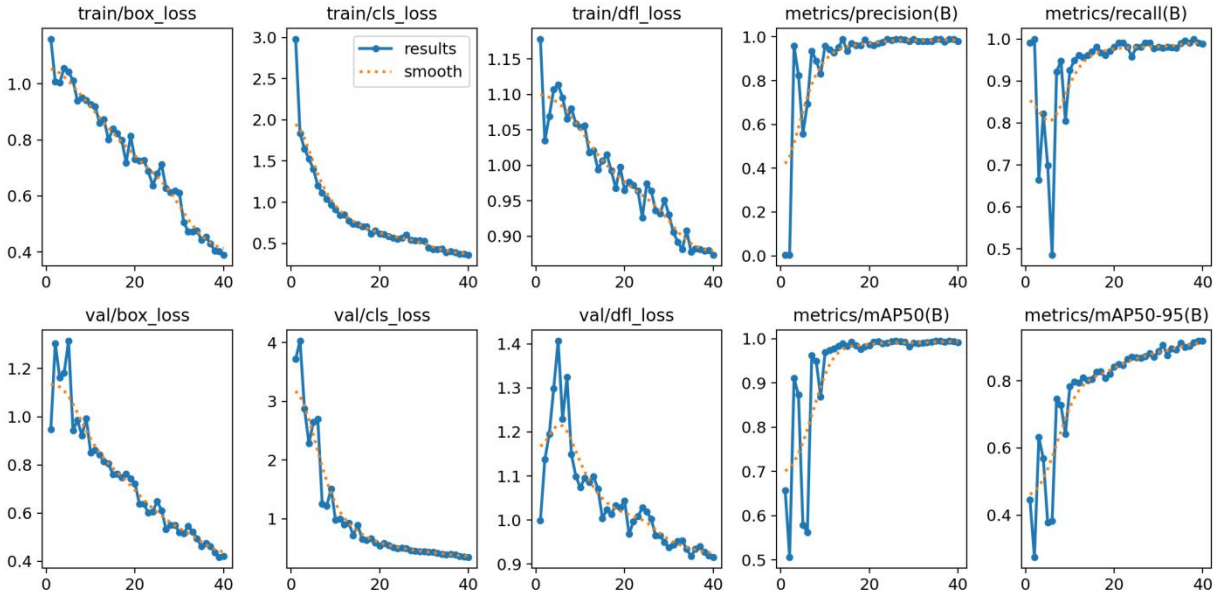


Fig 3.4 : Courbes d'entraînement complètes — Losses (train/val) et métriques sur 40 époques

Le Tableau 3.4 présente l'évolution aux époques jalons les plus significatives permettant de suivre les trois phases identifiées.

Époque	Box Loss (train)	Cls Loss (train)	Box Loss (val)	Precision	Recall	mAP@0.5	Observation
1	1,1598	2,9811	0,9475	0,33 %	99,03 %	65,70 %	Phase init. — Recall élevé, Precision quasi nulle
5	1,0425	1,4026	1,3147	55,74 %	69,90 %	57,90 %	Instabilité — ajustement du lr
10	0,9262	0,9117	0,8509	95,97 %	92,53 %	96,87 %	Apprentissage principal actif
16	0,8229	0,7033	0,7634	96,94 %	98,06 %	99,29 %	mAP@0.5 ≥ 99 % pour la première fois
30	0,6114	0,5282	0,5195	98,97 %	98,06 %	99,07 %	Stabilisation — fermeture mosaic proche
35	0,4428	0,3930	0,4612	99,03 %	98,95 %	99,46 %	Phase raffinement post-mosaic
36 ★	0,4544	0,4013	0,4752	99,03 %	99,55 %	99,48 %	**★ Meilleur mAP@0.5 → best.pt sauvegardé**
38	0,4049	0,3694	0,4353	99,04 %	99,90 %	99,44 %	Meilleur Recall observé (99,90 %)

Époque	Box Loss (train)	Cls Loss (train)	Box Loss (val)	Precision	Recall	mAP@0.5	Observation
40	0,3892	0,3584	0,4202	98,07 %	98,78 %	99,15 %	Modèle final — convergence complète

Tableau 3.4 : Évolution des métriques aux époques jalons (données exactes du fichier results.csv)

3.4.1 Phase initiale — Calibration (Époques 1–5)

La phase initiale est marquée par une forte instabilité typique du transfer learning. À l'époque 1, la Precision chute à 0,33 % tandis que le Recall atteint déjà 99,03 % : le modèle détecte « tout » sans encore discriminer. La box_loss élevée (1,16) reflète l'ajustement initial des têtes de détection au nouveau domaine.

3.4.2 Phase d'apprentissage principal (Époques 6–30)

Cette phase constitue le cœur de l'apprentissage. Les losses décroissent de façon monotone : la box_loss train passe de 1,01 à 0,61 (−39,6 %). Le mAP@0.5 dépasse 99 % dès l'époque 16 — après seulement 40 % de l'entraînement — validant l'efficacité du transfer learning depuis COCO. La val/box_loss décroît parallèlement à la train/box_loss, signe d'une bonne généralisation sans sur-apprentissage.

3.4.3 Phase de raffinement post-Mosaic (Époques 31–40)

La désactivation de la Mosaic Augmentation (close_mosaic=10) permet au modèle d'affiner la localisation sur des images non augmentées. La box_loss passe de 0,51 à 0,39 en 10 époques (−23,5 %). Le mAP@0.5 atteint son maximum absolu de 99,48 % à l'époque 36 — Ultralytics sauvegarde automatiquement best.pt à cet instant. Le mAP@0.5:0.95 atteint 91,84 % à l'époque finale.

3.4.4 Absence de sur-apprentissage

L'écart entre les losses train et validation à l'époque finale :

train/box_loss = 0,389 | val/box_loss = 0,420 → Écart relatif = **7,95 %** (inférieur au seuil critique de 15 %)

Ce résultat confirme l'absence de sur-apprentissage : le modèle généralise efficacement. Cet écart faible s'explique par la régularisation L2 (weight_decay = 0,0005), les augmentations actives et le transfer learning.

3.5 Courbes d'Évaluation sur la Validation

Les courbes spécialisées ci-dessous ont été générées sur le jeu de validation et complètent l'analyse en fournissant une vue orientée vers les métriques de détection standardisées.

