

## Projet de Fin d'Études

*Pour l'obtention du diplôme d'Ingénieur d'État*

Option : Génie Logiciel

---

# Système de Reconnaissance Automatique des Plaques Minéralogiques Marocaines : Implémentation sous Android et Parking Intelligent

---

*Réalisé par :*

KAMGA DJEMGOU Hisdaele Kavel

*Effectué à :*



*Sous l'encadrement :*

*Académique de :*

Pr. METRANE Abdelmoutalib  
Pr. KHALFI Hamza

*Professionnel de :*

M. GHOUAMI Marouane

Année Académique : 2020/2021

# Table des matières

<b>Table des figures</b>	iv
<b>Liste des tableaux</b>	vi
<b>Acronymes</b>	vii
<b>Introduction Générale</b>	1
<b>1 Contexte général du projet</b>	3
1.1 Introduction . . . . .	3
1.2 Présentation de l'entreprise . . . . .	3
1.3 Présentation du projet . . . . .	4
1.3.1 Caractéristiques des plaques marocaines . . . . .	4
1.3.2 Problématique et Objectifs du projet . . . . .	5
1.4 Conduite de projet . . . . .	5
1.4.1 Méthodologie de travail . . . . .	5
1.4.2 Outils d'organisation et de communication . . . . .	8
1.5 Conclusion . . . . .	9
<b>2 Etude des systèmes ANPR</b>	10
2.1 Introduction . . . . .	10
2.2 Architecture et Composants . . . . .	10
2.3 Systèmes existants . . . . .	12
2.4 Domaines d'application et Difficultés . . . . .	13
2.5 Conclusion . . . . .	14
<b>3 Traitement d'images</b>	15
3.1 Introduction . . . . .	15
3.2 Généralités sur les images numériques . . . . .	15
3.2.1 Qu'est-ce qu'une image numérique ? . . . . .	15
3.2.2 Représentation de l'image . . . . .	15
3.2.3 Caractéristiques d'une image numérique . . . . .	16
3.2.4 Formats des images numériques . . . . .	17
3.3 Opérateurs de traitement d'images . . . . .	18
3.3.1 Opérations morpho-mathématiques . . . . .	18
3.3.2 Détection des contours . . . . .	19
3.4 Conclusion . . . . .	20

<b>4 Reconnaissance Optique de Caractères</b>	<b>21</b>
4.1 Introduction . . . . .	21
4.2 Définition . . . . .	21
4.3 Phases d'OCR . . . . .	21
4.3.1 Prétraitement . . . . .	22
4.3.2 Segmentation . . . . .	22
4.3.3 Reconnaissance de caractères . . . . .	23
4.3.4 Post-traitement . . . . .	23
4.4 Outils . . . . .	23
4.5 Conclusion . . . . .	24
<b>5 Apprentissage Automatique</b>	<b>25</b>
5.1 Introduction . . . . .	25
5.2 Phases d'un cycle de Machine Learning . . . . .	25
5.3 Méthodes d'apprentissage automatique . . . . .	26
5.3.1 Apprentissage supervisé . . . . .	26
5.3.2 Apprentissage non-supervisé . . . . .	26
5.3.3 Apprentissage par renforcement . . . . .	26
5.4 Deep Learning . . . . .	27
5.4.1 Définition . . . . .	27
5.4.2 Réseaux de neurones artificiels . . . . .	27
5.4.3 Apprentissage d'un réseau de neurones . . . . .	29
5.4.4 Réseaux de neurones convolutifs . . . . .	30
5.4.5 Détection des objets . . . . .	32
5.4.5.1 Métriques d'évaluation . . . . .	32
5.4.5.2 Les algorithmes de détection d'objets . . . . .	33
5.5 Conclusion . . . . .	37
<b>6 Conception et mise en oeuvre</b>	<b>39</b>
6.1 Introduction . . . . .	39
6.2 Architecture . . . . .	39
6.3 Détection de la plaque . . . . .	40
6.3.1 Acquisition des données . . . . .	40
6.3.2 Nettoyage et préparation . . . . .	41
6.3.3 Entraînement du modèle . . . . .	41
6.3.4 Résultats et évaluation . . . . .	43
6.4 Lecture du numéro d'immatriculation . . . . .	44
6.4.1 Acquisition des données . . . . .	44
6.4.2 Nettoyage et préparation . . . . .	45
6.4.3 Entraînement du modèle . . . . .	45
6.4.4 Résultats et évaluation . . . . .	46
6.5 Conclusion . . . . .	47
<b>7 Applications</b>	<b>48</b>
7.1 Introduction . . . . .	48
7.2 Développement de l'application mobile . . . . .	48
7.2.1 Analyse et spécifications des besoins . . . . .	48

7.2.2	Conception	49
7.2.3	Réalisation	50
7.3	Parking intelligent	55
7.3.1	Architecture du système GoPark	55
7.3.2	Environnements matériel et logiciel	56
7.3.3	Résultats	58
7.4	Conclusion	59

# Table des figures

1.1	Modèle sur une ligne pour les véhicules automobiles . . . . .	4
1.2	Le cadre de travail SCRUM . . . . .	7
1.3	Logos des outils d'organisation et de planification . . . . .	9
2.1	Architecture de fonctionnement d'un système ANPR . . . . .	12
3.1	Différence entre image matricielle et image vectorielle . . . . .	16
3.2	Exemples de types d'images matricielles . . . . .	17
3.3	Dilatation . . . . .	18
3.4	Erosion . . . . .	18
3.5	Ouverture . . . . .	19
3.6	Fermeture . . . . .	19
3.7	Détection des contours avec différents filtres . . . . .	20
4.1	Processus d'OCR . . . . .	21
5.1	Modélisation d'un neurone artificiel . . . . .	28
5.2	Réseau de neurones artificiels . . . . .	29
5.3	Architecture d'un réseau de neurones convolutifs . . . . .	32
5.4	Différence entre classification et détection d'objets . . . . .	32
5.5	Equation de l'IoU source : pyimagesearch . . . . .	33
5.6	R-CNN . . . . .	34
5.7	Fast R-CNN . . . . .	34
5.9	Architecture du SSD . . . . .	35
5.8	Faster R-CNN . . . . .	35
5.10	Architecture de la première de version de YOLO . . . . .	36
5.11	NMS . . . . .	36
6.1	Architecture du système MoPlaZer . . . . .	40
6.2	Exemple d'images collectées sur Open Images . . . . .	41
6.3	Évolution de la métrique mAP lors de l'entraînement du modèle de détection de plaques. . . . .	42
6.4	Exemples de détection de plaques marocaines . . . . .	43
6.5	Exemple d'annotation de plaque sur LabelImg . . . . .	45
6.6	Proportions des classes . . . . .	45
6.7	Évolution de la performance du modèle . . . . .	46
6.8	Exemples de lecture des matricules des véhicules . . . . .	47
7.1	Diagramme de cas d'utilisation . . . . .	49

7.2	Diagramme de séquence pour le cas d'utilisation "Capturer une image" . . . . .	51
7.3	Architecture détaillée de notre application . . . . .	52
7.4	Diagramme d'activité pour la reconnaissance des plaques . . . . .	53
7.5	Quelques outils logiciels pour le développement de l'application mobile . . . . .	54
7.6	Exemples de quelques interfaces de l'application mobile . . . . .	55
7.7	Architecture du système GoPark . . . . .	56
7.8	Outils matériels pour la réalisation du parking intelligent . . . . .	57
7.9	Outils logiciels pour la réalisation du parking intelligent . . . . .	58
7.10	Fonctionnement général de GoPark . . . . .	59
7.11	Maquette du parking intelligent GoPark . . . . .	60

# Liste des tableaux

1.1	Scrum Team du projet . . . . .	8
5.1	Comparaison entre les différentes versions de YOLO . . . . .	37
6.1	Analyse de la collection des données de véhicules . . . . .	41
6.2	Quelques paramètres du fichier de configuration YOLO . . . . .	42
6.3	Evaluation du modèle de détection des plaques marocaines . . . . .	43
6.4	Caractéristiques du PC . . . . .	44
6.5	Quelques paramètres du fichier de configuration YOLO pour le modèle OCR . . . . .	46
6.6	Evaluation du modèle de lecture du matricule . . . . .	47
7.1	Spécifications de la carte Raspberry Pi 4 . . . . .	57

# Acronymes

- ALPR** Automatic License Plate Recognition. [10](#)
- ANN** Artificial Neural Network. [30](#)
- ANPR** Automatic Number Plate Recognition. [1, 5, 9](#)
- AP** Average Precision. [33](#)
- API** Application Programming Interface. [23](#)
- AUP** Agile Unified Process. [6](#)
- CNN** Convolutional Neural Network. [30](#)
- CPU** Central Processing Unit. [27](#)
- DL** Deep Learning. [27](#)
- DSDM** Crystal Dynamic Systems Development Method. [6](#)
- FDD** Feature Driven Development. [6](#)
- FPS** Frame Per Second. [33](#)
- GIF** Graphics Interchange Format. [17](#)
- GPU** Graphic Processing Unit. [27](#)
- IA** Intelligence Artificielle. [1](#)
- IoU** Intersection over Union. [33](#)
- JPG/JPEG** Joint Photographic Experts Group. [17](#)
- KNN** K-Nearest Neighbors. [26](#)
- LAPI** Lecture Automatisée de Plaques d’Immatriculation. [1, 10](#)
- mAP** mean Average Precision. [33](#)
- ML** Machine Learning. [27](#)
- MoPlaZer** Moroccan Plate Recognizer. [5](#)
- OCR** Optical Character Recognition. [1, 11, 12, 21](#)
- PNG** Portable Network Graphics. [17](#)
- R-CNN** Region-based Convolutional Neural Networks. [33, 34](#)

**RAPI** Reconnaissance Automatique de Plaques d’Immatriculation. [12](#)

**RFID** Radio-frequency identification. [5](#)

**RNN** Recurrent Neural Network. [30](#)

**RPN** Region Proposal Network. [35](#)

**SSD** Single Shot MultiBox Detector. [33](#), [35](#)

**SVG** Scalable Vector Graphics. [17](#)

**SVM** Support Vector Machines. [26](#)

**TIFF** Tagged Image File Format. [17](#)

**XP** eXtreme Programming. [6](#)

**YOLO** You Only Look Once. [33](#), [35](#)

# Introduction Générale

Selon une publication parue le 28 février 2020 au quotidien d'informations Aujourd'hui Le Maroc, le parc automobile marocain comptait en 2018 environ 4,3 millions de véhicules en circulation. Ceci correspond à une augmentation de plus de 50% par rapport à 2002 où on dénombrait environ 1,81 million de véhicules. Cette croissance rapide et continue n'est pas sans conséquence sur la société. On peut citer notamment la montée des fléaux tels que les vols de voitures, la violation du trafic routier, les collisions, les congestions et sans compter leurs impacts sur l'économie nationale. Face à cette situation, la mise en place des systèmes automatiques et performants de gestion du trafic routier devient cruciale. Un des éléments de haute importance qui entre dans ces systèmes est l'identification des véhicules. Et quoi de plus simple et efficace pour identifier les véhicules que les plaques d'immatriculation. Encore appelée plaque minéralogique, une plaque d'immatriculation est un objet généralement en forme rectangulaire placé sur un véhicule. Sur elle, est inscrite une combinaison unique de chiffres et de lettres qui contient des informations sur un véhicule et par ricochet sur son propriétaire.

Ces dernières années, les avancées technologiques dans le domaine de l'[IA](#) et l'émergence des citées dites intelligentes ont donné un regain d'intérêt aux chercheurs et entreprises internationaux en général et marocains en particulier sur la question de l'identification des véhicules via leur plaque. Dans le métier, on parle le plus souvent d'un dispositif de [Lecture Automatisée de Plaques d'Immatriculation \(LAPI\)](#) ou encore en anglais [Automatic Number Plate Recognition \(ANPR\)](#). C'est une technologie d'identification qui utilise la plupart du temps des techniques de traitements d'images , de vision par ordinateur (Computer Vision en anglais) et de reconnaissance optique de caractères ou [Optical Character Recognition \(OCR\)](#) pour lire les plaques d'immatriculation de véhicules. Toutefois, avec l'essor de l'[IA](#) qui fait déjà largement ses preuves dans plusieurs domaines (la médecine, l'industrie, l'aviation), des techniques modernes et plus performantes utilisant les algorithmes puissants d'apprentissage automatique (Machine Learning) se sont ajoutées au processus d'[ANPR](#).

Dans ce sens, il existe déjà sur le marché plusieurs solutions d'[ANPR](#) qui sont soit intégrées dans des caméras adaptées soit "consommables" via les services web. Si d'une part, la plupart de ces solutions sont payantes, d'autre part elles ne traitent généralement pas le cas des modèles de plaques marocaines qui contiennent des caractères en arabe. C'est dans ce cadre que s'inscrit notre projet de fin d'études : mettre en place une solution de reconnaissance automatique de plaques minéralogiques marocaines en intégrant les algorithmes de Machine Learning.

Le présent rapport qui fait étalage des travaux que nous avons réalisés est composé de quatre(4) chapitres. Le premier chapitre sera consacré à la présentation de l'organisme d'accueil, du projet de manière générale ainsi que de la méthodologie de travail suivie. Un panorama synthétique et organisé des travaux déjà réalisés sur les systèmes d'[ANPR](#) sous forme d'état de l'art sera fait dans le deuxième chapitre. Le troisième chapitre nous permettra de faire étalage en profondeur de l'approche que nous avons adoptée. La mise en œuvre ou encore l'implémentation de notre

solution sera exposée au dernier chapitre. Enfin en guise de conclusion générale, nous donnerons une synthèse des travaux suivie des perspectives.

# Chapitre 1

## Contexte général du projet

### 1.1 Introduction

Comme en littérature pour comprendre un mot, il faut recourir à son contexte, de même comprendre un projet nécessite la compréhension de son contexte. Une bonne connaissance de ce dernier permet d'avoir une vision globale de la problématique traitée. C'est donc l'objectif de ce chapitre. Pour l'atteindre, nous allons d'abord présenter l'entreprise où nous avons effectué notre stage. Ensuite, nous ferons une présentation générale du sujet. Enfin, nous verrons la méthodologie de travail adoptée.

### 1.2 Présentation de l'entreprise

Notre projet de fin d'études a été réalisé au sein d'une jeune et dynamique entreprise marocaine dont le siège se trouve dans la ville de Casablanca. Il s'agit de l'organisme **KF2Y Consulting**. Fondée en 2010, KF2Y met à la disposition de ses clients un ensemble de compétences et d'experts, pour le déploiement, la maîtrise et l'optimisation des systèmes d'information. L'entreprise propose aux autres entreprises de l'ensemble des secteurs économiques une approche nouvelle qui conjugue l'utilisation de méthodes novatrices et de bonnes pratiques, le recours à la technologie et l'expertise de son capital humain. Pour poursuivre son développement, KF2Y a misé sur 2 pôles :

- **L'activité Consulting** : Ce pôle est né du rapprochement de professionnels du conseil en management et systèmes d'information avec une expertise technologique qui consiste à aider ses clients, qu'ils soient entreprises privées, administrations publiques ou organisations non gouvernementales, à créer de la valeur via la construction et l'implémentation de solutions technologiques avec une ambition de construire des relations pérennes avec ses clients-partenaires afin de mieux les connaître, mieux anticiper leurs besoins et mieux les servir. Cela se manifeste par la création de valeur ajoutée chez ses clients au niveau des ressources internes ou par une activité de sourcing qui consiste à intervenir dès la phase de recrutement de la ressource pour répondre à un besoin de mission chez nos clients.
- **L'activité Recherche et Développement** : Elle vise à mobiliser des ressources internes dans une optique de création des solutions innovantes cherchant à répondre à un paramètre clé de cette mutation économique et digitale vu par le monde.

KF2Y Consulting couvre un champ d'applications très large et varié tel que :

- **Développement IT** : L'entreprise développe pour ses partenaires des applications mobiles et web en utilisant les technologies comme **Java EE, Angular, Angular JS, .Net, C#**,

**PL/SQL, SQL, ABAP/4, PHP, Python ;**

- **ERP / Intégration de progiciels**
- **Qualité logiciel et Testing**
- **Big Data et Machine Learning**
- **Conseil et formation**

Dans chacun de ces domaines d'action, on retrouve des groupes de personnes qui travaillent sur des projets innovants. En ce qui concerne notre projet, nous avons collaboré avec les membres de l'équipe Big Data et Machine Learning.

La branche Big Data et Machine Learning est une nouvelle branche au sein de l'entreprise. Elle a été créée pour répondre aux besoins sans cesse croissants des systèmes intelligents. L'équipe attachée à cette branche est composée d'un manager qui suit l'avancé des projets, d'un responsable qui conduit techniquement les projets et de certains développeurs qui réalisent des applications intégrant des algorithmes de Big Data et de Machine Learning.

## 1.3 Présentation du projet

Ces dernières années au Maroc, le nombre des véhicules en circulation ne cesse de croître rapidement. Ceci provoque généralement des violations et de l'anarchie dans le trafic routier. La reconnaissance automatique des plaques d'immatriculation devient donc une urgence. Certes il existe déjà sur le marché des outils qui permettent de lire automatiquement les plaques d'immatriculation. Toutefois, ceux-ci (appelés souvent ANPR) ne prennent pas toujours en compte la particularité des plaques marocaines. *Quelles sont donc les caractéristiques de ces plaques ?*

### 1.3.1 Caractéristiques des plaques marocaines

Généralement sous forme de rectangle ou carrée, la plaque d'immatriculation marocaine est un outil permettant d'identifier les véhicules enregistrés au Maroc. Depuis l'an 2000, les immatriculations doivent respecter une nouvelle norme. Cette nouvelle configuration est composée d'une série de cinq chiffres allant de **1 à 99 999** qui correspond au numéro d'enregistrement de la voiture. Une lettre de l'alphabet arabe est incrémentée au milieu de la plaque de contrôle, ce dernier prend en compte le numéro d'enregistrement de l'automobile. Pour conclure la combinaison alphanumérique de la plaque minéralogique marocaine, le nouveau système d'immatriculation en vigueur actuellement dans le royaume chérifien termine la combinaison par l'identifiant de la préfecture d'émission de la plaque. Ces numéros vont de **1 à 89**. Donc les plaques sont maintenant du style **##### | A | ##**. Elles ont un fond blanc et les lettres sont en noir. Selon le [ministère de l'Equipement, du transport, de la logistique et de l'eau](#), il existe plusieurs **modèles de plaques d'immatriculation valides** au Maroc. Néanmoins dans notre projet, nous allons nous focaliser sur le modèle le plus répandu illustré à travers la figure suivante :

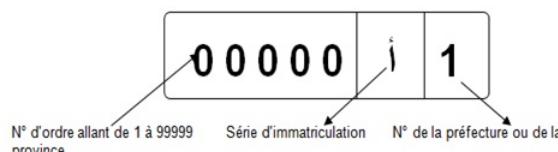


FIGURE 1.1 – Modèle sur une ligne pour les véhicules automobiles

### 1.3.2 Problématique et Objectifs du projet

Le projet sur lequel nous avons travaillé est la résultante de deux constats majeurs faits sur le trafic routier au Maroc :

1. Le premier est lié à l'**augmentation rapide du nombre de véhicules en circulation dans le Royaume chérifien**. Cette croissance accroît l'ampleur des problèmes liés au trafic routier comme le vol des voitures, les congestions, la violation des codes routiers et bien d'autres encore.
2. Le second est lié au **système actuel de gestion de parking dit intelligent** et particulièrement la méthode actuelle pour ouvrir la barrière afin d'accéder à la zone de stationnement. Pour le moment, la plupart des systèmes implantés dans le pays utilisent régulièrement des cartes **RFID**. Ce qui rend le système pas vraiment automatisé de bout en bout.

Face à ces constats, l'entreprise KF2Y Consulting dans sa démarche de création des villes intelligentes destinées au marché marocain a voulu mettre en place un système automatique de bout en bout pour la reconnaissance des plaques d'immatriculation marocaines. Cette mission nous a donc été confiée dans le cadre de notre projet de fin d'études avec les objectifs suivant :

1. **Mettre en place un système ANPR spécifique aux plaques marocaines appelé MoPlaZer performant (rapide et précis)** : Ce système doit être en mesure de localiser d'une part les plaques d'immatriculation sur une image ou une vidéo et d'autre part extraire sous format alphanumérique leur numéro d'immatriculation.
2. **Concevoir et développer une application mobile qui intègre le système** : L'application doit proposer deux modes de traitement
  - **Un mode statique** : Ici le système **MoPlaZer** traite les images fixes prises par une capture ou récupérées dans l'appareil mobile.
  - **Un mode temps réel** : Ici le système traite les séquences d'images continues (mode vidéo) à travers la caméra de l'appareil mobile
3. **Intégrer le système MoPlaZer dans un système embarqué pour un smart parking** : En effet le système **MoPlaZer** devra fournir sous format texte le numéro des plaques d'immatriculation des véhicules qui viennent à l'entrée du parking. Par la suite une vérification de l'existence du matricule détecté dans une base de données est faite. Dans le cas où le matricule se trouve dans la base de données, on déclenche l'ouverture de la barrière qui donne accès aux zones de stationnement.

## 1.4 Conduite de projet

Pour réussir tout projet, il est indispensable de mettre en place une bonne méthodologie de travail. Une bonne méthodologie de travail est celle qui donne à une équipe de pouvoir livrer un produit tout en respectant les délais, les budgets et les ressources disponibles. Par ailleurs, il faut associer à toute bonne méthodologie des outils d'organisation et de communication qui l'implémentent concrètement.

### 1.4.1 Méthodologie de travail

Dans le domaine de l'informatique, on retrouve plusieurs méthodologies de gestion des projets qui peuvent être classées principalement dans deux grands groupes :

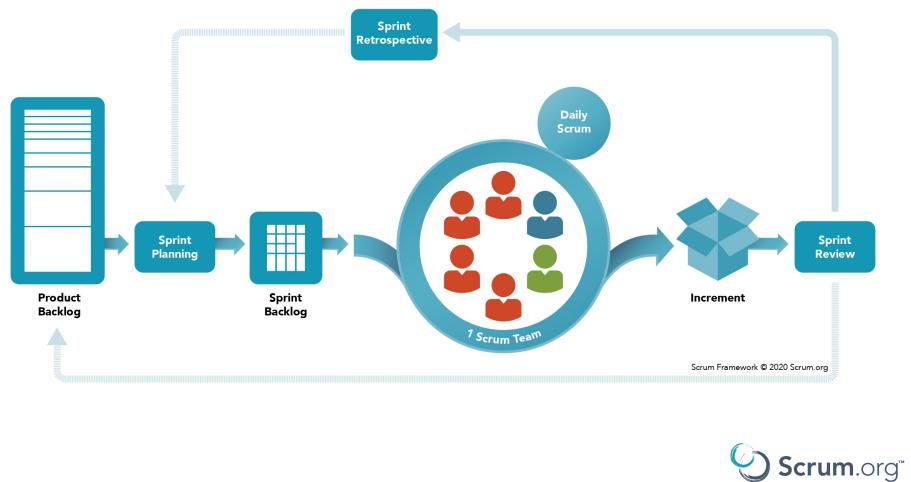
- **Les méthodes traditionnelles** : encore désignées comme classiques ou waterfall (cascade), ces méthodologies suivent la logique d'une chute d'eau. En appliquant cette méthode, l'équipe de projet suit un cahier de charges à la lettre et travaille sur la totalité du projet jusqu'à sa livraison. Rigide, elle ne prévoit pas des interactions permanentes avec le client qui ne pourra recevoir son produit qu'à la fin du projet. Toutefois, elle possède l'avantage d'être simple à mettre en œuvre.
- **Les méthodes agiles** : plus efficaces et moins rigides que les méthodologies classiques, les méthodes Agile placent les besoins du client au centre des priorités du projet. Elles offrent une plus grande flexibilité et une meilleure visibilité dans la gestion du projet, ce qui permet à l'équipe d'être plus réactive aux attentes du client. Le projet est ainsi découpé en mini-projets, chacun nécessitant la validation du client pour passer au suivant. Le dialogue avec le client est privilégié, les retours et les ajustements sont possibles. On prend davantage en considération l'évolution des besoins du client. Parmi les méthodes Agile largement connues, nous pouvons citer l'**eXtreme Programming (XP)**, **Scrum**, **Feature Driven Development (FDD)**, **Lean Software Development**, **Agile Unified Process (AUP)**, **Crystal Dynamic Systems Development Method (DSDM)**.

Toutes les méthodologies sus-citées possèdent chacune des avantages et des inconvénients. Ainsi quand il s'agit de choisir une méthode, nous ne cherchons pas la meilleure mais la plus adaptée à notre projet. Et pour ce qui concerne notre projet, puisque d'une part un cahier de charges fixe n'est pas déterminé dès le départ et d'autre part les besoins du client sont susceptibles de changer suivant l'avancement du projet, nous avons opté pour la souplesse de la méthodologie Agile et plus particulièrement la méthode Scrum.

Co-fondé dans **les années 1990** par **Ken Schwaber** et **Jeff Sutherland**, Scrum est un cadre dans lequel les gens peuvent résoudre des problèmes adaptatifs complexes, tout en fournissant de manière productive et créative des produits de la plus haute valeur possible. En bref, Scrum a besoin d'un Scrum Master pour favoriser un environnement où :

1. Un Product Owner ordonne le travail à faire pour résoudre un problème complexe dans le Product Backlog.
2. La Scrum Team transforme une sélection de ce travail en un Increment de valeur lors d'un Sprint.
3. La Scrum Team et ses parties prenantes inspectent les résultats et s'adaptent pour le prochain Sprint.
4. Répéter

## SCRUM FRAMEWORK



Scrum.org™

FIGURE 1.2 – Le cadre de travail SCRUM

Pour une implémentation réelle, le framework Scrum nécessite le regroupement et le respect d'un certain nombre d'éléments qui font sa particularité. Ce sont notamment :

- Les **piliers** : Scrum dispose de 3 grands piliers empiriques à savoir : la **transparence**, l'**inspection**, l'**adaptation**.
- Les **valeurs** : Pour réussir bien un projet Scrum, les membres de l'équipe doivent être capables de respecter cinq valeurs fondamentales : l' **engagement** à suivre les objectifs fixés, le **focus** sur le but commun, l' **ouverture** face aux difficultés professionnelles, **respect mutuel** entre les collaborateurs et enfin le **courage** pour une exécution de manière excellente des tâches et la relève des challenges.
- Les **événements** : 4 (quatre) événements majeurs permettant la création de la régularité font la méthodologie Scrum : les **Sprints** d'une durée fixe allant de 2 semaines à 1 mois au plus, le **Sprint Planning** qui lance un sprint, le **Daily Scrum** pour inspecter les progressions, le **Sprint Review** pour analyser les résultats et déterminer les adaptations futures.
- Les **artefacts** : Ceux-ci représentent un travail ou une valeur. Nous avons trois artefacts Scrum : le **Product Backlog** qui exprime l'objectif du produit, le **Sprint Backlog** définit l'objectif du sprint et l' **Increment** qui établit la signification d'un "*sprint terminé ou fini*".
- L' **équipe** : Scrum s'organise toujours autour d'une petite équipe d'au plus 10 (dix) personnes. Cette équipe est composée des **Developers** qui implémentent concrètement chaque incrément d'un sprint, le **Product Owner** qui détaille les objectifs du produit à livrer, le **Scrum Master** qui met en place le Scrum et assure l'efficacité de l'équipe.

Pour ce qui concerne notre projet, notre scrum team était constituée comme suit :

Fonction	Noms et Prénoms
Product Owner	M. Youssef
Scrum Master	M. Marouane GHOULAMI
Developers	M. Hisdaele KAMGA

TABLE 1.1 – Scrum Team du projet

Nous avons suivi des sprints d'une semaine. La définition des objectifs sont définies en début de semaine. A la fin de la semaine, un rapport sur le sprint achevé est effectué et contient l'état d'avancement du projet ainsi que les perspectives pour le prochain sprint.

#### 1.4.2 Outils d'organisation et de communication

Toute équipe de projet mise en place doit savoir d'une part s'organiser et d'autre part communiquer. Avec l'essor de l'informatique, ces tâches essentielles deviennent plus faciles. En effet, il existe plusieurs plateformes gratuites qui permettent aussi bien de planifier et suivre les activités d'un projet que de faciliter la communication au sein de groupe de travail. Pour notre projet, nous avons opté pour des outils simples et largement connus :

1. **Trello** : Lancé en septembre 2011, Trello est un outil en ligne de gestion de projet inspiré de la méthode Kanban de Toyota. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement. Ainsi Trello va nous servir à représenter les différentes informations sur les sprints en cours, effectués.
2. **TeamGantt** : Pour pouvoir planifier les différentes tâches et les visualiser, nous avons utilisé le **diagramme de Gantt**. C'est un outil pour l'ordonnancement et la gestion de projet. Elle permet d'un coup d'œil de **déterminer les dates de début et de fin des tâches**, **d'identifier les marges existantes sur certaines tâches** et de **connaître l'état d'avancement du projet en général**. Pour créer ce diagramme de Gantt pour notre projet, nous avons travaillé avec la plateforme en ligne TeamGantt. Elle permet facilement de placer de nouvelles tâches avec leurs dates de début et de fin dans un diagramme de Gantt.
3. **Skype** : C'est un logiciel qui permet de faire des échanges téléphoniques, vidéo et des messages via internet. Nous l'avons utilisé au cours de notre projet échanger des messages et partager certains fichiers ou liens utiles pour l'avancement des travaux.
4. **Microsoft OutLook** : C'est un gestionnaire d'informations personnelles et un client de courrier électronique propriétaire édité par Microsoft. Ce logiciel informatique fait partie de la suite bureautique Microsoft Office. Bien qu'il soit principalement utilisé en tant qu'application de courrier électronique, il propose aussi un calendrier et un gestionnaire de tâche et de contact. Nous avons utilisé ce logiciel pour communiquer officiellement avec le Product Owner et le Scrum Master. À travers cet outil, nous envoyons avec une certaine régularité les rapports d'avancement du projet avec les prochaines perspectives.
5. **Microsoft OneNote** : Pour les prises de notes importantes, nous avons opté pour Microsoft OneNote. C'est un programme développé par le géant Microsoft qui nous a permis de tracer les avancées journalières.
6. **Google Meet** : C'est un outil de vidéoconférence développé par Google. Nous l'avons utilisé pour faire certaines rencontres de mise au point en ligne avec le Manager M. Youssef.

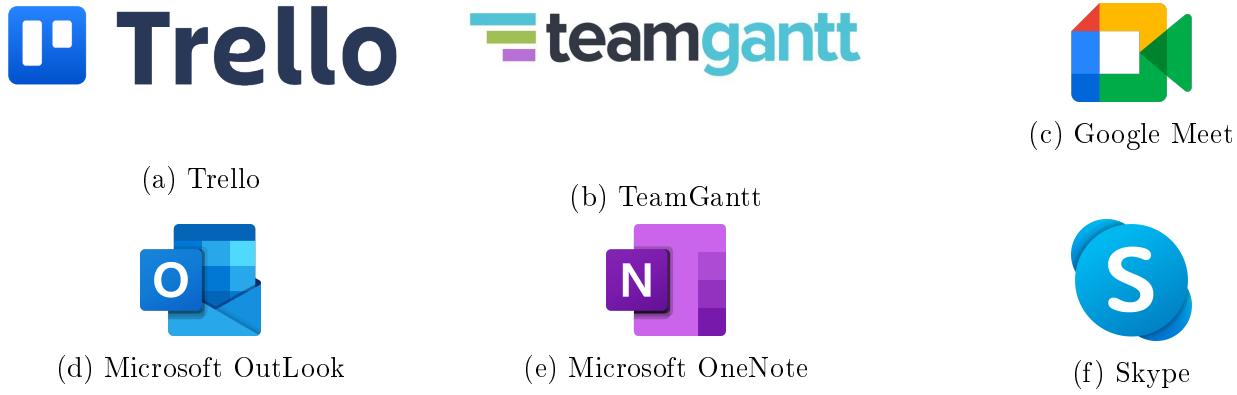


FIGURE 1.3 – Logos des outils d’organisation et de planification

## 1.5 Conclusion

Nous arrivons au terme de ce tout premier et important chapitre. Il nous a permis de mettre en lumière le contexte général dans lequel se situe notre projet de fin d’études. Il en ressort que ce projet s’inscrit dans une démarche innovante de la jeune et dynamique entreprise marocaine KF2Y Consulting. C’est une démarche pour rendre le trafic routier au Maroc plus facile, plus contrôlé et en quelque sorte intelligent. Et ceci à travers la mise en place d’un système rapide et précis de reconnaissance automatique de plaques minéralogiques marocaines qu’on appelle plus techniquement les systèmes ANPR. Si les systèmes ANPR ne sont pas encore très répandus au Maroc, dans le reste du monde ils sont déjà été et continuent à être l’objet de plusieurs études scientifiques depuis plus d’une vingtaine d’années. Nous allons donc consacrer le prochain chapitre à ces études.

# Chapitre 2

## Etude des systèmes ANPR

### 2.1 Introduction

Encore appelé [Lecture Automatisée de Plaques d’Immatriculation \(LAPI\)](#) ou encore [Automatic License Plate Recognition \(ALPR\)](#), l’ANPR est une technologie qui permet d’identifier les plaques d’immatriculation des véhicules en utilisant les techniques de traitement d’images et de vision par ordinateur. Inventés en 1976 au Royaume Uni au sein de la Police Scientific Development Branch [20], les systèmes ANPR ont fait leur chemin dans plusieurs autres pays dans le monde devenant ainsi un outil puissant et indispensable pour la sécurité du trafic routier. Nous allons à cet effet dédié ce chapitre pour répondre à quatre questions importantes autour de ces systèmes à savoir :

- *Quelle est l’architecture d’un système ANPR et de quoi se constitue-t-il principalement ?*
- *Quels sont les systèmes ANPR existants déjà sur le marché ?*
- *Dans quels champs de notre société un tel système serait-il utile ?*
- *Quels sont les problèmes qui peuvent entacher le bon fonctionnement de ces systèmes*

### 2.2 Architecture et Composants

Pour lire automatiquement les numéros des plaques d’immatriculation, un système ANPR s’étend généralement sur 4 grandes phases :

1. **Acquisition de l’image** : C’est l’entrée de tout système ANPR. À travers une caméra, le système reçoit une séquence d’images.
2. **Pré-traitement** : C’est une phase qui est déterminante pour la précision du système. Elle permet d’améliorer la qualité de l’image acquise pour faciliter les prochains traitements.
3. **Détection de la plaque** : C’est l’étape la plus importante et en même temps la plus difficile d’un système ANPR. Elle consiste à identifier sur l’ensemble de l’image la position exacte de la plaque et par la suite l’isoler du reste de l’image. Dans la littérature, plusieurs méthodes ont été proposées pour réussir cette phase à savoir :
  - **La méthode basée sur la texture** : La texture est une caractéristique qui peut être utilisée pour détecter les plaques d’immatriculation vue que les caractères d’une plaque ont une texture similaires. Dans cet algorithme, on traite les caractères d’une plaque comme une texture distincte du reste des objets contenus dans une image. La méthode est décomposée en quatre étapes qui sont les suivantes :
    - **Analyse de la texture de l’image** ;

- Décomposition de l'image en multi-segment ;
  - Choix du masque ;
  - Analyse des composantes connexes.
- **La méthode basée sur le contour et le gradient** : Les conditions de luminance non uniforme et la distance variable entre la caméra et le véhicule peuvent influencer sur le résultat de la détection des plaques d'immatriculation. Pour cela, il existe une méthode de détection des plaques d'immatriculation basée sur les caractéristiques du contour et des propriétés des caractères. L'algorithme proposé est basé sur les caractéristiques suivantes :
    - Les pixels présentant les caractères de la plaque ont souvent une valeur de contraste plus élevée par rapport aux pixels voisins.
    - Le contour des caractères d'une plaque est toujours un contour fermé.
    - Il y'a une relation de voisinage entre les caractères.De plus, cette approche qui se base sur le contour et le gradient est composée de cinq processus qui sont :
    - **Détection de contour** ;
    - **Sélection des régions candidates de caractères de la plaque d'immatriculation** ;
    - **Calcul du gradient magnitude des composantes connexes** ;
    - **Extraction des caractères** ;
    - **Localisation de la plaque d'immatriculation de véhicules.**[11]
  - **La méthode basée sur l'apprentissage profond** : Les méthodes précédentes sont dites déterministes car sont implémentées avec des fonctions mathématiques avec des paramètres fixés. A cet effet, dans l'article [18], TAO et al. proposent une méthode basée sur un modèle de Deep Learning pour la détection des plaques. En effet, à partir une masse de données d'images contenant les plaques d'immatriculation, on réalise un modèle à l'aide des algorithmes de détection des objets (Object Detection). Ce modèle servira par la suite pour détecter les plaques sur de nouvelles images. Cette approche est de plus en plus adoptée car plus efficace par rapport aux approches classiques déterministes.
4. **Reconnaissance des caractères** : Encore appelé **OCR**, le but de cette phase est d'extraire sous format textuel (alphanumérique), le numéro de la plaque de d'immatriculation. Si certaines méthodes [17, 23] utilisent l'approche classique qui passent par 3 phases (pré-traitement, segmentation de caractères et reconnaissance de caractères), d'autres [1, 18] par contre optent pour une approche nouvelle en utilisant directement un modèle de réseaux de neurones pour la reconnaissance des caractères sur l'image de la plaque. Toutefois quelque soit la méthode, une étape de post-traitement sur le texte détecté est nécessaire pour faire d'éventuelles corrections sous la base de la norme d'immatriculation des véhicules spécifique au pays. A la fin de la chaîne, le texte obtenu et nettoyé est soit enregistré dans une base de données ou envoyé à un autre système pour un traitement ultérieur.

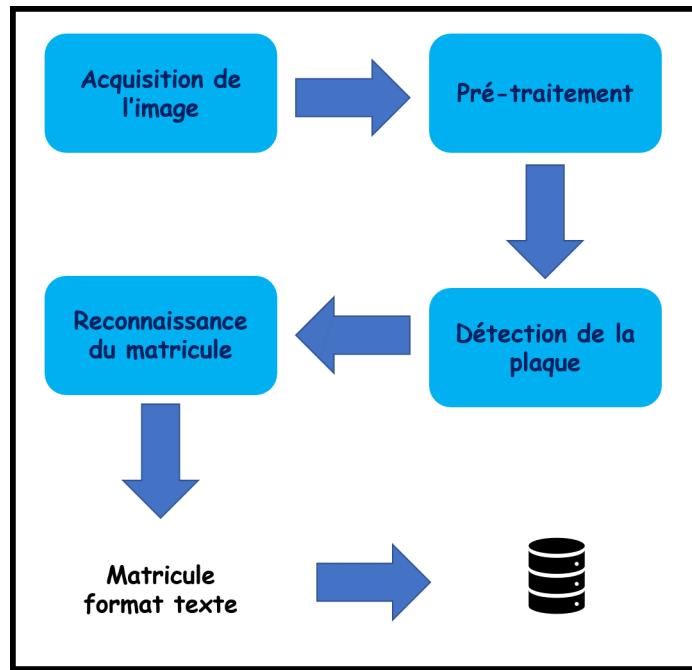


FIGURE 2.1 – Architecture de fonctionnement d'un système ANPR

Tout système ANPR est constitué essentiellement de deux parties :

1. **Une partie matérielle** : Elle est composée d'une caméra qui permet d'acquérir les images et d'un ordinateur standard sur lequel s'exécute la partie logicielle du système.
2. **Une partie logicielle** : C'est le programme qui traite les images en vue d'extraire les numéros des plaques d'immatriculation. Elle est la plupart du temps liée à une base de données où sont stockées les matricules.