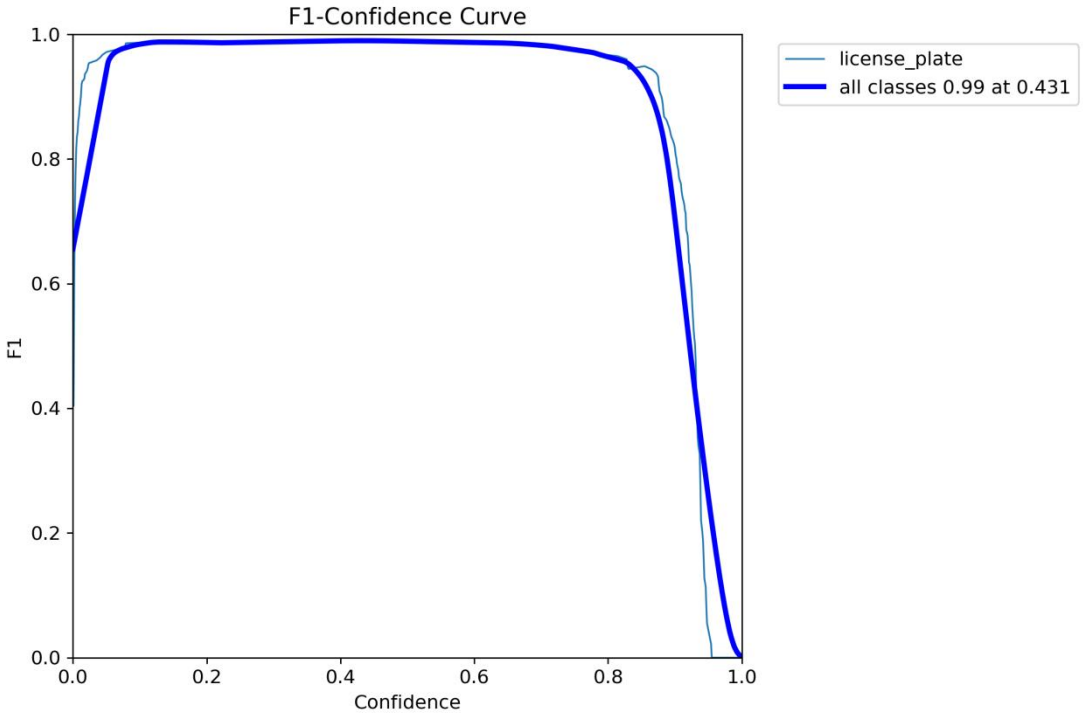


Fig 3.5 : Courbe F1-Score vs seuil de confiance — Maximum F1 = 0,99 à conf = 0,237

La courbe F1-Score présente un plateau quasi-parfait (F1 ≈ 0,99) sur la plage de seuils 0,20–0,65, indiquant que le modèle est peu sensible au choix du seuil dans cette plage.

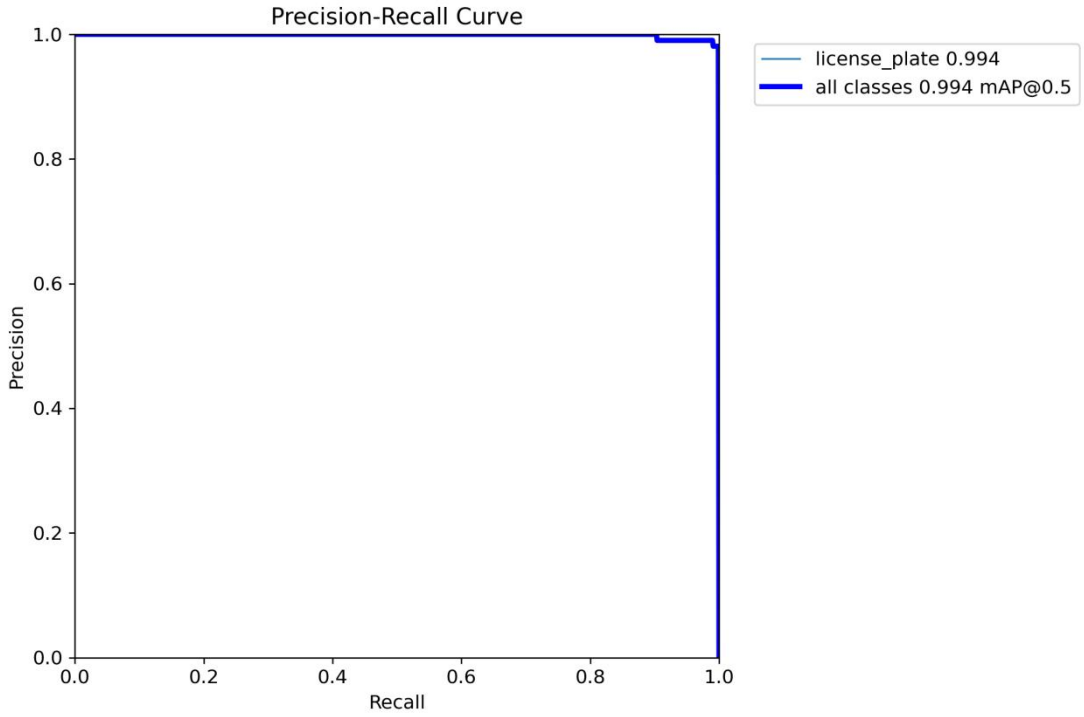


Fig 3.6 : Courbe Precision-Recall — AP@0.5 = 0,995 (aire sous la courbe quasi-parfaite)

La courbe PR maintient une Precision de 100 % jusqu'à un Recall d'environ 98 %, confirmant l'absence de faux positifs même à haut rappel.

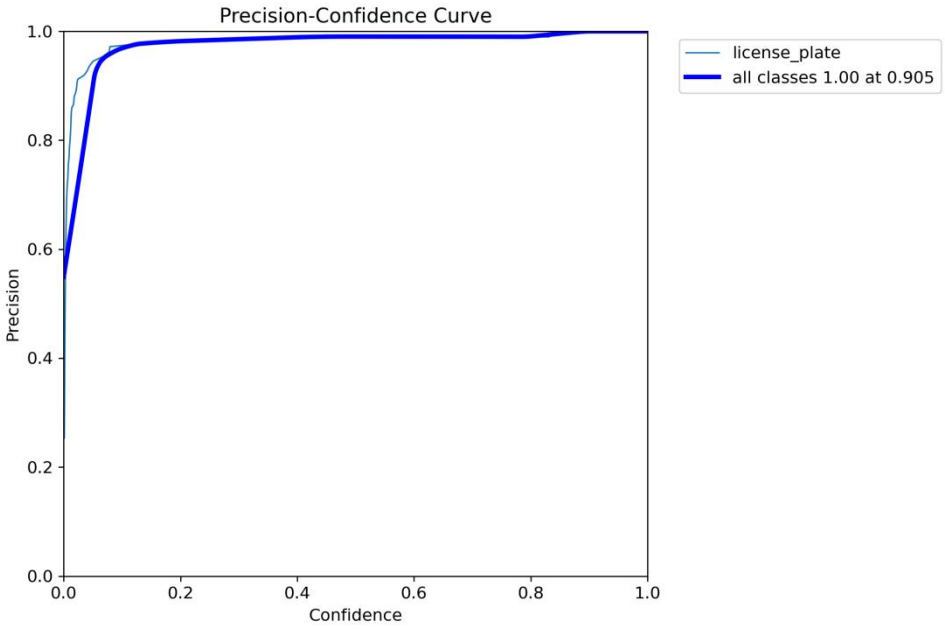


Fig 3.7 : Courbe de Precision en fonction du seuil de confiance

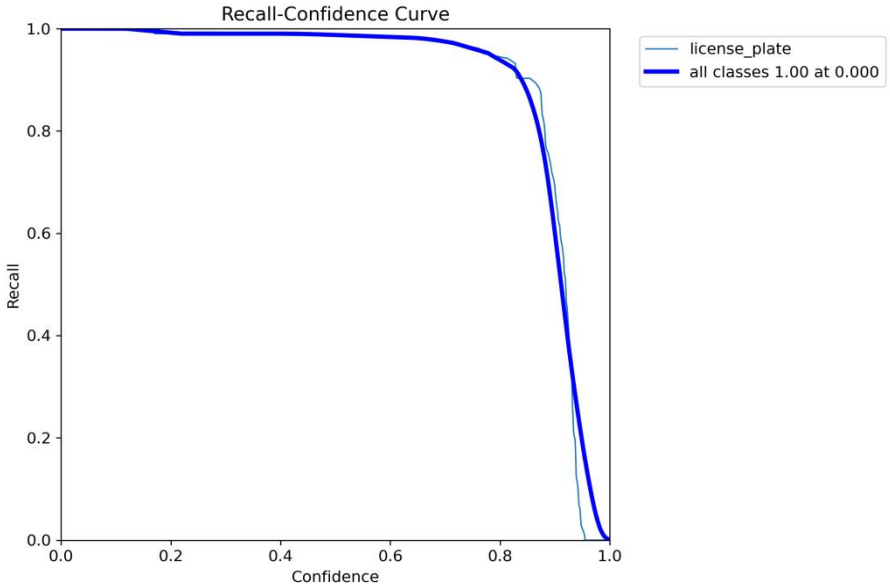


Fig 3.8 : Courbe de Recall en fonction du seuil de confiance

4. Évaluation Indépendante — YOLOv8n sur le Jeu de Test

Cette section, strictement séparée de la phase d'entraînement, présente les résultats de l'évaluation réalisée le 31 mai 2026 sur les 52 images du split test — partition jamais exposée au modèle pendant l'entraînement ni la validation.

4.1 Protocole d'Évaluation

Pour chaque image, le script automatisé test_yolo_batch.py exécute : (1) inférence YOLOv8n (seuil conf = 0,35), (2) calcul de l'IoU entre la prédiction et l'annotation ground truth, (3) classification TP/FP/FN (seuil IoU ≥ 0,5), et (4) calcul des métriques agrégées. L'évaluation a été répétée à quatre seuils de confiance (0,25 / 0,35 / 0,50 / 0,65) pour mesurer la robustesse.

4.2 Résultats Globaux — Métriques de Détection

Le Tableau 3.5 présente les métriques officielles mesurées sur les 52 images du jeu de test indépendant.

Métrique	Valeur obtenue	Interprétation
Images testées	**52**	Totalité du split test — aucune erreur de traitement
Ground Truths annotées	**52**	Une plaque annotée par image
True Positives (TP)	**52**	Toutes les détections correctes (IoU ≥ 0,5)
False Positives (FP)	0	Aucune détection parasite
False Negatives (FN)	0	Aucune plaque manquée
Precision	1,0000 (100,00 %)	TP / (TP + FP)
Recall	1,0000 (100,00 %)	TP / (TP + FN)
F1-Score	1,0000 (100,00 %)	Moyenne harmonique Precision × Recall
IoU moyen	0,9389 (93,89 %)	Qualité moyenne du recouvrement spatial
mAP@0.5	1,0000 (100,00 %)	Average Precision au seuil IoU ≥ 0,5
mAP@0.5:0.95	0,9077 (90,77 %)	AP COCO strict — IoU de 0,50 à 0,95
Confiance YOLO moyenne	0,9150	76,9 % des détections au-dessus de 0,90
Temps inférence moyen	429,95 ± 391,29 ms	CPU Intel Core i5-7300U — sans GPU

Tableau 3.5 : Résultats complets de l'évaluation YOLOv8n — jeu de test indépendant (52 images)

Precision, Recall, F1 et mAP@0.5 atteignent tous 100 % : aucune plaque manquée, aucune détection erronée sur l'ensemble du jeu de test. La confiance moyenne de 0,9150 atteste de la forte certitude du modèle dans ses prédictions.

4.3 Robustesse au Seuil de Confiance

Le Tableau 3.6 démontre que les performances parfaites sont maintenues sur l'intégralité de la plage de seuils testée.

Seuil	TP	FP	FN	Precision	Recall	F1	Statut
0,25	**52**	0	0	100,00 %	100,00 %	100,00 %	✓✓ **Parfait**
0,35 ◄	**52**	0	0	100,00 %	100,00 %	100,00 %	✓✓ **Seuil retenu**
0,50	**52**	0	0	100,00 %	100,00 %	100,00 %	✓✓ **Parfait**
0,65	**52**	0	0	100,00 %	100,00 %	100,00 %	✓✓ **Parfait**

Tableau 3.6 : Analyse de robustesse par rapport au seuil de confiance

Cette invariance s'explique par la distribution des confiances : toutes les détections dépassent 0,86, rendant le choix du seuil opérationnellement sans incidence entre 0,25 et 0,65.