## 2.3 Systèmes existants

De nos jours, on retrouve sur le marché international, une variété de systèmes ANPR commerciaux tels que :

- **AutoVu** : C'est le système de [Reconnaissance Automatique de Plaques d'Immatriculation \(RAPI\)](#) sur IP de Security Center qui automatise la lecture et vérification de plaques d'immatriculation de véhicules. Il utilise des caméras qui capturent des images de plaques et transmettent les données à un Patroller ou Security Center, qui recherche la plaque dans des listes de véhicules recherchés ou de permis. [7]
- **LAPI ENGINE** : C'est un [OCR](#) de lecture de plaques d'immatriculation permettant d'identifier et de tracer les véhicules passant sur une zone et se présentant devant une caméra, à partir d'un flux vidéo en direct ou de séquences vidéo d'événements détectés par les caméras. [19]
- En plus des systèmes cités plus haut, nous avons d'autres comme les **caméras de contrôle de vitesse TrajectControl** qui sont implantées au Pays-Bas depuis 2002, les **systèmes de reconnaissance Zamir Ltd** de Jérusalem en Israël, **SeeTec**, **Asia Vision Technology Limited** et bien d'autres encore. [2, 10]

## 2.4 Domaines d'application et Difficultés

Depuis leur invention, les systèmes de reconnaissance automatique de plaques d'immatriculation n'ont cessé d'étendre leurs champs d'action. Ils sont même devenus incontournables dans les systèmes modernes et intelligents de gestion de trafic routier. Parmi les larges gammes d'applications de l'ANPR, on peut citer :

- **Vol de voitures** : le système est déployé sur le bord des routes, et réalise une comparaison en temps réel entre les voitures qui passent et la liste des voitures volées. Lorsqu'une correspondance est trouvée, une alerte est déclenchée pour informer l'agent de police de la voiture détectée et les raisons pour arrêter la voiture.
- **Parking** : le numéro de la plaque d'immatriculation est utilisé pour le paiement du stationnement au parking pour les gens ayant des cartes près-payée pour les parkings, afin de calculer les frais de stationnement en comparant les temps d'entrée et de sortie au parking.
- **Péage** : le numéro de la voiture est utilisé pour calculer les frais de voyage dans une route à péage, ou utilisé pour vérifier le billet.
- **Contrôle d'accès** : l'ouverture automatique d'une porte pour les membres agréés dans une zone de sécurité. Ce genre de système est mis en place pour aider les agents de sécurité. Les événements sont enregistrés sur une base de données et peuvent être utilisés pour rechercher l'historique des événements en cas de besoin.
- **Contrôle des frontières** : le numéro de la voiture est enregistré à l'entrée ou à la sortie du pays, et utilisé pour surveiller les passages frontaliers. Chaque véhicule est enregistré dans une base de données centrale et lié à des informations supplémentaires telle que les données relatives aux passeports. Il est utilisé pour suivre tous les passages frontaliers.
- **Code pénal de la route** : le numéro de plaque est utilisé pour produire une amende de violation de vitesse ou de feux rouges. Le processus manuel de préparation d'une amende de violation est remplacé par un processus automatisé qui réduit les surcharges et les délais. Les amendes peuvent être consultées et payées en ligne. [2]

Depuis certaines années, les pays dans le monde normalisent le format de leurs plaques d'immatriculation. Cette normalisation intervient d'une part sur l'uniformisation des couleurs possibles et d'autre part la liste et le nombre de caractères que l'on peut trouver sur une plaque valide. Cette régularisation des plaques minéralogiques a rendu les systèmes ANPR plus spécifiques et donc plus performants. Néanmoins, comme tout système faisant du traitement d'image, les systèmes ANPR font face à des difficultés comme :

- **Une mauvaise résolution de l'image** qui est la conséquence soit d'une caméra de mauvaise qualité soit de l'éloignement de la plaque par rapport à la caméra ;
- **Des images floues** principalement causées par le mouvement ;
- **Les mauvaises conditions climatiques** : Les phénomènes naturels comme la pluie, la poussière et le brouillard peuvent empêcher une bonne capture des images et donc impacter négativement la précision d'un système ANPR.
- **La différence de polices de caractères de la plaque** à cause de la fantaisie faite par certains usagers. [20]

Si certains de ces problèmes peuvent être surmontés en agissant sur la partie logiciel du système, d'autres par contre nécessitent une intervention au niveau matériel.

## 2.5 Conclusion

Ce chapitre nous a permis de comprendre que les systèmes ANPR ne sont pas une nouveauté. En réalité, plusieurs études ont déjà été menées de long en large dans le domaine. Ces travaux ont ainsi contribué d'une part à standardiser plus au moins l'architecture générale d'un système ANPR et d'autre part à déterminer les composants essentiels de ce système. Ces travaux sont aussi le socle sur lequel plusieurs entreprises ont construit leur propre solution ANPR qui fait face plus ou moins bien aux difficultés que rencontrent la majorité des technologies de traitement d'images. Le prochain chapitre sera donc une occasion d'exposer les différentes techniques de traitements d'images.

# Chapitre 3

## Traitement d'images

### 3.1 Introduction

Les systèmes ANPR sont essentiellement des technologies axées sur le traitement d'images numériques. Les images sont dans une certaine mesure à un système ANPR ce que la matière première est pour l'industrie. Elles passent par de plusieurs transformations et d'opérations afin d'extraire une information qui est le numéro d'immatriculation sous format textuel. Ce chapitre permet donc de présenter ces différentes transformations et opérations. Mais avant d'y arriver commençons par exposer les généralités sur les images numériques.

### 3.2 Généralités sur les images numériques

#### 3.2.1 Qu'est-ce qu'une image numérique ?

Une image numérique est toute image (dessin, icône, photographie ...) acquise, créée, traitée ou stockée sous forme binaire (suite de 0 et de 1) :

- **Acquise** par des dispositifs comme les scanners, les appareils photo ou caméscopes numériques, les cartes d'acquisition vidéo (qui numérisent directement une source comme la télévision).
- **Créée** directement par des programmes informatiques, via la souris, les tablettes graphiques ou par la modélisation 3D (ce que l'on appelle par abus de langage les « images de synthèse »).
- **Traitée** grâce à des outils informatiques. Il est facile de la modifier en taille, en couleur, d'ajouter ou supprimer des éléments, d'appliquer des filtres variés, etc.
- **Stockée** sur un support informatique (disquette, disque dur, CD-ROM(Compact Disk Read Only Memory), ...) [21]

#### 3.2.2 Représentation de l'image

Selon la manière dont les images sont représentées pour être visualisées sur les écrans des ordinateurs, on distingue deux grandes classes d'images numériques :

- **Les images matricielles** : Encore appelées images bitmap, elles sont représentées à travers des matrices de points à plusieurs dimensions. Dans le cas de deux dimensions spatiales (largeur, hauteur), on parle d'**images 2D** et les points sont appelés **pixels (Picture Element)**. C'est le type le plus répandu. Si une image 2D possède en plus un composant

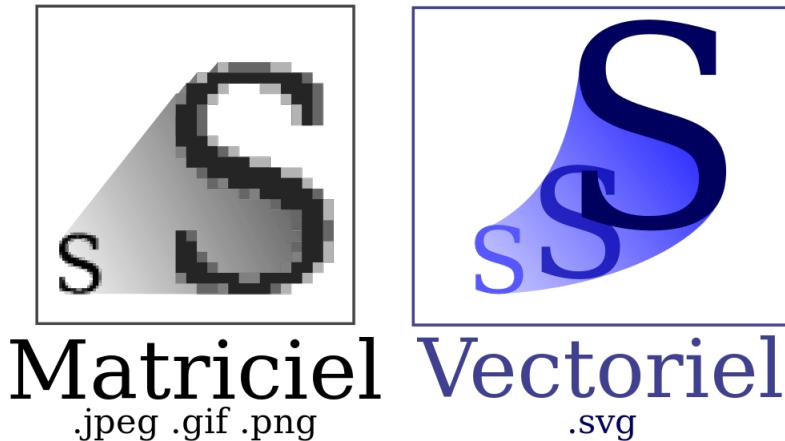


FIGURE 3.1 – Différence entre image matricielle et image vectorielle

Source: [Site Cours en ligne](#)

temporel (**image 2D + t**), on parle dans ce cas d'**animation**. Si par contre l'image possède trois dimensions spatiales (largeur, hauteur, profondeur), on parle d'**image 3D** ou **volume** et les points sont appelés **voxels (Volume Element)**.

- **Les images vectorielles** : Les données d'une image vectorielle sont représentées par des formes géométriques (cercle, rectangle, ...) qui sont décrites mathématiquement. Elles possèdent l'avantage contrairement aux images vectorielles d'occuper moins d'espace mémoire et la flexibilité dans le redimensionnement sans perte d'informations.

Dans la famille des images matricielles, on distingue trois types d'images :

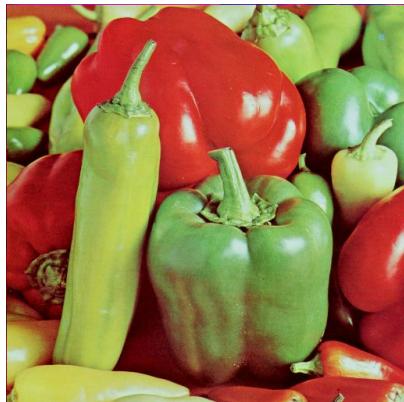
- **Les images binaires** : Elles sont uniquement en noir et blanc. La matrice de représentation contient uniquement des 0 et 1.
- **Les images en niveaux de gris** : La matrice de représentation contient des valeurs entières entre 0 et 255 qui correspondent au niveau d'intensité lumineuse.
- **Les images de couleur** : Elles sont régulièrement obtenues par une synthèse additive des trois couleurs fondamentales :rouge, vert, bleu (RVB ou RGB en anglais). La matrice de représentation est constituée dans ce cas de trois matrices : pour chaque couleur, une matrice dont chaque cellule donne l'intensité lumineuse de la couleur correspondante.

Dans le cadre de notre projet, nous travaillons exclusivement avec les images matricielles 2D. Donc pour la suite, lorsqu'on parle d'image, il s'agit d'une image matricielle 2D sauf s'il est mentionné le contraire.

### 3.2.3 Caractéristiques d'une image numérique

Une image a trois caractéristiques principales :

- **Les dimensions** : Elles représentent la largeur et la hauteur de l'image généralement exprimées en pixels.
- **La définition** : C'est le nombre de pixels constituant une image. Elle est obtenue en multipliant le nombre de pixels sur la largeur par le nombre de pixels sur la hauteur.
- **La résolution** : C'est le nombre de pixels que l'on retrouve sur une unité de longueur. Elle s'exprime généralement en ppp (point par pouce) ou encore en anglais dpi (dot per inch). Plus la résolution est importante, plus les points sont petits et nombreux, plus l'image est



(a) Image de couleur



(b) Image en niveau de gris



(c) Image binaire

FIGURE 3.2 – Exemples de types d'images matricielles

fine.

### 3.2.4 Formats des images numériques

Le format d'une image est la représentation informatique de cette image. Il donne les informations nécessaires sur la manière dont l'image a été codée et éventuellement des moyens de la décoder et de la manipuler. Il existe plusieurs formats qui plus ou moins adaptés pour certains cas d'utilisation :

- **Format Graphics Interchange Format (GIF)** : Ce format permet la transparence et les images animées plusieurs images séquentielles à l'intérieur du même fichier. Il est utilisé pour des logos, des icônes, des boutons et autres éléments de pages web. Le format d'image GIF n'atteigne au maximum que 256 couleurs, il n'est donc pas du tout adapté aux photos et à l'impression.
- **Format Tagged Image File Format (TIFF)** : Un des formats le plus couramment utilisé pour stocker des images, des photographies. Il est couramment utilisé dans les environnements professionnels et pour l'impression commerciale. Il est considéré comme étant le format le plus fiable pour des impressions de haute qualité comme pour le textile, les tissus.
- **Format Joint Photographic Experts Group (JPG/JPEG)** : C'est le format le plus adéquat pour administrer ses photos et les publier. Un des formats les plus utilisés sur le net (les navigateurs l'affichent correctement), et dans les mails. Les appareils photo numériques compacts prennent également les photos au format JPG.
- **Format Portable Network Graphics (PNG)** : Un des formats le plus couramment utilisé. Crée pour remplacer le GIF il est très peu connu du grand public. Performant, il réunit presque tous les avantages du JPEG et du GIF et permet les fonds transparents. La compression proposée par ce format est perte d'une qualité 5 à 25 % meilleure que la compression GIF. [11]
- **Format Scalable Vector Graphics (SVG)** : Contrairement aux précédents formats, SVG est un format d'images vectorielles. Il est conçu pour décrire un ensemble de graphiques vectoriels en utilisant XML. Cette conception permet aux images sous ce type de format d'être agrandies à l'infini sans impact sur la qualité. On utilise généralement ce format pour échanger des graphiques sur internet ou dans des logiciels de dessin assisté par ordinateur.

ordinateur(DAO) pour représenter les objets.

### 3.3 Opérateurs de traitement d'images

A l'instar des opérateurs mathématiques, les opérateurs de traitement d'images prennent en entrée une image et un ensemble d'informations relatives à cette image, effectuent des modifications sur ces entrées et retournent une nouvelle image ou un ensemble d'informations relatives aux données d'entrée. Ces opérateurs sont nombreux. Mais nous allons en citer quelques-unes.

#### 3.3.1 Opérations morpho-mathématiques

La morphologie est un vaste ensemble d'opérations de traitement d'images qui traitent des images en fonction de formes. Les opérations morphologiques appliquent un élément structurant à une image d'entrée, créant une image de sortie de même taille. Dans une opération morphologique, la valeur de chaque pixel de l'image de sortie est basée sur une comparaison du pixel correspondant de l'image d'entrée avec ses voisins. Elles peuvent être utilisées pour supprimer les bruits sur une image. Les principales opérations morphologiques sont :

- **La dilatation** : La valeur du pixel de sortie est la valeur maximale de tous les pixels du voisinage. Dans une image binaire, un pixel est défini sur 1 si l'un des pixels voisins a la valeur 1. La dilatation morphologique rend les objets plus visibles et comble les petits trous dans les objets.



FIGURE 3.3 – Dilatation

- **L'érosion** : La valeur du pixel de sortie est la valeur minimale de tous les pixels du voisinage. Dans une image binaire, un pixel est défini sur 0 si l'un des pixels voisins a la valeur 0. L'érosion morphologique supprime les îles et les petits objets de sorte que seuls les objets substantifs restent.



FIGURE 3.4 – Erosion

- **L'ouverture** : L'opération d'ouverture érode une image puis dilate l'image érodée, en utilisant le même élément structurant pour les deux opérations. L'ouverture morphologique est utile pour supprimer les petits objets d'une image tout en préservant la forme et la taille des objets plus gros dans l'image.



FIGURE 3.5 – Ouverture

- **La fermeture** : L'opération de fermeture dilate une image puis érode l'image dilatée, en utilisant le même élément structurant pour les deux opérations.La fermeture morphologique est utile pour combler les petits trous d'une image tout en préservant la forme et la taille des objets de l'image.[14]



FIGURE 3.6 – Fermeture

### 3.3.2 Détection des contours

La détection de contours est une étape essentielle du processus de traitement d'images qui permet une réduction importante de la quantité d'information relative à une image, tout en préservant des informations structurelles comme les contours et les frontières des images.Elle consiste à repérer les points d'une image numérique qui correspondent à un changement brutal de l'intensité lumineuse. En effet, un contour se matérialise par une rupture d'intensité dans l'image suivant une direction donnée. Plusieurs méthodes existent pour détecter cette rupture, les unes plus ou moins complexes, les autres plus ou moins gourmandes en calcul.

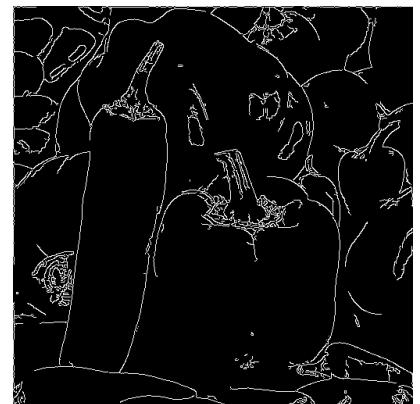
- **Le filtre de Prewitt** : Prewitt est l'un des premiers algorithmes de détection de contours dans le domaine du traitement d'image. Il s'agit d'une approximation du gradient par convolution de l'image avec des masques de convolution.
- **Le filtre de Sobel** : Le filtre de Sobel détecte séparément les bords horizontaux et verticaux sur une image en niveaux de gris.Et on peut aussi appliquer le détecteur de Sobel sur des images en couleurs en appliquant le même algorithme sur les différentes composantes RGB prises séparément. Ainsi qu'on peut également l'appliquer le uniquement sur la composante de luminance.
- **Le filtre de Canny** :La méthode de Canny implémente une estimation du gradient de l'image à l'aide du filtre de Sobel, suivi d'un seuillage par hystérésis du module de gradient. Un seuil haut et un seuil bas sont à définir. Tous les pixels où le module du gradient est supérieur au premier seuil sont classifiés comme appartenant aux contours de l'image, des contours de l'image sont ainsi formés. Les pixels ayant un module supérieur au seuil bas et qui sont segmentés précédents sont définis comme points de contour dans l'image binaire résultante.[11]



(a) Filtre de Prewitt



(b) Filtre de Sobel



(c) Filtre de Canny

FIGURE 3.7 – Détection des contours avec différents filtres

### 3.4 Conclusion

La vision par ordinateur n'est possible qu'à travers une série d'algorithmes de traitement de d'images qui ont été développées depuis de très longues années. Ces algorithmes puissants permettent d'une part de modifier une image numérique et d'autre part d'y extraire des informations. Au cours de ce chapitre, après avoir donné quelques généralités sur les images numériques, nous avons présenté brièvement quelques méthodes qui ont été proposées pour les traiter. Par ailleurs, le traitement d'images numériques possède de nombreuses applications dont l'une qui entre dans le processus des systèmes ANPR est la reconnaissance optique de caractères. C'est l'objet du chapitre suivant.

# Chapitre 4

## Reconnaissance Optique de Caractères

### 4.1 Introduction

Si les humains peuvent très facilement reconnaître les caractères sur un document, notamment une image est chose facile, ce n'est pas le cas pour les ordinateurs. En effet, comme nous l'avons vu au chapitre précédent, une image pour un ordinateur n'est qu'une série de points. Le besoin croissant de numérisation des documents imprimés a fait naître une nouvelle technologie qu'on appelle généralement [Optical Character Recognition \(OCR\)](#). Ce chapitre nous permettra de comprendre ce qu'est réellement cette technologie d'abord, ensuite nous présenterons les phases d'un processus d'OCR et nous terminerons par les outils utilisés pour réaliser l'OCR.

### 4.2 Définition

L'OCR ou encore la reconnaissance optique de caractères est une technologie complexe qui convertit les images contenant du texte en formats avec du texte modifiable. Il permet de traiter des livres numérisés, des captures d'écran et des photos contenant du texte et d'obtenir des documents modifiables tels que des fichiers TXT, DOC ou PDF.

### 4.3 Phases d'OCR

Pour arriver à extraire le texte se trouvant sur un document, la technologie OCR le fait passer par une succession de plusieurs étapes.



FIGURE 4.1 – Processus d'OCR

### 4.3.1 Prétraitement

L'objectif de cette étape est d'améliorer la qualité de l'image et de faciliter son traitement pour de meilleurs résultats. Pour cela, on utilise quelques méthodes comme :

- **La binarisation** : Il s'agit de convertir une image en couleur qui est en trois dimensions en une image bidimensionnelle noir sur blanc. Pour cela, on passe par une étape intermédiaire appelée le *grayscaleing* qui convertit l'image en gris et en utilisant le seuillage sur l'image obtenue pour aboutir à une image en noir et blanc.
- **Le redressement** : Lors du processus de numérisation des documents, il peut y arriver quelques déformations sur le document numérisé obtenu. Ces déformations peuvent être des inclinaisons du texte. Cette étape permet de redresser l'image afin que les textes par exemple soient placées horizontalement.
- **Le zonage** : Il permet de sélectionner la partie de l'image où l'on veut extraire les informations.
- **Le suppression des bruits** : Ici on enlève les éléments qui peuvent être considérés comme du bruit de l'image. Ceci peut des lignes verticaux, horizontaux, obliques, des points en arrière plan.
- **L'amincisement et la squelettisation** : Il permet d'uniformiser la taille du texte dans l'image.

### 4.3.2 Segmentation

Le but de cette phase est d'isoler les lignes et les caractères se trouvant sur l'image. Il existe donc trois types de segmentation :

- **La segmentation des lignes** : L'objectif principal de ce type est de déterminer les coordonnées des lignes dans l'image, ce qui peut diviser l'image en lignes. L'idée est que, si nous projetons horizontalement l'image binaire, les lignes qui ont les pixels de texte ont des pics plus élevés, en raison du plus grand nombre de pixels de premier plan et les lignes qui ont l'espacement entre les lignes ont des pics plus faibles, en raison du plus grand nombre de pixels d'arrière-plan.
- **La segmentation des mots** : L'objectif principal de ce type est de déterminer où nous devons segmenter l'image pour séparer les mots. L'idée est la même que l'étape précédente, mais le seul changement est ici que nous projetons l'image verticalement parce que nous séparons les mots verticalement. Les colonnes qui ont les pixels de texte ont des pics plus élevés, en raison de plus de nombre de pixels de premier plan et les colonnes qui ont l'espace entre les mots contiennent des pics plus faibles, en raison de plus de nombre de pixels d'arrière-plan.
- **La segmentation des caractères** : la tâche de la segmentation au niveau des caractères consiste à séparer les caractères uniques (comme les alphabets, les chiffres et autres symboles) des mots qui sont séparés de l'étape précédente.

Pour la segmentation des images, la technique la plus utilisée est la projection histogramme, on cite deux types de projection :

- **La projection horizontale** : Utilisée dans la segmentation au niveau des lignes.
- **La projection verticale** : Utilisée dans la segmentation au niveau des mots et des caractères.