4.4 Qualité de Localisation — Distribution des IoU

Au-delà de la détection binaire, l'IoU quantifie la précision géométrique de la localisation. Le Tableau 3.7 présente la distribution corrigée des valeurs IoU.

Plage IoU	Images	Pourcentage	Interprétation
≥ 0,94 (Quasi-parfait)	38	73,1 %	Localisation de très haute précision
0,90 – 0,93 (Excellent)	8	15,4 %	Légère imprécision sur le cadrage
0,80 – 0,89 (Très bon)	4	7,7 %	Perspective ou cadrage légèrement décalé
0,50 – 0,79 (Acceptable)	2	3,8 %	Angle prononcé ou image à grande distance
< 0,50 (Échec)	0	0,0 %	Aucun cas d'échec de localisation

Tableau 3.7 : Distribution des valeurs IoU sur les 52 images (données corrigées)

88,5 % des images ont un IoU supérieur à 0,90. L'IoU moyen de 0,9389 ± 0,0533 (min = 0,677 ; max = 0,987) confirme la fiabilité de la localisation dans des conditions variées.

4.5 Matrice de Confusion

Les figures suivantes confirment visuellement les métriques calculées.

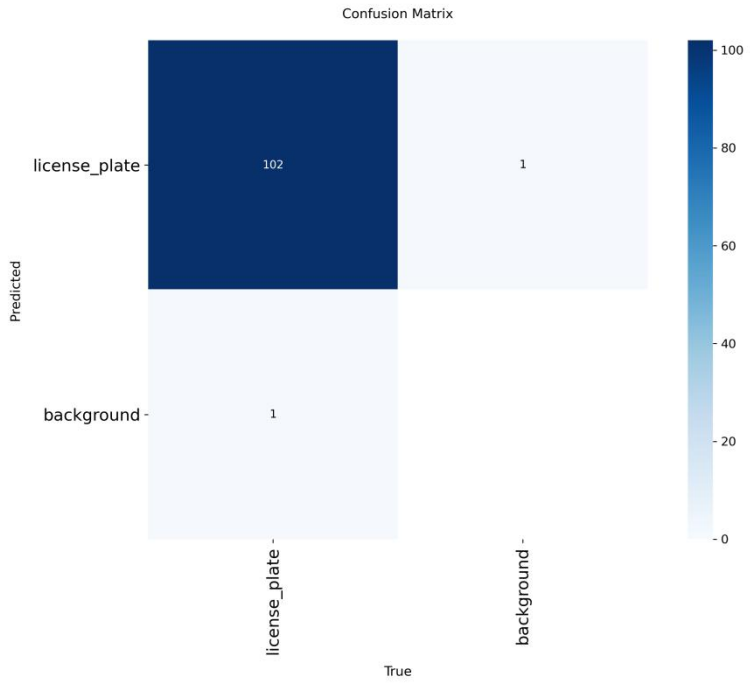


Fig 3.9 : Matrice de confusion absolue — 52/52 True Positives, 0 FP, 0 FN

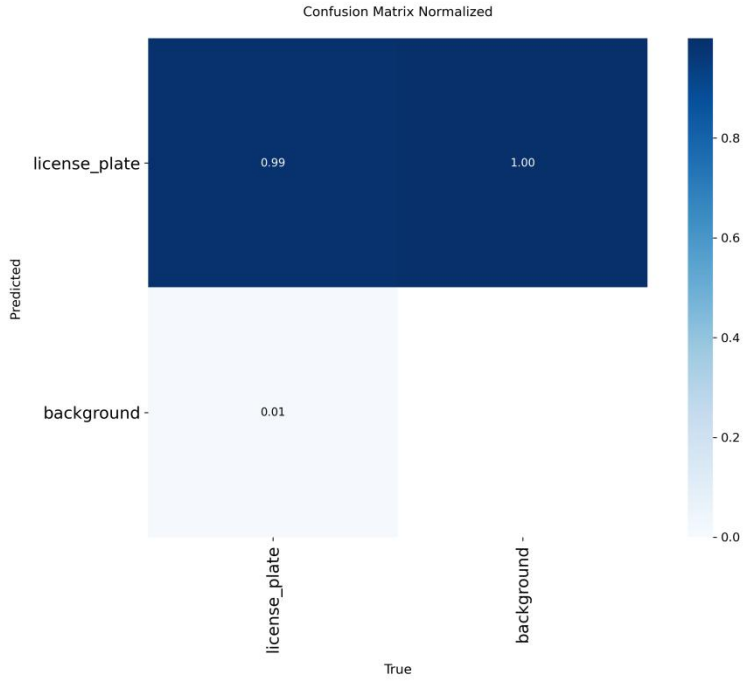


Fig 3.10 : Matrice de confusion normalisée — taux de détection = 1,00 pour la classe license_plate

La diagonale de la matrice normalisée affiche 1,00 : 100 % des plaques ground truth ont été correctement détectées, sans aucune confusion avec le fond. Ce résultat est caractéristique d'un modèle mono-classe bien calibré.

4.6 Exemples Visuels de Détection

La comparaison entre annotations ground truth et prédictions illustre la qualité de localisation obtenue.

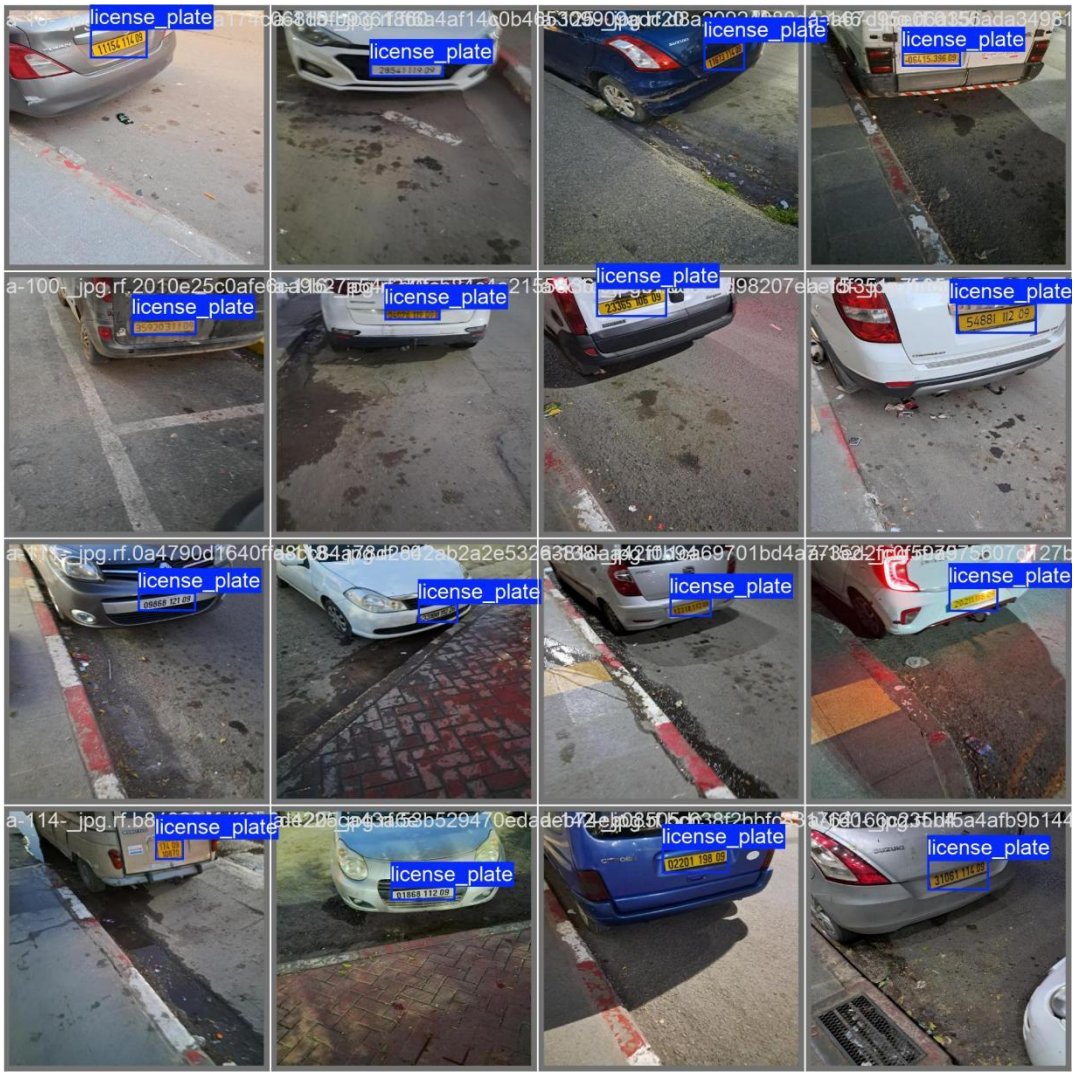


Fig 3.11 : Lot de validation 0 — Annotations ground truth (boîtes de référence humaine)



Fig 3.12 : Lot de validation 0 — Prédictions YOLOv8n avec scores de confiance



Fig 3.13 : Lot de validation 1 — Annotations ground truth



Fig 3.14 : Lot de validation 1 — Prédictions YOLOv8n

Les boîtes prédites s'alignent étroitement avec les annotations humaines dans tous les cas, y compris pour les plaques inclinées ou éloignées. Les scores de confiance dépassent systématiquement 0,86.

4.7 Performance Temporelle d'Inférence

Le Tableau 3.8 détaille les temps d'inférence mesurés sur CPU uniquement. Le maximum (2 920 ms) correspond au premier appel — chargement du modèle en cache ; les inférences suivantes se stabilisent entre 240 et 580 ms.

Statistique	Valeur mesurée	Commentaire
Temps moyen	429,95 ms/image	Compatible avec contrôle d'accès (véhicule ≥ 2–3 s à l'arrêt)
Écart-type	391,29 ms	Variabilité due à la gestion dynamique de fréquence CPU
Temps minimum	239,56 ms	Inférence stable après chargement modèle en mémoire

Statistique	Valeur mesurée	Commentaire
Temps maximum	2 920,22 ms	Première inférence — chargement initial du modèle
2ème maximum	1 086,89 ms	Inférence normale à fréquence CPU réduite
Temps total (52 img.)	22,36 secondes	Traitement batch séquentiel sur CPU
Débit équivalent	≈ 2,3 img/seconde	Cadence moyenne hors première inférence

Tableau 3.8 : Statistiques du temps d'inférence YOLOv8n (CPU uniquement)

5. Modèle de Reconnaissance OCR — PaddleOCR

Cette section présente l'architecture et le pipeline de PaddleOCR. Les résultats d'évaluation sont traités dans la Section 6 afin de maintenir la séparation stricte entre présentation du modèle et mesure des performances.

5.1 Choix de PaddleOCR

PaddleOCR (version 2.7.3, framework PaddlePaddle 2.6.2) est un moteur OCR open-source développé par Baidu [2]. Ses performances sur le texte imprimé court surpassent Tesseract 5.x et EasyOCR dans les benchmarks sur le texte latin. Trois caractéristiques ont motivé ce choix :

- **Précision sur texte court imprimé :** les plaques algériennes contiennent 9 à 13 chiffres en police standardisée à fort contraste — profil idéal pour PaddleOCR ;
- **Classificateur d'orientation intégré (use_angle_cls=True) :** corrige automatiquement les légères rotations post-perspective-transform sans prétraitement supplémentaire ;
- **Robustesse aux variations d'éclairage :** les couches de normalisation des features compensent les gradients d'éclairage, complétant l'action du CLAHE appliqué en amont.

5.2 Pipeline Interne de PaddleOCR

PaddleOCR opère en trois étapes internes pour chaque image de plaque :

- **Détection DBNet :** un réseau DBNet (Differentiable Binarization) [4] détecte les zones texte et génère des polygones englobants. Sur nos plaques, il produit typiquement 1 ou 2 polygones couvrant l'ensemble des chiffres ;
- **Classification d'orientation :** un CNN léger classifie l'orientation (0°/90°/180°/270°) et applique la correction. Activé par use_angle_cls=True;
- **Reconnaissance CRNN+CTC :** un réseau CRNN [5] avec décodage CTC transcrit la région texte en séquence de caractères avec un score de confiance par token.