### 4.3.3 Reconnaissance de caractères

Le but est d'identifier numériquement le caractères qui est en image. Pour y parvenir, il existe des techniques de reconnaissance qui peuvent classées en trois grands groupes :

1. **La classification par caractéristiques (Features)** : une forme à reconnaître est représentée par un vecteur de valeurs numériques - appelées features en anglais - calculées à partir de cette forme. Le nombre de features est de l'ordre de 100 à 300. Si les features sont bien choisies, une classe de caractères (par exemple l'ensemble des A majuscules) sera représentée par un « nuage » contigu de points dans l'espace vectoriel des features. Le rôle du classificateur est de déterminer à quel nuage (donc à quelle classe de caractères) la forme à reconnaître appartient le plus vraisemblablement. La classification fait généralement appel à divers types de réseaux de neurones artificiels entraînés sur de vastes bases de formes possibles.
2. **Les méthodes métriques** : consistent à comparer directement la forme à reconnaître, au moyen d'algorithmes de distance, avec un ensemble de modèles appris. Ce type de méthode est peu utilisé et peu valorisé par les chercheurs, car souvent plus naïf et vraisemblablement moins efficace que les méthodes à base de features.
3. **Les méthodes statistiques** : dans le domaine de la reconnaissance d'écriture manuscrite, il est fréquemment fait appel aux méthodes probabilistes comme les chaînes de Markov.[22]

### 4.3.4 Post-traitement

L'objectif de cette phase est de réduire les erreurs de reconnaissance. On utilise à cet effet des méthodes linguistiques et contextuelles pour identifier les mauvaises reconnaissances et proposer une éventuelle correction.

## 4.4 Outils

De nombreuses API et bibliothèques ont été développées pour effectuer les opérations d'OCR sur les documents. Parmi elles, nous pouvons citer :

- **Tesseract** : c'est un moteur de reconnaissance optique de caractères écrit en C et C++ pour divers systèmes d'exploitation (Linux, Windows et Mac OS X). Il s'agit d'un logiciel gratuit, publié sous la licence Apache, et le développement est parrainé par Google depuis 2006. En 2006, Tesseract était considéré comme l'un des moteurs OCR open source les plus précis. Avec la version 4, un moteur et des modèles OCR basés sur LSTM (Long-Short Term Memory) pour de nombreuses langues et scripts supplémentaires ont été ajoutés, portant ainsi le nombre de langues traitées à 116.
- **EasyOCR** : comme Tesseract, EasyOCR est une bibliothèque open source d'OCR. Développé par JaidedAI , il prend en charge plus de 80 langues et tous les scripts d'écriture courants, notamment le latin, le chinois, l'arabe, le cyrillique, etc. Par rapport à Tesseract, il possède l'avantage d'être rapide lors d'une exécution avec le GPU.
- **MLKit Text Recognition** : C'est API développé par Google pour l'OCR. Elle est principalement utilisée dans les applications mobiles et prend uniquement en compte les caractères latins.

## 4.5 Conclusion

L'OCR est la technologie qui permet aux ordinateurs de lire les documents comme les humains. Elle s'étend sur 4 principales phases dont l'une des plus cruciales est la reconnaissance des caractères. Des méthodes ont été proposées pour réussir de plus en plus cette phase. L'une des méthodes qui a fait ces preuves est l'apprentissage automatique ou le Machine Learning. Ce concept qui s'est répandu dans le monde de l'informatique sera l'objet du prochain chapitre.

# Chapitre 5

## Apprentissage Automatique

### 5.1 Introduction

Encore appelé Machine Learning ou apprentissage statistique, l'apprentissage automatique est une discipline, un champ d'étude de l'intelligence artificielle qui donnent aux ordinateurs la capacité d'apprendre à partir des données. En effet, à partir des ces données, les algorithmes de Machine Learning construisent des modèles qui sont comme des fonctions mathématiques qui s'appliqueront sur de nouvelles données. La machine devient ainsi capable de résoudre des problèmes sans être explicitement programmée.

### 5.2 Phases d'un cycle de Machine Learning

Pour construire un modèle de Machine Learning qui répond à un problème bien défini au préalable, il est indispensable de suivre une chaîne qui se décline en 5 phases :

1. **Acquisition des données** : la possession de données historiques est la source de tout apprentissage. De plus, plus la quantité est élevée, meilleur sera l'apprentissage. Ces données peuvent provenir de plusieurs sources différentes : Web, sources internes et bien d'autres encore.
2. **Nettoyage, Préparation, Manipulation des données** : les données ont besoin d'une retouche avant d'être utilisées. En effet, certains attributs sont inutiles, d'autre doivent être modifiés afin d'être compris par l'algorithme et enfin, certains éléments sont inutilisables car leurs données sont incomplètes. Plusieurs techniques telles que la Data Visualisation, la Data Transformation ou encore la Normalisation permettent de gérer cette problématique.
3. **Création du modèle** : une fois les données clarifiées, un ou plusieurs modèles sont créés à l'aide d'algorithmes de Machine Learning.
4. **Évaluation** : une fois le modèle réalisé, il faut l'évaluer pour prouver son efficacité et sa performance. En utilisant un second jeu de données, on identifie la précision du modèle et on répète l'étape 3 et 4 jusqu'à l'obtention de la meilleure performance.
5. **Déploiement** : la dernière étape est la mise en production du modèle. Il est désormais possible de l'utiliser pour les nouvelles données entrantes. Le système est itératif puisque ces nouvelles données sont réutilisées par l'algorithme pour améliorer le modèle. Le système est en perpétuel évolution et amélioration.

## 5.3 Méthodes d'apprentissage automatique

La création d'un modèle se fait à l'aide d'un algorithme de Machine Learning spécifique au type de problème à résoudre. On classe généralement ces algorithmes en plusieurs classes dont 3 sont les plus connues à l'heure actuelle.

### 5.3.1 Apprentissage supervisé

Les algorithmes d'apprentissage automatique supervisés peuvent appliquer ce qui a été appris dans le passé à de nouvelles données en utilisant des exemples étiquetés pour prédire des événements futurs. À partir de l'analyse d'un ensemble de données d'apprentissage connu, l'algorithme d'apprentissage produit une fonction inférée pour faire des prédictions sur les valeurs de sortie. Le système est capable de fournir des objectifs pour toute nouvelle entrée après une formation suffisante. L'algorithme d'apprentissage peut également comparer sa sortie avec la sortie correcte et prévue et trouver des erreurs afin de modifier le modèle en conséquence.<sup>[6]</sup> Selon le type de variable à prédire, on distingue deux types d'apprentissage supervisé : la **classification** pour les variables qualitatives, discrètes et la **régression** pour les variables quantitatives, continues. Les algorithmes les plus connus d'apprentissage supervisé sont :

- La **machine à vecteurs de support** ([SVM](#) en anglais)
- La **méthode des k plus proches voisins** ([KNN](#) en anglais)
- L'**arbre de décision**
- La **classification naïve bayésienne**
- Les **réseaux de neurones**

### 5.3.2 Apprentissage non-supervisé

Les algorithmes d'apprentissage automatique non supervisés ont utilisés lorsque les informations utilisées pour l'entraînement ne sont ni classées ni étiquetées. L'apprentissage non supervisé étudie comment les systèmes peuvent déduire une fonction pour décrire une structure cachée à partir de données non étiquetées. Le système ne trouve pas la bonne sortie, mais il explore les données et peut tirer des inférences à partir d'ensembles de données pour décrire des structures cachées à partir de données non étiquetées.<sup>[6]</sup> Parmi les algorithmes d'apprentissage non-supervisé, on retrouve :

- **K-Moyenne** (K-Means en anglais)
- La **réduction de la dimensionnalité**
- L'**analyse des composants principaux**

### 5.3.3 Apprentissage par renforcement

Le renforcement des algorithmes d'apprentissage automatique est une méthode d'apprentissage qui interagit avec son environnement en produisant des actions et découvre des erreurs ou des récompenses. La recherche par essais et erreurs et la récompense différée sont les caractéristiques les plus pertinentes de l'apprentissage par renforcement. Cette méthode permet aux machines et aux agents logiciels de déterminer automatiquement le comportement idéal dans un contexte spécifique afin de maximiser ses performances. Un simple retour de récompense est nécessaire pour que l'agent sache quelle action est la meilleure ; c'est ce qu'on appelle le signal de renforcement.<sup>[6]</sup>

L'une des applications de ces algorithmes se trouve dans le domaine de la vision par ordinateur avec la reconnaissance objets sur une image. Et pour cela, on utilise général une sous-branche du Machine Learning appelée Deep Learning ou encore apprentissage profond.

## 5.4 Deep Learning

### 5.4.1 Définition

Comme une branche du Machine Learning, le Deep Learning est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires. L'apprentissage profond se distingue de l'apprentissage automatique sur plusieurs points :

- D'une part, les algorithmes d'apprentissage classique nécessitent une expertise humaine pour l'extraction des caractéristiques. Ceci n'est pas nécessaire pour les algorithmes d'apprentissage profond qui font cette extraction de manière automatique via des transformations non-linéaires multi-couches.
- D'autre part, une différence notable entre le Machine Learning et le Deep Learning se situe au niveau de l'adaptabilité. En effet, les techniques d'apprentissage profond peuvent être adaptées à différents domaines et applications bien plus facilement que les algorithmes de ML classiques. Par exemple, en vision par ordinateur, les réseaux de classification d'images pré-entraînés sont souvent utilisés pour l'extraction de caractéristiques pour faire la détection et la segmentation des objets. L'utilisation de ces réseaux pré-entraînés facilite l'apprentissage du modèle et permet souvent d'obtenir des performances supérieures en un temps réduit.
- Un autre point est au niveau de l'évolution de la précision d'un modèle avec l'évolution des données d'entraînement. Pour augmenter la précision d'un modèle réalisé par les algorithmes de DL, il suffit généralement d'augmenter la quantité de données pour l'entraînement. Ce n'est pas toujours le cas pour les modèles faits par les algorithmes classiques de ML qui sont plus utiles avec une base de données de petite taille.
- En règle générale, un algorithme d'apprentissage profond prend beaucoup de temps pour faire l'apprentissage à cause du grand nombre de paramètres qu'il utilise. Par contre, les algorithmes traditionnels de **ML** prennent quelques secondes à quelques heures pour s'entraîner. Le scénario est complètement inversé en phase de test. Au moment du test, l'algorithme de **DL** prend beaucoup moins de temps à s'exécuter. [4]
- Un dernier point de divergence mais pas des moindres est les performances matérielles pour l'entraînement. Si d'un côté, algorithmes de **ML** peuvent effectuer l'entraînement d'un modèle sur le **CPU**, d'autre côté les algorithmes ont essentiellement besoin d'un **GPU** pour un entraînement efficace.

En général, nous remarquons que le **DL** possède des performances élevées par rapport au **ML**. Ceci s'explique principalement par le fait que les algorithmes sont construits pour imiter le processus de réflexions des hommes en utilisant les réseaux de neurones artificiels.

### 5.4.2 Réseaux de neurones artificiels

L'unité de base du calcul dans un réseau de neurones artificiel est le **neurone**. Un neurone artificiel reçoit des entrées de certains autres neurones ou d'une source externe ayant des valeurs numériques  $x_1, x_2, \dots x_n$  auxquels il est connecté par des synapses et calcule une sortie  $y$ . Chaque

entrée  $x_i$  a un poids associé  $w_i$ , qui est attribué en fonction de son importance relative par rapport aux autres entrées. La valeur d'entrée  $x$  du neurone correspond à la somme pondérée de ses entrées en ajoutant une autre entrée ayant un poids  $b$  appelé **biais**. Ensuite, le neurone applique une fonction  $f$  sur cette somme. La sortie du neurone  $y$ , appelée **activation de sortie**, est calculée par la formule suivante :

$$y = f\left(\sum_{i=1}^n w_i x_i + b\right) \quad (5.1)$$

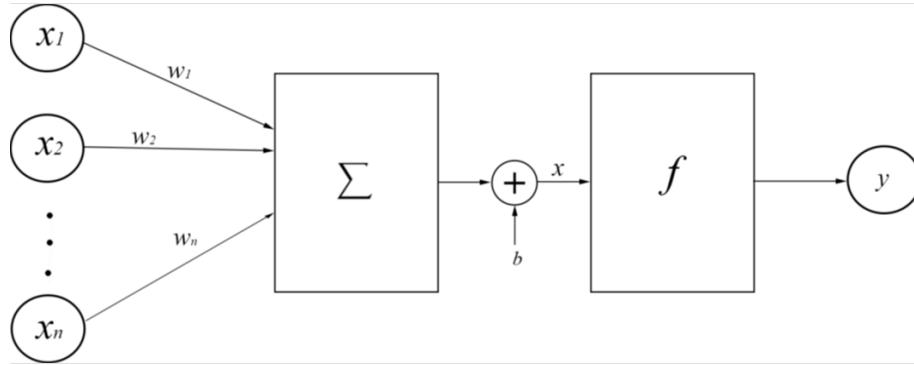


FIGURE 5.1 – Modélisation d'un neurone artificiel

La fonction  $f$ , appelée **fonction d'activation**, est non linéaire. Elle a pour but d'introduire la non-linéarité dans la sortie d'un neurone. Il existe plusieurs fonctions d'activation utilisées dans la pratique :

- **La fonction sigmoïde** : prend une entrée réelle et la réduit entre 0 et 1. Elle est définie par :

$$f(x) = \frac{1}{1 + e^{-x}} \quad (5.2)$$

- **La fonction tangente hyperbolique** : prend une entrée de valeur réelle et la réduit à une valeur dans  $[-1, 1]$ . Elle est définie par :

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (5.3)$$

- **La fonction unité de rectification linéaire (ReLU)** : prend une entrée réelle et retourne cette entrée si elle est positive et 0 sinon :

$$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases} \quad (5.4)$$

- **La fonction softmax** : prend un vecteur à valeurs réelles et le réduit à un vecteur de valeurs comprises entre zéro et un qui égal :

$$f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (5.5)$$

En connectant plusieurs neurones, on forme ainsi un réseau de neurones constitué de trois principaux types de couches :

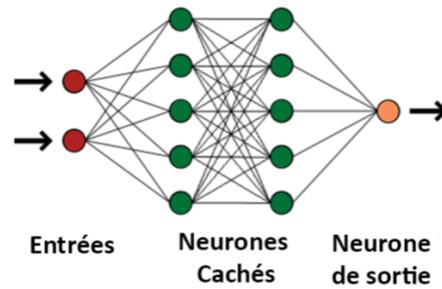


FIGURE 5.2 – Réseau de neurones artificiels

- **Une couche d'entrée** : Les neurones de cette couche dits d'entrée fournissent des informations du monde extérieur au réseau. Ils transmettent simplement les informations à la prochaine couche.
- **Une ou plusieurs couches cachées** : Les neurones de cette couche dits cachés n'ont aucune connexion directe avec le monde extérieur. Ils effectuent des calculs et transfèrent des informations à la couche suivante.
- **Une couche de sortie** : Les neurones de cette couche dits de sortie sont responsables des calculs et du transfert des informations du réseau vers le monde extérieur.

A partir de l'équation 5.1, on peut définir de manière générale un réseau de neurones comme suit. Soient :

- $r$  le nombre des neurones sur la couche d'entrée
- $s$  le nombre des neurones sur la couche de sortie
- $l$  le nombre total des couches du réseau de neurones
- $n_k$  le nombre des neurones sur la  $k^{\text{ème}}$  couche avec  $0 \leq k \leq l$  tel que  $n_0 = r$  et  $n_l = s$
- les fonctions affines  $A_i^k(x) = \sum_{j=1}^{n_{k-1}} w_{ij}^k x_j + b_j^k$  avec  $1 \leq k \leq l$ ,  $1 \leq i \leq n_k$  et  $x \in \mathbb{R}^{n_{k-1}}$
- $f$  la fonction d'activation du réseau de neurones
- le vecteur de sortie  $Y = (y_1, \dots, y_s)$  défini par la récurrence suivante :

$$\forall x \in \mathbb{R}^r, \begin{cases} a_0 = x \\ a_i^k = f(A_i^k(a^{k-1})) & \text{avec } 1 \leq k \leq l, 1 \leq i \leq n_k \\ y_i(x) = A_i^l(a^{l-1}) & \text{avec } 1 \leq i \leq s \end{cases} \quad (5.6)$$

[4]

### 5.4.3 Apprentissage d'un réseau de neurones

La phase d'apprentissage consiste à ajuster les poids d'un réseau de neurones afin d'obtenir les meilleurs résultats de régression ou de classification dans notre cas. L'objectif de cette optimisation est de minimiser la fonction de coût  $\psi$  (5.7) qui calcule l'erreur entre les données de la base d'apprentissage  $y$  et les données prédites par le réseau de neurones  $\hat{y}$ .

$$\psi(p) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2, \text{ avec } N \text{ le nombre de données d'apprentissage} \quad (5.7)$$

L'un des algorithmes d'optimisation proposés dans la littérature est l'algorithme de rétropropagation qui se base sur la **descente de gradient**. Cet algorithme permet d'approcher son minimum

local en cherchant les points auxquels son gradient est nul. Dans le cas des réseaux de neurones, cela signifie que, à chaque itération, la rétropropagation calcule la dérivée de la fonction de coût  $\psi$  par rapport à chaque poids et la soustrait de ce dernier selon l'équation :

$$p_{i+1} = p_i - \alpha \nabla_p \psi(p_i). \quad (5.8)$$

En pratique la base de données d'entraînement est important ( $N$  est très grand). A cet effet, on subdivise cette base de données en lots de petite taille appelés **batchs**. Ainsi l'équation 5.8 n'est pas appliquée d'un coup sur l'ensemble des données mais sur chaque lot successivement.

Le paramètre  $\alpha$  de l'équation 5.8 est le **pas de la descente** ou **learning rate**. C'est l'un des hyperparamètres importants pour l'optimisation des réseaux de neurones profonds en agissant sur sa convergence. En effet, un taux d'apprentissage trop élevé conduit à des mises à jour des poids importantes et la convergence devient instable. Par contre, pour un taux d'apprentissage faible, la convergence est ralentie avec une possibilité de tomber dans des minimas locaux. L'approche populaire utilisée dans l'apprentissage profond pour avoir le taux d'apprentissage optimal est de commencer l'apprentissage avec une valeur élevée afin d'accélérer la descente du gradient et de la réduire par la suite pour améliorer la précision.[4]

Plusieurs problèmes peuvent être rencontrés lors du processus d'entraînement d'un réseau de neurones. Parmi ceux-ci, nous pouvons citer :

- **Le surapprentissage** : dans ce cas, le réseau de neurones devient capable de capturer des informations qui ne sont pas utiles pour accomplir sa tâche ou il devient incapable de généraliser les caractéristiques des données, ce qui limite sa capacité de reconnaître de nouvelles données. Pour résoudre ce problème, on peut d'une part utiliser la technique **dropout** qui consiste éliminer certains neurones à chaque itération afin de diminuer le nombre de paramètres du réseau et de mettre à jour un nombre réduit de poids. D'autre part, une autre technique consiste à diversifier notre base de données en y ajoutant de nouvelles données.[4]
- **Le sous-apprentissage** : dans ce cas, le réseau de neurones est incapable de reconnaître même les données d'entraînement. Pour résoudre, on peut entraîner plus longtemps le modèle de réseau de neurones ou encore rendre le réseau de neurones plus complexes en y ajoutant plus de couches cachées.

Selon le type des données d'entraînement, on distingue des types de réseaux de neurones adaptés à savoir les **ANNs**, les **RNNs** et les **CNNs**. Pour un système ANPR qui traite essentiellement les images, les réseaux neurones convolutifs sont les plus adéquats.

#### 5.4.4 Réseaux de neurones convolutifs

Aussi appelés **CNN** ou ConvNets, les réseaux de neurones convolutifs sont constitués de plusieurs couches empilées. En 1989, LECUN et al. développent le premier réseau de neurones à convolution appelé LeNet [12]. Il servait à reconnaître les caractères comme les codes postaux et les chiffres. Pour pouvoir extraire et traiter les caractéristiques des données, les ConvNets se composent des couches suivantes :

1. **Couche de convolution** : Un réseau de neurones convolutif est un réseau de neurones qui utilise une opération mathématique qui s'appelle convolution ou produit de convolution. Il s'agit d'une opération linéaire. Chaque réseau de neurones convolutif contient au moins une couche de convolution. Soient  $f$  et  $g$  deux fonctions définies sur  $\mathbb{R}$ , le produit de convolution

entre  $f$  et  $g$  est généralement noté  $f * g$  et il est défini par l'équation suivante :

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(t)g(x - t) dt \quad (5.9)$$

Dans un réseau de neurones,  $f$  est assimilé à l'entrée et  $g$  au noyau de convolution. En apprentissage automatique, on utilise en réalité la convolution discrète avec l'entrée qui est toujours un tableau de données multidimensionnel, et le noyau est toujours un tableau de paramètres multidimensionnel. Dans ce cas, pour une entrée image  $I$  et pour un noyau  $K$ , la convolution discrète s'écrit :

$$(K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \quad (5.10)$$

Une couche de convolution est caractérisée par :

- **les dimensions des noyaux de convolution**, généralement une convolution à une dimension égale à 2 avec des noyaux carrés.
  - **le nombre des filtres de convolution  $C$** , c'est le nombre de cartes d'activations, ou cartes de caractéristiques, en sortie de la couche. Ces cartes sont représentées sous la forme de tenseurs de dimension 3 ( $H, W, C$ ) avec  $H$  la hauteur des cartes,  $W$  la largeur et  $C$  le nombre de canaux.
  - **le pas de convolution** -ou stride -  $s$ . C'est le pas de décalage du noyau de convolution à chaque calcul.
  - **le padding  $p$** . C'est le paramètre permettant de dépasser la taille de l'image pour appliquer la convolution en ajoutant des pixels autour de l'image.
2. **Couche d'échantillonnage (Pooling)** : Semblable à la couche de convolution, la couche d'échantillonnage est chargée de réduire la taille spatiale des cartes de caractéristiques, mais elle conserve les informations les plus importantes. Il existe différents types d'échantillonnage dont l'**échantillonnage maximum -ou Max Pooling-**, l'**échantillonnage moyen -ou Average Pooling**. Le Max Pooling renvoie la valeur maximale de la partie de l'image couverte par le noyau. Tandis que le Average Pooling prend la moyenne de tous les éléments couverts par le noyau. Si le Average Pooling ne permet juste qu'une réduction dimensionnelle, le Max Pooling par contre est un mécanisme de suppression de bruits. Ceci rend évidemment le Max Pooling plus performant et donc plus utilisé.
3. **Couche complètement connectée** : La couche complètement connectée est un réseau de neurones multi-couche traditionnel qui utilise une fonction d'activation (par exemple softmax) sur le vecteur de sortie afin d'ajouter la non-linéarité. La sortie des couches de convolution et d'échantillonnage représente des caractéristiques de haut niveau de l'image d'entrée. L'objectif de la couche complètement connectée est d'utiliser ces caractéristiques pour classer l'image d'entrée du réseau en différentes classes en fonction de la base de données d'apprentissage. Les couches entièrement connectées sont généralement suivies d'un dropout. Ce dernier agit sur les poids de ces couches afin de désactiver un certain nombre de neurones pour réduire le nombre de paramètres. Cela permet de contrôler le sur-apprentissage qui peut être causé par un nombre important de paramètres.[\[4\]](#)

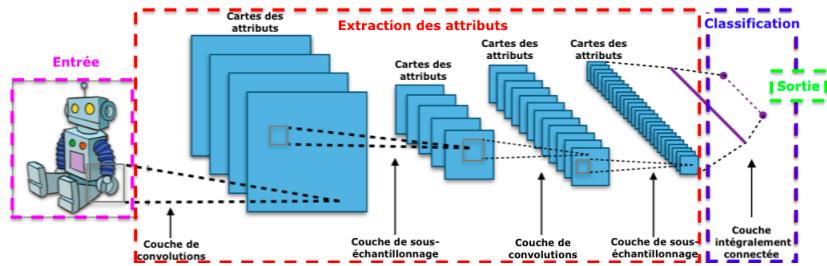


FIGURE 5.3 – Architecture d'un réseau de neurones convolutifs

Parmi les applications où les réseaux de neurones convolutifs sont utilisés avec succès, se trouvent la classification et la détection des objets dans une image. Si construire sa propre architecture CNN de classification des objets est une chose aisée, ce n'est pas le cas pour la détection des objets. En réalité, un réseau de neurones convolutifs dédié à la détection des objets est plus complexe dans son architecture qu'un CNN de classification. C'est la raison pour laquelle plusieurs chercheurs se sont penchés sur la question et ont proposé des algorithmes de création de modèles de détection des objets plus ou moins rapides et précis.

#### 5.4.5 Détection des objets

Contrairement à la classification, la détection des objets ne se limite pas à reconnaître les objets sur les images. Il faut aller plus loin en les localisant à travers des rectangles appelés *bounding boxes*. Pour y arriver, des spécialistes du domaine de la vision par ordinateur ont mis en place des algorithmes.

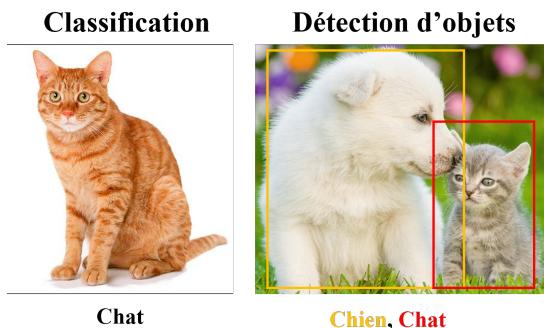


FIGURE 5.4 – Différence entre classification et détection d'objets

Mais avant de ces algorithmes, commençons par présenter quelques métriques d'évaluation utilisées dans ces algorithmes.

#### 5.4.5 Métriques d'évaluation

Les métriques d'évaluation utilisés régulièrement dans les algorithmes de détection d'objets sont :

- **La précision** : c'est le pourcentage de détections correctes. Il met en évidence l'exactitude des prédictions. Sa formule est :

$$prec = \frac{VP}{VP + FP} \quad (5.11)$$

- **Le rappel** : c'est un indicateur qui mesure la capacité du modèle à prédire l'ensemble des résultats attendus. Ainsi, un modèle peut avoir une très bonne précision mais avoir un mauvais rappel. En effet, un modèle peut-être exact lorsqu'il prédit la présence d'un objet, mais ne pas détecter des objets qui auraient dû l'être.

$$rap = \frac{VP}{VP + FN} \quad (5.12)$$

- VP : Vrais Positifs, nombre d'objets correctement détectés
- FP : Faux Positifs, nombre d'objets détectés mais non présents en réalité
- FN : Faux Négatifs, nombre d'objets non détectés mais présents en réalité
- **L'intersection sur l'union (IoU)** : il permet de comparer la région détectée "prédite" avec la région de la vérité terrain, de manière proportionnelle à la taille de l'objet recherché. Les régions comparées sont les boîtes englobantes : la comparaison des boîtes englobantes permettra de mesurer la qualité de la détection d'objets. Pour cela, on compare l'IoU à un seuil : on considère que l'objet est correctement détecté si et seulement si l'IoU est supérieur au seuil.

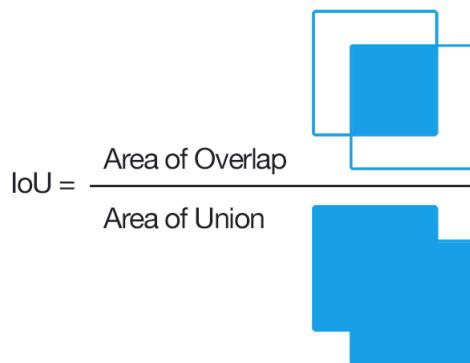


FIGURE 5.5 – Equation de l'IoU source : [pyimagesearch](#)

- **La précision moyenne (AP)** : A partir de la précision et du rappel, on peut tracer une courbe de précision en fonction du rappel. L'aire de cette courbe est définie comme la précision moyenne.
- **La moyenne de la précision moyenne (mAP)** : Si plusieurs valeurs sont choisies pour le seuil de l'IoU, pour chaque classe, l'AP moyenne est calculée sur l'ensemble de ces valeurs seuils et la moyenne des valeurs obtenues donne le mAP. [3]
- **Frame Per Second (FPS)** : Il représente le nombre d'images qui peuvent être traitées par seconde. Plus il est grand, plus le modèle de détection est rapide.

#### 5.4.5 Les algorithmes de détection d'objets

Les algorithmes de détection d'objets peuvent être divisés en deux grands groupes. D'une part, nous avons les **algorithmes de détection à deux étapes** : une étape de détection de régions possibles d'objets possibles en utilisant, une étape pour classer l'objet éventuel dans chaque région. D'autre part, nous retrouvons les **algorithmes de détection à une étape** qui fait la détection des objets par un seul passage dans un réseau de neurones. Dans le premier groupe, on a les algorithmes comme **R-CNN**, **Fast R-CNN**, **Faster R-CNN**. Dans le second groupe, on y trouve **YOLO**, **SSD**. Décrivons brièvement certains de ces algorithmes.

- **Region-based Convolutional Neural Networks (R-CNN)** : Il a été proposé par GIRSHICK et al. A partir d'un algorithme de recherche selective, on extrait sur l'image à traiter 2000 régions appélées régions de propositions. Chacune de ces régions est par la suite introduite dans un CNN pour en extraire les caractéristiques. Ces caractéristiques sont envoyées à un SVM qui classe la présence d'un objet sur la région candidate. En outre, l'algorithme prédit 4 valeurs dites de décalage pour la précision du cadre de délimitation.

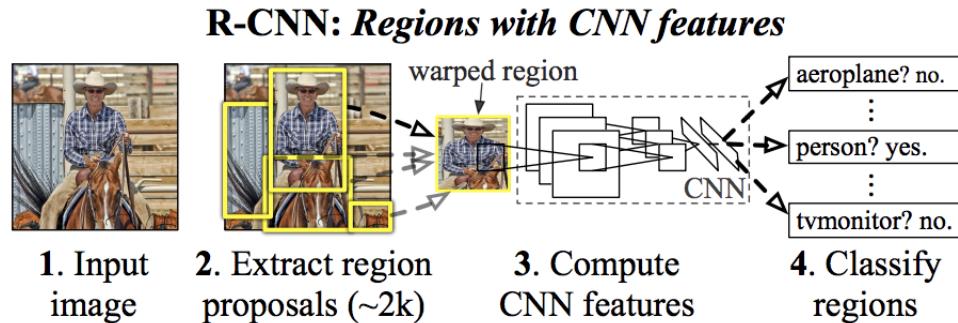


FIGURE 5.6 – R-CNN

- **Fast R-CNN** : Le même auteur de R-CNN a résolu certains des inconvénients de celui-ci en créant un algorithme de détection d'objets plus rapide appelé Fast R-CNN. L'approche est similaire à l'algorithme R-CNN. Mais, au lieu de fournir les propositions de région au CNN, nous transmettons l'image d'entrée au CNN pour générer une carte de caractéristiques convolutives. À partir de la carte des caractéristiques convolutives, nous identifions la région des propositions et les déformons en carrés et en utilisant une couche de regroupement de ROI, nous les remodelons en une taille fixe afin qu'elle puisse être introduite dans une couche entièrement connectée. À partir du vecteur de caractéristiques ROI, nous utilisons une couche softmax pour prédire la classe de la région proposée ainsi que les valeurs de décalage pour la boîte englobante.

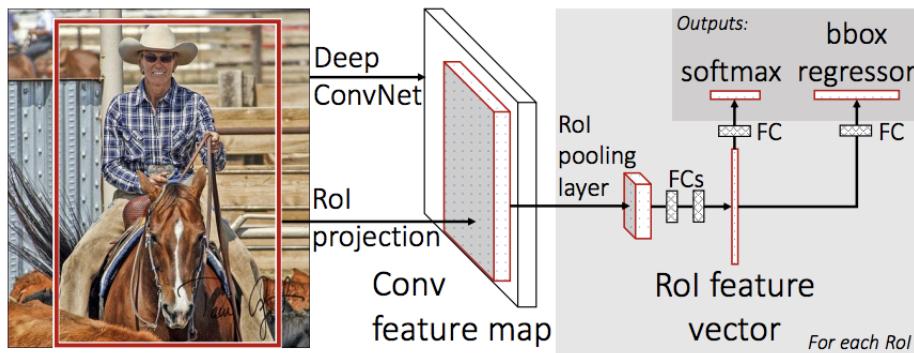


FIGURE 5.7 – Fast R-CNN

- **Faster-RCNN** : Les deux algorithmes ci-dessus (R-CNN et Fast R-CNN) utilisent une recherche sélective pour trouver les propositions de région. La recherche sélective est un processus lent et fastidieux qui affecte les performances du réseau. A la place de cette

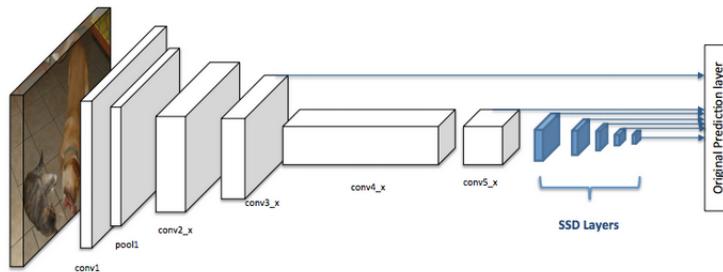


FIGURE 5.9 – Architecture du SSD

recherche, REN et al. ont proposé un réseau de neurones convolutifs appelé [Region Proposal Network \(RPN\)](#) qui prédit les propositions de région. Le reste des étapes est le même que celui d'un Fast-RCNN décrit plus haut.

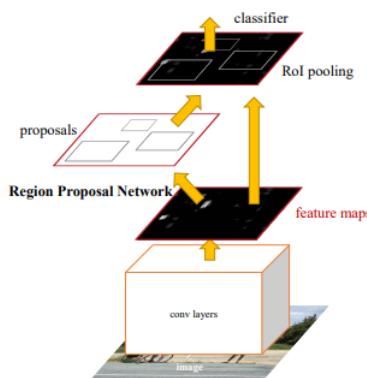


FIGURE 5.8 – Faster R-CNN

- [Single Shot MultiBox Detector \(SSD\)](#) : En fin novembre 2016, LIU et al. publient un article portant sur un nouvel algorithme de détection à une seule étape qu'ils choisissent d'appeler **SSD**. Cet algorithme permet de créer des modèles qui atteignent des précisions record par rapport à Faster R-CNN : 74.3% mAP à 59 FPS [13] sur les jeux de données populaires comme Pascal VOC. L'architecture SSD est composée de deux parties : un **modèle de base** et une **tête SSD**. Le modèle de base est généralement un réseau de classification d'images pré-entraîné pour extraire les caractéristiques. La tête SSD est constituée d'une ou plusieurs couches convolutives ajoutées à la première partie et les sorties sont interprétées comme les cadres de délimitation et les classes d'objets. Dans la figure ci-dessous le modèle de base est représenté par les boîtes blanches tandis que la tête SSD est représentée par les boîtes bleues.
- [You Only Look Once \(YOLO\)](#) : En 2016, REDMON et al. proposent une nouvelle approche qui permet de détecter les objets en prenant l'image en entier et en une seule instance. Ce qui justifie le nom donné à l'algorithme You Only Look Once qui signifie littéralement On ne regarde qu'une seule fois. Contrairement à Faster R-CNN, YOLO utilise un seul réseau convolutif pour prédire en même temps les boîtes englobantes (bounding boxes) et les probabilités de classe pour ces boîtes. De manière concrète et simple, on peut résumer le fonctionnement de YOLO comme suit :

1. YOLO prend en entrée une image

2. Il divise l'image en une grille SXS
3. La classification et la localisation des images sont effectuées sur chaque grille. Par après, YOLO prédit les cadres de délimitation et leurs probabilités de classe correspondantes pour les objets.

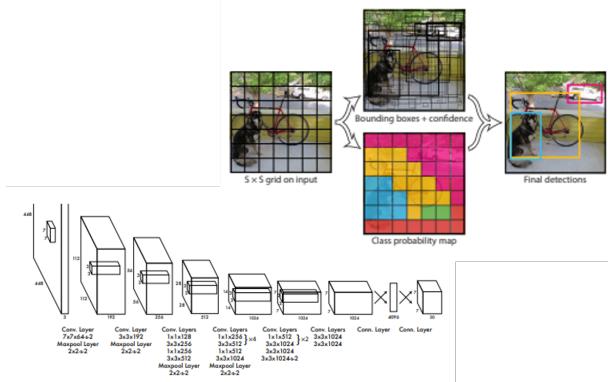


FIGURE 5.10 – Architecture de la première de version de YOLO

L'un des problèmes qui peut arriver avec YOLO comme la plupart des algorithmes de détection des objets est qu'un même objet est détecté plus d'une fois. Pour résoudre ce problème, on fait appel à l'algorithme NMS (Non Max Suppression) qui supprime les doublons de détections du même objet. L'algorithme est décrit

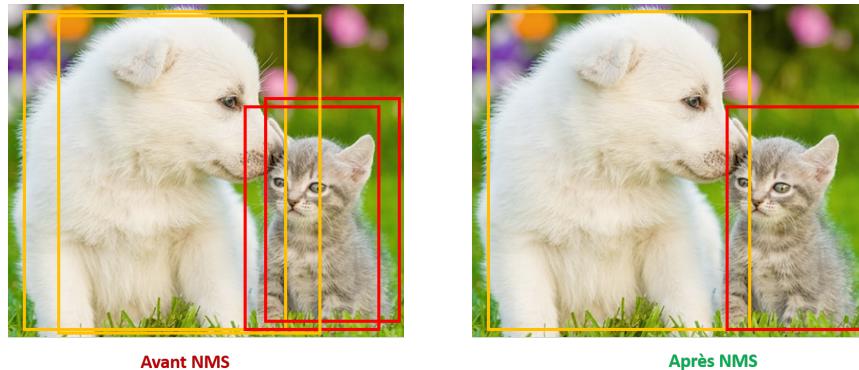


FIGURE 5.11 – NMS

**Algorithm 1** Non Max Suppression

---

```

1: function NMS( $\mathcal{B}, \mathcal{S}, \mathcal{N}_s$ )            $\triangleright \mathcal{B}$  :cadres initiaux,  $\mathcal{S}$  :scores initiaux,  $\mathcal{N}_s$  : seuil de NMS
2:    $\mathcal{B}_{nms} \leftarrow \emptyset$ 
3:   while  $\mathcal{B} \neq \emptyset$  do
4:      $m \leftarrow \text{findMax}(\mathcal{S})$             $\triangleright$  Recherche de l'indice du cadre de plus haut score
5:      $b_{max} \leftarrow \mathcal{B}[m]$ 
6:     for  $b_i \in \mathcal{B}$  do
7:       if  $iou(b_{max}, b_i) \geq \mathcal{N}_s$  then     $\triangleright$  On élimine tout cadre très proche du cadre actuel
8:          $\mathcal{B} \leftarrow \mathcal{B} - b_i$ 
9:          $\mathcal{S} \leftarrow \mathcal{S} - s_i$ 
10:      end if
11:    end for
12:     $\mathcal{B}_{nms} \leftarrow \mathcal{B}_{nms} \cup b_{max}$ 
13:   end while
14:   return  $\mathcal{B}_{nms}, \mathcal{S}$ 
15: end function

```

---

Après la première version de YOLO, d'autres nouvelles versions ont été développées pour augmenter les performances des modèles de détection d'objets. Ci-dessous se trouve le tableau comparatif des différentes versions de YOLO fait dans le cadre d'une étude réalisée par des chercheurs tunisiens sur la détection des plaques de véhicules.

YOLO	v2	v3	v3-tiny	v3-SPP	v4	v4-tiny
Anchors	5	9	6	9	9	6
Backbone	VGG-16	Darknet-53	Darknet-19	Darknet-53	CSPDarknet-53	CSPNet-15
FPN	Absent	Présent	Présent	Présent	Présent	Présent
SPP	Absent	Absent	Absent	Présent	Présent	Présent
PAN	Absent	Absent	Absent	Absent	Présent	Présent
mAP@0.5(%)	44.0	55.3	33.1	60.6	62.8	40.2
Vitesse (FPS)	40	66	345	38	55	330
Taille (MB)	275	236	33.7	240	245	23.1

TABLE 5.1 – Comparaison entre les différentes versions de YOLO

## 5.5 Conclusion

Dans ce chapitre, nous avons exploré l'apprentissage automatique qui est la tendance technologique utilisée dans les systèmes ANPR modernes. Pour cela, nous avons d'abord commencé par citer les 5 phases d'un cycle de Machine Learning. Par la suite, nous avons vu les 3 types principaux d'apprentissage automatique qui se diffèrent par le type des données utilisées pour faire l'entraînement. Enfin, dans la dernière section de ce chapitre, nous avons détaillé une sous-branche de l'apprentissage automatique qui a révolutionné l'intelligence artificielle ces dernières années : le Deep Learning. Dans cette section, nous avons décrit les réseaux de neurones qui sont la base du Deep Learning. Ces réseaux de neurones permettent de concevoir des modèles de détection d'objets grâce à des algorithmes soit à deux étapes soit à une étape. Si d'une part les algorithmes à une étape sont plus rapides, les algorithmes à deux étapes sont plus précis. Toutefois, en faisant un bon

compromis entre la vitesse et la précision, les premiers sont plus performants que les seconds. Dans ce sens, YOLO s'est fortement démarqué pour les applications en temps réel comme les systèmes ANPR. C'est l'une des raisons pour laquelle nous l'avons adopté lors de la conception de notre système de reconnaissance automatique des plaques de véhicules.