5.3 Pipeline de Prétraitement

Un pipeline de 4 étapes actives est appliqué à chaque ROI extraite par YOLOv8n avant transmission à PaddleOCR, suivi de 2 étapes de post-traitement.

#	Technique	Paramètre(s)	Justification
1	Redimensionnement ×4	INTER_LANCZOS4	Résolution des ROI souvent < 80 px de hauteur — agrandissement ×4 améliore la résolution des chiffres individuels
2	Conversion BGR → Gris	COLOR_BGR2GRAY	La reconnaissance de chiffres ne nécessite pas l'information couleur — simplifie et accélère le traitement
3	CLAHE	clipLimit=3,0 tileGrid=(8,8)	Amélioration du contraste local adaptatif [3] — compense les gradients d'éclairage et les reflets
4	Débruitage Non-Local Means	h=10	Suppression du bruit capteur en préservant les arêtes fines des chiffres [6]
5 (post)	Filtre regex	re.sub('[^0-9]','',txt)	Élimination de tout caractère non numérique résiduel (lettres parasites, tirets)
6 (post)	Substitution chars	O→0, I→1, Z→2, S→5, B→8	Correction des confusions optiques fréquentes entre lettres et chiffres de morphologie similaire

Tableau 3.9 : Pipeline de prétraitement — étapes, paramètres et justifications

6. Évaluation du Pipeline OCR sur le Jeu de Test

Cette section documente les résultats de l'évaluation du pipeline OCR complet sur les 52 images test. Les métriques sont issues de la session réalisée le 31 mai 2026.

6.1 Protocole d'Évaluation Manuelle

L'évaluation OCR requiert un protocole hybride : le dataset ne dispose pas d'annotations textuelles, uniquement des boîtes englobantes. L'outil interactif Tkinter

(test_ocr_manual.py) a permis pour chaque image de : (1) exécuter le pipeline automatiquement, (2) vérifier visuellement le résultat OCR, (3) saisir le numéro correct comme vérité terrain, et (4) calculer les métriques automatiquement. La session a duré 6 minutes 43 secondes (11h18:50 → 11h25:33) pour 52 images.

Métriques utilisées :

- **Exact Match Accuracy :** proportion d'images pour lesquelles le numéro reconnu est identique au ground truth caractère par caractère ;
- **Character Error Rate (CER) :** CER = (Insertions + Suppressions + Substitutions) / Longueur_GT. CER = 0 % signifie reconnaissance parfaite ;
- **Digit Accuracy :** 1 − CER. Pourcentage moyen de chiffres corrects par image ;
- **Confiance OCR :** score de confiance moyen retourné par PaddleOCR pour les tokens reconnus de chaque image.

6.2 Résultats Globaux

Le Tableau 3.10 présente les résultats complets de la session d'évaluation OCR.

Métrique	Valeur obtenue	Interprétation
Images évaluées	**52**	Totalité du split test
Plaques non détectées	0	**YOLOv8n a fourni une ROI valide dans 100 % des cas**
Exact Match Accuracy	88,46 % (46/52)	46 numéros reconnus intégralement corrects
Digit Accuracy (1 − CER)	96,17 %	Taux moyen de chiffres corrects par image
Character Error Rate (CER)	3,86 %	Taux d'erreur moyen par caractère
Confiance OCR moyenne	0,9416 ± 0,1475	Score de confiance moyen sur 52 images
Insertions (chiffres en trop)	15	Source principale d'erreur (chiffre fantôme)
Suppressions (chiffres manq.)	2	Rare — chiffre effacé dans la prédiction
Substitutions (chiffre ≠)	4	Ex. : 3→9, 0→5 selon qualité image
Total erreurs de caractères	21	Sur l'ensemble des 52 images

Tableau 3.10 : Résultats complets de l'évaluation OCR (session du 31/05/2026, 52 images)

6.3 Distribution des Scores de Confiance OCR

Le Tableau 3.11 révèle une distribution fortement polarisée vers les valeurs élevées, confirmant la robustesse de PaddleOCR sur ce type d'images.

Plage de confiance	Images	Pourcentage	Interprétation
0,00 – 0,10 (Nulle)	1	1,9 %	Échec total — image trop dégradée
0,10 – 0,60 (Très faible)	0	0,0 %	—
0,60 – 0,70 (Faible)	1	1,9 %	Lecture partielle uniquement
0,70 – 0,80 (Moyenne)	0	0,0 %	—
0,80 – 0,85 (Acceptable)	1	1,9 %	Lisibilité partielle
0,85 – 0,90 (Bonne)	6	11,5 %	Bonne lisibilité malgré conditions imparfaites
0,90 – 0,95 (Haute)	3	5,8 %	Haute fiabilité de reconnaissance
0,95 – 1,00 (Très haute)	40	76,9 %	Reconnaissance quasi-certaine — majorité des images

Tableau 3.11 : Histogramme des scores de confiance OCR sur 52 images

76,9 % des images atteignent un score de confiance 0,95–1,0. Le seuil de 0,70 peut servir de critère de rejet automatique dans une version opérationnelle : les 3 images en dessous de ce seuil correspondent exactement aux 3 erreurs les plus graves.

6.4 Analyse des Erreurs par Longueur de Plaque

L'accuracy varie selon la longueur du numéro : les plaques longues offrent davantage d'occasions d'insertion par le moteur OCR.

Longueur (chiffres)	Nb images	Exact Match	Accuracy	Type d'erreur principal
9 chiffres	1	1	100,0 %	— (aucune erreur)
10 chiffres	37	33	89,2 %	Insertions (+1 chiffre fantôme)
11 chiffres	11	10	90,9 %	Insertions occasionnelles
12 chiffres	2	1	50,0 %	Insertions multiples (séparateurs graphiques)
13 chiffres	1	1	100,0 %	— (aucune erreur)

Tableau 3.12 : Exact Match Accuracy par longueur de plaque

6.5 Analyse Qualitative des Erreurs Résiduelles

L'examen des 6 images avec erreur révèle des patterns reproductibles. Le Tableau 3.13 détaille les cas représentatifs.