# Chapitre 6

## Conception et mise en oeuvre

### 6.1 Introduction

En nous basant sur les travaux des chercheurs dans le domaine informatique, nous avons fait dans les chapitres précédents des études théoriques des axes principaux de notre sujet sur les systèmes ANPR. Ces études nous ont permis de concevoir et mettre en place aussi notre système de reconnaissance automatique des plaques d'immatriculation. Ce chapitre est donc pour mettre en lumière ce système. Pour cela, nous commencerons par présenter l'architecture simple que nous avons opté pour. Par la suite, nous détaillerons chaque partie de cette architecture. Il est bien de préciser que nous parlons ici de l'architecture logicielle de notre système et non pas du côté matériel.

### 6.2 Architecture

Le système ANPR que nous avons réalisé est essentiellement composé de deux grandes parties :

1. **Une partie de détection de la plaque** : Dans cette partie, on veut répondre à la question suivante : où se trouve la plaque sur l'image ? En effet, avant de reconnaître le numéro d'immatriculation, il est important de localiser premièrement la position de la plaque sur l'image reçue à travers la caméra. À partir de cette localisation, on peut isoler la plaque du reste de l'image pour l'envoyer à la partie suivante.
2. **Une partie de lecture des caractères** : L'entrée de cette partie est la sortie de la partie précédente. L'objectif de cette partie est de reconnaître l'ensemble des caractères sur l'image de la plaque. Cette partie du système retourne alors le numéro du matricule présent sur l'image en format textuel ou ASCII enregistrable dans une base de données.

La figure suivante illustre l'architecture que nous avons choisie pour notre système ANPR.



FIGURE 6.1 – Architecture du système MoPlaZer

### 6.3 Détection de la plaque

Pour détecter la plaque sur une image, nous avons choisi l'approche par apprentissage automatique. À cet effet, nous avons entraîné un modèle Deep Learning de détection d'objets en utilisant l'architecture YOLOv4 Tiny. Le choix de cette architecture s'explique par ses performances qui sont clairement définies dans le tableau 5.1. D'autre part, cette architecture est la plus adaptée pour les appareils mobiles et systèmes embarqués où nous allons déployer le modèle de détection. Décrivons maintenant chaque étape que nous avons suivie pour avoir le modèle.

### 6.3.1 Acquisition des données

L'entraînement d'un modèle de Deep Learning nécessite une masse importante de données. Pour le détection des plaques, nous avons collecté les images de véhicules sur la plateforme **Open Images de Google**. Open Images est un ensemble de données d'environ 9 millions d'images annotées avec des étiquettes au niveau des images, des cadres de délimitation d'objets, des masques de segmentation d'objets, des relations visuelles et des récits localisés [9]. Sur cette plateforme, nous avons pu avoir **5890 images avec les annotations sous format YOLO d'une taille d'environ 2Go**. Les annotations sont des fichiers texte contenant les coordonnées des cadres de délimitation des plaques.



FIGURE 6.2 – Exemple d’images collectées sur Open Images

Après une analyse des données collectées, voici quelques unes de ses caractéristiques :

Caractéristiques	Valeur
Largeur maximale des images	<b>2672px</b>
Hauteur maximale des images	<b>4000px</b>
Largeur minimale des images	<b>141px</b>
Hauteur minimale des images	<b>225px</b>
Nombre maximal d’annotations sur une image	<b>26</b>
Moyenne du nombre d’annotations par image	<b>1.43</b>
Nombres d’images dupliquées	<b>134</b>

TABLE 6.1 – Analyse de la collection des données de véhicules

### 6.3.2 Nettoyage et préparation

Avant d’utiliser les données collectées, il est important de les vérifier et corriger les anomalies afin d’éviter des mauvaises performances du modèle. A l’aide de l’outil d’annotations **LabelImg**, nous avons vérifié si toutes les images que nous avions chargées étaient bien annotées. Dans certains cas rares, il fallait recadrer le rectangle délimitant la plaque d’immatriculation. D’autre part, nous avons remarqué comme l’indique le tableau précédent que 134 images ont été dupliquées. A l’aide d’un script écrit en Python, nous avons supprimé toutes ces duplications. Nous obtenons finalement une base de données de 5756 images distinctes et correctement annotées. Par ailleurs, grâce à la plateforme en ligne Roboflow, nous avons effectué quelques opérations de prétraitement sur l’ensemble des images. Ces opérations permettent d’une part de réduire le temps d’entraînement du modèle et d’autre part d’améliorer la précision du modèle. Parmi ces opérations, on retrouve l’**auto-orientation** pour standardiser l’ordre des pixels et le **redimensionnement à 416 x 416** (taille d’entrée du modèle).

### 6.3.3 Entraînement du modèle

Pour l’évaluation de la performance du modèle au cours de l’entraînement, nous avons subdivisé notre base de données en deux grands groupes : 80% soit 4604 images pour l’entraînement et 20% pour le test. L’entraînement du modèle a été fait sur la plateforme en ligne Google Colab. Elle offre des ressources(puissance de GPU) gratuitement pour entraîner des modèles de Deep Learning. Voici quelques configurations prises en compte :

- **Type d’exécution : GPU**

- Utilisation de CUDA Version 11.2
- GPU : Tesla T4
- Architecture du modèle : YOLO V4
- Utilisation de Tensorflow version 2.3.0rc0
- Utilisation du [répertoire darknet de Roboflow](#)
- Utilisation de fichier de configuration [yolov4-tiny\\_3l.cfg](#) : Toutefois nous avons modifié certains paramètres dans ce fichier.

Paramètres	Nouvelle valeur
batch	64
subdivisions	32
width	416
height	416
max_batches	6000
steps	4800, 5400
classes	1
filters (couche de convolution avant couche yolo)	18

TABLE 6.2 – Quelques paramètres du fichier de configuration YOLO

### — Utilisation des poids pré-entraînés YOLO : [yolov4-tiny.conv.29](#)

L’entraînement a pratiquement pris 1.5 heures. Voici une courbe montrant l’évolution de la précision du modèle au cours de l’entraînement :

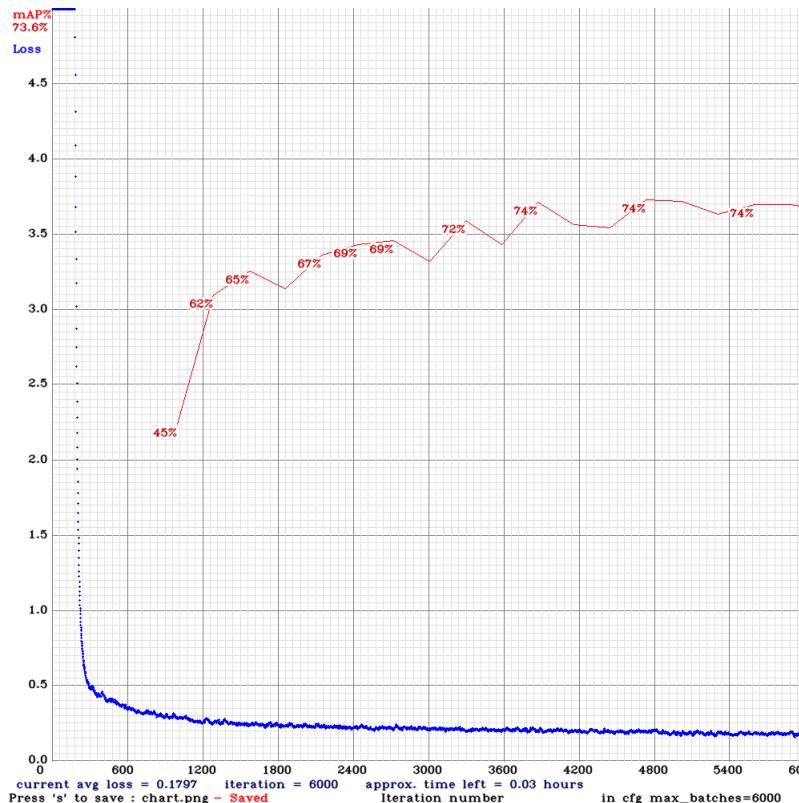


FIGURE 6.3 – Évolution de la métrique mAP lors de l’entraînement du modèle de détection de plaques.

À la fin de l'entraînement, le modèle est un fichier binaire extension **.weights**. Ce fichier contient les poids du réseaux de neurones. Par ailleurs, nous constatons à partir de la courbe précédente que la précision maximale qu'atteint le modèle au cours de son entraînement est de **74%**.

#### 6.3.4 Résultats et évaluation

Nous avons évalué notre modèle de détection des plaques sur deux types de données : les images contenant des véhicules au Maroc pour évaluer la précision du modèle et les vidéos de voitures en circulation pour mesurer la rapidité du modèle.

Pour le premier type, nous avons collecté des images sur les plateformes comme Kaggle, MSDA Datasets de UM6P et d'autres prises dans la rue. Après avoir réuni ces différentes sources de données, nous avons obtenu une base de données de 2229 images. On retrouve en moyenne une plaque par image. Le tableau d'évaluation est décrit comme suit :

Evaluation	Valeur
Nombre de plaques correctement détectées (VP)	<b>1994</b>
Nombre de plaques non détectées (FN)	<b>386</b>
Nombre de fausses détections (FP)	<b>6</b>
Précision	<b>99,7%</b>
Rappel	<b>83,78%</b>

TABLE 6.3 – Evaluation du modèle de détection des plaques marocaines

Puisque la précision et le rappel du modèle sont élevés, on peut dire que le modèle est performant en terme de précision de détection. La figure suivante traduit des exemples de détection faite par le modèle qui a été entraîné.



FIGURE 6.4 – Exemples de détection de plaques marocaines

Pour évaluer la rapidité de notre modèle, nous avons lancé le modèle sur des vidéos des véhicules en circulation. Puisque cette rapidité dépend à partie des performances matérielles de la machine sur laquelle le modèle est exécuté, voici ces caractéristiques :

Caractéristiques	Valeur
Processeur	<b>Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz</b>
RAM	<b>8.00 Go</b>
Disque dur SSD	<b>237 Go</b>
Type de système	<b>Système d'exploitation 64 bits, processeur x64</b>

TABLE 6.4 – Caractéristiques du PC

Avec un script Python en utilisant la librairie de traitement d’images OpenCV, nous avons constaté qu’avec les performances ci-dessus, notre modèle traite environ **14 à 18 images par secondes** en d’autres termes **14-18 FPS**.

## 6.4 Lecture du numéro d’immatriculation

Pour la lecture des caractères présents sur la plaque détectée, nous avons utilisé l’approche proposé par certains chercheurs de l’UM6P [1]. Cette approche consiste à utiliser un modèle pour détecter les caractères et classifier en même temps. Pour l’entraînement du modèle, ils sont appel à l’architecture YOLOv3. La précision obtenue sur les données d’entraînement est de **94,42%** et sur les données de test **82,38%**. Puisque ces résultats sont satisfaisants, nous avons adopté la même approche mais avec une version plus récente de YOLO : version 4. L’avantage avec cette approche par l’approche classique de l’OCR est que nous n’avons pas vraiment besoin d’une phase de prétraitement de l’image au préalable. Ceci participe donc à la rapidité de l’ensemble de système ANPR. Décrivons maintenant chaque étape suivie pour obtenir notre modèle.

### 6.4.1 Acquisition des données

Cette phase est très important car détermine la qualité du modèle. Une mauvaise qualité de données conduit inéluctablement à une mauvais modèle. Nous avons donc pris soin de collecter une importante masse diversifiée d’images de plaques marocaines. Pour y parvenir, nous avons pris les images contenant des véhicules au Maroc utilisées lors de la précédente phase. Grâce à un script Python et avec le modèle de détection de plaques conçu précédemment nous avons extrait les images des plaques. Nous avons donc pu réunir **3570 images**. A l’aide de l’outil d’annotation LabelImg, nous avons annoté ces images suivant **16 classes (0, 1, 2, 4, 5, 6, 7, 8, 9, a, b, waw, d, h, w)**.

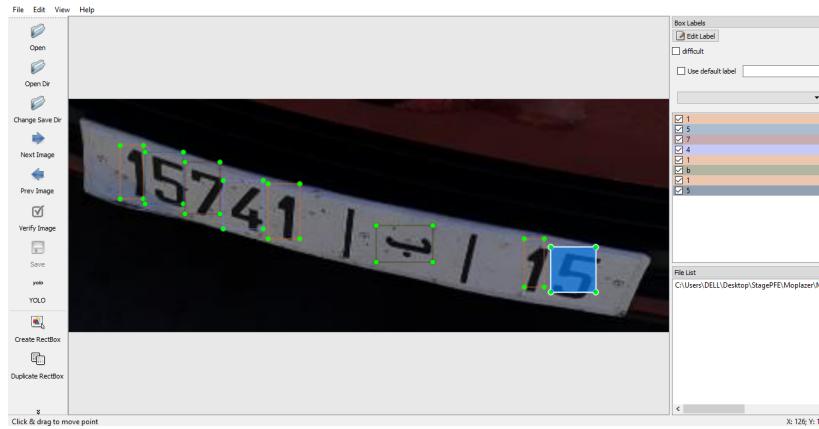


FIGURE 6.5 – Exemple d’annotation de plaque sur LabelImg

#### 6.4.2 Nettoyage et préparation

Pour cette phase, nous avons dans un premier temps effectuer quelques prétraitements sur les images collectées (redimensionnement et auto-orientation). Dans un second, pour augmenter le volume des données, nous avons créé de nouvelles données à partir des 3570 originales. Cette augmentation a été faite sur la plateforme Roboflow en utilisant plusieurs procédés comme la **saturation**, **modification de la luminosité**, **l’ajout des bruits et de flous**. Ces procédés nous ont permis d’avoir au total **8989 images annotées de plaques marocaines**. Nous avons analysé les proportions de chaque classes dans notre base de 8989 images et le graphe suivant les illustre.

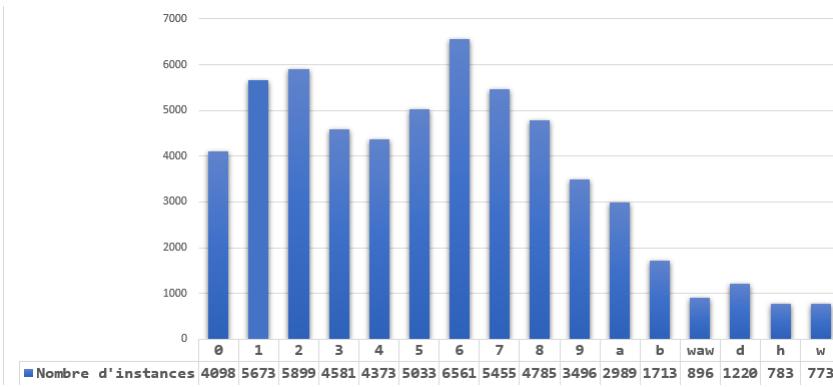


FIGURE 6.6 – Proportions des classes

#### 6.4.3 Entraînement du modèle

Avant l’entraînement proprement dit, nous avons subdivisé nos deux données en deux : 90% pour l’entraînement et 10% pour le test. Nous avons par la suite effectué l’entraînement sur Google Colab avec les même paramètres que précédemment sauf quelques modifications :

- Utilisation du [repertoire darknet d’AlexeyAB](#)
- Utilisation de fichier de configuration [yolov4-tiny\\_3l.cfg](#) : Toutefois nous avons modifié certains paramètres dans ce fichier.

Paramètres	Nouvelle valeur
batch	64
subdivisions	32
width	416
height	416
max_batches	10000
steps	8000, 9000
classes	16
filters (couche de convolution avant couche yolo)	63

TABLE 6.5 – Quelques paramètres du fichier de configuration YOLO pour le modèle OCR

Le graphe suivant montre l'évolution de la performance du modèle au cours de l'entraînement : la précision maximale atteint par le modèle est de **95,1%**.

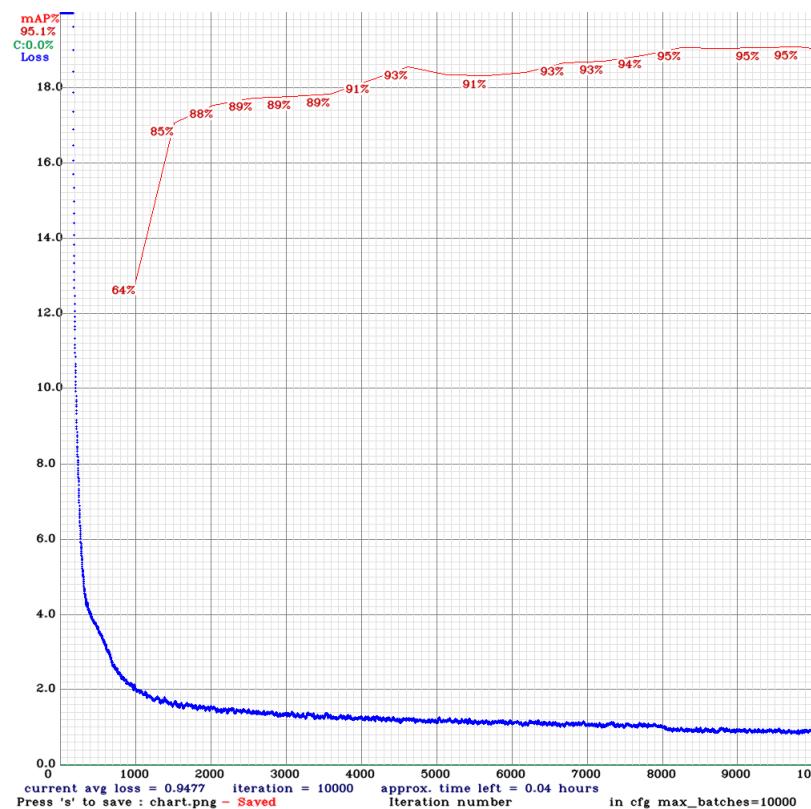


FIGURE 6.7 – Évolution de la performance du modèle

#### 6.4.4 Résultats et évaluation

Pour évaluer notre modèle de lecture du matricule, nous avons collecté des images contenant des véhicules. Dans un premier temps, nous avons utilisé le modèle de détection de plaques pour localiser les plaques sur les images. Dans un second temps, nous avons utilisé le modèle courant pour détecter les caractères sur les plaques. Le tableau suivant les résultats de la performance du modèle.

Evaluation	Valeur
Nombre de caractères correctement détectés (VP)	<b>302</b>
Nombre de caractères non détectés (FN)	<b>11</b>
Nombre de fausses détections (FP)	<b>12</b>
Précision	<b>96,18%</b>
Rappel	<b>96,48%</b>

TABLE 6.6 – Evaluation du modèle de lecture du matricule

Comme pour le modèle de détection de plaques, la précision et le rappel du modèle de lecture des matricules sont très élevés. Ceci nous permet de dire que notre système est globalement très précis. Voici quelques exemples de lecture des plaques d'immatriculation.



FIGURE 6.8 – Exemples de lecture des matricules des véhicules

## 6.5 Conclusion

Nous arrivons au terme d'un des chapitres les plus importants de notre document. Il nous a permis de présenter en détail notre système de reconnaissance des plaques d'immatriculation marocaines. Nous pouvons retenir que notre système ANPR que nous avons choisi d'appeler MoPlaZer est constitué de deux parties essentielles. La première partie est consacrée à la détection de la plaque d'immatriculation sur une image. Nous avons vu que pour arriver à détecter les plaques, nous avons entraîné un modèle de détection d'objets avec YOLOv4. La précision de ce modèle est de 99,7%. La seconde partie de notre système est dédiée à la lecture du matricule : extraire en format textuel le numéro d'immatriculation. Pour y arriver, nous avons aussi entraîné un modèle de détection d'objets avec YOLOv4. Ce modèle atteint une précision de plus de 96%. Nous pouvons donc constater que notre système MoPlaZer est globalement précis. Après avoir conçu et réalisé le système, il faut l'intégrer dans des cas d'utilisation particuliers. C'est l'objet du prochain et dernier chapitre.

# Chapitre 7

# Applications

## 7.1 Introduction

Nous entrons dans le dernier chapitre de ce rapport. Nous l'avons dédié aux différentes applications dans lesquelles nous avons intégré notre système de reconnaissance de plaques d'immatriculation marocaines. S'il est vrai qu'il existe plusieurs diverses cas d'utilisation pour les systèmes ANPR, nous sommes focalisés sur deux. La première est une application mobile qui utilise notre système MoPlaZer pour lire les plaques marocaines sur les images ou des vidéos en temps réel. La seconde est un parking intelligent dans lequel le système MoPlaZer servira à lire les plaques d'immatriculation pour donner accès ou non aux places de stationnement du parking. Ces deux cas d'utilisation feront l'objet de nos prochaines lignes.