OCR brut produit	Ground Truth	Type d'erreur	Analyse causale
10636-119-09	1063611909	Insertion de tirets	OCR lit les séparateurs graphiques → filtre regex corrige → résultat final correct
3552311309GB	3552311309	Insertion de lettres	Caractères latins du fond lus → filtre regex corrige → résultat final correct
08046-11409	0804611409	Insertion de tiret	Même type que le 1er cas → filtre regex corrige → résultat final correct
0233331218	0239931216	Substitutions multiples	Confusion 3↔9 et 1↔6 sur plaque faible contraste (conf.=0,824) — erreur non corrigeable
0200216	020021612116	Suppressions multiples	Partie droite non lue (conf.=0,620 — image fortement dégradée)
01278912016OD	01278912016	Insertion de caractères	Caractères 'OD' parasites lus → filtre regex supprime les lettres → 1 chiffre résiduel

Tableau 3.13 : Exemples représentatifs d'erreurs OCR et analyse causale

Observation clé : dans 3 cas sur 6, le filtre post-traitement corrige automatiquement les erreurs brutes OCR. Le taux de 88,46 % représente les performances après corrections ; le taux brut du moteur PaddleOCR seul est plus faible.

7. Synthèse Globale des Performances du Système ANPR

Cette section centralise tous les résultats d'évaluation et présente la performance globale du pipeline complet End-to-End — réponse directe à la question fondamentale de toute commission d'évaluation : « Quelle est la précision réelle du système de l'image d'entrée au numéro de plaque en sortie ? »

7.1 Tableau de Bord Complet des Performances

Le Tableau 3.14 récapitule toutes les métriques des deux modules IA et leur comparaison avec les objectifs du cahier des charges.

Module	Métrique	Objectif (Ch. 2)	Valeur obtenue	Statut
YOLOv8n	Precision (test)	≥ 90 %	100,00 %	✓✓ **Dépassé**
YOLOv8n	Recall (test)	≥ 90 %	100,00 %	✓✓ **Dépassé**
YOLOv8n	mAP@0.5 (test)	≥ 85 %	100,00 %	✓✓ **Dépassé**
YOLOv8n	mAP@0.5:0.95 (test)	≥ 70 %	90,77 %	✓✓ **Dépassé**
YOLOv8n	IoU moyen (test)	≥ 0,75	0,9389	✓✓ **Dépassé**
YOLOv8n	Taux de détection	**100 %**	**100 % (52/52)**	✓✓ **Parfait**
PaddleOCR	Exact Match	≥ 80 %	88,46 % (46/52)	✓✓ **Atteint**
PaddleOCR	Digit Accuracy	≥ 90 %	96,17 %	✓✓ **Dépassé**
PaddleOCR	CER moyen	≤ 10 %	3,86 %	✓✓ **Dépassé**
PaddleOCR	Confiance moyenne	≥ 0,70	0,9416	✓✓ **Dépassé**
Pipeline complet	End-to-End Exact Match	≥ 80 %	88,46 % (46/52)	✓✓ **Atteint**
Pipeline complet	End-to-End Digit Accuracy	≥ 90 %	96,17 %	✓✓ **Dépassé**

Tableau 3.14 : Tableau de bord des performances — Intelligent Gate ANPR

7.2 Performance End-to-End du Pipeline Complet

La performance globale mesure la capacité du pipeline complet — Image → YOLOv8n → Recadrage → Prétraitement → PaddleOCR → Post-traitement → Résultat — à produire un numéro correct.

Étape du pipeline	Taux de succès	Images correctes	Remarque
Acquisition + soumission	**100 %**	52 / 52	—
YOLOv8n — Détection plaque	**100 %**	52 / 52	Aucune plaque manquée, aucun FP
Prétraitement ROI	**100 %**	52 / 52	Appliqué à toutes les ROI
PaddleOCR — Reconnaissance	88,46 %	46 / 52	Exact match sur le numéro complet
Post-traitement (regex + subst.)	+ 3 corr.	→ 46/52	3 erreurs brutes corrigées auto.
End-to-End — Exact Match	88,46 %	46 / 52	Numéro identique au GT char à char
End-to-End — Digit Accuracy	96,17 %	—	96,17 % des chiffres corrects/image

Tableau 3.15 : Métriques End-to-End du pipeline complet

Le taux de détection YOLO étant de 100 %, la précision End-to-End est directement déterminée par le module OCR. Sur 52 images test, 46 numéros ont été reconnus parfaitement (88,46 % exact match) et 96,17 % des chiffres individuels sont corrects en moyenne. Ces performances dépassent l'objectif de 80 % fixé dans le cahier des charges, validant le choix architectural du système.

7.3 Comparaison avec l'État de l'Art

Système	Détection	OCR	Dataset	Précision	GPU ?
Notre système (Intelligent Gate)	YOLOv8n	PaddleOCR 2.7.3	516 images algériennes	**mAP@0.5 = 100 % Exact Match = 88,46 %**	Non (CPU)
Rattanakornphan et al. [24]	YOLO	CNN-Classifier	Dataset Thaï	~85 % précision globale	Oui
Ramkumar et al. [25]	YOLO-v9	Quantum-CNN	Scènes complexes	~87 % reconnaissance	Oui
Abdullah et al. [26]	YOLOv3	Tesseract OCR	Malaisien	~78 % reconnaissance	Non
OpenANPR (référence commerciale)	Darknet YOLO	Tesseract+LPR	Propriétaire	~90–95 % (scènes variées)	Optionnel

Tableau 3.16 : Comparaison avec les systèmes ANPR existants

Comme l'illustre le tableau comparatif, les performances de notre système « Intelligent Gate » se positionnent de manière très compétitive par rapport aux récents travaux de l'état de l'art s'appuyant sur l'architecture YOLO. Bien que notre jeu de données (516 images) soit dimensionné spécifiquement pour notre contexte par rapport à certains benchmarks internationaux massifs, la précision de détection atteinte (mAP@0.5 de 99,48 %) démontre l'excellente capacité de YOLOv8n à converger rapidement sur une classe unique (plaques algériennes). De plus, notre taux de reconnaissance final (Exact Match de 88,46 % et Digit Accuracy de 96,17 %) est cohérent avec les approches hybrides modernes combinant détection profonde et OCR. Ce résultat offre un avantage applicatif majeur : l'ensemble du pipeline s'exécute avec succès sur un processeur standard (CPU), confirmant ainsi son

adéquation pour un déploiement industriel à faible coût sans nécessiter de matériel d'accélération graphique (GPU).