## 7.2 Développement de l'application mobile

Notre mission est de réaliser une simple application sous Android qui utilise le système MoPlaZer pour reconnaître les matricules des véhicules au Maroc sur des images ou des vidéos en temps réel. Pour réaliser une application de qualité, il est important de passer par deux étapes primordiales : l'analyse des besoins et la conception.

### 7.2.1 Analyse et spécifications des besoins

Dans cette partie, nous allons tout simplement identifier les acteurs et leurs actions. Un acteur est une personne, une machine ou un autre système qui interagit avec le système étudié. Leurs différentes interactions avec le système constituent ce qu'on appelle cas d'utilisation.

Pour ce qui concerne notre application, nous avons un unique acteur qui est l'**utilisateur**. Il peut accéder à l'application pour :

- **Capturer une image** : L'utilisateur peut lancer la caméra et prendre une capture d'une photo. Cette photo est ensuite automatiquement traitée par le système pour faire la reconnaissance des plaques d'immatriculation éventuellement détectées sur la photo. Le système renvoie sur l'écran de l'application le résultat en image.
- **Lancer le traitement en temps réel** : L'utilisateur peut encore lancer la caméra. Mais cette fois-ci, le système traite en temps réel les images reçues par la caméra. Il trace sur ces images les rectangles sur les plaques éventuelles sans oublier les numéros de matricule sur les rectangles.

Pour représenter graphiquement ces fonctionnalités, nous allons utiliser ce qu'on appelle en langage UML le diagramme de cas d'utilisation. C'est un simple diagramme qui permet de décrire l'ensemble des opérations réalisables par un acteur. Ainsi notre diagramme de cas d'utilisation peut être représenté comme suit :

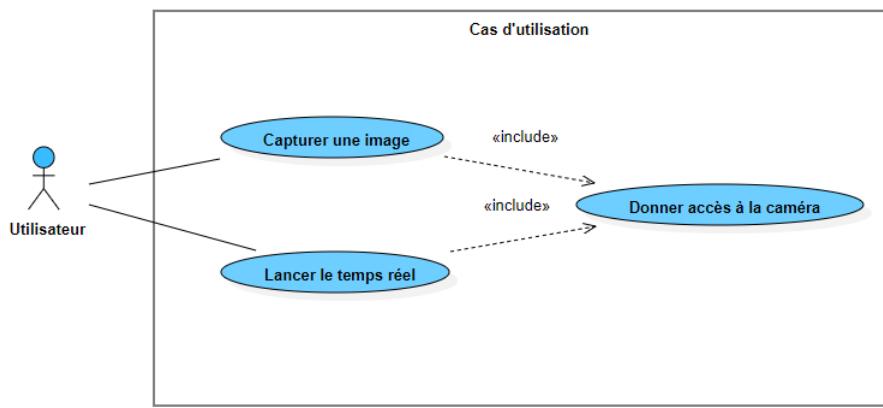


FIGURE 7.1 – Diagramme de cas d'utilisation

Ces fonctionnalités citées représentent les besoins fonctionnelles de notre application. Hors mis ce type de besoins, nous avons aussi les besoins non-fonctionnelles qui sont un ensemble des contraintes à respecter par l'application. Dans notre cas, on a :

- **L'ergonomie des interfaces** : Les interfaces de l'application doivent être simples et conviviales pour l'utilisateur. On doit aussi limiter au maximum les encombrements.
- **La performance** : Il s'agit principalement de la précision et la rapidité de la reconnaissance des plaques. En effet, notre application doit détecter le plus rapidement possible les plaques et identifier le numéro de plaque avec une grande précision.

### 7.2.2 Conception

A partir d'une bonne analyse des besoins, on réalise la conception qui est l'une sinon la plus importante étape entrant dans le processus de développement d'une application. La conception se matérialise généralement par des diagrammes UML tels que les **diagrammes de classes**, les **diagrammes de séquence** et les **diagrammes d'activités**. Un diagramme de classes permet de représenter les classes et les interfaces d'un système ainsi que les relations entre elles. Contrairement à un diagramme de classes qui est statique, un diagramme de séquence est dynamique et traduit les interactions entre acteurs et/ou le système pour un cas d'utilisation déterminé. Le déroulement d'un cas d'utilisation ou une méthode est présentée en utilisant un diagramme d'activités.

Pour mieux structurer notre application et faciliter ainsi les modifications et les maintenances, nous avons adopté pour une **architecture 3-tiers** (Figure 7.3) c'est-à-dire composée de trois couches principales qui communiquent entre elles en offrant les services les unes aux autres :

- La **couche métier** : Elle implémente la logique fonctionnelle de notre application. Elle est composée de plusieurs packages à savoir :
- Le package ***entities*** : qui contient les classes décrivant l'élément principal traité par notre application : la plaque d'immatriculation marocaine. Puisqu'il en existe plusieurs types, nous avons eu besoin de créer une classe abstraite qui prend en compte les caractéristiques et actions communes.

- Le package ***utils*** : qui contient des classes permettant de faire certains traitements sur les images ou manipulation entrant dans le cadre du *multithreading*.
- Le package ***detector*** : qui contient des classes permettant de localiser les plaques sur les images. Dans ce package, nous avons utilisé le **Design Pattern Builder** pour rendre plus flexible la création des détecteurs de plaques. Ceci pourra être utilisé si on veut par exemple utiliser une autre stratégie que la détection des objets avec YOLO.
- Le package ***ocr*** : qui contient les classes servant à extraire les matricules sur les plaques localisées. Pour les besoins d'évolution et de modification ultérieure, nous avons fait appel à un couplage faible en utilisant les interfaces à la place des classes concrètes.
- Le package ***analysers*** : qui contient une classe qui regroupe le processus de détection et de lecture des plaques d'immatriculation. Ici on utilise le principe d'**injection de dépendances par constructeur** pour initialiser le détecteur de plaques et le lecteur OCR.
- La **couche persistance** : Elle gère l'accès aux données et principalement la communication avec la base de données. Elle a été implémentée à travers un package appelé ***dao***. Nous y avons utilisé le **Design Pattern Singleton** pour assurer la création d'une seule instance de l'objet communiquant avec la base de données.
- La **couche présentation** : C'est la couche visible par l'utilisateur. Elle est composée des différentes interfaces graphiques de notre application. Nous l'avons regroupée dans un seul package appelé ***ui***. Ce package contient deux classes. Une pour décrivant la page principale et une autre décrivant la page secondaire où se réalise la reconnaissance des plaques en temps réel.

Cette architecture est illustrée à travers le diagramme de classes dans la figure 7.3. La figure 7.2 montre comment l'utilisateur interagit avec le système à l'aide d'un diagramme de séquence. La figure 7.4 est un diagramme d'activités représentant le déroulement de la reconnaissance des plaques en temps réel par le système.

### 7.2.3 Réalisation

Le développement de notre application mobile Android a été effectué en utilisant plusieurs outils matériels et logiciels. Sur le côté matériel, nous avons utilisé un PC dont les caractéristiques sont les suivantes :

- Système d'exploitation 64 bits Windows 10 ;
- Un disque dur SSD 237 Go ;
- Une mémoire RAM de 8 Go ;
- Un processeur Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz.

Les outils logiciels utilisés pour la réalisation de l'application sont :

- **Android Studio** : c'est un puissant environnement de développement (IDE) des applications Android créé par les ingénieurs de JetBrains et de Google. Tout le développement de notre application s'est fait sur cette plateforme.
- **Kotlin/Java** : pour développer les applications Android, on peut utiliser soit le langage de programmation Java soit le langage Kotlin ou même les deux à la fois. Presque similaires, ces deux langages sont interopérables : on peut appeler les méthodes d'une classe Java dans une classe Kotlin et vice versa. En ce qui nous concerne, nous avons choisi le langage Kotlin comme notre langage principal de développement d'une part parce que c'est le langage officiel conseillé par Google pour le développement des applications sous Android. D'autre part, nous avons préféré Kotlin à Java pour la simplicité de sa syntaxe qui rend le code moins

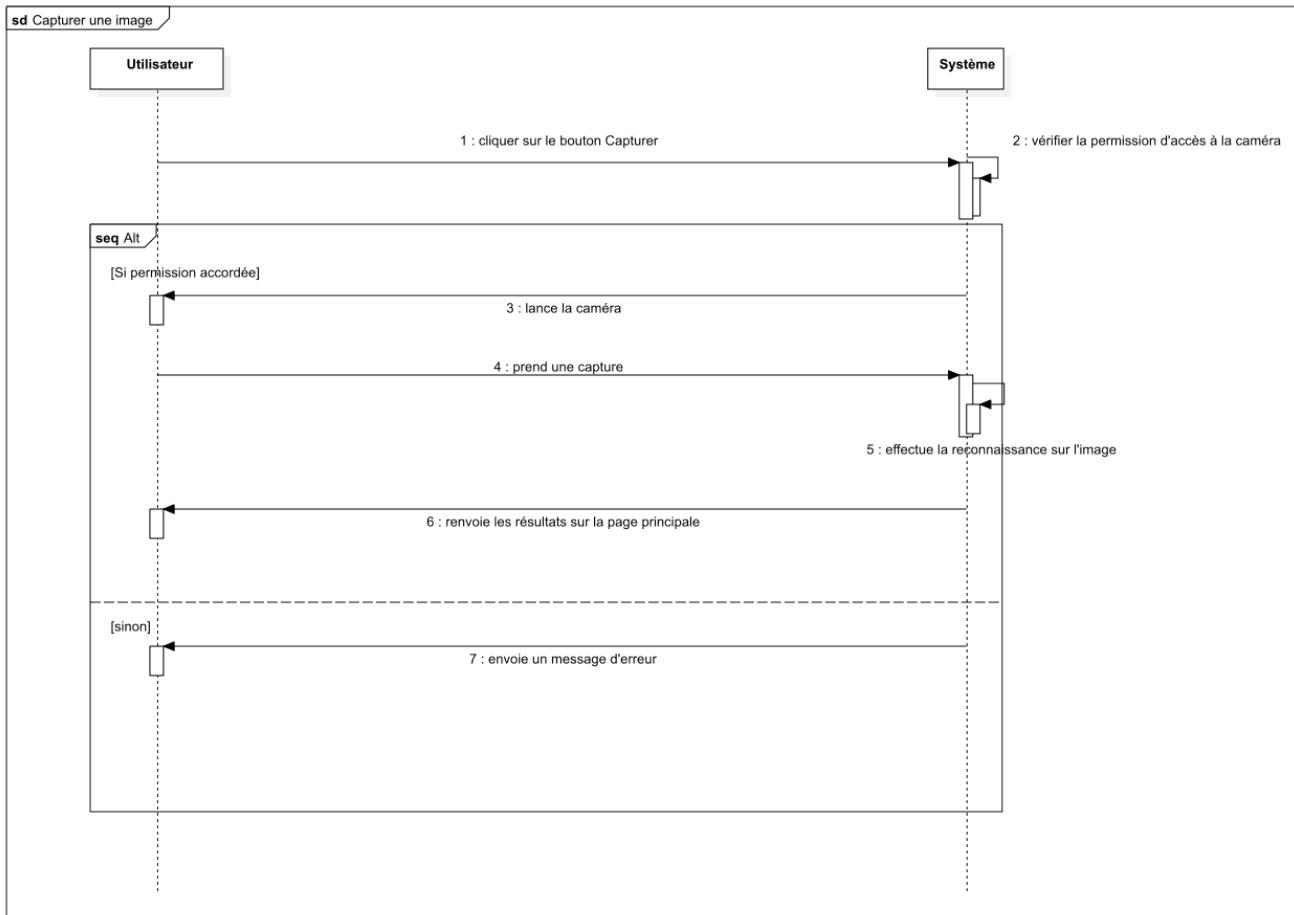


FIGURE 7.2 – Diagramme de séquence pour le cas d'utilisation "Capturer une image"

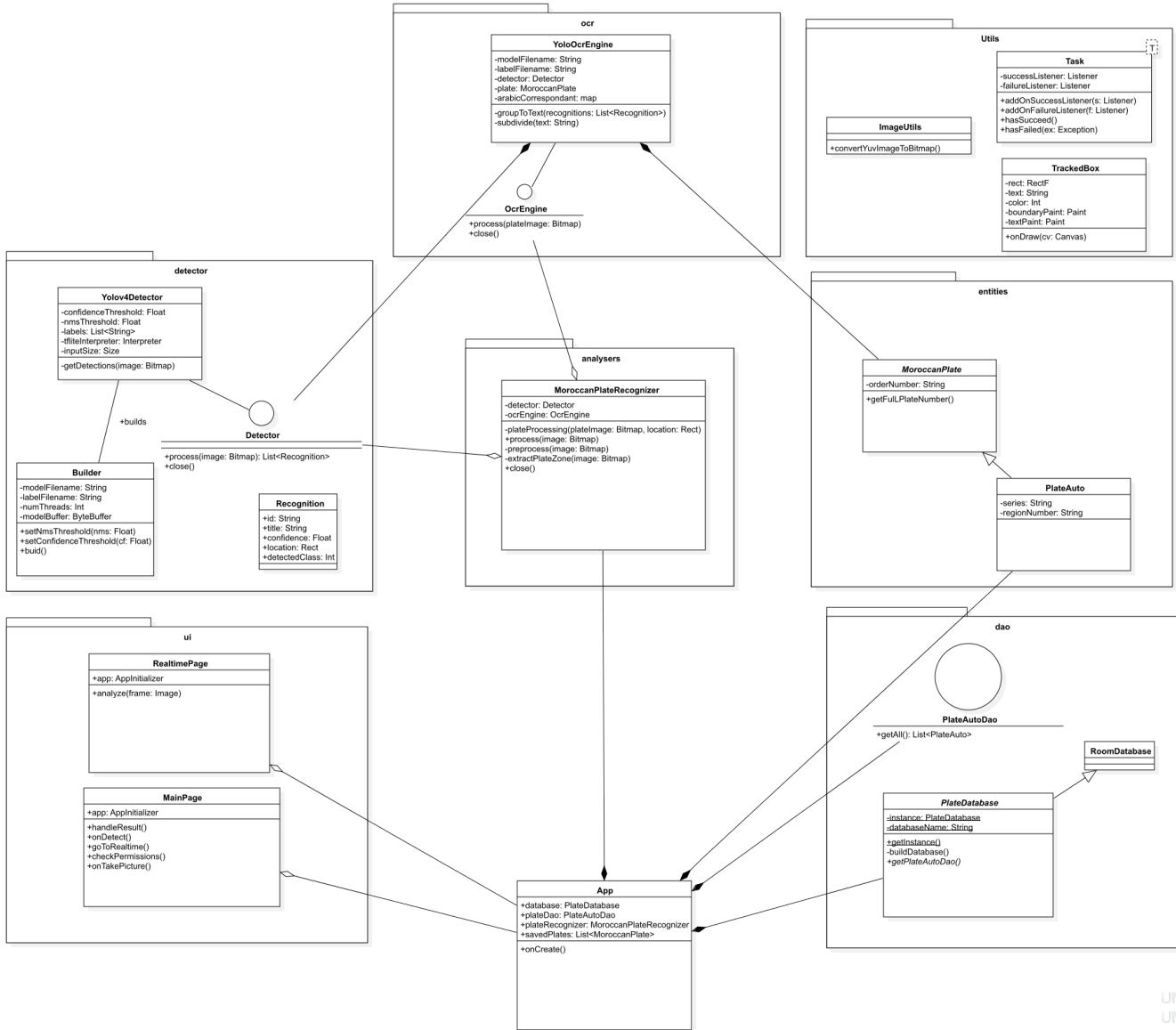


FIGURE 7.3 – Architecture détaillée de notre application

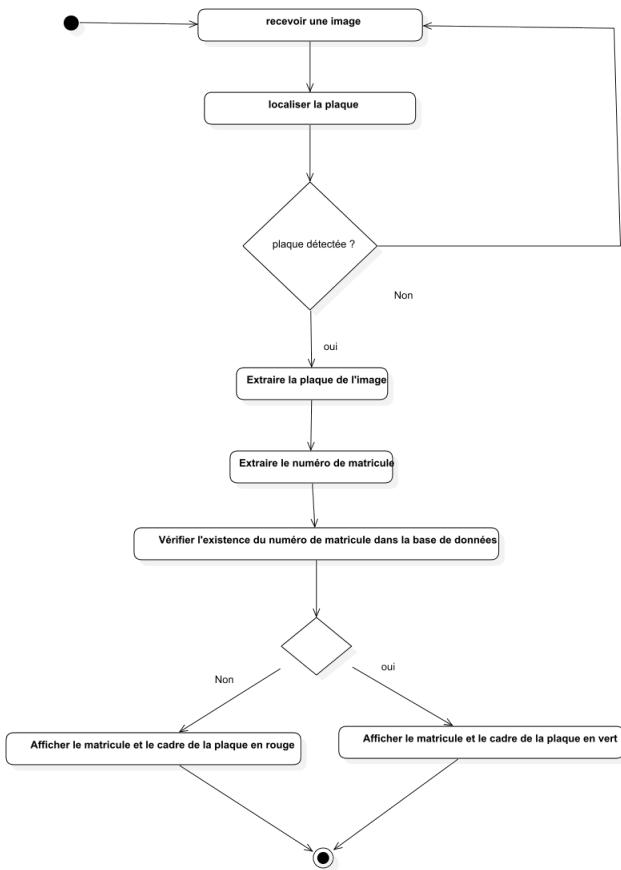


FIGURE 7.4 – Diagramme d'activité pour la reconnaissance des plaques

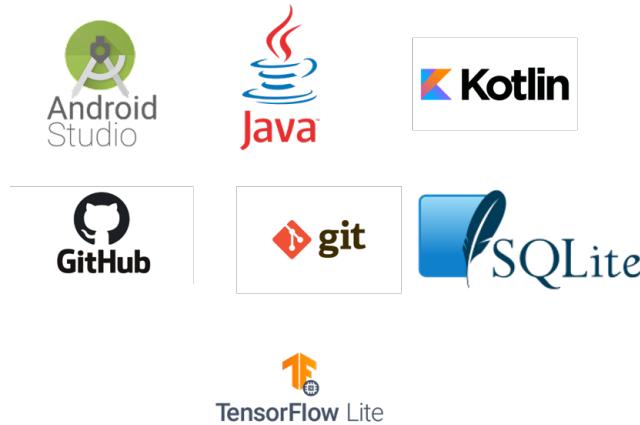
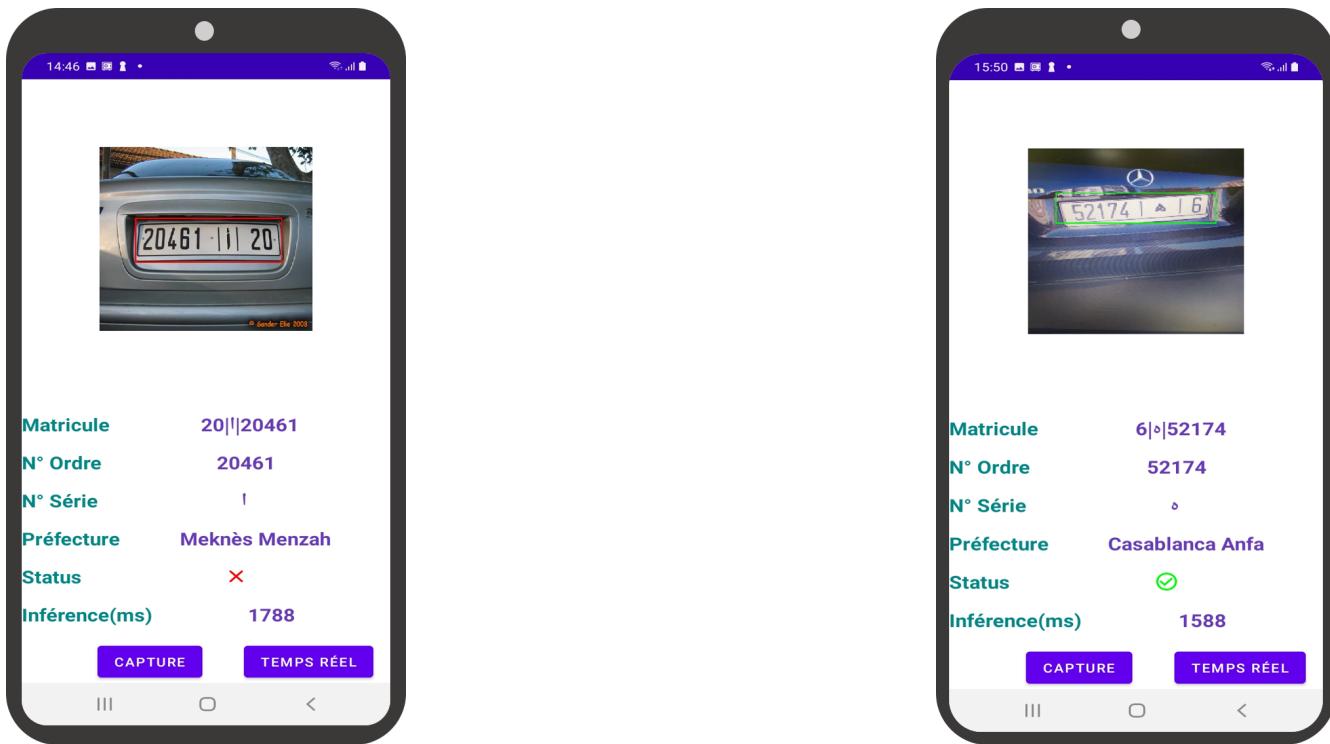


FIGURE 7.5 – Quelques outils logiciels pour le développement de l'application mobile

verbeux, l'ajout de certains extensions comme les Data Class, les coroutines qui facilitent énormément le développement. Néanmoins nous avons utilisé certaines classes utiles écrites en Java par d'autres développeurs.