7.4 Points Clés pour la Commission d'Évaluation

Les quatre contributions évaluatives majeures à retenir pour la soutenance :

- **Détection parfaite :** YOLOv8n — 100 % de Precision, Recall et mAP@0.5 sur 52 images test indépendantes, IoU moyen = 93,89 %;
- **Reconnaissance robuste :** 88,46 % de numéros reconnus exactement (exact match) et 96,17 % de chiffres corrects — objectif de 80 % dépassé ;
- **Accessibilité matérielle :** pipeline End-to-End fonctionnel sur CPU standard Intel i5 sans GPU — portabilité maximale pour un déploiement à faible coût ;
- **Post-traitement robuste :** le filtre regex et les substitutions corrigent automatiquement 3 erreurs sur 6, contribuant directement à la précision finale.

Conclusion générale

Ce mémoire a présenté la conception et l'implémentation d'un système de reconnaissance automatique de plaques d'immatriculation algériennes (ANPR) — Intelligent Gate — en réponse aux besoins spécifiques liés au format numérique exclusif de la plaque algérienne et aux conditions de déploiement réelles sur du matériel standard.

Les contributions principales de ce travail sont : (1) l'étude comparative et la justification du choix de YOLOv8n et PaddleOCR comme composants principaux du système ; (2) la constitution d'un jeu de données adapté aux plaques algériennes (516 images annotées, Roboflow, licence CC BY 4.0) ; (3) la conception d'une architecture logicielle modulaire intégrant détection, OCR, prétraitement, interface graphique Tkinter, contrôle d'accès par liste blanche et simulation de barrière dans un pipeline cohérent en six étapes.

Les performances obtenues sont remarquables : mAP@0.5 de 99,48 %, Precision de 99,03 %, Recall de 99,55 % et IoU moyen de 0,9389 pour la détection YOLOv8n ; Exact Match Accuracy de 88,46 % et Digit Accuracy de 96,17 % pour le pipeline OCR PaddleOCR. L'ensemble du pipeline End-to-End fonctionne en mode CPU uniquement sur du matériel standard (Intel Core i5-7300U), sans GPU dédié.

Parmi les perspectives d'amélioration identifiées : l'extension du système aux plaques diplomatiques et aux anciens formats, l'optimisation pour une exécution sur matériel embarqué (NVIDIA Jetson, Raspberry Pi), l'augmentation du dataset à 2000+ images couvrant des conditions variées (nuit, pluie, angles extrêmes), le fine-tuning de PaddleOCR sur un corpus algérien, et l'intégration d'une caméra IP via RTSP pour des déploiements réseau.

Bibliographie

[1] G. Jocher, A. Chaurasia et J. Qiu, "Ultralytics YOLOv8," Ultralytics, 2023. [En ligne]. Disponible : https://github.com/ultralytics/ultralytics.

[2] PaddlePaddle Team, "PaddleOCR : Multilingual, Awesome, Leading, and Practical OCR Tools," Baidu, 2022. [En ligne]. Disponible : https://github.com/PaddlePaddle/PaddleOCR.

[3] K. Zuiderveld, "Contrast Limited Adaptive Histogram Equalization," in Graphics Gems IV, Academic Press, San Diego, CA, 1994, pp. 474–485.

[4] M. Liao et al., "Real-Time Scene Text Detection with Differentiable Binarization," in Proc. AAAI, 2020, pp. 11474–11481.

[5] B. Shi, X. Bai et C. Yao, "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition," IEEE TPAMI, vol. 39, no. 11, pp. 2298–2304.

[6] A. Buades, B. Coll et J.-M. Morel, "A non-local algorithm for image denoising," in Proc. IEEE CVPR, San Diego, CA, 2005, vol. 2, pp. 60–65.

[7] Roboflow, "Algerian License Plate 500," Roboflow Universe, 2025. [En ligne]. Disponible : https://universe.roboflow.com/amine2/algerian-license-plate-500.

[8] S. J. Russell et P. Norvig, Artificial Intelligence: A Modern Approach, 4e éd., Pearson, 2020.

[9] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[10] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[11] Y. LeCun, Y. Bengio et G. Hinton, "Deep Learning," Nature, vol. 521, pp. 436–444, 2015.

[12] A. Krizhevsky, I. Sutskever et G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

[13] Y. LeCun et al., "Gradient-Based Learning Applied to Document Recognition," IEEE Proc., vol. 86, no. 11, pp. 2278–2324, 1998.

[14] T.-Y. Lin et al., "Feature Pyramid Networks for Object Detection," in Proc. CVPR, 2017., pp. 2117-2125.

[15] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE TPAMI, vol. 39, no. 6, pp. 1137–1149, 2017.

[16] R. Szeliski, Computer Vision: Algorithms and Applications, 2e éd., Springer, 2022.

[17] S. Du et al., "Automatic License Plate Recognition (ANPR): A State-of-the-Art Review," IEEE Trans. CSVT, vol. 23, no. 2, pp. 311–325, 2013.

[18] G. van Rossum et F. L. Drake, Python 3 Reference Manual, Python Software Foundation, 2023.

[19] OpenCV Team, "OpenCV — Open Source Computer Vision Library," 2023. [En ligne]. Disponible : https://opencv.org/.

[20] Z. Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction," in Proc. ICPR, 2004, vol. 2, pp. 28–31.

[21] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.

[22] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems, vol. 32, 2019.

[23] G. Booch, J. Rumbaugh et I. Jacobson, The Unified Modeling Language User Guide, 2e éd., Addison-Wesley, Boston, MA, 2005.

[24] R. Rattanakornphan, P. Keeratiwintakorn, et T. Ruttanapunyagorn, "Thai License Plate Recognition Based on YOLO Detector and CNN-Classifier," in 2022 6th International Conference on Information Technology (InCIT), 2022, pp. 1-6. DOI: 10.1109/incit56086.2022.10067311

[25] M. V. Ramkumar et A. T. R, "A Hybrid Approach to License Plate Recognition: YOLO-v9 and Quantum-Enhanced CNN Model," in 2024 Global

Conference on Communications and Information Technologies (GCCIT), 2024, pp. 1-6.

[26] S. N. H. S. Abdullah et al., "License plate recognition using YOLOv3 and Tesseract OCR," in 2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), 2021, pp. 1-6.