- **SQLite** : pour la base de données, nous avons opté SQLite qui un système de gestion de base de données à la fois puissant et léger et donc adapté pour une application mobile. Et comme recommandé par la documentation d'Android [5], nous avons utilisé la bibliothèque de persistance **Room** qui fournit une couche d'abstraction sur SQLite pour permettre un accès fluide à la base de données.
- **CameraX** : c'est une bibliothèque proposée par Android qui facilite l'utilisation de la Camera dans les applications Android. Elle offre déjà des services notamment **Image Analysis** qui permet de faire des opérations de Machine Learning. Nous l'avons utilisé pour le cas de la reconnaissance en temps réel pour traiter chaque *frame* renvoyé par la caméra.
- **Tensorflow Lite** : c'est un API développé par Google pour déployer facilement les modèles de Machine Learning sur les appareils mobiles. Nous l'avons donc utilisé pour déployer nos deux modèles YOLO. Mais au préalable, nous avons du convertir les modèles du format YOLO en format TFLite (format reconnu par l'API).
- **Git/GitHub** : pour conserver l'historique des modifications de notre code et collaborer avec les autres, nous avons opté pour le logiciel connu et apprécié Git. Pour héberger le code et conserver l'historique des modifications sur le Cloud, nous avons utilisé la plateforme GitHub. Nous y avons créé un répertoire privé qui contient les différentes versions du code de l'application.

La combinaison de ces outils nous a permis de développer une application simple, performante et facile à prendre en main dont voici quelques interfaces :



(a) Exemple d'une reconnaissance de plaque non enregistrée dans la base de données réalisée en 1788 ms

(b) Exemple d'une reconnaissance de plaque enregistrée dans la base de données réalisée en 1588 ms

FIGURE 7.6 – Exemples de quelques interfaces de l'application mobile

## 7.3 Parking intelligent

Une des applications pratiques et très utiles du système ANPR est le parking intelligent ou encore smart parking. Entrant dans le cadre de la mobilité intelligente pour les citées dites intelligentes, les smart parking sont une solution à la fois moderne et efficace pour résoudre le problème de stationnement dans les villes à forte agglomération et mobilité comme Casablanca ou Rabat. La solution développée par l'entreprise KF2Y Consulting se veut innovante car utilise les nouvelles techniques de Machine Learning à la place de capteurs ou carte RFID. Au cours de notre stage, une équipe de stagiaires étudiants en développement logiciel, système embarqué et services numériques et Machine Learning sous la coordination d'un responsable a mis en place une maquette plus ou moins réaliste d'un smart parking.

### 7.3.1 Architecture du système GoPark

Le système GoPark est constitué principalement de deux grandes parties communiquant entre elles via les services web à travers les requêtes HTTP :

- **L'application mobile GoPark :** c'est une application qui permet d'une part à des clients propriétaires de Parking de créer des annonces de location de places sur son Parking. D'autre part, elle permet à d'autres clients véhiculés d'effectuer des réservations de stationnement sur l'un des parkings présents dans les annonces. Par ailleurs, c'est une application multi



FIGURE 7.7 – Architecture du système GoPark

plateforme développée par les étudiants en ingénierie logiciel du groupe en utilisant le kit de développement logiciel Flutter.

- Le **système embarqué** : c'est un ensemble des composants électroniques et informatiques qui seront installés sur le parking pour la gestion automatique du stationnement. Dans le cadre de notre projet, notre système embarqué peut être subdivisé en deux grandes parties : la **barrière intelligente** qui donne accès ou non aux places de stationnement en fonction du numéro de matricule reconnu et le **détecteur de stationnement** qui permet signaler à l'application mobile la disponibilité d'une place du parking.

Dans les prochaines lignes, nous allons plus nous focaliser sur le système embarqué, la partie de l'application mobile ne faisant partie de nos domaines d'intervention durant notre période de stage.

### 7.3.2 Environnements matériel et logiciel

La réalisation de la maquette de GoPark a été possible grâce à la combinaison d'un assez grand nombre d'outils matériels et logiciels. Parmi les outils matériels, on retrouve :

1. Des **capteurs ultrasoniques** (1) : ils permettent de détecter la présence d'un objet et dans notre cas la présence d'un véhicule dans l'espace de stationnement. Son principe comme son nom l'indique repose sur l'utilisation des ultrasons (ondes acoustiques). Ils sont composés d'un émetteur et d'un récepteur. L'émetteur émet régulièrement un train d'ondes qui va se réfléchir sur l'objet détecté et ensuite se rediriger vers le récepteur. Pour les deux places disponibles dans notre maquette, nous avons utilisé deux capteurs ultrasoniques.
2. Des **microcontrôleurs** (2) : ce sont des circuits intégrés qui se composent des éléments essentiels et minimaux d'un ordinateur (processeur, mémoires, unité périphérique et interfaces entrées/sorties). Ils ont été utilisés pour le traitement avec les capteurs ultrasoniques et communiquer ainsi au serveur de l'application mobile les résultats de la détection de présence.

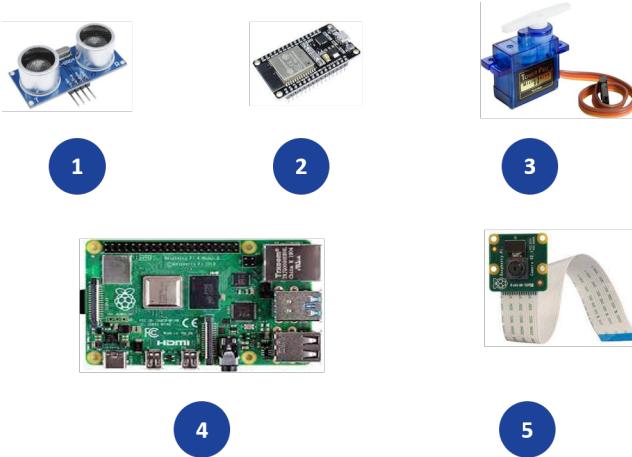


FIGURE 7.8 – Outils matériels pour la réalisation du parking intelligent

3. Un **servomoteur SG90 (3)** : c'est un actionneur qui déclenche un mouvement précis suite à une commande externe. C'est le composant qui servira à l'ouverture et la fermeture de la barrière de notre parking.
4. Une **carte Raspberry Pi 4 (4)** : c'est un petit ordinateur (nano ordinateur) de la taille d'une carte de crédit offrant toutes les fonctionnalités d'un ordinateur standard. C'est dans cette carte que nous allons déployer notre système de reconnaissance des plaques d'immatriculation marocaines. A cette carte seront connectés le servomoteur SG90 et une **caméra V2 de 8 mégapixels (5)** pour récupérer les images de l'entrée du parking en temps réel. Voici quelques spécifications de la carte Raspberry Pi 4 que nous avons utilisée :

Spécifications	Raspberry Pi 4
CPU	1,5 GHz quadricœur ARM Cortex-A72
GPU	Broadcom VideoCore VI, OpenGL ES 3.0
Puissance nominale	3A / 15W
Systèmes d'exploitation	Raspbian OS

TABLE 7.1 – Spécifications de la carte Raspberry Pi 4

Du côté logiciel, les outils dont nous nous sommes servis sont :

1. **L'IDE Arduino** : c'est une application multiplateforme qui permet d'écrire et de télécharger des programmes sur les cartes compatibles Arduino. Nous l'avons utilisé pour programmer les traitements opérés par le microcontrôleur.
2. **Le langage de programmation C++** : c'est le langage utilisé pour coder les programmes sous Arduino.
3. **Firebase** : c'est un ensemble de services d'hébergement pour les divers types d'applications. Nous l'avons utilisé pour le service de base de données NoSQL qui stocke en temps réel l'état d'une place envoyée par le microcontrôleur.
4. **Python** : c'est un langage de programmation interprété, multi-plateforme et multi-paradigme. Au vu de sa simplicité et de la grande documentation disponible, nous avons opté pour ce langage pour coder nos programmes sur la carte Raspberry.



FIGURE 7.9 – Outils logiciels pour la réalisation du parking intelligent

5. **OpenCV** : c'est une bibliothèque regroupant plusieurs fonctions de traitement d'images. Elle a été utile pour exécuter les différents modèles de détection et de lecture des plaques d'immatriculation sur les images en temps réel envoyées par la caméra.
6. **L'IDE Thonny** : c'est l'éditeur utilisé pour le codage en Python.

### 7.3.3 Résultats

Le smart parking matérialisé par la maquette 7.11 et réalisé par notre équipe de stagiaire de KF2Y Consulting fonctionne en plusieurs étapes comme le montre la figure 7.10 :

1. **Création d'une annonce de parking** : on accède à l'application mobile et on crée une annonce sur un parking en indiquant les places disponibles et le lieu ;
2. **Création d'une réservation** : on (en réalité un client véhiculé) accède à l'application et fait une réservation d'une place disponible pendant une tranche horaire dans un parking présent dans les annonces sans oublier de spécifier le numéro de matricule du véhicule ;
3. **Lecture de la plaque d'immatriculation** : le véhicule se dirige à l'entrée du parking où se trouve une caméra et une barrière intelligente. La caméra envoie au programme s'exécutant sur la carte Raspberry l'image du véhicule. Le programme localise dans un premier temps la position de la plaque et par la suite extrait sous format textuel le numéro du matricule ;
4. **Vérification de l'enregistrement du matricule** : le programme s'exécutant dans la carte envoie à travers les services web REST, le numéro de matricule détecté à l'application GoPark. L'application vérifie l'existence de ce matricule dans la base de données et renvoie la réponse au programme sur la carte ;
5. **Action sur la barrière** : Si la réponse reçue par la carte est positive, on ne déclenche aucune action sur la barrière donc elle reste fermée. Dans le cas contraire, le programme déclenche immédiatement l'ouverture de la barrière pour donner accès au véhicule à la zone de stationnement ;
6. **Signalisation de l'occupation d'une place** : lorsque le véhicule arrive sur sa place de stationnement, le capteur ultrason détecte sa présence et envoie une pulsion au microcontrôleur. Ce dernier signale cette occupation à l'application GoPark via Firebase ;

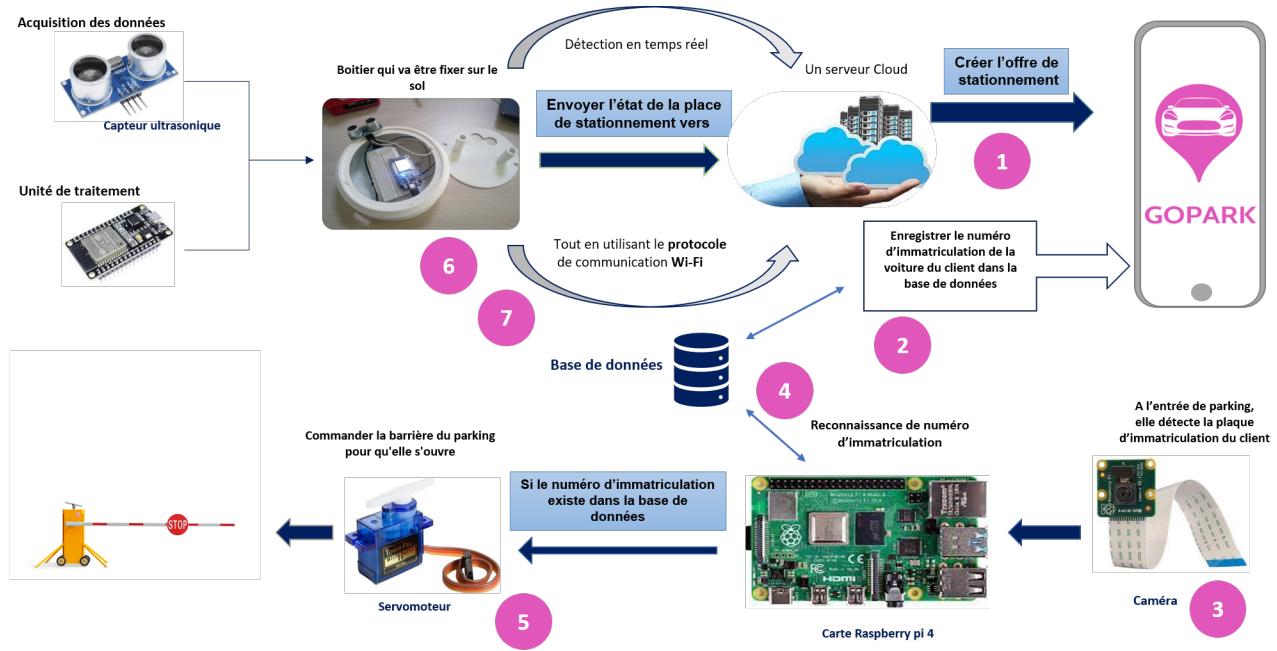


FIGURE 7.10 – Fonctionnement général de GoPark

7. **Signalisation de la libération d'une place** : au moment où le véhicule libère la place, les mêmes opérations précédentes sont faites par le capteur et le microcontrôleur mais cette fois-ci pour signaler la libération d'une place.

## 7.4 Conclusion

Nous arrivons au terme de ce dernier chapitre qui nous a permis de présenter les différentes applications dans lesquelles nous avons déployé notre système ANPR MoPlaZer. La première est une application mobile qui permet de faire la reconnaissance des plaques d'immatriculation marocaines d'une part sur une capture d'image prise par un utilisateur et d'autre part sur une vidéo en temps réel. La seconde est un parking intelligent appelé GoPark. Nous avons réalisé une maquette de ce parking qui utilise le système ANPR pour lire les plaques d'immatriculation des véhicules à l'entrée du parking. Le résultat donné par ce système permet de déterminer si on doit déclencher l'ouverture de la barrière ou non. Ce dernier chapitre loin d'être le moindre montre à quel point le système ANPR que nous avons développé peut être utile pour notre société et faciliter ainsi la circulation et le stationnement dans de grandes métropoles du Royaume comme Casablanca.



FIGURE 7.11 – Maquette du parking intelligent GoPark

# Bibliographie et Webographie

- [1] Abdelkrim ALAHYANE et al. « Open data for Moroccan license plates for OCR applications : data collection, labeling, and model construction ». In : *ArXiv* abs/2104.08244 (2021).
- [2] Hinde ANOUAL. « Détection et Localisation de texte dans les images de scènes naturelles :Application à la détection des plaques d'immatriculation marocaines ». Thèse de doct. Faculté des Sciences de Rabat, Soutenue le 14 Juillet 2012.
- [3] Lucie CAMANEZ. *Extraction d'objets pour la cartographie par deep-learning : évaluation du modèle*. Dernière Visite : 27-07-2021. URL : <https://makina-corpus.com/blog/metier/2020/extraction-dobjets-pour-la-cartographie-par-deep-learning-evaluation-du-modele>.
- [4] Khouloud DAHMANE. « Analyse d'images par méthode de Deep Learning appliquée au contexte routier en conditions météorologiques dégradées. » Thèse de doct. Université Clermont Auvergne, 2020.
- [5] Android DEVELOPERS. *Save data using SQLite*. Dernière Visite : 25-08-2021. URL : <https://developer.android.com/training/data-storage/sqlite>.
- [6] EXPERT.AI. *What is Machine Learning ? A Definition*. Dernière Visite : 20-07-2021. URL : <https://www.expert.ai/blog/machine-learning-definition/>.
- [7] GENETEC. *Manuel AutoVu 5.2 SR9*. URL : <https://downloadcenter1.genetec.com/products/SecurityCenter/5.2/SR9/AutoVu/FR.AutoVu%20Handbook%205.2%20SR9.pdf>.
- [8] Ross GIRSHICK et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv : [1311.2524 \[cs.CV\]](https://arxiv.org/abs/1311.2524).
- [9] Open IMAGES. Dernière Visite : 02-08-2021. URL : <https://storage.googleapis.com/openimages/web/index.html>.
- [10] NOR IMANE et SIDHOUM SOUAD. « Développement d'une application de détection et de reconnaissance de plaques d'immatriculation(LAPIA) ». Mém. de mast. Université Abou Bakr Belkaïd– Tlemcen d'Algérie, Soutenue le 02 Juillet 2017.
- [11] AKACEM Oum el KHEIR et RAHMANI NASSIRA. « Système de reconnaissance des plaques d'immatriculation Algérienne ». Mém. de mast. Faculté des Sciences et de la Technologie, Université d'Adrar, Soutenue en 2015.
- [12] Y. LECUN et al. « Backpropagation Applied to Handwritten Zip Code Recognition ». In : *Neural Computation* 1.4 (déc. 1989), p. 541-551. ISSN : 0899-7667. DOI : [10.1162/neco.1989.1.4.541](https://doi.org/10.1162/neco.1989.1.4.541). eprint : <https://direct.mit.edu/neco/article-pdf/1/4/541/811941/neco.1989.1.4.541.pdf>. URL : <https://doi.org/10.1162/neco.1989.1.4.541>.

- [13] Wei LIU et al. « SSD : Single Shot MultiBox Detector ». In : *Lecture Notes in Computer Science* (2016), p. 21-37. ISSN : 1611-3349. DOI : [10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2). URL : [http://dx.doi.org/10.1007/978-3-319-46448-0\\_2](http://dx.doi.org/10.1007/978-3-319-46448-0_2).
- [14] MATHWORKS. *Types of Morphological Operations*. Dernière Visite : 19-07-2021. URL : <https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html>.
- [15] Joseph REDMON et al. *You Only Look Once : Unified, Real-Time Object Detection*. 2016. arXiv : [1506.02640 \[cs.CV\]](https://arxiv.org/abs/1506.02640).
- [16] Shaoqing REN et al. *Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv : [1506.01497 \[cs.CV\]](https://arxiv.org/abs/1506.01497).
- [17] Krim SMAÏL. « Reconnaissance Automatique des plaques d'immatriculation (R.A.P.I) (Implémentation sur Raspberry Pi) ». Mém. de mast. Faculté des Sciences et de la Technologie, Université Mohamed Khider de Biskra, Soutenue le 09 juillet 2019.
- [18] Ting TAO et al. « Object Detection-Based License Plate Localization and Recognition in Complex Environments ». In : *Transportation Research Record* 2674.12 (2020), p. 212-223. DOI : [10.1177/0361198120954202](https://doi.org/10.1177/0361198120954202). eprint : <https://doi.org/10.1177/0361198120954202>. URL : <https://doi.org/10.1177/0361198120954202>.
- [19] Alphanumeric VISION. *Logiciel LAPI*. Dernière Visite : 13-07-2021. URL : <http://www.alphanumeric-vision.com/fr/logiciel-lapi/>.
- [20] WIKIPEDIA. *Automatic number-plate recognition*. Dernière Visite : 13-07-2021. URL : [https://en.wikipedia.org/wiki/Automatic\\_number-plate\\_recognition](https://en.wikipedia.org/wiki/Automatic_number-plate_recognition).
- [21] WIKIPEDIA. *Image numérique*. Dernière Visite : 19-07-2021. URL : [https://fr.wikipedia.org/wiki/Image\\_num%C3%A9rique](https://fr.wikipedia.org/wiki/Image_num%C3%A9rique).
- [22] WIKIPEDIA. *Reconnaissance optique de caractères*. Dernière Visite : 20-07-2021. URL : [https://fr.wikipedia.org/wiki/Reconnaissance\\_optique\\_de\\_caract%C3%A8res](https://fr.wikipedia.org/wiki/Reconnaissance_optique_de_caract%C3%A8res).
- [23] Fei XIE et al. « A Robust License Plate Detection and Character Recognition Algorithm Based on a Combined Feature Extraction Model and BPNN ». In : *Journal of Advanced Transportation* 2018 (sept. 2018), p. 1-14. DOI : [10.1155/2018/6737314](https://doi.org/10.1155/2018/6737314).